**CS 1410: Flappy Bird**

In computer science, classes and object-oriented programming are very commonly used when creating programs that involve computer graphics and graphical user interfaces. To display something visually on the screen, like a button, a class is used to represent a Button, and multiple instances of the Button class can be created to display multiple buttons on the screen at one time. This is the foundation on which all modern applications are created, whether it’s on your computer, phone, or television.

The Flappy Bird app was a game on iOS and Android platforms in late 2013 and early 2014. It has an interesting history. The basic idea is a bird trying to fly between oncoming obstacles, while gravity pulls the bird down. When the user taps the screen, the bird flaps its wings and it flies upward. Game score is counted by the number of obstacles the bird successfully passes. You can watch and listen as a couple of game reviewers play.

**Assignment**

Your assignment is to create a program using Python and Pygame that mimics the ideas of Flappy Bird.

At a minimum, your program should include:

- A player object that accelerates up when the user presses a designated key, and accelerates down every game frame.
- Pairs of obstacles with gaps between them that scroll from the side of the window toward the player. The position of the gaps should be chosen randomly to avoid repetitive game play. The player scores a point for every pair of obstacles that it passes.
- A score display to show the current score.
- A collision detection mechanism to stop the game when the player collides with an obstacle.
- A collision detection mechanism to stop the game when the player collides with the top or bottom of the screen.

You must use classes and objects to construct this program. There must be a top level class that represents the game, and classes and objects created for all of the elements in the game.

Start by downloading the provided Pygame starter kit by clicking here.

**Extra Challenges**

- Add a ground, a sky, or a background object that changes its display to give the illusion of forward progress.
- Draw the elements of the game with multiple shapes to give them better screen presence.
- Draw a part of the player display to help indicate the direction of travel.
- Add a restart option to the game so the player doesn’t have to exit the application and start it again to restart game play.
- Add a high score as well as the current score. This requires that the restart option is available.
- Add a high score file so that the game can remember the best score ever, not just the best score since the program was launched.
- Add sound.
- Add images for display.

**Hints**

- If you keep the player roughly circular and the obstacles roughly rectangular, the collision detection can be done using relatively simple geometry. No need to make the collision detection pixel-perfect.
- Before starting, grab a piece of paper and create a rough sketch of your game. Use this to help you visualize what game elements you need, and how to draw them. Use this to plan the classes that you will need to
create, and to approximate the coordinates that you will need when drawing the individual shapes.

- Refer to the class example when planning your classes and how they will relate to each other, and also to see examples of how to move elements from frame to frame.

- Refer to the Pygame documentation to understand which parameters are necessary when calling each of the Pygame draw methods. Specifically, you should be interested in `pygame.draw` and `pygame.Rect`.

- When creating colors, use a helpful tool to determine the RGB values. Here are two good options: color.adobe.com and colorpicker.com

- Instructions for installing PyGame.

**Sample**

An example running program: