pyrogi Documentation

Release 0.1.3

Ben Weedon

Nov 05, 2017

Contents:

CHAPTER 1

Pyrogi Core

The top level package for the pyrogi engine. This package defines the components of the engine which will be included in every game, such as events, and the classes *Backend* and *Screen*.

pyrogi.**get_window_dimensions**()

> **Returns** The dimensions in tiles of the game window.
>
> **Return type** Vec2

pyrogi.**get_tile_dimensions**()

> **Returns** The dimensions in pixels of each game tile.
>
> **Return type** Vec2

pyrogi.**get_caption**()

> **Returns** The title caption of the game window.
>
> **Return type** str

pyrogi.**get_mouse_position**()

> **Returns** The current position in tiles of the mouse.
>
> **Return type** Vec2

class pyrogi.**Backend**(*window_dimensions*, *tile_dimensions*, *caption*)

The *Backend* class is intended to be extended (for example, by PyGameBackend). *Backend* forwards events such as key events and ticks on to the current *Screen*. Its implementations are expected to implement the *run()* method, however, which should start up the main loop for the backend.

The __init__ method of a *Backend* is as follows:

Parameters

- **window_dimensions** (*Vec2*) – The dimensions, in terms of of tiles, for the game window to be.
- **tile_dimensions** (*Vec2*) – The dimensions, in terms of pixels, for each tile to be.

- **caption** (*str*) – The title text of the game window.

get_current_screen()

Returns The screen currently set in the engine.

Return type *Screen*

go_back_n_screens(*n*)

Return to the nth screen in the stack, popping off screens as you go. Because screens are removed from the stack to reach the nth most recent screen, they cannot be returned to without constructing new ones.

Parameters n (*int*) – The number of screens to go back.

handle_key_down(*event*)

Called by the *Backend* implementation when a key down event has occured. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every time the event is triggered. The event is passed on to the backend's current screen.

Parameters event (KeyDownEvent) – The event triggered.

handle_key_up(*event*)

Called by the *Backend* implementation when a key up event has occured. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every time the event is triggered. The event is passed on to the backend's current screen.

Parameters event (KeyUpEvent) – The event triggered.

handle_mouse_button_down(*event*)

Called by the *Backend* implementation when a mouse button down event has occured. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every time the event is triggered. The event is passed on to the backend's current screen.

Parameters event (MouseButtonDownEvent) – The event triggered.

handle_mouse_button_up(*event*)

Called by the *Backend* implementation when a mouse button up event has occured. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every time the event is triggered. The event is passed on to the backend's current screen.

Parameters event (MouseButtonUpEvent) – The event triggered.

handle_mouse_moved(*event*)

Called by the *Backend* implementation when a mouse moved event has occured. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every time the event is triggered. The event is passed on to the backend's current screen.

Parameters event (MouseMovedEvent) – The event triggered.

handle_mouse_wheel_scrolled(*event*)

Called by the *Backend* implementation when a mouse wheel scrolled has occured. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every time the event is triggered. The event is passed on to the backend's current screen.

Parameters event (MouseWheelScrolledEvent) – The event triggered.

on_draw(*g*)

Called by the *Backend* implementation when a draw should occur in the game loop. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every draw. The draw is passed on to the backend's current screen.

Parameters g (*Graphics*) – The Graphics object to be used to perform the drawing.

on_tick(*millis*)

Called by the *Backend* implementation when a tick should occur in the game loop. This should never be overriden, but can be extended if the *Backend* implementation wishes to do something every tick. The tick is passed on to the backend's current screen.

Parameters **millis** (*int*) – The number of milliseconds since the last tick.

run()

Run the main loop for pyrogi.

This must be implemented by subclasses of *Backend*.

set_screen(*screen*)

Sets the current screen in the engine. It is added to a stack of other screens, which maintains history and allows the engine to return to previous screens.

Parameters **screen** (Screen) – The screen to be added to the stack.

class pyrogi.**Event**

An event, such as a key event or mouse event, handled by the *Backend*.

class pyrogi.**KeyDownEvent**(*character*, *key*, *modifier*)

An event triggered when a key is pressed down.

The __init__ method is as follows:

Parameters

- **character** (*str*) – The character representing the key pressed.
- **key** (*int*) – The key pressed.
- **modifier** (*int*) – The modifier key pressed.

class pyrogi.**KeyUpEvent**(*key*, *modifier*)

An event triggered when a key is released.

The __init__ method is as follows:

Parameters

- **key** (*int*) – The key released.
- **modifier** (*int*) – The modifier key released.

class pyrogi.**MouseButtonDownEvent**(*position*, *button*)

An event triggered when a mouse button is pressed down.

The __init__ method is as follows:

Parameters

- **position** (*tuple*) – The current position of the mouse.
- **button** (*button*) – The button pressed.

class pyrogi.**MouseButtonUpEvent**(*position*, *button*)

An event triggered when a mouse button is released.

The __init__ method is as follows:

Parameters

- **position** (*tuple*) – The current position of the mouse.
- **button** (*button*) – The button released.

class `pyrogi.`**`MouseMovedEvent`**(*position*, *relative_position*, *buttons*)

An event triggered when the mouse is moved.

The `__init__` method is as follows:

Parameters

- **`position`** (*`tuple`*) – The current position of the mouse.
- **`relative_position`** (*`tuple`*) – The vector representing the motion of the mouse.
- **`buttons`** (*`list`*) – The buttons pressed on the mouse.

class `pyrogi.`**`MouseWheelScrolledEvent`**(*position*, *scroll_amount*)

An event triggered when the mouse wheel is scrolled.

The `__init__` method is as follows:

Parameters

- **`position`** (*`tuple`*) – The current position of the mouse.
- **`scroll_amount`** (*`int`*) – The amount scrolled.

class `pyrogi.`**`Screen`**

The *`Screen`* class is the top level `UIElementContainer`. It represents a single screen in a game. An example may be the menu screen, the main game screen, or the shop screen if you are placed in a view separate from the game world to shop. Because they are the top level container, a screen's position is always (0, 0), and its dimensions is always equal to that of the game window. A stack of screens is maintained in the *`Backend`* object.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index