Lab 1
Computer Graphics CS4052

Edmond O’ Flynn
12304742

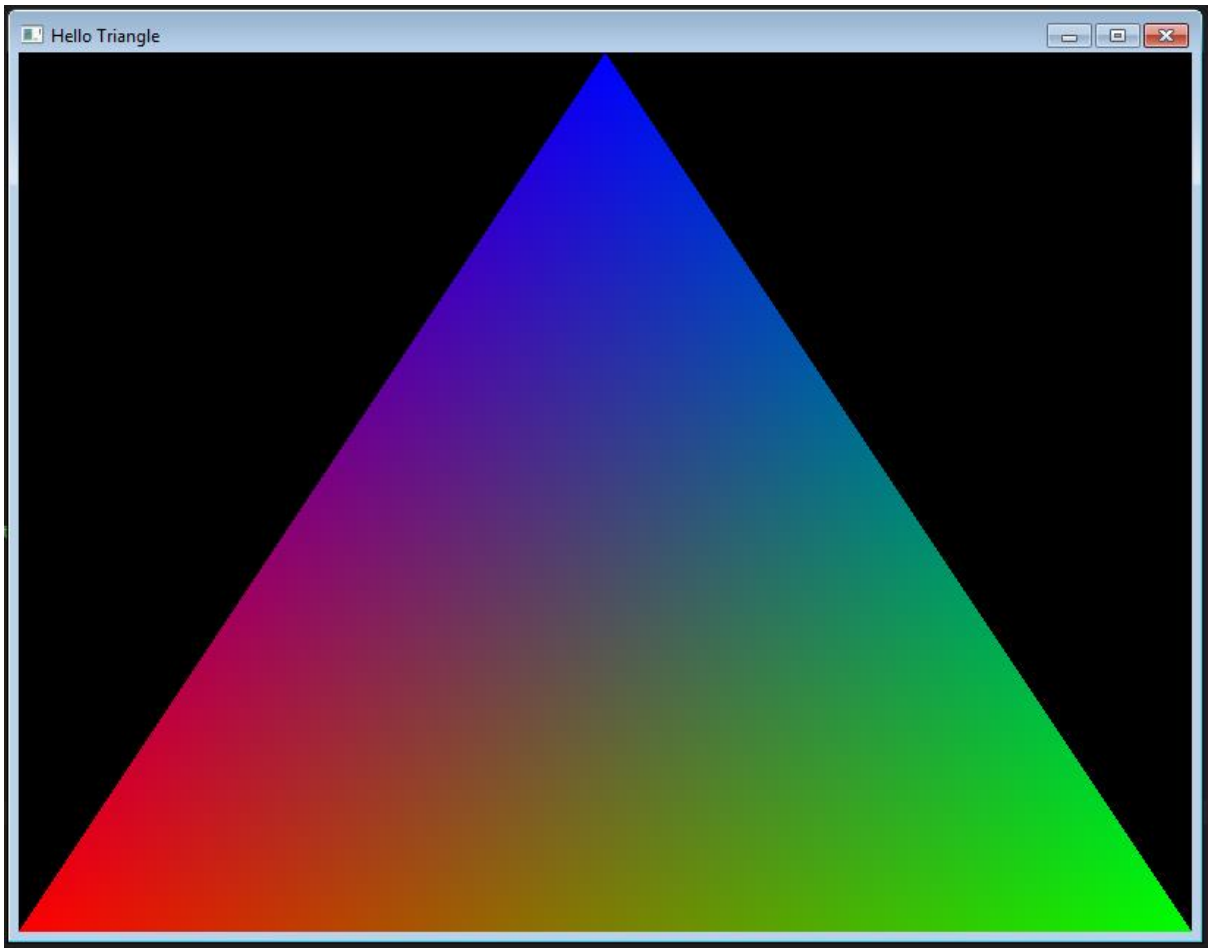
Part A.

“Now make the fragment shader use the colours specified in the vertex buffer object (the resulting triangle should be multi-coloured)”

I modified the sample code given specifically in the fragment shader to receive an input of a colour vector, and assign it to its output. Initially, the triangle’s colours were all assigned red which overrode the colours in the colours array. I removed this override and used its input as its output, noting the output of the vertex shader had the same name as the input of the fragment shader.

```
static const char* pFS = "                                                  \n\
#version 330                                                                \n\
in vec4 color;                                                              \n\
out vec4 FragColor;                                                         \n\
                                                                            \n\
void main()                                                                 \n\
{                                                                           \n\
    FragColor = color;                                                      \n\
}";
```

Modified fragment shader with input and output assignment



Resulting triangle on screen with various colours as per the colour array

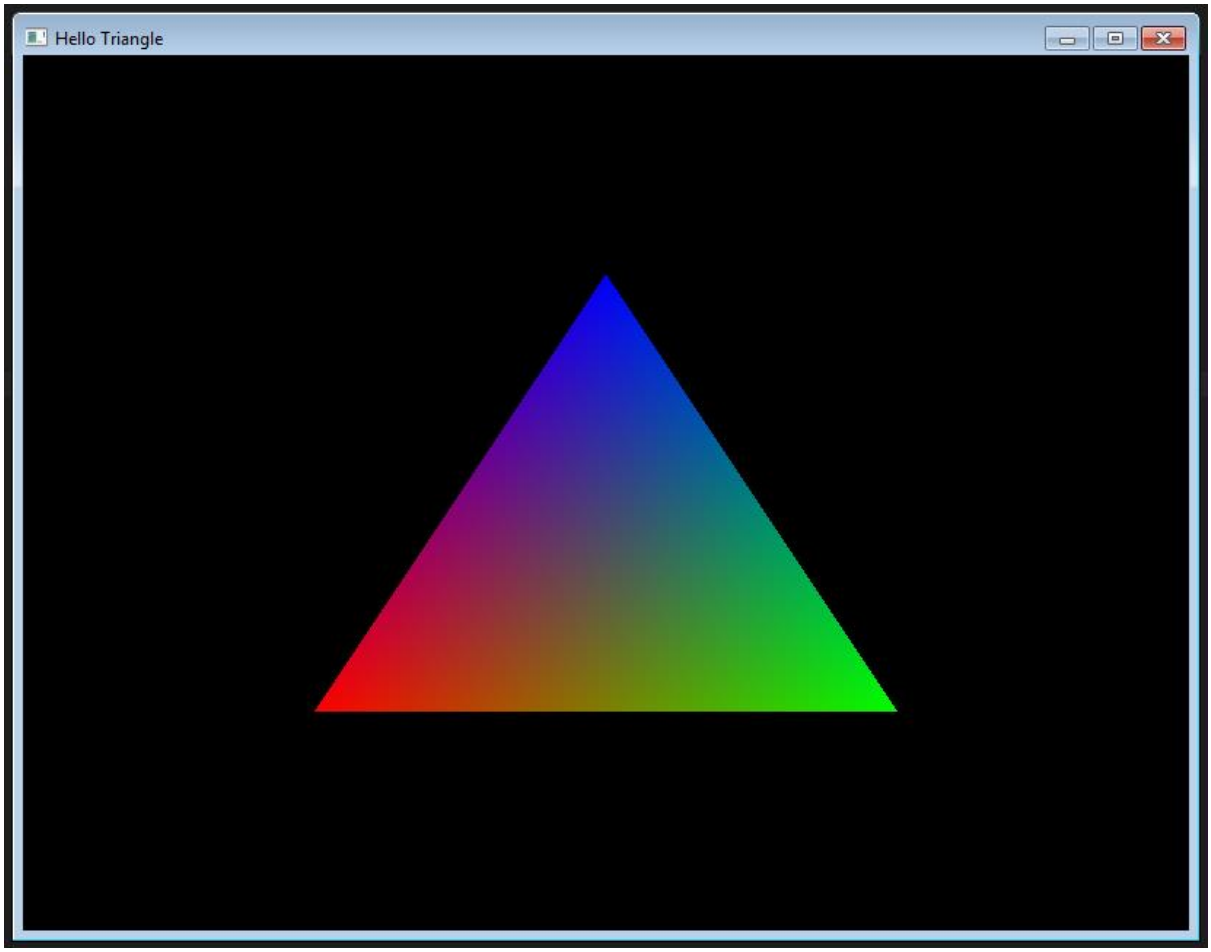
Part B.

“Try to make the triangle be half as big onscreen WITHOUT altering the vertices in the vertices array”

This time I modified the vertex shader to divide all vertices by a constant (in this case by 2). I kept the vertex array the same and only modified the vertex shader.

```
static const char* pVS = "                                                    \n\
#version 330                                                                  \n\
                                                                              \n\
in vec3 vPosition;                                                            \n\
in vec4 vColor;                                                               \n\
out vec4 color;                                                               \n\
                                                                              \n\
void main()                                                                   \n\
{                                                                             \n\
    gl_Position = vec4(vPosition.x / 2, vPosition.y / 2, vPosition.z / 2, 1.0);     \n\
    color = vColor;                                                           \n\
}";
```

Modified vertex shader with its gl_Position being assigned a vector subject to scaling by a constant



Output of the modified vertex shader resulting in a scaled by half over the original

Part C.

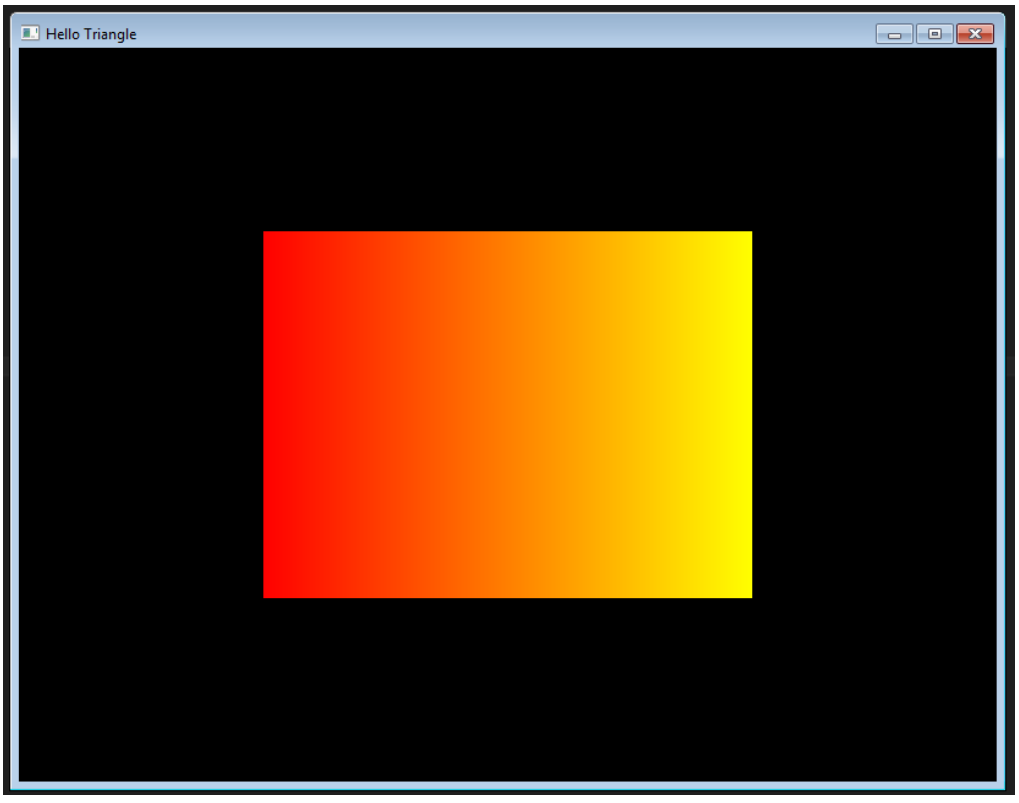
“Change the triangle to a square with 2 yellow and 2 red vertices”

This time I modified the vertices and colours arrays in init() in order to change the shape from a triangle to a square. I added a second triangle and modified the colours to display correctly on corners ranging from (1,0,0,1) to (1,1,0,1) in order to display red and yellow accordingly.

```
void init()
{
    // Create 3 vertices that make up a triangle that fits on the viewport
    GLfloat vertices[] = {
        //first triangle
        -1.0f, -1.0f, 0.0f,
        -1.0f, 1.0f, 0.0f,
        1.0f, -1.0f, 0.0f,

        //second triangle
        1.0f, 1.0f, 0.0f,
        -1.0f, 1.0f, 0.0f,
        1.0f, -1.0f, 0.0f
    };
    // Create a color array that identfies the colors of each vertex (format R, G, B, A)
    GLfloat colors[] = {
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 1.0f, 0.0f, 1.0f,
        1.0f, 1.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 1.0f, 0.0f, 1.0f,
        1.0f, 1.0f, 0.0f, 1.0f
    };
    // Set up the shaders
    GLuint shaderProgramID = CompileShaders();
    // Put the vertices and colors into a vertex buffer object
    generateObjectBuffer(vertices, colors);
    // Link the current buffer to the shader
    linkCurrentBuffertoShader(shaderProgramID);
}
```

Modified init() to show the change in vertices and colours



Resulting output with 2 red corner and 2 yellow corners in a square shape