GraphMap Help

1. GraphMap

In GraphMap, there are two ways of drawing objects. Both should be used accordingly depending on the needs of the developer. In order to do so, a windows object needs to be initialized (to draw in) along with a loop (which controls the flow of the application). Below is an example of a simple application.

```
import GraphMap as gm

# Window
window = gm.Window("GraphMap New", (500, 500), background=(80, 80, 80))

# Inplace draw
gm.Circle(window, (250, 250), radius=25, color=(200, 100, 100), inplace_draw=False)

# Object draw
circle2 = gm.Circle(window, (250, 250), radius=25, color=(200, 100, 100),
inplace_draw=False)
circle2.draw()

# Main loop
while window.running:
      window.tick()

# Termination
window.quit()
```

For more information on any part of the code above or to learn more, keep on reading below.

2. Window Object

A window object, as the name implies, is an object that takes care of all the functionality of a window. Many windows may be defined, but only one should be used at a time.

- **Window**(title, size, fullscreen, background, frame_cap)
  - **Description**: Window object constructor
  - **title**: Optional String - Defines the window's name
  - **size**: Optional Tuple - Defines the window's size
  - **fullscreen**: Optional Boolean - Defines the window's full screen status
  - **background**: Optional Tuple - Defines the window's background color – form: (R, G, B)
  - **frame_cap**: Optional Integer - Defines the window's FPS
  - **Sample use:**

```
window = gm.Window("Example", (100, 100), true, (0, 0, 0), 50)
```

- **change_icon**(path)
  - **Description**: Changes a window's icon

    - **path**: Mandatory String - Defines the icon's image path relative to main

    - **Sample use**:

```
window.change_icon("images/sample_icon.ico")
```

- **tick**()
    - **Description**: Updates the screen and events, should be used in a loop

    - **Sample use**:

```
window.tick()
```

- **clear**()
    - **Description**: Clears all items drawn on screen, should be used when inplace_drawn = True. It's an expensive operation, so use it sparingly

    - **Sample use**:

```
window.clear()
```

- **width**()
    - **Description**: Returns Integer width of the screen in pixels.

    - **Sample use**:

```
width = window.width()
```

- **height**()
    - **Description**: Returns Integer height of the screen in pixels.

    - **Sample use**:

```
height = window.height()
```

- **Getters**
    - **window**: Canvas Object – Window canvas (tkinter object)
    - **running**: Boolean – Window running status
    - **title**: String – Title of window
    - **fullscreen**: Boolean – Fullscreen status
    - **background**: Tuple – Background color
    - **last_key**: String – Last key pressed
    - **last_mouse**: Array – Last mouse button pressed (L, M, R)
    - **last_motion**: Tuple – Last saved mouse position
    - **last_update**: Float – Last time the screen was updated
    - **frame**: Integer – Frame number of window
    - **size:** Tuple – Size of window
- **Setters**
    - **background**(color): Tuple – Background color (R, G, B)

3. General Graphic Methods

Methods listed here are applicable to all graphic objects and will hence only be mentioned in this section.

- **move**(x, y)
    - **Description**: Moves the object by the desired amount across the x and y coordinates
    - **x**: Optional Integer - Defines by how much the object should move across the x axis
    - **y**: Optional Integer - Defines by how much the object should move across the y axis
    - **Sample use**:

```
object.move(x=-2, y=5)
```

- **draw**()
    - **Description**: Draws object unto a window. Will give a warning if object is already drawn
    - **Sample use**:

```
object.draw()
```

- **undraw**()
    - **Description**: Undraws and object from a window. Will give a warning if object is already undrawn
    - **Sample use**:

```
object.undraw()
```

- **Is_drawn**()
    - **Description**: Returns a True Boolean when the object is drawn and a False one when it isn’t.
    - **Sample use**:

```
if object.isdrawn(): print(“Object is drawn”)
```

- **copy**()
    - **Description**: Returns a copy of the object
    - **Sample use**:

```
copy = object.copy()
```

4. Line Object

- **Line**(window_name, point_a, point_b, color, width, dash, inplace_draw)
    - **Description**: Line object constructor
    - **window_name**: Mandatory Window Object - Denotes window to be used
    - **point_a**: Mandatory Tuple - Defines the first point of the line (x, y)
    - **point_b**: Mandatory Tuple - Defines the second point of the line (x, y)
    - **color**: Optional Tuple - Defines the color of the line (R, G, B) – Default = (0, 0, 0)
    - **width**: Optional Integer - Defines the width of the line – Default = 1
    - **dash**: Optional Tuple - Defines the dash type of the line (line_length, line_space, line_length, line_space…. <length is undefined so that any pattern can be made>) – Default = ()

- **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn't need to be stored – Default = True

- **Sample use**:

```
line = gm.Line(window, (10, 10), (100, 100), (20, 40, 20), 4, (4, 3))
```

- **length**()
    - **Description:** Returns Float length of the line
    - **Sample use**:

```
length = line.length()
```

- **Getters**
    - **window**: Canvas object – Window canvas (tkinter object)
    - **point_a**: Tuple – First point of the line (x, y)
    - **point_b**: Tuple – Second point of the line (x, y)
    - **color**: Tuple – Color of the line (R, G, B)
    - **width**: Integer – Width of the line
    - **dash**: Tuple – Dash style of the line (Refer to the constructor for an explanation)
- **Setters**
    - **point_a**(point): Tuple – First point of the line (x, y)
    - **point_b**(point): Tuple – Second point of the line (x, y)
    - **color**(color): Tuple – Color of the line (R, G, B)
    - **width**(width): Integer – Width of the line
    - **dash**(dash): Tuple – Dash style of the line (Refer to constructor for an explanation)

5. Rectangle Object

- **Rectangle**(window_name, point_a, point_b, color, outline_color, outline_width, inplace_draw)
    - **Description**: Rectangle object constructor
    - **window_name**: Mandatory Window Object - Denotes window to be used
    - **point_a**: Mandatory Tuple - Defines one of the corner points of the rectangle (x, y)
    - **point_b**: Mandatory Tuple - Defines one of the corner points of the rectangle (x, y)
    - **color**: Optional Tuple - Defines the color of the rectangle (R, G, B) – Default = (0, 0, 0)
    - **outline_color**: Optional Tuple - Defines the color of the outline of the rectangle (R, G, B) – Default (0, 0, 0)
    - **outline_width**: Optional integer - Defines the width of the outline of the rectangle – Default = 0
    - **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn't need to be stored – Default = True
    - **Sample use**:

```
rectangle = gm.Rectangle(window, (10, 10), (100, 100), (50, 50, 50),
outline_width=10)
```

- **Getters**
    - **window**: Canvas object – Window canvas (tkinter object)
    - **point_a**: Tuple – Corner point of the rectangle (x, y)
    - **point_b**: Tuple – Corner point of the rectangle (x, y)
    - **color**: Tuple – Color of the rectangle (R, G, B)
    - **outline_color**: Tuple – Color of the outline of the rectangle (R, G, B)
    - **outline_width**: Integer – Width of the outline of the rectangle
- **Setters**
    - **point_a**(point): Tuple – Corner point of the rectangle (x, y)
    - **point_b**(point): Tuple – Corner point of the rectangle (x, y)
    - **color**(color): Tuple – Color of the rectangle (R, G, B)
    - **outline_color**(color): Tuple – Color of the outline of the rectangle (R, G, B)
    - **outline_width**(width): Integer – Width of the outline of the rectangle

6. Polygon Object

- **Polygon**(window_name, points, color, outline_color, outline_width, inplace_draw)
    - **Description**: Polygon object constructor

    - **window_name**: Mandatory Window Object - Denotes window to be used
    - **points**: Mandatory Tuple - Defines the points of the polygon (x1, y1, y2, y2, … <length is undefined so that any number of points can be made>)
    - **color**: Optional Tuple - Defines the color of the polygon (R, G, B) – Default = (0, 0, 0)
    - **outline_color**: Optional Tuple - Defines the color of the outline of the polygon (R, G, B) – Default = (0, 0, 0)
    - **outline_width**: Optional Integer - Defines the width of the outline of the polygon – Default = 0
    - **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn't need to be stored – Default = True

---

    - **Sample use**:

```
polygon = gm.Polygon(window, (50, 10, 150, 10, 100, 150), color=(60, 60,
120), outline_color=(30, 30, 60), outline_width=4)
```

- **Getters**
    - **window**: Canvas object – Window canvas (tkinter object)
    - **points**: Tuple – Points of the polygon (Refer to constructor for an explanation)
    - **color**: Tuple – Color of polygon (R, G, B)
    - **outline_color**: Tuple – Color of outline of polygon (R, G, B)
    - **outline_width**: Integer – Width of polygon outline
- **Setters**
    - **color**(color): Tuple – Color of the polygon (R, G, B)
    - **outline_color**(color): Tuple – Color of the outline of the polygon (R, G, B)
    - **outline_width**(width): Integer – Width of the outline of the polygon

7. Circle Object

- **Circle**(window_name, point, radius, color, outline_color, outline_width, inplace_draw)
  - **Description**: Circle object constructor

  - **window_name**: Mandatory Window Object - Denotes window to be used
  - **point**: Mandatory Tuple - Defines the center point of the circle – form: (x, y)
  - **radius**: Mandatory Integer - Defines the radius of the circle
  - **color**: Optional Tuple - Defines the color of the circle (R, G, B) – Default = (0, 0, 0)
  - **outline_color**: Optional Tuple - Defines the color of the outline of the circle (R, G, B) – Default = (0, 0, 0)
  - **outline_width**: Optional Integer - Defines the width of the outline of the circle – Default = 0
  - **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn't need to be stored – Default = True

  ---
  - **Sample use**:

```
circle = gm.Circle(window, (250, 250), radius=25, color=(200, 100, 100),
inplace_draw=False)
```

- **Getters**
  - **window**: Canvas object – Window canvas (tkinter object)
  - **point:** Tuple – Center point of the circle (x, y)
  - **radius**: Integer – Radius of the circle
  - **color**: Tuple – Color of circle (R, G, B)
  - **outline_color**: Tuple – Color of outline of circle (R, G, B)
  - **outline_width**: Integer – Width of circle outline
- **Setters**
  - **point**(point): Tuple – Center point of the circle (x, y)
  - **color**(color): Tuple – Color of the circle (R, G, B)
  - **outline_color**(color): Tuple – Color of the outline of the circle (R, G, B)
  - **outline_width**(width): Integer – Width of the outline of the circle

8. Oval Object

- **Oval**(window_name, point_a, point_b, color, outline_color, outline_width, inplace_draw)
  - **Description**: Oval object constructor

  - **window_name**: Mandatory Window Object - Denotes window to be used
  - **point_a**: Mandatory Tuple - Defines one of the corner points of an imaginary enclosing rectangle around the oval (x, y)
  - **point_b**: Mandatory Tuple - Defines one of the corner points of an imaginary enclosing rectangle around the oval (x, y)
  - **color**: Optional Tuple - Defines the color of the oval (R, G, B) – Default = (0, 0, 0)
  - **outline_color**: Optional Tuple Defines the color of the outline of the oval – form: (R, G, B) – Default = (0, 0, 0)

- **outline_width**: Optional Integer - Defines the width of the outline of the oval – Default = 0
- **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn't need to be stored – Default = True

---

- **Sample use**:

```
oval = gm.Oval(window, (10, 10), (90, 150), (50, 50, 50), outline_width=10)
```

- **Getters**
  - **window**: Canvas object – Window canvas (tkinter object)
  - **point_a**: Tuple – Corner point of the oval (x, y)
  - **point_b**: Tuple – Corner point of the oval (x, y)
  - **color**: Tuple – Color of the oval (R, G, B)
  - **outline_color**: Tuple – Color of the outline of the oval (R, G, B)
  - **outline_width**: Integer – Width of the outline of the oval
- **Setters**
  - **point_a**(point): Tuple – Corner point of the oval (x, y)
  - **point_b**(point): Tuple – Corner point of the oval (x, y)
  - **color**(color): Tuple – Color of the oval (R, G, B)
  - **outline_color**(color): Tuple – Color of the outline of the oval (R, G, B)
  - **outline_width**(width): Integer – Width of the outline of the oval

9. Text Object

- **Text**(window_name, point, text, font, size, color, bold, italic, anchor, inplace_draw)
  - **Description**: Text object constructor

  - **window_name**: Mandatory Window Object - Denotes window to be used
  - **point**: Mandatory Tuple - Defines the point from which the text will be drawn (anchor dependent) (x, y)
  - **text**: Mandatory String – Defines the text to be displayed
  - **font**: Mandatory String – Defines the font family (System fonts supported)
  - **size**: Mandatory Integer – Defines the font size
  - **color**: Optional Tuple - Defines the color of the text (R, G, B) – Default = (0, 0, 0)
  - **bold**: Optional Boolean – Defines text bold status – Default = False
  - **italic**: Optional Boolean – Defines text italic status – Default = False
  - **anchor**: Optional String – Defines where the text will be drawn from ("nw" = northwest <upper left of text will be placed on the previously defined point>, "center" = center <center of text will be placed on the previously defined point>) – Default = "nw"
  - **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn't need to be stored – Default = True

---

  - **Sample use**:

```
text = gm.Text(window, (10, 10), “Sample”, “Consolas”, 15)
```

- **Getters**
    - **window**: Canvas object – Window canvas (tkinter object)
    - **point**: Tuple – Point from which the text will be drawn (anchor dependent) (x, y)
    - **text**: String – Text displayed
    - **font**: String – Font family
    - **size**: Integer – Font size
    - **color**: Tuple – Font color (R, G, B)
    - **bold**: Boolean – Text bold status
    - **italic**: Boolean – Text italic status
    - **anchor**: String – Text draw position (Refer to constructor for an explanation)
- **Setters**
    - **point**(point): Tuple – Point from which the text will be drawn (anchor dependent) (x, y)
    - **text**(text): String – Text displayed
    - **font**(font): String – Font family (System fonts supported)
    - **size**(size): Integer – Font size
    - **color**(color): Tuple – Font color (R, G, B)
    - **bold**(bold): Boolean – Text bold status
    - **italic**(italic): Boolean – Text italic status
    - **anchor**(anchor): String - Text draw position (Refer to constructor for an explanation)

10. Image Object

An image object loads an image and displays it when drawn. Supported image formats include PNG, GIF, JPEG, PPM, TIFF and BMP.

- **Image**(window_name, point, path, anchor_type, inplace_draw)
    - **Description**: Image object constructor
    - **window_name**: Mandatory Window Object - Denotes window to be used
    - **point**: Mandatory Tuple - Defines the point from which the image will be drawn (anchor_type dependent) (x, y)
    - **path**: Mandatory String – Defines the path of the image relative to main
    - **anchor_type**: Optional String – Defines where the image will be drawn from ("nw" = northwest <upper left of image will be placed on the previously defined point>, "center" = center <center of image will be placed on the previously defined point>) – Default = "nw"
    - **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn't need to be stored – Default = True
    - **Sample use**:

```
image = gm.Image(window, (10, 10), “images/sample_image.png”)
```

- **width**()
    - **Description**: Returns Integer width of the image in pixels.
    - **Sample use**:

```
width = object.width()
```

- **height**()
  - **Description**: Returns Integer height of the image in pixels.
  - **Sample use**:

```
height = object.height()
```

- **Getters**
  - **window**: Canvas object – Window canvas (tkinter object)
  - **path**: String – Path of the image relative to main
  - **image**: String – Path of the image relative to main
  - **point**: Tuple – Point from which the image will be drawn (anchor_type dependent)
  - **anchor_type**: String – Image draw position (Refer to constructor for an explanation)
- **Setters**
  - **path**(path): String – Path of the image relative to main
  - **image**(path): String – Path of the image relative to main
  - **point**(point): Tuple – Point from which the text will be drawn (anchor dependent) (x, y)
  - **anchor_type**(anchor): String - Image draw position (Refer to constructor for an explanation)

11. Button Object

- **Button**(window_name, point_a, point_b, box_color, outline_color, hovered_box_color, hovered_outline_color, clicked_box_color, clicked_outline_color, outline_width, text, font, font_size, font_color, hovered_font_color, clicked_font_color, bold, italic, image_path, hovered_image_path, clicked_image_path, inplace_draw)
  - **Description**: Button object constructor
  - **window_name**: Mandatory Window Object - Denotes window to be used
  - **point_a**: Mandatory Tuple - Defines one of the corner points of the button (x, y)
  - **point_b**: Mandatory Tuple - Defines one of the corner points of the button (x, y)
  - **box_color**: Optional Tuple – Defines the color of the button (R, G, B) – Default = (0, 0, 0)
  - **outline_color**: Optional Tuple – Defines the color of the outline of the button (R, G, B) – Default = (0, 0, 0)
  - **hovered_box_color**: Optional Tuple – Defines the color of the button when hovered (R, G, B) – Default = (40, 40, 40)
  - **hovered_outline_color**: Optional Tuple – Defines the color of the outline of the button when hovered (R, G, B) – Default = (40, 40, 40)
  - **clicked_box_color**: Optional Tuple - Defines the color of the button when clicked (R, G, B) – Default = (80, 80, 80)
  - **clicked_outline_color**: Optional Tuple - Defines the color of the outline of the button when clicked (R, G, B) – Default = (80, 80, 80)
  - **outline_width**: Optional Integer – Defines the width of the outline of the button – Default = 0
  - **text**: Optional String – Defines the text inside the button – Default = “”

- **font**: Optional String – Defines the font of the text inside the button (System fonts supported) – Default = “Times”
- **font_size**: Optional Integer – Defines the font size of the text inside the button – Default = 20
- **font_color**: Optional Tuple – Defines the color of the text inside the button (R, G, B) – Default = (255, 255, 255)
- **hovered_font_color**: Optional Tuple - Defines the color of the text inside the button when hovered (R, G, B) – Default = (210, 210, 210)
- **clicked_font_color**: Optional Tuple - Defines the color of the text inside the button when clicked (R, G, B) – Default = (170, 170, 170)
- **bold**: Optional Boolean – Defines the bold status of the text inside the button – Default = False
- **italic**: Optional Boolean – Defines the italic status of the text inside the button – Default = False
- **image_path**: Optional String – Defines the path of the image inside of the button relative to main – Default = “”
- **hovered_image_path**: Optional String – Defines the path of the image inside of the button relative to main when hovered – Default = “”
- **clicked_image_path**: Optional String – Defines the path of the image inside of the button relative to main when clicked – Default = “”
- **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn’t need to be stored – Default = True

---

- **Sample use**:

```
button = gm.Button(window, (40, 250), (140, 350), text="BUTTON",
font="Consolas")
```

- **Getters**
  - **point_a**: Tuple – Corner point of the button (x, y)
  - **point_b**: Tuple – Corner point of the button (x, y)
- **Setters**
  - **point_a**(point): Tuple – Corner point of the button (x, y)
  - **point_b**(point): Tuple – Corner point of the button (x, y)

12. InputBox Object

- **InputBox**(window_name, point, length, height, text, bar_color, outline_color, outline_width, text_y_margin, text_x_margin, font, font_size, font_color, bold, italic, pointer_width, pointer_length, pointer_x_margin, pointer_y_margin, pointer_blink_frames, pointer_color, force_case, inplace_draw)
  - **Description**: InputBox object constructor

  - **window_name**: Mandatory Window Object - Denotes window to be used
  - **point**: Mandatory Tuple - Defines where the box will be drawn (x, y)
  - **length**: Optional Integer – Defines the length of the box – Default = 200

- **height**: Optional Integer – Defines the height of the box – Default = 30
- **text**: Optional String – Defines the default text on the box – Default = “SEARCH”
- **bar_color**: Optional Tuple – Defines the color of the box (R, G, B) – Default = (200, 200, 200)
- **outline_color**: Optional Tuple – Defines the color of the outline of the box (R, G, B) – Default = (0, 0, 0)
- **outline_width**: Optional Integer – Defines the width of the outline of the box – Default = 0
- **text_y_margin**: Optional Integer – Defines the margin the text has from the top of the box in pixels – Default = -1
- **text_x_margin**: Optional Integer – Defines the margin the text has from the left of the box in pixels – Default = 6
- **font**: Optional String – Defines the font of the text in the box (System fonts supported) – Default = “Consolas”
- **font_size**: Optional Integer – Defines the font size of the text in the box – Default = 20
- **font_color:** Optional Tuple – Defines the color of the text in the box – Default = (20, 20, 20)
- **bold**: Optional Boolean – Defines the bold status of the text in the box – Default = False
- **italic**: Optional Boolean – Defines the italic status of the text in the box – Default = False
- **pointer_width**: Optional Integer – Defines the width of the pointer – Default = 2
- **pointer_length**: Optional Integer – Defines the length of the pointer – Default = 10
- **pointer_x_margin**: Optional Integer – Defines the margin the pointer has from the right of the text – Default = 4
- **pointer_y_margin**: Optional Integer – Defines the margin the pointer has from the top of the box – Default = 4
- **pointer_blink_frames**: Optional Integer – Defines the number of frames it takes the pointer to blink – Default = 400
- **pointer_color**: Optional Tuple – Defines the color of the pointer (R, G, B) – Default = (0, 0, 0)
- **force_case**: Optional String - Defines the case of the text <”NONE” = standard, “UPPER” = force uppercase> - Default = “NONE”
- **inplace_draw**: Optional Boolean - Defines inplace draw status. If true, object will be drawn without the need of calling object.draw(), this also means that the object doesn’t need to be stored – Default = True

---

- **Sample use**:

```
input = gm.InputBox(window, (10, 10), 1000)
```

- **Getters**
  - **point**: Tuple – Point where the box will be drawn (x, y)
- **Setters**
  - **point**(point): Tuple – Point where the box will be drawn (x, y)