TASK 1: Drawing Rectangle (Create a new python file)

In previous assignments, for the most part, you're asked to apply what you have learned in the class. In this class, you're asked to search for a solution and apply that solution to the problem stated.

1. Draw two triangles so that it can make a perfect rectangle using just what we learned in lab 4 and 5.
2. If you draw the rectangle, you, most probably, used 6 vertices. If you noticed, two pairs of vertices are the same, leaving you with 4 distinct vertices. The two additional vertices cause wastage of processing time of our machine and its storage. If we are drawing 100,000 of rectangle, around 30 %, 30, 000 * 12 bytes = 240Kb space and its processing time is wasted on these unnecessary vertices. Please read on Element Buffer Array and use only 4 vertices to draw the box.

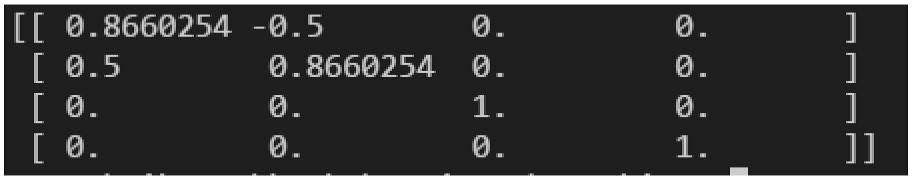
You may roughly add 7 lines of code and change 1 line of code.

3. What is wireframe mode? Implement on your rectangle. Add only 1 line of code.

TASK 2: Uniform variables (USE THE LAB 6 python file)

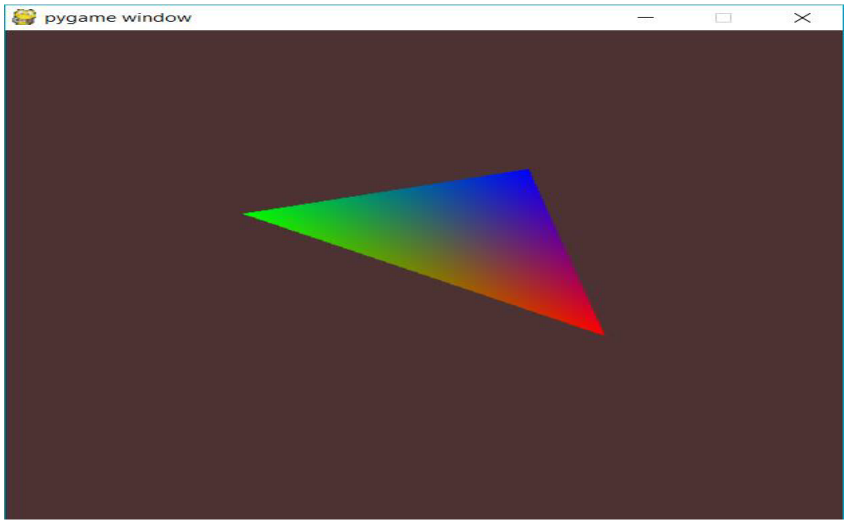
We can control our shaders with uniform variables from our OpenGL application on the CPU. Please read about uniform variables from the OpenGL book.

1. In our vertex shader create a uniform variable called transform with data type of mat4 and assign it to a rotation matrix of 30 degrees. Since we have 4 attributes for the position, we need to have a 4 X 4 matrix. Use this matrix



```
[[ 0.8660254 -0.5        0.         0.        ]
 [ 0.5        0.8660254  0.         0.        ]
 [ 0.         0.         1.         0.        ]
 [ 0.         0.         0.         1.        ]]
```

2. Assign gl_Position to the given vertex position multiplied by transform.
3. Run your application to see if the triangle rotated -30 degrees.



4. Just above the glDrawArrays function

```
rotateMatLocation = glGetUniformLocation(program, "transform");
rotateMat = transform.rotationMatrix(90)
glUniformMatrix4fv(rotateMatLocation, 1, GL_FALSE, rotateMat)
```

a. the first line gives the transform attribute location in the vertex shader you just created
b. the second line creates rotation matrix of 60 degree
c. third line assigns the rotationMat to the transform variable in the vertex shader (GPU) from our program (CPU). Please refer to glGetUniformLocation and glUniformMatrix4fv from the book or OpenGL documentation. Don't forget to import transform4d.py. the transformation returns 4 X 4 instead of the old 2D matrix.

If you run your application you will get a triangle that rotated 90 degrees

5. Try to make the triangle rotate 360 degrees continuously instead of just rotating a certain given degree and sticking at it.

TASK 3: (Use the CUBE_LAB.PY)
Create 5 more Rectangles to make a cube
1. Calculate and find 4 more vertices that make up cube with first rectangle
2. In our vertices array, add the 10 more triangles made from the 8 vertices, making it totally 12 triangles.
3. Change the glDrawArrays function to include all vertices
4. Use the following functions to make the drawing proper cube in the draw function.

```
#enable clearing depth buffer
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
#enable depth test
glEnable(GL_DEPTH_TEST)
#apply the depth checking function
glDepthFunc(GL_LESS)
```

Rotate the cube
1. Create a 3D rotation Matrix
2. Using Lab 5 uniform variable, apply the rotation of 30 degree, in the axis (0, 0.6, 0.4)
3. Continuously change the rotation degree to have movement just like rotation of the triangle from lab 5

Instruction:

1. I expect a zipped file with the three different python files for each task and the necessary files you used.(shaders, images etc…)
2. Name each python file and the folder as YOURNAME_ID_SECTION.py

DEADLINE: JUNE 12 MID NIGHT