Lesson 10

Introduction to Graphical User Interfaces II

In this lesson we are going to produce more GUIs using the Tkinter library. The main goal is to be able to draw shapes using Tkinter functions.

Tkinter has various functions especially for drawing shapes.

- **create_line()** - draws a line
- **create_rectangle()** - draws a rectangle or a square
- **create_oval()** - draws a circle or an oval

Creating a Line

In order to draw a line, we use the function in the following way:

create_line(x0, y0, x1, y1, option…)

Creating a Rectangle

In order to draw a rectangle or square, we use the function in the following way:

create_rectangle(x0, y0, x1, y1, option…)

Creating an Oval

In order to draw an oval or circle, we use the function in the following way:

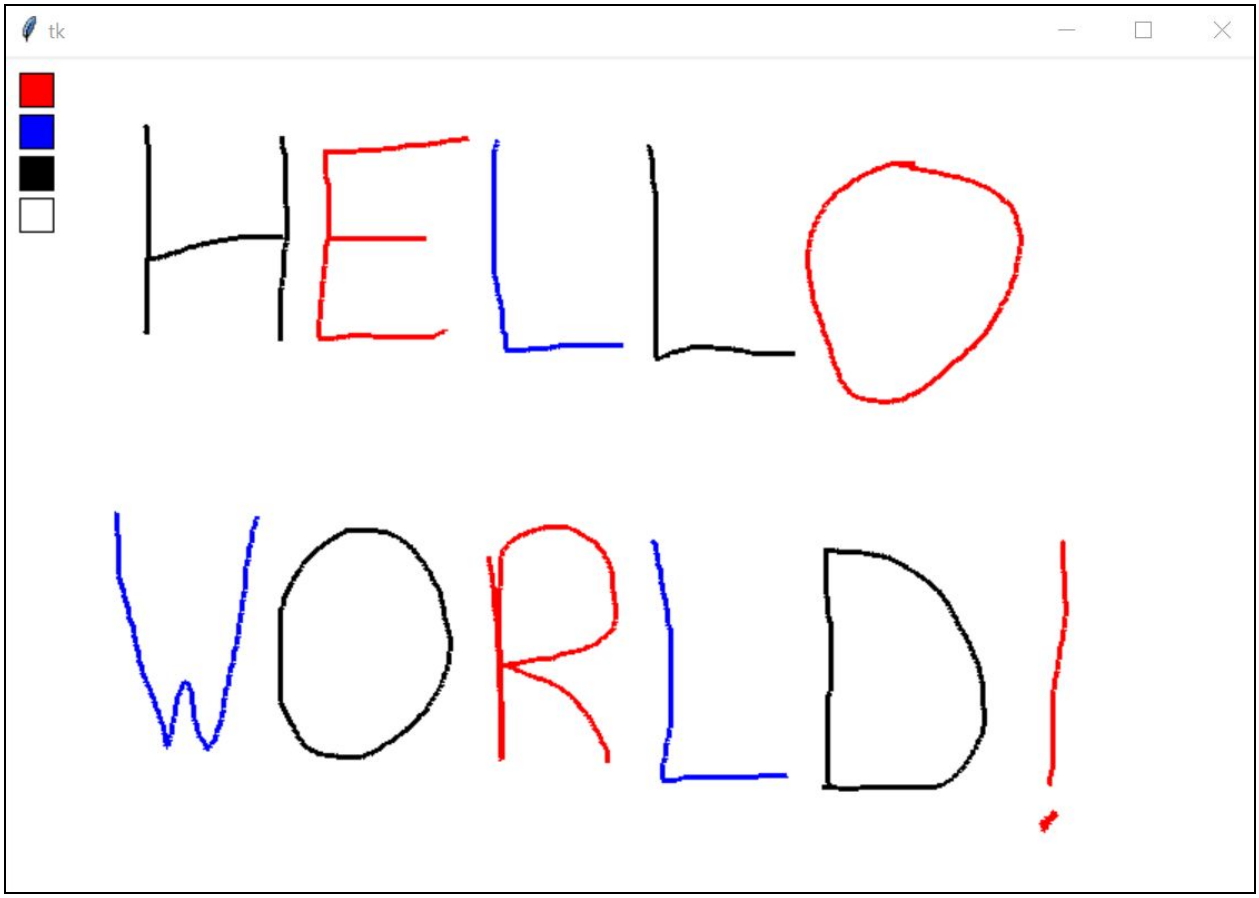
create_oval(x0, y0, x1, y1, option…)

Tasks:

1. Draw a stickman using the functions above.
2. Fill the stickman's head with a colour (this is one of the options in the function, look this up on http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html).

Make a Masterpiece Project

Write a program that allows you to make your own ‘paintings’ by dragging your mouse-pointer around the screen, just like in Microsoft Paint.



Tasks:

1. Import the tkinter module.

2. Add a couple of ***print()*** functions to explain how to use the program.

```
print("To draw, hold down the left mouse button and move your mouse around.")
print("To change your brush colour, click on one of the colours.")
```

3. Tell the computer to create a canvas when the program runs.

```
window = tkinter.Tk()
canvas = tkinter.Canvas(window, width=750, height=500, bg="white")
canvas.pack()
```

4. Create two **global** variables to represent the **x** and **y coordinates** of the mouse pointer (you can create two at the same time, separated by commas). To start with, both variables should be set to 0. Set a starting colour for your paint, too.

```
colour = "black"
lastX, lastY = 0, 0
```

5. Create a function to keep track of the mouse-pointer, so you can draw lines as it moves. The function in the box below is called ***storePosition()*** but you may call this something different if you like.

```
def storePosition(event):
    global lastX, lastY

    lastX = event.x
    lastY = event.y
```

6. Create a function to tell the computer what to do when you click on the canvas. Store the position of the click by calling the ***storePosition()*** function.

```
def onClick(event):
    storePosition(event)
```

7. Add another function to draw a line when the mouse-pointer is dragged across the canvas.

```
def onDrag(event):
    global colour
    global lastX, lastY

    canvas.create_line(lastX, lastY, event.x, event.y, fill=colour, width=3)
    storePosition(event)
```

8. Connect your ***onClick()*** and ***onDrag()*** functions to clicks and drags on the canvas using ***bind()***.

```
canvas.bind("<Button-1>", onClick)
canvas.bind("<B1-Motion>", onDrag)
```

9. Call the main loop to repeatedly show the window.

```
window.mainloop()
```

10. To paint in colours, add a palette of clickable squares. To create and position these on the canvas, you need to use coordinates...

```
redID = canvas.create_rectangle(10,10,30,30,fill="red")
blueID = canvas.create_rectangle(10,35,30,55,fill="blue")
blackID = canvas.create_rectangle(10,60,30,80,fill="black")
whiteID = canvas.create_rectangle(10,85,30,105,fill="white")
```

11. To change the paint colour, you will need a separate function for each colour on your palette.

```
def setColourRed(event):
    global colour
    colour = "red"
```

The code above will set the colour to red. You will need to come up with the functions to set the colour to blue, black and white.

12. You can use a tkinter function called **tag_bind()** to link the act of clicking each square in the palette to your **setColour** function.

```
canvas.tag_bind(redID, "<Button-1>", setColourRed)
canvas.tag_bind(blueID, "<Button-1>", setColourBlue)
canvas.tag_bind(blackID, "<Button-1>", setColourBlack)
canvas.tag_bind(whiteID, "<Button-1>", setColourWhite)
```

Extra Tasks:

1. Add three more colours to your palette.
2. Add buttons that will increase and decrease the size of your brush.

Hint: If we want to add in some text to our buttons, for example a plus and a minus symbol, there is no easy way of adding these to our rectangle buttons. Therefore, what we can do instead is add in just the text and treat the text as a button.

```
increaseID = canvas.create_text((20, 130), text="+")
decreaseID = canvas.create_text((20, 155), text="-")
```