Assignment 3 (Exercise 6.6)

Ross C. Hoyt

Oct. 14, 2019

**Assignment Objective:** Modify Exercise 5.3 (RotatingColorfulLetter.cpp) so that the letter rotates in response to user input, the letter rotates in three dimensions, and a matrix is used for transformations.

**How Achieved:** Before starting Exercise 6.6, I fixed the problems with discontinuities in my original H letter design by adding more vertices and triangles where needed. Then I completed Exercise 5.3 to better understand Chapter 5 material, duplicated that exercise’s application file (*RotatingColorfulLetter.cpp*) and renamed it to *Rotate3DLetter.cpp*.

Starting Exercise 6.6, I first decided to create more space in *Rotate3DLetter.cpp* by moving the vertex and pixel shaders to separate *.shader* files, in an internal project resources folder (I chose the .shader extension because it provides some syntax highlighting in VS 2019). To read the shaders from files, the call to *LinkProgramViaCode()* in method *InitShader()* was changed to:

*LinkProgramViaFile(“res/shaders/vertex.shader”, “res/shaders/pixel.shader”);*

Then, I added code to the Vertex Shader to support the use of a matrix. The full Vertex Shader:

*#version 130*

*in vec2 point;*

*in vec3 color;*

*out vec4 vColor;*

*uniform mat4 view;*

*void main() {*

*gl\_Position = view \* vec4(point, 0, 1);*

*vColor = vec4(color, 1);*

*}*

Next, I added variables related to mouse position in the application window, and how the application should interpret those variables, to *Rotate3DLetter.cpp*:

*float rotSpeed = .3f; // deg rotation per #pixels dragged by mouse*

*vec2 mouseDown(0, 0); // location of last mouse down*

*vec2 rotOld(0, 0), rotNew(0, 0); // .x is rotation about Y-axis, in deg; .y about X-axis*

Then I added callback methods to respond to mouse clicks and mouse movement:

*void MouseButton(GLFWwindow\* w, int butn, int action, int mods);*

*void MouseMove(GLFWwindow\* w, double x, double y);*

These callbacks were registered in *main* by adding lines:

*glfwSetMouseButtonCallback(window, MouseButton);*

*glfwSetCursorPosCallback(window, MouseMove);*

Then, I changed method *Display()* to compute the transformation matrix using routines in VecMath.h:

*mat4 view = RotateY(rotNew.y) \* RotateX(rotNew.x);*

*SetUniform(program, "view", view);*

**Resulting images:**

