PyGraphica *0.1.1*

By Luke Campbell

PyGraphica is an easy-to-learn GUI module designed for Python, built on the Python bindings (pysdl2) for SDL-2.

One important note before we start. For all coordinates and lengths, if represented as an integer will be interpreted as a measurement in pixels, and if represented as a string (e.g. '35') will be interpreted as a percentage of the height or width.

Contents

Installation

The most simple installation is via pip, i.e

`py -m pip install PyGraphica` *(for windows)*

`python -m pip install PyGraphica` *(for linux)*

However the source code can be downloaded from GitHub if pip install fails.

Once the module has been imported, add it to your Python project like this:

```
from PyGraphica import draw, colours, origins, fonts
```

Colours, Origins, and Fonts files

Colours

The colours file contains a series of default RGB tuples, which can be used to simplify coding. The colours can be accessed by the following variable names:

BLACK	DARK_GREY	GREY	LIGHT_GREY	WHITE
DARK_RED	RED	PASTEL_PINK	ORANGE	PINK
DARK_GREEN	GREEN	PASTEL_GREEN	LIGHT_GREEN	NEON_GREEN
NAVY_BLUE	DARK_BLUE	PASTEL_BLUE	BLUE	PURPLE
MUSTARD	YELLOW	LIGHT_YELLOW		
TURQUISE	LIGHT_BLUE	PALE_BLUE		
DARK_PINK	LIGHTPINK	PALE_PINK		

For example:

```
colours.PASTEL_BLUE
```

Origins

The Origins file contains the five origin options, top-left, top-right, bottom-left, bottom-right, and centre. These origins can be accessed using the following variable names:

TOP_LEFT

TOP_RIGHT

BOTTOM_LEFT

BOTTOM_RIGHT

CENTRE

For example:

```
origins.BOTTOM_LEFT
```

Fonts

The fonts file contains the path to the default fonts included in PyGraphica. These paths can be accessed using the following names:

Variable	Font
Arial	Hello world!
Baskerville	Hello world!
BrushScript	Hello world!
Calibri	Hello world!
Courier	Hello world!
Garamond	Hello world!
Helvetica	Hello world!
Impact	Hello world!
OpenDyslexic	Hello world!
TimesNewRoman	Hello world!
Trebuchet	Hello world!

For example:

```
fonts.OpenDyslexic
```

Creating a window

The *window* class can be defined by the following attributes:

attribute	default	type	description
name	‘PyGraphica’	string	name that will appear above the window
size	(800, 600)	tuple of two integers or strings of integers	dimensions (in px) of the screen
resizable	False	boolean	whether the user can resize the window by dragging
icon	False	Path (string) to file, or False for no icon	The image that will be displayed in the top left corner of the window
position	(0,20)	tuple of two integers or strings of integers	The location (px) of the window on the screen
origin	top left	variable from origins file	The location of the origin and corresponding coordinate system
colour	black	variable from colours file or RGB tuple	background colour of the window

An example window could be:

```
NavCS = draw.window("NavCS", (400,600), True, "NavCS_logo_icon.jpg", (0,20),
origins.CENTRE, colours.WHITE)
```

Once the window has been created the following attributes can be called:

attribute	description
keys	keys currently held by the user
comms	command keys currently held by the user
key_changes	keys newly pressed by the user (not held)
comm_changes	command keys newly pressed by the user (not held)
caps	whether capslock is on or shift is held
mouse_x	x position of the mouse
mouse_y	y position of the mouse
mouse_down	whether the mouse button is held down
mouse_held	whether the mouse button is held down for more than one cycle

Creating a line

The line class can be defined by the following attributes:

attribute	type	description
window	window object	window which the line will be drawn to
x1	integer or string of integer	x component of the start coordinate
y1	integer or string of integer	y component of the start coordinate
x2	integer or string of integer	x component of the end coordinate
y2	integer or string of integer	y component of the end coordinate
colour	variable from colours file or RGB tuple	colour of the line

For example

```
line1 = draw.line(app, 100, 70, 250, 300, colours.RED)
```

Once the line has been created the following attributes can be called:

name	description
visible	whether the line is displayed to the screen or not

Creating a rectangle

The rectangle class can be defined by the following attributes:

name	default	type	description
window	NA	window object	window which the line will be drawn to
x1	NA	integer or string of integer	x component of the start coordinate
x2	NA	integer or string of integer	x component of the end coordinate
y1	NA	integer or string of integer	y component of the start coordinate
y2	NA	integer or string of integer	y component of the end coordinate
colour	False	variable from colours file, RGB tuple, or False for no fill	fill colour of rectangle
border_colour	False	integer or False for no border	colour of the border of the rectangle
border_thickness	1	positive integer	thickness of the border

For example:

```
my_rect = draw.rectangle(app, 10, 20, 600, 750, colours.PURPLE,
colours.YELLOW, 2)
```

Once the rectangle has been created the following attributes can be called:

name	description
visible	whether the rectangle is displayed to the screen or not
hover	whether the mouse is hovered over the rectangle
clicked	whether theuser has selected the rectangle

For example:

```
if my_rect.clicked:
    my_rect.visible = False
```

Creating text

The text class can be defined by the following attributes:

name	default	type	description
window	NA	window object	window which the text will be drawn to
x1	NA	integer	x component of the start coordinate
y1	NA	integer	y component of the start coordinate
size	NA	positive integer or string of positive integer	height
colour	NA	variable from colours file or RGB tuple	colour of the text
content	NA	string	content of the text
font	fonts.Calibri	variable from fonts file or path to ttf/otf file	font of the text

For example:

```
title = draw.text(app, 100, -50, 20, colours.NAVY_BLUE, "Hello world!",
fonts.OpenDyslexic)
```

Once the text object has been creeated the following attributes can be called:

name	description
visible	whether the text is displayed to the screen or not
hover	whether the mouse is hovered over the text
clicked	whether the user has selected the text
x2	x component of the end coordinate
y2	y component of the end coordinate

For example:

```
if title.clicked:
    title.colour = colours.PASTEL_BLUE
elif title.hover:
    title.colour = colours.BLUE
else:
    title.colour = colours.NAVY_BLUE
```

Creating a textbox

A textbox is an input field in which users can type text. A textbox can be defined by the following attributes:

name	default	type	description
window	NA	window object	window to which the textbox will be drawn
x1	NA	integer or string of integer	x component of the start coordinate
y1	NA	integer or string of integer	y component of the start coordinate
size	NA	integer or string of integer	height of text
width	1	integer or string of integer	width of textbox (NB: textbox will expand if text goes out of textbox)
font	fonts.Calibri	variable fromfont file or path to ttf/otf file	font in which the text will be displyed
default_ text	Type here…	string	text which will be displayed in lighter shade if the user has not yet typed anything

For example:

```
code_input = draw.textbox(app,"20","10","3","80",fonts.OpenDyslexic,"ENTER CODE HERE")
```

Once the textbox has been created the following attributes can be used:

name	description
visible	whether the textbox is displayed to the screen or not
hover	whether the mouse is hovered over the textbox
clicked	whether the user has selected the textbox
x2	x component of the end coordinate
y2	y component of the end coordinate
content	the content the user has typed into the textbox

For example:

```
def submit():
    response = code_input.content
    if response == "open sesame":
        #let them to the next stage
    else:
        code_input.content = "
```

Creating an image

The image class can be defined by the following parameters:

Name	Default	Type	Description
window	NA	window object	window to which the image will be drawn
path	NA	string	path to image file
x1	NA	integer or string of integer	x component of the start coordinate
y1	NA	integer or string of integer	y component of the start coordinate
height	False	positive integer, string of positive integer or False if height is to be defined by width and aspect ratio	height of the image
width	False	positive integer, string of positive integer or False if width is to be defined by height and aspect ratio	width of the image

For example:

```
background = draw.image(app, "/my_images/background_image.jpg", 0, 0,  width
= "100")
```

Other functions

Collision function

The collision function takes two objects as input, and then returns a boolean for whether their hitboxes overlap. For example:

```
if draw.collision(rect1,image2):
    #do something
```

To_front and to_back functions

These functions can be used to reorder the elements on the screen. For example:

```
draw.to_back(background_img)
```

Or

```
draw.to_font(player)
```

Delete function

This function is used to delete an object. For example:

```
draw.delete(rect1)
```