pygamelib Documentation

Release 1.3.0

Arnaud Dupuis

Contents (API reference):

1	Forewords	3
2	Introduction	5
3	Tutorials	7
	3.1 actuators	7
	3.2 assets	34
	3.3 base	190
	3.4 board_items	206
	3.5 constants	431
	3.6 engine	432
	3.7 gfx	479
	3.8 Credits	599
	3.9 Release notes	600
4	Indices and tables	607
Рy	ython Module Index	609
In	ndex	611



CHAPTER 1

Forewords

Historically, this library was (and still is) used as a base to teach coding to kids from 6 to 15. It aims at giving an environment to new and learning developers (including kids) that let them focus on the algorithm instead of the lousy display or precise management.

It started as a very simple library with very little capabilities, but over time it became something more. To the point that it is now possible to make very decent terminal games with it.

So this is **obviously** still extremely simple compared to other game framework and it still does not have the pretention of being anything serious for real game developers. However, it can now be used by aspiring game developers for an introduction to 2D games development.

CHAPTER 2

Introduction

First of all, his module is exclusively compatible with python 3.6+.

The core concept is that writting a game mostly involve the *Game* object, the *Board* object and the derivatives of *board_items*.

More advanced game will use the ui module to create terminal user interfaces (or TUI) and the GFX core module to improve the graphics with Sprite and Color.

Here is an example of what the current version allow to build:

And a quick peak at the new features in the most recent version:

CHAPTER 3

Tutorials

Most tutorials to teach you how to use the library to build games are (or will be) on the wiki.

Tutorials that teach you how to expand the library are (or will be) centralized here.

The complete API documentation is referenced bellow.

3.1 actuators

This module contains the base classes for simple and advanced actuators. These classes are the base contract for actuators. If you wish to create your own one, you need to inherit from one of these base class.

3.1.1 Actuator

```
class pygamelib.actuators.Actuator(parent)
    Bases: pygamelib.base.PglBaseObject
```

Actuator is the base class for all Actuators. It is mainly a contract class with some utility methods.

By default, all actuators are considered movement actuators. So the base class only require next_move() to be implemented.

Parameters parent – the item parent.

```
___init___(parent)
```

The constructor take only one (positional) parameter: the parent object.

Important: The default state of ALL actuators is RUNNING. If you want your actuator to be in a different state (PAUSED for example), you have to do it yourself.

Methods

1 11 (Th
init(parent)	The constructor take only one (positional) parameter:
	the parent object.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load serialized data, create and returns a new actua-
	tor out of these data.
next_move()	That method needs to be implemented by all actu-
	ators or a NotImplementedError exception will be
	raised.
notify([modifier])	Notify all the observers that a change occurred.
pause()	Set the actuator state to PAUSED.
serialize()	Serializes the actuator and returns it as a dict.
start()	Set the actuator state to RUNNING.
stop()	Set the actuator state to STOPPED.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (*PglBaseObject*) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PqlBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

load(data: dict = None)

Load serialized data, create and returns a new actuator out of these data.

That method needs to be implemented by all actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

next_move()

That method needs to be implemented by all actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serializes the actuator and returns it as a dict.

That method needs to be implemented by all actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

start()

Set the actuator state to RUNNING.

If the actuator state is not RUNNING, actuators' next_move() function (and all derivatives) should not return anything.

Example:

```
mygame.start()
```

stop()

Set the actuator state to STOPPED.

Example:

```
mygame.stop()
```

$\verb|store_screen_position|| (\textit{row: int, column: int})| \rightarrow bool|$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.1.2 Behavioral

class pygamelib.actuators.Behavioral(parent)

```
Bases: pygamelib.actuators.Actuator
```

The behavioral actuator is inheriting from Actuator and is adding a next_action() method. The actual actions are left to the actuator that implements Behavioral.

Parameters parent – the item parent.

```
___init___(parent)
```

The constructor simply construct an Actuator. It takes on positional parameter: the parent object.

10 Chapter 3. Tutorials

Methods

The second secon
The constructor simply construct an Actuator.
Attach an observer to this instance.
Detach an observer from this instance.
A virtual method that needs to be implemented by
the observer.
Load serialized data, create and returns a new actua-
tor out of these data.
That method needs to be implemented by all behav-
ioral actuators or a NotImplementedError exception
will be raised.
That method needs to be implemented by all actu-
ators or a NotImplementedError exception will be
raised.
Notify all the observers that a change occurred.
Set the actuator state to PAUSED.
Serializes the actuator and returns it as a dict.
Set the actuator state to RUNNING.
Set the actuator state to ROTTING.
Set the actuator state to STOPPED.

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

load(data: dict = None)

Load serialized data, create and returns a new actuator out of these data.

That method needs to be implemented by all actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

next action()

That method needs to be implemented by all behavioral actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

next move()

That method needs to be implemented by all actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- value(Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the actuator state to PAUSED.

```
mygame.pause()
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serializes the actuator and returns it as a dict.

That method needs to be implemented by all actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

start()

Set the actuator state to RUNNING.

If the actuator state is not RUNNING, actuators' next_move() function (and all derivatives) should not return anything.

Example:

```
mygame.start()
```

stop()

Set the actuator state to STOPPED.

Example:

```
mygame.stop()
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.1.3 PathActuator

class pygamelib.actuators.PathActuator(path=None, parent=None)

```
Bases: pygamelib.actuators.Actuator
```

The path actuator is a subclass of Actuator. The move inside the function next_move depends on path and index. If the state is not running it returns None otherwise it increments the index & then, further compares the

index with length of the path. If they both are same then, index is set to value zero and the move is returned back.

Parameters

- path (list) A list of paths.
- parent (pygamelib.board_items.BoardItem) The parent object to actuate.

__init__ (path=None, parent=None)

The constructor take only one (positional) parameter: the parent object.

Important: The default state of ALL actuators is RUNNING. If you want your actuator to be in a different state (PAUSED for example), you have to do it yourself.

Methods

init([path, parent])	The constructor take only one (positional) parameter:
	the parent object.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new PathActuator out of it.
next_move()	Return the movement based on current index
notify([modifier])	Notify all the observers that a change occurred.
pause()	Set the actuator state to PAUSED.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_path(path)	Defines a new path
start()	Set the actuator state to RUNNING.
stop()	Set the actuator state to STOPPED.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- subject (PglBaseObject) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

classmethod load(data: dict = None)

Load data and create a new PathActuator out of it.

```
Parameters data (dict) - Data to create a new actuator (usually generated by serialize())
```

Returns A new actuator.

Return type PathActuator

Example:

```
path_actuator = PathActuator.load(actuator_data)
```

next_move()

Return the movement based on current index

The movement is selected from path if state is RUNNING, otherwise it returns NO_DIR from the constants module. When state is RUNNING, the movement is selected before incrementing the index by 1. When the index equal the length of path, the index should return back to 0.

Returns The next movement

Return type int | pygamelib.constants.NO_DIR

Example:

```
path_actuator.next_move()
```

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
\texttt{serialize}\,(\,)\,\to dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_path (path)

Defines a new path

This will also reset the index back to 0.

Parameters path (list) – A list of movements.

```
\verb|path_actuator.set_path([constants.UP, constants.DOWN, constants.LEFT, constants.\\ \\ \hookrightarrow \verb|RIGHT]|)|
```

start()

Set the actuator state to RUNNING.

If the actuator state is not RUNNING, actuators' next_move() function (and all derivatives) should not return anything.

Example:

```
mygame.start()
```

stop()

Set the actuator state to STOPPED.

Example:

```
mygame.stop()
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.1.4 PatrolActuator

class pygamelib.actuators.PatrolActuator(path=None, parent=None)

```
Bases: pygamelib.actuators.PathActuator
```

The patrol actuator is a subclass of <code>PathActuator</code>. The move inside the function next_move depends on path and index and the mode. Once it reaches the end of the move list it will start cycling back to the beginning of the list. Once it reaches the beginning it will start moving forwards If the state is not running it returns None otherwise it increments the index & then, further compares the index with length of the path. If they both are same then, index is set to value zero and the move is returned back.

Parameters path (list) – A list of directions.

```
___init___(path=None, parent=None)
```

The constructor take only one (positional) parameter: the parent object.

Important: The default state of ALL actuators is RUNNING. If you want your actuator to be in a different state (PAUSED for example), you have to do it yourself.

Methods

init([path, parent])	The constructor take only one (positional) parameter:
	the parent object.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new PatrolActuator out of it.
next_move()	Return the movement based on current index
notify([modifier])	Notify all the observers that a change occurred.
pause()	Set the actuator state to PAUSED.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_path(path)	Defines a new path
start()	Set the actuator state to RUNNING.
stop()	Set the actuator state to STOPPED.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

handle notification (*subject*, *attribute=None*, *value=None*)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

classmethod load(data: dict = None)

Load data and create a new PatrolActuator out of it.

```
Parameters data (dict) - Data to create a new actuator (usually generated by serialize())
```

Returns A new actuator.

Return type PatrolActuator

Example:

```
patrol_actuator = PatrolActuator.load(actuator_data)
```

next_move()

Return the movement based on current index

The movement is selected from path if state is RUNNING, otherwise it returns NO_DIR from the *constants* module. When state is RUNNING, the movement is selected before incrementing the index by 1. When the index equals the length of path, the index should return back to 0 and the path list should be reversed before the next call.

Returns The next movement

Return type int | pygamelib.constants.NO_DIR

Example:

```
patrol_actuator.next_move()
```

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_path(path)

Defines a new path

This will also reset the index back to 0.

Parameters path (list) - A list of movements.

Example:

```
\verb|path_actuator.set_path([constants.UP, constants.DOWN, constants.LEFT, constants.\\ \\ \hookrightarrow \verb|RIGHT]|)|
```

start()

Set the actuator state to RUNNING.

If the actuator state is not RUNNING, actuators' next_move() function (and all derivatives) should not return anything.

Example:

```
mygame.start()
```

stop()

Set the actuator state to STOPPED.

Example:

```
mygame.stop()
```

20 Chapter 3. Tutorials

```
store\_screen\_position (row: int, column: int) \rightarrow bool Store the screen position of the object.
```

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.1.5 PathFinder

```
class pygamelib.actuators.PathFinder(game=None, actuated\_object=None, circle_waypoints=True, parent=None, algorithm=90000100)
```

Bases: pygamelib.actuators.Behavioral

Important: This module assume a one step movement. If you need more than one step, you will need to sub-class this module and re-implement next_waypoint().

This actuator is a bit different than the simple actuators (SimpleActuators) as it requires the knowledge of both the game object and the actuated object.

The constructor takes the following parameters:

Parameters

- game (pygamelib.engine.Game) A reference to the instantiated game engine.
- actuated_object (pygamelib.board_items.BoardItem) The object to actuate. Deprecated in favor of parent. Only kept for backward compatibility.
- parent (pygamelib.board_items.BoardItem) The parent object to actuate.
- circle_waypoints (bool) If True the next_waypoint() method is going to circle between the waypoints (when the last is visited, go back to the first)
- algorithm (constant) ALGO BFS BFS, ALGO ASTAR AStar

```
__init__(game=None, actuated_object=None, circle_waypoints=True, parent=None, algorithm=90000100)
```

The constructor simply construct an Actuator. It takes on positional parameter: the parent object.

Methods

init([game, actuated_object,])	The constructor simply construct an Actuator.
add_waypoint(row, column)	Add a waypoint to the list of waypoints.
attach(observer)	Attach an observer to this instance.
clear_waypoints()	Empty the waypoints stack.
current_path()	This method simply return a copy of the current path
	of the actuator.
	Continued on post page

Continued on next page

Table 9 – continued from previous page

current_waypoint()		Return the currently active waypoint.
detach(observer)		Detach an observer from this instance.
find_path()		Find a path to the destination.
handle_notification(subject[, at	ttribute,	A virtual method that needs to be implemented by
value])		the observer.
load(data)		Load data and create a new PathFinder out of it.
next_action()		That method needs to be implemented by all behav-
		ioral actuators or a NotImplementedError exception
		will be raised.
next_move()		This method return the next move calculated by this
		actuator.
next_waypoint()		Return the next active waypoint.
notify([modifier])		Notify all the observers that a change occurred.
pause()		Set the actuator state to PAUSED.
remove_waypoint(row, column)		Remove a waypoint from the stack.
serialize()		Return a dictionary with all the attributes of this ob-
		ject.
set_destination([row, column])		Set the targeted destination.
start()		Set the actuator state to RUNNING.
stop()		Set the actuator state to STOPPED.
store_screen_position(row, column))	Store the screen position of the object.

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

add_waypoint (row, column)

Add a waypoint to the list of waypoints.

Waypoints are used one after the other on a FIFO basis (First In, First Out).

If not destination (i.e destination == (None, None)) have been set yet, that method sets it.

Parameters

- row (int) The "row" part of the waypoint's coordinate.
- column The "column" part of the waypoint's coordinate.

Raises PglInvalidTypeException - If any of the parameters is not an int.

Example:

```
pf = PathFinder(game=mygame, actuated_object=npc1)
pf.add_waypoint(3,5)
pf.add_waypoint(12,15)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

clear_waypoints()

Empty the waypoints stack.

Example:

```
pf.clear_waypoints()
```

current_path()

This method simply return a copy of the current path of the actuator.

The current path is to be understood as: the list of positions still remaining. All positions that have already been gone through are removed from the stack.

Important: A copy of the path is returned for every call to that function so be wary of the performances impact.

Example:

current_waypoint()

Return the currently active waypoint.

If no waypoint have been added, this function return None.

Returns Either a None tuple or the current waypoint.

Return type A None tuple or a tuple of integer.

Example:

```
(row,column) = pf.current_waypoint()
pf.set_destination(row,column)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

find_path()

Find a path to the destination.

Destination (PathFinder.destination) has to be set beforehand.

Example:

Warning: PathFinder.destination is a tuple! Please use PathFinder.set_destination(x,y) to avoid problems.

Path Finding Algorithm Description:

Breadth First Search: This method implements a Breadth First Search algorithm (Wikipedia: BFS) to find the shortest path to destination.

A* Search: This method implements a A* Search algorithm (Wikipedia: A*) to find the shortest path to destination.

```
handle_notification (subject, attribute=None, value=None)
```

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

classmethod load(data: dict = None)

Load data and create a new PathFinder out of it.

Parameters data (dict) - Data to create a new actuator (usually generated by serialize())

Returns A new actuator.

Return type PathFinder

Example:

```
path_finder = PathFinder.load(actuator_data)
```

next_action()

That method needs to be implemented by all behavioral actuators or a NotImplementedError exception will be raised.

Raises NotImplementedError

next move()

This method return the next move calculated by this actuator.

In the case of this PathFinder actuator, next move does the following:

- If the destination is not set return NO_DIR (see constants) If the destination is set, but the path is empty and actuated object's position is different from destination: call find_path()
- Look at the current waypoint, if the actuated object is not at that position return a direction from the *constants* module. The direction is calculated from the difference between actuated object's position and waypoint's position.
- If the actuated object is at the waypoint position, then call next_waypoint(), set the destination and return a direction. In this case, also call find_path().
- In any case, if there is no more waypoints in the path this method returns NO_DIR (see constants)

Example:

next_waypoint()

Return the next active waypoint.

If no waypoint have been added, this function return None. If there is no more waypoint in the stack:

- if PathFinder.circle_waypoints is True this function reset the waypoints stack and return the first one.
- else, return None.

Returns Either a None tuple or the next waypoint.

Return type A None tuple or a tuple of integer.

Example:

```
pf.circle_waypoints = True
(row,column) = pf.next_waypoint()
pf.set_destination(row,column)
```

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

remove_waypoint (row, column)

Remove a waypoint from the stack.

This method removes the first occurrence of a waypoint in the stack.

If the waypoint cannot be found, it raises a ValueError exception. If the row and column parameters are not int, an PgIInvalidTypeException is raised.

Parameters

- row (int) The "row" part of the waypoint's coordinate.
- **column** The "column" part of the waypoint's coordinate.

Raises

- PglInvalidTypeException If any of the parameters is not an int.
- **ValueError** If the waypoint is not found in the stack.

Example:

```
path_finder.remove_waypoint(2,5)
```

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_destination (row=0, column=0)

Set the targeted destination.

Parameters

- row (int) "row" coordinate on the board grid
- column (int) "column" coordinate on the board grid

Raises PglInvalidTypeException - if row or column are not int.

Example:

```
mykillernpc.actuator.set_destination(
    mygame.player.pos[0], mygame.player.pos[1]
)
```

start()

Set the actuator state to RUNNING.

If the actuator state is not RUNNING, actuators' next_move() function (and all derivatives) should not return anything.

Example:

```
mygame.start()
```

stop()

Set the actuator state to STOPPED.

Example:

```
mygame.stop()
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.1.6 RandomActuator

```
class pygamelib.actuators.RandomActuator(moveset=None, parent=None)
    Bases: pygamelib.actuators.Actuator
```

A class that implements a random choice of movement.

The random actuator is a subclass of Actuator. It is simply implementing a random choice in a predefined move set.

Parameters

- moveset (list) A list of movements.
- parent (pygamelib.board_items.BoardItem) The parent object to actuate.

___init__ (moveset=None, parent=None)

The constructor take only one (positional) parameter: the parent object.

Important: The default state of ALL actuators is RUNNING. If you want your actuator to be in a different state (PAUSED for example), you have to do it yourself.

Methods

init([moveset, parent])	The constructor take only one (positional) parameter:
	the parent object.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new RandomActuator out of
	it.
next_move()	Return a randomly selected movement
notify([modifier])	Notify all the observers that a change occurred.
pause()	Set the actuator state to PAUSED.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
start()	Set the actuator state to RUNNING.
stop()	Set the actuator state to STOPPED.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

moveset	Return the moveset.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

classmethod load(data: dict = None)

Load data and create a new RandomActuator out of it.

```
Parameters data (dict) - Data to create a new actuator (usually generated by serialize())
```

Returns A new actuator.

Return type RandomActuator

Example:

```
npc2.actuator = actuators.RandomActuator.load( npc1.actuator.serialize() )
```

moveset

Return the moveset.

Returns The moveset.

Return type list

next_move()

Return a randomly selected movement

The movement is randomly selected from moveset if state is RUNNING, otherwise it returns NO_DIR from the constants module.

Returns The next movement

Return type int | pygamelib.constants.NO_DIR

Example:

```
random_actuator.next_move()
```

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

start()

Set the actuator state to RUNNING.

If the actuator state is not RUNNING, actuators' next_move() function (and all derivatives) should not return anything.

```
mygame.start()
```

stop()

Set the actuator state to STOPPED.

Example:

```
mygame.stop()
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.1.7 Unidirectional Actuator

class pygamelib.actuators.**UnidirectionalActuator**(*direction=10000100*, *parent=None*) Bases: pygamelib.actuators.Actuator

A class that implements a single movement.

The unidirectional actuator is a subclass of Actuator. It is simply implementing a mono directional movement. It is primarily target at projectiles.

Parameters

- direction (int) A single direction from the Constants module.
- parent (pygamelib.board_items.BoardItem) The parent object to actuate.

```
___init___(direction=10000100, parent=None)
```

The constructor take only one (positional) parameter: the parent object.

Important: The default state of ALL actuators is RUNNING. If you want your actuator to be in a different state (PAUSED for example), you have to do it yourself.

Methods

init([direction, parent])		The constructor take only one (positional) parameter:
		the parent object.
attach(observer)		Attach an observer to this instance.
detach(observer)		Detach an observer from this instance.
handle_notification(subject[,	attribute,	A virtual method that needs to be implemented by
value])		the observer.

Continued on next page

IADIE	1.7	- (:())	11()111	previous page

Load data and create a new UnidirectionalActuator
out of it.
Return the direction.
Notify all the observers that a change occurred.
Set the actuator state to PAUSED.
Return a dictionary with all the attributes of this ob-
ject.
Set the actuator state to RUNNING.
Set the actuator state to STOPPED.
Store the screen position of the object.

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

classmethod load(data: dict = None)

Load data and create a new Unidirectional Actuator out of it.

Parameters data (dict) - Data to create a new actuator (usually generated by serialize())

Returns A new actuator.

Return type UnidirectionalActuator

Example:

```
unidir_actuator = UnidirectionalActuator.load(actuator_data)
```

next_move()

Return the direction.

The movement is always direction if state is RUNNING, otherwise it returns NO_DIR from the constants module.

Returns The next movement

Return type int | pygamelib.constants.NO_DIR

Example:

```
unidirectional_actuator.next_move()
```

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

3.1. actuators 33

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\textbf{serialize}\,(\,)\,\to dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

start()

Set the actuator state to RUNNING.

If the actuator state is not RUNNING, actuators' next_move() function (and all derivatives) should not return anything.

Example:

```
mygame.start()
```

stop()

Set the actuator state to STOPPED.

Example:

```
mygame.stop()
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.2 assets

The assets sub-module holds all the classes that are adding features without being core features. The graphics module is a good example of that: it is cool to have and provides a nice default set of assets to build games. But the library can work without it.

3.2.1 graphics

Important: The Graphics module was introduced in version 1.1.0.

The Graphics module hold many variables that aims at simplifying the use of unicode characters in the game development process.

This module also import colorama. All styling features are accessible through:

- Graphics.Fore for Foreground colors.
- · Graphics.Back for Background colors.
- Graphics. Style for styling options.

For convenience, the different entities are scattered in grouping classes:

- All emojis are in the Models class.
- The UI/box drawings are grouped into the BoxDrawings class.
- The block glyphs are in the Blocks class.
- The geometric shapes are in the GeometricShapes class.

This modules defines a couple of colored squares and rectangles that should displays correctly in all terminals.

These are kept for legacy purpose (I personally have a lot of kids that are still using it), but for anyone starting fresh, it is better to use the <color>_rect() and <color>_square() static methods of the Sprixel class. Particularly if you are going to use them as background for your Board.

Colored rectangles:

- WHITE RECT
- BLUE_RECT
- RED_RECT
- MAGENTA_RECT
- GREEN RECT
- YELLOW_RECT
- BLACK_RECT
- CYAN_RECT

Then colored squares:

- WHITE_SQUARE
- MAGENTA_SQUARE
- GREEN_SQUARE
- RED_SQUARE
- BLUE_SQUARE
- YELLOW_SQUARE
- BLACK_SQUARE
- CYAN_SQUARE

And finally an example of composition of rectangles to make different colored squares:

- RED_BLUE_SQUARE = RED_RECT+BLUE_RECT
- YELLOW_CYAN_SQUARE = YELLOW_RECT+CYAN_RECT

The Graphics module contains the following classes:

Blocks

class pygamelib.assets.graphics.Blocks

Bases: object

Block elements (unicode)

Here is the list of supported glyphs:

- UPPER_HALF_BLOCK =
- LOWER_ONE_EIGHTH_BLOCK =
- LOWER_ONE_QUARTER_BLOCK =
- LOWER_THREE_EIGHTHS_BLOCK =
- LOWER_HALF_BLOCK =
- LOWER_FIVE_EIGHTHS_BLOCK =
- LOWER_THREE_QUARTERS_BLOCK =
- LOWER_SEVEN_EIGHTHS_BLOCK =
- FULL_BLOCK =
- LEFT_SEVEN_EIGHTHS_BLOCK =
- LEFT_THREE_QUARTERS_BLOCK =
- LEFT_FIVE_EIGHTHS_BLOCK =
- LEFT_HALF_BLOCK =
- LEFT_THREE_EIGHTHS_BLOCK =
- LEFT_ONE_QUARTER_BLOCK =
- LEFT_ONE_EIGHTH_BLOCK =
- RIGHT_HALF_BLOCK =
- LIGHT_SHADE =
- MEDIUM_SHADE =
- DARK_SHADE =
- UPPER_ONE_EIGHTH_BLOCK =
- RIGHT_ONE_EIGHTH_BLOCK =
- QUADRANT_LOWER_LEFT =
- QUADRANT_LOWER_RIGHT =
- QUADRANT_UPPER_LEFT =
- QUADRANT_UPPER_LEFT_AND_LOWER_LEFT_AND_LOWER_RIGHT =

- QUADRANT_UPPER_LEFT_AND_LOWER_RIGHT =
- QUADRANT_UPPER_LEFT_AND_UPPER_RIGHT_AND_LOWER_LEFT =
- QUADRANT_UPPER_LEFT_AND_UPPER_RIGHT_AND_LOWER_RIGHT =
- QUADRANT_UPPER_RIGHT =
- QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT =
- QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT_AND_LOWER_RIGHT =

___init___()

Initialize self. See help(type(self)) for accurate signature.

Methods

init Initialize self.
Attributes
DARK_SHADE
FULL_BLOCK
LEFT_FIVE_EIGHTHS_BLOCK
LEFT_HALF_BLOCK
LEFT_ONE_EIGHTH_BLOCK
LEFT_ONE_QUARTER_BLOCK
LEFT_SEVEN_EIGHTHS_BLOCK
LEFT_THREE_EIGHTHS_BLOCK
LEFT_THREE_QUARTERS_BLOCK
LIGHT_SHADE
LOWER_FIVE_EIGHTHS_BLOCK
LOWER_HALF_BLOCK
LOWER_ONE_EIGHTH_BLOCK
LOWER_ONE_QUARTER_BLOCK
LOWER_SEVEN_EIGHTHS_BLOCK
LOWER_THREE_EIGHTHS_BLOCK
LOWER_THREE_QUARTERS_BLOCK
MEDIUM_SHADE
QUADRANT_LOWER_LEFT
QUADRANT_LOWER_RIGHT
QUADRANT_UPPER_LEFT
QUADRANT_UPPER_LEFT_AND_LOWER_LEFT_AND_LOWER_RIGHT
QUADRANT_UPPER_LEFT_AND_LOWER_RIGHT
QUADRANT_UPPER_LEFT_AND_UPPER_RIGHT_AND_LOWER_LEFT
QUADRANT_UPPER_LEFT_AND_UPPER_RIGHT_AND_LOWER_RIGHT
QUADRANT_UPPER_RIGHT
QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT
QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT_AND_LOWER_RIGHT
RIGHT_HALF_BLOCK
RIGHT_ONE_EIGHTH_BLOCK
UPPER_HALF_BLOCK

Table 16 – continued from previous page

UPPER_ONE_EIGHTH_BLOCK

```
DARK SHADE = ''
FULL_BLOCK = ''
LEFT_FIVE_EIGHTHS_BLOCK = ''
LEFT_HALF_BLOCK = ''
LEFT_ONE_EIGHTH_BLOCK = ''
LEFT_ONE_QUARTER_BLOCK = ''
LEFT_SEVEN_EIGHTHS_BLOCK = ''
LEFT THREE EIGHTHS BLOCK = ''
LEFT_THREE_QUARTERS_BLOCK = ''
LIGHT_SHADE = ''
LOWER_FIVE_EIGHTHS_BLOCK = ''
LOWER HALF BLOCK = ''
LOWER_ONE_EIGHTH_BLOCK = ''
LOWER_ONE_QUARTER_BLOCK = ''
LOWER_SEVEN_EIGHTHS_BLOCK = ''
LOWER_THREE_EIGHTHS_BLOCK = ''
LOWER_THREE_QUARTERS_BLOCK = ''
MEDIUM_SHADE = ''
QUADRANT_LOWER_LEFT = ''
QUADRANT LOWER RIGHT = ''
QUADRANT_UPPER_LEFT = ''
QUADRANT_UPPER_LEFT_AND_LOWER_LEFT_AND_LOWER_RIGHT = ''
QUADRANT_UPPER_LEFT_AND_LOWER_RIGHT = ''
QUADRANT UPPER LEFT AND UPPER RIGHT AND LOWER LEFT = ''
QUADRANT_UPPER_LEFT_AND_UPPER_RIGHT_AND_LOWER_RIGHT = ''
QUADRANT_UPPER_RIGHT = ''
QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT = ''
QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT_AND_LOWER_RIGHT = ''
RIGHT_HALF_BLOCK = ''
RIGHT_ONE_EIGHTH_BLOCK = ''
UPPER_HALF_BLOCK = ''
UPPER ONE EIGHTH BLOCK = ''
```

BoxDrawings

class pygamelib.assets.graphics. BoxDrawings

Bases: object

Box drawing elements (unicode)

Here is the list of supported glyphs:

- LIGHT_HORIZONTAL = -
- HEAVY_HORIZONTAL =
- LIGHT_VERTICAL =
- HEAVY_VERTICAL =
- LIGHT_TRIPLE_DASH_HORIZONTAL =
- HEAVY_TRIPLE_DASH_HORIZONTAL =
- LIGHT_TRIPLE_DASH_VERTICAL =
- HEAVY_TRIPLE_DASH_VERTICAL =
- LIGHT_QUADRUPLE_DASH_HORIZONTAL =
- HEAVY_QUADRUPLE_DASH_HORIZONTAL =
- LIGHT_QUADRUPLE_DASH_VERTICAL =
- HEAVY_QUADRUPLE_DASH_VERTICAL =
- LIGHT_DOWN_AND_RIGHT =
- DOWN_LIGHT_AND_RIGHT_HEAVY =
- DOWN_HEAVY_AND_RIGHT_LIGHT =
- HEAVY_DOWN_AND_RIGHT =
- LIGHT_DOWN_AND_LEFT =
- DOWN_LIGHT_AND_LEFT_HEAVY =
- DOWN_HEAVY_AND_LEFT_LIGHT =
- HEAVY_DOWN_AND_LEFT =
- LIGHT_UP_AND_RIGHT = L
- UP_LIGHT_AND_RIGHT_HEAVY =
- UP_HEAVY_AND_RIGHT_LIGHT =
- HEAVY_UP_AND_RIGHT =
- LIGHT_UP_AND_LEFT =
- UP_LIGHT_AND_LEFT_HEAVY =
- UP_HEAVY_AND_LEFT_LIGHT =
- HEAVY_UP_AND_LEFT =
- LIGHT_VERTICAL_AND_RIGHT = |-
- VERTICAL_LIGHT_AND_RIGHT_HEAVY =
- UP_HEAVY_AND_RIGHT_DOWN_LIGHT =

- DOWN_HEAVY_AND_RIGHT_UP_LIGHT =
- VERTICAL_HEAVY_AND_RIGHT_LIGHT =
- DOWN_LIGHT_AND_RIGHT_UP_HEAVY =
- UP_LIGHT_AND_RIGHT_DOWN_HEAVY =
- HEAVY VERTICAL AND RIGHT =
- LIGHT VERTICAL AND LEFT =
- VERTICAL_LIGHT_AND_LEFT_HEAVY =
- UP_HEAVY_AND_LEFT_DOWN_LIGHT =
- DOWN_HEAVY_AND_LEFT_UP_LIGHT =
- VERTICAL_HEAVY_AND_LEFT_LIGHT =
- DOWN_LIGHT_AND_LEFT_UP_HEAVY =
- UP_LIGHT_AND_LEFT_DOWN_HEAVY =
- HEAVY_VERTICAL_AND_LEFT =
- LIGHT_DOWN_AND_HORIZONTAL =
- LEFT_HEAVY_AND_RIGHT_DOWN_LIGHT =
- RIGHT_HEAVY_AND_LEFT_DOWN_LIGHT =
- DOWN LIGHT AND HORIZONTAL HEAVY =
- DOWN_HEAVY_AND_HORIZONTAL_LIGHT =
- RIGHT_LIGHT_AND_LEFT_DOWN_HEAVY =
- LEFT_LIGHT_AND_RIGHT_DOWN_HEAVY =
- HEAVY_DOWN_AND_HORIZONTAL =
- LIGHT_UP_AND_HORIZONTAL =
- LEFT_HEAVY_AND_RIGHT_UP_LIGHT =
- RIGHT_HEAVY_AND_LEFT_UP_LIGHT =
- UP LIGHT AND HORIZONTAL HEAVY =
- UP_HEAVY_AND_HORIZONTAL_LIGHT =
- RIGHT_LIGHT_AND_LEFT_UP_HEAVY =
- LEFT_LIGHT_AND_RIGHT_UP_HEAVY =
- HEAVY_UP_AND_HORIZONTAL =
- LIGHT_VERTICAL_AND_HORIZONTAL =
- LEFT_HEAVY_AND_RIGHT_VERTICAL_LIGHT =
- RIGHT_HEAVY_AND_LEFT_VERTICAL_LIGHT =
- VERTICAL_LIGHT_AND_HORIZONTAL_HEAVY =
- UP_HEAVY_AND_DOWN_HORIZONTAL_LIGHT =
- DOWN_HEAVY_AND_UP_HORIZONTAL_LIGHT =
- VERTICAL_HEAVY_AND_HORIZONTAL_LIGHT =

- LEFT_UP_HEAVY_AND_RIGHT_DOWN_LIGHT =
- RIGHT_UP_HEAVY_AND_LEFT_DOWN_LIGHT =
- LEFT_DOWN_HEAVY_AND_RIGHT_UP_LIGHT =
- RIGHT_DOWN_HEAVY_AND_LEFT_UP_LIGHT =
- DOWN LIGHT AND UP HORIZONTAL HEAVY =
- UP LIGHT AND DOWN HORIZONTAL HEAVY =
- RIGHT_LIGHT_AND_LEFT_VERTICAL_HEAVY =
- LEFT_LIGHT_AND_RIGHT_VERTICAL_HEAVY =
- HEAVY_VERTICAL_AND_HORIZONTAL =
- LIGHT_DOUBLE_DASH_HORIZONTAL =
- HEAVY_DOUBLE_DASH_HORIZONTAL =
- LIGHT_DOUBLE_DASH_VERTICAL =
- HEAVY_DOUBLE_DASH_VERTICAL =
- DOUBLE_HORIZONTAL =
- DOUBLE_VERTICAL =
- DOWN SINGLE AND RIGHT DOUBLE =
- DOWN_DOUBLE_AND_RIGHT_SINGLE =
- DOUBLE_DOWN_AND_RIGHT =
- DOWN_SINGLE_AND_LEFT_DOUBLE =
- DOWN_DOUBLE_AND_LEFT_SINGLE =
- DOUBLE_DOWN_AND_LEFT =
- UP_SINGLE_AND_RIGHT_DOUBLE =
- UP_DOUBLE_AND_RIGHT_SINGLE =
- DOUBLE_UP_AND_RIGHT =
- UP_SINGLE_AND_LEFT_DOUBLE =
- UP_DOUBLE_AND_LEFT_SINGLE =
- DOUBLE_UP_AND_LEFT =
- VERTICAL_SINGLE_AND_RIGHT_DOUBLE =
- VERTICAL_DOUBLE_AND_RIGHT_SINGLE =
- DOUBLE_VERTICAL_AND_RIGHT =
- VERTICAL_SINGLE_AND_LEFT_DOUBLE =
- VERTICAL_DOUBLE_AND_LEFT_SINGLE =
- DOUBLE_VERTICAL_AND_LEFT =
- DOWN_SINGLE_AND_HORIZONTAL_DOUBLE =
- DOWN_DOUBLE_AND_HORIZONTAL_SINGLE =
- DOUBLE DOWN AND HORIZONTAL =

- UP_SINGLE_AND_HORIZONTAL_DOUBLE =
- UP_DOUBLE_AND_HORIZONTAL_SINGLE =
- DOUBLE_UP_AND_HORIZONTAL =
- VERTICAL_SINGLE_AND_HORIZONTAL_DOUBLE =
- VERTICAL_DOUBLE_AND_HORIZONTAL_SINGLE =
- DOUBLE_VERTICAL_AND_HORIZONTAL =
- LIGHT_ARC_DOWN_AND_RIGHT =
- LIGHT_ARC_DOWN_AND_LEFT =
- LIGHT_ARC_UP_AND_LEFT =
- LIGHT_ARC_UP_AND_RIGHT =
- LIGHT_DIAGONAL_UPPER_RIGHT_TO_LOWER_LEFT =
- LIGHT_DIAGONAL_UPPER_LEFT_TO_LOWER_RIGHT = \
- LIGHT_DIAGONAL_CROSS =
- LIGHT_LEFT =
- LIGHT_UP =
- LIGHT_RIGHT =
- LIGHT_DOWN =
- HEAVY_LEFT =
- HEAVY_UP =
- HEAVY_RIGHT =
- HEAVY_DOWN =
- LIGHT_LEFT_AND_HEAVY_RIGHT =
- LIGHT_UP_AND_HEAVY_DOWN =
- HEAVY_LEFT_AND_LIGHT_RIGHT =
- HEAVY_UP_AND_LIGHT_DOWN =

___init___()

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__ Initialize self.

Attributes

DOUBLE_DOWN_AND_HORIZONTAL

DOUBLE_DOWN_AND_LEFT

DOUBLE_DOWN_AND_RIGHT

DOUBLE_HORIZONTAL

Continued on next page

Table 18 – continued from previous page

Table 18 – continued from previo	ous page
DOUBLE_UP_AND_HORIZONTAL	
DOUBLE_UP_AND_LEFT	
DOUBLE_UP_AND_RIGHT	
DOUBLE_VERTICAL	
DOUBLE_VERTICAL_AND_HORIZONTAL	
DOUBLE_VERTICAL_AND_LEFT	
DOUBLE_VERTICAL_AND_RIGHT	
DOWN_DOUBLE_AND_HORIZONTAL_SINGLE	
DOWN_DOUBLE_AND_LEFT_SINGLE	
DOWN_DOUBLE_AND_RIGHT_SINGLE	
DOWN_HEAVY_AND_HORIZONTAL_LIGHT	
DOWN_HEAVY_AND_LEFT_LIGHT	
DOWN_HEAVY_AND_LEFT_UP_LIGHT	
DOWN_HEAVY_AND_RIGHT_LIGHT	
DOWN_HEAVY_AND_RIGHT_UP_LIGHT	
DOWN_HEAVY_AND_UP_HORIZONTAL_LIGHT	
DOWN_LIGHT_AND_HORIZONTAL_HEAVY	
DOWN_LIGHT_AND_LEFT_HEAVY	
DOWN_LIGHT_AND_LEFT_UP_HEAVY	
DOWN_LIGHT_AND_RIGHT_HEAVY	
DOWN_LIGHT_AND_RIGHT_UP_HEAVY	
DOWN_LIGHT_AND_UP_HORIZONTAL_HEAVY	
DOWN_SINGLE_AND_HORIZONTAL_DOUBLE	
DOWN_SINGLE_AND_LEFT_DOUBLE	
DOWN_SINGLE_AND_RIGHT_DOUBLE	
HEAVY_DOUBLE_DASH_HORIZONTAL	
HEAVY_DOUBLE_DASH_VERTICAL	
HEAVY_DOWN	
HEAVY_DOWN_AND_HORIZONTAL	
HEAVY_DOWN_AND_LEFT	
HEAVY_DOWN_AND_RIGHT	
HEAVY_HORIZONTAL	
HEAVY_LEFT	
HEAVY_LEFT_AND_LIGHT_RIGHT	
HEAVY_QUADRUPLE_DASH_HORIZONTAL	
HEAVY_QUADRUPLE_DASH_VERTICAL	
HEAVY_RIGHT	
HEAVY_TRIPLE_DASH_HORIZONTAL	
HEAVY_TRIPLE_DASH_VERTICAL	
HEAVY_UP	
HEAVY_UP_AND_HORIZONTAL	
HEAVY_UP_AND_LEFT	
HEAVY_UP_AND_LIGHT_DOWN	
HEAVY_UP_AND_RIGHT	
HEAVY_VERTICAL	
HEAVY_VERTICAL_AND_HORIZONTAL	
HEAVY_VERTICAL_AND_LEFT	
HEAVY_VERTICAL_AND_RIGHT	
LEFT_DOWN_HEAVY_AND_RIGHT_UP_LIGHT	
LEFT_HEAVY_AND_RIGHT_DOWN_LIGHT	
	Continued on next page

Table 18 – continued from previous page

Table 18 – continued from previous	page
LEFT_HEAVY_AND_RIGHT_UP_LIGHT	
LEFT_HEAVY_AND_RIGHT_VERTICAL_LIGHT	
LEFT_LIGHT_AND_RIGHT_DOWN_HEAVY	
LEFT_LIGHT_AND_RIGHT_UP_HEAVY	
LEFT_LIGHT_AND_RIGHT_VERTICAL_HEAVY	
LEFT_UP_HEAVY_AND_RIGHT_DOWN_LIGHT	
LIGHT_ARC_DOWN_AND_LEFT	
LIGHT_ARC_DOWN_AND_RIGHT	
LIGHT_ARC_UP_AND_LEFT	
LIGHT_ARC_UP_AND_RIGHT	
LIGHT_DIAGONAL_CROSS	
LIGHT_DIAGONAL_UPPER_LEFT_TO_LOWER_RIGHT	
LIGHT_DIAGONAL_UPPER_RIGHT_TO_LOWER_LEFT	
LIGHT_DOUBLE_DASH_HORIZONTAL	
LIGHT_DOUBLE_DASH_VERTICAL	
LIGHT_DOWN	
LIGHT_DOWN_AND_HORIZONTAL	
LIGHT_DOWN_AND_LEFT	
LIGHT_DOWN_AND_RIGHT	
LIGHT_HORIZONTAL	
LIGHT_LEFT	
LIGHT_LEFT_AND_HEAVY_RIGHT	
LIGHT_QUADRUPLE_DASH_HORIZONTAL	
LIGHT_QUADRUPLE_DASH_VERTICAL	
LIGHT_RIGHT	
LIGHT_TRIPLE_DASH_HORIZONTAL	
LIGHT_TRIPLE_DASH_VERTICAL	
LIGHT_UP	
LIGHT_UP_AND_HEAVY_DOWN	
LIGHT_UP_AND_HORIZONTAL	
LIGHT_UP_AND_LEFT	
LIGHT_UP_AND_RIGHT	
LIGHT_VERTICAL	
LIGHT_VERTICAL_AND_HORIZONTAL	
LIGHT_VERTICAL_AND_LEFT	
LIGHT_VERTICAL_AND_RIGHT	
RIGHT_DOWN_HEAVY_AND_LEFT_UP_LIGHT	
RIGHT_HEAVY_AND_LEFT_DOWN_LIGHT	
RIGHT_HEAVY_AND_LEFT_UP_LIGHT	
RIGHT_HEAVY_AND_LEFT_VERTICAL_LIGHT	
RIGHT_LIGHT_AND_LEFT_DOWN_HEAVY	
RIGHT_LIGHT_AND_LEFT_UP_HEAVY	
RIGHT_LIGHT_AND_LEFT_VERTICAL_HEAVY	
RIGHT_UP_HEAVY_AND_LEFT_DOWN_LIGHT	
UP_DOUBLE_AND_HORIZONTAL_SINGLE	
UP_DOUBLE_AND_LEFT_SINGLE	
UP_DOUBLE_AND_RIGHT_SINGLE	
UP_HEAVY_AND_DOWN_HORIZONTAL_LIGHT	
UP_HEAVY_AND_HORIZONTAL_LIGHT	
UP_HEAVY_AND_LEFT_DOWN_LIGHT	
	Continued on next page

Table 18 – continued from previous page

UP_HEAVY_AND_LEFT_LIGHT
UP_HEAVY_AND_RIGHT_DOWN_LIGHT
UP_HEAVY_AND_RIGHT_LIGHT
UP_LIGHT_AND_DOWN_HORIZONTAL_HEAVY
UP_LIGHT_AND_HORIZONTAL_HEAVY
UP_LIGHT_AND_LEFT_DOWN_HEAVY
UP_LIGHT_AND_LEFT_HEAVY
UP_LIGHT_AND_RIGHT_DOWN_HEAVY
UP_LIGHT_AND_RIGHT_HEAVY
UP_SINGLE_AND_HORIZONTAL_DOUBLE
UP_SINGLE_AND_LEFT_DOUBLE
UP_SINGLE_AND_RIGHT_DOUBLE
VERTICAL_DOUBLE_AND_HORIZONTAL_SINGLE
VERTICAL_DOUBLE_AND_LEFT_SINGLE
VERTICAL_DOUBLE_AND_RIGHT_SINGLE
VERTICAL_HEAVY_AND_HORIZONTAL_LIGHT
VERTICAL_HEAVY_AND_LEFT_LIGHT
VERTICAL_HEAVY_AND_RIGHT_LIGHT
VERTICAL_LIGHT_AND_HORIZONTAL_HEAVY
VERTICAL_LIGHT_AND_LEFT_HEAVY
VERTICAL_LIGHT_AND_RIGHT_HEAVY
VERTICAL_SINGLE_AND_HORIZONTAL_DOUBLE
VERTICAL_SINGLE_AND_LEFT_DOUBLE
VERTICAL_SINGLE_AND_RIGHT_DOUBLE

```
DOUBLE_DOWN_AND_HORIZONTAL = ''
DOUBLE_DOWN_AND_LEFT = ''
DOUBLE_DOWN_AND_RIGHT = ''
DOUBLE_HORIZONTAL = ''
DOUBLE_UP_AND_HORIZONTAL = ''
DOUBLE UP AND LEFT = ''
DOUBLE_UP_AND_RIGHT = ''
DOUBLE_VERTICAL = ''
DOUBLE_VERTICAL_AND_HORIZONTAL = ''
DOUBLE_VERTICAL_AND_LEFT = ''
DOUBLE_VERTICAL_AND_RIGHT = ''
DOWN_DOUBLE_AND_HORIZONTAL_SINGLE = ''
DOWN_DOUBLE_AND_LEFT_SINGLE = ''
DOWN_DOUBLE_AND_RIGHT_SINGLE = ''
DOWN_HEAVY_AND_HORIZONTAL_LIGHT = ''
DOWN_HEAVY_AND_LEFT_LIGHT = ''
DOWN_HEAVY_AND_LEFT_UP_LIGHT = ''
DOWN_HEAVY_AND_RIGHT_LIGHT = ''
```

```
DOWN HEAVY AND RIGHT UP LIGHT = ''
DOWN_HEAVY_AND_UP_HORIZONTAL_LIGHT = ''
DOWN_LIGHT_AND_HORIZONTAL_HEAVY = ''
DOWN_LIGHT_AND_LEFT_HEAVY = ''
DOWN LIGHT AND LEFT UP HEAVY = ''
DOWN LIGHT AND RIGHT HEAVY = ''
DOWN_LIGHT_AND_RIGHT_UP_HEAVY = ''
DOWN_LIGHT_AND_UP_HORIZONTAL_HEAVY = ''
DOWN_SINGLE_AND_HORIZONTAL_DOUBLE = ''
DOWN_SINGLE_AND_LEFT_DOUBLE = ''
DOWN_SINGLE_AND_RIGHT_DOUBLE = ''
HEAVY_DOUBLE_DASH_HORIZONTAL = ''
HEAVY_DOUBLE_DASH_VERTICAL = ''
HEAVY DOWN = ''
HEAVY_DOWN_AND_HORIZONTAL = ''
HEAVY DOWN AND LEFT = ''
HEAVY DOWN AND RIGHT = ''
HEAVY HORIZONTAL = ''
HEAVY_LEFT = ''
HEAVY LEFT AND LIGHT RIGHT = ''
HEAVY_QUADRUPLE_DASH_HORIZONTAL = ''
HEAVY_QUADRUPLE_DASH_VERTICAL = ''
HEAVY_RIGHT = ''
HEAVY TRIPLE DASH HORIZONTAL = ''
HEAVY TRIPLE DASH VERTICAL = ''
HEAVY_UP = ''
HEAVY_UP_AND_HORIZONTAL = ''
HEAVY_UP_AND_LEFT = ''
HEAVY UP AND LIGHT DOWN = ''
HEAVY_UP_AND_RIGHT = ''
HEAVY_VERTICAL = ''
HEAVY_VERTICAL_AND_HORIZONTAL = ''
HEAVY_VERTICAL_AND_LEFT = ''
HEAVY_VERTICAL_AND_RIGHT = ''
LEFT_DOWN_HEAVY_AND_RIGHT_UP_LIGHT = ''
LEFT_HEAVY_AND_RIGHT_DOWN_LIGHT = ''
```

```
LEFT HEAVY AND RIGHT UP LIGHT = ''
LEFT HEAVY AND RIGHT VERTICAL LIGHT = ''
LEFT_LIGHT_AND_RIGHT_DOWN_HEAVY = ''
LEFT_LIGHT_AND_RIGHT_UP_HEAVY = ''
LEFT LIGHT AND RIGHT VERTICAL HEAVY = ''
LEFT UP HEAVY AND RIGHT DOWN LIGHT = ''
LIGHT_ARC_DOWN_AND_LEFT = ''
LIGHT_ARC_DOWN_AND_RIGHT = ''
LIGHT ARC UP AND LEFT = ''
LIGHT ARC UP AND RIGHT = ''
LIGHT_DIAGONAL_CROSS = ''
LIGHT_DIAGONAL_UPPER_LEFT_TO_LOWER RIGHT = '\'
LIGHT DIAGONAL UPPER RIGHT TO LOWER LEFT = ''
LIGHT DOUBLE DASH HORIZONTAL = ''
LIGHT DOUBLE DASH VERTICAL = ''
LIGHT_DOWN = ''
LIGHT DOWN AND HORIZONTAL = ''
LIGHT DOWN AND LEFT = ''
LIGHT_DOWN_AND_RIGHT = ''
LIGHT HORIZONTAL = '-'
LIGHT LEFT = ''
LIGHT_LEFT_AND_HEAVY_RIGHT = ''
LIGHT_QUADRUPLE_DASH_HORIZONTAL = ''
LIGHT QUADRUPLE DASH VERTICAL = ''
LIGHT RIGHT = ''
LIGHT TRIPLE DASH HORIZONTAL = ''
LIGHT_TRIPLE_DASH_VERTICAL = ''
LIGHT UP = ''
LIGHT UP AND HEAVY DOWN = ''
LIGHT_UP_AND_HORIZONTAL = ''
LIGHT_UP_AND_LEFT = ''
LIGHT UP AND RIGHT = 'L'
LIGHT VERTICAL = ' '
LIGHT VERTICAL AND HORIZONTAL = ''
LIGHT_VERTICAL_AND_LEFT = ''
LIGHT_VERTICAL_AND_RIGHT = ' -'
```

```
RIGHT DOWN HEAVY AND LEFT UP LIGHT = ''
RIGHT_HEAVY_AND_LEFT_DOWN_LIGHT = ''
RIGHT_HEAVY_AND_LEFT_UP_LIGHT = ''
RIGHT_HEAVY_AND_LEFT_VERTICAL_LIGHT = ''
RIGHT LIGHT AND LEFT DOWN HEAVY = ''
RIGHT LIGHT AND LEFT UP HEAVY = ''
RIGHT_LIGHT_AND_LEFT_VERTICAL_HEAVY = ''
RIGHT_UP_HEAVY_AND_LEFT_DOWN_LIGHT = ''
UP DOUBLE AND HORIZONTAL SINGLE = ''
UP_DOUBLE_AND_LEFT_SINGLE = ''
UP_DOUBLE_AND_RIGHT_SINGLE = ''
UP_HEAVY_AND_DOWN_HORIZONTAL_LIGHT = ''
UP_HEAVY_AND_HORIZONTAL_LIGHT = ''
UP HEAVY AND LEFT DOWN LIGHT = ''
UP HEAVY AND LEFT LIGHT = ''
UP HEAVY AND RIGHT DOWN LIGHT = ''
UP HEAVY AND RIGHT LIGHT = ''
UP_LIGHT_AND_DOWN_HORIZONTAL_HEAVY = ''
UP_LIGHT_AND_HORIZONTAL_HEAVY = ''
UP LIGHT AND LEFT DOWN HEAVY = ''
UP_LIGHT_AND_LEFT_HEAVY = ''
UP_LIGHT_AND_RIGHT_DOWN_HEAVY = ''
UP_LIGHT_AND_RIGHT_HEAVY = ''
UP SINGLE AND HORIZONTAL DOUBLE = ''
UP SINGLE AND LEFT DOUBLE = ''
UP_SINGLE_AND_RIGHT_DOUBLE = ''
VERTICAL_DOUBLE_AND_HORIZONTAL_SINGLE = ''
VERTICAL DOUBLE AND LEFT SINGLE = ''
VERTICAL DOUBLE AND RIGHT SINGLE = ''
VERTICAL_HEAVY_AND_HORIZONTAL_LIGHT = ''
VERTICAL_HEAVY_AND_LEFT_LIGHT = ''
VERTICAL_HEAVY_AND_RIGHT_LIGHT = ''
VERTICAL_LIGHT_AND_HORIZONTAL_HEAVY = ''
VERTICAL_LIGHT_AND_LEFT_HEAVY = ''
VERTICAL_LIGHT_AND_RIGHT_HEAVY = ''
VERTICAL SINGLE AND HORIZONTAL DOUBLE = ''
```

```
VERTICAL_SINGLE_AND_LEFT_DOUBLE = ''
VERTICAL_SINGLE_AND_RIGHT_DOUBLE = ''
```

GeometricShapes

```
class pygamelib.assets.graphics.GeometricShapes
    Bases: object
```

Geometric shapes elements (unicode)

Here is the list of supported glyphs:

- BLACK SQUARE =
- BLACK_LARGE_SQUARE =
- WHITE_SQUARE =
- WHITE_SQUARE_WITH_ROUNDED_CORNERS =
- WHITE_SQUARE_CONTAINING_BLACK_SMALL_SQUARE =
- SQUARE_WITH_HORIZONTAL_FILL =
- SQUARE_WITH_VERTICAL_FILL =
- SQUARE_WITH_ORTHOGONAL_CROSSHATCH_FILL =
- SQUARE_WITH_UPPER_LEFT_TO_LOWER_RIGHT_FILL =
- SQUARE_WITH_UPPER_RIGHT_TO_LOWER_LEFT_FILL =
- SQUARE_WITH_DIAGONAL_CROSSHATCH_FILL =
- BLACK_SMALL_SQUARE =
- WHITE_SMALL_SQUARE =
- BLACK_RECTANGLE =
- WHITE_RECTANGLE =
- BLACK_VERTICAL_RECTANGLE =
- WHITE_VERTICAL_RECTANGLE =
- BLACK_PARALLELOGRAM =
- WHITE_PARALLELOGRAM =
- BLACK_UP_POINTING_TRIANGLE =
- WHITE_UP_POINTING_TRIANGLE =
- BLACK UP POINTING SMALL TRIANGLE =
- WHITE_UP_POINTING_SMALL_TRIANGLE =
- BLACK RIGHT POINTING TRIANGLE =
- WHITE_RIGHT_POINTING_TRIANGLE =
- BLACK_RIGHT_POINTING_SMALL_TRIANGLE =
- WHITE_RIGHT_POINTING_SMALL_TRIANGLE =
- BLACK_RIGHT_POINTING_POINTER =

- WHITE RIGHT POINTING POINTER =
- BLACK_DOWN_POINTING_TRIANGLE =
- WHITE_DOWN_POINTING_TRIANGLE =
- BLACK_DOWN_POINTING_SMALL_TRIANGLE =
- WHITE DOWN POINTING SMALL TRIANGLE =
- BLACK_LEFT_POINTING_TRIANGLE =
- WHITE_LEFT_POINTING_TRIANGLE =
- BLACK_LEFT_POINTING_SMALL_TRIANGLE =
- WHITE_LEFT_POINTING_SMALL_TRIANGLE =
- BLACK_LEFT_POINTING_POINTER =
- WHITE_LEFT_POINTING_POINTER =
- BLACK_DIAMOND =
- WHITE_DIAMOND =
- WHITE_DIAMOND_CONTAINING_BLACK_SMALL_DIAMOND =
- FISHEYE =
- LOZENGE =
- WHITE CIRCLE =
- DOTTED_CIRCLE =
- CIRCLE_WITH_VERTICAL_FILL =
- BULLSEYE =
- BLACK_CIRCLE =
- CIRCLE_WITH_LEFT_HALF_BLACK =
- CIRCLE_WITH_RIGHT_HALF_BLACK =
- CIRCLE_WITH_LOWER_HALF_BLACK =
- CIRCLE_WITH_UPPER_HALF_BLACK =
- CIRCLE_WITH_UPPER_RIGHT_QUADRANT_BLACK =
- CIRCLE_WITH_ALL_BUT_UPPER_LEFT_QUADRANT_BLACK =
- LEFT HALF BLACK CIRCLE =
- RIGHT_HALF_BLACK_CIRCLE =
- INVERSE_BULLET =
- INVERSE_WHITE_CIRCLE =
- UPPER_HALF_INVERSE_WHITE_CIRCLE =
- LOWER_HALF_INVERSE_WHITE_CIRCLE =
- UPPER_LEFT_QUADRANT_CIRCULAR_ARC =
- UPPER_RIGHT_QUADRANT_CIRCULAR_ARC =
- LOWER_RIGHT_QUADRANT_CIRCULAR_ARC =

- LOWER_LEFT_QUADRANT_CIRCULAR_ARC =
- UPPER_HALF_CIRCLE =
- LOWER_HALF_CIRCLE =
- BLACK_LOWER_RIGHT_TRIANGLE =
- BLACK LOWER LEFT TRIANGLE =
- BLACK UPPER LEFT TRIANGLE =
- BLACK_UPPER_RIGHT_TRIANGLE =
- WHITE_BULLET = \circ
- BULLET = •
- RING_OPERATOR =
- SQUARE_WITH_LEFT_HALF_BLACK =
- SQUARE_WITH_RIGHT_HALF_BLACK =
- SQUARE_WITH_UPPER_LEFT_DIAGONAL_HALF_BLACK =
- SQUARE_WITH_LOWER_RIGHT_DIAGONAL_HALF_BLACK =
- WHITE_SQUARE_WITH_VERTICAL_BISECTING_LINE =
- WHITE_UP_POINTING_TRIANGLE_WITH_DOT =
- UP_POINTING_TRIANGLE_WITH_LEFT_HALF_BLACK =
- UP_POINTING_TRIANGLE_WITH_RIGHT_HALF_BLACK =
- LARGE_CIRCLE = ()
- WHITE_SQUARE_WITH_UPPER_LEFT_QUADRANT =
- WHITE_SQUARE_WITH_LOWER_LEFT_QUADRANT =
- WHITE_SQUARE_WITH_LOWER_RIGHT_QUADRANT =
- WHITE_SQUARE_WITH_UPPER_RIGHT_QUADRANT =
- WHITE_CIRCLE_WITH_UPPER_LEFT_QUADRANT =
- WHITE_CIRCLE_WITH_LOWER_LEFT_QUADRANT =
- WHITE_CIRCLE_WITH_LOWER_RIGHT_QUADRANT =
- WHITE_CIRCLE_WITH_UPPER_RIGHT_QUADRANT =
- UPPER LEFT TRIANGLE =
- UPPER_RIGHT_TRIANGLE =
- LOWER_LEFT_TRIANGLE =
- WHITE_MEDIUM_SQUARE =
- BLACK_MEDIUM_SQUARE =
- WHITE_MEDIUM_SMALL_SQUARE =
- BLACK_MEDIUM_SMALL_SQUARE =
- LOWER_RIGHT_TRIANGLE =

```
___init___()
```

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__ Initialize self.

Attributes

BLACK CIRCLE	
BLACK DIAMOND	
BLACK DOWN POINTING SMALL TRIANGLE	
BLACK_DOWN_POINTING_TRIANGLE	
BLACK LARGE SQUARE	
BLACK LEFT POINTING POINTER	
BLACK_LEFT_POINTING_SMALL_TRIANGLE	
BLACK_LEFT_POINTING_TRIANGLE	
BLACK LOWER LEFT TRIANGLE	
BLACK_LOWER_RIGHT_TRIANGLE	
BLACK_MEDIUM_SMALL_SQUARE	
BLACK MEDIUM SQUARE	
BLACK_PARALLELOGRAM	
BLACK RECTANGLE	
BLACK_RIGHT_POINTING_POINTER	
BLACK RIGHT POINTING SMALL TRIANGLE	
BLACK_RIGHT_POINTING_TRIANGLE	
BLACK_SMALL_SQUARE	
BLACK_SQUARE	
BLACK_UPPER_LEFT_TRIANGLE	
BLACK_UPPER_RIGHT_TRIANGLE	
BLACK_UP_POINTING_SMALL_TRIANGLE	
BLACK_UP_POINTING_TRIANGLE	
BLACK_VERTICAL_RECTANGLE	
BULLET	
BULLSEYE	
CIRCLE_WITH_ALL_BUT_UPPER_LEFT_QUADRANT_BLACK	
CIRCLE_WITH_LEFT_HALF_BLACK	
CIRCLE_WITH_LOWER_HALF_BLACK	
CIRCLE_WITH_RIGHT_HALF_BLACK	
CIRCLE_WITH_UPPER_HALF_BLACK	
CIRCLE_WITH_UPPER_RIGHT_QUADRANT_BLACK	
CIRCLE_WITH_VERTICAL_FILL	
DOTTED_CIRCLE	
FISHEYE	
INVERSE_BULLET	
INVERSE_WHITE_CIRCLE	
LARGE_CIRCLE	
LEFT_HALF_BLACK_CIRCLE	
LOWER_HALF_CIRCLE	
Contin	nued on nevt nage

Continued on next page

Table 20 – continued from previous page

rable 20 – continued from previous pag	
LOWER_HALF_INVERSE_WHITE_CIRCLE	
LOWER_LEFT_QUADRANT_CIRCULAR_ARC	
LOWER_LEFT_TRIANGLE	
LOWER_RIGHT_QUADRANT_CIRCULAR_ARC	
LOWER_RIGHT_TRIANGLE	
LOZENGE	
RIGHT_HALF_BLACK_CIRCLE	
RING_OPERATOR	
SQUARE_WITH_DIAGONAL_CROSSHATCH_FILL	
SQUARE_WITH_HORIZONTAL_FILL	
SQUARE_WITH_LEFT_HALF_BLACK	
SQUARE_WITH_LOWER_RIGHT_DIAGONAL_HALF_BLACK	
SQUARE_WITH_ORTHOGONAL_CROSSHATCH_FILL	
SQUARE_WITH_RIGHT_HALF_BLACK	
SQUARE_WITH_UPPER_LEFT_DIAGONAL_HALF_BLACK	
SQUARE_WITH_UPPER_LEFT_TO_LOWER_RIGHT_FILL	
SQUARE_WITH_UPPER_RIGHT_TO_LOWER_LEFT_FILL	
SQUARE WITH VERTICAL FILL	
UPPER_HALF_CIRCLE	
UPPER HALF INVERSE WHITE CIRCLE	
UPPER LEFT QUADRANT CIRCULAR ARC	
UPPER LEFT TRIANGLE	
UPPER RIGHT QUADRANT CIRCULAR ARC	
UPPER RIGHT TRIANGLE	
UP POINTING TRIANGLE WITH LEFT HALF BLACK	
UP_POINTING_TRIANGLE_WITH_RIGHT_HALF_BLACK	
WHITE BULLET	
WHITE CIRCLE	
WHITE CIRCLE WITH LOWER LEFT QUADRANT	
WHITE CIRCLE WITH LOWER RIGHT QUADRANT	
WHITE_CIRCLE_WITH_UPPER_LEFT_QUADRANT	
WHITE_CIRCLE_WITH_UPPER_RIGHT_QUADRANT	
WHITE DIAMOND	
WHITE DIAMOND CONTAINING BLACK SMALL DIAMOND	
WHITE DOWN POINTING SMALL TRIANGLE	
WHITE_DOWN_POINTING_TRIANGLE	
WHITE LEFT POINTING POINTER	
WHITE LEFT POINTING SMALL TRIANGLE	
WHITE LEFT POINTING TRIANGLE	
WHITE MEDIUM SMALL SOUARE	
WHITE MEDIUM SOUARE	
WHITE PARALLELOGRAM	
WHITE RECTANGLE	
WHITE RIGHT POINTING POINTER	
WHITE RIGHT POINTING SMALL TRIANGLE	
WHITE RIGHT POINTING TRIANGLE WHITE RIGHT POINTING TRIANGLE	
WHITE_SMALL_SQUARE	
WHITE_SQUARE WHITE SOURCE CONTAINING BLACK SMALL SOURCE	
WHITE_SQUARE_CONTAINING_BLACK_SMALL_SQUARE	
WHITE_SQUARE_WITH_LOWER_LEFT_QUADRANT	Continued on next page

Table 20 – continued from previous page

```
WHITE_SQUARE_WITH_LOWER_RIGHT_QUADRANT

WHITE_SQUARE_WITH_ROUNDED_CORNERS

WHITE_SQUARE_WITH_UPPER_LEFT_QUADRANT

WHITE_SQUARE_WITH_UPPER_RIGHT_QUADRANT

WHITE_SQUARE_WITH_VERTICAL_BISECTING_LINE

WHITE_UP_POINTING_SMALL_TRIANGLE

WHITE_UP_POINTING_TRIANGLE_WITH_DOT

WHITE_VERTICAL_RECTANGLE
```

```
BLACK CIRCLE = ''
BLACK_DIAMOND = ''
BLACK_DOWN_POINTING_SMALL_TRIANGLE = ''
BLACK_DOWN_POINTING_TRIANGLE = ''
BLACK LARGE SQUARE = ''
BLACK LEFT POINTING POINTER = ''
BLACK_LEFT_POINTING_SMALL_TRIANGLE = ''
BLACK_LEFT_POINTING_TRIANGLE = ''
BLACK LOWER LEFT TRIANGLE = ''
BLACK LOWER RIGHT TRIANGLE = ''
BLACK_MEDIUM_SMALL_SQUARE = ''
BLACK_MEDIUM_SQUARE = ''
BLACK_PARALLELOGRAM = ''
BLACK_RECTANGLE = ''
BLACK_RIGHT_POINTING_POINTER = ''
BLACK RIGHT POINTING SMALL TRIANGLE = ''
BLACK_RIGHT_POINTING_TRIANGLE = ''
BLACK_SMALL_SQUARE = ''
BLACK_SQUARE = ''
BLACK UPPER LEFT TRIANGLE = ''
BLACK UPPER RIGHT TRIANGLE = ''
BLACK_UP_POINTING_SMALL_TRIANGLE = ''
BLACK_UP_POINTING_TRIANGLE = ''
BLACK VERTICAL RECTANGLE = ''
BULLET = '•'
BULLSEYE = ''
CIRCLE_WITH_ALL_BUT_UPPER_LEFT_QUADRANT_BLACK = ''
CIRCLE_WITH_LEFT_HALF_BLACK = ''
```

```
CIRCLE WITH LOWER HALF BLACK = ''
CIRCLE_WITH_RIGHT_HALF_BLACK = ''
CIRCLE_WITH_UPPER_HALF_BLACK = ''
CIRCLE_WITH_UPPER_RIGHT_QUADRANT_BLACK = ''
CIRCLE WITH VERTICAL FILL = ''
DOTTED CIRCLE = ''
FISHEYE = ''
INVERSE_BULLET = ''
INVERSE WHITE CIRCLE = ''
LARGE_CIRCLE = '()'
LEFT_HALF_BLACK_CIRCLE = ''
LOWER_HALF_CIRCLE = ''
LOWER HALF INVERSE WHITE CIRCLE = ''
LOWER LEFT QUADRANT CIRCULAR ARC = ''
LOWER LEFT TRIANGLE = ''
LOWER RIGHT QUADRANT CIRCULAR ARC = ''
LOWER RIGHT TRIANGLE = ''
LOZENGE = ''
RIGHT_HALF_BLACK_CIRCLE = ''
RING OPERATOR = ''
SQUARE_WITH_DIAGONAL_CROSSHATCH_FILL = ''
SQUARE_WITH_HORIZONTAL_FILL = ''
SQUARE_WITH_LEFT_HALF_BLACK = ''
SQUARE WITH LOWER RIGHT DIAGONAL HALF BLACK = ''
SQUARE WITH ORTHOGONAL CROSSHATCH FILL = ''
SQUARE_WITH_RIGHT_HALF_BLACK = ''
SQUARE_WITH_UPPER_LEFT_DIAGONAL_HALF_BLACK = ''
SQUARE_WITH_UPPER_LEFT_TO_LOWER_RIGHT_FILL = ''
SQUARE WITH UPPER RIGHT TO LOWER LEFT FILL = ''
SQUARE_WITH_VERTICAL_FILL = ''
UPPER_HALF_CIRCLE = ''
UPPER_HALF_INVERSE_WHITE_CIRCLE = ''
UPPER_LEFT_QUADRANT_CIRCULAR_ARC = ''
UPPER_LEFT_TRIANGLE = ''
UPPER_RIGHT_QUADRANT_CIRCULAR_ARC = ''
UPPER RIGHT TRIANGLE = ''
```

```
UP POINTING TRIANGLE WITH LEFT HALF BLACK = ''
UP POINTING TRIANGLE WITH RIGHT HALF BLACK = ''
WHITE BULLET = 'o'
WHITE CIRCLE = ''
WHITE CIRCLE WITH LOWER LEFT QUADRANT = ''
WHITE CIRCLE WITH LOWER RIGHT QUADRANT = ''
WHITE_CIRCLE_WITH_UPPER_LEFT_QUADRANT = ''
WHITE_CIRCLE_WITH_UPPER_RIGHT_QUADRANT = ''
WHITE DIAMOND = ''
WHITE_DIAMOND_CONTAINING_BLACK_SMALL_DIAMOND = ''
WHITE_DOWN_POINTING_SMALL_TRIANGLE = ''
WHITE_DOWN_POINTING_TRIANGLE = ''
WHITE LEFT POINTING POINTER = ''
WHITE LEFT POINTING SMALL TRIANGLE = ''
WHITE LEFT POINTING TRIANGLE = ''
WHITE MEDIUM SMALL SQUARE = ''
WHITE MEDIUM SQUARE = ''
WHITE PARALLELOGRAM = ''
WHITE_RECTANGLE = ''
WHITE RIGHT POINTING POINTER = ''
WHITE_RIGHT_POINTING_SMALL_TRIANGLE = ''
WHITE_RIGHT_POINTING_TRIANGLE = ''
WHITE_SMALL_SQUARE = ''
WHITE SQUARE = ''
WHITE SQUARE CONTAINING BLACK SMALL SQUARE = ''
WHITE_SQUARE_WITH_LOWER_LEFT_QUADRANT = ''
WHITE_SQUARE_WITH_LOWER_RIGHT_QUADRANT = ''
WHITE SQUARE WITH ROUNDED CORNERS = ''
WHITE SQUARE WITH UPPER LEFT QUADRANT = ''
WHITE_SQUARE_WITH_UPPER_RIGHT_QUADRANT = ''
WHITE_SQUARE_WITH_VERTICAL_BISECTING_LINE = ''
WHITE_UP_POINTING_SMALL_TRIANGLE = ''
WHITE_UP_POINTING_TRIANGLE = ''
WHITE_UP_POINTING_TRIANGLE_WITH_DOT = ''
WHITE_VERTICAL_RECTANGLE = ''
```

MiscTechnicals

class pygamelib.assets.graphics.MiscTechnicals

Bases: object

Miscellanous Technical block (unicode)

Here is the list of supported glyphs:

- DIAMETER_SIGN = ""
- ELECTRIC_ARROW = ""
- HOUSE = ""
- UP_ARROWHEAD = ""
- DOWN ARROWHEAD = ""
- PROJECTIVE = ""
- PERSPECTIVE = ""
- WAVY LINE = ""
- LEFT_CEILING = ""
- RIGHT_CEILING = ""
- LEFT_FLOOR = ""
- RIGHT_FLOOR = ""
- BOTTOM_RIGHT_CROP = ""
- BOTTOM_LEFT_CROP = ""
- TOP_RIGHT_CROP = ""
- TOP_LEFT_CROP = ""
- REVERSED_NOT_SIGN = ""
- SQUARE LOZENGE = ""
- ARC = ""
- SEGMENT = ""
- SECTOR = ""
- TELEPHONE_RECORDER = ""
- POSITION_INDICATOR = ""
- VIEWDATA_SQUARE = ""
- PLACE_OF_INTEREST_SIGN = ""
- TURNED_NOT_SIGN = ""
- WATCH = ""
- HOURGLASS = ""
- TOP_LEFT_CORNER = ""
- TOP RIGHT CORNER = ""
- BOTTOM_LEFT_CORNER = ""

- BOTTOM RIGHT CORNER = ""
- TOP_HALF_INTEGRAL = ""
- BOTTOM_HALF_INTEGRAL = ""
- FROWN = ""
- SMILE = ""
- UP ARROWHEAD BETWEEN TWO HORIZONTAL BARS = ""
- OPTION_KEY = ""
- ERASE_TO_THE_RIGHT = ""
- X_IN_A_RECTANGLE_BOX = ""
- KEYBOARD = ""
- LEFT_POINTING_ANGLE_BRACKET = "\"
- RIGHT_POINTING_ANGLE_BRACKET = "\"
- ERASE_TO_THE_LEFT = ""
- BENZENE RING = ""
- CYLINDRICITY = ""
- ALL_AROUND_PROFILE = ""
- **SYMMETRY** = ""
- TOTAL_RUNOUT = ""
- DIMENSION_ORIGIN = ""
- CONICAL TAPER = ""
- SLOPE = ""
- COUNTERBORE = ""
- COUNTERSINK = ""
- APL_FUNCTIONAL_SYMBOL_I_BEAM = ""
- APL_FUNCTIONAL_SYMBOL_SQUISH_QUAD = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_EQUAL = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_DIVIDE = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_DIAMOND = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_JOT = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_CIRCLE = ""
- APL_FUNCTIONAL_SYMBOL_CIRCLE_STILE = ""
- APL_FUNCTIONAL_SYMBOL_CIRCLE_JOT = ""
- APL_FUNCTIONAL_SYMBOL_SLASH_BAR = ""
- APL_FUNCTIONAL_SYMBOL_BACKSLASH_BAR = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_SLASH = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_BACKSLASH = ""

- APL FUNCTIONAL SYMBOL QUAD LESS THAN = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_GREATER_THAN = ""
- APL_FUNCTIONAL_SYMBOL_LEFTWARDS_VANE = ""
- APL_FUNCTIONAL_SYMBOL_RIGHTWARDS_VANE = ""
- APL FUNCTIONAL SYMBOL QUAD LEFTWARDS ARROW = ""
- APL FUNCTIONAL SYMBOL QUAD RIGHTWARDS ARROW = ""
- APL_FUNCTIONAL_SYMBOL_CIRCLE_BACKSLASH = ""
- APL_FUNCTIONAL_SYMBOL_DOWN_TACK_UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_DELTA_STILE = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_DOWN_CARET = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_DELTA = ""
- APL_FUNCTIONAL_SYMBOL_DOWN_TACK_JOT = ""
- APL_FUNCTIONAL_SYMBOL_UPWARDS_VANE = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_UPWARDS_ARROW = ""
- APL_FUNCTIONAL_SYMBOL_UP_TACK_OVERBAR = ""
- APL FUNCTIONAL SYMBOL DEL STILE = ""
- APL FUNCTIONAL SYMBOL QUAD UP CARET = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_DEL = ""
- APL_FUNCTIONAL_SYMBOL_UP_TACK_JOT = ""
- APL_FUNCTIONAL_SYMBOL_DOWNWARDS_VANE = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_DOWNWARDS_ARROW = ""
- APL_FUNCTIONAL_SYMBOL_QUOTE_UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_DELTA_UNDERBAR = ""
- APL FUNCTIONAL SYMBOL DIAMOND UNDERBAR = ""
- APL FUNCTIONAL SYMBOL JOT UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_CIRCLE_UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_UP_SHOE_JOT = ""
- APL FUNCTIONAL SYMBOL QUOTE QUAD = ""
- APL_FUNCTIONAL_SYMBOL_CIRCLE_STAR = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_COLON = ""
- APL_FUNCTIONAL_SYMBOL_UP_TACK_DIAERESIS = ""
- APL_FUNCTIONAL_SYMBOL_DEL_DIAERESIS = ""
- APL_FUNCTIONAL_SYMBOL_STAR_DIAERESIS = ""
- APL_FUNCTIONAL_SYMBOL_JOT_DIAERESIS = ""
- APL FUNCTIONAL SYMBOL CIRCLE DIAERESIS = ""
- APL FUNCTIONAL SYMBOL DOWN SHOE STILE = ""

- APL_FUNCTIONAL_SYMBOL_LEFT_SHOE_STILE = ""
- APL_FUNCTIONAL_SYMBOL_TILDE_DIAERESIS = ""
- APL_FUNCTIONAL_SYMBOL_GREATER_THAN_DIAERESIS = ""
- APL_FUNCTIONAL_SYMBOL_COMMA_BAR = ""
- APL FUNCTIONAL SYMBOL DEL TILDE = ""
- APL FUNCTIONAL SYMBOL ZILDE = ""
- APL_FUNCTIONAL_SYMBOL_STILE_TILDE = ""
- APL_FUNCTIONAL_SYMBOL_SEMICOLON_UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_NOT_EQUAL = ""
- APL_FUNCTIONAL_SYMBOL_QUAD_QUESTION = ""
- APL_FUNCTIONAL_SYMBOL_DOWN_CARET_TILDE = ""
- APL_FUNCTIONAL_SYMBOL_UP_CARET_TILDE = ""
- APL_FUNCTIONAL_SYMBOL_IOTA = ""
- APL FUNCTIONAL SYMBOL RHO = ""
- APL_FUNCTIONAL_SYMBOL_OMEGA = ""
- APL FUNCTIONAL SYMBOL ALPHA UNDERBAR = ""
- APL FUNCTIONAL SYMBOL EPSILON UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_IOTA_UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_OMEGA_UNDERBAR = ""
- APL_FUNCTIONAL_SYMBOL_ALPHA = ""
- NOT_CHECK_MARK = ""
- RIGHT_ANGLE_WITH_DOWNWARDS_ZIGZAG_ARROW = ""
- SHOULDERED_OPEN_BOX = ""
- BELL_SYMBOL = ""
- VERTICAL LINE WITH MIDDLE DOT = ""
- INSERTION_SYMBOL = ""
- CONTINUOUS_UNDERLINE_SYMBOL = ""
- DISCONTINUOUS UNDERLINE SYMBOL = ""
- EMPHASIS_SYMBOL = ""
- COMPOSITION_SYMBOL = ""
- WHITE_SQUARE_WITH_CENTRE_VERTICAL_LINE = ""
- ENTER_SYMBOL = ""
- ALTERNATIVE_KEY_SYMBOL = ""
- HELM_SYMBOL = ""
- CIRCLED_HORIZONTAL_BAR_WITH_NOTCH = ""
- CIRCLED TRIANGLE DOWN = ""

- BROKEN_CIRCLE_WITH_NORTHWEST_ARROW = ""
- UNDO_SYMBOL = ""
- MONOSTABLE_SYMBOL = ""
- HYSTERESIS SYMBOL = ""
- OPEN CIRCUIT OUTPUT H TYPE SYMBOL = ""
- OPEN CIRCUIT OUTPUT L TYPE SYMBOL = ""
- PASSIVE_PULL_DOWN_OUTPUT_SYMBOL = ""
- PASSIVE_PULL_UP_OUTPUT_SYMBOL = ""
- DIRECT_CURRENT_SYMBOL_FORM_TWO = ""
- SOFTWARE_FUNCTION_SYMBOL = ""
- APL_FUNCTIONAL_SYMBOL_QUAD = ""
- DECIMAL_SEPARATOR_KEY_SYMBOL = ""
- PREVIOUS_PAGE = ""
- NEXT PAGE = ""
- PRINT_SCREEN_SYMBOL = ""
- CLEAR SCREEN SYMBOL = ""
- LEFT PARENTHESIS UPPER HOOK = ""
- LEFT_PARENTHESIS_EXTENSION = ""
- LEFT_PARENTHESIS_LOWER_HOOK = ""
- RIGHT_PARENTHESIS_UPPER_HOOK = ""
- RIGHT_PARENTHESIS_EXTENSION = ""
- RIGHT_PARENTHESIS_LOWER_HOOK = ""
- LEFT_SQUARE_BRACKET_UPPER_CORNER = ""
- LEFT_SQUARE_BRACKET_EXTENSION = ""
- LEFT SQUARE BRACKET LOWER CORNER = ""
- RIGHT_SQUARE_BRACKET_UPPER_CORNER = ""
- RIGHT_SQUARE_BRACKET_EXTENSION = ""
- RIGHT SQUARE BRACKET LOWER CORNER = ""
- LEFT_CURLY_BRACKET_UPPER_HOOK = ""
- LEFT_CURLY_BRACKET_MIDDLE_PIECE = ""
- LEFT_CURLY_BRACKET_LOWER_HOOK = ""
- CURLY_BRACKET_EXTENSION = ""
- RIGHT_CURLY_BRACKET_UPPER_HOOK = ""
- RIGHT_CURLY_BRACKET_MIDDLE_PIECE = ""
- RIGHT CURLY BRACKET LOWER HOOK = ""
- INTEGRAL EXTENSION = ""

- HORIZONTAL LINE EXTENSION = ""
- UPPER_LEFT_OR_LOWER_RIGHT_CURLY_BRACKET_SECTION = ""
- UPPER_RIGHT_OR_LOWER_LEFT_CURLY_BRACKET_SECTION = ""
- SUMMATION TOP = ""
- SUMMATION BOTTOM = ""
- TOP SQUARE BRACKET = ""
- BOTTOM_SQUARE_BRACKET = ""
- BOTTOM_SQUARE_BRACKET_OVER_TOP_SQUARE_BRACKET = ""
- RADICAL SYMBOL BOTTOM = ""
- LEFT_VERTICAL_BOX_LINE = ""
- RIGHT_VERTICAL_BOX_LINE = ""
- HORIZONTAL_SCAN_LINE_1 = ""
- HORIZONTAL_SCAN_LINE_3 = ""
- HORIZONTAL SCAN LINE 7 = ""
- HORIZONTAL SCAN LINE 9 = " "
- DENTISTRY SYMBOL LIGHT VERTICAL AND TOP RIGHT = ""
- DENTISTRY SYMBOL LIGHT VERTICAL AND BOTTOM RIGHT = ""
- DENTISTRY_SYMBOL_LIGHT_VERTICAL_WITH_CIRCLE = ""
- DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_CIRCLE = ""
- DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_CIRCLE = ""
- DENTISTRY_SYMBOL_LIGHT_VERTICAL_WITH_TRIANGLE = ""
- DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_TRIANGLE = ""
- DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_TRIANGLE = ""
- DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_WAVE = ""
- DENTISTRY SYMBOL LIGHT DOWN AND HORIZONTAL WITH WAVE = ""
- DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_WAVE = ""
- DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL = ""
- DENTISTRY SYMBOL LIGHT UP AND HORIZONTAL = ""
- DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_TOP_LEFT = ""
- DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_BOTTOM_LEFT = ""
- SQUARE_FOOT = ""
- RETURN_SYMBOL = ""
- EJECT_SYMBOL = ""
- VERTICAL_LINE_EXTENSION = ""
- METRICAL BREVE = ""
- METRICAL LONG OVER SHORT = ""

- METRICAL_SHORT_OVER_LONG = ""
- METRICAL_LONG_OVER_TWO_SHORTS = ""
- METRICAL_TWO_SHORTS_OVER_LONG = ""
- METRICAL_TWO_SHORTS_JOINED = ""
- METRICAL TRISEME = ""
- METRICAL TETRASEME = ""
- METRICAL_PENTASEME = ""
- EARTH_GROUND = ""
- FUSE = ""
- TOP_PARENTHESIS = ""
- BOTTOM_PARENTHESIS = ""
- TOP_CURLY_BRACKET = ""
- BOTTOM_CURLY_BRACKET = ""
- TOP TORTOISE SHELL BRACKET = ""
- BOTTOM_TORTOISE_SHELL_BRACKET = ""
- WHITE_TRAPEZIUM = ""
- BENZENE RING WITH CIRCLE = ""
- STRAIGHTNESS = ""
- FLATNESS = ""
- AC CURRENT = ""
- ELECTRICAL_INTERSECTION = ""
- DECIMAL_EXPONENT_SYMBOL = ""
- BLACK_RIGHT_POINTING_DOUBLE_TRIANGLE = ""
- BLACK_LEFT_POINTING_DOUBLE_TRIANGLE = ""
- BLACK UP POINTING DOUBLE TRIANGLE = ""
- BLACK_DOWN_POINTING_DOUBLE_TRIANGLE = ""
- BLACK_RIGHT_POINTING_DOUBLE_TRIANGLE_WITH_VERTICAL_BAR = ""
- BLACK LEFT POINTING DOUBLE TRIANGLE WITH VERTICAL BAR = ""
- BLACK_RIGHT_POINTING_TRIANGLE_WITH_DOUBLE_VERTICAL_BAR = ""
- ALARM_CLOCK = ""
- STOPWATCH = ""
- TIMER CLOCK = ""
- HOURGLASS_WITH_FLOWING_SAND = ""
- BLACK_MEDIUM_LEFT_POINTING_TRIANGLE = ""
- BLACK_MEDIUM_RIGHT_POINTING_TRIANGLE = ""
- BLACK_MEDIUM_UP_POINTING_TRIANGLE = ""

- BLACK_MEDIUM_DOWN_POINTING_TRIANGLE = ""
- DOUBLE_VERTICAL_BAR = ""
- BLACK_SQUARE_FOR_STOP = ""
- BLACK_CIRCLE_FOR_RECORD = ""
- POWER_SYMBOL = ""
- POWER_ON_OFF_SYMBOL = ""
- POWER_ON_SYMBOL = ""
- POWER_SLEEP_SYMBOL = ""
- OBSERVER_EYE_SYMBOL = ""

Attributes

DIAMETER_SIGN	
ELECTRIC_ARROW	
HOUSE	
UP_ARROWHEAD	
DOWN_ARROWHEAD	
PROJECTIVE	
PERSPECTIVE	
WAVY_LINE	
LEFT_CEILING	
RIGHT_CEILING	
LEFT_FLOOR	
RIGHT_FLOOR	
BOTTOM_RIGHT_CROP	
BOTTOM_LEFT_CROP	
TOP_RIGHT_CROP	
TOP_LEFT_CROP	
REVERSED_NOT_SIGN	
SQUARE_LOZENGE	
ARC	
SEGMENT	
SECTOR	
TELEPHONE_RECORDER	
POSITION_INDICATOR	
VIEWDATA_SQUARE	
PLACE_OF_INTEREST_SIGN	
TURNED_NOT_SIGN	
WATCH	
HOURGLASS	
TOP_LEFT_CORNER	
TOP_RIGHT_CORNER	
BOTTOM_LEFT_CORNER	
BOTTOM_RIGHT_CORNER	
TOP_HALF_INTEGRAL	
BOTTOM_HALF_INTEGRAL	
	0

Continued on next page

Table 21 – continued from previous page

Table 21 – continued from previous pa	ige
FROWN	
SMILE	
UP_ARROWHEAD_BETWEEN_TWO_HORIZONTAL_BARS	
OPTION_KEY	
ERASE_TO_THE_RIGHT	
X_IN_A_RECTANGLE_BOX	
KEYBOARD	
LEFT_POINTING_ANGLE_BRACKET	
RIGHT_POINTING_ANGLE_BRACKET	
ERASE_TO_THE_LEFT	
BENZENE_RING	
CYLINDRICITY	
ALL_AROUND_PROFILE	
SYMMETRY	
TOTAL_RUNOUT	
DIMENSION_ORIGIN	
CONICAL_TAPER	
SLOPE	
COUNTERBORE	
COUNTERSINK	
APL_FUNCTIONAL_SYMBOL_I_BEAM	
APL_FUNCTIONAL_SYMBOL_SQUISH_QUAD	
APL_FUNCTIONAL_SYMBOL_QUAD_EQUAL	
APL_FUNCTIONAL_SYMBOL_QUAD_DIVIDE	
APL_FUNCTIONAL_SYMBOL_QUAD_DIAMOND	
APL_FUNCTIONAL_SYMBOL_QUAD_JOT	
APL_FUNCTIONAL_SYMBOL_QUAD_CIRCLE	
APL_FUNCTIONAL_SYMBOL_CIRCLE_STILE	
APL_FUNCTIONAL_SYMBOL_CIRCLE_JOT	
APL_FUNCTIONAL_SYMBOL_SLASH_BAR	
APL_FUNCTIONAL_SYMBOL_BACKSLASH_BAR	
APL_FUNCTIONAL_SYMBOL_QUAD_SLASH	
APL_FUNCTIONAL_SYMBOL_QUAD_BACKSLASH	
APL_FUNCTIONAL_SYMBOL_QUAD_LESS_THAN	
APL_FUNCTIONAL_SYMBOL_QUAD_GREATER_THAN	
APL_FUNCTIONAL_SYMBOL_LEFTWARDS_VANE	
APL_FUNCTIONAL_SYMBOL_RIGHTWARDS_VANE	
APL_FUNCTIONAL_SYMBOL_QUAD_LEFTWARDS_ARROW	
APL_FUNCTIONAL_SYMBOL_QUAD_RIGHTWARDS_ARROW	
APL_FUNCTIONAL_SYMBOL_CIRCLE_BACKSLASH	
APL FUNCTIONAL SYMBOL DOWN TACK UNDERBAR	
APL FUNCTIONAL SYMBOL DELTA STILE	
APL FUNCTIONAL SYMBOL QUAD DOWN CARET	
APL_FUNCTIONAL_SYMBOL_QUAD_DELTA	
APL_FUNCTIONAL_SYMBOL_DOWN_TACK_JOT	
APL_FUNCTIONAL_SYMBOL_UPWARDS_VANE	
APL_FUNCTIONAL_SYMBOL_QUAD_UPWARDS_ARROW	
APL_FUNCTIONAL_SYMBOL_UP_TACK_OVERBAR	
APL FUNCTIONAL SYMBOL DEL STILE	
APL_FUNCTIONAL_SYMBOL_QUAD_UP_CARET	
	Continued on next page

Table 21 – continued from previous page

Table 21 – continued from previous page
APL_FUNCTIONAL_SYMBOL_QUAD_DEL
APL_FUNCTIONAL_SYMBOL_UP_TACK_JOT
APL_FUNCTIONAL_SYMBOL_DOWNWARDS_VANE
APL_FUNCTIONAL_SYMBOL_QUAD_DOWNWARDS_ARROW
APL_FUNCTIONAL_SYMBOL_QUOTE_UNDERBAR
APL_FUNCTIONAL_SYMBOL_DELTA_UNDERBAR
APL_FUNCTIONAL_SYMBOL_DIAMOND_UNDERBAR
APL_FUNCTIONAL_SYMBOL_JOT_UNDERBAR
APL_FUNCTIONAL_SYMBOL_CIRCLE_UNDERBAR
APL_FUNCTIONAL_SYMBOL_UP_SHOE_JOT
APL_FUNCTIONAL_SYMBOL_QUOTE_QUAD
APL_FUNCTIONAL_SYMBOL_CIRCLE_STAR
APL_FUNCTIONAL_SYMBOL_QUAD_COLON
APL_FUNCTIONAL_SYMBOL_UP_TACK_DIAERESIS
APL_FUNCTIONAL_SYMBOL_DEL_DIAERESIS
APL_FUNCTIONAL_SYMBOL_STAR_DIAERESIS
APL_FUNCTIONAL_SYMBOL_JOT_DIAERESIS
APL_FUNCTIONAL_SYMBOL_CIRCLE_DIAERESIS
APL_FUNCTIONAL_SYMBOL_DOWN_SHOE_STILE
APL_FUNCTIONAL_SYMBOL_LEFT_SHOE_STILE
APL_FUNCTIONAL_SYMBOL_TILDE_DIAERESIS
APL_FUNCTIONAL_SYMBOL_GREATER_THAN_DIAERESIS
APL_FUNCTIONAL_SYMBOL_COMMA_BAR
APL_FUNCTIONAL_SYMBOL_DEL_TILDE
APL_FUNCTIONAL_SYMBOL_ZILDE
APL_FUNCTIONAL_SYMBOL_STILE_TILDE
APL_FUNCTIONAL_SYMBOL_SEMICOLON_UNDERBAR
APL_FUNCTIONAL_SYMBOL_QUAD_NOT_EQUAL
APL_FUNCTIONAL_SYMBOL_QUAD_QUESTION
APL_FUNCTIONAL_SYMBOL_DOWN_CARET_TILDE
APL_FUNCTIONAL_SYMBOL_UP_CARET_TILDE
APL_FUNCTIONAL_SYMBOL_IOTA
APL_FUNCTIONAL_SYMBOL_RHO
APL_FUNCTIONAL_SYMBOL_OMEGA
APL_FUNCTIONAL_SYMBOL_ALPHA_UNDERBAR
APL_FUNCTIONAL_SYMBOL_EPSILON_UNDERBAR
APL_FUNCTIONAL_SYMBOL_IOTA_UNDERBAR
APL_FUNCTIONAL_SYMBOL_OMEGA_UNDERBAR
APL_FUNCTIONAL_SYMBOL_ALPHA
NOT_CHECK_MARK
RIGHT_ANGLE_WITH_DOWNWARDS_ZIGZAG_ARROW
SHOULDERED_OPEN_BOX
BELL_SYMBOL
VERTICAL_LINE_WITH_MIDDLE_DOT
INSERTION_SYMBOL
CONTINUOUS_UNDERLINE_SYMBOL
DISCONTINUOUS_UNDERLINE_SYMBOL
EMPHASIS_SYMBOL
COMPOSITION_SYMBOL
WHITE_SQUARE_WITH_CENTRE_VERTICAL_LINE
Continued on next page

Table 21 – continued from previous page

Table 21 – continued from previous pag	<u> </u>
ENTER_SYMBOL	
ALTERNATIVE_KEY_SYMBOL	
HELM_SYMBOL	
CIRCLED_HORIZONTAL_BAR_WITH_NOTCH	
CIRCLED_TRIANGLE_DOWN	
BROKEN_CIRCLE_WITH_NORTHWEST_ARROW	
UNDO_SYMBOL	
MONOSTABLE_SYMBOL	
HYSTERESIS_SYMBOL	
OPEN_CIRCUIT_OUTPUT_H_TYPE_SYMBOL	
OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL	
PASSIVE_PULL_DOWN_OUTPUT_SYMBOL	
PASSIVE_PULL_UP_OUTPUT_SYMBOL	
DIRECT_CURRENT_SYMBOL_FORM_TWO	
SOFTWARE_FUNCTION_SYMBOL	
APL_FUNCTIONAL_SYMBOL_QUAD	
DECIMAL_SEPARATOR_KEY_SYMBOL	
PREVIOUS_PAGE	
NEXT_PAGE	
PRINT_SCREEN_SYMBOL	
CLEAR_SCREEN_SYMBOL	
LEFT_PARENTHESIS_UPPER_HOOK	
LEFT_PARENTHESIS_EXTENSION	
LEFT_PARENTHESIS_LOWER_HOOK	
RIGHT_PARENTHESIS_UPPER_HOOK	
RIGHT_PARENTHESIS_EXTENSION	
RIGHT_PARENTHESIS_LOWER_HOOK	
LEFT_SQUARE_BRACKET_UPPER_CORNER	
LEFT_SQUARE_BRACKET_EXTENSION	
LEFT_SQUARE_BRACKET_LOWER_CORNER	
RIGHT_SQUARE_BRACKET_UPPER_CORNER	
RIGHT_SQUARE_BRACKET_EXTENSION	
RIGHT_SQUARE_BRACKET_LOWER_CORNER	
LEFT_CURLY_BRACKET_UPPER_HOOK	
LEFT_CURLY_BRACKET_MIDDLE_PIECE	
LEFT_CURLY_BRACKET_LOWER_HOOK	
CURLY_BRACKET_EXTENSION	
RIGHT_CURLY_BRACKET_UPPER_HOOK	
RIGHT_CURLY_BRACKET_MIDDLE_PIECE	
RIGHT_CURLY_BRACKET_LOWER_HOOK	
INTEGRAL_EXTENSION	
HORIZONTAL_LINE_EXTENSION	
UPPER_LEFT_OR_LOWER_RIGHT_CURLY_BRACKET_SECTION	
UPPER_RIGHT_OR_LOWER_LEFT_CURLY_BRACKET_SECTION	
SUMMATION_TOP	
SUMMATION_BOTTOM	
TOP_SQUARE_BRACKET	
BOTTOM SQUARE BRACKET	
BOTTOM SQUARE BRACKET OVER TOP SQUARE BRACKET	
RADICAL SYMBOL BOTTOM	
	Continued on next page

Table 21 – continued from previous page

Table 21 – continued from previous page
LEFT_VERTICAL_BOX_LINE
RIGHT_VERTICAL_BOX_LINE
HORIZONTAL_SCAN_LINE_1
HORIZONTAL_SCAN_LINE_3
HORIZONTAL_SCAN_LINE_7
HORIZONTAL_SCAN_LINE_9
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_TOP_RIGHT
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_BOTTOM_RIGHT
DENTISTRY_SYMBOL_LIGHT_VERTICAL_WITH_CIRCLE
_DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_CIRCLE
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_CIRCLE
DENTISTRY_SYMBOL_LIGHT_VERTICAL_WITH_TRIANGLE
DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_TRIANGLE
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_TRIANGLE
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_WAVE
DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_WAVE
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_WAVE
DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_TOP_LEFT
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_BOTTOM_LEFT
SQUARE_FOOT
RETURN_SYMBOL
EJECT_SYMBOL
VERTICAL_LINE_EXTENSION
METRICAL_BREVE
METRICAL_LONG_OVER_SHORT
METRICAL_SHORT_OVER_LONG
METRICAL_LONG_OVER_TWO_SHORTS
METRICAL_TWO_SHORTS_OVER_LONG
METRICAL_TWO_SHORTS_JOINED
METRICAL_TRISEME
METRICAL_TETRASEME
METRICAL_PENTASEME
EARTH_GROUND
FUSE
TOP_PARENTHESIS
BOTTOM_PARENTHESIS
TOP_CURLY_BRACKET
BOTTOM_CURLY_BRACKET
TOP_TORTOISE_SHELL_BRACKET
BOTTOM_TORTOISE_SHELL_BRACKET
WHITE_TRAPEZIUM
BENZENE_RING_WITH_CIRCLE
STRAIGHTNESS
FLATNESS
AC_CURRENT
ELECTRICAL_INTERSECTION
DECIMAL_EXPONENT_SYMBOL
BLACK_RIGHT_POINTING_DOUBLE_TRIANGLE
Continued on next page

Table 21 – continued from previous page

```
BLACK_LEFT_POINTING_DOUBLE_TRIANGLE
BLACK UP POINTING DOUBLE TRIANGLE
BLACK_DOWN_POINTING_DOUBLE_TRIANGLE
BLACK_RIGHT_POINTING_DOUBLE_TRIANGLE_WITH_VERTICAL_BAR
BLACK_LEFT_POINTING_DOUBLE_TRIANGLE_WITH_VERTICAL_BAR
BLACK_RIGHT_POINTING_TRIANGLE_WITH_DOUBLE_VERTICAL_BAR
ALARM CLOCK
STOPWATCH
TIMER_CLOCK
HOURGLASS_WITH_FLOWING_SAND
BLACK_MEDIUM_LEFT_POINTING_TRIANGLE
BLACK_MEDIUM_RIGHT_POINTING_TRIANGLE
BLACK_MEDIUM_UP_POINTING_TRIANGLE
BLACK MEDIUM DOWN POINTING TRIANGLE
DOUBLE_VERTICAL_BAR
BLACK_SQUARE_FOR_STOP
BLACK CIRCLE FOR RECORD
POWER_SYMBOL
POWER_ON_OFF_SYMBOL
POWER_ON_SYMBOL
POWER SLEEP SYMBOL
OBSERVER_EYE_SYMBOL
```

```
AC CURRENT = ''
ALARM_CLOCK = ''
ALL_AROUND_PROFILE = ''
ALTERNATIVE_KEY_SYMBOL = ''
APL_FUNCTIONAL_SYMBOL_ALPHA = ''
APL_FUNCTIONAL_SYMBOL_ALPHA_UNDERBAR = ''
APL_FUNCTIONAL_SYMBOL_BACKSLASH_BAR = ''
APL FUNCTIONAL SYMBOL CIRCLE BACKSLASH = ''
APL FUNCTIONAL SYMBOL CIRCLE DIAERESIS = ''
APL FUNCTIONAL SYMBOL CIRCLE JOT = ''
APL_FUNCTIONAL_SYMBOL_CIRCLE_STAR = ''
APL_FUNCTIONAL_SYMBOL_CIRCLE_STILE = ''
APL FUNCTIONAL SYMBOL CIRCLE UNDERBAR = ''
APL_FUNCTIONAL_SYMBOL_COMMA_BAR = ''
APL_FUNCTIONAL_SYMBOL_DELTA_STILE = ''
APL_FUNCTIONAL_SYMBOL_DELTA_UNDERBAR = ''
APL_FUNCTIONAL_SYMBOL_DEL_DIAERESIS = ''
APL FUNCTIONAL SYMBOL DEL STILE = ''
APL_FUNCTIONAL_SYMBOL_DEL_TILDE = ''
APL FUNCTIONAL SYMBOL DIAMOND UNDERBAR = ''
```

```
APL FUNCTIONAL SYMBOL DOWNWARDS VANE = ''
APL FUNCTIONAL SYMBOL DOWN CARET TILDE = ''
APL_FUNCTIONAL_SYMBOL_DOWN_SHOE_STILE = ''
APL_FUNCTIONAL_SYMBOL_DOWN_TACK_JOT = ''
APL FUNCTIONAL SYMBOL DOWN TACK UNDERBAR = ''
APL FUNCTIONAL SYMBOL EPSILON UNDERBAR = ''
APL_FUNCTIONAL_SYMBOL_GREATER_THAN_DIAERESIS = ''
APL_FUNCTIONAL_SYMBOL_IOTA = ''
APL FUNCTIONAL SYMBOL IOTA UNDERBAR = ''
APL_FUNCTIONAL_SYMBOL_I_BEAM = ''
APL_FUNCTIONAL_SYMBOL_JOT_DIAERESIS = ''
APL_FUNCTIONAL_SYMBOL_JOT_UNDERBAR = ''
APL FUNCTIONAL SYMBOL LEFTWARDS VANE = ''
APL FUNCTIONAL SYMBOL LEFT SHOE STILE = ''
APL FUNCTIONAL SYMBOL OMEGA = ''
APL FUNCTIONAL SYMBOL OMEGA UNDERBAR = ''
APL FUNCTIONAL SYMBOL QUAD = ''
APL_FUNCTIONAL_SYMBOL_QUAD_BACKSLASH = ''
APL_FUNCTIONAL_SYMBOL_QUAD_CIRCLE = ''
APL FUNCTIONAL SYMBOL QUAD COLON = ''
APL_FUNCTIONAL_SYMBOL_QUAD_DEL = ''
APL_FUNCTIONAL_SYMBOL_QUAD_DELTA = ''
APL_FUNCTIONAL_SYMBOL_QUAD_DIAMOND = ''
APL FUNCTIONAL SYMBOL QUAD DIVIDE = ''
APL FUNCTIONAL SYMBOL QUAD DOWNWARDS ARROW = ''
APL_FUNCTIONAL_SYMBOL_QUAD_DOWN_CARET = ''
APL_FUNCTIONAL_SYMBOL_QUAD_EQUAL = ''
APL FUNCTIONAL SYMBOL QUAD GREATER THAN = ''
APL FUNCTIONAL SYMBOL QUAD JOT = ''
APL_FUNCTIONAL_SYMBOL_QUAD_LEFTWARDS_ARROW = ''
APL_FUNCTIONAL_SYMBOL_QUAD_LESS_THAN = ''
APL_FUNCTIONAL_SYMBOL_QUAD_NOT_EQUAL = ''
APL_FUNCTIONAL_SYMBOL_QUAD_QUESTION = ''
APL_FUNCTIONAL_SYMBOL_QUAD_RIGHTWARDS_ARROW = ''
APL_FUNCTIONAL_SYMBOL_QUAD_SLASH = ''
APL FUNCTIONAL SYMBOL QUAD UPWARDS ARROW = ''
```

```
APL FUNCTIONAL SYMBOL QUAD UP CARET = ''
APL FUNCTIONAL SYMBOL QUOTE QUAD = ''
APL_FUNCTIONAL_SYMBOL_QUOTE_UNDERBAR = ''
APL FUNCTIONAL SYMBOL RHO = ''
APL FUNCTIONAL SYMBOL RIGHTWARDS VANE = ''
APL FUNCTIONAL SYMBOL SEMICOLON UNDERBAR = ''
APL_FUNCTIONAL_SYMBOL_SLASH_BAR = ''
APL_FUNCTIONAL_SYMBOL_SQUISH_QUAD = ''
APL FUNCTIONAL SYMBOL STAR DIAERESIS = ''
APL FUNCTIONAL SYMBOL STILE TILDE = ''
APL_FUNCTIONAL_SYMBOL_TILDE_DIAERESIS = ''
APL_FUNCTIONAL_SYMBOL_UPWARDS_VANE = ''
APL FUNCTIONAL SYMBOL UP CARET TILDE = ''
APL FUNCTIONAL SYMBOL UP SHOE JOT = ''
APL FUNCTIONAL SYMBOL UP TACK DIAERESIS = ''
APL FUNCTIONAL SYMBOL UP TACK JOT = ''
APL FUNCTIONAL SYMBOL UP TACK OVERBAR = ''
APL FUNCTIONAL SYMBOL ZILDE = ''
ARC = ''
BELL SYMBOL = ''
BENZENE RING = ''
BENZENE_RING_WITH_CIRCLE = ''
BLACK_CIRCLE_FOR_RECORD = ''
BLACK DOWN POINTING DOUBLE TRIANGLE = ''
BLACK LEFT POINTING DOUBLE TRIANGLE = ''
BLACK LEFT POINTING DOUBLE TRIANGLE WITH VERTICAL BAR = ''
BLACK_MEDIUM_DOWN_POINTING_TRIANGLE = ''
BLACK MEDIUM LEFT POINTING TRIANGLE = ''
BLACK MEDIUM RIGHT POINTING TRIANGLE = ''
BLACK_MEDIUM_UP_POINTING_TRIANGLE = ''
BLACK_RIGHT_POINTING_DOUBLE_TRIANGLE = ''
BLACK RIGHT POINTING DOUBLE TRIANGLE WITH VERTICAL BAR = ''
BLACK_RIGHT_POINTING_TRIANGLE_WITH_DOUBLE_VERTICAL_BAR = ''
BLACK SQUARE FOR STOP = ''
BLACK_UP_POINTING_DOUBLE_TRIANGLE = ''
BOTTOM CURLY BRACKET = ''
```

```
BOTTOM HALF INTEGRAL = ''
BOTTOM LEFT CORNER = ''
BOTTOM LEFT CROP = ''
BOTTOM PARENTHESIS = ''
BOTTOM RIGHT CORNER = ''
BOTTOM RIGHT CROP = ''
BOTTOM_SQUARE_BRACKET = ''
BOTTOM_SQUARE_BRACKET_OVER_TOP_SQUARE_BRACKET = ''
BOTTOM TORTOISE SHELL BRACKET = ''
BROKEN_CIRCLE_WITH_NORTHWEST_ARROW = ''
CIRCLED_HORIZONTAL_BAR_WITH_NOTCH = ''
CIRCLED_TRIANGLE_DOWN = ''
CLEAR SCREEN SYMBOL = ''
COMPOSITION SYMBOL = ''
CONICAL TAPER = ''
CONTINUOUS UNDERLINE SYMBOL = ''
COUNTERBORE = ''
COUNTERSINK = ''
CURLY BRACKET EXTENSION = ''
CYLINDRICITY = ''
DECIMAL_EXPONENT_SYMBOL = ''
DECIMAL_SEPARATOR_KEY_SYMBOL = ''
DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL = ''
DENTISTRY SYMBOL LIGHT DOWN AND HORIZONTAL WITH CIRCLE = ''
DENTISTRY SYMBOL LIGHT DOWN AND HORIZONTAL WITH TRIANGLE = ''
DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_WAVE = ''
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL = ''
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_CIRCLE = ''
DENTISTRY SYMBOL LIGHT UP AND HORIZONTAL WITH TRIANGLE = ''
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_WAVE = ''
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_BOTTOM_LEFT = ''
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_BOTTOM_RIGHT = ''
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_TOP_LEFT = ''
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_TOP_RIGHT = ''
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_WAVE = ''
DENTISTRY_SYMBOL_LIGHT_VERTICAL_WITH_CIRCLE = ''
```

```
DENTISTRY SYMBOL LIGHT VERTICAL WITH TRIANGLE = ''
DIAMETER SIGN = ''
DIMENSION_ORIGIN = ''
DIRECT_CURRENT_SYMBOL_FORM_TWO = ''
DISCONTINUOUS UNDERLINE SYMBOL = ''
DOUBLE VERTICAL BAR = ''
DOWN_ARROWHEAD = ''
EARTH_GROUND = ''
EJECT SYMBOL = ''
ELECTRICAL INTERSECTION = ''
ELECTRIC_ARROW = ''
EMPHASIS_SYMBOL = ''
ENTER_SYMBOL = ''
ERASE TO THE LEFT = ''
ERASE TO THE RIGHT = ''
FLATNESS = ''
FROWN = ''
FUSE = ''
HELM_SYMBOL = ''
HORIZONTAL LINE EXTENSION = ''
HORIZONTAL_SCAN_LINE_1 = ''
HORIZONTAL_SCAN_LINE_3 = ''
HORIZONTAL_SCAN_LINE_7 = ''
HORIZONTAL SCAN LINE 9 = ' '
HOURGLASS = ''
HOURGLASS_WITH_FLOWING_SAND = ''
HOUSE = ''
HYSTERESIS SYMBOL = ''
INSERTION SYMBOL = ''
INTEGRAL_EXTENSION = ''
KEYBOARD = ''
LEFT_CEILING = ''
LEFT_CURLY_BRACKET_LOWER_HOOK = ''
LEFT_CURLY_BRACKET_MIDDLE_PIECE = ''
LEFT_CURLY_BRACKET_UPPER_HOOK = ''
LEFT FLOOR = ''
```

```
LEFT PARENTHESIS EXTENSION = ''
LEFT_PARENTHESIS_LOWER_HOOK = ''
LEFT_PARENTHESIS_UPPER_HOOK = ''
LEFT_POINTING_ANGLE_BRACKET = '('
LEFT SQUARE BRACKET EXTENSION = ''
LEFT SQUARE BRACKET LOWER CORNER = ''
LEFT_SQUARE_BRACKET_UPPER_CORNER = ''
LEFT_VERTICAL_BOX_LINE = ''
METRICAL BREVE = ''
METRICAL_LONG_OVER_SHORT = ''
METRICAL_LONG_OVER_TWO_SHORTS = ''
METRICAL_PENTASEME = ''
METRICAL_SHORT_OVER_LONG = ''
METRICAL TETRASEME = ''
METRICAL TRISEME = ''
METRICAL TWO SHORTS JOINED = ''
METRICAL_TWO_SHORTS_OVER_LONG = ''
MONOSTABLE_SYMBOL = ''
NEXT_PAGE = ''
NOT CHECK MARK = ''
OBSERVER_EYE_SYMBOL = ''
OPEN_CIRCUIT_OUTPUT_H_TYPE_SYMBOL = ''
OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL = ''
OPTION KEY = ''
PASSIVE PULL DOWN OUTPUT SYMBOL = ''
PASSIVE_PULL_UP_OUTPUT_SYMBOL = ''
PERSPECTIVE = ''
PLACE_OF_INTEREST_SIGN = ''
POSITION INDICATOR = ''
POWER_ON_OFF_SYMBOL = ''
POWER_ON_SYMBOL = ''
POWER_SLEEP_SYMBOL = ''
POWER_SYMBOL = ''
PREVIOUS_PAGE = ''
PRINT_SCREEN_SYMBOL = ''
PROJECTIVE = ''
```

```
RADICAL SYMBOL BOTTOM = ''
RETURN SYMBOL = ''
REVERSED_NOT_SIGN = ''
RIGHT_ANGLE_WITH_DOWNWARDS_ZIGZAG_ARROW = ''
RIGHT CEILING = ''
RIGHT CURLY BRACKET LOWER HOOK = ''
RIGHT_CURLY_BRACKET_MIDDLE_PIECE = ''
RIGHT_CURLY_BRACKET_UPPER_HOOK = ''
RIGHT FLOOR = ''
RIGHT_PARENTHESIS_EXTENSION = ''
RIGHT_PARENTHESIS_LOWER_HOOK = ''
RIGHT_PARENTHESIS_UPPER_HOOK = ''
RIGHT_POINTING_ANGLE_BRACKET = ')'
RIGHT SQUARE BRACKET EXTENSION = ''
RIGHT_SQUARE_BRACKET_LOWER_CORNER = ''
RIGHT SQUARE BRACKET UPPER CORNER = ''
RIGHT VERTICAL BOX LINE = ''
SECTOR = ''
SEGMENT = ''
SHOULDERED OPEN BOX = ''
SLOPE = ''
SMILE = ''
SOFTWARE_FUNCTION_SYMBOL = ''
SQUARE FOOT = ''
SQUARE LOZENGE = ''
STOPWATCH = ''
STRAIGHTNESS = ''
SUMMATION_BOTTOM = ''
SUMMATION TOP = ''
SYMMETRY = ''
TELEPHONE_RECORDER = ''
TIMER_CLOCK = ''
TOP_CURLY_BRACKET = ''
TOP_HALF_INTEGRAL = ''
TOP_LEFT_CORNER = ''
TOP LEFT CROP = ''
```

```
TOP PARENTHESIS = ''
TOP RIGHT CORNER = ''
TOP_RIGHT_CROP = ''
TOP_SQUARE_BRACKET = ''
TOP TORTOISE SHELL BRACKET = ''
TOTAL RUNOUT = ''
TURNED_NOT_SIGN = ''
UNDO_SYMBOL = ''
UPPER_LEFT_OR_LOWER_RIGHT_CURLY_BRACKET_SECTION = ''
UPPER_RIGHT_OR_LOWER_LEFT_CURLY_BRACKET_SECTION = ''
UP_ARROWHEAD = ''
UP_ARROWHEAD_BETWEEN_TWO_HORIZONTAL_BARS = ''
VERTICAL_LINE_EXTENSION = ''
VERTICAL_LINE_WITH_MIDDLE_DOT = ''
VIEWDATA SQUARE = ''
WATCH = ''
WAVY LINE = ''
WHITE_SQUARE_WITH_CENTRE_VERTICAL_LINE = ''
WHITE_TRAPEZIUM = ''
X IN A RECTANGLE BOX = ''
```

Models

```
\begin{tabular}{ll} \textbf{class} & \textbf{pygamelib.assets.graphics.Models} \\ & \textbf{Bases:} & \textbf{object} \\ \end{tabular}
```

List of models (emojis by unicode denomination)

Models are filtered emojis. This class does not map the entire specification.

Models replaces the previous Sprites class. Renaming that class is necessary with the introduction of a real Sprite class in the GFX module.

This class contains 1328 emojis (this is not the full list). All emoji codes come from: https://unicode.org/emoji/charts/full_emoji_list.html Additional emojis can be added by codes.

The complete list of aliased emojis is:

- GRINNING_FACE =
- GRINNING_FACE_WITH_BIG_EYES =
- GRINNING_FACE_WITH_SMILING_EYES =
- BEAMING_FACE_WITH_SMILING_EYES =
- GRINNING_SQUINTING_FACE =
- GRINNING_FACE_WITH_SWEAT =

- ROLLING_ON_THE_FLOOR_LAUGHING =
- FACE_WITH_TEARS_OF_JOY =
- SLIGHTLY_SMILING_FACE =
- UPSIDE_DOWN_FACE =
- WINKING FACE =
- SMILING_FACE_WITH_SMILING_EYES =
- SMILING_FACE_WITH_HALO =
- SMILING_FACE_WITH_HEARTS =
- SMILING_FACE_WITH_HEART_EYES =
- STAR_STRUCK =
- FACE_BLOWING_A_KISS =
- KISSING_FACE =
- SMILING_FACE =
- KISSING_FACE_WITH_CLOSED_EYES =
- KISSING_FACE_WITH_SMILING_EYES =
- SMILING_FACE_WITH_TEAR =
- FACE_SAVORING_FOOD =
- FACE_WITH_TONGUE =
- WINKING_FACE_WITH_TONGUE =
- ZANY_FACE =
- SQUINTING_FACE_WITH_TONGUE =
- MONEY_MOUTH_FACE =
- HUGGING_FACE =
- FACE_WITH_HAND_OVER_MOUTH =
- SHUSHING_FACE =
- THINKING_FACE =
- ZIPPER_MOUTH_FACE =
- FACE_WITH_RAISED_EYEBROW =
- NEUTRAL_FACE =
- EXPRESSIONLESS_FACE =
- FACE_WITHOUT_MOUTH =
- SMIRKING_FACE =
- UNAMUSED_FACE =
- FACE_WITH_ROLLING_EYES =
- GRIMACING_FACE =
- LYING_FACE =

- RELIEVED_FACE =
- PENSIVE_FACE =
- SLEEPY_FACE =
- DROOLING_FACE =
- SLEEPING_FACE =
- FACE_WITH_MEDICAL_MASK =
- FACE_WITH_THERMOMETER =
- FACE_WITH_HEAD_BANDAGE =
- NAUSEATED_FACE =
- FACE_VOMITING =
- SNEEZING_FACE =
- HOT_FACE =
- COLD_FACE =
- WOOZY_FACE =
- DIZZY_FACE =
- EXPLODING_HEAD =
- COWBOY_HAT_FACE =
- PARTYING_FACE =
- DISGUISED_FACE =
- SMILING_FACE_WITH_SUNGLASSES =
- NERD_FACE =
- FACE_WITH_MONOCLE =
- CONFUSED_FACE =
- WORRIED_FACE =
- SLIGHTLY_FROWNING_FACE =
- FROWNING_FACE =
- FACE_WITH_OPEN_MOUTH =
- HUSHED_FACE =
- ASTONISHED_FACE =
- FLUSHED_FACE =
- PLEADING_FACE =
- FROWNING_FACE_WITH_OPEN_MOUTH =
- ANGUISHED_FACE =
- FEARFUL_FACE =
- ANXIOUS_FACE_WITH_SWEAT =
- SAD_BUT_RELIEVED_FACE =

- CRYING_FACE =
- LOUDLY_CRYING_FACE =
- FACE_SCREAMING_IN_FEAR =
- CONFOUNDED_FACE =
- PERSEVERING_FACE =
- DISAPPOINTED_FACE =
- DOWNCAST_FACE_WITH_SWEAT =
- WEARY_FACE =
- TIRED_FACE =
- YAWNING_FACE =
- FACE_WITH_STEAM_FROM_NOSE =
- POUTING_FACE =
- ANGRY_FACE =
- FACE_WITH_SYMBOLS_ON_MOUTH =
- SMILING_FACE_WITH_HORNS =
- ANGRY_FACE_WITH_HORNS =
- SKULL =
- SKULL_AND_CROSSBONES =
- PILE_OF_POO =
- CLOWN_FACE =
- OGRE =
- GOBLIN =
- GHOST =
- ALIEN =
- ALIEN_MONSTER =
- ROBOT =
- GRINNING_CAT =
- GRINNING_CAT_WITH_SMILING_EYES =
- CAT_WITH_TEARS_OF_JOY =
- SMILING_CAT_WITH_HEART_EYES =
- CAT_WITH_WRY_SMILE =
- KISSING_CAT =
- WEARY_CAT =
- CRYING_CAT =
- POUTING_CAT =
- SEE_NO_EVIL_MONKEY =

- HEAR_NO_EVIL_MONKEY =
- SPEAK_NO_EVIL_MONKEY =
- KISS_MARK =
- LOVE_LETTER =
- HEART_WITH_ARROW =
- HEART_WITH_RIBBON =
- SPARKLING_HEART =
- GROWING_HEART =
- BEATING_HEART =
- REVOLVING_HEARTS =
- TWO_HEARTS =
- HEART_DECORATION =
- HEART_EXCLAMATION =
- BROKEN_HEART =
- RED_HEART =
- ORANGE_HEART =
- YELLOW_HEART =
- GREEN_HEART =
- BLUE_HEART =
- PURPLE_HEART =
- BROWN_HEART =
- BLACK_HEART =
- WHITE_HEART =
- HUNDRED_POINTS =
- ANGER_SYMBOL =
- COLLISION =
- DIZZY =
- SWEAT_DROPLETS =
- DASHING_AWAY =
- HOLE =
- BOMB =
- SPEECH_BALLOON =
- LEFT_SPEECH_BUBBLE =
- RIGHT_ANGER_BUBBLE =
- THOUGHT_BALLOON =
- ZZZ =

- WAVING_HAND =
- RAISED_BACK_OF_HAND =
- HAND_WITH_FINGERS_SPLAYED =
- RAISED_HAND =
- VULCAN_SALUTE =
- OK_HAND =
- PINCHED_FINGERS =
- PINCHING_HAND =
- VICTORY_HAND =
- CROSSED_FINGERS =
- LOVE_YOU_GESTURE =
- SIGN_OF_THE_HORNS =
- CALL_ME_HAND =
- BACKHAND_INDEX_POINTING_LEFT =
- BACKHAND_INDEX_POINTING_RIGHT =
- BACKHAND_INDEX_POINTING_UP =
- MIDDLE_FINGER =
- BACKHAND_INDEX_POINTING_DOWN =
- INDEX_POINTING_UP =
- THUMBS_UP =
- THUMBS_DOWN =
- RAISED_FIST =
- ONCOMING_FIST =
- LEFT_FACING_FIST =
- RIGHT_FACING_FIST =
- CLAPPING_HANDS =
- RAISING_HANDS =
- OPEN_HANDS =
- PALMS_UP_TOGETHER =
- HANDSHAKE =
- FOLDED_HANDS =
- WRITING_HAND =
- NAIL_POLISH =
- SELFIE =
- FLEXED_BICEPS =
- MECHANICAL_ARM =

- MECHANICAL_LEG =
- LEG =
- FOOT =
- EAR =
- EAR_WITH_HEARING_AID =
- NOSE =
- BRAIN =
- ANATOMICAL_HEART =
- LUNGS =
- TOOTH =
- BONE =
- EYES =
- EYE =
- TONGUE =
- MOUTH =
- BABY =
- CHILD =
- BOY =
- GIRL =
- PERSON =
- PERSON_BLOND_HAIR =
- MAN =
- MAN_BEARD =
- WOMAN =
- OLDER_PERSON =
- $OLD_MAN =$
- OLD_WOMAN =
- PERSON_FROWNING =
- PERSON_POUTING =
- PERSON_GESTURING_NO =
- PERSON_GESTURING_OK =
- PERSON_TIPPING_HAND =
- PERSON_RAISING_HAND =
- DEAF_PERSON =
- PERSON_BOWING =
- PERSON_FACEPALMING =

- PERSON_SHRUGGING =
- POLICE_OFFICER =
- DETECTIVE =
- GUARD =
- NINJA =
- CONSTRUCTION_WORKER =
- PRINCE =
- PRINCESS =
- PERSON_WEARING_TURBAN =
- PERSON_WITH_SKULLCAP =
- WOMAN_WITH_HEADSCARF =
- PERSON_IN_TUXEDO =
- PERSON_WITH_VEIL =
- PREGNANT_WOMAN =
- BREAST_FEEDING =
- BABY_ANGEL =
- SANTA_CLAUS =
- MRS_CLAUS =
- SUPERHERO =
- SUPERVILLAIN =
- MAGE =
- FAIRY =
- VAMPIRE =
- MERPERSON =
- ELF =
- GENIE =
- ZOMBIE =
- PERSON_GETTING_MASSAGE =
- PERSON_GETTING_HAIRCUT =
- PERSON_WALKING =
- PERSON_STANDING =
- PERSON_KNEELING =
- PERSON_RUNNING =
- WOMAN_DANCING =
- MAN_DANCING =
- PERSON_IN_SUIT_LEVITATING =

- PEOPLE_WITH_BUNNY_EARS =
- PERSON_IN_STEAMY_ROOM =
- PERSON_CLIMBING =
- PERSON_FENCING =
- HORSE_RACING =
- SKIER =
- SNOWBOARDER =
- PERSON_GOLFING =
- PERSON_SURFING =
- PERSON_ROWING_BOAT =
- PERSON_SWIMMING =
- PERSON_BOUNCING_BALL =
- PERSON_LIFTING_WEIGHTS =
- PERSON_BIKING =
- PERSON_MOUNTAIN_BIKING =
- PERSON_CARTWHEELING =
- PEOPLE_WRESTLING =
- PERSON_PLAYING_WATER_POLO =
- PERSON_PLAYING_HANDBALL =
- PERSON_JUGGLING =
- PERSON_IN_LOTUS_POSITION =
- PERSON_TAKING_BATH =
- PERSON_IN_BED =
- WOMEN_HOLDING_HANDS =
- WOMAN_AND_MAN_HOLDING_HANDS =
- MEN_HOLDING_HANDS =
- KISS =
- COUPLE_WITH_HEART =
- FAMILY =
- SPEAKING_HEAD =
- BUST_IN_SILHOUETTE =
- BUSTS_IN_SILHOUETTE =
- PEOPLE_HUGGING =
- FOOTPRINTS =
- LIGHT SKIN TONE =
- MEDIUM_LIGHT_SKIN_TONE =

- MEDIUM_SKIN_TONE =
- MEDIUM_DARK_SKIN_TONE =
- DARK_SKIN_TONE =
- RED_HAIR =
- CURLY_HAIR =
- WHITE_HAIR =
- BALD =
- MONKEY_FACE =
- MONKEY =
- GORILLA =
- ORANGUTAN =
- DOG_FACE =
- DOG =
- GUIDE_DOG =
- POODLE =
- WOLF =
- FOX =
- RACCOON =
- CAT_FACE =
- CAT =
- LION =
- TIGER_FACE =
- TIGER =
- LEOPARD =
- HORSE_FACE =
- HORSE =
- UNICORN =
- ZEBRA =
- DEER =
- BISON =
- COW_FACE =
- OX =
- WATER_BUFFALO =
- COW =
- PIG_FACE =
- PIG =

- BOAR =
- PIG_NOSE =
- RAM =
- EWE =
- GOAT =
- CAMEL =
- TWO_HUMP_CAMEL =
- LLAMA =
- GIRAFFE =
- ELEPHANT =
- MAMMOTH =
- RHINOCEROS =
- HIPPOPOTAMUS =
- MOUSE_FACE =
- MOUSE =
- RAT =
- HAMSTER =
- RABBIT_FACE =
- RABBIT =
- CHIPMUNK =
- BEAVER =
- HEDGEHOG =
- BAT =
- BEAR =
- KOALA =
- PANDA =
- SLOTH =
- OTTER =
- SKUNK =
- KANGAROO =
- BADGER =
- PAW_PRINTS =
- TURKEY =
- CHICKEN =
- ROOSTER =
- HATCHING_CHICK =

86

- BABY_CHICK =
- FRONT_FACING_BABY_CHICK =
- BIRD =
- PENGUIN =
- DOVE =
- EAGLE =
- DUCK =
- SWAN =
- OWL =
- DODO =
- FEATHER =
- FLAMINGO =
- PEACOCK =
- PARROT =
- FROG =
- CROCODILE =
- TURTLE =
- LIZARD =
- SNAKE =
- DRAGON_FACE =
- DRAGON =
- SAUROPOD =
- T_REX =
- SPOUTING_WHALE =
- WHALE =
- DOLPHIN =
- SEAL =
- FISH =
- TROPICAL_FISH =
- BLOWFISH =
- SHARK =
- OCTOPUS =
- SPIRAL_SHELL =
- SNAIL =
- BUTTERFLY =
- BUG =

- ANT =
- HONEYBEE =
- BEETLE =
- LADY_BEETLE =
- CRICKET =
- COCKROACH =
- SPIDER =
- SPIDER_WEB =
- SCORPION =
- MOSQUITO =
- FLY =
- WORM =
- MICROBE =
- BOUQUET =
- CHERRY_BLOSSOM =
- WHITE_FLOWER =
- ROSETTE =
- ROSE =
- WILTED_FLOWER =
- HIBISCUS =
- SUNFLOWER =
- BLOSSOM =
- TULIP =
- SEEDLING =
- POTTED_PLANT =
- EVERGREEN_TREE =
- DECIDUOUS_TREE =
- PALM_TREE =
- CACTUS =
- SHEAF_OF_RICE =
- HERB =
- SHAMROCK =
- FOUR_LEAF_CLOVER =
- MAPLE_LEAF =
- FALLEN_LEAF =
- LEAF_FLUTTERING_IN_WIND =

- GRAPES =
- MELON =
- WATERMELON =
- TANGERINE =
- LEMON =
- BANANA =
- PINEAPPLE =
- MANGO =
- RED_APPLE =
- GREEN_APPLE =
- PEAR =
- PEACH =
- CHERRIES =
- STRAWBERRY =
- BLUEBERRIES =
- KIWI FRUIT =
- TOMATO =
- OLIVE =
- COCONUT =
- AVOCADO =
- EGGPLANT =
- POTATO =
- CARROT =
- EAR_OF_CORN =
- HOT_PEPPER =
- BELL_PEPPER =
- CUCUMBER =
- LEAFY_GREEN =
- BROCCOLI =
- GARLIC =
- ONION =
- MUSHROOM =
- PEANUTS =
- CHESTNUT =
- BREAD =
- CROISSANT =

- BAGUETTE_BREAD =
- FLATBREAD =
- PRETZEL =
- BAGEL =
- PANCAKES =
- WAFFLE =
- CHEESE_WEDGE =
- MEAT_ON_BONE =
- POULTRY_LEG =
- CUT_OF_MEAT =
- BACON =
- HAMBURGER =
- FRENCH_FRIES =
- PIZZA =
- HOT_DOG =
- SANDWICH =
- TACO =
- BURRITO =
- TAMALE =
- STUFFED_FLATBREAD =
- FALAFEL =
- EGG =
- COOKING =
- SHALLOW_PAN_OF_FOOD =
- POT_OF_FOOD =
- FONDUE =
- BOWL_WITH_SPOON =
- GREEN_SALAD =
- POPCORN =
- BUTTER =
- SALT =
- CANNED_FOOD =
- BENTO_BOX =
- RICE_CRACKER =
- RICE_BALL =
- COOKED_RICE =

- CURRY_RICE =
- STEAMING_BOWL =
- SPAGHETTI =
- ROASTED_SWEET_POTATO =
- ODEN =
- SUSHI =
- FRIED_SHRIMP =
- FISH_CAKE_WITH_SWIRL =
- MOON_CAKE =
- DANGO =
- DUMPLING =
- FORTUNE_COOKIE =
- TAKEOUT_BOX =
- CRAB =
- LOBSTER =
- SHRIMP =
- SQUID =
- OYSTER =
- SOFT_ICE_CREAM =
- SHAVED_ICE =
- ICE_CREAM =
- DOUGHNUT =
- COOKIE =
- BIRTHDAY_CAKE =
- SHORTCAKE =
- CUPCAKE =
- PIE =
- CHOCOLATE_BAR =
- CANDY =
- LOLLIPOP =
- CUSTARD =
- HONEY_POT =
- BABY_BOTTLE =
- GLASS_OF_MILK =
- HOT_BEVERAGE =
- TEAPOT =

- TEACUP_WITHOUT_HANDLE =
- SAKE =
- BOTTLE_WITH_POPPING_CORK =
- WINE_GLASS =
- COCKTAIL_GLASS =
- TROPICAL_DRINK =
- BEER_MUG =
- CLINKING_BEER_MUGS =
- CLINKING_GLASSES =
- TUMBLER_GLASS =
- CUP_WITH_STRAW =
- BUBBLE_TEA =
- BEVERAGE_BOX =
- MATE =
- ICE =
- CHOPSTICKS =
- FORK_AND_KNIFE_WITH_PLATE =
- FORK_AND_KNIFE =
- SPOON =
- KITCHEN_KNIFE =
- AMPHORA =
- GLOBE_SHOWING_EUROPE_AFRICA =
- GLOBE_SHOWING_AMERICAS =
- GLOBE_SHOWING_ASIA_AUSTRALIA =
- GLOBE_WITH_MERIDIANS =
- WORLD_MAP =
- MAP_OF_JAPAN =
- COMPASS =
- SNOW_CAPPED_MOUNTAIN =
- MOUNTAIN =
- VOLCANO =
- MOUNT_FUJI =
- CAMPING =
- BEACH_WITH_UMBRELLA =
- DESERT =
- DESERT_ISLAND =

- NATIONAL_PARK =
- STADIUM =
- CLASSICAL_BUILDING =
- BUILDING_CONSTRUCTION =
- BRICK =
- ROCK =
- WOOD =
- HUT =
- HOUSES =
- DERELICT_HOUSE =
- HOUSE =
- HOUSE_WITH_GARDEN =
- OFFICE_BUILDING =
- JAPANESE_POST_OFFICE =
- POST_OFFICE =
- HOSPITAL =
- BANK =
- HOTEL =
- LOVE_HOTEL =
- CONVENIENCE_STORE =
- SCHOOL =
- DEPARTMENT_STORE =
- FACTORY =
- JAPANESE_CASTLE =
- CASTLE =
- WEDDING =
- TOKYO_TOWER =
- STATUE_OF_LIBERTY =
- CHURCH =
- MOSQUE =
- HINDU_TEMPLE =
- SYNAGOGUE =
- SHINTO_SHRINE =
- KAABA =
- FOUNTAIN =
- TENT =

- FOGGY =
- NIGHT_WITH_STARS =
- CITYSCAPE =
- SUNRISE_OVER_MOUNTAINS =
- SUNRISE =
- CITYSCAPE_AT_DUSK =
- SUNSET =
- BRIDGE_AT_NIGHT =
- HOT_SPRINGS =
- CAROUSEL_HORSE =
- FERRIS_WHEEL =
- ROLLER_COASTER =
- BARBER_POLE =
- CIRCUS_TENT =
- LOCOMOTIVE =
- RAILWAY_CAR =
- HIGH_SPEED_TRAIN =
- BULLET_TRAIN =
- TRAIN =
- METRO =
- LIGHT_RAIL =
- STATION =
- TRAM =
- MONORAIL =
- MOUNTAIN_RAILWAY =
- $TRAM_CAR =$
- BUS =
- ONCOMING_BUS =
- TROLLEYBUS =
- MINIBUS =
- AMBULANCE =
- FIRE_ENGINE =
- POLICE_CAR =
- ONCOMING_POLICE_CAR =
- TAXI =
- ONCOMING_TAXI =

- AUTOMOBILE =
- ONCOMING_AUTOMOBILE =
- SPORT_UTILITY_VEHICLE =
- PICKUP_TRUCK =
- DELIVERY_TRUCK =
- ARTICULATED_LORRY =
- TRACTOR =
- RACING_CAR =
- MOTORCYCLE =
- MOTOR_SCOOTER =
- MANUAL_WHEELCHAIR =
- MOTORIZED_WHEELCHAIR =
- AUTO_RICKSHAW =
- BICYCLE =
- KICK_SCOOTER =
- SKATEBOARD =
- ROLLER_SKATE =
- BUS_STOP =
- MOTORWAY =
- RAILWAY_TRACK =
- OIL_DRUM =
- FUEL_PUMP =
- POLICE_CAR_LIGHT =
- HORIZONTAL_TRAFFIC_LIGHT =
- VERTICAL_TRAFFIC_LIGHT =
- STOP_SIGN =
- CONSTRUCTION =
- ANCHOR =
- SAILBOAT =
- CANOE =
- SPEEDBOAT =
- PASSENGER_SHIP =
- FERRY =
- MOTOR_BOAT =
- SHIP =
- AIRPLANE =

- SMALL_AIRPLANE =
- AIRPLANE_DEPARTURE =
- AIRPLANE_ARRIVAL =
- PARACHUTE =
- SEAT =
- HELICOPTER =
- SUSPENSION_RAILWAY =
- MOUNTAIN_CABLEWAY =
- AERIAL_TRAMWAY =
- SATELLITE =
- ROCKET =
- FLYING_SAUCER =
- BELLHOP_BELL =
- LUGGAGE =
- HOURGLASS_DONE =
- HOURGLASS_NOT_DONE =
- WATCH =
- ALARM_CLOCK =
- STOPWATCH =
- TIMER_CLOCK =
- MANTELPIECE_CLOCK =
- TWELVE_OCLOCK =
- TWELVE_THIRTY =
- ONE_OCLOCK =
- ONE_THIRTY =
- TWO_OCLOCK =
- TWO_THIRTY =
- THREE_OCLOCK =
- THREE_THIRTY =
- FOUR_OCLOCK =
- FOUR_THIRTY =
- FIVE_OCLOCK =
- FIVE_THIRTY =
- SIX_OCLOCK =
- SIX_THIRTY =
- SEVEN_OCLOCK =

- SEVEN_THIRTY =
- EIGHT_OCLOCK =
- EIGHT_THIRTY =
- NINE_OCLOCK =
- NINE_THIRTY =
- TEN_OCLOCK =
- TEN_THIRTY =
- ELEVEN_OCLOCK =
- ELEVEN_THIRTY =
- NEW_MOON =
- WAXING_CRESCENT_MOON =
- FIRST_QUARTER_MOON =
- WAXING_GIBBOUS_MOON =
- FULL_MOON =
- WANING_GIBBOUS_MOON =
- LAST_QUARTER_MOON =
- WANING_CRESCENT_MOON =
- CRESCENT_MOON =
- NEW_MOON_FACE =
- FIRST_QUARTER_MOON_FACE =
- LAST_QUARTER_MOON_FACE =
- THERMOMETER =
- SUN =
- FULL_MOON_FACE =
- SUN_WITH_FACE =
- RINGED_PLANET =
- STAR =
- GLOWING_STAR =
- SHOOTING_STAR =
- MILKY_WAY =
- CLOUD =
- SUN_BEHIND_CLOUD =
- CLOUD_WITH_LIGHTNING_AND_RAIN =
- SUN_BEHIND_SMALL_CLOUD =
- SUN_BEHIND_LARGE_CLOUD =
- SUN_BEHIND_RAIN_CLOUD =

- CLOUD_WITH_RAIN =
- CLOUD_WITH_SNOW =
- CLOUD_WITH_LIGHTNING =
- TORNADO =
- FOG =
- WIND_FACE =
- CYCLONE =
- RAINBOW =
- CLOSED_UMBRELLA =
- UMBRELLA =
- UMBRELLA_WITH_RAIN_DROPS =
- UMBRELLA_ON_GROUND =
- HIGH_VOLTAGE =
- SNOWFLAKE =
- SNOWMAN =
- SNOWMAN_WITHOUT_SNOW =
- COMET =
- FIRE =
- DROPLET =
- WATER_WAVE =
- JACK_O_LANTERN =
- CHRISTMAS_TREE =
- FIREWORKS =
- SPARKLER =
- FIRECRACKER =
- SPARKLES =
- BALLOON =
- PARTY_POPPER =
- CONFETTI_BALL =
- TANABATA_TREE =
- PINE_DECORATION =
- JAPANESE_DOLLS =
- CARP_STREAMER =
- WIND_CHIME =
- MOON_VIEWING_CEREMONY =
- RED_ENVELOPE =

- RIBBON =
- WRAPPED_GIFT =
- REMINDER_RIBBON =
- ADMISSION_TICKETS =
- TICKET =
- MILITARY_MEDAL =
- TROPHY =
- SPORTS_MEDAL =
- FIRST_PLACE_MEDAL =
- SECOND_PLACE_MEDAL =
- THIRD_PLACE_MEDAL =
- SOCCER_BALL =
- BASEBALL =
- SOFTBALL =
- BASKETBALL =
- VOLLEYBALL =
- AMERICAN_FOOTBALL =
- RUGBY_FOOTBALL =
- TENNIS =
- FLYING_DISC =
- BOWLING =
- CRICKET_GAME =
- FIELD_HOCKEY =
- ICE_HOCKEY =
- LACROSSE =
- PING_PONG =
- BADMINTON =
- BOXING_GLOVE =
- MARTIAL_ARTS_UNIFORM =
- GOAL_NET =
- FLAG_IN_HOLE =
- ICE_SKATE =
- FISHING_POLE =
- DIVING_MASK =
- RUNNING_SHIRT =
- SKIS =

- SLED =
- CURLING_STONE =
- DIRECT_HIT =
- YO_YO =
- KITE =
- BALL =
- CRYSTAL_BALL =
- MAGIC_WAND =
- NAZAR_AMULET =
- VIDEO_GAME =
- JOYSTICK =
- SLOT_MACHINE =
- GAME_DIE =
- PUZZLE_PIECE =
- TEDDY_BEAR =
- PIñATA =
- NESTING_DOLLS =
- SPADE_SUIT =
- HEART_SUIT =
- DIAMOND_SUIT =
- CLUB_SUIT =
- CHESS_PAWN =
- JOKER =
- MAHJONG_RED_DRAGON =
- FLOWER_PLAYING_CARDS =
- PERFORMING_ARTS =
- FRAMED_PICTURE =
- ARTIST_PALETTE =
- THREAD =
- SEWING_NEEDLE =
- YARN =
- KNOT =
- GLASSES =
- SUNGLASSES =
- GOGGLES =
- LAB_COAT =

- SAFETY_VEST =
- NECKTIE =
- T_SHIRT =
- JEANS =
- SCARF =
- GLOVES =
- COAT =
- SOCKS =
- DRESS =
- KIMONO =
- SARI =
- ONE_PIECE_SWIMSUIT =
- BRIEFS =
- SHORTS =
- BIKINI =
- WOMANS_CLOTHES =
- PURSE =
- HANDBAG =
- CLUTCH_BAG =
- SHOPPING_BAGS =
- BACKPACK =
- THONG_SANDAL =
- MANS_SHOE =
- RUNNING_SHOE =
- HIKING_BOOT =
- FLAT_SHOE =
- HIGH_HEELED_SHOE =
- WOMANS_SANDAL =
- BALLET_SHOES =
- WOMANS_BOOT =
- CROWN =
- WOMANS_HAT =
- TOP_HAT =
- GRADUATION_CAP =
- BILLED_CAP =
- MILITARY_HELMET =

- RESCUE_WORKERS_HELMET =
- PRAYER_BEADS =
- LIPSTICK =
- RING =
- GEM_STONE =
- MUTED_SPEAKER =
- SPEAKER_LOW_VOLUME =
- SPEAKER_MEDIUM_VOLUME =
- SPEAKER_HIGH_VOLUME =
- LOUDSPEAKER =
- MEGAPHONE =
- POSTAL_HORN =
- BELL =
- BELL_WITH_SLASH =
- MUSICAL_SCORE =
- MUSICAL_NOTE =
- MUSICAL_NOTES =
- STUDIO_MICROPHONE =
- LEVEL_SLIDER =
- CONTROL_KNOBS =
- MICROPHONE =
- HEADPHONE =
- RADIO =
- SAXOPHONE =
- ACCORDION =
- GUITAR =
- MUSICAL_KEYBOARD =
- TRUMPET =
- VIOLIN =
- BANJO =
- DRUM =
- LONG_DRUM =
- MOBILE_PHONE =
- MOBILE_PHONE_WITH_ARROW =
- TELEPHONE =
- TELEPHONE_RECEIVER =

- PAGER =
- FAX_MACHINE =
- BATTERY =
- ELECTRIC_PLUG =
- LAPTOP =
- DESKTOP_COMPUTER =
- PRINTER =
- KEYBOARD =
- COMPUTER_MOUSE =
- TRACKBALL =
- COMPUTER_DISK =
- FLOPPY_DISK =
- OPTICAL_DISK =
- DVD =
- ABACUS =
- MOVIE_CAMERA =
- FILM_FRAMES =
- FILM_PROJECTOR =
- CLAPPER_BOARD =
- TELEVISION =
- CAMERA =
- CAMERA_WITH_FLASH =
- VIDEO_CAMERA =
- VIDEOCASSETTE =
- MAGNIFYING_GLASS_TILTED_LEFT =
- MAGNIFYING_GLASS_TILTED_RIGHT =
- CANDLE =
- LIGHT_BULB =
- FLASHLIGHT =
- RED_PAPER_LANTERN =
- DIYA_LAMP =
- NOTEBOOK_WITH_DECORATIVE_COVER =
- CLOSED_BOOK =
- OPEN_BOOK =
- GREEN_BOOK =
- BLUE_BOOK =

- ORANGE_BOOK =
- BOOKS =
- NOTEBOOK =
- LEDGER =
- PAGE_WITH_CURL =
- SCROLL =
- PAGE_FACING_UP =
- NEWSPAPER =
- ROLLED_UP_NEWSPAPER =
- BOOKMARK_TABS =
- BOOKMARK =
- LABEL =
- MONEY_BAG =
- COIN =
- YEN_BANKNOTE =
- DOLLAR_BANKNOTE =
- EURO_BANKNOTE =
- POUND_BANKNOTE =
- MONEY_WITH_WINGS =
- CREDIT_CARD =
- RECEIPT =
- CHART_INCREASING_WITH_YEN =
- ENVELOPE =
- E_MAIL =
- INCOMING_ENVELOPE =
- ENVELOPE_WITH_ARROW =
- OUTBOX_TRAY =
- INBOX_TRAY =
- PACKAGE =
- CLOSED_MAILBOX_WITH_RAISED_FLAG =
- CLOSED_MAILBOX_WITH_LOWERED_FLAG =
- OPEN_MAILBOX_WITH_RAISED_FLAG =
- OPEN_MAILBOX_WITH_LOWERED_FLAG =
- POSTBOX =
- BALLOT_BOX_WITH_BALLOT =
- PENCIL =

- BLACK_NIB =
- FOUNTAIN_PEN =
- PEN =
- PAINTBRUSH =
- CRAYON =
- MEMO =
- BRIEFCASE =
- FILE_FOLDER =
- OPEN_FILE_FOLDER =
- CARD_INDEX_DIVIDERS =
- CALENDAR =
- TEAR_OFF_CALENDAR =
- SPIRAL_NOTEPAD =
- SPIRAL_CALENDAR =
- CARD_INDEX =
- CHART_INCREASING =
- CHART_DECREASING =
- BAR_CHART =
- CLIPBOARD =
- PUSHPIN =
- ROUND_PUSHPIN =
- PAPERCLIP =
- LINKED_PAPERCLIPS =
- STRAIGHT_RULER =
- TRIANGULAR_RULER =
- SCISSORS =
- CARD_FILE_BOX =
- FILE_CABINET =
- WASTEBASKET =
- LOCKED =
- UNLOCKED =
- LOCKED_WITH_PEN =
- LOCKED_WITH_KEY =
- KEY =
- OLD_KEY =
- HAMMER =

- AXE =
- PICK =
- HAMMER_AND_PICK =
- HAMMER_AND_WRENCH =
- DAGGER =
- CROSSED_SWORDS =
- PISTOL =
- BOOMERANG =
- BOW_AND_ARROW =
- SHIELD =
- CARPENTRY_SAW =
- WRENCH =
- SCREWDRIVER =
- NUT_AND_BOLT =
- GEAR =
- CLAMP =
- BALANCE_SCALE =
- WHITE_CANE =
- LINK =
- CHAINS =
- HOOK =
- TOOLBOX =
- MAGNET =
- LADDER =
- ALEMBIC =
- TEST_TUBE =
- PETRI_DISH =
- DNA =
- MICROSCOPE =
- TELESCOPE =
- SATELLITE_ANTENNA =
- SYRINGE =
- DROP_OF_BLOOD =
- PILL =
- ADHESIVE_BANDAGE =
- STETHOSCOPE =

106 Chapter 3. Tutorials

- DOOR =
- ELEVATOR =
- MIRROR =
- WINDOW =
- BED =
- COUCH_AND_LAMP =
- CHAIR =
- TOILET =
- PLUNGER =
- SHOWER =
- BATHTUB =
- MOUSE_TRAP =
- RAZOR =
- LOTION_BOTTLE =
- SAFETY_PIN =
- BROOM =
- BASKET =
- ROLL_OF_PAPER =
- BUCKET =
- SOAP =
- TOOTHBRUSH =
- SPONGE =
- FIRE_EXTINGUISHER =
- SHOPPING_CART =
- CIGARETTE =
- COFFIN =
- HEADSTONE =
- FUNERAL_URN =
- MOAI =
- PLACARD =
- ATM_SIGN =
- LITTER_IN_BIN_SIGN =
- POTABLE_WATER =
- WHEELCHAIR_SYMBOL =
- MENS_ROOM =
- WOMENS_ROOM =

- RESTROOM =
- BABY_SYMBOL =
- WATER_CLOSET =
- PASSPORT_CONTROL =
- CUSTOMS =
- BAGGAGE_CLAIM =
- LEFT_LUGGAGE =
- WARNING =
- CHILDREN_CROSSING =
- NO_ENTRY =
- PROHIBITED =
- NO_BICYCLES =
- NO_SMOKING =
- NO_LITTERING =
- NON_POTABLE_WATER =
- NO_PEDESTRIANS =
- NO_MOBILE_PHONES =
- NO_ONE_UNDER_EIGHTEEN =
- RADIOACTIVE =
- BIOHAZARD =
- UP_ARROW =
- UP_RIGHT_ARROW =
- RIGHT_ARROW =
- DOWN_RIGHT_ARROW =
- DOWN_ARROW =
- DOWN_LEFT_ARROW =
- LEFT_ARROW =
- UP_LEFT_ARROW =
- UP_DOWN_ARROW =
- LEFT_RIGHT_ARROW =
- RIGHT_ARROW_CURVING_LEFT =
- LEFT_ARROW_CURVING_RIGHT =
- RIGHT_ARROW_CURVING_UP =
- RIGHT_ARROW_CURVING_DOWN =
- CLOCKWISE_VERTICAL_ARROWS =
- COUNTERCLOCKWISE_ARROWS_BUTTON =

108 Chapter 3. Tutorials

- BACK_ARROW =
- END_ARROW =
- ON_ARROW =
- SOON_ARROW =
- TOP_ARROW =
- PLACE_OF_WORSHIP =
- ATOM_SYMBOL =
- OM =
- STAR_OF_DAVID =
- WHEEL_OF_DHARMA =
- YIN_YANG =
- LATIN_CROSS =
- ORTHODOX_CROSS =
- STAR_AND_CRESCENT =
- PEACE_SYMBOL =
- MENORAH =
- DOTTED_SIX_POINTED_STAR =
- ARIES =
- TAURUS =
- GEMINI =
- CANCER =
- LEO =
- VIRGO =
- LIBRA =
- SCORPIO =
- SAGITTARIUS =
- CAPRICORN =
- AQUARIUS =
- PISCES =
- OPHIUCHUS =
- SHUFFLE_TRACKS_BUTTON =
- REPEAT_BUTTON =
- REPEAT_SINGLE_BUTTON =
- PLAY_BUTTON =
- FAST_FORWARD_BUTTON =
- NEXT_TRACK_BUTTON =

- PLAY_OR_PAUSE_BUTTON =
- REVERSE_BUTTON =
- FAST_REVERSE_BUTTON =
- LAST_TRACK_BUTTON =
- UPWARDS_BUTTON =
- FAST_UP_BUTTON =
- DOWNWARDS_BUTTON =
- FAST_DOWN_BUTTON =
- PAUSE_BUTTON =
- STOP_BUTTON =
- RECORD_BUTTON =
- EJECT_BUTTON =
- CINEMA =
- DIM_BUTTON =
- BRIGHT_BUTTON =
- ANTENNA_BARS =
- VIBRATION_MODE =
- MOBILE_PHONE_OFF =
- FEMALE_SIGN =
- MALE_SIGN =
- TRANSGENDER_SYMBOL =
- MULTIPLY =
- PLUS =
- MINUS =
- DIVIDE =
- INFINITY =
- DOUBLE_EXCLAMATION_MARK =
- EXCLAMATION_QUESTION_MARK =
- QUESTION_MARK =
- WHITE_QUESTION_MARK =
- WHITE_EXCLAMATION_MARK =
- EXCLAMATION_MARK =
- WAVY_DASH =
- CURRENCY_EXCHANGE =
- HEAVY_DOLLAR_SIGN =
- MEDICAL_SYMBOL =

110 Chapter 3. Tutorials

- RECYCLING_SYMBOL =
- FLEUR_DE_LIS =
- TRIDENT_EMBLEM =
- NAME_BADGE =
- JAPANESE_SYMBOL_FOR_BEGINNER =
- HOLLOW_RED_CIRCLE =
- CHECK_MARK_BUTTON =
- CHECK_BOX_WITH_CHECK =
- CHECK_MARK = ✓
- CROSS_MARK =
- CROSS_MARK_BUTTON =
- CURLY_LOOP =
- DOUBLE_CURLY_LOOP =
- PART_ALTERNATION_MARK =
- EIGHT_SPOKED_ASTERISK =
- EIGHT_POINTED_STAR =
- SPARKLE =
- COPYRIGHT = ©
- REGISTERED = ®
- TRADE_MARK = TM
- INPUT_LATIN_UPPERCASE =
- INPUT_LATIN_LOWERCASE =
- INPUT_NUMBERS =
- INPUT_SYMBOLS =
- INPUT_LATIN_LETTERS =
- A_BUTTON_BLOOD_TYPE =
- AB_BUTTON_BLOOD_TYPE =
- B_BUTTON_BLOOD_TYPE =
- CL_BUTTON =
- COOL_BUTTON =
- FREE_BUTTON =
- INFORMATION =
- ID_BUTTON =
- CIRCLED_M =
- NEW_BUTTON =
- NG_BUTTON =

- O_BUTTON_BLOOD_TYPE =
- OK_BUTTON =
- P_BUTTON =
- SOS_BUTTON =
- UP_BUTTON =
- VS_BUTTON =
- JAPANESE_HERE_BUTTON =
- JAPANESE_SERVICE_CHARGE_BUTTON =
- JAPANESE_MONTHLY_AMOUNT_BUTTON =
- JAPANESE_NOT_FREE_OF_CHARGE_BUTTON =
- JAPANESE_RESERVED_BUTTON =
- JAPANESE_BARGAIN_BUTTON =
- JAPANESE_DISCOUNT_BUTTON =
- JAPANESE_FREE_OF_CHARGE_BUTTON =
- JAPANESE_PROHIBITED_BUTTON =
- JAPANESE_ACCEPTABLE_BUTTON =
- JAPANESE_APPLICATION_BUTTON =
- JAPANESE_PASSING_GRADE_BUTTON =
- JAPANESE_VACANCY_BUTTON =
- JAPANESE_CONGRATULATIONS_BUTTON =
- JAPANESE_SECRET_BUTTON =
- JAPANESE_OPEN_FOR_BUSINESS_BUTTON =
- JAPANESE_NO_VACANCY_BUTTON =
- RED_CIRCLE =
- ORANGE_CIRCLE =
- YELLOW_CIRCLE =
- GREEN_CIRCLE =
- BLUE_CIRCLE =
- PURPLE_CIRCLE =
- BROWN_CIRCLE =
- BLACK_CIRCLE =
- WHITE_CIRCLE =
- RED_SQUARE =
- ORANGE_SQUARE =
- YELLOW_SQUARE =
- GREEN_SQUARE =

112 Chapter 3. Tutorials

- BLUE_SQUARE =
- PURPLE_SQUARE =
- BROWN_SQUARE =
- BLACK_LARGE_SQUARE =
- WHITE LARGE SQUARE =
- BLACK_MEDIUM_SQUARE =
- WHITE_MEDIUM_SQUARE =
- BLACK_MEDIUM_SMALL_SQUARE =
- WHITE_MEDIUM_SMALL_SQUARE =
- BLACK_SMALL_SQUARE =
- WHITE_SMALL_SQUARE =
- LARGE_ORANGE_DIAMOND =
- LARGE_BLUE_DIAMOND =
- SMALL_ORANGE_DIAMOND =
- SMALL_BLUE_DIAMOND =
- RED_TRIANGLE_POINTED_UP =
- RED_TRIANGLE_POINTED_DOWN =
- DIAMOND_WITH_A_DOT =
- RADIO_BUTTON =
- WHITE_SQUARE_BUTTON =
- BLACK_SQUARE_BUTTON =
- CHEQUERED_FLAG =
- TRIANGULAR_FLAG =
- CROSSED_FLAGS =
- BLACK_FLAG =
- WHITE_FLAG =

___init___()

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__ Initialize self.

Attributes

ABACUS

AB_BUTTON_BLOOD_TYPE

Continued on next page

Table 23 – continued from previous page

rable 23 – continued from p	nevious page
ACCORDION	
ADHESIVE_BANDAGE	
ADMISSION_TICKETS	
AERIAL_TRAMWAY	
AIRPLANE	
AIRPLANE_ARRIVAL	
AIRPLANE_DEPARTURE	
ALARM_CLOCK	
ALEMBIC	
ALIEN	
ALIEN_MONSTER	
AMBULANCE	
AMERICAN_FOOTBALL	
AMPHORA	
ANATOMICAL_HEART	
ANCHOR	
ANGER_SYMBOL	
ANGRY_FACE	
ANGRY_FACE_WITH_HORNS	
ANGUISHED FACE	
ANT	
ANTENNA BARS	
ANXIOUS_FACE_WITH_SWEAT	
AQUARIUS	
ARIES	
ARTICULATED_LORRY	
ARTIST_PALETTE	
ASTONISHED_FACE	
ATM_SIGN	
ATOM_SYMBOL	
AUTOMOBILE	
AUTO_RICKSHAW	
AVOCADO	
AXE	
A_BUTTON_BLOOD_TYPE	
BABY	
BABY ANGEL	
BABY BOTTLE	
BABY_CHICK	
BABY_SYMBOL	
BACKHAND_INDEX_POINTING_DOWN	
BACKHAND_INDEX_POINTING_LEFT	
BACKHAND_INDEX_POINTING_RIGHT	
BACKHAND_INDEX_POINTING_UP	
BACKPACK	
BACK_ARROW	
BACON	
BADGER	
BADMINTON	
BAGEL	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from previous	page
BAGGAGE_CLAIM	
BAGUETTE_BREAD	
BALANCE_SCALE	
BALD	
BALL	
BALLET_SHOES	
BALLOON	
BALLOT_BOX_WITH_BALLOT	
BANANA	
BANJO	
BANK	
BARBER_POLE	
BAR_CHART	
BASEBALL	
BASKET	
BASKETBALL	
BAT	
BATHTUB	
BATTERY	
BEACH_WITH_UMBRELLA	
BEAMING_FACE_WITH_SMILING_EYES	
BEAR	
BEATING_HEART	
BEAVER	
BED	
BEER_MUG	
BEETLE	
BELL	
BELLHOP_BELL	
BELL PEPPER	
BELL_WITH_SLASH	
BENTO_BOX	
BEVERAGE_BOX	
BICYCLE	
BIKINI	
BILLED_CAP	
BIOHAZARD	
BIRD	
BIRTHDAY_CAKE	
BISON	
BLACK_CIRCLE	
BLACK_FLAG	
BLACK_HEART	
BLACK_LARGE_SQUARE	
BLACK_MEDIUM_SMALL_SQUARE	
BLACK_MEDIUM_SQUARE	
BLACK_NIB	
BLACK_SMALL_SQUARE	
BLACK_SQUARE_BUTTON	
BLOSSOM	
	Continued on next page

Table 23 – continued from previous page

Table 23 – continued fro	m previous page
BLOWFISH	
BLUEBERRIES	
BLUE_BOOK	
BLUE_CIRCLE	
BLUE_HEART	
BLUE_SQUARE	
BOAR	
BOMB	
BONE	
BOOKMARK	
BOOKMARK_TABS	
BOOKS	
BOOMERANG	
BOTTLE_WITH_POPPING_CORK	
BOUQUET	
BOWLING	
BOWL_WITH_SPOON	
BOW_AND_ARROW	
BOXING_GLOVE	
BOY	
BRAIN	
BREAD	
BREAST_FEEDING	
BRICK	
BRIDGE_AT_NIGHT	
BRIEFCASE	
BRIEFS	
BRIGHT_BUTTON	
BROCCOLI	
BROKEN_HEART	
BROOM	
BROWN_CIRCLE	
BROWN_HEART	
BROWN_SQUARE	
BUBBLE TEA	
BUCKET	
BUG	
BUILDING_CONSTRUCTION	
BULLET_TRAIN	
BURRITO	
BUS	
BUSTS_IN_SILHOUETTE	
BUST_IN_SILHOUETTE	
BUS_STOP	
BUTTER	
BUTTERFLY	
B_BUTTON_BLOOD_TYPE	
CACTUS	
CALENDAR	
CALL_ME_HAND	
V.1111_111111	Continued on next page

Table 23 – continued from previous page

rable 23 – continued from prev	nous page
CAMEL	
CAMERA	
CAMERA_WITH_FLASH	
CAMPING	
CANCER	
CANDLE	
CANDY	
CANNED_FOOD	
CANOE	
CAPRICORN	
CARD_FILE_BOX	
CARD_INDEX	
CARD_INDEX_DIVIDERS	
CAROUSEL_HORSE	
CARPENTRY_SAW	
CARP_STREAMER	
CARROT	
CASTLE	
CAT	
CAT_FACE	
CAT_WITH_TEARS_OF_JOY	
CAT_WITH_WRY_SMILE	
CHAINS	
CHAIR	
CHART_DECREASING	
CHART_INCREASING	
CHART_INCREASING_WITH_YEN	
CHECK_BOX_WITH_CHECK	
CHECK_MARK	
CHECK_MARK_BUTTON	
CHEESE_WEDGE	
CHEQUERED_FLAG	
CHERRIES	
CHERRY_BLOSSOM	
CHESS_PAWN	
CHESTNUT CHICKEN	
CHILD	
CHILDREN_CROSSING	
CHIPMUNK	
CHOCOLATE_BAR	
CHOPSTICKS	
CHRISTMAS_TREE	
CHURCH	
CIGARETTE	
CINEMA	
CIRCLED_M	
CIRCUS_TENT	
CITYSCAPE	
CITYSCAPE_AT_DUSK	
	Continued on next page

Table 23 – continued from previous page

rable 23 – continued from pr	evious page
CLAMP	
CLAPPER_BOARD	
CLAPPING_HANDS	
CLASSICAL_BUILDING	
CLINKING_BEER_MUGS	
CLINKING_GLASSES	
CLIPBOARD	
CLOCKWISE_VERTICAL_ARROWS	
CLOSED_BOOK	
CLOSED_MAILBOX_WITH_LOWERED_FLAG	
CLOSED_MAILBOX_WITH_RAISED_FLAG	
CLOSED_UMBRELLA	
CLOUD	
CLOUD_WITH_LIGHTNING	
CLOUD_WITH_LIGHTNING_AND_RAIN	
CLOUD_WITH_RAIN	
CLOUD_WITH_SNOW	
CLOWN_FACE	
CLUB_SUIT	
CLUTCH_BAG	
CL BUTTON	
COAT	
COCKROACH	
COCKTAIL GLASS	
COCONUT	
COFFIN	
COIN	
COLD FACE	
COLLISION	
COMET	
COMPASS	
COMPUTER DISK	
COMPUTER_MOUSE	
CONFETTI_BALL	
CONFOUNDED_FACE	
CONFUSED_FACE	
CONSTRUCTION	
CONSTRUCTION_WORKER	
CONTROL KNOBS	
CONVENIENCE_STORE	
COOKED RICE	
COOKIE	
COOKING	
COOL_BUTTON	
COPYRIGHT	
COUCH AND LAMP	
COUNTERCLOCKWISE_ARROWS_BUTTON	
COUPLE_WITH_HEART	
COW COUPLE_WITH_HEART	
COWBOY HAT FACE	
COMPOT_HAT_FACE	Continued on next page

Table 23 – continued from previous page

iable 23 – continued from	r previous page
COW_FACE	
CRAB	
CRAYON	
CREDIT_CARD	
CRESCENT_MOON	
CRICKET	
CRICKET_GAME	
CROCODILE	
CROISSANT	
CROSSED_FINGERS	
CROSSED_FLAGS	
CROSSED_SWORDS	
CROSS_MARK	
CROSS_MARK_BUTTON	
CROWN	
CRYING_CAT	
CRYING_FACE	
CRYSTAL_BALL	
CUCUMBER	
CUPCAKE	
CUP_WITH_STRAW	
CURLING_STONE	
CURLY_HAIR	
CURLY_LOOP	
CURRENCY_EXCHANGE	
CURRY_RICE	
CUSTARD	
CUSTOMS	
CUT_OF_MEAT	
CYCLONE	
DAGGER	
DANGO	
DARK_SKIN_TONE	
DASHING_AWAY	
DEAF_PERSON	
DECIDUOUS_TREE	
DEER	
DELIVERY_TRUCK	
DEPARTMENT_STORE	
DERELICT_HOUSE	
DESERT	
DESERT_ISLAND	
DESKTOP_COMPUTER	
DETECTIVE	
DIAMOND_SUIT	
DIAMOND_WITH_A_DOT	
DIM_BUTTON	
DIRECT_HIT	
DISAPPOINTED_FACE	
DISGUISED_FACE	
	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from previous p	page
DIVIDE	
DIVING_MASK	
DIYA_LAMP	
DIZZY	
DIZZY_FACE	
DNA	
DODO	
DOG	
DOG_FACE	
DOLLAR_BANKNOTE	
DOLPHIN	
DOOR	
DOTTED_SIX_POINTED_STAR	
DOUBLE_CURLY_LOOP	
DOUBLE_EXCLAMATION_MARK	
DOUGHNUT	
DOVE	
DOWNCAST_FACE_WITH_SWEAT	
DOWNWARDS_BUTTON	
DOWN_ARROW	
DOWN_LEFT_ARROW	
DOWN_RIGHT_ARROW	
DRAGON	
DRAGON_FACE	
DRESS	
DROOLING_FACE	
DROPLET	
DROP_OF_BLOOD	
DRUM	
DUCK	
DUMPLING	
DVD	
EAGLE	
EAR	
EAR_OF_CORN	
EAR_WITH_HEARING_AID	
EGG	
EGGPLANT	
EIGHT_OCLOCK	
EIGHT_POINTED_STAR	
EIGHT_SPOKED_ASTERISK	
EIGHT_THIRTY	
EJECT_BUTTON	
ELECTRIC_PLUG	
ELEPHANT	
ELEVATOR	
ELEVEN_OCLOCK	
ELEVEN_THIRTY	
ELF	
END_ARROW	
	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from	previous page
ENVELOPE	
ENVELOPE_WITH_ARROW	
EURO_BANKNOTE	
EVERGREEN_TREE	
EWE	
EXCLAMATION_MARK	
EXCLAMATION_QUESTION_MARK	
EXPLODING_HEAD	
EXPRESSIONLESS_FACE	
EYE	
EYES	
E_MAIL	
FACE_BLOWING_A_KISS	
FACE_SAVORING_FOOD	
FACE_SCREAMING_IN_FEAR	
FACE_VOMITING	
FACE_WITHOUT_MOUTH	
FACE_WITH_HAND_OVER_MOUTH	
FACE_WITH_HEAD_BANDAGE	
FACE_WITH_MEDICAL_MASK	
FACE_WITH_MONOCLE	
FACE_WITH_OPEN_MOUTH	
FACE_WITH_RAISED_EYEBROW	
FACE_WITH_ROLLING_EYES	
FACE_WITH_STEAM_FROM_NOSE	
FACE_WITH_SYMBOLS_ON_MOUTH	
FACE_WITH_TEARS_OF_JOY	
FACE_WITH_THERMOMETER	
FACE_WITH_TONGUE	
FACTORY	
FAIRY	
FALAFEL	
FALLEN_LEAF	
FAMILY	
FAST_DOWN_BUTTON	
FAST_FORWARD_BUTTON	
FAST_REVERSE_BUTTON	
FAST UP BUTTON	
FAX MACHINE	
FEARFUL_FACE	
FEATHER	
FEMALE_SIGN	
FERRIS_WHEEL	
FERRY	
FIELD_HOCKEY	
FILE_CABINET	
FILE FOLDER	
FILM_FRAMES	
FILM PROJECTOR	
FIRE	
	Continued on next page

Table 23 – continued from previous page

Table 25 – Continued from pr	evious page
FIRECRACKER	
FIREWORKS	
FIRE_ENGINE	
FIRE_EXTINGUISHER	
FIRST_PLACE_MEDAL	
FIRST_QUARTER_MOON	
FIRST_QUARTER_MOON_FACE	
FISH	
FISHING_POLE	
FISH_CAKE_WITH_SWIRL	
FIVE OCLOCK	
FIVE_THIRTY	
FLAG_IN_HOLE	
FLAMINGO	
FLASHLIGHT	
FLATBREAD	
FLAT_SHOE	
FLEUR_DE_LIS	
FLEXED_BICEPS	
FLOPPY_DISK	
FLOWER_PLAYING_CARDS	
FLUSHED_FACE FLY	
FLYING_DISC	
FLYING_SAUCER	
FOG	
FOGGY	
FOLDED_HANDS	
FONDUE	
FOOT	
FOOTPRINTS	
FORK_AND_KNIFE	
FORK_AND_KNIFE_WITH_PLATE	
FORTUNE_COOKIE	
FOUNTAIN	
FOUNTAIN_PEN	
FOUR_LEAF_CLOVER	
FOUR_OCLOCK	
FOUR_THIRTY	
FOX	
FRAMED_PICTURE	
FREE_BUTTON	
FRENCH_FRIES	
FRIED_SHRIMP	
FROG	
FRONT_FACING_BABY_CHICK	
FROWNING_FACE	
FROWNING_FACE_WITH_OPEN_MOUTH	
FUEL_PUMP	
FULL_MOON	
	Continued on next page

Table 23 – continued from previous page

rable 23 – continued from prev	lous page
FULL_MOON_FACE	
FUNERAL_URN	·
GAME_DIE	
GARLIC	
GEAR	
GEMINI	
GEM_STONE	
GENIE	
GHOST	
GIRAFFE	
GIRL	
GLASSES	
GLASS_OF_MILK	
GLOBE_SHOWING_AMERICAS	
GLOBE_SHOWING_ASIA_AUSTRALIA	
GLOBE_SHOWING_EUROPE_AFRICA	
GLOBE_WITH_MERIDIANS	
GLOVES	
GLOWING_STAR	
GOAL NET	
GOAT	
GOBLIN	
GOGGLES	
GORILLA	
GRADUATION_CAP	
GRAPES	
GREEN_APPLE	
GREEN_BOOK	
GREEN_CIRCLE	
GREEN_HEART	
GREEN_SALAD	
GREEN_SQUARE	
GRIMACING_FACE	
GRINNING_CAT	
GRINNING_CAT_WITH_SMILING_EYES	
GRINNING_FACE	
GRINNING_FACE_WITH_BIG_EYES	
GRINNING_FACE_WITH_SMILING_EYES	
GRINNING_FACE_WITH_SWEAT	
GRINNING_SQUINTING_FACE	
GROWING_HEART	
GUARD	
GUIDE_DOG	
GUITAR	
HAMBURGER	
HAMMER	
HAMMER_AND_PICK	
HAMMER_AND_WRENCH	
HAMSTER	
HANDBAG	

Table 23 – continued from previous page

Table 23 – continued fro	m previous page
HANDSHAKE	
HAND_WITH_FINGERS_SPLAYED	
HATCHING_CHICK	
HEADPHONE	
HEADSTONE	
HEART_DECORATION	
HEART_EXCLAMATION	
HEART_SUIT	
HEART_WITH_ARROW	
HEART_WITH_RIBBON	
HEAR_NO_EVIL_MONKEY	
HEAVY_DOLLAR_SIGN	
HEDGEHOG	
HELICOPTER	
HERB	
HIBISCUS	
HIGH_HEELED_SHOE	
HIGH_SPEED_TRAIN	
HIGH_VOLTAGE	
HIKING_BOOT	
HINDU_TEMPLE	
HIPPOPOTAMUS	
HOLE	
HOLLOW_RED_CIRCLE	
HONEYBEE	
HONEY POT	
HOOK	
HORIZONTAL_TRAFFIC_LIGHT	
HORSE	
HORSE FACE	
HORSE RACING	
HOSPITAL	
HOTEL	
HOT_BEVERAGE	
HOT_DOG	
HOT_FACE	
HOT_PEPPER	
HOT_SPRINGS	
HOURGLASS DONE	
HOURGLASS_NOT_DONE	
HOUSE	
HOUSES	
HOUSE_WITH_GARDEN	
HUGGING_FACE	
HUNDRED_POINTS	
HUSHED_FACE	
HUT	
ICE	
ICE CREAM	
ICE_HOCKEY	
100_11001111	Continued on next page

Table 23 – continued from previous page

rable 23 – continued from previou	us page
ICE_SKATE	
ID_BUTTON	
INBOX_TRAY	
INCOMING_ENVELOPE	
INDEX_POINTING_UP	
INFINITY	
INFORMATION	
INPUT_LATIN_LETTERS	
INPUT_LATIN_LOWERCASE	
INPUT_LATIN_UPPERCASE	
INPUT_NUMBERS	
INPUT_SYMBOLS	
JACK_O_LANTERN	
JAPANESE_ACCEPTABLE_BUTTON	
JAPANESE_APPLICATION_BUTTON	
JAPANESE_BARGAIN_BUTTON	
JAPANESE_CASTLE	
JAPANESE_CONGRATULATIONS_BUTTON	
JAPANESE_DISCOUNT_BUTTON	
JAPANESE_DOLLS	
JAPANESE_FREE_OF_CHARGE_BUTTON	
JAPANESE_HERE_BUTTON	
JAPANESE MONTHLY AMOUNT BUTTON	
JAPANESE_NOT_FREE_OF_CHARGE_BUTTON	
JAPANESE_NO_VACANCY_BUTTON	
JAPANESE_OPEN_FOR_BUSINESS_BUTTON	
JAPANESE_PASSING_GRADE_BUTTON	
JAPANESE_POST_OFFICE	
JAPANESE_PROHIBITED_BUTTON	
JAPANESE_RESERVED_BUTTON	
JAPANESE_SECRET_BUTTON	
JAPANESE_SERVICE_CHARGE_BUTTON	
JAPANESE_SYMBOL_FOR_BEGINNER	
JAPANESE_VACANCY_BUTTON	
JEANS	
JOKER	
JOYSTICK	
KAABA	
KANGAROO	
KEY	
KEYBOARD	
KICK_SCOOTER	
KIMONO	
KISS	
KISSING_CAT	
KISSING_FACE	
KISSING FACE WITH CLOSED EYES	
KISSING FACE WITH SMILING EYES	
KISS_MARK	
KITCHEN KNIFE	
_	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from previ	ous page
KITE	
KIWI_FRUIT	
KNOT	
KOALA	
LABEL	
LAB_COAT	
LACROSSE	
LADDER	
LADY_BEETLE	
LAPTOP	
LARGE_BLUE_DIAMOND	
LARGE_ORANGE_DIAMOND	
LAST_QUARTER_MOON	
LAST_QUARTER_MOON_FACE	
LAST_TRACK_BUTTON	
LATIN_CROSS	
 LEAFY_GREEN	
 LEAF_FLUTTERING_IN_WIND	
 LEDGER	
LEFT_ARROW_CURVING_RIGHT	
LEFT_FACING_FIST	
LEFT_LUGGAGE	
LEFT_RIGHT_ARROW	
LEFT_SPEECH_BUBBLE	
LEG	
LEMON	
LEO	
LEOPARD	
LEVEL_SLIDER	
LIBRA	
LIGHT_BULB	
 LIGHT_RAIL	
 LIGHT_SKIN_TONE	
LINK	
LINKED_PAPERCLIPS	
LION	
LIPSTICK	
LITTER_IN_BIN_SIGN	
LIZARD	
LLAMA	
LOBSTER	
LOCKED	
LOCKED_WITH_KEY	
LOCKED_WITH_PEN	
LOCOMOTIVE	
LOLLIPOP	
LONG_DRUM	
LOTION_BOTTLE	
LOUDLY_CRYING_FACE	
	Continued on next page

Table 23 – continued from previous page

rable 23 – continued from prev	ious page
LOUDSPEAKER	
LOVE_HOTEL	
LOVE_LETTER	
LOVE_YOU_GESTURE	
LUGGAGE	
LUNGS	
LYING_FACE	
MAGE	
MAGIC_WAND	
MAGNET	
MAGNIFYING_GLASS_TILTED_LEFT	
MAGNIFYING_GLASS_TILTED_RIGHT	
MAHJONG_RED_DRAGON	
MALE_SIGN	
MAMMOTH	
MAN	
MANGO	
MANS_SHOE	
MANTELPIECE_CLOCK	
MANUAL WHEELCHAIR	
MAN BEARD	
MAN DANCING	
MAPLE_LEAF	
MAP_OF_JAPAN	
MARTIAL_ARTS_UNIFORM	
MATE	
MEAT_ON_BONE	
MECHANICAL_ARM	
MECHANICAL LEG	
MEDICAL_SYMBOL	
MEDIUM_DARK_SKIN_TONE	
MEDIUM_LIGHT_SKIN_TONE	
MEDIUM_SKIN_TONE	
MEGAPHONE	
MELON	
MEMO	
MENORAH	
MENS_ROOM	
MEN_HOLDING_HANDS	
MERPERSON	
METRO	
MICROBE	
MICROPHONE	
MICROSCOPE	
MIDDLE_FINGER	
MILITARY_HELMET	
MILITARY MEDAL	
MILKY_WAY	
MINIBUS	
MINUS	
	Continued on next page

Table 23 – continued from previous page

Table 23 – continued fro	m previous page
MIRROR	
MOAI	
MOBILE_PHONE	
MOBILE_PHONE_OFF	
MOBILE_PHONE_WITH_ARROW	
MONEY_BAG	
MONEY_MOUTH_FACE	
MONEY_WITH_WINGS	
MONKEY	
MONKEY_FACE	
MONORAIL	
MOON_CAKE	
MOON_VIEWING_CEREMONY	
MOSQUE	
MOSQUITO	
MOTORCYCLE	
MOTORIZED_WHEELCHAIR	
MOTORWAY	
MOTOR_BOAT	
MOTOR_SCOOTER	
MOUNTAIN	
MOUNTAIN_CABLEWAY	
MOUNTAIN_RAILWAY	
MOUNT_FUJI	
MOUSE	
MOUSE FACE	
MOUSE_TRAP	
MOUTH	
MOVIE CAMERA	
MRS CLAUS	
MULTIPLY	
MUSHROOM	
MUSICAL KEYBOARD	
MUSICAL NOTE	
MUSICAL_NOTES	
MUSICAL_SCORE	
MUTED_SPEAKER	
NAIL_POLISH	
NAME_BADGE	
NATIONAL_PARK	
NAUSEATED_FACE	
NAZAR AMULET	
NECKTIE	
NECTILE NERD_FACE	
NESTING DOLLS	
NEUTRAL FACE	
_	
NEW PUTTON	
NEW_BUTTON	
NEW_MOON_FINCE	
NEW_MOON_FACE	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from pro	evious page
NEXT_TRACK_BUTTON	
NG_BUTTON	
NIGHT_WITH_STARS	
NINE_OCLOCK	
NINE_THIRTY	
NINJA	
NON_POTABLE_WATER	
NOSE	
NOTEBOOK	
NOTEBOOK_WITH_DECORATIVE_COVER	
NO_BICYCLES	
NO_ENTRY	
NO_LITTERING	
NO_MOBILE_PHONES	
NO_ONE_UNDER_EIGHTEEN	
NO_PEDESTRIANS	
NO_SMOKING	
NUT_AND_BOLT	
OCTOPUS	
ODEN	
OFFICE_BUILDING	
OGRE	
OIL_DRUM	
OK_BUTTON	
OK_HAND	
OLDER_PERSON	
OLD_KEY	
OLD_MAN	
OLD_WOMAN	
OLIVE	
OM	
ONCOMING_AUTOMOBILE	
ONCOMING_BUS	
ONCOMING_FIST	
ONCOMING_POLICE_CAR	
ONCOMING_TAXI	
ONE_OCLOCK	
ONE_PIECE_SWIMSUIT	
ONE_THIRTY	
ONION	
ON_ARROW	
OPEN_BOOK	
OPEN_FILE_FOLDER	
OPEN_HANDS	
OPEN_MAILBOX_WITH_LOWERED_FLAG	
OPEN_MAILBOX_WITH_RAISED_FLAG	
OPHIUCHUS	
OPTICAL_DISK	
ORANGE_BOOK	
ORANGE_CIRCLE	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from	previous page
ORANGE_HEART	
ORANGE_SQUARE	
ORANGUTAN	
ORTHODOX_CROSS	
OTTER	
OUTBOX_TRAY	
OWL	
OX	
OYSTER	
O_BUTTON_BLOOD_TYPE	
PACKAGE	
PAGER	
PAGE_FACING_UP	
PAGE_WITH_CURL	
PAINTBRUSH	
PALMS_UP_TOGETHER	
PALM_TREE	
PANCAKES	
PANDA	
PAPERCLIP	
PARACHUTE	
PARROT	
PARTYING_FACE	
PARTY_POPPER	
PART_ALTERNATION_MARK	
PASSENGER_SHIP	
PASSPORT_CONTROL	
PAUSE_BUTTON	
PAW_PRINTS	
PEACE_SYMBOL	
PEACH	
PEACOCK	
PEANUTS	
PEAR	
PEN	
PENCIL	
PENGUIN	
PENSIVE_FACE	
PEOPLE_HUGGING	
PEOPLE_WITH_BUNNY_EARS	
PEOPLE_WRESTLING	
PERFORMING_ARTS	
PERSEVERING_FACE	
PERSON	
PERSON_BIKING	
PERSON_BLOND_HAIR	
PERSON_BOUNCING_BALL	
PERSON_BOWING	
PERSON_CARTWHEELING	
PERSON_CLIMBING	
	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from previous page	9
PERSON_FACEPALMING	
PERSON_FENCING	
PERSON_FROWNING	
PERSON_GESTURING_NO	
PERSON_GESTURING_OK	
PERSON_GETTING_HAIRCUT	
PERSON_GETTING_MASSAGE	
PERSON_GOLFING	
PERSON_IN_BED	
PERSON_IN_LOTUS_POSITION	
PERSON_IN_STEAMY_ROOM	
PERSON_IN_SUIT_LEVITATING	
PERSON_IN_TUXEDO	
PERSON_JUGGLING	
PERSON_KNEELING	
PERSON_LIFTING_WEIGHTS	
PERSON_MOUNTAIN_BIKING	
PERSON_PLAYING_HANDBALL	
PERSON_PLAYING_WATER_POLO	
PERSON_POUTING	
PERSON_RAISING_HAND	
PERSON_ROWING_BOAT	
PERSON_RUNNING	
PERSON_SHRUGGING	
PERSON_STANDING	
PERSON_SURFING	
PERSON_SWIMMING	
PERSON_TAKING_BATH	
PERSON_TIPPING_HAND	
PERSON_WALKING	
PERSON_WEARING_TURBAN	
PERSON_WITH_SKULLCAP	
PERSON_WITH_VEIL	
PETRI_DISH	
PICK	
PICKUP_TRUCK	
PIE	
PIG	
PIG_FACE	
PIG_NOSE	
PILE_OF_POO	
PILL	
PINCHED_FINGERS	
PINCHING_HAND	
PINEAPPLE	
PINE_DECORATION	
PING_PONG	
PISCES	
PISTOL	
PIZZA	
	Continued on next page

Table 23 – continued from previous page

Table 23 - continued	Tom previous page
PIÑATA	
PLACARD	
PLACE_OF_WORSHIP	
PLAY_BUTTON	
PLAY_OR_PAUSE_BUTTON	
PLEADING_FACE	
PLUNGER	
PLUS	
POLICE_CAR	
POLICE_CAR_LIGHT	
POLICE_OFFICER	
POODLE	
POPCORN	
POSTAL_HORN	
POSTBOX	
POST_OFFICE	
POTABLE WATER	
POTATO	
POTTED_PLANT	
POT_OF_FOOD	
POULTRY_LEG	
POUND BANKNOTE	
POUTING_CAT	
POUTING_FACE	
PRAYER_BEADS	
PREGNANT_WOMAN	
PRETZEL	
PRINCE	
PRINCESS	
PRINTER	
PROHIBITED	
PURPLE_CIRCLE	
PURPLE_HEART	
PURPLE_SQUARE	
PURSE	
PUSHPIN	
PUZZLE_PIECE	
P_BUTTON	
QUESTION_MARK	
RABBIT	
RABBIT_FACE	
RACCOON	
RACING_CAR	
RADIO	
RADIOACTIVE	
RADIO_BUTTON	
RAILWAY_CAR	
RAILWAY_TRACK	
RAINBOW	
RAISED_BACK_OF_HAND	
	Continued on next page

Table 23 – continued from previous page

rable 23 – continued from pr	evious page
RAISED_FIST	
RAISED_HAND	
RAISING_HANDS	
RAM	
RAT	
RAZOR	
RECEIPT	
RECORD_BUTTON	
RECYCLING_SYMBOL	
RED_APPLE	
RED_CIRCLE	
RED_ENVELOPE	
RED_HAIR	
RED_HEART	
RED_PAPER_LANTERN	
RED_SQUARE	
RED_TRIANGLE_POINTED_DOWN	
RED_TRIANGLE_POINTED_UP	
REGISTERED	
RELIEVED FACE	
REMINDER RIBBON	
REPEAT BUTTON	
REPEAT_SINGLE_BUTTON	
RESCUE WORKERS HELMET	
RESTROOM	
REVERSE_BUTTON	
REVOLVING_HEARTS	
RHINOCEROS	
RIBBON	
RICE BALL	
RICE_CRACKER	
RIGHT_ANGER_BUBBLE	
RIGHT_ARROW	
RIGHT_ARROW_CURVING_DOWN	
RIGHT_ARROW_CURVING_LEFT	
RIGHT_ARROW_CURVING_UP	
RIGHT_FACING_FIST	
RING	
RINGED_PLANET	
ROASTED_SWEET_POTATO	
ROBOT	
ROCK	
ROCKET	
ROLLED_UP_NEWSPAPER	
ROLLER_COASTER	
ROLLER_SKATE	
ROLLING_ON_THE_FLOOR_LAUGHING	
ROLL_OF_PAPER	
ROOSTER	
ROSE	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from prev	lious page
ROSETTE	
ROUND_PUSHPIN	
RUGBY_FOOTBALL	
RUNNING_SHIRT	
RUNNING_SHOE	
SAD_BUT_RELIEVED_FACE	
SAFETY_PIN	
SAFETY_VEST	
SAGITTARIUS	
SAILBOAT	
SAKE	
SALT	
SANDWICH	
SANTA_CLAUS	
SARI	
SATELLITE	
SATELLITE_ANTENNA	
SAUROPOD	
SAXOPHONE	
SCARF	
SCHOOL	
SCISSORS	
SCORPIO	
SCORPION	
SCREWDRIVER	
SCROLL	
SEAL	
SEAT	
SECOND PLACE MEDAL	
SEEDLING	
SEE_NO_EVIL_MONKEY	
SELFIE	
SEVEN_OCLOCK	
SEVEN_THIRTY	
SEWING_NEEDLE	
SHALLOW_PAN_OF_FOOD	
SHAMROCK	
SHARK	
SHAVED_ICE	
SHEAF_OF_RICE	
SHIELD	
SHINTO_SHRINE	
SHIP	
SHOOTING_STAR	
SHOPPING_BAGS	
SHOPPING CART	
SHORTCAKE	
SHORTS	
SHOWER	
SHRIMP	
<u></u>	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from previou	is page
SHUFFLE_TRACKS_BUTTON	
SHUSHING_FACE	
SIGN_OF_THE_HORNS	
SIX_OCLOCK	
SIX_THIRTY	
SKATEBOARD	
SKIER	
SKIS	
SKULL	
SKULL_AND_CROSSBONES	
SKUNK	
SLED	
SLEEP ING_FACE	
SLEEPY_FACE	
SLIGHTLY_FROWNING_FACE	
SLIGHTLY_SMILING_FACE	
SLOTH	
SLOT_MACHINE	
SMALL_AIRPLANE	
SMALL_BLUE_DIAMOND	
SMALL_ORANGE_DIAMOND	
SMILING_CAT_WITH_HEART_EYES	
SMILING_FACE	
SMILING_FACE_WITH_HALO	
SMILING_FACE_WITH_HEARTS	
SMILING_FACE_WITH_HEART_EYES	
SMILING_FACE_WITH_HORNS	
SMILING_FACE_WITH_SMILING_EYES	
SMILING_FACE_WITH_SUNGLASSES	
SMILING_FACE_WITH_TEAR	
SMIRKING_FACE	
SNAIL	
SNAKE	
SNEEZING_FACE	
SNOWBOARDER	
SNOWFLAKE	
SNOWMAN	
SNOWMAN_WITHOUT_SNOW	
SNOW_CAPPED_MOUNTAIN	
SOAP	
SOCCER_BALL	
SOCKS	
SOFTBALL	
SOFT_ICE_CREAM	
SOON_ARROW	
SOS_BUTTON	
SPADE_SUIT	
SPAGHETTI	
SPARKLE	
SPARKLER	
	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from previous page

Table 25 – Continued not	n providuo pago
SUSHI	
SUSPENSION_RAILWAY	
SWAN	
SWEAT_DROPLETS	
SYNAGOGUE	
SYRINGE	
TACO	
TAKEOUT_BOX	
TAMALE	
TANABATA_TREE	
TANGERINE	
TAURUS	
TAXI	
TEACUP_WITHOUT_HANDLE	
TEAPOT	
TEAR_OFF_CALENDAR	
TEDDY_BEAR	
TELEPHONE	
TELEPHONE_RECEIVER	
TELESCOPE	
TELEVISION	
TENNIS	
TENT	
TEN_OCLOCK	
TEN_THIRTY	
TEST_TUBE	
THERMOMETER	
THINKING_FACE	
THIRD_PLACE_MEDAL	
THONG_SANDAL	
THOUGHT_BALLOON	
THREAD	
THREE_OCLOCK	
THREE_THIRTY	
THUMBS_DOWN	
THUMBS_UP	
TICKET	
TIGER	
TIGER_FACE	
TIMER_CLOCK	
TIRED_FACE	
TOILET	
TOKYO_TOWER	
TOMATO	
TONGUE	
TOOLBOX	
TOOTH	
TOOTHBRUSH	
TOP_ARROW	
TOP_HAT	
101_1111	Continued on next page

Table 23 – continued from previous page

rable 23 – continued from previo	ouo pugo
TORNADO	
TRACKBALL	
TRACTOR	
TRADE_MARK	
TRAIN	
TRAM	
TRAM_CAR	
TRANSGENDER_SYMBOL	
TRIANGULAR_FLAG	
TRIANGULAR_RULER	
TRIDENT_EMBLEM	
TROLLEYBUS	
TROPHY	
TROPICAL_DRINK	
TROPICAL_FISH	
TRUMPET	
TULIP	
TUMBLER_GLASS	
TURKEY	
TURTLE	
TWELVE_OCLOCK	
TWELVE_THIRTY	
TWO HEARTS	
TWO_HUMP_CAMEL	
TWO_OCLOCK	
TWO_THIRTY	
T_REX	
T_SHIRT	
UMBRELLA	
UMBRELLA_ON_GROUND	
UMBRELLA_WITH_RAIN_DROPS	
UNAMUSED_FACE	
UNICORN	
UNLOCKED	
UPSIDE DOWN FACE	
UPWARDS_BUTTON	
UP_ARROW	
UP_BUTTON	
UP DOWN ARROW	
UP_LEFT_ARROW	
UP RIGHT ARROW	
VAMPIRE	
VERTICAL_TRAFFIC_LIGHT	
VIBRATION_MODE	
VICTORY_HAND	
VIDEOCASSETTE	
VIDEO_CAMERA	
VIDEO_GAME	
VIOLIN	
VIRGO	
V 11.00	Continued on next page

Table 23 – continued from previous page

Table 23 – continued from	previous page
VOLCANO	
VOLLEYBALL	
VS_BUTTON	
VULCAN_SALUTE	
WAFFLE	
WANING_CRESCENT_MOON	
WANING_GIBBOUS_MOON	
WARNING	
WASTEBASKET	
WATCH	
WATERMELON	
WATER_BUFFALO	
WATER_CLOSET	
WATER_WAVE	
WAVING_HAND	
WAVY_DASH	
WAXING_CRESCENT_MOON	
WAXING_GIBBOUS_MOON	
WEARY CAT	
WEARY FACE	
WEDDING	
WHALE	
WHEELCHAIR_SYMBOL	
WHEEL_OF_DHARMA	
WHITE CANE	
WHITE_CIRCLE	
WHITE EXCLAMATION MARK	
WHITE FLAG	
WHITE FLOWER	
WHITE HAIR	
WHITE_HEART	
WHITE_LARGE_SQUARE	
WHITE_MEDIUM_SMALL_SQUARE	
WHITE_MEDIUM_SQUARE	
WHITE_QUESTION_MARK	
WHITE_SMALL_SQUARE	
WHITE SQUARE BUTTON	
WILTED_FLOWER	
WINDOW	
WIND_CHIME	
WIND FACE	
WINE_GLASS	
WINKING FACE	
WINKING_FACE_WITH_TONGUE	
WOLF	
WOMAN	
WOMANS BOOT	
WOMANS_CLOTHES	
WOMANS_HAT	
WOMANS SANDAL	
	Continued on next page

Table 23 – continued from previous page

Table 25 Continued from provided page
WOMAN_AND_MAN_HOLDING_HANDS
WOMAN_DANCING
WOMAN_WITH_HEADSCARF
WOMENS_ROOM
WOMEN_HOLDING_HANDS
WOOD
WOOZY_FACE
WORLD_MAP
WORM
WORRIED_FACE
WRAPPED_GIFT
WRENCH
WRITING_HAND
YARN
YAWNING_FACE
YELLOW_CIRCLE
YELLOW_HEART
YELLOW_SQUARE
YEN_BANKNOTE
YIN_YANG
YO_YO
ZANY_FACE
ZEBRA
ZIPPER_MOUTH_FACE
ZOMBIE
ZZZ

```
ABACUS = ''
AB_BUTTON_BLOOD_TYPE = ''
ACCORDION = '\U0001fa97'
ADHESIVE_BANDAGE = '\U0001fa79'
ADMISSION_TICKETS = ''
AERIAL_TRAMWAY = ''
AIRPLANE = ''
AIRPLANE_ARRIVAL = ''
AIRPLANE_DEPARTURE = ''
ALARM_CLOCK = ''
ALEMBIC = ''
ALIEN = ''
ALIEN_MONSTER = ''
AMBULANCE = ''
AMERICAN_FOOTBALL = ''
AMPHORA = ''
ANATOMICAL_HEART = '\U0001fac0'
```

140 Chapter 3. Tutorials

```
ANCHOR = ''
ANGER_SYMBOL = ''
ANGRY_FACE = ''
ANGRY_FACE_WITH_HORNS = ''
ANGUISHED_FACE = ''
ANT = ''
ANTENNA_BARS = ''
ANXIOUS_FACE_WITH_SWEAT = ''
AQUARIUS = ''
ARIES = ''
ARTICULATED_LORRY = ''
ARTIST_PALETTE = ''
ASTONISHED_FACE = ''
ATM_SIGN = ''
ATOM SYMBOL = ''
AUTOMOBILE = ''
AUTO_RICKSHAW = '\U0001f6fa'
AVOCADO = ''
AXE = '\U0001fa93'
A_BUTTON_BLOOD_TYPE = ''
BABY = ''
BABY_ANGEL = ''
BABY_BOTTLE = ''
BABY CHICK = ''
BABY SYMBOL = ''
BACKHAND_INDEX_POINTING_DOWN = ''
BACKHAND_INDEX_POINTING_LEFT = ''
BACKHAND_INDEX_POINTING_RIGHT = ''
BACKHAND_INDEX_POINTING_UP = ''
BACKPACK = ''
BACK_ARROW = ''
BACON = ''
BADGER = ''
BADMINTON = ''
BAGEL = ''
BAGGAGE_CLAIM = ''
```

```
BAGUETTE BREAD = ''
BALANCE_SCALE = ''
BALD = ''
BALL = ''
BALLET_SHOES = '\U0001fa70'
BALLOON = ''
BALLOT_BOX_WITH_BALLOT = ''
BANANA = ''
BANJO = '\U0001fa95'
BANK = ''
BARBER_POLE = ''
BAR_CHART = ''
BASEBALL = ''
BASKET = ''
BASKETBALL = ''
BAT = ''
BATHTUB = ''
BATTERY = ''
BEACH_WITH_UMBRELLA = ''
BEAMING_FACE_WITH_SMILING_EYES = ''
BEAR = ''
BEATING_HEART = ''
BEAVER = '\U0001f9ab'
BED = ''
BEER MUG = ''
BEETLE = '\U0001fab2'
BELL = ''
BELLHOP_BELL = ''
BELL_PEPPER = '\U0001fad1'
BELL_WITH_SLASH = ''
BENTO_BOX = ''
BEVERAGE_BOX = '\U0001f9c3'
BICYCLE = ''
BIKINI = ''
BILLED CAP = ''
BIOHAZARD = ''
```

```
BIRD = ''
BIRTHDAY_CAKE = ''
BISON = '\U0001f9ac'
BLACK_CIRCLE = ''
BLACK FLAG = ''
BLACK HEART = ''
BLACK_LARGE_SQUARE = ''
BLACK_MEDIUM_SMALL_SQUARE = ''
BLACK_MEDIUM_SQUARE = ''
BLACK_NIB = ''
BLACK_SMALL_SQUARE = ''
BLACK_SQUARE_BUTTON = ''
BLOSSOM = ''
BLOWFISH = ''
BLUEBERRIES = '\U0001fad0'
BLUE BOOK = ''
BLUE_CIRCLE = ''
BLUE_HEART = ''
BLUE_SQUARE = '\U0001f7e6'
BOAR = ''
BOMB = ''
BONE = ''
BOOKMARK = ''
BOOKMARK TABS = ''
BOOKS = ''
BOOMERANG = '\U0001fa83'
BOTTLE_WITH_POPPING_CORK = ''
BOUQUET = ''
BOWLING = ''
BOWL_WITH_SPOON = ''
BOW_AND_ARROW = ''
BOXING_GLOVE = ''
BOY = ''
BRAIN = ''
BREAD = ''
BREAST_FEEDING = ''
```

```
BRICK = ''
BRIDGE_AT_NIGHT = ''
BRIEFCASE = ''
BRIEFS = '\U0001fa72'
BRIGHT BUTTON = ''
BROCCOLI = ''
BROKEN_HEART = ''
BROOM = ''
BROWN_CIRCLE = '\U0001f7e4'
BROWN_HEART = '\U0001f90e'
BROWN_SQUARE = '\U0001f7eb'
BUBBLE_TEA = '\U0001f9cb'
BUCKET = '\U0001faa3'
BUG = ''
BUILDING_CONSTRUCTION = ''
BULLET TRAIN = ''
BURRITO = ''
BUS = ''
BUSTS_IN_SILHOUETTE = ''
BUST_IN_SILHOUETTE = ''
BUS_STOP = ''
BUTTER = '\U0001f9c8'
BUTTERFLY = ''
B_BUTTON_BLOOD_TYPE = ''
CACTUS = ''
CALENDAR = ''
CALL_ME_HAND = ''
CAMEL = ''
CAMERA = ''
CAMERA_WITH_FLASH = ''
CAMPING = ''
CANCER = ''
CANDLE = ''
CANDY = ''
CANNED_FOOD = ''
CANOE = ''
```

```
CAPRICORN = ''
CARD_FILE_BOX = ''
CARD_INDEX = ''
CARD_INDEX_DIVIDERS = ''
CAROUSEL HORSE = ''
CARPENTRY SAW = '\U0001fa9a'
CARP_STREAMER = ''
CARROT = ''
CASTLE = ''
CAT = ''
CAT_FACE = ''
CAT_WITH_TEARS_OF_JOY = ''
CAT_WITH_WRY_SMILE = ''
CHAINS = ''
CHAIR = '\U0001fa91'
CHART DECREASING = ''
CHART_INCREASING = ''
CHART_INCREASING_WITH_YEN = ''
CHECK_BOX_WITH_CHECK = ''
CHECK_MARK = '√'
CHECK_MARK_BUTTON = ''
CHEESE_WEDGE = ''
CHEQUERED_FLAG = ''
CHERRIES = ''
CHERRY BLOSSOM = ''
CHESS_PAWN = ''
CHESTNUT = ''
CHICKEN = ''
CHILD = ''
CHILDREN_CROSSING = ''
CHIPMUNK = ''
CHOCOLATE_BAR = ''
CHOPSTICKS = ''
CHRISTMAS_TREE = ''
CHURCH = ''
CIGARETTE = ''
```

```
CINEMA = ''
CIRCLED_M = ''
CIRCUS_TENT = ''
CITYSCAPE = ''
CITYSCAPE_AT_DUSK = ''
CLAMP = ''
CLAPPER_BOARD = ''
CLAPPING_HANDS = ''
CLASSICAL_BUILDING = ''
CLINKING_BEER_MUGS = ''
CLINKING_GLASSES = ''
CLIPBOARD = ''
CLOCKWISE_VERTICAL_ARROWS = ''
CLOSED_BOOK = ''
CLOSED_MAILBOX_WITH_LOWERED_FLAG = ''
CLOSED MAILBOX WITH RAISED FLAG = ''
CLOSED_UMBRELLA = ''
CLOUD = ''
CLOUD_WITH_LIGHTNING = ''
CLOUD_WITH_LIGHTNING_AND_RAIN = ''
CLOUD_WITH_RAIN = ''
CLOUD_WITH_SNOW = ''
CLOWN_FACE = ''
CLUB SUIT = ''
CLUTCH BAG = ''
CL BUTTON = ''
COAT = ''
COCKROACH = '\U0001fab3'
COCKTAIL GLASS = ''
COCONUT = ''
COFFIN = ''
COIN = '\U0001fa99'
COLD_FACE = ''
COLLISION = ''
COMET = ''
COMPASS = ''
```

```
COMPUTER DISK = ''
COMPUTER_MOUSE = ''
CONFETTI_BALL = ''
CONFOUNDED_FACE = ''
CONFUSED FACE = ''
CONSTRUCTION = ''
CONSTRUCTION_WORKER = ''
CONTROL_KNOBS = ''
CONVENIENCE_STORE = ''
COOKED_RICE = ''
COOKIE = ''
COOKING = ''
COOL_BUTTON = ''
COPYRIGHT = '@'
COUCH_AND_LAMP = ''
COUNTERCLOCKWISE ARROWS BUTTON = ''
COUPLE_WITH_HEART = ''
COW = ''
COWBOY_HAT_FACE = ''
COW FACE = ''
CRAB = ''
CRAYON = ''
CREDIT_CARD = ''
CRESCENT MOON = ''
CRICKET = ''
CRICKET_GAME = ''
CROCODILE = ''
CROISSANT = ''
CROSSED FINGERS = ''
CROSSED_FLAGS = ''
CROSSED_SWORDS = ''
CROSS_MARK = ''
CROSS_MARK_BUTTON = ''
CROWN = ''
CRYING_CAT = ''
CRYING_FACE = ''
```

```
CRYSTAL_BALL = ''
CUCUMBER = ''
CUPCAKE = ''
CUP_WITH_STRAW = ''
CURLING STONE = ''
CURLY HAIR = ''
CURLY_LOOP = ''
CURRENCY_EXCHANGE = ''
CURRY_RICE = ''
CUSTARD = ''
CUSTOMS = ''
CUT_OF_MEAT = ''
CYCLONE = ''
DAGGER = ''
DANGO = ''
DARK SKIN TONE = ''
DASHING_AWAY = ''
DEAF_PERSON = '\U0001f9cf'
DECIDUOUS_TREE = ''
DEER = ''
DELIVERY_TRUCK = ''
DEPARTMENT_STORE = ''
DERELICT_HOUSE = ''
DESERT = ''
DESERT_ISLAND = ''
DESKTOP_COMPUTER = ''
DETECTIVE = ''
DIAMOND_SUIT = ''
DIAMOND_WITH_A_DOT = ''
DIM_BUTTON = ''
DIRECT_HIT = ''
DISAPPOINTED_FACE = ''
DISGUISED_FACE = '\U0001f978'
DIVIDE = ''
DIVING MASK = '\U0001f93f'
DIYA_LAMP = '\U0001fa94'
```

```
DIZZY = ''
DIZZY_FACE = ''
DNA = ''
DODO = '\U0001f9a4'
DOG = ''
DOG FACE = ''
DOLLAR_BANKNOTE = ''
DOLPHIN = ''
DOOR = ''
DOTTED_SIX_POINTED_STAR = ''
DOUBLE_CURLY_LOOP = ''
DOUBLE_EXCLAMATION_MARK = ''
DOUGHNUT = ''
DOVE = ''
DOWNCAST_FACE_WITH_SWEAT = ''
DOWNWARDS BUTTON = ''
DOWN_ARROW = ''
DOWN_LEFT_ARROW = ''
DOWN_RIGHT_ARROW = ''
DRAGON = ''
DRAGON_FACE = ''
DRESS = ''
DROOLING_FACE = ''
DROPLET = ''
DROP_OF_BLOOD = '\U0001fa78'
DRUM = ''
DUCK = ''
DUMPLING = ''
DVD = ''
EAGLE = ''
EAR = ''
EAR_OF_CORN = ''
EAR_WITH_HEARING_AID = '\U0001f9bb'
EGG = ''
EGGPLANT = ''
EIGHT_OCLOCK = ''
```

```
EIGHT POINTED STAR = ''
EIGHT_SPOKED_ASTERISK = ''
EIGHT_THIRTY = ''
EJECT_BUTTON = ''
ELECTRIC PLUG = ''
ELEPHANT = ''
ELEVATOR = '\U0001f6d7'
ELEVEN_OCLOCK = ''
ELEVEN THIRTY = ''
ELF = ''
END_ARROW = ''
ENVELOPE = ''
ENVELOPE_WITH_ARROW = ''
EURO_BANKNOTE = ''
EVERGREEN TREE = ''
EWE = ''
EXCLAMATION_MARK = ''
EXCLAMATION_QUESTION_MARK = ''
EXPLODING_HEAD = ''
EXPRESSIONLESS FACE = ''
EYE = ''
EYES = ''
E_MAIL = ''
FACE_BLOWING_A_KISS = ''
FACE_SAVORING_FOOD = ''
FACE_SCREAMING_IN_FEAR = ''
FACE_VOMITING = ''
FACE_WITHOUT_MOUTH = ''
FACE_WITH_HAND_OVER_MOUTH = ''
FACE_WITH_HEAD_BANDAGE = ''
FACE_WITH_MEDICAL_MASK = ''
FACE_WITH_MONOCLE = ''
FACE_WITH_OPEN_MOUTH = ''
FACE_WITH_RAISED_EYEBROW = ''
FACE_WITH_ROLLING_EYES = ''
FACE_WITH_STEAM_FROM_NOSE = ''
```

```
FACE WITH SYMBOLS ON MOUTH = ''
FACE_WITH_TEARS_OF_JOY = ''
FACE_WITH_THERMOMETER = ''
FACE_WITH_TONGUE = ''
FACTORY = ''
FAIRY = ''
FALAFEL = '\U0001f9c6'
FALLEN_LEAF = ''
FAMILY = ''
FAST_DOWN_BUTTON = ''
FAST_FORWARD_BUTTON = ''
FAST_REVERSE_BUTTON = ''
FAST_UP_BUTTON = ''
FAX MACHINE = ''
FEARFUL FACE = ''
FEATHER = '\U0001fab6'
FEMALE SIGN = ''
FERRIS_WHEEL = ''
FERRY = ''
FIELD HOCKEY = ''
FILE_CABINET = ''
FILE_FOLDER = ''
FILM_FRAMES = ''
FILM PROJECTOR = ''
FIRE = ''
FIRECRACKER = ''
FIREWORKS = ''
FIRE ENGINE = ''
FIRE EXTINGUISHER = ''
FIRST_PLACE_MEDAL = ''
FIRST_QUARTER_MOON = ''
FIRST_QUARTER_MOON_FACE = ''
FISH = ''
FISHING_POLE = ''
FISH_CAKE_WITH_SWIRL = ''
FIVE_OCLOCK = ''
```

```
FIVE THIRTY = ''
FLAG_IN_HOLE = ''
FLAMINGO = '\U0001f9a9'
FLASHLIGHT = ''
FLATBREAD = '\U0001fad3'
FLAT SHOE = ''
FLEUR_DE_LIS = ''
FLEXED_BICEPS = ''
FLOPPY_DISK = ''
FLOWER_PLAYING_CARDS = ''
FLUSHED_FACE = ''
FLY = '\U0001fab0'
FLYING_DISC = ''
FLYING_SAUCER = ''
FOG = ''
FOGGY = ''
FOLDED_HANDS = ''
FONDUE = '\U0001fad5'
FOOT = ''
FOOTPRINTS = ''
FORK_AND_KNIFE = ''
FORK_AND_KNIFE_WITH_PLATE = ''
FORTUNE_COOKIE = ''
FOUNTAIN = ''
FOUNTAIN_PEN = ''
FOUR_LEAF_CLOVER = ''
FOUR_OCLOCK = ''
FOUR_THIRTY = ''
FOX = ''
FRAMED_PICTURE = ''
FREE_BUTTON = ''
FRENCH_FRIES = ''
FRIED_SHRIMP = ''
FROG = ''
FRONT_FACING_BABY_CHICK = ''
FROWNING_FACE = ''
```

```
FROWNING_FACE_WITH_OPEN_MOUTH = ''
FUEL PUMP = ''
FULL_MOON = ''
FULL_MOON_FACE = ''
FUNERAL URN = ''
GAME DIE = ''
GARLIC = '\U0001f9c4'
GEAR = ''
GEMINI = ''
GEM STONE = ''
GENIE = ''
GHOST = ''
GIRAFFE = ''
GIRL = ''
GLASSES = ''
GLASS OF MILK = ''
GLOBE_SHOWING_AMERICAS = ''
GLOBE_SHOWING_ASIA_AUSTRALIA = ''
GLOBE_SHOWING_EUROPE_AFRICA = ''
GLOBE_WITH_MERIDIANS = ''
GLOVES = ''
GLOWING_STAR = ''
GOAL_NET = ''
GOAT = ''
GOBLIN = ''
GOGGLES = ''
GORILLA = ''
GRADUATION_CAP = ''
GRAPES = ''
GREEN_APPLE = ''
GREEN_BOOK = ''
GREEN_CIRCLE = '\U0001f7e2'
GREEN_HEART = ''
GREEN_SALAD = ''
GREEN_SQUARE = '\U0001f7e9'
GRIMACING_FACE = ''
```

```
GRINNING_CAT = ''
GRINNING_CAT_WITH_SMILING_EYES = ''
GRINNING_FACE = ''
GRINNING_FACE_WITH_BIG_EYES = ''
GRINNING_FACE_WITH_SMILING_EYES = ''
GRINNING_FACE_WITH_SWEAT = ''
GRINNING_SQUINTING_FACE = ''
GROWING_HEART = ''
GUARD = ''
GUIDE_DOG = '\U0001f9ae'
GUITAR = ''
HAMBURGER = ''
HAMMER = ''
HAMMER_AND_PICK = ''
HAMMER_AND_WRENCH = ''
HAMSTER = ''
HANDBAG = ''
HANDSHAKE = ''
HAND_WITH_FINGERS_SPLAYED = ''
HATCHING CHICK = ''
HEADPHONE = ''
HEADSTONE = '\U0001faa6'
HEART_DECORATION = ''
HEART EXCLAMATION = ''
HEART_SUIT = ''
HEART_WITH_ARROW = ''
HEART_WITH_RIBBON = ''
HEAR_NO_EVIL_MONKEY = ''
HEAVY_DOLLAR_SIGN = ''
HEDGEHOG = ''
HELICOPTER = ''
HERB = ''
HIBISCUS = ''
HIGH_HEELED_SHOE = ''
HIGH_SPEED_TRAIN = ''
HIGH_VOLTAGE = ''
```

```
HIKING BOOT = ''
HINDU_TEMPLE = '\U0001f6d5'
HIPPOPOTAMUS = ''
HOLE = ''
HOLLOW RED CIRCLE = ''
HONEYBEE = ''
HONEY_POT = ''
HOOK = '\U0001fa9d'
HORIZONTAL_TRAFFIC_LIGHT = ''
HORSE = ''
HORSE_FACE = ''
HORSE_RACING = ''
HOSPITAL = ''
HOTEL = ''
HOT BEVERAGE = ''
HOT DOG = ''
HOT_FACE = ''
HOT_PEPPER = ''
HOT_SPRINGS = ''
HOURGLASS_DONE = ''
HOURGLASS_NOT_DONE = ''
HOUSE = ''
HOUSES = ''
HOUSE WITH GARDEN = ''
HUGGING_FACE = ''
HUNDRED_POINTS = ''
HUSHED_FACE = ''
HUT = '\U0001f6d6'
ICE = '\U0001f9ca'
ICE_CREAM = ''
ICE_HOCKEY = ''
ICE_SKATE = ''
ID_BUTTON = ''
INBOX_TRAY = ''
INCOMING ENVELOPE = ''
INDEX_POINTING_UP = ''
```

```
INFINITY = ''
INFORMATION = ''
INPUT_LATIN_LETTERS = ''
INPUT_LATIN_LOWERCASE = ''
INPUT LATIN UPPERCASE = ''
INPUT NUMBERS = ''
INPUT_SYMBOLS = ''
JACK_O_LANTERN = ''
JAPANESE_ACCEPTABLE_BUTTON = ''
JAPANESE_APPLICATION_BUTTON = ''
JAPANESE_BARGAIN_BUTTON = ''
JAPANESE_CASTLE = ''
JAPANESE_CONGRATULATIONS_BUTTON = ''
JAPANESE DISCOUNT BUTTON = ''
JAPANESE_DOLLS = ''
JAPANESE FREE OF CHARGE BUTTON = ''
JAPANESE HERE BUTTON = ''
JAPANESE_MONTHLY_AMOUNT_BUTTON = ''
JAPANESE_NOT_FREE_OF_CHARGE_BUTTON = ''
JAPANESE_NO_VACANCY_BUTTON = ''
JAPANESE_OPEN_FOR_BUSINESS_BUTTON = ''
JAPANESE_PASSING_GRADE_BUTTON = ''
JAPANESE_POST_OFFICE = ''
JAPANESE PROHIBITED BUTTON = ''
JAPANESE RESERVED BUTTON = ''
JAPANESE_SECRET_BUTTON = ''
JAPANESE_SERVICE_CHARGE_BUTTON = ''
JAPANESE_SYMBOL_FOR_BEGINNER = ''
JAPANESE_VACANCY_BUTTON = ''
JEANS = ''
JOKER = ''
JOYSTICK = ''
KAABA = ''
KANGAROO = ''
KEY = ''
KEYBOARD = ''
```

```
KICK SCOOTER = ''
KIMONO = ''
KISS = ''
KISSING_CAT = ''
KISSING FACE = ''
KISSING FACE WITH CLOSED EYES = ''
KISSING_FACE_WITH_SMILING_EYES = ''
KISS_MARK = ''
KITCHEN_KNIFE = ''
KITE = '\U0001fa81'
KIWI_FRUIT = ''
KNOT = '\U0001faa2'
KOALA = ''
LABEL = ''
LAB COAT = ''
LACROSSE = ''
LADDER = '\U0001fa9c'
LADY_BEETLE = ''
LAPTOP = ''
LARGE_BLUE_DIAMOND = ''
LARGE_ORANGE_DIAMOND = ''
LAST_QUARTER_MOON = ''
LAST_QUARTER_MOON_FACE = ''
LAST TRACK BUTTON = ''
LATIN CROSS = ''
LEAFY_GREEN = ''
LEAF_FLUTTERING_IN_WIND = ''
LEDGER = ''
LEFT ARROW = ''
LEFT_ARROW_CURVING_RIGHT = ''
LEFT_FACING_FIST = ''
LEFT_LUGGAGE = ''
LEFT_RIGHT_ARROW = ''
LEFT_SPEECH_BUBBLE = ''
LEG = ''
LEMON = ''
```

```
LEO = ''
LEOPARD = ''
LEVEL_SLIDER = ''
LIBRA = ''
LIGHT_BULB = ''
LIGHT_RAIL = ''
LIGHT_SKIN_TONE = ''
LINK = ''
LINKED_PAPERCLIPS = ''
LION = ''
LIPSTICK = ''
LITTER_IN_BIN_SIGN = ''
LIZARD = ''
LLAMA = ''
LOBSTER = ''
LOCKED = ''
LOCKED_WITH_KEY = ''
LOCKED_WITH_PEN = ''
LOCOMOTIVE = ''
LOLLIPOP = ''
LONG_DRUM = '\U0001fa98'
LOTION_BOTTLE = ''
LOUDLY_CRYING_FACE = ''
LOUDSPEAKER = ''
LOVE HOTEL = ''
LOVE_LETTER = ''
LOVE_YOU_GESTURE = ''
LUGGAGE = ''
LUNGS = '\U0001fac1'
LYING_FACE = ''
MAGE = ''
MAGIC_WAND = '\U0001fa84'
MAGNET = ''
MAGNIFYING_GLASS_TILTED_LEFT = ''
MAGNIFYING_GLASS_TILTED_RIGHT = ''
MAHJONG_RED_DRAGON = ''
```

```
MALE SIGN = ''
MAMMOTH = '\U0001f9a3'
MAN = ''
MANGO = ''
MANS SHOE = ''
MANTELPIECE CLOCK = ''
MANUAL_WHEELCHAIR = '\U0001f9bd'
MAN_BEARD = ''
MAN DANCING = ''
MAPLE_LEAF = ''
MAP_OF_JAPAN = ''
MARTIAL_ARTS_UNIFORM = ''
MATE = '\U0001f9c9'
MEAT_ON_BONE = ''
MECHANICAL_ARM = '\U0001f9be'
MECHANICAL LEG = '\U0001f9bf'
MEDICAL_SYMBOL = ''
MEDIUM_DARK_SKIN_TONE = ''
MEDIUM_LIGHT_SKIN_TONE = ''
MEDIUM_SKIN_TONE = ''
MEGAPHONE = ''
MELON = ''
MEMO = ''
MENORAH = ''
MENS ROOM = ''
MEN_HOLDING_HANDS = ''
MERPERSON = ''
METRO = ''
MICROBE = ''
MICROPHONE = ''
MICROSCOPE = ''
MIDDLE_FINGER = ''
MILITARY_HELMET = '\U0001fa96'
MILITARY_MEDAL = ''
MILKY_WAY = ''
MINIBUS = ''
```

```
MINUS = ''
MIRROR = '\U0001fa9e'
MOAI = ''
MOBILE_PHONE = ''
MOBILE_PHONE_OFF = ''
MOBILE_PHONE_WITH_ARROW = ''
MONEY_BAG = ''
MONEY_MOUTH_FACE = ''
MONEY_WITH_WINGS = ''
MONKEY = ''
MONKEY_FACE = ''
MONORAIL = ''
MOON_CAKE = ''
MOON_VIEWING_CEREMONY = ''
MOSQUE = ''
MOSQUITO = ''
MOTORCYCLE = ''
MOTORIZED_WHEELCHAIR = '\U0001f9bc'
MOTORWAY = ''
MOTOR_BOAT = ''
MOTOR_SCOOTER = ''
MOUNTAIN = ''
MOUNTAIN_CABLEWAY = ''
MOUNTAIN RAILWAY = ''
MOUNT_FUJI = ''
MOUSE = ''
MOUSE_FACE = ''
MOUSE_TRAP = '\U0001faa4'
MOUTH = ''
MOVIE_CAMERA = ''
MRS_CLAUS = ''
MULTIPLY = ''
MUSHROOM = ''
MUSICAL_KEYBOARD = ''
MUSICAL NOTE = ''
MUSICAL_NOTES = ''
```

```
MUSICAL SCORE = ''
MUTED_SPEAKER = ''
NAIL_POLISH = ''
NAME_BADGE = ''
NATIONAL PARK = ''
NAUSEATED FACE = ''
NAZAR_AMULET = ''
NECKTIE = ''
NERD_FACE = ''
NESTING_DOLLS = '\U0001fa86'
NEUTRAL_FACE = ''
NEWSPAPER = ''
NEW BUTTON = ''
NEW MOON = ''
NEW_MOON_FACE = ''
NEXT TRACK BUTTON = ''
NG BUTTON = ''
NIGHT_WITH_STARS = ''
NINE_OCLOCK = ''
NINE_THIRTY = ''
NINJA = '\U0001f977'
NON_POTABLE_WATER = ''
NOSE = ''
NOTEBOOK = ''
NOTEBOOK_WITH_DECORATIVE_COVER = ''
NO_BICYCLES = ''
NO_ENTRY = ''
NO_LITTERING = ''
NO MOBILE PHONES = ''
NO_ONE_UNDER_EIGHTEEN = ''
NO_PEDESTRIANS = ''
NO_SMOKING = ''
NUT_AND_BOLT = ''
OCTOPUS = ''
ODEN = ''
OFFICE_BUILDING = ''
```

```
OGRE = ''
OIL_DRUM = ''
OK_BUTTON = ''
OK_HAND = ''
OLDER PERSON = ''
OLD KEY = ''
OLD_MAN = ''
OLD_WOMAN = ''
OLIVE = '\U0001fad2'
OM = ''
ONCOMING_AUTOMOBILE = ''
ONCOMING_BUS = ''
ONCOMING_FIST = ''
ONCOMING_POLICE_CAR = ''
ONCOMING_TAXI = ''
ONE OCLOCK = ''
ONE_PIECE_SWIMSUIT = '\U0001fa71'
ONE_THIRTY = ''
ONION = '\U0001f9c5'
ON ARROW = ''
OPEN_BOOK = ''
OPEN_FILE_FOLDER = ''
OPEN_HANDS = ''
OPEN MAILBOX WITH LOWERED FLAG = ''
OPEN_MAILBOX_WITH_RAISED_FLAG = ''
OPHIUCHUS = ''
OPTICAL_DISK = ''
ORANGE_BOOK = ''
ORANGE_CIRCLE = '\U0001f7e0'
ORANGE_HEART = ''
ORANGE_SQUARE = '\U0001f7e7'
ORANGUTAN = '\0001f9a7'
ORTHODOX_CROSS = ''
OTTER = '\U0001f9a6'
OUTBOX TRAY = ''
OWL = ''
```

```
OX = ''
OYSTER = '\U0001f9aa'
O_BUTTON_BLOOD_TYPE = ''
PACKAGE = ''
PAGER = ''
PAGE FACING UP = ''
PAGE_WITH_CURL = ''
PAINTBRUSH = ''
PALMS_UP_TOGETHER = ''
PALM_TREE = ''
PANCAKES = ''
PANDA = ''
PAPERCLIP = ''
PARACHUTE = '\U0001fa82'
PARROT = ''
PARTYING FACE = ''
PARTY POPPER = ''
PART_ALTERNATION_MARK = ''
PASSENGER_SHIP = ''
PASSPORT CONTROL = ''
PAUSE_BUTTON = ''
PAW_PRINTS = ''
PEACE_SYMBOL = ''
PEACH = ''
PEACOCK = ''
PEANUTS = ''
PEAR = ''
PEN = ''
PENCIL = ''
PENGUIN = ''
PENSIVE_FACE = ''
PEOPLE_HUGGING = '\U0001fac2'
PEOPLE_WITH_BUNNY_EARS = ''
PEOPLE_WRESTLING = ''
PERFORMING_ARTS = ''
PERSEVERING_FACE = ''
```

```
PERSON = ''
PERSON_BIKING = ''
PERSON_BLOND_HAIR = ''
PERSON_BOUNCING_BALL = ''
PERSON BOWING = ''
PERSON CARTWHEELING = ''
PERSON_CLIMBING = ''
PERSON_FACEPALMING = ''
PERSON FENCING = ''
PERSON_FROWNING = ''
PERSON_GESTURING_NO = ''
PERSON_GESTURING_OK = ''
PERSON_GETTING_HAIRCUT = ''
PERSON_GETTING_MASSAGE = ''
PERSON_GOLFING = ''
PERSON IN BED = ''
PERSON_IN_LOTUS_POSITION = ''
PERSON_IN_STEAMY_ROOM = ''
PERSON_IN_SUIT_LEVITATING = ''
PERSON_IN_TUXEDO = ''
PERSON_JUGGLING = ''
PERSON_KNEELING = '\U0001f9ce'
PERSON_LIFTING_WEIGHTS = ''
PERSON MOUNTAIN BIKING = ''
PERSON PLAYING HANDBALL = ''
PERSON_PLAYING_WATER_POLO = ''
PERSON_POUTING = ''
PERSON_RAISING_HAND = ''
PERSON ROWING BOAT = ''
PERSON_RUNNING = ''
PERSON_SHRUGGING = ''
PERSON_STANDING = '\U0001f9cd'
PERSON_SURFING = ''
PERSON_SWIMMING = ''
PERSON_TAKING_BATH = ''
PERSON_TIPPING_HAND = ''
```

```
PERSON WALKING = ''
PERSON_WEARING_TURBAN = ''
PERSON_WITH_SKULLCAP = ''
PERSON_WITH_VEIL = ''
PETRI DISH = ''
PICK = ''
PICKUP_TRUCK = '\U0001f6fb'
PIE = ''
PIG = ''
PIG_FACE = ''
PIG_NOSE = ''
PILE_OF_POO = ''
PILL = ''
PINCHED_FINGERS = '\U0001f90c'
PINCHING_HAND = '\U0001f90f'
PINEAPPLE = ''
PINE_DECORATION = ''
PING_PONG = ''
PISCES = ''
PISTOL = ''
PIZZA = ''
PIñATA = '\U0001fa85'
PLACARD = '\U0001faa7'
PLACE_OF_WORSHIP = ''
PLAY BUTTON = ''
PLAY_OR_PAUSE_BUTTON = ''
PLEADING_FACE = ''
PLUNGER = '\U0001faa0'
PLUS = ''
POLICE_CAR = ''
POLICE_CAR_LIGHT = ''
POLICE_OFFICER = ''
POODLE = ''
POPCORN = ''
POSTAL HORN = ''
POSTBOX = ''
```

```
POST_OFFICE = ''
POTABLE_WATER = ''
POTATO = ''
POTTED_PLANT = '\U0001fab4'
POT_OF_FOOD = ''
POULTRY_LEG = ''
POUND_BANKNOTE = ''
POUTING_CAT = ''
POUTING_FACE = ''
PRAYER_BEADS = ''
PREGNANT_WOMAN = ''
PRETZEL = ''
PRINCE = ''
PRINCESS = ''
PRINTER = ''
PROHIBITED = ''
PURPLE_CIRCLE = '\U0001f7e3'
PURPLE_HEART = ''
PURPLE_SQUARE = '\U0001f7ea'
PURSE = ''
PUSHPIN = ''
PUZZLE_PIECE = ''
P_BUTTON = ''
QUESTION MARK = ''
RABBIT = ''
RABBIT_FACE = ''
RACCOON = ''
RACING_CAR = ''
RADIO = ''
RADIOACTIVE = ''
RADIO_BUTTON = ''
RAILWAY_CAR = ''
RAILWAY_TRACK = ''
RAINBOW = ''
RAISED_BACK_OF_HAND = ''
RAISED_FIST = ''
```

```
RAISED HAND = ''
RAISING HANDS = ''
RAM = ''
RAT = ''
RAZOR = '\U0001fa92'
RECEIPT = ''
RECORD_BUTTON = ''
RECYCLING_SYMBOL = ''
RED APPLE = ''
RED_CIRCLE = ''
RED_ENVELOPE = ''
RED_HAIR = ''
RED_HEART = ''
RED_PAPER_LANTERN = ''
RED_SQUARE = '\U0001f7e5'
RED TRIANGLE POINTED DOWN = ''
RED_TRIANGLE_POINTED_UP = ''
REGISTERED = '®'
RELIEVED_FACE = ''
REMINDER RIBBON = ''
REPEAT_BUTTON = ''
REPEAT_SINGLE_BUTTON = ''
RESCUE_WORKERS_HELMET = ''
RESTROOM = ''
REVERSE BUTTON = ''
REVOLVING_HEARTS = ''
RHINOCEROS = ''
RIBBON = ''
RICE BALL = ''
RICE_CRACKER = ''
RIGHT_ANGER_BUBBLE = ''
RIGHT_ARROW = ''
RIGHT_ARROW_CURVING_DOWN = ''
RIGHT_ARROW_CURVING_LEFT = ''
RIGHT_ARROW_CURVING_UP = ''
RIGHT_FACING_FIST = ''
```

```
RING = ''
RINGED_PLANET = '\U0001fa90'
ROASTED_SWEET_POTATO = ''
ROBOT = ''
ROCK = '\U0001faa8'
ROCKET = ''
ROLLED_UP_NEWSPAPER = ''
ROLLER_COASTER = ''
ROLLER_SKATE = '\U0001f6fc'
ROLLING_ON_THE_FLOOR_LAUGHING = ''
ROLL_OF_PAPER = ''
ROOSTER = ''
ROSE = ''
ROSETTE = ''
ROUND_PUSHPIN = ''
RUGBY FOOTBALL = ''
RUNNING_SHIRT = ''
RUNNING_SHOE = ''
SAD_BUT_RELIEVED_FACE = ''
SAFETY_PIN = ''
SAFETY_VEST = '\U0001f9ba'
SAGITTARIUS = ''
SAILBOAT = ''
SAKE = ''
SALT = ''
SANDWICH = ''
SANTA_CLAUS = ''
SARI = '\U0001f97b'
SATELLITE = ''
SATELLITE_ANTENNA = ''
SAUROPOD = ''
SAXOPHONE = ''
SCARF = ''
SCHOOL = ''
SCISSORS = ''
SCORPIO = ''
```

```
SCORPION = ''
SCREWDRIVER = '\U0001fa9b'
SCROLL = ''
SEAL = '\U0001f9ad'
SEAT = ''
SECOND PLACE MEDAL = ''
SEEDLING = ''
SEE_NO_EVIL_MONKEY = ''
SELFIE = ''
SEVEN_OCLOCK = ''
SEVEN_THIRTY = ''
SEWING_NEEDLE = '\U0001faa1'
SHALLOW_PAN_OF_FOOD = ''
SHAMROCK = ''
SHARK = ''
SHAVED ICE = ''
SHEAF_OF_RICE = ''
SHIELD = ''
SHINTO_SHRINE = ''
SHIP = ''
SHOOTING_STAR = ''
SHOPPING_BAGS = ''
SHOPPING_CART = ''
SHORTCAKE = ''
SHORTS = '\U0001fa73'
SHOWER = ''
SHRIMP = ''
SHUFFLE_TRACKS_BUTTON = ''
SHUSHING_FACE = ''
SIGN_OF_THE_HORNS = ''
SIX_OCLOCK = ''
SIX_THIRTY = ''
SKATEBOARD = ''
SKIER = ''
SKIS = ''
SKULL = ''
```

```
SKULL_AND_CROSSBONES = ''
SKUNK = '\U0001f9a8'
SLED = ''
SLEEPING_FACE = ''
SLEEPY FACE = ''
SLIGHTLY FROWNING FACE = ''
SLIGHTLY_SMILING_FACE = ''
SLOTH = '\U0001f9a5'
SLOT MACHINE = ''
SMALL_AIRPLANE = ''
SMALL_BLUE_DIAMOND = ''
SMALL_ORANGE_DIAMOND = ''
SMILING_CAT_WITH_HEART_EYES = ''
SMILING_FACE = ''
SMILING_FACE_WITH_HALO = ''
SMILING FACE WITH HEARTS = ''
SMILING_FACE_WITH_HEART_EYES = ''
SMILING_FACE_WITH_HORNS = ''
SMILING_FACE_WITH_SMILING_EYES = ''
SMILING_FACE_WITH_SUNGLASSES = ''
SMILING_FACE_WITH_TEAR = '\U0001f972'
SMIRKING_FACE = ''
SNAIL = ''
SNAKE = ''
SNEEZING FACE = ''
SNOWBOARDER = ''
SNOWFLAKE = ''
SNOWMAN = ''
SNOWMAN WITHOUT SNOW = ''
SNOW_CAPPED_MOUNTAIN = ''
SOAP = ''
SOCCER_BALL = ''
SOCKS = ''
SOFTBALL = ''
SOFT ICE CREAM = ''
SOON_ARROW = ''
```

```
SOS BUTTON = ''
SPADE SUIT = ''
SPAGHETTI = ''
SPARKLE = ''
SPARKLER = ''
SPARKLES = ''
SPARKLING_HEART = ''
SPEAKER_HIGH_VOLUME = ''
SPEAKER_LOW_VOLUME = ''
SPEAKER_MEDIUM_VOLUME = ''
SPEAKING_HEAD = ''
SPEAK_NO_EVIL_MONKEY = ''
SPEECH_BALLOON = ''
SPEEDBOAT = ''
SPIDER = ''
SPIDER WEB = ''
SPIRAL CALENDAR = ''
SPIRAL_NOTEPAD = ''
SPIRAL_SHELL = ''
SPONGE = ''
SPOON = ''
SPORTS_MEDAL = ''
SPORT_UTILITY_VEHICLE = ''
SPOUTING WHALE = ''
SQUID = ''
SQUINTING_FACE_WITH_TONGUE = ''
STADIUM = ''
STAR = ''
STAR_AND_CRESCENT = ''
STAR_OF_DAVID = ''
STAR_STRUCK = ''
STATION = ''
STATUE_OF_LIBERTY = ''
STEAMING_BOWL = ''
STETHOSCOPE = '\U0001fa7a'
STOPWATCH = ''
```

```
STOP_BUTTON = ''
STOP_SIGN = ''
STRAIGHT_RULER = ''
STRAWBERRY = ''
STUDIO MICROPHONE = ''
STUFFED FLATBREAD = ''
SUN = ''
SUNFLOWER = ''
SUNGLASSES = ''
SUNRISE = ''
SUNRISE_OVER_MOUNTAINS = ''
SUNSET = ''
SUN_BEHIND_CLOUD = ''
SUN_BEHIND_LARGE_CLOUD = ''
SUN_BEHIND_RAIN_CLOUD = ''
SUN_BEHIND_SMALL_CLOUD = ''
SUN_WITH_FACE = ''
SUPERHERO = ''
SUPERVILLAIN = ''
SUSHI = ''
SUSPENSION_RAILWAY = ''
SWAN = ''
SWEAT_DROPLETS = ''
SYNAGOGUE = ''
SYRINGE = ''
TACO = ''
TAKEOUT_BOX = ''
TAMALE = '\U0001fad4'
TANABATA TREE = ''
TANGERINE = ''
TAURUS = ''
TAXI = ''
TEACUP_WITHOUT_HANDLE = ''
TEAPOT = '\U0001fad6'
TEAR_OFF_CALENDAR = ''
TEDDY_BEAR = ''
```

```
TELEPHONE = ''
TELEPHONE_RECEIVER = ''
TELESCOPE = ''
TELEVISION = ''
TENNIS = ''
TENT = ''
TEN_OCLOCK = ''
TEN_THIRTY = ''
TEST_TUBE = ''
THERMOMETER = ''
THINKING_FACE = ''
THIRD_PLACE_MEDAL = ''
THONG_SANDAL = '\U0001fa74'
THOUGHT_BALLOON = ''
THREAD = ''
THREE OCLOCK = ''
THREE_THIRTY = ''
THUMBS_DOWN = ''
THUMBS_UP = ''
TICKET = ''
TIGER = ''
TIGER_FACE = ''
TIMER_CLOCK = ''
TIRED FACE = ''
TOILET = ''
TOKYO_TOWER = ''
TOMATO = ''
TONGUE = ''
TOOLBOX = ''
TOOTH = ''
TOOTHBRUSH = '\U0001faa5'
TOP_ARROW = ''
TOP_HAT = ''
TORNADO = ''
TRACKBALL = ''
TRACTOR = ''
```

```
TRADE_MARK = 'TM'
TRAIN = ''
TRAM = ''
TRAM_CAR = ''
TRANSGENDER_SYMBOL = ''
TRIANGULAR_FLAG = ''
TRIANGULAR_RULER = ''
TRIDENT_EMBLEM = ''
TROLLEYBUS = ''
TROPHY = ''
TROPICAL_DRINK = ''
TROPICAL_FISH = ''
TRUMPET = ''
TULIP = ''
TUMBLER_GLASS = ''
TURKEY = ''
TURTLE = ''
TWELVE_OCLOCK = ''
TWELVE_THIRTY = ''
TWO HEARTS = ''
TWO_HUMP_CAMEL = ''
TWO_OCLOCK = ''
TWO_THIRTY = ''
T REX = ''
T SHIRT = ''
UMBRELLA = ''
UMBRELLA_ON_GROUND = ''
UMBRELLA_WITH_RAIN_DROPS = ''
UNAMUSED FACE = ''
UNICORN = ''
UNLOCKED = ''
UPSIDE_DOWN_FACE = ''
UPWARDS_BUTTON = ''
UP_ARROW = ''
UP_BUTTON = ''
UP_DOWN_ARROW = ''
```

```
UP LEFT ARROW = ''
UP_RIGHT_ARROW = ''
VAMPIRE = ''
VERTICAL_TRAFFIC_LIGHT = ''
VIBRATION MODE = ''
VICTORY HAND = ''
VIDEOCASSETTE = ''
VIDEO_CAMERA = ''
VIDEO_GAME = ''
VIOLIN = ''
VIRGO = ''
VOLCANO = ''
VOLLEYBALL = ''
VS BUTTON = ''
VULCAN_SALUTE = ''
WAFFLE = '\U0001f9c7'
WANING_CRESCENT_MOON = ''
WANING_GIBBOUS_MOON = ''
WARNING = ''
WASTEBASKET = ''
WATCH = ''
WATERMELON = ''
WATER_BUFFALO = ''
WATER CLOSET = ''
WATER WAVE = ''
WAVING_HAND = ''
WAVY_DASH = ''
WAXING_CRESCENT_MOON = ''
WAXING_GIBBOUS_MOON = ''
WEARY_CAT = ''
WEARY_FACE = ''
WEDDING = ''
WHALE = ''
WHEELCHAIR_SYMBOL = ''
WHEEL OF DHARMA = ''
WHITE_CANE = '\U0001f9af'
```

```
WHITE CIRCLE = ''
WHITE_EXCLAMATION_MARK = ''
WHITE_FLAG = ''
WHITE_FLOWER = ''
WHITE HAIR = ''
WHITE_HEART = '\U0001f90d'
WHITE_LARGE_SQUARE = ''
WHITE_MEDIUM_SMALL_SQUARE = ''
WHITE_MEDIUM_SQUARE = ''
WHITE_QUESTION_MARK = ''
WHITE_SMALL_SQUARE = ''
WHITE_SQUARE_BUTTON = ''
WILTED_FLOWER = ''
WINDOW = '\U0001fa9f'
WIND CHIME = ''
WIND FACE = ''
WINE_GLASS = ''
WINKING_FACE = ''
WINKING_FACE_WITH_TONGUE = ''
WOLF = ''
WOMAN = ''
WOMANS_BOOT = ''
WOMANS_CLOTHES = ''
WOMANS HAT = ''
WOMANS_SANDAL = ''
WOMAN_AND_MAN_HOLDING_HANDS = ''
WOMAN_DANCING = ''
WOMAN_WITH_HEADSCARF = ''
WOMENS_ROOM = ''
WOMEN_HOLDING_HANDS = ''
WOOD = '\U0001fab5'
WOOZY_FACE = ''
WORLD_MAP = ''
WORM = '\U0001fab1'
WORRIED FACE = ''
WRAPPED_GIFT = ''
```

```
WRENCH = ''
WRITING_HAND = ''
YARN = ''
YAWNING_FACE = '\U0001f971'
YELLOW_CIRCLE = '\U0001f7e1'
YELLOW_HEART = ''
YELLOW_SQUARE = '\U0001f7e8'
YEN_BANKNOTE = ''
YIN_YANG = ''
YO_YO = '\U0001fa80'
ZANY_FACE = ''
ZEBRA = ''
ZIPPER_MOUTH_FACE = ''
ZOMBIE = ''
ZZZ = ''
```

3.2.2 Fonts

Fonts in the pygamelib are nothing more than a specially organized sprite collection.

The way to use it is extremely simple: you instantiate a Font object and ask it to load the data from a specific font.

For example to load the 8bits font, you do:

Example:: from pygamelib.gfx import core

```
my_font = core.Font('8bits')
```

That's it! The you can use it to format Text objects.

8bits

New in version 1.3.0.

How to use?

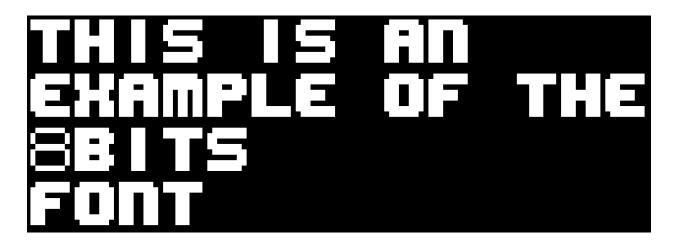
Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("8bits")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

3.2. assets 177

What does it look like?



figlet-caligraphy

New in version 1.3.0.

How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-caligraphy")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?

* ********** ** *			
* *stototototok *sk *sksk	stotok		
* * * * ** *			
** * ** ** ** ***	****		
* *otok *ok *ok * *otok * * *otok *	**************************************	yololok yolok yololok	
** **	*** ** ****	* **** * **** **** * ***** ***	
** *** *** *** *** *** ***	** ****	sk skolokok skok skokosk skok skok skok	
** ** ** ** ** *** ***	** *** ** ***	** ** ** ** ** **	
** ** ** ** **	*** ****	** ** ***	
** * ** ** ** ***	** **** **	** ** **	
xotok * ** *ok *ok * *okotok *	** * *** *	** ** **	
sololokokok sok sok sokok se sololok	yokok * yokok	stokołok stok stokok stokok	
*** ** **	***	*** ** ***	
*			
*			
*			
*			
	***	*oks*	*
	***	** ***	* **
	*ok	work work	kok kok
	**	**	** **
yolok yolok	***	**** **	********* **
*** * *** **** * **** *** **** ****	* *** * ** ***	* *** * *****	*okokolololok *ok *okok *okok
	* * *** ** * ***	* ***	** ** *** ***
* **** *** *** ** *** ** *** *** *** *	** ** ** ** ** ***	** ** ** **	** ** ** ***
** *** *** ** ** ** ** **	** ** ** ** ***	** ** **	** ** ** ** ***
kotototototok	dok dok dokolokok	yok yok yok	kok kok kokokokokok
** * *** ** ** ** **	** ** **	** ** **	** ** **
kototok * * kokok * kok * kok * kok * kok * kok	skolokokok sk skolok sk	*otolokokok *kok	tok tok tokokok tok
******* * *** **** *** *** ***		**** **	** ** ** ****
****** ** *** *** ***	* ** *** ****	**	** ** ****
	**		*
	**		*
	**		*
			4
***		***	*
*** *** ** ***		*** *** *	
*** ****	*o*	***	*
** ***	** **	***	* ** ** **
** ***	kokololokok kok	***	* ** ** ** ** *** *** ***
** ***	** ** *** ****** ****** *****	***	* ** ** ** ** ** ** ** ** **
** ***	** ** ******* ** * *** * ***	*** * ** *** ** *** ** *** **** ***	* ** ** ** ** ** ** ** ** ** **
** ***	** ** *** ****** ****** *****	### # ## *## ## ### #### #### ## *## #### ##	* ** ** ** ** ** ** ** ** **
** ***	** ** ** ** ** ** ** ** ** **	### # ## *## ## ### #### #### ## *## #### ##	* ** ** ** ** ** ** ** ** ** ** ** ** *
** *** * *** *** *** *** *** *** *** *	** ** ** ** ** ** ** ** ** ** **	### # ### #### #### #### #### #### #### ####	* ** ** ** ** ** ** ** ** **
## ### # ### ## ### ### ### ### #######	** ** ** ** ** ** ** ** ** ** ** ** **	*** * ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** **	* ** ** ** ** ** ** ** ** ** **
** *** * ** ** *** ** ** ** ** ** ** ** **	** ** ** ** ** ** ** ** ** ** ** ** **	*** * ** *** *** * ** *** *** * ** *** *** * ** *** * ** *** * ** *** * ** **	* ** ** ** ** ** ** ** ** **
## ### ### ### ### ### ### ### ########	** ** ** ** ** ** ** ** ** **	*** * *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** **	* ** ** ** ** ** ** ** ** **
## ### # ### ### ### #################	** ** ** ** ** ** ** ** ** ** ** ** **	*** * ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *** *** *** ** *** *** *** *** *** *** *** *** *** *	* ** ** ** ** ** ** ** ** **
## \$44	** ** ** ** ** ** ** ** ** ** ** ** **	### # ### #### #### #### #### #### #### ####	** ** ** ** ** ** ** ** ** **
## ### ## ## ## ### ### ##############	** ** ** ** ** ** ** ** ** **	*** * ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* ** ** ** ** ** ** ** ** **
## \$44	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	** ** ** ** ** ** ** ** ** **
## ### ## ### ### ####################	** ** ** ** ** ** ** ** ** **	*** * ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### ## ### ### ####################	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### ## ### ### ####################	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### # ### ## ### ### ##############	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### ### ### ### ### ###############	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### ## ### ### ### ################	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### # ### ### #####################	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### ## ## ## ## ### ### ### ########	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ***	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ***	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ***	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ***	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ***	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### ##	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ### ### ### ### ### ### ### ### ###	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *
## ***	** ** ** ** ** ** ** ** ** ** ** ** **	*** ** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** *** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **	* * * * * * * * * * * * * * * * * * *

More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

3.2. assets 179

figlet-doom

New in version 1.3.0.

How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-doom")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?



More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

figlet-graffiti

New in version 1.3.0.

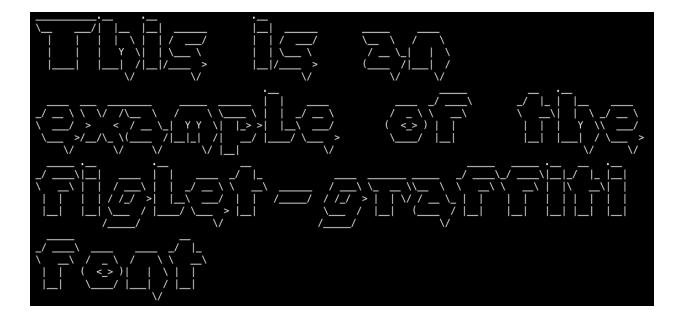
How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-graffiti")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?



More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

3.2. assets 181

figlet-mirror

New in version 1.3.0.

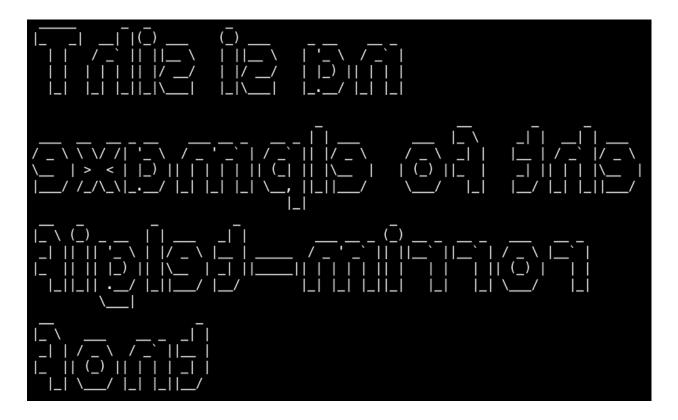
How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-mirror")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?



More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

figlet-pepper

New in version 1.3.0.

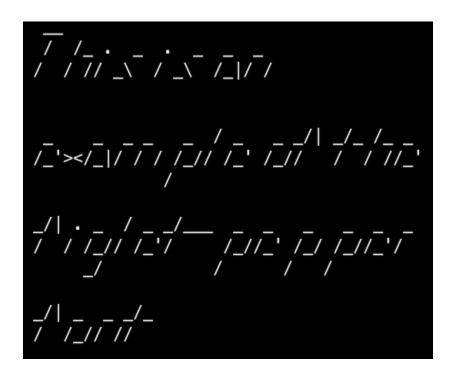
How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-pepper")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?



More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

figlet-poison

New in version 1.3.0.

3.2. assets 183

How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-poison")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?

```
| Secretaria | Sec
```

More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

figlet-puffy

New in version 1.3.0.

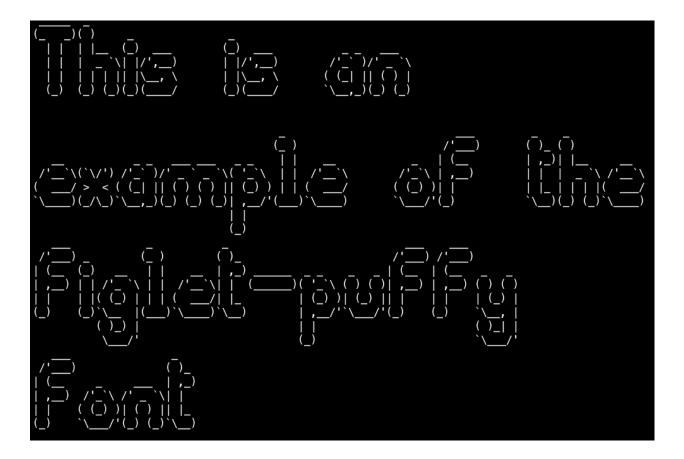
How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-puffy")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?



More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

3.2. assets 185

figlet-rounded

New in version 1.3.0.

How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-rounded")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?



More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

figlet-stampatello

New in version 1.3.0.

How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-stampatello")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?

```
This is an example of the figlet-stampatello font
```

3.2. assets 187

More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

figlet-univers

New in version 1.3.0.

How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-univers")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?

More

Please visit the FIGlet project: http://www.figlet.org/

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

figlet-wavy

New in version 1.3.0.

How to use?

Example:

```
from pygamelib.gfx.core import Font
from pygamelib.base import Text

my_font = Font("figlet-wavy")
my_text = Text("Lorem Ipsum", font=my_font)
my_text.print_formatted()
```

What does it look like?

More

Please visit the FIGlet project: http://www.figlet.org/

3.2. assets 189

The fonts prefixed by "figlet-" are but a small portion of what's available here.

To easily convert FIGlet fonts to the pygamelib's format you can you the figlet-to-pygamelib script available here: https://github.com/pygamelib/figlet-to-pygamelib

3.3 base

The base module provide basic objects and exceptions that are used by the entire library.

3.3.1 Console

```
class pygamelib.base.Console
    Bases: object
```

The Console class is a singleton wrapper around the blessed. Terminal() class. Since the library is using Terminal a lot, it is both useful and efficient to have a quick access to a single instance of the class.

This class only expose one method: instance() that returns the singleton instance.

Methods

instance()

Returns the instance of the blessed. Terminal object.

classmethod instance()

Returns the instance of the blessed. Terminal object.

New in version 1.3.0.

The pygamelib extensively use the Terminal object from the blessed module. However we find ourselves in need of a Terminal instance a lot, so to help with memory and execution time we just encapsulate the Terminal object in a singleton so any object can use it without instantiating it many times (and messing up with the contexts).

Returns Instance of blessed. Terminal object

Example:

```
term = Console.instance()
```

3.3.2 Math

```
class pygamelib.base.Math
```

Bases: object

The math class regroup math functions required for game development.

New in version 1.2.0.

For the moment there is only static methods in that class but it will evolve in the future.

```
init ()
```

Initialize self. See help(type(self)) for accurate signature.

Methods

init()	Initialize self.
distance(row1, column1, row2, column2)	Return the euclidean distance between to points.
<pre>intersect(row1, column1, width1, height1,)</pre>	This function check if 2 rectangles intersect.
lerp(a, b, t)	Return the linear interpolation between 2 values rel-
	ative to a third value.

static distance (row1: int, column1: int, row2: int, column2: int) \rightarrow float Return the euclidean distance between to points.

Points are identified by their row and column. If you want the distance in number of cells, you need to round the result (see example).

Parameters

- row1 (int) the row number (coordinate) of the first point.
- column1 (int) the column number (coordinate) of the first point.
- row2 (int) the row number (coordinate) of the second point.
- column2 (int) the column number (coordinate) of the second point.

Returns The distance between the 2 points.

Return type float

Example:

static intersect (row1: int, column1: int, width1: int, height1: int, row2: int, column2: int, width2: int, height2: int) \rightarrow bool This function check if 2 rectangles intersect.

The 2 rectangles are defined by their positions (row, column) and dimension (width and height).

Parameters

- row1 (int) The row of the first rectangle
- **column1** (int) The column of the first rectangle
- width1 (int) The width of the first rectangle
- height1 (int) The height of the first rectangle
- row2 (int) The row of the second rectangle
- column2 The column of the second rectangle
- width2 (int) The width of the second rectangle
- height2 (int) The height of the second rectangle

Returns A boolean, True if the rectangles intersect False, otherwise.

Example:

static lerp (a: float, b: float, t: float) \rightarrow float

Return the linear interpolation between 2 values relative to a third value.

New in version 1.3.0.

Parameters

- **a** (float) Start value of the interpolation. Returned if t is 0.
- **b** (float) End value of the interpolation. Returned if t is 1.
- t (float) A value between 0 and 1 used to interpolate between a and b.

Example:

```
value = lerp(0, 100, 0.5) # 50
```

3.3.3 PglBaseObject

```
class pygamelib.base.PglBaseObject
```

Bases: object

The base object of most of the pygamelib's classes.

New in version 1.3.0.

The PglBaseObject has 2 goals:

- Store the object's screen position.
- Implements a modified observer design pattern.

It is "modified" as it acts both as the observer and the client. The idea behind it is that any object can observe and be observed by any other objects.

The base logic of the pattern is already implemented and probably does not require re-implementation on the child object. However, the <code>handle_notification()</code> method needs to be implemented in each client. The actual processing of the notification is indeed specific to each object.

Storing the screen position is particularly useful for <code>BoardItem</code> subclasses as they only know their position relative to the <code>Board</code> but might need to know their absolute screen coordinates.

This is a lightweight solution to that issue. It is not foolproof however! The screen_row and screen_column attributes are not wrapped properties and can be modified to mess up things. It shouldn't be done lightly. You have been warned!

```
\underline{\hspace{1cm}}init\underline{\hspace{1cm}}() \rightarrow None
```

Like the object class, this class constructor takes no parameter.

Methods

init()	Like the object class, this class constructor takes no
	parameter.
	Continued on next page

T	~~		•		
Iania	ソドー	continued	trom	nravinie	nana
iabic	20	CONTINUCA	11 0111	picvious	page

attach(observer)		Attach an observer to this instance.
detach(observer)		Detach an observer from this instance.
handle_notification(subject[,	attribute,	A virtual method that needs to be implemented by
value])		the observer.
notify([modifier])		Notify all the observers that a change occurred.
store_screen_position(row, colu	ımn)	Store the screen position of the object.

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

• **subject** (*PqlBaseObject*) – The object that has changed.

- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

 $store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.3.4 PglException

```
exception pygamelib.base.PglException(error, message)
```

Exception raised for non specific errors in the pygamelib.

3.3.5 PglInvalidLevelException

exception pygamelib.base.PglInvalidLevelException(message)

Exception raised if a level is not associated to a board in Game().

3.3.6 PglInvalidTypeException

exception pygamelib.base.PglInvalidTypeException(message)

Exception raised for invalid types.

3.3.7 PglInventoryException

exception pygamelib.base.PglInventoryException(error, message)

Exception raised for issue related to the inventory. The error is an explicit string, and the message explains the error.

3.3.8 PglObjectIsNotMovableException

exception pygamelib.base.PglObjectIsNotMovableException (message)

Exception raised if the object that is being moved is not a subclass of Movable.

3.3.9 PglOutOfBoardBoundException

exception pygamelib.base.PglOutOfBoardBoundException (message)

Exception for out of the board's boundaries operations.

3.3.10 Text

 $\textbf{class} \text{ pygamelib.base.} \textbf{Text} (\textit{text}=", \textit{fg_color=None}, \textit{bg_color=None}, \textit{style="}", \textit{font=None})$

Bases: pygamelib.base.PglBaseObject

An object to manipulate and display text in multiple contexts.

New in version 1.2.0.

The Text class is a collection of text formatting and display static methods.

You can either instantiate an object or use the static methods.

The Text object allow for easy text manipulation through its collection of independent attributes. They help to set the text, its style and the foreground and background colors.

The Text object can be converted to a *Sprite* through the Sprite.from_text() method. This is particularly useful to the place text on the game *Board*.

__init__ (text=", fg_color=None, bg_color=None, style=", font=None)

Parameters

- **text** (str) The text to manipulate
- **fg_color** (*Color*) The foreground color for the text.
- **bg_color** (*Color*) The background color for the text.
- **style** (str) The style for the text.

• **font** (Font) – The font in which the text is going to be displayed (only works when using Screen.place() and Screen.update())

Methods

init([text, fg_color, bg_color, styl	le, font])	param text The text to manipulate
attach(observer)		Attach an observer to this instance.
<pre>handle_notification(target[, value])</pre>	attribute,	A virtual method that needs to be implemented by the observer.
black(message)		This method works exactly the way green_bright() work with different color.
black_bright(message)		This method works exactly the way green_bright() work with different color.
black_dim(message)		This method works exactly the way green_bright() work with different color.
blue(message)		This method works exactly the way green_bright() work with different color.
blue_bright(message)		This method works exactly the way green_bright() work with different color.
blue_dim(message)		This method works exactly the way green_bright() work with different color.
cyan(message)		This method works exactly the way green_bright() work with different color.
cyan_bright(message)		This method works exactly the way green_bright() work with different color.
cyan_dim(message)		This method works exactly the way green_bright() work with different color.
debug(message)		Print a debug message.
detach(observer)		Detach an observer from this instance.
fatal(message)		Print a fatal message.
green(message)		This method works exactly the way green_bright() work with different color.
green_bright(message)		Return a string formatted to be bright green
green_dim(message)		This method works exactly the way green_bright() work with different color.
info(message)		Print an informative message.
magenta(message)		This method works exactly the way green_bright() work with different color.
magenta_bright(message)		This method works exactly the way green_bright() work with different color.
magenta_dim(message)		This method works exactly the way green_bright() work with different color.
notify([modifier])		Notify all the observers that a change occurred.
print_white_on_red(message)		Print a white message over a red background.
red(message)		This method works exactly the way green_bright() work with different color.
red_bright(message)		This method works exactly the way green_bright() work with different color.
		Continued on next page

Table 28 – continued from previous page

red_dim(message)	This method works exactly the way green_bright()
	work with different color.
render_to_buffer(buffer, row, column,)	Render the Text object from the display buffer to the
	frame buffer.
warn(message)	Print a warning message.
white(message)	This method works exactly the way green_bright()
	work with different color.
white_bright(message)	This method works exactly the way green_bright()
	work with different color.
white_dim(message)	This method works exactly the way green_bright()
	work with different color.
yellow(message)	This method works exactly the way green_bright()
	work with different color.
yellow_bright(message)	This method works exactly the way green_bright()
	work with different color.
yellow_dim(message)	This method works exactly the way green_bright()
	work with different color.

Attributes

bg_color	The bg_color attribute sets the background color.
fg_color	The fg_color attribute sets the foreground color.
length	Return the true length of the text.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
text	The text attribute.
Text.style	
Text.parent	

$\mathtt{attach}\ (observer)$

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

bg_color

The bg_color attribute sets the background color. It needs to be a Color.

New in version 1.3.0.

When the background color is changed, the observers are notified of the change with the pygamelib.base.Text.bg_color:changed event. The new color is passed as the *value* parameter.

```
static black (message)
```

This method works exactly the way green_bright() work with different color.

static black_bright (message)

This method works exactly the way green_bright() work with different color.

static black_dim(message)

This method works exactly the way green_bright() work with different color.

static blue (message)

This method works exactly the way green_bright() work with different color.

static blue_bright (message)

This method works exactly the way green_bright() work with different color.

static blue_dim(message)

This method works exactly the way green_bright() work with different color.

static cyan (message)

This method works exactly the way green_bright() work with different color.

static cyan bright(message)

This method works exactly the way green_bright() work with different color.

static cyan_dim(message)

This method works exactly the way green_bright() work with different color.

static debug(message)

Print a debug message.

The debug message is a regular message prefixed by INFO in blue on a green background.

Parameters message (str) – The message to print.

Example:

```
base.Text.debug("This is probably going to success, eventually...")
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

static fatal(message)

Print a fatal message.

The fatal message is a regular message prefixed by FATAL in white on a red background.

Parameters message (str) – The message to print.

Example:

```
base.Text.fatal("|x_x|")
```

fg color

The fg_color attribute sets the foreground color. It needs to be a Color.

New in version 1.3.0.

When the foreground color is changed, the observers are notified of the change with the pygamelib.base.Text.fg_color:changed event. The new color is passed as the *value* parameter.

static green (message)

This method works exactly the way green bright() work with different color.

static green_bright(message)

Return a string formatted to be bright green

Parameters message (str) – The message to format.

Returns The formatted string

Return type str

Example:

```
print( Text.green_bright("This is a formatted message") )
```

static green_dim(message)

This method works exactly the way green bright() work with different color.

handle_notification (target, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PqlBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

static info(message)

Print an informative message.

The info is a regular message prefixed by INFO in white on a blue background.

Parameters message (str) – The message to print.

Example:

```
base.Text.info("This is a very informative message.")
```

length

Return the true length of the text.

New in version 1.3.0.

With UTF8 and emojis the length of a string as returned by python's len() function is often very wrong. For example, the len("x1b[48;2;139;22;19mx1b[38;2;160;26;23mx1b[0m") returns 39 when it should return 1.

This method returns the actual printing/display size of the text.

Note: This is a read only value. It is automatically updated when the text property is changed.

Example:

```
game.screen.place(my_text, 0, game.screen.width - my_text.length)
```

classmethod load(data: dict = None)

Load data and create a new Text object out of it.

New in version 1.3.0.

```
Parameters data (dict) - Data to create a new actuator (usually generated by serialize())
```

Returns A new Text object.

Return type Text

Example:

```
title = base.Text.load( previous_title.serialize() )
```

static magenta(message)

This method works exactly the way green_bright() work with different color.

static magenta_bright (message)

This method works exactly the way green_bright() work with different color.

static magenta_dim(message)

This method works exactly the way green_bright() work with different color.

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

parent = None

This object's parent. It needs to be a BoardItem.

print formatted()

Print the text with the current font activated.

New in version 1.3.0.

If the font is not set, it is strictly equivalent to use Python's print(text_object).

static print_white_on_red(message)

Print a white message over a red background.

Parameters message (str) – The message to print.

Example:

```
base.Text.print_white_on_red("This is bright!")
```

static red(message)

This method works exactly the way green_bright() work with different color.

static red_bright(message)

This method works exactly the way green_bright() work with different color.

static red_dim(message)

This method works exactly the way green_bright() work with different color.

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render the Text object from the display buffer to the frame buffer.

New in version 1.3.0.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Return a dictionary with all the attributes of this object.

New in version 1.3.0.

Returns A dictionary with all the attributes of this object.

Return type dict

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

• row (int) – The row (or y) coordinate.

• **column** (*int*) – The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

style = None

The style attribute sets the style of the text. It needs to be a str.

text

The text attribute. It needs to be a str.

New in version 1.3.0.

When the text is changed, the observers are notified of the change with the pygamelib.base. Text.text:changed event. The new text is passed as the *value* parameter.

static warn(message)

Print a warning message.

The warning is a regular message prefixed by WARNING in black on a yellow background.

Parameters message (str) – The message to print.

Example:

```
base.Text.warn("This is a warning.")
```

static white (message)

This method works exactly the way green_bright() work with different color.

static white_bright(message)

This method works exactly the way green_bright() work with different color.

static white_dim(message)

This method works exactly the way green_bright() work with different color.

static yellow(message)

This method works exactly the way green_bright() work with different color.

static yellow_bright(message)

This method works exactly the way green_bright() work with different color.

static yellow_dim(message)

This method works exactly the way green_bright() work with different color.

3.3.11 Vector2D

```
class pygamelib.base.Vector2D(row=0.0, column=0.0)
Bases: object
```

A 2D vector class.

New in version 1.2.0.

Contrary to the rest of the library Vector2D uses floating point numbers for its coordinates/direction/orientation. However since the rest of the library uses integers, the numbers are rounded to 2 decimals. You can alter that behavior by increasing or decreasing the rounding_precision parameter (if you want integer for example).

Vector2D use the row/column internal naming convention as it is easier to visualize for developers that are still learning python or the pygamelib. If it is a concept that you already understand and are more familiar with the x/y coordinate system you can also use x and y.

- x is equivalent to column
- · y is equivalent to row

Everything else is the same.

Vectors can be printed and supports basic operations:

- · addition
- substraction
- · multiplication

Let's elaborate a bit more on the multiplication. The product behaves in 2 different ways:

If you multiply a vector with a scalar (int or float), the return value is a Vector2D with each vector component multiplied by said scalar.

If you multiply a Vector2D with another Vector2D you ask for the the cross product of vectors. This is an undefined mathematical operation in 2D as the cross product is supposed to be perpendicular to the 2 other vectors (along the z axis in our case). Since we don't have depth (z) in 2D, this will return the magnitude of the signed cross product of the 2 vectors.

Example of products:

```
v1 = base.Vector2D(1,2)
v2 = base.Vector2D(3,4)
# This returns -2
mag = v1 * v2
# This returns a Vector2D with values (-1, -2)
inv = v1 * -1
# This return a Vector2D with values (2.85, 3.8) or 95% of v2
dim = v2 * 0.95
```

Parameters

- row (int) The row/y parameter.
- **column** (*int*) The column/x parameter.

Example:

```
gravity = Vector2D(9.81, 0)
# Remember that minus on row is up.
speed = Vector2D(-0.123, 0.456)
# In that case you might want to increase the rounding precision
speed.rounding_precision = 3
```

```
__init___(row=0.0, column=0.0)
```

Initialize self. See help(type(self)) for accurate signature.

Methods

init([row, column])	Initialize self.
from_direction(direction, step)	Build and return a Vector2D from a direction.
length()	Returns the length of a vector.
load(data)	Loads a vector from a dictionary.

Continued on next page

Table 30 – continued from previous page

serialize()	Returns a dictionary with the attributes of the vector.
unit()	Returns a normalized unit vector.

Attributes

column	The column component of the vector.
row	The row component of the vector.
X	x is an alias for column.
y	y is an alias for row.

column

The column component of the vector.

classmethod from_direction(direction, step)

Build and return a Vector2D from a direction.

Directions are from the constants module.

Parameters

- **direction** (*int*) A direction from the constants module.
- **step** (int) The number of cell to cross in one movement.

Example:

```
v2d_up = Vector2D.from_direction(constants.UP, 1)
```

length()

Returns the length of a vector.

Return type float

Example:

```
if speed.length() == 0.0:
    print('We are not moving... at all...')
```

classmethod load(data)

Loads a vector from a dictionary.

New in version 1.3.0.

Parameters data (dict) – A dictionary with the attributes of the vector.

Returns A vector.

Return type Vector2D

Example:

```
gravity_dict = {"row": 9.81, "column": 0}
gravity = Vector2D.load(gravity_dict)
```

rounding_precision = None

The rounding_precision attributes is used when vectors values are calculated and the result rounded for convenience. It can be changed anytime to increase or decrease the precision anytime.

row

The row component of the vector.

serialize()

Returns a dictionary with the attributes of the vector.

New in version 1.3.0.

Returns A dictionary with the attributes of the vector.

Return type dict

Example:

```
gravity = Vector2D(9.81, 0)
gravity_dict = gravity.serialize()
print(gravity_dict)
```

unit()

Returns a normalized unit vector.

Returns A unit vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

x

x is an alias for column.

У

y is an alias for row.

3.3.12 Deprecated objects

These are the deprecated objects of the pygamelib.base module. They should not be used as they are going to be removed in future versions.

HacException

```
exception pygamelib.base.HacException(error, message)
```

A simple forward to PglException

Deprecated since version 1.3.0.

HacInvalidLevelException

exception pygamelib.base.HacInvalidLevelException(message)

Forward to PglInvalidLevelException

Deprecated since version 1.3.0.

HacInvalidTypeException

```
exception pygamelib.base.HacInvalidTypeException(message)
```

A simple forward to PglInvalidTypeException

Deprecated since version 1.3.0.

HacObjectIsNotMovableException

```
exception pygamelib.base.HacObjectIsNotMovableException (message)
```

Simple forward to PglObjectIsNotMovableException

Deprecated since version 1.3.0.

HacOutOfBoardBoundException

```
\textbf{exception} \  \, \texttt{pygamelib.base.} \textbf{HacOutOfBoardBoundException} \, (\textit{message})
```

Simple forward to PglOutOfBoardBoundException

Deprecated since version 1.3.0.

3.4 board_items

3.4.1 Actionable

Bases: pygamelib.board items. Immovable

This class derives *Immovable*. It adds the ability to an Immovable BoardItem to be triggered and execute some code.

If an actionable board item is activated by an item (this mechanism is taken care of by the Board class), the function passed as the *action* parameter is called with *action_parameters* as parameters. Subclass may implement a different mechanism for activation so please read their documentations.

Parameters

- action (function) the reference to a function (Attention: no parentheses at the end of the function name). It needs to be callable.
- $action_parameters(list)$ the parameters to the action function.
- perm(constants) The permission that defines what types of items can actually activate the actionable. The permission has to be one of the permissions defined in constants. By default it is set to constants.PLAYER_AUTHORIZED.

On top of these parameters Actionable accepts all parameters from *Immovable* and therefor from *BoardItem*.

Note: The common way to use this class is to use GenericActionableStructure. Please refer to GenericActionableStructure for more details.

Important: There's a complete tutorial about Actionable items on the pygamelib wiki

__init__ (action=None, action_parameters=None, perm=None, **kwargs)
Like the object class, this class constructor takes no parameter.

Methods

init([action, action_parameters, perm])	Like the object class, this class constructor takes no parameter.		
activate()	This function is calling the action function with the action_parameters.		
attach(observer)	Attach an observer to this instance.		
can_move()	Return the capability of moving of an item.		
collides_with(other, projection_offset)	Tells if this item collides with another item.		
debug_info()	Return a string with the list of the attributes and their		
	current value.		
detach(observer)	Detach an observer from this instance.		
display()	Print the model WITHOUT carriage return.		
distance_to(other)	Calculates the distance with an item.		
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by		
value])	the observer.		
load(data)	Load data and create a new BoardItem out of it.		
notify([modifier])	Notify all the observers that a change occurred.		
overlappable()	Returns True if the item is overlappable, False other-		
	wise.		
pickable()	Returns True if the item is pickable, False otherwise.		
position_as_vector()	Returns the current item position as a Vector2D		
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a screen buffer).		
restorable()	Returns True if the item is restorable, False other-		
V	wise.		
serialize()	Return a dictionary with all the attributes of this ob-		
·	ject.		
set_can_move(value)	Set the value of the can_move property to value.		
set_overlappable(value)	Set the value of the overlappable property to value.		
set_pickable(value)	Set the value of the pickable property to value.		
set_restorable(value)	Set the value of the restorable property to value.		
store_position(row, column, layer)	Store the BoardItem position for self access.		
store_screen_position(row, column)	Store the screen position of the object.		

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
	Continued on next page

3.4. board_items 207

Table	33 –	continued	from	previous	page
-------	------	-----------	------	----------	------

	- commerce work browners bridge
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

activate()

This function is calling the action function with the action_parameters.

The action callback function should therefor have a signature like:

```
def my_callback_function(actionable, action_parameters)
```

With actionable being the Actionable current reference to self.

Usually it's automatically called by move () when a Player or NPC (see board_items)

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)
Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

3.4. board items 209

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*Pg1BaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the Inventory.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

3.4. board items 211

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
    print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\textbf{serialize}\,(\,)\,\to dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable (value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.2 ActionableTile

```
class pygamelib.board_items.ActionableTile(**kwargs)
    Bases: pygamelib.board_items.Actionable, pygamelib.board_items.Tile
```

The ActionableTile is the complex (i.e: multi-cells items) version of the <code>GenericActionableStructure</code>. It allows you to create any type of in game object that is represented with more than one character in the terminal and that is <code>Actionable</code>. Actionable object have a callback system that is automatically called when the player collide with the object.

Important: There's a complete tutorial about Actionable items on the pygamelib wiki

___init___(**kwargs)

Please have a look at the documentation for *Tile* and *Actionable* for the list of possible constructor's parameters.

Methods

init(**kwargs)	Please have a look at the documentation for Tile
	and Actionable for the list of possible construc-
	tor's parameters.
activate()	This function is calling the action function with the
	action_parameters.
attach(observer)	Attach an observer to this instance.
can_move()	A Tile cannot move.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new Tile out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
<pre>position_as_vector()</pre>	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

activate()

This function is calling the action function with the action_parameters.

The action callback function should therefor have a signature like:

```
def my_callback_function(actionable, action_parameters)
```

With actionable being the Actionable current reference to self.

Usually it's automatically called by move () when a Player or NPC (see board_items)

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
```

(continues on next page)

(continued from previous page)

```
# screen will be notified of all changes in myboard myboard.attach(screen)
```

can_move()

A Tile cannot move.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pg1BaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification(subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

item(row, column)

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises PglOutOfBoardBoundException – if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new Tile out of it.

Parameters data (dict) – Data to create a new tile (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.Tile

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

• modifier (PglBaseObject) – An optional parameter that identify the modifier object to exclude it from the notified objects.

- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

$store_position$ (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the Board. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position(row: int, column: int) \rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.3 BoardComplexItem

 $Bases: \ pygamelib.board_items.BoardItem$

New in version 1.2.0.

A BoardComplexItem is the base item for multi cells elements. It inherits from <code>BoardItem</code> and accepts all its parameters.

The main difference is that a complex item can use Sprite as representation.

You can see a complex item as a collection of other items that are ruled by the same laws. They behave as one but a complex item is actually made of complex components. At first it is not important but you may want to exploit that as a feature for your game.

On top of *BoardItem* the constructor accepts the following parameters:

Parameters

- **sprite** (*Sprite*) A sprite representing the item.
- **size** (array [int]) The size of the item as [WIDTH, HEIGHT]. It impact movement and collision detection amongst other things. If it is left empty the Sprite size is used. If no sprite is given to the constructor the default size is 2x2.
- base_item_type (BoardItemComplexComponent) the building block of the complex item. The complex item is built from a 2D array of base items.

Null_sprixel The null_sprixel is a bit of a special parameter: during construction a null sprixel is replaced by a BoardItemVoid. This is a trick to show the background (i.e transparency). A sprixel can take the color of the background but a complex item with a null_sprixel that correspond to transparent zone of a sprite will really be transparent and show the background.

Null_sprixel Sprixel

__init__ (sprite=None, size=None, null_sprixel=None, base_item_type=None, **kwargs)
Like the object class, this class constructor takes no parameter.

Methods

init([sprite, size, null_sprixel,])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Returns True if the item can move, False otherwise.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
	Continued on next page

224 Chapter 3. Tutorials

Table 36 – continued from previous page

table 30 continued from previous page	
load(data)	Load data and create a new BoardComplexItem out
	of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can move()

Returns True if the item can move, False otherwise.

Example:

```
if board.item(4,5).can_move():
    print('The item can move')
```

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

item(row, column)

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises Pq10utOfBoardBoundException - if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load (data)

Load data and create a new BoardComplexItem out of it.

Parameters data (dict) - Data to create a new complex item (usually generated by serialize())

Returns A new complex item.

Return type ~pygamelib.board_items.BoardComplexItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
    print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

```
render_to_buffer (buffer, row, column, height, width)
```

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

```
Parameters value (int) – the screen column
```

Return type int

screen_row

A property to get/set the screen row.

```
Parameters value (int) – the screen row
```

Return type int

```
serialize() \rightarrow dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

```
set can move (value)
```

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable (value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

(continues on next page)

(continued from previous page)

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
```

(continues on next page)

(continued from previous page)

```
item.update_sprite()
board.move(item, constants.RIGHT, 1)
time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.4 BoardItemComplexComponent

```
class pygamelib.board_items.BoardItemComplexComponent(**kwargs)
```

Bases: pygamelib.board_items.BoardItem

The default component of a complex item.

It is literally just a BoardItem but is subclassed for easier identification.

It is however scanning its parent for the item's basic properties (overlappable, restorable, etc.)

A component can never be pickable by itself.

```
___init___(**kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Returns True if the item can move, False otherwise.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns False.
	0 11 1

Continued on next page

Table 38 – continued from previous page

position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

234 Chapter 3. Tutorials

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Returns True if the item can move, False otherwise.

Example:

```
if board.item(4,5).can_move():
    print('The item can move')
```

 $collides_with (other, projection_offset: pygamelib.base.Vector2D = None)$

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.

• **value** (Any) – An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

Returns False. A component is never pickable by itself (either the whole complex item is pickable or not, but not partially)

Example:

```
if item.item(4,5).pickable():
    print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column(int) The column (or x) coordinate.

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width **Return type** int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.5 BoardItem

Bases: pygamelib.base.PglBaseObject

Base class for any item that will be placed on a Board.

Parameters

- **type** (str) A type you want to give your item. It can be any string. You can then use the type for sorting or grouping for example.
- name (str) A name for this item. For identification purpose.
- **pos** (list) the position of this item. When the item is managed by the Board and Game engine this member hold the last updated position of the item. It is not updated if you manually move the item. It must be an array of 2 integers [row,column]
- model (str) The model to use to display this item on the Board. Be mindful of the space it will require. Default value is '*'. This parameter is now deprecated in favor of "sprixel". If both "sprixel" and "model" are specified, "model" is ignored.
- parent The parent object of the board item. Usually a Board or Game object.
- **sprixel** (Sprixel) The sprixel that will represent the item on the Board.
- pickable (bool) Represent the capacity for a BoardItem to be pick-up by player or NPC. This parameter is True or False. If sets to None, it'll be set to False.
- **overlappable** (bool) Represent to be overlapped by another BoardItem. This parameter is True or False. If sets to None, it'll be set to False.
- **restorable** (bool) Represent the capacity for an Immovable BoardItem to be restored by the board if the item is overlappable and has been overlapped by another BoardItem. This parameter is True or False. If sets to None, it'll be set to False.
- can_move (bool) Represent the ability of the BoardItem to move on the Board. If this parameter is False, the Board.move() method will not allow the item to move. This parameter is True or False. If sets to None, it'll be set to False.
- **pos** The position of the BoardItem on a *Board*. Please make sure that you understand what you do before changing that parameter. The position of an item is managed by the

Board object and will be updated. In most cases you don't need to use that parameter. The position is a list of 2 or 3 int: [row, column, layer].

- **value** (*int* / *float*) The value of an item. It can be used for any game purpose: a score indicator, a trade value, the amount of XP to grant to a player on a kill, etc.
- **inventory_space** (*int*) The space that the item takes in the *pygamelib*. *engine*. *Inventory*. This parameter used to be available only for *Immovable* items but since 1.3.0, every BoardItem can be configured to be pickable, so every BoardItem can now take space in the inventory. Default value is 1.
- animation (Animation) An animation to animate the item sprixel.
- particle_emitter (ParticleEmitter) A particle emitter that is attached to this item.

Note: Starting with version 1.2.0 and introduction of complex items, BoardItems have a size. That size **CANNOT** be set. It is always 1x1. This is because a BoardItem always takes 1 cell, regardless of its actual number of characters. The size is a read-only property.

Important: In version 1.3.0 the BoardItem object has been reworked to make sure that the pickable, restorable, overlappable and can_move properties are configurable for all items independently of their type. This fixes an issue with restorable: only *Immovable* objects could be restorable. Now all items can be any combination of these properties. As a developer you are now encouraged to use the corresponding functions to determine the abilities of an item.

Warning: An item cannot be restorable and pickable at the same time. If it's pickable, it's put into the inventory of the item overlapping it. Therefor, it cannot be restored. If both restorable and pickable are set to True, one of the 2 is set to False depending on the value of overlappable: if True restorable is set to True and pickable to False and the contrary if overlappable is False.

__init__ (sprixel=None, model=None, name=None, item_type=None, parent=None, pickable=False, overlappable=False, restorable=False, can_move=False, pos=None, value=None, inventory_space=1, animation: pygamelib.gfx.core.Animation = None, particle_emitter=None)
Like the object class, this class constructor takes no parameter.

Methods

init([sprixel, model, name, item_type,])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Returns True if the item can move, False otherwise.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.

Continued on next page

242 Chapter 3. Tutorials

Table 40 – continued from previous page

A virtual method that needs to be implemented by
the observer.
Load data and create a new BoardItem out of it.
Notify all the observers that a change occurred.
Returns True if the item is overlappable, False other-
wise.
Returns True if the item is pickable, False otherwise.
Returns the current item position as a Vector2D
Render the board item into a display buffer (not a
screen buffer).
Returns True if the item is restorable, False other-
wise.
Return a dictionary with all the attributes of this ob-
ject.
Set the value of the can_move property to value.
Set the value of the overlappable property to value.
Set the value of the pickable property to value.
Set the value of the restorable property to value.
Set the value of the restorable property to value. Store the BoardItem position for self access.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a

notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) - An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Returns True if the item can move, False otherwise.

Example:

```
if board.item(4,5).can_move():
    print('The item can move')
```

 $collides_with (other, projection_offset: pygamelib.base.Vector2D = None)$

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.

- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

$\verb"set_overlappable" (value)$

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

```
# This is a silly example because the Board object does not allow
# that use case.
if item.column + item.size[0] >= board.width:
    Game.instance().screen.display_line(
        f"{item.name} cannot be placed at {item.pos}."
    )
```

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.6 BoardItemVoid

```
class pygamelib.board_items.BoardItemVoid(**kwargs)
```

Bases: pygamelib.board_items.BoardItem

A class that represent a void cell.

```
___init___(**kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Returns True if the item can move, False otherwise.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	A BoardItemVoid is obviously overlappable (so
	player and NPC can walk over).
pickable()	A BoardItemVoid is not pickable, therefor this
	method return false.
<pre>position_as_vector()</pre>	Returns the current item position as a Vector2D
<pre>render_to_buffer(buffer, row, column,)</pre>	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
	Continued on next page

250 Chapter 3. Tutorials

Table 42 – continued from previous page

	<u> </u>
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
```

(continues on next page)

(continued from previous page)

```
# screen will be notified of all changes in myboard myboard.attach(screen)
```

can_move()

Returns True if the item can move, False otherwise.

Example:

```
if board.item(4,5).can_move():
    print('The item can move')
```

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PqlBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) - The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (PqlBaseObject) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

A BoardItemVoid is obviously overlappable (so player and NPC can walk over).

Returns True

particle emitter

pickable()

A BoardItemVoid is not pickable, therefor this method return false.

Returns False

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

256 Chapter 3. Tutorials

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position(row: int, column: int) \rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.7 Camera

```
class pygamelib.board_items.Camera(actuator=None, **kwargs)
    Bases: pygamelib.board_items.Movable
```

New in version 1.3.0.

A Camera is a special item: it does not appear on the Board and actually is not even registered on it. It is only an item that you can center the board on (when using partial display). It helps for cut scenes for example.

The main difference with a regular BoardItem is that the row and column properties are writable. This means that you can directly manipulate its coordinates and partially render a huge board around that focal point.

The *Screen* buffer rendering system introduced in version 1.3.0 require a board item to be declared as the focus point of the board if partial display is enabled.

The Camera object inherits from Movable and can accept an actuator parameter. However, it is up to the developer to activate the actuators mechanics as the Camera object does not register as a NPC or a Player. The support for actuators is mainly thought for pre-scripted cut-scenes.

Example:

```
# This example leverage the Screen buffer system introduced in v1.3.0.
# It pans the camera over a huge map. The Screen.update() method automatically
# uses the Board.partial_display_focus coordinates to adjust the displayed area.
camera = Camera()
huge_board.partial_display_focus = camera
while camera.column < huge_board.width:
    camera.column += 1
    game.screen.update()</pre>
```

___init___(actuator=None, **kwargs)

Like the object class, this class constructor takes no parameter.

Methods

Like the object class, this class constructor takes no
parameter.
Attach an observer to this instance.
Movable implements can_move().
Tells if this item collides with another item.
Return a string with the list of the attributes and their
current value.
Detach an observer from this instance.
Print the model WITHOUT carriage return.
Calculates the distance with an item.
A virtual method that needs to be implemented by
the observer.
This is a virtual method that must be implemented in
deriving class.
Load data and create a new Movable out of it.
Notify all the observers that a change occurred.
Returns True if the item is overlappable, False other-
wise.
Returns True if the item is pickable, False otherwise.

258 Chapter 3. Tutorials

Table 44 – continued from previous page

	, , ,
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Serialize the Immovable object.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.
	·

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

$\mathtt{can_move}() \rightarrow bool$

Movable implements can_move().

Returns True

Return type Boolean

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PqlBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

${\tt has_inventory}\,(\,)\,\to bool$

This is a virtual method that must be implemented in deriving class. This method has to return True or False. This represent the capacity for a Movable to have an inventory.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new Movable out of it.

Parameters data (dict) - Data to create a new movable item (usually generated by serialize())

Returns A new complex item.

Return type ~pygamelib.board_items.Movable

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
    print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (*int*) – the screen row

Return type int

$serialize() \rightarrow dict$

Serialize the Immovable object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

```
set can move (value)
```

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column(int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width **Return type** int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.8 Character

```
class pygamelib.board_items.Character(max_hp=None, hp=None, max_mp=None, mp=None, mp=Non
```

Bases: pygamelib.board_items.Movable

A base class for a character (playable or not)

Parameters

- agility (int) Represent the agility of the character
- **attack_power** (*int*) Represent the attack power of the character.
- **defense_power** (*int*) Represent the defense_power of the character
- **hp** (*int*) Represent the hp (Health Point) of the character
- intelligence (int) Represent the intelligence of the character
- max_hp (int) Represent the max_hp of the character
- $max_mp(int)$ Represent the max_mp of the character
- mp (int) Represent the mp (Mana/Magic Point) of the character
- **remaining_lives** (*int*) Represent the remaining_lives of the character. For a NPC it is generally a good idea to set that to 1. Unless the NPC is a multi phased boss.
- **strength** (*int*) Represent the strength of the character

These characteristics are here to be used by the game logic but very few of them are actually used by the Game (*pygamelib.engine*) engine.

```
__init__ (max_hp=None, hp=None, max_mp=None, mp=None, remaining_lives=None, at-
tack_power=None, defense_power=None, strength=None, intelligence=None,
agility=None, **kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

266

init([max_hp, hp, max_mp, mp,])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
	Continued on next page

Table 46 – continued from previous page

can_move()	Movable implements can_move().
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
has_inventory()	This is a virtual method that must be implemented in
	deriving class.
load(data)	Load data and create a new Character out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Serialize the Character object.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
	Continued on next page

Table 47 – continued from previous page

width

Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

${\tt can_move}\,(\,)\,\to bool$

Movable implements can move().

Returns True

Return type Boolean

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

268 Chapter 3. Tutorials

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

• **subject** (*PqlBaseObject*) – The object that has changed.

- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

$has_inventory() \rightarrow bool$

This is a virtual method that must be implemented in deriving class. This method has to return True or False. This represent the capacity for a Movable to have an inventory.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the Inventory.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new Character out of it.

Parameters data (dict) - Data to create a new character item (usually generated by serialize())

Returns A new character item.

Return type ~pygamelib.board_items.Character

model

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (*int*) – the screen row

Return type int

$serialize() \rightarrow dict$

Serialize the Character object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable (value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable (value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.9 ComplexDoor

```
class pygamelib.board_items.ComplexDoor(**kwargs)
```

Bases: pygamelib.board_items.Door, pygamelib.board_items.BoardComplexItem

New in version 1.2.0.

A complex door is nothing more than a Door mashed with a BoardComplexItem.

It supports all parameters of both with inheritance going first to Door and second to BoardComplexItem.

The main interest is of course the multiple cell representation and the Sprites support.

Example:

```
__init___(**kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.

Continued on next page

Table 48 – continued from previous page

can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new ComplexDoor out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
<pre>render_to_buffer(buffer, row, column,)</pre>	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
Beore Boreen posteron (10w, column)	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
	Continued on next page

Table 49 – continued from previous page

	<u> </u>
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) - An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PqlBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the Inventory.

Returns The size of the item.

Return type int

item(row, column)

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises PglOutOfBoardBoundException – if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new ComplexDoor out of it.

Parameters data (dict) - Data to create a new complex door (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.ComplexDoor

model

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

```
position_as_vector()
```

Returns the current item position as a Vector2D

Returns The position as a 2D vector

```
Return type Vector2D
```

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set can move (value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column(int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.10 ComplexNPC

```
class pygamelib.board_items.ComplexNPC(**kwargs)
```

Bases: pygamelib.board_items.NPC, pygamelib.board_items.BoardComplexItem

New in version 1.2.0.

A complex NPC is nothing more than a NPC mashed with a BoardComplexItem.

It supports all parameters of both with inheritance going first to NPC and second to BoardComplexItem.

The main interest is of course the multiple cell representation and the Sprites support.

Example:

```
___init___(**kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Movable implements can_move().
collides_with(other, projection_offset)	Tells if this item collides with another item.

Continued on next page

Table 50 – continued from previous page

Table 30 - Continued	a irom previous page
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
has_inventory()	Define if the NPC has an inventory.
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new ComplexNPC out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Define if the NPC is pickable.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Serialize the NPC object.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
	Continued on next page

284 Chapter 3. Tutorials

Table 51 – continued from previous page

sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

$\mathtt{can_move} () \rightarrow bool$

Movable implements can_move().

Returns True

Return type Boolean

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

has_inventory()

Define if the NPC has an inventory.

This method returns false because the game engine doesn't manage NPC inventory yet but it could be in the future. It's a good habit to check the value returned by this function.

Returns False

Return type Boolean

Example:

```
if mynpc.has_inventory():
    print("Cool: we can pickpocket that NPC!")
else:
    print("No pickpocketing XP for us today :(")
```

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

item(row, column)

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises PglOutOfBoardBoundException – if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new ComplexNPC out of it.

Parameters data (dict) - Data to create a new complex npc (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.ComplexNPC

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Define if the NPC is pickable.

Obviously this method always return False.

Returns False

Return type Boolean

Example:

```
if mynpc.pickable():
    Utils.warn("Something is fishy, that NPC is pickable"
        "but is not a Pokemon...")
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- **row** (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Serialize the NPC object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$\verb|store_screen_position|| (\textit{row: int, column: int})| \rightarrow bool|$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.11 ComplexPlayer

```
class pygamelib.board_items.ComplexPlayer(**kwargs)
```

Bases: pygamelib.board_items.Player, pygamelib.board_items.BoardComplexItem

New in version 1.2.0.

A complex player is nothing more than a Player mashed with a BoardComplexItem.

It supports all parameters of both with inheritance going first to Player and second to BoardComplexItem.

The main interest is of course the multiple cell representation and the Sprites support.

Example:

```
__init__(**kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Movable implements can_move().
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
has_inventory()	This method returns True (a player has an inventory).
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new ComplexPlayer out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	This method returns False (a player is obviously not
	pickable).
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Serialize the Character object.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

$\mathtt{can_move} () \rightarrow bool$

Movable implements can_move().

Returns True

Return type Boolean

 $\verb|collides_with| (other, projection_offset: pygamelib.base. Vector 2D = None)|$

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

has_inventory()

This method returns True (a player has an inventory).

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

item(row, column)

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises Pg1OutOfBoardBoundException - if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new ComplexPlayer out of it.

Parameters data (dict) - Data to create a new complex player (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.ComplexPlayer

mode1

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

This method returns False (a player is obviously not pickable).

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
    print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Serialize the Character object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

$\mathtt{set_overlappable}$ (value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the Board. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.12 ComplexTreasure

```
class pygamelib.board_items.ComplexTreasure(**kwargs)
```

Bases: pygamelib.board_items.Treasure, pygamelib.board_items.BoardComplexItem

New in version 1.2.0.

A complex treasure is nothing more than a Treasure mashed with a BoardComplexItem.

It supports all parameters of both with inheritance going first to Treasure and second to BoardComplexItem.

The main interest is of course the multiple cell representation and the Sprites support.

Example:

___init___(**kwargs)

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new ComplexTreasure out of
	it.
<pre>notify([modifier])</pre>	Notify all the observers that a change occurred.
overlappable()	This represent the capacity for a Treasure to be over-
	lapped by player or NPC.
pickable()	This represent the capacity for a Treasure to be
	picked-up by player or NPC.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	This represent the capacity for a Treasure to be re-
	stored after being overlapped.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
	Continued on next page

302 Chapter 3. Tutorials

Table 55 – continued from previous page

inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

 $\verb|collides_with| (other, projection_offset: pygamelib.base.Vector2D = None)|$

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on

different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (Vector2D) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

item(row, column)

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises Pq10utOfBoardBoundException - if row or column are out of bound.

laver

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new ComplexTreasure out of it.

Parameters data (dict) - Data to create a new complex treasure (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.ComplexTreasure

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

This represent the capacity for a Treasure to be overlapped by player or NPC.

A treasure is not overlappable.

Returns False

Return type bool

particle_emitter

pickable()

This represent the capacity for a Treasure to be picked-up by player or NPC.

A treasure is obviously pickable by the player and potentially NPCs. Board puts the Treasure in the Inventory if the picker implements has_inventory()

Returns True

Return type bool

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

This represent the capacity for a Treasure to be restored after being overlapped.

A treasure is not overlappable, therefor is not restorable.

Returns False

Return type bool

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\textbf{serialize}\,(\,)\,\to dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable (value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

308 Chapter 3. Tutorials

Example:

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.13 ComplexWall

```
class pygamelib.board_items.ComplexWall(**kwargs)
```

Bases: pygamelib.board_items.Wall, pygamelib.board_items.BoardComplexItem

New in version 1.2.0.

A complex wall is nothing more than a Wall mashed with a BoardComplexItem.

It supports all parameters of both with inheritance going first to Wall and second to BoardComplexItem.

The main interest is of course the multiple cell representation and the Sprites support.

Example:

```
__init___(**kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new ComplexWall out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	This represent the capacity for a BoardItem to be
	overlapped by player or NPC.
pickable()	This represent the capacity for a BoardItem to be
	pick-up by player or NPC.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	This represent the capacity for an Immovable
	Movable item.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
<pre>store_position(row, column, layer)</pre>	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	

Continued on next page

T			•		
Ianie	り /	- continued	trom	nrevinis	nage
IUDIC	\circ	COLLULIACA	11 0111	picvious	page

particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

${\tt can_move}\,(\,)$

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

 $collides_with (other, projection_offset: pygamelib.base.Vector2D = None)$

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

• other (BoardItem) - The item you want to check for collision.

• **projection_offset** (Vector2D) – A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (*Pg1BaseObject*) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the Inventory.

Returns The size of the item.

Return type int

```
item(row, column)
```

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises Pq10utOfBoardBoundException - if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new ComplexWall out of it.

Parameters data (dict) - Data to create a new complex wall item (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.ComplexWall

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

This represent the capacity for a *BoardItem* to be overlapped by player or NPC.

Returns False

Return type bool

particle_emitter

pickable()

This represent the capacity for a *BoardItem* to be pick-up by player or NPC.

Returns False

Return type bool

Example:

```
if mywall.pickable():
    print('Whoaa this wall is really light... and small...')
else:
    print('Really? Trying to pick-up a wall?')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- **row** (*int*) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

This represent the capacity for an *Immovable* Movable item. A wall is not overlappable.

Returns False

Return type bool

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\texttt{serialize}\,(\,)\,\to dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.14 Door

```
class pygamelib.board_items.Door(**kwargs)
    Bases: pygamelib.board_items.GenericStructure
```

A Door is a *GenericStructure* that is not pickable, overlappable and restorable. It has a value of 0 and a size of 1 by default. It is an helper class that allows to focus on game design and mechanics instead of small building blocks.

Parameters

- model(str) The model that will represent the door on the map
- value (int) The value of the door, it is useless in that case. The default value is 0.
- **inventory_space** (*int*) The size of the door in the inventory. Unless you make the door pickable (I have no idea why you would do that...), this parameter is not used.
- **type** (*str*) The type of the door. It is often used as a type identifier for your game main loop. For example: unlocked door or locked door.
- pickable (Boolean) Is this door pickable by the player? Default value is False.

- **overlappable** (Boolean) Is this door overlappable by the player? Default value is True.
- **restorable** (Boolean) Is this door restorable after being overlapped? Default value is True.

Note: All the options from *GenericStructure* are also available to this constructor.

Example:

```
door1 = Door(model=graphics.Models.DOOR,type='locked_door')
```

__init__(**kwargs)

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

320 Chapter 3. Tutorials

	A property to get and get on The installant for this
animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

 $\verb|collides_with| (other, projection_offset: pygamelib.base.Vector2D = None)|$

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (PqlBaseObject) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load (data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

```
Return type Vector2D
```

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- **row** (*int*) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
    print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

```
# This is a silly example because the Board object does not allow
# that use case.
if item.column + item.size[0] >= board.width:
```

(continues on next page)

(continued from previous page)

```
Game.instance().screen.display_line(
    f"{item.name} cannot be placed at {item.pos}."
)
```

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

 $store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.15 GenericActionableStructure

```
class pygamelib.board_items.GenericActionableStructure(**kwargs)
    Bases: pygamelib.board_items.GenericStructure, pygamelib.board_items.
    Actionable
```

A GenericActionableStructure is the combination of a GenericStructure and an Actionable. It is only a helper combination.

Please see the documentation for GenericStructure and Actionable for more information.

Important: There's a complete tutorial about Actionable items on the pygamelib wiki

___init___(**kwargs)

Like the object class, this class constructor takes no parameter.

Methods

activate() Activa	Darameter. This function is calling the action function with the action_parameters. Attach an observer to this instance. Return the capability of moving of an item. Tells if this item collides with another item. Return a string with the list of the attributes and their
attach(observer) can_move() collides_with(other, projection_offset) T	Attach an observer to this instance. Return the capability of moving of an item. Fells if this item collides with another item.
attach(observer) A can_move() R collides_with(other, projection_offset) T	Attach an observer to this instance. Return the capability of moving of an item. Fells if this item collides with another item.
can_move() R collides_with(other, projection_offset) T	Return the capability of moving of an item. Tells if this item collides with another item.
collides_with(other, projection_offset) T	Tells if this item collides with another item.
debug info() R	Return a string with the list of the attributes and their
accag_info()	tream a string with the fist of the attributes and then
	current value.
,	Detach an observer from this instance.
	Print the model WITHOUT carriage return.
_ ` '	Calculates the distance with an item.
handle_notification(subject[, attribute, A	A virtual method that needs to be implemented by
value]) th	he observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable() R	Returns True if the item is overlappable, False other-
W	vise.
pickable() R	Returns True if the item is pickable, False otherwise.
	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,) R	Render the board item into a display buffer (not a
	screen buffer).
restorable() R	Returns True if the item is restorable, False other-
W	vise.
serialize() R	Return a dictionary with all the attributes of this ob-
je	ect.
	Set the value of the can_move property to value.
	Set the value of the overlappable property to value.
	Set the value of the pickable property to value.
set_restorable(value) S	Set the value of the restorable property to value.
	Store the BoardItem position for self access.
store_screen_position(row, column) S	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
	Continued on next page

continued on next page

Table	61 ·	continued	from	previous	page
-------	------	-----------------------------	------	----------	------

	John Mada Ham providuo pugo
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

activate()

This function is calling the action function with the action_parameters.

The *action* callback function should therefor have a signature like:

```
def my_callback_function(actionable, action_parameters)
```

With actionable being the Actionable current reference to self.

Usually it's automatically called by move () when a Player or NPC (see board_items)

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)
Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*Pg1BaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

332

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
    print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
    print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\textbf{serialize}\,(\,)\,\to dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable (value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable (value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

334 Chapter 3. Tutorials

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.16 GenericStructureComplexComponent

```
class pygamelib.board_items.GenericStructureComplexComponent(**kwargs)
    Bases: pygamelib.board_items.GenericStructure, pygamelib.board_items.
    BoardItemComplexComponent
```

A ComplexComponent specifically for generic structures.

__init__(**kwargs)

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns False.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
	Continued on next page

336 Chapter 3. Tutorials

Table 63 – continued from previous page

	, , ,
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

• other (BoardItem) - The item you want to check for collision.

• **projection_offset** (Vector2D) – A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (*Pg1BaseObject*) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the Inventory.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

Returns False. A component is never pickable by itself (either the whole complex item is pickable or not, but not partially)

Example:

```
if item.item(4,5).pickable():
    print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
\texttt{serialize}\,(\,)\,\to dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable (value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

```
store_position (row: int, column: int, layer: int = 0)
```

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.17 GenericStructure

```
class pygamelib.board_items.GenericStructure(value=0, **kwargs)
    Bases: pygamelib.board_items.Immovable
```

A GenericStructure is as the name suggest, a generic object to create all kind of structures.

It can be tweaked with all the properties of *BoardItem*, *Immovable* and it can be made pickable, overlappable or restorable or any combination of these.

If you need an action to be done when a Player and/or a NPC touch the structure please have a look at pygamelib.board_items.GenericActionableStructure.

Parameters

• pickable (bool) – Define if the structure can be picked-up by a Player or NPC.

- overlappable (bool) Define if the structure can be overlapped by a Player or NPC.
- **restorable** (bool) Define if the structure can be restored by the Board after a Player or NPC passed through. For example, you want a door or an activator structure (see Generic ActionableStructure for that) to remain on the board after it's been overlapped by a player. But you could also want to develop some kind of Space Invaders game were the protection block are overlappable but not restorable.
- **value** (*int* / *float*) The value of the structure. It can be used for scoring, resource spending, etc.

On top of these, this object takes all parameters of <code>BoardItem</code> and <code>Immovable</code>

Important: If you need a structure with a permission system please have a look at GenericActionableStructure. This class has a permission system for activation.

___init___(value=0, **kwargs)

Like the object class, this class constructor takes no parameter.

Methods

init([value])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

344 Chapter 3. Tutorials

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.
	-

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

```
Return type Vector2D
```

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- **row** (*int*) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
    print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

```
# This is a silly example because the Board object does not allow
# that use case.
if item.column + item.size[0] >= board.width:
```

(continues on next page)

(continued from previous page)

```
Game.instance().screen.display_line(
    f"(item.name) cannot be placed at (item.pos)."
)
```

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

 $store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.18 Immovable

```
class pygamelib.board_items.Immovable(inventory_space: int = None, **kwargs)
    Bases: pygamelib.board_items.BoardItem
```

This class derive <code>BoardItem</code> and describe an object that cannot move or be moved (like a wall). <code>can_move()</code> cannot be configured and return False. The other properties can be configured. They have the same default values than <code>BoardItem</code>.

Parameters inventory_space (*int*) – The space the immovable item takes into an *Inventory* (in case the item is pickable). By default it is 0.

__init__ (*inventory_space*: *int* = *None*, **kwargs)

Like the object class, this class constructor takes no parameter.

Methods

attach(observer) Attach an observer to this instance. can_move() Return the capability of moving of an item. collides_with(other, projection_offset) debug_info() Return a string with the list of the attributes and their current value. detach(observer) Detach an observer from this instance. display() Print the model WITHOUT carriage return. distance_to(other) Andle_notification(subject[, attribute, value]) load(data) Load data and create a new BoardItem out of it. notify([modifier]) Notify all the observers that a change occurred. vise. pickable() Returns True if the item is overlappable, False otherwise. position_as_vector() Returns True if the item is pickable, False otherwise. position_as_vector() Returns True if the item is restorable, False otherwise. ser_alize() Returns True if the item is restorable, False otherwise. set_can_move(value) Set the value of the can_move property to value. set_overlappable(value) Set the value of the overlappable property to value. set_position(row, column, layer) Store the BoardItem position for hel object.	init(inventory_space, **kwargs)	Like the object class, this class constructor takes no
can_move()Return the capability of moving of an item.collides_with(other, projection_offset)Tells if this item collides with another item.debug_info()Return a string with the list of the attributes and their current value.detach(observer)Detach an observer from this instance.display()Print the model WITHOUT carriage return.distance_to(other)Calculates the distance with an item.handle_notification(subject[, attribute, value])A virtual method that needs to be implemented by the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Returns True if the item is restorable, False otherwise.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.		parameter.
collides_with(other, projection_offset)Tells if this item collides with another item.debug_info()Return a string with the list of the attributes and their current value.detach(observer)Detach an observer from this instance.display()Print the model WITHOUT carriage return.distance_to(other)Calculates the distance with an item.handle_notification(subject[, attribute, value])A virtual method that needs to be implemented by the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Returns True if the item is restorable, False otherwise.set_overlappable(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.	attach(observer)	Attach an observer to this instance.
debug_info()Return a string with the list of the attributes and their current value.detach(observer)Detach an observer from this instance.display()Print the model WITHOUT carriage return.distance_to(other)Calculates the distance with an item.handle_notification(subject[, value])attribute, the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Returns True if the item is restorable property to value.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	can_move()	Return the capability of moving of an item.
detach(observer)Detach an observer from this instance.display()Print the model WITHOUT carriage return.distance_to(other)Calculates the distance with an item.handle_notification(subject[, attribute, value])A virtual method that needs to be implemented by the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	collides_with(other, projection_offset)	Tells if this item collides with another item.
detach(observer)Detach an observer from this instance.display()Print the model WITHOUT carriage return.distance_to(other)Calculates the distance with an item.handle_notification(subject[, attribute, value])A virtual method that needs to be implemented by the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	debug_info()	Return a string with the list of the attributes and their
display()Print the model WITHOUT carriage return.distance_to(other)Calculates the distance with an item.handle_notification(subject[, value])attribute, attribute, the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_pickable(value)Set the value of the overlappable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.		
distance_to(other)Calculates the distance with an item.handle_notification(subject[, attribute, value])A virtual method that needs to be implemented by the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	detach(observer)	Detach an observer from this instance.
handle_notification(subject[, value])attribute, the observer.A virtual method that needs to be implemented by the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_overlappable(value)Set the value of the can_move property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	display()	Print the model WITHOUT carriage return.
value])the observer.load(data)Load data and create a new BoardItem out of it.notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	distance_to(other)	Calculates the distance with an item.
Load data and create a new BoardItem out of it. notify([modifier]) Notify all the observers that a change occurred. Notify all the observers that a change occurred. Returns True if the item is overlappable, False otherwise. pickable() Returns True if the item is pickable, False otherwise. position_as_vector() Returns the current item position as a Vector2D render_to_buffer(buffer, row, column,) Render the board item into a display buffer (not a screen buffer). restorable() Returns True if the item is restorable, False otherwise. serialize() Return a dictionary with all the attributes of this object. set_can_move(value) Set the value of the can_move property to value. set_overlappable(value) Set the value of the overlappable property to value. set_restorable(value) Set the value of the restorable property to value. set_restorable(value) Set the value of the restorable property to value. set_restorable(value) Set the value of the restorable property to value. store_position(row, column, layer) Store the BoardItem position for self access.	handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
notify([modifier])Notify all the observers that a change occurred.overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	value])	the observer.
overlappable()Returns True if the item is overlappable, False otherwise.pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	load(data)	Load data and create a new BoardItem out of it.
wise. pickable() Returns True if the item is pickable, False otherwise. Returns the current item position as a Vector2D render_to_buffer(buffer, row, column,) Render the board item into a display buffer (not a screen buffer). Returns True if the item is restorable, False otherwise. Returns True if the item is restorable, False otherwise. Serialize() Return a dictionary with all the attributes of this object. Set_can_move(value) Set the value of the can_move property to value. Set_overlappable(value) Set the value of the pickable property to value. Set_restorable(value) Set the value of the restorable property to value. Set_restorable(value) Set the value of the restorable property to value. Set_position(row, column, layer) Store the BoardItem position for self access.	notify([modifier])	Notify all the observers that a change occurred.
pickable()Returns True if the item is pickable, False otherwise.position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the restorable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	overlappable()	Returns True if the item is overlappable, False other-
position_as_vector()Returns the current item position as a Vector2Drender_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.		wise.
render_to_buffer(buffer, row, column,)Render the board item into a display buffer (not a screen buffer).restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	pickable()	Returns True if the item is pickable, False otherwise.
screen buffer). Returns True if the item is restorable, False otherwise. serialize() Return a dictionary with all the attributes of this object. set_can_move(value) set_overlappable(value) set_pickable(value) set_restorable(value) set_restorable(value) store_position(row, column, layer) Set the value of the restorable property to value. Set the value of the restorable property to value. Set the value of the restorable property to value.	position_as_vector()	Returns the current item position as a Vector2D
restorable()Returns True if the item is restorable, False otherwise.serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
wise. Serialize() Return a dictionary with all the attributes of this object. Set_can_move(value) Set the value of the can_move property to value. Set_overlappable(value) Set the value of the overlappable property to value. Set_pickable(value) Set the value of the pickable property to value. Set_restorable(value) Set the value of the restorable property to value. Set_restorable(value) Set the value of the restorable property to value. Store_position(row, column, layer) Store the BoardItem position for self access.		screen buffer).
serialize()Return a dictionary with all the attributes of this object.set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	restorable()	Returns True if the item is restorable, False other-
set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.		11-2-2-1
set_can_move(value)Set the value of the can_move property to value.set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	serialize()	Return a dictionary with all the attributes of this ob-
set_overlappable(value)Set the value of the overlappable property to value.set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.		ject.
set_pickable(value)Set the value of the pickable property to value.set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	set_can_move(value)	Set the value of the can_move property to value.
set_restorable(value)Set the value of the restorable property to value.store_position(row, column, layer)Store the BoardItem position for self access.	set_overlappable(value)	Set the value of the overlappable property to value.
store_position(row, column, layer) Store the BoardItem position for self access.	set_pickable(value)	Set the value of the pickable property to value.
	set_restorable(value)	Set the value of the restorable property to value.
store_screen_position(row, column) Store the screen position of the object.	store_position(row, column, layer)	Store the BoardItem position for self access.
	store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.

Continued on next page

352 Chapter 3. Tutorials

Table	67 –	 continued 	from	previous	page
-------	------	-------------------------------	------	----------	------

rainers or services and production productions			
layer	Convenience method to get the current stored layer		
	number of the item.		
model			
particle_emitter			
row	Convenience method to get the current stored row of		
	the item.		
screen_column	A property to get/set the screen column.		
screen_row	A property to get/set the screen row.		
size	A read-only property that gives the size of the item		
	as a 2 dimensions list.		
width	Convenience method to get the width of the item.		

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) - An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

• other (BoardItem) - The item you want to check for collision.

• **projection_offset** (Vector2D) – A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pg1BaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the Inventory.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
    print('The item is pickable')
```

```
position_as_vector()
```

Returns the current item position as a Vector2D

Returns The position as a 2D vector

```
Return type Vector2D
```

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render to buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set can move (value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable (value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.19 Movable

Bases: pygamelib.board_items.BoardItem

A class representing BoardItem capable of movements.

Movable subclasses BoardItem.

Parameters

- **step** (*int*) the amount of cell a movable can cross in one turn. Default value: 1.
- **step_vertical** (*int*) the amount of cell a movable can vertically cross in one turn. Default value: step value.
- **step_horizontal** (*int*) the amount of cell a movable can horizontally cross in one turn. Default value: step value.

• movement_speed (int/float) - The time (in seconds) between 2 movements of a Movable. It is used by all the Game's actuation methods to enforce move speed of NPC and projectiles.

The movement_speed parameter is only used when the Game is configured with MODE_RT. Additionally the dtmove property is used to accumulate time between frames. It is entirely managed by the Game object and most of the time you shouldn't mess up with it. Unless you want to manage movements by yourself. If so, have fun! That's the point of the pygamelib to let you do whatever you like.

This class derive BoardItem and describe an object that can move or be moved (like a player or NPC). Thus this class implements BoardItem.can_move(). However it does not implement BoardItem.pickable() or BoardItem.overlappable()

Methods

	Till all all and a second and a second and a second
init(step, step_vertical,)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Movable implements can_move().
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
has_inventory()	This is a virtual method that must be implemented in
	deriving class.
load(data)	Load data and create a new Movable out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Serialize the Immovable object.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

360 Chapter 3. Tutorials

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

${\tt can_move}\:(\:)\:\to bool$

Movable implements can_move().

Returns True

Return type Boolean

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)
Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

$has_inventory() \rightarrow bool$

This is a virtual method that must be implemented in deriving class. This method has to return True or False. This represent the capacity for a Movable to have an inventory.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new Movable out of it.

Parameters data (dict) - Data to create a new movable item (usually generated by serialize())

Returns A new complex item.

Return type ~pygamelib.board_items.Movable

model

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

${\tt overlappable}\,(\,)$

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

```
Return type Vector2D
```

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

${\tt restorable}\,(\,)$

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
    print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Serialize the Immovable object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

$\verb"set_pickable"\,(value)$

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

$\verb|set_restorable| (value)$

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.20 NPC

```
class pygamelib.board_items.NPC(actuator=None, **kwargs)
    Bases: pygamelib.board_items.Character
```

A class that represent a non playable character controlled by the computer. For the NPC to be successfully managed by the Game, you need to set an actuator.

None of the parameters are mandatory, however it is advised to make good use of some of them (like type or name) for game design purpose.

In addition to its own member variables, this class inherits all members from:

- pygamelib.board_items.Character
- pygamelib.board_items.Movable
- pygamelib.board_items.BoardItem

This class sets a couple of variables to default values:

- max_hp: 10
- hp: 10
- remaining_lives: 1
- attack_power: 5
- movement_speed: 0.25 (one movement every 0.25 second). Only useful if the game mode is set to MODE_RT.

Parameters actuator (pygamelib.actuators.Actuator) – An actuator, it can be any class but it need to implement pygamelib.actuators.Actuator.

Example:

```
mynpc = NPC(name='Idiot McStupid', type='dumb_enemy')
mynpc.step = 1
mynpc.actuator = RandomActuator()
```

```
___init___(actuator=None, **kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init([actuator])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Movable implements can_move().
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
has_inventory()	Define if the NPC has an inventory.
load(data)	Load data and create a new NPC out of it.
notify([modifier])	Notify all the observers that a change occurred.
	Continued on post page

Continued on next page

Table 70 – continued from previous page

	a wew breathers
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Define if the NPC is pickable.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Serialize the NPC object.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

```
\mathtt{can\_move}() \rightarrow bool
```

Movable implements can_move().

Returns True

Return type Boolean

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

has_inventory()

Define if the NPC has an inventory.

This method returns false because the game engine doesn't manage NPC inventory yet but it could be in the future. It's a good habit to check the value returned by this function.

Returns False

Return type Boolean

Example:

```
if mynpc.has_inventory():
    print("Cool: we can pickpocket that NPC!")
else:
    print("No pickpocketing XP for us today :(")
```

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new NPC out of it.

Parameters data (dict) – Data to create a new npc (usually generated by serialize())

Returns A new npc.

Return type ~pygamelib.board_items.NPC

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
    print('The item is overlappable')
```

particle_emitter

pickable()

Define if the NPC is pickable.

Obviously this method always return False.

Returns False

Return type Boolean

Example:

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\textbf{serialize}\,(\,)\,\to dict$

Serialize the NPC object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

set can move (value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

$store_position$ (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

• row (int) – the row of the item in the Board.

- column (int) the column of the item in the Board.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

 $store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.21 Player

```
class pygamelib.board_items.Player(inventory=None, **kwargs)
    Bases: pygamelib.board_items.Character
```

A class that represent a player controlled by a human.

This can take all parameter from Character, Movable and obviously BoardItem.

It is a specific board item as the whole Game class assumes only one player. Aside from the wrapper functions (like Game.move_player for example), there is no reel limitations to use more than one player.

The player also has a couple of attributes that are added for your convenience. You are free to use them or not. They are (name and default value):

- max_hp: 100
- hp: 100
- remaining_lives: 3
- attack_power: 10
- movement_speed: 0.1 (one movement every 0.1 second). Only useful if the game mode is set to MODE RT.
- inventory: A *Inventory* object. If none is provided, one is created automatically.

A player can be animated by providing a Animation object to its animation attribute.

Like all other board items, you can specify a *sprixel* attribute that will be the representation of the player on the board.

Example:

```
player = Player(
    name="Player",
    # A sprixel with "@" as the model, no background color, a cyan foreground
    # color and we set the background to be transparent.
    sprixel=core.Sprixel("@", None, core.Color(0, 255, 255), True),
    max_hp=200,
)
```

__init__ (inventory=None, **kwargs)

Like the object class, this class constructor takes no parameter.

Methods

init([inventory])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Movable implements can_move().
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
has_inventory()	This method returns True (a player has an inventory).
load(data)	Load data and create a new Character out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	This method returns False (a player is obviously not
	pickable).
<pre>position_as_vector()</pre>	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Serialize the Character object.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
<pre>store_position(row, column, layer)</pre>	Store the BoardItem position for self access.
<pre>store_screen_position(row, column)</pre>	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

${\tt can_move}\,(\,)\,\to bool$

Movable implements can_move().

Returns True

Return type Boolean

 $collides_with (other, projection_offset: pygamelib.base.Vector2D = None)$

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

has_inventory()

This method returns True (a player has an inventory).

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new Character out of it.

Parameters data (dict) - Data to create a new character item (usually generated by serialize())

Returns A new character item.

Return type ~pygamelib.board_items.Character

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

This method returns False (a player is obviously not pickable).

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render to buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Serialize the Character object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

```
# This is a silly example because the Board object does not allow
# that use case.
if item.column + item.size[0] >= board.width:
```

(continues on next page)

(continued from previous page)

```
Game.instance().screen.display_line(
    f"{item.name} cannot be placed at {item.pos}."
)
```

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

 $store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.22 Projectile

Bases: pygamelib.board items.Movable

A class representing a projectile type board item. That class can be sub-classed to represent all your needs (fireballs, blasters shots, etc.).

That class support the 2 types of representations: model and animations. The animation cases are slightly more evolved than the regular item.animation. It does use the item.animation but with more finesse as a projectile can travel in many directions. So it also keeps track of models and animation per travel direction.

You probably want to subclass Projectile. It is totally ok to use it as it, but it is easier to create a subclass that contains all your Projectile information and let the game engine deal with orientation, range keeping, etc. Please see examples/07_projectiles.py for a good old fireball example.

By default, Projectile travels in straight line in one direction. This behavior can be overwritten by setting a specific actuator (a projectile is a *Movable* so you can use my_projectile.actuator).

The general way to use it is as follow:

- Create a factory object with your static content (usually the static models, default direction and hit callback)
- Add the direction related models and/or animation (keep in mind that animation takes precedence over static models)
- deep copy that object when needed and add it to the projectiles stack of the game object.
- use Game.actuate_projectiles(level) to let the Game engine do the heavy lifting.

The Projectile constructor takes the following parameters:

Parameters

- direction (int) A direction from the constants module
- range (int) The maximum range of the projectile in number of cells that can be crossed.
 When range is attained the hit_callback is called with a BoardItemVoid as a collision object.
- **step** (*int*) the amount of cells a projectile can cross in one turn
- model(str) the default model of the projectile.
- movement_animation (Animation) the default animation of a projectile. If a projectile is sent in a direction that has no explicit and specific animation, then movement_animation is used if defined.
- hit_animation (Animation) the animation used when the projectile collide with something.
- hit_model (str) the model used when the projectile collide with something.
- hit_callback (function) A reference to a function that will be called upon collision. The hit_callback is receiving the object it collides with as first parameter.
- **is_aoe** (bool) Is this an 'area of effect' type of projectile? Meaning, is it doing something to everything around (mass heal, exploding rocket, fireball, etc.)? If yes, you must set

that parameter to True and set the aoe_radius. If not, the Game object will only send the colliding object in front of the projectile.

- **aoe_radius** (*int*) the radius of the projectile area of effect. This will force the Game object to send a list of all objects in that radius.
- callback_parameters (1ist) A list of parameters to pass to hit_callback.
- movement_speed (int | float) The movement speed of the projectile
- **collision_exclusions** (list) A list of **TYPES** of objects that should not collides with that projectile. It is usually a good idea to put the projectile type in the exclusion list. This prevent the projectile to collide with other instances of itself. Adding the projectile's emitter is also a valid idea.
- parent The parent object (usually a Board object or some sort of BoardItem).

Important: The effects of a Projectile are determined by the callback. No callback == no effect!

Example:

__init___(name='projectile', direction=10000100, step=1, range=5, model='', movement_animation=None, hit_animation=None, hit_model=None, hit_callback=None, is_aoe=False, aoe_radius=0, parent=None, callback_parameters=None, movement_speed=0.15, collision_exclusions=None, **kwargs) Like the object class, this class constructor takes no parameter.

Methods

init([name, direction, step, range,])	Like the object class, this class constructor takes no
	parameter.
add_directional_animation(direction, ani-	Add an animation for a specific direction.
mation)	
add_directional_model(direction, model)	Add an model for a specific direction.
attach(observer)	Attach an observer to this instance.
can_move()	Movable implements can_move().
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
directional_animation(direction)	Return the animation for a specific direction.
directional_model(direction)	Return the model for a specific direction.
display()	Print the model WITHOUT carriage return.
	Continued on next page

386 Chapter 3. Tutorials

Table 74 – continued from previous page

distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
has_inventory()	Projectile cannot have inventory by default.
hit(objects)	A method that is called when the projectile hit some-
	thing.
load(data)	Load data and create a new Movable out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Projectile are overlappable by default.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
remove_directional_animation(direction)	Remove an animation for a specific direction.
remove_directional_model(direction)	Remove the model for a specific direction.
<pre>render_to_buffer(buffer, row, column,)</pre>	Render the board item into a display buffer (not a
	screen buffer).
restorable()	We assume that by default, Projectiles are restorable.
serialize()	Serialize the Immovable object.
set_can_move(value)	Set the value of the can_move property to value.
set_direction(direction)	Set the direction of a projectile
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
(1) (2) (2) (1) (2)	
<pre>store_position(row, column, layer)</pre>	Store the BoardItem position for self access. Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
direction	The direction of the projectile.
dtmove	
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

add_directional_animation (direction, animation)

Add an animation for a specific direction.

Parameters

- **direction** (*int*) A direction from the constants module.
- animation (Animation) The animation for the direction

Example:

```
fireball.add_directional_animation(constants.UP, constants.UP, animation)
```

add_directional_model (direction, model)

Add an model for a specific direction.

Parameters

- **direction** (*int*) A direction from the constants module.
- model(str) The model for the direction

Example:

```
fireball.add_directional_animation(constants.UP, upward_animation)
```

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

$\mathtt{can_move}() \rightarrow bool$

Movable implements can_move().

Returns True

Return type Boolean

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on

different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (Vector2D) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

direction

The direction of the projectile.

Updating this property also updates the Unidirectional Actuator's direction.

Parameters value (int | Vector2D) - some param

Warning: If your projectile uses directional model and/or animation you should use $set_direction()$ to set the projectile direction.

Example:

```
bullet.direction = Vector2D(0, 1)
```

directional_animation (direction)

Return the animation for a specific direction.

Parameters direction (*int*) – A direction from the constants module.

Return type Animation

Example:

```
# No more animation for the UP direction fireball.directional_animation(constants.UP)
```

directional_model (direction)

Return the model for a specific direction.

Parameters direction (int) – A direction from the constants module.

Return type str

Example:

```
fireball.directional_model(constants.UP)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

dtmove

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

• **subject** (*PglBaseObject*) – The object that has changed.

- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

has_inventory()

Projectile cannot have inventory by default.

Returns False

Return type Boolean

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

hit (objects)

A method that is called when the projectile hit something.

That method is automatically called by the Game object when the Projectile collide with another object or is at the end of its range.

Here are the call cases covered by the Game object:

• range is reached without collision and projectile IS NOT an AoE type: hit() is called with a single BoardItemVoid in the objects list.

- range is reached without collision and projectile IS an AoE type: hit() is called with the list of all objects within aoe_radius (including structures).
- projectile collide with something and IS NOT an AoE type: hit() is called with the single colliding object in the objects list.
- projectile collide with something and IS an AoE type: hit() is called with the list of all objects within aoe_radius (including structures).

In turn, that method calls the hit callback with the following parameters (in that order):

- 1. the projectile object
- 2. the list of colliding objects (that may contain only one object)
- 3. the callback parameters (from the constructor callback_parameters)

Parameters objects – A list of objects hit by or around the projectile.

Example:

```
my_projectile.hit([npc1])
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load (data)

Load data and create a new Movable out of it.

Parameters data (dict) - Data to create a new movable item (usually generated by serialize())

Returns A new complex item.

Return type ~pygamelib.board_items.Movable

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.

• **value** (Any) – An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Projectile are overlappable by default.

Returns True

Return type Boolean

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

remove_directional_animation(direction)

Remove an animation for a specific direction.

Parameters direction (int) – A direction from the constants module.

Example:

```
# No more animation for the UP direction
fireball.remove_directional_animation(constants.UP)
```

remove_directional_model(direction)

Remove the model for a specific direction.

Parameters direction (*int*) – A direction from the constants module.

Example:

```
fireball.directional_model(constants.UP)
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

We assume that by default, Projectiles are restorable.

Returns True

Return type bool

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Serialize the Immovable object.

This returns a dictionary that contains all the key/value pairs that makes up the object.

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_direction (direction)

Set the direction of a projectile

This method will set a Unidirectional Actuator with the direction. It will also take care of updating the model and animation for the given direction if they are specified.

Parameters direction (*int*) – A direction from the constants module.

Example:

```
fireball.set_direction(constants.UP)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

 $store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.23 TextItem

```
class pygamelib.board_items.TextItem(text=None, **kwargs)
    Bases: pygamelib.board_items.BoardComplexItem
```

New in version 1.2.0.

The text item is a board item that can contains text. The text can then be manipulated and placed on a Board.

It is overall a *BoardComplexItem* (so it takes all the parameters of that class). The big difference is that the first parameter is the text you want to display.

The text parameter can be either a regular string or a Text object (in case you want formatting and colors).

Parameters text (str | Text) – The text you want to display.

Example:

```
city_name = TextItem('Super City')
fancy_city_name = TextItem(text=base.Text('Super City', base.Fore.GREEN,
    base.Back.BLACK,
    base.Style.BRIGHT
))
my_board.place_item(city_name, 0, 0)
my_board.place_item(fancy_city_name, 1, 0)
```

```
__init___(text=None, **kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init([text])	Like the object class, this class constructor takes no
([])	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Returns True if the item can move, False otherwise.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
item(row, column)	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new TextItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	A property to get and set the size that the BoardItem
	takes in the Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
	Continued on next page

Iabla	//	aantiniiad	trom	previous page
1aune	, ,	— (.0)	11 ()1 1 1	DIEVIOUS DAGE

row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
text	The text within the item.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Returns True if the item can move, False otherwise.

Example:

```
if board.item(4,5).can_move():
    print('The item can move')
```

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

• **other** (*BoardItem*) – The item you want to check for collision.

• **projection_offset** (Vector2D) – A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (*Pg1BaseObject*) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

```
Return type Vector2D
```

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

A property to get and set the size that the BoardItem takes in the *Inventory*.

Returns The size of the item.

Return type int

```
item(row, column)
```

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises Pq10utOfBoardBoundException - if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new TextItem out of it.

Parameters data (dict) - Data to create a new text item (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.TextItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

```
Return type Vector2D
```

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
    print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
serialize() \rightarrow dict
```

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable (value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

```
# This is a silly example because the Board object does not allow
# that use case.
if item.column + item.size[0] >= board.width:
```

(continues on next page)

(continued from previous page)

```
Game.instance().screen.display_line(
    f"{item.name} cannot be placed at {item.pos}."
)
```

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) - The sprite to set

Example:

```
npc1 = board_items.ComplexNpc(
                                 sprite=npc_sprite_collection['npc1_idle']
# to access the sprite:
if npc1.sprite.width * npc1.sprite.height > CONSTANT_BIG_GUY:
   game.screen.place(
        base.Text (
            'Big boi detected!!!',
            core.Color(255,0,0),
            style=constants.BOLD,
        ),
        notifications.row,
        notifications.column,
    )
# And to set it:
if game.player in game.neighbors(3, npc1):
   npc1.sprite = npc_sprite_collection['npc1_fight']
```

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column(int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

text

The text within the item.

TextItem.text can be set to either a string or a Text object.

It will always return a Text object.

Internally it translate the text to a *Sprite* to display it correctly on a *Board*. If print()-ed it will do so like the *Text* object.

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
   print('The item is too big for the board.')
```

3.4.24 Tile

```
class pygamelib.board_items.Tile(**kwargs)
    Bases: pygamelib.board_items.BoardComplexItem, pygamelib.board_items.
    GenericStructure
```

New in version 1.2.0.

A Tile is a standard <code>BoardComplexItem</code> configured by default to:

- be overlappable
- be restorable
- · be not pickable

• be immovable.

Aside from the movable attributes (it inherit from GenericStructure so it's an Immovable object), everything else is configurable.

It is particularly useful to display a Sprite on the background or to create terrain.

Example:

```
grass_sprite = Sprite.load_from_ansi_file('textures/grass.ans')
for pos in grass_positions:
   outdoor_level.place_item( Tile(sprite=grass_sprite), pos[0], pos[1] )
```

```
___init___(**kwargs)
```

Parameters

- **overlappable** (bool) Defines if the Tile can be overlapped.
- **restorable** (bool) Defines is the Tile should be restored after being overlapped.
- pickable (bool) Defines if the Tile can be picked up by the Player or NPC.

Please see BoardComplexItem for additional parameters.

Methods

init(**kwargs)	
	param overlappable Defines if the Tile
	can be overlapped.
attach(observer)	Attach an observer to this instance.
can_move()	A Tile cannot move.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
<u> </u>	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
<pre>item(row, column)</pre>	Return the item component at the row, column posi-
	tion if it is within the complex item's boundaries.
load(data)	Load data and create a new Tile out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	Returns True if the item is overlappable, False other-
	wise.
pickable()	Returns True if the item is pickable, False otherwise.
position_as_vector()	Returns the current item position as a Vector2D
<pre>render_to_buffer(buffer, row, column,)</pre>	Render the complex board item from the display
	buffer to the frame buffer.
restorable()	Returns True if the item is restorable, False other-
	wise.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
	Continued on next page

406 Chapter 3. Tutorials

Table 78 – continued from previous page

set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.
update_sprite()	Update the complex item with the current sprite.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
sprite	A property to easily access and update a complex
	item's sprite.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

A Tile cannot move.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- **other** (*BoardItem*) The item you want to check for collision.
- **projection_offset** (Vector2D) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PqlBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance to (other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (PqlBaseObject) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

item(row, column)

Return the item component at the row, column position if it is within the complex item's boundaries.

Return type ~pygamelib.board_items.BoardItem

Raises PglOutOfBoardBoundException – if row or column are out of bound.

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load(data)

Load data and create a new Tile out of it.

Parameters data (dict) – Data to create a new tile (usually generated by serialize())

Returns A new complex npc.

Return type ~pygamelib.board_items.Tile

model

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

• modifier (PglBaseObject) – An optional parameter that identify the modifier object to exclude it from the notified objects.

- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

Returns True if the item is overlappable, False otherwise.

Example:

```
if board.item(4,5).overlappable():
   print('The item is overlappable')
```

particle_emitter

pickable()

Returns True if the item is pickable, False otherwise.

Example:

```
if board.item(4,5).pickable():
   print('The item is pickable')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the complex board item from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

Returns True if the item is restorable, False otherwise.

Example:

```
if board.item(4,5).restorable():
   print('The item is restorable')
```

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

```
# This is a silly example because the Board object does not allow
# that use case.
if item.column + item.size[0] >= board.width:
    Game.instance().screen.display_line(
        f"{item.name} cannot be placed at {item.pos}."
    )
```

sprite

A property to easily access and update a complex item's sprite.

Parameters new_sprite (Sprite) – The sprite to set

Example:

$store_position$ (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- layer the layer of the item in the Board. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position(row: int, column: int) \rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_sprite()

Update the complex item with the current sprite.

Note: This method use to need to be called every time the sprite was changed. Starting with version 1.3.0, it is no longer a requirement as BoardComplexItem.sprite was turned into a property that takes care of calling update_sprite().

Example:

```
item = BoardComplexItem(sprite=position_idle)
for s in [walk_1, walk_2, walk_3, walk_4]:
    # This is not only no longer required but also wasteful as
    # update_sprite() is called twice here.
    item.sprite = s
    item.update_sprite()
    board.move(item, constants.RIGHT, 1)
    time.sleep(0.2)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.25 Treasure

```
class pygamelib.board_items.Treasure(value=10, **kwargs)
    Bases: pygamelib.board_items.Immovable
```

A Treasure is an *Immovable* that is pickable and with a non zero value. It is an helper class that allows to focus on game design and mechanics instead of small building blocks.

Parameters

- model(str) The model that will represent the treasure on the map
- **value** (*int*) The value of the treasure, it is usually used to calculate the score.
- **inventory_space** (*int*) The space occupied by the treasure. It is used by *Inventory* as a measure of space. If the treasure's size exceed the Inventory size (or the cumulated size of all items + the treasure exceed the inventory max_size()) the Inventory will refuse to add the treasure.

Note: All the options from *Immovable* are also available to this constructor.

Example:

```
money_bag = Treasure(
    model=graphics.Models.MONEY_BAG,value=100,inventory_space=2
)
print(f"This is a money bag {money_bag}")
player.inventory.add_item(money_bag)
print(f"The inventory value is {player.inventory.value()} and is at
    {player.inventory.size()}/{player.inventory.max_size}")
```

```
___init___(value=10, **kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init([value])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	This represent the capacity for a Treasure to be over-
	lapped by player or NPC.
	0

Continued on next page

Table 80 – continued from previous page

pickable()	This represent the capacity for a Treasure to be
	picked-up by player or NPC.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	This represent the capacity for a Treasure to be re-
	stored after being overlapped.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

416 Chapter 3. Tutorials

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*Pg1BaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

heading

418

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
   print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load (data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

• **modifier** (*Pg1BaseObject*) – An optional parameter that identify the modifier object to exclude it from the notified objects.

- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

This represent the capacity for a Treasure to be overlapped by player or NPC.

A treasure is not overlappable.

Returns False

Return type bool

particle_emitter

pickable()

This represent the capacity for a Treasure to be picked-up by player or NPC.

A treasure is obviously pickable by the player and potentially NPCs. Board puts the Treasure in the Inventory if the picker implements has_inventory()

Returns True

Return type bool

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

restorable()

This represent the capacity for a Treasure to be restored after being overlapped.

A treasure is not overlappable, therefor is not restorable.

Returns False

Return type bool

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (*int*) – the screen row

Return type int

$serialize() \rightarrow dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

set_can_move(value)

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

set_overlappable(value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

set_pickable(value)

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_pickable(False)
```

```
set restorable (value)
```

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- **row** (*int*) the row of the item in the *Board*.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width

Return type int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.4.26 Wall

```
class pygamelib.board_items.Wall(**kwargs)
    Bases: pygamelib.board_items.Immovable
```

A Wall is a specialized *Immovable* object that as unmodifiable characteristics:

- It is not pickable (and cannot be).
- It is not overlappable (and cannot be).
- It is not restorable (and cannot be).

As such it's an object that cannot be moved, cannot be picked up or modified by Player or NPC and block their ways. It is therefor advised to create one per board and reuse it in many places.

Parameters

- **model** (*str*) The representation of the Wall on the Board.
- name (str) The name of the Wall.
- **size** (*int*) The size of the Wall. This parameter will probably be deprecated as size is only used for pickable objects.

```
___init___(**kwargs)
```

Like the object class, this class constructor takes no parameter.

Methods

init(**kwargs)	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
can_move()	Return the capability of moving of an item.
collides_with(other, projection_offset)	Tells if this item collides with another item.
debug_info()	Return a string with the list of the attributes and their
	current value.
detach(observer)	Detach an observer from this instance.
display()	Print the model WITHOUT carriage return.
distance_to(other)	Calculates the distance with an item.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load data and create a new BoardItem out of it.
notify([modifier])	Notify all the observers that a change occurred.
overlappable()	This represent the capacity for a BoardItem to be
	overlapped by player or NPC.
	0

Continued on next page

Table 82 – continued from previous page

pickable()	This represent the capacity for a BoardItem to be
	pick-up by player or NPC.
position_as_vector()	Returns the current item position as a Vector2D
render_to_buffer(buffer, row, column,)	Render the board item into a display buffer (not a
	screen buffer).
restorable()	This represent the capacity for an Immovable
	Movable item.
serialize()	Return a dictionary with all the attributes of this ob-
	ject.
set_can_move(value)	Set the value of the can_move property to value.
set_overlappable(value)	Set the value of the overlappable property to value.
set_pickable(value)	Set the value of the pickable property to value.
set_restorable(value)	Set the value of the restorable property to value.
store_position(row, column, layer)	Store the BoardItem position for self access.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

	A
animation	A property to get and set an Animation for this
	item.
column	Convenience method to get the current stored col-
	umn of the item.
heading	Return the heading of the item.
height	Convenience method to get the height of the item.
inventory_space	Return the size that the Immovable item takes in the
	Inventory.
layer	Convenience method to get the current stored layer
	number of the item.
model	
particle_emitter	
row	Convenience method to get the current stored row of
	the item.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
size	A read-only property that gives the size of the item
	as a 2 dimensions list.
width	Convenience method to get the width of the item.

animation

A property to get and set an Animation for this item.

Important: When an animation is set, the item is setting the animation's parent to itself.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

424 Chapter 3. Tutorials

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

can_move()

Return the capability of moving of an item.

Obviously an Immovable item is not capable of moving. So that method always returns False.

Returns False

Return type bool

collides_with (other, projection_offset: pygamelib.base.Vector2D = None)

Tells if this item collides with another item.

Important: collides_with() does not take the layer into account! It is not desirable for the pygamelib to assume that 2 items on different layers wont collide. For example, if a player is over a door, they are on different layers, but logically speaking they are colliding. The player is overlapping the door. Therefor, it is the responsibility of the developer to check for layers in collision, if it is important to the game logic.

Parameters

- other (BoardItem) The item you want to check for collision.
- **projection_offset** (*Vector2D*) A vector to offset this board item's position (not the position of the *other* item). Use this to detect a collision before moving the board item. You can pass the movement vector before moving to check if a collision will occur when moving.

Return type bool

Example:

```
if projectile.collides_with(game.player):
    game.player.hp -= 5
```

column

Convenience method to get the current stored column of the item.

This is absolutely equivalent to access to item.pos[1].

Returns The column coordinate

Return type int

Example:

```
if item.column != item.pos[1]:
    print('Something extremely unlikely just happened...')
```

debug_info()

Return a string with the list of the attributes and their current value.

Return type str

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

display()

Print the model WITHOUT carriage return.

distance_to(other)

Calculates the distance with an item.

Parameters other (BoardItem) – The item you want to calculate the distance to.

Returns The distance between this item and the other.

Return type float

Example:

```
if npc.distance_to(game.player) <= 2.0:
    npc.seek_and_destroy = True</pre>
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*Pg1BaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

heading

Return the heading of the item.

This is a read only property that is updated by store_position().

The property represent the orientation and movement of the item in the board. It gives the difference between the item's centroid current and previous position. Thus, giving you both the direction and the distance of the movement. You can get the angle from here.

One of the possible usage of that property is to set the sprite/sprixel/model of a moving item.

Returns The heading of the item.

Return type Vector2D

Example:

```
if my_item.heading.column > 0:
    my_item.sprixel.model = item_models["heading_right"]
```

Warning: Just after placing an item on the board, and before moving it, the heading cannot be trusted! The heading represent the direction and orientation of the **movement**, therefore, it is not reliable before the item moved.

height

Convenience method to get the height of the item.

This is absolutely equivalent to access to item.size[1].

Returns The height

Return type int

Example:

```
if item.height > board.height:
    print('The item is too big for the board.')
```

inventory_space

Return the size that the Immovable item takes in the *Inventory*.

Returns The size of the item.

Return type int

layer

Convenience method to get the current stored layer number of the item.

This is absolutely equivalent to access to item.pos[2].

Returns The layer number

Return type int

Example:

```
if item.layer != item.pos[2]:
    print('Something extremely unlikely just happened...')
```

classmethod load (data)

Load data and create a new BoardItem out of it.

Parameters data (dict) – Data to create a new item (usually generated by serialize())

Returns A new item.

Return type ~pygamelib.board_items.BoardItem

model

```
notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.
```

Parameters

• **modifier** (*Pg1BaseObject*) – An optional parameter that identify the modifier object to exclude it from the notified objects.

- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

overlappable()

This represent the capacity for a *BoardItem* to be overlapped by player or NPC.

Returns False

Return type bool

particle_emitter

pickable()

This represent the capacity for a BoardItem to be pick-up by player or NPC.

Returns False

Return type bool

Example:

```
if mywall.pickable():
    print('Whoaa this wall is really light... and small...')
else:
    print('Really? Trying to pick-up a wall?')
```

position_as_vector()

Returns the current item position as a Vector2D

Returns The position as a 2D vector

Return type Vector2D

Example:

```
gravity = Vector2D(9.81, 0)
next_position = item.position_as_vector() + gravity.unit()
```

render_to_buffer (buffer, row, column, height, width)

Render the board item into a display buffer (not a screen buffer).

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

```
restorable()
```

This represent the capacity for an *Immovable* Movable item. A wall is not overlappable.

Returns False

Return type bool

row

Convenience method to get the current stored row of the item.

This is absolutely equivalent to access to item.pos[0].

Returns The row coordinate

Return type int

Example:

```
if item.row != item.pos[0]:
    print('Something extremely unlikely just happened...')
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\textbf{serialize}\,(\,)\,\to dict$

Return a dictionary with all the attributes of this object.

Returns A dictionary with all the attributes of this object.

Return type dict

```
set_can_move(value)
```

Set the value of the can_move property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_can_move(False)
```

$\mathtt{set_overlappable}$ (value)

Set the value of the overlappable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_overlappable(False)
```

$\verb"set_pickable"\,(value)$

Set the value of the pickable property to value.

Parameters value (bool) – The value to set.

Example:

3.4. board_items 429

```
item.set_pickable(False)
```

set_restorable(value)

Set the value of the restorable property to value.

Parameters value (bool) – The value to set.

Example:

```
item.set_restorable(False)
```

size

A read-only property that gives the size of the item as a 2 dimensions list. The first element is the width and the second the height.

Returns The size.

Return type list

Example:

store_position (row: int, column: int, layer: int = 0)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self position. It is a redundant information and might not be synchronized.

Parameters

- row (int) the row of the item in the Board.
- **column** (*int*) the column of the item in the *Board*.
- **layer** the layer of the item in the *Board*. By default layer is set to 0.

Example:

```
item.store_position(3,4)
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column(int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

Convenience method to get the width of the item.

This is absolutely equivalent to access to item.size[0].

Returns The width **Return type** int

Example:

```
if item.width > board.width:
    print('The item is too big for the board.')
```

3.5 constants

Accessible constants are the following:

General purpose:

• PYGAMELIB_VERSION

Directions:

- NO_DIR: This one is used when no direction can be provided by an actuator (destination reached for a PathFinder for example)
- UP
- DOWN
- LEFT
- RIGHT
- DRUP: Diagonal right up
- DRDOWN: Diagonal right down
- DLUP: Diagonal Left up
- DLDOWN: Diagonal left down

Permissions:

- PLAYER_AUTHORIZED
- NPC_AUTHORIZED
- ALL_PLAYABLE_AUTHORIZED (deprecated in 1.2.0 in favor of ALL_CHARACTERS_AUTHORIZED)
- ALL_CHARACTERS_AUTHORIZED
- ALL_MOVABLE_AUTHORIZED
- NONE_AUTHORIZED

UI positions:

- ORIENTATION_HORIZONTAL
- ORIENTATION_VERTICAL
- ALIGN_LEFT
- ALIGN_RIGHT

3.5. constants 431

ALIGN CENTER

Actions states (for Actuators for example):

- RUNNING
- PAUSED
- STOPPED

Accepted input (mainly used in pygamelib.gfx.ui for input dialogs): * INTEGER_FILTER * PRINTABLE_FILTER Path Finding Algorithm Constants:

- ALGO_BFS
- ALGO ASTAR

Text styling constants:

- BOLD
- UNDERLINE

Special constants:

- NO_PLAYER: That constant is used to tell the Game object not to manage the player.
- MODE_RT: Set the game object to Real Time mode. The game runs independently from the user input.
- MODE_TBT: Set the game object to Turn By Turn mode. The game runs turn by turn and pause between each user input.

3.6 engine

3.6.1 Board

Bases: pygamelib.base.PglBaseObject

A class that represent a game board.

The board object is a 2D matrix of board items. This means that you can visualize it as a chessboard for example. All board items are positioned on this chessboard-like object and can be moved around.

The Board object is the base object to build a level. Once created to your liking you can add items from the board_items module. You can also derived BoardItem to create your own board items, specific to your game.

If you want a detailed introduction to the Board object, go the the pygamelib wiki and read the "Getting started: the Board" article.

Note: In version 1.3.0 a new screen rendering stack was introduced. With this came the need for some object to hold more information about their state. This is the case for Board. To use partial display with the *Screen* buffer system the board itself needs to hold the information about were to draw and on what to focus on. The

existing code will still work as the *Game* object takes care of forwarding the information to the Board. However, it is now possible to exploit the *Camera* object to create cut scenes and more interesting movements.

Important: Partial display related parameters are information used by the <code>display_around()</code> method and the <code>Screen</code> object to either display directly the board (display_around) or render the Board in the frame buffer. **You have to make sure that the focus element's position is updated**. If you use the player, you have nothing to do but the Camera object needs to be manually updated for example.

Warning: in 1.3.0 the notion of layers was added to the Board object. Layers are used to better manage items overlapping. For the moment, layers are automatically managed to expand and shrink on demand (or on a need basis). You can use the layer system to add some depth to your game but you should be warned that you may experience some issues. If it is the case please report them on the Github issues page. For existing code, the entire Board object behaves exactly like in version 1.2.x.

__init__ (name: str = 'Board', size: list = None, ui_borders: str = None, ui_border_bottom:
 str = '-', ui_border_top: str = '-', ui_border_left: str = '\', ui_border_right: str =
 '\', ui_board_void_cell=' ', ui_board_void_cell_sprixel: pygamelib.gfx.core.Sprixel
 = None, player_starting_position: list = None, DISPLAY_SIZE_WARNINGS=False,
 parent=None, partial_display_viewport=None, partial_display_focus=None, en able_partial_display=False)

Parameters

- name (str) the name of the Board
- **size** (list) array [width,height] with width and height being int. The size of the board. If layers is not specified it is set to 5.
- player_starting_position (list) array [row,column] with row and column being int. The coordinates at which Game will place the player on change_level().
- ui_borders (str) To set all the borders to the same value
- ui_border_left (str) A string that represents the left border.
- ui_border_right (str) A string that represents the right border.
- ui_border_top (str) A string that represents the top border.
- ui border bottom (str) A string that represents the bottom border.
- ui_board_void_cell (str) A string that represents an empty cell. This option is going to be the model of the BoardItemVoid (see pygamelib.board_items. BoardItemVoid)
- parent (Game) The parent object (usually the Game object).
- **DISPLAY_SIZE_WARNINGS** (bool) A boolean to show or hide the warning about boards bigger than 80 rows and/or columns.
- enable_partial_display (bool) A boolean to tell the Board to enable or not partial display of boards. Default: False.
- partial_display_viewport (list) A 2 int elements array that gives the radius of the partial display in number of row and column. Please see display_around().

• partial_display_focus (BoardItem or Vector2D) - An item to focus (i.e center) the view on. When partial display is enabled the rendered view will be centered on this focus point/item. It can be an item or a vector.

Methods

init(name, size, ui_borders,[,])	param name the name of the Board
attach(observer)	Attach an observer to this instance.
check_sanity()	Check the board sanity.
<pre>clear_cell(row, column[, layer])</pre>	Clear cell (row, column, layer)
detach(observer)	Detach an observer from this instance.
display()	Display the entire board.
display_around(item, row_radius, col-	Display only a part of the board.
umn_radius)	
<pre>generate_void_cell()</pre>	This method return a void cell.
<pre>get_immovables(**kwargs)</pre>	Return a list of all the Immovable objects in the
	Board.
get_movables(**kwargs)	Return a list of all the Movable objects in the Board.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
init_board()	Initialize the board with BoardItemVoid that uses
	ui_board_void_cell_sprixel or ui_board_void_cell
	(in that order of preference) as model.
<pre>init_cell(row, column[, layer])</pre>	Initialize a specific cell of the board with Board-
	ItemVoid that uses ui_board_void_cell as model.
instantiate_item(data)	Instantiate a BoardItem from its serialized data.
<pre>item(row, column[, layer])</pre>	Return the item at the row, column, layer position if
	within board's boundaries.
layers(row, column)	A method to get the number of layers at the Board's
	given coordinates.
load(data)	Create a new Board object based on serialized data.
<pre>move(item, direction[, step])</pre>	Board.move() is a routing function.
neighbors(obj, radius)	Returns a list of neighbors (non void item) around an object.
notify([modifier])	Notify all the observers that a change occurred.
place_item(item, row, column, layer, auto_layer)	Place an item at coordinates row, column and layer.
remove_item(item)	Remove an item from the board.
render_cell(row, column)	New in version 1.3.0.
render_to_buffer(buffer, row, column,)	Render the board into from the display buffer to the frame buffer.
serialize()	Return a serialized version of the board.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

434 Chapter 3. Tutorials

height	A convenience read only property to get the height
	of the Board.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
width	A convenience read only property to get the width of
	the Board.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

$check_sanity() \rightarrow None$

Check the board sanity.

This is essentially an internal method called by the constructor.

```
clear_cell (row, column, layer=0)
```

Clear cell (row, column, layer)

This method clears a cell, meaning it position a void_cell BoardItemVoid at these coordinates.

It also removes the items from the list of movables and immovables.

Parameters

- row (int) The row of the item to remove
- column (int) The column of the item to remove
- layer (int) The layer of the item to remove. The default value is 0 to remain coherent with previous version of the library.

Example:

```
myboard.clear_cell(3,4,0)
```

Warning: This method does not check the content before, it will overwrite the content.

Important: In the case of a BoardComplexItem derivative (Tile, ComplexPlayer, ComplexNPC, etc.) clearing one cell of the entire item is enough to remove the entire item from the list of movables or immovables.

Note: Starting in 1.3.0 and the addition of board's layers, there is no more overlapping matrix. With no more moving items around this method should be a little faster. It also means that the layer parameter is really important (a wrong layer means that you'll clear the wrong cell). Be ready to catch an IndexError exception

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

$display() \rightarrow None$

Display the entire board.

This method display the Board (as in print()), taking care of displaying the borders, and everything inside.

It uses the <u>__str__</u> method of the item, which by default uses (in order) BoardItem.sprixel and (if no sprixel is defined) BoardItem.model. If you want to override this behavior you have to subclass BoardItem.

```
display\_around(item, row\_radius, column\_radius) \rightarrow None
```

Display only a part of the board.

This method behaves like display() but only display a part of the board around an item (usually the player). Example:

```
# This will display only a total of 30 cells vertically and
# 60 cells horizontally.
board.display_around(player, 15, 30)
```

Parameters

- item (BoardItem) an item to center the view on (it has to be a subclass of BoardItem)
- row_radius (int) The radius of display in number of rows showed. Remember that it is a radius not a diameter...
- **column_radius** (*int*) The radius of display in number of columns showed. Remember that... Well, same thing.

It uses the same display algorithm than the regular display() method.

generate_void_cell()

This method return a void cell.

If ui_board_void_cell_sprixel is defined it uses it, otherwise use ui_board_void_cell to generate the void item.

Returns A void board item

Return type BoardItemVoid

Example:

```
board.generate_void_cell()
```

get_immovables (**kwargs)

Return a list of all the Immovable objects in the Board.

See pygamelib.board_items.Immovable for more on an Immovable object.

Parameters **kwargs – an optional dictionnary with keys matching Immovables class members and value being something **contained** in that member.

Returns A list of Immovable items

Example:

```
for m in myboard.get_immovables():
    print(m.name)

# Get all the Immovable objects that type contains "wall"
    AND name contains fire
walls = myboard.get_immovables(type="wall",name="fire")
```

get movables (**kwargs)

Return a list of all the Movable objects in the Board.

See pygamelib.board_items.Movable for more on a Movable object.

Parameters **kwargs – an optional dictionnary with keys matching Movables class members and value being something contained in that member.

Returns A list of Movable items

Example:

```
for m in myboard.get_movables():
    print(m.name)

# Get all the Movable objects that has a type that contains "foe"
foes = myboard.get_movables(type="foe")
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

height

A convenience read only property to get the height of the Board.

It is absolutely equivalent to access to board.size[1].

Returns The height of the board.

Return type int

Example:

```
if board.size[1] != board.height:
    print('Houston, we have a problem...')
```

init board()

Initialize the board with BoardItemVoid that uses ui_board_void_cell_sprixel or ui_board_void_cell (in that order of preference) as model.

This method is automatically called by the Board's constructor.

Example:

```
myboard.init_board()
```

$init_cell(row, column, layer=0) \rightarrow None$

Initialize a specific cell of the board with BoardItemVoid that uses ui_board_void_cell as model.

Parameters

- **row** (*int*) the row coordinate.
- **column** (*int*) the column coordinate.

Example:

```
myboard.init_cell(2,3,0)
```

static instantiate_item(data: dict)

Instantiate a BoardItem from its serialized data.

Parameters data (dict) – The data to use to build the item.

Returns an instance of a BoardItem.

Important: The actual object depends on the serialized data. It can be any derivative of BoardItem (even custom objects as long as they inherit from BoardItem) as long as they are importable by this class.

Example:

```
# First get some board item serialization data. For example:
data = super_duper_npc.serialize()
# Then instantiate a new one:
another_super_duper_npc = Board.instantiate_item(data)
```

item(row, column, layer=-1)

Return the item at the row, column, layer position if within board's boundaries.

Parameters

- row(int) The row to probe.
- column(int) The column to probe.
- **layer** (*int*) The layer to probe (default: -1 i.e the top item).

Return type pygamelib.board_items.BoardItem

Raises Pq10utOfBoardBoundException - if row, column or layer are out of bound.

```
layers (row, column) \rightarrow int
```

A method to get the number of layers at the Board's given coordinates.

Returns The number of layers of the board.

Return type int

Example:

```
if board.layers(game.player.row, game.player.column) > 1:
    print('The player is stomping on something!')
```

classmethod load(data: dict = None)

Create a new Board object based on serialized data.

If data is None, None is returned.

If a color component is missing from data, it is set to 0 (see examples).

Raises an exception if the color components are not integer.

Parameters data (dict) – Data loaded from JSON data (serialized).

Returns Either a Board object or None if data where empty.

Return type Board | NoneType

Raise PglInvalidTypeException

Example:

```
# Loading from parsed JSON data
new_board = Board.load(json.load("board_lvl_01.json"))
```

move (item, direction, step=1)

Board.move() is a routing function. It does 2 things:

- **1 If the direction is a** *Vector2D***, round the** values to the nearest integer (as move works with entire board cells, i.e integers).
- 2 route toward the right moving function depending if the item is complex or not.

Move an item in the specified direction for a number of steps.

Parameters

- item (pygamelib.board_items.Movable) an item to move (it has to be a subclass of Movable)
- direction (pygamelib.constants or Vector2D) a direction from constants
- **step** (*int*) the number of steps to move the item.

If the number of steps is greater than the Board, the item will be move to the maximum possible position.

If the item is not a subclass of Movable, an PglObjectIsNotMovableException exception (see pygamelib.base.PglObjectIsNotMovableException).

Example:

```
board.move(player,constants.UP,1)
```

Important: if the move is successful, an empty BoardItemVoid (see pygamelib.boards_item. BoardItemVoid) will be put at the departure position (unless the movable item is over an overlappable item). If the movable item is over an overlappable item, the overlapped item is restored.

Important: Also important: If the direction is a *Vector2D*, the values will be rounded to the nearest integer (as move works with entire board cells). It allows for movement accumulation before actually moving. The step parameter is not used in that case.

```
neighbors(obj, radius: int = 1)
```

Returns a list of neighbors (non void item) around an object.

This method returns a list of objects that are all around an object between the position of an object and all the cells at **radius**.

Parameters

- radius (int) The radius in which non void item should be included
- **obj** (BoardItem) The central object. The neighbors are calculated for that object.

Returns A list of BoardItem. No BoardItemVoid is included.

Raises PglInvalidTypeException - If radius is not an int.

Example:

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

place_item(item, row: int, column: int, layer: int = 0, auto_layer: bool = True)

Place an item at coordinates row, column and layer.

If row, column or layer are out of the board boundaries, a PglOutOfBoardBoundException is raised.

If the item is not a subclass of BoardItem, a PglInvalidTypeException

The observers are notified of a successful placement with the pygamelib.engine.Board.place_item:item_placed event. The item that was deleted is passed as the value of the event.

Warning: Nothing prevents you from placing an object on top of another. Be sure to check that. This method will check for items that are both overlappable **and** restorable to save them, but that's the extend of it.

remove_item(item)

Remove an item from the board.

If the item is a single BoardItem, this method is absolutely equivalent to calling <code>clear_cell()</code>. If item is a derivative of BoardComplexItem, it is not as clear_cell() only clears a specific cell (that can be part of a complex item). This method actually remove the entire item and clears all its cells.

The observers are notified of a successful removal with the pygamelib.engine.Board.remove_item:item_removed event. The item that was deleted is passed as the value of the event.

Parameters item (BoardItem) – The item to remove.

Example:

```
game.current_board().remove_item(game.player)
```

```
render_cell (row, column)
```

New in version 1.3.0.

Render the cell at given position.

This method always return a *Sprixel* (it could be an empty one though). It automatically render the highest item (if items are overlapping for example). If the rendered *Sprixel* is configured to have transparent background, this method is going to go through the layers to make sure that it is rendering the sprixels correctly (i.e. with the right background color).

For basic usage of the library it is unlikely that you will use it. It is part of the screen rendering stack introduced in version 1.3.0. Actually unless you need to write a different rendering system you won't use that method.

Parameters

- row (int) The row to render.
- **column** (*int*) The column to render.

Return type Sprixel

Raises Pq10utOfBoardBoundException - if row or column are out of bound.

Example:

```
# This renders the board from the top left corner of the screen.
for row in range(0, myboard.height):
    for column in range(0, myboard.height):
        myscreen.place(
            myboard.render_cell(row, column)
        ),
        row,
        column,
```

 $\verb"render_to_buffer" (\textit{buffer}, \textit{row}, \textit{column}, \textit{buffer_height}, \textit{buffer_width}) \rightarrow \texttt{None}$

Render the board into from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (*numpy.array*) A frame buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

screen_column

A property to get/set the screen column.

```
Parameters value (int) – the screen column
```

Return type int

screen_row

A property to get/set the screen row.

```
Parameters value (int) – the screen row
```

Return type int

serialize()

Return a serialized version of the board.

Returns A dictionary containing the board's attributes.

Example:

```
serialized_board_data = myboard.serialize()
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- **row** (*int*) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

width

A convenience read only property to get the width of the Board.

It is absolutely equivalent to access to board.size[0].

Returns The width of the board.

Return type int

Example:

```
if board.size[0] != board.width:
    print('Houston, we have a problem...')
```

3.6.2 Game

```
class pygamelib.engine.Game (name='Game', player=None, boards={}, menu={}, cur-
rent_level=None, enable_partial_display=False, par-
tial_display_viewport=None, partial_display_focus=None,
mode=90000003, user_update=None, input_lag=0.01,
user_update_paused=None)
```

Bases: pygamelib.base.PglBaseObject

A class that serve as a game engine.

This object is the central system that allow the management of a game. It holds boards (see *pygamelib*. engine.Board), associate it to level, takes care of level changing, etc.

Note: The game object has an object_library member that is always an empty array except just after loading a board. In this case, if the board have a "library" field, it is going to be used to populate object_library. This library is accessible through the Game object mainly so people have access to it across different Boards during level design in the editor. That architecture decision is debatable.

Note: The constructor of Game takes care of initializing the terminal to properly render the colors on Windows.

Important: The Game object automatically assumes ownership over the Player.

__init__ (name='Game', player=None, boards={}, menu={}, current_level=None, enable_partial_display=False, partial_display_viewport=None, partial_display_focus=None, mode=90000003, user_update=None, input_lag=0.01, user_update_paused=None)

Parameters

- name (str) The Game name.
- boards (dict) A dictionary of boards with the level number as key and a board reference as value.
- **menu** (dict) A dictionary of menus with a category (str) as key and another dictionary (key: a shortcut, value: a description) as value.
- **current_level** (*int*) The current level.
- **enable_partial_display** (bool) A boolean to tell the Game object to enable or not partial display of boards. Default: False.
- partial_display_viewport (list) A 2 int elements array that gives the radius of the partial display in number of row and column. Please see display_around().
- partial_display_focus (BoardItem) The object that is going to be the center of the view when the board is displayed.
- mode (int) The mode parameter configures the way the run() method is going to behave. The default value is constants.MODE_TBT. TBT is short for "Turn By Turn". In that mode, the Game object wait for an user input before looping. Exactly like when you wait for user input with get_key(). The other possible value is constants.MODE_RT. RT stands for "Real Time". In that mode, the Game object waits for a minimal amount of time (0.01 i.e 100 FPS, configurable through the input_lag parameter) in order to get the input from the user and call the update function right away. This parameter is *only* useful if you use Game.run().

- **user_update** (function) A reference to the main program update function. The update function is called for each new frame. It is called with 3 parameters: the game object, the user input (can be None) and the elapsed time since last frame.
- user_update_paused (function) A reference to the update function called when the game is paused. It is called with the same 3 parameters than the regular update function: the game object, the user input (can be None) and the elapsed time since last frame. If not specified, the regular update function is called but nothing is done regarding NPCs, projectiles, animations, etc.
- input_lag (float/int) The amount of time the run() function is going to wait for a user input before returning None and calling the update function. Default is 0.01.

Methods

init([name, player, boards, menu,])	
	param name The Game name.
actuate_npcs(level_number[, elapsed_time])	Actuate all NPCs on a given level
actuate_projectiles(level_number[,	Actuate all Projectiles on a given level
	Actuate an Projectnes on a given level
elapsed_time])	Add a board for the level number.
add_board(level_number, board)	
add_menu_entry(category, shortcut, message)	Add a new entry to the menu.
add_npc(level_number, npc[, row, column,])	Add a NPC to the game.
add_projectile(level_number, projectile[,	Add a Projectile to the game.
])	
animate_items(level_number[, elapsed_time])	That method goes through all the BoardItems of a
	given map and call Animation.next_frame().
attach(observer)	Attach an observer to this instance.
change_level(level_number)	Change the current level, load the board and place
	the player to the right place.
clear_screen()	Clear the whole screen (i.e: remove everything writ-
	ten in terminal)
clear_session_logs()	Delete all the log lines from the logs.
config(section)	Get the content of a previously loaded configuration
	section.
create_config(section)	Initialize a new config section.
current_board()	This method return the board object corresponding
_	to the current level.
delete_all_levels()	Delete all boards and their associated levels from the
	game object.
delete_level(lvl_number)	Delete a level and its associated Board from the game
(object.
	Delete an entire category from the menu.
detach(observer)	Detach an observer from this instance.
display_board()	Display the current board.
display_menu(category[, orientation, paginate])	Display the menu.
display_player_stats([life_model,	Display the player name and health.
void_model])	-17 F7
get_board(level_number)	This method returns the board associated with a level
9	number.
get_key()	Reads the next key-stroke returning it as a string.
300_no1()	Continued on next page
	Continued on next page

444 Chapter 3. Tutorials

Table 86 – continued from previous page

<pre>get_menu_entry(category, shortcut)</pre>	Get an entry of the menu.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
<pre>insert_board(level_number, board)</pre>	Insert a board for the level number.
instance(*args, **kwargs)	Returns the instance of the Game object
load_board(filename[, lvl_number])	Load a saved board
load_config(filename, section)	Load a configuration file from the disk.
<pre>move_player(direction[, step])</pre>	Easy wrapper for Board.move().
neighbors([radius, obj])	Get a list of neighbors (non void item) around an ob-
	ject.
notify([modifier])	Notify all the observers that a change occurred.
pause()	Set the game engine state to PAUSE.
remove_npc(level_number, npc)	This methods remove the NPC from the level in pa-
	rameter.
run()	New in version 1.2.0.
save_board(lvl_number, filename)	Save a board to a JSON file
<pre>save_config(section, filename, append)</pre>	Save a configuration section.
session_log(line)	Add a line to the session logs.
session_logs()	Return the complete session logs since instantiation.
start()	Set the game engine state to RUNNING.
stop()	Set the game engine state to STOPPED.
store_screen_position(row, column)	Store the screen position of the object.
update_menu_entry(category, shortcut, mes-	Update an entry of the menu.
sage)	

Attributes

screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
state	Get/set the state of the game.

actuate_npcs (level_number, elapsed_time=0.0)

Actuate all NPCs on a given level

This method actuate all NPCs on a board associated with a level. At the moment it means moving the NPCs but as the Actuators become more capable this method will evolve to allow more choice (like attack use objects, etc.)

When all NPCs have been successfully actuated, the observers are notified of the change with the pygamelib.engine.Game.actuate_npcs:npcs_actuated event. Their is value passed for that event.

Parameters

- level_number (int) The number of the level to actuate NPCs in.
- **elapsed_time** (float) The amount of time that passed since last call. This parameter is not mandatory.

Example:

mygame.actuate_npcs(1)

Note: This method only move NPCs when their actuator state is RUNNING. If it is PAUSED or STOPPED, the NPC is not moved.

Note: Since version 1.2.0 it's possible for a Movable item to have different vertical and horizontal movement steps, so actuate_npc respect that by integrating the steps with a unit direction vector. It should be completely transparent and you should not expect any change. Just more movement freedom. If you do experience issues, please report a bug.

Note: Since version 1.2.0 and the appearance of the realtime mode, we have to account for movement speed. This method does it.

actuate_projectiles (level_number, elapsed_time=0.0)

Actuate all Projectiles on a given level

This method actuate all Projectiles on a board associated with a level. This method differs from actuate_npcs() as some logic is involved with projectiles that NPC do not have. This method decrease the available range by projectile.step each time it's called. It also detects potential collisions. If the available range falls to 0 or a collision is detected the projectile hit_callback is called.

This method respects the Projectile.collision_exclusions parameter and does not register collisions with objects of a type present in that list.

Important: In this method, projectiles do not collide with overlappable items. If you want to detect collisions with overlappable objects, please implement your own projectile actuation method.

Parameters

- **level_number** (*int*) The number of the level to actuate Projectiles in.
- **elapsed_time** (float) The amount of time that passed since last call. This parameter is not mandatory.

When all Projectiles have been successfully actuated, the observers are notified of the change with the pygamelib.engine.Game.actuate_projectiles:projectiles_actuated event. Their is value passed for that event.

Example:

```
mygame.actuate_projectiles(1)
```

Note: This method only move Projectiles when their actuator state is RUNNING. If it is PAUSED or STOPPED, the Projectile is not moved.

Important: Please have a look at the *pygamelib.board_items.Projectile.hit()* method for more information on the projectile hit mechanic.

add_board (level_number: int, board: pygamelib.engine.Board) \rightarrow None Add a board for the level number.

446 Chapter 3. Tutorials

This method associate a Board (pygamelib.engine.Board) to a level number.

If the partial display is enabled at Game level (i.e: partial_display_viewport is not None and enable_partial_display is True), this method propagate the settings to the board automatically. Same for partial_display_focus.

Example:

```
game.add_board(1,myboard)
```

Parameters

- **level_number** (*int*) the level number to associate the board to.
- board (pygamelib.engine.Board) a Board object corresponding to the level number.

Raises *PglInvalidTypeException* – If either of these parameters are not of the correct type.

add_menu_entry (category, shortcut, message, data=None)

Add a new entry to the menu.

Deprecated since version 1.3.0: This function will be removed in version 1.4.0

Add another shortcut and message to the specified category.

Categories help organize the different sections of a menu or dialogues.

Parameters

- **category** (*str*) The category to which the entry should be added.
- **shortcut** (*str*) A shortcut (usually one key) to display.
- message (various) a message that explains what the shortcut does.
- data a data that you can get from the menu object.

The shortcut and data is optional.

Example:

```
game.add_menu_entry('main_menu','d','Go right',constants.RIGHT)
game.add_menu_entry('main_menu',None,'-----')
game.add_menu_entry('main_menu','v','Change game speed')
```

add_npc (level_number, npc, row=None, column=None, layer=None, auto_layer=True)

Add a NPC to the game. It will be placed on the board corresponding to the level_number. If row and column are not None, the NPC is placed at these coordinates. Else, it's randomly placed in an empty cell.

Example:

```
game.add_npc(1,my_evil_npc,5,2)
```

Parameters

- **level_number** (*int*) the level number of the board.
- npc (pygamelib.board_items.NPC) the NPC to place.
- row (int) the row coordinate to place the NPC at.
- **column** (*int*) the column coordinate to place the NPC at.

If either of these parameters are not of the correct type, a PglInvalidTypeException exception is raised.

Important: If the NPC does not have an actuator, this method is going to affect a pygamelib.actuators.RandomActuator() to npc.actuator. And if npc.step == None, this method sets it to 1

add projectile (level number, projectile, row=None, column=None)

Add a Projectile to the game. It will be placed on the board corresponding to level_number. Neither row nor column can be None.

Example:

```
game.add_projectile(1, fireball, 5, 2)
```

Parameters

- **level_number** (*int*) the level number of the board.
- projectile (Projectile) the Projectile to place.
- row (int) the row coordinate to place the Projectile at.
- **column** (*int*) the column coordinate to place the Projectile at.

If either of these parameters are not of the correct type, a PgIInvalidTypeException exception is raised.

Important: If the Projectile does not have an actuator, this method is going to affect pygamelib.actuators.RandomActuator(moveset=[RIGHT]) to projectile.actuator. And if projectile.step == None, this method sets it to 1.

animate_items (level_number, elapsed_time=0.0)

That method goes through all the BoardItems of a given map and call Animation.next_frame().

When all items have been successfully animated, the observers are notified of the change with the pygamelib.engine.Game.animate_items:items_animated event. Their is value passed for that event.

Parameters

- **level_number** (*int*) The number of the level to animate items in.
- **elapsed_time** (float) The amount of time that passed since last call. This parameter is not mandatory.

Raise PglInvalidLevelException PglInvalidTypeException

Example:

```
mygame.animate_items(1)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

change_level ($level_number: int) \rightarrow None$

Change the current level, load the board and place the player to the right place.

Example:

```
game.change_level(1)
```

Parameters level_number (int) – the level number to change to.

Raises base. PglInvalidTypeException - If parameter is not an int.

clear_screen()

Clear the whole screen (i.e: remove everything written in terminal)

Deprecated since version 1.2.0: Starting 1.2.0 we are using the pygamelib.engine. Screen object to manage the screen. That function is a simple forward and is kept for backward compatibility only. You should use Game.screen.clear()

$clear_session_logs() \rightarrow None$

Delete all the log lines from the logs.

Example:

```
game = Game.instance()
game.clear_logs()
```

Note: The session log system is nothing more than a list to do your "debug prints". If you want a real logging system, please use Python logging module.

```
config (section: str = 'main') \rightarrow dict
```

Get the content of a previously loaded configuration section.

Parameters section (str) – The name of the section.

Example:

```
if mygame.config('main')['pgl-version-required'] < 10200:
    print('The pygamelib version 1.2.0 or greater is required.')
    exit()</pre>
```

$create_config(section: str) \rightarrow None$

Initialize a new config section.

The new section is a dictionary.

Parameters section (str) – The name of the new section.

Example:

```
if mygame.config('high_scores') is None:
    mygame.create_config('high_scores')
mygame.config('high_scores')['first_place'] = mygame.player.name
```

current_board() → pygamelib.engine.Board

This method return the board object corresponding to the current_level.

Example:

```
game.current_board().display()
```

If current_level is set to a value with no corresponding board a PglException exception is raised with an invalid_level error.

delete_all_levels()

Delete all boards and their associated levels from the game object.

You might want to think twice before using that function...

Example:

```
game.delete_all_levels()
```

delete_level (lvl_number: int = None)

Delete a level and its associated Board from the game object.

Both the level and the board can't be used after that (unless they are reloaded or replaced of course).

Parameters lvl_number (int) – The number of the level to remove.

Raises

- base.PglInvalidTypeException If parameter is not an int.
- base.PglInvalidLevelException If parameter is not a valid level.

Example:

```
my_game.delete_level(1)
```

delete_menu_category (category=None)

Delete an entire category from the menu.

Deprecated since version 1.3.0: This function will be removed in version 1.4.0

That function removes the entire list of messages that are attached to the category.

Parameters category (str) – The category to delete.

Raises PglInvalidTypeException – If the category is not a string

Important: If the entry have no shortcut it's advised not to try to update unless you have only one NoneType as a shortcut.

Example:

```
game.add_menu_entry('main_menu','d','Go right')
game.update_menu_entry('main_menu','d','Go LEFT',constants.LEFT)
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

display board()

Display the current board.

The behavior of that function is dependant on how you configured this object. If you set enable_partial_display to True AND partial_display_viewport is set to a correct value, it will call Game.current_board().display_around() with the correct parameters. The partial display will be centered on the player (Game.player). Otherwise it will just call Game.current_board().display().

If the player is not set or is set to constants.NO_PLAYER partial display won't activate automatically.

Example:

```
mygame.enable_partial_display = True
# Number of rows, number of column (on each side, total viewport
# will be 20x20 in that case).
mygame.partial_display_viewport = [10, 10]
# This will call Game.current_board().display_around()
mygame.display()
mygame.enable_partial_display = False
# This will call Game.current_board().display()
mygame.display()
```

display_menu (category, orientation=30000010, paginate=10)

Display the menu.

Deprecated since version 1.3.0: This function will be removed in version 1.4.0

This method display the whole menu for a given category.

Parameters

- category (str) The category to display. Mandatory parameter.
- orientation (pygamelib.constants) The shortcut of the entry you want to get.
- paginate (int) pagination parameter (how many items to display before changing line or page).

Example:

```
game.display_menu('main_menu')
game.display_menu('main_menu', constants.ORIENTATION_HORIZONTAL, 5)
```

Deprecated since version This: method is completely deprecated and not even compatible with the Screen Buffer system. **It will be removed in 1.4.0**.

This method print the Player name, a health bar (20 blocks of life_model). When life is missing the complement (20-life missing) is printed using void_model. It also display the inventory value as "Score".

Parameters

• life_model (str) - The character(s) that should be used to represent the *remaining* life.

• **void_model** (str) – The character(s) that should be used to represent the *lost* life.

Note: This method might change in the future. Particularly it could take a template of what to display.

get_board (*level_number: int*) → pygamelib.engine.Board

This method returns the board associated with a level number: :param level_number: The number of the level. :type level_number: int

Raises PglInvalidTypeException – if the level_number is not an int.

Example:

```
level1_board = mygame.get_board(1)
```

static get_key()

Reads the next key-stroke returning it as a string.

Example:

```
key = Utils.get_key()
if key == Utils.key.UP:
    print("Up")
elif key == "q"
    exit()
```

Note: See *readkey* documentation in *readchar* package.

get menu entry(category, shortcut)

Get an entry of the menu.

Deprecated since version 1.3.0: This function will be removed in version 1.4.0

This method return a dictionnary with 3 entries:

- · shortcut
- message
- data

Parameters

- **category** (str) The category in which the entry is located.
- **shortcut** (str) The shortcut of the entry you want to get.

Returns The menu entry or None if none was found

Return type dict

Example:

```
ent = game.get_menu_entry('main_menu','d')
game.move_player(int(ent['data']),1)
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

 $\verb"insert_board" (\textit{level_number: int, board: pygamelib.engine.Board}) \rightarrow None$

Insert a board for the level number.

This method does basically the same thing than add_board() except that if the level number is already associated it re-affect the numbers down.

Example:

```
game.insert_board(1,myboard_1)
# level number 1 is associated with myboard_1
game.insert_board(2,myboard_2)
# level number 1 is associated with myboard_1
# level number 2 is associated with myboard_2
game.insert_board(2,myboard_3)
# level number 1 is associated with myboard_1
# level number 2 is now associated with myboard_1
# level number 3 is associated with myboard_3
# level number 3 is associated with myboard_2
```

Parameters

- **level_number** (*int*) the level number to associate the board to.
- board (pygamelib.engine.Board) a Board object corresponding to the level number.

Raises *PglInvalidTypeException* – If either of these parameters are not of the correct type.

```
classmethod instance(*args, **kwargs)
```

Returns the instance of the Game object

Creates a Game object on first call an then returns the same instance on further calls

Returns Instance of Game object

```
load_board (filename, lvl_number=0)
```

Load a saved board

Load a Board saved on the disk as a JSON file. This method creates a new Board object, populate it with all the elements (except a Player) and then return it.

If the filename argument is not an existing file, the open function is going to raise an exception.

This method, load the board from the JSON file, populate it with all BoardItem included, check for sanity, init the board with BoardItemVoid and then associate the freshly created board to a lvl_number. It then create the NPCs and add them to the board.

Parameters

- **filename** (str) The file to load
- **1v1 number** (*int*) The level number to associate the board to. Default is 0.

Returns a newly created board (see pygamelib.engine.Board)

Example:

```
mynewboard = game.load_board( 'awesome_level.json', 1 )
game.change_level( 1 )
```

load_config (filename: str, section: str = 'main') \rightarrow dict

Load a configuration file from the disk. The configuration file must respect the INI syntax. The goal of these methods is to simplify configuration files management.

Parameters

- **filename** (str) The filename to load. does not check for existence.
- **section** (*str*) The section to put the read config file into. This allow for multiple files for multiple purpose. Section is a human readable unique identifier.

Raises

- FileNotFoundError If filename is not found on the disk.
- json.decoder.JSONDecodeError If filename could not be decoded as JSON.

Returns The parsed data.

Return type dict

Warning: breaking changes: before v1.1.0 that method use to load file using the configpraser module. This have been dumped in favor of json files. Since that methods was apparently not used, there is no backward compatibility.

Example:

```
mygame.load_config('game_controls.json','game_control')
```

move_player (direction, step=1)

Easy wrapper for Board.move().

Example:

```
mygame.move_player(constants.RIGHT,1)
```

neighbors (radius=1, obj=None)

Get a list of neighbors (non void item) around an object.

This method returns a list of objects that are all around an object between the position of an object and all the cells at **radius**.

Parameters

- radius (int) The radius in which non void item should be included
- **object** (pygamelib.board_items.BoardItem) The central object. The neighbors are calculated for that object. If None, the player is the object.

Returns A list of BoardItem. No BoardItemVoid is included.

Raises PglInvalidTypeException - If radius is not an int.

Example:

```
for item in game.neighbors(2):
    print(f'{item.name} is around player at coordinates '
          '({item.pos[0]}, {item.pos[1]})')
```

notify (*modifier=None*, *attribute:* str = None, *value:* Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

pause()

Set the game engine state to PAUSE.

Example:

```
mygame.pause()
```

remove_npc (level_number, npc)

This methods remove the NPC from the level in parameter.

Parameters

- **level** (*int*) The number of the level from where the NPC is to be removed.
- npc (NPC) The NPC object to remove.

Example:

```
mygame.remove_npc(1, dead_npc)
```

run()

New in version 1.2.0.

The run() method act as the main game loop and does a number of things for you:

- 1. It grabs the user input. If the Game object is configured with MODE_TBT (the default), nothing happen until the user hit a key. If the mode is set to MODE_RT, it will wait for input_lag secondes for a user input before going to step 3.
- 2. It calculate the elapsed time between 2 frames.
- 3. Accumulates the elapsed time in the player dtmove variable (if there is a player object configured)
- 4. It sets the cursor position to 0,0 (meaning that your user_update function will draw on top of the previously drawn window). The Board.display() and Board.display_around() method clean the end of their line.

- 5. It calls the user_update function with 3 parameters: the game object, the key hit by the user (it can be None) and the elapsed time between to calls.
- 6. Clears the end of the screen.
- 7. Actuates NPCs (If there is at least one Board manage by Game).
- 8. Actuates projectiles (If there is at least one Board manage by Game).
- 9. Animates items (If there is at least one Board manage by Game).

On the subject of particle emitters, the *Board* object automatically update the ones that are attached to BoardItems. For all other particle emitters you need to call the update method of the emitters yourself (for now).

In version 1.2.X, there was a bug when the game was paused. In that case nothing was happening anymore. The user update function was not called and events were not processed. On top of that it was impossible to use run() without associating a board object with a level. Starting with version 1.3.0, it is now possible to use run() without associating a board object with a level. There is also a new parameter to the constructor (user_update_paused) that allows you to specify a function that will be called when the game is paused. This function will be called with the same 3 parameters than the regular update function: the game object, the user input (can be None) and the elapsed time since last frame.

Important: If you try to set the game state to PAUSED and the user_update_paused function is not defined, a notification will be issued and the game will continue to run. The notification message is pygamelib.engine.Game.run:PauseNotAvailable

Raises PglInvalidTypeException, PglInvalidTypeException

Example:

```
mygame.run()
```

save board(lvl number, filename)

Save a board to a JSON file

This method saves a Board and everything in it but the BoardItemVoid.

Not check are done on the filename, if anything happen you get the exceptions from open().

Parameters

- **lvl_number** (*int*) The level number to get the board from.
- **filename** (str) The path to the file to save the data to.

Raises

- PglInvalidTypeException If any parameter is not of the right type
- **PglInvalidLevelException** If the level is not associated with a Board.

Example:

```
game.save_board( 1, 'hac-maps/level1.json')
```

If Game.object_library is not an empty array, it will be saved also.

Warning: In version 1.3.0 the *Board* class changed a lot and a layer system has been added. Therefor, boards saved from version 1.3.0+ are *not* compatible with previous version. Previous boards can be loaded (*Game.load_board*() is backward compatible), but when saved they will be converted to the new format.

save_config (section: str = None, filename: str = None, append: bool = False) \rightarrow None Save a configuration section.

Parameters

- **section** (str) The name of the section to save on disk.
- **filename** (str) The file to write in. If not provided it will write in the file that was used to load the given section. If section was not loaded from a file, save will raise an exception.
- append (bool) Do we need to append to the file or replace the content (True = append, False = replace)

Example:

```
mygame.save_config('game_controls', 'data/game_controls.json')
```

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

```
session\_log(line: str) \rightarrow None
```

Add a line to the session logs.

Session logs needs to be activated first.

Parameters line (str) – The line to add to the logs.

Example:

```
game = Game.instance()
game.ENABLE_SESSION_LOGS = True
game.session_log('Game engine initialized')
```

Note: The session log system is nothing more than a list to do your "debug prints". If you want a real logging system, please use Python logging module.

```
session_logs() \rightarrow list
```

Return the complete session logs since instantiation.

Example:

```
game = Game.instance()
game.ENABLE_SESSION_LOGS = True
for line in game.logs():
    print(line)
```

Note: The session log system is nothing more than a list to do your "debug prints". If you want a real logging system, please use Python logging module.

start()

Set the game engine state to RUNNING.

The game has to be RUNNING for actuate_npcs() and move_player() to do anything.

Example:

```
mygame.start()
```

state

Get/set the state of the game.

Parameters value (*int*) – The new state of the game (from the constants module).

Returns The state of the game.

Return type int

The observers are notified of a change of state with the pygamelib.engine.Game.state event. The new state is passed as the value of the event.

stop()

Set the game engine state to STOPPED.

Example:

```
mygame.stop()
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

update_menu_entry (category, shortcut, message, data=None)

Update an entry of the menu.

Deprecated since version 1.3.0: This function will be removed in version 1.4.0

Update the message associated to a category and a shortcut.

Parameters

• **category** (str) – The category in which the entry is located.

- **shortcut** (*str*) The shortcut of the entry you want to update.
- message (various) a message that explains what the shortcut does.
- data a data that you can get from the menu object.

Important: If the entry have no shortcut it's advised not to try to update unless you have only one NoneType as a shortcut.

Example:

```
game.add_menu_entry('main_menu','d','Go right')
game.update_menu_entry('main_menu','d','Go LEFT',constants.LEFT)
```

3.6.3 Inventory

class pygamelib.engine.Inventory(max_size=10, parent=None)
 Bases: pygamelib.base.PglBaseObject

A class that represent the Player (or NPC) inventory.

This class is pretty straightforward: it is an object container, you can add, get and remove items and you can get a value from the objects in the inventory.

On top of that, starting with version 1.3.0, a constraints system has been added. It allows to specify a certain amount of constraints that will be applied to the items when they are added to the inventory.

For the moment, constraints are limited to the number of items with a given type/ name/value (any combination of these three).

When a constraint is violated, the item is not added to the inventory and a notification is broadcasted to the observers of the inventory. A PglInventoryException is also raised with name "constraint_violation" and the constraint details in description.

Note: You can print() the inventory. This is mostly useful for debug as you want to have a better display in your game.

Warning: The *Game* engine and *Player* takes care to initiate an inventory for the player, you don't need to do it.

```
___init__ (max_size=10, parent=None)
```

The constructor takes two parameters: the maximum size of the inventory. And the Inventory owner/parent.

Each *BoardItem* that is going to be put in the inventory has a size (default is 1), the total addition of all these size cannot exceed max size.

Parameters

- max_size (int) The maximum size of the inventory. Default value: 10.
- parent The parent object (usually a BoardItem).

Methods

init([max_size, parent])	The constructor takes two parameters: the maximum
	size of the inventory.
<pre>add_constraint(constraint_name, item_type,</pre>	
)	
add_item(item)	Add an item to the inventory.
attach(observer)	Attach an observer to this instance.
available_space()	Return the available space in the inventory.
clear_constraints()	Remove all constraints from the inventory.
delete_item(name)	Delete THE FIRST item matching the name given in
	argument.
delete_items(name)	Delete ALL items matching the name given in argu-
	ment.
detach(observer)	Detach an observer from this instance.
empty()	Empty the inventory.
<pre>get_item(name)</pre>	Return the FIRST item corresponding to the name
	given in argument.
<pre>get_items(name)</pre>	Return ALL items matching the name given in argu-
	ment.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
items_name()	Return the list of all items names in the inventory.
load(data)	Load serialized data into a new Inventory object.
notify([modifier])	Notify all the observers that a change occurred.
remove_constraint(constraint_name)	
search(query)	Search for objects in the inventory.
serialize()	Serialize the inventory in a dictionary.
size()	Return the cumulated size of the inventory.
store_screen_position(row, column)	Store the screen position of the object.

Attributes

constraints	
items	Return the list of all items in the inventory.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

Add a constraint to the inventory.

Parameters

- **constraint_name** (*str*) the name of the constraint.
- $item_type(str)$ the type of the item.
- item_name (int) the name of the item.
- item_value the value of the item.

460 Chapter 3. Tutorials

• max_number (int) – the maximum number of items that match the item_* parameters that can be in the inventory.

The observers are notified of the addition of the constraint with the pygamelib.engine.Inventory.add_constraint event. The constraint that was added is passed as the value of the event as a dictionnary.

New in version 1.3.0.

add item(item)

Add an item to the inventory.

This method will add an item to the inventory unless:

- it is not an instance of BoardItem,
- you try to add an item that is not pickable,
- there is no more space left in the inventory (i.e. the cumulated size of the inventory + your item.inventory_space is greater than the inventory max_size)
- An existing constraint is violated.

```
Parameters item (BoardItem) – the item you want to add
```

Returns The index of the newly added item in the inventory or None if the item could not be added.

Return type intlNone

```
Raise PalInventoryException, PalInvalidTypeException
```

When an item is successfully added, the observers are notified of the change with the pygamelib.engine.Inventory.add_item event. The item that was added is passed as the value of the event.

When something goes wrong exceptions are raised. The following exceptions can be raised (PglInventoryException):

- not_pickable: The item you try to add is not pickable.
- not_enough_space: There is not enough space left in the inventory.
- constraint_violation: A constraint is violated.

A PqlInvalidTypeException is raised when the item you try to add is not a BoardItem.

Example:

```
item = Treasure(model=graphics.Models.MONEY_BAG, size=2, name='Money bag')
try:
    mygame.player.inventory.add_item(item)
expect PglInventoryException as e:
    if e.error == 'not_enough_space':
        print(f"Impossible to add {item.name} to the inventory, there is no"
        "space left in it!")
        print(e.message)
elif e.error == 'not_pickable':
        print(e.message)
```

Note: In versions prior to 1.3.0, the inventory object was changing the name of the item if another item with the same name was already in the inventory. This is (fortunately) not the case anymore. The Inventory

class does NOT modify the items that are stored into it anymore.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) - An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

available_space() \rightarrow int

Return the available space in the inventory.

That is to say, Inventory.max_size - Inventory.size().

The returned number is comprised between 0 and Inventory.max_size.

Returns The size as an int.

Return type int

Example:

```
method()
```

clear_constraints()

Remove all constraints from the inventory.

The observers are notified with the pygamelib.engine.Inventory.clear_constraints event. The value is set to None for this event.

New in version 1.3.0.

constraints

Return the list of all constraints in the inventory.

Returns a list of constraints (dict)

Return type list

Example:

```
for cstr in game.player.inventory.constraints:
    print(f" - {cstr[name]}")
```

delete item(name)

Delete THE FIRST item matching the name given in argument.

Parameters name (str) – the name of the items you want to delete.

When an item is successfully removed, the observers are notified of the change with the pygamelib.engine.Inventory.delete_item event. The item that was deleted is passed as the value of the event.

Example:

```
mygame.player.inventory.delete_item('heart_1')
```

Important: Starting with version 1.3.0 this method does not raise exceptions anymore. It's behavior also changed from deleting a precise item to deleting the first one that matches the name.

delete_items (name)

Delete ALL items matching the name given in argument.

Parameters name (str) – the name of the items you want to delete.

The observers are notified of each deletion with the pygamelib.engine.Inventory.delete_item event. The item that was deleted is passed as the value of the event.

Example:

```
mygame.player.inventory.delete_items('heart_1')
```

New in version 1.3.0.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

empty()

Empty the inventory.

The observers are notified that the Inventory has been emptied with the pygamelib.engine.Inventory.empty event. Nothing is passed as the value.

Example:

```
if inventory.size() > 0:
   inventory.empty()
```

get_item(name)

Return the FIRST item corresponding to the name given in argument.

Parameters name (str) – the name of the item you want to get.

Returns An item.

Return type BoardItem | None

Example:

```
life_container = mygame.player.inventory.get_item('heart_1')
if isinstance(life_container,GenericActionableStructure):
    life_container.action(life_container.action_parameters)
```

Note: Please note that the item object reference is returned but nothing is changed in the inventory. The item hasn't been removed.

Important: Starting with version 1.3.0 this method does not raise exceptions anymore. Instead it returns None if no item is found. It's behavior also changed from returning a precise item to the first one that matches the name.

get_items (name)

Return ALL items matching the name given in argument.

Parameters name (str) – the name of the item you want to get.

Returns An array of items.

Return type list

Example:

```
for life_container in mygame.player.inventory.get_items('heart_1'):
    if isinstance(life_container, GenericActionableStructure):
        life_container.action(life_container.action_parameters)
```

Note: Please note that the item object reference is returned but nothing is changed in the inventory. The item hasn't been removed.

New in version 1.3.0.

handle_notification(subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

items

Return the list of all items in the inventory.

Returns a list of BoardItem

Return type list

Example:

```
for item in game.player.inventory.items:
    print(f"This is a mighty item: {item.name}")
```

items name()

Return the list of all items names in the inventory.

Returns a list of string representing the items names.

Return type list

classmethod load(data: dict)

Load serialized data into a new Inventory object.

Parameters data (dict) - The serialized data

Returns A new Inventory object.

Return type Inventory

New in version 1.3.0.

Example:

```
my_player.inventory = Inventory.load(data)
```

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

remove_constraint (constraint_name: str)

Remove a constraint from the inventory.

Parameters constraint_name (str) - the name of the constraint.

The observers are notified of the removal of the constraint with the pygamelib.engine.Inventory.remove_constraint event. The constraint that was removed is passed as the value of the event as a dictionnary.

New in version 1.3.0.

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

3.6. engine 465

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

search (query)

Search for objects in the inventory.

All objects that matches the query are going to be returned. Search is performed on the name and type of the object.

Parameters query – the query that items in the inventory have to match to be returned

Returns a list of BoardItems.

Return type list

Example:

```
for item in game.player.inventory.search('mighty'):
    print(f"This is a mighty item: {item.name}")
```

serialize()

Serialize the inventory in a dictionary.

Returns The serialized data.

Return type dict

New in version 1.3.0.

Example:

```
json.dump(my_inventory.serialize(), out_file)
```

size()

Return the cumulated size of the inventory. It can be used in the UI to display the size compared to max_size for example.

Returns size of inventory

Return type int

Example:

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

value()

Return the cumulated value of the inventory. It can be used for scoring for example.

Returns value of inventory

Return type int

Example:

```
if inventory.value() >= 10:
    print('Victory!')
    break
```

3.6.4 Screen

```
class pygamelib.engine.Screen (width: int = None, height: int = None)
Bases: pygamelib.base.PglBaseObject
```

The screen object is pretty straightforward: it is an object that allow manipulation of the screen.

Warning: Starting with version 1.3.0 the terminal parameter has been removed. The Screen object now takes advantage of base.Console.instance() to get a reference to a blessed.Terminal object.

Version 1.3.0 introduced a new way of managing the screen. It rely on an internally managed display buffer that allows for easier positioning and more regular rendering. This comes at a cost though as the performances takes a hit. The screen should still be able to be refreshed between 50 and 60+ times per seconds (and still around 30 times per second within a virtual machine). These numbers obviously depends on the terminal used, the screen size and the content to display.

This change introduce two ways of displaying things on the screen:

- The **Improved Screen Management** stack (referred to as ISM later in the doc).
- The Legacy Direct Display stack.

It is safer to consider them mutually incompatible. In reality the **Improved Screen Management** will always use the whole display but you can use the methods from the **Direct Display** stack to write over the buffer. It is really **NOT** advised.

We introduced the **Improved Screen Management** stack because the direct display is messy and does not allow us to do what we want in term of positioning, UI, etc.

A typical usage consist of:

- Placing elements on the screen with place ()
- Update the screen with update ()

That's it! The screen maintain its own state and knows when to re-render the display buffer. You don't need to manually call render(). This helps with performances as the frame buffer is only rendered when needed.

Example:

```
screen = Screen()
# The next 3 lines do the same thing: display a message centered on the screen.
# Screen Buffer style
```

(continues on next page)

3.6. engine 467

(continued from previous page)

```
screen.place('This is centered', screen.vcenter, screen.hcenter)
screen.update()
# Direct Display style
screen.display_at('This is centered', screen.vcenter, screen.hcenter)
# The rest of this example uses the Screen Buffer (because placing a Board
# anywhere on the Screen is not supported by the Direct Display stack).
# delete the previous message and place a Board at the center of the screen
screen.delete(screen.vcenter, screen.hcenter)
screen.place(
    my_awesome_board,
    screen.vcenter - int(my_awesome_board.height/2),
    screen.hcenter - int(my_awesome_board.width/2)
)
screen.update()
```

Precisions about the Improved Screen Management stack:

You don't need to know how the frame buffer works to use it. However, if you are interested in more details, here they are.

The Improved Screen Management stacks uses a double numpy buffer to represent the screen. One buffer is used to place elements as objects (that's the buffer managed by place() or delete()). It is never directly printed to the screen. It is here to simplify screen maintenance. This buffer is called the **display buffer**. It is practical to use to place, move and delete elements on the screen space. But as said before it cannot be directly printed to the screen. It needs to be rendered first.

For example, if you want to use a sprite on a title screen and want to move it around (or animate the screen). Normally (i.e with Direct Display) you would display the sprite at a specific position and then would either call <code>clear()</code> or overwrite all the sprite with spaces to erase and replace and/or move it. And that's very slow.

With the **Improved Screen Management** you place() the sprite and then just delete() it. And since it is only one object reference it is a very fast operation (we only place or delete one cell of the buffer).

When update() is called, it first look at the state of the buffers and call render() if needed (i.e. if something has change in the display buffer). The buffers are only rendered when needed.

When render() is called it goes through the display buffer and render each elements transforming it into a printable sequence that is stored in the frame buffer. The rendering is done from the bottom right corner of the screen to the top left corner. This allows for cleaning junk characters at no additional cost.

TL;DR: The **display buffer** hold the objects placed on the screen while the **frame buffer** hold the rendered representation of the display buffer.

The Screen object also inherits from the PglBaseObject and if the object that is place()-ed is an instance of PglBaseObject, the screen will automatically attach itself to the object. When notified of a change it will trigger a render cycle before the next update.

In terms of performances, depending on your terminal emulator and CPU you will most certainly achieve over 30 FPS. Here are a couple of benchmark results:

- On an Intel Core i7 @ 4.20 GHz: 50 to 70 FPS.
- On an AMD Ryzen 9 5950X @ 4.80 GHz: 60 to 100 FPS.

The new **Improved Screen Management** is faster than the legacy stack in most of the cases. The only case when the legacy Direct Display stack might be faster is in the case of a game or application with only simple ASCII characters and not a lot of things to display.

Here are some compiled benchmark results of both of systems over 150 runs:

Benchmark	Improved Screen Man-	Legacy Direct Display
	agement	
Sprite (place, render and update screen), Sprite	10.0 msec. or 71 FPS	380.0 msec. or 3 FPS
size: 155x29		
Sprite 200 updates	620.0 msec. or 76 FPS	9830.0 msec. or 20 FPS
Phase 1 - 500 frames. Single board avg load	11.02 msec. per frame or	12.65 msec. per frame or
	91 FPS	79 FPS
Phase 2 - 500 frames. Dual board high load	18.18 msec. per frame or	28.34 msec. per frame or
	55 FPS	35 FPS
Overall - 1000 frames.	14.60 msec. per frame or	20.49 msec. per frame or
	68 FPS	49 FPS

You can use the 2 benchmark scripts to compare on your system:

- · benchmark-screen-buffer.py
- · benchmark-screen-direct-display.py

The frame buffer system has been tested on the following terminals:

- xterm-256color
- Konsole
- Kitty
- Alacritty
- GNOME Terminal

Performances are consistants across the different terminals. The only exception is the GNOME Terminal, which is slower than the others (about 20~30 % slower).

```
\_init\_ (width: int = None, height: int = None)
```

The constructor takes the following (optional) parameters.

Parameters

- width (int) The width of the screen.
- **height** (*int*) The height of the screen.

Setting any of these parameters fixes the screen size regardless of the actual console/terminal resolution. Leaving any of these parameters unset will let the constructor use the actual console/terminal resolution instead.

Please have a look at the examples for more on this topic.

Example:

```
# Let's assume a terminal resolution of 170(width)x75(height).
screen = Screen()
# Next line display: "Screen width=170 height=75"
print(f"Screen width={screen.width} height={screen.height}")
screen = Screen(50)
# Next line display: "Screen width=50 height=75"
print(f"Screen width={screen.width} height={screen.height}")
screen = Screen(height=50)
# Next line display: "Screen width=170 height=50"
print(f"Screen width={screen.width} height={screen.height}")
screen = Screen(50, 50)
```

(continues on next page)

3.6. engine 469

(continued from previous page)

```
# Next line display: "Screen width=50 height=50"
print(f"Screen width={screen.width} height={screen.height}")
```

Methods

init(width, height)	The constructor takes the following (optional) pa-
	rameters.
attach(observer)	Attach an observer to this instance.
clear()	This methods clear the screen.
clear_buffers()	This methods clear the Screen's buffers (both display
	and frame buffer).
clear_frame_buffer()	This methods clear the frame buffer (but not the dis-
	play buffer).
delete([row, column])	Delete a element on screen.
detach(observer)	Detach an observer from this instance.
display_at(text[, row, column, clear_eol,])	Displays text at a given position.
display_line(*text[, end, file, flush])	A wrapper to Python's print() builtin function except
	it will always add an ANSI sequence to clear the end
	of the line.
display_sprite(sprite[, filler, file, flush])	Displays a sprite at the current cursor position.
<pre>display_sprite_at(sprite[, row, column,])</pre>	Displays a sprite at a given position.
force_render()	Force the immediate rendering of the display buffer.
force_update()	Same as force_render() but also force the im-
	mediate screen update.
get(row, column)	Get an element from the display buffer at the speci-
	fied screen coordinates.
handle_notification(subject[, attribute,	When a Screen object is notified, it set the display
value])	buffer to be rendered before the next update.
notify([modifier])	Notify all the observers that a change occurred.
place([element, row, column, rendering_pass])	Place an element on the screen.
render()	Render the display buffer into the frame buffer.
store_screen_position(row, column)	Store the screen position of the object.
trigger_rendering()	Trigger the frame buffer for rendering at the next up-
	date.
update()	Update the screen.

Attributes

buffer	The buffer property return a numpy.array as a
	writable frame buffer.
hcenter	Return the horizontal center of the screen as an int.
height	This property returns the height of the terminal win-
	dow in number of characters.
need_rendering	This property return True if the display buffer has
	been updated since the last rendering cycle and the
	screen needs to re-render the frame buffer.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
	Continued on next nage

Continued on next page

Table 91 – continued from previous page

vcenter	Return the vertical center of the screen as an int.
width	This property returns the width of the terminal win-
	dow in number of characters.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PqlBaseObject) - An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

buffer

The buffer property return a numpy.array as a writable frame buffer.

The buffer is a 2D plane (like a screen) and anything can render in it. However, it is recommended to place objects through Screen.place() and update the screen with Screen.update() (update calls render() if needed and do the actual display).

Warning: Everything that is stored in the buffer *must* be printable. Each cell of the frame buffer represent a single character on screen, so you need to take care of that when you write into that buffer or you will corrupt the display. If <code>need_rendering</code> returns True, you need to manually call <code>render()</code> before writing anything into the frame buffer. Or else it will be squashed in the next rendering cycle.

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

clear()

This methods clear the screen.

clear buffers()

This methods clear the Screen's buffers (both display and frame buffer).

Make sure that you really want to clear the buffers before doing so, because this is a slow operation.

Once the buffer is cleared nothing is left in it, you have to reposition (place) everything.

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

3.6. engine 471

clear frame buffer()

This methods clear the frame buffer (but not the display buffer). This means that the next time update () is called, rendering will be triggered.

Make sure that you really want to clear the buffers before doing so, because this is a slow operation. It might however be faster than manually update screen cells.

Once the buffer is cleared nothing is left in it, it sets the Screen for a rendering update.

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

delete (row=None, column=None)

Delete a element on screen.

It is important to note that if you placed an element that occupies more than 1 cell, you only have to erase that specific position not the entire area.

Parameters

- **row** (*int*) The row coordinate of the element to delete.
- **column** (*int*) The column coordinate of the element to delete.

Example:

```
board = Board(size=[20,20])
screen.place(board, 2, 2)
# With this we have placed a board at screen coordinates 2,2 and the board
# will display on screen coordinates from 2,2 to 22,22.
# However, to delete the board we don't need to clean all these cells.
# Just the one where we placed the board:
screen.delete(2, 2)
```

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pg1BaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

472 Chapter 3. Tutorials

Displays text at a given position. If clear_eol is True, also clear the end of line. Additionally you can specify all the parameters of a regular print() if you need to.

Parameters

- **text** (str) The text to display. Please note that in that case text is a single string.
- **row** (*int*) The row position in the terminal window.
- **column** (*int*) The column position in the terminal window.
- **clear_eol** (bool) If True this clears the end of the line (everything after the last character displayed by that method).
- end (str) end sub string added to the printed text. Usually a carriage return.
- file (stream) -
- flush (bool) -

Important: The cursor is only moved for printing the text. It is returned to its previous position after.

Note: The position respect the row/column convention across the library. It is reversed compared to the blessed module.

Example:

Note: This method is part of the **Legacy Direct Display** rendering stack and is incompatible with the methods identified as being part of the **Improved Screen Management** stack.

A wrapper to Python's print() builtin function except it will always add an ANSI sequence to clear the end of the line. Making it more suitable to use in a user_update callback.

The reason is that with line with variating length, if you use run() but not clear(), some characters will remain on screen because run(), for performances concerns does not clear the entire screen. It just bring the cursor back to the top left corner of the screen. So if you want to benefit from the increase performances you should use display_line().

Parameters

- ***text** (str/objects) objects that can serialize to str. The ANSI sequence to clear the end of the line is *always* appended to the text.
- end (str) end sub string added to the printed text. Usually a carriage return.
- file (stream) -

3.6. engine 473

• flush (bool) -

Example:

```
screen.display_line(f'This line will display correctly: {elapsed_time}')
# That line will have trailing characters that are not cleared after redraw
# if you don't use clear().
print(f'That one won't: {elapsed_time}')
```

New in version 1.2.0.

Note: This method is part of the **Legacy Direct Display** rendering stack and is incompatible with the methods identified as being part of the **Improved Screen Management** stack.

Displays a sprite at the current cursor position. If a Sprixel is empty, then it's going to be replaced by filler.

Parameters

- **sprite** (*Sprite*) The sprite object to display.
- **filler** (*Sprixel*) A sprixel object to replace all empty sprixels in sprite.
- file (stream) -
- **flush** print() parameter to flush the stream after printing

Examples:

```
screen.display_sprite(panda_sprite)
```

New in version 1.3.0.

Note: This method is part of the **Legacy Direct Display** rendering stack and is incompatible with the methods identified as being part of the **Improved Screen Management** stack.

Displays a sprite at a given position. If a Sprixel is empty, then it's going to be replaced by filler.

Parameters

- **sprite** (*Sprite*) The sprite object to display.
- **row** (*int*) The row position in the terminal window.
- **column** (*int*) The column position in the terminal window.
- **filler** (*Sprixel*) A sprixel object to replace all empty sprixels in sprite.
- file (stream) -
- **flush** (bool) print() parameter to flush the stream after printing

Example:

New in version 1.3.0.

Note: This method is part of the **Legacy Direct Display** rendering stack and is incompatible with the methods identified as being part of the **Improved Screen Management** stack.

force_render()

Force the immediate rendering of the display buffer.

If you just want to mark the frame buffer for rendering before the next update use trigger_rendering() instead.

Example:

```
screen.force_render()
```

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

force_update()

Same as force render () but also force the immediate screen update.

Example:

```
screen.force_update()
```

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

```
get (row: int, column: int)
```

Get an element from the display buffer at the specified screen coordinates.

The element is returned from the display buffer (pre-rendering).

Parameters

- row (int) The row of the element to get.
- **column** (*int*) The column of the element to get.

Example:

```
board = Board(size=[20,20])
screen.place(board, 2, 2)
my_board = screen.get(2,2)
```

3.6. engine 475

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

handle_notification (subject, attribute=None, value=None)

When a Screen object is notified, it set the display buffer to be rendered before the next update.

hcenter

Return the horizontal center of the screen as an int.

Example:

```
screen.place('horizontally centered', 0, screen.hcenter)
```

height

This property returns the height of the terminal window in number of characters.

need rendering

This property return True if the display buffer has been updated since the last rendering cycle and the screen needs to re-render the frame buffer.

It returns False otherwise.

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

place (element=None, row=None, column=None, rendering_pass=1)

Place an element on the screen.

This method places an element in the screen display buffer. The element is then going to be rendered in the frame buffer before being printed on screen.

The following elements can be placed on screen:

· All BoardItem derivatives.

- All BoardComplexItem derivatives.
- · Board object.
- Text objects.
- Sprite objects.
- Sprixel objects.
- · Regular Python str.
- Any object that expose a render_to_buffer() method.

Here is the required signature for render_to_buffer:

render_to_buffer(self, buffer, row, column, buffer_height, buffer_width)

The buffer parameter will always be a numpy array, row and column are the position to render to. Finally buffer_height and buffer_width are the dimension of the buffer.

The buffer is rendered in 2 passes. By default all elements are rendered in pass 1. But if for some reason something needs to be drawn over other elements (like if a dialog/popup is needed for example), the element can be set to be rendered only during the second pass.

Parameters

- element (various) The element to place.
- row (int) The row to render to.
- **column** (*int*) The column to render to.
- **rendering_pass** (*int*) When to render the element. You can have any number of rendering passes but you have to be careful of performances. Higher passes render on top of lower passes. You can see the render passes as plane to write on. The default pass is 1.

Warning: to be rendered on the second+ pass an element *needs* to implement render_to_buffer(...). This excludes all standard types (but not Text). Regular Python strings and object that can be print() can still be used in the first pass.

Example:

```
screen.place(my_sprite, 0, 0)
```

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

render()

Render the display buffer into the frame buffer.

Example:

```
screen.render()
screen.update()
```

3.6. engine 477

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\verb|store_screen_position|| (\textit{row: int, column: int})| \rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

trigger_rendering()

Trigger the frame buffer for rendering at the next update.

Example:

```
screen.trigger_rendering()
```

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Direct Display** stack.

update()

Update the screen. Update means write the frame buffer on screen.

Example:

```
mygame = Game()
sc = core.SpriteCollection.load_json_file('title_screens.spr')
mygame.screen.place(sc['welcome_screen'], 0, 0)
mygame.screen.update()
```

New in version 1.3.0.

Note: This method is part of the **Improved Screen Management** rendering stack and is incompatible with the methods identified as being part of the **Legacy Direct Display** stack.

vcenter

Return the vertical center of the screen as an int.

Example:

```
screen.place('vertically centered', screen.vcenter, 0)
```

width

This property returns the width of the terminal window in number of characters.

3.7 gfx

The gfx (for graphics) sub-module holds all the classes related to the graphics system.

3.7.1 core

Animation

Bases: object

The Animation class is used to give the ability to have more than one model for a BoardItem. A BoardItem can have an animation and all of them that are available to the Game object can be animated through Game.animate_items(lvl_number). To benefit from that, BoardItem.animation must be set explicitly. An animation is controlled via the same state system than the Actuators.

The frames are all stored in a list called frames, that you can access through Animation.frames.

Parameters

- **display_time** (float) The time each frame is displayed
- **auto_replay** (bool) controls the auto replay of the animation, if false once the animation is played it stays on the last frame of the animation.
- **frames** (array[strl Sprixel | Sprite] | SpriteCollection) an array of "frames" (string, sprixel, sprite) or a sprite collection
- animated_object (BoardItem) The object to animate. This parameter is deprecated. Please use parent instead. It is only kept for backward compatibility. The parent parameter always takes precedence over this one.
- parent (BoardItem) The parent object. It is also the object to animate. Important: We cannot animate anything else that BoardItems and subclasses.
- **refresh_screen** (function) The callback function that controls the redrawing of the screen. This function reference should come from the main game.

3.7. qfx 479

Important: When a *SpriteCollection* is used as the *frames* parameter the sprites' names are ordered so the frames are displayed in correct order. This means that 'walk_1' is going to be displayed before 'walk_2'. Otherwise SpriteCollection is un-ordered.

Example

__init__ (display_time=0.05, auto_replay=True, frames=None, animated_object=None, refresh_screen=None, initial_index=None, parent=None)
Initialize self. See help(type(self)) for accurate signature.

Methods

init([display_time, auto_replay,])	Initialize self.
add_frame(frame)	Add a frame to the animation.
current_frame()	Return the current frame.
load(data)	Load a serialized Animation object.
next_frame()	Update the parent's model, sprixel or sprite with the
	next frame of the animation.
pause()	Set the animation state to PAUSED.
play_all()	Play the entire animation once.
remove_frame(index)	Remove a frame from the animation.
reset()	Reset the Animation to the first frame.
search_frame(frame)	Search a frame in the animation.
serialize()	Serialize the Animation object.
start()	Set the animation state to constants.RUNNING.
stop()	Set the animation state to STOPPED.

Attributes

dtanimate	The time elapsed since the last frame was displayed.

add_frame (frame)

Add a frame to the animation.

The frame has to be a string (that includes sprites from the Sprite module and squares from the Utils module).

Raise an exception if frame is not a string.

Parameters frame (strl:class: Sprite'l:class: 'Sprixel') - The frame to add to the animation.

Raise pygamelib.base.PglInvalidTypeException

480 Chapter 3. Tutorials

Example:

```
item.animation.add_frame(Sprite.ALIEN)
item.animation.add_frame(Sprite.ALIEN_MONSTER)
```

current_frame()

Return the current frame.

Example:

```
item.model = item.animation.current_frame()
```

dtanimate

The time elapsed since the last frame was displayed.

classmethod load(data)

Load a serialized Animation object.

Parameters data (dict) – The serialized Animation object.

Returns The loaded Animation object.

Return type Animation

next frame()

Update the parent's model, sprixel or sprite with the next frame of the animation.

That method takes care of automatically resetting the animation if the last frame is reached if the state is constants.RUNNING.

If the state is PAUSED it still update the parent.model and returning the current frame. It does NOT actually go to next frame.

If parent is not a sub class of BoardItem an exception is raised.

Raise PglInvalidTypeException

Example:

```
item.animation.next_frame()
```

Warning: If you use Sprites as frames, you need to make sure your Animation is attached to a <code>BoardComplexItem</code>.

pause()

Set the animation state to PAUSED.

Example:

```
item.animation.pause()
```

play_all()

Play the entire animation once.

That method plays the entire animation only once, there is no auto replay as it blocks the game (for the moment).

If the the state is PAUSED or STOPPED, the animation does not play and the method return False.

If parent is not a sub class of BoardItem an exception is raised.

3.7. gfx 481

If screen_refresh is not defined or is not a function an exception is raised.

```
Raise PglInvalidTypeException
```

Example:

```
item.animation.play_all()
```

remove_frame (index)

Remove a frame from the animation.

That method remove the frame at the specified index and return it if it exists.

If the index is out of bound an exception is raised. If the index is not an int an exception is raised.

Parameters index (int) – The index of the frame to remove.

Return type str

Raise IndexError, PglInvalidTypeException

Example:

```
item.animation.remove_frame( item.animation.search_frame(
    Sprite.ALIEN_MONSTER)
)
```

reset()

Reset the Animation to the first frame.

Example:

```
item.animation.reset()
```

search_frame (frame)

Search a frame in the animation.

That method is returning the index of the first occurrence of "frame".

Raise an exception if frame is not a string.

Parameters frame (str) – The frame to find.

Return type int

Raise PglInvalidTypeException

Example:

```
item.animation.remove_frame(
   item.animation.search_frame(Sprite.ALIEN_MONSTER)
)
```

serialize()

Serialize the Animation object.

The refresh_screen callback function is not serialized. Neither is the parent.

Returns A dictionary containing the Animation object's data.

Return type dict

start()

Set the animation state to constants.RUNNING.

If the animation state is not constants.RUNNING, animation's next_frame() function return the last frame returned.

Example:

```
item.animation.start()
```

stop()

Set the animation state to STOPPED.

Example:

```
item.animation.stop()
```

Font

```
class pygamelib.gfx.core.Font (font_name: str = None, search_directories: list = None)
    Bases: object
```

New in version 1.3.0.

The Font class allow to load and manipulate a pygamelib "font". A font consist of a sprite collection and a configuration file.

If you want to create your own font, please have a look at the font creation tutorial.

In general the Font class is not used directly but passed to a Text object. The text is then rendered using the font.

For performance consideration, it is advised to load the font once and to reuse the object in multiple text objects.

Glyphs are cached (particularly if you change the colors) so it is always beneficial to reuse a font object.

Example:

```
myfont = Font("8bits")
# If you print() mytext, it will use the terminal font and print in cyan.
# But if you Sreen.place() it, it will render using the 8bits sprite font.
mytext = Text("Here's a cool text", fg_color = Color(0,255,255), font=myfont)
```

```
__init__ (font\_name: str = None, search\_directories: list = None) <math>\rightarrow None
```

Parameters

- **font_name** (str) The name of the font to load upon object construction.
- **search_directories** (*list*) A list of directories to search for the font. The items of the list are strings representing a relative or absolute path.

Important: The search directories **must** contain a "fonts" directory, that itself contains the font at the correct format.

Note: Version 1.3.0 comes with a pygamelib specific font called 8bits. It also comes with a handfull of fonts imported from the figlet fonts. Please go to http://www.figlet.org/ for more information.

The conversion script will be made available in the Pygamelib Github organization (https://github.com/pygamelib).

3.7. gfx 483

Example:

```
myfont = Font("8bits")
```

Methods

init(font_name, search_directories)	
	<pre>param font_name The name of the font to load upon object construction.</pre>
glyph(glyph_name, fg_color, bg_color)	This method take a glyph name in parameter and re-
	turns its representation as a Sprite.
load(font_name)	Load a font by name.

Attributes

	Detume the "colorehility" of the fant or angelfod in
colorable	Returns the "colorability" of the font as specified in
	the font config file.
glyphs_map	Returns the glyph map of the font as specified in the
	font config file.
height	Returns the height of the font as specified in the font
	config file.
horizontal_spacing	Returns the horizontal spacing recommended by the
	font (as specified in the font config file).
monospace	Returns if the font is monospace as specified in the
	font config file.
name	Return the name of the font.
scalable	Returns the scalability of the font as specified in the
	font config file.
vertical_spacing	Returns the vertical spacing recommended by the
	font (as specified in the font config file).

colorable

Returns the "colorability" of the font as specified in the font config file.

Return type bool

```
glyph (glyph\_name: str = None, fg\_color: pygamelib.gfx.core.Color = None, bg\_color: pygamelib.gfx.core.Color = None) <math>\rightarrow pygamelib.gfx.core.Sprite

This method take a glyph name in parameter and returns its representation as a Sprite.
```

The glyph name is usually the name of a character (like "a") but it is not mandatory and can be anything. The default glyph (returned when no glyph matches the requested glyph) is called "default" for example.

```
Parameters glyph_name (str) - The glyph name
Returns A glyphe as a Sprite
Return type Sprite
```

Example:

```
myfont = Font("8bits")
row = 5
```

(continues on next page)

(continued from previous page)

```
column = 10
for letter in "this is a text":
    glyph = myfont.glyph(letter)
    screen.place(glyph, row, column)
    column += glyph.width + myfont.horizontal_spacing()

# Please note that in real life you would just do this
mytext = Text("this is a text", font=myfont)
screen.place(mytext, row, column)
```

glyphs_map

Returns the glyph map of the font as specified in the font config file.

Return type dict

height

Returns the height of the font as specified in the font config file.

Return type int

Example:

```
screen.place(text, last_row + myfont.height, first_text_column)
```

horizontal_spacing

Returns the horizontal spacing recommended by the font (as specified in the font config file).

As a user of the font class using the Font class to change the look of some text, you will rarely use that value directly (it is directly used by Text.render_to_buffer()).

If your goal is to use the Font class to do glyph rendering as you see fit, use the horizontal spacing value to place each glyph relatively to the one on its left or right.

Return type int

```
load (font name: str = None) \rightarrow None
```

Load a font by name. Once the font is loaded glyphs can be accessed through the glyph () method.

This method is automatically called is the Font constructor is called with a font name.

Parameters font_name (str) - The name of the font to load upon object construction.

Example:

```
# The 2 following examples do exactly the same thing.
# Example 1: instantiate and load
myfont = Font()
myfont.load("8bits")
# Example 2: load from instantiation
myfont2 = Font("8bits")
# At that point myfont and myfont2 are exactly the same (and there is no
# good justification to instantiate or load the font twice).
```

monospace

Returns if the font is monospace as specified in the font config file.

Return type bool

name

Return the name of the font. The name is the string that was used to load the font.

Example:

3.7. qfx 485

```
myfont = Font("8bits")
if myfont.name() != "8bits":
    print("Something very wrong just occurred!")
```

scalable

Returns the scalability of the font as specified in the font config file.

Return type bool

vertical_spacing

Returns the vertical spacing recommended by the font (as specified in the font config file).

Return type int

Example:

```
screen.place(
   text,
   last_row + myfont.height() + myfont.vertical_spacing(),
   first_text_column
)
```

SpriteCollection

```
class pygamelib.gfx.core.SpriteCollection(data=None)
    Bases: collections.UserDict
```

SpriteCollection is a dictionnary class that derives collections. UserDict.

Its main goal is to provide an easy to use object to load and save sprite files. On top of traditional dict method, it provides the following capabilities:

- · loading and writing from and to JSON files,
- data serialization,
- shortcut to add sprites to the dictionnary.

A SpriteCollection is an unordered indexed list of Sprites (i.e a dictionnary).

Sprites are indexed by their names in that collection.

Example:

```
# Load a sprite file
sprites_village1 = SpriteCollection.load_json_file('gfx/village1.spr')
# display the Sprites with their name
for sprite_name in sprites_village1:
    print(f'{sprite_name}:\n(sprites_village1[sprite_name]}')
# Add an empty sprite with name 'house_placeholder'
sprites_village1.add( Sprite(name='house_placeholder') )
# This is absolutely equivalent to:
sprites_village1['house_placeholder'] = Sprite(name='house_placeholder')
# And now rewrite the sprite file with the new placeholder house
sprites_village1.to_json_file('gfx/village1.spr')
```

```
___init___(data=None)
```

Initialize self. See help(type(self)) for accurate signature.

Methods

<pre>clear() copy() fromkeys(iterable[, value]) get(k[,d]) items() keys()</pre>	Sprite to the collection.
<pre>copy() fromkeys(iterable[, value]) get(k[,d]) items() keys()</pre>	
<pre>fromkeys(iterable[, value]) get(k[,d]) items() keys()</pre>	
<pre>get(k[,d]) items() keys()</pre>	
<pre>items() keys()</pre>	
keys()	
_ :	
I and	
	serialized data and return a new SpriteCollec-
tion o	v .
<pre>load_json_file(filename)</pre> Load	a JSON sprite file into a new SpriteCollection
object	•
pop(k[,d]) If key	is not found, d is returned if given, otherwise
KeyE	ror is raised.
popitem() as a 2	-tuple; but raise KeyError if D is empty.
rename(old_key, new_key) Renam	ne a key in the collection.
serialize() Return	a serialized version of the SpriteCollection.
setdefault(k[,d])	
to_json_file(filename) Expor	t the SpriteCollection object in JSON and
writes	it on the disk.
update([E,]**F)	resent and has a .keys() method, does: for k in
E: D[J	K = E[K] If E present and lacks .keys() method,
does:	for (k, v) in E: $D[k] = v$ In either case, this is
follow	ved by: for k, v in F.items(): $D[k] = v$
values()	

add (sprite)

Add a Sprite to the collection. This method is simply a shortcut to the usual dictionnary affectation. The collection requires the name of the Sprite to be the key. That method does that automatically.

Parameters sprite (Sprite) – A Sprite object to add to the collection.

Warning: As SpriteCollection index Sprites by their name if you change the Sprite's name *after* adding it to the collection you will need to manually update the keys.

Example:

```
sprites_village1 = SpriteCollection.load_json_file('gfx/village1.spr')
new_village = SpriteCollection()
new_village.add( copy.deepcopy( sprites_village1.get('bakery') ) )
print( new_village['bakery'] )
```

 ${\tt clear}() o {\tt None}.$ Remove all items from D. ${\tt copy}()$ ${\tt classmethod}$ ${\tt fromkeys}(iterable, value=None)$ ${\tt get}(k{\tt l},d{\tt l}) o {\tt D[k]}$ if k in D, else d. d defaults to None. ${\tt items}() o {\tt a}$ set-like object providing a view on D's items

3.7. gfx 487

keys () \rightarrow a set-like object providing a view on D's keys

classmethod load(data)

Load serialized data and return a new SpriteCollection object.

Parameters data (str) – Serialized data that need to be expanded into objects.

Returns A new SpriteCollection object.

Return type SpriteCollection

Example:

```
sprites_village1 = SpriteCollection.load(
    sprites_village_template.serialize()
)
```

static load_json_file(filename)

Load a JSON sprite file into a new SpriteCollection object.

Parameters filename (str) – The complete path (relative or absolute) to the sprite file.

Returns A new SpriteCollection object.

Return type SpriteCollection

Example:

```
sprites_village1 = SpriteCollection.load_json_file('gfx/village1.spr')
```

 $pop(k[,d]) \rightarrow v$, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised.

```
popitem() \rightarrow (k, v), remove and return some (key, value) pair as a 2-tuple; but raise KeyError if D is empty.
```

```
rename (old_key, new_key)
```

Rename a key in the collection.

This methods also takes care of renaming the Sprite associated with the old key name.

Parameters

- old_key (str) The key to rename
- **new_key** (str) The new key name

Example:

```
my_collection.rename('panda', 'panda walk 01')
```

serialize()

Return a serialized version of the SpriteCollection. The serialized data can be pass to the JSON module to export.

Returns The SpriteCollection object serialized as a dictionnary.

Return type dict

Example:

```
data = sprites_village1.serialize()
```

setdefault $(k[,d]) \rightarrow D.get(k,d)$, also set D[k]=d if k not in D

```
to json file (filename)
```

Export the SpriteCollection object in JSON and writes it on the disk.

Parameters filename (str) – The complete path (relative or absolute) to the sprite file to write.

Example:

```
sprites_village1.to_json_file('gfx/village1.spr')
```

update ([E], **F) \rightarrow None. Update D from mapping/iterable E and F.

If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

values () \rightarrow an object providing a view on D's values

Sprite

The Sprite object represent a 2D "image" that can be used to represent any complex item. Obviously, a sprite in the pygamelib is not really an image, it is a series of glyphs (or characters) with colors (foreground and background) information.

A Sprite object is a 2D array of Sprixel.

If you use the climage python module, you can load the generated result into a Sprite through Sprite.load from ansi file().

Parameters

- sprixels (list) A 2D array of Sprixel.
- **default_sprixel** (*Sprixel*) A default Sprixel to complete lines that are not long enough. By default, it's an empty Sprixel.
- parent (BoardComplexItem (suggested)) The parent object of this Sprite. If it's left to None, the BoardComplexItem constructor takes ownership of the sprite.
- **size** (list) A 2 elements list that represent the width and height ([width, height]) of the Sprite. It is only needed if you create an empty Sprite. If you load from a file or provide an array of sprixels it's obviously calculated automatically. Default value: [2, 2].
- name (str) The name of sprite. If none is given, an UUID will be automatically generated.

Example:

```
void = Sprixel()
# This represent a panda
panda_sprite = Sprite(
    sprixels=[
        [void, void, void, void, void, void, void],
        [
            Sprixel.black_rect(),
            Sprixel.black_rect(),
            void,
            void,
            void,
            void,
            void,
```

(continues on next page)

3.7. qfx 489

(continued from previous page)

```
void,
    Sprixel.black_rect(),
    Sprixel.black_rect(),
],
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
],
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.black_rect(),
    Sprixel.black_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.black_rect(),
    Sprixel.black_rect(),
],
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.red_rect(),
    Sprixel.red_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
],
Γ
    void,
    void,
    Sprixel.black_rect(),
    Sprixel.black_rect(),
    Sprixel.black_rect(),
    Sprixel.black_rect(),
    void,
    void,
],
    void,
    void,
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.white_rect(),
    Sprixel.black_rect(),
    Sprixel.black_rect(),
],
    void,
    void,
```

(continues on next page)

(continued from previous page)

__init__ (sprixels=None, default_sprixel=None, parent=None, size=[2, 2], name=None)
Like the object class, this class constructor takes no parameter.

Methods

init([sprixels, default_sprixel,])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
calculate_size()	Calculate the size of the sprite and update the size
	variable.
copy()	Returns a (deep) copy of the sprite.
detach(observer)	Detach an observer from this instance.
empty()	Empty the sprite and fill it with default sprixels.
flip_horizontally()	Flip the sprite horizontally.
flip_vertically()	Flip the sprite vertically (i.e upside/down).
<pre>from_text(text_object)</pre>	Create a Sprite from a Text object.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Create a new Sprite object based on serialized data.
load_from_ansi_file(filename[, de-	Load an ANSI encoded file into a Sprite object.
fault_sprixel])	
modulate(color, ratio)	Modulate the sprite colors with the color in parame-
	ters.
<pre>notify([modifier])</pre>	Notify all the observers that a change occurred.
render_to_buffer(buffer, row, column,)	Render the sprite from the display buffer to the frame buffer.
scale([ratio])	Scale a sprite up and down using the nearest neighbor
scare([lauo])	algorithm.
serialize()	Serialize a Sprite into a dictionary.
set_sprixel(row, column, value)	Set a specific sprixel in the sprite to the given value.
set_transparency(state)	This method enable transparent background to all the
bee_eransparency (suic)	sprite's sprixels.
sprixel([row, column])	Return a sprixel at a specific position within the
	sprite.
store_screen_position(row, column)	Store the screen position of the object.
tint(color, ratio)	Tint a copy of the sprite with the color.

Attributes

3.7. gfx 491

height	Property that returns the height of the Sprite.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
width	Property that returns the width of the Sprite.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

calculate_size()

Calculate the size of the sprite and update the size variable.

The size is immediately returned.

It is done separately for concerns about performances of doing that everytime the size is requested.

Return type list

Example:

copy()

Returns a (deep) copy of the sprite.

New in version 1.3.0.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

empty()

Empty the sprite and fill it with default sprixels.

Example:

```
player_sprite.empty()
```

flip_horizontally()

Flip the sprite horizontally.

This method performs a symmetry versus the vertical axis.

At the moment, glyph are not inverted. Only the position of the sprixels.

The flipped sprite is returned (original sprite is not modified).

```
Return type Sprite
```

Example:

```
reflection_sprite = player_sprite.flip_horizontally()
```

flip_vertically()

Flip the sprite vertically (i.e upside/down).

At the moment, glyph are not inverted. Only the position of the sprixels. There is one exception however, as climage uses the '' utf8 glyph as a marker, that specific glyph is inverted to '' and vice versa.

The flipped sprite is returned (original sprite is not modified).

```
Return type Sprite
```

Example:

```
reflection_sprite = player_sprite.flip_vertically()
```

classmethod from_text(text_object)

Create a Sprite from a Text object.

Parameters text_object (*Text*) – A text object to transform into Sprite.

Example:

```
# The Text object allow for easy manipulation of text
village_name = base.Text('Khukdale',fg_red, bg_green)
# It can be converted into a Sprite to be displayed on the Board
village_sign = board_items.Tile(sprite=Sprite.from_text(village_name))
# And can be used as formatted text
notifications.push( f'You enter the dreaded village of {village_name}' )
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

• **subject** (*PglBaseObject*) – The object that has changed.

3.7. qfx 493

- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

height

Property that returns the height of the Sprite.

New in version 1.3.0.

Contrary to Sprite.size[1], this property always calls Sprite.calculate size() before returning the height.

classmethod load(data)

Create a new Sprite object based on serialized data.

New in version 1.3.0.

Parameters data (dict) – Data loaded from a JSON sprite file (deserialized).

Return type Sprite

Example:

```
new_sprite = Sprite.load(json_parsed_data)
```

classmethod load_from_ansi_file (filename, default_sprixel=None)

Load an ANSI encoded file into a Sprite object.

This class method can load a file produced by the climage python module and load it into a Sprite class. Each character is properly decoded into a Sprixel with model, background and foreground colors.

A Sprite is rectangular (at least for the moment), so in case the file is not shaped as a rectangle, this method automatically fills the void with a default sprixel (to make sure all lines in the sprite have the same length). By default, it fills the table with None "values" but you can specify a default sprixel.

The reasons the default sprixel is set to None is because None values in a sprite are not translated into a component in <code>BoardComplexItem</code> (i.e no sub item is generated).

Parameters

- **filename** (str) The path to a file to load.
- **default_sprixel** (None | *Sprixel*) The default Sprixel to fill a non rectangular shaped sprite.

Example:

```
player_sprite = gfx_core.Sprite.load_from_ansi_file('gfx/models/player.ans')
```

modulate (color: pygamelib.gfx.core.Color, ratio: float = 0.5)

Modulate the sprite colors with the color in parameters.

New in version 1.3.0.

This method tint all the sprixels of the sprite with the color at the specified ratio. **The original sprite IS modified**.

If you want to keep the original sprite intact consider using tint().

Parameters

- **color** (*Color*) The modulation color.
- ratio (float) The modulation ratio between 0.0 and 1.0 (default: 0.5)

Returns None

When this method is called, the observers are notified of the change with the pygamelib.core.Sprite.color:modulated event. No arguments are passed along this event.

Example:

```
player_sprites = core.SpriteCollection.load_json_file("gfx/player.spr")
# After that, the sprite is quite not "normal" anymore...
player_sprites["normal"].modulate(core.Color(0, 255, 0), 0.3)
```

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render the sprite from the display buffer to the frame buffer.

New in version 1.3.0.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

scale (ratio=1.0)

Scale a sprite up and down using the nearest neighbor algorithm.

New in version 1.3.0.

Parameters ratio (float) – The scaling ration.

Returns An upscaled/downscaled sprite.

Return type Sprite

Note: The sprites generated with pgl-converter.py don't scale well yet if the –unicode flag is active

Example:

3.7. qfx 495

```
bigger_sprite = original_sprite.scale(2)
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serialize a Sprite into a dictionary.

New in version 1.3.0.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( sprite.serialize() )
```

set_sprixel (row, column, value)

Set a specific sprixel in the sprite to the given value.

Parameters

- row (int) The row of the sprite (WARNING: internal sprite coordinates)
- **column** (*int*) The column of the sprite (same warning)
- **value** (*Sprixel*) The sprixel to set at [row, column]

When a sprixel is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprite.sprixel:changed event. A structure is passed as the *value* parameter. This structure has 3 members: row, column and sprixel.

Example:

```
my_sprite.set_sprixel(1, 2, Sprixel("#",fg_color=green))
```

set_transparency (state)

This method enable transparent background to all the sprite's sprixels.

New in version 1.3.0.

Parameters state – a boolean to enable or disable background transparency

When the transparency is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprite.transparency:changed event. The new transparency state is passed as the *value* parameter.

Example:

```
player_sprite.set_transparency(True)
```

Warning: This set background transparency on all sprixels, make sure you are not using background colors as part of your sprite before doing that. It can also be used as a game/rendering mechanic. Just make sure you know what you do. As a reminder, by default, sprixels with no background have transparent background enable.

sprixel (row=0, column=None)

Return a sprixel at a specific position within the sprite.

If the column is set to None, the whole row is returned.

Parameters

- row (int) The row to access within the sprite.
- **column** (*int*) The column to access within the sprite.

Returns Sprixel

Example:

```
# Return the entire line at row index 2
scanline = house_sprite.sprixel(2)
# Return the specific sprixel at sprite internal coordinate 2,3
house_sprixel = house_sprite.sprixel(2, 3)
```

Warning: For performance consideration sprixel() does not check the size of its matrix. This method is called many times during rendering and 2 calls to len() in a row are adding up pretty quickly. It checks the boundary of the sprite using the cached size. Make sure it is up to date!

```
store\_screen\_position (row: int, column: int) \rightarrow bool
```

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

tint (color: pygamelib.gfx.core.Color, ratio: float = 0.5)

Tint a copy of the sprite with the color.

New in version 1.3.0.

This method creates a copy of the sprite and tint all its sprixels with the color at the specified ratio. It then returns the new sprite. **The original sprite is NOT modified**.

Parameters

- color (Color) The tint color.
- ratio (float) The tint ration between 0.0 and 1.0 (default: 0.5)

Returns Sprite

Example:

3.7. qfx 497

width

Property that returns the width of the Sprite.

New in version 1.3.0.

Contrary to Sprite.size[0], this property always calls Sprite.calculate_size() before returning the width.

Sprixel

```
class pygamelib.gfx.core.Sprixel(model=", bg_color=None, fg_color=None, is_bg_transparent=None)
    Bases: pygamelib.base.PqlBaseObject
```

A sprixel is the representation of 1 cell of the sprite or one cell on the Board. It is not really a pixel but it is the closest notion we'll have. A Sprixel has a background color, a foreground color and a model. All regular BoardItems can now use a sprixel instead of a model (but simple model is still supported of course).

In the terminal, a sprixel is represented by a single character.

If the background color and the is_bg_transparent are None, the sprixel will be automatically configured with transparent background. In that case, as we cannot really achieve transparency in the console, the sprixel will take the background color of whatever it is overlapping.

Important: BREAKING CHANGE: in version 1.3.0 background and foreground colors use the new *Color* object. Therefor, Sprixel does not accept ANSI sequences anymore for the bg_color and fg_color parameters.

Example:

___init__ (model=", bg_color=None, fg_color=None, is_bg_transparent=None)

Parameters

- model (str) The model, it can be any string. Preferrably a single character.
- **bg_color** (*Color*) A Color object to configure the background color.
- fg color (Color) A Color object to configure the foreground color.
- is_bg_transparent (bool) Set the background of the Sprixel to be transparent. It tells the engine to replace the background of the Sprixel by the background color of the overlapped sprixel.

Methods

init([model, bg_color, fg_color,])	
	param model The model, it can be any
	string. Preferrably a single character.
attach(observer)	Attach an observer to this instance.
black_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.BLACK_RECT.
black_square()	This class method returns a
	sprixel that is the equivalent of
	pygamelib.assets.graphics.BLACK_SQUARE.
blue_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.BLUE_RECT.
blue_square()	This class method returns a sprixel that is the equiva-
	lent of pygamelib.assets.graphics.BLUE_SQUARE.
copy()	Returns a (deep) copy of the sprixel.
cyan_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.CYAN_RECT.
cyan_square()	This class method returns a sprixel that is the equiva-
	lent of pygamelib.assets.graphics.CYAN_SQUARE.
detach(observer)	Detach an observer from this instance.
<pre>from_ansi(string[, model])</pre>	Takes an ANSI string, parse it and return a Sprixel.
green_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.GREEN_RECT.
green_square()	This class method returns a
	sprixel that is the equivalent of
	pygamelib.assets.graphics.GREEN_SQUARE.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Create a new Sprixel object based on serialized data.
magenta_rect()	This class method returns a
	sprixel that is the equivalent of
	pygamelib.assets.graphics.MAGENTA_RECT.
magenta_square()	This class method returns a
	sprixel that is the equivalent of
	pygamelib.assets.graphics.MAGENTA_SQUARE.
notify([modifier])	Notify all the observers that a change occurred.
red_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.RED_RECT.
red_square()	alent of pygamelib.assets.graphics.RED_RECT. This class method returns a sprixel that is the equiv-
	This class method returns a sprixel that is the equiv-
red_square()	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE.
<pre>red_square() render_to_buffer(buffer, row, column,)</pre>	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the
red_square()	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer.
<pre>red_square() render_to_buffer(buffer, row, column,) serialize()</pre>	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary.
<pre>red_square() render_to_buffer(buffer, row, column,) serialize() store_screen_position(row, column)</pre>	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary. Store the screen position of the object. This class method returns a sprixel that is the equiv-
red_square() render_to_buffer(buffer, row, column,) serialize() store_screen_position(row, column) white_rect()	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary. Store the screen position of the object.
<pre>red_square() render_to_buffer(buffer, row, column,) serialize() store_screen_position(row, column)</pre>	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary. Store the screen position of the object. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_RECT. This class method returns a
red_square() render_to_buffer(buffer, row, column,) serialize() store_screen_position(row, column) white_rect()	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary. Store the screen position of the object. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_RECT. This class method returns a sprixel that is the equivalent of
<pre>red_square() render_to_buffer(buffer, row, column,) serialize() store_screen_position(row, column) white_rect() white_square()</pre>	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary. Store the screen position of the object. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_RECT. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_SQUARE.
red_square() render_to_buffer(buffer, row, column,) serialize() store_screen_position(row, column) white_rect()	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary. Store the screen position of the object. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_RECT. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_SQUARE. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_SQUARE.
<pre>red_square() render_to_buffer(buffer, row, column,) serialize() store_screen_position(row, column) white_rect() white_square()</pre>	This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. Render the sprixel from the display buffer to the frame buffer. Serialize a Sprixel into a dictionary. Store the screen position of the object. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_RECT. This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_SQUARE.

3.7. gfx 499

Table 99 – continued from previous page

yellow_square()	This	class	n	nethod	returns	a	
	sprixel	that	is	the	equivalent	of	
	pygamelib.assets.graphics.YELLOW_SQUARE.						

Attributes

bg_color	A property to get/set the background color of the
	Sprixel.
fg_color	A property to get/set the foreground color of the
	Sprixel.
length	Return the true length of the model.
model	A property to get/set the model of the Sprixel.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

bg_color

A property to get/set the background color of the Sprixel.

```
Parameters value (Color) – The new color
```

When the bg_color is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprixel.bg_color:changed event. The new bg_color is passed as the *value* parameter.

Example:

```
# Access the sprixel's color
sprix.bg_color
# Set the sprixel's background color to some blue
sprix.bg_color = Color(0,128,255)
```

classmethod black_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLACK_RECT. The difference is that BLACK_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.black_rect()
```

classmethod black_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLACK_SQUARE. The difference is that BLACK_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.black_square()
```

classmethod blue_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLUE_RECT. The difference is that BLUE_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.blue_rect()
```

classmethod blue_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLUE_SQUARE. The difference is that BLUE_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.blue_square()
```

copy()

Returns a (deep) copy of the sprixel.

New in version 1.3.0.

classmethod cyan_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.CYAN_RECT. The difference is that CYAN_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.cyan_rect()
```

classmethod cyan_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.CYAN_SQUARE. The difference is that CYAN_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.cyan_square()
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

fg_color

A property to get/set the foreground color of the Sprixel.

```
Parameters value (Color) – The new color
```

When the fg_color is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprixel.fg_color:changed event. The new fg_color is passed as the *value* parameter.

Example:

```
# Access the sprixel's color
sprix.fg_color
# Set the sprixel's foreground color to some green
sprix.fg_color = Color(0,255,128)
```

static from_ansi(string, model='')

Takes an ANSI string, parse it and return a Sprixel.

Parameters

- **string** (*str*) The ANSI string to parse.
- mode1 (str) The character used to represent the sprixel in the ANSI sequence. Default
 is ""

Example:

```
new_sprixel = Sprixel.from_ansi(
    "\x1b[48;2;139;22;19m\x1b[38;2;160;26;23m\x1b[0m"])
```

Warning: This has mainly be tested with ANSI string generated by climage. If you find any issue, please report it

classmethod green_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.GREEN_RECT. The difference is that GREEN_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.green_rect()
```

classmethod green_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.GREEN_SQUARE. The difference is that GREEN_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.green_square()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

length

Return the true length of the model.

New in version 1.3.0.

With UTF8 and emojis the length of a string as returned by python's len() function is often very wrong. For example, the len("x1b[48;2;139;22;19mx1b[38;2;160;26;23mx1b[0m") returns 39 when it should return 1.

This method returns the actual printing/display size of the sprixel's model.

Note: This is a read only value. It is automatically updated when the model is changed.

Example:

```
if sprix.length > 2:
    print(
        f"Warning: that sprixel {sprix} will break the rest of the "
        "board's alignement"
        )
```

classmethod load(data)

Create a new Sprixel object based on serialized data.

New in version 1.3.0.

Parameters data (dict) – Data loaded from JSON data (descrialized).

Return type Sprixel

Example:

```
new_sprite = Sprixel.load(json_parsed_data['default_sprixel'])
```

classmethod magenta_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.MAGENTA_RECT. The difference is that MAGENTA_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.magenta_rect()
```

classmethod magenta_square()

This class method returns a sprixel that is the equivalent of

3.7. qfx 503

pygamelib.assets.graphics.MAGENTA_SQUARE. The difference is that MAGENTA_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.magenta_square()
```

model

A property to get/set the model of the Sprixel.

Parameters value (str) – The new model

When the model is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprixel.model:changed event. The new model is passed as the *value* parameter.

Example:

```
# Get the sprixel's model
sprix.model
# Set the sprixel's model to "@"
sprix.model = "@"
```

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

classmethod red_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_RECT. The difference is that RED_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.red_rect()
```

classmethod red_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. The difference is that RED_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.red_square()
```

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render the sprixel from the display buffer to the frame buffer.

New in version 1.3.0.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serialize a Sprixel into a dictionary.

New in version 1.3.0.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( sprixel.serialize() )
```

 $store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

classmethod white rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_RECT. The difference is that WHITE_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.white_rect()
```

classmethod white_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_SQUARE. The difference is that WHITE_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.white_square()
```

classmethod yellow_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.YELLOW_RECT. The difference is that YELLOW_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Note: Yellow is often rendered as brown.

Example:

```
sprixel = Sprixel.yellow_rect()
```

classmethod yellow_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.YELLOW_SQUARE. The difference is that YELLOW_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Note: Yellow is often rendered as brown.

Example:

```
sprixel = Sprixel.yellow_square()
```

Color

```
class pygamelib.gfx.core.Color(r=0, g=0, b=0)
Bases: pygamelib.base.PqlBaseObject
```

New in version 1.3.0.

A color represented by red, green and blue (RGB) components. Values are integer between 0 and 255 (both included).

Parameters

- **r** (*int*) The red component of the color.
- **g** (*int*) The green component of the color.
- **b** (*int*) The blue component of the color.

Example:

```
# color is blue
color = Color(0, 0, 255)
# and now color is pink
color.r = 255
```

```
\_init\_ (r=0, g=0, b=0)
```

Like the object class, this class constructor takes no parameter.

Methods

init([r, g, b])	Like the object class, this class constructor takes no
	parameter.
attach(observer)	Attach an observer to this instance.
blend(other_color[, fraction])	Blend the color with another one.
copy()	Returns a (deep) copy of this color.
detach(observer)	Detach an observer from this instance.
from_ansi(string)	Create and return a Color object based on an ANSI
	color string.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Create a new Color object based on serialized data.
notify([modifier])	Notify all the observers that a change occurred.
notify([modifier]) random()	Notify all the observers that a change occurred. Create and return a new random color.
	•
random()	Create and return a new random color.
random()	Create and return a new random color. Set a random value for each of the components of an

Attributes

b	The b property controls the intensity of the blue
~	color.
g	The g property controls the intensity of the green
	color.
r	The r property controls the intensity of the red color.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

b

The b property controls the intensity of the blue color. You can set it to an integer between 0 and 255 (both included).

When this property is set, the observers are notified with the pygamelib.gfx.core.Color.b:changed event. The value of the event is the new value of the property.

Example:

```
color = Color(128, 128, 0)
print(f"Value for b is {color.b}")
color.b = 255
print(f"New value for b is {color.b}")
```

blend (other_color, fraction=0.5)

Blend the color with another one. Fraction controls the amount of other_color that is included (0 means no inclusion at all).

Parameters

- other_color (Color) The color to blend with.
- **fraction** (*float*) The blending modulation factor between 0 and 1.

Returns A new Color object that contains the blended color.

Return type Color

Example:

```
a = Color(200, 200, 200)
b = Color(25, 25, 25)
# c is going to be Color(112, 112, 112)
c = a.blend(b, 0.5)
```

copy()

Returns a (deep) copy of this color.

Example:

```
red = Color(255, 0, 0)
red2 = red.copy()
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

classmethod from_ansi(string)

Create and return a Color object based on an ANSI color string.

Important: The string must be RGB, i.e '[38;2;RED;GREEN;BLUEm' or '[48;2;RED;GREEN;BLUEm' for foreground and background colors. This method will return None if the color string is not RGB. It is also important to understand that Color is independent from the

foreground of background, it is just a color. Therefor '[38;2;89;32;93m' and '[48;2;89;32;93m' will both be parsed into Color(89, 32, 93).

Parameters string (str) – The ANSI color string to convert.

Example:

```
color = Color.from_ansi()
```

g

The g property controls the intensity of the green color. You can set it to an integer between 0 and 255 (both included).

When this property is set, the observers are notified with the pygamelib.gfx.core.Color.g:changed event. The value of the event is the new value of the property.

Example:

```
color = Color(128, 128, 0)
print(f"Value for g is {color.g}")
color.g = 255
print(f"New value for g is {color.g}")
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- subject (PglBaseObject) The object that has changed.
- **attribute** (*str*) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

classmethod load(data)

Create a new Color object based on serialized data.

If data is None, None is returned.

If a color component is missing from data, it is set to 0 (see examples).

Raises an exception if the color components are not integer.

Parameters data (dict) – Data loaded from JSON data (descrialized).

Returns Either a Color object or None if data where empty.

Return type Color | NoneType

Raise PglInvalidTypeException

Example:

3.7. qfx 509

```
# Loading from parsed JSON data
new_color = Color.load(json_parsed_data['default_sprixel']['fg_color'])

# Loading from incomplete data
color = Color.load({'red':25,'green':35})
# Result in the following Color object:
# Color(25, 35, 0)
```

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

r

The r property controls the intensity of the red color. You can set it to an integer between 0 and 255 (both included).

When this property is set, the observers are notified with the pygamelib.gfx.core.Color.r:changed event. The value of the event is the new value of the property.

Example:

```
color = Color(128, 128, 0)
print(f"Value for r is {color.r}")
color.r = 255
print(f"New value for r is {color.r}")
```

classmethod random()

Create and return a new random color.

```
Return type Color
```

Example:

```
my_color = Color.random()
```

randomize()

Set a random value for each of the components of an existing color.

When this method is called, the observers are notified with the pygamelib.gfx.core.Color.randomized event. The value of the event is the new color.

Returns None

Return type NoneType

Example:

```
color = Color()
color.randomize()
```

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serialize a Color into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( color.serialize() )
```

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

3.7.2 ui

Warning: The UI module is in alpha version. Some things might change over time.

The ui module contains the classes to easily build full screen Terminal User Interface (TUI) for your games (or applications).

Important: It works exclusively with the screen buffer system (place, delete, render, update, etc.). It doesn't work with Screen functions tagged "direct display" like display_at().

Box

Bases: object

A simple object to draw a box on screen.

The Box object's looks and feel is highly configurable through the *UiConfig* object.

__init__ (width: int, height: int, title: str = ", config: pygamelib.gfx.ui.UiConfig = None, fill: bool = False, filling_sprixel: pygamelib.gfx.core.Sprixel = None, title_alignment: int = 30000101)

The box constructor takes the following parameters.

Parameters

- width (int) The width of the box.
- **height** (*int*) The height of the box.
- **title** (str | Text) The title of the box (encased in the top border).
- **config** (*UiConfig*) The configuration object.
- **fill** (bool) A tag to tell the box object to fill its inside (or not).
- **filling_sprixel** (Sprixel) If fill is True, the filling Sprixel is used to fill the inside of the box.
- title_alignment (int) The alignment of the title in the top bar. It is a constant from the constant module and can be ALIGN_LEFT, ALIGN_RIGHT and ALIGN_CENTER. THIS FEATURE IS NOT YET IMPLEMENTED.

Todo: Implement the title alignment.

Example:

```
config = UiConfig(bg_color=None)
box = Box(30, 10, 'This is a box')
screen.place(box, 20, 20)
screen.update()
```

Methods

init(width, height, title, config, fill,)	The box constructor takes the following parameters.
render_to_buffer(buffer, row, column,)	Render the box from the display buffer to the frame
	buffer.

Attributes

config	Get and set the config object (UiConfig).
height	Get and set the height of the box, only accept int.
title	Get and set the title, only accepts str or Text.
width	Get and set the width of the box, only accept int.

config

Get and set the config object (UiConfig).

height

Get and set the height of the box, only accept int.

render_to_buffer (buffer, row, column, buffer_height, buffer_width) \rightarrow None Render the box from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

title

Get and set the title, only accepts str or Text.

width

Get and set the width of the box, only accept int.

ColorPickerDialog

```
class pygamelib.gfx.ui.ColorPickerDialog(title: str = None, pygamelib.gfx.ui.UiConfig = None)
Bases: pygamelib.gfx.ui.Dialog
```

The ColorPickerDialog is a dialog wrapper around the ColorPicker widget.

It serves the same purpose: present a way to easily select a custom color to the user.

It does it as an immediately usable dialog.

The show() method returns the Color selected by the user. If the user pressed the ESC key, it returns None.

Key mapping:

- ESC: Exit from the show() method and return None.
- ENTER: Exit from the show() method. Returns the currently selected color.
- UP / DOWN: Increase/decrease the currently selected channel by 1.
- PAGE_UP / PAGE_DOWN: Increase/decrease the currently selected channel by 10.
- LEFT / RIGHT: Navigate between color channels.

Like all dialogs, it is automatically destroyed on exit of the show() method. It is also deleted from the screen buffer.

```
__init__(title: str = None, config: pygamelib.gfx.ui.UiConfig = None) <math>\rightarrow None The constructor only take the configuration as parameter.
```

Parameters config (*UiConfig*) – The configuration object.

Example:

3.7. qfx 513

```
color_dialog = ColorPickerDialog(conf)
color_dialog.set_color( core.Color(128, 128, 128) )
screen.place(color_dialog, 10, 10)
new_color = color_dialog.show()
```

Methods

init(title, config)	The constructor only take the configuration as pa-
	rameter.
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.
set_color(color)	Set the color shown in the dialog.
set_selection(selection)	Set the channel selection.
show()	Show the dialog and execute the event loop.

Attributes

config	Get and set the config object (UiConfig).
title	Get / set the dialog title, it needs to be a str.
user_input	Facility to store and retrieve the user input.

config

Get and set the config object (*UiConfig*).

render_to_buffer (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

 $\mathtt{set_color}(color: pygamelib.gfx.core.Color) \rightarrow \mathsf{None}$

Set the color shown in the dialog.

Parameters color (Color) – The color to edit.

Example:

```
color_dialog.set_color( core.Color(128, 128, 128) )
```

set_selection (selection: int = 0)

Set the channel selection.

Parameters selection (int) – The number of the channel to select (0 = red, 1 = green and 2 = blue).

Example:

```
color_dialog.set_selection(1)
```

show()

Show the dialog and execute the event loop. Until this method returns, all keyboards event are processed by the local event loop. This is also true if called from the main event loop.

This event loop returns the edited Color or None if the user pressed the ESC key.

Returns The editor color.

Return type Color

Example:

```
new_color = color_dialog.show()
```

title

Get / set the dialog title, it needs to be a str.

user_input

Facility to store and retrieve the user input.

ColorPicker

```
class pygamelib.gfx.ui.ColorPicker (orientation: int = None, config: pygamelib.gfx.ui.UiConfig = None)

Bases: object
```

The ColorPicker widget is a simple object to select the red, green and blue components of a color.

It provides the API to set/get each color channel independently as well as the mechanism to select and draw a selection box around one specific channel to give the user a visual cue about what he is modifying.

```
__init__ (orientation: int = None, config: pygamelib.gfx.ui.UiConfig = None) \rightarrow None The constructor is really simple and takes only 2 arguments.
```

Parameters

- ullet or orientation (int) One of the 2 orientation constants pygamelib. constants.ORIENTATION_HORIZONTAL or pygamelib.constants.ORIENTATION_VERTICAL
- **config** (*UiConfig*) The configuration object.

The default orientation is horizontal.

Warning: The orientation parameter is ignored for the moment.

Example:

```
color_picker = ColorPicker(constants.ORIENTATION_HORIZONTAL, conf)
screen.place(color_picker, 10, 10)
screen.update()
```

Methods

	The constructor is really simple and takes only 2 ar-
	guments.
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.

Attributes

blue	Get / set the blue component of the color, the value needs to be an int between 0 and 255.
color	Get / set the edited color.
green	Get / set the green component of the color, the value
	needs to be an int between 0 and 255.
red	Get / set the red component of the color, the value
	needs to be an int between 0 and 255.
selection	Get / set the selection, it needs to be an int between
	0 and 2 included.

blue

Get / set the blue component of the color, the value needs to be an int between 0 and 255.

color

Get / set the edited color.

The setter automatically forward the individual red, green and blue values to to the proper properties of that widget.

Parameters value (*Color*) – The color object.

Example:

```
current_color = color_picker.color
current_color.r += 10
color_picker.color = current_color
```

green

Get / set the green component of the color, the value needs to be an int between 0 and 255.

red

Get / set the red component of the color, the value needs to be an int between 0 and 255.

 $render_to_buffer$ (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

selection

Get / set the selection, it needs to be an int between 0 and 2 included.

0 correspond to the red channel, 1 to the green channel and 2 to the blue channel.

When this widget is rendered a Box will be rendered around the specified channel.

Dialog

```
class pygamelib.gfx.ui.Dialog(config=None)
    Bases: object
```

Dialog is a virtual class that can be subclassed to create actual dialogs.

All classes that inherits from Dialog have the following constraints:

- They need to implement a show() method.
- They are automatically rendered on the second pass by the *Screen* object.

It stores the *UiConfig* object and provide a helper attribute for user inputs.

```
__init__ (config=None) \rightarrow None This constructor takes only one parameter.
```

Parameters config (UiConfig.) - The config object.

Methods

init([config])	This constructor takes only one parameter.
show()	This is a virtual method, calling it directly will only
	raise a NotImplementedError.

Attributes

config	Get and set the config object (UiConfig).
user_input	Facility to store and retrieve the user input.

config

Get and set the config object (*UiConfig*).

show()

This is a virtual method, calling it directly will only raise a NotImplementedError. Each class that inheritate Dialog needs to implement show().

user_input

Facility to store and retrieve the user input.

FileDialog

The FileDialog is a file selection dialog: it allow the user to select a file on disk in a relatively easy way. File can then be use for any purpose by the program, like for "save as" or "open" features.

The show() method returns the path selected by the user.

Key mapping:

- ESC: set the path to None and exit from the show() method.
- ENTER: Exit from the show() method. Returns the currently selected path.
- BACKSPACE / DELETE: delete a character (both keys have the same result).
- UP / DOWN: Navigate between the files.
- LEFT / RIGHT: Navigate between the directories.
- All other keys input characters in the input field.

In all cases, when the dialog is closed, a path is returned. It can be a file name entered by the user or an existing file. The returned value can also be None if the user pressed ESC. There is no guarantee that the returned path is correct. Please, check it before doing anything with it.

Like all dialogs, it is automatically destroyed on exit of the show() method. It is also deleted from the screen buffer.

```
__init__ (path: pathlib.Path = None, width: int = 20, height: int = 10, title: str = 'File dialog', show_hidden_files: bool = False, filter: str = '*', config: pygamelib.gfx.ui.UiConfig = None) → None
```

Parameters

- path (pathlib.Path) The path to start in. This path is made absolute by the constructor.
- width (int) The width of the file dialog widget (in number of screen cells).
- height (int) The height of the file dialog widget (in number of screen cells).
- **title** (*str*) The title of the dialog (written in the upper border).
- show_hidden_files (bool) Does the file dialog needs to show the hidden files or not.
- **filter** (str) A string that will be used to filter the files shown to the user. For example "*.spr".
- **config** (*UiConfig*) The configuration object.

Example:

```
file_dialog = FileDialog( Path("."), 30, 10, "Open file", False, conf)
screen.place(file_dialog, 10, 10)
file = file_dialog.show()
```

Methods

init(path, width, height, title,)	
	<pre>param path The path to start in. This path is made absolute by the</pre>
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.
show()	Show the dialog and execute the event loop.

Attributes

config	Get and set the config object (UiConfig).
filter	Get/set the current file filter.
path	Get/set the current path.
show_hidden_files	Get/set the property, if True the file dialog is going
	to show hidden files, and , if False, it won't.
user_input	Facility to store and retrieve the user input.

config

Get and set the config object (UiConfig).

filter

Get/set the current file filter.

Returns The dialog's current filter.

Return type str

path

Get/set the current path.

Returns The dialog's current path.

Return type pathlib.Path

 $render_to_buffer$ (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

show () \rightarrow pathlib.Path

Show the dialog and execute the event loop. Until this method returns, all keyboards event are processed by the local event loop. This is also true if called from the main event loop.

This event loop returns a pathlib.Path object or None if the user pressed the ESC key. The path can point to an existing file or not.

Example:

```
fields = multi_input.show()
```

show hidden files

Get/set the property, if True the file dialog is going to show hidden files, and, if False, it won't.

Returns The dialog's current show_hidden_files value.

Return type bool

user_input

Facility to store and retrieve the user input.

GridSelectorDialog

```
class pygamelib.gfx.ui.GridSelectorDialog (choices: list = None, max_height: int = None, max_width: int = None, title: str = None, config: pygamelib.gfx.ui.UiConfig = None)
```

Bases: pygamelib.gfx.ui.Dialog

The GridSelectorDialog is an easy wrapper around the GridSelector object. It offers a simple interface for the programmer to present a GridSelector to the user and retrieve its selection.

The show() method returns the path selected by the user.

Key mapping:

- ESC: set the selected item to an empty Sprixel and exit from the show() method.
- ENTER: Exit from the show() method. Returns the currently selected sprixel.
- UP / DOWN / LEFT / RIGHT: Navigate between the files.
- PAGE_UP / PAGE_DOWN: Go to previous / next page if there's any.

In all cases, when the dialog is closed, a *Sprixel* is returned.

Like all dialogs, it is automatically destroyed on exit of the show() method. It is also deleted from the screen buffer.

```
__init__ (choices: list = None, max_height: int = None, max_width: int = None, title: str = None, config: pygamelib.gfx.ui.UiConfig = None) \rightarrow None
```

Parameters

- **choices** (list) A list of choices to present to the user. The elements of the list needs to be str or Sprixel.
- max_height (int) The maximum height of the grid selector.
- max_width (int) The maximum width of the grid selector.
- **config** (*UiConfig*) The configuration object.

Example:

```
choices = ["@","#","$","%","&","*","[","]"]
grid_dialog = GridSelector(choices, 10, 30, conf)
screen.place(grid_dialog, 10, 10)
grid_dialog.show()
```

Methods

```
___init__(choices, max_height, max_width, ...)

param choices A list of choices to present to the user. The elements of the

render_to_buffer(buffer, row, column, ...)

Render the object from the display buffer to the frame buffer.

show()

Show the dialog and execute the event loop.
```

Attributes

config	Get and set the config object (UiConfig).
grid_selector	Get / set the GridSelector object, it has to be a
	GridSelector object.
title	Get / set the title of the dialog, it needs to be a str.
user_input	Facility to store and retrieve the user input.

config

Get and set the config object (*UiConfig*).

grid_selector

Get / set the GridSelector object, it has to be a GridSelector object.

 $render_to_buffer$ (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

show()

Show the dialog and execute the event loop. Until this method returns, all keyboards event are processed by the local event loop. This is also true if called from the main event loop.

This event loop returns the selected item as a Sprixel or None if the user pressed the ESC key.

Returns The selected item.

Return type Sprixel

Example:

```
item = grid_dialog.show()
```

title

Get / set the title of the dialog, it needs to be a str.

user_input

Facility to store and retrieve the user input.

GridSelector

Bases: object

The GridSelector is a widget that present a list of elements as a grid to the user.

It also provides the API to draw and manage the cursor and to retrieve the selected element.

Warning: In the first version of that widget, only the characters that have a length of 1 are supported. This excludes some UTF8 characters and most of the emojis.

```
__init__(choices: list = None, max_height: int = None, max_width: int = None, config: pygamelib.gfx.ui.UiConfig = None) \rightarrow None
```

Parameters

- **choices** (list) A list of choices to present to the user. The elements of the list needs to be str or Sprixel.
- max_height (int) The maximum height of the grid selector.
- $max_width(int)$ The maximum width of the grid selector.
- **config** (*UiConfig*) The configuration object.

Example:

```
choices = ["@","#","$","%","&","*","[","]"]
grid_selector = GridSelector(choices, 10, 30, conf)
screen.place(grid_selector, 10, 10)
screen.update()
```

Methods

init(choices, max_height, max_width, con-	
fig)	param choices A list of choices to
	present to the user. The elements of
	the

current_sprixel()	Returns the currently selected sprixel.
cursor_down()	Move the selection cursor one row down.
cursor_left()	Move the selection cursor one column to the left.
cursor_right()	Move the selection cursor one column to the right.
cursor_up()	Move the selection cursor one row up.
items_per_page()	Returns the number of items per page.
nb_pages()	Returns the number of pages.
page_down()	Change the current page to the one immediately
	down (current_page + 1).
page_up()	Change the current page to the one immediately up
	(current_page - 1).
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.

Attributes

choices	Get and set the list of choices, it has to be a list of
	Sprixel or str.
current_choice	Get and set the currently selected item's index (the
	current choice), it needs to be an int.
	Continued on next page

Table 116 - continued from previous page

current_page	Get and set the current page of the grid selector, it
	needs to be an int.
max_height	Get and set the maximum height of the grid selector,
	it needs to be an int.
max_width	Get and set the maximum width of the grid selector,
	it needs to be an int.

choices

Get and set the list of choices, it has to be a list of Sprixel or str.

current_choice

Get and set the currently selected item's index (the current choice), it needs to be an int. Use <code>current_sprixel()</code> to get the actual current item.

current_page

Get and set the current page of the grid selector, it needs to be an int.

$current_sprixel() \rightarrow pygamelib.gfx.core.Sprixel$

Returns the currently selected sprixel.

cursor down () \rightarrow None

Move the selection cursor one row down.

$\mathtt{cursor_left}$ () \to None

Move the selection cursor one column to the left.

$cursor_right() \rightarrow None$

Move the selection cursor one column to the right.

$\textbf{cursor_up}\,(\,)\,\to None$

Move the selection cursor one row up.

$items_per_page() \rightarrow int$

Returns the number of items per page.

max_height

Get and set the maximum height of the grid selector, it needs to be an int.

${\tt max_width}$

Get and set the maximum width of the grid selector, it needs to be an int.

$nb_pages() \rightarrow int$

Returns the number of pages.

$page_down() \rightarrow None$

Change the current page to the one immediately down (current_page + 1).

$\textbf{page_up} \ (\) \ \rightarrow None$

Change the current page to the one immediately up (current_page - 1).

render_to_buffer (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.

- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

LineInputDialog

```
class pygamelib.gfx.ui.LineInputDialog(title=None, label='Input a value:', default=", fil-
ter=50000002, config=None)
```

```
Bases: pygamelib.gfx.ui.Dialog
```

The LineInputDialog allows the user to enter and edit a single line of text.

This dialog can be configured to accept either anything printable or only digits.

The show() method returns the user input.

Key mapping:

- ESC: set the user input to "" and exit from the show() method.
- ENTER: Exit from the show() method. Returns the user input.
- BACKSPACE / DELETE: delete a character (both keys have the same result)
- All other keys input characters in the input field.

In all cases, when the dialog is closed, the user input is returned.

Like all dialogs, it is automatically destroyed on exit of the show() method. It is also deleted from the screen buffer.

```
__init__ (title=None, label='Input a value:', default="', filter=50000002, config=None) → None
```

Parameters

- **title** (*str*) The short title of the dialog. Only used when the dialog is not borderless.
- label (str | base. Text) The label of the dialog (usually a one line instruction).
- **default** (str) The default value in the input field.
- **filter** (constants.PRINTABLE_FILTER|constants.INTEGER_FILTER) Sets the type of accepted input. It comes from the constants module.
- **config** (*UiConfig*) The configuration object.

Example:

```
line_input = LineInputDialog(
    "Name the pet",
    "Enter the name of your pet:",
    "Stupido",
)
screen.place(line_input, 10, 10)
pet_name = line_input.show()
```

Methods

init([title, label, default, filter, config])	param title The short title of the dialog. Only used when the dialog is not
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the frame buffer.
show()	Show the dialog and execute the event loop.

Attributes

config	Get and set the config object (UiConfig).
label	Get and set the label of the dialog, it has to be a str
	or base. Text.
title	Get and set the title of the dialog, it has to be a str.
user_input	Facility to store and retrieve the user input.

config

Get and set the config object (*UiConfig*).

label

Get and set the label of the dialog, it has to be a str or base. Text.

 $render_to_buffer$ (buffer, row, column, buffer_height, buffer_width) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

show()

Show the dialog and execute the event loop. Until this method returns, all keyboards event are processed by the local event loop. This is also true if called from the main event loop.

This event loop returns the either "" or what is displayed in the input field.

Example:

```
value = line_input.show()
```

title

Get and set the title of the dialog, it has to be a str.

user_input

Facility to store and retrieve the user input.

Menu

```
class pygamelib.gfx.ui.Menu (title: pygamelib.base.Text = None, entries: list = None, padding: int = 1, config: pygamelib.gfx.ui.UiConfig = None)
Bases: object
```

The Menu object consists of a list of other Menu objects and/or MenuAction objects.

It has a title that is used in a MenuBar and the list of its entries is displayed when the menu is expanded.

A Menu object can contains an arbitrary number of entries with an arbitrary depth of submenus.

```
__init__(title: pygamelib.base.Text = None, entries: list = None, padding: int = 1, config: pygamelib.gfx.ui.UiConfig = None) \rightarrow None The constructor takes the following parameters.
```

Parameters

- **title** (str | *Text*) The title of the action (i.e. its label)
- entries (list) A list of MenuAction or other Menu objects.
- **padding** (*int*) The horizontal padding, i.e the number of space characters added to the left and right of the title.
- **config** (*UiConfig*) The configuration object.

Example

Methods

activate() Activates the menu. add_entry(entry) Add an entry to the menu. collapse() Collapse the menu. current_entry() Return the currently selected menu entry. expand() Expand the menu. menu_width() Calculate and return the maximum width of the menu. based on the widest element.
collapse() Collapse the menu. current_entry() Return the currently selected menu entry. expand() Expand the menu. menu_width() Calculate and return the maximum width of the menu. based on the widest element.
current_entry() Return the currently selected menu entry. expand() Expand the menu. menu_width() Calculate and return the maximum width of the menu. based on the widest element.
expand() Expand the menu. menu_width() Calculate and return the maximum width of the maximum width width width width width width width width widt
menu_width() Calculate and return the maximum width of the maximum width width width width of the maximum width width width width width width width width w
based on the widest element.
render_to_buffer(buffer, row, column,) Render the object from the display buffer to
frame buffer.
select_next() Select the next entry in the menu.

Continued on next page

Table 119 - continued from previous page

select_previous()	Select the previous entry in the menu.
title_width()	Return the actual width of the menu title.

Attributes

config	Get / set the config of the Menu, it needs to be a
	UiConfig.
entries	Get / set the entries of the Menu, it needs to be a list
	of MenuAction objects.
padding	Get / set the padding before and after the menu, it
	needs to be an int.
selected	Get / set the selected status of the Menu, it needs to
	be a boolean.
title	Get / set the title of the Menu, it needs to be a Text
	object or a python str.

activate()

Activates the menu. This method contains its own event loop a bit like the show() methods of Dialogs. It expands the menu if it wasn't already the case and listen to keyboard key strokes.

- SPACE or ENTER activates (i.e execute) menu actions.
- DOWN select the next entry.
- UP select the previous entry.
- ESC or LEFT close the menu.
- RIGHT activate (i.e expand) a submenu.

Example:

```
menu.activate()
```

add_entry(entry)

Add an entry to the menu. An entry can be a MenuAction or a Menu. Entries are displayed in the order of there additions from left to right.

Important: The config of the entry is overwritten by the config of the Menu. That is why it's not mandatory for *Menu and MenuAction*.

Parameters entry (MenuAction | Menu) - The entry to add.

Example:

```
menu.add_entry( Menu('File') )
menu.add_entry( MenuAction('Exit', quit_application) )
```

collapse()

Collapse the menu. A menu is automatically collapsed after activation.

Example:

```
file_menu.collapse()
```

config

Get / set the config of the Menu, it needs to be a UiConfig.

current_entry()

Return the currently selected menu entry.

It can be either a Menu object or a MenuAction object.

entries

Get / set the entries of the Menu, it needs to be a list of MenuAction objects.

expand()

Expand the menu. A menu is automatically expanded when activated.

Example:

```
file_menu.expand()
```

menu width() \rightarrow int

Calculate and return the maximum width of the menu based on the widest element. This includes the padding.

Returns the menu width.

Return type int

padding

Get / set the padding before and after the menu, it needs to be an int.

The padding is only used when the menu is nested into another menu.

 $render_to_buffer$ (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

select_next()

Select the next entry in the menu.

The selected entry is rendered differently to give a visual feedback to the user. Please see the *UiConfig* class for the styling option available to the Menu object.

Example:

```
menu.select_next()
```

select_previous()

Select the previous entry in the menu.

The selected entry is rendered differently to give a visual feedback to the user. Please see the *UiConfig* class for the styling option available to the Menu object.

Example:

```
menu.select_previous()
```

selected

Get / set the selected status of the Menu, it needs to be a boolean.

This changes the representation (way it's drawn) of the menu entry.

title

Get / set the title of the Menu, it needs to be a Text object or a python str.

The title is used in the MenuBar. In the following image, the title of the expanded menu is "File".



$title_width() \rightarrow int$

Return the actual width of the menu title. This takes into account the padding.

Example:

```
menu.title_width()
```

MenuAction

```
class pygamelib.gfx.ui.MenuAction(title: pygamelib.base.Text = None, action=None, parameter=None, padding: int = 1, config: pygamelib.gfx.ui.UiConfig = None)
```

Bases: object

A menu action is a menu entry that executes a callback when activated. Usually a Menuaction represents an action from the user interface like open file, save, quit, etc.

Therefor a MenuAction is fairly simple, at its simplest it has a title and a callable reference to a function.

An action cannot be used by itself but can be added to a MenuBar or a Menu.

Like everything in the UI module, MenuAction are styled through a *UiConfig* object. Unlike the other classes of that module however, the configuration object is not mandatory when instanciating this class. The reason is that the *MenuBar* object impose the configuration to its managed *MenuAction* and *Menu*.

```
__init__ (title: pygamelib.base.Text = None, action=None, parameter=None, padding: int = 1, config: pygamelib.gfx.ui.UiConfig = None) \rightarrow None The constructor takes the following parameters.
```

Parameters

• title (str | Text) - The title of the action (i.e. its label)

3.7. qfx 529

- **action** (callable) A reference to a callable function that is going to be executed when the action is activated. If set to None, nothing will happen when the action is activated.
- parameter (Any) A parameter that is passed to the callback action if not None.
- **padding** (*int*) The horizontal padding, i.e the number of space characters added to the left and right of the action.
- **config** (*UiConfig*) The configuration object.

Example

Methods

init(title[, action, parameter])	The constructor takes the following parameters.
activate()	Execute and return the result of the callback.
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.
title_width()	Return the actual width of the action's title.

Attributes

action	Get / set the action's callback, it needs to be a
	callable.
config	Get / set the config of the MenuAction, it needs to be
	a UiConfig.
padding	Get / set the padding before and after the menu ac-
	tion, it needs to be an int.
selected	Get / set the selected of the MenuAction, it needs to
	be a boolean.
title	Get / set the title of the action, it needs to be a str or
	a Text object.

action

Get / set the action's callback, it needs to be a callable.

activate()

Execute and return the result of the callback.

Example:

```
file_save_action.activate()
```

config

Get / set the config of the MenuAction, it needs to be a *UiConfig*.

padding

Get / set the padding before and after the menu action, it needs to be an int.

 $render_to_buffer$ (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

selected

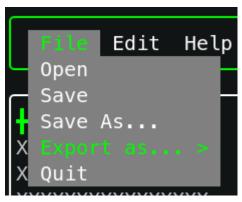
Get / set the selected of the MenuAction, it needs to be a boolean.

This changes the representation (way it's drawn) of the menu entry.

title

Get / set the title of the action, it needs to be a str or a Text object.

The title is used in the *Menu*. In the following image, the title of the first action in the expanded menu is "Open", followed by "Save".



title_width()

Return the actual width of the action's title. This takes into account the padding.

Example:

```
menu_action.title_width()
```

MenuBar

```
class pygamelib.gfx.ui.MenuBar(entries: list = None, spacing: int = 2, config: pygamelib.gfx.ui.UiConfig = None)

Bases: object
```

The MenuBar widget is exactly that: an horizontal bar that can hold Menu or MenuAction objects.

Contrary to these 2 classes, MenuBar does *not* have an activate() method. The reason is that the menubar cannot block rendering with its own event loop as it is supposed to be showned at all times. So the management of interactions are left to the programmer to implement.

A typical implementation would look like this:

Example:

```
# First create a menubar
menubar = MenuBar(config=UiConfig.instance(game=Game.instance()))
# Then create a Menu
file_menu = Menu(
    "File",
    Γ
        MenuAction("Open", open_file),
        MenuAction("Save", save_file),
       MenuAction("Save as", save_file_as),
        MenuAction("Quit", exit_application),
    ]
menubar.add_entry( file_menu )
menubar.add_entry( MenuAction("Help", display_help) )
# Place the menubar on screen
screen.place(menubar, 0, 0)
screen.update()
# Then, somewhere in an event loop, manage the inputs for example in the user
# update function
def user_update(game, inkey, elapsed_time):
    if inkey == engine.key.DOWN:
        if menubar.current_entry() is not None:
            menubar.current_entry().activate()
    elif inkey == engine.key.LEFT:
        menubar.select_previous()
    elif inkey == engine.key.RIGHT:
        menubar.select_next()
    elif inkey.name == "KEY_ENTER":
        if menubar.current_entry() is not None:
            menubar.current_entry().activate()
    elif inkey.name == "KEY_ESCAPE":
        menubar.close()
```

__init__ (*entries: list = None, spacing: int = 2, config: pygamelib.gfx.ui.UiConfig = None*) \rightarrow None The constructor takes the following parameters.

Parameters

- entries (list) A list of MenuAction or Menu objects.
- **spacing** The horizontal spacing between entries.

• **config** (*UiConfig*) – The configuration object.

Methods

init(entries, spacing, config)	The constructor takes the following parameters.
add_entry(entry)	Add an entry to the menu bar.
close()	Close and unselect menu entries/submenu.
current_entry()	Return the currently selected menu entry.
length()	Returns the total length of the menubar.
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.
select_next()	Select the next element in the menubar.
select_previous()	Select the previous element in the menubar.

Attributes

config	Get / set the config of the MenuBar, it needs to be a
	UiConfig.
current_index	Get / set the currently selected menu entry, it needs
	to be an int.
entries	Get / set the entries of the MenuBar, it needs to be a
	list of MenuAction or Menu objects.
spacing	Get / set the spacing between menu entries, it needs
	to be an int.

add_entry(entry)

Add an entry to the menu bar. An entry can be a *MenuAction* or a *Menu*. Entries are displayed in the order of there additions from left to right.

Important: The config of the entry is overwritten by the config of the MenuBar. That is why it's not mandatory for *Menu* and *MenuAction*.

Parameters entry (MenuAction | Menu) - The entry to add.

Example:

```
menubar.add_entry( Menu('File') )
menubar.add_entry( MenuAction('Exit', quit_application) )
```

close()

Close and unselect menu entries/submenu.

Please call that method when the menu bar loses focus.

config

Get / set the config of the MenuBar, it needs to be a UiConfig.

Important: The MenuBar's config is imposed on the managed items (Menu and MenuAction).

current entry()

Return the currently selected menu entry.

It can be either a Menu object or a MenuAction object.

current index

Get / set the currently selected menu entry, it needs to be an int. When setting the current_index, if the previous index was corresponding to a selected entry, said entry is first unselected.

entries

Get / set the entries of the MenuBar, it needs to be a list of MenuAction or Menu objects.

length() \rightarrow int

Returns the total length of the menubar. This is computed everytime the method is called and it includes the spacing.

render_to_buffer (buffer, row: int, column: int, buffer_height: int, buffer_width: int) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

select_next()

Select the next element in the menubar.

Example

```
if user_input.name == 'KEY_RIGHT':
    menubar.select_next()
```

$select_previous()$

Select the previous element in the menubar.

Example

```
if user_input.name == 'KEY_RIGHT':
    menubar.select_previous()
```

spacing

Get / set the spacing between menu entries, it needs to be an int.

MessageDialog

```
class pygamelib.gfx.ui.MessageDialog(data: list = None, width: int = 20, height: int = None, adaptive\_height: bool = True, alignment: int = None, title: <math>str = None, config: pygamelib.gfx.ui.UiConfig = None)
```

Bases: pygamelib.gfx.ui.Dialog

The message dialog is a popup that can display multiple lines of text.

It supports formatted text (base.Text), python strings, pygamelib.gfx.core.Sprixel, core. Sprite and more generally anything that can be rendered on screen (i.e. posess a render_to_buffer(self, buffer, row, column, buffer height, buffer width) method).

Each line can be aligned separately using constants.ALIGN_RIGHT, constants.ALIGN_LEFT or constants.ALIGN_CENTER. Please see add_line().

It also implements the *show()* virtual method of *Dialog*. This method is blocking and has its own event loop. It does not return anything.

ESC or ENTER close the dialog.

For the moment, the full message dialog needs to be displayed on screen. There is no pagination, but it is going to be implemented in a future release.

As all dialogs it also has a *user_input* property that reflects the user input. It is not used here however.

Like all dialogs, it is automatically destroyed on exit of the show() method. It is also deleted from the screen buffer.

Todo: Implements pagination.

```
__init__ (data: list = None, width: int = 20, height: int = None, adaptive_height: bool = True, alignment: int = None, title: str = None, config: pygamelib.gfx.ui.UiConfig = None) \rightarrow None
```

Parameters

- data (list) A list of data to display inside the MessageDialog. Elements in the list can contain various data types like base. Text, python strings, pygamelib.gfx.core. Sprixel, core.Sprite
- width (int) The width of the message dialog widget (in number of screen cells).
- **height** (*int*) The height of the message dialog widget (in number of screen cells).
- adaptive_height (bool) If True, the dialog height will be automatically adapted to match the content size.
- alignment (int) The alignment to apply to the data parameter. Please use the constants. ALIGN * constants. The default value is constants. ALIGN LEFT
- **title** (str) The short title of the dialog. Only used when the dialog is not borderless.
- config (*UiConfig*) The configuration object.

Example:

(continues on next page)

3.7. qfx 535

(continued from previous page)

```
screen.place(msg, 10, 10)
msg.show()
```

Methods

init(data, width, height,)	param data A list of data to display inside the MessageDialog. Elements in
add_line(data[, alignment])	Add a line to the message dialog.
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.
show()	Show the dialog and execute the event loop.

Attributes

config	Get and set the config object (UiConfig).
height	Get and set the height of the message dialog, it has
	to be an int.
title	Get and set the title of the dialog, it has to be a str.
user_input	Facility to store and retrieve the user input.

add_line (data, alignment=30000011) \rightarrow None

Add a line to the message dialog.

The line can be any type of data that can be rendered on screen. This means that any object that expose a render_to_buffer(self, buffer, row, column, buffer_height, buffer_width) method can be added as a "line". Python strings are also obviously accepted.

Here is a non-exhaustive list of supported types:

- Text,
- python strings (str),
- Sprixel,
- Sprite,
- most board items,
- etc.

Parameters

- data (various) The data to add to the message dialog.
- alignment (constants.ALIGN_RIGHT | constants.ALIGN_LEFT | constants.ALIGN_CENTER) The alignment of the line to add.

Example:

```
msg.add_line(
   base.Text(
     'This is centered and very red',
     core.Color(255,0,0),
   ),
   constants.ALGIN_CENTER,
)
```

config

Get and set the config object (UiConfig).

height

Get and set the height of the message dialog, it has to be an int.

```
render_to_buffer (buffer, row, column, buffer_height, buffer_width) \rightarrow None Render the object from the display buffer to the frame buffer.
```

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

$show() \rightarrow None$

Show the dialog and execute the event loop. Until this method returns, all keyboards event are processed by the local event loop. This is also true if called from the main event loop.

This event loop returns the key pressed.

Example:

```
key_pressed = msg.show()
if key_pressed.name = 'KEY_ENTER':
    // do something
else:
    print('Good bye')
```

title

Get and set the title of the dialog, it has to be a str.

user_input

Facility to store and retrieve the user input.

MultiLineInputDialog

The MultiLineInputDialog behave essentially like the LineInputDialog but is more configurable to allow the user to enter and edit a multiple lines of text.

Each field of this dialog can be individually configured to accept either anything printable or only digits.

The show() method returns the user input.

Key mapping:

- ESC: set the user input to "" and exit from the show() method.
- ENTER: Exit from the show() method. Returns the user input.
- BACKSPACE / DELETE: delete a character (both keys have the same result).
- TAB: cycle through the fields.
- All other keys input characters in the input field.

In all cases, when the dialog is closed, the user input is returned.

Like all dialogs, it is automatically destroyed on exit of the show() method. It is also deleted from the screen buffer.

```
__init__ (fields=[{'label': 'Input a value:', 'default': ", 'filter': 50000002}], title: str = None, config=None) \rightarrow None
```

Parameters

- **fields** (*list*) A list of dictionnary that represent the fields to present to the user. Please see bellow for a description of the dictionnary.
- **title** (str) The short title of the dialog. Only used when the dialog is not borderless.
- **config** (*UiConfig*) The configuration object.

The fields needs to be a list that contains dictionaries. Each of the dictionaries needs to contain 3 fields:

- "label": A one line instruction displayed over the field. This is a string.
- "default": A string that is going to pre-fill the input field.
- "filter": A filter to configure the acceptable inputs.

The filters are coming from the constants module and can be either constants.INTEGER_FILTER or constants.PRINTABLE_FILTER.

Example:

```
fields = [
        "label": "Enter the height of the new sprite:",
        "default": "",
        "filter": constants.INTEGER_FILTER,
    },
        "label": "Enter the width of the new sprite:",
        "default": "",
        "filter": constants.INTEGER_FILTER,
    },
        "label": "Enter the name of the new sprite:",
        "default": f"Sprite {len(sprite_list)}",
        "filter": constants.PRINTABLE_FILTER,
    },
multi_input = MultiLineInput(fields, conf)
screen.place(multi_input, 10, 10)
completed_fields = multi_input.show()
```

Methods

init([fields, config])	
	param fields A list of dictionnary that
	represent the fields to present to the
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the
	frame buffer.
show()	Show the dialog and execute the event loop.

Attributes

config	Get and set the config object (UiConfig).
fields	Get and set the fields of the dialog, see the construc-
	tor for the format or this list.
title	Get and set the title of the dialog, it has to be a str.
user_input	Facility to store and retrieve the user input.

config

Get and set the config object (UiConfig).

fields

Get and set the fields of the dialog, see the constructor for the format or this list.

 $render_to_buffer$ (buffer, row, column, buffer_height, buffer_width) \rightarrow None Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

show()

Show the dialog and execute the event loop. Until this method returns, all keyboards event are processed by the local event loop. This is also true if called from the main event loop.

This event loop returns a list of dictionaries with the content of each fields. The list of dictionaries is the same than the fields constructor parameter but each key has an additional 'user_input' field that contains the user input.

If the fields parameter was:

The returned value would be:

Example:

```
fields = multi_input.show()
```

title

Get and set the title of the dialog, it has to be a str.

user_input

Facility to store and retrieve the user input.

ProgressBar

A simple horizontal progress bar widget.

```
__init__ (value=0, maximum=100, width=20, progress_marker='', empty_marker='', config=None)
```

Parameters

- **value** (*int*) The initial value parameter. It represents the progression.
- maximum (int) The maximum value held by the progress bar. Any value over the maximum is ignored.
- width (int) The width of the progress bar widget (in number of screen cells).
- **progress_marker** (*pygamelib.gfx.core.Sprixel*) The progress marker is displayed on progression. It is the sprixel that fills the bar. Please see below.
- **empty_marker** (*pygamelib.gfx.core.Sprixel*) The empty marker is displayed instead of the progress marker when the bar should be empty (when the value is too low to fill the bar for example). Please see below.
- **config** (*UiConfig*) The configuration object.

Here is a representation of were the progress and empty markers are used.

```
Progress marker

[====----]

Empty marker
```

Example:

```
# Create a default progress bar with the default configuration
progress_bar = ProgressBar(config=UiConfig.instance())
# Place the progress bar in the middle of the screen
screen.place(
    progress_bar, screen.vcenter, screen.hcenter - int(progress_bar.width)
)
for progress in range(progress_bar.maximum + 1):
    # Do something useful
    progress_bar.value = progress
    screen.update()
```

Methods

init([value, maximum, width,])	param value The initial value parameter. It represents the progression.
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the frame buffer.

Attributes

config	Get and set the config object (UiConfig).
empty_marker	Get and set the empty marker, preferrably a
	Sprixel but could be a str.
maximum	Get and set the maximum possible progress, it has to
	be an int.
progress_marker	Get and set the progress marker, preferrably a
	Sprixel but could be a str.
value	Get and set the current progress value, it has to be an
	int.

config

Get and set the config object (*UiConfig*).

empty_marker

Get and set the empty marker, preferrably a Sprixel but could be a str.

maximum

Get and set the maximum possible progress, it has to be an int.

progress_marker

Get and set the progress marker, preferrably a Sprixel but could be a str.

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row(int) The row to render in.

- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

value

Get and set the current progress value, it has to be an int.

ProgressDialog

```
class pygamelib.gfx.ui.ProgressDialog(label=Progress dialog[0m, value=0, maximum=100, width=20, progress_marker='', empty_marker='', adaptive_width=True, destroy_on_complete=True, config=None)
```

Bases: pygamelib.gfx.ui.Dialog

ProgressDialog is a progress bar widget as a dialog (or popup). The main difference with a progress bar with borders is that it is automatically rendered on the second pass by the screen object (therefore, is visible on top of other graphical elements).

This dialog requires external interactions so it is the only dialog widget that does not provide a useful show() implementation. As a matter of fact, show do nothing at all.

ProgressDialog is mainly a label, a box and a *ProgressBar* bundled together.

```
__init__(label=Progress dialog[0m, value=0, maximum=100, width=20, progress_marker='', empty_marker='', adaptive_width=True, destroy_on_complete=True, config=None)

The constructor accepts the following parameters.
```

Parameters

- label (str | base. Text) A label to display on top of the progress bar.
- **value** (*int*) The initial value parameter. It represents the progression.
- maximum (int) The maximum value held by the progress bar. Any value over the maximum is ignored.
- width (int) The width of the progress bar widget (in number of screen cells).
- **progress_marker** (*pygamelib.gfx.core.Sprixel*) The progress marker is displayed on progression. It is the sprixel that fills the bar. Please see below.
- **empty_marker** (*pygamelib.gfx.core.Sprixel*) The empty marker is displayed instead of the progress marker when the bar should be empty (when the value is too low to fill the bar for example). Please see below.
- adaptive_width (bool) If True, the dialog will automatically adapt to the size of the label.
- **destroy_on_complete** If True, the dialog will remove itself from the screen when complete (i.e: when value == maximum)
- **config** (*UiConfig*) The configuration object.

Example:

```
# Create a default progress bar with the default configuration
progress_dial = ProgressDialog(
    "Please wait while I'm doing something super duper important",
    config=UiConfig.instance(),
```

(continues on next page)

(continued from previous page)

```
# Place the progress bar in the middle of the screen
screen.place(
    progress_dial, screen.vcenter, screen.hcenter - int(progress_bar.width)
)
for progress in range(progress_dial.maximum + 1):
    # Do something useful
    progress_dial.value = progress
    screen.update()
```

Methods

init([label, value, maximum, width,])	The constructor accepts the following parameters.	
render_to_buffer(buffer, row, column,)	Render the object from the display buffer to the	
	frame buffer.	
show()	The show method does nothing in the ProgressDia-	
	log.	

Attributes

config	Get and set the config object (UiConfig).
label	Get and set the label of the dialog, it has to be a str
	or base. Text.
maximum	Get and set the maximum possible progress, it has to
	be an int.
user_input	Facility to store and retrieve the user input.
value	Get and set the current progress value, it has to be an
	int.

config

Get and set the config object (UiConfig).

label

Get and set the label of the dialog, it has to be a str or base. Text.

maximum

Get and set the maximum possible progress, it has to be an int.

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render the object from the display buffer to the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

show()

The show method does nothing in the ProgressDialog. It is a notable exception and the only dialog widget in the UI module to do that.

user_input

Facility to store and retrieve the user input.

value

Get and set the current progress value, it has to be an int.

UiConfig

```
class pygamelib.gfx.ui.UiConfig(game=None,
                                                                           box vertical border='|',
                                                                            box_top_left_corner=
                                          box_horizontal_border='-',
                                          box_top_right_corner='',
                                                                         box_bottom_left_corner=''
                                          box_bottom_right_corner='',
                                                                        box_vertical_and_right='\',
                                          box_vertical_and_left='', fg_color=Color(255, 255, 255),
                                          bg_color=Color(0, 128, 128), fg_color_inactive=Color(128,
                                                128), bg_color_selected=Color(128,
                                                                                     128,
                                          bg_color_not_selected=None,
                                                                        fg\_color\_selected=Color(0,
                                          255, 0), fg_color_not_selected=Color(255, 255, 255),
                                          bg_color_menu_not_selected=Color(128, 128, 128), bor-
                                          der_fg_color=Color(255, 255, 255), border_bg_color=None,
                                          borderless dialog=True)
```

Bases: object

A configuration object for the UI module. TEST

This object's purpose is to configure the look and feel of the UI widgets. It does nothing by itself.

Parameters

- game (Game) The game object.
- **box_vertical_border** (*str*) The vertical border of a box.
- **box_horizontal_border** (*str*) The horizontal border of a box.
- box_top_left_corner (str) The top left corner of a box.
- **box_top_right_corner** (*str*) The top right corner of a box.
- **box_bottom_left_corner** (*str*) The bottom left corner of a box.
- box_bottom_right_corner (str) The bottom right corner of a box.
- box_vertical_and_right (str) The left junction between two boxes.
- box_vertical_and_left (str) The right junction between two boxes.
- $fg_color(Color)$ The foreground color (for text and content).
- $bg_color(Color)$ The background color (for text and content).
- **fg_color_inactive** (*Color*) The foreground color for inactive items like menu entries.
- **bg_color_selected** (Color) The background color (for selected text and content).
- **bg_color_not_selected** (Color) The background color (for non selected text and content).
- $fg_color_selected(Color)$ The foreground color (for selected text and content).

- **fg_color_not_selected** (Color) The foreground color (for non selected text and content).
- **bg_color_menu_not_selected** (*Color*) The menu background color (for expanded menu items).
- border_fg_color (Color) The foreground color (for borders).
- border_bg_color (Color) The background color (for borders).
- borderless_dialog (bool) Is the dialog borderless or not.

Example:

```
config_ui_red = UiConfig(
   fg_color=Color(255,0,0),
   border_fg_color=Color(255,0,0)
)
```

__init__ (game=None, box_vertical_border='|', box_horizontal_border='-', box top left corner="', box top right corner="', box bottom left corner=" box_bottom_right_corner='', box_vertical_and_right='\', box_vertical_and_left='', $fg\ color=Color(255,$ 255, 255), bg color=Color(0,128), fg_color_inactive=Color(128, 128, 128), bg_color_selected=Color(128, 128. bg_color_not_selected=None, $fg_color_selected=Color(0,$ 0), fg_color_not_selected=Color(255, 255, 255), bg_color_menu_not_selected=Color(128, 128, 128), border_fg_color=Color(255, 255, 255), border_bg_color=None, borderless_dialog=True)

Initialize self. See help(type(self)) for accurate signature.

Methods

init([game, box_vertical_border,])	Initialize self.
instance(*args, **kwargs)	Returns the instance of the UiConfig object

```
classmethod instance(*args, **kwargs)
```

Returns the instance of the UiConfig object

Creates an UiConfig object on first call an then returns the same instance on further calls. Useful for a default configuration. It accepts all the parameters from the constructor.

Returns Instance of Game object

3.7.3 particles

New in version 1.3.0.

Starting with version 1.3.0, the pygamelib now provides a particle system. It is for now a first limited version and it has a number of limitations.

First, the particles are "non interactive" objects. They are not affected by board items or anything drawn on screen nor can they affect them. All particles are drawn on top of an already rendered screen.

This means no fancy particle physics out of the box. It doesn't means that it is not doable. It just means that it is not existing out of the box.

Second, although I did my best to make the particle system as efficient as possible, drawing a lot of moving elements in the terminal is very slow. So be mindful of the performances when using it.

Now despite the limitations, the particle system still allow to do some very cool stuff. Here is a video example:

This is the benchmark of the particle system, the code is available on Ghithub.

Important: Like the UI module, the particles system works exclusively with the screen buffer system (place, delete, render, update, etc.). It doesn't work with Screen functions tagged "direct display" like display_at().

CircleEmitter

class pygamelib.gfx.particles.CircleEmitter(emitter_properties:

pygamelib.gfx.particles.EmitterProperties =
None)

Bases: pygamelib.gfx.particles.ParticleEmitter

The CircleEmitter differs from the ParticleEmitter in only one thing: it emits its particle in a circular shape, like this:

Aside from that specificity it's exactly the same as a regular particle emitter.

__init__(emitter_properties: pygamelib.gfx.particles.EmitterProperties = None) \rightarrow None The CircleEmitter takes the same parameters than the ParticleEmitter and make use of Emitter-Properties.radius.

The radius is used as the initial distance from the center of the circle (i.e the emitter's position).

Methods

init(emitter_properties)	The CircleEmitter takes the same parameters than		
	the ParticleEmitter and make use of Emitter-		
	Properties.radius.		
apply_force(force)	Apply a force to all alive particles.		
attach(observer)	Attach an observer to this instance.		
detach(observer)	Detach an observer from this instance.		
emit(amount)	Emit a certain amount of particles.		
finished()	Returns True if the emitter is finished.		
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by		
value])	the observer.		
load(data)	Load a particle emitter from serialized data.		
notify([modifier])	Notify all the observers that a change occurred.		
render_to_buffer(buffer, row, column,)	Render all the particles of that emitter in the frame		
	buffer.		
resize_pool(new_size)	In substance, this method is an alias for		
	ParticleEmitter.particle_pool.		
	resize().		
serialize()	Serialize the particle emitter.		
store_screen_position(row, column)	Store the screen position of the object.		
toggle_active()	Toggle the emitter's state between active and inac-		
	tive.		
	Continued on next page		

546 Chapter 3. Tutorials

Table 134 – continued from previous page

	<u> </u>
update()	Update all the particles in the pool.

Attributes

active	Access and set the active property.		
column	Access and set the column property (i.e: x).		
particle_pool	This property holds this emitter's instance of a		
	ParticlePool.		
row	Access and set the row property (i.e: y).		
screen_column	A property to get/set the screen column.		
screen_row	A property to get/set the screen row.		
X	Access and set the x property (i.e: column).		
Y	Access and set the y property (i.e: row).		

active

Access and set the active property.

An emitter only emits particles if he is active. Emitted particles keeps being updated even if the emitter is not active anymore, for obvious reasons.

apply_force (force: pygamelib.base.Vector2D)

Apply a force to all alive particles.

The force needs to be a Vector2D.

Parameters force (Vector2D) – The force to apply to the particles.

Example:

```
my_emitter.apply_force(base.Vector2D(0,0.3)) # slight wind.
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property (i.e. x).

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

emit(amount: int = None)

Emit a certain amount of particles.

The emitter will request particles from the particle pool. This in turn will trigger the recycling of dead particles if needed.

Calling this method faster than the configured emit_rate is not going to emit more particles. An emitter cannot emit particles faster than its emit_rate.

If amount is None, the emitter emits emit_number particles.

Parameters amount (*int*) – The amount (number) of particles to be emitted.

Example:

```
my_emitter.emit(50)
```

finished()

Returns True if the emitter is finished.

A finished emitter has both:

- Reach the end of its lifespan (i.e lifespan == 0)
- And all particles are finished too.

This means that an emitter will, in most cases, not be finished as soon as its lifespan reaches 0 but a bit after. When all of its managed particles are dead.

This is on purpose for both aesthetic reasons (avoiding particles sudden removal) and for optimization (counting active particles is a O(n) operation and can be very long when there's a lot of particles so we want to do it only when necessary).

Example:

```
if my_emitter.finished():
    screen.delete(my_emitter.row, my_emitter.column)
```

handle_notification(subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

classmethod load (data)

Load a particle emitter from serialized data.

Parameters data (dict) – The serialized data.

Returns The loaded particle emitter.

Return type ParticleEmitter

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

particle_pool

This property holds this emitter's instance of a ParticlePool.

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render all the particles of that emitter in the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- **buffer** (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- column (int) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

```
resize_pool (new_size: int = None)
```

In substance, this method is an alias for ParticleEmitter.particle_pool.resize(). However, called without parameter, it will try to resize the particle pool to emit_number * particle_lifespan. It will do so only if the resulting number is greater than the current particle pool size.

Parameters new_size (int) - The desired new size of the pool.

Example:

```
my_emitter.resize_pool(3000)
```

row

Access and set the row property (i.e: y).

screen_column

A property to get/set the screen column.

```
Parameters value (int) – the screen column
```

Return type int

screen_row

A property to get/set the screen row.

```
Parameters value (int) – the screen row
```

Return type int

serialize()

Serialize the particle emitter.

Returns A dictionary containing all the emitter's properties.

Return type dict

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

toggle_active()

Toggle the emitter's state between active and inactive.

An inactive emitter does not emit new particles but keeps processing particles that have already been emitted.

Example:

```
if not my_emitter.active:
    my_emitter.toggle_active()
```

update()

Update all the particles in the pool.

Updating a particle means applying particle_acceleration to every particle and then call Particle.

update().

Example:

```
my_emitter.update()
```

v

Access and set the x property (i.e. column).

У

Access and set the y property (i.e: row).

ColorParticle

```
class pygamelib.gfx.particles.ColorParticle(row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None, sprixel: pygamelib.gfx.particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particles.Particle
```

Bases: pygamelib.gfx.particles.Particle

This class is an extension of *Particle*. It adds the possibility to gradually go from a starting color to an end color over time. It is linked with the lifespan of the particle.

```
__init___(row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifes-pan: int = None, sprixel: pygamelib.gfx.particles.ParticleSprixel = None, start_color: pygamelib.gfx.core.Color = None, stop_color: pygamelib.gfx.core.Color = None) \rightarrow None The constructor takes the following parameters.
```

Parameters

- row (int) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.
- **sprixel** (*Sprixel*) The sprixel that represent the particle when drawn on screen.
- **start color** (Color) The color of the particle at the beginning of its lifespan.
- **stop_color** (*Color*) The color of the particle at the end of its lifespan.

Example:

```
single_particle = ColorParticle(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
   lifespan=10,
   start_color=core.Color(255, 0, 0),
   stop_color=core.Color(0, 255, 0),
)
```

Methods

init(row, column, velocity, lifespan,)	The constructor takes the following parameters.	
apply_force(force)	Apply a force to the particle's acceleration vector.	
attach(observer)	Attach an observer to this instance.	
detach(observer)	Detach an observer from this instance.	
finished()	Return True if the particle is done living (i.e its lifes-	
	pan is lesser or equal to 0).	
handle_notification(subject[, attribute	e, A virtual method that needs to be implemented by	
value])	the observer.	
	Continued on next nage	

3.7. gfx 551

Continued on next page

Table	136 -	 continued 	from	previous page
IUDIO		COLLINGCA		pi ovious page

load(data)	Load a ColorParticle from a dictionary.
notify([modifier])	Notify all the observers that a change occurred.
render(sprixel)	Render the particle as a Sprixel.
reset(row, column, velocity, lifespan)	Reset a particle in its initial state.
reset_lifespan(lifespan)	Reset the particle lifespan (including the initial lifes-
	pan).
serialize()	Serialize a ColorParticle into a dictionary.
store_screen_position(row, column)	Store the screen position of the object.
terminate()	Terminate a particle, i.e sets its lifespan to -1.
update()	The update method perform the calculations required
	to process the new particle position.

Attributes

column	Access and set the column property.
row	Access and set the row property.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
X	Access and set the x property.
Y	Access and set the y property.

$\verb"apply_force" (force: pygamelib.base.Vector2D)" \to None$

Apply a force to the particle's acceleration vector.

You are more likely to apply forces to all particles of an emitter through the apply_force() method of the emitter class.

Parameters force (Vector2D) – The force to apply.

Example:

```
gravity = Vector2D(-0.2, 0.0)
my_particle.apply_force(gravity)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property. Equivalent to the x property.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (*PglBaseObject*) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

finished() \rightarrow bool

Return True if the particle is done living (i.e its lifespan is lesser or equal to 0). It returns False otherwise.

Return type bool

Example:

```
if not my_particle.finished():
   my_particle.update()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

classmethod load(data)

Load a ColorParticle from a dictionary.

Parameters data (dict) – The dictionary to load from

Returns The loaded ColorParticle

Return type ColorParticle

Example:

```
particle = ColorParticle.load( json.load( open("particle.json") ) )
```

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None

Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.

• **value** (Any) – An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

render (sprixel: pygamelib.gfx.core.Sprixel = None)

Render the particle as a *Sprixel*. This method is called by the *ParticleEmitter* render_to_buffer method.

It takes a *Sprixel* as a parameter. This Sprixel is given by the ParticleEmitter.render_to_buffer() method and if it is not None, the particle will render itself into that *Sprixel* and return it.

Important: This method must be called after everything else as rendered or else there will be *Sprixel* that will be overwritten during their rendering cycle. Other elements could also have their *Sprixel* corrupted and replaced by the particle's one.

Parameters sprixel (Sprixel) - A sprixel already rendered in the screen buffer.

Example:

```
p = my_particle
buffer[p.row][p.column] = p.render(buffer[p.row][p.column])
```

reset (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None) Reset a particle in its initial state. This is particularly useful for the reuse of particles.

This method takes almost the same parameters than the constructor.

Parameters

- row (int) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.

Example:

```
single_particle.reset(
    row=5,
    column=5,
    velocity=base.Vector2D(-0.5, 0.0),
    lifespan=10,
)
```

$reset_lifespan (lifespan: int = 20) \rightarrow None$

Reset the particle lifespan (including the initial lifespan).

Parameters lifespan (*int*) – The particle lifespan in number of movements/turns.

Example:

```
my_particle.reset_lifespan(10)
```

row

Access and set the row property. Equivalent to the y property.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (*int*) – the screen row

Return type int

serialize()

Serialize a ColorParticle into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( particle.serialize() )
```

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

$\textbf{terminate}\,()\,\to None$

Terminate a particle, i.e sets its lifespan to -1.

In that case the ParticleEmitter and ParticlePool will recycle it. That is *IF* you are managing the particle through an emitter and/or a pool of course.

Example:

```
p = my_particle
if p.row >= screen, height or p.column >= screen.width:
    p.terminate()
```

update()

The update method perform the calculations required to process the new particle position. It mainly adds the acceleration to the velocity vector and update the position accordingly.

After calling update() the acceleration is "consumed" in the velocity and therefor reset.

The update() method takes no parameters and returns nothing.

Example:

```
my_particle.update()
```

x

Access and set the x property. Equivalent to the column property.

У

Access and set the y property. Equivalent to the row property.

ColorPartitionParticle

```
class pygamelib.gfx.particles.ColorPartitionParticle(row:
                                                                               = 0.
                                                                                         col-
                                                                          int = 0, velocity:
                                                                  umn:
                                                                  pygamelib.base.Vector2D
                                                                  = None, lifespan:
                                                                                       int =
                                                                  None, partition: list = None,
                                                                  partition_blending_table:
                                                                  list = None, start color:
                                                                  pygamelib.gfx.core.Color
                                                                        None.
                                                                                   stop color:
                                                                  pygamelib.gfx.core.Color
                                                                  None)
```

Bases: pygamelib.gfx.particles.PartitionParticle

This class is basically the same as ColorParticle but its base class is PartitionParticle instead of Particle. Everything else is the same.

It serves the same purpose as the ColorParticle with the added partition particle capabilities.

```
__init__ (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None, partition: list = None, partition_blending_table: list = None, start_color: pygamelib.gfx.core.Color = None, stop_color: pygamelib.gfx.core.Color = None) → None The constructor takes the following parameters.
```

Parameters

- row (int) The initial row position of the particle on the screen.
- column(int) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.
- partition (list) The partition of the particle.
- partition_blending_table (list) The blending table of the particle.
- **start_color** (*Color*) The color of the particle at the beginning of its lifespan.
- **stop_color** (*Color*) The color of the particle at the end of its lifespan.

Example:

```
single_particle = RandomColorPartitionParticle(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
```

(continues on next page)

(continued from previous page)

```
lifespan=10,
```

Methods

init(row, column, velocity, lifespan,)	The constructor takes the following parameters.
apply_force(force)	Apply a force to the particle's acceleration vector.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
finished()	Return True if the particle is done living (i.e its lifes-
	pan is lesser or equal to 0).
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load a ColorPartitionParticle from a dictionary.
notify([modifier])	Notify all the observers that a change occurred.
render(sprixel)	This method first calls the Particle.render() method.
reset(row, column, velocity, lifespan)	Reset a particle in its initial state.
reset_lifespan(lifespan)	Reset the particle lifespan (including the initial lifes-
	pan).
serialize()	Serialize a ColorPartitionParticle into a dictionary.
store_screen_position(row, column)	Store the screen position of the object.
terminate()	Terminate a particle, i.e sets its lifespan to -1.
update()	This method first calls the Particle.update() method,
	then calculates the quadrant position, i.e. the actual
	position of the particle within a console character.

Attributes

column	Access and set the column property.
row	Access and set the row property.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
X	Access and set the x property.
Y	Access and set the y property.

$apply_force(force: pygamelib.base.Vector2D) \rightarrow None$

Apply a force to the particle's acceleration vector.

You are more likely to apply forces to all particles of an emitter through the $apply_force()$ method of the emitter class.

Parameters force (Vector2D) – The force to apply.

Example:

```
gravity = Vector2D(-0.2, 0.0)
my_particle.apply_force(gravity)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property. Equivalent to the x property.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PqlBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

$finished() \rightarrow bool$

Return True if the particle is done living (i.e its lifespan is lesser or equal to 0). It returns False otherwise.

Return type bool

Example:

```
if not my_particle.finished():
    my_particle.update()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

classmethod load(data)

Load a ColorPartitionParticle from a dictionary.

Parameters data (dict) – The dictionary to load from

Returns The loaded ColorPartitionParticle

Return type ColorPartitionParticle

Example:

```
particle = ColorPartitionParticle.load( json.load( open("particle.json") ) )
```

notify (*modifier=None*, *attribute:* str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

render (sprixel: pygamelib.gfx.core.Sprixel = None)

This method first calls the Particle.render() method. Then it updates the rendered particle's model based on the blending table.

Parameters sprixel (Sprixel) – A sprixel already rendered in the screen buffer.

Example:

```
p = my_particle
buffer[p.row][p.column] = p.render(buffer[p.row][p.column])
```

reset (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None) Reset a particle in its initial state. This is particularly useful for the reuse of particles.

This method takes almost the same parameters than the constructor.

Parameters

- row (int) The initial row position of the particle on the screen.
- column (int) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.

Example:

```
single_particle.reset(
    row=5,
    column=5,
    velocity=base.Vector2D(-0.5, 0.0),
    lifespan=10,
)
```

```
reset_lifespan (lifespan: int = 20) \rightarrow None
```

Reset the particle lifespan (including the initial lifespan).

Parameters lifespan (int) – The particle lifespan in number of movements/turns.

Example:

```
my_particle.reset_lifespan(10)
```

row

Access and set the row property. Equivalent to the y property.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (*int*) – the screen row

Return type int

serialize()

Serialize a ColorPartitionParticle into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( particle.serialize() )
```

store screen position (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

$\textbf{terminate}\,(\,)\,\to None$

Terminate a particle, i.e sets its lifespan to -1.

In that case the ParticleEmitter and ParticlePool will recycle it. That is *IF* you are managing the particle through an emitter and/or a pool of course.

Example:

```
p = my_particle
if p.row >= screen, height or p.column >= screen.width:
    p.terminate()
```

560 Chapter 3. Tutorials

update()

This method first calls the Particle.update() method, then calculates the quadrant position, i.e: the actual position of the particle within a console character. It then updates the particle's model based on this internal position.

Example:

```
my_particle.update()
```

x

Access and set the x property. Equivalent to the column property.

У

Access and set the y property. Equivalent to the row property.

EmitterProperties

```
class pygamelib.gfx.particles.EmitterProperties (row: int = 0, column: int = 0, variance: float = 2.0, emit_number: int = 1, emit_rate: float = 0.1, lifespan: int = 200, parent=None, particle_velocity=None, particle_acceleration=None, particle_lifespan: float = 5.0, radius: float = 1.0, particle: pygamelib.gfx.particles.Particle = 1.0
```

Bases: object

EmitterProperties is a class that hold configuration variables for a particle emitter. The idea is that it's easier to carry around for multiple emitters with the same configuration than multiple values in the emitter's constructor.

It holds all possible parmeters for all types of emitters. Emitters uses only the ones that they really need.

Important: In most cases these values are copied by the emitter's constructor. So changing the values during an emitter's alive cycle is not going to do anything.

Note: This class should be a @dataclass. However, support for keyword only data classes is specific to python 3.10+. So for now, it is a regular class.

```
__init__ (row: int = 0, column: int = 0, variance: float = 2.0, emit_number: int = 1, emit_rate: float = 0.1, lifespan: int = 200, parent=None, particle_velocity=None, particle_acceleration=None, particle_lifespan: float = 5.0, radius: float = 1.0, particle: pygamelib.gfx.particles.Particle = None) \rightarrow None
```

Parameters

- **row** (*int*) The row where the emitter is. It is only important for the first rendering cycle. After that, the emitter will know its position on screen.
- **column** (*int*) The row where the emitter is. It is only important for the first rendering cycle. After that, the emitter will know its position on screen.
- **variance** (float) The variance is the amount of randomness that is allowed when emitting a particle. The exact use of this parameter is specific to each emitter.

- **emit_number** (*int*) The number of particle emitted at each timer tick.
- **emit_rate** (*float*) The rate of emission in seconds. This value needs to be understood as "the emitter will emit **emit_number** particles every **emit_rate** seconds".
- **lifespan** (*int*) The lifespan of the emitter in number of emission cycle. If lifespan is set to 1 for example, the emitter will only emit one burst of particles.
- parent (BoardItem) A parent board item. If you do that manually, you will probably want to set it specifically for each emitter.
- particle_velocity (Vector2D) The initial particle velocity. Please read the documentation of each emitter for the specific use of particle velocity.
- particle_acceleration (Vector2D) The initial particle acceleration. Please read the documentation of each emitter for the specific use of particle acceleration.
- particle_lifespan (int) The lifespan of the particle in number of cycles.
- **radius** (*float*) For emitter that supports it (like the CircleEmitter), sets the radius of emission (which translate into a velocity vector for each particle).
- **particle** (*Particle*) The particle that the emitter will emit. This can be a class reference or a fully instantiated particle. Emitters will copy it in the particle pool.

Example:

```
props = EmitterProperties(emit_number=10, emit_rate=0.1, lifespan=10)
```

Methods

```
__init__(row, column, variance, emit_number,
...)

param row The row where the emitter
is. It is only important for the first

load(data)

Load an EmitterProperties from a dictionary.

serialize()

Serialize an EmitterProperties into a dictionary.
```

classmethod load(data)

Load an EmitterProperties from a dictionary.

Parameters data (dict) – The dictionary to load from.

Returns The EmitterProperties object

Return type EmitterProperties

Example:

serialize()

Serialize an EmitterProperties into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( emitter_properties.serialize() )
```

ParticleEmitter

```
class pygamelib.gfx.particles.ParticleEmitter(emitter_properties=None)
    Bases: pygamelib.base.PqlBaseObject
```

The particle emitter is a key piece of the pygamelib's particle system: it's the part that actually do something!

The emitter takes care of managing the particles' life cycle. It emits, move, apply forces, update and draw particles on screen. It also provide convenient methods to manage the particle pool or apply forces to all active particles in the pool.

Particle emitters are configured with *EmitterProperties*. This is a convenient way to place multiple emitters with the same configuration. For example, if you create a "torch fire" emitter, you can use the same properties to create multiple emitters. It's less cumbersome than having the parameters tied to an instance of the emitter.

Here is an example of that taken from examples/benchmark-particle-system:

Example:

```
# The torch fire properties
emt_props = particles.EmitterProperties(
    screen.vcenter, # Position is not important as it will be updated by the
    screen.hcenter, # ParticleEmitter.render_to_buffer method.
    lifespan=150,
    variance=0.3,
    emit number=10,
    emit rate=0.1.
    particle=particles.ColorPartitionParticle(
        start_color=core.Color(45, 151, 227),
        stop_color=core.Color(7, 2, 40),
    ),
    particle_lifespan=5,
    radius=0.4,
# Now create multiple emitters at different position with the same properties.
for c in [[20, 24], [20, 35], [20, 122], [20, 133]]:
   bench_state.particle_emitters.append(particles.CircleEmitter(emt_props))
    screen.place(
        bench state.particle emitters[-1],
        screen.vcenter - int(bench_state.altar_sprite.height / 2) + c[0],
        c[1],
        2, # Always set your emitters to be rendered on the second pass.
    )
```

Important: The entire particle system is build around the **Screen Buffer** system and is completely incompatible with the direct display system. If you want to use the particle system you **have to** use Screen.place() and the other methods of the **Screen Buffer** system.

An emitter should always be placed on screen and set to render on the second rendering pass.

It is important if you want to avoid artifacts (like particles being rendered only under the position of the emitter).

The particles by themselves are not able to render on screen, the emitter is doing that job for them.

It also means that the particles are rendered and displayed over a screen that is already rendered. Therefor, by default and for the moment, they cannot interact with elements on screen or items in a board. It also means that there is no built in particle physics (for the moment).

__init__(emitter_properties=None) → None The constructor takes the following parameter:

Parameters emitter_properties (*EmitterProperties*) – The properties of that particle emitter.

Methods

init([emitter_properties])	The constructor takes the following parameter:
apply_force(force)	Apply a force to all alive particles.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
emit(amount)	Emit a certain amount of particles.
finished()	Returns True if the emitter is finished.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load a particle emitter from serialized data.
notify([modifier])	Notify all the observers that a change occurred.
render_to_buffer(buffer, row, column,)	Render all the particles of that emitter in the frame
	buffer.
resize_pool(new_size)	In substance, this method is an alias for
	ParticleEmitter.particle_pool.
	resize().
serialize()	Serialize the particle emitter.
store_screen_position(row, column)	Store the screen position of the object.
toggle_active()	Toggle the emitter's state between active and inac-
	.•
	tive.

Attributes

active	Access and set the active property.
column	Access and set the column property (i.e: x).
particle_pool	This property holds this emitter's instance of a
	ParticlePool.
row	Access and set the row property (i.e: y).
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
X	Access and set the x property (i.e: column).
Y	Access and set the y property (i.e: row).

active

Access and set the active property.

An emitter only emits particles if he is active. Emitted particles keeps being updated even if the emitter is not active anymore, for obvious reasons.

564 Chapter 3. Tutorials

```
apply_force (force: pygamelib.base.Vector2D)
```

Apply a force to all alive particles.

The force needs to be a Vector2D.

Parameters force (Vector2D) – The force to apply to the particles.

Example:

```
my_emitter.apply_force(base.Vector2D(0,0.3)) # slight wind.
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property (i.e. x).

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

```
emit (amount: int = None) \rightarrow None
```

Emit a certain amount of particles.

The emitter will request particles from the particle pool. This in turn will trigger the recycling of dead particles if needed.

Calling this method faster than the configured emit_rate is not going to emit more particles. An emitter cannot emit particles faster than its emit_rate.

If amount is None, the emitter emits emit_number particles.

Parameters amount (int) – The amount (number) of particles to be emitted.

Example:

```
my_emitter.emit(50)
```

finished()

Returns True if the emitter is finished.

A finished emitter has both:

- Reach the end of its lifespan (i.e lifespan == 0)
- And all particles are finished too.

This means that an emitter will, in most cases, not be finished as soon as its lifespan reaches 0 but a bit after. When all of its managed particles are dead.

This is on purpose for both aesthetic reasons (avoiding particles sudden removal) and for optimization (counting active particles is a O(n) operation and can be very long when there's a lot of particles so we want to do it only when necessary).

Example:

```
if my_emitter.finished():
    screen.delete(my_emitter.row, my_emitter.column)
```

handle_notification(subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

classmethod load(data)

Load a particle emitter from serialized data.

Parameters data (dict) – The serialized data.

Returns The loaded particle emitter.

Return type ParticleEmitter

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (*str*) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

particle_pool

This property holds this emitter's instance of a ParticlePool.

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render all the particles of that emitter in the frame buffer.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- **row** (*int*) The row to render in.
- column (int) The column to render in.
- height (int) The total height of the display buffer.
- width (int) The total width of the display buffer.

```
resize_pool (new_size: int = None)
```

In substance, this method is an alias for ParticleEmitter.particle_pool.resize(). However, called without parameter, it will try to resize the particle pool to emit_number * particle_lifespan. It will do so only if the resulting number is greater than the current particle pool size.

Parameters new_size (int) - The desired new size of the pool.

Example:

```
my_emitter.resize_pool(3000)
```

row

Access and set the row property (i.e. y).

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serialize the particle emitter.

Returns A dictionary containing all the emitter's properties.

Return type dict

$\verb|store_screen_position|| (\textit{row: int, column: int})| \rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

toggle_active()

Toggle the emitter's state between active and inactive.

An inactive emitter does not emit new particles but keeps processing particles that have already been emitted.

Example:

```
if not my_emitter.active:
    my_emitter.toggle_active()
```

update()

Update all the particles in the pool.

Updating a particle means applying particle_acceleration to every particle and then call Particle.

update().

Example:

```
my_emitter.update()
```

X

Access and set the x property (i.e. column).

У

Access and set the y property (i.e: row).

ParticlePool

Bases: object

The particle pool is a structure that holds a large number of particles and make them available to the emitters.

Its main role is to optimize the performances (both speed and memory usage). It works by pre-instantiating a desired number of particles according to the *EmitterProperties* that is given to the constructor.

The particle pool is optimized to avoid searching for available particles. It sets its own size to avoid relying on anything but its last known particle made available to the emitter. So unless for specific behavior, it is probably a good idea to let it sets its own size.

It also recycle particles that are finished() to avoid a constant cycle of creation/destruction of a large amount of particle objects.

```
\_init\_ (size: int = None, emitter\_properties: pygamelib.gfx.particles.EmitterProperties = None) \rightarrow None
```

The constructor takes the following parameters:

Parameters

- **size** (*int*) The size of the pool in number of particles. For this to be efficient, be sure to have enough particles to cover for enough cycles before your first emitted particles are finished. The <code>ParticleEmitter</code> uses the following rule to size the pool: emit_rate * particle_lifespan. It is the default value if size is not specified.
- **emitter_properties** (*EmitterProperties*) The properties of the particles that needs to be pre-instantiated.

Example:

```
my_particle_pool = ParticlePool(500, my_properties)
```

Methods

init(size, emitter_properties)	The constructor takes the following parameters:
count_active_particles()	Returns the number of active particle (i.e not fin-
	ished) in the pool.
<pre>get_particles(amount)</pre>	Returns the requested amount of particles.
resize(new_size)	Resize the particle pool to a new size.

Attributes

pool	A read-only property that returns the particle pool
	tuple.

$count_active_particles() \rightarrow int$

Returns the number of active particle (i.e not finished) in the pool.

Important: The only way to know the amount of alive particles is to go through the entire pool. Be aware of the performance impact on large particle pools.

Returns the number of active particles.

Return type int

Example:

```
if emitter.particles.count_active_particles() > 0:
    emitter.apply_force(gravity)
```

$get_particles(amount: int = None) \rightarrow tuple$

Returns the requested amount of particles.

It is important to know that no particle is created during that call. This method returns available particles in the pool. Particles are recycled after they "died".

If amount is not specified the pool returns EmitterProperties.emit_number particles.

Parameters amount (int) – The amount of particles to return.

Returns A tuple containing the desired amount of particles.

Return type tuple

Example:

```
fresh_particles = my_particle_pool.get_particles(30)
```

pool

A read-only property that returns the particle pool tuple.

```
resize(new_size: int)
```

Resize the particle pool to a new size.

If the new size is greater than the old one, the pool will be filled by pre-instanciated particles. If it's shorter however, the extra particles will be destroyed.

Parameters new_size (int) - The new size of the pool.

Example:

```
# Resize the particle pool to hold 100 particles.
my_particle_pool.resize(100)
```

Particle

```
class pygamelib.gfx.particles.Particle (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None, sprixel: pygamelib.gfx.particles.ParticleSprixel = None)
```

Bases: pygamelib.base.PglBaseObject

New in version 1.3.0.

The Particle class is the base class that is inherited from by all other particles. It is mostly a "data class" in the sense that it is a class used for calculations but is not able to render on screen by itself. All operations are pure data operations until the emitter draw the particles.

Altought the Particle class can be used on its own, it is most likely to be used as a template for a particle emitter.

```
__init__(row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None, sprixel: pygamelib.gfx.particles.ParticleSprixel = None) \rightarrow None The constructor takes the following parameters.
```

Parameters

- row (int) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.
- **sprixel** (Sprixel) The sprixel that represent the particle when drawn on screen.

Example:

```
single_particle = Particle(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
   lifespan=10,
```

(continues on next page)

(continued from previous page)

```
sprixel=core.Sprixel(graphics.GeometricShapes.BLACK_CIRCLE)
```

Methods

init(row, column, velocity, lifespan,)	The constructor takes the following peremeters
	The constructor takes the following parameters.
apply_force(force)	Apply a force to the particle's acceleration vector.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
finished()	Return True if the particle is done living (i.e its lifes-
	pan is lesser or equal to 0).
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load a Particle from a dictionary.
notify([modifier])	Notify all the observers that a change occurred.
render(sprixel)	Render the particle as a Sprixel.
reset(row, column, velocity, lifespan)	Reset a particle in its initial state.
reset_lifespan(lifespan)	Reset the particle lifespan (including the initial lifes-
	pan).
serialize()	Serialize a Particle into a dictionary.
store_screen_position(row, column)	Store the screen position of the object.
terminate()	Terminate a particle, i.e sets its lifespan to -1.
update()	The update method perform the calculations required
	to process the new particle position.

Attributes

column	Access and set the column property.
row	Access and set the row property.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
X	Access and set the x property.
Y	Access and set the y property.

$apply_force(force: pygamelib.base.Vector2D) \rightarrow None$

Apply a force to the particle's acceleration vector.

You are more likely to apply forces to all particles of an emitter through the apply_force() method of the emitter class.

Parameters force (Vector2D) – The force to apply.

Example:

```
gravity = Vector2D(-0.2, 0.0)
my_particle.apply_force(gravity)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property. Equivalent to the x property.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PqlBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

$finished() \rightarrow bool$

Return True if the particle is done living (i.e its lifespan is lesser or equal to 0). It returns False otherwise.

Return type bool

Example:

```
if not my_particle.finished():
    my_particle.update()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- **value** (Any) The new value of the attribute. This can be None.

classmethod load(data)

Load a Particle from a dictionary.

Parameters data (dict) – The dictionary to load from

Returns The loaded Particle

Return type Particle

Example:

```
particle = Particle.load( json.load( open("particle.json") ) )
```

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- value (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

render (sprixel: pygamelib.gfx.core.Sprixel = None)

Render the particle as a *Sprixel*. This method is called by the *ParticleEmitter* render_to_buffer method.

It takes a *Sprixel* as a parameter. This Sprixel is given by the ParticleEmitter.render_to_buffer() method and if it is not None, the particle will render itself into that *Sprixel* and return it.

Important: This method must be called after everything else as rendered or else there will be <code>Sprixel</code> that will be overwritten during their rendering cycle. Other elements could also have their <code>Sprixel</code> corrupted and replaced by the particle's one.

Parameters sprixel (Sprixel) – A sprixel already rendered in the screen buffer.

Example:

```
p = my_particle
buffer[p.row][p.column] = p.render(buffer[p.row][p.column])
```

reset (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None) Reset a particle in its initial state. This is particularly useful for the reuse of particles.

This method takes almost the same parameters than the constructor.

Parameters

- row (int) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.

3.7. qfx 573

```
single_particle.reset(
    row=5,
    column=5,
    velocity=base.Vector2D(-0.5, 0.0),
    lifespan=10,
)
```

$reset_lifespan (lifespan: int = 20) \rightarrow None$

Reset the particle lifespan (including the initial lifespan).

Parameters lifespan (*int*) – The particle lifespan in number of movements/turns.

Example:

```
my_particle.reset_lifespan(10)
```

row

Access and set the row property. Equivalent to the y property.

screen column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serialize a Particle into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( particle.serialize() )
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column(int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

$\textbf{terminate}\,(\,)\,\to None$

Terminate a particle, i.e sets its lifespan to -1.

In that case the ParticleEmitter and ParticlePool will recycle it. That is *IF* you are managing the particle through an emitter and/or a pool of course.

Example:

```
p = my_particle
if p.row >= screen,height or p.column >= screen.width:
    p.terminate()
```

$update() \rightarrow None$

The update method perform the calculations required to process the new particle position. It mainly adds the acceleration to the velocity vector and update the position accordingly.

After calling update() the acceleration is "consumed" in the velocity and therefor reset.

The update() method takes no parameters and returns nothing.

Example:

```
my_particle.update()
```

x

Access and set the x property. Equivalent to the column property.

У

Access and set the y property. Equivalent to the row property.

ParticleSprixel

```
 \begin{array}{c} \textbf{class} \text{ pygamelib.gfx.particles.ParticleSprixel} (\textit{model}=", & \textit{bg\_color=None}, \\ \textit{fg\_color=None}, & \textit{fg\_color=None}, \\ \textit{is\_bg\_transparent=None}) \\ \textbf{Bases: pygamelib.gfx.core.Sprixel} \end{array}
```

New in version 1.3.0.

The ParticleSprixel is nothing more than a Sprixel. Its only role is to help differentiate rendered sprixels for Partition Particles.

```
__init__ (model=", bg_color=None, fg_color=None, is_bg_transparent=None)
```

Parameters

- model(str) The model, it can be any string. Preferrably a single character.
- **bg_color** (Color) A Color object to configure the background color.
- **fg_color** (Color) A Color object to configure the foreground color.
- **is_bg_transparent** (bool) Set the background of the Sprixel to be transparent. It tells the engine to replace the background of the Sprixel by the background color of the overlapped sprixel.

Methods

init([model, bg_color, fg_color,])	
	param model The model, it can be any
	string. Preferrably a single character.
attach(observer)	Attach an observer to this instance.
black_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.BLACK_RECT.
black_square()	This class method returns a
	sprixel that is the equivalent of
	pygamelib.assets.graphics.BLACK_SQUARE.
blue_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.BLUE_RECT.
blue_square()	This class method returns a sprixel that is the equiva-
	lent of pygamelib.assets.graphics.BLUE_SQUARE.
copy()	Returns a (deep) copy of the sprixel.
cyan_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.CYAN_RECT.
cyan_square()	This class method returns a sprixel that is the equiva-
	lent of pygamelib.assets.graphics.CYAN_SQUARE.
detach(observer)	Detach an observer from this instance.
from_ansi(string[, model])	Takes an ANSI string, parse it and return a Sprixel.
green_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.GREEN_RECT.
green_square()	This class method returns a
	sprixel that is the equivalent of
	pygamelib.assets.graphics.GREEN_SQUARE.
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by the observer.
value]) 10ad(data)	Create a new Sprixel object based on serialized data.
magenta_rect()	mi.
magenta_rect()	This class method returns a sprixel that is the equivalent of
	pygamelib.assets.graphics.MAGENTA_RECT.
magenta_square()	This class method returns a
magenea_square()	sprixel that is the equivalent of
	pygamelib.assets.graphics.MAGENTA_SQUARE.
notify([modifier])	Notify all the observers that a change occurred.
red_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.RED_RECT.
red_square()	This class method returns a sprixel that is the equiv-
_ 1	alent of pygamelib.assets.graphics.RED_SQUARE.
render_to_buffer(buffer, row, column,)	Render the sprixel from the display buffer to the
	frame buffer.
serialize()	Serialize a Sprixel into a dictionary.
store_screen_position(row, column)	Store the screen position of the object.
white_rect()	This class method returns a sprixel that is the equiv-
	alent of pygamelib.assets.graphics.WHITE_RECT.
white_square()	This class method returns a
	sprixel that is the equivalent of
	pygamelib.assets.graphics.WHITE_SQUARE.
yellow_rect()	This class method returns a sprixel that is the equiva-
	lent of pygamelib.assets.graphics.YELLOW_RECT.
	Continued on next page

576 Chapter 3. Tutorials

Table 147 – continued from previous page

yellow_square()	This	class	n	nethod	returns	a
	sprixel	that	is	the	equivalent	of
	pygamel	ib.assets.g	graphic	es.YELL	OW_SQUARE	

Attributes

ha aslan	A property to get/get the healtground solar of the
bg_color	A property to get/set the background color of the
	Sprixel.
fg_color	A property to get/set the foreground color of the
	Sprixel.
length	Return the true length of the model.
model	A property to get/set the model of the Sprixel.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (*PglBaseObject*) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

bg_color

A property to get/set the background color of the Sprixel.

Parameters value (Color) - The new color

When the bg_color is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprixel.bg_color:changed event. The new bg_color is passed as the *value* parameter.

Example:

```
# Access the sprixel's color
sprix.bg_color
# Set the sprixel's background color to some blue
sprix.bg_color = Color(0,128,255)
```

classmethod black_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLACK_RECT. The difference is that BLACK_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.black_rect()
```

classmethod black_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLACK_SQUARE. The difference is that BLACK_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.black_square()
```

classmethod blue_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLUE_RECT. The difference is that BLUE_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.blue_rect()
```

classmethod blue_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.BLUE_SQUARE. The difference is that BLUE_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.blue_square()
```

copy()

Returns a (deep) copy of the sprixel.

New in version 1.3.0.

classmethod cyan_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.CYAN_RECT. The difference is that CYAN_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.cyan_rect()
```

classmethod cyan_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.CYAN_SQUARE. The difference is that CYAN_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.cyan_square()
```

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

fg_color

A property to get/set the foreground color of the Sprixel.

```
Parameters value (Color) - The new color
```

When the fg_color is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprixel.fg_color:changed event. The new fg_color is passed as the *value* parameter.

Example:

```
# Access the sprixel's color
sprix.fg_color
# Set the sprixel's foreground color to some green
sprix.fg_color = Color(0,255,128)
```

static from_ansi(string, model='')

Takes an ANSI string, parse it and return a Sprixel.

Parameters

- **string** (*str*) The ANSI string to parse.
- model (str) The character used to represent the sprixel in the ANSI sequence. Default is ""

Example:

```
new_sprixel = Sprixel.from_ansi(
    "\x1b[48;2;139;22;19m\x1b[38;2;160;26;23m\x1b[0m"])
```

Warning: This has mainly be tested with ANSI string generated by climage. If you find any issue, please report it

classmethod green_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.GREEN_RECT. The difference is that GREEN_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.green_rect()
```

classmethod green_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.GREEN_SQUARE. The difference is that GREEN_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.green_square()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

length

Return the true length of the model.

New in version 1.3.0.

With UTF8 and emojis the length of a string as returned by python's len() function is often very wrong. For example, the len("x1b[48;2;139;22;19mx1b[38;2;160;26;23mx1b[0m") returns 39 when it should return 1.

This method returns the actual printing/display size of the sprixel's model.

Note: This is a read only value. It is automatically updated when the model is changed.

Example:

```
if sprix.length > 2:
    print(
        f"Warning: that sprixel {sprix} will break the rest of the "
        "board's alignement"
        )
```

classmethod load(data)

Create a new Sprixel object based on serialized data.

New in version 1.3.0.

Parameters data (dict) – Data loaded from JSON data (descrialized).

Return type Sprixel

Example:

```
new_sprite = Sprixel.load(json_parsed_data['default_sprixel'])
```

${\tt classmethod\ magenta_rect}\ (\,)$

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.MAGENTA_RECT. The difference is that MAGENTA_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.magenta_rect()
```

classmethod magenta_square()

This class method returns a sprixel that is the equivalent of

580 Chapter 3. Tutorials

pygamelib.assets.graphics.MAGENTA_SQUARE. The difference is that MAGENTA_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.magenta_square()
```

model

A property to get/set the model of the Sprixel.

Parameters value (str) – The new model

When the model is changed, the observers are notified of the change with the pygamelib.gfx.core.Sprixel.model:changed event. The new model is passed as the *value* parameter.

Example:

```
# Get the sprixel's model
sprix.model
# Set the sprixel's model to "@"
sprix.model = "@"
```

notify (modifier = None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

classmethod red_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_RECT. The difference is that RED_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.red_rect()
```

classmethod red_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.RED_SQUARE. The difference is that RED_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.red_square()
```

render_to_buffer (buffer, row, column, buffer_height, buffer_width)

Render the sprixel from the display buffer to the frame buffer.

New in version 1.3.0.

This method is automatically called by pygamelib.engine.Screen.render().

Parameters

- buffer (numpy.array) A screen buffer to render the item into.
- row (int) The row to render in.
- **column** (*int*) The column to render in.
- **height** (*int*) The total height of the display buffer.
- width (int) The total width of the display buffer.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (*int*) – the screen row

Return type int

serialize()

Serialize a Sprixel into a dictionary.

New in version 1.3.0.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( sprixel.serialize() )
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

classmethod white rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_RECT. The difference is that WHITE_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.white_rect()
```

582 Chapter 3. Tutorials

classmethod white square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.WHITE_SQUARE. The difference is that WHITE_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Example:

```
sprixel = Sprixel.white_square()
```

classmethod yellow_rect()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.YELLOW_RECT. The difference is that YELLOW_RECT is a string and this one is a Sprixel that can be manipulated more easily.

Note: Yellow is often rendered as brown.

Example:

```
sprixel = Sprixel.yellow_rect()
```

classmethod yellow_square()

This class method returns a sprixel that is the equivalent of pygamelib.assets.graphics.YELLOW_SQUARE. The difference is that YELLOW_SQUARE is a string and this one is a Sprixel that can be manipulated more easily.

Note: Yellow is often rendered as brown.

Example:

```
sprixel = Sprixel.yellow_square()
```

PartitionParticle

Bases: pygamelib.gfx.particles.Particle

New in version 1.3.0.

The PartitionParticle is a more precise Particle. Its main difference is that it is additive. This means that the PartitionParticle posess the ability to complement a sprixel that is already drawn. Or to add to a sprixel that is already drawn.

As a matter of facts, the primary goal of the PartitionParticle is to modify an already drawn sprixel to improve the visuals/graphical effects.

For example, if two particles occupy the same space on screen, with a regular <code>Particle</code> the last to render is the one that will be displayed. If one particle is represented by "and the other by ", only the second will be displayed.

In the case of PartitionParticles, an addition of the 2 sprixels will be displayed! So in the previous example the addition of the 2 particles would result in "because " + " = ".

3.7. qfx 583

It comes at a cost though as the PartitionParticle is slower to render than the Particle class.

The partition particle achieve that by using a partition and a blending table. The blending table is crucial for the performances to be not too catastrophic. The size of the blending table is directly linked to the performances of the PartitionParticle (the bigger the blending table the slower the rendering).

The blending table is a dictionnary of strings that covers all possible operations.

Example:

```
partition_blending_table = {
    gb.QUADRANT_UPPER_LEFT
    + gb.QUADRANT_UPPER_RIGHT: gb.UPPER_HALF_BLOCK,
    gb.QUADRANT_UPPER_LEFT + gb.QUADRANT_LOWER_LEFT: gb.LEFT_HALF_BLOCK,
    gb.QUADRANT_UPPER_LEFT
    + gb.QUADRANT_LOWER_RIGHT: gb.QUADRANT_UPPER_LEFT_AND_LOWER_RIGHT,
    # it goes on for many lines...
}
```

By default, the PartitionParticle has a blending table that is using the UTF8 Blocks.QUADRANT_* characters. If you want to use a different one, you need to define a new blending table and pass it as parameter to the constructor.

The partition itself is a 2x2 array that contains the 4 quadrants of a character displayed in the terminal.

As an example, if a full character were a block: "the partition would be: [[", "], [", "]].

You can conceive the partition as the exploded version of the character/sprixel and the blending table as the rules to blend them together.

The PartitionParticle can also be used to create reinforcement effects. For example, if the partition is composed solely of "and the partition table only define one rule: "+" = ". It is a powerful particle that can be used to create a lot of different effects.

Important: A limit of the current implementation is that the partition table must be a 2x2 array. It cannot be otherwise. Even if all the quadrants are the same.

```
__init__(row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None, partition: list = None, partition_blending_table: list = None) \rightarrow None The constructor takes the following parameters.
```

Parameters

- row (int) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.
- partition (list) The 2x2 array that defines the partition of the sprixel.
- partition_blending_table (list) The blending table that defines the rules to blend the 2 sprixels.

Example:

Methods

init(row, column, velocity, lifespan,)	The constructor takes the following parameters.
apply_force(force)	Apply a force to the particle's acceleration vector.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
finished()	Return True if the particle is done living (i.e its lifes-
	pan is lesser or equal to 0).
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load a PartitionParticle from a dictionary.
notify([modifier])	Notify all the observers that a change occurred.
render(sprixel)	This method first calls the Particle.render() method.
reset(row, column, velocity, lifespan)	Reset a particle in its initial state.
reset_lifespan(lifespan)	Reset the particle lifespan (including the initial lifes-
	pan).
serialize()	Serialize a PartitionParticle into a dictionary.
store_screen_position(row, column)	Store the screen position of the object.
terminate()	Terminate a particle, i.e sets its lifespan to -1.
update()	This method first calls the Particle.update() method,
	then calculates the quadrant position, i.e: the actual
	position of the particle within a console character.

Attributes

column	Access and set the column property.
row	Access and set the row property.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
X	Access and set the x property.
Y	Access and set the y property.

```
apply_force(force: pygamelib.base.Vector2D) \rightarrow None
```

Apply a force to the particle's acceleration vector.

You are more likely to apply forces to all particles of an emitter through the apply_force() method of the emitter class.

Parameters force (Vector2D) – The force to apply.

Example:

```
gravity = Vector2D(-0.2, 0.0)
my_particle.apply_force(gravity)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property. Equivalent to the x property.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

$finished() \rightarrow bool$

Return True if the particle is done living (i.e its lifespan is lesser or equal to 0). It returns False otherwise.

Return type bool

Example:

```
if not my_particle.finished():
    my_particle.update()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PglBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

classmethod load(data)

Load a PartitionParticle from a dictionary.

Parameters data(dict) – The dictionary to load from

Returns The loaded PartitionParticle

Return type PartitionParticle

Example:

```
particle = PartitionParticle.load( json.load( open("particle.json") ) )
```

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

render (sprixel: pygamelib.gfx.core.Sprixel = None)

This method first calls the Particle.render() method. Then it updates the rendered particle's model based on the blending table.

Parameters sprixel (Sprixel) – A sprixel already rendered in the screen buffer.

Example:

```
p = my_particle
buffer[p.row][p.column] = p.render(buffer[p.row][p.column])
```

reset (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None) Reset a particle in its initial state. This is particularly useful for the reuse of particles.

This method takes almost the same parameters than the constructor.

Parameters

3.7. qfx 587

- **row** (*int*) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.

```
single_particle.reset(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
   lifespan=10,
)
```

reset_lifespan (lifespan: int = 20) \rightarrow None

Reset the particle lifespan (including the initial lifespan).

Parameters lifespan (int) – The particle lifespan in number of movements/turns.

Example:

```
my_particle.reset_lifespan(10)
```

row

Access and set the row property. Equivalent to the y property.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serialize a PartitionParticle into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( particle.serialize() )
```

$store_screen_position$ (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

```
an_object.store_screen_coordinate(3,8)
```

$terminate() \rightarrow None$

Terminate a particle, i.e sets its lifespan to -1.

In that case the ParticleEmitter and ParticlePool will recycle it. That is *IF* you are managing the particle through an emitter and/or a pool of course.

Example:

```
p = my_particle
if p.row >= screen,height or p.column >= screen.width:
    p.terminate()
```

update()

This method first calls the Particle.update() method, then calculates the quadrant position, i.e: the actual position of the particle within a console character. It then updates the particle's model based on this internal position.

Example:

```
my_particle.update()
```

x

Access and set the x property. Equivalent to the column property.

У

Access and set the y property. Equivalent to the row property.

RandomColorParticle

```
class pygamelib.gfx.particles.RandomColorParticle (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None, sprixel: pygamelib.gfx.particles.ParticleSprixel = None, color: pygamelib.gfx.core.Color = None)
```

Bases: pygamelib.gfx.particles.Particle

This class is a <code>Particle</code> that has a random foreground color.

By default, if both the sprixel and color parameters are not specified, the model of the *Sprixel* is going to be '•' and the color will be randomly chosen.

You can also specify a color and a model.

```
__init__(row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None, sprixel: pygamelib.gfx.particles.ParticleSprixel = None, color: pygamelib.gfx.core.Color = None) \rightarrow None The constructor takes the following parameters.
```

Parameters

- row (int) The initial row position of the particle on the screen.
- column (int) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.

3.7. qfx 589

- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.
- **sprixel** (*Sprixel*) The sprixel that represent the particle when drawn on screen.
- **color** (Color) The color of the particle (if you want a specific color instead of a random one).

```
single_particle = RandomColorParticle(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
   lifespan=10,
)
```

Methods

init(row, column, velocity, lifespan,)	The constructor takes the following parameters.
apply_force(force)	Apply a force to the particle's acceleration vector.
attach(observer)	Attach an observer to this instance.
detach(observer)	Detach an observer from this instance.
finished()	Return True if the particle is done living (i.e its lifes-
	pan is lesser or equal to 0).
handle_notification(subject[, attribute,	A virtual method that needs to be implemented by
value])	the observer.
load(data)	Load a PartitionParticle from a dictionary.
notify([modifier])	Notify all the observers that a change occurred.
render(sprixel)	Render the particle as a Sprixel.
reset(row, column, velocity, lifespan)	Reset a particle in its initial state.
reset_lifespan(lifespan)	Reset the particle lifespan (including the initial lifes-
	pan).
serialize()	Serialize a RandomColorParticle into a dictionary.
store_screen_position(row, column)	Store the screen position of the object.
terminate()	Terminate a particle, i.e sets its lifespan to -1.
update()	The update method perform the calculations required
	to process the new particle position.

Attributes

column	Access and set the column property.
row	Access and set the row property.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
X	Access and set the x property.
Y	Access and set the y property.

apply_force (force: pygamelib.base.Vector2D) \rightarrow None Apply a force to the particle's acceleration vector.

You are more likely to apply forces to all particles of an emitter through the apply_force() method of the emitter class.

Parameters force (Vector2D) – The force to apply.

Example:

```
gravity = Vector2D(-0.2, 0.0)
my_particle.apply_force(gravity)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property. Equivalent to the x property.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (PglBaseObject) – An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard.
myboard.detach(screen)
```

$finished() \rightarrow bool$

Return True if the particle is done living (i.e its lifespan is lesser or equal to 0). It returns False otherwise.

Return type bool

Example:

```
if not my_particle.finished():
    my_particle.update()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PqlBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

classmethod load (data)

Load a PartitionParticle from a dictionary.

Parameters data (dict) – The dictionary to load from

Returns The loaded PartitionParticle

Return type PartitionParticle

Example:

```
particle = RandomColorParticle.load( json.load( open("particle.json") ) )
```

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- **attribute** (str) An optional parameter that identify the attribute that has changed.
- **value** (*Any*) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

render (sprixel: pygamelib.gfx.core.Sprixel = None)

Render the particle as a *Sprixel*. This method is called by the *ParticleEmitter* render_to_buffer method.

It takes a *Sprixel* as a parameter. This Sprixel is given by the ParticleEmitter.render_to_buffer() method and if it is not None, the particle will render itself into that *Sprixel* and return it.

Important: This method must be called after everything else as rendered or else there will be *Sprixel* that will be overwritten during their rendering cycle. Other elements could also have their *Sprixel* corrupted and replaced by the particle's one.

Parameters sprixel (Sprixel) – A sprixel already rendered in the screen buffer.

Example:

```
p = my_particle
buffer[p.row][p.column] = p.render(buffer[p.row][p.column])
```

reset (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None) Reset a particle in its initial state. This is particularly useful for the reuse of particles.

This method takes almost the same parameters than the constructor.

Parameters

- row (int) The initial row position of the particle on the screen.
- column (int) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.

Example:

```
single_particle.reset(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
   lifespan=10,
)
```

reset_lifespan (lifespan: int = 20) \rightarrow None

Reset the particle lifespan (including the initial lifespan).

Parameters lifespan (*int*) – The particle lifespan in number of movements/turns.

Example:

```
my_particle.reset_lifespan(10)
```

row

Access and set the row property. Equivalent to the y property.

screen_column

A property to get/set the screen column.

Parameters value (*int*) – the screen column

Return type int

screen row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

$\verb"serialize"()$

Serialize a RandomColorParticle into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( particle.serialize() )
```

store screen position (row: int, column: int) \rightarrow bool

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

3.7. qfx 593

- row (int) The row (or y) coordinate.
- **column** (*int*) The column (or x) coordinate.

```
an_object.store_screen_coordinate(3,8)
```

$\texttt{terminate}\,()\,\to None$

Terminate a particle, i.e sets its lifespan to -1.

In that case the ParticleEmitter and ParticlePool will recycle it. That is *IF* you are managing the particle through an emitter and/or a pool of course.

Example:

```
p = my_particle
if p.row >= screen,height or p.column >= screen.width:
    p.terminate()
```

$update() \rightarrow None$

The update method perform the calculations required to process the new particle position. It mainly adds the acceleration to the velocity vector and update the position accordingly.

After calling update() the acceleration is "consumed" in the velocity and therefor reset.

The update() method takes no parameters and returns nothing.

Example:

```
my_particle.update()
```

x

Access and set the x property. Equivalent to the column property.

У

Access and set the y property. Equivalent to the row property.

RandomColorPartitionParticle

```
class pygamelib.gfx.particles.RandomColorPartitionParticle (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D
= None, lifespan: int
= None, partition: list = None, partition: list = None, color: pygamelib.gfx.core.Color
= None)
```

Bases: pygamelib.gfx.particles.PartitionParticle

This class is basically the same as RandomColorParticle but its base class is PartitionParticle instead of Particle. Everything else is the same.

```
__init__ (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifes-pan: int = None, partition: list = None, partition_blending_table: list = None, color: pygamelib.gfx.core.Color = None) → None
The constructor takes the following parameters.
```

594 Chapter 3. Tutorials

Parameters

- row (int) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.
- partition (list) The partition of the particle.
- partition_blending_table (list) The blending table of the particle.
- **color** (Color) The color of the particle (if you want a specific color instead of a random one).

Example:

```
single_particle = RandomColorPartitionParticle(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
   lifespan=10,
)
```

Methods

The constructor takes the following parameters.		
Apply a force to the particle's acceleration vector.		
Attach an observer to this instance.		
Detach an observer from this instance.		
Return True if the particle is done living (i.e its lifes-		
pan is lesser or equal to 0).		
A virtual method that needs to be implemented by		
the observer.		
Load a RandomColorPartitionParticle from a dictio-		
nary.		
Notify all the observers that a change occurred.		
This method first calls the Particle.render() method.		
Reset a particle in its initial state.		
Reset the particle lifespan (including the initial lifes-		
pan).		
Serialize a RandomColorPartitionParticle into a dic-		
tionary.		
Store the screen position of the object.		
Terminate a particle, i.e sets its lifespan to -1.		
This method first calls the Particle.update() method,		
then calculates the quadrant position, i.e: the actual		
position of the particle within a console character.		

Attributes

column	Access and set the column property.
row	Access and set the row property.
screen_column	A property to get/set the screen column.
screen_row	A property to get/set the screen row.
X	Access and set the x property.
Y	Access and set the y property.

$apply_force (force: pygamelib.base.Vector2D) \rightarrow None$

Apply a force to the particle's acceleration vector.

You are more likely to apply forces to all particles of an emitter through the apply_force() method of the emitter class.

Parameters force (Vector2D) – The force to apply.

Example:

```
gravity = Vector2D(-0.2, 0.0)
my_particle.apply_force(gravity)
```

attach (observer)

Attach an observer to this instance. It means that until it is detached, it will be notified every time that a notification is issued (usually on changes).

An object cannot add itself to the list of observers (to avoid infinite recursions).

Parameters observer (PglBaseObject) – An observer to attach to this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
myboard = Board()
screen = Game.instance().screen
# screen will be notified of all changes in myboard
myboard.attach(screen)
```

column

Access and set the column property. Equivalent to the x property.

detach (observer)

Detach an observer from this instance. If observer is not in the list this returns False.

Parameters observer (Pq1BaseObject) - An observer to detach from this object.

Returns True or False depending on the success of the operation.

Return type bool

Example:

```
# screen will no longer be notified of the changes in myboard. myboard.detach(screen)
```

$finished() \rightarrow bool$

Return True if the particle is done living (i.e its lifespan is lesser or equal to 0). It returns False otherwise.

Return type bool

Example:

```
if not my_particle.finished():
    my_particle.update()
```

handle_notification (subject, attribute=None, value=None)

A virtual method that needs to be implemented by the observer. By default it does nothing but each observer needs to implement it if something needs to be done when notified.

This method always receive the notifying object as first parameter. The 2 other parameters are optional and can be None.

You can use the attribute and value as you see fit. You are free to consider attribute as an event and value as the event's value.

Parameters

- **subject** (*PqlBaseObject*) The object that has changed.
- attribute (str) The attribute that has changed, it is usually a "FQDN style" string. This can be None.
- value (Any) The new value of the attribute. This can be None.

classmethod load (data)

Load a RandomColorPartitionParticle from a dictionary.

Parameters data (dict) – The dictionary to load from

Returns The loaded RandomColorPartitionParticle

Return type RandomColorPartitionParticle

Example:

notify (modifier=None, attribute: str = None, value: Any = None) \rightarrow None Notify all the observers that a change occurred.

Parameters

- modifier (PglBaseObject) An optional parameter that identify the modifier object to exclude it from the notified objects.
- attribute (str) An optional parameter that identify the attribute that has changed.
- **value** (Any) An optional parameter that identify the new value of the attribute.

Example:

```
# This example is silly, you would usually notify other objects from inside
# an object that changes a value that's important for the observers.
color = Color(255,200,125)
color.attach(some_text_object)
color.notify()
```

render (sprixel: pygamelib.gfx.core.Sprixel = None)

This method first calls the Particle.render() method. Then it updates the rendered particle's model based on the blending table.

Parameters sprixel (*Sprixel*) – A sprixel already rendered in the screen buffer.

Example:

3.7. qfx 597

```
p = my_particle
buffer[p.row][p.column] = p.render(buffer[p.row][p.column])
```

reset (row: int = 0, column: int = 0, velocity: pygamelib.base.Vector2D = None, lifespan: int = None) Reset a particle in its initial state. This is particularly useful for the reuse of particles.

This method takes almost the same parameters than the constructor.

Parameters

- row (int) The initial row position of the particle on the screen.
- **column** (*int*) The initial column position of the particle on the screen.
- **velocity** (*Vector2D*) The initial velocity of the particle.
- **lifespan** (*int*) The particle lifespan in number of movements/turns. A particle with a lifespan of 3 will move for 3 turns before being finished.

Example:

```
single_particle.reset(
   row=5,
   column=5,
   velocity=base.Vector2D(-0.5, 0.0),
   lifespan=10,
)
```

$reset_lifespan (lifespan: int = 20) \rightarrow None$

Reset the particle lifespan (including the initial lifespan).

Parameters lifespan (int) – The particle lifespan in number of movements/turns.

Example:

```
my_particle.reset_lifespan(10)
```

row

Access and set the row property. Equivalent to the y property.

screen_column

A property to get/set the screen column.

Parameters value (int) – the screen column

Return type int

screen_row

A property to get/set the screen row.

Parameters value (int) – the screen row

Return type int

serialize()

Serialize a RandomColorPartitionParticle into a dictionary.

Returns The class as a dictionary

Return type dict

Example:

```
json.dump( particle.serialize() )
```

$store_screen_position$ (row: int, column: int) $\rightarrow bool$

Store the screen position of the object.

This method is automatically called by Screen.place().

Parameters

- row (int) The row (or y) coordinate.
- column (int) The column (or x) coordinate.

Example:

```
an_object.store_screen_coordinate(3,8)
```

$\texttt{terminate}\,(\,)\,\to None$

Terminate a particle, i.e sets its lifespan to -1.

In that case the ParticleEmitter and ParticlePool will recycle it. That is *IF* you are managing the particle through an emitter and/or a pool of course.

Example:

```
p = my_particle
if p.row >= screen,height or p.column >= screen.width:
    p.terminate()
```

update()

This method first calls the Particle.update() method, then calculates the quadrant position, i.e: the actual position of the particle within a console character. It then updates the particle's model based on this internal position.

Example:

```
my_particle.update()
```

x

Access and set the x property. Equivalent to the column property.

У

Access and set the y property. Equivalent to the row property.

3.8 Credits

3.8.1 Development Lead

• Arnaud Dupuis (@arnauddupuis)

3.8.2 Contributors

- Kalil de Lima (@kaozdl)
- Muhammad Syuqri (@Dansyuqri)
- Ryan Brown (@grimmjow8)

3.8. Credits 599

- Chase Miller (@ Arekenaten)
- Gunjan Rawal (@gunjanraval)
- Anshul Choudhary (@achoudh5)
- Raymond Beaudoin (@synackray)
- Felipe Rodrigues (@fbidu)
- Bastien Wirtz (@bwirtz)
- Franz Osorio (@f-osorio)
- Guillermo Eijo (@guilleijo)
- Diego Cáceres (@diego-caceres)
- Spassarop (@spassarop)
- Javier Hernán Caballero García (@caballerojavier13)
- Olle Lögdahl (@ollelogdahl)
- MaryEtta Morris (@morrme)
- Peter Szabo (@szabopeter)
- Frans Ramirez (@Frans06)
- Krunal Rank (@KRHero03)
- Juan Picca (@jumapico)
- Harshini (@harshiniwho)
- Tammysalmon (@tammysalmon)
- JayC (@jayc13)
- Rikil Gajarla (@RikilG)
- Melsaa (@melsaa)

3.9 Release notes

3.9.1 1.3.0 (2022-10-07)

This release is massive. Please read the documentation for specific changes to classes. It is available at https://pygamelib.readthedocs.io/en/latest/index.html.

Important one: the whole pygamelib has been migrated to its own Github organization: https://github.com/pygamelib please update your links! The library's repository is now available at https://github.com/pygamelib/pygamelib.

Main updates

- New feature: A lot of new tools have been developed for the library and are all available on the organization's Github: https://github.com/pygamelib.
- New feature: The pygamelib.engine.Screen class now has a new Improved Screen Management double buffered system. This set of methods allow for a simplified management of the console screen. It is also faster than the *Legacy Direct Display* system. Please read the documentation (https://pygamelib.readthedocs.io/en/latest/pygamelib.engine.Screen.html) and the wiki on the Github repository for more about the differences.

You will probably want to switch to the new stack as soon as possible. Both systems are clearly identified in the documentation by visible tags. Most of the new features of this release are **NOT** compatible with the *Legacy Direct Display* system. It still received updates and new features but will probably be deprecated in future updates.

- New feature: Introducing the pygamelib.gfx.ui module! The beginning of a module for all your game/application user interface needs. The module is in alpha for the moment, feel free to voice your feedback. This module is only compatible with the Improved Screen Management.
- New feature: A new tool has been added to the library: pgl-sprite-editor. An editor to create or edit sprites and sprite based animations.
- New feature: pygamelib.engine.Game can now be created as a Singleton through the instance() method.
- New feature: Add a particle system to the library! It includes a number of new classes that are located in the pygamelib.gfx.particles submodule. This module is only compatible with the Improved Screen Management.
- New feature: introducing pygamelib.gfx.core.Font, a Sprite based font system. This release come with an "8bits" font and a couple of font imported from FIGlet!
- New feature: Add a Color class (pygamelib.gfx.core.Color) to entirely abstract the color system.
- New feature: All objects can now be properly serialized and loaded through a streamlined process. Look for the *serialize()* and *load()* methods.
- New feature: New base object pygamelib.base.PglBaseObject, all objects that inherits from python's object are now inheriting from this new one. It implements a couple of base features but the most important is the modified *Observer* design pattern that is the base of a refactoring to event base communication within the library.
- New feature: Added a new board item: pygamelib.board_item.Camera. It is a specific item that is not shown on the board. It can be used for cinematic for example. Please read the documentation for more information.
- New feature/improvement: The *Board* object has been reworked to allow for a third dimension. It now has a new property called *layer*. Layers are automatically added and removed to fit the need of overlapping items. *Board.place_item()* also accept a new layer parameter to set the layer (if you want to put stuff over the player for example). An example is visible here: https://www.youtube.com/watch?v=9cOt63ZAJOk.
- *Improvement:* Most resources intensive array/list have been replaced by numpy arrays. This brings better performances for pygamelib.engine.Board and for pygamelib.engine.Screen.
- Improvement: Add a new algorithm to the PathFinder actuator: A*.
- Improvement: pygamelib.gfx.core.Sprite can now be tinted or modulated with a color. Both operation do the same thing: change the color of the sprite by applying a color at a given ratio. However, tint() returns a new sprite and does not modify the original sprite while modulate() returns nothing and modify the sprite directly.

Breaking changes

- pygamelib.board_items.BoardItem constructor parameter changed: type is now item_type.
- pygamelib.board_items.BoardItem: there was a conflict with inventory_space. It was defined both as a property and a method. The method has been removed and *BoardItem.inventory_space* is now a proper python property. Concretely: you might have to remove parenthesis when using any_item.inventory_space().

3.9. Release notes 601

- The new pygamelib.gfx.core.Color replaces Terminal.on_color_rgb() and Terminal.color_rgb(). It is much easier to use (just use the Color object and the pygamelib will manage foreground and background differences) but it requires to change the initialization of every Sprixel and Text of your game (sorry...).
- When using the new Improved Screen Management stack and partial display at the same time, you now have to set Board.partial_display_focus. It is not breaking anything in existing code but it will not behave as you want is you just Screen.place() your board (that uses partial display) without setting the partial display focus to the player first.

Other changes

- *Improvement*: pgl-editor now uses Sprixels instead of regular characters allowing for more possible customization and features in the Board and Screen.
- Improvement: in pgl-editor it is now possible to generate a random color in the color editor.
- *Improvement*: All actuators now return pygamelib.constants.NO_DIR if there is no direction available to next_move(). This makes the actuators behavior more consistent particularly when they are overloaded.
- *Improvement*: The RandomActuator behavior has been reworked. It now choose a direction and follow it for a certain distance before choosing a new direction. It also detect when it is stuck an, in that case, pick a new direction.
- *Improvement*: Add display_sprite() and display_sprite_at() method to Screen. These methods can display a pygamelib.gfx.core.Sprite on screen.
- *Improvement*: Inventory has been improved to be more versatile and less limited. It now behaves like an enhanced list of objects. A rudimentary constraints system was added (for example to limit the number of certain types of items). The new inventory is also fully plugged into the observer/notifications system.
- Improvement: All BoardItem now have configurable properties for restorable, overlappable, pickable and can move.
- Improvement: pygamelib.board_items.BoardComplexItem.sprite is now a @property instead of a class variable. That property automatically call update_sprite().
- Improvement: When Game.mode is set to pygamelib.constants.MODE_RT, all pygamelib. board_items.Movable now accumulate movement vectors (when using vectors). This means that non unit movement patterns are now possible.
- Improvement: The new pygamelib.base.Console implements a Singleton design pattern. You can now get a unique reference to the blessed.Terminal (the object wrapped in Console) object by calling Console.instance().
- Fixed a bug in pygamelib.engine.Screen.display_at(): it was not possible to display anything after (below a Board). It is now possible.
- *Improvement*: pygamelib.base.Text has improved a lot. It can now use the Font system, has new attributes and is now a *PglBaseObject*. Please read the documentation for more.
- *Improvement*: Sprixels and Sprites now have their own deepcopy operator: Sprixel.copy() and Sprite.copy().
- *Improvement*: It is now possible to set the transparency of all sprixels of a sprite by using Sprite. set_transparency().
- Fixed a bug with *restorable* items: now all board items can be set to be restorable.
- Fixed a bug in pgl-editor when editing large boards that require partial display. The viewport was not correctly restored.

602 Chapter 3. Tutorials

- Fixed issues with the library's inheritance graph.
- Fixed a bug in pygamelib.engine.Game where the partial display settings (when set at in the Game instance), were not correctly passed down to the Board.
- Fixed the sphinx dependencies (for building the doc).
- Fixed the mess in the sphinx files to generate the documentation.
- Fixed an issue with linting dependencies.
- Removed legacy files from older version of the library.

I would like to thank all the contributors (https://pygamelib.readthedocs.io/en/latest/authors.html) for their work on this massive update.

The new pygamelib logo was done by an awesome artist: Jack Tseng (https://hellojacktseng.carrd.co/ https://twitter.com/HelloJackTseng) please have a look at their amazing work!!

3.9.2 1.2.3 (2020-09-01)

Emergency release: fix a regression introduced by v1.2.2.

3.9.3 1.2.2 (2020-09-01)

- Fix issue with imports for Python 3.6
- Fix an issue with the way pygamelib.engine. Screen test the terminal on Windows.

3.9.4 1.2.0 (2020-08-29)

- Renamed the entire library from hac-game-lib to pygamelib.
- *Breaking change:* The library has been heavily refactored and this creates some issues. Please have a look at the migration notes
- New feature: Items that can be represented on more than one cell. We call them complex items. There's a lot of new complex items: ComplexPlayer and ComplexNPC of course, but also ComplexWall, ComplexDoor, ComplexTreasure and the general purpose Tile object.
- New feature: Going, with complex item we now have a proper sprite system with the gfx.core.Sprite class.
- New feature: In addition to the regular model we now have a new concept: the Sprixel. A Sprite is made of many Sprixels.
- New feature: New JSON based file format to save, load and distribute sprites and/or sprixels.
- New feature: All these sprites can be grouped into a SpriteCollection that in turn can be saved in our new sprite file format.
- New feature: New Math library. This one starts small but will grow. It makes calculating the distance and intersections easier.
- New feature: New Vector2D class to represent forces and movement as a vector. It is now possible to give a vector to the move() method.
- New feature: Gave some love to text. There are now 2 objects dedicated to text: base. Text to manipulate text and board_items. TextItem to easily place text on a board.
- New feature: A Screen object has been added to make the screen manipulation simpler.

3.9. Release notes 603

- New feature: The Game object now has a run() method that act as the main game loop. It calls a user defined update function and takes care of a lot of things. It runs until the Game.state is set to STOPPED.
- **New feature:** The Game object can now turn by turn or real time. All movables can be configured to have time based or turn based movement speed.
- Improvement: The Animation class now support both regular strings (models), Sprixel and Sprite.
- *Improvement*: All complex items obviously support (actually requires) sprites but all regular board items now supports sprixels.
- Improvement: Test coverage dramatically improved. It has jumped from 25% to 98%.
- *Improvement*: Lots of objects now have attributes to easily access and/or set properties like position (mostly read only), width, height, etc.
- *Improvement*: Converted the editor to pygamelib and renamed it pgl-editor.py. Also added a multi page selector and integrated the new graphic assets.
- Improvement: All movables can now have different vertical and horizontal "steps" parameters.
- Cleaned up the repository (it was becoming seriously messy).
- Change the prefix of all exceptions from HAc to Pgl.
- Added a NO_PLAYER constant to tell the game object that he should not expect a player object.
- Improve the generated documentation.
- Various improvements in exceptions raising across the library. Please see the documentation (that was also updated).
- Various bug fixing in the Suparex example.

I also need to give some kudos to the kids of the Hyrule Astronomy Club for thorough testing of Suparex. They found well hidden bug and exploitable bugs. Special thanks to Arthur who found many glitches. Congratulations to Arthur and Hadrien that successfully exploited them to achieve extremely high scores (up to 12000!!!).

3.9.5 1.1.1 (2020-07-18)

- Fix a bug in hgl-editor: when using previously recorded parameters to create a board the editor was crashing.
- *Improvement*: Automatically enable partial display and map bigger than 40x40.
- Fix a bug a coordinates in Board.item()

3.9.6 1.1.0 (2020-06-12)

- Fix many issues with strings all across the library.
- Fix many issues with variables interpolation in exceptions.
- Fix a bug in Game.load_board() that was causing corruptions.
- Fix multiple typos in the documentation.
- Fix an issue with the user directory in hgl-editor
- Fix many issues with the PatrolActuator.
- New feature: partial display (dynamically display only a part of a board)
- New feature: new mono directional actuator.

- New feature: projectiles (can be sent and completely managed by the game object)
- New feature: new assets module to hold many non core submodules.
- New feature: Assets.Graphics that add thousands of glyphs (including emojis) to the current capacities of the library.
- **New feature:** Add support for PatrolActuator in hgl-editor.
- New feature: Add support for PathFinder actuator in hgl-editor.
- New feature: Add an object parent system.
- New feature: Add a configuration system to hgl-editor.
- *Improvement*: Add full configuration features to the Game object.
- *Improvement*: Add a new example in the form of a full procedural generation platform game (see examples/suparex).
- *Improvement*: Improved performances particularly around the features that relies on Board.place_item(). Up to 70 times faster.
- *Improvement*: It is now possible to specify the first frame index in Animation.
- Improvement: Formatted all the code with black.
- Improvement: PathFinder.add_waypoint() now sets the destination if it wasn't set before.

3.9.7 1.0.1 (2020-05-17)

- Fix a huge default save directory issue (see complete announcement) in hgl-editor.
- Fix lots of strings in hgl-editor.
- Fix a type issue in the Inventory class for the not_enough_space exception.
- Improve Board.display() performances by 15% (average).

3.9.8 1.0.0 (2020-03-20)

- Add AdvancedActuators.PathFinder @arnauddupuis
- Add test cases for BoardItem @grimmjow8 @Arekenaten
- Add test cases for Board @grimmjow8 @Arekenaten
- Add support to load files from the directories in directories.json @kaozdl
- Add a new SimpleActuators.PatrolActuator @kaozdl
- Add Animation capabilities @arnauddupuis
- Improve navigation in hgl-editor by using arrow keys @bwirtz
- Improve selection of maps in hgl-editor @gunjanraval @kaozdl
- Improve documentation for SimpleActuators.PathActuator @achoudh5
- Improve documentation for launching the test suite @bwirtz
- Migration from pip install to pipenv @kaozdl
- Fix board saving bug in hgl-editor @gunjanraval
- Fix back menu issues in hgl-editor @synackray

3.9. Release notes 605

- Fix README and setup.py @fbidu
- Make the module compatible with Flake8: @bwirtz @arnauddupuis @kaozdl @f-osorio @guilleijo @diego-caceres @spassarop
- CircleCI integration @caballerojavier13 @bwirtz

3.9.9 2019.5

• Please see the official website.

3.9.10 pre-2019.5

• Please see the Github for history.

$\mathsf{CHAPTER}\, 4$

Indices and tables

- genindex
- modindex
- search

р

pygamelib.constants,432

610 Python Module Index

Symbols	init()
init() (pygamelib.actuators.Actuator method), 7	method), 258
init() (pygamelib.actuators.Behavioral	init() (pygamelib.board_items.Character
method), 10	method), 266
init() (pygamelib.actuators.PathActuator	init() (pygamelib.board_items.ComplexDoor
method), 14	method), 274
init() (pygamelib.actuators.PathFinder	init() (pygamelib.board_items.ComplexNPC
method), 21	method), 283
init() (pygamelib.actuators.PatrolActuator	init() (pygamelib.board_items.ComplexPlayer
method), 17	method), 293
init() (pygamelib.actuators.RandomActuator	init() (pygamelib.board_items.ComplexTreasure
method), 28	method), 301
init() (pygamelib.actuators.UnidirectionalActuat	or_init() (pygamelib.board_items.ComplexWall
method), 31	method), 310
init() (pygamelib.assets.graphics.Blocks	init() (pygamelib.board_items.Door method), 320
method), 37	
init() (pygamelib.assets.graphics.BoxDrawings	init() (pygamelib.board_items.GenericActionableStructure method), 328
method), 42	init () (magnelih hogrd itams Generic Structure
	nesinit() (pygamelib.board_items.GenericStructure method), 344
method), 51	init() (pygamelib.board_items.GenericStructureComplexCompone
init() (pygamelib.assets.graphics.Models	method), 336
method), 113	init() (pygamelib.board_items.Immovable
init() (pygamelib.base.Math method), 190	method), 352
init() (pygamelib.base.PglBaseObject method), 192	init() (pygamelib.board_items.Movable
init() (pygamelib.base.Text method), 195	method), 360
init() (pygamelib.base.Vector2D method), 203	init() (pygamelib.board_items.NPC method),
init() (pygametib.board_items.Actionable	368
method), 207	init() (pygamelib.board_items.Player method),
init() (pygamelib.board_items.ActionableTile	377
method), 215	init()
init() (pygamelib.board_items.BoardComplexItem	m method), 386
method), 224	init()
init() (pygamelib.board_items.BoardItem	method), 396
method), 242	init() (pygamelib.board_items.Tile method),
init() (pygamelib.board_items.BoardItemComple	exComponen 106
method), 233	tnit() (pygameub.boara_uems.treasure
init() (pygamelib.board_items.BoardItemVoid	method), 415
method), 250	init() (pygamelib.board_items.Wall method),
	423

	A D 525
init() (pygamelib.engine.Board method), 433	method), 535
init() (pygamelib.engine.Game method), 443	init() (pygamelib.gfx.ui.MultiLineInputDialog
init() (pygamelib.engine.Inventory method),	method), 538
459	init() (pygamelib.gfx.ui.ProgressBar method),
init() (pygamelib.engine.Screen method), 469	540
init() (pygamelib.gfx.core.Animation method), 480	init() (pygamelib.gfx.ui.ProgressDialog method), 542
init() (pygamelib.gfx.core.Color method), 506	init() (pygamelib.gfx.ui.UiConfig method), 545
init() (pygamelib.gfx.core.Font method), 483	A
init() (pygamelib.gfx.core.Sprite method), 491	A
init() (pygamelib.gfx.core.SpriteCollection	A_BUTTON_BLOOD_TYPE
method), 486	(pygamelib.assets.graphics.Models attribute),
init() (pygamelib.gfx.core.Sprixel method), 498	141
init() (pygamelib.gfx.particles.CircleEmitter	AB_BUTTON_BLOOD_TYPE
method), 546	(pygamelib.assets.graphics.Models attribute),
init() (pygamelib.gfx.particles.ColorParticle	140
method), 551	ABACUS (pygamelib.assets.graphics.Models attribute),
init() (pygamelib.gfx.particles.ColorPartitionPartiti	ticle 140
method), 556	AC_CURRENT (pygamelib.assets.graphics.MiscTechnicals
init() (pygamelib.gfx.particles.EmitterProperties	attribute), 69
method), 561	ACCORDION (pygamelib.assets.graphics.Models at-
init() (pygamelib.gfx.particles.Particle	tribute), 140
method), 570	action (pygamelib.gfx.ui.MenuAction attribute), 530
init() (pygamelib.gfx.particles.ParticleEmitter	Actionable (class in pygamelib.board_items), 206
method), 564	ActionableTile (class in pygamelib.board_items),
init() (pygamelib.gfx.particles.ParticlePool	214
method), 568	activate() (pygamelib.board_items.Actionable
init() (pygamelib.gfx.particles.ParticleSprixel	method), 208
method), 575	<pre>activate() (pygamelib.board_items.ActionableTile</pre>
init() (pygamelib.gfx.particles.PartitionParticle	method), 216
method), 584	activate() (pygamelib.board_items.GenericActionableStructure
init() (pygamelib.gfx.particles.RandomColorPart	,,,
method), 589	activate() (pygamelib.gfx.ui.Menu method), 527
init() (pygamelib.gfx.particles.RandomColorPart	4.0
method), 594	530
init() (pygamelib.gfx.ui.Box method), 512	active (pygamelib.gfx.particles.CircleEmitter at-
init() (pygamelib.gfx.ui.ColorPicker method),	tribute), 547
515	active (pygamelib.gfx.particles.ParticleEmitter at-
init() (pygamelib.gfx.ui.ColorPickerDialog method), 513	tribute), 564
	actuate_npcs() (pygamelib.engine.Game method),
init() (pygamelib.gfx.ui.Dialog method), 517 init() (pygamelib.gfx.ui.FileDialog method),	445
	actuate_projectiles() (pygamelib.engine.Game
init() (pygamelib.gfx.ui.GridSelector method),	method), 446
522	Actuator (class in pygamelib.actuators), 7
init() (pygamelib.gfx.ui.GridSelectorDialog	add() (pygamelib.gfx.core.SpriteCollection method),
method), 520	487
init() (pygamelib.gfx.ui.LineInputDialog	add_board() (pygamelib.engine.Game method), 446
method), 524	add_constraint() (pygamelib.engine.Inventory
init() (pygamelib.gfx.ui.Menu method), 526	<pre>method), 460 add_directional_animation()</pre>
init() (pygametib.gfx.ui.MenuAction method),	(pygamelib.board_items.Projectile method),
529	(pygameno.boara_nems.Frojectite meinoa), 388
init() (pygamelib.gfx.ui.MenuBar method), 532	add_directional_model()
init() (pygamelib.gfx.ui.MessageDialog	(pygamelib.hoard items.Projectile method).

388 add_entry() (pygamelib.gfx.ui.Menu method), 527	ANATOMICAL_HEART (pygamelib.assets.graphics.Models attribute), 140
add_entry() (pygamelib.gfx.ui.MenuBar method), 533	ANCHOR (pygamelib.assets.graphics.Models attribute), 140
add_frame() (pygamelib.gfx.core.Animation method), 480	ANGER_SYMBOL (pygamelib.assets.graphics.Models attribute), 141
add_item() (pygamelib.engine.Inventory method), 461	ANGRY_FACE (pygamelib.assets.graphics.Models attribute), 141
add_line() (pygamelib.gfx.ui.MessageDialog	ANGRY_FACE_WITH_HORNS (pygamelib.assets.graphics.Models attribute),
method), 536 add_menu_entry() (pygamelib.engine.Game	141
method), 447	ANGUISHED_FACE (pygamelib.assets.graphics.Models
add_npc() (pygamelib.engine.Game method), 447	attribute), 141
add_projectile() (pygamelib.engine.Game method), 448	animate_items() (pygamelib.engine.Game method), 448
	Animation (class in pygamelib.gfx.core), 479
method), 22	animation (pygamelib.board_items.Actionable at-
ADHESIVE_BANDAGE (pygamelib.assets.graphics.Models	
attribute), 140 ADMISSION_TICKETS	animation (pygamelib.board_items.ActionableTile attribute), 216
140	animation(pygamelib.board_items.BoardComplexItem attribute), 225
AERIAL_TRAMWAY (pygamelib.assets.graphics.Models attribute), 140	animation (pygamelib.board_items.BoardItem at- tribute), 243
AIRPLANE (pygamelib.assets.graphics.Models attribute), 140	animation (pygamelib.board_items.BoardItemComplexComponent attribute), 234
AIRPLANE_ARRIVAL (pygamelib.assets.graphics.Models attribute), 140	sanimation (pygamelib.board_items.BoardItemVoid at- tribute), 251
AIRPLANE_DEPARTURE	animation (pygamelib.board_items.Camera attribute),
(pygamelib.assets.graphics.Models attribute),	259
140	animation (pygamelib.board_items.Character at-
ALARM_CLOCK (pygamelib.assets.graphics.MiscTechnical	
attribute), 69	animation (pygamelib.board_items.ComplexDoor at-
ALARM_CLOCK (pygamelib.assets.graphics.Models at-	tribute), 276
tribute), 140	animation (pygamelib.board_items.ComplexNPC at-
ALEMBIC (pygamelib.assets.graphics.Models attribute),	tribute), 285
140	animation (pygamelib.board_items.ComplexPlayer
ALIEN (pygamelib.assets.graphics.Models attribute),	attribute), 294
140	animation (pygamelib.board_items.ComplexTreasure
ALIEN_MONSTER (pygamelib.assets.graphics.Models	attribute), 303
attribute), 140	animation (pygamelib.board_items.ComplexWall at-
ALL_AROUND_PROFILE	tribute), 312
(pygamelib.assets.graphics.MiscTechnicals attribute), 69	animation (pygamelib.board_items.Door attribute), 321
ALTERNATIVE_KEY_SYMBOL	animation(pygamelib.board_items.GenericActionableStructure
(pygame lib. assets. graphics. MiscTechnicals	attribute), 329
attribute), 69	animation (pygamelib.board_items.GenericStructure
AMBULANCE (pygamelib.assets.graphics.Models at-	attribute), 345
tribute), 140	animation(pygamelib.board_items.GenericStructureComplexComponen
AMERICAN_FOOTBALL	attribute), 337
(pygamelib.assets.graphics.Models attribute), 140	animation (pygamelib.board_items.Immovable at- tribute), 353
AMPHORA (pygamelib.assets.graphics.Models attribute),	animation (pygamelib.board_items.Movable at-
140	tribute), 361

- animation (pygamelib.board_items.NPC attribute), 369
- animation (pygamelib.board_items.Player attribute), 378
- animation (pygamelib.board_items.Projectile attribute), 388
- animation (pygamelib.board_items.TextItem attribute), 398
- animation (pygamelib.board_items.Tile attribute), 407 animation (pygamelib.board_items.Treasure attribute), 416
- animation (pygamelib.board_items.Wall attribute), 424
- ANT (pygamelib.assets.graphics.Models attribute), 141 ANTENNA_BARS (pygamelib.assets.graphics.Models attribute), 141
- ANXIOUS_FACE_WITH_SWEAT (pygamelib.assets.graphics.Models attribute),
 141
- APL_FUNCTIONAL_SYMBOL_ALPHA (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_ALPHA_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_BACKSLASH_BAR (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_CIRCLE_BACKSLASH (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_CIRCLE_DIAERESIS (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_CIRCLE_JOT (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_CIRCLE_STAR (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_CIRCLE_STILE (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_CIRCLE_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_COMMA_BAR (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_DEL_DIAERESIS (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_DEL_STILE (pygamelib.assets.graphics.MiscTechnicals

- attribute), 69
- APL_FUNCTIONAL_SYMBOL_DEL_TILDE (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_DELTA_STILE (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_DELTA_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_DIAMOND_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 69
- APL_FUNCTIONAL_SYMBOL_DOWN_CARET_TILDE (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_DOWN_SHOE_STILE (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_DOWN_TACK_JOT (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_DOWN_TACK_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_DOWNWARDS_VANE (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_EPSILON_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_GREATER_THAN_DIAERESIS (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_I_BEAM (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_IOTA (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_IOTA_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_JOT_DIAERESIS (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_JOT_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_LEFT_SHOE_STILE (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_LEFTWARDS_VANE (pygamelib.assets.graphics.MiscTechnicals

- attribute), 70
- APL_FUNCTIONAL_SYMBOL_OMEGA (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL FUNCTIONAL SYMBOL OMEGA UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_BACKSLASH (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_CIRCLE (pygamelib. assets. graphics. Misc Technicalsattribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_COLON (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL FUNCTIONAL SYMBOL QUAD DEL (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_DELTA (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_DIAMOND (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_DIVIDE (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_DOWN_CARET (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL FUNCTIONAL SYMBOL OUAD DOWNWARDS ARRAWL FUNCTIONAL SYMBOL SOUISH OUAD (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_EQUAL (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_JOT (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_LESS_THAN (pygamelib. assets. graphics. Misc Technicalsattribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_NOT_EQUAL (pygamelib.assets.graphics.MiscTechnicals

- attribute), 70
- APL FUNCTIONAL SYMBOL QUAD QUESTION (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL FUNCTIONAL SYMBOL QUAD RIGHTWARDS ARROW (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL FUNCTIONAL SYMBOL QUAD SLASH (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_UP_CARET (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUAD_UPWARDS_ARROW (pygamelib.assets.graphics.MiscTechnicals attribute), 70
- APL_FUNCTIONAL_SYMBOL_QUOTE_QUAD (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- APL FUNCTIONAL SYMBOL QUOTE UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- APL_FUNCTIONAL_SYMBOL_RHO (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- APL FUNCTIONAL SYMBOL RIGHTWARDS VANE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- APL_FUNCTIONAL_SYMBOL_SEMICOLON_UNDERBAR (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- APL_FUNCTIONAL_SYMBOL_SLASH_BAR (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- (pygamelib.assets.graphics.MiscTechnicals attribute), 71
 - APL_FUNCTIONAL_SYMBOL_STAR_DIAERESIS (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- APL FUNCTIONAL SYMBOL QUAD GREATER THAN APL FUNCTIONAL SYMBOL STILE TILDE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
 - APL_FUNCTIONAL_SYMBOL_TILDE_DIAERESIS (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- APL_FUNCTIONAL_SYMBOL_QUAD_LEFTWARDS_ARRAWL_FUNCTIONAL_SYMBOL_UP_CARET_TILDE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
 - APL_FUNCTIONAL_SYMBOL_UP_SHOE_JOT (pygamelib.assets.graphics.MiscTechnicals attribute), 71
 - APL_FUNCTIONAL_SYMBOL_UP_TACK_DIAERESIS (pygamelib.assets.graphics.MiscTechnicals at-

tribute), 71 method), 18	
APL_FUNCTIONAL_SYMBOL_UP_TACK_JOT attach() (pygamelib.actuators.RandomActuator	
(pygamelib.assets.graphics.MiscTechnicals method), 28	
attribute), 71 attach() (pygamelib.actuators.UnidirectionalActuator APL_FUNCTIONAL_SYMBOL_UP_TACK_OVERBAR method), 32	
(pygamelib.assets.graphics.MiscTechnicals attach() (pygamelib.base.PglBaseObject method),	
attribute), 71	
APL_FUNCTIONAL_SYMBOL_UPWARDS_VANE attach() (pygamelib.base.Text method), 197	
(pygamelib.assets.graphics.MiscTechnicals attach() (pygamelib.board_items.Actionable	
attribute), 71 method), 208	
APL_FUNCTIONAL_SYMBOL_ZILDE attach() (pygamelib.board_items.ActionableTile (pygamelib.assets.graphics.MiscTechnicals method), 216	
attribute), 71 attach () (pygamelib.board_items.BoardComplexItem	
apply_force() (pygamelib.gfx.particles.CircleEmitter method), 225	
method), 547 attach() (pygamelib.board_items.BoardItem method),	
apply_force() (pygamelib.gfx.particles.ColorParticle 243	
method), 552 attach() (pygamelib.board_items.BoardItemComplexCompon	ent
apply_force() (pygamelib.gfx.particles.ColorPartitionParticle method), 234 method), 557 attach() (pygamelib.board_items.BoardItemVoid	
apply_force() (pygamelib.gfx.particles.Particle method), 251	
method), 571 attach() (pygamelib.board_items.Camera method),	
apply_force() (pygamelib.gfx.particles.ParticleEmitter 259	
method), 564 attach() (pygamelib.board_items.Character method),	
apply_force() (pygamelib.gfx.particles.PartitionParticle 268 method), 586 attach() (pygamelib.board_items.ComplexDoor	
apply_force() (pygamelib.gfx.particles.RandomColorParticle method), 276	
method), 590 attach() (pygamelib.board_items.ComplexNPC	
apply_force() (pygamelib.gfx.particles.RandomColorPartitionParticle), 285	
method), 596 attach() (pygamelib.board_items.ComplexPlayer AQUARIUS (pygamelib.assets.graphics.Models at method), 294	
tribute), 141 attach() (pygamelib.board_items.ComplexTreasure	
ARC (pygamelib.assets.graphics.MiscTechnicals at- method), 303	
tribute), 71 attach() (pygamelib.board_items.ComplexWall	
ARIES (pygamelib.assets.graphics.Models attribute), method), 312	
attach() (pygamelib.board_items.Door method), 321	_
ARTICULATED_LORRY attach() (pygamelib.board_items.GenericActionableStructure (pygamelib.assets.graphics.Models attribute), method), 329	<i>!</i>
141 attach() (pygamelib.board_items.GenericStructure	
ARTIST_PALETTE (pygamelib.assets.graphics.Models method), 345	
attribute), 141 attach() (pygamelib.board_items.GenericStructureComplexC	omponent
ASTONISHED_FACE (pygamelib.assets.graphics.Models method), 337	
attribute), 141 attach() (pygamelib.board_items.Immovable	
ATM_SIGN (pygamelib.assets.graphics.Models at method), 353	
tribute), 141 attach() (pygamelib.board_items.Movable method), ATOM_SYMBOL (pygamelib.assets.graphics.Models at-	
tribute), 141 attach () (pygamelib.board_items.NPC method), 369	
attach() (pygamelib.actuators.Actuator method), 8 attach() (pygamelib.board_items.Player method),	
attach() (pygamelib.actuators.Behavioral method), 378	
11 attach() (pygamelib.board_items.Projectile method),	
attach() (pygamelib.actuators.PathActuator method), 388	
14 attach() (pygamelib.board_items.TextItem method),	
a++ a ab / \ (magamalib gatugtons Path Finder math a A) 200	
attach() (pygamelib.actuators.PathFinder method), 398 22 attach() (pygamelib.board_items.Tile method), 407	

416	BACKHAND_INDEX_POINTING_DOWN
attach() (pygamelib.board_items.Wall method), 424	(pygamelib.assets.graphics.Models attribute),
attach() (pygamelib.engine.Board method), 435	141
attach() (pygamelib.engine.Game method), 448	BACKHAND_INDEX_POINTING_LEFT
attach() (pygamelib.engine.Inventory method), 462	(pygamelib.assets.graphics.Models attribute),
attach() (pygamelib.engine.Screen method), 471	141
attach() (pygamelib.gfx.core.Color method), 507	BACKHAND_INDEX_POINTING_RIGHT
attach() (pygamelib.gfx.core.Sprite method), 492	(pygamelib.assets.graphics.Models attribute),
attach() (pygamelib.gfx.core.Sprixel method), 500	141
attach() (pygamelib.gfx.particles.CircleEmitter	BACKHAND_INDEX_POINTING_UP
method), 547	(pygamelib.assets.graphics.Models attribute),
attach() (pygamelib.gfx.particles.ColorParticle	141
method), 552	BACKPACK (pygamelib.assets.graphics.Models at-
attach() (pygamelib.gfx.particles.ColorPartitionParticl	
method), 557	BACON (pygamelib.assets.graphics.Models attribute),
attach() (pygamelib.gfx.particles.Particle method),	141
571	BADGER (pygamelib.assets.graphics.Models attribute),
attach() (pygamelib.gfx.particles.ParticleEmitter	141
method), 565	BADMINTON (pygamelib.assets.graphics.Models at-
attach() (pygamelib.gfx.particles.ParticleSprixel	tribute), 141
method), 577	BAGEL (pygamelib.assets.graphics.Models attribute),
attach() (pygamelib.gfx.particles.PartitionParticle	141
method), 586	BAGGAGE_CLAIM (pygamelib.assets.graphics.Models
attach() (pygamelib.gfx.particles.RandomColorParticle	
method), 591	BAGUETTE_BREAD (pygamelib.assets.graphics.Models
attach() (pygamelib.gfx.particles.RandomColorPartitio	
method), 596	BALANCE_SCALE (pygamelib.assets.graphics.Models
AUTO_RICKSHAW (pygamelib.assets.graphics.Models	attribute), 142
attribute), 141	BALD (pygamelib.assets.graphics.Models attribute), 142
AUTOMOBILE (pygamelib.assets.graphics.Models	BALL (pygamelib.assets.graphics.Models attribute), 142
attribute), 141	BALLET_SHOES (pygamelib.assets.graphics.Models at-
available_space() (pygamelib.engine.Inventory	tribute), 142
method), 462	BALLOON (pygamelib.assets.graphics.Models attribute), 142
AVOCADO (pygamelib.assets.graphics.Models attribute), 141	
1.1	BALLOT_BOX_WITH_BALLOT (pygamelib.assets.graphics.Models attribute),
AXE (pygamelib.assets.graphics.Models attribute), 141	(pygametio.asseis.graphics.models attribute), 142
В	
	BANANA (pygamelib.assets.graphics.Models attribute), 142
b (pygamelib.gfx.core.Color attribute), 507	BANJO (pygamelib.assets.graphics.Models attribute),
B_BUTTON_BLOOD_TYPE	142
(pygamelib.assets.graphics.Models attribute),	BANK (pygamelib.assets.graphics.Models attribute), 142
144	BAR_CHART (pygamelib.assets.graphics.Models at-
BABY (pygamelib.assets.graphics.Models attribute), 141	tribute), 142
BABY_ANGEL (pygamelib.assets.graphics.Models	BARBER_POLE (pygamelib.assets.graphics.Models at-
attribute), 141	***
BABY_BOTTLE (pygamelib.assets.graphics.Models at-	tribute), 142
tribute), 141	BASEBALL (pygamelib.assets.graphics.Models attribute), 142
BABY_CHICK (pygamelib.assets.graphics.Models	DACKET (magmalib assets avanhias Madels attaileuts)
attribute), 141	BASKET (pygamelib.assets.graphics.Models attribute),
attribute), 141 BABY_SYMBOL (pygamelib.assets.graphics.Models at-	142
attribute), 141 BABY_SYMBOL (pygamelib.assets.graphics.Models attribute), 141	142 BASKETBALL (pygamelib.assets.graphics.Models
attribute), 141 BABY_SYMBOL (pygamelib.assets.graphics.Models attribute), 141 BACK_ARROW (pygamelib.assets.graphics.Models	142 BASKETBALL (pygamelib.assets.graphics.Models attribute), 142
attribute), 141 BABY_SYMBOL (pygamelib.assets.graphics.Models attribute), 141	142 BASKETBALL (pygamelib.assets.graphics.Models

142	143
BATTERY (pygamelib.assets.graphics.Models attribute),	black() (pygamelib.base.Text static method), 197
142	black_bright() (pygamelib.base.Text static
BEACH_WITH_UMBRELLA	method), 198
(pygamelib.assets.graphics.Models attribute), 142	BLACK_CIRCLE (pygamelib.assets.graphics.GeometricShapes attribute), 54
BEAMING_FACE_WITH_SMILING_EYES	BLACK_CIRCLE (pygamelib.assets.graphics.Models at-
(pygamelib.assets.graphics.Models attribute), 142	tribute), 143 BLACK_CIRCLE_FOR_RECORD
BEAR (pygamelib.assets.graphics.Models attribute), 142	(pygamelib.assets.graphics.MiscTechnicals
BEATING_HEART (pygamelib.assets.graphics.Models	attribute), 71
attribute), 142	BLACK_DIAMOND (pygamelib.assets.graphics.GeometricShapes
BEAVER (pygamelib.assets.graphics.Models attribute),	attribute), 54
142	black_dim() (pygamelib.base.Text static method), 198
BED (pygamelib.assets.graphics.Models attribute), 142	BLACK_DOWN_POINTING_DOUBLE_TRIANGLE
BEER_MUG (pygamelib.assets.graphics.Models attribute), 142	(pygamelib.assets.graphics.MiscTechnicals attribute), 71
BEETLE (pygamelib.assets.graphics.Models attribute),	BLACK_DOWN_POINTING_SMALL_TRIANGLE
142	(pygame lib. assets. graphics. Geometric Shapes
Behavioral (class in pygamelib.actuators), 10	attribute), 54
BELL (pygamelib.assets.graphics.Models attribute), 142	BLACK_DOWN_POINTING_TRIANGLE
BELL_PEPPER (pygamelib.assets.graphics.Models at-	(pygamelib.assets.graphics.GeometricShapes
tribute), 142	attribute), 54
BELL_SYMBOL (pygamelib.assets.graphics.MiscTechnica attribute), 71	lsBLACK_FLAG (pygamelib.assets.graphics.Models attribute), 143
BELL_WITH_SLASH (pygamelib.assets.graphics.Models attribute), 142	BLACK_HEART (pygamelib.assets.graphics.Models attribute), 143
BELLHOP_BELL (pygamelib.assets.graphics.Models at-	
tribute), 142	(pygamelib.assets.graphics.GeometricShapes
BENTO_BOX (pygamelib.assets.graphics.Models at-	attribute), 54
tribute), 142	BLACK_LARGE_SQUARE
BENZENE_RING (pygamelib.assets.graphics.MiscTechnic attribute), 71	rals (pygamelib.assets.graphics.Models attribute), 143
BENZENE_RING_WITH_CIRCLE	BLACK_LEFT_POINTING_DOUBLE_TRIANGLE
(pygamelib.assets.graphics.MiscTechnicals attribute), 71	(pygamelib.assets.graphics.MiscTechnicals attribute), 71
BEVERAGE_BOX (pygamelib.assets.graphics.Models at-	BLACK_LEFT_POINTING_DOUBLE_TRIANGLE_WITH_VERTICATION
tribute), 142	(pygamelib.assets.graphics.MiscTechnicals at-
bg_color (pygamelib.base.Text attribute), 197	tribute), 71
bg_color (pygamelib.gfx.core.Sprixel attribute), 500	BLACK_LEFT_POINTING_POINTER
<pre>bg_color (pygamelib.gfx.particles.ParticleSprixel at- tribute), 577</pre>	(pygamelib.assets.graphics.GeometricShapes attribute), 54
BICYCLE (pygamelib.assets.graphics.Models attribute),	BLACK_LEFT_POINTING_SMALL_TRIANGLE
142	(pygame lib. assets. graphics. Geometric Shapes
BIKINI (pygamelib.assets.graphics.Models attribute),	attribute), 54
142	BLACK_LEFT_POINTING_TRIANGLE
BILLED_CAP (pygamelib.assets.graphics.Models attribute), 142	(pygamelib.assets.graphics.GeometricShapes attribute), 54
BIOHAZARD (pygamelib.assets.graphics.Models at-	BLACK_LOWER_LEFT_TRIANGLE
tribute), 142	(pygamelib.assets.graphics.GeometricShapes
BIRD (pygamelib.assets.graphics.Models attribute), 142	attribute), 54
BIRTHDAY_CAKE (pygamelib.assets.graphics.Models	BLACK_LOWER_RIGHT_TRIANGLE
attribute), 143	(pygame lib. assets. graphics. Geometric Shapes
BISON (pygamelib.assets.graphics.Models attribute),	attribute), 54

- BLACK MEDIUM DOWN POINTING TRIANGLE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK_MEDIUM_LEFT_POINTING_TRIANGLE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK MEDIUM RIGHT POINTING TRIANGLE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK_MEDIUM_SMALL_SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK_MEDIUM_SMALL_SQUARE (pygamelib.assets.graphics.Models attribute),
- BLACK_MEDIUM_SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK MEDIUM SQUARE (pygamelib.assets.graphics.Models attribute),
- BLACK MEDIUM UP POINTING TRIANGLE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK_NIB (pygamelib.assets.graphics.Models attribute), 143
- BLACK_PARALLELOGRAM (pygamelib.assets.graphics.GeometricShapes attribute), 54
- black rect() (pygamelib.gfx.core.Sprixel class method), 500
- black_rect() (pygamelib.gfx.particles.ParticleSprixel class method), 577
- BLACK_RECTANGLE (pygamelib.assets.graphics.GeometricShapes (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK RIGHT POINTING DOUBLE TRIANGLE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK_RIGHT_POINTING_DOUBLE_TRIANGLE_WITH_VERTICASL_BAR (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK RIGHT POINTING POINTER (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK_RIGHT_POINTING_SMALL_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK_RIGHT_POINTING_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK_RIGHT_POINTING_TRIANGLE_WITH_DOUBLELVERHEERT_BAR (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK SMALL SQUARE

- (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK SMALL SQUARE (pygamelib.assets.graphics.Models attribute),
- BLACK SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- black_square() (pygamelib.gfx.core.Sprixel class method), 501
- black_square() (pygamelib.gfx.particles.ParticleSprixel class method), 578
- BLACK_SQUARE_BUTTON (pygamelib.assets.graphics.Models attribute), 143
- BLACK_SQUARE_FOR_STOP (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK UP POINTING DOUBLE TRIANGLE (pygamelib.assets.graphics.MiscTechnicals attribute), 71
- BLACK_UP_POINTING_SMALL_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK UP POINTING TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK_UPPER_LEFT_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK_UPPER_RIGHT_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 54
- BLACK_VERTICAL_RECTANGLE
- attribute), 54
- blend() (pygamelib.gfx.core.Color method), 508
- Blocks (class in pygamelib.assets.graphics), 36
- BLOSSOM (pygamelib.assets.graphics.Models attribute),
- BLOWFISH (pygamelib.assets.graphics.Models attribute), 143
- blue (pygamelib.gfx.ui.ColorPicker attribute), 516
- blue () (pygamelib.base.Text static method), 198
- BLUE_BOOK (pygamelib.assets.graphics.Models attribute), 143
- blue_bright() (pygamelib.base.Text static method), 198
- BLUE_CIRCLE (pygamelib.assets.graphics.Models attribute), 143
- blue_dim() (pygamelib.base.Text static method), 198
- attribute), 143
 - blue_rect() (pygamelib.gfx.core.Sprixel class method), 501

blue_rect() (pygamelib.gfx.particles.ParticleSprixel attribute), 72 class method), 578 BOTTOM SQUARE BRACKET OVER TOP SQUARE BRACKET BLUE SQUARE (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.MiscTechnicals attribute), 143 tribute), 72 blue square() (pygamelib.gfx.core.Sprixel BOTTOM_TORTOISE_SHELL_BRACKET method), 501 (pygamelib.assets.graphics.MiscTechnicals blue square() (pygamelib.gfx.particles.ParticleSprixel attribute), 72 class method), 578 BOUQUET (pygamelib.assets.graphics.Models attribute), BLUEBERRIES (pygamelib.assets.graphics.Models at-143 tribute), 143 BOW_AND_ARROW (pygamelib.assets.graphics.Models BOAR (pygamelib.assets.graphics.Models attribute), 143 attribute), 143 Board (class in pygamelib.engine), 432 BOWL_WITH_SPOON (pygamelib.assets.graphics.Models BoardComplexItem inattribute), 143 pygamelib.board_items), 224 BOWLING (pygamelib.assets.graphics.Models attribute), BoardItem (class in pygamelib.board_items), 241 143 BoardItemComplexComponent (class Box (class in pygamelib.gfx.ui), 512 pygamelib.board_items), 233 BoxDrawings (class in pygamelib.assets.graphics), 39 BoardItemVoid (class in pygamelib.board_items), BOXING_GLOVE (pygamelib.assets.graphics.Models attribute), 143 BOMB (pygamelib.assets.graphics.Models attribute), 143 BOY (pygamelib.assets.graphics.Models attribute), 143 BONE (pygamelib.assets.graphics.Models attribute), 143 BRAIN (pygamelib.assets.graphics.Models attribute), BOOKMARK (pygamelib.assets.graphics.Models tribute), 143 BREAD (pygamelib.assets.graphics.Models attribute), BOOKMARK TABS (pygamelib.assets.graphics.Models BREAST_FEEDING (pygamelib.assets.graphics.Models attribute), 143 BOOKS (pygamelib.assets.graphics.Models attribute), attribute), 143 143 (pygamelib.assets.graphics.Models attribute), BRICK (pygamelib.assets.graphics.Models BOOMERANG attribute), 143 BRIDGE_AT_NIGHT (pygamelib.assets.graphics.Models BOTTLE_WITH_POPPING_CORK attribute), 144 (pygamelib.assets.graphics.Models attribute), BRIEFCASE (pygamelib.assets.graphics.Models at-143 tribute), 144 BOTTOM_CURLY_BRACKET BRIEFS (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.MiscTechnicals 144 attribute), 71 BRIGHT_BUTTON (pygamelib.assets.graphics.Models BOTTOM HALF INTEGRAL attribute), 144 (pygamelib.assets.graphics.MiscTechnicals BROCCOLI (pygamelib.assets.graphics.Models atattribute), 71 tribute), 144 BOTTOM_LEFT_CORNER BROKEN_CIRCLE_WITH_NORTHWEST_ARROW (pygamelib.assets.graphics.MiscTechnicals (pygamelib.assets.graphics.MiscTechnicals attribute), 72 attribute), 72 BOTTOM_LEFT_CROP (pygamelib.assets.graphics.MiscTechnickIsn_HEART (pygamelib.assets.graphics.Models atattribute), 72 tribute), 144 (pygamelib.assets.graphics.Models attribute), BOTTOM_PARENTHESIS BROOM (pygamelib.assets.graphics.MiscTechnicals attribute), 72 BROWN_CIRCLE (pygamelib.assets.graphics.Models at-BOTTOM_RIGHT_CORNER tribute), 144 (pygamelib.assets.graphics.MiscTechnicals BROWN_HEART (pygamelib.assets.graphics.Models attribute), 144 attribute), 72 BOTTOM_RIGHT_CROP BROWN_SQUARE (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.MiscTechnicals tribute), 144 attribute), 72 BUBBLE TEA (pygamelib.assets.graphics.Models BOTTOM SQUARE BRACKET attribute), 144

620 Index

BUCKET (pygamelib.assets.graphics.Models attribute),

(pygamelib.assets.graphics.MiscTechnicals

144	<pre>can_move() (pygamelib.board_items.BoardItem</pre>
buffer (pygamelib.engine.Screen attribute), 471	method), 244
BUG (pygamelib.assets.graphics.Models attribute), 144 BUILDING_CONSTRUCTION	can_move() (pygamelib.board_items.BoardItemComplexComponent method), 235
(pygamelib.assets.graphics.Models attribute), 144	<pre>can_move() (pygamelib.board_items.BoardItemVoid method), 252</pre>
BULLET (pygamelib.assets.graphics.GeometricShapes attribute), 54	can_move() (pygamelib.board_items.Camera method), 260
BULLET_TRAIN (pygamelib.assets.graphics.Models attribute), 144	can_move() (pygamelib.board_items.Character method), 268
BULLSEYE (pygamelib.assets.graphics.GeometricShapes attribute), 54	can_move() (pygamelib.board_items.ComplexDoor method), 276
BURRITO (pygamelib.assets.graphics.Models attribute), 144	can_move() (pygamelib.board_items.ComplexNPC method), 285
BUS (pygamelib.assets.graphics.Models attribute), 144	<pre>can_move() (pygamelib.board_items.ComplexPlayer</pre>
BUS_STOP (pygamelib.assets.graphics.Models at-	method), 294
tribute), 144	can_move() (pygamelib.board_items.ComplexTreasure
BUST_IN_SILHOUETTE (pygamelib.assets.graphics.Models attribute),	method), 303 can_move() (pygamelib.board_items.ComplexWall
(pygameto.assets.graphics.modets airribate), 144	method), 312
BUSTS_IN_SILHOUETTE	can_move() (pygamelib.board_items.Door method),
(pygamelib.assets.graphics.Models attribute),	321
144	can_move() (pygamelib.board_items.GenericActionableStructure
BUTTER (pygamelib.assets.graphics.Models attribute), 144	method), 329 can_move() (pygamelib.board_items.GenericStructure
BUTTERFLY (pygamelib.assets.graphics.Models at-	method), 345
tribute), 144	can_move() (pygamelib.board_items.GenericStructureComplexComponemethod), 337
C	can_move() (pygamelib.board_items.Immovable
CACTUS (pygamelib.assets.graphics.Models attribute),	method), 353
144	can_move() (pygamelib.board_items.Movable
calculate_size() (pygamelib.gfx.core.Sprite	method), 361 can_move() (pygamelib.board_items.NPC method),
method), 492 CALENDAR (pygamelib.assets.graphics.Models at-	370
tribute), 144	can_move() (pygamelib.board_items.Player method),
CALL_ME_HAND (pygamelib.assets.graphics.Models at-	378
tribute), 144	can_move() (pygamelib.board_items.Projectile
CAMEL (pygamelib.assets.graphics.Models attribute),	method), 388 can_move() (pygamelib.board_items.TextItem
144 Camera (class in pygamelib.board_items), 258	method), 398
CAMERA (pygamelib.assets.graphics.Models attribute), 144	can_move() (pygamelib.board_items.Tile method), 408
CAMERA_WITH_FLASH	can_move() (pygamelib.board_items.Treasure
(pygamelib.assets.graphics.Models attribute), 144	method), 417 can_move() (pygamelib.board_items.Wall method),
CAMPING (pygamelib.assets.graphics.Models attribute), 144	425 CANCER (pygamelib.assets.graphics.Models attribute),
can_move() (pygamelib.board_items.Actionable method), 208	144 CANDLE (pygamelib.assets.graphics.Models attribute),
<pre>can_move() (pygamelib.board_items.ActionableTile</pre>	144
method), 217	CANDY (pygamelib.assets.graphics.Models attribute), 144
<pre>can_move() (pygamelib.board_items.BoardComplexItem method), 226</pre>	CANNED_FOOD (pygamelib.assets.graphics.Models at-
тетои), 220	tribute), 144

435

CHEESE_WEDGE (pygamelib.assets.graphics.Models at-CANOE (pygamelib.assets.graphics.Models attribute), 144 tribute), 145 CAPRICORN (pygamelib.assets.graphics.Models CHEQUERED FLAG (pygamelib.assets.graphics.Models tribute), 144 attribute), 145 CARD_FILE_BOX (pygamelib.assets.graphics.Models CHERRIES (pygamelib.assets.graphics.Models attribute), 145 tribute), 145 (pygamelib.assets.graphics.Models CHERRY BLOSSOM (pygamelib.assets.graphics.Models CARD INDEX attribute), 145 attribute), 145 CARD_INDEX_DIVIDERS CHESS_PAWN (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), attribute), 145 CHESTNUT (pygamelib.assets.graphics.Models at-CAROUSEL_HORSE (pygamelib.assets.graphics.Models tribute), 145 attribute), 145 CHICKEN (pygamelib.assets.graphics.Models attribute), CARP_STREAMER (pygamelib.assets.graphics.Models 145 (pygamelib.assets.graphics.Models attribute), attribute), 145 CHILD CARPENTRY_SAW (pygamelib.assets.graphics.Models 145 attribute), 145 CHILDREN_CROSSING CARROT (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), CASTLE (pygamelib.assets.graphics.Models attribute), CHIPMUNK (pygamelib.assets.graphics.Models tribute), 145 CAT (pygamelib.assets.graphics.Models attribute), 145 CHOCOLATE_BAR (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), 145 CAT_FACE atchoices (pygamelib.gfx.ui.GridSelector attribute), 523 tribute), 145 CAT_WITH_TEARS_OF_JOY CHOPSTICKS (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), attribute), 145 CHRISTMAS_TREE (pygamelib.assets.graphics.Models CAT_WITH_WRY_SMILE attribute), 145 (pygamelib.assets.graphics.Models attribute), CHURCH (pygamelib.assets.graphics.Models attribute), 145 CHAINS (pygamelib.assets.graphics.Models attribute), CIGARETTE (pygamelib.assets.graphics.Models at-145 tribute), 145 (pygamelib.assets.graphics.Models attribute), CINEMA (pygamelib.assets.graphics.Models attribute), 145 145 change_level() (pygamelib.engine.Game method), CIRCLE_WITH_ALL_BUT_UPPER_LEFT_QUADRANT_BLACK (pygamelib.assets.graphics.GeometricShapes Character (class in pygamelib.board items), 266 attribute), 54 CHART_DECREASING (pygamelib.assets.graphics.Models CIRCLE_WITH_LEFT_HALF_BLACK attribute), 145 (pygamelib.assets.graphics.GeometricShapes CHART_INCREASING (pygamelib.assets.graphics.Models attribute), 54 attribute), 145 CIRCLE WITH LOWER HALF BLACK CHART INCREASING WITH YEN (pygamelib.assets.graphics.GeometricShapes (pygamelib.assets.graphics.Models attribute), attribute), 54 145 CIRCLE_WITH_RIGHT_HALF_BLACK CHECK_BOX_WITH_CHECK (pygamelib.assets.graphics.GeometricShapes (pygamelib.assets.graphics.Models attribute), attribute), 55 145 CIRCLE_WITH_UPPER_HALF_BLACK (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.GeometricShapes CHECK_MARK attribute), 145 attribute), 55 CHECK_MARK_BUTTON CIRCLE_WITH_UPPER_RIGHT_QUADRANT_BLACK (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.GeometricShapes attribute), 55 check_sanity() (pygamelib.engine.Board method), CIRCLE_WITH_VERTICAL_FILL

622 Index

(pygamelib.assets.graphics.GeometricShapes

attribute), 55 CIRCLED HORIZONTAL BAR WITH NOTCH (pygamelib.assets.graphics.MiscTechnicals attribute), 72 CIRCLED M (pygamelib.assets.graphics.Models tribute), 146 CIRCLED TRIANGLE DOWN (pygamelib.assets.graphics.MiscTechnicals attribute), 72 CircleEmitter (class in pygamelib.gfx.particles), CIRCUS_TENT (pygamelib.assets.graphics.Models attribute), 146 (pygamelib.assets.graphics.Models CITYSCAPE tribute), 146 CITYSCAPE_AT_DUSK (pygamelib.assets.graphics.Models attribute), CL_BUTTON (pygamelib.assets.graphics.Models tribute), 146 CLAMP (pygamelib.assets.graphics.Models attribute), CLAPPER_BOARD (pygamelib.assets.graphics.Models attribute), 146 CLAPPING_HANDS (pygamelib.assets.graphics.Models attribute), 146 CLASSICAL_BUILDING (pygamelib.assets.graphics.Models attribute), clear () (pygamelib.engine.Screen method), 471 clear () (pygamelib.gfx.core.SpriteCollection method), 487 clear_buffers() (pygamelib.engine.Screen *method*), 471 clear_cell() (pygamelib.engine.Board method), 435 clear_constraints() (pygamelib.engine.Inventory method), 462 clear_frame_buffer() (pygamelib.engine.Screen *method*), 471 clear_screen() (pygamelib.engine.Game method), CLEAR SCREEN SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 72 (pygamelib.engine.Game clear_session_logs() method), 449 clear_waypoints() (pygamelib.actuators.PathFinder *method*), 23 CLINKING_BEER_MUGS

(pygamelib.assets.graphics.Models attribute),

CLINKING_GLASSES (pygamelib.assets.graphics.Models

attribute), 146

CLIPBOARD (pygamelib.assets.graphics.Models tribute), 146 CLOCKWISE VERTICAL ARROWS (pygamelib.assets.graphics.Models attribute), close() (pygamelib.gfx.ui.MenuBar method), 533 CLOSED BOOK (pygamelib.assets.graphics.Models attribute), 146 CLOSED_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), CLOSED_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), CLOSED_UMBRELLA (pygamelib.assets.graphics.Models attribute), 146 (pygamelib.assets.graphics.Models attribute), CLOUD 146 CLOUD WITH LIGHTNING (pygamelib.assets.graphics.Models attribute), CLOUD WITH LIGHTNING AND RAIN (pygamelib.assets.graphics.Models attribute), CLOUD_WITH_RAIN (pygamelib.assets.graphics.Models attribute), 146 CLOUD_WITH_SNOW (pygamelib.assets.graphics.Models attribute), 146 (pygamelib.assets.graphics.Models CLOWN_FACE attribute), 146 CLUB_SUIT (pygamelib.assets.graphics.Models attribute), 146 CLUTCH_BAG (pygamelib.assets.graphics.Models attribute), 146 COAT (pygamelib.assets.graphics.Models attribute), 146 COCKROACH (pygamelib.assets.graphics.Models attribute), 146 COCKTAIL_GLASS (pygamelib.assets.graphics.Models attribute), 146 COCONUT (pygamelib.assets.graphics.Models attribute), COFFIN (pygamelib.assets.graphics.Models attribute), COIN (pygamelib.assets.graphics.Models attribute), 146 COLD_FACE (pygamelib.assets.graphics.Models tribute), 146 collapse() (pygamelib.gfx.ui.Menu method), 527 collides_with() (pygamelib.board_items.Actionable method), 208 collides_with() (pygamelib.board_items.ActionableTile method), 217

collides_with() (pygamelib.board_items.BoardComplexItem

collides with () (pygamelib.board items.BoardItem

method), 226

method), 244	ColorP	icker (class in pygamelib.gfx.ui), 515
collides_with()(pygamelib.board_items.BoardItemC	Complex©	inapamentalog (class in pygamelib.gfx.ui), 513
method), 235	column	(pygamelib.base.Vector2D attribute), 204
collides_with() (pygamelib.board_items.BoardItemV	∕∞idlumn	(pygamelib.board_items.Actionable attribute),
method), 252		209
collides_with() (pygamelib.board_items.Camera	column	(pygamelib.board items.ActionableTile at-
method), 260		tribute), 217
collides_with()(pygamelib.board_items.Character		
method), 268		tribute), 226
collides_with() (pygamelib.board_items.ComplexDo		
method), 276	WO I WINIT	244
collides_with() (pygamelib.board_items.ComplexNF	26011mn	
method), 285		attribute), 235
collides_with() (pygamelib.board_items.ComplexPla		
method), 294		tribute), 252
collides_with()(pygamelib.board_items.ComplexTre		
method), 303		(pygamelib.board_items.Character attribute),
collides_with()(pygamelib.board_items.ComplexWa		268
method), 312	column	- 1
collides_with() (pygamelib.board_items.Door		tribute), 277
method), 321	column	_ 1
collides_with()(pygamelib.board_items.GenericAct	ionableStr	atrillourte), 286
method), 329	column	(pygamelib.board_items.ComplexPlayer at-
collides_with()(pygamelib.board_items.GenericStru	ıcture	tribute), 295
method), 345	column	(pygamelib.board_items.ComplexTreasure at-
collides_with()(pygamelib.board_items.GenericStru	uctureCom	plikxGo)µβ04ent
method), 337	column	(pygamelib.board_items.ComplexWall at-
<pre>collides_with() (pygamelib.board_items.Immovable</pre>		<i>tribute</i>), 313
method), 353		(pygamelib.board_items.Door attribute), 322
		(pygamelib.board_items.GenericActionableStructure
method), 361		attribute), 330
		(pygamelib.board_items.GenericStructure at-
method), 370		tribute), 346
		(pygamelib.board_items.GenericStructureComplexComponent
method), 378		attribute), 338
collides_with() (pygamelib.board_items.Projectile		
method), 388	COLUMNI	354
	aalumn	
collides_with() (pygamelib.board_items.TextItem	COLUMNI	
method), 398	7	362
collides_with() (pygamelib.board_items.Tile		(pygamelib.board_items.NPC attribute), 370
method), 408		(pygamelib.board_items.Player attribute), 379
collides_with() (pygamelib.board_items.Treasure	column	(pygamelib.board_items.Projectile attribute),
method), 417		389
collides_with() (pygamelib.board_items.Wall	column	(pygamelib.board_items.TextItem attribute),
method), 425		399
COLLISION (pygamelib.assets.graphics.Models at-		(pygamelib.board_items.Tile attribute), 408
tribute), 146	column	(pygamelib.board_items.Treasure attribute),
Color (class in pygamelib.gfx.core), 506		417
color (pygamelib.gfx.ui.ColorPicker attribute), 516	column	(pygamelib.board_items.Wall attribute), 425
colorable (pygamelib.gfx.core.Font attribute), 484	column	(pygamelib.gfx.particles.CircleEmitter at-
ColorParticle (class in pygamelib.gfx.particles),		tribute), 547
551	column	(pygamelib.gfx.particles.ColorParticle at-
ColorPartitionParticle (class in		tribute), 552
nyoamelih ofx narticles) 556	column	(pygamelih gfx.particles, ColorPartitionParticle

attribute), 558 CONICAL_TAPER (pygamelib.assets.graphics.MiscTechnicals column (pygamelib.gfx.particles.Particle attribute), 572 attribute), 72 (pygamelib.gfx.particles.ParticleEmitter Console (class in pygamelib.base), 190 tribute), 565 constraints (pygamelib.engine.Inventory attribute), column (pygamelib.gfx.particles.PartitionParticle attribute), 586 CONSTRUCTION (pygamelib.assets.graphics.Models atcolumn (pygamelib.gfx.particles.RandomColorParticle tribute), 147 CONSTRUCTION WORKER attribute), 591 (pygamelib.assets.graphics.Models attribute). column (pygamelib.gfx.particles.RandomColorPartitionParticle attribute), 596 (pygamelib.assets.graphics.Models attribute), CONTINUOUS_UNDERLINE_SYMBOL COMET (pygamelib.assets.graphics.MiscTechnicals COMPASS (pygamelib.assets.graphics.Models attribute), attribute), 72 CONTROL_KNOBS (pygamelib.assets.graphics.Models 146 ComplexDoor (class in pygamelib.board_items), 274 attribute), 147 ComplexNPC (class in pygamelib.board_items), 283 CONVENIENCE_STORE ComplexPlayer (class in pygamelib.board_items), (pygamelib.assets.graphics.Models attribute), ComplexTreasure (class in pygamelib.board_items), COOKED_RICE (pygamelib.assets.graphics.Models attribute), 147 ComplexWall (class in pygamelib.board_items), 310 COOKIE (pygamelib.assets.graphics.Models attribute), COMPOSITION SYMBOL 147 (pygamelib.assets.graphics.MiscTechnicals COOKING (pygamelib.assets.graphics.Models attribute), attribute), 72 COMPUTER_DISK (pygamelib.assets.graphics.Models COOL_BUTTON (pygamelib.assets.graphics.Models atattribute), 146 tribute), 147 ${\tt COMPUTER_MOUSE} \ \ (pygamelib. assets. graphics. Models$ copy () (pygamelib.gfx.core.Color method), 508 copy () (pygamelib.gfx.core.Sprite method), 492 attribute), 147 copy() (pygamelib.gfx.core.SpriteCollection method), CONFETTI_BALL (pygamelib.assets.graphics.Models attribute), 147 487 copy () (pygamelib.gfx.core.Sprixel method), 501 config (pygamelib.gfx.ui.Box attribute), 513 config (pygamelib.gfx.ui.ColorPickerDialog attribute), (pygamelib.gfx.particles.ParticleSprixel copy() method), 578 config (pygamelib.gfx.ui.Dialog attribute), 517 COPYRIGHT (pygamelib.assets.graphics.Models config (pygamelib.gfx.ui.FileDialog attribute), 519 tribute), 147 confia (pygamelib.gfx.ui.GridSelectorDialog COUCH_AND_LAMP (pygamelib.assets.graphics.Models attribute), 521 attribute), 147 config (pygamelib.gfx.ui.LineInputDialog attribute), count_active_particles() 525 (pygamelib.gfx.particles.ParticlePool method), config (pygamelib.gfx.ui.Menu attribute), 528 569 config (pygamelib.gfx.ui.MenuAction attribute), 531 COUNTERBORE (pygamelib.assets.graphics.MiscTechnicals config (pygamelib.gfx.ui.MenuBar attribute), 533 attribute), 72 config (pygamelib.gfx.ui.MessageDialog attribute), COUNTERCLOCKWISE ARROWS BUTTON 537 (pygamelib.assets.graphics.Models attribute), config (pygamelib.gfx.ui.MultiLineInputDialog tribute), 539 COUNTERSINK (pygamelib.assets.graphics.MiscTechnicals config (pygamelib.gfx.ui.ProgressBar attribute), 541 attribute), 72 config (pygamelib.gfx.ui.ProgressDialog attribute), COUPLE_WITH_HEART 543 (pygamelib.assets.graphics.Models attribute), config() (pygamelib.engine.Game method), 449 CONFOUNDED_FACE (pygamelib.assets.graphics.Models COW (pygamelib.assets.graphics.Models attribute), 147 (pygamelib.assets.graphics.Models attribute), 147 COW_FACE CONFUSED_FACE (pygamelib.assets.graphics.Models tribute), 147 attribute), 147 COWBOY HAT FACE (pygamelib.assets.graphics.Models

attribute), 147 CRAB (pygamelib.assets.graphics.Models attribute), 147 CRAYON (pygamelib.assets.graphics.Models attribute), create_config() (pygamelib.engine.Game method), CREDIT CARD (pygamelib.assets.graphics.Models attribute), 147 ${\tt CRESCENT_MOON} \quad \textit{(pygamelib.assets.graphics.Models}$ attribute), 147 CRICKET (pygamelib.assets.graphics.Models attribute), CRICKET_GAME (pygamelib.assets.graphics.Models attribute), 147 (pygamelib.assets.graphics.Models CROCODILE attribute), 147 (pygamelib.assets.graphics.Models CROISSANT attribute), 147 CROSS_MARK (pygamelib.assets.graphics.Models attribute), 147 CROSS_MARK_BUTTON (pygamelib.assets.graphics.Models attribute), CROSSED_FINGERS (pygamelib.assets.graphics.Models attribute), 147 CROSSED FLAGS (pygamelib.assets.graphics.Models attribute), 147 ${\tt CROSSED_SWORDS} \ \ \textit{(pygamelib.assets.graphics.Models}$ attribute), 147 (pygamelib.assets.graphics.Models attribute), CROWN 147 CRYING_CAT (pygamelib.assets.graphics.Models attribute), 147 CRYING_FACE (pygamelib.assets.graphics.Models attribute), 147 CRYSTAL_BALL (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.Models CUCUMBER tribute), 148 CUP_WITH_STRAW (pygamelib.assets.graphics.Models attribute), 148 CUPCAKE (pygamelib.assets.graphics.Models attribute), CURLING_STONE (pygamelib.assets.graphics.Models attribute), 148 CURLY_BRACKET_EXTENSION (pygamelib.assets.graphics.MiscTechnicals attribute), 72 (pygamelib.assets.graphics.Models CURLY_HAIR attribute), 148 CURLY_LOOP (pygamelib.assets.graphics.Models attribute), 148

(pygamelib.assets.graphics.Models attribute),

CURRENCY EXCHANGE

148 current_board() (pygamelib.engine.Game method), current_choice (pygamelib.gfx.ui.GridSelector attribute), 523 current_entry() (pygamelib.gfx.ui.Menu method), 528 current_entry() (pygamelib.gfx.ui.MenuBar method), 533 current_frame() (pygamelib.gfx.core.Animation *method*), 481 (pygamelib.gfx.ui.MenuBar current_index tribute), 534 current_page (pygamelib.gfx.ui.GridSelector tribute), 523 current_path() (pygamelib.actuators.PathFinder method), 23 current_sprixel() (pygamelib.gfx.ui.GridSelector method), 523 current waypoint() (pygamelib.actuators.PathFinder method), 23 CURRY_RICE (pygamelib.assets.graphics.Models attribute), 148 cursor_down() (pygamelib.gfx.ui.GridSelector method), 523 (pygamelib.gfx.ui.GridSelector cursor_left() method), 523 (pygamelib.gfx.ui.GridSelector cursor_right() method), 523 cursor_up() (pygamelib.gfx.ui.GridSelector method), 523 CUSTARD (pygamelib.assets.graphics.Models attribute), 148 CUSTOMS (pygamelib.assets.graphics.Models attribute), CUT OF MEAT (pygamelib.assets.graphics.Models attribute), 148 cyan () (pygamelib.base.Text static method), 198 cyan_bright() (pygamelib.base.Text static method), cyan_dim() (pygamelib.base.Text static method), 198 cyan rect() (pygamelib.gfx.core.Sprixel method), 501 cyan_rect() (pygamelib.gfx.particles.ParticleSprixel class method), 578 cyan_square() (pygamelib.gfx.core.Sprixel class method), 501 cyan_square() (pygamelib.gfx.particles.ParticleSprixel class method), 578 CYCLONE (pygamelib.assets.graphics.Models attribute),

CYLINDRICITY (pygamelib.assets.graphics.MiscTechnicals

attribute), 72

D	debug_info() (pygamelib.board_items.Player
DAGGER (pygamelib.assets.graphics.Models attribute),	method), 379
148	debug_info() (pygamelib.board_items.Projectile method), 389
DANGO (pygamelib.assets.graphics.Models attribute), 148	debug_info() (pygamelib.board_items.TextItem
DARK_SHADE (pygamelib.assets.graphics.Blocks at-	method), 399
tribute), 38	debug_info() (pygamelib.board_items.Tile method),
DARK_SKIN_TONE (pygamelib.assets.graphics.Models	408 debug_info() (pygamelib.board_items.Treasure
attribute), 148 DASHING_AWAY (pygamelib.assets.graphics.Models at-	method), 417
tribute), 148	debug_info() (pygamelib.board_items.Wall method),
DEAF_PERSON (pygamelib.assets.graphics.Models at-	425
tribute), 148	DECIDUOUS_TREE (pygamelib.assets.graphics.Models attribute), 148
debug() (pygamelib.base.Text static method), 198	DECIMAL_EXPONENT_SYMBOL
debug_info() (pygamelib.board_items.Actionable method), 209	(pygamelib.assets.graphics.MiscTechnicals
debug_info() (pygamelib.board_items.ActionableTile	attribute), 72
method), 217	DECIMAL_SEPARATOR_KEY_SYMBOL
<pre>debug_info() (pygamelib.board_items.BoardComplex.</pre>	attribute), 72
debug_info() (pygamelib.board_items.BoardItem	DEER (pygamelib.assets.graphics.Models attribute), 148
method), 245	delete() (pygamelib.engine.Screen method), 472 நடித்துந்துத்தி_levels() (pygamelib.engine.Game
debug_info()(pygamelib.board_items.BoardItemCommethod), 235	methoa), 450
<pre>debug_info() (pygamelib.board_items.BoardItemVoid</pre>	delete_item() (pygamelib.engine.Inventory method), 462
debug_info() (pygamelib.board_items.Camera	delete_items() (pygamelib.engine.Inventory
method), 260	method), 463
<pre>debug_info() (pygamelib.board_items.Character method), 269</pre>	delete_level() (pygamelib.engine.Game method), 450
$\verb debug_info (pygamelib.board_items.ComplexDoor $	delete_menu_category()
method), 277	(pygamelib.engine.Game method), 450 DELIVERY_TRUCK (pygamelib.assets.graphics.Models
debug_info() (pygamelib.board_items.ComplexNPC method), 286	attribute), 148
debug_info() (pygamelib.board_items.ComplexPlayer	DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL (pygamelib.assets.graphics.MiscTechnicals at-
<pre>method), 295 debug_info() (pygamelib.board_items.ComplexTreasu</pre>	7
method), 304	DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_C
debug_info() (pygamelib.board_items.ComplexWall	(pygamelib.assets.graphics.MiscTechnicals at-
method), 313	tribute), 72
debug_info() (pygamelib.board_items.Door	DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_TE (pygamelib.assets.graphics.MiscTechnicals at-
method), 322 debug_info() (pygamelib.board_items.GenericActional	***
method), 330	DENTISTRY_SYMBOL_LIGHT_DOWN_AND_HORIZONTAL_WITH_WA
debug_info() (pygamelib.board_items.GenericStructumethod), 346	re (pygamelib.assets.graphics.MiscTechnicals at- tribute), 72
debug_info() (pygamelib.board_items.GenericStructu	reConflexComponental
method), 338	(pygamelib.assets.graphics.MiscTechnicals at-
debug_info() (pygamelib.board_items.Immovable	<pre>tribute), 72 DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_CIRC</pre>
method), 354 debug_info() (pygamelib.board_items.Movable	(pygamelib.assets.graphics.MiscTechnicals at-
method), 362	tribute), 72
debug_info() (pygamelib.board_items.NPC	DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTAL_WITH_TRIA
method), 370	(pygamelib.assets.graphics.MiscTechnicals at-

```
tribute), 72
                                                            method), 217
DENTISTRY_SYMBOL_LIGHT_UP_AND_HORIZONTALdefallowAV@pygamelib.board_items.BoardComplexItem
        (pygamelib.assets.graphics.MiscTechnicals at-
                                                            method), 227
        tribute), 72
                                                   detach() (pygamelib.board_items.BoardItem method),
DENTISTRY SYMBOL LIGHT VERTICAL AND BOTTOM LEFT245
        (pygamelib.assets.graphics.MiscTechnicals at- detach() (pygamelib.board items.BoardItemComplexComponent
        tribute), 72
                                                            method), 235
DENTISTRY SYMBOL LIGHT VERTICAL AND BOTTOMERAGET
                                                                   (pygamelib.board_items.BoardItemVoid
        (pygamelib.assets.graphics.MiscTechnicals at-
                                                            method), 252
        tribute), 72
                                                    detach()
                                                               (pygamelib.board_items.Camera method),
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_TOP_LEFT
                                                            260
        (pygamelib.assets.graphics.MiscTechnicals at- detach() (pygamelib.board_items.Character method),
        tribute), 72
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_TOP_REGUTH ()
                                                                    (pygamelib.board_items.ComplexDoor
        (pygamelib.assets.graphics.MiscTechnicals at-
                                                            method), 277
        tribute), 72
                                                    detach()
                                                                    (pygamelib.board_items.ComplexNPC
DENTISTRY_SYMBOL_LIGHT_VERTICAL_AND_WAVE
                                                            method), 286
        (pygamelib.assets.graphics.MiscTechnicals at- detach()
                                                                   (pygamelib.board_items.ComplexPlayer
                                                            method), 295
        tribute), 72
DENTISTRY SYMBOL LIGHT VERTICAL WITH CIRCLETA ()
                                                                 (pygamelib.board items.ComplexTreasure
        (pygamelib.assets.graphics.MiscTechnicals at-
                                                            method), 304
                                                                    (pygamelib.board_items.ComplexWall
                                                    detach()
DENTISTRY_SYMBOL_LIGHT_VERTICAL_WITH_TRIANGLE method), 313
        (pygamelib.assets.graphics.MiscTechnicals at- detach() (pygamelib.board_items.Door method), 322
                                                    detach() (pygamelib.board_items.GenericActionableStructure
        tribute), 72
DEPARTMENT_STORE (pygamelib.assets.graphics.Models
                                                            method), 330
        attribute), 148
                                                                 (pygamelib.board_items.GenericStructure
                                                    detach()
DERELICT_HOUSE (pygamelib.assets.graphics.Models
                                                            method), 346
        attribute), 148
                                                    detach() (pygamelib.board_items.GenericStructureComplexComponent
DESERT (pygamelib.assets.graphics.Models attribute),
                                                            method), 338
        148
                                                    detach()
                                                                       (pygamelib.board_items.Immovable
DESERT_ISLAND (pygamelib.assets.graphics.Models
                                                            method), 354
        attribute), 148
                                                    detach() (pygamelib.board_items.Movable method),
DESKTOP_COMPUTER (pygamelib.assets.graphics.Models
                                                            362
        attribute), 148
                                                    detach() (pygamelib.board_items.NPC method), 370
detach() (pygamelib.actuators.Actuator method), 8
                                                    detach() (pygamelib.board_items.Player method),
detach() (pygamelib.actuators.Behavioral method),
                                                            379
        11
                                                   detach() (pygamelib.board_items.Projectile method),
detach() (pygamelib.actuators.PathActuator method),
                                                   detach() (pygamelib.board_items.TextItem method),
        15
detach() (pygamelib.actuators.PathFinder method),
        23
                                                    detach() (pygamelib.board_items.Tile method), 408
                                                   detach() (pygamelib.board_items.Treasure method),
                  (pygamelib.actuators.PatrolActuator
detach()
        method), 18
                                                            418
                                                   detach() (pygamelib.board_items.Wall method), 426
detach()
                (pygamelib.actuators.RandomActuator
        method), 29
                                                    detach() (pygamelib.engine.Board method), 436
detach() (pygamelib.actuators.UnidirectionalActuator
                                                   detach() (pygamelib.engine.Game method), 450
        method), 32
                                                    detach() (pygamelib.engine.Inventory method), 463
detach()
           (pygamelib.base.PglBaseObject method),
                                                   detach() (pygamelib.engine.Screen method), 472
        193
                                                    detach() (pygamelib.gfx.core.Color method), 508
detach() (pygamelib.base.Text method), 198
                                                    detach() (pygamelib.gfx.core.Sprite method), 492
                  (pygamelib.board_items.Actionable
                                                   detach() (pygamelib.gfx.core.Sprixel method), 501
detach()
        method), 209
                                                    detach()
                                                                    (pygamelib.gfx.particles.CircleEmitter
               (pygamelib.board items.ActionableTile
detach()
                                                            method), 547
```

<pre>detach() (pygamelib.gfx.particles.ColorParticle method), 552</pre>	<pre>method), 218 display() (pygamelib.board_items.BoardComplexItem</pre>
detach() (pygamelib.gfx.particles.ColorPartitionParticles.method), 558	
$\verb detach() (pygame lib. gfx. particles. Particle method),$	method), 245
detach() (pygamelib.gfx.particles.ParticleEmitter	display() (pygamelib.board_items.BoardItemComplexComponent method), 236
method), 565	display() (pygamelib.board_items.BoardItemVoid
detach() (pygamelib.gfx.particles.ParticleSprixel method), 578	method), 253 display() (pygamelib.board_items.Camera method),
<pre>detach() (pygamelib.gfx.particles.PartitionParticle</pre>	261 display() (pygamelib.board_items.Character
detach() (pygamelib.gfx.particles.RandomColorParticle	
method), 591	display() (pygamelib.board_items.ComplexDoor
<pre>detach() (pygamelib.gfx.particles.RandomColorPartition</pre>	mrarncie meinoa), 277 display()
DETECTIVE (pygamelib.assets.graphics.Models at-	method), 286
tribute), 148 Dialog (class in pygamelib.gfx.ui), 517	display() (pygamelib.board_items.ComplexPlayer method), 295
$\verb DIAMETER_SIGN (pygamelib. assets. graphics. Misc Techn$	icalissplay() (pygamelib.board_items.ComplexTreasure
attribute), 73 DIAMOND_SUIT (pygamelib.assets.graphics.Models at-	method), 304 display() (pygamelib.board_items.ComplexWall
tribute), 148	method), 313
DIAMOND_WITH_A_DOT (pygamelib.assets.graphics.Models attribute),	display() (pygamelib.board_items.Door method), 322
148	display() (pygamelib.board_items.GenericActionableStructure
DIM_BUTTON (pygamelib.assets.graphics.Models attribute), 148	method), 330 display() (pygamelib.board_items.GenericStructure
DIMENSION_ORIGIN (pygamelib.assets.graphics.MiscT	echnicals method), 346
<pre>attribute), 73 DIRECT_CURRENT_SYMBOL_FORM_TWO</pre>	<pre>display() (pygamelib.board_items.GenericStructureComplexComponer</pre>
(pygame lib. assets. graphics. Misc Technicals	display() (pygamelib.board_items.Immovable
attribute), 73 DIRECT HIT (pygamelib.assets.graphics.Models	method), 354 display() (pygamelib.board_items.Movable method),
attribute), 148	362
direction (pygamelib.board_items.Projectile attribute), 389	display() (pygamelib.board_items.NPC method), 371 display() (pygamelib.board_items.Player method),
directional_animation()	379
(pygamelib.board_items.Projectile method), 390	display() (pygamelib.board_items.Projectile method), 390
directional_model()	display() (pygamelib.board_items.TextItem method),
(pygamelib.board_items.Projectile method), 390	399 display() (pygamelib.board_items.Tile method), 409
DISAPPOINTED_FACE	display() (pygamelib.board_items.Treasure method),
(pygamelib.assets.graphics.Models attribute), 148	418 display() (pygamelib.board_items.Wall method), 426
DISCONTINUOUS_UNDERLINE_SYMBOL	display() (pygamelib.engine.Board method), 436
(pygamelib.assets.graphics.MiscTechnicals attribute), 73	display_around() (pygamelib.engine.Board method), 436
DISGUISED_FACE (pygamelib.assets.graphics.Models	display_at() (pygamelib.engine.Screen method),
attribute), 148 display() (pygamelib.board_items.Actionable	472 display_board() (pygamelib.engine.Game method),
method), 209 display() (pygamelih hoggd items Actionable Tile)	451 display line() (pygamelib.engine.Screen method),
albertay () (pygameno.boara nems.AcnonableThe	albert in the contraction of the

473	distance_to() (pygamelib.board_items.TextItem
display_menu() (pygamelib.engine.Game method),	method), 399
451	distance_to() (pygamelib.board_items.Tile
display_player_stats()	method), 409
(pygamelib.engine.Game method), 451 display_sprite() (pygamelib.engine.Screen	distance_to() (pygamelib.board_items.Treasure method), 418
method), 474	distance_to() (pygamelib.board_items.Wall
display_sprite_at() (pygamelib.engine.Screen	method), 426
method), 474	DIVIDE (pygamelib.assets.graphics.Models attribute),
distance() (pygamelib.base.Math static method), 191	148
	DIVING_MASK (pygamelib.assets.graphics.Models at-
method), 209	tribute), 148
<pre>distance_to() (pygamelib.board_items.ActionableTile</pre>	tribute), 148
<pre>distance_to() (pygamelib.board_items.BoardComplex</pre>	148
<pre>distance_to() (pygamelib.board_items.BoardItem</pre>	DIZZY_FACE (pygamelib.assets.graphics.Models attribute), 149
<pre>distance_to() (pygamelib.board_items.BoardItemCon</pre>	
method), 236	DODO (pygamelib.assets.graphics.Models attribute), 149
distance_to() (pygamelib.board_items.BoardItemVoid	
method), 253 distance_to() (pygamelib.board_items.Camera	DOG_FACE (pygamelib.assets.graphics.Models attribute), 149
method), 261	DOLLAR_BANKNOTE (pygamelib.assets.graphics.Models
distance_to() (pygamelib.board_items.Character	attribute), 149
method), 269	DOLPHIN (pygamelib.assets.graphics.Models attribute),
distance_to()(pygamelib.board_items.ComplexDoor	
method), 277	Door (class in pygamelib.board_items), 319
distance_to() (pygamelib.board_items.ComplexNPC	
method), 286	DOTTED_CIRCLE (pygamelib.assets.graphics.GeometricShape.
<pre>distance_to() (pygamelib.board_items.ComplexPlaye</pre>	or attribute), 55 DOTTED_SIX_POINTED_STAR
distance_to() (pygamelib.board_items.ComplexTreas	
method), 304	149
<pre>distance_to() (pygamelib.board_items.ComplexWall</pre>	(pygamelib.assets.graphics.Models attribute),
distance_to() (pygamelib.board_items.Door	149
method), 322	DOUBLE_DOWN_AND_HORIZONTAL
distance_to() (pygamelib.board_items.GenericAction	
method), 330	attribute), 45
distance_to() (pygamelib.board_items.GenericStruct	
method), 346	(pygamelib.assets.graphics.BoxDrawings
distance_to() (pygamelib.board_items.GenericStruct	
<pre>method), 338 distance_to() (pygamelib.board_items.Immovable</pre>	DOUBLE_DOWN_AND_RIGHT (pygamelib.assets.graphics.BoxDrawings
method), 354	attribute), 45
distance_to() (pygamelib.board_items.Movable	DOUBLE_EXCLAMATION_MARK
method), 362	(pygamelib.assets.graphics.Models attribute),
distance_to() (pygamelib.board_items.NPC	149
method), 371	DOUBLE_HORIZONTAL
distance_to() (pygamelib.board_items.Player	(pygamelib.assets.graphics.BoxDrawings
method), 379	attribute), 45
distance_to() (pygamelib.board_items.Projectile	DOUBLE_UP_AND_HORIZONTAL
method), 390	(pygamelib.assets.graphics.BoxDrawings

attribute), 45 attribute), 46 DOUBLE_UP_AND_LEFT DOWN_LEFT_ARROW (pygamelib.assets.graphics.Models attribute), 149 (pygamelib.assets.graphics.BoxDrawings attribute), 45 DOWN_LIGHT_AND_HORIZONTAL_HEAVY DOUBLE UP AND RIGHT (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 46 attribute), 45 DOWN LIGHT AND LEFT HEAVY DOUBLE_VERTICAL (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 45 attribute), 46 DOWN_LIGHT_AND_LEFT_UP_HEAVY DOUBLE_VERTICAL_AND_HORIZONTAL (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 45 attribute), 46 DOUBLE_VERTICAL_AND_LEFT DOWN_LIGHT_AND_RIGHT_HEAVY (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 45 attribute), 46 DOWN_LIGHT_AND_RIGHT_UP_HEAVY DOUBLE_VERTICAL_AND_RIGHT (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 45 attribute), 46 DOUBLE_VERTICAL_BAR DOWN LIGHT AND UP HORIZONTAL HEAVY (pygamelib.assets.graphics.MiscTechnicals (pygamelib.assets.graphics.BoxDrawings attribute), 73 attribute), 46 DOUGHNUT (pygamelib.assets.graphics.Models DOWN_RIGHT_ARROW (pygamelib.assets.graphics.Models tribute), 149 attribute), 149 DOVE (pygamelib.assets.graphics.Models attribute), 149 DOWN SINGLE AND HORIZONTAL DOUBLE DOWN ARROW (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.BoxDrawings attribute), 149 attribute), 46 DOWN_ARROWHEAD (pygamelib.assets.graphics.MiscTechnicals*N_SINGLE_AND_LEFT_DOUBLE (pygamelib.assets.graphics.BoxDrawings attribute), 73 DOWN_DOUBLE_AND_HORIZONTAL_SINGLE attribute), 46 (pygamelib.assets.graphics.BoxDrawings DOWN_SINGLE_AND_RIGHT_DOUBLE attribute), 45 (pygamelib.assets.graphics.BoxDrawings DOWN_DOUBLE_AND_LEFT_SINGLE attribute), 46 (pygamelib.assets.graphics.BoxDrawings DOWNCAST_FACE_WITH_SWEAT attribute), 45 (pygamelib.assets.graphics.Models attribute), DOWN DOUBLE AND RIGHT SINGLE (pygamelib.assets.graphics.BoxDrawings DOWNWARDS BUTTON (pygamelib.assets.graphics.Models attribute), 45 attribute), 149 DOWN_HEAVY_AND_HORIZONTAL_LIGHT DRAGON (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.BoxDrawings 149 attribute), 45 DRAGON_FACE (pygamelib.assets.graphics.Models at-DOWN HEAVY AND LEFT LIGHT tribute), 149 (pygamelib.assets.graphics.BoxDrawings DRESS (pygamelib.assets.graphics.Models attribute), attribute), 45 DOWN_HEAVY_AND_LEFT_UP_LIGHT DROOLING_FACE (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.BoxDrawings attribute), 149 attribute), 45 DROP_OF_BLOOD (pygamelib.assets.graphics.Models DOWN_HEAVY_AND_RIGHT_LIGHT attribute), 149 (pygamelib.assets.graphics.BoxDrawings DROPLET (pygamelib.assets.graphics.Models attribute), attribute), 45 DOWN_HEAVY_AND_RIGHT_UP_LIGHT DRUM (pygamelib.assets.graphics.Models attribute), 149 (pygamelib.assets.graphics.BoxDrawings dtanimate (pygamelib.gfx.core.Animation attribute), attribute), 45 DOWN_HEAVY_AND_UP_HORIZONTAL_LIGHT dtmove (pygamelib.board items.Camera attribute), 261 (pygamelib.assets.graphics.BoxDrawings dtmove (pygamelib.board items.Character attribute),

	269	ELEPHANT (pygamelib.assets.graphics.Models at-
dtmove		tribute), 150
	tribute), 286	ELEVATOR (pygamelib.assets.graphics.Models at-
dtmove	e (pygamelib.board_items.ComplexPlayer at- tribute), 296	tribute), 150 ELEVEN_OCLOCK (pygamelib.assets.graphics.Models
dtmove		attribute), 150 ELEVEN_THIRTY (pygamelib.assets.graphics.Models
dtmove	e (pygamelib.board_items.NPC attribute), 371	attribute), 150
	e (pygamelib.board_items.Player attribute), 380	ELF (pygamelib.assets.graphics.Models attribute), 150
dtmove	e (pygamelib.board_items.Projectile attribute), 390	emit () (pygamelib.gfx.particles.CircleEmitter method), 548
	ygamelib.assets.graphics.Models attribute), 149	emit() (pygamelib.gfx.particles.ParticleEmitter
DUMPLI	1.0	method), 565
DIID (mu	tribute), 149	EmitterProperties (class in
E (Py	gamelib.assets.graphics.Models attribute), 149	pygamelib.gfx.particles), 561 EMPHASIS_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 73
	(magmalih assats aranhias Madala, attributa)	empty() (pygamelib.engine.Inventory method), 463
E_MAIL	(pygamelib.assets.graphics.Models attribute), 150	empty() (pygamelib.gfx.core.Sprite method), 492
EAGLE	(pygamelib.assets.graphics.Models attribute), 149	empty_marker (pygamelib.gfx.ui.ProgressBar attribute), 541
EAR (pyg	gamelib.assets.graphics.Models attribute), 149	END_ARROW (pygamelib.assets.graphics.Models at-
EAR_OF	C_CORN (pygamelib.assets.graphics.Models at-	tribute), 150
	tribute), 149	ENTER_SYMBOL (pygamelib.assets.graphics.MiscTechnicals
EAR_WI	TH_HEARING_AID	attribute), 73 entries (pygamelib.gfx.ui.Menu attribute), 528
	(pygamelib.assets.graphics.Models attribute), 149	entries (pygamelib.gfx.ui.MenuBar attribute), 528
EARTH_	_GROUND (pygamelib.assets.graphics.MiscTechnica attribute), 73	
EGG (pv	gamelib.assets.graphics.Models attribute), 149	ENVELOPE_WITH_ARROW
EGGPLA		(pygamelib.assets.graphics.Models attribute), 150
EIGHT_	OCLOCK (pygamelib.assets.graphics.Models at-	ERASE_TO_THE_LEFT
	tribute), 149	(pygame lib. assets. graphics. MiscTechnicals
EIGHT_	POINTED_STAR	attribute), 73
	(pygamelib.assets.graphics.Models attribute),	ERASE_TO_THE_RIGHT
DICHE	149	(pygamelib.assets.graphics.MiscTechnicals attribute), 73
EIGHI_	_SPOKED_ASTERISK (pygamelib.assets.graphics.Models attribute),	EURO_BANKNOTE (pygamelib.assets.graphics.Models
	150	attribute), 150
EIGHT_	_THIRTY (pygamelib.assets.graphics.Models attribute), 150	EVERGREEN_TREE (pygamelib.assets.graphics.Models attribute), 150
EJECT_	BUTTON (pygamelib.assets.graphics.Models attribute), 150	EWE (pygamelib.assets.graphics.Models attribute), 150 EXCLAMATION_MARK (pygamelib.assets.graphics.Models
EJECT_	_SYMBOL (pygamelib.assets.graphics.MiscTechnica	als attribute), 150
	attribute), 73	EXCLAMATION_QUESTION_MARK
ELECTR	RIC_ARROW (pygamelib.assets.graphics.MiscTechnattribute), 73	150
ELECTR	RIC_PLUG (pygamelib.assets.graphics.Models attribute), 150	expand() (pygamelib.gfx.ui.Menu method), 528 EXPLODING_HEAD (pygamelib.assets.graphics.Models
ELECTR	RICAL_INTERSECTION	attribute), 150
	(pygamelib.assets.graphics.MiscTechnicals	EXPRESSIONLESS_FACE
	attribute), 73	(pygamelib.assets.graphics.Models attribute), 150

EYE (pygamelib.assets.graphics.Models attribute), 150 FACTORY (pygamelib.assets.graphics.Models attribute), EYES (pygamelib.assets.graphics.Models attribute), 150 151 (pygamelib.assets.graphics.Models attribute), FAIRY F FALAFEL (pygamelib.assets.graphics.Models attribute), FACE BLOWING A KISS 151 (pygamelib.assets.graphics.Models attribute), FALLEN LEAF (pygamelib.assets.graphics.Models at-150 *tribute*), 151 FACE_SAVORING_FOOD FAMILY (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), 151 150 FAST_DOWN_BUTTON (pygamelib.assets.graphics.Models FACE_SCREAMING_IN FEAR attribute), 151 (pygamelib.assets.graphics.Models attribute), FAST FORWARD BUTTON (pygamelib.assets.graphics.Models attribute), FACE_VOMITING (pygamelib.assets.graphics.Models attribute), 150 FAST_REVERSE_BUTTON FACE WITH HAND OVER MOUTH (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), FAST_UP_BUTTON (pygamelib.assets.graphics.Models FACE_WITH_HEAD_BANDAGE attribute), 151 (pygamelib.assets.graphics.Models attribute), fatal () (pygamelib.base.Text static method), 198 150 FAX_MACHINE (pygamelib.assets.graphics.Models at-FACE WITH MEDICAL MASK tribute), 151 (pygamelib.assets.graphics.Models attribute), FEARFUL FACE (pygamelib.assets.graphics.Models at-150 FACE_WITH_MONOCLE tribute), 151 FEATHER (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), 151 FEMALE_SIGN (pygamelib.assets.graphics.Models at-FACE_WITH_OPEN_MOUTH tribute), 151 (pygamelib.assets.graphics.Models attribute), FERRIS_WHEEL (pygamelib.assets.graphics.Models at-150 *tribute*), 151 FACE_WITH_RAISED_EYEBROW FERRY (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), 151 150 fg_color (pygamelib.base.Text attribute), 198 FACE_WITH_ROLLING EYES fg color (pygamelib.gfx.core.Sprixel attribute), 502 (pygamelib.assets.graphics.Models attribute), fg_color (pygamelib.gfx.particles.ParticleSprixel at*tribute*), 579 FACE_WITH_STEAM_FROM_NOSE FIELD_HOCKEY (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.Models attribute), tribute), 151 150 (pygamelib.gfx.ui.MultiLineInputDialog atfields FACE WITH SYMBOLS ON MOUTH tribute), 539 (pygamelib.assets.graphics.Models attribute), FILE_CABINET (pygamelib.assets.graphics.Models at-150 tribute), 151 FACE_WITH_TEARS_OF_JOY FILE_FOLDER (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.Models attribute), tribute), 151 151 FileDialog (class in pygamelib.gfx.ui), 517 FACE_WITH_THERMOMETER FILM_FRAMES (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.Models attribute), tribute), 151 $\verb|FACE_WITH_TONGUE| (pygamelib. assets. graphics. Models \verb|FILM_PROJECTOR| (pygamelib. assets. graphics. Models assets. graphics. grap$ attribute), 151 attribute), 151 filter (pygamelib.gfx.ui.FileDialog attribute), 519 FACE WITHOUT MOUTH (pygamelib.actuators.PathFinder find path() (pygamelib.assets.graphics.Models attribute), method), 24 150 finished() (pygamelib.gfx.particles.CircleEmitter

method), 548 (pygamelib.assets.graphics.Models FLATBREAD (pygamelib.gfx.particles.ColorParticle finished() tribute), 152 *method*), 553 FLATNESS (pygamelib.assets.graphics.MiscTechnicals finished() (pygamelib.gfx.particles.ColorPartitionParticle attribute), 73 *method*), 558 FLEUR_DE_LIS (pygamelib.assets.graphics.Models atfinished() (pygamelib.gfx.particles.Particle tribute), 152 FLEXED BICEPS (pygamelib.assets.graphics.Models method), 572 finished() (pygamelib.gfx.particles.ParticleEmitter attribute), 152 *method*), 565 flip_horizontally() (pygamelib.gfx.core.Sprite finished() (pygamelib.gfx.particles.PartitionParticle *method*), 493 method), 586 flip_vertically() (pygamelib.gfx.core.Sprite*method*), 493 finished()(pygamelib.gfx.particles.RandomColorParticle *method*), 591 FLOPPY_DISK (pygamelib.assets.graphics.Models atfinished() (pygamelib.gfx.particles.RandomColorPartitionParticleribute), 152 method), 596 FLOWER_PLAYING_CARDS FIRE (pygamelib.assets.graphics.Models attribute), 151 (pygamelib.assets.graphics.Models attribute), FIRE_ENGINE (pygamelib.assets.graphics.Models at-152 tribute), 151 FLUSHED_FACE (pygamelib.assets.graphics.Models at-FIRE_EXTINGUISHER tribute), 152 (pygamelib.assets.graphics.Models attribute), FLY (pygamelib.assets.graphics.Models attribute), 152 FLYING_DISC (pygamelib.assets.graphics.Models at-FIRECRACKER (pygamelib.assets.graphics.Models attribute), 152 tribute), 151 FLYING_SAUCER (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models FIREWORKS attribute), 152 tribute), 151 FOG (pygamelib.assets.graphics.Models attribute), 152 FIRST PLACE MEDAL FOGGY (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), 152 FOLDED_HANDS (pygamelib.assets.graphics.Models at-151 FIRST_QUARTER_MOON tribute), 152 (pygamelib.assets.graphics.Models attribute), FONDUE (pygamelib.assets.graphics.Models attribute), 151 152 FIRST_QUARTER_MOON_FACE Font (class in pygamelib.gfx.core), 483 (pygamelib.assets.graphics.Models attribute), FOOT (pygamelib.assets.graphics.Models attribute), 152 (pygamelib.assets.graphics.Models FOOTPRINTS FISH (pygamelib.assets.graphics.Models attribute), 151 attribute), 152 FISH CAKE WITH SWIRL force_render() (pygamelib.engine.Screen method), (pygamelib.assets.graphics.Models attribute), force_update() (pygamelib.engine.Screen method), FISHEYE (pygamelib.assets.graphics.GeometricShapes attribute), 55 FORK_AND_KNIFE (pygamelib.assets.graphics.Models FISHING POLE (pygamelib.assets.graphics.Models atattribute), 152 tribute), 151 FORK AND KNIFE WITH PLATE FIVE_OCLOCK (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.Models attribute), tribute), 151 152 FORTUNE_COOKIE (pygamelib.assets.graphics.Models FIVE_THIRTY (pygamelib.assets.graphics.Models atattribute), 152 tribute), 151 FLAG_IN_HOLE (pygamelib.assets.graphics.Models at-FOUNTAIN (pygamelib.assets.graphics.Models tribute), 152 *tribute*), 152 FLAMINGO (pygamelib.assets.graphics.Models FOUNTAIN_PEN (pygamelib.assets.graphics.Models attribute), 152 tribute), 152 FOUR_LEAF_CLOVER (pygamelib.assets.graphics.Models FLASHLIGHT (pygamelib.assets.graphics.Models attribute), 152 attribute), 152 FLAT SHOE (pygamelib.assets.graphics.Models at-FOUR_OCLOCK (pygamelib.assets.graphics.Models attribute), 152 tribute), 152

FOUR THIRTY (pygamelib.assets.graphics.Models at- GEAR (pygamelib.assets.graphics.Models attribute), 153 tribute), 152 (pygamelib.assets.graphics.Models at-GEM STONE FOX (pygamelib.assets.graphics.Models attribute), 152 *tribute*), 153 FRAMED_PICTURE (pygamelib.assets.graphics.Models GEMINI (pygamelib.assets.graphics.Models attribute), attribute), 152 FREE BUTTON (pygamelib.assets.graphics.Models atgenerate void cell() (pygamelib.engine.Board method), 436 tribute), 152 FRENCH_FRIES (pygamelib.assets.graphics.Models at-GenericActionableStructure (class in tribute), 152 pygamelib.board_items), 327 FRIED_SHRIMP (pygamelib.assets.graphics.Models at-GenericStructure (class inpygamelib.board_items), 343 tribute), 152 GenericStructureComplexComponent (class in FROG (pygamelib.assets.graphics.Models attribute), 152 (pygamelib.gfx.core.Color pygamelib.board_items), 335 from_ansi() class (pygamelib.assets.graphics.Models attribute), method), 508 GENIE (pygamelib.gfx.core.Sprixel 153 from_ansi() static method), 502 GeometricShapes (class infrom_ansi() (pygamelib.gfx.particles.ParticleSprixel pygamelib.assets.graphics), 49 static method), 579 get () (pygamelib.engine.Screen method), 475 from_direction() (pygamelib.base.Vector2D class (pygamelib.gfx.core.SpriteCollection method), get() method), 204 from_text() (pygamelib.gfx.core.Sprite class get_board() (pygamelib.engine.Game method), 452 *method*), 493 get immovables() (pygamelib.engine.Board (pygamelib.gfx.core.SpriteCollection method), 437 fromkeys() class method), 487 (pygamelib.engine.Inventory method), get item() FRONT FACING BABY CHICK 463 (pygamelib.assets.graphics.Models attribute), get_items() (pygamelib.engine.Inventory method), 464 (pygamelib.assets.graphics.MiscTechnicals atget_key() (pygamelib.engine.Game static method), FROWN tribute), 73 452 FROWNING_FACE (pygamelib.assets.graphics.Models (pygamelib.engine.Game get_menu_entry() attribute), 152 method), 452 FROWNING_FACE_WITH_OPEN_MOUTH get_movables() (pygamelib.engine.Board method), (pygamelib.assets.graphics.Models attribute), get_particles() (pygamelib.gfx.particles.ParticlePool FUEL PUMP (pygamelib.assets.graphics.Models method), 569 at-(pygamelib.assets.graphics.Models attribute), tribute), 153 GHOST FULL BLOCK (pygamelib.assets.graphics.Blocks at-153 tribute), 38 GIRAFFE (pygamelib.assets.graphics.Models attribute), FULL MOON (pygamelib.assets.graphics.Models 153 attribute), 153 GIRL (pygamelib.assets.graphics.Models attribute), 153 FULL MOON FACE (pygamelib.assets.graphics.Models GLASS OF MILK (pygamelib.assets.graphics.Models attribute), 153 attribute), 153 GLASSES (pygamelib.assets.graphics.Models attribute), FUNERAL URN (pygamelib.assets.graphics.Models at-153 tribute), 153 (pygamelib. assets. graphics. Misc TechnicalsFUSE GLOBE_SHOWING_AMERICAS tribute), 73 (pygamelib.assets.graphics.Models attribute), 153 G GLOBE_SHOWING_ASIA_AUSTRALIA (pygamelib.assets.graphics.Models attribute), g (pygamelib.gfx.core.Color attribute), 509 Game (class in pygamelib.engine), 443 GLOBE_SHOWING_EUROPE_AFRICA GAME_DIE (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), tribute), 153 153 GARLIC (pygamelib.assets.graphics.Models attribute),

Index 635

153

GLOBE WITH MERIDIANS

(pygamelib.assets.graphics.Models attribute),	tribute), 153
153	GRINNING_CAT_WITH_SMILING_EYES
GLOVES (pygamelib.assets.graphics.Models attribute), 153	(pygamelib.assets.graphics.Models attribute), 154
GLOWING_STAR (pygamelib.assets.graphics.Models attribute), 153	GRINNING_FACE (pygamelib.assets.graphics.Models attribute), 154
glyph() (pygamelib.gfx.core.Font method), 484	GRINNING_FACE_WITH_BIG_EYES
glyphs_map (pygamelib.gfx.core.Font attribute), 485 GOAL_NET (pygamelib.assets.graphics.Models at-	(pygamelib.assets.graphics.Models attribute), 154
tribute), 153	GRINNING_FACE_WITH_SMILING_EYES
GOAT (pygamelib.assets.graphics.Models attribute), 153 GOBLIN (pygamelib.assets.graphics.Models attribute),	(pygamelib.assets.graphics.Models attribute), 154
153	GRINNING_FACE_WITH_SWEAT
GOGGLES (pygamelib.assets.graphics.Models attribute), 153	(pygamelib.assets.graphics.Models attribute), 154
GORILLA (pygamelib.assets.graphics.Models attribute), 153	GRINNING_SQUINTING_FACE (pygamelib.assets.graphics.Models attribute),
GRADUATION_CAP (pygamelib.assets.graphics.Models attribute), 153	154 GROWING_HEART (pygamelib.assets.graphics.Models
GRAPES (pygamelib.assets.graphics.Models attribute),	attribute), 154
153 green (pygamelib.gfx.ui.ColorPicker attribute), 516	GUARD (pygamelib.assets.graphics.Models attribute), 154
green () (pygamelib.base.Text static method), 199	GUIDE_DOG (pygamelib.assets.graphics.Models at-
GREEN_APPLE (pygamelib.assets.graphics.Models at-	tribute), 154
tribute), 153	${\tt GUITAR}\ \ (pygame lib. assets. graphics. Models\ \ attribute),$
GREEN_BOOK (pygamelib.assets.graphics.Models attribute), 153	154
<pre>green_bright() (pygamelib.base.Text static</pre>	Н
method), 199	HacException, 205
GREEN_CIRCLE (pygamelib.assets.graphics.Models at-	HacInvalidLevelException, 205
tribute), 153	HacInvalidTypeException, 206
green_dim() (pygamelib.base.Text static method), 199 GREEN_HEART (pygamelib.assets.graphics.Models at-	HacObjectIsNotMovableException, 206
tribute), 153	HacOutOfBoardBoundException, 206 HAMBURGER (pygamelib.assets.graphics.Models at-
green_rect() (pygamelib.gfx.core.Sprixel class	tribute), 154
method), 502 green_rect() (pygamelib.gfx.particles.ParticleSprixel	HAMMER (pygamelib.assets.graphics.Models attribute), 154
class method), 579	HAMMER_AND_PICK (pygamelib.assets.graphics.Models
GREEN_SALAD (pygamelib.assets.graphics.Models at-	attribute), 154
tribute), 153	HAMMER_AND_WRENCH
GREEN_SQUARE (pygamelib.assets.graphics.Models attribute), 153	(pygamelib.assets.graphics.Models attribute), 154
<pre>green_square() (pygamelib.gfx.core.Sprixel class method), 502</pre>	HAMSTER (pygamelib.assets.graphics.Models attribute), 154
<pre>green_square() (pygamelib.gfx.particles.ParticleSprix</pre>	adand_with_fingers_splayed
class method), 579	(pygamelib.assets.graphics.Models attribute),
<pre>grid_selector(pygamelib.gfx.ui.GridSelectorDialog</pre>	154
attribute), 521	HANDBAG (pygamelib.assets.graphics.Models attribute),
GridSelector (class in pygamelib.gfx.ui), 521	154
GridSelectorDialog (class in pygamelib.gfx.ui),	handle_notification()
520	(and a many 121 and
CRIMACING FACE (nyaqmalih assats aranhics Madals	(pygamelib.actuators.Actuator method),
GRIMACING_FACE (pygamelib.assets.graphics.Models attribute), 153	<pre>(pygamelib.actuators.Actuator method), 9 handle_notification()</pre>

12	handle_notification()
handle_notification()	(pygamelib.board_items.ComplexTreasure
(pygamelib.actuators.PathActuator method),	method), 305
15	handle_notification()
handle_notification()	(pygamelib.board_items.ComplexWall
(pygamelib.actuators.PathFinder method),	method), 313
24	handle_notification()
handle_notification()	(pygamelib.board_items.Door method), 323
(pygamelib.actuators.PatrolActuator method),	handle_notification() (magnetib heard items ConsistationableStructure)
handle_notification()	(pygamelib.board_items.GenericActionableStructure method), 331
(pygamelib.actuators.RandomActuator	handle_notification()
method), 29	(pygamelib.board_items.GenericStructure
handle_notification()	method), 347
(pygamelib.actuators.UnidirectionalActuator	handle_notification()
method), 32	(pygamelib.board_items.GenericStructureComplexComponent
handle_notification()	method), 338
(pygamelib.base.PglBaseObject method),	handle_notification()
(pygameno.oase.1 gibaseOojeci memoa), 193	(pygamelib.board_items.Immovable method),
handle_notification() (pygamelib.base.Text	354
method), 199	handle_notification()
handle_notification()	(pygamelib.board_items.Movable method),
(pygamelib.board_items.Actionable method),	363
210	handle_notification()
handle_notification()	(pygamelib.board_items.NPC method), 371
(pygamelib.board_items.ActionableTile	handle_notification()
method), 218	(pygamelib.board_items.Player method),
handle_notification()	380
(pygamelib.board_items.BoardComplexItem	handle_notification()
method), 227	(pygamelib.board_items.Projectile method),
handle_notification()	390
(pygamelib.board_items.BoardItem method),	handle notification()
245	(pygamelib.board_items.TextItem method),
handle_notification()	399
(pygamelib.board_items.BoardItemComplexCom	phaendle notification()
method), 236	(pygamelib.board_items.Tile method), 409
handle_notification()	handle_notification()
(pygamelib.board_items.BoardItemVoid	(pygamelib.board_items.Treasure method),
method), 253	418
handle_notification()	handle_notification()
(pygamelib.board_items.Camera method),	(pygamelib.board_items.Wall method), 426
261	handle_notification() (pygamelib.engine.Board
handle_notification()	method), 437
(pygamelib.board_items.Character method),	handle_notification() (pygamelib.engine.Game
269	method), 452
handle_notification()	handle_notification()
(pygamelib.board_items.ComplexDoor	(pygamelib.engine.Inventory method), 464
method), 277	handle_notification()
handle_notification()	(pygamelib.engine.Screen method), 476
(pygamelib.board_items.ComplexNPC	handle_notification()
method), 286	(pygamelib.gfx.core.Color method), 509
handle_notification()	handle_notification()
(pygamelib.board_items.ComplexPlayer	(pygamelib.gfx.core.Sprite method), 493
method) 296	handle notification()

(pygamelib.gfx.core.Sprixel method), 502 handle_notification()	attribute), 227 heading (pygamelib.board_items.BoardItem attribute),
(pygamelib.gfx.particles.CircleEmitter method), 548	245 heading (pygamelib.board_items.BoardItemComplexComponent
handle_notification()	attribute), 236
(pygamelib.gfx.particles.ColorParticle method), 553	heading (pygamelib.board_items.BoardItemVoid attribute), 253
handle_notification()	heading (pygamelib.board_items.Camera attribute),
<pre>(pygamelib.gfx.particles.ColorPartitionParticle method), 558 handle_notification()</pre>	261 heading (pygamelib.board_items.Character attribute), 270
(pygamelib.gfx.particles.Particle method), 572	heading (pygamelib.board_items.ComplexDoor at- tribute), 278
handle_notification() (pygamelib.gfx.particles.ParticleEmitter	heading (pygamelib.board_items.ComplexNPC attribute), 287
method), 566	heading (pygamelib.board_items.ComplexPlayer at-
handle_notification() (pygamelib.gfx.particles.ParticleSprixel	tribute), 296 heading (pygamelib.board_items.ComplexTreasure at-
method), 579	tribute), 305
handle_notification() (pygamelib.gfx.particles.PartitionParticle	heading (pygamelib.board_items.ComplexWall at- tribute), 314
method), 586	heading (pygamelib.board_items.Door attribute), 323
<pre>handle_notification()</pre>	heading (pygamelib.board_items.GenericActionableStructure attribute), 331
method), 591	heading (pygamelib.board_items.GenericStructure at-
handle_notification()	tribute), 347
(pygametib.gfx.particles.KandomColorPartitionF method), 597	Panticleting (pygamelib.board_items.GenericStructureComplexComponent attribute), 339
	heading (pygamelib.board_items.Immovable attribute), 355
has_inventory() (pygamelib.board_items.Camera method), 261	heading (pygamelib.board_items.Movable attribute), 363
has_inventory()(pygamelib.board_items.Character	
<pre>method), 270 has_inventory() (pygamelib.board_items.ComplexNI</pre>	heading (pygamelib.board_items.Player attribute), 380 PGeading (pygamelib.board_items.Projectile attribute),
method), 287	391
has_inventory()(pygamelib.board_items.ComplexPl method), 296	400
has_inventory() (pygamelib.board_items.Movable method), 363	heading (pygamelib.board_items.Tile attribute), 409 heading (pygamelib.board_items.Treasure attribute),
has_inventory() (pygamelib.board_items.NPC	418
method), 371	heading (pygamelib.board_items.Wall attribute), 426
has_inventory() (pygamelib.board_items.Player method), 380	HEADPHONE (pygamelib.assets.graphics.Models at- tribute), 154
has_inventory()(pygamelib.board_items.Projectile method), 391	HEADSTONE (pygamelib.assets.graphics.Models attribute), 154
HATCHING_CHICK (pygamelib.assets.graphics.Models	HEAR_NO_EVIL_MONKEY
attribute), 154 hcenter (pygamelib.engine.Screen attribute), 476	(pygamelib.assets.graphics.Models attribute), 154
heading (pygamelib.board_items.Actionable attribute),	HEART_DECORATION (pygamelib.assets.graphics.Models
210	attribute), 154
heading (pygamelib.board_items.ActionableTile attribute), 218	HEART_EXCLAMATION (pygamelib.assets.graphics.Models attribute),
heading (pygamelib.board_items.BoardComplexItem	154

HEART_SUIT (pygamelib.assets.graphics.Models	(pygamelib.assets.graphics.BoxDrawings
attribute), 154	attribute), 46
HEART_WITH_ARROW (pygamelib.assets.graphics.Models	
attribute), 154	(pygamelib.assets.graphics.BoxDrawings
HEART_WITH_RIBBON	attribute), 46
(pygamelib.assets.graphics.Models attribute), 154	
	(pygamelib.assets.graphics.BoxDrawings attribute), 46
HEAVY_DOLLAR_SIGN (magnetic anaphies Models attribute)	
(pygamelib.assets.graphics.Models attribute), 154	HEAVY_VERTICAL (pygamelib.assets.graphics.BoxDrawings attribute), 46
HEAVY_DOUBLE_DASH_HORIZONTAL	HEAVY_VERTICAL_AND_HORIZONTAL
(pygamelib. assets. graphics. Box Drawings	(pygamelib.assets.graphics.BoxDrawings
attribute), 46	attribute), 46
HEAVY_DOUBLE_DASH_VERTICAL	HEAVY_VERTICAL_AND_LEFT
(pygamelib. assets. graphics. Box Drawings	(pygamelib.assets.graphics.BoxDrawings
attribute), 46	attribute), 46
HEAVY_DOWN (pygamelib.assets.graphics.BoxDrawings	HEAVY_VERTICAL_AND_RIGHT
attribute), 46	(pygamelib.assets.graphics.BoxDrawings
HEAVY_DOWN_AND_HORIZONTAL	attribute), 46
(pygamelib.assets.graphics.BoxDrawings attribute), 46	HEDGEHOG (pygamelib.assets.graphics.Models attribute), 154
HEAVY_DOWN_AND_LEFT	height (pygamelib.board_items.Actionable attribute),
(pygamelib.assets.graphics.BoxDrawings	210
attribute), 46	height (pygamelib.board_items.ActionableTile at-
HEAVY_DOWN_AND_RIGHT	tribute), 219
(pygamelib.assets.graphics.BoxDrawings	height (pygamelib.board_items.BoardComplexItem at-
attribute), 46	tribute), 228
HEAVY_HORIZONTAL (pygamelib.assets.graphics.BoxDrattribute), 46	ам ы вунс (pygameno.boara_nems.boaranem anribute), 246
	height (pygamelib.board_items.BoardItemComplexComponent
attribute), 46	attribute), 237
HEAVY_LEFT_AND_LIGHT_RIGHT	height (pygamelib.board_items.BoardItemVoid at-
(pygamelib.assets.graphics.BoxDrawings	tribute), 254
attribute), 46	height (pygamelib.board_items.Camera attribute), 262
HEAVY_QUADRUPLE_DASH_HORIZONTAL	height (pygamelib.board_items.Character attribute),
(pygamelib. assets. graphics. BoxDrawings	270
attribute), 46	height (pygamelib.board_items.ComplexDoor at-
HEAVY_QUADRUPLE_DASH_VERTICAL	tribute), 278
(pygamelib.assets.graphics.BoxDrawings	height (pygamelib.board_items.ComplexNPC at-
attribute), 46	tribute), 287
HEAVY_RIGHT (pygamelib.assets.graphics.BoxDrawings	
attribute), 46	tribute), 296
HEAVY_TRIPLE_DASH_HORIZONTAL	height (pygamelib.board_items.ComplexTreasure at-
(pygamelib.assets.graphics.BoxDrawings	tribute), 305
attribute), 46 HEAVY_TRIPLE_DASH_VERTICAL	height (pygamelib.board_items.ComplexWall at-
(pygamelib.assets.graphics.BoxDrawings	tribute), 314 height (pygamelib.board_items.Door attribute), 323
attribute), 46	height (pygamelib.board_items.GenericActionableStructure
HEAVY_UP (pygamelib.assets.graphics.BoxDrawings at-	attribute), 331
tribute), 46	height (pygamelib.board_items.GenericStructure at-
HEAVY_UP_AND_HORIZONTAL	tribute), 347
(pygamelib.assets.graphics.BoxDrawings	height (pygamelib.board_items.GenericStructureComplexComponent
attribute), 46	attribute), 339
HEAVY_UP_AND_LEFT	height (pygamelib.board_items.Immovable attribute),

355	attribute), 73
height (pygamelib.board_items.Movable attribute),	HORIZONTAL_SCAN_LINE_3
363	(pygame lib. assets. graphics. Misc Technicals
height (pygamelib.board_items.NPC attribute), 372	attribute), 73
height (pygamelib.board_items.Player attribute), 380	HORIZONTAL_SCAN_LINE_7
height (pygamelib.board_items.Projectile attribute), 391	(pygamelib.assets.graphics.MiscTechnicals attribute), 73
height (pygamelib.board_items.TextItem attribute),	HORIZONTAL_SCAN_LINE_9
400	(pygame lib. assets. graphics. Misc Technicals
height (pygamelib.board_items.Tile attribute), 410	attribute), 73
height (pygamelib.board_items.Treasure attribute), 419	horizontal_spacing (pygamelib.gfx.core.Font attribute), 485
height (pygamelib.board_items.Wall attribute), 427	HORIZONTAL_TRAFFIC_LIGHT
height (pygamelib.engine.Board attribute), 437	(pygamelib.assets.graphics.Models attribute),
height (pygamelib.engine.Screen attribute), 476	155
height (pygamelib.gfx.core.Font attribute), 485	HORSE (pygamelib.assets.graphics.Models attribute),
height (pygamelib.gfx.core.Sprite attribute), 494	155
height (pygamelib.gfx.ui.Box attribute), 513	HORSE_FACE (pygamelib.assets.graphics.Models
height (pygamelib.gfx.ui.MessageDialog attribute), 537	attribute), 155
HELICOPTER (pygamelib.assets.graphics.Models	HORSE_RACING (pygamelib.assets.graphics.Models attribute), 155
attribute), 154	HOSPITAL (pygamelib.assets.graphics.Models at-
HELM_SYMBOL (pygamelib.assets.graphics.MiscTechnical	
attribute), 73	HOT_BEVERAGE (pygamelib.assets.graphics.Models at-
HERB (pygamelib.assets.graphics.Models attribute), 154	tribute), 155
	HOT_DOG (pygamelib.assets.graphics.Models attribute),
tribute), 154	155
HIGH_HEELED_SHOE (pygamelib.assets.graphics.Model.	s HOT_FACE (pygamelib.assets.graphics.Models at-
attribute), 154	tribute), 155
$\verb HIGH_SPEED_TRAIN (pygamelib. assets. graphics. Model and the state of the stat$	
attribute), 154	attribute), 155
HIGH_VOLTAGE (pygamelib.assets.graphics.Models attribute), 154	HOT_SPRINGS (pygamelib.assets.graphics.Models attribute), 155
HIKING_BOOT (pygamelib.assets.graphics.Models attribute), 154	HOTEL (pygamelib.assets.graphics.Models attribute), 155
HINDU_TEMPLE (pygamelib.assets.graphics.Models attribute), 155	HOURGLASS (pygamelib.assets.graphics.MiscTechnicals attribute), 73
HIPPOPOTAMUS (pygamelib.assets.graphics.Models attribute), 155	
hit () (pygamelib.board_items.Projectile method), 391	HOURGLASS_NOT_DONE
HOLE (pygamelib.assets.graphics.Models attribute), 155	(pygamelib.assets.graphics.Models attribute),
HOLLOW_RED_CIRCLE	155
(pygamelib.assets.graphics.Models attribute),	HOURGLASS_WITH_FLOWING_SAND
155	(pygame lib. assets. graphics. Misc Technicals
HONEY_POT (pygamelib.assets.graphics.Models at-	attribute), 73
tribute), 155	HOUSE (pygamelib.assets.graphics.MiscTechnicals at-
HONEYBEE (pygamelib.assets.graphics.Models at-	tribute), 73
tribute), 155	HOUSE (pygamelib.assets.graphics.Models attribute),
HOOK (pygamelib.assets.graphics.Models attribute), 155	155
HORIZONTAL_LINE_EXTENSION	HOUSE_WITH_GARDEN
(pygamelib.assets.graphics.MiscTechnicals attribute), 73	(pygamelib.assets.graphics.Models attribute), 155
HORIZONTAL_SCAN_LINE_1 (pygamelib.assets.graphics.MiscTechnicals	HOUSES (pygamelib.assets.graphics.Models attribute), 155

```
HUGGING_FACE (pygamelib.assets.graphics.Models at instance () (pygamelib.base.Console class method),
              tribute), 155
                                                                                                  190
HUNDRED POINTS (pygamelib.assets.graphics.Models
                                                                                    instance() (pygamelib.engine.Game class method),
              attribute), 155
                                                                                                  453
HUSHED_FACE (pygamelib.assets.graphics.Models at-
                                                                                    instance()
                                                                                                                (pygamelib.gfx.ui.UiConfig
                                                                                                                                                              class
             tribute), 155
                                                                                                  method), 545
HUT (pygamelib.assets.graphics.Models attribute), 155
                                                                                                                                 (pygamelib.engine.Board
                                                                                    instantiate item()
HYSTERESIS_SYMBOL
                                                                                                  static method), 438
              (pygamelib.assets.graphics.MiscTechnicals
                                                                                    INTEGRAL EXTENSION
                                                                                                  (pygamelib.assets.graphics.MiscTechnicals
              attribute), 73
                                                                                                  attribute), 73
                                                                                    intersect() (pygamelib.base.Math static method),
ICE (pygamelib.assets.graphics.Models attribute), 155
ICE_CREAM (pygamelib.assets.graphics.Models
                                                                                    Inventory (class in pygamelib.engine), 459
                                                                                    inventory_space(pygamelib.board_items.Actionable
              tribute), 155
                                                                                                  attribute), 211
ICE HOCKEY
                               (pygamelib.assets.graphics.Models
                                                                                    inventory_space(pygamelib.board_items.ActionableTile
             attribute), 155
                                                                                                  attribute), 219
ICE_SKATE
                     (pygamelib.assets.graphics.Models
                                                                            at-
                                                                                    inventory_space(pygamelib.board_items.BoardComplexItem
             tribute), 155
                                                                                                  attribute), 228
ID_BUTTON (pygamelib.assets.graphics.Models
                                                                            at-
                                                                                    inventory_space(pygamelib.board_items.BoardItem
             tribute), 155
                                                                                                  attribute), 246
Immovable (class in pygamelib.board_items), 351
                                                                                    inventory_space(pygamelib.board_items.BoardItemComplexCompon
INBOX_TRAY
                               (pygamelib.assets.graphics.Models
                                                                                                  attribute), 237
             attribute), 155
INCOMING_ENVELOPE
                                                                                    inventory_space(pygamelib.board_items.BoardItemVoid
                                                                                                  attribute), 254
              (pygamelib.assets.graphics.Models attribute),
                                                                                    inventory_space (pygamelib.board_items.Camera
                                                                                                  attribute), 262
INDEX POINTING UP
                                                                                    inventory_space(pygamelib.board_items.Character
              (pygamelib.assets.graphics.Models attribute),
                                                                                                  attribute), 270
              155
                                                                                    inventory_space(pygamelib.board_items.ComplexDoor
INFINITY
                      (pygamelib.assets.graphics.Models
                                                                                                  attribute), 278
              tribute), 155
                                                                                    inventory_space(pygamelib.board_items.ComplexNPC
info() (pygamelib.base.Text static method), 199
                                                                                                  attribute), 288
INFORMATION (pygamelib.assets.graphics.Models at-
                                                                                    inventory_space(pygamelib.board_items.ComplexPlayer
             tribute), 156
init_board() (pygamelib.engine.Board method), 438
                                                                                                  attribute), 297
                                                                                    inventory_space(pygamelib.board_items.ComplexTreasure
init_cell() (pygamelib.engine.Board method), 438
                                                                                                  attribute), 305
INPUT_LATIN_LETTERS
                                                                                    \verb"inventory_space" (pygame lib.board\_items. Complex Wall")
              (pygamelib.assets.graphics.Models attribute),
                                                                                                  attribute), 314
              156
                                                                                    inventory_space (pygamelib.board_items.Door at-
INPUT_LATIN_LOWERCASE
                                                                                                  tribute), 324
              (pygamelib.assets.graphics.Models attribute),
                                                                                    inventory_space(pygamelib.board_items.GenericActionableStructure
                                                                                                  attribute), 332
INPUT_LATIN_UPPERCASE
                                                                                    inventory_space(pygamelib.board_items.GenericStructure
              (pygamelib.assets.graphics.Models attribute),
              156
                                                                                                  attribute), 348
                                                                                    inventory_space (pygamelib.board_items.GenericStructureComplexC
INPUT_NUMBERS (pygamelib.assets.graphics.Models
                                                                                                  attribute), 339
              attribute), 156
                                                                                    \verb"inventory_space" (pygamelib.board_items.Immovable")
INPUT SYMBOLS
                               (pygamelib.assets.graphics.Models
                                                                                                  attribute), 355
             attribute), 156
                                                                                    inventory_space (pygamelib.board_items.Movable
insert_board() (pygamelib.engine.Game method),
                                                                                                  attribute), 364
{\tt INSERTION\_SYMBOL}\ (pygamelib. assets. graphics. \textit{MiscTechnikals} {\tt tory\_space}\ (pygamelib. board\_items. NPC\ atsumed to the property of the property 
                                                                                                  tribute), 372
              attribute), 73
```

inventory_space (pygamelib.board_items.Player attribute), 381	JAPANESE_CASTLE (pygamelib.assets.graphics.Models attribute), 156
inventory_space(pygamelib.board_items.Projectile attribute), 392	JAPANESE_CONGRATULATIONS_BUTTON (pygamelib.assets.graphics.Models attribute),
inventory_space (pygamelib.board_items.TextItem	156
attribute), 400	JAPANESE_DISCOUNT_BUTTON
inventory_space (pygamelib.board_items.Tile attribute), 410	(pygamelib.assets.graphics.Models attribute), 156
inventory_space (pygamelib.board_items.Treasure attribute), 419	JAPANESE_DOLLS (pygamelib.assets.graphics.Models attribute), 156
<pre>inventory_space (pygamelib.board_items.Wall at- tribute), 427</pre>	JAPANESE_FREE_OF_CHARGE_BUTTON (pygamelib.assets.graphics.Models attribute),
INVERSE_BULLET (pygamelib.assets.graphics.Geometric	cShapes 156
attribute), 55	JAPANESE_HERE_BUTTON
INVERSE_WHITE_CIRCLE	(pygamelib.assets.graphics.Models attribute),
(pygamelib.assets.graphics.GeometricShapes	156
attribute), 55	JAPANESE_MONTHLY_AMOUNT_BUTTON
item() (pygamelib.board_items.ActionableTile method), 219	(pygamelib.assets.graphics.Models attribute), 156
item() (pygamelib.board_items.BoardComplexItem	JAPANESE_NO_VACANCY_BUTTON
method), 228 item() (pygamelib.board_items.ComplexDoor	(pygamelib.assets.graphics.Models attribute), 156
method), 278	JAPANESE_NOT_FREE_OF_CHARGE_BUTTON
<pre>item() (pygamelib.board_items.ComplexNPC method),</pre>	(pygamelib.assets.graphics.Models attribute), 156
<pre>item() (pygamelib.board_items.ComplexPlayer</pre>	JAPANESE_OPEN_FOR_BUSINESS_BUTTON (pygamelib.assets.graphics.Models attribute),
<pre>item() (pygamelib.board_items.ComplexTreasure method), 306</pre>	156 JAPANESE_PASSING_GRADE_BUTTON
item() (pygamelib.board_items.ComplexWall method), 315	(pygamelib.assets.graphics.Models attribute), 156
item() (pygamelib.board_items.TextItem method), 401	JAPANESE_POST_OFFICE
<pre>item() (pygamelib.board_items.Tile method), 410 item() (pygamelib.engine.Board method), 438</pre>	(pygamelib.assets.graphics.Models attribute), 156
items (pygamelib.engine.Inventory attribute), 464	JAPANESE_PROHIBITED_BUTTON
items() (pygamelib.gfx.core.SpriteCollection method), 487	(pygamelib.assets.graphics.Models attribute), 156
<pre>items_name() (pygamelib.engine.Inventory method),</pre>	JAPANESE_RESERVED_BUTTON
465 items_per_page() (pygamelib.gfx.ui.GridSelector	(pygamelib.assets.graphics.Models attribute), 156
method), 523	JAPANESE_SECRET_BUTTON
J	(pygamelib.assets.graphics.Models attribute), 156
JACK_O_LANTERN (pygamelib.assets.graphics.Models	JAPANESE_SERVICE_CHARGE_BUTTON
attribute), 156	(pygamelib.assets.graphics.Models attribute), 156
JAPANESE_ACCEPTABLE_BUTTON	JAPANESE_SYMBOL_FOR_BEGINNER
(pygamelib.assets.graphics.Models attribute), 156	(pygamelib.assets.graphics.Models attribute), 156
JAPANESE_APPLICATION_BUTTON (magnetic assets araphics Models attribute)	JAPANESE_VACANCY_BUTTON
(pygamelib.assets.graphics.Models attribute), 156	(pygamelib.assets.graphics.Models attribute), 156
JAPANESE_BARGAIN_BUTTON (magnetic anaphics Models attribute)	JEANS (pygamelib.assets.graphics.Models attribute),
(pygamelib.assets.graphics.Models attribute), 156	156

JOKER (pygamelib.assets.graphics.Models attribute), LADY_BEETLE (pygamelib.assets.graphics.Models at-156 tribute), 157 JOYSTICK (pygamelib.assets.graphics.Models LAPTOP (pygamelib.assets.graphics.Models attribute), tribute), 156 157 LARGE BLUE DIAMOND K (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), LARGE CIRCLE (pygamelib.assets.graphics.GeometricShapes 156 attribute), 55 **KANGAROO** (pygamelib.assets.graphics.Models at-LARGE_ORANGE_DIAMOND tribute), 156 (pygamelib.assets.graphics.Models attribute), KEY (pygamelib.assets.graphics.Models attribute), 156 157 KEYBOARD (pygamelib.assets.graphics.MiscTechnicals LAST_QUARTER_MOON KEYBOARD (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), attribute), 156 LAST_QUARTER_MOON_FACE keys() (pygamelib.gfx.core.SpriteCollection method), (pygamelib.assets.graphics.Models attribute), 157 KICK_SCOOTER (pygamelib.assets.graphics.Models at-LAST_TRACK_BUTTON tribute), 156 (pygamelib.assets.graphics.Models attribute), KIMONO (pygamelib.assets.graphics.Models attribute), 157 157 LATIN_CROSS (pygamelib.assets.graphics.Models at-KISS (pygamelib.assets.graphics.Models attribute), 157 tribute), 157 KISS_MARK (pygamelib.assets.graphics.Models (pygamelib.board items.Actionable attribute), layer tribute), 157 KISSING_CAT (pygamelib.assets.graphics.Models at-211 layer (pygamelib.board_items.ActionableTile tribute), 157 tribute), 219 KISSING_FACE (pygamelib.assets.graphics.Models atlayer (pygamelib.board_items.BoardComplexItem attribute), 157 tribute), 228 KISSING_FACE_WITH_CLOSED_EYES (pygamelib.board_items.BoardItem attribute), (pygamelib.assets.graphics.Models attribute), layer 246 157 layer(pygamelib.board_items.BoardItemComplexComponent KISSING FACE WITH SMILING EYES attribute), 237 (pygamelib.assets.graphics.Models attribute), (pygamelib.board_items.BoardItemVoid layer tribute), 254 KITCHEN_KNIFE (pygamelib.assets.graphics.Models layer (pygamelib.board_items.Camera attribute), 262 attribute), 157 layer (pygamelib.board items.Character attribute), KITE (pygamelib.assets.graphics.Models attribute), 157 (pygamelib.assets.graphics.Models KIWI_FRUIT layer (pygamelib.board_items.ComplexDoor attribute), attribute), 157 KNOT (pygamelib.assets.graphics.Models attribute), 157 layer (pygamelib.board_items.ComplexNPC attribute), KOALA (pygamelib.assets.graphics.Models attribute), 157 (pygamelib.board_items.ComplexPlayer layer tribute), 297 (pygamelib.board_items.ComplexTreasure layer LAB_COAT (pygamelib.assets.graphics.Models attribute), 306 tribute), 157 layer (pygamelib.board_items.ComplexWall attribute), LABEL (pygamelib.assets.graphics.Models attribute), 315 layer (pygamelib.board_items.Door attribute), 324 label (pygamelib.gfx.ui.LineInputDialog attribute), 525 layer(pygamelib.board_items.GenericActionableStructure label (pygamelib.gfx.ui.ProgressDialog attribute), 543 attribute), 332 LACROSSE (pygamelib.assets.graphics.Models (pygamelib.board_items.GenericStructure layer tribute), 157

Index 643

LADDER (pygamelib.assets.graphics.Models attribute),

157

attribute), 348

layer (pygamelib.board items.GenericStructureComplexComponent

attribute), 340	(pygamelib.assets.graphics.BoxDrawings
<pre>layer (pygamelib.board_items.Immovable attribute),</pre>	attribute), 46
356	LEFT_HEAVY_AND_RIGHT_VERTICAL_LIGHT
layer (pygamelib.board_items.Movable attribute), 364	(pygamelib.assets.graphics.BoxDrawings
layer (pygamelib.board_items.NPC attribute), 372	attribute), 47
layer (pygamelib.board_items.Player attribute), 381	LEFT_LIGHT_AND_RIGHT_DOWN_HEAVY
layer (pygamelib.board_items.Projectile attribute), 392	(pygamelib.assets.graphics.BoxDrawings
layer (pygamelib.board_items.TextItem attribute), 401	attribute), 47
layer (pygamelib.board_items.Tile attribute), 410	LEFT_LIGHT_AND_RIGHT_UP_HEAVY
layer (pygamelib.board_items.Treasure attribute), 419	(pygamelib.assets.graphics.BoxDrawings
layer (pygamelib.board_items.Wall attribute), 427	attribute), 47
layers() (pygamelib.engine.Board method), 439	LEFT_LIGHT_AND_RIGHT_VERTICAL_HEAVY
LEAF_FLUTTERING_IN_WIND	(pygamelib.assets.graphics.BoxDrawings
(pygamelib.assets.graphics.Models attribute),	attribute), 47
157	LEFT_LUGGAGE (pygamelib.assets.graphics.Models at-
LEAFY_GREEN (pygamelib.assets.graphics.Models at-	tribute), 157
tribute), 157	LEFT_ONE_EIGHTH_BLOCK
LEDGER (pygamelib.assets.graphics.Models attribute),	(pygamelib.assets.graphics.Blocks attribute),
157	38
LEFT_ARROW (pygamelib.assets.graphics.Models	LEFT_ONE_QUARTER_BLOCK
attribute), 157	(pygamelib.assets.graphics.Blocks attribute),
LEFT_ARROW_CURVING_RIGHT	38
(pygamelib.assets.graphics.Models attribute),	LEFT_PARENTHESIS_EXTENSION
157	(pygamelib.assets.graphics.MiscTechnicals
LEFT_CEILING (pygamelib.assets.graphics.MiscTechnica	
attribute), 73	LEFT_PARENTHESIS_LOWER_HOOK
LEFT_CURLY_BRACKET_LOWER_HOOK	(pygamelib.assets.graphics.MiscTechnicals
(pygamelib.assets.graphics.MiscTechnicals	attribute), 74
attribute), 73	LEFT_PARENTHESIS_UPPER_HOOK
LEFT_CURLY_BRACKET_MIDDLE_PIECE	(pygamelib.assets.graphics.MiscTechnicals
(pygamelib.assets.graphics.MiscTechnicals	attribute), 74
attribute), 73	LEFT_POINTING_ANGLE_BRACKET
LEFT_CURLY_BRACKET_UPPER_HOOK	(pygamelib.assets.graphics.MiscTechnicals
(pygamelib.assets.graphics.MiscTechnicals	attribute), 74
attribute), 73	LEFT_RIGHT_ARROW (pygamelib.assets.graphics.Models
LEFT_DOWN_HEAVY_AND_RIGHT_UP_LIGHT	attribute), 157
(pygamelib.assets.graphics.BoxDrawings	LEFT_SEVEN_EIGHTHS_BLOCK
attribute), 46	(pygamelib.assets.graphics.Blocks attribute),
LEFT_FACING_FIST (pygamelib.assets.graphics.Models	
attribute), 157	LEFT_SPEECH_BUBBLE
LEFT_FIVE_EIGHTHS_BLOCK	(pygamelib.assets.graphics.Models attribute),
(pygamelib.assets.graphics.Blocks attribute),	157
38	LEFT_SQUARE_BRACKET_EXTENSION
LEFT_FLOOR (pygamelib.assets.graphics.MiscTechnicals	(pygamelib.assets.graphics.MiscTechnicals
attribute), 73	attribute), 74
LEFT_HALF_BLACK_CIRCLE	LEFT_SQUARE_BRACKET_LOWER_CORNER
(pygamelib.assets.graphics.GeometricShapes	(pygamelib.assets.graphics.MiscTechnicals
attribute), 55	attribute), 74
LEFT_HALF_BLOCK (pygamelib.assets.graphics.Blocks	LEFT_SQUARE_BRACKET_UPPER_CORNER
attribute), 38	(pygamelib.assets.graphics.MiscTechnicals
LEFT_HEAVY_AND_RIGHT_DOWN_LIGHT	attribute), 74
(pygamelib.assets.graphics.BoxDrawings	LEFT_THREE_EIGHTHS_BLOCK
attribute), 46	(pygamelib.assets.graphics.Blocks attribute),
LEFT HEAVY AND RIGHT UP LIGHT	38

LEFT_THREE_QUARTERS_BLOCK attribute), 47 (pygamelib.assets.graphics.Blocks attribute), LIGHT DOWN (pygamelib.assets.graphics.BoxDrawings attribute), 47 LEFT_UP_HEAVY_AND_RIGHT_DOWN_LIGHT LIGHT_DOWN_AND_HORIZONTAL (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 47 attribute), 47 LEFT VERTICAL BOX LINE LIGHT DOWN AND LEFT (pygamelib.assets.graphics.MiscTechnicals (pygamelib.assets.graphics.BoxDrawings attribute), 74 attribute), 47 LIGHT_DOWN_AND_RIGHT LEG (pygamelib.assets.graphics.Models attribute), 157 (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.Models attribute), 157 attribute), 47 length (pygamelib.base.Text attribute), 199 $\verb|LIGHT_HORIZONTAL| (pygamelib. assets. graphics. BoxDrawings$ length (pygamelib.gfx.core.Sprixel attribute), 503 attribute), 47 length (pygamelib.gfx.particles.ParticleSprixel LIGHT_LEFT (pygamelib.assets.graphics.BoxDrawings attribute), 580 attribute), 47 length() (pygamelib.base.Vector2D method), 204 LIGHT_LEFT_AND_HEAVY_RIGHT length() (pygamelib.gfx.ui.MenuBar method), 534 (pygamelib.assets.graphics.BoxDrawings LEO (pygamelib.assets.graphics.Models attribute), 157 attribute), 47 LEOPARD (pygamelib.assets.graphics.Models attribute), LIGHT QUADRUPLE DASH HORIZONTAL 158 (pygamelib.assets.graphics.BoxDrawings lerp() (pygamelib.base.Math static method), 192 attribute), 47 LEVEL_SLIDER (pygamelib.assets.graphics.Models at-LIGHT_QUADRUPLE_DASH_VERTICAL tribute), 158 (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.Models attribute), attribute), 47 LIBRA (pygamelib.assets.graphics.Models LIGHT RAIL LIGHT_ARC_DOWN_AND_LEFT attribute), 158 (pygamelib.assets.graphics.BoxDrawings LIGHT_RIGHT (pygamelib.assets.graphics.BoxDrawings attribute), 47 attribute), 47 LIGHT_ARC_DOWN_AND_RIGHT LIGHT_SHADE (pygamelib.assets.graphics.Blocks at-(pygamelib.assets.graphics.BoxDrawings tribute), 38 attribute), 47 LIGHT_SKIN_TONE (pygamelib.assets.graphics.Models LIGHT_ARC_UP_AND_LEFT attribute), 158 (pygamelib.assets.graphics.BoxDrawings LIGHT_TRIPLE_DASH_HORIZONTAL attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT_ARC_UP_AND_RIGHT attribute), 47 (pygamelib. assets. graphics. BoxDrawingsLIGHT TRIPLE DASH VERTICAL attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT BULB (pygamelib.assets.graphics.Models attribute), 47 attribute), 158 LIGHT_UP (pygamelib.assets.graphics.BoxDrawings at-LIGHT DIAGONAL CROSS tribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT UP AND HEAVY DOWN attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT_DIAGONAL_UPPER_LEFT_TO_LOWER_RIGHT attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT_UP_AND_HORIZONTAL tribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT_DIAGONAL_UPPER_RIGHT_TO_LOWER_LEFT attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT_UP_AND_LEFT attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT_DOUBLE_DASH_HORIZONTAL attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT_UP_AND_RIGHT attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT DOUBLE DASH VERTICAL attribute), 47 (pygamelib.assets.graphics.BoxDrawings LIGHT VERTICAL (pygamelib.assets.graphics.BoxDrawings

	attribute), 47		method), 271
LIGHT_	VERTICAL_AND_HORIZONTAL	load()	(pygamelib.board_items.ComplexDoor class
	(pygamelib.assets.graphics.BoxDrawings		method), 279
	attribute), 47	load()	
LIGHT_	VERTICAL_AND_LEFT		method), 288
	(pygamelib.assets.graphics.BoxDrawings	load()	
	attribute), 47		method), 297
LIGHT_	VERTICAL_AND_RIGHT	load()	(pygamelib.board_items.ComplexTreasure class
	(pygamelib.assets.graphics.BoxDrawings attribute), 47	l a a d ()	method), 306 (pygamelib.board_items.ComplexWall class
TinoTn	aputDialog (class in pygamelib.gfx.ui), 524	load()	method), 315
	ygamelib.assets.graphics.Models attribute), 158	load()	(pygamelib.board_items.Door class method),
	_PAPERCLIPS	1044()	324
DIIVICED,		load()	(pygamelib.board_items.GenericActionableStructure
	158	(/	class method), 332
LION (p	ygamelib.assets.graphics.Models attribute), 158	load()	(pygamelib.board_items.GenericStructure class
LIPSTI			method), 348
	tribute), 158	load()	$(pygame lib. board_items. Generic Structure Complex Component$
LITTER	_IN_BIN_SIGN		class method), 340
	(pygame lib. assets. graphics. Models attribute),	load()	(pygamelib.board_items.Immovable class
	158		method), 356
LIZARD	(pygamelib.assets.graphics.Models attribute),	load()	(pygamelib.board_items.Movable class
	158		method), 364
LLAMA	(pygamelib.assets.graphics.Models attribute),	load()	
1 a a d ()	158 (magnetic actuators Actuator method) 0	l a a d ()	(magnetic board items Player class method)
	(pygamelib.actuators.Actuator method), 9 (pygamelib.actuators.Behavioral method), 12	Ioad()	(pygamelib.board_items.Player class method), 381
load()		load()	(pygamelib.board_items.Projectile class
1044()	method), 15	1044()	method), 392
load()	(pygamelib.actuators.PathFinder class method),	load()	(pygamelib.board_items.TextItem class
(/	24	(,	method), 401
load()	(pygamelib.actuators.PatrolActuator class	load()	(pygamelib.board_items.Tile class method), 410
	method), 19	load()	(pygamelib.board_items.Treasure class
load()	(pygamelib.actuators.RandomActuator class		method), 419
	method), 29	load()	
load()	***		427
	class method), 33		(pygamelib.engine.Board class method), 439
	(pygamelib.base.Text class method), 200		(pygamelib.engine.Inventory class method), 465
	(pygamelib.base.Vector2D class method), 204	load()	(pygamelib.gfx.core.Animation class method),
load()	(pygamelib.board_items.Actionable class method), 211	load()	481 (pygamelib.gfx.core.Color class method), 509
load()			(pygamelib.gfx.core.Font method), 485
1044()	method), 219		(pygamelib.gfx.core.Sprite class method), 494
load()	(pygamelib.board_items.BoardComplexItem	load()	(pygamelib.gfx.core.SpriteCollection class
(/	class method), 228	(/	method), 488
load()	(pygamelib.board_items.BoardItem class	load()	(pygamelib.gfx.core.Sprixel class method), 503
	method), 246	load()	(pygamelib.gfx.particles.CircleEmitter class
load()	$(pygame lib. board_items. Board Item Complex Complex$	ponent	method), 548
	class method), 237	load()	(pygamelib.gfx.particles.ColorParticle class
load()	_		method), 553
	method), 254	load()	(pygamelib.gfx.particles.ColorPartitionParticle
load()	(pygamelib.board_items.Camera class method),	_	class method), 558
	262	load()	(pygamelib.gfx.particles.EmitterProperties
load()	(pygamelib.board_items.Character class		class method), 562

load() (pygamelib.gfx.particles.Particle class method), attribute), 55 LOWER HALF INVERSE WHITE CIRCLE load() (pygamelib.gfx.particles.ParticleEmitter class (pygamelib.assets.graphics.GeometricShapes attribute), 55 method), 566 load() (pygamelib.gfx.particles.ParticleSprixel class LOWER LEFT QUADRANT CIRCULAR ARC method), 580 (pygamelib.assets.graphics.GeometricShapes load() (pygamelib.gfx.particles.PartitionParticle class attribute), 55 LOWER LEFT TRIANGLE method), 587 load() (pygamelib.gfx.particles.RandomColorParticle (pygamelib.assets.graphics.GeometricShapes class method), 592 attribute), 55 load() (pygamelib.gfx.particles.RandomColorPartitionPaltionHeel_ONE_EIGHTH_BLOCK class method), 597 (pygamelib.assets.graphics.Blocks attribute), load_board() (pygamelib.engine.Game method), 453 load_config() (pygamelib.engine.Game method), LOWER_ONE_QUARTER_BLOCK 454 (pygamelib.assets.graphics.Blocks attribute), load_from_ansi_file() 38 (pygamelib.gfx.core.Sprite class method), LOWER_RIGHT_QUADRANT_CIRCULAR_ARC (pygamelib.assets.graphics.GeometricShapes load_json_file() (pygamelib.gfx.core.SpriteCollection attribute), 55 LOWER RIGHT TRIANGLE static method), 488 LOBSTER (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.GeometricShapes attribute), 55 LOCKED (pygamelib.assets.graphics.Models attribute), LOWER_SEVEN_EIGHTHS_BLOCK (pygamelib.assets.graphics.Blocks attribute). 158 38 LOCKED_WITH_KEY (pygamelib.assets.graphics.Models attribute), 158 LOWER THREE EIGHTHS BLOCK LOCKED_WITH_PEN (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Blocks attribute), attribute), 158 (pygamelib.assets.graphics.Models LOWER_THREE_QUARTERS_BLOCK LOCOMOTIVE (pygamelib.assets.graphics.Blocks attribute), attribute), 158 LOLLIPOP (pygamelib.assets.graphics.Models 38 tribute), 158 LOZENGE (pygamelib.assets.graphics.GeometricShapes LONG_DRUM (pygamelib.assets.graphics.Models attribute), 55 tribute), 158 LUGGAGE (pygamelib.assets.graphics.Models attribute), LOTION_BOTTLE (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), attribute), 158 LUNGS LOUDLY CRYING FACE 158 (pygamelib.assets.graphics.Models attribute), LYING_FACE (pygamelib.assets.graphics.Models 158 attribute), 158 LOUDSPEAKER (pygamelib.assets.graphics.Models at-Μ tribute), 158 LOVE_HOTEL (pygamelib.assets.graphics.Models MAGE (pygamelib.assets.graphics.Models attribute), 158 attribute), 158 magenta() (pygamelib.base.Text static method), 200 LOVE_LETTER (pygamelib.assets.graphics.Models atmagenta_bright() (pygamelib.base.Text static tribute), 158 method), 200 LOVE_YOU_GESTURE (pygamelib.assets.graphics.Models magenta_dim() (pygamelib.base.Text static method), attribute), 158 200 LOWER_FIVE_EIGHTHS_BLOCK magenta_rect() (pygamelib.gfx.core.Sprixel class (pygamelib.assets.graphics.Blocks attribute), method), 503 magenta_rect() (pygamelib.gfx.particles.ParticleSprixel LOWER_HALF_BLOCK (pygamelib.assets.graphics.Blocks class method), 580 attribute), 38 magenta_square() (pygamelib.gfx.core.Sprixel class LOWER HALF CIRCLE method), 503 (pygamelib.assets.graphics.GeometricShapes

magenta_square() (pygamelib.gfx.particles.ParticleSpMEdHANICAL_LEG (pygamelib.assets.graphics.Models attribute), 159 class method), 580 MAGIC WAND (pygamelib.assets.graphics.Models MEDICAL SYMBOL (pygamelib.assets.graphics.Models attribute), 158 attribute), 159 MAGNET (pygamelib.assets.graphics.Models attribute), MEDIUM DARK SKIN TONE (pygamelib.assets.graphics.Models attribute), MAGNIFYING GLASS TILTED LEFT MEDIUM LIGHT SKIN TONE (pygamelib.assets.graphics.Models attribute), 158 (pygamelib.assets.graphics.Models attribute), 159 MAGNIFYING_GLASS_TILTED_RIGHT MEDIUM_SHADE (pygamelib.assets.graphics.Blocks at-(pygamelib.assets.graphics.Models attribute), 158 tribute), 38 MAHJONG_RED_DRAGON MEDIUM_SKIN_TONE (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), attribute), 159 (pygamelib.assets.graphics.Models at-158 MEGAPHONE MALE_SIGN (pygamelib.assets.graphics.Models tribute), 159 (pygamelib.assets.graphics.Models attribute), tribute), 158 MELON MAMMOTH (pygamelib.assets.graphics.Models attribute), 159 MEMO (pygamelib.assets.graphics.Models attribute), 159 MAN (pygamelib.assets.graphics.Models attribute), 159 MEN HOLDING HANDS MAN BEARD (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), tribute), 159 MAN_DANCING (pygamelib.assets.graphics.Models at-MENORAH (pygamelib.assets.graphics.Models attribute), 159 *tribute*), 159 MANGO (pygamelib.assets.graphics.Models attribute), MENS ROOM (pygamelib.assets.graphics.Models tribute), 159 (pygamelib.assets.graphics.Models Menu (class in pygamelib.gfx.ui), 526 MANS_SHOE menu_width() (pygamelib.gfx.ui.Menu method), 528 *tribute*), 159 MANTELPIECE_CLOCK MenuAction (class in pygamelib.gfx.ui), 529 (pygamelib.assets.graphics.Models attribute), MenuBar (class in pygamelib.gfx.ui), 532 159 MERPERSON (pygamelib.assets.graphics.Models at-MANUAL_WHEELCHAIR tribute), 159 (pygamelib.assets.graphics.Models attribute), MessageDialog (class in pygamelib.gfx.ui), 534 METRICAL_BREVE (pygamelib.assets.graphics.MiscTechnicals MAP_OF_JAPAN (pygamelib.assets.graphics.Models atattribute), 74 tribute), 159 METRICAL_LONG_OVER_SHORT MAPLE LEAF (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.MiscTechnicals attribute), 159 attribute), 74 MARTIAL_ARTS_UNIFORM METRICAL_LONG_OVER_TWO_SHORTS (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.MiscTechnicals attribute), 74 MATE (pygamelib.assets.graphics.Models attribute), 159 METRICAL PENTASEME Math (class in pygamelib.base), 190 (pygamelib.assets.graphics.MiscTechnicals max_height (pygamelib.gfx.ui.GridSelector attribute), attribute), 74 METRICAL_SHORT_OVER_LONG 523 max_width (pygamelib.gfx.ui.GridSelector attribute), (pygamelib.assets.graphics.MiscTechnicals attribute), 74 maximum (pygamelib.gfx.ui.ProgressBar attribute), 541 METRICAL_TETRASEME maximum (pygamelib.gfx.ui.ProgressDialog attribute), (pygamelib.assets.graphics.MiscTechnicals attribute), 74 MEAT_ON_BONE (pygamelib.assets.graphics.Models at-METRICAL_TRISEME (pygamelib.assets.graphics.MiscTechnicals *tribute*), 159 attribute), 74 MECHANICAL_ARM (pygamelib.assets.graphics.Models METRICAL TWO SHORTS JOINED attribute), 159 (pygamelib.assets.graphics.MiscTechnicals

attribute), 74	288
METRICAL_TWO_SHORTS_OVER_LONG	model (pygamelib.board_items.ComplexPlayer at-
(pygame lib. assets. graphics. Misc Technicals	tribute), 297
attribute), 74	model (pygamelib.board_items.ComplexTreasure
METRO (pygamelib.assets.graphics.Models attribute),	attribute), 306
159 MICROPE (magnetih assets graphics Models attribute)	model (pygamelib.board_items.ComplexWall attribute), 315
MICROBE (pygamelib.assets.graphics.Models attribute), 159	model (pygamelib.board_items.Door attribute), 324
MICROPHONE (pygamelib.assets.graphics.Models	mode1 (pygamelib.board_items.GenericActionableStructure
attribute), 159	attribute), 332
MICROSCOPE (pygamelib.assets.graphics.Models	model (pygamelib.board_items.GenericStructure
attribute), 159	attribute), 348
MIDDLE_FINGER (pygamelib.assets.graphics.Models	$\verb model (pygamelib.board_items. Generic Structure Complex Component) $
attribute), 159	attribute), 340
MILITARY_HELMET (pygamelib.assets.graphics.Models attribute), 159	model (pygamelib.board_items.Immovable attribute), 356
MILITARY_MEDAL (pygamelib.assets.graphics.Models	model (pygamelib.board_items.Movable attribute), 364
attribute), 159	<pre>model (pygamelib.board_items.NPC attribute), 373</pre>
MILKY_WAY (pygamelib.assets.graphics.Models at-	model (pygamelib.board_items.Player attribute), 381
tribute), 159	model (pygamelib.board_items.Projectile attribute), 392
MINIBUS (pygamelib.assets.graphics.Models attribute),	model (pygamelib.board_items.TextItem attribute), 401
159 MINUS (meanwhib agasta quanhias Madela, attributa)	model (pygamelib.board_items.Tile attribute), 410
MINUS (pygamelib.assets.graphics.Models attribute), 159	model (pygamelib.board_items.Treasure attribute), 419 model (pygamelib.board_items.Wall attribute), 427
MIRROR (pygamelib.assets.graphics.Models attribute),	mode1 (pygametib.boara_tiems. wait dirtibute), 427 mode1 (pygamelib.gfx.core.Sprixel attribute), 504
160	model (pygametis.gfx.core.sprixet utirioute), 304 model (pygametis.gfx.particles.ParticleSprixel at-
MiscTechnicals (class in	tribute), 581
pygamelib.assets.graphics), 57	Models (class in pygamelib.assets.graphics), 76
MOAI (pygamelib.assets.graphics.Models attribute), 160	modulate() (pygamelib.gfx.core.Sprite method), 494
MOBILE_PHONE (pygamelib.assets.graphics.Models at-	MONEY_BAG (pygamelib.assets.graphics.Models at-
tribute), 160	tribute), 160
	s MONEY_MOUTH_FACE (pygamelib.assets.graphics.Models
attribute), 160	attribute), 160
MOBILE_PHONE_WITH_ARROW (pygamelib.assets.graphics.Models attribute),	MONEY_WITH_WINGS (pygamelib.assets.graphics.Models attribute), 160
(pygameno.assers.graphics.moders announc),	MONKEY (pygamelib.assets.graphics.Models attribute),
model (pygamelib.board_items.Actionable attribute),	160
211	MONKEY_FACE (pygamelib.assets.graphics.Models at-
model (pygamelib.board_items.ActionableTile at-	tribute), 160
tribute), 219	MONORAIL (pygamelib.assets.graphics.Models at-
<pre>model (pygamelib.board_items.BoardComplexItem at-</pre>	tribute), 160
tribute), 229	monospace (pygamelib.gfx.core.Font attribute), 485
model (pygamelib.board_items.BoardItem attribute),	MONOSTABLE_SYMBOL
246	(pygamelib.assets.graphics.MiscTechnicals
<pre>model (pygamelib.board_items.BoardItemComplexCompo attribute), 237</pre>	onent attribute), 74 MOON_CAKE (pygamelib.assets.graphics.Models at-
model (pygamelib.board_items.BoardItemVoid at-	tribute), 160
tribute), 254	MOON_VIEWING_CEREMONY
model (pygamelib.board_items.Camera attribute), 262	(pygamelib.assets.graphics.Models attribute),
model (pygamelib.board_items.Character attribute),	160
271	MOSQUE (pygamelib.assets.graphics.Models attribute),
$\verb model (pygame lib.board_items. Complex Door\ attribute),$	160
279	MOSQUITO (pygamelib.assets.graphics.Models at-
<pre>model (pygamelib.board_items.ComplexNPC attribute),</pre>	tribute), 160

(pygamelib.assets.graphics.Models MOTOR BOAT attribute), 160 (pygamelib.assets.graphics.Models MOTOR SCOOTER attribute), 160 MOTORCYCLE (pygamelib.assets.graphics.Models attribute), 160 MOTORIZED WHEELCHAIR (pygamelib.assets.graphics.Models attribute), 160 MOTORWAY (pygamelib.assets.graphics.Models attribute), 160 (pygamelib.assets.graphics.Models MOUNT_FUJI attribute), 160 (pygamelib.assets.graphics.Models MOUNTAIN at*tribute*), 160 MOUNTAIN_CABLEWAY (pygamelib.assets.graphics.Models attribute), ${\tt MOUNTAIN_RAILWAY} \ (pygamelib. assets. graphics. Models\ {\tt neighbors}\ ()\ \ (pygamelib. engine. Board\ method),\ 440$ attribute), 160 MOUSE (pygamelib.assets.graphics.Models attribute), MOUSE_FACE (pygamelib.assets.graphics.Models attribute), 160 MOUSE TRAP (pygamelib.assets.graphics.Models attribute), 160 (pygamelib.assets.graphics.Models attribute), MOUTH Movable (class in pygamelib.board_items), 359 move () (pygamelib.engine.Board method), 439 move_player() (pygamelib.engine.Game method), 454 (pygamelib.actuators.RandomActuator moveset tribute), 29 MOVIE_CAMERA (pygamelib.assets.graphics.Models attribute), 160 MRS CLAUS (pygamelib.assets.graphics.Models *tribute*), 160 MultiLineInputDialog (class in pygamelib.gfx.ui), 537 (pygamelib.assets.graphics.Models MULTIPLY tribute), 160 MUSHROOM (pygamelib.assets.graphics.Models tribute), 160 ${\tt MUSICAL_KEYBOARD} \ (pygamelib. assets. graphics. Models \ {\tt next_move} \ ()$ attribute), 160 MUSICAL_NOTE (pygamelib.assets.graphics.Models at*tribute*), 160 MUSICAL_NOTES (pygamelib.assets.graphics.Models attribute), 160 MUSICAL_SCORE (pygamelib.assets.graphics.Models

attribute), 160

attribute), 161

MUTED_SPEAKER (pygamelib.assets.graphics.Models

Ν

- NAIL_POLISH (pygamelib.assets.graphics.Models attribute), 161
- name (pygamelib.gfx.core.Font attribute), 485
- NAME BADGE (pygamelib.assets.graphics.Models attribute), 161
- NATIONAL_PARK (pygamelib.assets.graphics.Models attribute), 161
- NAUSEATED_FACE (pygamelib.assets.graphics.Models attribute), 161
- NAZAR_AMULET (pygamelib.assets.graphics.Models attribute), 161
- nb_pages() (pygamelib.gfx.ui.GridSelector method),
- NECKTIE (pygamelib.assets.graphics.Models attribute),
- need_rendering (pygamelib.engine.Screen tribute), 476
- neighbors () (pygamelib.engine.Game method), 454
- NERD_FACE (pygamelib.assets.graphics.Models attribute), 161
- NESTING DOLLS (pygamelib.assets.graphics.Models attribute), 161
- NEUTRAL_FACE (pygamelib.assets.graphics.Models attribute), 161
- NEW BUTTON (pygamelib.assets.graphics.Models attribute), 161
- NEW_MOON (pygamelib.assets.graphics.Models tribute), 161
- NEW_MOON_FACE (pygamelib.assets.graphics.Models attribute), 161
- (pygamelib.assets.graphics.Models NEWSPAPER tribute), 161
- next_action() (pygamelib.actuators.Behavioral method), 12
- (pygamelib.actuators.PathFinder next_action() method), 25
- (pygamelib.gfx.core.Animation next_frame() method), 481
- next_move() (pygamelib.actuators.Actuator method), 9
- next_move() (pygamelib.actuators.Behavioral method), 12
- (pygamelib.actuators.PathActuator method), 15
- (pygamelib.actuators.PathFinder next_move() method), 25
- (pygamelib.actuators.PatrolActuator next_move() method), 19
- next_move() (pygamelib.actuators.RandomActuator method), 29
- next_move() (pygamelib.actuators.UnidirectionalActuator method), 33

NEXT_PAGE (pygamelib.assets.graphics.MiscTechnicals attribute), 74	notify() (pygamelib.actuators.UnidirectionalActuator method), 33
NEXT_TRACK_BUTTON	notify() (pygamelib.base.PglBaseObject method),
(pygamelib.assets.graphics.Models attribute),	194
161	notify() (pygamelib.base.Text method), 200
<pre>next_waypoint() (pygamelib.actuators.PathFinder</pre>	
NG_BUTTON (pygamelib.assets.graphics.Models attribute), 161	
NIGHT_WITH_STARS (pygamelib.assets.graphics.Model.	
attribute), 161	method), 229
NINE_OCLOCK (pygamelib.assets.graphics.Models attribute), 161	notify() (pygamelib.board_items.BoardItem method), 247
NINE_THIRTY (pygamelib.assets.graphics.Models attribute), 161	notify() (pygamelib.board_items.BoardItemComplexComponent method), 237
NINJA (pygamelib.assets.graphics.Models attribute), 161	notify() (pygamelib.board_items.BoardItemVoid method), 254
NO_BICYCLES (pygamelib.assets.graphics.Models attribute), 161	
NO_ENTRY (pygamelib.assets.graphics.Models at-	notify() (pygamelib.board_items.Character method),
tribute), 161 NO_LITTERING (pygamelib.assets.graphics.Models at-	notify() (pygamelib.board_items.ComplexDoor
tribute), 161	method), 279
NO_MOBILE_PHONES (pygamelib.assets.graphics.Model.	
attribute), 161	method), 288
NO_ONE_UNDER_EIGHTEEN (pygamelib.assets.graphics.Models attribute),	notify() (pygamelib.board_items.ComplexPlayer method), 297
(pygameno.assers.graphics.moders announe), 161	notify() (pygamelib.board_items.ComplexTreasure
NO_PEDESTRIANS (pygamelib.assets.graphics.Models	method), 306
attribute), 161	notify() (pygamelib.board_items.ComplexWall
NO_SMOKING (pygamelib.assets.graphics.Models	method), 315
attribute), 161	notify() (pygamelib.board_items.Door method), 324
NON_POTABLE_WATER	notify() (pygamelib.board_items.GenericActionableStructure
(pygamelib.assets.graphics.Models attribute),	method), 332
161	notify() (pygamelib.board_items.GenericStructure
NOSE (pygamelib.assets.graphics.Models attribute), 161	method), 348
	memoa), 546 niwałsify () (pygamelib.board_items.GenericStructureComplexComponent
attribute), 74	method), 340
	notify() (pygamelib.board_items.Immovable
<i>tribute</i>), 161	method), 356
NOTEBOOK_WITH_DECORATIVE_COVER	notify() (pygamelib.board_items.Movable method),
(pygamelib.assets.graphics.Models attribute),	364
161	notify() (pygamelib.board_items.NPC method), 373
notify() (pygamelib.actuators.Actuator method), 9	notify() (pygamelib.board_items.Player method),
notify() (pygamelib.actuators.Behavioral method),	381
12	notify() (pygamelib.board_items.Projectile method),
<pre>notify() (pygamelib.actuators.PathActuator method),</pre>	392
16	notify() (pygamelib.board_items.TextItem method),
<pre>notify() (pygamelib.actuators.PathFinder method),</pre>	401
25	notify() (pygamelib.board_items.Tile method), 410
notify() (pygamelib.actuators.PatrolActuator	notify() (pygamelib.board_items.Treasure method),
method), 19	419
notify() (pygamelib.actuators.RandomActuator	notify() (pygamelib.board_items.Wall method), 427
method), 30	notify() (pygamelib.engine.Board method), 440

notify () (pygamelib.agic.enc.Screen method), 476 notify () (pygamelib.agic.core.Sprite method), 504 notify () (pygamelib.gfc.core.Sprite method), 504 notify () (pygamelib.gfc.core.Sprite method), 504 notify () (pygamelib.gfc.core.Sprite method), 504 notify () (pygamelib.gfc.particles.CircleEmitter method), 553 notify () (pygamelib.gfc.particles.ColorParticle method), 550 notify () (pygamelib.gfc.particles.ColorParticle method), 550 notify () (pygamelib.gfc.particles.Particle method), 550 notify () (pygamelib.gfc.particles.Particle method), 573 notify () (pygamelib.gfc.particles.Particle method), 573 notify () (pygamelib.gfc.particles.ParticleSprixel method), 581 notify () (pygamelib.gfc.particles.ParticleSprixel method), 581 notify () (pygamelib.gfc.particles.ParticleSprixel method), 581 notify () (pygamelib.gfc.particles.ParticleSprixel method), 587 notify () (pygamelib.gfc.particles.ParticleSprixel method), 587 notify () (pygamelib.gfc.particles.ParticleSprixel method), 580 notify () (pygamelib.gfc.particles.ParticleSprixel method), 580 notify () (pygamelib.gfc.particles.ColorParticle method), 580 notify () (pygamelib.gfc.particles.ColorParticle method), 580 notify () (pygamelib.gfc.particles.ColorParticle method), 580 notify () (pygamelib.gfc.particles.ParticleSprixel method), 581 notify () (pygamelib.gfc.particles.ParticleSprixel method), 580 notify () (pygamelib.gfc.particles.P	notify() (pygamelib.engine.Game method), 455	OLDER_PERSON (pygamelib.assets.graphics.Models at-
notify() (pygamelib gfx.core. Sprite method), 504 notify() (pygamelib gfx.core. Sprite method), 504 notify() (pygamelib gfx.core. Sprixed method), 504 notify() (pygamelib gfx.core. Sprixed method), 504 notify() (pygamelib gfx.particles. Circle mitter method), 559 notify() (pygamelib gfx.particles. ColorParticle method), 559 notify() (pygamelib gfx.particles. Particle method), 550 notify() (pygamelib gfx.particles. Particle method), 557 notify() (pygamelib gfx.particles. Particle method), 566 notify() (pygamelib gfx.particles. Particle method), 567 notify() (pygamelib gfx.particles. Particle method), 587 notify() (pygamelib gfx.particles. PartitionParticle method), 589 notify() (pygamelib gfx.particles. PartitionParticle method), 587 notify() (pygamelib gfx.particles. PartitionParticle method), 581 notify() (pygamelib gfx.particles. PartitionParticle method), 581 notify() (
notify() (pygamelib.gfx.core.Sprite method), 594 notify() (pygamelib.gfx.core.Sprite method), 504 notify() (pygamelib.gfx.particles.CircleEmitter method), 539 notify() (pygamelib.gfx.particles.ColorParticle method), 553 notify() (pygamelib.gfx.particles.ColorParticle method), 553 notify() (pygamelib.gfx.particles.Particle method), 557 notify() (pygamelib.gfx.particles.Particle method), 566 notify() (pygamelib.gfx.particles.Particle method), 567 notify() (pygamelib.gfx.particles.Particle method), 566 notify() (pygamelib.gfx.particles.Particle method), 567 notify() (pygamelib.gfx.particles.ParticleTemitter method), 568 notify() (pygamelib.gfx.particles.ParticleTemitter method), 566 notify() (pygamelib.gfx.particles.ParticleTemitter method), 567 notify() (pygamelib.gfx.particleServalleTemitter method), 567 notify() (pygamelib.gfx.particleTemitter method), 567 notify() (pygamelib.gfx.particleTemitter method), 567 notify() (pygamelib.gfx.particleTem		
notify () (pygamelib.gk.particles.CircleEmitre method), 549 notify () (pygamelib.gk.particles.ColorParticle method), 553 notify () (pygamelib.gk.particles.ColorParticle method), 550 notify () (pygamelib.gk.particles.Particle method), 560 notify () (pygamelib.gk.particles.ParticleEmitre method), 581 notify () (pygamelib.gk.particles.Particle method), 581 notify () (pygamelib.gk.particles.Particle method), 587 notify () (pygamelib.gk.particles.Particle method), 592 notify () (pygamelib.gk.particles.Particle method), 597 notify () (pygamelib.gk.particles.Particle method), 592 notify () (pygamelib.assets.graphics.Models attribute), 162 O_EUTION_BOUT (pygamelib.assets.graphics.Models attribute), 162 O_EUTION_BOUT (pygamelib.assets.graphics.Models attr		
notify () (pygamelib gfx.particles. ColorParticle method), 553 notify () (pygamelib gfx.particles. ColorParticle method), 553 notify () (pygamelib gfx.particles. Particle method), 550 notify () (pygamelib gfx.particles. Particle method), 550 notify () (pygamelib gfx.particles. Particle method), 550 notify () (pygamelib gfx.particles. Particle method), 560 notify () (pygamelib gfx.particles. Particle method), 581 notify () (pygamelib gfx.particles. Particle method), 581 notify () (pygamelib gfx.particles. Particle method), 587 notify () (pygamelib gfx.particles. Partition Particle method), 587 notify () (pygamelib assets. graphics. Models attribute), 162 ONCOMING_EUST (pygamelib.assets. graphics. Models attribute), 162 ONCOMING_EUST (pygamelib.asset		
method), 549 notify () (pygamelib gfx.particles.ColorParticle method), 553 notify () (pygamelib gfx.particles.Particle method), 550 notify () (pygamelib gfx.particles.Particle mitter method), 560 notify () (pygamelib gfx.particles.Particle Emitter method), 581 notify () (pygamelib gfx.particles.Particles.Particle Emitter method), 581 notify () (pygamelib gfx.particles.Parti	- 170	
notify() (pygamelib.gfx.particles.ColorParticle method), 553 notify() (pygamelib.gfx.particles.Particle method), 550 notify() (pygamelib.gfx.particles.Particle method), 550 notify() (pygamelib.gfx.particles.Particle method), 560 notify() (pygamelib.gfx.particles.Particle method), 581 notify() (pygamelib.gfx.particles.Particles.Particles.particl		
method), 553 notify () (pygamelib.gfx.particles.ColorPartitionParticleONCOMING_BUS (pygamelib.assets.graphics.Models attribute), 162 notify () (pygamelib.gfx.particles.Particle method), 566 notify () (pygamelib.gfx.particles.ParticleSprixel method), 581 notify () (pygamelib.gfx.particles.PartitionParticle method), 582 notify () (pygamelib.gfx.particles.RandomColorParticle method), 592 notify () (pygamelib.gfx.particles.RandomColorParticle method), 597 notify () (pygamelib.gfx.particles.ParticleSprixel method), 590 notify () (pygamelib.gfx.particles.ParticleSprixel method), 590 notify () (pygamelib.gfx.particles.ParticleSprixel method), 591 notify () (pygamelib.gfx.particles.ParticleSprixel method), 592 notify () (pygamelib.gfx.particles.ParticleSprixel particleSprixel		
method), 559 notify () (pygamelib.gfx.particles.Particle method), 560 notify () (pygamelib.gfx.particles.Particles.	method), 553	162
DOLIFY () (pygamelib.gfx.particles.Particle method), 566 DOLIFY () (pygamelib.gfx.particles.ParticleSprixel method), 581 DOLIFY () (pygamelib.gfx.particles.PartitionParticle method), 581 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 587 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 587 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 592 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 592 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 597 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 587 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 592 DOLIFY () (pygamelib.gfx.particles.RandomColorParticle method), 592 DOLIFY () (pygamelib.assets.graphics.Models attribute), 162 DOLIFY () (pygamelib.assets.graphics.Models a		
attribute), 162 notify () (pygamelib, gfx.particles.ParticleEmiter method), 586 notify () (pygamelib, gfx.particles.ParticleSprixel method), 581 notify () (pygamelib, gfx.particles.ParticleSprixel method), 587 notify () (pygamelib, gfx.particles.PartitionParticle method), 587 notify () (pygamelib, gfx.particles.RandomColorParticle method), 597 notify () (pygamelib.assets.graphics.Models attribute), 162 ONE_OLOCK (pygamelib.assets.graphics.Models at		
notify() (pygamelib.gfx.particles.ParticleEmitter method), 561 notify() (pygamelib.gfx.particles.ParticleSprixel method), 581 notify() (pygamelib.gfx.particles.PartitionParticle method), 587 notify() (pygamelib.gfx.particles.RandomColorParticle method), 593 notify() (pygamelib.gfx.particles.RandomColorParticle method), 597 notify() (pygamelib.gfx.particles.RandomColorParticle method), 597 notify() (pygamelib.gfx.particles.RandomColorParticle method), 597 NOTIGINO (pygamelib.gfx.particles.RandomColorParticle method), 597 NOTIGINO (pygamelib.board_items), 367 NOTIGINO (pygamelib.board_items), 367 NOTIGINO (pygamelib.assets.graphics.Models attribute), 161 OO OBUTTON_BLOOD_TYPE (pygamelib.assets.graphics.Models attribute), 163 OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.Models attribute), 163 OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 162 OREAND (pygamelib.assets.graphics.Models attribute), 162 OREAND (pygamelib.assets.graphics.Models attribute), 162 OREAND (pygamelib.assets.graphics.Models attribute), 162 OREAND (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.		
method), 566 notify() (pygamelib.gfx.particles.ParticleSprixel method), 581 notify() (pygamelib.gfx.particles.Part		
Dotify () (pygamelib.gfx.particles.ParticleSprixel method), \$587 ONCOMING_TAXI (pygamelib.assets.graphics.Models attribute), 162 ONCOMING_TAXI (pygamelib.assets.graphics.Models attribute), 162		
method), 581 notify () (pygamelib.gfx.particles.PartitionParticle method), 587 notify () (pygamelib.gfx.particles.RandomColorParticle method), 592 notify () (pygamelib.gfx.particles.RandomColorParticle method), 592 notify () (pygamelib.gfx.particles.RandomColorParticle method), 597 notify () (pygamelib.gfx.particles.RandomColorParticle method), 592 notify () (pygamelib.gfx.particles.RandomColorParticle method), 592 notify () (pygamelib.gfx.particles.RandomColorParticle method), 592 notify () (pygamelib.gfx.particles.RandomColorParticle attribute), 161 NDEOL (pygamelib.assets.graphics.Models attribute), 163 OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.Models attribute), 164 OESERVER_EYE_SYMBOL (pygamelib.assets.graphics.Models attribute), 161 OESERVER_EYE_SYMBOL (pygamelib.assets.graphics.Models attribute), 162 OED_MAN (pygamelib.assets.graphics.Models attribute), 162 OED_MAN (pygamelib.assets.graphics.Models attribute), 162 OED_MOMAN (pygamelib.assets.graphics.Models attribute), 162 OED_MOMAN (pygamelib.assets.graphics.Models attribute), 162 OED_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OED_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OED_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OED_WOMAN (pyga		
Dotify () (pygamelib gfx.particles.PartitionParticle method), 587		ONCOMING_TAXI (pygamelib.assets.graphics.Models
notify() (pygamelib.gfx.particles.RandomColorParticle method), 592 Notify() (pygamelib.gfx.particles.RandomColorPartitionParticle (pygamelib.assets.graphics.Models attribute), 162 NPC (class in pygamelib.board_items), 367 NUT_AND_BOLT (pygamelib.assets.graphics.Models attribute), 161 O OBUTTON_BLOOD_TYPE	notify() (pygamelib.gfx.particles.PartitionParticle	
method), 592 NPC (class in pygamelib.board_items), 367 NUT_AND_BOLT (pygamelib.assets.graphics.Models attribute), 161 OBUTTON_BLOOD_TYPE	method), 587	ONE_OCLOCK (pygamelib.assets.graphics.Models
notify () (pygamelib.gfx.particles.RandomColorPartitionParticle (pygamelib.assets.graphics.Models attribute), 597 NUT_AND_BOLT (pygamelib.assets.graphics.Models attribute), 161 O_BUTTON_BLOOD_TYPE	notify()(pygamelib.gfx.particles.RandomColorParticle	e attribute), 162
method), 597 NPC (class in pygamelib.board_items), 367 NUT_AND_BOLT (pygamelib.assets.graphics.Models attribute), 161 O O OBUTTON_BLOOD_TYPE	method), 592	ONE_PIECE_SWIMSUIT
NPC (class in pygamelib.board_items), 367 NUT_AND_BOLT (pygamelib.assets.graphics.Models attribute), 161 O O_BUTTON_BLOOD_TYPE	notify()(pygamelib.gfx.particles.RandomColorPartition	onParticle (pygamelib.assets.graphics.Models attribute),
NUT_AND_BOLT (pygamelib.assets.graphics.Models attribute), 161 OONION 162 OPEN_BOOK (pygamelib.assets.graphics.Models attribute), 162 OPEN_CIRCUIT_OUTPUT_H_TYPE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 161 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.Models attribute), 161 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.Models attribute), 161 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.Models attribute), 161 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.Models attribute), 162 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets	method), 597	162
ONION (pygamelib.assets.graphics.Models attribute), 162 Open_BUTTON_BLOOD_TYPE (pygamelib.assets.graphics.Models attribute), 163 OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.Models attribute), 164 OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OFEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute)		
O_BUTTON_BLOOD_TYPE (pygamelib.assets.graphics.Models attribute), 163 OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OFEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamel		
OPEN_BOOK (pygamelib.assets.graphics.Models attribute), 162 OPEN_BOOK (pygamelib.assets.graphics.Models attribute), 163 OPEN_BOOK (pygamelib.assets.graphics.Models attribute), 164 OPEN_CIRCUIT_OUTPUT_H_TYPE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HAND (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute)	tribute), 161	
O_BUTTON_BLOOD_TYPE (pygamelib.assets.graphics.Models attribute), 163 OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.as	0	
(pygamelib.assets.graphics.Models attribute), 163 OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 74 OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OPEN_CIRCU	O BUTTON BUOOD TYPE	
163		
OBSERVER_EYE_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 74 OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 162 OGRE (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74		
OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL attribute), 74 OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 163 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 164 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 165 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 164 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 165 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 164 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 165 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 166 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 167 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 168 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 169 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 169 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 160 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 160 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 160 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 160 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 161 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 161 OPT		
CTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTICN_KEY (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 162		OPEN_CIRCUIT_OUTPUT_L_TYPE_SYMBOL
OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 34 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 34 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 162		(lib MiTb-il-
ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute) attribute), 74		(pygametib.assets.grapnics.Misc1ecnnicals
ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute)		
OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute).	OCTOPUS (pygamelib.assets.graphics.Models attribute),	attribute), 74
attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute).	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162
OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute).	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162
tribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162
OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 (pygamelib.assets.graphics.Models attribute), 162 (pygamelib.assets.graphics.Models attribute), 162 (pygamelib.assets.graphics.Models attribute), 162 (OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 (pygamelib.assets.graphics.Models attribute), 162 (pygamelib.assets.graphics.Models attribute), 162 (OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 (OPTION_KEY (pygamelib.assets.graphics.Models attribute), 162 (OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 74 (ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 (ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 162 (OPTION_KEY (pygamelib.assets.graphics.Models attribute), 74 (ORANGE_BOOK (pygamelib.assets.gra	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG
tribute), 162 (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models at-	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models at-	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute),
OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute) attribute), 162	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162
OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute)	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models at-	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG
162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models at-	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute),
162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 tribute), 162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models attribute), 162	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute),	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162
OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162 tribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models at-	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162
162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models at-	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute),	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162
tribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models at-	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162
tribute), 162 ORANGE_BOOK (pygamelib.assets.graphics.Models at-	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals
tribute), 162	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 74
	OCTOPUS (pygamelib.assets.graphics.Models attribute), 161 ODEN (pygamelib.assets.graphics.Models attribute), 161 OFFICE_BUILDING (pygamelib.assets.graphics.Models attribute), 161 OGRE (pygamelib.assets.graphics.Models attribute), 161 OIL_DRUM (pygamelib.assets.graphics.Models attribute), 162 OK_BUTTON (pygamelib.assets.graphics.Models attribute), 162 OK_HAND (pygamelib.assets.graphics.Models attribute), 162 OLD_KEY (pygamelib.assets.graphics.Models attribute), 162 OLD_MAN (pygamelib.assets.graphics.Models attribute), 162 OLD_WOMAN (pygamelib.assets.graphics.Models attribute), 162	attribute), 74 OPEN_FILE_FOLDER (pygamelib.assets.graphics.Models attribute), 162 OPEN_HANDS (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_LOWERED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPEN_MAILBOX_WITH_RAISED_FLAG (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPHIUCHUS (pygamelib.assets.graphics.Models attribute), 162 OPTICAL_DISK (pygamelib.assets.graphics.Models attribute), 162 OPTION_KEY (pygamelib.assets.graphics.MiscTechnicals attribute), 74 ORANGE_BOOK (pygamelib.assets.graphics.Models at-

```
ORANGE_CIRCLE (pygamelib.assets.graphics.Models overlappable()
                                                                                                                         (pygamelib.board_items.Player
              attribute), 162
                                                                                                  method), 381
ORANGE HEART (pygamelib.assets.graphics.Models at-
                                                                                    overlappable() (pygamelib.board_items.Projectile
              tribute), 162
                                                                                                  method), 393
ORANGE SQUARE
                              (pygamelib.assets.graphics.Models
                                                                                    overlappable()
                                                                                                                      (pygamelib.board_items.TextItem
              attribute), 162
                                                                                                  method), 401
                     (pygamelib.assets.graphics.Models
                                                                                                                             (pygamelib.board items.Tile
ORANGUTAN
                                                                                    overlappable()
              tribute), 162
                                                                                                  method), 411
ORTHODOX_CROSS (pygamelib.assets.graphics.Models
                                                                                    overlappable()
                                                                                                                      (pygamelib.board_items.Treasure
              attribute), 162
                                                                                                  method), 420
              (pygamelib.assets.graphics.Models attribute),
                                                                                    overlappable()
                                                                                                                            (pygamelib.board_items.Wall
OTTER
                                                                                                  method), 428
              162
OUTBOX_TRAY (pygamelib.assets.graphics.Models at-
                                                                                    OWL (pygamelib.assets.graphics.Models attribute), 162
                                                                                    OX (pygamelib.assets.graphics.Models attribute), 162
              tribute), 162
                                                                                    OYSTER (pygamelib.assets.graphics.Models attribute),
overlappable() (pygamelib.board_items.Actionable
              method), 211
                                                                                                   163
\verb|overlappable(|)| \textit{(pygamelib.board\_items.ActionableTile}| \textbf{D}|
              method), 220
overlappable() (pygamelib.board_items.BoardComplexItemg_TTON
                                                                                                           (pygamelib.assets.graphics.Models
                                                                                                                                                                 at-
              method), 229
                                                                                                  tribute), 166
overlappable() (pygamelib.board_items.BoardItem PACKAGE (pygamelib.assets.graphics.Models attribute),
              method), 247
                                                                                                   163
overlappable() (pygamelib.board_items.BoardItemComplexComponentib.gfx.ui.Menu attribute), 528
              method), 238
                                                                                    padding (pygamelib.gfx.ui.MenuAction attribute), 531
overlappable() (pygamelib.board_items.BoardItemVoipage_down() (pygamelib.gfx.ui.GridSelector method),
              method), 255
                                                                                                   523
overlappable()
                                   (pygamelib.board_items.Camera PAGE_FACING_UP (pygamelib.assets.graphics.Models
              method), 263
                                                                                                  attribute), 163
overlappable() (pygamelib.board_items.Character
                                                                                                          (pygamelib.gfx.ui.GridSelector method),
                                                                                   page_up()
              method), 271
                                                                                                   523
\verb|overlappable(|)| (pygamelib.board\_items. Complex Door \texttt{PAGE\_WITH\_CURL}| (pygamelib.assets. graphics. Models)| (pygamelib.assets. graphics. graphics
              method), 279
                                                                                                   attribute), 163
overlappable() (pygamelib.board_items.ComplexNPCPAGER
                                                                                                  (pygamelib.assets.graphics.Models attribute),
              method), 288
                                                                                                   163
overlappable() (pygamelib.board_items.ComplexPlay@AINTBRUSH
                                                                                                                    (pygamelib.assets.graphics.Models
              method), 298
                                                                                                  attribute), 163
overlappable()(pygamelib.board_items.ComplexTreaspace_M_TREE
                                                                                                            (pygamelib.assets.graphics.Models
              method), 306
                                                                                                   tribute), 163
overlappable()(pygamelib.board_items.ComplexWallPALMS_UP_TOGETHER
              method), 315
                                                                                                  (pygamelib.assets.graphics.Models attribute),
                                       (pygamelib.board items.Door
overlappable()
              method), 324
                                                                                    PANCAKES
                                                                                                           (pygamelib.assets.graphics.Models
overlappable () (pygamelib.board_items.GenericActionableStruquibate), 163
              method), 332
                                                                                                  (pygamelib.assets.graphics.Models attribute),
                                                                                    PANDA
overlappable() (pygamelib.board_items.GenericStructure
                                                                                                   163
              method), 348
                                                                                    PAPERCLIP
                                                                                                            (pygamelib.assets.graphics.Models
overlappable() (pygamelib.board_items.GenericStructureComplexiGampancent
              method), 340
                                                                                    PARACHUTE
                                                                                                           (pygamelib.assets.graphics.Models
                                                                                                                                                                 at-
overlappable() (pygamelib.board_items.Immovable
                                                                                                   tribute), 163
              method), 356
                                                                                    parent (pygamelib.base.Text attribute), 200
overlappable()
                                  (pygamelib.board_items.Movable
                                                                                    PARROT (pygamelib.assets.graphics.Models attribute),
              method), 364
                                                                                                   163
overlappable()
                                       (pygamelib.board items.NPC
                                                                                    PART_ALTERNATION_MARK
              method), 373
                                                                                                   (pygamelib.assets.graphics.Models attribute),
```

```
163
                                                     particle_pool (pygamelib.gfx.particles.CircleEmitter
Particle (class in pygamelib.gfx.particles), 570
                                                              attribute), 549
particle_emitter(pygamelib.board_items.Actionableparticle_pool(pygamelib.gfx.particles.ParticleEmitter
         attribute), 212
                                                              attribute), 567
particle_emitter(pygamelib.board_items.ActionableFilecticleEmitter(class in pygamelib.gfx.particles),
        attribute), 220
                                                              563
particle_emitter(pygamelib.board_items.BoardComplexhtem1ePool(class in pygamelib.gfx.particles), 568
                                                     ParticleSprixel (class in pygamelib.gfx.particles),
         attribute), 229
particle_emitter(pygamelib.board_items.BoardItem
                                                              575
        attribute), 247
                                                     PartitionParticle
                                                                                      (class
particle_emitter(pygamelib.board_items.BoardItemComplexGpygpondib.gfx.particles), 583
                                                     PARTY_POPPER (pygamelib.assets.graphics.Models at-
        attribute), 238
particle_emitter(pygamelib.board_items.BoardItemVoid
                                                              tribute), 163
                                                     PARTYING_FACE (pygamelib.assets.graphics.Models
        attribute), 255
particle_emitter (pygamelib.board_items.Camera
                                                              attribute), 163
         attribute), 263
                                                     PASSENGER_SHIP (pygamelib.assets.graphics.Models
particle_emitter(pygamelib.board_items.Character
                                                              attribute), 163
                                                     PASSIVE PULL DOWN OUTPUT SYMBOL
        attribute), 271
particle_emitter(pygamelib.board_items.ComplexDoor
                                                              (pygamelib.assets.graphics.MiscTechnicals
        attribute), 279
                                                              attribute), 74
particle_emitter(pygamelib.board_items.ComplexNPCSSIVE_PULL_UP_OUTPUT_SYMBOL
        attribute), 289
                                                              (pygamelib.assets.graphics.MiscTechnicals
particle_emitter(pygamelib.board_items.ComplexPlayer
                                                              attribute), 74
                                                     PASSPORT_CONTROL (pygamelib.assets.graphics.Models
        attribute), 298
particle_emitter(pygamelib.board_items.ComplexTreasure
                                                             attribute), 163
        attribute), 307
                                                     path (pygamelib.gfx.ui.FileDialog attribute), 519
particle_emitter(pygamelib.board_items.ComplexWaldthActuator(class in pygamelib.actuators), 13
        attribute), 315
                                                     PathFinder (class in pygamelib.actuators), 21
                        (pygamelib.board_items.Door
                                                    PatrolActuator (class in pygamelib.actuators), 17
particle_emitter
        attribute), 324
                                                     pause () (pygamelib.actuators.Actuator method), 9
particle_emitter (pygamelib.board_items.GenericActionable(Str(ucygrumelib.actuators.Behavioral method), 12
        attribute), 333
                                                     pause() (pygamelib.actuators.PathActuator method),
particle_emitter(pygamelib.board_items.GenericStructure
                                                             16
        attribute), 348
                                                     pause () (pygamelib.actuators.PathFinder method), 26
particle_emitter(pygamelib.board_items.GenericStructureCom(phygGovelpibacentactors.PatrolActuator method),
        attribute), 340
particle_emitter(pygamelib.board_items.Immovablepause()
                                                                     (pygamelib.actuators.RandomActuator
        attribute), 356
                                                              method), 30
particle_emitter(pygamelib.board_items.Movable pause() (pygamelib.actuators.UnidirectionalActuator
                                                              method), 33
        attribute), 365
particle_emitter(pygamelib.board_items.NPC at-
                                                    pause () (pygamelib.engine.Game method), 455
         tribute), 373
                                                     pause () (pygamelib.gfx.core.Animation method), 481
                                                    PAUSE_BUTTON (pygamelib.assets.graphics.Models at-
particle_emitter (pygamelib.board_items.Player
                                                              tribute), 163
        attribute), 382
particle_emitter(pygamelib.board_items.Projectile PAW_PRINTS
                                                                         (pygamelib.assets.graphics.Models
        attribute), 393
                                                              attribute), 163
particle_emitter(pygamelib.board_items.TextItem PEACE_SYMBOL(pygamelib.assets.graphics.Models at-
        attribute), 401
                                                              tribute), 163
particle_emitter (pygamelib.board_items.Tile at- PEACH
                                                             (pygamelib.assets.graphics.Models attribute),
        tribute), 411
particle_emitter(pygamelib.board_items.Treasure PEACOCK(pygamelib.assets.graphics.Models attribute),
        attribute), 420
                                                              163
particle_emitter (pygamelib.board_items.Wall at- PEANUTS (pygamelib.assets.graphics.Models attribute),
         tribute), 428
                                                              163
```

PEAR (pygamelib.assets.graphics.Models attribute), 163	164
PEN (pygamelib.assets.graphics.Models attribute), 163	PERSON_GOLFING (pygamelib.assets.graphics.Models
${\tt PENCIL} \ \ (pygame lib. assets. graphics. Models \ \ attribute),$	attribute), 164
163	PERSON_IN_BED (pygamelib.assets.graphics.Models
PENGUIN (pygamelib.assets.graphics.Models attribute),	attribute), 164
163	PERSON_IN_LOTUS_POSITION
PENSIVE_FACE (pygamelib.assets.graphics.Models attribute), 163	(pygamelib.assets.graphics.Models attribute), 164
PEOPLE_HUGGING (pygamelib.assets.graphics.Models attribute), 163	PERSON_IN_STEAMY_ROOM (pygamelib.assets.graphics.Models attribute),
PEOPLE_WITH_BUNNY_EARS	164
(pygamelib.assets.graphics.Models attribute), 163	PERSON_IN_SUIT_LEVITATING (pygamelib.assets.graphics.Models attribute),
PEOPLE_WRESTLING (pygamelib.assets.graphics.Model	
attribute), 163	PERSON_IN_TUXEDO (pygamelib.assets.graphics.Models
PERFORMING_ARTS (pygamelib.assets.graphics.Models	
attribute), 163	PERSON_JUGGLING (pygamelib.assets.graphics.Models
${\tt PERSEVERING_FACE}\ (pygame lib. assets. graphics. Model and the property of the property $	ls attribute), 164
attribute), 163 PERSON (pygamelib.assets.graphics.Models attribute),	PERSON_KNEELING (pygamelib.assets.graphics.Models attribute), 164
163	PERSON_LIFTING_WEIGHTS
PERSON_BIKING (pygamelib.assets.graphics.Models attribute), 164	(pygamelib.assets.graphics.Models attribute), 164
PERSON_BLOND_HAIR	PERSON_MOUNTAIN_BIKING
(pygamelib.assets.graphics.Models attribute), 164	(pygamelib.assets.graphics.Models attribute), 164
PERSON_BOUNCING_BALL	PERSON_PLAYING_HANDBALL
(pygamelib.assets.graphics.Models attribute), 164	(pygamelib.assets.graphics.Models attribute), 164
PERSON_BOWING (pygamelib.assets.graphics.Models	PERSON_PLAYING_WATER_POLO
attribute), 164	(pygamelib.assets.graphics.Models attribute),
PERSON_CARTWHEELING	164
(pygamelib.assets.graphics.Models attribute), 164	PERSON_POUTING (pygamelib.assets.graphics.Models attribute), 164
PERSON_CLIMBING (pygamelib.assets.graphics.Models	PERSON_RAISING_HAND
attribute), 164 PERSON FACEPALMING	(pygamelib.assets.graphics.Models attribute), 164
(pygamelib.assets.graphics.Models attribute),	PERSON_ROWING_BOAT
164 PERSON_FENCING (pygamelib.assets.graphics.Models	(pygamelib.assets.graphics.Models attribute), 164
attribute), 164	PERSON_RUNNING (pygamelib.assets.graphics.Models
PERSON_FROWNING (pygamelib.assets.graphics.Models	
attribute), 164	PERSON_SHRUGGING (pygamelib.assets.graphics.Models
PERSON_GESTURING_NO	attribute), 164
(pygamelib.assets.graphics.Models attribute), 164	PERSON_STANDING (pygamelib.assets.graphics.Models attribute), 164
PERSON_GESTURING_OK	PERSON_SURFING (pygamelib.assets.graphics.Models
(pygamelib.assets.graphics.Models attribute), 164	attribute), 164 PERSON_SWIMMING (pygamelib.assets.graphics.Models
PERSON_GETTING_HAIRCUT	attribute), 164
(pygamelib.assets.graphics.Models attribute),	
164	(pygamelib.assets.graphics.Models attribute), 164
PERSON_GETTING_MASSAGE (pygamelib.assets.graphics.Models attribute),	

(pygamelib.assets.graphics.Models attribute), 164	<pre>pickable() (pygamelib.board_items.GenericActionableStructure</pre>
PERSON_WALKING (pygamelib.assets.graphics.Models attribute), 164	pickable() (pygamelib.board_items.GenericStructure method), 348
PERSON_WEARING_TURBAN	pickable()(pygamelib.board_items.GenericStructureComplexCompone
(pygamelib.assets.graphics.Models attribute),	method), 340
165	<pre>pickable() (pygamelib.board_items.Immovable</pre>
PERSON_WITH_SKULLCAP	method), 356
(pygamelib.assets.graphics.Models attribute), 165	pickable() (pygamelib.board_items.Movable method), 365
PERSON_WITH_VEIL (pygamelib.assets.graphics.Model attribute), 165	spickable() (pygamelib.board_items.NPC method), 373
PERSPECTIVE (pygamelib.assets.graphics.MiscTechnica. attribute), 74	lspickable() (pygamelib.board_items.Player method), 382
PETRI_DISH (pygamelib.assets.graphics.Models attribute), 165	method), 393
PglBaseObject (class in pygamelib.base), 192	pickable() (pygamelib.board_items.TextItem
PglException, 194	method), 401
PglInvalidLevelException, 195	pickable() (pygamelib.board_items.Tile method),
PglInvalidTypeException, 195	411
PglInventoryException, 195	pickable() (pygamelib.board_items.Treasure
PglObjectIsNotMovableException, 195 PglOutOfBoardBoundException, 195	method), 420 pickable() (pygamelib.board_items.Wall method),
PIÑATA (pygamelib.assets.graphics.Models attribute),	428
165	PICKUP_TRUCK (pygamelib.assets.graphics.Models at-
PICK (pygamelib.assets.graphics.Models attribute), 165	tribute), 165
pickable() (pygamelib.board_items.Actionable	PIE (pygamelib.assets.graphics.Models attribute), 165
method), 212	PIG (pygamelib.assets.graphics.Models attribute), 165
<pre>pickable() (pygamelib.board_items.ActionableTile</pre>	
method), 220	tribute), 165
<pre>pickable() (pygamelib.board_items.BoardComplexItem</pre>	tribute), 165
method), 247	PILE_OF_POO (pygamelib.assets.graphics.Models attribute), 165
pickable()(<i>pygamelib.board_items.BoardItemComple.</i>	
method), 238	PINCHED_FINGERS (pygamelib.assets.graphics.Models
pickable() (pygamelib.board_items.BoardItemVoid	attribute), 165
method), 255 pickable() (pygamelib.board_items.Camera	PINCHING_HAND (pygamelib.assets.graphics.Models
pickable() (pygamelib.board_items.Camera method), 263	attribute), 165 PINE_DECORATION (pygamelib.assets.graphics.Models
pickable() (pygamelib.board_items.Character	attribute), 165
method), 271	PINEAPPLE (pygamelib.assets.graphics.Models at-
pickable() (pygamelib.board_items.ComplexDoor	tribute), 165
method), 279	PING_PONG (pygamelib.assets.graphics.Models at-
<pre>pickable() (pygamelib.board_items.ComplexNPC</pre>	tribute), 165
method), 289	PISCES (pygamelib.assets.graphics.Models attribute),
pickable() (pygamelib.board_items.ComplexPlayer	165
method), 298	PISTOL (pygamelib.assets.graphics.Models attribute),
pickable() (pygamelib.board_items.ComplexTreasure	165
method), 307 pickable() (pygamelib.board_items.ComplexWall	PIZZA (pygamelib.assets.graphics.Models attribute), 165
pickable() (pygamelib.board_items.ComplexWall method), 315	PLACARD (pygamelib.assets.graphics.Models attribute),
pickable() (pygamelib.board_items.Door method),	165
324	place() (pygamelib.engine.Screen method), 476

```
place_item() (pygamelib.engine.Board method), 440
                                                            (pygamelib.board_items.Camera
                                                                                             method),
PLACE_OF_INTEREST_SIGN
                                                            263
                                                   position as vector()
        (pygamelib.assets.graphics.MiscTechnicals
                                                            (pygamelib.board_items.Character
        attribute), 74
                                                                                             method),
PLACE_OF_WORSHIP (pygamelib.assets.graphics.Models
        attribute), 165
                                                   position_as_vector()
play_all() (pygamelib.gfx.core.Animation method),
                                                            (pygamelib.board items.ComplexDoor
                                                            method), 279
PLAY_BUTTON (pygamelib.assets.graphics.Models at-
                                                   position_as_vector()
                                                            (pygamelib.board_items.ComplexNPC
        tribute), 165
PLAY_OR_PAUSE_BUTTON
                                                            method), 289
        (pygamelib.assets.graphics.Models attribute), position_as_vector()
                                                            (pygamelib.board_items.ComplexPlayer
Player (class in pygamelib.board_items), 376
                                                            method), 298
PLEADING_FACE (pygamelib.assets.graphics.Models
                                                   position_as_vector()
        attribute), 165
                                                            (pygamelib.board_items.ComplexTreasure
PLUNGER (pygamelib.assets.graphics.Models attribute),
                                                            method), 307
                                                   position_as_vector()
PLUS (pygamelib.assets.graphics.Models attribute), 165
                                                            (pygamelib.board_items.ComplexWall
                   (pygamelib.assets.graphics.Models
POLICE CAR
                                                            method), 316
        attribute), 165
                                                   position_as_vector()
POLICE_CAR_LIGHT (pygamelib.assets.graphics.Models
                                                            (pygamelib.board_items.Door method), 325
        attribute), 165
                                                   position_as_vector()
POLICE_OFFICER (pygamelib.assets.graphics.Models
                                                            (pygamelib.board items.GenericActionableStructure
        attribute), 165
                                                            method), 333
POODLE (pygamelib.assets.graphics.Models attribute),
                                                   position_as_vector()
                                                            (pygamelib.board_items.GenericStructure
pool (pygamelib.gfx.particles.ParticlePool attribute),
                                                            method), 349
                                                   position_as_vector()
        (pygamelib.gfx.core.SpriteCollection method),
                                                            (pygamelib.board_items.GenericStructureComplexComponent
pop()
                                                            method), 341
POPCORN (pygamelib.assets.graphics.Models attribute),
                                                   position_as_vector()
        165
                                                            (pygamelib.board_items.Immovable
                                                                                             method),
                 (pygamelib.gfx.core.SpriteCollection
popitem()
        method), 488
                                                   position_as_vector()
                                                            (pygamelib.board_items.Movable
position_as_vector()
                                                                                             method),
        (pygamelib.board items.Actionable
                                         method),
                                                            365
        212
                                                   position_as_vector()
position_as_vector()
                                                            (pygamelib.board_items.NPC method), 373
        (pygamelib.board_items.ActionableTile
                                                   position_as_vector()
        method), 220
                                                            (pygamelib.board items.Player
                                                                                             method),
position_as_vector()
                                                            382
        (pygamelib.board_items.BoardComplexItem
                                                   position_as_vector()
        method), 229
                                                            (pygamelib.board_items.Projectile
                                                                                             method),
position_as_vector()
        (pygamelib.board_items.BoardItem method), position_as_vector()
        247
                                                            (pygamelib.board_items.TextItem
                                                                                             method),
position_as_vector()
                                                            402
        (pygamelib.board_items.BoardItemComplexComponenttion_as_vector()
        method), 238
                                                            (pygamelib.board_items.Tile method), 411
position_as_vector()
                                                   position_as_vector()
        (pygamelib.board items.BoardItemVoid
                                                            (pygamelib.board_items.Treasure
                                                                                             method),
        method), 255
                                                            420
position_as_vector()
                                                   position_as_vector()
```

(pygamelib.board_items.Wall method), 428	method), 200
POSITION_INDICATOR	PRINTER (pygamelib.assets.graphics.Models attribute),
(pygamelib.assets.graphics.MiscTechnicals	166
attribute), 74	progress_marker (pygamelib.gfx.ui.ProgressBar at-
POST_OFFICE (pygamelib.assets.graphics.Models at-	tribute), 541
tribute), 165	ProgressBar (class in pygamelib.gfx.ui), 540
POSTAL_HORN (pygamelib.assets.graphics.Models at-	ProgressDialog (class in pygamelib.gfx.ui), 542
tribute), 165	PROHIBITED (pygamelib.assets.graphics.Models
POSTBOX (pygamelib.assets.graphics.Models attribute),	attribute), 166
165	Projectile (class in pygamelib.board_items), 385
POT_OF_FOOD (pygamelib.assets.graphics.Models attribute), 166	PROJECTIVE (pygamelib.assets.graphics.MiscTechnicals attribute), 74
POTABLE_WATER (pygamelib.assets.graphics.Models	PURPLE_CIRCLE (pygamelib.assets.graphics.Models
attribute), 166	attribute), 166
	PURPLE_HEART (pygamelib.assets.graphics.Models at-
166	<i>tribute</i>), 166
POTTED_PLANT (pygamelib.assets.graphics.Models at-	PURPLE_SQUARE (pygamelib.assets.graphics.Models
tribute), 166	attribute), 166
POULTRY_LEG (pygamelib.assets.graphics.Models at-	PURSE (pygamelib.assets.graphics.Models attribute),
tribute), 166	166
POUND_BANKNOTE (pygamelib.assets.graphics.Models	PUSHPIN (pygamelib.assets.graphics.Models attribute),
attribute), 166	166
POUTING_CAT (pygamelib.assets.graphics.Models at-	
tribute), 166 POUTING_FACE (pygamelib.assets.graphics.Models at-	<pre>tribute), 166 pygamelib.constants (module), 432</pre>
tribute), 166	pygamerib.constants (mounte), 432
POWER_ON_OFF_SYMBOL	Q
(pygamelib.assets.graphics.MiscTechnicals	QUADRANT_LOWER_LEFT
attribute), 74	(pygamelib.assets.graphics.Blocks attribute),
POWER_ON_SYMBOL (pygamelib.assets.graphics.MiscTec	hnicals 38
attribute), 74	QUADRANT_LOWER_RIGHT
POWER_SLEEP_SYMBOL	(pygamelib.assets.graphics.Blocks attribute),
(pygamelib.assets.graphics.MiscTechnicals	38
attribute), 74	QUADRANT_UPPER_LEFT
POWER_SYMBOL (pygamelib.assets.graphics.MiscTechnical	4.28
attribute), 74 PRAYER_BEADS (pygamelib.assets.graphics.Models at-	38
tribute), 166	QUADRANT_UPPER_LEFT_AND_LOWER_LEFT_AND_LOWER_RIGHT
PREGNANT_WOMAN (pygamelib.assets.graphics.Models	(pygamelib.assets.graphics.Blocks attribute), 38
attribute), 166	QUADRANT_UPPER_LEFT_AND_LOWER_RIGHT
PRETZEL (pygamelib.assets.graphics.Models attribute),	(pygamelib.assets.graphics.Blocks attribute),
166	38
$\verb"PREVIOUS_PAGE" (pygame lib. assets. graphics. Misc Technical Control of the property of th$	$cm{q}m{b}$ adrant_upper_left_and_upper_right_and_lower_left
attribute), 74	(pygamelib.assets.graphics.Blocks attribute),
PRINCE (pygamelib.assets.graphics.Models attribute),	38
166	QUADRANT_UPPER_LEFT_AND_UPPER_RIGHT_AND_LOWER_RIGHT
PRINCESS (pygamelib.assets.graphics.Models at-	(pygamelib.assets.graphics.Blocks attribute),
<pre>tribute), 166 print_formatted() (pygamelib.base.Text method),</pre>	38
200	QUADRANT_UPPER_RIGHT (magnetic anaphics Placks attribute)
PRINT_SCREEN_SYMBOL	(pygamelib.assets.graphics.Blocks attribute), 38
(pygamelib.assets.graphics.MiscTechnicals	QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT
attribute), 74	(pygamelib.assets.graphics.Blocks attribute),
<pre>print_white_on_red() (pygamelib.base.Text static</pre>	38

QUADRANT_UPPER_RIGHT_AND_LOWER_LEFT_AND_ROWERDRBUHTON (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Blocks attribute). attribute), 167 RECYCLING SYMBOL (pygamelib.assets.graphics.Models QUESTION_MARK (pygamelib.assets.graphics.Models attribute), 167 attribute), 166 red (pygamelib.gfx.ui.ColorPicker attribute), 516 red() (pygamelib.base.Text static method), 201 R RED APPLE (pygamelib.assets.graphics.Models *tribute*), 167 r (pygamelib.gfx.core.Color attribute), 510 red_bright() (pygamelib.base.Text static method), RABBIT (pygamelib.assets.graphics.Models attribute), 201 RED_CIRCLE (pygamelib.assets.graphics.Models RABBIT FACE (pygamelib.assets.graphics.Models atattribute), 167 tribute), 166 red_dim() (pygamelib.base.Text static method), 201 RACCOON (pygamelib.assets.graphics.Models attribute), RED_ENVELOPE (pygamelib.assets.graphics.Models at-166 tribute), 167 RACING CAR (pygamelib.assets.graphics.Models RED_HAIR (pygamelib.assets.graphics.Models atattribute), 166 tribute), 167 RADICAL_SYMBOL_BOTTOM (pygamelib.assets.graphics.Models RED_HEART (pygamelib.assets.graphics.MiscTechnicals tribute), 167 attribute), 74 RED PAPER LANTERN (pygamelib.assets.graphics.Models attribute), RADIO (pygamelib.assets.graphics.Models attribute), RADIO_BUTTON (pygamelib.assets.graphics.Models at-(pygamelib.gfx.core.Sprixel red rect() class tribute), 166 method), 504 RADIOACTIVE (pygamelib.assets.graphics.Models atred rect() (pygamelib.gfx.particles.ParticleSprixel tribute), 166 class method), 581 RAILWAY_CAR (pygamelib.assets.graphics.Models at-RED_SQUARE (pygamelib.assets.graphics.Models tribute), 166 attribute), 167 RAILWAY TRACK (pygamelib.assets.graphics.Models (pygamelib.gfx.core.Sprixel red_square() attribute), 166 method), 504 RAINBOW (pygamelib.assets.graphics.Models attribute), red_square() (pygamelib.gfx.particles.ParticleSprixel 166 class method), 581 RAISED BACK OF HAND RED_TRIANGLE_POINTED_DOWN (pygamelib.assets.graphics.Models attribute), (pygamelib.assets.graphics.Models attribute), RAISED_FIST (pygamelib.assets.graphics.Models at-RED TRIANGLE POINTED UP tribute), 166 (pygamelib.assets.graphics.Models attribute), RAISED_HAND (pygamelib.assets.graphics.Models at-167 tribute), 166 REGISTERED (pygamelib.assets.graphics.Models RAISING_HANDS (pygamelib.assets.graphics.Models attribute), 167 attribute), 167 RELIEVED FACE (pygamelib.assets.graphics.Models RAM (pygamelib.assets.graphics.Models attribute), 167 attribute), 167 random() (pygamelib.gfx.core.Color class method), REMINDER_RIBBON (pygamelib.assets.graphics.Models 510 attribute), 167 RandomActuator (class in pygamelib.actuators), 27 remove_constraint() RandomColorParticle (class in (pygamelib.engine.Inventory method), 465 pygamelib.gfx.particles), 589 remove_directional_animation() RandomColorPartitionParticle (class (pygamelib.board_items.Projectile method), pygamelib.gfx.particles), 594 393 randomize() (pygamelib.gfx.core.Color method), 510 remove_directional_model() RAT (pygamelib.assets.graphics.Models attribute), 167 (pygamelib.board_items.Projectile method), RAZOR (pygamelib.assets.graphics.Models attribute), 393 remove_frame() (pygamelib.gfx.core.Animation RECEIPT (pygamelib.assets.graphics.Models attribute),

Index 659

167

method), 482

```
method), 289
remove_item() (pygamelib.engine.Board method),
        441
                                                   render_to_buffer()
                                                            (pygamelib.board items.ComplexPlayer
remove_npc() (pygamelib.engine.Game method), 455
remove_waypoint()
                                                            method), 298
        (pygamelib.actuators.PathFinder
                                         method),
                                                   render_to_buffer()
                                                            (pygamelib.board items.ComplexTreasure
                 (pygamelib.gfx.core.SpriteCollection
                                                            method), 307
rename()
                                                   render_to_buffer()
        method), 488
                                                            (pygamelib.board_items.ComplexWall
render() (pygamelib.engine.Screen method), 477
                (pygamelib.gfx.particles.ColorParticle
render()
                                                            method), 316
        method), 554
                                                   render_to_buffer()
render () (pygamelib.gfx.particles.ColorPartitionParticle
                                                            (pygamelib.board_items.Door method), 325
        method), 559
                                                   render_to_buffer()
                                                            (pygamelib.board_items.GenericActionableStructure
render() (pygamelib.gfx.particles.Particle method),
        573
                                                            method), 333
render()
             (pygamelib.gfx.particles.PartitionParticle render_to_buffer()
                                                            (pygamelib.board_items.GenericStructure
        method), 587
render() (pygamelib.gfx.particles.RandomColorParticle
                                                            method), 349
                                                   render_to_buffer()
        method), 592
render() (pygamelib.gfx.particles.RandomColorPartitionParticle (pygamelib.board_items.GenericStructureComplexComponent
        method), 597
                                                            method), 341
render_cell() (pygamelib.engine.Board method),
                                                   render to buffer()
                                                            (pygamelib.board_items.Immovable method),
                               (pygamelib.base.Text
render_to_buffer()
                                                            357
        method), 201
                                                   render_to_buffer()
render_to_buffer()
                                                            (pygamelib.board_items.Movable
                                                                                             method),
        (pygamelib.board_items.Actionable
                                         method),
        212
                                                   render_to_buffer() (pygamelib.board_items.NPC
render_to_buffer()
                                                            method), 374
        (pygamelib.board_items.ActionableTile
                                                   render_to_buffer()
        method), 220
                                                            (pygamelib.board_items.Player
                                                                                             method),
render_to_buffer()
                                                            382
        (pygamelib.board_items.BoardComplexItem
                                                   render_to_buffer()
                                                            (pygamelib.board_items.Projectile
        method), 229
                                                                                             method),
render to buffer()
                                                            393
        (pygamelib.board_items.BoardItem
                                         method),
                                                   render to buffer()
        247
                                                            (pygamelib.board items.TextItem
                                                                                             method),
render_to_buffer()
        (pygamelib.board_items.BoardItemComplexComponenter_to_buffer() (pygamelib.board_items.Tile
        method), 238
                                                            method), 411
render to buffer()
                                                   render to buffer()
        (pygamelib.board_items.BoardItemVoid
                                                            (pygamelib.board_items.Treasure
                                                                                             method),
        method), 255
render_to_buffer()
                                                   render_to_buffer() (pygamelib.board_items.Wall
        (pygamelib.board_items.Camera
                                         method),
                                                            method), 428
        263
                                                   render_to_buffer()
                                                                               (pygamelib.engine.Board
render_to_buffer()
                                                            method), 441
        (pygamelib.board_items.Character
                                         method),
                                                   render_to_buffer()
                                                                              (pygamelib.gfx.core.Sprite
        272
                                                            method), 495
render_to_buffer()
                                                   render_to_buffer()
                                                                             (pygamelib.gfx.core.Sprixel
        (pygamelib.board_items.ComplexDoor
                                                            method), 504
        method), 280
                                                   render_to_buffer()
render_to_buffer()
                                                            (pygamelib.gfx.particles.CircleEmitter
        (pygamelib.board items.ComplexNPC
                                                            method), 549
```

<pre>render_to_buffer()</pre>	method), 559
(pygamelib.gfx.particles.ParticleEmitter	reset() (pygamelib.gfx.particles.Particle method), 573
method), 567	reset() (pygamelib.gfx.particles.PartitionParticle
render_to_buffer()	method), 587
(pygamelib.gfx.particles.ParticleSprixel	reset() (pygamelib.gfx.particles.RandomColorParticle
method), 581	method), 592
render_to_buffer() (pygamelib.gfx.ui.Box	reset() (pygamelib.gfx.particles.RandomColorPartitionParticle
method), 513	method), 598
render_to_buffer()	reset_lifespan() (pygamelib.gfx.particles.ColorParticle method), 554
(pygamelib.gfx.ui.ColorPicker method), 516	
render_to_buffer()	reset_lifespan() (pygamelib.gfx.particles.ColorPartitionParticle
(pygamelib.gfx.ui.ColorPickerDialog method),	method), 559
514	reset_lifespan() (pygamelib.gfx.particles.Particle
render_to_buffer() (pygamelib.gfx.ui.FileDialog	method), 574
method), 519	reset_lifespan() (pygamelib.gfx.particles.PartitionParticle
render_to_buffer()	method), 588
(pygamelib.gfx.ui.GridSelector method),	- 110
523	method), 593
render_to_buffer()	reset_lifespan() (pygamelib.gfx.particles.RandomColorPartitionPart
(pygamelib.gfx.ui.GridSelectorDialog method),	method), 598
521	resize() (pygamelib.gfx.particles.ParticlePool
render_to_buffer()	method), 570
(pygamelib.gfx.ui.LineInputDialog method),	resize_pool()(pygamelib.gfx.particles.CircleEmitter
525	method), 549
render_to_buffer() (pygamelib.gfx.ui.Menu	resize_pool() (pygamelib.gfx.particles.ParticleEmitter
method), 528	method), 567
render_to_buffer()	restorable() (pygamelib.board_items.Actionable
(pygamelib.gfx.ui.MenuAction method), 531	method), 212
<pre>render_to_buffer() (pygamelib.gfx.ui.MenuBar</pre>	restorable()(pygamelib.board_items.ActionableTile
method), 534	method), 220
render_to_buffer()	restorable()(pygamelib.board_items.BoardComplexItem
(pygamelib.gfx.ui.MessageDialog method),	method), 230
537	restorable() (pygamelib.board_items.BoardItem
<pre>render_to_buffer()</pre>	method), 248
(pygamelib.gfx.ui.MultiLineInputDialog	$\verb restorable() (pygame lib.board_items.Board ItemComplexComponent) \\$
method), 539	method), 238
render_to_buffer()	restorable()(pygamelib.board_items.BoardItemVoid
(pygamelib.gfx.ui.ProgressBar method),	method), 255
541	restorable() (pygamelib.board_items.Camera
<pre>render_to_buffer()</pre>	method), 263
(pygamelib.gfx.ui.ProgressDialog method),	restorable() (pygamelib.board_items.Character
543	method), 272
REPEAT_BUTTON (pygamelib.assets.graphics.Models	restorable() (pygamelib.board_items.ComplexDoor
attribute), 167	method), 280
REPEAT_SINGLE_BUTTON	restorable() (pygamelib.board_items.ComplexNPC
(pygamelib.assets.graphics.Models attribute),	method), 289
167	restorable() (pygamelib.board_items.ComplexPlayer
RESCUE_WORKERS_HELMET	method), 298
(pygamelib.assets.graphics.Models attribute),	restorable() (pygamelib.board_items.ComplexTreasure
167	method), 307
reset () (pygamelib.gfx.core.Animation method), 482	restorable() (pygamelib.board_items.ComplexWall
reset() (pygametio.gyx.tore.rm.mation method), +82 reset() (pygamelib.gfx.particles.ColorParticle	method), 316
method), 554	restorable() (pygamelib.board_items.Door
reset () (pygamelib.gfx.particles.ColorPartitionParticle	method), 325
= (, p)oseoyparticles.cotori artitletti artitletti	

```
restorable() (pygamelib.board items.GenericActionableStructureRROW CURVING LEFT
        method), 333
                                                             (pygamelib.assets.graphics.Models attribute),
restorable() (pygamelib.board items.GenericStructure
                                                             167
                                                    RIGHT_ARROW_CURVING_UP
        method), 349
restorable() (pygamelib.board_items.GenericStructureComplexCppygpoomehib.assets.graphics.Models attribute),
        method), 341
                                                             167
                  (pygamelib.board items.Immovable
                                                    RIGHT CEILING (pygamelib.assets.graphics.MiscTechnicals
restorable()
        method), 357
                                                             attribute), 75
                    (pygamelib.board_items.Movable
restorable()
                                                    RIGHT CURLY BRACKET LOWER HOOK
        method), 365
                                                            (pygamelib.assets.graphics.MiscTechnicals
restorable()
                        (pygamelib.board_items.NPC
                                                            attribute), 75
                                                    RIGHT_CURLY_BRACKET_MIDDLE_PIECE
        method), 374
restorable()
                      (pygamelib.board_items.Player
                                                            (pygamelib.assets.graphics.MiscTechnicals
        method), 382
                                                             attribute), 75
restorable()
                   (pygamelib.board_items.Projectile
                                                    RIGHT_CURLY_BRACKET_UPPER_HOOK
        method), 394
                                                             (pygamelib.assets.graphics.MiscTechnicals
restorable()
                     (pygamelib.board_items.TextItem
                                                            attribute), 75
        method), 402
                                                    RIGHT DOWN HEAVY AND LEFT UP LIGHT
restorable() (pygamelib.board_items.Tile method),
                                                             (pygamelib.assets.graphics.BoxDrawings
        411
                                                             attribute), 47
restorable()
                    (pygamelib.board_items.Treasure
                                                    RIGHT_FACING_FIST
        method), 420
                                                             (pygamelib.assets.graphics.Models attribute),
                                                            167
restorable() (pygamelib.board_items.Wall method),
                                                    RIGHT FLOOR (pygamelib.assets.graphics.MiscTechnicals
        428
                                                            attribute), 75
RESTROOM
             (pygamelib.assets.graphics.Models
                                               at-
        tribute), 167
                                                    RIGHT HALF BLACK CIRCLE
RETURN_SYMBOL (pygamelib.assets.graphics.MiscTechnicals
                                                             (pygamelib.assets.graphics.GeometricShapes
        attribute), 75
                                                             attribute), 55
REVERSE_BUTTON (pygamelib.assets.graphics.Models
                                                    RIGHT_HALF_BLOCK (pygamelib.assets.graphics.Blocks
        attribute), 167
                                                            attribute), 38
REVERSED_NOT_SIGN
                                                    RIGHT_HEAVY_AND_LEFT_DOWN_LIGHT
        (pygamelib.assets.graphics.MiscTechnicals
                                                             (pygamelib.assets.graphics.BoxDrawings
        attribute), 75
                                                            attribute), 48
REVOLVING_HEARTS (pygamelib.assets.graphics.Models RIGHT_HEAVY_AND_LEFT_UP_LIGHT
        attribute), 167
                                                             (pygamelib.assets.graphics.BoxDrawings
                   (pygamelib.assets.graphics.Models
RHINOCEROS
                                                            attribute), 48
        attribute), 167
                                                    RIGHT HEAVY AND LEFT VERTICAL LIGHT
RIBBON (pygamelib.assets.graphics.Models attribute),
                                                            (pygamelib.assets.graphics.BoxDrawings
        167
                                                             attribute), 48
RICE_BALL (pygamelib.assets.graphics.Models
                                                    RIGHT_LIGHT_AND_LEFT_DOWN_HEAVY
                                                             (pygamelib.assets.graphics.BoxDrawings
        tribute), 167
RICE_CRACKER (pygamelib.assets.graphics.Models at-
                                                            attribute), 48
                                                    RIGHT LIGHT AND LEFT UP HEAVY
        tribute), 167
                                                            (pygamelib.assets.graphics.BoxDrawings
RIGHT_ANGER_BUBBLE
        (pygamelib.assets.graphics.Models attribute),
                                                            attribute), 48
                                                    RIGHT_LIGHT_AND_LEFT_VERTICAL_HEAVY
        167
RIGHT_ANGLE_WITH_DOWNWARDS_ZIGZAG_ARROW
                                                            (pygamelib.assets.graphics.BoxDrawings
        (pygamelib.assets.graphics.MiscTechnicals at-
                                                             attribute), 48
        tribute), 75
                                                    RIGHT_ONE_EIGHTH_BLOCK
RIGHT_ARROW (pygamelib.assets.graphics.Models at-
                                                             (pygamelib.assets.graphics.Blocks
                                                                                             attribute),
        tribute), 167
                                                            38
RIGHT_ARROW_CURVING_DOWN
                                                    RIGHT PARENTHESIS EXTENSION
        (pygamelib.assets.graphics.Models attribute),
                                                             (pygamelib.assets.graphics.MiscTechnicals
        167
                                                             attribute), 75
```

RIGHT PARENTHESIS LOWER HOOK ROUND_PUSHPIN (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.MiscTechnicals attribute), 168 attribute), 75 rounding precision (pygamelib.base.Vector2D at-RIGHT_PARENTHESIS_UPPER_HOOK tribute), 204 (pygamelib.assets.graphics.MiscTechnicals row (pygamelib.base.Vector2D attribute), 204 attribute), 75 row (pygamelib.board items.Actionable attribute), 212 RIGHT POINTING ANGLE BRACKET row (pygamelib.board items.ActionableTile attribute), (pygamelib.assets.graphics.MiscTechnicals (pygamelib.board_items.BoardComplexItem attribute), 75 row RIGHT_SQUARE_BRACKET_EXTENSION tribute), 230 (pygamelib.assets.graphics.MiscTechnicals row (pygamelib.board_items.BoardItem attribute), 248 attribute), 75 row (pygamelib.board_items.BoardItemComplexComponent RIGHT_SQUARE_BRACKET_LOWER_CORNER attribute), 239 row (pygamelib.board_items.BoardItemVoid attribute), (pygamelib.assets.graphics.MiscTechnicals 255 attribute), 75 RIGHT_SQUARE_BRACKET_UPPER_CORNER row (pygamelib.board_items.Camera attribute), 264 (pygamelib.assets.graphics.MiscTechnicals row (pygamelib.board_items.Character attribute), 272 attribute), 75 (pygamelib.board_items.ComplexDoor attribute), RIGHT_UP_HEAVY_AND_LEFT_DOWN_LIGHT (pygamelib.assets.graphics.BoxDrawings row (pygamelib.board items.ComplexNPC attribute), attribute), 48 RIGHT VERTICAL BOX LINE row (pygamelib.board_items.ComplexPlayer attribute), (pygamelib.assets.graphics.MiscTechnicals (pygamelib.board items.ComplexTreasure attribute), 75 row RING (pygamelib.assets.graphics.Models attribute), 167 tribute), 307 RING_OPERATOR (pygamelib.assets.graphics.GeometricShapes (pygamelib.board_items.ComplexWall attribute), attribute), 55 (pygamelib.assets.graphics.Models row (pygamelib.board_items.Door attribute), 325 RINGED_PLANET row (pygamelib.board_items.GenericActionableStructure attribute), 168 ROASTED_SWEET_POTATO attribute), 333 (pygamelib.assets.graphics.Models attribute), (pygamelib.board_items.GenericStructure attribute), 349 row (pygamelib.board_items.GenericStructureComplexComponent (pygamelib.assets.graphics.Models attribute), attribute), 341 ROCK (pygamelib.assets.graphics.Models attribute), 168 row (pygamelib.board items.Immovable attribute), 357 ROCKET (pygamelib.assets.graphics.Models attribute), row (pygamelib.board_items.Movable attribute), 365 row (pygamelib.board items.NPC attribute), 374 ROLL_OF_PAPER (pygamelib.assets.graphics.Models row (pygamelib.board_items.Player attribute), 382 attribute), 168 row (pygamelib.board items.Projectile attribute), 394 ROLLED_UP_NEWSPAPER row (pygamelib.board_items.TextItem attribute), 402 (pygamelib.assets.graphics.Models attribute), row (pygamelib.board items. Tile attribute), 412 row (pygamelib.board items. Treasure attribute), 421 row (pygamelib.board items. Wall attribute), 429 ROLLER_COASTER (pygamelib.assets.graphics.Models attribute), 168 (pygamelib.gfx.particles.CircleEmitter attribute), ROLLER_SKATE (pygamelib.assets.graphics.Models attribute), 168 (pygamelib.gfx.particles.ColorParticle attribute), ROLLING_ON_THE_FLOOR_LAUGHING 555 row (pygamelib.gfx.particles.ColorPartitionParticle at-(pygamelib.assets.graphics.Models attribute), tribute), 560 row (pygamelib.gfx.particles.Particle attribute), 574 ROOSTER (pygamelib.assets.graphics.Models attribute), row (pygamelib.gfx.particles.ParticleEmitter attribute), ROSE (pygamelib.assets.graphics.Models attribute), 168 ROSETTE (pygamelib.assets.graphics.Models attribute), row (pygamelib.gfx.particles.PartitionParticle attribute),

Index 663

588

168

row (pygamelib.gfx.particles.RandomColorParticle at- SCORPION (pygamelib.assets.graphics.Models attribute), 593 *tribute*), 168 row (pygamelib.gfx.particles.RandomColorPartitionParticl&creen (class in pygamelib.engine), 467 attribute), 598 (pygamelib.actuators.Actuator screen_column RUGBY_FOOTBALL (pygamelib.assets.graphics.Models attribute), 9 attribute), 168 screen column (pygamelib.actuators.Behavioral atrun () (pygamelib.engine.Game method), 455 tribute), 13 RUNNING_SHIRT (pygamelib.assets.graphics.Models screen_column (pygamelib.actuators.PathActuator attribute), 168 attribute), 16 RUNNING_SHOE (pygamelib.assets.graphics.Models atscreen_column (pygamelib.actuators.PathFinder attribute), 168 tribute), 26 screen_column (pygamelib.actuators.PatrolActuator S attribute), 20 screen_column (pygamelib.actuators.RandomActuator SAD_BUT_RELIEVED_FACE attribute), 30 (pygamelib.assets.graphics.Models attribute), screen_column (pygamelib.actuators. Unidirectional Actuator 168 attribute), 33 SAFETY_PIN (pygamelib.assets.graphics.Models screen_column (pygamelib.base.PglBaseObject atattribute), 168 tribute), 194 SAFETY_VEST (pygamelib.assets.graphics.Models atscreen column (pygamelib.base.Text attribute), 201 tribute), 168 screen_column (pygamelib.board_items.Actionable SAGITTARIUS (pygamelib.assets.graphics.Models atattribute), 212 tribute), 168 screen_column(pygamelib.board_items.ActionableTile SAILBOAT (pygamelib.assets.graphics.Models atattribute), 221 tribute), 168 screen_column(pygamelib.board_items.BoardComplexItem SAKE (pygamelib.assets.graphics.Models attribute), 168 attribute), 230 SALT (pygamelib.assets.graphics.Models attribute), 168 screen_column (pygamelib.board_items.BoardItem SANDWICH (pygamelib.assets.graphics.Models attribute), 248 *tribute*), 168 screen_column(pygamelib.board_items.BoardItemComplexComponent SANTA_CLAUS (pygamelib.assets.graphics.Models atattribute), 239 tribute), 168 screen_column(pygamelib.board_items.BoardItemVoid SARI (pygamelib.assets.graphics.Models attribute), 168 attribute), 256 SATELLITE (pygamelib.assets.graphics.Models screen_column (pygamelib.board_items.Camera at*tribute*), 168 *tribute*), 264 SATELLITE_ANTENNA screen_column (pygamelib.board_items.Character (pygamelib.assets.graphics.Models attribute), attribute), 272 screen column (pygamelib.board items.ComplexDoor SAUROPOD (pygamelib.assets.graphics.Models attribute), 280 tribute), 168 screen_column(pygamelib.board_items.ComplexNPC save board() (pygamelib.engine.Game method), 456 attribute), 290 save_config() (pygamelib.engine.Game method), $screen_column(pygamelib.board_items.ComplexPlayer)$ 457 attribute), 299 SAXOPHONE (pygamelib.assets.graphics.Models atscreen_column (pygamelib.board_items.ComplexTreasure tribute), 168 attribute), 307 scalable (pygamelib.gfx.core.Font attribute), 486 screen_column (pygamelib.board_items.ComplexWall scale() (pygamelib.gfx.core.Sprite method), 495 attribute), 316 SCARF (pygamelib.assets.graphics.Models attribute), screen_column (pygamelib.board_items.Door attribute), 325 SCHOOL (pygamelib.assets.graphics.Models attribute), screen_column (pygamelib.board_items.GenericActionableStructure 168 attribute), 333 (pygamelib.assets.graphics.Models SCISSORS atscreen_column (pygamelib.board_items.GenericStructure tribute), 168 attribute), 349 SCORPIO (pygamelib.assets.graphics.Models attribute), screen_column (pygamelib.board_items.GenericStructureComplexCom 168 attribute), 341

screen_column (pygamelib.board_items.Immovable (pygamelib.actuators.PathActuator screen row attribute), 357 attribute), 16 (pygamelib.actuators.PathFinder screen column (pygamelib.board items.Movable atscreen row *tribute*), 366 tribute), 26 screen_row (pygamelib.actuators.PatrolActuator atscreen column (pygamelib.board_items.NPC attribute), 374 tribute), 20 screen column (pygamelib.board items.Player at-(pygamelib.actuators.RandomActuator screen row tribute), 382 attribute), 30 screen_column (pygamelib.board_items.Projectile screen row (pygamelib.actuators. Unidirectional Actuator attribute), 394 attribute), 34 screen_column (pygamelib.board_items.TextItem at-(pygamelib.base.PglBaseObject atscreen_row tribute), 402 tribute), 194 (pygamelib.board_items.Tile screen_row (pygamelib.base.Text attribute), 201 screen_column *tribute*), 412 screen_row (pygamelib.board_items.Actionable atscreen_column (pygamelib.board_items.Treasure at*tribute*), 213 tribute), 421 (pygamelib.board_items.ActionableTile screen_row screen_column (pygamelib.board_items.Wall attribute), 221 tribute), 429 screen_row (pygamelib.board_items.BoardComplexItem screen_column (pygamelib.engine.Board attribute), attribute), 230 screen_row (pygamelib.board_items.BoardItem atscreen_column (pygamelib.engine.Game attribute), tribute), 248 screen_row(pygamelib.board_items.BoardItemComplexComponent (pygamelib.engine.Inventory attribute), 239 screen_column screen row (pygamelib.board items.BoardItemVoid tribute), 465 screen_column (pygamelib.engine.Screen attribute), attribute), 256 screen_row (pygamelib.board_items.Camera screen_column (pygamelib.gfx.core.Color attribute), tribute), 264 (pygamelib.board_items.Character screen_row screen_column (pygamelib.gfx.core.Sprite attribute), attribute), 272 496 (pygamelib.board_items.ComplexDoor screen_row screen_column (pygamelib.gfx.core.Sprixel attribute), 280 tribute), 505 screen_row (pygamelib.board_items.ComplexNPC atscreen_column (pygamelib.gfx.particles.CircleEmitter tribute), 290 attribute), 549 screen_row (pygamelib.board_items.ComplexPlayer screen_column(pygamelib.gfx.particles.ColorParticle attribute), 299 attribute), 555 screen_row (pygamelib.board_items.ComplexTreasure screen_column (pygamelib.gfx.particles.ColorPartitionParticle attribute), 308 screen_row (pygamelib.board_items.ComplexWall atattribute), 560 screen_column (pygamelib.gfx.particles.Particle attribute), 317 tribute), 574 screen_row (pygamelib.board_items.Door attribute), screen_column (pygamelib.gfx.particles.ParticleEmitter attribute), 567 screen_row (pygamelib.board_items.GenericActionableStructure screen_column (pygamelib.gfx.particles.ParticleSprixel attribute), 334 attribute), 582 screen_row (pygamelib.board_items.GenericStructure screen_column (pygamelib.gfx.particles.PartitionParticle attribute), 349 screen_row(pygamelib.board_items.GenericStructureComplexCompone attribute), 588 screen_column (pygamelib.gfx.particles.RandomColorParticle attribute), 341 attribute), 593 screen_row (pygamelib.board_items.Immovable atscreen_column (pygamelib.gfx.particles.RandomColorPartitionPartiti attribute), 598 (pygamelib.board_items.Movable screen_row screen_row (pygamelib.actuators.Actuator attribute), *tribute*), 366 10 screen_row (pygamelib.board_items.NPC attribute), (pygamelib.actuators.Behavioral 374 screen row at-

Index 665

screen row (pygamelib.board items.Player attribute),

tribute), 13

383	169
screen_row (pygamelib.board_items.Projectile attribute), 394	SEEDLING (pygamelib.assets.graphics.Models at- tribute), 169
screen_row (pygamelib.board_items.TextItem at- tribute), 402	SEGMENT (pygamelib.assets.graphics.MiscTechnicals attribute), 75
screen_row (pygamelib.board_items.Tile attribute), 412	<pre>select_next() (pygamelib.gfx.ui.Menu method), 528 select_next() (pygamelib.gfx.ui.MenuBar method),</pre>
screen_row (pygamelib.board_items.Treasure at- tribute), 421	534 select_previous() (pygamelib.gfx.ui.Menu
screen_row (pygamelib.board_items.Wall attribute), 429	method), 528 select_previous() (pygamelib.gfx.ui.MenuBar
screen_row (pygamelib.engine.Board attribute), 442	method), 534
screen_row (pygamelib.engine.Game attribute), 457 screen_row (pygamelib.engine.Inventory attribute), 466	selected (pygamelib.gfx.ui.Menu attribute), 529 selected (pygamelib.gfx.ui.MenuAction attribute), 531 selection (pygamelib.gfx.ui.ColorPicker attribute),
screen_row (pygamelib.engine.Screen attribute), 478	516
screen_row (pygamelib.gfx.core.Color attribute), 511 screen_row (pygamelib.gfx.core.Sprite attribute), 496	SELFIE (pygamelib.assets.graphics.Models attribute), 169
screen_row (pygamelib.gfx.core.Sprixel attribute), 505	serialize() (pygamelib.actuators.Actuator method), 10
screen_row (pygamelib.gfx.particles.CircleEmitter attribute), 550	serialize() (pygamelib.actuators.Behavioral method), 13
screen_row (pygamelib.gfx.particles.ColorParticle attribute), 555	serialize() (pygamelib.actuators.PathActuator method), 16
screen_row(pygamelib.gfx.particles.ColorPartitionParattribute), 560	ti cle rialize() (pygamelib.actuators.PathFinder method), 26
screen_row (pygamelib.gfx.particles.Particle attribute), 574	serialize() (pygamelib.actuators.PatrolActuator method), 20
screen_row (pygamelib.gfx.particles.ParticleEmitter attribute), 567	serialize() (pygamelib.actuators.RandomActuator method), 30
screen_row (pygamelib.gfx.particles.ParticleSprixel attribute), 582	serialize() (pygamelib.actuators.UnidirectionalActuator method), 34
screen_row (pygamelib.gfx.particles.PartitionParticle attribute), 588	serialize() (pygamelib.base.Text method), 201 serialize() (pygamelib.base.Vector2D method), 205
screen_row (pygamelib.gfx.particles.RandomColorPart attribute), 593	
screen_row(pygamelib.gfx.particles.RandomColorPart attribute), 598	titiomParticleze() (pygamelib.board_items.ActionableTile method), 221
SCREWDRIVER (pygamelib.assets.graphics.Models attribute), 169	serialize() (pygamelib.board_items.BoardComplexItem method), 230
SCROLL (pygamelib.assets.graphics.Models attribute), 169	serialize() (pygamelib.board_items.BoardItem method), 248
SEAL (pygamelib.assets.graphics.Models attribute), 169 search() (pygamelib.engine.Inventory method), 466	<pre>serialize() (pygamelib.board_items.BoardItemComplexComponent</pre>
<pre>search_frame()</pre>	serialize() (pygamelib.board_items.BoardItemVoid method), 256
SEAT (pygamelib.assets.graphics.Models attribute), 169 SECOND_PLACE_MEDAL	serialize() (pygamelib.board_items.Camera method), 264
(pygamelib.assets.graphics.Models attribute), 169	serialize() (pygamelib.board_items.Character method), 272
SECTOR (pygamelib.assets.graphics.MiscTechnicals attribute), 75	serialize() (pygamelib.board_items.ComplexDoor method), 280
SEE_NO_EVIL_MONKEY (pygamelib.assets.graphics.Models attribute),	serialize() (pygamelib.board_items.ComplexNPC method), 290

```
serialize() (pygamelib.board_items.ComplexPlayer
                                                             method), 567
        method), 299
                                                    serialize() (pygamelib.gfx.particles.ParticleSprixel
serialize() (pygamelib.board items.ComplexTreasure
                                                             method), 582
                                                    serialize() (pygamelib.gfx.particles.PartitionParticle
        method), 308
serialize()
                (pygamelib.board_items.ComplexWall
                                                             method), 588
                                                    serialize() (pygamelib.gfx.particles.RandomColorParticle
        method), 317
serialize() (pygamelib.board_items.Door method),
                                                             method), 593
                                                    serialize() (pygamelib.gfx.particles.RandomColorPartitionParticle
serialize() (pygamelib.board_items.GenericActionableStructuremethod), 598
        method), 334
                                                    session_log() (pygamelib.engine.Game method),
serialize() (pygamelib.board_items.GenericStructure
                                                             457
        method), 350
                                                    session_logs() (pygamelib.engine.Game method),
serialize() (pygamelib.board_items.GenericStructureComplexComponent
        method), 342
                                                    set_can_move() (pygamelib.board_items.Actionable
serialize()
                  (pygamelib.board_items.Immovable
                                                             method), 213
        method), 357
                                                    set_can_move() (pygamelib.board_items.ActionableTile
                     (pygamelib.board_items.Movable
serialize()
                                                             method), 221
        method), 366
                                                    set_can_move() (pygamelib.board_items.BoardComplexItem
serialize() (pygamelib.board_items.NPC method),
                                                             method), 230
        374
                                                    set_can_move() (pygamelib.board_items.BoardItem
serialize()
                      (pygamelib.board_items.Player
                                                             method), 248
                                                    set_can_move() (pygamelib.board_items.BoardItemComplexCompone
        method), 383
serialize()
                    (pygamelib.board_items.Projectile
                                                             method), 239
        method), 394
                                                    set can move() (pygamelib.board items.BoardItemVoid
serialize()
                     (pygamelib.board_items.TextItem
                                                             method), 256
        method), 403
                                                                         (pygamelib.board_items.Camera
                                                    set_can_move()
serialize() (pygamelib.board_items.Tile method),
                                                             method), 264
                                                    \verb"set_can_move"() \ \textit{(pygamelib.board_items.Character}
        412
                     (pygamelib.board_items.Treasure
serialize()
                                                             method), 272
                                                    set\_can\_move() (pygamelib.board_items.ComplexDoor
        method), 421
serialize() (pygamelib.board_items.Wall method),
                                                             method), 281
                                                    set_can_move() (pygamelib.board_items.ComplexNPC
        429
serialize() (pygamelib.engine.Board method), 442
                                                             method), 290
serialize() (pygamelib.engine.Inventory method),
                                                    set_can_move() (pygamelib.board_items.ComplexPlayer
        466
                                                             method), 299
serialize() (pygamelib.gfx.core.Animation method),
                                                    set_can_move() (pygamelib.board_items.ComplexTreasure
        482
                                                             method), 308
serialize() (pygamelib.gfx.core.Color method), 511
                                                    set_can_move() (pygamelib.board_items.ComplexWall
serialize() (pygamelib.gfx.core.Sprite method), 496
                                                             method), 317
serialize()
                 (pygamelib.gfx.core.SpriteCollection
                                                                            (pygamelib.board_items.Door
                                                    set_can_move()
        method), 488
                                                             method), 326
serialize()
               (pygamelib.gfx.core.Sprixel method),
                                                    set_can_move() (pygamelib.board_items.GenericActionableStructure
        505
                                                             method), 334
serialize() (pygamelib.gfx.particles.CircleEmitter
                                                    set_can_move() (pygamelib.board_items.GenericStructure
        method), 550
                                                             method), 350
               (pygamelib.gfx.particles.ColorParticle
                                                    set_can_move() (pygamelib.board_items.GenericStructureComplexCo
serialize()
        method), 555
                                                             method), 342
serialize() (pygamelib.gfx.particles.ColorPartitionPartiele_can_move() (pygamelib.board_items.Immovable
        method), 560
                                                             method), 358
                                                                         (pygamelib.board_items.Movable
serialize() (pygamelib.gfx.particles.EmitterProperties set_can_move()
        method), 562
                                                             method), 366
serialize()
                     (pygamelib.gfx.particles.Particle
                                                    set_can_move()
                                                                            (pygamelib.board_items.NPC
        method), 574
                                                             method), 374
serialize() (pygamelib.gfx.particles.ParticleEmitter set_can_move()
                                                                           (pygamelib.board items.Player
```

<pre>method), 383 set_can_move() (pygamelib.board_items.Projectile</pre>	<pre>set_overlappable() (pygamelib.board_items.ComplexWall</pre>
method), 394	method), 317
set_can_move() (pygamelib.board_items.TextItem method), 403	<pre>set_overlappable()</pre>
set_can_move() (pygamelib.board_items.Tile	set_overlappable()
method), 412	(pygamelib.board_items.GenericActionableStructure
set_can_move() (pygamelib.board_items.Treasure method), 421	<pre>method), 334 set_overlappable()</pre>
set_can_move() (pygamelib.board_items.Wall method), 429	(pygamelib.board_items.GenericStructure method), 350
set_color() (pygamelib.gfx.ui.ColorPickerDialog	
method), 514	(pygamelib.board_items.GenericStructureComplexComponent
set_destination() (magnetic actuators Path Finder method)	method), 342
(pygamelib.actuators.PathFinder method), 27	<pre>set_overlappable()</pre>
set_direction() (pygamelib.board_items.Projectile method), 394	358 set_overlappable()
set_overlappable()	(pygamelib.board_items.Movable method),
(pygamelib.board_items.Actionable method),	366
213	set_overlappable() (pygamelib.board_items.NPC
<pre>set_overlappable()</pre>	<pre>method), 375 set_overlappable()</pre>
method), 221	(pygamelib.board_items.Player method),
set_overlappable()	383
(pygamelib.board_items.BoardComplexItem method), 231	<pre>set_overlappable()</pre>
set_overlappable()	395
(pygamelib.board_items.BoardItem method), 248	<pre>set_overlappable()</pre>
set_overlappable()	403
method), 239	pomentoverlappable() (pygamelib.board_items.Tile method), 412
<pre>set_overlappable()</pre>	set_overlappable() (progradib board items Tragsura method)
method), 256	(pygamelib.board_items.Treasure method), 421
<pre>set_overlappable()</pre>	<pre>set_overlappable() (pygamelib.board_items.Wall method), 429</pre>
264	set_path() (pygamelib.actuators.PathActuator
set_overlappable()	method), 16
(pygamelib.board_items.Character method), 273	<pre>set_path() (pygamelib.actuators.PatrolActuator method), 20</pre>
set_overlappable()	set_pickable() (pygamelib.board_items.Actionable
(pygamelib.board_items.ComplexDoor	method), 213
<pre>method), 281 set_overlappable()</pre>	<pre>set_pickable() (pygamelib.board_items.ActionableTile method), 221</pre>
$(pygamelib.board_items.ComplexNPC$	set_pickable()(pygamelib.board_items.BoardComplexItem
<pre>method), 290 set_overlappable()</pre>	<pre>method), 231 set_pickable() (pygamelib.board_items.BoardItem</pre>
(pygamelib.board_items.ComplexPlayer	method), 248
method), 299	$\verb set_pickable () \textit{ (pygamelib.board_items.BoardItemComplexComponed)} \\$
set_overlappable()	method), 239
(pygamelib.board_items.ComplexTreasure method), 308	<pre>set_pickable() (pygamelib.board_items.BoardItemVoid method), 256</pre>

```
(pygamelib.board_items.Camera set_restorable()(pygamelib.board_items.Character
set_pickable()
             method), 264
                                                                                              method), 273
set_pickable() (pygamelib.board_items.Character set_restorable() (pygamelib.board_items.ComplexDoor
                                                                                              method), 281
             method), 273
set_pickable() (pygamelib.board_items.ComplexDoorset_restorable() (pygamelib.board_items.ComplexNPC
             method), 281
                                                                                              method), 290
set pickable() (pygamelib.board items.ComplexNPCset restorable() (pygamelib.board items.ComplexPlayer
             method), 290
                                                                                              method), 299
set_pickable() (pygamelib.board_items.ComplexPlayset_restorable() (pygamelib.board_items.ComplexTreasure
             method), 299
                                                                                              method), 308
\verb|set_pickable()| (pygamelib.board\_items. Complex Treasmet\_restorable()| (pygamelib.board\_items. Complex Wall)| (pygamelib.board\_item
             method), 308
                                                                                              method), 317
set_pickable() (pygamelib.board_items.ComplexWall set_restorable()
                                                                                                                      (pygamelib.board_items.Door
             method), 317
                                                                                              method), 326
                                     (pygamelib.board\_items.Door\  set_restorable() (pygamelib.board\_items.GenericActionableStructus)
set_pickable()
             method), 326
                                                                                              method), 334
set_pickable() (pygamelib.board_items.GenericActionadpleStreus:tuoecable() (pygamelib.board_items.GenericStructure
                                                                                              method), 350
             method), 334
set_pickable() (pygamelib.board_items.GenericStructureComplex
             method), 350
                                                                                              method), 342
set_pickable() (pygamelib.board_items.GenericStructuretComplexComplex(Complex) (pygamelib.board_items.Immovable
             method), 342
                                                                                              method), 358
set_pickable() (pygamelib.board_items.Immovable
                                                                                 set_restorable() (pygamelib.board_items.Movable
             method), 358
                                                                                              method), 366
                                (pygamelib.board_items.Movable
set_pickable()
                                                                                 set_restorable()
                                                                                                                      (pygamelib.board items.NPC
             method), 366
                                                                                              method), 375
                                     (pygamelib.board_items.NPC
                                                                                                                    (pygamelib.board_items.Player
set_pickable()
                                                                                 set_restorable()
             method), 375
                                                                                              method), 383
                                                                                 set_restorable() (pygamelib.board_items.Projectile
set_pickable()
                                   (pygamelib.board_items.Player
                                                                                              method), 395
             method), 383
set_pickable() (pygamelib.board_items.Projectile
                                                                                 set_restorable() (pygamelib.board_items.TextItem
             method), 395
                                                                                              method), 403
set_pickable()
                                (pygamelib.board_items.TextItem
                                                                                 set_restorable()
                                                                                                                        (pygamelib.board_items.Tile
             method), 403
                                                                                              method), 413
set_pickable()
                                                                                 set_restorable() (pygamelib.board_items.Treasure
                                       (pygamelib.board items.Tile
             method), 412
                                                                                              method), 421
set pickable()
                                (pygamelib.board_items.Treasure
                                                                                 set restorable()
                                                                                                                       (pygamelib.board items.Wall
                                                                                              method), 430
             method), 421
set_pickable()
                                      (pygamelib.board items.Wall
                                                                                set_selection() (pygamelib.gfx.ui.ColorPickerDialog
             method), 429
                                                                                              method), 514
set_restorable() (pygamelib.board_items.Actionableset_sprixel() (pygamelib.gfx.core.Sprite method),
                                                                                               496
             method), 213
set_restorable()(pygamelib.board_items.Actionable ##&_transparency()
                                                                                                                           (pygamelib.gfx.core.Sprite
             method), 222
                                                                                              method), 496
set_restorable() (pygamelib.board_items.BoardComplex.blenfiault()
                                                                                                            (pygamelib.gfx.core.SpriteCollection
             method), 231
                                                                                              method), 488
set_restorable() (pygamelib.board_items.BoardItemSEVEN_OCLOCK (pygamelib.assets.graphics.Models at-
             method), 249
                                                                                              tribute), 169
set_restorable() (pygamelib.board_items.BoardItemをចាក់ pygamelib.assets.graphics.Models at-
             method), 240
                                                                                              tribute), 169
set_restorable() (pygamelib.board_items.BoardItem Modifing_NEEDLE (pygamelib.assets.graphics.Models
             method), 256
                                                                                              attribute), 169
set_restorable() (pygamelib.board_items.Camera SHALLOW_PAN_OF_FOOD
             method), 265
                                                                                               (pygamelib.assets.graphics.Models attribute),
```

169	attribute), 169
SHAMROCK (pygamelib.assets.graphics.Models attribute), 169	SIX_THIRTY (pygamelib.assets.graphics.Models attribute), 169
SHARK (pygamelib.assets.graphics.Models attribute), 169	size (pygamelib.board_items.Actionable attribute), 213 size (pygamelib.board_items.ActionableTile attribute),
SHAVED_ICE (pygamelib.assets.graphics.Models	222
attribute), 169 SHEAF_OF_RICE (pygamelib.assets.graphics.Models	size (pygamelib.board_items.BoardComplexItem at- tribute), 231
attribute), 169 SHIELD (pygamelib.assets.graphics.Models attribute), 169	size (pygamelib.board_items.BoardItem attribute), 249 size (pygamelib.board_items.BoardItemComplexComponent attribute), 240
SHINTO_SHRINE (pygamelib.assets.graphics.Models attribute), 169	size (pygamelib.board_items.BoardItemVoid attribute), 256
SHIP (pygamelib.assets.graphics.Models attribute), 169	size (pygamelib.board_items.Camera attribute), 265
SHOOTING_STAR (pygamelib.assets.graphics.Models	size (pygamelib.board_items.Character attribute), 273
attribute), 169	size (pygamelib.board_items.ComplexDoor attribute),
SHOPPING_BAGS (pygamelib.assets.graphics.Models	281
attribute), 169	<pre>size (pygamelib.board_items.ComplexNPC attribute),</pre>
SHOPPING_CART (pygamelib.assets.graphics.Models	291
attribute), 169 SHORTCAKE (pygamelib.assets.graphics.Models at-	size (pygamelib.board_items.ComplexPlayer attribute), 299
SHORTCAKE (pygamelib.assets.graphics.Models attribute), 169	size (pygamelib.board_items.ComplexTreasure at-
SHORTS (pygamelib.assets.graphics.Models attribute),	tribute), 308
169	size (pygamelib.board_items.ComplexWall attribute),
SHOULDERED_OPEN_BOX	317
(pygamelib.assets.graphics.MiscTechnicals	size (pygamelib.board_items.Door attribute), 326
attribute), 75	$\verb size (pygame lib.board_items. Generic Actionable Structure$
show() (pygamelib.gfx.ui.ColorPickerDialog method),	attribute), 334
515	size (pygamelib.board_items.GenericStructure at-
show () (pygamelib.gfx.ui.Dialog method), 517	tribute), 350
show() (pygamelib.gfx.ui.FileDialog method), 519 show() (pygamelib.gfx.ui.GridSelectorDialog method),	size (pygamelib.board_items.GenericStructureComplexComponent attribute), 342
521	size (pygamelib.board_items.Immovable attribute), 358
show() (pygamelib.gfx.ui.LineInputDialog method),	size (pygamelib.board_items.Movable attribute), 366
525	size (pygamelib.board_items.NPC attribute), 375
show() (pygamelib.gfx.ui.MessageDialog method), 537	size (pygamelib.board_items.Player attribute), 383
show() (pygamelib.gfx.ui.MultiLineInputDialog	size (pygamelib.board_items.Projectile attribute), 395
method), 539	size (pygamelib.board_items.TextItem attribute), 403
show() (pygamelib.gfx.ui.ProgressDialog method), 543	size (pygamelib.board_items.Tile attribute), 413
show_hidden_files (pygamelib.gfx.ui.FileDialog	size (pygamelib.board_items.Treasure attribute), 422
attribute), 519	size (pygamelib.board_items.Wall attribute), 430
SHOWER (pygamelib.assets.graphics.Models attribute), 169	size() (pygamelib.engine.Inventory method), 466 SKATEBOARD (pygamelib.assets.graphics.Models
SHRIMP (pygamelib.assets.graphics.Models attribute),	attribute), 169
169	SKIER (pygamelib.assets.graphics.Models attribute),
SHUFFLE_TRACKS_BUTTON	169
(pygamelib.assets.graphics.Models attribute),	SKIS (pygamelib.assets.graphics.Models attribute), 169
169	SKULL (pygamelib.assets.graphics.Models attribute),
SHUSHING_FACE (pygamelib.assets.graphics.Models	169
attribute), 169 SIGN_OF_THE_HORNS	SKULL_AND_CROSSBONES (pygamelib.assets.graphics.Models attribute),
(pygamelib.assets.graphics.Models attribute),	169
169	SKUNK (pygamelib.assets.graphics.Models attribute),
SIX_OCLOCK (pygamelib.assets.graphics.Models	170

- SLED (pygamelib.assets.graphics.Models attribute), 170
 SLEEPING_FACE (pygamelib.assets.graphics.Models attribute), 170
 SLEEPY_FACE (pygamelib.assets.graphics.Models attribute), 170
 SLIGHTLY FROWNING FACE
- (pygamelib.assets.graphics.Models attribute), 170
- SLIGHTLY_SMILING_FACE (pygamelib.assets.graphics.Models attribute), 170
- SLOPE (pygamelib.assets.graphics.MiscTechnicals attribute), 75
- SLOT_MACHINE (pygamelib.assets.graphics.Models attribute), 170
- SLOTH (pygamelib.assets.graphics.Models attribute), 170
- SMALL_AIRPLANE (pygamelib.assets.graphics.Models attribute), 170
- SMALL_BLUE_DIAMOND (pygamelib.assets.graphics.Models attribute),
 170
- SMALL_ORANGE_DIAMOND (pygamelib.assets.graphics.Models attribute), 170
- SMILE (pygamelib.assets.graphics.MiscTechnicals attribute), 75
- SMILING_CAT_WITH_HEART_EYES (pygamelib.assets.graphics.Models attribute),
 170
- SMILING_FACE (pygamelib.assets.graphics.Models attribute), 170
- SMILING_FACE_WITH_HALO (pygamelib.assets.graphics.Models attribute),
- ${\tt SMILING_FACE_WITH_HEART_EYES} \\ (pygamelib.assets.graphics.Models \quad attribute),$
- SMILING_FACE_WITH_HEARTS (pygamelib.assets.graphics.Models attribute),
 170
- SMILING_FACE_WITH_HORNS (pygamelib.assets.graphics.Models attribute), 170
- SMILING_FACE_WITH_SMILING_EYES
 (pygamelib.assets.graphics.Models attribute),
 170
- SMILING_FACE_WITH_SUNGLASSES (pygamelib.assets.graphics.Models attribute),
- SMILING_FACE_WITH_TEAR

 (pygamelib.assets.graphics.Models attribute),

 170

 SMIRKING FACE (pygamelib.assets.graphics.Models

- attribute), 170
 SNAIL (pygamelib.assets.graphics.Models attribute), 170
- SNAKE (pygamelib.assets.graphics.Models attribute), 170
- SNEEZING_FACE (pygamelib.assets.graphics.Models attribute), 170
- SNOW_CAPPED_MOUNTAIN
 (pygamelib.assets.graphics.Models attribute),
 170
- SNOWBOARDER (pygamelib.assets.graphics.Models attribute), 170
- SNOWFLAKE (pygamelib.assets.graphics.Models attribute), 170
- SNOWMAN (pygamelib.assets.graphics.Models attribute), 170
- SNOWMAN_WITHOUT_SNOW
 (pygamelib.assets.graphics.Models attribute),
 170
- SOAP (pygamelib.assets.graphics.Models attribute), 170 SOCCER_BALL (pygamelib.assets.graphics.Models attribute), 170
- SOCKS (pygamelib.assets.graphics.Models attribute), 170
- SOFT_ICE_CREAM (pygamelib.assets.graphics.Models attribute), 170
- SOFTBALL (pygamelib.assets.graphics.Models attribute), 170
- SOFTWARE_FUNCTION_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 75
- SOON_ARROW (pygamelib.assets.graphics.Models attribute), 170
- SOS_BUTTON (pygamelib.assets.graphics.Models attribute), 170
- spacing (pygamelib.gfx.ui.MenuBar attribute), 534
- SPADE_SUIT (pygamelib.assets.graphics.Models attribute), 171
- SPAGHETTI (pygamelib.assets.graphics.Models attribute), 171
- SPARKLE (pygamelib.assets.graphics.Models attribute), 171
- SPARKLER (pygamelib.assets.graphics.Models attribute), 171
- SPARKLES (pygamelib.assets.graphics.Models attribute), 171
- SPARKLING_HEART (pygamelib.assets.graphics.Models attribute), 171
- SPEAK_NO_EVIL_MONKEY
 (pygamelib.assets.graphics.Models attribute),
 171
- SPEAKER_HIGH_VOLUME (pygamelib.assets.graphics.Models attribute), 171

- SPEAKER_LOW_VOLUME

 (pygamelib.assets.graphics.Models attribute),

 171

 SPEAKER_MEDIUM_VOLUME
 - (pygamelib.assets.graphics.Models attribute), 171
- SPEAKING_HEAD (pygamelib.assets.graphics.Models attribute), 171
- SPEECH_BALLOON (pygamelib.assets.graphics.Models attribute), 171
- SPEEDBOAT (pygamelib.assets.graphics.Models at tribute), 171
- SPIDER (pygamelib.assets.graphics.Models attribute),
 171
- SPIDER_WEB (pygamelib.assets.graphics.Models attribute), 171
- SPIRAL_CALENDAR (pygamelib.assets.graphics.Models attribute), 171
- SPIRAL_NOTEPAD (pygamelib.assets.graphics.Models attribute), 171
- SPIRAL_SHELL (pygamelib.assets.graphics.Models attribute), 171
- SPONGE (pygamelib.assets.graphics.Models attribute), 171
- SPOON (pygamelib.assets.graphics.Models attribute), 171
- SPORT_UTILITY_VEHICLE
 (pygamelib.assets.graphics.Models attribute),
 171
- SPORTS_MEDAL (pygamelib.assets.graphics.Models attribute), 171
- SPOUTING_WHALE (pygamelib.assets.graphics.Models attribute), 171
- Sprite (class in pygamelib.gfx.core), 489
- sprite (pygamelib.board_items.ActionableTile attribute), 222
- sprite (pygamelib.board_items.BoardComplexItem attribute), 231
- sprite (pygamelib.board_items.ComplexDoor attribute), 281
- sprite (pygamelib.board_items.ComplexNPC attribute), 291
- sprite (pygamelib.board_items.ComplexPlayer attribute), 300
- sprite (pygamelib.board_items.ComplexTreasure attribute), 309
- sprite (pygamelib.board_items.ComplexWall attribute), 318
- sprite (pygamelib.board_items.TextItem attribute),
- sprite (pygamelib.board_items.Tile attribute), 413
- SpriteCollection (class in pygamelib.gfx.core), 486
- Sprixel (class in pygamelib.gfx.core), 498

- sprixel() (pygamelib.gfx.core.Sprite method), 497 SQUARE_FOOT (pygamelib.assets.graphics.MiscTechnicals attribute), 75
- SQUARE_LOZENGE (pygamelib.assets.graphics.MiscTechnicals attribute), 75
- SQUARE_WITH_DIAGONAL_CROSSHATCH_FILL (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_HORIZONTAL_FILL (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_LEFT_HALF_BLACK (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_LOWER_RIGHT_DIAGONAL_HALF_BLACK (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_ORTHOGONAL_CROSSHATCH_FILL (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_RIGHT_HALF_BLACK (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_UPPER_LEFT_DIAGONAL_HALF_BLACK (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_UPPER_LEFT_TO_LOWER_RIGHT_FILL (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_UPPER_RIGHT_TO_LOWER_LEFT_FILL (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUARE_WITH_VERTICAL_FILL (pygamelib.assets.graphics.GeometricShapes attribute), 55
- SQUID (pygamelib.assets.graphics.Models attribute), 171
- SQUINTING_FACE_WITH_TONGUE (pygamelib.assets.graphics.Models attribute), 171
- STADIUM (pygamelib.assets.graphics.Models attribute), 171
- STAR (pygamelib.assets.graphics.Models attribute), 171 STAR_AND_CRESCENT
 - (pygamelib.assets.graphics.Models attribute), 171
- STAR_OF_DAVID (pygamelib.assets.graphics.Models attribute), 171
- STAR_STRUCK (pygamelib.assets.graphics.Models attribute), 171
- start() (pygamelib.actuators.Actuator method), 10
- start() (pygamelib.actuators.Behavioral method), 13
- start() (pygamelib.actuators.PathActuator method),
 16

```
start () (pygamelib.actuators.PathFinder method), 27
                                                             method), 273
                                                    \verb|store_position()| \textit{(pygamelib.board_items.ComplexDoor}|
start() (pygamelib.actuators.PatrolActuator method),
                                                             method), 282
start()
                (pygamelib.actuators.RandomActuator
                                                    store_position() (pygamelib.board_items.ComplexNPC
        method), 30
                                                             method), 291
start() (pygamelib.actuators.UnidirectionalActuator
                                                    store position()(pygamelib.board items.ComplexPlayer
        method), 34
                                                             method), 300
start () (pygamelib.engine.Game method), 458
                                                    store_position()(pygamelib.board_items.ComplexTreasure
start () (pygamelib.gfx.core.Animation method), 482
                                                             method), 309
state (pygamelib.engine.Game attribute), 458
                                                    store_position() (pygamelib.board_items.ComplexWall
STATION (pygamelib.assets.graphics.Models attribute),
                                                             method), 318
                                                                            (pygamelib.board_items.Door
        171
                                                    store_position()
STATUE_OF_LIBERTY
                                                             method), 327
        (pygamelib.assets.graphics.Models attribute),
                                                    store_position() (pygamelib.board_items.GenericActionableStructu
                                                             method), 335
STEAMING_BOWL (pygamelib.assets.graphics.Models
                                                    store_position() (pygamelib.board_items.GenericStructure
        attribute), 171
                                                             method), 351
STETHOSCOPE (pygamelib.assets.graphics.Models at-
                                                    store_position() (pygamelib.board_items.GenericStructureComplex
        tribute), 171
                                                             method), 342
stop () (pygamelib.actuators.Actuator method), 10
                                                    store position()(pygamelib.board items.Immovable
stop () (pygamelib.actuators.Behavioral method), 13
                                                             method), 358
stop() (pygamelib.actuators.PathActuator method), 17
                                                    store_position() (pygamelib.board_items.Movable
stop () (pygamelib.actuators.PathFinder method), 27
                                                             method), 367
stop() (pygamelib.actuators.PatrolActuator method),
                                                                             (pygamelib.board items.NPC
                                                    store position()
        20
                                                             method), 375
                                                    store_position()
stop()
                (pygamelib.actuators.RandomActuator
                                                                           (pygamelib.board_items.Player
        method), 31
                                                             method), 384
          (pygamelib.actuators.UnidirectionalActuator
                                                    store_position() (pygamelib.board_items.Projectile
stop()
        method), 34
                                                             method), 395
stop () (pygamelib.engine.Game method), 458
                                                    store_position() (pygamelib.board_items.TextItem
stop () (pygamelib.gfx.core.Animation method), 483
                                                             method), 404
STOP_BUTTON (pygamelib.assets.graphics.Models at-
                                                    store_position()
                                                                              (pygamelib.board_items.Tile
        tribute), 171
                                                             method), 413
STOP_SIGN
             (pygamelib.assets.graphics.Models
                                                    store_position() (pygamelib.board_items.Treasure
                                                             method), 422
        tribute), 172
{\tt STOPWATCH}\ (pygamelib. assets. graphics. Misc Technicals
                                                                             (pygamelib.board items.Wall
                                                    store_position()
        attribute), 75
                                                             method), 430
STOPWATCH (pygamelib.assets.graphics.Models
                                                    store_screen_position()
                                                             (pygamelib.actuators.Actuator
                                                                                               method),
        tribute), 171
                                                             10
store_position() (pygamelib.board_items.Actionable
        method), 214
                                                    store screen position()
store_position() (pygamelib.board_items.ActionableTile
                                                             (pygamelib.actuators.Behavioral
                                                                                               method),
        method), 222
store_position() (pygamelib.board_items.BoardComplexMtem_screen_position()
        method), 232
                                                             (pygamelib.actuators.PathActuator
                                                                                               method),
store_position() (pygamelib.board_items.BoardItem
                                                             17
        method), 249
                                                    store_screen_position()
store_position() (pygamelib.board_items.BoardItemComplexCoprypamerhib.actuators.PathFinder
                                                                                               method),
        method), 240
store_position() (pygamelib.board_items.BoardItem\iotidre_screen_position()
        method), 257
                                                             (pygamelib.actuators.PatrolActuator method),
store_position() (pygamelib.board_items.Camera
        method), 265
                                                    store_screen_position()
store_position() (pygamelib.board_items.Character
                                                             (pygamelib.actuators.RandomActuator
```

<pre>method), 31 store_screen_position()</pre>		(pygamelib.board_items.GenericStruc method), 351	ture
(pygamelib.actuators.UnidirectionalActuator	store	screen_position()	
method), 34	_	(pygamelib.board_items.GenericStruc	tureComplexComponent
store_screen_position()		method), 343	1 1
(pygamelib.base.PglBaseObject method),	store_	screen_position()	
194		(pygamelib.board_items.Immovable	method),
<pre>store_screen_position() (pygamelib.base.Text</pre>		359	
method), 201	store_	screen_position()	
store_screen_position()		(pygamelib.board_items.Movable	method),
(pygamelib.board_items.Actionable method),		367	
214	store_	screen_position()	276
store_screen_position()		(pygamelib.board_items.NPC method)), 3/6
(pygamelib.board_items.ActionableTile method), 223	store_	screen_position() (pygamelib.board_items.Player	mathad)
store_screen_position()		384	method),
(pygamelib.board_items.BoardComplexItem	store	screen_position()	
method), 232	00010_	. -	method),
store_screen_position()		396	,
(pygamelib.board_items.BoardItem method),	store_	screen_position()	
249		(pygamelib.board_items.TextItem	method),
store_screen_position()		404	
$(pygame lib. board_items. Board Item Complex Complex$	p onent ∈_	screen_position()	
method), 240		(pygamelib.board_items.Tile method),	414
store_screen_position()	store_	screen_position()	
(pygamelib.board_items.BoardItemVoid method), 257		(pygamelib.board_items.Treasure 422	method),
store_screen_position()	store_	screen_position()	
(pygamelib.board_items.Camera method),		(pygamelib.board_items.Wall method)	, 430
265	store_	screen_position()	
store_screen_position()		(pygamelib.engine.Board method), 44.	2
(pygamelib.board_items.Character method), 274	store_	screen_position() (pygamelib.engine.Game method), 458)
store_screen_position()	store	screen_position())
(pygamelib.board_items.ComplexDoor	2016_	(pygamelib.engine.Inventory method),	466
method), 282	store	screen_position()	100
store_screen_position()		(pygamelib.engine.Screen method), 47	8
(pygamelib.board_items.ComplexNPC	store_	screen_position()	
method), 292		(pygamelib.gfx.core.Color method), 51	11
store_screen_position()	store_	screen_position()	
(pygamelib.board_items.ComplexPlayer		(pygamelib.gfx.core.Sprite method), 49	97
method), 300	store_	screen_position()	
store_screen_position()		(pygamelib.gfx.core.Sprixel method), 5	505
(pygamelib.board_items.ComplexTreasure	store_	screen_position()	
method), 309		(pygamelib.gfx.particles.CircleEmitter	•
store_screen_position()		method), 550	
(pygamelib.board_items.ComplexWall method), 318	store_	screen_position()	
store_screen_position()		(pygamelib.gfx.particles.ColorParticle method), 555	;
(pygamelib.board_items.Door method), 327	store	screen_position()	
store_screen_position()	5010_	(pygamelib.gfx.particles.ColorPartitio	nParticle
(pygamelib.board_items.GenericActionableStruc	ture	method), 560	
method), 335		screen_position()	
store screen position()			method),

574	(pygamelib.assets.graphics.Models attribute),
store_screen_position()	172
(pygamelib.gfx.particles.ParticleEmitter method), 567	SUNSET (pygamelib.assets.graphics.Models attribute), 172
store_screen_position()	SUPERHERO (pygamelib.assets.graphics.Models at-
(pygamelib.gfx.particles.ParticleSprixel method), 582	tribute), 172 SUPERVILLAIN (pygamelib.assets.graphics.Models at-
store_screen_position()	tribute), 172
(pygamelib.gfx.particles.PartitionParticle method), 588	SUSHI (pygamelib.assets.graphics.Models attribute), 172
store_screen_position()	SUSPENSION_RAILWAY
(pygamelib.gfx.particles.RandomColorParticle method), 593	(pygamelib.assets.graphics.Models attribute), 172
store_screen_position()	SWAN (pygamelib.assets.graphics.Models attribute), 172
(pygamelib.gfx.particles.RandomColorPartitionP method), 599	asticle T_DROPLETS (pygamelib.assets.graphics.Models attribute), 172
STRAIGHT_RULER (pygamelib.assets.graphics.Models	
attribute), 172 STRAIGHTNESS (pygamelib.assets.graphics.MiscTechnical	attribute), 75 BYNAGOGUE (pygamelib.assets.graphics.Models at-
attribute), 75	tribute), 172
STRAWBERRY (pygamelib.assets.graphics.Models attribute), 172	SYRINGE (pygamelib.assets.graphics.Models attribute), 172
STUDIO_MICROPHONE	-
(pygamelib.assets.graphics.Models attribute),	T
172	T_REX (pygamelib.assets.graphics.Models attribute),
STUFFED_FLATBREAD (pygamelib.assets.graphics.Models attribute),	174 T. CHIDT (magnetik assets anaphies Models attribute)
172	T_SHIRT (pygamelib.assets.graphics.Models attribute), 174
style (pygamelib.base.Text attribute), 202	TACO (pygamelib.assets.graphics.Models attribute), 172
SUMMATION_BOTTOM (pygamelib.assets.graphics.MiscTe attribute), 75	chnicalsut_BOX (pygamelib.assets.graphics.Models attribute), 172
SUMMATION_TOP (pygamelib.assets.graphics.MiscTechniattribute), 75	
SUN (pygamelib.assets.graphics.Models attribute), 172	TANABATA_TREE (pygamelib.assets.graphics.Models
${\tt SUN_BEHIND_CLOUD}\ (pygamelib. assets. graphics. Models)$	attribute), 172
attribute), 172	TANGERINE (pygamelib.assets.graphics.Models at-
SUN_BEHIND_LARGE_CLOUD (magnetic agents arenhics Models, attribute)	tribute), 172
172	TAURUS (pygamelib.assets.graphics.Models attribute), 172
SUN_BEHIND_RAIN_CLOUD	TAXI (pygamelib.assets.graphics.Models attribute), 172
(pygamelib.assets.graphics.Models attribute),	TEACUP_WITHOUT_HANDLE
172 SUN_BEHIND_SMALL_CLOUD	(pygamelib.assets.graphics.Models attribute), 172
(pygamelib.assets.graphics.Models attribute),	TEAPOT (pygamelib.assets.graphics.Models attribute),
172	172
SUN_WITH_FACE (pygamelib.assets.graphics.Models attribute), 172	TEAR_OFF_CALENDAR (pygamelib.assets.graphics.Models attribute),
SUNFLOWER (pygamelib.assets.graphics.Models at-	172
tribute), 172 SUNGLASSES (pygamelib.assets.graphics.Models	TEDDY_BEAR (pygamelib.assets.graphics.Models
attribute), 172	attribute), 172 TELEPHONE (pygamelib.assets.graphics.Models at-
SUNRISE (pygamelib.assets.graphics.Models attribute),	tribute), 172
172	TELEPHONE_RECEIVER
SUNRISE OVER MOUNTAINS	(nygamelih assets granhics Models attribute)

173 TELEPHONE_RECORDER	TICKET (pygamelib.assets.graphics.Models attribute), 173
(pygamelib.assets.graphics.MiscTechnicals attribute), 75	TIGER (pygamelib.assets.graphics.Models attribute), 173
TELESCOPE (pygamelib.assets.graphics.Models at- tribute), 173	TIGER_FACE (pygamelib.assets.graphics.Models attribute), 173
TELEVISION (pygamelib.assets.graphics.Models	Tile (class in pygamelib.board_items), 405
attribute), 173 TEN_OCLOCK (pygamelib.assets.graphics.Models	TIMER_CLOCK (pygamelib.assets.graphics.MiscTechnicals attribute), 75
attribute), 173 TEN_THIRTY (pygamelib.assets.graphics.Models	TIMER_CLOCK (pygamelib.assets.graphics.Models attribute), 173
attribute), 173 TENNIS (pygamelib.assets.graphics.Models attribute),	tint() (pygamelib.gfx.core.Sprite method), 497 TIRED_FACE (pygamelib.assets.graphics.Models
173	attribute), 173
TENT (pygamelib.assets.graphics.Models attribute), 173	title (pygamelib.gfx.ui.Box attribute), 513
terminate() (pygamelib.gfx.particles.ColorParticle method), 555	title (pygamelib.gfx.ui.ColorPickerDialog attribute), 515
terminate() (pygamelib.gfx.particles.ColorPartitionPamethod), 560	rtiàlele (pygamelib.gfx.ui.GridSelectorDialog attribute), 521
terminate() (pygamelib.gfx.particles.Particle method), 574	title (pygamelib.gfx.ui.LineInputDialog attribute), 525 title (pygamelib.gfx.ui.Menu attribute), 529
terminate() (pygamelib.gfx.particles.PartitionParticle method), 589	title (pygamelib.gfx.ui.MenuAction attribute), 531 title (pygamelib.gfx.ui.MessageDialog attribute), 537
terminate() (pygamelib.gfx.particles.RandomColorParmethod), 594	
terminate() (pygamelib.gfx.particles.RandomColorParmethod), 599	
TEST_TUBE (pygamelib.assets.graphics.Models at-	method), 531
tribute), 173	<pre>to_json_file() (pygamelib.gfx.core.SpriteCollection</pre>
Text (class in pygamelib.base), 195	method), 488
text (pygamelib.base.Text attribute), 202 text (pygamelib.board_items.TextItem attribute), 405	toggle_active() (pygamelib.gfx.particles.CircleEmitter method), 550
TextItem (class in pygamelib.board_items), 396	<pre>toggle_active() (pygamelib.gfx.particles.ParticleEmitte</pre>
THERMOMETER (pygamelib.assets.graphics.Models attribute), 173	TOILET (pygamelib.assets.graphics.Models attribute),
THINKING_FACE (pygamelib.assets.graphics.Models	173
attribute), 173 THIRD_PLACE_MEDAL	TOKYO_TOWER (pygamelib.assets.graphics.Models attribute), 173
(pygamelib.assets.graphics.Models attribute), 173	TOMATO (pygamelib.assets.graphics.Models attribute), 173
THONG_SANDAL (pygamelib.assets.graphics.Models attribute), 173	TONGUE (pygamelib.assets.graphics.Models attribute), 173
THOUGHT_BALLOON (pygamelib.assets.graphics.Models attribute), 173	TOOLBOX (pygamelib.assets.graphics.Models attribute), 173
THREAD (pygamelib.assets.graphics.Models attribute), 173	TOOTH (pygamelib.assets.graphics.Models attribute), 173
THREE_OCLOCK (pygamelib.assets.graphics.Models attribute), 173	TOOTHBRUSH (pygamelib.assets.graphics.Models attribute), 173
THREE_THIRTY (pygamelib.assets.graphics.Models attribute), 173	TOP_ARROW (pygamelib.assets.graphics.Models attribute), 173
THUMBS_DOWN (pygamelib.assets.graphics.Models attribute), 173	TOP_CURLY_BRACKET (pygamelib.assets.graphics.MiscTechnicals
THUMBS_UP (pygamelib.assets.graphics.Models at-	attribute), 75
<i>tribute</i>), 173	TOP_HALF_INTEGRAL

(pygamelib.assets.graphics.MiscTechnicals attribute), 75	attribute), 174 TRUMPET (pygamelib.assets.graphics.Models attribute),
TOP_HAT (pygamelib.assets.graphics.Models attribute),	174
173	TULIP (pygamelib.assets.graphics.Models attribute),
TOP_LEFT_CORNER (pygamelib.assets.graphics.MiscTec attribute), 75	hnicals 174 TUMBLER_GLASS (pygamelib.assets.graphics.Models
TOP_LEFT_CROP (pygamelib.assets.graphics.MiscTechnic	***
attribute), 75	TURKEY (pygamelib.assets.graphics.Models attribute),
TOP_PARENTHESIS (pygamelib.assets.graphics.MiscTec attribute), 75	hnicals 174 TURNED_NOT_SIGN (pygamelib.assets.graphics.MiscTechnicals
TOP_RIGHT_CORNER (pygamelib.assets.graphics.MiscTe	echnicals attribute), 76
attribute), 76	TURTLE (pygamelib.assets.graphics.Models attribute), nicals 174
TOP_RIGHT_CROP (pygamelib.assets.graphics.MiscTech attribute), 76	TWELVE_OCLOCK (pygamelib.assets.graphics.Models
TOP_SQUARE_BRACKET	attribute), 174
(pygamelib.assets.graphics.MiscTechnicals attribute), 76	TWELVE_THIRTY (pygamelib.assets.graphics.Models attribute), 174
TOP_TORTOISE_SHELL_BRACKET	TWO_HEARTS (pygamelib.assets.graphics.Models
(pygamelib.assets.graphics.MiscTechnicals	attribute), 174
attribute), 76 TORNADO (pygamelib.assets.graphics.Models attribute),	TWO_HUMP_CAMEL (pygamelib.assets.graphics.Models attribute), 174
173	TWO_OCLOCK (pygamelib.assets.graphics.Models
TOTAL_RUNOUT (pygamelib.assets.graphics.MiscTechnic	als attribute), 174
attribute), 76 TRACKBALL (pygamelib.assets.graphics.Models at-	TWO_THIRTY (pygamelib.assets.graphics.Models attribute), 174
TRACKBALL (pygamelib.assets.graphics.Models attribute), 173	
TRACTOR (pygamelib.assets.graphics.Models attribute),	U
173 TRADE_MARK (pygamelib.assets.graphics.Models	UiConfig (class in pygamelib.gfx.ui), 544
TRADE MARK (pygamelib.assets.graphics.Models	
	UMBRELLA (pygamelib.assets.graphics.Models at- tribute) 174
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute),	umbrella (pygamelib.assets.graphics.Models at- tribute), 174 umbrella_on_ground
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute),
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute),
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute),
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in pygamelib.actuators), 31
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models attribute), 174 trigger_rendering() (pygamelib.engine.Screen	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in pygamelib.actuators), 31 unit () (pygamelib.base.Vector2D method), 205 UNLOCKED (pygamelib.assets.graphics.Models at-
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in pygamelib.actuators), 31 unit () (pygamelib.base.Vector2D method), 205
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models attribute), 174 trigger_rendering() (pygamelib.engine.Screen method), 478 TROLLEYBUS (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in pygamelib.actuators), 31 unit () (pygamelib.base.Vector2D method), 205 UNLOCKED (pygamelib.assets.graphics.Models attribute), 174 UP_ARROW (pygamelib.assets.graphics.Models attribute), 174
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models attribute), 174 trigger_rendering() (pygamelib.engine.Screen method), 478 TROLLEYBUS (pygamelib.assets.graphics.Models	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in pygamelib.actuators), 31 unit () (pygamelib.assets.graphics.Models at- tribute), 174 UP_ARROW (pygamelib.assets.graphics.Models at- tribute), 174 UP_ARROWHEAD (pygamelib.assets.graphics.Models at- tribute), 174 UP_ARROWHEAD (pygamelib.assets.graphics.MiscTechnicals
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models attribute), 174 trigger_rendering() (pygamelib.engine.Screen method), 478 TROLLEYBUS (pygamelib.assets.graphics.Models attribute), 174 TROPHY (pygamelib.assets.graphics.Models attribute), 174 TROPHY (pygamelib.assets.graphics.Models attribute), 174 TROPICAL_DRINK (pygamelib.assets.graphics.Models	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in pygamelib.actuators), 31 unit () (pygamelib.base.Vector2D method), 205 UNLOCKED (pygamelib.assets.graphics.Models attribute), 174 UP_ARROW (pygamelib.assets.graphics.Models attribute), 174
attribute), 173 TRAIN (pygamelib.assets.graphics.Models attribute), 174 TRAM (pygamelib.assets.graphics.Models attribute), 174 TRAM_CAR (pygamelib.assets.graphics.Models attribute), 174 TRANSGENDER_SYMBOL (pygamelib.assets.graphics.Models attribute), 174 Treasure (class in pygamelib.board_items), 415 TRIANGULAR_FLAG (pygamelib.assets.graphics.Models attribute), 174 TRIANGULAR_RULER (pygamelib.assets.graphics.Models attribute), 174 TRIDENT_EMBLEM (pygamelib.assets.graphics.Models attribute), 174 trigger_rendering() (pygamelib.engine.Screen method), 478 TROLLEYBUS (pygamelib.assets.graphics.Models attribute), 174 TROPHY (pygamelib.assets.graphics.Models attribute), 174	tribute), 174 UMBRELLA_ON_GROUND (pygamelib.assets.graphics.Models attribute), 174 UMBRELLA_WITH_RAIN_DROPS (pygamelib.assets.graphics.Models attribute), 174 UNAMUSED_FACE (pygamelib.assets.graphics.Models attribute), 174 UNDO_SYMBOL (pygamelib.assets.graphics.MiscTechnicals attribute), 76 UNICORN (pygamelib.assets.graphics.Models attribute), s 174 UnidirectionalActuator (class in pygamelib.actuators), 31 unit () (pygamelib.base.Vector2D method), 205 UNLOCKED (pygamelib.assets.graphics.Models attribute), 174 UP_ARROW (pygamelib.assets.graphics.Models attribute), 174 UP_ARROWHEAD (pygamelib.assets.graphics.Models attribute), 174 UP_ARROWHEAD (pygamelib.assets.graphics.MiscTechnicals attribute), 76

attribute), 55

UP_BUTTON (pygamelib.assets.graphics.Models at- UP POINTING TRIANGLE WITH RIGHT HALF BLACK tribute), 174 (pygamelib.assets.graphics.GeometricShapes UP_DOUBLE_AND_HORIZONTAL_SINGLE attribute), 56 (pygamelib.assets.graphics.BoxDrawings UP_RIGHT_ARROW (pygamelib.assets.graphics.Models attribute), 48 attribute), 175 UP DOUBLE AND LEFT SINGLE UP SINGLE AND HORIZONTAL DOUBLE (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 48 attribute), 48 UP_DOUBLE_AND_RIGHT_SINGLE UP_SINGLE_AND_LEFT_DOUBLE (pygamelib.assets.graphics.BoxDrawings (pygamelib.assets.graphics.BoxDrawings attribute), 48 attribute), 48 UP_DOWN_ARROW (pygamelib.assets.graphics.Models UP_SINGLE_AND_RIGHT_DOUBLE (pygamelib.assets.graphics.BoxDrawings attribute), 174 UP_HEAVY_AND_DOWN_HORIZONTAL_LIGHT attribute), 48 update() (pygamelib.engine.Screen method), 478 (pygamelib.assets.graphics.BoxDrawings attribute), 48 update() (pygamelib.gfx.core.SpriteCollection UP_HEAVY_AND_HORIZONTAL_LIGHT method), 489 (pygamelib.assets.graphics.BoxDrawings update() (pygamelib.gfx.particles.CircleEmitter attribute), 48 method), 550 UP_HEAVY_AND_LEFT_DOWN_LIGHT update() (pygamelib.gfx.particles.ColorParticle (pygamelib.assets.graphics.BoxDrawings method), 555 attribute), 48 update() (pygamelib.gfx.particles.ColorPartitionParticle UP_HEAVY_AND_LEFT_LIGHT method), 560 (pygamelib.assets.graphics.BoxDrawings update() (pygamelib.gfx.particles.Particle method), attribute), 48 575 UP_HEAVY_AND_RIGHT_DOWN_LIGHT update() (pygamelib.gfx.particles.ParticleEmitter (pygamelib.assets.graphics.BoxDrawings method), 568 attribute), 48 (pygamelib.gfx.particles.PartitionParticle update() UP_HEAVY_AND_RIGHT_LIGHT method), 589 (pygamelib.assets.graphics.BoxDrawings update() (pygamelib.gfx.particles.RandomColorParticle attribute), 48 method), 594 UP_LEFT_ARROW (pygamelib.assets.graphics.Models update()(pygamelib.gfx.particles.RandomColorPartitionParticle method), 599 attribute), 174 UP_LIGHT_AND_DOWN_HORIZONTAL_HEAVY update_menu_entry() (pygamelib.engine.Game (pygamelib.assets.graphics.BoxDrawings method), 458 attribute), 48 update_sprite() (pygamelib.board_items.ActionableTile UP LIGHT AND HORIZONTAL HEAVY method), 223 (pygamelib.assets.graphics.BoxDrawings update_sprite() (pygamelib.board_items.BoardComplexItem attribute), 48 method), 232 UP_LIGHT_AND_LEFT_DOWN_HEAVY update_sprite() (pygamelib.board_items.ComplexDoor (pygamelib.assets.graphics.BoxDrawings method), 282 attribute), 48 update_sprite() (pygamelib.board_items.ComplexNPC UP LIGHT AND LEFT HEAVY *method*), 292 (pygamelib.assets.graphics.BoxDrawings update_sprite() (pygamelib.board_items.ComplexPlayer attribute), 48 *method*), 301 UP_LIGHT_AND_RIGHT_DOWN_HEAVY update_sprite() (pygamelib.board_items.ComplexTreasure (pygamelib.assets.graphics.BoxDrawings *method*), 310 attribute), 48 update_sprite() (pygamelib.board_items.ComplexWall UP_LIGHT_AND_RIGHT_HEAVY *method*), 319 (pygamelib.assets.graphics.BoxDrawings update_sprite() (pygamelib.board_items.TextItem attribute), 48 method), 405 UP_POINTING_TRIANGLE_WITH_LEFT_HALF_BLACMpdate_sprite() (pygamelib.board items.Tile (pygamelib.assets.graphics.GeometricShapes method), 414

678 Index

UPPER HALF BLOCK (pygamelib.assets.graphics.Blocks

attribute), 38	VAMPIRE (pygamelib.assets.graphics.Models attribute), 175
UPPER_HALF_CIRCLE	1,0
(pygamelib.assets.graphics.GeometricShapes	vcenter (pygamelib.engine.Screen attribute), 479
attribute), 55	Vector2D (class in pygamelib.base), 202
UPPER_HALF_INVERSE_WHITE_CIRCLE	VERTICAL_DOUBLE_AND_HORIZONTAL_SINGLE
(pygamelib.assets.graphics.GeometricShapes attribute), 55	(pygamelib.assets.graphics.BoxDrawings attribute), 48
UPPER_LEFT_OR_LOWER_RIGHT_CURLY_BRACKET	_WERTIONL_DOUBLE_AND_LEFT_SINGLE
(pygamelib.assets.graphics.MiscTechnicals attribute), 76	(pygamelib.assets.graphics.BoxDrawings attribute), 48
UPPER_LEFT_QUADRANT_CIRCULAR_ARC	VERTICAL_DOUBLE_AND_RIGHT_SINGLE
(pygamelib.assets.graphics.GeometricShapes attribute), 55	(pygamelib.assets.graphics.BoxDrawings attribute), 48
UPPER_LEFT_TRIANGLE	VERTICAL_HEAVY_AND_HORIZONTAL_LIGHT
(pygamelib.assets.graphics.GeometricShapes	(pygamelib.assets.graphics.BoxDrawings
attribute), 55	attribute), 48
UPPER_ONE_EIGHTH_BLOCK	VERTICAL_HEAVY_AND_LEFT_LIGHT
(pygamelib.assets.graphics.Blocks attribute), 38	(pygamelib.assets.graphics.BoxDrawings attribute), 48
UPPER_RIGHT_OR_LOWER_LEFT_CURLY_BRACKET	
(pygamelib.assets.graphics.MiscTechnicals attribute), 76	(pygamelib.assets.graphics.BoxDrawings attribute), 48
UPPER_RIGHT_QUADRANT_CIRCULAR_ARC	VERTICAL_LIGHT_AND_HORIZONTAL_HEAVY
(pygamelib.assets.graphics.GeometricShapes	(pygamelib.assets.graphics.BoxDrawings
attribute), 55	attribute), 48
UPPER_RIGHT_TRIANGLE	VERTICAL_LIGHT_AND_LEFT_HEAVY
(pygame lib. assets. graphics. Geometric Shapes	(pygamelib.assets.graphics.BoxDrawings
attribute), 55	attribute), 48
${\tt UPSIDE_DOWN_FACE}\ (pygamelib. assets. graphics. Model for the control of th$	
attribute), 174	(pygamelib.assets.graphics.BoxDrawings
UPWARDS_BUTTON (pygamelib.assets.graphics.Models	attribute), 48
attribute), 174	VERTICAL_LINE_EXTENSION
user_input (pygamelib.gfx.ui.ColorPickerDialog attribute), 515	(pygamelib.assets.graphics.MiscTechnicals attribute), 76
user_input (pygamelib.gfx.ui.Dialog attribute), 517	VERTICAL_LINE_WITH_MIDDLE_DOT
user_input (pygamelib.gfx.ui.FileDialog attribute), 519	(pygamelib.assets.graphics.MiscTechnicals attribute), 76
<pre>user_input (pygamelib.gfx.ui.GridSelectorDialog at-</pre>	VERTICAL_SINGLE_AND_HORIZONTAL_DOUBLE
tribute), 521	(pygamelib.assets.graphics.BoxDrawings
user_input (pygamelib.gfx.ui.LineInputDialog	attribute), 48
attribute), 525	VERTICAL_SINGLE_AND_LEFT_DOUBLE
user_input (pygamelib.gfx.ui.MessageDialog at- tribute), 537	(pygamelib.assets.graphics.BoxDrawings attribute), 48
user_input (pygamelib.gfx.ui.MultiLineInputDialog attribute), 540	VERTICAL_SINGLE_AND_RIGHT_DOUBLE (pygamelib.assets.graphics.BoxDrawings
user_input (pygamelib.gfx.ui.ProgressDialog at- tribute), 544	<pre>attribute), 49 vertical_spacing (pygamelib.gfx.core.Font at-</pre>
V	tribute), 486 VERTICAL_TRAFFIC_LIGHT
value (pygamelib.gfx.ui.ProgressBar attribute), 542	(pygamelib.assets.graphics.Models attribute),
value (pygametib.gfx.ui.ProgressDialog attribute), 542	175
value() (pygamelib.engine.Inventory method), 467	VIBRATION_MODE (pygamelib.assets.graphics.Models
values() (pygametis.trgue:wettacty metitocty, 107 values() (pygametis.trgue:wettacty metitocty, 107 values()	attribute), 175
method), 489	$\verb"VICTORY_HAND" (pygame lib. assets. graphics. Models \ at-order assets) and the substitution of the property of the propert$

tribute), 175

tribute), 175 WAVY_LINE (pygamelib.assets.graphics.MiscTechnicals VIDEO_CAMERA (pygamelib.assets.graphics.Models atattribute), 76 tribute), 175 WAXING CRESCENT MOON (pygamelib.assets.graphics.Models attribute), VIDEO_GAME (pygamelib.assets.graphics.Models attribute), 175 VIDEOCASSETTE (pygamelib.assets.graphics.Models WAXING GIBBOUS MOON (pygamelib.assets.graphics.Models attribute), attribute), 175 VIEWDATA_SQUARE (pygamelib.assets.graphics.MiscTechnicals attribute), 76 WEARY CAT (pygamelib.assets.graphics.Models VIOLIN (pygamelib.assets.graphics.Models attribute), tribute), 175 (pygame lib. assets. graphics. ModelsWEARY_FACE (pygamelib.assets.graphics.Models attribute), VIRGO attribute), 175 WEDDING (pygamelib.assets.graphics.Models attribute), VOLCANO (pygamelib.assets.graphics.Models attribute), 175 175 (pygamelib.assets.graphics.Models attribute), WHALE VOLLEYBALL (pygamelib.assets.graphics.Models attribute), 175 WHEEL_OF_DHARMA (pygamelib.assets.graphics.Models VS BUTTON (pygamelib.assets.graphics.Models attribute), 175 WHEELCHAIR SYMBOL tribute), 175 VULCAN_SALUTE (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.Models attribute), attribute), 175 white () (pygamelib.base.Text static method), 202 W (pygamelib.base.Text white_bright() static method), 202 WAFFLE (pygamelib.assets.graphics.Models attribute), 175 WHITE_BULLET (pygamelib.assets.graphics.GeometricShapes attribute), 56 Wall (class in pygamelib.board_items), 423 WHITE_CANE (pygamelib.assets.graphics.Models WANING_CRESCENT_MOON attribute), 175 (pygamelib.assets.graphics.Models attribute), WHITE_CIRCLE (pygamelib.assets.graphics.GeometricShapes attribute), 56 WANING GIBBOUS MOON (pygamelib.assets.graphics.Models attribute), WHITE_CIRCLE (pygamelib.assets.graphics.Models attribute), 175 WHITE_CIRCLE_WITH_LOWER_LEFT_QUADRANT warn () (pygamelib.base.Text static method), 202 (pygamelib.assets.graphics.GeometricShapes WARNING (pygamelib.assets.graphics.Models attribute), attribute), 56 175 WHITE_CIRCLE_WITH_LOWER_RIGHT_QUADRANT WASTEBASKET (pygamelib.assets.graphics.Models at-(pygamelib.assets.graphics.GeometricShapes tribute), 175 attribute), 56 WATCH (pygamelib.assets.graphics.MiscTechnicals at-WHITE_CIRCLE_WITH_UPPER_LEFT_QUADRANT tribute), 76 (pygamelib.assets.graphics.GeometricShapes (pygamelib.assets.graphics.Models attribute), WATCH attribute), 56 WHITE CIRCLE WITH UPPER RIGHT QUADRANT WATER_BUFFALO (pygamelib.assets.graphics.Models (pygamelib.assets.graphics.GeometricShapes attribute), 175 attribute), 56 WATER_CLOSET (pygamelib.assets.graphics.Models at- $\verb|WHITE_DIAMOND| (pygamelib. assets. graphics. Geometric Shapes$ tribute), 175 (pygamelib.assets.graphics.Models attribute), 56 WATER_WAVE WHITE_DIAMOND_CONTAINING_BLACK_SMALL_DIAMOND attribute), 175 (pygamelib.assets.graphics.GeometricShapes WATERMELON (pygamelib.assets.graphics.Models attribute), 56 attribute), 175 white_dim() (pygamelib.base.Text static method), 202 WAVING HAND (pygamelib.assets.graphics.Models at-WHITE_DOWN_POINTING_SMALL_TRIANGLE tribute), 175 (pygamelib.assets.graphics.GeometricShapes WAVY_DASH (pygamelib.assets.graphics.Models atattribute), 56

680 Index

WHITE DOWN POINTING TRIANGLE

- (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE EXCLAMATION MARK (pygamelib.assets.graphics.Models attribute),
- WHITE FLAG (pygamelib.assets.graphics.Models attribute), 176
- WHITE_FLOWER (pygamelib.assets.graphics.Models attribute), 176
- (pygamelib.assets.graphics.Models WHITE_HAIR attribute), 176
- WHITE_HEART (pygamelib.assets.graphics.Models attribute), 176
- WHITE_LARGE_SQUARE (pygamelib.assets.graphics.Models attribute), 176
- WHITE_LEFT_POINTING_POINTER (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_LEFT_POINTING_SMALL_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_LEFT_POINTING_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE MEDIUM SMALL SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_MEDIUM_SMALL_SQUARE (pygamelib.assets.graphics.Models attribute), 176
- WHITE_MEDIUM_SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE MEDIUM SOUARE (pygamelib.assets.graphics.Models attribute), 176
- WHITE PARALLELOGRAM (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE QUESTION MARK (pygamelib.assets.graphics.Models attribute),
- (pygamelib.gfx.core.Sprixel white_rect() class method), 505
- white_rect() (pygamelib.gfx.particles.ParticleSprixel WHITE_UP_POINTING_SMALL_TRIANGLE class method), 582
- WHITE_RECTANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 56 attribute), 56
- WHITE_RIGHT_POINTING_POINTER (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE RIGHT POINTING SMALL TRIANGLE (pygamelib.assets.graphics.GeometricShapes

- attribute), 56
- WHITE RIGHT POINTING TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE SMALL SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE SMALL SQUARE (pygamelib.assets.graphics.Models attribute), 176
- WHITE_SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- white_square() (pygamelib.gfx.core.Sprixel class method), 505
- white_square() (pygamelib.gfx.particles.ParticleSprixel class method), 582
- WHITE_SQUARE_BUTTON (pygamelib.assets.graphics.Models attribute),
- WHITE SQUARE CONTAINING BLACK SMALL SQUARE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_SQUARE_WITH_CENTRE_VERTICAL_LINE (pygamelib.assets.graphics.MiscTechnicals attribute), 76
- WHITE SQUARE WITH LOWER LEFT QUADRANT (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_SQUARE_WITH_LOWER_RIGHT_QUADRANT (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_SQUARE_WITH_ROUNDED_CORNERS (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE SOUARE WITH UPPER LEFT OUADRANT (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_SQUARE_WITH_UPPER_RIGHT_QUADRANT (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE SQUARE WITH VERTICAL BISECTING LINE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- $\verb"WHITE_TRAPEZIUM" (pygamelib. assets. graphics. Misc Technicals$ attribute), 76
- (pygamelib.assets.graphics.GeometricShapes
- WHITE_UP_POINTING_TRIANGLE (pygamelib.assets.graphics.GeometricShapes attribute), 56
- WHITE_UP_POINTING_TRIANGLE_WITH_DOT (pygamelib.assets.graphics.GeometricShapes attribute), 56

WHITE_	_VERTICAL_RECTANGLE	tribute), 176
	(pygamelib.assets.graphics.GeometricShapes attribute), 56	WINDOW (pygamelib.assets.graphics.Models attribute), 176
width	(pygamelib.board_items.Actionable attribute), 214	WINE_GLASS (pygamelib.assets.graphics.Models attribute), 176
width	(pygamelib.board_items.ActionableTile at- tribute), 223	WINKING_FACE (pygamelib.assets.graphics.Models attribute), 176
width	(pygamelib.board_items.BoardComplexItem attribute), 233	WINKING_FACE_WITH_TONGUE (pygamelib.assets.graphics.Models attribute),
width	(pygamelib.board_items.BoardItem attribute), 250	176 WOLF (pygamelib.assets.graphics.Models attribute), 176
width	(pygamelib.board_items.BoardItemComplexCompo attribute), 240	
width		WOMAN_AND_MAN_HOLDING_HANDS (pygamelib.assets.graphics.Models attribute),
width	(pygamelib.board_items.Camera attribute), 265	176
	(pygamelib.board_items.Character attribute), 274	WOMAN_DANCING (pygamelib.assets.graphics.Models attribute), 176
width	(pygamelib.board_items.ComplexDoor attribute),	WOMAN_WITH_HEADSCARF
width	283 (pygamelib.board_items.ComplexNPC attribute),	(pygamelib.assets.graphics.Models attribute), 176
	292	WOMANS_BOOT (pygamelib.assets.graphics.Models at-
width	(pygamelib.board_items.ComplexPlayer at-	tribute), 176
	tribute), 301	WOMANS_CLOTHES (pygamelib.assets.graphics.Models
width	(pygamelib.board_items.ComplexTreasure	attribute), 176
d+b	attribute), 310	WOMANS_HAT (pygamelib.assets.graphics.Models attribute), 176
width	(pygamelib.board_items.ComplexWall attribute), 319	WOMANS_SANDAL (pygamelib.assets.graphics.Models
width	(pygamelib.board_items.Door attribute), 327	attribute), 176
	(pygamelib.board_items.GenericActionableStructus	
	attribute), 335	(pygamelib.assets.graphics.Models attribute),
width	(pygamelib.board_items.GenericStructure	176
	attribute), 351	WOMENS_ROOM (pygamelib.assets.graphics.Models at-
width	(pygamelib.board_items.GenericStructureComplex)	=
	attribute), 343	WOOD (pygamelib.assets.graphics.Models attribute), 176
width	(pygamelib.board_items.Immovable attribute), 359	WOOZY_FACE (pygamelib.assets.graphics.Models attribute), 176
	(pygamelib.board_items.Movable attribute), 367	WORLD_MAP (pygamelib.assets.graphics.Models at-
	(pygamelib.board_items.NPC attribute), 376	tribute), 176
	(pygamelib.board_items.Player attribute), 384	WORM (pygamelib.assets.graphics.Models attribute), 176
	(pygamelib.board_items.Projectile attribute), 396	WORRIED_FACE (pygamelib.assets.graphics.Models at-
	(pygamelib.board_items.TextItem attribute), 405 (pygamelib.board_items.Tile attribute), 414	tribute), 176
	(pygametib.board_items.Treasure attribute), 422	WRAPPED_GIFT (pygamelib.assets.graphics.Models attribute), 176
	(pygamelib.board_items.Wall attribute), 430	WRENCH (pygamelib.assets.graphics.Models attribute),
	(pygamelib.engine.Board attribute), 442	176
	(pygamelib.engine.Screen attribute), 479	WRITING_HAND (pygamelib.assets.graphics.Models at-
	(pygamelib.gfx.core.Sprite attribute), 498	tribute), 177
	(pygamelib.gfx.ui.Box attribute), 513	
WILTE	O_FLOWER (pygamelib.assets.graphics.Models	X
	attribute), 176	x (pygamelib.base.Vector2D attribute), 205
WIND_C		$\times (pygamelib.gfx.particles.CircleEmitter\ attribute), 550$
T.T	attribute), 176	x (pygamelib.gfx.particles.ColorParticle attribute), 556
WIND_H	FACE (pygamelib.assets.graphics.Models at-	

(pygamelib.gfx.particles.ColorPartitionParticle tribute), 561 x (pygamelib.gfx.particles.Particle attribute), 575 Ζ x (pygamelib.gfx.particles.ParticleEmitter attribute), 568 (pygamelib.gfx.particles.PartitionParticle attribute), (pygamelib.gfx.particles.RandomColorParticle tribute), 594 ${\color{blue} \times (pygamelib.gfx.particles.RandomColorPartitionParticle} \\$ attribute), 599 X_IN_A_RECTANGLE_BOX (pygamelib.assets.graphics.MiscTechnicals attribute), 76 Y y (pygamelib.base.Vector2D attribute), 205 y (pygamelib.gfx.particles.CircleEmitter attribute), 550 y (pygamelib.gfx.particles.ColorParticle attribute), 556 (pygamelib.gfx.particles.ColorPartitionParticle attribute), 561 y (pygamelib.gfx.particles.Particle attribute), 575 y (pygamelib.gfx.particles.ParticleEmitter attribute), 568 (pygamelib.gfx.particles.PartitionParticle attribute), 589 (pygamelib.gfx.particles.RandomColorParticle tribute), 594 y (pygamelib.gfx.particles.RandomColorPartitionParticle attribute), 599 YARN (pygamelib.assets.graphics.Models attribute), 177 YAWNING_FACE (pygamelib.assets.graphics.Models attribute), 177 yellow() (pygamelib.base.Text static method), 202 (pygamelib.base.Text yellow bright() static *method*), 202 YELLOW_CIRCLE (pygamelib.assets.graphics.Models attribute), 177 yellow_dim() (pygamelib.base.Text static method), 202 YELLOW_HEART (pygamelib.assets.graphics.Models attribute), 177 yellow_rect() (pygamelib.gfx.core.Sprixel class *method*), 506 yellow_rect() (pygamelib.gfx.particles.ParticleSprixel class method), 583 YELLOW SQUARE (pygamelib.assets.graphics.Models attribute), 177 yellow_square() (pygamelib.gfx.core.Sprixel class method), 506yellow_square() (pygamelib.gfx.particles.ParticleSprixel class method), 583 YEN BANKNOTE (pygamelib.assets.graphics.Models attribute), 177 (pygamelib.assets.graphics.Models YIN_YANG

tribute), 177

YO_YO (pygamelib.assets.graphics.Models attribute),

ZANY_FACE (pygamelib.assets.graphics.Models attribute), 177

ZEBRA (pygamelib.assets.graphics.Models attribute),
177

ZIPPER_MOUTH_FACE

(pygamelib.assets.graphics.Models attribute), 177

ZOMBIE (pygamelib.assets.graphics.Models attribute),
177

ZZZ (pygamelib.assets.graphics.Models attribute), 177