A Bouncing Ball

I. Today we will do a graphics simulation of a bouncing ball (no gravity, rebounding off the ceiling.)

Here is the idea. We will make a ball, starting at the bottom of the window. It will move up one unit each unit of time until it gets to the top of the window. At that point, it should move down at the same rate, until it gets to the bottom. Then, it should move up again, ...

In order to do this, we will create a ball class. Eventually, this will be the algorithm for the program - don't do it yet:

Set up the window
Make a ball
Start things going - put the ball in the window
repeat forever
    move the ball

A. Let's start by thinking about the layout. Let the window be twenty by twenty, but with the origin in the middle. (You decide how many pixels.)

B. Now let's design the ball. Basically, it's going to be a moving circle. To make life easy, give it a radius of 1. We have to keep track of where it is (where its center is) and what direction it is going. Let's give it a color, too.

So, we want the instance data to be as follows:
  circle - a Circle
  direction - the direction the circle is going. Let's use the convention that up is 1 and down is -1, so direction is an integer.

define a class BBall (for bouncy ball)
Write the constructor
  __init__(self, color, center, dir=1)
Notice that the parameters don't exactly match the instance data we discussed: a Circle and a direction. But we can use these parameters to initialize the instance data.
  • Set circle to a Circle of radius 1, whose center is center.
  • Fill the circle with color.
  • Set the direction to dir. (The dir=1 in the header means, if the user doesn't give enough arguments, so dir wouldn't get a value, make dir = 1. That is, if the user doesn't specify otherwise, the ball will start out going up.)

C. So that we can see if things are working, give the BBall class a draw method
draw(self, win)
that draws the Circle (self.circle) to the window win.
D. Write a main to test that everything is working. main is not part of the class BBall.
   - main should create the window.
   - Then it can create the BBall and tell the BBall to draw itself.

What value should c (the argument for the center) be if we want the ball to be sitting at the bottom of the window, in the horizontal middle?

```python
c = Point( , )
```

main should construct a BBall object: `bouncyBall=BBall("red", c)`
Then bouncyBall should call its draw method. Try it.
You will probably have some debugging to do!

II. A. Let's add a method to the BBall class, to move the ball. We can make use of the move function for the Circle class in graphics.py, `move(dx, dy)`
If we want to move only up or down, not sideways, what should we use for dx?

The value of dy is 1 or -1, depending on whether the BBall should be moving up or down. That depends on self.direction, except

if the ball was moving up and got to the top already
```
What is self.circle.getCenter().getY() in that case?
```

or if the ball was moving down and got to the bottom already
```
What is self.circle.getCenter().getY() in that case?
```

(Since `self.circle` is a Circle, we can use the getCenter() method of the Circle class. getCenter() returns a Point, so that Point can invoke the `getY()` method of the Point class.)

You are now ready to write the BBall class method
```
move(self, win)
```
It will need to make use of the instance variables
```
self.direction and self.circle
```

There are two situations where the value of self.direction needs to change, but most of the time it will stay the same. This is a good place to use `if` `elif` without an `else`.

B. Try it out in main. Tell bouncyBall to move.

C. To do more testing of move, make another BBall, `downBall`, at the top of the window, with x=5, and color "blue". Do you see it? Tell it to move. Did it move down a little?

D. Make 2 more BBalls, `midUp` and `midDown`, both with y=0, with x=-3 and with x=-6, of different colors. One should have direction=1 and the other should have direction=-1. Have them each invoke their move method. Do they move in the right directions?
III. Time to get the ball bouncing.
   A. For now, we will only try to get bounceBall moving. Create an infinite loop
      while True:
      Inside of it have bounceBall move.

   Try it. bounceBall should be bouncing!
   (You will probably have to do some debugging.)

   B. Let's have a party! Inside the loop have all the balls move.
   Try it.

   C. Stopping the program is kind of a pain. It would be nice if we could press the q key
      (for quit) and the program stopped.
      We have seen a method (for the GraphWin class) getKey() that waits until the user enters
      a key, then continues once a key is pressed. We don't really want the balls to stop after
      each small move, waiting to see if we want to quit. Just to understand what happens,
      change the loop condition to
      while win.getKey()!='q'

      After the loop body (outside of the loop) add
      win.close()

      This should close the window if the program ever gets out of the loop.
      Run the program.
      What happens if you don't press a key?

      What happens each time you press a key other than q?

      What happens when you press q?

   D. There is another method in the GraphWin class, checkKey(), which checks to see if a
      key has been pressed since the last time it was called. If a key was pressed, it returns the
      value of that key. If no key was pressed, it returns the value None.

      Change the getKey to checkKey. Run the program.

      What happens if you don't press a key?

      What happens each time you press a key other than q?

      What happens when you press q?
IV. It would be nice if bouncyBall's color changed every time we clicked the mouse. Since we are just clicking the mouse, there is no way to specify which color to change to, so let's make it random.

A. So far, we have no way to change the color of a BBall once we have created it. Write a method (in BBall class) changeColor(self)

In it, create a list of colors, like

```
colors=['red','orange','yellow','green','blue','indigo','violet']
```

but you can pick your own.

Then have your program pick a random number, to correspond to an index of a color in your list. (Remember to add `from random import randrange` at the top of your program.)

In changeColor set self.circle's color to the randomly chosen color.

To test changeColor, have bouncyBall change its color each pass through the loop.

B. As with getKey and checkKey, there is a method in the GraphWin class, checkMouse that returns the Point where the mouse was clicked, if it was clicked, but returns None if the mouse has not been clicked since the last time the mouse was checked.

In your code, where you had bouncyBall change its color in part A, insert a call to checkMouse. (Recall this method belongs to win). Have bouncyBall only change its color if the returned value of checkMouse is not None.

Check that your program is working.

Print the file.

V. Submitting your work.
• Make sure your name is on this lab packet.
• Staple your final program from IV B, to the Lab 18 packet.
• Turn the lab in at the file cabinet.