Homework 2 Report

Zack Fravel
010646947

Problem Statement:

The goal of this programming assignment was to extend Homework 1 to generate a 3D model in OpenGL of our letter. Very similar to the previous project, we ask the user input a letter they wish to generate (my program handles L, I, O, U, and E) along with X, Y, and Z - axis scaling information; after this, the program will output an OpenGL window with our desired letter drawn out in a rotatable 3D model.

The error handling required for this program is similar to the last in that the user inputs need to all be validated. I have the program output all the calculated points to the console as well, as a way of validating the outputted shape.

Design:

Thankfully I designed my Homework 1 in a way that was very easy to expand functionality to perform this task. To create a third dimension, I just had to add a z variable to my Point class and give it respective get() and set() functions. I then had to expand the program to be able and display an OpenGL window.

To display the points in OpenGL, I had to make a few modifications. Instead of the variables being passed around being declared in main I had to make them global variables to give the OpenGL display() function access to the points array and the variable for number of sides. The pro's of structuring my design like this are that it is extremely simple to feed GL_LINE_LOOP() an array of vertices.

Implementation:

I began with some sample code (src/building.cpp) that helped me formalize a way of solving this problem. The sample code mostly helped me learn how the OpenGL functions are called and how data needs to be passed. Along with that, I also found an implementation for rotating a 3D model using the keyboard in the same program. This made the rotation functionality very trivial to implement.

Another difference is now that after the user enters the desired letter, I create a copy of the X,Y points in the array that come directly after the first set of points. These represent the "back" of the model where the first set represents the "front." After initializing the points[] array from gathering user input, I extended the original program by now calling a createModel() function that sets the respective depth points for a 3D model with the X and Y points already in place. These points are then passed to the OpenGL display function which uses a two different GL_LINE_LOOP() iterations. The first iteration creates the outline of the front and back, the second iteration connects the two. The main thing to look out for is I had to add an if-else-if statement when creating a line for the final point in each set to make sure it was set to the correct depth and not creating diagonals.

Testing:

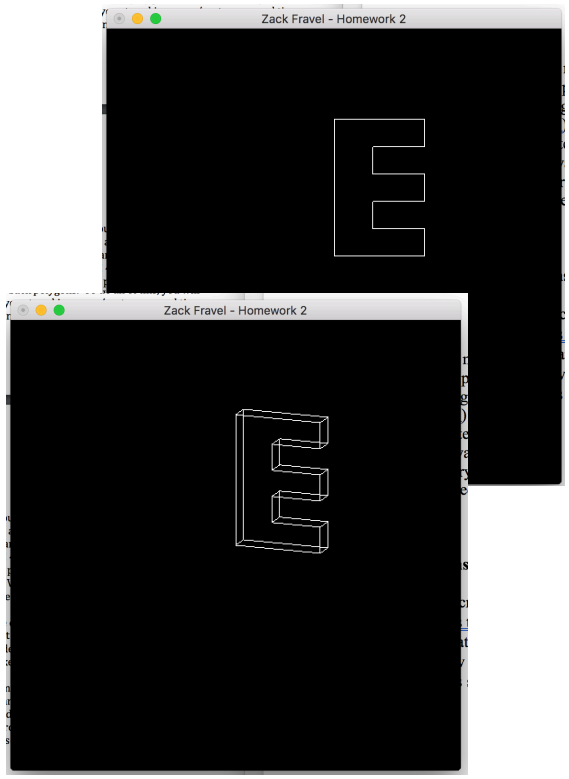
The main way I tested my program was through very incremental development and always printing to the console variables of interest to make sure I always knew what was working and what wasn't. The most involved testing came when developing the drawLetter() function since this was the main functionality of the program. Since I was able to rotate and look at a 3D model it was fairly easy to debug and see how the algorithm was working and tracing out the letter to find points of failure.

Everything works exactly as expected, no shortcomings that I have found. Blow I have included a sample execution.

```
Zacks-MacBook-Pro:hw2 Meryl$ ./hw2

Enter a single uppercase character to output [A-Z]: E

Generating data for the letter E

Enter desired X-axis scaling: 20
Enter desired Y-axis scaling: 30
Enter desired depth: 5
```

```
Keyboard commands:
   'x' - rotate x-axis -5 degrees
   'X' - rotate x-axis +5 degrees
   'y' - rotate y-axis -5 degrees
   'Y' - rotate y-axis +5 degrees
   'z' - rotate z-axis -5 degrees
   'Z' - rotate z-axis +5 degrees
```

```
Front
(0,0,0)
(0,30,0)
(20,30,0)
(20,24,0)
(8.57143,24,0)
(8.57143,18,0)
(20,18,0)
(20,12,0)
(8.57143,12,0)
(8.57143,6,0)
(20,6,0)
(20,0,0)
Back
(0,0,5)
(0,30,5)
(20,30,5)
(20,24,5)
(8.57143,24,5)
(8.57143,18,5)
(20,18,5)
(20,12,5)
(8.57143,12,5)
(8.57143,6,5)
(20,6,5)
(20,0,5)
```



Conclusions:

The overall result of the assignment was a huge success! We are able to successfully take in a user inputted capital letter along with scaling and depth information and output a 3D model of the letter in OpenGL. If I were to do it again I would do everything the same most likely because it was fairly trivial to expand my previous assignment to accomplish this. The project took a total of probably around 3 or 4 hours.