Assignment 0 - Rendering Triangles in OpenGL

Guilherme Gomes Haetinger - 00274792

August 3, 2021

---

1 Setup

I had problems building the code from the Github provided source. Thus, I adapted the source code to use my local GLEW and GLFW3 setup. If you are to run my code, make sure you have them installed in your lib path.

Not quite sure what the problem was, but it might have something to do with the compilation of the `vermilion` files.

2 Code

As a task, we also had to change how the triangles were built, removing one of them from the final image and making the remaining one colorful. To do so, we have to add in a few enums:

```
enum Buffer_IDs { ArrayBuffer, ColorBuffer, NumBuffers };
enum Attrib_IDs { vPosition = 0, vColor = 1 };
```

Also, as we remove one of the triangles, we also need to define a color array, leaving us with:

```
GLfloat vertices[NumVertices][2] = {{-0.90f, -0.90f}, {0.85f, -0.90f}, {-0.90f, 0.85f}};
GLfloat colors[NumVertices][3] = {{1.0f, 0.0f, 0.0f}, {0.0f, 1.0f, 0.0f}, {0.0f, 0.0f, 1.0f}};
```

Once we have them defined, we bind the `ColorBuffer`, set its storage to the size of the array and enable it.

```
glBindBuffer(GL_ARRAY_BUFFER, Buffers[ColorBuffer]);
glBufferStorage(GL_ARRAY_BUFFER, sizeof(colors), colors, 0);

glVertexAttribPointer(vColor, 3, GL_FLOAT, GL_FALSE, 0, 0);
glEnableVertexAttribArray(vColor);
```

All this data will go to the vertex shader and can be interpolated onto separate fragments processed by the fragment shader. So, the first thing we do is to receive and send a color `vec3` in the vertex shader:

```
#version 400 core

layout( location = 0 ) in vec4 vPosition;
layout( location = 1 ) in vec4 vColor;

out vec4 color;

void
main()
{
    gl_Position = vPosition;
    color = vColor;
}
```

And the fragment shader receives and forwards it.

```
#version 450 core

in vec4 color;
out vec4 fColor;

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

3 Results

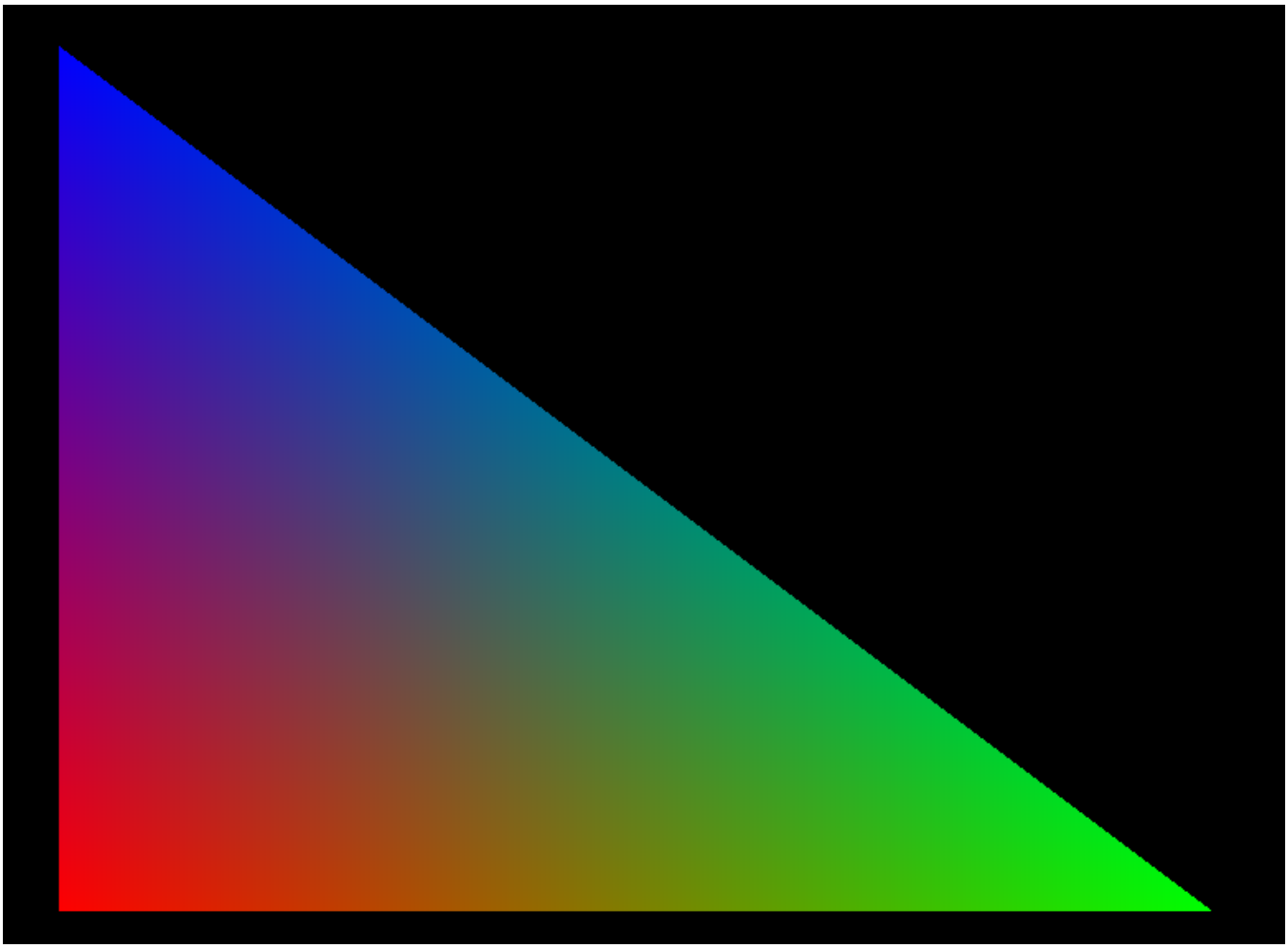

Figure 1: Colorful Triangle