COSC 1436 - Assignment 1

## 1. Brief Introdunction to Pygame Zero:

We are going to use Pygame Zero, a 2D game engine in our Python program. Game engines are used by programmers to make the tasking of building their program (game) easier. For this assignment we need five main things from Pygame Zero:

* Window Control with WIDTH, HEIGHT, TITLE.
* The Rect function.
* The filled\_rect function.
* The draw( ) function.
* The update( ) function, although we will not modify this function in this assignment.

When we open the Python file, you may see the three sentences on the top like;

**TITLE = “Assignment 1”**

**WIDTH = 1000**

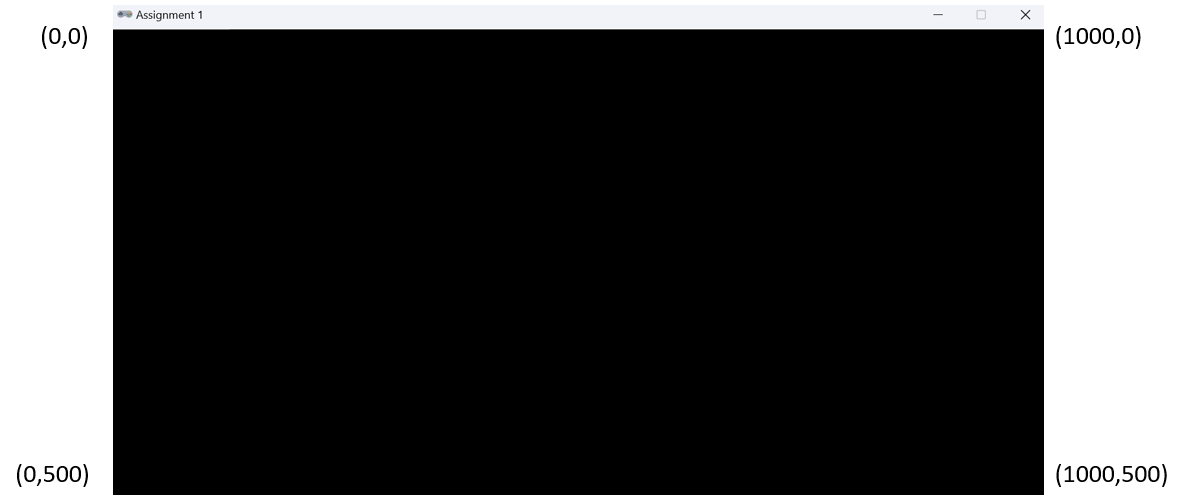
**HEIGHT = 500**

With the TITLE, you can control the title of the window:



With the WIDTH and HEIGHT, you can control the length of width and height of window. In this case, each number represents the number of the pixels. For instance, the width is composed 1000 pixels and the height is composed 500 pixels for this window.

When it comes to window composition, the x value increases by going to the right side, and the y value increases by going to the downward. (x, y)



In Pygame Zero, we can declare a rectangle by using Rect function by following instruction:

**Rect((x\_position, y\_position), (rect\_width, rect\_height))**

The Rect function is included in Pygame Rect class which is available as a built-in, and it helps to declare various rectangles.

To draw a rectangle on the screen we need to use the screen.draw.filled\_rect() function. This needs a Rect() argument and the color.

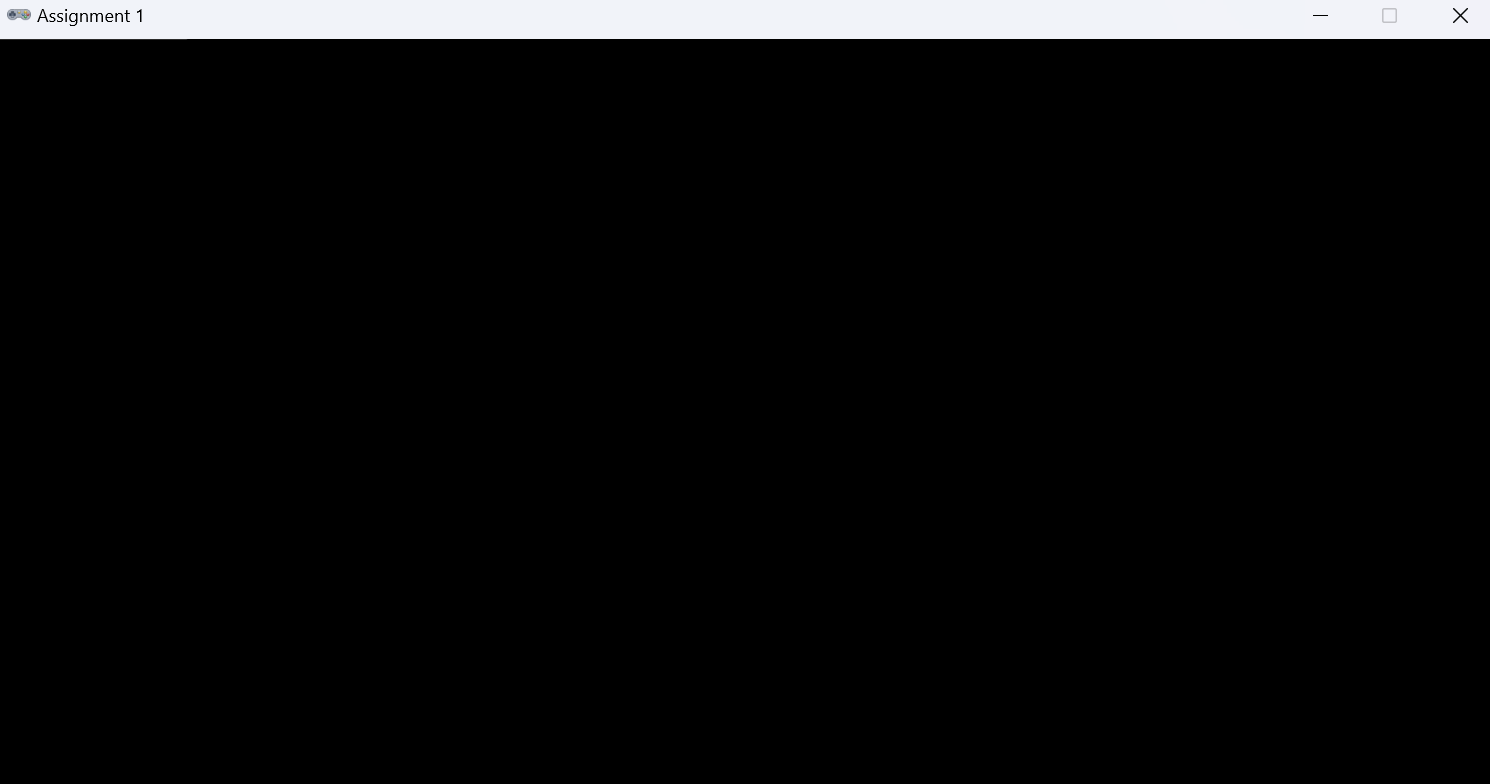
An example of using the draw rect function is:

**screen.draw.filled\_rect(Rect((x\_position, y\_position), (rect\_width, rect\_height)), “color”)**

Pygame Zero engine automatically looks for the function named ***draw***, and then calls (executes) it 60 times per second. This function draws all the game contents on the screen. Just like the ***draw*** function, the ***update*** function is automatically called (executed) by Pygame Zero 60 times per second by Pygame Zero engine.

## 2. Opening the Assignment with Mu

* 1. Download the files of Assignment 1 from the webpage of this course. Decompress the zip file. If you are using Mac OS, double click Assignment1.zip to unzip the file. If you are using Windows, right click Assignment1.zip and choose "extract all" to unzip the file.
  2. Go to the folder called *Assignment1* that you have just unzipped. Then, you will see the code file “assignment1.py”.
  3. Using **Mu**, click on the “Load” button on the menu bar. Browse to the location where you saved “assignment1.py”, and open the file.
  4. Make sure that Mu’s Mode is set to Pygame Zero mode.
  5. To run the program, you need to click on the “Play” button. Initially, you will get the following screen:



* 1. After you click on the “Play” button, the button will change to a “Stop” button. To terminate the game, you will need to click the “Stop” button.

## 3. Assignment Description:

In this assignment we are going to start implementing a 2D fighting game. We are going to program some features of Mortal Kombat through multiple assignments. In assignment 1, we are going to code an essential feature of any fighting game, the health bar. A health bar is a visual representation of how much health a fighter (game character) has. Let us see several examples from popular fighting games:



All fighting games have the health of the player represented as a bar form. For example, in the original Mortal Kombat game, a full health is represented as a green bar. As the player loses health, the green color is replaced with a red color. In the example below, Scorpion has full health while Sub-Zero lost half of his health. A player with no health is represented as a full red bar.



In our first assignment will learn how to create a health bar similar to the one in the original Mortal Kombat. So how are we going to implement a health bar?

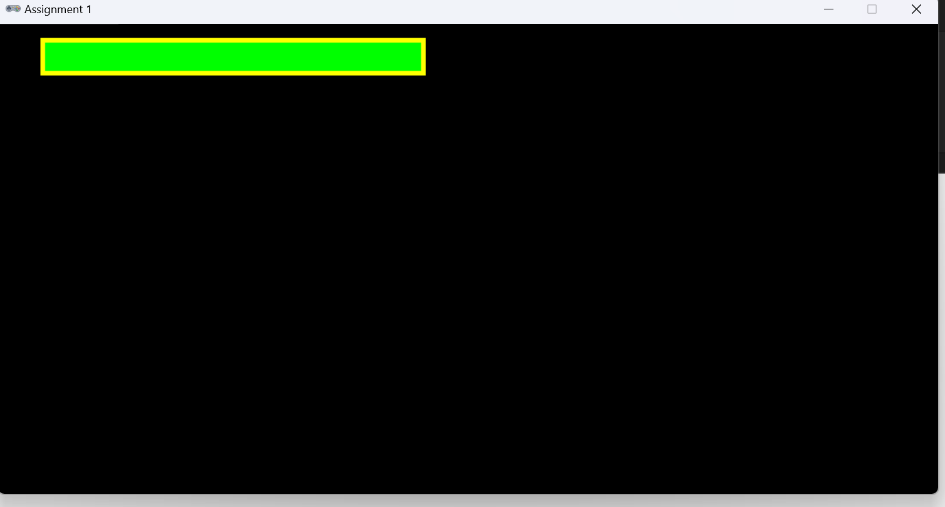
We need an integer variable to represent the health of a player as a numeric value. We also want two rectangles, a red and a green one, over each other. Which rectangle you think should be on top?

Let us continue with our example. As Scorpion loses health, the green bar should shrink from right to left while the red bar should stay the same:



## Assignment Questions:

1. Create a new integer variable called *maxHealth*. The value of this variable should be 100.
2. Create a new integer variable called *player1Health*. The initial value of this variable should be *maxHealth*.
3. Inside the draw function, draw three rectangles on the screen: a red rectangle to represent the lack of health, a green rectangle to represent health, and a yellow border rectangle. The length of the red, and green rectangle should be 400 pixels. Here is how the screen should look like:



We want the *player1Health* variable to control the green health bar. For example, if we modify *player1Health* to 70, then the health bar and the screen should be modified on its own to:

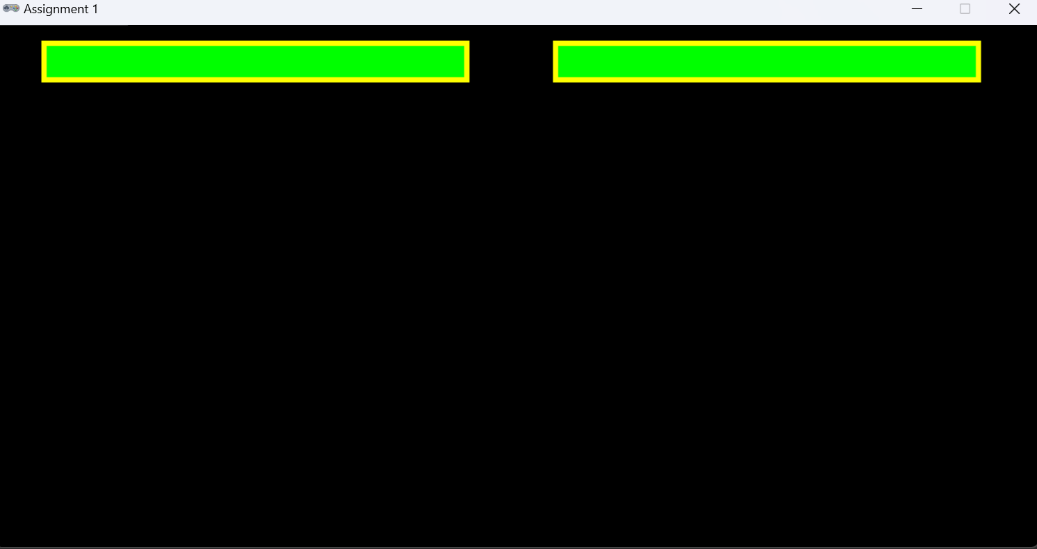


On the other hand, if we set the value of *player1Health* to 40, then the health bar will now be the following:



Notice, as the player’s health drops the green bar shrinks in length starting from the right side to the left side. **Hint**: you need to use the *player1Health* variable while you draw the green triangle.

1. Create a new integer variable called *player2Health*. The initial value of this variable should be *maxHealth*.
2. Inside the draw function, draw another three rectangles on the screen to represent the health bar of player 2. Here is how the screen should look like:



Notice, that in case of player 2 when health decrease say to 50 the green bar shrinks from left side to the right side.



## What to hand in:

Submit your project electronically through D2L by attaching and submitting your Python program file (assignment1.py).