
EzPyGame Documentation

Release 0.3.1

Markus Meskanen

Apr 14, 2017

Contents

Python Module Index

5

Easier and more pythonic usage of `pygame`.

class `ezpygame.Application` (*title=None, resolution=None, update_rate=None*)

A simple wrapper around `pygame` for running games easily.

Also makes scene management seamless together with the `Scene` class.

Parameters

- **title** (*str/None*) – title to display in the window’s title bar
- **resolution** (*tuple[int, int]/None*) – resolution of the game window
- **update_rate** (*int/None*) – how many times per second to update

If any parameters are left to `None`, these settings must be defined either manually through `application.<setting> = value` or via `Scene`’s class variable settings.

Example usage:

```
class Menu(ezpygame.Scene):
    ...

class Game(ezpygame.Scene):
    ...

app = ezpygame.Application(
    title='My First EzPyGame Application',
    resolution=(1280, 720),
    update_rate=60,
)
main_menu = Menu()
app.run(main_menu)
```

active_scene

The currently active scene. Can be `None`.

change_scene (*scene*)

Change the currently active scene.

This will invoke `Scene.on_exit()` and `Scene.on_enter()` methods on the switching scenes.

If `None` is provided, the application’s execution will end.

Parameters *scene* (`Scene/None`) – the scene to change into

run (*scene=None*)

Execute the application.

Parameters *scene* (`Scene/None`) – scene to start the execution from

class `ezpygame.Scene` (*title=None, resolution=None, update_rate=None*)

An isolated scene which can be ran by an application.

Create your own scene by subclassing and overriding any methods. The hosting `Application` instance is accessible through the `application` property.

Example usage with two scenes interacting:

```
class Menu(Scene):

    def __init__(self):
        self.font = pygame.font.Font(...)
```

```
def on_enter(self, previous_scene):
    self.application.title = 'Main Menu'
    self.application.resolution = (640, 480)
    self.application.update_rate = 30

def draw(self, screen):
    pygame.draw.rect(...)
    text = self.font.render(...)
    screen.blit(text, ...)

def handle_event(self, event):
    if event.type == pygame.MOUSEBUTTONDOWN:
        if event.button == 1:
            game_size = self._get_game_size(event.pos)
            self.change_scene(Game(game_size))

def _get_game_size(self, mouse_pos_upon_click):
    ...

class Game(ezpygame.Scene):
    title = 'The Game!'
    resolution = (1280, 720)
    update_rate = 60

    def __init__(self, size):
        super().__init__()
        self.size = size
        self.player = ...
        ...

    def on_enter(self, previous_scene):
        super().on_enter(previous_scene)
        self.previous_scene = previous_scene

    def draw(self, screen):
        self.player.draw(screen)
        for enemy in self.enemies:
            ...

    def update(self, dt):
        self.player.move(dt)
        ...
        if self.player.is_dead():
            self.application.change_scene(self.previous_scene)
        elif self.player_won():
            self.application.change_scene(...)

    def handle_event(self, event):
        ... # Player movement etc.
```

The above two classes use different approaches for changing the application's settings when the scene is entered:

1. Manually set them in `on_enter()`, as seen in Menu
2. Use class variables, as I did with Game

When using class variables (2), you can leave out any setting (defaults to None) to not override that particular setting. If you override `on_enter()` in the subclass, you must call `super()`.

`on_enter(previous_scene)` to use the class variables.

These settings can further be overridden in individual instances:

```
my_scene0 = MyScene()
my_scene0.resolution = (1280, 720)
my_scene1 = MyScene(title='My Second Awesome Scene')
```

application

The host application that's currently running the scene.

draw(screen)

Override this with the scene drawing.

Parameters `screen` (`pygame.Surface`) – screen to draw the scene on

handle_event(event)

Override this to handle an event in the scene.

All of `pygame`'s events are sent here, so filtering should be applied manually in the subclass.

Parameters `event` (`pygame.event.Event`) – event to handle

on_enter(previous_scene)

Override this to initialize upon scene entering.

The `application` property is initialized at this point, so you are free to access it through `self.application`. Stuff like changing resolution etc. should be done here.

If you override this method and want to use class variables to change the application's settings, you must call `super().on_enter(previous_scene)` in the subclass.

Parameters `previous_scene` (`Scene` / `None`) – previous scene to run

on_exit(next_scene)

Override this to deinitialize upon scene exiting.

The `application` property is still initialized at this point. Feel free to do saving, settings reset, etc. here.

Parameters `next_scene` (`Scene` / `None`) – next scene to run

update(dt)

Override this with the scene update tick.

Parameters `dt` (`int`) – time in milliseconds since the last update

e

ezpygame, [1](#)

A

active_scene (ezpygame.Application attribute), 1
Application (class in ezpygame), 1
application (ezpygame.Scene attribute), 3

C

change_scene() (ezpygame.Application method), 1

D

draw() (ezpygame.Scene method), 3

E

ezpygame (module), 1

H

handle_event() (ezpygame.Scene method), 3

O

on_enter() (ezpygame.Scene method), 3
on_exit() (ezpygame.Scene method), 3

R

run() (ezpygame.Application method), 1

S

Scene (class in ezpygame), 1

U

update() (ezpygame.Scene method), 3