hac-game-lib Documentation

Release 1.1.1

Arnaud Dupuis

Jul 18, 2020

Contents:

CHAPTER 1

Board

This module contains the Board class. It is the base class for all levels.

class gamelib.Board.**Board**(***kwargs*)

Bases: object

A class that represent a game board.

The board is being represented by a square matrix. For the moment a board only support one player.

The Board object is the base object to build a level : you create a Board and then you add BoardItems (or objects derived from BoardItem).

Parameters

- **name** (*str*) – the name of the Board
- **size** (*list*) – array [width,height] with width and height being int. The size of the board.
- **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 *gamelib.BoardItem.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 columns.

check_sanity()
Check the board sanity.

This is essentially an internal method called by the constructor.

clear_cell(*row*, *column*)
Clear cell (row, column)

This method clears a cell, meaning it position a void_cell BoardItemVoid at these coordinates.

Parameters

- **row** (*int*) – The row of the item to remove
- **column** (*int*) – The column of the item to remove

Example:

```
myboard.clear_cell(3,4)
```

Warning: This method does not check the content before, it *will* overwrite the content.

display()
Display the entire board.

This method display the Board (as in print()), taking care of displaying the borders, and everything inside.

It uses the __str__ method of the item, which by default is BoardItem.model. If you want to override this behavior you have to subclass BoardItem.

display_around(*object*, *row_radius*, *column_radius*)
Display only a part of the board.

This method behaves like display() but only display a part of the board around an object (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

- **object** (*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.

get_immovables(***kwargs*)
Return a list of all the Immovable objects in the Board.

See *gamelib.Immovable.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 *gamelib.Movable.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")
```

init_board()

Initialize the board with BoardItemVoid that uses ui_board_void_cell as model.

Example:

```
myboard.init_board()
```

init_cell(*row*, *column*)

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)
```

item(*row*, *column*)

Return the item at the row, column position if within board's boundaries.

Return type *gamelib.BoardItem.BoardItem*

Raises ***HacOutOfBoardBoundException*** – if row or column are out of bound.

move(*item*, *direction*, *step*)

Move an item in the specified direction for a number of steps.

Example:

```
board.move(player,Constants.UP,1)
```

Parameters

- **item** (*gamelib.Movable.Movable*) – an item to move (it has to be a subclass of Movable)
- **direction** (*gamelib.Constants*) – 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 HacObjectIsNotMovableException exception (see *gamelib.HacExceptions.HacObjectIsNotMovableException*).

Important: if the move is successfull, an empty BoardItemVoid (see *gamelib.BoardItem.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.

Note: It could be interesting here, instead of relying on storing the overlapping item in a property of a Movable (*gamelib.Movable.Movable*) object, to have another dimension on the board matrix to push and pop objects on a cell. Only the first item would be rendered and it would avoid the complicated and error prone logic in this method. If anyone feel up to the challenge, PR are welcome ;-).

Todo: check all types!

---

place_item(*item*, *row*, *column*)

Place an item at coordinates row and column.

If row or column are our of the board boundaries, an HacOutOfBoardBoundException is raised.

If the item is not a subclass of BoardItem, an HacInvalidTypeException

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.

CHAPTER 2

BoardItem

This module contains the basic board items classes (regular and void items).

BoardItem(**kwargs)	Base class for any item that will be placed on a Board.
BoardItemVoid(**kwargs)	A class that represent a void cell.

2.1 BoardItem

class gamelib.BoardItem.**BoardItem**(***kwargs*)

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** (*array*) – 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 '*'.
- **parent** – The parent object of the board item. Usually a Board or Game object.

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	This is a virtual method that must be implemented in deriving classes.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
overlappable()	This is a virtual method that must be implemented in deriving class.
pickable()	This is a virtual method that must be implemented in deriving class.
size()	This is a virtual method that must be implemented in deriving class.
store_position(row, column)	Store the BoardItem position for self access.

2.2 BoardItemVoid

class gamelib.BoardItem.**BoardItemVoid**(***kwargs*)

A class that represent a void cell.

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	This is a virtual method that must be implemented in deriving classes.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
overlappable()	A BoardItemVoid is obviously overlappable (so player and NPC can walk over).
pickable()	A BoardItemVoid is not pickable, therefor this method return false.
size()	This is a virtual method that must be implemented in deriving class.
store_position(row, column)	Store the BoardItem position for self access.

class gamelib.BoardItem.**BoardItem**(***kwargs*)

Bases: object

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** (*array*) – 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 '*'.
- **parent** – The parent object of the board item. Usually a Board or Game object.

can_move()
This is a virtual method that must be implemented in deriving classes. This method has to return True or False. This represent the capacity for a BoardItem to be moved by the Board.

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

Return type str

display()
Print the model WITHOUT carriage return.

overlappable()
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 BoardItem to be overlapped by another BoardItem.

pickable()
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 BoardItem to be pick-up by player or NPC.

size()
This is a virtual method that must be implemented in deriving class. This method has to return an integer. This represent the size of the BoardItem. It is used for example to evaluate the space taken in the inventory.

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.BoardItem.**BoardItemVoid**(**kwargs*)
Bases: *gamelib.BoardItem.BoardItem*

A class that represent a void cell.

overlappable()
A BoardItemVoid is obviously overlappable (so player and NPC can walk over).

Returns True

pickable()
A BoardItemVoid is not pickable, therefor this method return false.

Returns False

CHAPTER 3

Characters

This module contains the base classes for both playable and non playable characters.

Character(**kwargs)	A base class for a character (playable or not)
NPC(**kwargs)	A class that represent a non playable character controlled by the computer.
Player(**kwargs)	A class that represent a player controlled by a human.

3.1 Character

class gamelib.Characters.**Character**(***kwargs*)

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 (*gamelib.Game*) engine.

__init__(***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(**kwargs)	Initialize self.

3.2 NPC

class gamelib.Characters.**NPC**(***kwargs*)
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:

- *gamelib.Characters.Character*
- *gamelib.Movable.Movable*
- *gamelib.BoardItem.BoardItem*

Parameters **actuator** (*gamelib.Actuators.Actuator*) – An actuator, it can be any class but it need to implement gamelib.Actuator.Actuator.

Example:

```
mynpc = NPC(name='Idiot McStupid', type='dumb_enemy')
mynpc.step = 1
mynpc.actuator = RandomActuator()
```

__init__(***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(**kwargs)	Initialize self.
can_move()	Movable implements can_move().
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
has_inventory()	Define if the NPC has an inventory.
overlappable()	Define if the NPC is overlappable.
pickable()	Define if the NPC is pickable.
size()	This is a virtual method that must be implemented in deriving class.
store_position(row, column)	Store the BoardItem position for self access.

3.3 Player

class gamelib.Characters.**Player**(***kwargs*)

A class that represent a player controlled by a human. It accepts all the parameters from *Character* and is a *Movable*.

Note: If no inventory is passed as parameter a default one is created.

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	Movable implements can_move().
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
has_inventory()	This method returns True (a player has an inventory).
overlappable()	This method returns false (a player cannot be overlapped).
pickable()	This method returns False (a player is obviously not pickable).
size()	This is a virtual method that must be implemented in deriving class.
store_position(row, column)	Store the BoardItem position for self access.

class gamelib.Characters.**Character**(***kwargs*)

Bases: object

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 (*gamelib.Game*) engine.

class gamelib.Characters.**NPC**(***kwargs*)

Bases: *gamelib.Movable.Movable*, *gamelib.Characters.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:

- *gamelib.Characters.Character*
- *gamelib.Movable.Movable*
- *gamelib.BoardItem.BoardItem*

Parameters **actuator** (*gamelib.Actuators.Actuator*) – An actuator, it can be any class but it need to implement gamelib.Actuator.Actuator.

Example:

```
mynpc = NPC(name='Idiot McStupid', type='dumb_enemy')
mynpc.step = 1
mynpc.actuator = RandomActuator()
```

can_move()

Movable implements can_move().

Returns True

Return type Boolean

debug_info()

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

Return type str

display()

Print the model WITHOUT carriage return.

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 :(")
```

overlappable()

Define if the NPC is overlappable.

Obviously this method also always return False.

Returns False

Return type Boolean

Example:

```
if mynpc.overlappable():
    Utils.warn("Something is fishy, that NPC is overlappable but"
        "is not a Ghost...")
```

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...")
```

size()
This is a virtual method that must be implemented in deriving class. This method has to return an integer. This represent the size of the BoardItem. It is used for example to evaluate the space taken in the inventory.

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.Characters.**Player**(***kwargs*)
Bases: *gamelib.Movable.Movable*, *gamelib.Characters.Character*

A class that represent a player controlled by a human. It accepts all the parameters from *Character* and is a *Movable*.

Note: If no inventory is passed as parameter a default one is created.

---

can_move()
Movable implements can_move().

Returns True

Return type Boolean

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

Return type str

display()
Print the model WITHOUT carriage return.

has_inventory()
This method returns True (a player has an inventory).

overlappable()
This method returns false (a player cannot be overlapped).

Note: If you wish your player to be overlappable, you need to inherit from that class and re-implement overlappable().

---

pickable()
This method returns False (a player is obviously not pickable).

size()
This is a virtual method that must be implemented in deriving class. This method has to return an integer. This represent the size of the BoardItem. It is used for example to evaluate the space taken in the inventory.

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

CHAPTER 4

Constants

Accessible constants are the following:

General purpose:

- HAC_GAME_LIB_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
- NONE_AUTHORIZED

UI positions:

- POS_TOP
- POS_BOTTOM

- ORIENTATION_HORIZONTAL
- ORIENTATION_VERTICAL

Actions states (for Actuators for example):

- RUNNING
- PAUSED
- STOPPED

CHAPTER 5

Game

class gamelib.Game.**Game**(*name='Game'*, *boards={}*, *menu={}*, *current_level=None*, *enable_partial_display=False*, *partial_display_viewport=None*)

Bases: `object`

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 *gamelib.Board.Board*), associate it to level, takes care of level changing, etc.

Parameters

- **name** (*str*) – The Game name.
- **boards** (*dict*) – A dictionnary of boards with the level number as key and a board reference as value.
- **menu** (*dict*) – A dictionnary of menus with a category (str) as key and another dictionnary (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()*.

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.

actuate_npcs(*level_number*)
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.)

Parameters **level_number** – The number of the level to actuate NPCs in.

Example:

```
mygame.actuate_npcs(1)
```

Note: This method only move NPCs when their actuator state is RUNNING. If it is PAUSED or STOPPED, theNPC is not moved.

actuate_projectiles(*level_number*)
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.

Parameters **level_number** – The number of the level to actuate Projectiles in.

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.

add_board(*level_number*, *board*)
Add a board for the level number.

This method associate a Board (*gamelib.Board.Board*) to a level number.

Example:

```
game.add_board(1,myboard)
```

Parameters

- **level_number** (*int*) – the level number to associate the board to.
- **board** (gamelib.Board.Board) – a Board object corresponding to the level number.

Raises ***HacInvalidTypeException*** – 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.

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*)

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** (gamelib.Characters.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 HacInvalidTypeException exception is raised.

Important: If the NPC does not have an actuator, this method is going to affect a gamelib.Actuators.SimpleActuators.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 HacInvalidTypeException exception is raised.

Important: If the Projectile does not have an actuator, this method is going to affect gamelib.Actuators.SimpleActuators.RandomActuator(moveset=[RIGHT]) to projectile.actuator. And if projectile.step == None, this method sets it to 1.

---

animate_items(*level_number*)

That method goes through all the BoardItems of a given map and call Animation.next_frame() :param level_number: The number of the level to animate items in. :type level_number: int

Raise *gamelib.HacExceptions.HacInvalidLevelException*
class:*gamelib.HacExceptions.HacInvalidTypeException*

Example:

```
mygame.animate_items(1)
```

change_level(*level_number*)

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 ***HacInvalidTypeException*** – If parameter is not an int.

clear_screen()

Clear the whole screen (i.e: remove everything written in terminal)

config(*section='main'*)

Get the content of a previously loaded configuration section.

Parameters **section** (*str*) – The name of the section.

Example:

```
if mygame.config('main')['hgl-version-required'] < 10100:
    print('The hac-game-lib version 1.1.0 or greater is required.')
    exit()
```

create_config(*section*)

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()
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 HacException exception is raised with an invalid_level error.

delete_menu_category(*category=None*)
Delete an entire category from the menu.

That function removes the entire list of messages that are attached to the category.

Parameters **category** (*str*) – The category to delete.

Raises ***HacInvalidTypeException*** – 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)
```

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().

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=10010000*, *paginate=10*)
Display the menu.

This method display the whole menu for a given category.

Parameters

- **category** (*str*) – The category to display. **Mandatory** parameter.
- **orientation** (gamelib.Constants.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)
```

display_player_stats (*life_model='\x1b[41m\x1b[0m'*, *void_model='\x1b[40m\x1b[0m'*)
Display the player name and health.

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*)
This method returns the board associated with a level number. :param level_number: The number of the level. :type level_number: int

Raises ***HacInvalidTypeException*** – if the level_number is not an int.

Example:

```
level1_board = mygame.get_board(1)
```

get_menu_entry (*category*, *shortcut*)
Get an entry of the menu.

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)
```

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
- **lvl_number** (*int*) – The level number to associate the board to. Default is 0.

Returns a newly created board (see *gamelib.Board.Board*)

Example:

```
mynewboard = game.load_board( 'awesome_level.json', 1 )
game.change_level( 1 )
```

load_config(*filename*, *section='main'*)

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 configparser 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*)

Easy wrapper for Board.move().

Example:

```
mygame.move_player(Constants.RIGHT,1)
```

neighbors(*radius=1*, *object=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** (gamelib.BoardItem.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 ***HacInvalidTypeException*** – 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]})')
```

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)
```

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

- ***HacInvalidTypeException*** – If any parameter is not of the right type
- ***HacInvalidLevelException*** – 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.

save_config(*section=None*, *filename=None*, *append=False*)

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')
```

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()
```

stop()

Set the game engine state to STOPPED.

Example:

```
mygame.stop()
```

update_menu_entry(*category*, *shortcut*, *message*, *data=None*)

Update an entry of the menu.

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)
```

CHAPTER 6

HacExceptions

This module regroup all the specific exceptions of the library. The idea behind most exceptions is to provide more context and info that the standard exceptions.

exception gamelib.HacExceptions.**HacException**(*error*, *message*)
Bases: Exception

Exception raised for non specific errors in HAC-GAME-LIB.

exception gamelib.HacExceptions.**HacInvalidLevelException**(*message*)
Bases: Exception

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

exception gamelib.HacExceptions.**HacInvalidTypeException**(*message*)
Bases: Exception

Exception raised for invalid types.

exception gamelib.HacExceptions.**HacInventoryException**(*error*, *message*)
Bases: Exception

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

exception gamelib.HacExceptions.**HacObjectIsNotMovableException**(*message*)
Bases: Exception

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

exception gamelib.HacExceptions.**HacOutOfBoardBoundException**(*message*)
Bases: Exception

Exception for out of the board's boundaries operations.

CHAPTER 7

Immovable

This module contains the Immovable and Actionable classes.

Immovable(**kwargs)	This class derive BoardItem and describe an object that cannot move or be moved (like a wall).
Actionable(**kwargs)	This class derives *Immovable*.

7.1 Immovable

class gamelib.Immovable.**Immovable**(***kwargs*)

This class derive BoardItem and describe an object that cannot move or be moved (like a wall). Thus this class implements BoardItem.can_move(). However it does not implement BoardItem.pickable() or BoardItem.overlappable()

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	Return the capability of moving of an item.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
overlappable()	This is a virtual method that must be implemented in deriving class.
pickable()	This is a virtual method that must be implemented in deriving class.

Continued on next page

Table 2 – continued from previous page

restorable()	This is a virtual method that must be implemented in deriving class.
size()	Return the size of the Immovable Item.
store_position(row, column)	Store the BoardItem position for self access.

7.2 Actionable

class gamelib.Immovable.**Actionable**(***kwargs*)

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

Parameters

- **action** (*function*) – the reference to a function (Attention: no parentheses at the end of the function name).
- **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*

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.

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
activate()	This function is calling the action function with the action_parameters.
can_move()	Return the capability of moving of an item.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
overlappable()	This is a virtual method that must be implemented in deriving class.
pickable()	This is a virtual method that must be implemented in deriving class.
restorable()	This is a virtual method that must be implemented in deriving class.
size()	Return the size of the Immovable Item.
store_position(row, column)	Store the BoardItem position for self access.

class gamelib.Immovable.**Actionable**(***kwargs*)

Bases: *gamelib.Immovable.Immovable*

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

some code.

Parameters

- **action** (*function*) – the reference to a function (Attention: no parentheses at the end of the function name).
- **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*

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.

---

activate()
This function is calling the action function with the action_parameters.

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

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

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

Return type str

display()
Print the model WITHOUT carriage return.

overlappable()
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 BoardItem to be overlapped by another BoardItem.

pickable()
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 BoardItem to be pick-up by player or NPC.

restorable()
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 an Immovable BoardItem to be restored by the board if the item is overlappable and has been overlapped by another Movable (*Movable*) item.

size()
Return the size of the Immovable Item.

Returns The size of the item.

Return type int

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.Immovable.**Immovable**(***kwargs*)

Bases: *gamelib.BoardItem.BoardItem*

This class derive BoardItem and describe an object that cannot move or be moved (like a wall). Thus this class implements BoardItem.can_move(). However it does not implement BoardItem.pickable() or BoardItem.overlappable()

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

debug_info()

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

Return type str

display()

Print the model WITHOUT carriage return.

overlappable()

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 BoardItem to be overlapped by another BoardItem.

pickable()

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 BoardItem to be pick-up by player or NPC.

restorable()

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 an Immovable BoardItem to be restored by the board if the item is overlappable and has been overlapped by another Movable (*Movable*) item.

size()

Return the size of the Immovable Item.

Returns The size of the item.

Return type int

store_position(*row*, *column*)

Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

CHAPTER 8

Inventory

This module contains the Inventory class.

class gamelib.Inventory.**Inventory**(*max_size=10*, *parent=None*)
Bases: object

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.

The constructor takes only one parameter: the maximum size of the inventory. 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. Deafult value: 10.
- **parent** – The parent object (usually a BoardItem).

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.

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.size is greater than the inventory max_size)

Parameters **item** (*BoardItem*) – the item you want to add

Raises HacInventoryException, HacInvalidTypeException

Example:

```
item = Treasure(model=Sprites.MONEY_BAG,size=2,name='Money bag')
try:
    mygame.player.inventory.add_item(item)
expect HacInventoryException 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)
```

Warning: if you try to add more than one item with the same name (or if the name is empty), this function will automatically change the name of the item by adding a UUID to it.

delete_item(*name*)

Delete the item corresponding to the name given in argument.

Parameters **name** (*str*) – the name of the item you want to delete.

Note: in case an execpetion is raised, the error will be 'no_item_by_that_name' and the message is giving the specifics.

See also:

gamelib.HacExceptions.HacInventoryException.

Example:

```
life_container = mygame.player.inventory.get_item('heart_1')
if isinstance(life_container,GenericActionableStructure):
    life_container.action(life_container.action_parameters)
    mygame.player.inventory.delete_item('heart_1')
```

empty()

Empty the inventory Example:

```
if inventory.size() > 0:
    inventory.empty()
```

get_item(*name*)

Return the 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*

Raises HacInventoryException

Note: in case an execpetion is raised, the error will be 'no_item_by_that_name' and the message is giving the specifics.

See also:

gamelib.HacExceptions.HacInventoryException.

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.

---

items_name()

Return the list of all items names in the inventory.

Returns a list of string representing the items names.

Return type list

search(*query*)

Search for objects in the inventory.

All objects that matches the query are going to be returned. :param query: the query that items in the inventory have to match to be returned :type name: str :returns: a table of BoardItems. :rtype: list

Example:

```
for item in game.player.inventory.search('mighty'):
    print(f"This is a mighty item: {item.name}")
```

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:

```
print(f"Inventory: {mygame.player.inventory.size()}/"
"{mygame.player.inventory.max_size}")
```

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
```

CHAPTER 9

Movable

This module contains the Movable class. It can potentially hold more movement related classes.

Movable(**kwargs)	A class representing BoardItem capable of movements.
Projectile([name, direction, step, range, ...])	A class representing a projectile type board item.

9.1 Movable

class gamelib.Movable.**Movable**(***kwargs*)

A class representing BoardItem capable of movements.

Movable subclasses `BoardItem`.

Parameters **step** (*int*) – the amount of cell a movable can cross in one turn.

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()

This class contains a private member called _overlapping. This private member is used to store the reference to an overlappable object while a movable occupy its position. The Board then restore the overlapped object. You should let the Board class take care of that.

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	Movable implements can_move().

Continued on next page

Table 2 – continued from previous page

debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
has_inventory()	This is a virtual method that must be implemented in deriving class.
overlappable()	This is a virtual method that must be implemented in deriving class.
pickable()	This is a virtual method that must be implemented in deriving class.
size()	This is a virtual method that must be implemented in deriving class.
store_position(row, column)	Store the BoardItem position for self access.

9.2 Projectile

class gamelib.Movable.**Projectile**(*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*, **args*)

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.
- **args** – extra parameters to pass to hit_callback.
- **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:

```
fireball = Projectile(
                        name="fireball",
                        model=Utils.red_bright(black_circle),
                        hit_model=Sprites.EXPLOSION,
                    )
fireball.set_direction(Constants.RIGHT)
my_game.add_projectile(1, fireball,
                       my_game.player.pos[0], my_game.player.pos[1] + 1)
```

__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*, **args*)
Initialize self. See help(type(self)) for accurate signature.

Methods

__init__([name, direction, step, range, ...])	Initialize self.
add_directional_animation(direction, animation)	Add an animation for a specific direction.
add_directional_model(direction, model)	Add an model for a specific direction.
can_move()	Movable implements can_move().
debug_info()	Return a string with the list of the attributes and their current value.
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.
has_inventory()	Projectile cannot have inventory by default.

Continued on next page

Table 3 – continued from previous page

hit(objects)	A method that is called when the projectile hit something.
overlappable()	Projectile are overlappable by default.
pickable()	This is a virtual method that must be implemented in deriving class.
remove_directional_animation(direction)	Remove an animation for a specific direction.
remove_directional_model(direction)	Remove the model for a specific direction.
restorable()	We assume that by default, Projectiles are restorable.
set_direction(direction)	Set the direction of a projectile
size()	This is a virtual method that must be implemented in deriving class.
store_position(row, column)	Store the BoardItem position for self access.

class gamelib.Movable.**Movable**(***kwargs*)

Bases: *gamelib.BoardItem.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.

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()

This class contains a private member called _overlapping. This private member is used to store the reference to an overlappable object while a movable occupy its position. The Board then restore the overlapped object. You should let the Board class take care of that.

can_move()

Movable implements can_move().

Returns True

Return type Boolean

debug_info()

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

Return type str

display()

Print the model WITHOUT carriage return.

has_inventory()

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.

overlappable()

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 BoardItem to be overlapped by another BoardItem.

pickable()

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 BoardItem to be pick-up by player or NPC.

size()

This is a virtual method that must be implemented in deriving class. This method has to return an integer. This represent the size of the BoardItem. It is used for example to evaluate the space taken in the inventory.

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.Movable.**Projectile**(*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*, **args*)

Bases: *gamelib.Movable.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.
- **args** – extra parameters to pass to hit_callback.
- **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:

```
fireball = Projectile(
                        name="fireball",
                        model=Utils.red_bright(black_circle),
                        hit_model=Sprites.EXPLOSION,
                    )
fireball.set_direction(Constants.RIGHT)
my_game.add_projectile(1, fireball,
                       my_game.player.pos[0], my_game.player.pos[1] + 1)
```

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, updward_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, updward_animation)
```

can_move()
Movable implements can_move().

Returns True

Return type Boolean

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

Return type str

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.

has_inventory()
Projectile cannot have inventory by default.

Returns False

Return type Boolean

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])
```

overlappable()
Projectile are overlappable by default.

Returns True

Return type Boolean

pickable()
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 BoardItem to be pick-up by player or NPC.

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)
```

restorable()
We assume that by default, Projectiles are restorable.

Returns True

Return type bool

set_direction(*direction*)
Set the direction of a projectile

This method will set a UnidirectionalActuator 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)
```

size()
This is a virtual method that must be implemented in deriving class. This method has to return an integer. This represent the size of the BoardItem. It is used for example to evaluate the space taken in the inventory.

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

CHAPTER 10

Assets.Graphics

Important: The Graphics module was introduced in version 1.1.0.

The Graphics module contains the following classes:

Sprites	List of sprites (emojis by unicode denomination)
Blocks	Block elements (unicode)
BoxDrawings	Box drawing elements (unicode)
GeometricShapes	Geometric shapes elements (unicode)

10.1 Sprites

class gamelib.Assets.Graphics.**Sprites**

List of sprites (emojis by unicode denomination)

Sprites are filtered emojis. This class does not map the entire specification. It is however a significant improvement over the gamelib.Sprites module (now deprecated). 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 =

- 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 =

- 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 =

- 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 =

- 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 = ™
- 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 =

- 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 3 – continued from previous 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 3 – 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 3 – continued from 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

Continued on next page

Table 3 – continued from previous 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 3 – continued from previous 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
COWBOY_HAT_FACE

Continued on next page

Table 3 – continued from 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 3 – continued from previous 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 3 – 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 3 – continued from previous 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 3 – continued from previous 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

Continued on next page

Table 3 – continued from 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

Continued on next page

Table 3 – continued from previous 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 3 – continued from previous 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
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 3 – continued from previous 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 3 – continued from 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
NERD_FACE
NESTING_DOLLS
NEUTRAL_FACE
NEWSPAPER
NEW_BUTTON
NEW_MOON
NEW_MOON_FACE

Continued on next page

Table 3 – continued from previous 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 3 – 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 3 – continued from previous page

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 3 – continued from 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 3 – continued from previous 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 3 – continued from previous 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

Continued on next page

Table 3 – continued from previous page

SHUFFLE_TRACKS_BUTTON
SHUSHING_FACE
SIGN_OF_THE_HORNS
SIX_OCLOCK
SIX_THIRTY
SKATEBOARD
SKIER
SKIS
SKULL
SKULL_AND_CROSSBONES
SKUNK
SLED
SLEEPING_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 3 – continued from previous page

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
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

Continued on next page

Table 3 – continued from previous page

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

Continued on next page

Table 3 – continued from previous page

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

Continued on next page

Table 3 – 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 3 – continued from previous 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

10.2 Blocks

class gamelib.Assets.Graphics.**Blocks**

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

Continued on next page

Table 5 – continued from previous page

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

10.3 BoxDrawings

class gamelib.Assets.Graphics.**BoxDrawings**

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 = └
- 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
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

Continued on next page

Table 7 – continued from previous page

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

Continued on next page

Table 7 – continued from previous page

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
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

10.4 GeometricShapes

class gamelib.Assets.Graphics.**GeometricShapes**

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 = ○

- 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

Continued on next page

Table 9 – continued from previous page

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
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
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

Continued on next page

Table 9 – continued from previous page

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_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

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 Sprites 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.

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

class gamelib.Assets.Graphics.**Blocks**
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 =

class gamelib.Assets.Graphics.**BoxDrawings**

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 = └
- 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 =

class gamelib.Assets.Graphics.**GeometricShapes**

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 = ○
- 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 =

class gamelib.Assets.Graphics.**Sprites**

List of sprites (emojis by unicode denomination)

Sprites are filtered emojis. This class does not map the entire specification. It is however a significant improvement over the gamelib.Sprites module (now deprecated). 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 =
- 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 =
- 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 =
- 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 =
- 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 = ™
- 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 =
- 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 =

CHAPTER 11

Sprites

Deprecated since version 1.1.0: Use: *gamelib.Assets.Graphics.Sprites* instead.

Sprites are simply filtered emojis. Explore this file for a complete list. All emoji codes from: https://unicode.org/emoji/charts/full-emoji-list.html

The complete list of aliased emojis is:

- COWBOY =
- DEAMON_HAPPY =
- DAEMON_ANGRY =
- SKULL =
- SKULL_CROSSBONES =
- POO =
- CLOWN =
- OGRE =
- HAPPY_GHOST =
- ALIEN =
- ALIEN_MONSTER =
- ROBOT =
- CAT =
- CAT_FACE =
- CAT_LOVE =
- CAT_WEARY =
- CAT_CRY =
- CAT_ANGRY =

- HEART =
- HEART_SPARKLING =
- HEART_BROKEN =
- HEART_ORANGE =
- HEART_YELLOW =
- HEART_GREEN =
- HEART_BLUE =
- EXPLOSION =
- DIZZY =
- DASH =
- HOLE =
- BOMB =
- BRAIN =
- BOY =
- GIRL =
- MAN =
- MAN_BEARD =
- WOMAN =
- WOMAN_BLOND =
- MAN_OLD =
- WOMAN_OLD =
- POLICE =
- SUPER_HERO =
- SUPER_VILAIN =
- MAGE =
- FAIRY =
- VAMPIRE =
- MERMAID =
- ELF =
- GENIE =
- ZOMBIE =
- PERSON_RUNNING =
- PERSON_WALKING =
- PERSON_FENCING =
- PERSON_SLEEPING =
- PERSON_YOGA =

- PERSON_BATHING =
- MONKEY =
- GORILLA =
- DOG =
- DOG_FACE =
- WOLF_FACE =
- FOX_FACE =
- RACCOON_FACE =
- LION_FACE =
- TIGER_FACE =
- HORSE_FACE =
- HORSE =
- UNICORN_FACE =
- DEER_FACE =
- COW_FACE =
- COW =
- OX =
- BUFFALO =
- PIG =
- PIG_FACE =
- RAM =
- SHEEP =
- GOAT =
- LLAMA =
- GIRAFFE =
- ELEPHANT =
- RHINOCEROS_FACE =
- MOUSE =
- RABBIT =
- CHIPMUNK =
- BAT =
- PANDA_FACE =
- TURKEY =
- CHICKEN =
- CHICK =
- EAGLE =

- DUCK =
- OWL =
- FROG_FACE =
- CROCODILE =
- TURTLE =
- LIZARD =
- SNAKE =
- DRAGON =
- DINOSAUR =
- TREX =
- WHALE =
- DOLPHIN =
- SHARK =
- OCTOPUS =
- SPIDER =
- SPIDER_WEB =
- SCORPION =
- MICROBE =
- SUNFLOWER =
- CHERRY_BLOSSOM =
- FLOWER =
- ROSE =
- TREE_PINE =
- TREE =
- TREE_PALM =
- CACTUS =
- CLOVER =
- CLOVER_LUCKY =
- CHEESE =
- MEAT_BONE =
- MEAT =
- BACON =
- EGG =
- CRAB =
- LOBSTER =
- SHRIMP =

- SQUID =
- KNIFE =
- AMPHORA =
- EARTH_GLOBE =
- WALL =
- HOUSE =
- CASTLE =
- MON =
- FOUNTAIN =
- ROCKET =
- FLYING_SAUCER =
- HOURGLASS =
- CYCLONE =
- RAINBOW =
- ZAP =
- SNOWMAN =
- COMET =
- FIRE =
- WATER_DROP =
- JACK_O_LANTERN =
- DYNAMITE =
- SPARKLES =
- GIFT =
- TROPHY =
- CROWN =
- GEM_STONE =
- CANDLE =
- LIGHT_BULB =
- BOOK_OPEN =
- SCROLL =
- MONEY_BAG =
- BANKNOTE_DOLLARS =
- BANKNOTE_EUROS =
- BANKNOTE_WINGS =
- DOLLAR =
- LOCKED =

- UNLOCKED =
- KEY =
- PICK =
- SWORD =
- SWORD_CROSSED =
- PISTOL =
- BOW =
- SHIELD =
- COFFIN =
- RADIOACTIVE =
- FLAG_GOAL =
- DOOR =

CHAPTER 12

Structures

This module contains many "helpers" classes to populate your game with structures. It contains many directly usable structures and some generic ones that can be turned in anything you like.

Wall(**kwargs)	A Wall is a specialized *Immovable* object that as unmodifiable characteristics:
Treasure(**kwargs)	A Treasure is an *Immovable* that is pickable and with a non zero value.
Door(**kwargs)	A Door is a *GenericStructure* that is not pickable, overlappable and restorable.
GenericStructure(**kwargs)	A GenericStructure is as the name suggest, a generic object to create all kind of structures.
GenericActionableStructure(**kwargs)	A GenericActionableStructure is the combination of a *GenericStructure* and an *Actionable*.

12.1 Wall

class gamelib.Structures.**Wall**(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	Return the capability of moving of an item.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
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.
restorable()	This represent the capacity for an *Immovable* Movable item.
size()	Return the size of the Immovable Item.
store_position(row, column)	Store the BoardItem position for self access.

12.2 Treasure

class gamelib.Structures.**Treasure**(***kwargs*)
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.
- **size** (*str*) – The size of 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=Sprites.MONEY_BAG,value=100,size=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__(***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(**kwargs)	Initialize self.
can_move()	Return the capability of moving of an item.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
overlappable()	This represent the capacity for a Treasure to be overlapped by player or NPC.
pickable()	This represent the capacity for a Treasure to be picked-up by player or NPC.
restorable()	This represent the capacity for a Treasure to be restored after being overlapped.
size()	Return the size of the Immovable Item.
store_position(row, column)	Store the BoardItem position for self access.

12.3 Door

class gamelib.Structures.**Door**(***kwargs*)

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.
- **size** (*str*) – The size of the door. 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=Sprites.DOOR,type='locked_door')
```

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	Return the capability of moving of an item.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
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 picked-up by player or NPC.
restorable()	This represent the capacity for an *Immovable BoardItem* (in this case a GenericStructure item) to be restored by the board if the item is overlappable and has been overlapped by another *Movable* item.
set_overlappable(val)	Make the structure overlappable or not.
set_pickable(val)	Make the structure pickable or not.
set_restorable(val)	Make the structure restorable or not.
size()	Return the size of the Immovable Item.
store_position(row, column)	Store the BoardItem position for self access.

12.4 GenericStructure

class gamelib.Structures.**GenericStructure**(***kwargs*)

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 *gamelib.Structures.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 GenericActionableStructure 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.

On top of these, this object takes all parameters of *BoardItem* and *Immovable*

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__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
can_move()	Return the capability of moving of an item.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
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 picked-up by player or NPC.
restorable()	This represent the capacity for an *Immovable BoardItem* (in this case a GenericStructure item) to be restored by the board if the item is overlappable and has been overlapped by another *Movable* item.
set_overlappable(val)	Make the structure overlappable or not.
set_pickable(val)	Make the structure pickable or not.
set_restorable(val)	Make the structure restorable or not.
size()	Return the size of the Immovable Item.
store_position(row, column)	Store the BoardItem position for self access.

12.5 GenericActionableStructure

class gamelib.Structures.**GenericActionableStructure**(***kwargs*)

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.

__init__(***kwargs*)

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

Methods

__init__(**kwargs)	Initialize self.
activate()	This function is calling the action function with the action_parameters.
can_move()	Return the capability of moving of an item.
debug_info()	Return a string with the list of the attributes and their current value.
display()	Print the model WITHOUT carriage return.
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 picked-up by player or NPC.
restorable()	This represent the capacity for an *Immovable BoardItem* (in this case a GenericStructure item) to be restored by the board if the item is overlappable and has been overlapped by another *Movable* item.
set_overlappable(val)	Make the structure overlappable or not.

Continued on next page

Table 6 – continued from previous page

set_pickable(val)	Make the structure pickable or not.
set_restorable(val)	Make the structure restorable or not.
size()	Return the size of the Immovable Item.
store_position(row, column)	Store the BoardItem position for self access.

class gamelib.Structures.**Door**(***kwargs*)

Bases: *gamelib.Structures.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.
- **size** (*str*) – The size of the door. 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=Sprites.DOOR,type='locked_door')
```

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

debug_info()

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

Return type str

display()

Print the model WITHOUT carriage return.

overlappable()

This represent the capacity for a *BoardItem* to be overlapped by player or NPC.

To set this value please use *set_overlappable()*

Returns False

Return type bool

See also:

set_overlappable()

pickable()
This represent the capacity for a BoardItem to be picked-up by player or NPC.

To set this value please use *set_pickable()*

Returns True or False

Return type bool

See also:

set_pickable()

restorable()
This represent the capacity for an *Immovable BoardItem* (in this case a GenericStructure item) to be restored by the board if the item is overlappable and has been overlapped by another *Movable* item.

The value of this property is set with *set_restorable()*

Returns False

Return type bool

See also:

set_restorable()

set_overlappable(*val*)
Make the structure overlappable or not.

Parameters **val** (*bool*) – True or False depending on the fact that the structure can be overlapped (i.e that a Player or NPC can step on it) or not.

Example:

```
myneatstructure.set_overlappable(True)
```

set_pickable(*val*)
Make the structure pickable or not.

Parameters **val** (*bool*) – True or False depending on the pickability of the structure.

Example:

```
myneatstructure.set_pickable(True)
```

set_restorable(*val*)
Make the structure restorable or not.

Parameters **val** (*bool*) – True or False depending on the restorability of the structure.

Example:

```
myneatstructure.set_restorable(True)
```

size()
Return the size of the Immovable Item.

Returns The size of the item.

Return type int

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.Structures.**GenericActionableStructure**(***kwargs*)
Bases: *gamelib.Structures.GenericStructure*, *gamelib.Immovable.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.

activate()
This function is calling the action function with the action_parameters.

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

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

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

Return type str

display()
Print the model WITHOUT carriage return.

overlappable()
This represent the capacity for a *BoardItem* to be overlapped by player or NPC.

To set this value please use *set_overlappable()*

Returns False

Return type bool

See also:

set_overlappable()

pickable()
This represent the capacity for a BoardItem to be picked-up by player or NPC.

To set this value please use *set_pickable()*

Returns True or False

Return type bool

See also:

set_pickable()

restorable()
This represent the capacity for an *Immovable BoardItem* (in this case a GenericStructure item) to be restored by the board if the item is overlappable and has been overlapped by another *Movable* item.

The value of this property is set with *set_restorable()*

Returns False

Return type bool

See also:

set_restorable()

set_overlappable(*val*)
Make the structure overlappable or not.

Parameters **val** (*bool*) – True or False depending on the fact that the structure can be overlapped (i.e that a Player or NPC can step on it) or not.

Example:

```
myneatstructure.set_overlappable(True)
```

set_pickable(*val*)
Make the structure pickable or not.

Parameters **val** (*bool*) – True or False depending on the pickability of the structure.

Example:

```
myneatstructure.set_pickable(True)
```

set_restorable(*val*)
Make the structure restorable or not.

Parameters **val** (*bool*) – True or False depending on the restorability of the structure.

Example:

```
myneatstructure.set_restorable(True)
```

size()
Return the size of the Immovable Item.

Returns The size of the item.

Return type int

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.Structures.**GenericStructure**(***kwargs*)
Bases: *gamelib.Immovable.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 *gamelib.Structures.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 GenericActionableStructure 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.

On top of these, this object takes all parameters of *BoardItem* and *Immovable*

Important: If you need a structure with a permission system please have a look at *GenericActionableStructure*. This class has a permission system for activation.

---

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

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

Return type str

display()
Print the model WITHOUT carriage return.

overlappable()
This represent the capacity for a *BoardItem* to be overlapped by player or NPC.

To set this value please use *set_overlappable()*

Returns False

Return type bool

See also:

set_overlappable()

pickable()
This represent the capacity for a BoardItem to be picked-up by player or NPC.

To set this value please use *set_pickable()*

Returns True or False

Return type bool

See also:

set_pickable()

restorable()
This represent the capacity for an *Immovable BoardItem* (in this case a GenericStructure item) to be restored by the board if the item is overlappable and has been overlapped by another *Movable* item.

The value of this property is set with *set_restorable()*

Returns False

Return type bool

See also:

set_restorable()

set_overlappable(*val*)
Make the structure overlappable or not.

Parameters **val** (*bool*) – True or False depending on the fact that the structure can be overlapped (i.e that a Player or NPC can step on it) or not.

Example:

```
myneatstructure.set_overlappable(True)
```

set_pickable(*val*)
Make the structure pickable or not.

Parameters **val** (*bool*) – True or False depending on the pickability of the structure.

Example:

```
myneatstructure.set_pickable(True)
```

set_restorable(*val*)
Make the structure restorable or not.

Parameters **val** (*bool*) – True or False depending on the restorability of the structure.

Example:

```
myneatstructure.set_restorable(True)
```

size()
Return the size of the Immovable Item.

Returns The size of the item.

Return type int

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.Structures.**Treasure**(***kwargs*)

Bases: *gamelib.Immovable.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.
- **size** (*str*) – The size of 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=Sprites.MONEY_BAG,value=100,size=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}")
```

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

debug_info()

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

Return type str

display()

Print the model WITHOUT carriage return.

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

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

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

size()
Return the size of the Immovable Item.

Returns The size of the item.

Return type int

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

class gamelib.Structures.**Wall**(***kwargs*)
Bases: *gamelib.Immovable.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.

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

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

Return type str

display()
Print the model WITHOUT carriage return.

overlappable()
This represent the capacity for a *BoardItem* to be overlapped by player or NPC.

Returns False

Return type bool

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?')
```

restorable()
This represent the capacity for an *Immovable* Movable item. A wall is not overlappable.

Returns False

Return type bool

size()
Return the size of the Immovable Item.

Returns The size of the item.

Return type int

store_position(*row*, *column*)
Store the BoardItem position for self access.

The stored position is used for consistency and quick access to the self postion. 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*.

Example:

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

CHAPTER 13

Utils

This module regroup different utility functions and constants.

gamelib.Utils.**black**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**black_bright**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**black_dim**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**blue**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**blue_bright**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**blue_dim**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**clear_screen**()
This methods clear the screen

gamelib.Utils.**cyan**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**cyan_bright**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**cyan_dim**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**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:

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

gamelib.Utils.**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:

```
Utils.fatal("|x_x|")
```

gamelib.Utils.**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.

gamelib.Utils.**green**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**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( Utils.green_bright("This is a formatted message") )
```

gamelib.Utils.**green_dim**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**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:

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

gamelib.Utils.**init_term_colors**()
This function is a forward to colorama.init()

gamelib.Utils.**magenta**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**magenta_bright**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**magenta_dim**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**print_white_on_red**(*message*)
Print a white message over a red background.

Parameters **message** (*str*) – The message to print.

Example:

```
Utils.print_white_on_red("This is bright!")
```

gamelib.Utils.**red**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**red_bright**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**red_dim**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**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:

```
Utils.warn("This is a warning.")
```

gamelib.Utils.**white**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**white_bright**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**white_dim**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**yellow**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**yellow_bright**(*message*)
This method works exactly the way green_bright() work with different color.

gamelib.Utils.**yellow_dim**(*message*)
This method works exactly the way green_bright() work with different color.

CHAPTER 14

Actuators

14.1 SimpleActuators

This module contains the simple actuators classes. Simple actuators are movement related one. They allow for predetermined movements patterns.

class gamelib.Actuators.SimpleActuators.**PathActuator**(*path=None*, *parent=None*)

Bases: *gamelib.Actuators.Actuator.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** (gamelib.BoardItem.BoardItem) – The parent object to actuate.

next_move()

Return the movement based on current index

The movement is selected from path if state is RUNNING, otherwise it should return None. 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 | None

Example:

```
pathactuator.next_move()
```

pause()

Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

set_path(*path*)
Defines a new path

This will also reset the index back to 0.

Parameters **path** (*list*) – A list of movements.

Example:

```
pathactuator.set_path([Constants.UP,Constants.DOWN,Constants.LEFT,Constants.
↪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()
```

class gamelib.Actuators.SimpleActuators.**PatrolActuator**(*path=None*, *parent=None*)
Bases: *gamelib.Actuators.SimpleActuators.PathActuator*

The patrol actuator is a subclass of PathActuator. 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 beggining of the list. Once it reaches the beggining 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.

next_move()
Return the movement based on current index

The movement is selected from path if state is RUNNING, otherwise it should return None. 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 | None

Example:

```
patrolactuator.next_move()
```

pause()
Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

set_path(*path*)
Defines a new path

This will also reset the index back to 0.

Parameters **path** (*list*) – A list of movements.

Example:

```
pathactuator.set_path([Constants.UP,Constants.DOWN,Constants.LEFT,Constants.
↪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()
```

class gamelib.Actuators.SimpleActuators.**RandomActuator**(*moveset=None*, *parent=None*)

Bases: *gamelib.Actuators.Actuator.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** (gamelib.BoardItem.BoardItem) – The parent object to actuate.

next_move()
Return a randomly selected movement

The movement is randomly selected from moveset if state is RUNNING, otherwise it should return None.

Returns The next movement

Return type int | None

Example:

```
randomactuator.next_move()
```

pause()
Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

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()
```

class gamelib.Actuators.SimpleActuators.**UnidirectionalActuator**(*direction=10000100*, *parent=None*)

Bases: *gamelib.Actuators.Actuator.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** (gamelib.BoardItem.BoardItem) – The parent object to actuate.

next_move()
Return the direction.

The movement is always direction if state is RUNNING, otherwise it returns None.

Returns The next movement

Return type int | None

Example:

```
unidirectional_actuator.next_move()
```

pause()
Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

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()
```

14.2 AdvancedActuators

This module contains the more advanced actuators. AdvancedActuators allow for more actions and not only movement. It can also be more advanced movement classes.

class gamelib.Actuators.AdvancedActuators.**PathFinder**(*game=None*, *actuated_object=None*, *circle_waypoints=True*, *parent=None*)
Bases: *gamelib.Actuators.Actuator.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** (gamelib.Game.Game) – A reference to the instanciated game engine.
- **actuated_object** (gamelib.BoardItem.BoardItem) – The object to actuate. Deprecated in favor of parent. Only kept for backward compatibility.
- **parent** (gamelib.BoardItem.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)

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 *HacInvalidTypeException* – 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)
```

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:

```
mykillernpc.actuator = PathFinder(
                        game=mygame,
                        actuated_object=mykillernpc
                    )
mykillernpc.actuator.set_destination(
                        mygame.player.pos[0],
                        mygame.player.pos[1]
                    )
mykillernpc.actuator.find_path()
for i in mykillernpc.actuator.current_path():
    print(i)
```

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)
```

find_path()
Find a path to the destination.

Destination (PathFinder.destination) has to be set beforehand. This method implements a Breadth First Search algorithm (Wikipedia) to find the shortest path to destination.

Example:

```
mykillernpc.actuator = PathFinder(
        game=mygame, actuated_object=mykillernpc
```

(continues on next page)

(continued from previous page)

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

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

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 betwen 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:

```
seeker = NPC(model=Sprites.SKULL)
seeker.actuator = PathFinder(game=mygame,actuated_object=seeker)
while True:
    seeker.actuator.set_destination(mygame.player.pos[0],mygame.player.pos[1])
    # next_move() will call find_path() for us.
    next_move = seeker.actuator.next_move()
    if next_move == Constants.NO_DIR:
        seeker.actuator.set_destination(mygame.player.pos[0],mygame.player.
↪pos[1])
    else:
        mygame.current_board().move(seeker,next_move,1)
```

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)
```

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 HacInvalidTypeException is raised.

Parameters

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

Raises

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

Example:

```
method()
```

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 ***HacInvalidTypeException*** – 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()
```

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 inheritate from one of these base class.

class gamelib.Actuators.Actuator.**Actuator**(*parent*)
Bases: object

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.

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

Raises NotImplementedError

pause()
Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

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()
```

class gamelib.Actuators.Actuator.**Behavioral**(*parent*)
Bases: *gamelib.Actuators.Actuator.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.

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

pause()
Set the actuator state to PAUSED.

Example:

```
mygame.pause()
```

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()
```

CHAPTER 15

Animation

This module contains the animation relation classes (so far only Animation).

class gamelib.Animation.**Animation**(*display_time=0.05*, *auto_replay=True*, *frames=None*, *animated_object=None*, *refresh_screen=None*, *initial_index=None*, *parent=None*)

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 explicitely. 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[str]*) – an array of "frames" (string)
- **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.

Example

```
def redraw_screen(game_object):
    game_object.clear_screen()
```

(continues on next page)

(continued from previous page)

```
    game_object.display_board()

item = BoardItem(model=Sprite.ALIEN, name='Friendly Alien')
# By default BoardItem does not have any animation, we have to
# explicitely create one
item.animation = Animation(display_time=0.1, parent=item,
                           refresh_screen=redraw_screen)
```

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** (*str*) – The frame to add to the animation.

Raise *gamelib.HacExceptions.HacInvalidTypeException*

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()
```

next_frame()

Update the parent.model with the next frame of the animation.

That method takes care of automatically replaying the animation if the last frame is reached if the state is RUNNING.

If the 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 *HacInvalidTypeException*

Example:

```
item.animation.next_frame()
```

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.

If screen_refresh is not defined or is not a function an exception is raised.

Raise *HacInvalidTypeException*

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, HacInvalidTypeException

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 *gamelib.HacExceptions.HacInvalidTypeException*

Example:

```
item.animation.remove_frame(
    item.animation.search_frame(Sprite.ALIEN_MONSTER)
)
```

start()

Set the animation state to RUNNING.

If the animation state is not 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()
```

CHAPTER 16

Credits

16.1 Development Leads

- Arnaud Dupuis (@arnauddupuis)

16.2 Top Contributors

- Kalil de Lima (@kaozdl)

16.3 Contributors

- Muhammad Syuqri (@Dansyuqri)
- Ryan Brown (@grimmjow8)
- 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)

CHAPTER 17

History

17.1 1.1.1 (2020-07-15)

- 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.

17.2 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.

17.3 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).

17.4 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
- 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

17.5 2019.5

- Please see the official website.

17.6 pre-2019.5

- Please see the Github for history.

CHAPTER 18

Forewords

This python3 module is a base for the programming lessons of the Hyrule Astronomy Club. It is not meant to be a comprehensive game building library.

It is however meant (and used) to teach core programming concept to kids from age 6 to 13.

CHAPTER 19

Introduction

First of all, his module is exclusively compatible with python 3.

The core concept is that it revolve around the *Game* object, the *Board* object and the derivatives of *BoardItem*.

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

The base game makes use of:

- The main "game engine" (gamelib.Game.Game)
- **Many different types of structures (from gamelib.Structures), like:**
  - Wall (well the walls. . .),
  - Treasure (gems and money bag),
  - GenericStructure (trees),
  - GenericActionnableStructure (hearts and portals).
- Game()'s menu capabilities.
- Player and NPC (from gamelib.Characters)
- Inventory (from gamelib.Inventory)
- Player and Inventory stats
- **Simple actuators (gamelib.SimpleActuators) like:**
  - RandomActuator (NPCs in level 2),
  - PathActuator (NPCs in level 1).

CHAPTER 20

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Y