Shaders OpenGL Exercises

Victor Gordan

1 Easy Difficulty

These only require the change of some signs or variables!

Exercise 1. Flip the triangles upside down using the Vertex Shader
Hint: you simply have to put two minuses somewhere

Exercise 2. Invert the colors of the triangles
Hint: The inverse of an RGB normalized color, is 1 - that color

2 Medium Difficulty

This requires a bit more typing!

Exercise 1. Make it so the bigger the triangle is, the more bright it is
Hint: make use of the `scale` uniform in the Fragment Shader

3 Hard Difficulty

This requires the use of extra functions!

Exercise 1. Make the triangles pulsate
Hint: use a sin wave to change `scale`, and use the function `glfwGetTime()` for time

Solutions

Each image shows the solution to one exercise

default.vert / Ex1.1

```
15 void main()
16 {
17     // Outputs the positions/coordinates of all vertices
18     gl_Position = vec4(aPos.x + aPos.x * scale, -aPos.y - aPos.y * scale, aPos.z + aPos.z *
   scale, 1.0);
19     // Assigns the colors from the Vertex Data to "color"
20     color = aColor;
21 }
```

default.frag / Ex1.2

```
11 void main()
12 {
13     FragColor = vec4(1.0f - color.x, 1.0f - color.y, 1.0f - color.z, 1.0f);
14 }
```

default.frag / Ex2.1

```
10 uniform float scale;
11
12 void main()
13 {
14     FragColor = vec4(color.x * scale, color.y * scale, color.z * scale, 1.0f);
15 }
```

main.cpp / Ex3.1

```cpp
// Gets ID of uniform called "scale"
    GLuint uniID = glGetUniformLocation(shaderProgram.ID, "scale");
    // The time of the last tick
    float lastTime = 0.0f;
    // added scale for triangle
    float scale = 0.0f;
    // Main while loop
    while (!glfwWindowShouldClose(window))
    {
        // Specify the color of the background
        glClearColor(0.07f, 0.13f, 0.17f, 1.0f);
        // Clean the back buffer and assign the new color to it
        glClear(GL_COLOR_BUFFER_BIT);
        // Tell OpenGL which Shader Program we want to use
        shaderProgram.Activate();
        // Changes "scale" regularly
        if (glfwGetTime() - lastTime > 1.0f / 60.0f)
        {
            scale += 0.05f;
            lastTime = glfwGetTime();
        }
        // Updates uniform value
        glUniform1f(uniID, 0.1f * sin(scale));
        // Bind the VAO so OpenGL knows to use it
        VAO1.Bind();
        // Draw primitives, number of indices, datatype of indices, index of indices
        glDrawElements(GL_TRIANGLES, 9, GL_UNSIGNED_INT, 0);
        // Swap the back buffer with the front buffer
        glfwSwapBuffers(window);
        // Take care of all GLFW events
        glfwPollEvents();
    }
```