There was no Assignment 4F; this assignment is required for all students.

PIC 16, Winter 2018 – Assignment 5M

Assigned 2/5/2018. Code (a single `.py` file) due by the end of class 2/9/2018 on CCLE. Hand in a printout of this document with the self-assessment portion completed by the end of class on 2/9/2018.

In this assignment, you will create a simple animation of a ball bouncing around inside a window.

Task

Begin by following an example from the preparation assignment to create a window 600px wide by 400px tall with a white background (not the default gray) and a 30px diameter red circle at the top left corner of the window. Hint: I don't know of a method for setting a background color, but we all know how to paint a rectangle…. Likewise, I don't know of a method for painting a circle, but I could see if the `QPainter` documentation has a method for painting an ellipse.... Use what you know, and experiment to get things (colors, for instance) as desired.

If you didn't already, create instance variables for the diameter, x coordinate, and y coordinate of the "ball" in the initializer, and use those fields throughout your class instead of hardcoding the diameter and position. Create two additional fields to represent the x and y components of the ball's *velocity* and set them to 1 (i.e. we will make the ball move 1 pixel each time we update the scene).

Write a method `animate` that: increments the x and y position of the ball by the corresponding "velocity" component and calls the `update()` method inherited from `QWidget`, which requests that `paintEvent` be called. You should not call `paintEvent` yourself.

If you were to run your code now, the ball would stay put because the `animate` method is never called. Create a `QTimer` that calls your `animate` method every 30 ms or so. Remember, you have to keep a reference to 'timer', otherwise the timer can be garbage-collected. It is up to you to decide where to put create the `QTimer` timer and how to "keep a reference".

If you were successful, running your code now should show the ball moving in your window. It will not bounce, however, when it reaches the edge. Instead, it will move beyond the edge. It's still there! You just can't see it unless you resize your window. In your `animate` method (you decide exactly where), call a method `checkCollision` that determines whether a collision between the edge of the ball and the edge of the screen has occurred, and if so, change the ball's velocity accordingly. The method should take less than 10 lines, but you may find writing the conditions a bit tricky. If so, debugging this code is good practice! Use the debugger and/or print output to the console to figure out what's going wrong. Write your method so that it will work even if the user resizes the window. Hint: check the widget's `geometry`.

When your code is working, feel free to tweak the velocity and `QTimer` period to see how these parameters affect the apparent speed and smoothness of the animation.

Self-Assessment

Does your code draw a red ball starting in the very top left corner? 20 points.

Does your code animate the ball moving across the screen? 40 points.

Does the ball bounce around the window? 20 points.

Does the ball bounce *exactly* when the perimeter touches the boundary? 10 points.

Does the ball bounce exactly when it touches the boundary even when the window is resized? 10 points.

Circle the points you deserve above and indicate your total score here: