Hamilton College CS110 Graphics Library

1.0

Generated by Doxygen 1.6.1

Fri Jul 7 11:02:21 2017

Contents

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 Package cs110graphics

Contains a CSPy-friendly version of a Tkinter based graphics library.

Classes

- class Window

  This window acts as a canvas which other objects can be put onto.

- class Event

  An event which gets bound to an object.

- class EventHandler

  Handles an event.

- class GraphicalObject

  This window is a parent class of any object which can be put into Window.

- class Fillable

  This window is a parent class of any object which can have its colors modified.

- class Image

  An image, which can be added to a Window object.

- class Text

  Text which can be added to a Window object.

- class Polygon

  A Polygon, which can be added to a Window object.

- class Circle

  A circle, which can be added to a Window object.

- class Oval

An oval, which can be added to a Window object.

- class Square

  A square, which can be added to a Window object.

- class Rectangle

  A rectangle, which can be added to a Window object.

- class Timer

  A class which continually runs a function after a delay.

- class _RunWithYieldDelay

  A class which uses a function which returns a generator to rerun until the generator stops generating numbers.

Functions

- def StartGraphicsSystem

  This initalizes the graphics system.

- def RunWithYieldDelay

  A wrapper for the _RunWithYieldDelay class.

4.1.1 Detailed Description

Contains a CSPy-friendly version of a Tkinter based graphics library. Paul Magnus '18, Ines Ayara '20, Matthew R. Jenkins '20

Summer 2017

4.1.2 Function Documentation

4.1.2.1 def cs110graphics.RunWithYieldDelay (*window*, *func*)

A wrapper for the _RunWithYieldDelay class. THIS SHOULD BE USED INSTEAD OF CREATING AN _RunWithYieldDelay INSTANCE.

Required Parameters:

- window - Window
- func - function which returns a generator of int

4.1.2.2 def cs110graphics.StartGraphicsSystem (*first_function*, *width* = `400`, *height* = `400`, *background* = `"white"`, *name* = `"Graphics Window"`)

This initalizes the graphics system. Required Parameters:

- first_function - func

Optional Parameters:

- width - int
- height - int
- background - string
- name - string

Chapter 5

Class Documentation

5.1 cs110graphics._RunWithYieldDelay Class Reference

A class which uses a function which returns a generator to rerun until the generator stops generating numbers.

Public Member Functions

- def **__init__**

5.1.1 Detailed Description

A class which uses a function which returns a generator to rerun until the generator stops generating numbers. NOTE: DO NOT INITALIZE THIS CLASS ANYWHERE IN YOUR PROGRAM. THE WRAPPER FUNCTION RunWithYieldDelay SHOULD BE USED INSTEAD.

Required Parameters:

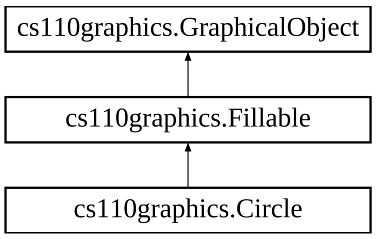
- window - Window - the window which the object with yield delay is on.
- func - function which returns a generator of int - a function with a few necessary parameters which allow it to run with yield delay. A function needs to return a generator of int, needs a yield statement with an int which represents the delay (in milliseconds), and it needs a raise StopIteration statement at the end of the function.

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

- cs110graphics.py

5.2 cs110graphics.Circle Class Reference

A circle, which can be added to a Window object. Inheritance diagram for cs110graphics.Circle::



Public Member Functions

- def **__init__**
- def set_radius

  Sets the radius of the Circle.

5.2.1 Detailed Description

A circle, which can be added to a Window object. Required Parameters:

- window - Window - the window which the object will be added to.

Optional Parameters:

- radius - int - sets the radius of the Circle. (default: 40)
- center - tuple - sets the center of the Circle. (default: (200, 200))

5.2.2 Member Function Documentation

5.2.2.1 def cs110graphics.Circle.set_radius (*self*, *radius*)

Sets the radius of the Circle. Required Parameters:

- radius - int

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

- cs110graphics.py

5.3 cs110graphics.Event Class Reference

An event which gets bound to an object.

Public Member Functions

- def **__init__**
- def get_button

  Returns the mouse button that is attached to the event.
- def get_description

  Returns the description of the event.
- def get_key

  Returns the keyboard key that is attached to the event.
- def get_mouse_location

  Returns a tuple of the x and y coordinates of the mouse location in the canvas.
- def get_root_mouse_location

  Returns a tuple of the x and y coordinates of the mouse location in the window.

5.3.1 Detailed Description

An event which gets bound to an object. Used by EventHandler objects.

Required Parameters:

- event - TkEvent - The event which the user want applied an an object.

5.3.2 Member Function Documentation

5.3.2.1 def cs110graphics.Event.get_button (*self*)

Returns the mouse button that is attached to the event. Returns None if the button fails to exist (like if the Event handles a key press).

5.3.2.2 def cs110graphics.Event.get_description (*self*)

Returns the description of the event.

5.3.2.3 def cs110graphics.Event.get_key (*self*)

Returns the keyboard key that is attached to the event. Returns None if the key fails to exist (like if the Event handles a mouse press).

5.3.2.4 def cs110graphics.Event.get_mouse_location (*self*)

Returns a tuple of the x and y coordinates of the mouse location in the canvas.

5.3.2.5 def cs110graphics.Event.get_root_mouse_location (*self*)

Returns a tuple of the x and y coordinates of the mouse location in the window.

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

- cs110graphics.py

5.4 cs110graphics.EventHandler Class Reference

Handles an event.

Public Member Functions

- def **__init__**
- def handle_key_press

  Handles a key press.
- def handke_key_release

  Handles a key release.
- def handle_mouse_enter

  Handles when a mouse enters an object.
- def handle_mouse_leave

  Handles when a mouse leaves an object.
- def handle_mouse_move

  Handles a mouse move.
- def handle_mouse_press

  Handles a mouse press.
- def handle_mouse_release

  Handles a mouse release.

5.4.1 Detailed Description

Handles an event. These are overloaded by the user, so by default they're empty except for the pass command.

5.4.2 Member Function Documentation

5.4.2.1 def cs110graphics.EventHandler.handke_key_release (*self*, *event*)

Handles a key release. Optional Parameters:

- event - Event - when included, you can use any Event method whenever this function is run.

5.4.2.2 def cs110graphics.EventHandler.handle_key_press (*self*, *event*)

Handles a key press. Optional Parameters:

- event - Event - when included, you can use any Event method whenever this function is run.

5.4.2.3 def cs110graphics.EventHandler.handle_mouse_enter (*self*, *event*)

Handles when a mouse enters an object. Optional Parameters:

- event - Event - when included, you can use any Event method whenever this function is run.

5.4.2.4 def cs110graphics.EventHandler.handle_mouse_leave (*self*, *event*)

Handles when a mouse leaves an object. Optional Parameters:

- event - Event - when included, you can use any Event method whenever this function is run.

5.4.2.5 def cs110graphics.EventHandler.handle_mouse_move (*self*, *event*)

Handles a mouse move. Optional Parameters:

- event - Event - when included, you can use any Event method whenever this function is run.

5.4.2.6 def cs110graphics.EventHandler.handle_mouse_press (*self*, *event*)

Handles a mouse press. Optional Parameters:

- event - Event - when included, you can use any Event method whenever this function is run.

5.4.2.7 def cs110graphics.EventHandler.handle_mouse_release (*self*, *event*)

Handles a mouse release. Optional Parameters:

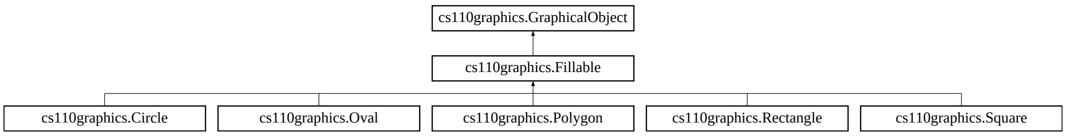
- event - Event - when included, you can use any Event method whenever this function is run.

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

- cs110graphics.py

5.5 cs110graphics.Fillable Class Reference

This window is a parent class of any object which can have its colors modified. Inheritance diagram for cs110graphics.Fillable::



Public Member Functions

- def **__init__**
- def get_border_color

    Returns the border color of a Fillable.

- def get_border_width

    Returns the border width of a Fillable.

- def get_fill_color

    Returns the depth of a Fillable.

- def get_pivot

    Returns the pivot point of a Fillable.

- def rotate

    Rotates the object.

- def scale

    Scales the Fillable up or down depending on the factor.

- def set_border_color

    Sets the border color of the Fillable.

- def set_border_width

    Sets the border width of the Fillable.

- def set_fill_color

    Sets the fill color of the Fillable.

- def set_pivot

    Sets the pivot point of the Fillable.

5.5.1 Detailed Description

This window is a parent class of any object which can have its colors modified. No constructor exists in this class, but its methods are used by other objects that extend/inherit this class.

5.5.2 Member Function Documentation

5.5.2.1 def cs110graphics.Fillable.get_border_color (*self*)

Returns the border color of a Fillable.

5.5.2.2 def cs110graphics.Fillable.get_border_width (*self*)

Returns the border width of a Fillable.

5.5.2.3 def cs110graphics.Fillable.get_fill_color (*self*)

Returns the depth of a Fillable.

5.5.2.4 def cs110graphics.Fillable.get_pivot (*self*)

Returns the pivot point of a Fillable.

5.5.2.5 def cs110graphics.Fillable.rotate (*self*, *degrees*)

Rotates the object. Required Parameters:

- degrees - int

5.5.2.6 def cs110graphics.Fillable.scale (*self*, *factor*)

Scales the Fillable up or down depending on the factor. Required Parameters:

- factor - float

5.5.2.7 def cs110graphics.Fillable.set_border_color (*self*, *color*)

Sets the border color of the Fillable. Required Parameters:

- color - string

5.5.2.8 def cs110graphics.Fillable.set_border_width (*self*, *width*)

Sets the border width of the Fillable. Required Parameters:

- width - int

5.5.2.9 def cs110graphics.Fillable.set_fill_color (*self*, *color*)

Sets the fill color of the Fillable. Required Parameters:

- color - string

5.5.2.10 def cs110graphics.Fillable.set_pivot (*self*, *pivot*)

Sets the pivot point of the Fillable. Required Parameters:

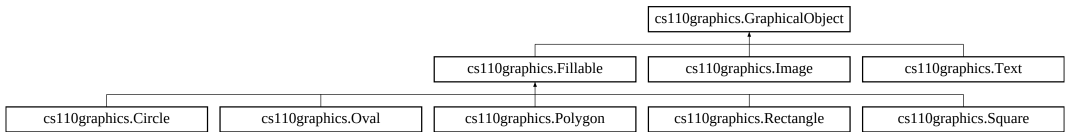
- pivot - tuple of (int $*$ int)

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

- cs110graphics.py

5.6 cs110graphics.GraphicalObject Class Reference

This window is a parent class of any object which can be put into Window. Inheritance diagram for cs110graphics.GraphicalObject::



Public Member Functions

- def **__init__**
- def add_handler

  Adds a handler to the graphical object.
- def get_center

  Returns the center of the graphical object.
- def get_depth

  Returns the depth of the graphical object.
- def move

  Moves a graphical object dx pixels horizontally and dy pixels vertically.
- def move_to

  Moves a graphical object to a point.
- def set_depth

  Sets the depth of the GraphicalObject.

5.6.1 Detailed Description

This window is a parent class of any object which can be put into Window. No constructor exists in this class, but its methods are used by other objects that extend/inherit this class.

5.6.2 Member Function Documentation

5.6.2.1 def cs110graphics.GraphicalObject.add_handler (*self*, *handler_object*)

Adds a handler to the graphical object. Required Parameters:

- handler_object - an object with a GraphicalObject representation within it (such as an object which has a Circle object in it)

5.6.2.2 def cs110graphics.GraphicalObject.get_center (*self*)

Returns the center of the graphical object.

5.6.2.3 def cs110graphics.GraphicalObject.get_depth (*self*)

Returns the depth of the graphical object.

5.6.2.4 def cs110graphics.GraphicalObject.move (*self*, *dx*, *dy*)

Moves a graphical object dx pixels horizontally and dy pixels vertically. Required Parameters:

- dx - int
- dy - int

Reimplemented in cs110graphics.Image, and cs110graphics.Text.

5.6.2.5 def cs110graphics.GraphicalObject.move_to (*self*, *point*)

Moves a graphical object to a point. Required Parameters:

- point - tuple of (int ∗ int)

Reimplemented in cs110graphics.Image, and cs110graphics.Text.

5.6.2.6 def cs110graphics.GraphicalObject.set_depth (*self*, *depth*)

Sets the depth of the GraphicalObject. Required Parameters:

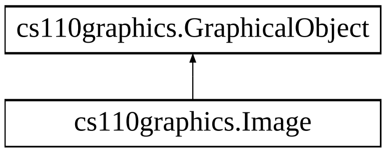
- depth - int

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

- cs110graphics.py

5.7 cs110graphics.Image Class Reference

An image, which can be added to a Window object. Inheritance diagram for cs110graphics.Image::



Public Member Functions

- def **__init__**
- def move

  Moves a graphical object dx pixels horizontally and dy pixels vertically.

- def move_to

  Moves a graphical object to a point.

- def resize

  Resizes the Image.

- def rotate

  Rotates an object by degrees.

- def scale

  Scales the image according to the factor.

- def size

  Returns a tuple of the width and height of the image.

5.7.1 Detailed Description

An image, which can be added to a Window object. Required Parameters:

- window - Window - the window which the object will be added to.
- image_loc - str - The name of an image within the current working directory . (If the current working directory is /foo/bar, then the image the user wants to use has to be in that directory. There is no support for using internet links at this time.)

Optional Parameters:

- center - tuple of int $*$ int - sets the center of the Image. (default: (200, 200))
- width - int - sets the width of the image. (default: 25)
- height - int - sets the height of the image. (default: 25)

5.7.2 Member Function Documentation

5.7.2.1 def cs110graphics.Image.move (*self*, *dx*, *dy*)

Moves a graphical object dx pixels horizontally and dy pixels vertically. Required Parameters:

- dx - int
- dy - int

Reimplemented from cs110graphics.GraphicalObject.

5.7.2.2 def cs110graphics.Image.move_to (*self*, *point*)

Moves a graphical object to a point. Required Parameters:

- point - tuple of (int $*$ int)

Reimplemented from cs110graphics.GraphicalObject.

5.7.2.3 def cs110graphics.Image.resize (*self*, *width*, *height*)

Resizes the Image. Required Parameters:

- width - int
- height - int

5.7.2.4 def cs110graphics.Image.rotate (*self*, *degrees*)

Rotates an object by degrees. Required Parameters:

- degrees - int

5.7.2.5 def cs110graphics.Image.scale (*self*, *factor*)

Scales the image according to the factor. Required Parameters:

- factor - float

5.7.2.6 def cs110graphics.Image.size (*self*)

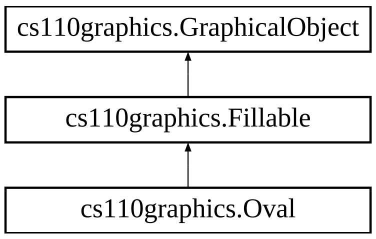
Returns a tuple of the width and height of the image.

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

- cs110graphics.py

5.8 cs110graphics.Oval Class Reference

An oval, which can be added to a Window object. Inheritance diagram for cs110graphics.Oval::



Public Member Functions

- def **__init__**
- def set_radii

  Sets the horizontal and vertical radii of the Oval.

5.8.1 Detailed Description

An oval, which can be added to a Window object. Required Parameters:

- window - Window - the window which the object will be added to.

Optional Parameters:

- radiusX - int - sets the radius of the Oval. (default: 40)
- radiusY - int - sets the radius of the Oval. (default: 60)
- center - tuple - sets the center of the Oval. (default: (200, 200))

5.8.2 Member Function Documentation

5.8.2.1 def cs110graphics.Oval.set_radii (*self*, *radiusX*, *radiusY*)

Sets the horizontal and vertical radii of the Oval. Required Parameters:

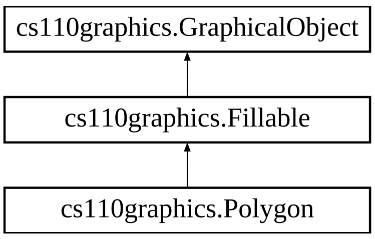
- radiusX - int
- radiusY - int

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

- cs110graphics.py

5.9 cs110graphics.Polygon Class Reference

A Polygon, which can be added to a Window object. Inheritance diagram for cs110graphics.Polygon::



Public Member Functions

- def **__init__**

5.9.1 Detailed Description

A Polygon, which can be added to a Window object. Required Parameters:

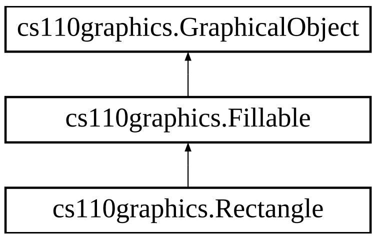
- window - Window - the window which the object will be added to.
- points - list of tuples of int ∗ int - each tuple corresponds to an xy point.

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

- cs110graphics.py

5.10 cs110graphics.Rectangle Class Reference

A rectangle, which can be added to a Window object. Inheritance diagram for cs110graphics.Rectangle::



Public Member Functions

- def **__init__**
- def set_side_lengths

    Sets the width and height of the Rectangle.

5.10.1 Detailed Description

A rectangle, which can be added to a Window object. Required Parameters:

- window - Window - the window which the object will be added to.

Optional Parameters:

- width - int - sets the width of the Square. (default: 40)
- height - int - sets the height of the Square. (default: 40)
- center - tuple - sets the center of the Square. (default: (200, 200))

5.10.2 Member Function Documentation

5.10.2.1 def cs110graphics.Rectangle.set_side_lengths (*self*, *width*, *height*)

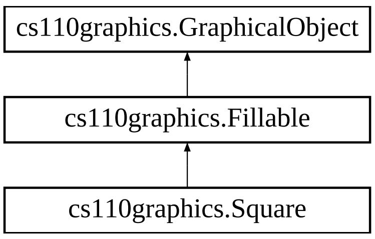
Sets the width and height of the Rectangle.

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

- cs110graphics.py

5.11 cs110graphics.Square Class Reference

A square, which can be added to a Window object. Inheritance diagram for cs110graphics.Square::



Public Member Functions

- def **__init__**
- def set_side_length

    Sets the side length of the Square.

5.11.1 Detailed Description

A square, which can be added to a Window object. Required Parameters:

- window - Window - the window which the object will be added to.

Optional Parameters:

- side_length - int - sets the side length of the Square. (default: 40)
- center - tuple - sets the center of the Square. (default: (200, 200))

5.11.2 Member Function Documentation

5.11.2.1 def cs110graphics.Square.set_side_length (*self*, *side_length*)

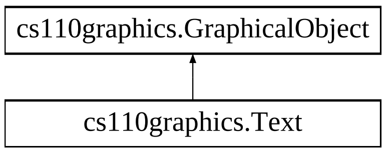
Sets the side length of the Square.

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

- cs110graphics.py

5.12 cs110graphics.Text Class Reference

Text which can be added to a Window object. Inheritance diagram for cs110graphics.Text::



Public Member Functions

- def **__init__**
- def move

  Moves a graphical object dx pixels horizontally and dy pixels vertically.
- def move_to

  Moves a graphical object to a point.
- def set_size

  Sets the point size of the text.
- def set_text

  Sets the text.

5.12.1 Detailed Description

Text which can be added to a Window object. Required Parameters:

- window - Window - the window which the object will be added to.
- text - str - The text which is displayed.

Optional Parameters:

- center - tuple of int $*$ int - sets the center of the Image. (default: (200, 200))
- width - int - sets the size of the text. (default: 12)

5.12.2 Member Function Documentation

5.12.2.1 def cs110graphics.Text.move (*self*, *dx*, *dy*)

Moves a graphical object dx pixels horizontally and dy pixels vertically. Required Parameters:

- dx - int
- dy - int

Reimplemented from cs110graphics.GraphicalObject.

5.12.2.2 def cs110graphics.Text.move_to (*self*, *point*)

Moves a graphical object to a point. Required Parameters:

- point - tuple of (int ∗ int)

Reimplemented from cs110graphics.GraphicalObject.

5.12.2.3 def cs110graphics.Text.set_size (*self*, *size*)

Sets the point size of the text. Required Parameters:

- size - int

5.12.2.4 def cs110graphics.Text.set_text (*self*, *text*)

Sets the text. Required Parameters:

- text - string

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

- cs110graphics.py

5.13 cs110graphics.Timer Class Reference

A class which continually runs a function after a delay.

Public Member Functions

- def **__init__**
- def set_function

  Sets the function which is going to be run.

- def set_interval

  Sets the interval between executions of the function.

- def start

  Starts the timer.

- def stop

  Stops the timer.

5.13.1 Detailed Description

A class which continually runs a function after a delay. Required Parameters:

- window - Window - the window which the timer will use to start and stop the animation.
- interval - int - the time (in milliseconds) that the function will refresh.
- func - function - the function which will be run.

5.13.2 Member Function Documentation

5.13.2.1 def cs110graphics.Timer.set_function (*self*, *func*)

Sets the function which is going to be run. Required Parameters:

- func - function

5.13.2.2 def cs110graphics.Timer.set_interval (*self*, *interval*)

Sets the interval between executions of the function. Required Parameters:

- interval - int

5.13.2.3 def cs110graphics.Timer.start (*self*)

Starts the timer.

5.13.2.4 def cs110graphics.Timer.stop (*self*)

Stops the timer.

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

- cs110graphics.py

5.14 cs110graphics.Window Class Reference

This window acts as a canvas which other objects can be put onto.

Public Member Functions

- def **__init__**
- def add

  Adds an object of type GraphicalObject to the Window object.
- def remove

  Removes an object of type GraphicalObject to the Window object, assuming the object being deleted exists.
- def set_background

  Sets the background color of the canvas.
- def set_height

  Sets the height of the canvas.
- def set_title

  Sets the title of the window holding the canvas.
- def set_width

  Sets the width of the canvas.

5.14.1 Detailed Description

This window acts as a canvas which other objects can be put onto. Required Parameters:

- width - int - Width of canvas.
- height - int - Height of canvas.
- background - str - Background color of canvas. Can be either the name of a color ("yellow"), or a hex code ("#FFFF00").
- name - str - The title of the window.
- first_function - proc(Window) - When the window is created, it runs this function after everything is run. (default: None)
- master - unknown type - necessary for the creation of the Tkinter widgets. (default: None)

5.14.2 Member Function Documentation

5.14.2.1 def cs110graphics.Window.add (*self*, *graphic*)

Adds an object of type GraphicalObject to the Window object. Required Parameters:

- graphic - GraphicalObject

5.14.2.2 def cs110graphics.Window.remove (*self*, *graphic*)

Removes an object of type GraphicalObject to the Window object, assuming the object being deleted exists. Required Parameters:

- graphic - GraphicalObject

5.14.2.3 def cs110graphics.Window.set_background (*self*, *background*)

Sets the background color of the canvas. Required Parameters:

- background - string

5.14.2.4 def cs110graphics.Window.set_height (*self*, *height*)

Sets the height of the canvas. Required Parameters:

- height - int

5.14.2.5 def cs110graphics.Window.set_title (*self*, *name*)

Sets the title of the window holding the canvas. Required Parameters:

- name - string

5.14.2.6 def cs110graphics.Window.set_width (*self*, *width*)

Sets the width of the canvas. Required Parameters:

- width - height

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

- cs110graphics.py

Index