
PyPi Package Example Documentation

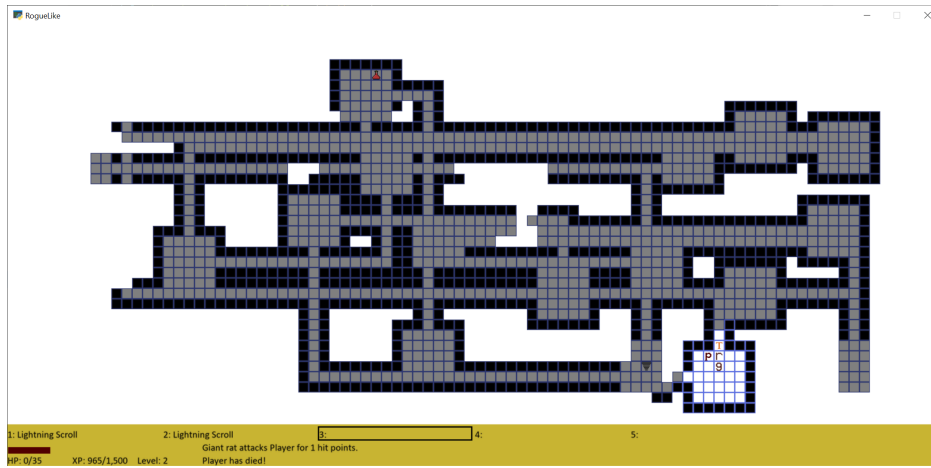
Release 1.0.0

Paul Vincent Craven

Dec 05, 2020

CONTENTS

1	How To Install	3
2	How To Play	5
2.1	Key Bindings	5
2.2	Combat	5
3	Features	7
4	Code Documentation	9
4.1	Game Window	9
4.2	Game Engine	10
	Python Module Index	13
	Index	15



This is a [rogue-like](#) adventure written in the [Python](#) computer language using the [Arcade Library](#).

Use this code to hack away at creating your own adventure game!

HOW TO INSTALL

- Download or clone the code from GitHub: <https://github.com/pythonarcade/roguelike>
- Switch to the directory, or open in an IDE like PyCharm
- Create a new virtual environment
- Install the depended packages through your IDE or via `pip install -r requirements.txt`
- Run with `python source`

HOW TO PLAY

2.1 Key Bindings

- Move with the number pad in 8 directions (num lock off)
- Pick up an item with G or Num 5
- Select an item with the numbers 1 - 9
- Use the selected item with U
- Drop the selected item with D
- Save game with S
- Load game with L
- Bring up the character screen with C
 - If you have ability points, click on the + to increase that stat
- Cancel the grid selection, character screen, etc. with ESC

2.2 Combat

- Move 'into' a monster to attack it
- Fireball is an area of effect weapon, and can damage the player.
- Lightning attacks the closest monster

FEATURES

- Procedural dungeons
- Character leveling system
- Ranged lighting spell
- Area of effect spell
- Field of vision
- Monster table
- Inventory management system
- A-star path-finding for monsters
- Can save/restore dungeon via JSON formatted data
- Message/event system

CODE DOCUMENTATION

At the top level, the `__main__.py` file will create an instance of the `GameWindow` class. This class manages the GUI and responds to events.

The next layer down is the `GameEngine`. This is more of a logic layer, although it isn't completely divorced from the display.

4.1 Game Window

Main Window Manager.

```
class source.game_window.MyGame(width, height, title)
```

Main application class. Manage the GUI

```
    __init__(width, height, title)
```

Parameters

- **width** (`int`) –
- **height** (`int`) –
- **title** (`str`) –

```
    check_for_player_movement()
```

Figure out if we should move the player or not based on keys currently held down.

```
    draw_character_screen()
```

```
    draw_hp_and_status_bar()
```

```
    draw_in_normal_state()
```

```
    draw_in_select_location_state()
```

```
    draw_inventory()
```

```
    draw_messages()
```

```
    draw_mouse_over_text()
```

```
    draw_sprites_and_status_panel()
```

```
    handle_character_screen_click(x, y)
```

```
    handle_messages()
```

```
    load()
```

Load the game from disk.

on_draw()

Render the screen.

on_key_press(*key*, *modifiers*)

Manage key-down events

Parameters

- **key**(*int*) –
- **modifiers**(*int*) –

on_key_release(*key*, *modifiers*)

Called when the user releases a key.

Parameters

- **key**(*int*) –
- **modifiers**(*int*) –

on_mouse_motion(*x*, *y*, *dx*, *dy*)

Handle mouse motion, mostly just used for mouse-over text.

on_mouse_press(*x*, *y*, *button*, *modifiers*)

Handle mouse-down events

Parameters

- **x**(*float*) –
- **y**(*float*) –
- **button**(*int*) –
- **modifiers**(*int*) –

on_update(*delta_time*)

Manage regular updates for the game

Parameters **delta_time**(*float*) –

save()

Save the current game to disk.

setup()

Set up the game here. Call this function to restart the game.

`source.game_window.main()`

Main method for starting the rogue-like game

4.2 Game Engine

Define the game engine

class `source.game_engine.GameEngine`

This is the main game engine class, that manages the game and its actions.

__init__()

Set the game engine's attributes

check_experience_level()

See if the player should level up

dying (*target*)

Handle event of an entity dying

Parameters **target** (*Entity*) –

Return type *list*

get_dict ()

Get a dictionary object for the entire game. Used in serializing the game state for saving to disk or sending over the network.

grid_click (*grid_x*, *grid_y*)

Handle a click on the grid

move_enemies ()

Process enemy movement.

move_player (*cx*, *cy*)

Process player movement

Parameters

- **cx** (*int*) –

- **cy** (*int*) –

pick_up ()

Handle a pick-up item entity request.

process_action_queue (*delta_time*)

Process the action queue, kind of a dispatch-center for the game.

Parameters **delta_time** (*float*) –

restore_from_dict (*data*)

Restore this object from a dictionary object. Used in recreating a game from a saved state, or from over the network.

Parameters **data** (*dict*) –

setup ()

Set up the game here. Call this function to restart the game.

setup_level (*level_number*)

Parameters **level_number** (*int*) –

Return type *GameLevel*

use_stairs ()

class `source.game_engine.GameLevel`

__init__ ()

Initialize level instance.

Other links:

- [Source on GitHub](#)
- [Arcade Library](#)
- [License \(MIT\)](#)

PYTHON MODULE INDEX

S

`source.game_engine`, [10](#)

`source.game_window`, [9](#)

Symbols

`__init__()` (source.game_engine.GameEngine method), 10
`__init__()` (source.game_engine.GameLevel method), 11
`__init__()` (source.game_window.MyGame method), 9

C

`check_experience_level()` (source.game_engine.GameEngine method), 10
`check_for_player_movement()` (source.game_window.MyGame method), 9

D

`draw_character_screen()` (source.game_window.MyGame method), 9
`draw_hp_and_status_bar()` (source.game_window.MyGame method), 9
`draw_in_normal_state()` (source.game_window.MyGame method), 9
`draw_in_select_location_state()` (source.game_window.MyGame method), 9
`draw_inventory()` (source.game_window.MyGame method), 9
`draw_messages()` (source.game_window.MyGame method), 9
`draw_mouse_over_text()` (source.game_window.MyGame method), 9
`draw_sprites_and_status_panel()` (source.game_window.MyGame method), 9
`dying()` (source.game_engine.GameEngine method), 10

G

`GameEngine` (class in source.game_engine), 10
`GameLevel` (class in source.game_engine), 11
`get_dict()` (source.game_engine.GameEngine method), 11
`grid_click()` (source.game_engine.GameEngine method), 11

H

`handle_character_screen_click()` (source.game_window.MyGame method), 9
`handle_messages()` (source.game_window.MyGame method), 9

L

`load()` (source.game_window.MyGame method), 9

M

`main()` (in module source.game_window), 10
module
 source.game_engine, 10
 source.game_window, 9
`move_enemies()` (source.game_engine.GameEngine method), 11
`move_player()` (source.game_engine.GameEngine method), 11
`MyGame` (class in source.game_window), 9

O

`on_draw()` (source.game_window.MyGame method), 9
`on_key_press()` (source.game_window.MyGame method), 10
`on_key_release()` (source.game_window.MyGame method), 10
`on_mouse_motion()` (source.game_window.MyGame method), 10
`on_mouse_press()` (source.game_window.MyGame method), 10

`on_update()` (*source.game_window.MyGame method*), [10](#)

P

`pick_up()` (*source.game_engine.GameEngine method*), [11](#)

`process_action_queue()` (*source.game_engine.GameEngine method*), [11](#)

R

`restore_from_dict()` (*source.game_engine.GameEngine method*), [11](#)

S

`save()` (*source.game_window.MyGame method*), [10](#)

`setup()` (*source.game_engine.GameEngine method*), [11](#)

`setup()` (*source.game_window.MyGame method*), [10](#)

`setup_level()` (*source.game_engine.GameEngine method*), [11](#)

`source.game_engine`
module, [10](#)

`source.game_window`
module, [9](#)

U

`use_stairs()` (*source.game_engine.GameEngine method*), [11](#)