

## Abstract Art Generator

Generated by Doxygen 1.8.17



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 canvas.canvas Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 <code>__init__()</code>	8
4.1.3 Member Function Documentation	8
4.1.3.1 <code>draw_layers()</code>	8
4.1.3.2 <code>draw_to_canvas()</code>	9
4.1.3.3 <code>generate_bg()</code>	9
4.1.3.4 <code>get_canvas()</code>	9
4.2 color_palette.color_palette Class Reference	9
4.2.1 Detailed Description	11
4.2.2 Constructor & Destructor Documentation	11
4.2.2.1 <code>__init__()</code>	11
4.2.3 Member Function Documentation	11
4.2.3.1 <code>draw_ui_dynamic()</code>	11
4.2.3.2 <code>draw_ui_static()</code>	12
4.2.3.3 <code>events()</code>	12
4.2.3.4 <code>get_background_color()</code>	12
4.2.3.5 <code>get_colors_from_palette()</code>	12
4.2.3.6 <code>get_foreground_colors()</code>	13
4.2.3.7 <code>get_name_of_palette()</code>	13
4.2.3.8 <code>refresh_ui_static()</code>	13
4.3 generators.generators Class Reference	13
4.3.1 Detailed Description	14
4.3.2 Constructor & Destructor Documentation	14
4.3.2.1 <code>__init__()</code>	14
4.3.3 Member Function Documentation	15
4.3.3.1 <code>draw_circles()</code>	15
4.3.3.2 <code>draw_curves()</code>	15
4.3.3.3 <code>draw_dots()</code>	16
4.3.3.4 <code>draw_fpolygons()</code>	16
4.3.3.5 <code>draw_hpolygons()</code>	16
4.3.3.6 <code>draw_lines()</code>	17

4.3.3.7 draw_rings()	17
4.3.3.8 draw_squares()	18
4.4 help.help Class Reference	18
4.4.1 Detailed Description	19
4.4.2 Constructor & Destructor Documentation	19
4.4.2.1 __init__()	20
4.4.3 Member Function Documentation	20
4.4.3.1 draw_ui_dynamic()	20
4.4.3.2 draw_ui_static()	20
4.4.3.3 events()	20
4.5 layer.layer Class Reference	21
4.5.1 Detailed Description	22
4.5.2 Constructor & Destructor Documentation	22
4.5.2.1 __init__()	22
4.5.3 Member Function Documentation	23
4.5.3.1 draw_ui_dynamic()	23
4.5.3.2 draw_ui_static()	23
4.5.3.3 events()	23
4.5.3.4 get_layer_complexity()	24
4.5.3.5 get_layer_shape()	24
4.5.3.6 get_layer_size()	24
4.5.3.7 get_layer_style()	24
4.5.3.8 get_layer_transparency()	25
4.6 overlay.overlay Class Reference	25
4.6.1 Detailed Description	26
4.6.2 Constructor & Destructor Documentation	26
4.6.2.1 __init__()	26
4.6.3 Member Function Documentation	26
4.6.3.1 draw_ui_dynamic()	26
4.6.3.2 draw_ui_static()	27
4.6.3.3 events()	27
4.6.3.4 get_active_overlay()	27
4.7 switch_theme.switch_theme Class Reference	28
4.7.1 Detailed Description	28
4.7.2 Constructor & Destructor Documentation	28
4.7.2.1 __init__()	28
4.7.3 Member Function Documentation	29
4.7.3.1 draw_ui_static()	29
4.7.3.2 events()	29
4.7.3.3 getDarkMode()	29
4.8 text_overlay.text_overlay Class Reference	30
4.8.1 Detailed Description	31

4.8.2 Constructor & Destructor Documentation	31
4.8.2.1 <code>__init__()</code>	31
4.8.3 Member Function Documentation	31
4.8.3.1 <code>draw_ui_dynamic()</code>	31
4.8.3.2 <code>draw_ui_static()</code>	32
4.8.3.3 <code>events()</code>	32
4.9 <code>ui_controller.ui_controller</code> Class Reference	32
4.9.1 Detailed Description	33
4.9.2 Constructor & Destructor Documentation	34
4.9.2.1 <code>__init__()</code>	34
4.9.3 Member Function Documentation	34
4.9.3.1 <code>draw_ui_dynamic()</code>	34
4.9.3.2 <code>draw_ui_static()</code>	34
4.9.3.3 <code>process_events()</code>	34
4.9.3.4 <code>run()</code>	35
4.9.4 Member Data Documentation	35
4.9.4.1 <code>resolutions_list</code>	35
4.10 <code>widget.widget</code> Class Reference	35
4.10.1 Detailed Description	36
4.10.2 Constructor & Destructor Documentation	36
4.10.2.1 <code>__init__()</code>	36
4.10.3 Member Function Documentation	37
4.10.3.1 <code>draw_ui_dynamic()</code>	37
4.10.3.2 <code>draw_ui_static()</code>	37
4.10.3.3 <code>events()</code>	37
4.11 <code>widget_storage.widget_storage</code> Class Reference	37
4.11.1 Detailed Description	38
<b>5 File Documentation</b>	<b>39</b>
5.1 <code>assets.py</code> File Reference	39
5.1.1 Detailed Description	40
5.1.2 Author(s)	40
5.1.3 Function Documentation	40
5.1.3.1 <code>text_to_screen()</code>	40
5.2 <code>canvas.py</code> File Reference	40
5.2.1 Detailed Description	41
5.2.2 Author(s)	41
5.3 <code>color_palette.py</code> File Reference	41
5.3.1 Detailed Description	41
5.3.2 Author(s)	41
5.4 <code>generators.py</code> File Reference	41
5.4.1 Detailed Description	42

---

5.4.2 Author(s) . . . . .	42
5.4.3 Variable Documentation . . . . .	42
5.4.3.1 art_styles_list . . . . .	42
5.5 help.py File Reference . . . . .	42
5.5.1 Detailed Description . . . . .	42
5.5.2 Author(s) . . . . .	42
5.6 layer.py File Reference . . . . .	43
5.6.1 Detailed Description . . . . .	43
5.6.2 Author(s) . . . . .	43
5.7 overlay.py File Reference . . . . .	43
5.7.1 Detailed Description . . . . .	43
5.7.2 Author(s) . . . . .	43
5.8 switch_theme.py File Reference . . . . .	43
5.8.1 Detailed Description . . . . .	44
5.8.2 Author(s) . . . . .	44
5.9 text_overlay.py File Reference . . . . .	44
5.9.1 Detailed Description . . . . .	44
5.9.2 Author(s) . . . . .	44
5.10 ui_controller.py File Reference . . . . .	44
5.10.1 Detailed Description . . . . .	45
5.10.2 Author(s) . . . . .	45
5.11 widget.py File Reference . . . . .	45
5.11.1 Detailed Description . . . . .	45
5.11.2 Author(s) . . . . .	45
5.12 widget_storage.py File Reference . . . . .	45
5.12.1 Detailed Description . . . . .	46
5.12.2 Author(s) . . . . .	46
5.12.3 Variable Documentation . . . . .	46
5.12.3.1 widgets . . . . .	46
<b>Index</b>	<b>47</b>

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

canvas.canvas . . . . .	7
generators.generators . . . . .	13
ui_controller.ui_controller . . . . .	32
widget_storage.widget_storage . . . . .	37
ABC	
widget.widget . . . . .	35
widget	
color_palette.color_palette . . . . .	9
help.help . . . . .	18
layer.layer . . . . .	21
overlay.overlay . . . . .	25
switch_theme.switch_theme . . . . .	28
text_overlay.text_overlay . . . . .	30





## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">canvas.canvas</a>	
The canvas class . . . . .	7
<a href="#">color_palette.color_palette</a>	
The color palette widget class . . . . .	9
<a href="#">generators.generators</a>	
Generators class that provides layer generation functionality to the layer module . . . . .	13
<a href="#">help.help</a>	
The help widget class . . . . .	18
<a href="#">layer.layer</a>	
The layer widget class . . . . .	21
<a href="#">overlay.overlay</a>	
The overlay widget class . . . . .	25
<a href="#">switch_theme.switch_theme</a>	
The theme switch widget class . . . . .	28
<a href="#">text_overlay.text_overlay</a>	
The text overlay widget class . . . . .	30
<a href="#">ui_controller.ui_controller</a>	
The <a href="#">ui_controller</a> class . . . . .	32
<a href="#">widget.widget</a>	
An abstract class for widgets to extend . . . . .	35
<a href="#">widget_storage.widget_storage</a>	
Storage for program widgets . . . . .	37



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">assets.py</a>	Stores various assets useful to other modules . . . . .	39
<a href="#">canvas.py</a>	Defines the canvas class . . . . .	40
<a href="#">color_palette.py</a>	Defines the color_palette class . . . . .	41
<a href="#">generators.py</a>	Defines the generator class which houses the various layer generation algorithms . . . . .	41
<a href="#">help.py</a>	Defines the help class . . . . .	42
<a href="#">layer.py</a>	Defines the layer class . . . . .	43
<a href="#">overlay.py</a>	Defines the overlay class . . . . .	43
<a href="#">switch_theme.py</a>	Defines the switch theme class . . . . .	43
<a href="#">text_overlay.py</a>	Defines the text_overlay class . . . . .	44
<a href="#">ui_controller.py</a>	Defines and initializes the ui_controller class . . . . .	44
<a href="#">widget.py</a>	Defines the widget abstract class . . . . .	45
<a href="#">widget_storage.py</a>	Defines and initializes the widget_storage class . . . . .	45



## Chapter 4

# Class Documentation

### 4.1 canvas.canvas Class Reference

The canvas class.

#### Public Member Functions

- def `__init__` (self, x, y, width, height, display\_width, display\_height, window)  
*Initializes the canvas.*
- def `draw` (self)  
*Draws the canvas to the ui.*
- def `draw_layers` (self)  
*Calls various widgets to draw to their layers.*
- def `draw_to_canvas` (self)  
*Blits layers to the canvas.*
- def `generate_bg` (self, color)  
*Fill the canvas background with a color.*
- def `get_canvas` (self)  
*Gets the pygame surface the canvas draws on.*
- def `get_height` (self)  
*Gets the height of the canvas.*
- def `get_width` (self)  
*Gets the width of the canvas.*

#### Public Attributes

- `bg_layer`  
*The single color, background layer of the canvas.*
- `canvas`  
*The surface the art is drawn to.*
- `display_canvas`  
*The surface that is drawn to the ui as a display port.*

### 4.1.1 Detailed Description

The canvas class.

Provides the canvas the program draws on along with functions for drawing layers to the canvas and drawing the canvas to the ui.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 `__init__()`

```
def canvas.canvas.__init__ (
    self,
    x,
    y,
    width,
    height,
    display_width,
    display_height,
    window )
```

Initializes the canvas.

#### Parameters

<i>x</i>	Horizontal position to draw the canvas at on the ui.
<i>y</i>	Vertical position to draw the canvas at on the ui.
<i>width</i>	Width of the canvas surface.
<i>height</i>	Height of the canvas surface.
<i>display_width</i>	Width of the ui's canvas display port.
<i>display_height</i>	Height of the ui's canvas display port.
<i>window</i>	Ui window to draw the canvas to.

### 4.1.3 Member Function Documentation

#### 4.1.3.1 `draw_layers()`

```
def canvas.canvas.draw_layers (
    self )
```

Calls various widgets to draw to their layers.

Calls `draw_canvas` in all the drawing widgets.

#### 4.1.3.2 draw\_to\_canvas()

```
def canvas.canvas.draw_to_canvas (
    self )
```

Blits layers to the canvas.

Combines the currently drawn layers into the canvas.

#### 4.1.3.3 generate\_bg()

```
def canvas.canvas.generate_bg (
    self,
    color )
```

Fill the canvas background with a color.

##### Parameters

<i>color</i>	Color to fill the background with.
--------------	------------------------------------

#### 4.1.3.4 get\_canvas()

```
def canvas.canvas.get_canvas (
    self )
```

Gets the pygame surface the canvas draws on.

##### Returns

The pygame surface the canvas draws on.

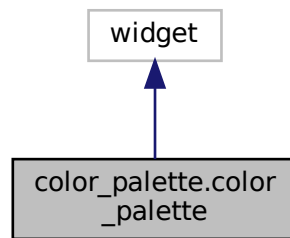
The documentation for this class was generated from the following file:

- [canvas.py](#)

## 4.2 color\_palette.color\_palette Class Reference

The color palette widget class.

Inheritance diagram for `color_palette.color_palette`:



## Public Member Functions

- `def __init__ (self, x, y, window, ui_manager)`  
*Initializes the color palette widget.*
- `def change_colors (self)`  
*Change the theme colors.*
- `def draw_ui_dynamic (self)`  
*Draws the dynamic ui elements for the color palette widget.*
- `def draw_ui_static (self)`  
*Draws the static ui elements for the color palette widget.*
- `def events (self, event)`  
*Processes pygame events for the color palette widget.*
- `def get_background_color (self)`  
*Get the background color in hex form.*
- `def get_colors_from_palette (self)`  
*Get the list of colors for the current palette in hex form.*
- `def get_foreground_colors (self)`  
*Get the list of foreground colors in hex form.*
- `def get_name_of_palette (self)`  
*Get the name of the current palette.*
- `def randomize (self)`  
*Randomize the current color palette and background color.*
- `def refresh_ui_static (self)`  
*Refreshes the static ui elements for the color palette widget.*

## Public Attributes

- `background_index`  
*The palette color selected as the background.*
- `background_index_buttons`
- `background_lock`  
*1 if randomization of the background color is locked, 0 otherwise*
- `color`



- `palette_colors`  
*Theme color for background.*
- `palette_lock`  
*Colors of currently selected palette.*
- `palette_name`  
*1 if randomization of the palette is locked, 0 otherwise*
- `ui_h1_color`  
*Name of currently selected palette.*
- `ui_h1_color`  
*Theme color for text.*

### 4.2.1 Detailed Description

The color palette widget class.

Provides a ui and functionality to specify the current color palette and the background color.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 `__init__()`

```
def color_palette.color_palette.__init__ (
    self,
    x,
    y,
    window,
    ui_manager )
```

Initializes the color palette widget.

#### Parameters

<code>x</code>	Horizontal position to draw the widget at on the ui.
<code>y</code>	Vertical position to draw the widget at on the ui.
<code>window</code>	Ui window to draw the widget to.
<code>ui_manager</code>	Pygame_gui element manager to tie pygame_gui elements to.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 `draw_ui_dynamic()`

```
def color_palette.color_palette.draw_ui_dynamic (
    self )
```

Draws the dynamic ui elements for the color palette widget.

Draws the text, lock icons, and color swatches.

#### 4.2.3.2 draw\_ui\_static()

```
def color_palette.color_palette.draw_ui_static (
    self )
```

Draws the static ui elements for the color palette widget.

Draws the palette dropdown, lock buttons, and color swatch buttons.

#### 4.2.3.3 events()

```
def color_palette.color_palette.events (
    self,
    event )
```

Processes pygame events for the color palette widget.

Handles the palette dropdown, lock buttons, and background color buttons.

##### Parameters

<i>event</i>	The pygame event being processed.
--------------	-----------------------------------

#### 4.2.3.4 get\_background\_color()

```
def color_palette.color_palette.get_background_color (
    self )
```

Get the background color in hex form.

##### Returns

The background color.

#### 4.2.3.5 get\_colors\_from\_palette()

```
def color_palette.color_palette.get_colors_from_palette (
    self )
```

Get the list of colors for the current palette in hex form.

##### Returns

A list of the palette colors.

#### 4.2.3.6 get\_foreground\_colors()

```
def color_palette.color_palette.get_foreground_colors (
    self )
```

Get the list of foreground colors in hex form.

##### Returns

A list of the palette colors excluding the background color.

#### 4.2.3.7 get\_name\_of\_palette()

```
def color_palette.color_palette.get_name_of_palette (
    self )
```

Get the name of the current palette.

##### Returns

The palette name.

#### 4.2.3.8 refresh\_ui\_static()

```
def color_palette.color_palette.refresh_ui_static (
    self )
```

Refreshes the static ui elements for the color palette widget.

Changes how many color swatch buttons display based on the length of the color palette.

The documentation for this class was generated from the following file:

- [color\\_palette.py](#)

## 4.3 generators.generators Class Reference

Generators class that provides layer generation functionality to the layer module.

## Public Member Functions

- def `__init__` (self, width, height)  
*Initializes the generators utility.*
- def `draw_circles` (self, layer, complexity, cp, style, magnitude)  
*Draws circles to a layer.*
- def `draw_curves` (self, layer, complexity, cp, style, magnitude)  
*Draws curves to a layer.*
- def `draw_dots` (self, layer, complexity, cp, style, magnitude)  
*Draws dots to a layer.*
- def `draw_fpolygons` (self, layer, complexity, cp, style, magnitude)  
*Draws filled polygons to a layer.*
- def `draw_hpolygons` (self, layer, complexity, cp, style, magnitude)  
*Draws hollow polygons to a layer.*
- def `draw_lines` (self, layer, complexity, cp, style, magnitude)  
*Draws lines to a layer.*
- def `draw_rings` (self, layer, complexity, cp, style, magnitude)  
*Draws rings to a layer.*
- def `draw_squares` (self, layer, complexity, cp, style, magnitude)  
*Draws squares to a layer.*

## Public Attributes

- `height`
- `width`

### 4.3.1 Detailed Description

Generators class that provides layer generation functionality to the layer module.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 `__init__()`

```
def generators.generators.__init__ (
    self,
    width,
    height )
```

Initializes the generators utility.

#### Parameters

<i>width</i>	Width of the canvas.
<i>height</i>	Height of the canvas.

### 4.3.3 Member Function Documentation

#### 4.3.3.1 draw\_circles()

```
def generators.generators.draw_circles (
    self,
    layer,
    complexity,
    cp,
    style,
    magnitude )
```

Draws circlces to a layer.

##### Parameters

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

#### 4.3.3.2 draw\_curves()

```
def generators.generators.draw_curves (
    self,
    layer,
    complexity,
    cp,
    style,
    magnitude )
```

Draws curves to a layer.

##### Parameters

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

#### 4.3.3.3 draw\_dots()

```
def generators.generators.draw_dots (
    self,
    layer,
    complexity,
    cp,
    style,
    magnitude )
```

Draws dots to a layer.

##### Parameters

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

#### 4.3.3.4 draw\_fpolygons()

```
def generators.generators.draw_fpolygons (
    self,
    layer,
    complexity,
    cp,
    style,
    magnitude )
```

Draws filled polygons to a layer.

##### Parameters

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

#### 4.3.3.5 draw\_hpolygons()

```
def generators.generators.draw_hpolygons (
    self,
    layer,
```

```
complexity,  
cp,  
style,  
magnitude )
```

Draws hollow polygons to a layer.

#### Parameters

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

#### 4.3.3.6 draw\_lines()

```
def generators.generators.draw_lines (  
    self,  
    layer,  
    complexity,  
    cp,  
    style,  
    magnitude )
```

Draws lines to a layer.

#### Parameters

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

#### 4.3.3.7 draw\_rings()

```
def generators.generators.draw_rings (  
    self,  
    layer,  
    complexity,  
    cp,  
    style,  
    magnitude )
```

Draws rings to a layer.

**Parameters**

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

**4.3.3.8 draw\_squares()**

```
def generators.generators.draw_squares (
    self,
    layer,
    complexity,
    cp,
    style,
    magnitude )
```

Draws squares to a layer.

**Parameters**

<i>layer</i>	The layer to draw to.
<i>complexity</i>	The complexity of the layer.
<i>cp</i>	The color palette to draw with.
<i>style</i>	The style of the layer.
<i>magnitude</i>	The magnitude of the layer.

The documentation for this class was generated from the following file:

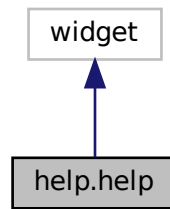
- [generators.py](#)

**4.4 help.help Class Reference**

The help widget class.



Inheritance diagram for help.help:



## Public Member Functions

- `def __init__ (self, x, y, window, ui_manager)`  
*Initializes the help widget.*
- `def change_colors (self)`  
*Change the theme colors.*
- `def draw_ui_dynamic (self)`  
*Draws the dynamic ui elements for the help widget.*
- `def draw_ui_static (self)`  
*Draws the static ui elements for the help widget.*
- `def events (self, event)`  
*Processes pygame events for the help widget.*

## Public Attributes

- `bg_color`  
*Theme color for background.*
- `font_color`  
*Theme color for normal font.*
- `font_color_emph`  
*Theme color for emphasized font.*
- `help_opt`  
*Whether the help dialogue should be displayed or not.*

### 4.4.1 Detailed Description

The help widget class.

Displays a ui help button that displays the program instructions when clicked.

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 `__init__()`

```
def help.help.__init__ (
    self,
    x,
    y,
    window,
    ui_manager )
```

Initializes the help widget.

##### Parameters

<i>x</i>	Horizontal position to draw the widget at on the ui.
<i>y</i>	Vertical position to draw the widget at on the ui.
<i>window</i>	Ui window to draw the widget to.
<i>ui_manager</i>	Pygame_gui element manager to tie pygame_gui elements to.

### 4.4.3 Member Function Documentation

#### 4.4.3.1 `draw_ui_dynamic()`

```
def help.help.draw_ui_dynamic (
    self )
```

Draws the dynamic ui elements for the help widget.

Draws a dialog with the instructions for using the program.

#### 4.4.3.2 `draw_ui_static()`

```
def help.help.draw_ui_static (
    self )
```

Draws the static ui elements for the help widget.

Draws a button with "help" written on it.

#### 4.4.3.3 `events()`

```
def help.help.events (
    self,
    event )
```

Processes pygame events for the help widget.

If event in the help button being pressed display the instructions dialog.

## Parameters

<i>event</i>	The pygame event being processed.
--------------	-----------------------------------

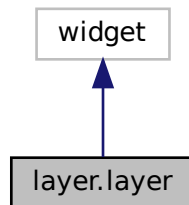
The documentation for this class was generated from the following file:

- [help.py](#)

## 4.5 layer.layer Class Reference

The layer widget class.

Inheritance diagram for layer.layer:



### Public Member Functions

- def `__init__` (self, x, y, window, ui\_manager, layer\_num)  
*Initializes the layer widget.*
- def `change_colors` (self)  
*Change the theme colors.*
- def `clean_layer` (self)  
*Clean the layer by setting it to be blank and see-through.*
- def `draw_canvas` (self)  
*Draw the layer to self.layer based on the current widget settings.*
- def `draw_ui_dynamic` (self)  
*Draws the dynamic ui elements for the layer widget.*
- def `draw_ui_static` (self)  
*Draws the static ui elements for the layer widget.*
- def `events` (self, event)  
*Processes pygame events for the layer widget.*
- def `get_layer_complexity` (self)
- def `get_layer_shape` (self)
- def `get_layer_size` (self)
- def `get_layer_style` (self)
- def `get_layer_transparency` (self)
- def `randomize` (self)  
*Randomize the shape, style, complexity, and size of the layer drawing algorithm.*

## Public Attributes

- [color](#)  
*Theme color for background.*
- [complexity](#)  
*The complexity of the layer's drawing.*
- [complexity\\_lock](#)  
*1 if radomization of the complexity is locked, 0 otherwise*
- [layer](#)  
*The pygame surface the layer draws to.*
- [shape](#)  
*The shape the layer draws.*
- [shape\\_lock](#)  
*1 if radomization of the shape is locked, 0 otherwise*
- [size](#)  
*The size of the drawn shapes.*
- [size\\_lock](#)  
*1 if radomization of the size is locked, 0 otherwise*
- [style](#)  
*The style or pattern the layer draws in.*
- [style\\_lock](#)  
*1 if radomization of the style is locked, 0 otherwise*
- [transparency](#)  
*The transparency of the layer.*
- [transparency\\_lock](#)  
*1 if radomization of the transparency is locked, 0 otherwise*
- [u1\\_h1\\_color](#)  
*Theme color for text.*

### 4.5.1 Detailed Description

The layer widget class.

Provides a ui and functionality to specify a drawing algorithm and draw to a pygame surface.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 `__init__()`

```
def layer.layer.__init__ (
    self,
    x,
    y,
    window,
    ui_manager,
    layer_num )
```

Initializes the layer widget.

## Parameters

<i>x</i>	Horizontal position to draw the widget at on the ui.
<i>y</i>	Vertical position to draw the widget at on the ui.
<i>window</i>	Ui window to draw the widget to.
<i>ui_manager</i>	Pygame_gui element manager to tie pygame_gui elements to.

## 4.5.3 Member Function Documentation

### 4.5.3.1 draw\_ui\_dynamic()

```
def layer.layer.draw_ui_dynamic (
    self )
```

Draws the dynamic ui elements for the layer widget.

Draws the text and lock icons.

### 4.5.3.2 draw\_ui\_static()

```
def layer.layer.draw_ui_static (
    self )
```

Draws the static ui elements for the layer widget.

Draws the shape and style dropdowns, the complexity and size sliders, and lock button.

### 4.5.3.3 events()

```
def layer.layer.events (
    self,
    event )
```

Processes pygame events for the layer widget.

Handles the shape and style dropdowns, the complexity and size sliders, and lock button.

## Parameters

<i>event</i>	The pygame event being processed.
--------------	-----------------------------------

#### 4.5.3.4 `get_layer_complexity()`

```
def layer.layer.get_layer_complexity (
    self )
```

##### Returns

The layer complexity.

#### 4.5.3.5 `get_layer_shape()`

```
def layer.layer.get_layer_shape (
    self )
```

##### Returns

The layer shape.

#### 4.5.3.6 `get_layer_size()`

```
def layer.layer.get_layer_size (
    self )
```

##### Returns

The layer size.

#### 4.5.3.7 `get_layer_style()`

```
def layer.layer.get_layer_style (
    self )
```

##### Returns

The layer style.

#### 4.5.3.8 get\_layer\_transparency()

```
def layer.layer.get_layer_transparency (
    self )
```

##### Returns

The layer transparency

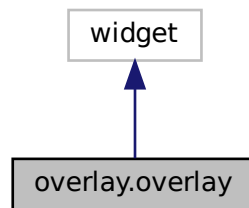
The documentation for this class was generated from the following file:

- [layer.py](#)

## 4.6 overlay.overlay Class Reference

The overlay widget class.

Inheritance diagram for overlay.overlay:



### Public Member Functions

- def [\\_\\_init\\_\\_](#) (self, x, y, window, ui\_manager)  
*Initializes the overlay widget.*
- def [change\\_colors](#) (self)  
*Change the theme colors.*
- def [clean\\_layer](#) (self)  
*Clean the overlay by setting it to be blank and see-through.*
- def [draw\\_canvas](#) (self)  
*Draw the currently selected overlay image to self.overlay\_layer.*
- def [draw\\_ui\\_dynamic](#) (self)  
*Draws the dynamic ui elements for the overlay widget.*
- def [draw\\_ui\\_static](#) (self)  
*Draws the static ui elements for the overlay widget.*
- def [events](#) (self, event)  
*Processes pygame events for the overlay widget.*
- def [get\\_active\\_overlay](#) (self)

## Public Attributes

- [active\\_overlay](#)  
*The currently selected overlay.*
- [color](#)  
*Theme color for background.*
- [overlay\\_layer](#)  
*The pygame surface the overlay draws to.*
- [ui\\_h1\\_color](#)  
*Theme color for text.*

### 4.6.1 Detailed Description

The overlay widget class.

Provides a ui and functionality to specify an overlay and draw to a pygame surface.

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 `__init__()`

```
def overlay.overlay.__init__ (
    self,
    x,
    y,
    window,
    ui_manager )
```

Initializes the overlay widget.

#### Parameters

<i>x</i>	Horizontal position to draw the widget at on the ui.
<i>y</i>	Vertical position to draw the widget at on the ui.
<i>window</i>	Ui window to draw the widget to.
<i>ui_manager</i>	Pygame_gui element manager to tie pygame_gui elements to.

### 4.6.3 Member Function Documentation

#### 4.6.3.1 `draw_ui_dynamic()`

```
def overlay.overlay.draw_ui_dynamic (
    self )
```



Draws the dynamic ui elements for the overlay widget.

Draws the text and overlay thumbnails.

#### 4.6.3.2 draw\_ui\_static()

```
def overlay.overlay.draw_ui_static (
    self )
```

Draws the static ui elements for the overlay widget.

Draws the overlay selection buttons.

#### 4.6.3.3 events()

```
def overlay.overlay.events (
    self,
    event )
```

Processes pygame events for the overlay widget.

Handles the overlay selection buttons.

##### Parameters

<i>event</i>	The pygame event being processed.
--------------	-----------------------------------

#### 4.6.3.4 get\_active\_overlay()

```
def overlay.overlay.get_active_overlay (
    self )
```

##### Returns

The active overlay.

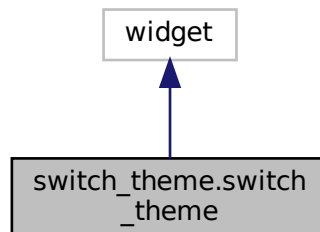
The documentation for this class was generated from the following file:

- [overlay.py](#)

## 4.7 switch\_theme.switch\_theme Class Reference

The theme switch widget class.

Inheritance diagram for switch\_theme.switch\_theme:



### Public Member Functions

- `def __init__ (self, x, y, window, ui_manager)`  
*Initializes the theme widget.*
- `def draw_ui_static (self)`  
*Draws the static ui elements for the theme widget.*
- `def events (self, event)`  
*Processes pygame events for the theme widget.*
- `def getDarkMode (self)`

### Public Attributes

- `switch_theme_dark`  
*True if in dark mode, False if in light.*

### 4.7.1 Detailed Description

The theme switch widget class.

Displays a ui switch theme button that changes the interface theme when clicked.

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 \_\_init\_\_()

```
def switch_theme.switch_theme.__init__ (
    self,
    x,
    y,
    window,
    ui_manager )
```

Initializes the theme widget.

## Parameters

<i>x</i>	Horizontal position to draw the widget at on the ui.
<i>y</i>	Vertical position to draw the widget at on the ui.
<i>window</i>	Ui window to draw the widget to.
<i>ui_manager</i>	Pygame_gui element manager to tie pygame_gui elements to.

### 4.7.3 Member Function Documentation

#### 4.7.3.1 draw\_ui\_static()

```
def switch_theme.switch_theme.draw_ui_static (
    self )
```

Draws the static ui elements for the theme widget.

Draws a button with "theme" written on it.

#### 4.7.3.2 events()

```
def switch_theme.switch_theme.events (
    self,
    event )
```

Processes pygame events for the theme widget.

If event is the switch theme button being pressed change the interface theme.

## Parameters

<i>event</i>	The pygame event being processed.
--------------	-----------------------------------

#### 4.7.3.3 getDarkMode()

```
def switch_theme.switch_theme.getDarkMode (
    self )
```

## Returns

True if in dark mode, False if in light mode.

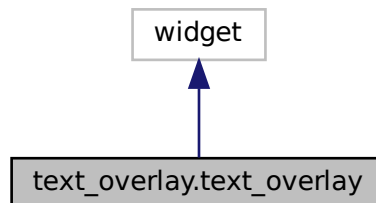
The documentation for this class was generated from the following file:

- [switch\\_theme.py](#)

## 4.8 text\_overlay.text\_overlay Class Reference

The text overlay widget class.

Inheritance diagram for text\_overlay.text\_overlay:



### Public Member Functions

- `def __init__ (self, x, y, window, ui_manager)`  
*Initializes the text overlay widget.*
- `def change_colors (self)`  
*Change the theme colors.*
- `def clean_layer (self)`  
*Clean the layer by setting it to be blank and see-through.*
- `def draw_canvas (self)`  
*Randomizes the text color and draws text to self.layer based on the current widget settings.*
- `def draw_ui_dynamic (self)`  
*Draws the dynamic ui elements for the text overlay widget.*
- `def draw_ui_static (self)`  
*Draws the static ui elements for the text overlay widget.*
- `def events (self, event)`  
*Processes pygame events for the text overlay widget.*
- `def text_to_canvas (self)`  
*Draws text to self.layer based on the current widget settings.*

### Public Attributes

- `bg_color`  
*Theme color for background.*
- `color`  
*The color to draw the text with.*
- `font`  
*The font used to draw text.*
- `layer`  
*The pygame surface the text overlay draws to.*
- `pos`

*The x and y position of the text on the canvas.*

- [size](#)

*The size of the text.*

- [text](#)

*The text to draw.*

- [ui\\_color](#)

*Theme color for smaller text.*

- [ui\\_h1\\_color](#)

*Theme color for text.*

## 4.8.1 Detailed Description

The text overlay widget class.

Provides a ui and functionality to specify text to be drawn to a pygame surface.

## 4.8.2 Constructor & Destructor Documentation

### 4.8.2.1 \_\_init\_\_()

```
def text_overlay.text_overlay.__init__ (
    self,
    x,
    y,
    window,
    ui_manager )
```

Initializes the text overlay widget.

#### Parameters

<i>x</i>	Horizontal position to draw the widget at on the ui.
<i>y</i>	Vertical position to draw the widget at on the ui.
<i>window</i>	Ui window to draw the widget to.
<i>ui_manager</i>	Pygame_gui element manager to tie pygame_gui elements to.

## 4.8.3 Member Function Documentation

### 4.8.3.1 draw\_ui\_dynamic()

```
def text_overlay.text_overlay.draw_ui_dynamic (
    self )
```

Draws the dynamic ui elements for the text overlay widget.

Draws the text.

#### 4.8.3.2 draw\_ui\_static()

```
def text_overlay.text_overlay.draw_ui_static (
    self )
```

Draws the static ui elements for the text overlay widget.

Draws the font dropdown, size and position sliders, and the text entry box.

#### 4.8.3.3 events()

```
def text_overlay.text_overlay.events (
    self,
    event )
```

Processes pygame events for the text overlay widget.

Handles the font dropdown, size and position sliders, and the text entry box.

##### Parameters

<i>event</i>	The pygame event being processed.
--------------	-----------------------------------

The documentation for this class was generated from the following file:

- [text\\_overlay.py](#)

## 4.9 ui\_controller.ui\_controller Class Reference

The [ui\\_controller](#) class.

### Public Member Functions

- def [\\_\\_init\\_\\_](#) (self)  
*Initializes [ui\\_controller](#).*
- def [draw\\_ui\\_dynamic](#) (self)  
*Draws the dynamic ui.*
- def [draw\\_ui\\_static](#) (self)  
*Draws the static ui.*
- def [export\\_art](#) (self)  
*Exports the canvas to a png image.*
- def [process\\_events](#) (self)  
*Processes pygame events.*
- def [run](#) (self)  
*Main loop.*

## Public Attributes

- `canvas`  
*The canvas to draw the generated art on.*
- `export_resolution`  
*Current canvas export resolution.*
- `isrunning`  
*A boolean that specifies if the program is running, program terminates if False.*
- `ui_manager`  
*Manages pygame\_gui elements and events.*
- `window`  
*The program window.*

## Static Public Attributes

- tuple `canvas_display_size` = (int(`SW`//1.8), int(`SH`//1.8))  
*Canvas ui display port size.*
- tuple `canvas_pos` = ((`SW` - `canvas_display_size`[0])//2, (`SH` - `canvas_display_size`[1])//2)  
*Position of the canvas on the ui.*
- tuple `canvas_size` = (3840, 2160)  
*Canvas internal size.*
- tuple `help_pos` = (284, 60)  
*Position of the help widget.*
- tuple `layer_one_pos` = (`_ui_menus_left`, 30)  
*Position of the layer one widget.*
- tuple `layer_three_pos` = (`_ui_menus_left`, `layer_two_pos`[1]+230)  
*Position of the layer three widget.*
- tuple `layer_two_pos` = (`_ui_menus_left`, `layer_one_pos`[1]+230)  
*Position of the layer two widget.*
- tuple `overlay_pos` = (`palette_pos`[0], `palette_pos`[1]+145)  
*Position of the overlay widget.*
- tuple `palette_pos` = (`_ui_menus_right`, 15)  
*Position of the color palette widget.*
- list `resolutions_list`  
*Possible canvas export resolutions.*
- int `SH` = 720  
*Application window height.*
- int `SW` = 1280  
*Application window width.*
- tuple `switch_theme_pos` = (284, 90)  
*Position of the switch theme widget.*
- tuple `text_overlay_pos` = (`_ui_menus_right`, `overlay_pos`[1]+360)  
*Position of the text overlay widget.*

### 4.9.1 Detailed Description

The `ui_controller` class.

A high level class that calls all other modules. Orchestrates pygame event handling, ui drawing, art generation, randomization, and art exporting.

## 4.9.2 Constructor & Destructor Documentation

### 4.9.2.1 `__init__()`

```
def ui_controller.ui_controller.__init__ (
    self )
```

Initializes [ui\\_controller](#).

Initializes pygame, the application window, the canvas, and all widgets.

## 4.9.3 Member Function Documentation

### 4.9.3.1 `draw_ui_dynamic()`

```
def ui_controller.ui_controller.draw_ui_dynamic (
    self )
```

Draws the dynamic ui.

Draws the background color and some text itself and calls canvas and widgets for all other drawing.

### 4.9.3.2 `draw_ui_static()`

```
def ui_controller.ui_controller.draw_ui_static (
    self )
```

Draws the static ui.

Draws generation and export controls itself and calls widgets for all other drawing.

### 4.9.3.3 `process_events()`

```
def ui_controller.ui_controller.process_events (
    self )
```

Processes pygame events.

Handles generation and export controls itself and calls events() in widgets for all other event processing.



#### 4.9.3.4 run()

```
def ui_controller.ui_controller.run (
    self )
```

Main loop.

Draws static ui then enters loop where it processes events and draws the dynamic ui.

### 4.9.4 Member Data Documentation

#### 4.9.4.1 resolutions\_list

```
list ui_controller.ui_controller.resolutions_list [static]
```

**Initial value:**

```
= [
    "4K: 3840x2160",
    "Full HD: 1920x1080",
    "HD: 1280x720"
]
```

Possible canvas export resolutions.

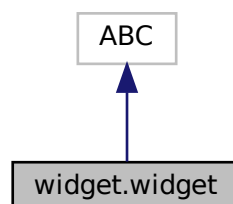
The documentation for this class was generated from the following file:

- [ui\\_controller.py](#)

## 4.10 widget.widget Class Reference

An abstract class for widgets to extend.

Inheritance diagram for widget.widget:



## Public Member Functions

- def `__init__` (self, x, y, window, ui\_manager)  
*Initializes the widget.*
- def `draw_canvas` (self)  
*Draw to the canvas.*
- def `draw_ui_dynamic` (self)  
*Draw ui elements that need to be refreshed each frame.*
- def `draw_ui_static` (self)  
*Draw ui elements that only need to be drawn once.*
- def `events` (self, event)  
*Handle pygame events for the widget.*
- def `randomize` (self)  
*Randomize the widget settings.*
- def `refresh_ui_static` (self)  
*Refresh the static ui elements.*

### 4.10.1 Detailed Description

An abstract class for widgets to extend.

Provides an interface widgets typically use.

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 `__init__()`

```
def widget.widget.__init__ (
    self,
    x,
    y,
    window,
    ui_manager )
```

Initializes the widget.

#### Parameters

<i>x</i>	Horizontal position to draw the widget at on the ui.
<i>y</i>	Vertical position to draw the widget at on the ui.
<i>window</i>	Ui window to draw the widget to.
<i>ui_manager</i>	Pygame_gui element manager to tie pygame_gui elements to.

### 4.10.3 Member Function Documentation

#### 4.10.3.1 draw\_ui\_dynamic()

```
def widget.widget.draw_ui_dynamic (
    self )
```

Draw ui elements that need to be refreshed each frame.

For our purposes draws everything that isn't a pygame\_gui element.

#### 4.10.3.2 draw\_ui\_static()

```
def widget.widget.draw_ui_static (
    self )
```

Draw ui elements that only need to be drawn once.

For our purposes draws pygame\_gui elements.

#### 4.10.3.3 events()

```
def widget.widget.events (
    self,
    event )
```

Handle pygame events for the widget.

##### Parameters

<i>event</i>	The event to be processed.
--------------	----------------------------

The documentation for this class was generated from the following file:

- [widget.py](#)

## 4.11 widget\_storage.widget\_storage Class Reference

Storage for program widgets.

### Public Member Functions

- `def __init__(self)`

## Public Attributes

- [color\\_palette](#)  
*The color palette widget.*
- [generators](#)  
*The generators utility.*
- [help](#)  
*The help button widget.*
- [layer\\_one](#)  
*The layer one widget.*
- [layer\\_three](#)  
*The layer three widget.*
- [layer\\_two](#)  
*The layer two widget.*
- [overlay](#)  
*The overlay widget.*
- [switch\\_theme](#)  
*The theme switch widget.*
- [text\\_overlay](#)  
*The text overlay widget.*

### 4.11.1 Detailed Description

Storage for program widgets.

Allows all other modules to access program widgets.

The documentation for this class was generated from the following file:

- [widget\\_storage.py](#)

## Chapter 5

# File Documentation

### 5.1 assets.py File Reference

Stores various assets useful to other modules.

#### Functions

- def `assets.text_to_screen` (window, text, color, pos, font\_size)  
*Draws text to ui.*

#### Variables

- tuple `assets.active_color` = (90, 90, 90)  
*Color used to indicate active settings.*
- `assets.background_color` = pg.Color("#322f3d")  
*Background\_color of ui.*
- list `assets.font_sizes` = [12, 14, 18, 24, 30, 40]  
*Font size numbers that correspond with defined font sizes.*
- list `assets.fonts` = [xs\_font, small\_font, medium\_font, large\_font, xl\_font, xxl\_font]  
*List of font sizes.*
- tuple `assets.inactive_color` = (20, 20, 20)  
*Color used to indicate inactive settings.*
- `assets.large_font` = pg.freetype.Font("fonts/Basic.ttf", 24)  
*Large font.*
- `assets.lock_disabled` = pg.transform.scale(pg.image.load("assets/lock\_disabled.png"), (20, 20))  
*Lock disabled graphic.*
- `assets.lock_enabled` = pg.transform.scale(pg.image.load("assets/lock\_enabled.png"), (20, 20))  
*Lock enabled graphic.*
- `assets.logo` = pg.image.load("assets/logo.png")  
*Program logo.*
- `assets.medium_font` = pg.freetype.Font("fonts/Basic.ttf", 18)  
*Medium font.*
- `assets.small_font` = pg.freetype.Font("fonts/Basic.ttf", 14)  
*Small font.*

- `assets.ui_color` = `pg.Color("#DFD6FF")`  
*Color used for smaller text elements.*
- tuple `assets.ui_h1_color` = (250, 250, 250)  
*Color used for larger text elements.*
- `assets.xl_font` = `pg.freetype.Font("fonts/Basic.ttf", 30)`  
*Extra large font.*
- `assets.xs_font` = `pg.freetype.Font("fonts/Basic.ttf", 12)`  
*Extra small font.*
- `assets.xxl_font` = `pg.freetype.Font("fonts/Basic.ttf", 40)`  
*Extra extra large font.*

### 5.1.1 Detailed Description

Stores various assets useful to other modules.

### 5.1.2 Author(s)

- Created by Jessica Dawson on 03/16/2022.

### 5.1.3 Function Documentation

#### 5.1.3.1 `text_to_screen()`

```
def assets.text_to_screen (
    window,
    text,
    color,
    pos,
    font_size )
```

Draws text to ui.

#### Parameters

<i>window</i>	Ui window to draw to.
<i>text</i>	Text to draw.
<i>color</i>	Color of text.
<i>pos</i>	Position of text.
<i>font_size</i>	Size of text.

## 5.2 `canvas.py` File Reference

Defines the canvas class.

## Classes

- class [canvas.canvas](#)  
*The canvas class.*

### 5.2.1 Detailed Description

Defines the canvas class.

### 5.2.2 Author(s)

- Created by Jessica Dawson on 03/16/2022.

## 5.3 color\_palette.py File Reference

Defines the color\_palette class.

## Classes

- class [color\\_palette.color\\_palette](#)  
*The color palette widget class.*

### 5.3.1 Detailed Description

Defines the color\_palette class.

### 5.3.2 Author(s)

- Created by Jessica Dawson on 03/16/2022.

## 5.4 generators.py File Reference

Defines the generator class which houses the various layer generation algorithms.

## Classes

- class [generators.generators](#)  
*Generators class that provides layer generation functionality to the layer module.*

## Variables

- list `generators.art_styles_list`

### 5.4.1 Detailed Description

Defines the generator class which houses the various layer generation algorithms.

### 5.4.2 Author(s)

- Created by Jessica Dawson on 03/17/2022.

### 5.4.3 Variable Documentation

#### 5.4.3.1 `art_styles_list`

```
list generators.art_styles_list
```

##### Initial value:

```
1 = [  
2     "Chaotic",  
3     "Striped Horizontal",  
4     "Striped Vertical",  
5     "Mosaic",  
6     "Cornered",  
7     "Centered",  
8     "Empty"  
9 ]
```

## 5.5 `help.py` File Reference

Defines the help class.

## Classes

- class `help.help`  
*The help widget class.*

### 5.5.1 Detailed Description

Defines the help class.

### 5.5.2 Author(s)

- Created by Jessica Dawson on 03/16/2022.



## 5.6 layer.py File Reference

Defines the layer class.

### Classes

- class [layer.layer](#)  
*The layer widget class.*

#### 5.6.1 Detailed Description

Defines the layer class.

#### 5.6.2 Author(s)

- Created by Aamina Hussain on 03/17/2022.

## 5.7 overlay.py File Reference

Defines the overlay class.

### Classes

- class [overlay.overlay](#)  
*The overlay widget class.*

#### 5.7.1 Detailed Description

Defines the overlay class.

#### 5.7.2 Author(s)

- Created by Jessica Dawson on 03/17/2022.
- Modified by Aamina Hussain on 04/05/2022.

## 5.8 switch\_theme.py File Reference

Defines the switch theme class.

## Classes

- class [switch\\_theme.switch\\_theme](#)  
*The theme switch widget class.*

### 5.8.1 Detailed Description

Defines the switch theme class.

### 5.8.2 Author(s)

- Created by Fady Morcos on 04/04/2022.

## 5.9 text\_overlay.py File Reference

Defines the text\_overlay class.

## Classes

- class [text\\_overlay.text\\_overlay](#)  
*The text overlay widget class.*

### 5.9.1 Detailed Description

Defines the text\_overlay class.

### 5.9.2 Author(s)

- Created by Jessica Dawson on 03/22/2022.

## 5.10 ui\_controller.py File Reference

Defines and initializes the ui\_controller class.

## Classes

- class [ui\\_controller.ui\\_controller](#)  
*The ui\_controller class.*

## Variables

- `ui_controller.controller` = `ui_controller()`  
*The ui\_controller instance that initializes the program.*

### 5.10.1 Detailed Description

Defines and initializes the ui\_controller class.

### 5.10.2 Author(s)

- Created by Jessica Dawson on 03/16/2022.
- Modified by Aamina Hussain on 03/17/2022.

## 5.11 widget.py File Reference

Defines the widget abstract class.

## Classes

- class `widget.widget`  
*An abstract class for widgets to extend.*

### 5.11.1 Detailed Description

Defines the widget abstract class.

### 5.11.2 Author(s)

- Created by Jessica Dawson on 03/16/2022.

## 5.12 widget\_storage.py File Reference

Defines and initializes the widget\_storage class.

## Classes

- class `widget_storage.widget_storage`  
*Storage for program widgets.*

## Variables

- `widget_storage.widgets` = None

*Instance of widget\_storage to access widgets through.*

### 5.12.1 Detailed Description

Defines and initializes the widget\_storage class.

### 5.12.2 Author(s)

- Created by Jessica Dawson on 03/16/2022.
- Modified by Aamina Hussain on 03/17/2022.

### 5.12.3 Variable Documentation

#### 5.12.3.1 widgets

```
widget_storage.widgets = None
```

Instance of widget\_storage to access widgets through.

Import this instance and access widgets with `widgets.widget_name`

# Index

- `__init__`
  - `canvas.canvas`, 8
  - `color_palette.color_palette`, 11
  - `generators.generators`, 14
  - `help.help`, 19
  - `layer.layer`, 22
  - `overlay.overlay`, 26
  - `switch_theme.switch_theme`, 28
  - `text_overlay.text_overlay`, 31
  - `ui_controller.ui_controller`, 34
  - `widget.widget`, 36
- `art_styles_list`
  - `generators.py`, 42
- `assets.py`, 39
  - `text_to_screen`, 40
- `canvas.canvas`, 7
  - `__init__`, 8
  - `draw_layers`, 8
  - `draw_to_canvas`, 8
  - `generate_bg`, 9
  - `get_canvas`, 9
- `canvas.py`, 40
- `color_palette.color_palette`, 9
  - `__init__`, 11
  - `draw_ui_dynamic`, 11
  - `draw_ui_static`, 12
  - `events`, 12
  - `get_background_color`, 12
  - `get_colors_from_palette`, 12
  - `get_foreground_colors`, 12
  - `get_name_of_palette`, 13
  - `refresh_ui_static`, 13
- `color_palette.py`, 41
- `draw_circles`
  - `generators.generators`, 15
- `draw_curves`
  - `generators.generators`, 15
- `draw_dots`
  - `generators.generators`, 15
- `draw_fpolygons`
  - `generators.generators`, 16
- `draw_hpolygons`
  - `generators.generators`, 16
- `draw_layers`
  - `canvas.canvas`, 8
- `draw_lines`
  - `generators.generators`, 17
- `draw_rings`
  - `generators.generators`, 17
- `draw_squares`
  - `generators.generators`, 18
- `draw_to_canvas`
  - `canvas.canvas`, 8
- `draw_ui_dynamic`
  - `color_palette.color_palette`, 11
  - `help.help`, 20
  - `layer.layer`, 23
  - `overlay.overlay`, 26
  - `text_overlay.text_overlay`, 31
  - `ui_controller.ui_controller`, 34
  - `widget.widget`, 37
- `draw_ui_static`
  - `color_palette.color_palette`, 12
  - `help.help`, 20
  - `layer.layer`, 23
  - `overlay.overlay`, 27
  - `switch_theme.switch_theme`, 29
  - `text_overlay.text_overlay`, 32
  - `ui_controller.ui_controller`, 34
  - `widget.widget`, 37
- `events`
  - `color_palette.color_palette`, 12
  - `help.help`, 20
  - `layer.layer`, 23
  - `overlay.overlay`, 27
  - `switch_theme.switch_theme`, 29
  - `text_overlay.text_overlay`, 32
  - `widget.widget`, 37
- `generate_bg`
  - `canvas.canvas`, 9
- `generators.generators`, 13
  - `__init__`, 14
  - `draw_circles`, 15
  - `draw_curves`, 15
  - `draw_dots`, 15
  - `draw_fpolygons`, 16
  - `draw_hpolygons`, 16
  - `draw_lines`, 17
  - `draw_rings`, 17
  - `draw_squares`, 18
- `generators.py`, 41
  - `art_styles_list`, 42
- `get_active_overlay`
  - `overlay.overlay`, 27
- `get_background_color`

- color\_palette.color\_palette, 12
- get\_canvas
  - canvas.canvas, 9
- get\_colors\_from\_palette
  - color\_palette.color\_palette, 12
- get\_foreground\_colors
  - color\_palette.color\_palette, 12
- get\_layer\_complexity
  - layer.layer, 23
- get\_layer\_shape
  - layer.layer, 24
- get\_layer\_size
  - layer.layer, 24
- get\_layer\_style
  - layer.layer, 24
- get\_layer\_transparency
  - layer.layer, 24
- get\_name\_of\_palette
  - color\_palette.color\_palette, 13
- getDarkMode
  - switch\_theme.switch\_theme, 29
- help.help, 18
  - \_\_init\_\_, 19
  - draw\_ui\_dynamic, 20
  - draw\_ui\_static, 20
  - events, 20
- help.py, 42
- layer.layer, 21
  - \_\_init\_\_, 22
  - draw\_ui\_dynamic, 23
  - draw\_ui\_static, 23
  - events, 23
  - get\_layer\_complexity, 23
  - get\_layer\_shape, 24
  - get\_layer\_size, 24
  - get\_layer\_style, 24
  - get\_layer\_transparency, 24
- layer.py, 43
- overlay.overlay, 25
  - \_\_init\_\_, 26
  - draw\_ui\_dynamic, 26
  - draw\_ui\_static, 27
  - events, 27
  - get\_active\_overlay, 27
- overlay.py, 43
- process\_events
  - ui\_controller.ui\_controller, 34
- refresh\_ui\_static
  - color\_palette.color\_palette, 13
- resolutions\_list
  - ui\_controller.ui\_controller, 35
- run
  - ui\_controller.ui\_controller, 34
- switch\_theme.py, 43
- switch\_theme.switch\_theme, 28
  - \_\_init\_\_, 28
  - draw\_ui\_static, 29
  - events, 29
  - getDarkMode, 29
- text\_overlay.py, 44
- text\_overlay.text\_overlay, 30
  - \_\_init\_\_, 31
  - draw\_ui\_dynamic, 31
  - draw\_ui\_static, 32
  - events, 32
- text\_to\_screen
  - assets.py, 40
- ui\_controller.py, 44
- ui\_controller.ui\_controller, 32
  - \_\_init\_\_, 34
  - draw\_ui\_dynamic, 34
  - draw\_ui\_static, 34
  - process\_events, 34
  - resolutions\_list, 35
  - run, 34
- widget.py, 45
- widget.widget, 35
  - \_\_init\_\_, 36
  - draw\_ui\_dynamic, 37
  - draw\_ui\_static, 37
  - events, 37
- widget\_storage.py, 45
  - widgets, 46
- widget\_storage.widget\_storage, 37
- widgets
  - widget\_storage.py, 46