Name: Abdullah Shahzad
Course: CS 152
Section: B
Date: 11/2/2021

Project 7 Report:

Title: *Modelling Collisions between Boxes and Balls in 2 Dimensions*

Abstract:

In this project, I’m making balls collide with boxes, either one of them could be moving, and the boxes disappear on impact, similar to **Atari.** The program uses physics equations to model the movement of objects and then checks the coordinates of the objects to determine whether or not they are colliding. This type of code could be, like I mentioned earlier, used in games, or to simulate collisions of cars, stars, molecules or any other object/group of objects.
This program makes use of Object Oriented Programming, which is a programming approach that treats each item in the code as an object, with attributes and functions that it performs called methods. In my code in specific, an object would be the ball, its attributes/fields would be its size, mass, velocity, and color, and its functions would be to move, accelerate, draw itself onto your screen, or undraw itself.

Results:

Required Element 1:

My collision function is in collide.py. The function creates a box and a ball, the box isn’t moving and the ball is set to collide with the box. The function loops endlessly to check the status of the objects, this means that the function refreshes to see where the objects have moved compared to their original position and keeps track of their locations. The function then uses an if statement (a statement that checks if something is true or not) to see if the distance between the objects is larger than the sum of the radius of the circle and half of the height or the width of the box (depending on if the collision happens from above/below compared to the sides). If the distance between the two objects is less than equal to that value, the square redraws itself, basically indicating that a collision happened and that the box was terminated as a result.

(Video of this attached as V1 coderun)

Required Element 2:

I added blocks at the bottom of the fall.py program and made a video.

(Video of this attached as V2 coderun)

Required Element 3:

I set up a way for the user to interact with the program. At the start, the terminal asks the user to pick a direction using the WASD keys. When the user does, the program starts by throwing the ball in that direction

(Video of this attached as V3 coderun)

Reflection:

Coding concepts from this program can be used in a wide variety of fields. Like I mentioned in my abstract, it can be used to map the collisions between molecules, or to make video games, or to visualize sports like cricket or soccer, where balls collide with players or objects and then perform an action.

Lecture concepts utilized in this project include Object Oriented Programming, which I also mentioned in my abstract and our introduction to Zelle Graphics as a graphics package, which we can now use to make simple shapes and have them interact with other things

Follow-up Questions

1. What is the purpose of having get and set functions for a class? Why not just access the object fields directly?

Having get/set functions firstly makes the process shorter, instead of having to write the code for the entire function, we can just call a function that's already written,, this is especially useful with larger set functions like the setPosition functions in this program. It also makes it easier to avoid returning important encapsulated fields that are being used by the object, instead we can return copies of that field.

2. Given a list of Zelle Graphics objects, write a for loop that would move each object in the list by (5, 10) in x and y.

```
for item in list:
    item.move(5, 10)
```

3. The Zelle GraphWin class has functions getMouse() and checkMouse(). What is the difference between them?

getMouse waits for the user to click to return a value, the checkMouse function constantly returns "None" until the user clicks and gives it an input.

4. What is the difference between simulation space and visualization space?

It is a difference in coordinates, normally, we'd work with euclidean coordinates but in a visualization space, we work with Zelle graphics coordinates, where the y-axis is inverted and so, to put an object in space at a particular y-value, we must subtract that y-value from the height of the window.

Extension (optional):

1) **Colored circles generated in one of the lab activities:**

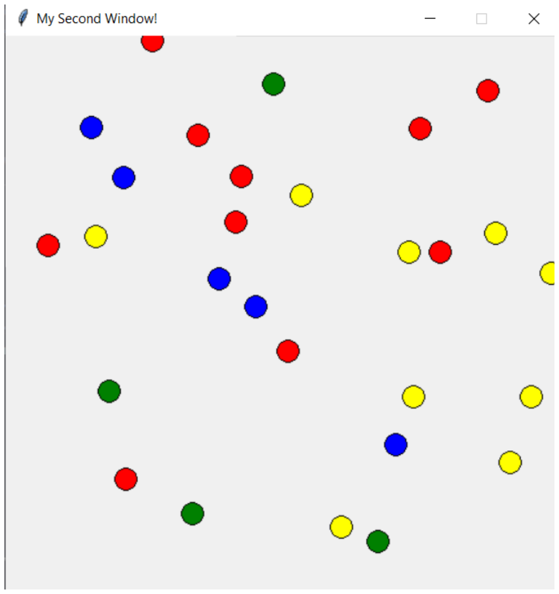

Fig 1.1

In figure 1.1, you can see that each ball generated is a random color. To do this, in the file that I defined the class object, I made a list of the strings of colors called ‘colors’. I then added the self.color attribute to the init function and set it equal to the result of calling the setFill() builtin function for the circle with the argument colors[index] and used the random.randint() function to generate a random index to randomly choose a color.

2) **Colored circles and squares every time you refresh the project program:**

This phenomena is clear in the videos attached to this submission. I coded the colors of the blocks and the circles in a similar way to how I coded the first extension, it was just a little bit more work.

3) **Balls bouncing off squares in the project program**

This phenomenon is clear in the attached video, **V4 coderun**, the circle bounces off of the squares on collision, consuming this as well. To do this, I used the same collision code as before but this time, after the conditional, I wrote code to make the ball bounce again to a large y height and a random side of the x-axis using the random.randint() function.

Acknowledgements: Nkechi Badi, Professor Allen Harper , graphicsPlus, time, random