CS 5110/6110 Program 1 (20 points)

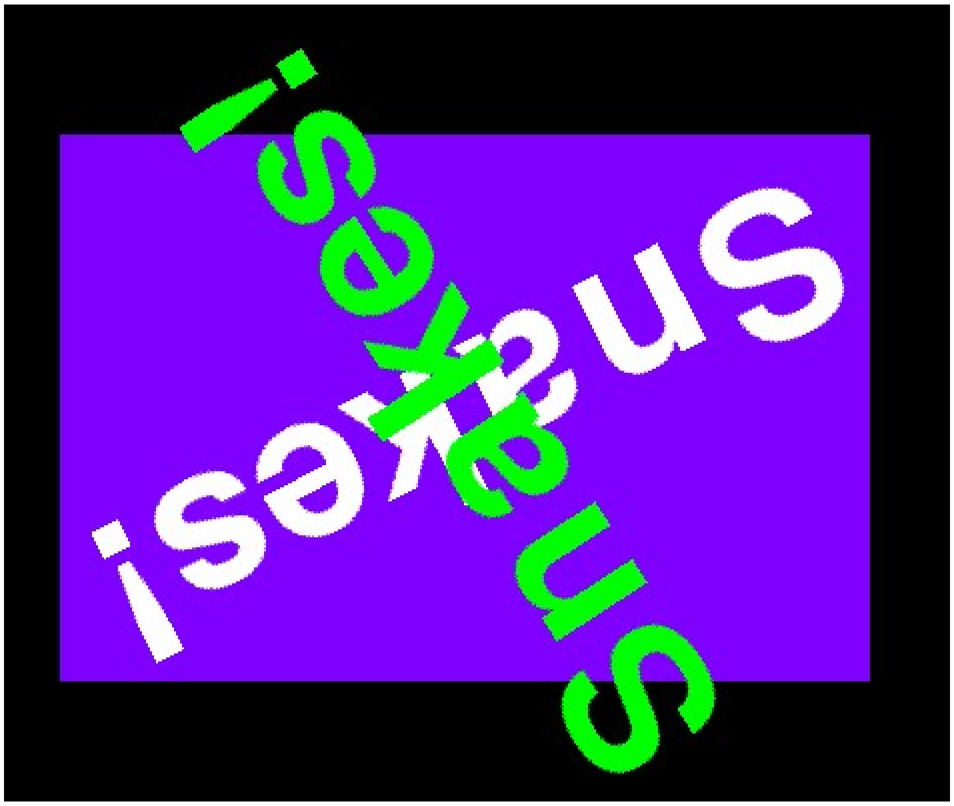
Project

Consult the material in the course module named Prelimaries. Download python, pycharm, and install pygame (see video). Several in the class are tutors, so feel free to go to the tutor room for help installing.

Using wormy.py as a starting point, make the changes listed below. Since the program is written for one apple and one worm, you will need to plan your data structures carefully. Test your results after each step.

a) Change the name of the game (which displays at the beginning) to be something other than “wormy”

b) Change the colors that display at the beginning. For example:



c) Use a “larger” grid.

d) Have multiple apples on the grid. New apples will be generated as they are eaten.

e) Allow the user(s) to control two snakes using the arrow keys (for one snake) and four other keys (for the other snake)

f) The keypad arrow keys will be used to move both worms in the same direction at the same time. The keypad arrow key names are K_KP2, K_KP4, K_KP6, and K_KP8.

g) Show the scores for both snakes. You can use whatever scoring system you like.

h) The snake dies when it hits a wall, the other snake, or itself.

i) Implement some variation of this idea: Allow the snakes to shoot each other. If the snake is hit, it cuts him in pieces (based on where the ray hits him). The tail is left behind as a stone. If a stone is hit by either snake, the snake dies.

Create a short video demo of your project. Submit the video with your project. In the video, explain the original feature you implemented. I used obs studio https://www.wikihow.com/Create-a-Screencast. The instructions were perfect!

Create a readMe.txt file telling the user what keys to use for motion

Submit the entire project as one zip file.

Hints:

1. For testing, slow the movement down by changing the frames per second (FPS) to 5 or less.
2. If you aren't careful in your design, this will be a mess. Create separate parameterized modules to do various things.
3. Use lots of lists. They are easy to pull apart by the inclusion of for loops.
4. Don't hesitate to print to the console while using pygame. That was very helpful when I was debugging
5. Parameter passing is a little tough to get used to (as it is neither by reference or by value). I often returned the values I needed (so there was no concern with how parameters were passed).
   For example, I made building the worms a function which returned the list of worms and the directions they were to go initally.

```
def getWormCoords(ct):
    directions = [RIGHT,LEFT]
    multiplier = [-1,1]
    wormCoords = []
    for i in range(ct):
        #build worms
    return wormCoords,directions
```

6. You can step through two parallel arrays in sync by first zipping the lists together. See the following example in which there are several worms that are colored differently.

```
def drawWorms(wormCoords,colors):
    for aworm,color in zip(wormCoords,colors):
        for coord in aworm:
            x = coord['x'] * CELLSIZE
            y = coord['y'] * CELLSIZE
            wormSegmentRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
            pygame.draw.rect(DISPLAYSURF, color[0], wormSegmentRect)
            wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, CELLSIZE - 8, CELLSIZE - 8)
            pygame.draw.rect(DISPLAYSURF, color[1], wormInnerSegmentRect)
```

For example, if there are two worms, wormCoords may look like:

[[{'x': 20, 'y': 26}, {'x': 20, 'y': 25}, {'x': 20, 'y': 24}], [{'x': 27, 'y': 33}, {'x': 27, 'y': 32}, {'x': 27, 'y': 31}]]

Colors represents a pair of colors for each worm and may look like: [[(0, 255, 0), (0, 155, 0)], [(30, 144, 244), (176, 224, 230)]]