**Pygame** **Project** **–** **Snake!**

The format for the project will be as follows­

* You will pick a partner and work together. Make sure that you email files to each other so you each have the code you work on!
* You will have time to work on this project in class. There will be a schedule setup so that you may ask for my help on certain days for a set amount of time during classes. Otherwise, you should attend tutorials.
* There will be two checkpoint dates before the end of the semester. You and your partner will need to have these checkpoints completed to earn full credit for the project.

# Game Play

The goal of the project is to implement the basic game play described below.

The game starts with a randomly placed apple and a 3 segment length snake that will move in a random direction. The board needs to be 40 by 30 blocks (this equates to 800 pixels by 600 pixels). The top left corner square of the board has coordinates (0, 0) and the bottom right corner square has coordinates (40, 30) (the y-axis is still flipped). As the snake eats apples it will increase in length and the score will increase by 1 point for each apple. If it moves off the board or runs into itself then the game play will end.

# Basic Rules

1. The snake may not move off the board or move into a space already occupied by itself. It should not be able to turn back on itself.
2. When the head of the snake hits an apple it ‘eats’ the apple. A new apple will then appear somewhere randomly on the board.
3. As the snake eats more apples it grows. The score for the game is updated for each apple that is eaten.
4. If the snake breaks the first rule, the game ends and a ‘Game Over!’ message is displayed with a prompt to play again.

# User Interaction

The user can use the arrow keys to direct the snake. They should also be able to display a grid on the board using a toggle key.

# Project Design

As discussed in class, you have a starter file that has all the class and method definitions. You will have to implement the methods to make your game work. We will do this step by step – starting small and extending the game features as we go along. You’ll work with your partner to try your best to implement the methods; I will periodically email out code to the whole class to give you hints on how to work with the trickier parts.

Most of the methods that you need to implement have just one statement, **pass**, that tells Python that the method doesn’t do anything currently. All the places where you will need to add code have a comment ‘**YOUR** **CODE** **HERE**’. At the end of the project, you should have code in all the places where you find this comment.

**READ** **ALL** **THE** **INSTRUCTIONS** **IN** **A** **GIVEN** **SECTION** **BEFORE** **YOU** **START** **WRITING** **ANY** **CODE.** **MAKE** **SURE** **THAT** **YOUR** **CODE** **WORKS** **BEFORE** **MOVING** **ON** **TO** **THE** **NEXT** **SECTION.**

**1.** **Snake** **Classes** **Overview**

Get a copy of the file snake template.py from the course webpage.

Take a look at the file. There are several classes included in this file. Read the documentation at the top to get an idea of how everything works together. The Window class implements the functionality of the game board and will be used to display anything that happens in the game.

Read through the rest of the file and familiarize yourself with the different classes and their attributes and methods. Take a look at the Apple and Snake classes as well since they also have some additional attributes and methods. Feel free to add to the colors; you can choose to make game elements any color you want!

In previous classwork you used **pygame.display** to create a window where you can draw objects. For our Snake project, we will create a window surface where the game will take place.

 Implement the \_\_init\_\_ method for the Window class. Run your code and make sure the window appears with a black background. You should have all of the attributes as listed under the class declaration.

# 2. Displaying intro and game\_over screens

Let’s make things a bit more interesting. We want to display more than just a black screen. If you look at the display\_intro method you can see that it is using a helper method called display\_text. You should study the display\_text and create\_text helper functions at the bottom of the class. Make sure you understand how these two functions work – you will need it to implement the display\_game\_over method and display\_score methods.

Alright, now that you have done a bit of studying, let’s add to the update method. We need to check what the state of the game is and then display the pertinent screen. For example, if the game is in the ‘intro’ state, then the update method should run the display\_intro method. Go ahead and add code to the update method so that it will be able to display the intro screen.

 Now that you have this code implemented go ahead and test the update method by uncommenting the Window test case. You should see the intro screen display.

Now let’s implement the display\_game\_over method. You should create the following screen using the display\_text function. 

 Once you have this implemented, try running the test 1 again. The screen should flip between the intro and game\_over screen repeatedly.

Now we need to figure out how we will display the game objects during the ‘game\_play’ state. You should ask yourself at this point… what do we need to draw on the screen?

Well, this is fairly simple to figure out – we will need an apple, a snake, and the score. We will also need a background color (you can change this from black if you like). The problem is that we haven’t yet fully implemented the Apple and Snake classes or the display\_score method. We will need to tackle each of these first before we move on.

The overall look of the game will be similar to what’s shown below.



The score needs to be located at (10, 10) pixels and should display the score as shown in the picture. The apple will just be a single block sized square with a body color and shadow outline. You will need to implement the \_\_init\_\_ and draw methods for this class. You may want to look through the Snake class before you work on the Apple class to give you some ideas. For the Snake class you should notice the only methods you need to implement are the move and eat methods. This means you should be able to draw the snake with your current code.

Try drawing the snake in the update method from the Window class. Use the test case 2 and note as the screens flip between the intro and game\_over screens the snake should just stay on the window. Test this before you move on.

Now let’s fill in the Apple class. Note that the location you use will be based on the grid locations (NOT PIXEL LOCATION). Once you have the apple class filled in test your draw method using test case 3. Once again, the snake and apple should just stay on the window as it flips between the intro and game\_over screens.

 **CHECKOFF** **1** **(Due** **TUESDAY 2/21):** **Show me sometime during the day that your code works up to this point.**