CS 120 (Spring 22): Introduction to Computer Programming II

Long Project #3 - Animated Graphics

due at 5pm, Tue 8 Feb 2022

1 Animated Graphics

In this project, you will either (roughly) replicate one of the example animations that I've provided, or else come up with something of comparable complexity.

Feel free to be creative! Your TA will be grading you on your engagement with the project - that means that you have to write a little bit of non-trivial code. And of course, if your code doesn't work - if it can't even draw to the screen, or it isn't animated - then you will lose most or all of the points for this part of the project.

Of course, you'll need to know how to use graphics - don't worry! I've put lots of information down at the bottom of this spec - and I've also put up a video that will give you a good intro to graphics here: `https://youtu.be/FxarUVRzD40`

(Note: The video mentions recursion, and recommends it for one of the demo programs. We haven't covered recursion yet, this semester, so it's certainly not required, but if you know it, feel free to use it.)

1.1 How to Turn in the Graphics Program

Your graphics program must be a complete program (not just a function). You may name the file whatever you like. When the TA runs your program, you should create a graphics window and draw the animation.

If you want to also include a README file (either a text file or PDF), explaining how to run the program (or any cool features), you may - but you aren't required to.

2 Appendix: `graphics.py`

Ben Dicken has created a file, `graphics.py`, which many of you have used in the past: it gives a slightly-nicer wrapper around a classic Python library, known as `tkinter`. Ben's library encapsulates `tkinter` in a nice object, and provides some methods that you can use to draw a scene.

If you've used `graphics.py` before, please **download a new copy from me.** I've added a few features.

2.1 (x, y) Coordinates

As is common in computer graphics, the (0,0) point of the screen is the **upper** left corner; positive x is going right, and positive y is going down.

(spec contines of the next page)

2.2 EXAMPLE

Here's an example of an extremely trivial drawing. Note that this one does not contain any animation - you must write your program so that it draws a different picture on every "frame."

```
import graphics

def main():
    win = graphics.graphics(400,400, "Example")

    while not win.is_destroyed():
        win.clear()                                # wipe all previous drawings
        win.ellipse  (200,200,  50,100, "red")
        win.rectangle(200,200, 100, 50, "green")
        win.update_frame(20)                       # sleep for 50 milliseconds

if __name__ == "__main__":
    main()
```

2.3 Creating a Window

To create a window, `import graphics`, and then call the following function:

```
win = graphics.graphics(width, height, window_title)
```

This is a constructor for an object; save the reference into a variable, because all of the calls below are **methods** on this object.

NOTE: Please restrict your window to no larger than 800x800, so that we can be sure that the window will fit on your TA's screen.

2.4 `win.is_destroyed()`

Returns `True` if the window has been closed. Use this in your `while` loop to know when to terminate your program.

2.5 `win.ellipse(x,y, w,h, fill)`

Used to draw circles and ellipses. (x,y) is the center of the figure. (w,h) is the width and height of the figure; to draw a circle, give the same value for both width and height. Since this is the **total** width and height, it is the **diameter** of the circle, not the radius.

The fill color is a string; it can be a standard color, such as `"black", "red", "green",` etc. , or it can be an html hex color, like `"#ff00ff"`.

2.6 `win.rectangle(x,y, w,h, fill)`

Used to draw rectangles and squares. (x,y) is the upper-left corner (**NOT** the center). (w,h) is the width and height.

The fill color works the same way as `ellipse()`.

2.7 `win.line(x1,y1, x2,y2, fill, width=3)`

Draws a line between two points. The fill color works the same as `ellipse()`.

The `width` is an optional parmeters. If you don't provide it, then it will default to 3. If you provide it, then you can change the width of the line you are drawing.

2.8 `win.triangle(x1,y1, x2,y2, x3,y3, fill)`

Draws a filled triangle, connecting the three points. The fill color works the same way as `ellipse()`.

2.9 `win.clear()`

Clears the window. Destroys all drawings that you've made so far. (This should be the **first** thing you do before drawing another frame of the animation!)

2.10 `win.update_frame(speed)`

Makes the window visible to the user for a while. (If you don't call this, the user will probably see an empty window, and may get a warning about an "unresponsive" program.)

This function will pause your program for $\frac{1}{speed}$ seconds; after that, it will return do you.