CS4052 Assignment 1

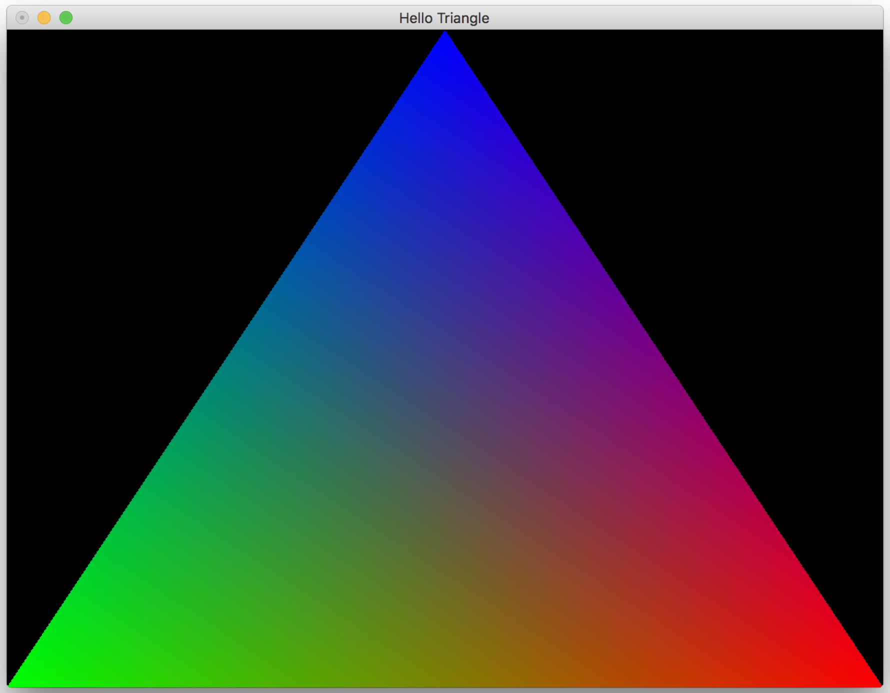
Luke Lau

September 22, 2018

1 Fragment shader colours

In order to get the fragment shader to use the position as the colour, I had to add an input `vec4` for the `vec4 color` that the vertex shader outputted. Since the inputs to the fragment shader are from the vertex shader, I had to use `color` and not `vColor`.

```
#version 330
in vec4 color;
out vec4 FragColor;

void main() {
        FragColor = color;
}
```



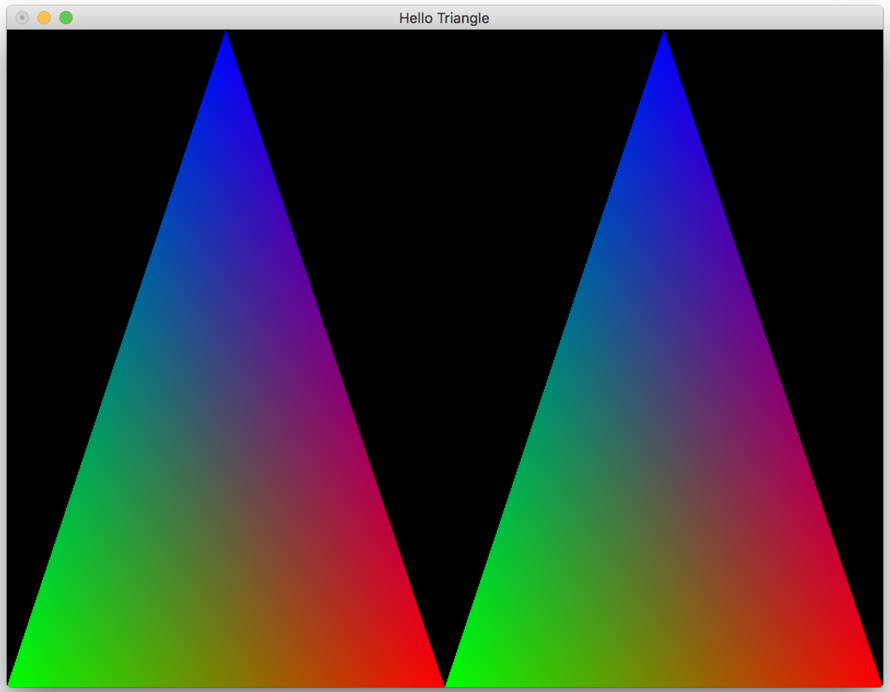
2 Two triangles

Two add two triangles, I had to increase the number of vertices drawn to 6

```
glDrawArrays(GL_TRIANGLES, 0, 6);
```

And add 9 new coordinates and colours for them.

```
GLfloat vertices[] = {
        0.0f, -1.0f, 0.0f,
        1.0f, -1.0f, 0.0f,
        0.5f, 1.0f, 0.0f,
        -1.0f, -1.0f, 0.0f,
        0.0f, -1.0f, 0.0f,
        -0.5f, 1.0f, 0.0f
}
GLfloat colors[] = {
        0, 1, 0, 1,
        1, 0, 0, 1,
        0, 0, 1, 1,
        0, 1, 0, 1,
        1, 0, 0, 1,
        0, 0, 1, 1
};
```



3 Two triangles, two VAOs and two VBOs

In order to use two VAOs and two VBOs, I had to combine the VAO setup and the VBO setup into one function, since you need to bind the VBO at the time that you create the corresponding VAO.

```
#define BUFFER_OFFSET(i) ((char *)NULL + (i))

GLuint setupBuffers(GLfloat* vertices, GLuint progId) {

        GLfloat colors[] = {
                0, 1, 0, 1,
                1, 0, 0, 1,
                0, 0, 1, 1
        };

        GLuint numVerts = 3;

        GLuint vbo;
        glGenBuffers(1, &vbo);

        GLuint vao;
        glGenVertexArrays(1, &vao);

        GLuint posId = glGetAttribLocation(progId, "vPosition");
        GLuint colorId = glGetAttribLocation(progId, "vColor");

        GLuint vertsLen = numVerts * 3 * sizeof(GLfloat);
        GLuint colorsLen = numVerts * 4 * sizeof(GLfloat);

        glBindBuffer(GL_ARRAY_BUFFER, vbo);
        glBufferData(GL_ARRAY_BUFFER, vertsLen + colorsLen, NULL, GL_STATIC_DRAW);

        glBufferSubData(GL_ARRAY_BUFFER, 0, vertsLen, vertices);
        glBufferSubData(GL_ARRAY_BUFFER, vertsLen, colorsLen, colors);

        glBindVertexArray(vao);

        glEnableVertexAttribArray(posId);
        glEnableVertexAttribArray(colorId);

        glVertexAttribPointer(posId, 3, GL_FLOAT, GL_FALSE, 0, 0);
        glVertexAttribPointer(colorId, 4, GL_FLOAT, GL_FALSE, 0, BUFFER_OFFSET(numVert

        return vao;
}
```

This could then be used like so:

```
GLfloat vertices[2][9] = {
        {
                0.0f, -1.0f, 0.0f,
                1.0f, -1.0f, 0.0f,
                0.5f, 1.0f, 0.0f
        },

        {
                -1.0f, -1.0f, 0.0f,
                0.0f, -1.0f, 0.0f,
                -0.5f, 1.0f, 0.0f
        }
};
GLuint* vaos = new GLuint[2];
vaos[0] = setupBuffers(vertices[0], progId);
vaos[1] = setupBuffers(vertices[1], progId);
```

When drawing, the VAOs needed to be switched out:

```
void display() {
        glClear(GL_COLOR_BUFFER_BIT);
        for (int i = 0; i < 2; i++) {
                glBindVertexArray(vaos[i]);
                glDrawArrays(GL_TRIANGLES, 0, 3);
        }
        glutSwapBuffers();
}
```

4 Two separate shaders

I modified the compileShaders function to load in the shader source from a file:

```
GLuint compileShaders(char* vertexShader, char* fragmentShader) {
        GLuint progId = glCreateProgram();

        attachShader(progId, vertexShader, GL_VERTEX_SHADER);
        attachShader(progId, fragmentShader, GL_FRAGMENT_SHADER);

        glLinkProgram(progId);
        GLint success = 0;
        glGetProgramiv(progId, GL_LINK_STATUS, &success);
        if (!success) {
                GLchar log[1024];
                glGetProgramInfoLog(progId, sizeof(log), NULL, log);
                fprintf(stderr, "error␣linking:␣%s\n", log);
                exit(1);
        }

        return progId;
}
```

Which meant I could easily swap out the programs used when setting up the buffers:

```
progIds = new GLuint[2];

GLuint progId1 = compileShaders((char*)"vertex.glsl", (char*)"fragment.glsl");
vaos[0] = setupBuffers(vertices[0], progId1);
progIds[0] = progId1;
```

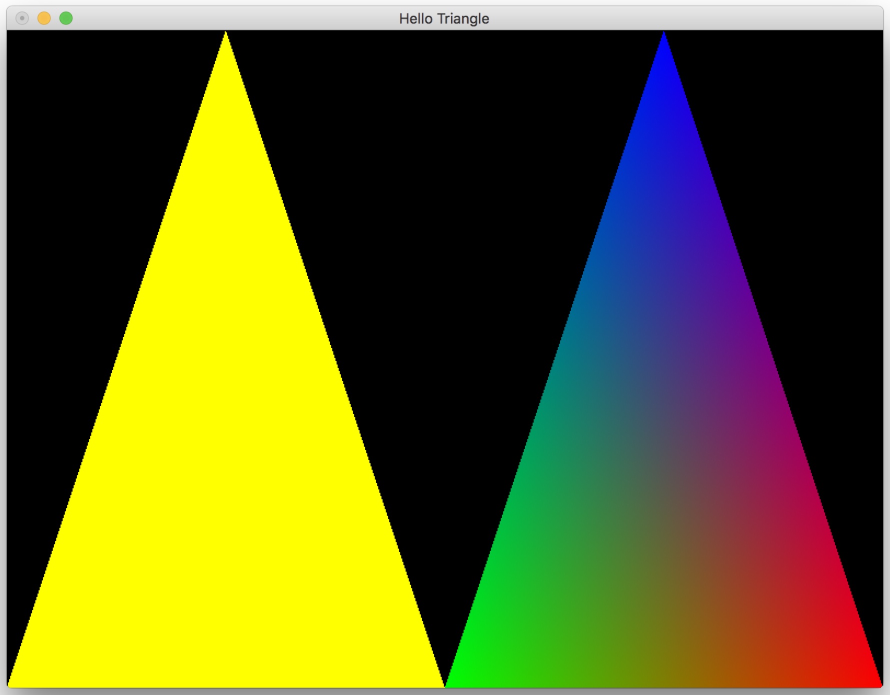
```
validateProgram(progId1);

GLuint progId2 = compileShaders((char*)"vertex.glsl", (char*)"yellow.glsl");
vaos[1] = setupBuffers(vertices[1], progId2);
progIds[1] = progId2;
validateProgram(progId2);
```

The new yellow shader looked like this:

```
#version 330
out vec4 FragColor;
void main() {
        FragColor = vec4(1.0, 1.0, 0.0, 1.0);
}
```

I needed to switch programs during the display function:

```
void display() {
        glClear(GL_COLOR_BUFFER_BIT);
        for (int i = 0; i < 2; i++) {
                glUseProgram(progIds[i]);
                glBindVertexArray(vaos[i]);
                glDrawArrays(GL_TRIANGLES, 0, 3);
        }
        glutSwapBuffers();
}
```


Hello Triangle