Name: **Abdullah Shahzad**
Course: **CS 152**
Section: **B**
Date: **11/3/2021**

Project 8 Report:

Title: **Modelling 2-Dimensional Collisions in Pinball**

Abstract:

For this project, I made a pinball game. The game involves a moving ball that interacts with stationary obstacles in a course. The ball bounces off obstacles and the speed and direction of the ball are modelled using physics equations of motion and momentum. When you run the code, the program will open up a window for you that has the obstacles and the ball drawn on it. The program will then give the ball a random initial velocity and acceleration and start the game.

This week's project makes use of inheritance and dictionaries. Inheritance is a coding technique in object oriented programming (a programming approach that treats each item in the code as an object, with attributes and functions that it performs called methods). Inheritance is when you realize that many of the objects that you are coding come under a more generalized class because of them having similar attributes and methods. For example, if you're coding a cat and a dog, both could come under the more generalized pet class because they have similar attributes like having weight, a coat color, or a certain breed; they even have similar methods like sleeping, eating, or moving.

Dictionaries, on the other hand, are a way of storing data in python. A dictionary stores ordered pairs of a key and a value. A key is supposed to be unique and immutable, which means that in a particular dictionary there can only be one key with a particular value and it can't be changed. The value stored with that key is just called value. In this project I used dictionaries to call the different collision functions depending on the types of objects that were colliding. For example, the 'ball, block' key would have the value that calls the collision function between a ball and a block.

Results:

Required Output 1: The link to a video of my obstacle course in action is attached along with this submission as **'Vid_1'**

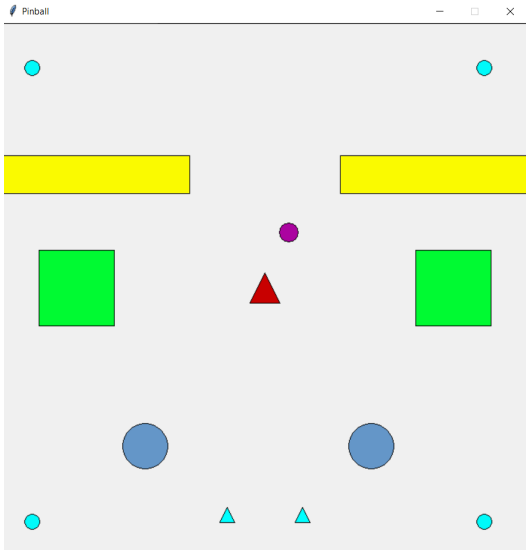

Fig 1.1

For the obstacle course, I coded in circles, squares, and triangles. The triangles are modelled as circles for the purpose of collisions with their center in the middle of the triangle and the radius being the distance from the center to any of the vertices. Besides the obstacles, there are walls on every side of the window as well that the user can't see to keep the ball from falling out of sight. All the obstacles are a different color and the ball is a different color as well. I coded it so that the ball would be a different color every time the program was run by using random.randint(0, 255) in place of all the rgb value entries of the set_color function.

Required Output 2:

The link to a video of my snowman bouncing around is attached along with this submission as **'Vid_2'**

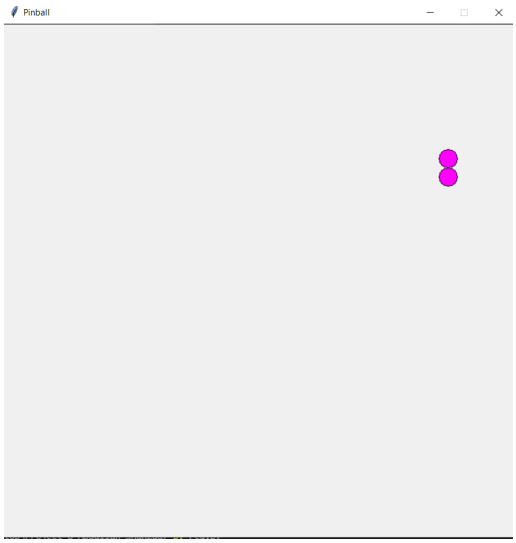

Fig 1.2

I used the get_position method to position two circles together, making a snowman. The entire thing bounces around as 1 object modelled as a circle according to project instructions.

Reflection:

Using the concept of inheritance in my code made it significantly more concise than it would have otherwise been. Instead of having to code the same attributes and methods again and again, I used a

parent class to define all the common attributes and methods. Using dictionaries also made this project more achievable by being a very convenient way to call certain collision functions in this project, as I also mentioned in the abstract. All I had to do was call for the types of the objects involved in the collision and using that as a key for the dictionary instantly called the relevant function.
Real life applications of the things I learned in this project could be mapping collisions between other objects such as cars, molecules, or stars. We can also make more games or simulate sports like soccer or cricket where something happens when a ball collides with another object or player.

(in a brief paragraph) draws connections between lecture concepts utilized in this project and real-world problems that interest you**.** *How else could these concepts apply to our everyday lives? Also addresses all of the follow up questions included at the end of the project (below).*

Follow-up Questions

Q1. What is inheritance?

Inheritance is when we identify that the different classes we'll be coding can come under a more general category. For example, cats, dogs, and rabbits come under the broad category of pets. After identifying this, we can see that they share a lot of characteristics (attributes) and do a lot of similar things (methods). Inheritance allows us to have a parent class (in this case that would be the Pet class) and inherit all of its attributes and methods in a child class (in this case the dog, cat, or the rabbit). Through this we can avoid a great amount of repetition we would otherwise have in our code.

Q2. What does it mean for a child class to override a method?

This means that the child class can have the same method as a parent class but have it perform a different function because the method in the child class overrides the method inherited from the parent class.

Q3. What is a class variable or class global variable?

A class variable or a class global variable is a variable that can be shared by all object instances of a class. Class variables are declared when making the class and aren't inside any of the methods because that would mean that they only exist if the method is called.

Q4. What is a field of an object?

A field of an object is a characteristic (also called an attribute) that an object has. For example, a field of a Ball would be its radius or its weight or its color. We list down the fields of an object when coding the __init__ method of the class.

Extension (optional):

1) I made another game. This game is about a car that starts at the bottom of the screen and faces a series of other cars and cones on its way to the top. The user needs to move out of the screen through the top to finish the game. On finishing the game, a prompt shows up saying 'You Won!' and gives the user instructions on how to quit the game. The other cars are a different color from the user's car and are moving. The cones are orange and aren't moving. However, the cones spawn in at a different location each time the program is run. A video of this game is included with this submission as **"Vid_3"**

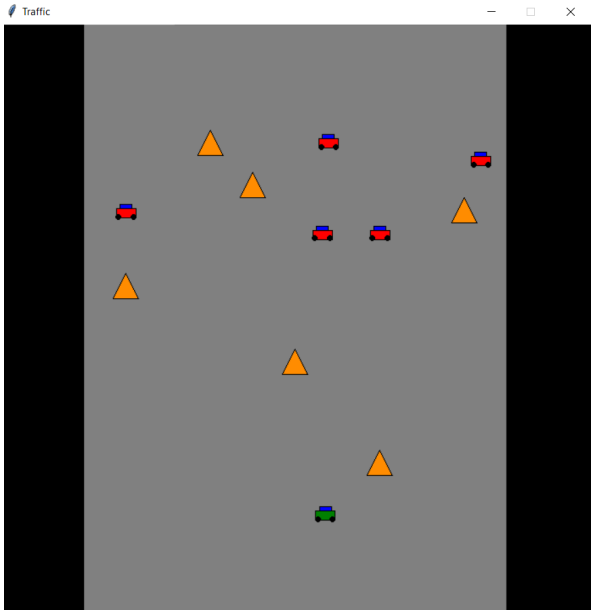

Fig 1.3

Fig 1.4

I made a separate group of cars and put them in a list and used the same principle as the collision of a ball and an object to code collisions between two cars. The red cars don't collide with each other but the

red cars do collide with the cones and the green car. The green car, as you can see in Fig 1.4 is the car that the user is controlling. The user needs to use the controls to avoid the cars and the cones which otherwise throw the car closer to the bottom of the screen. I used the random.randint() function to change the positions of the cones each time the program is run and removed the wall at the top of the screen. I also introduced two black rectangles at the sides to imitate a sidewalk. I coded the car by using the get_position function and positioning objects in the self.vis list around the center accordingly—2 circles and 2 rectangles. I also wrote a separate set color function for the Car object because I needed each object in the self.vis list to be a different color. This function also allowed me to distinguish between the user's car and the other cars. Just now, I added code to make the enemy cars different random colors that aren't green which you can see in Fig 1.5.

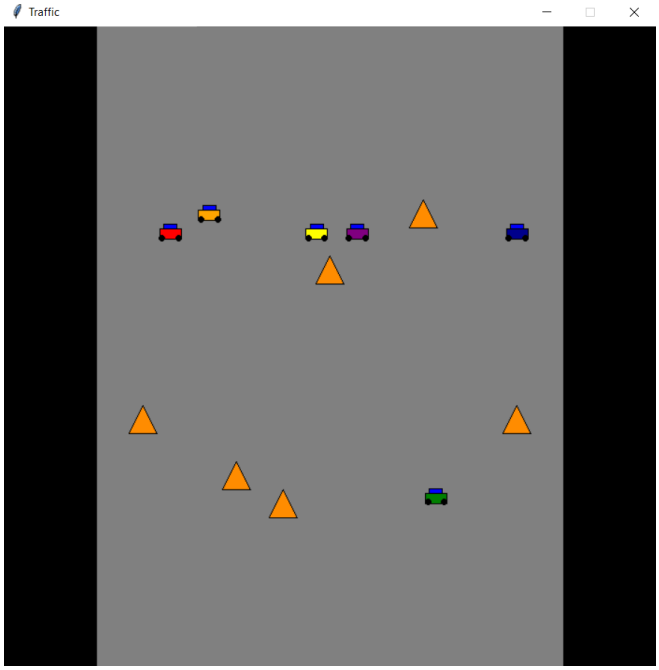

Fig 1,5

For the mechanics of this game, I reused some of the code from previous projects. I used the code that checks if the ball fell out of the screen to check when the car made it out of the top of the screen and I used the Text operator of the graphicsPlus library to have the program print 'You Won' onto the user's screen at the end of the game. Lastly, I set up the user's controls. The user can press w to make the car go down but in the right or left direction (the direction is random) to navigate their way around these obstacles, on going down, the car is also given a high upward acceleration so it comes back up and goes on until the user presses 'w' again.

2) I added the ability to whack the ball in pinball. When the user presses 'w', the ball jumps up. The ball could go either left or right, I coded this by using the random.randint() function and giving it a range from -5 to +5.

Acknowledgements: Professor Allen Harper, Libraries Imported: random, graphicsPlus, collision