GamesDrawingTkinter

February 20, 2020

1 Games: Drawing graphics with Tkinter

We will use Tkinter to discusse the basic building blocks of games.

The following example illustrates the use of the library Tkinter for drawing arbitrary graphics on a separate window.

It draws a ball attached to a stick.

The basics of drawing can be summarized as follows:

1. Create a `canvas`, i.e., a "piece of the/a window" (technically, a *widget*) that can be addressed by pixels. That is, we can say `highlight this and that pixel with this and that color`.
2. Actually `draw` something on the canvas
3. Have the canvas display all that (`update the canvas`)

So, a bit more in general, we that we need
1. **A window**. It's either given, or we need to instantiate one (most common) 2. **A widget** (it's the technical word for all objects we see in graphical user interfaces -GUIs-, wether clickable or not) bound to that window that *has pixels*. Such type of widget is called a ***canvas***.

A general reference and documentation on Tkinter for Python can be found at https://docs.python.org/2/library/tkinter.html

[2]:
```
from tkinter import *
import math

def _create_circle(self, x, y, r, **kwargs):
    return self.create_oval(x-r, y-r, x+r, y+r, **kwargs)

Canvas.create_circle = _create_circle

#The Tkinter window
window = Tk()
window.title("Ball and stick")

#window width and height in pixels
winsz = [600, 600]

#The canvas that everything shows up on
widget = Canvas(window,
```

```
                width=winsz[0],
                height=winsz[1],
                bg="#ffffff"
                )
widget.pack()

widget.create_circle(winsz[0]//2, 0, 2,
                     fill="#000000",
                     tags="pin"
                     )

def draw(canvas, pos):
    canvas.delete("pend")
    canvas.create_line(winsz[0]//2, 0, *pos, tags="pend")
    canvas.create_circle(*pos, 10, fill="#000000", tags="pend")
    canvas.update()

#while True:
draw(widget, [winsz[0]//2, winsz[1]//2])
```

Here another example. This creates random rectangles on the canvas.

[ ]:
```
tk = Tk()
canvas = Canvas(tk, width=400, height=400)
canvas.pack()

def rndm_rect(width, height):
    x1 = (random.randrange(width))
    y1 = (random.randrange(height))
    x2 = x1 + (random.randrange(width))
    y2 = y1 + (random.randrange(width))
    canvas.create_rectangle(x1, y1, x2, y2)

rndm_rect(400, 400)


for x in range(0, 500):
    rndm_rect(400, 400)

tk.mainloop()
```

2 Assignment 2 Term 3

This assignment is intended as an introduction to simulations. This will be the topic of this term's project. See the videos on (http://dragon.vellgraphy.com/G12/CS/Material/) for more details.

2.0.1 Due date: Tuesday, Feb. 25 2020

1. Wrap those two examples into classes `BouncingBall` and `RandomRect` such that the following function displays either as expected

```
def draw(something):
    if something == 'ball':
        bs = BouncingBall()
        bs.show()
    elif something == 'rectangles':
        rr = RandomRect()
        rr.show()
    else:
        print('I have no clue how to draw ',something)
```

2. Create the classes `BouncingBall` and `RandomRect` by appropriately wrapping the previous two examples on `Tkinter`
3. Modify the class `RandomRect` so that one can specify the size of the window when instantiating an object
4. Modify the class `RandomRect` so that initially there is a delay of 1 second between the first rectangle drawn and the second. Then at each iteration, it draws them faster and faster until it does so as fast as in the previous questions. Hint: Find out about the function `time.sleep`
5. Modify the class `Bouncing` so that it moves a ball with a random constant speed till it reaches border of the canvas:
    1. Includes a method `move` that moves each time step. This function should return the new coordinates of the ball
    2. After each move, check if the ball's coordinates lie outside of the window. If so, stop the program by calling `sys.exit()` -after importing `sys`
    3. If the ball lies still inside the canvas, repeat step 1.

[ ]: