Brian Peters

CS-499

Module 7 Final Project

**Artifact 1 – Software Design and Engineering**

I created this in the lamp shader, so it goes into the vertex shader and out to the fragment shader.

(line 171)



Staying in the lamp shader, I added the integer “flip” where the function will be found through the vertex.

(Line 176)



Then connecting the “iflip” and “flip”

(Line 182)

A screen shot of a computer

Description automatically generated

The next lines of code have the integer “iflip” flat in and which is now “flip” and “iflip” is connected and this is in the fragment shader source code

(Line 192)

A screenshot of a computer

Description automatically generated

Then the next part within the fragment shader source code, I took out the original and made if-else statements to have it change specific colors.

(Line 195-211)

A screen shot of a computer program

Description automatically generated

Then I made an integer “flip” of a value of one to have it where it has a certain value and for this case, I made it as one.

(Line 269)

A black background with white text

Description automatically generated

And here is where it ties all together, which I stumbled on quite of bit. Flip + 1 % 5 is going to change five times and then start over. This is part of the functionality with the if-else statements. This is in the mouse button callback function

(Line 475)

A computer screen shot of text

Description automatically generated

Here's another part that I had a hard time with how this works. glUniformLocation is going to locate the lamp program and the “flip function. Next is to set up the vertex shaders

A screen shot of a computer

Description automatically generated

After all that, the lamp can change colors. With this process, there was a lot of learning just because of enhancing code as your project.

**Artifact 2 – Algorithms and Data Structure**

To recap what I will be reusing is the color change if-else code

(Line 203-219)

A screen shot of a computer program

Description automatically generated

Then I created a Boolean “flipActive” that equaled to false. This was created when I made another if-else to incorporate it into whether the loop was active.



Next, I added the double elapse to work with the delta time and it’s equal to zero as a starting point.



On line 328, elapse is equal to delta time, and what the variable is to the delta time that is currently equal to will add to the elapse as the same variable. (+=)

A screen shot of a computer code

Description automatically generated

After that, I set up elapse greater than 1. Flip = flip +1 will just add one as a variable. % 5 will allow it to go five times. Then elapse equal to 0 would start it over in the code. Lastly, the flipActive is added to the loop.

(Line 329-332)

A screen shot of a computer program

Description automatically generated

Then I set the integer of the number of colors



Then I set the loop up for the number of colors equal to 5 which is the number of colors I set up and made the flipActive equal to false to stop after the cycle. Then had the number of colors equal to 0 to go back to the starting point.

(Line 334-336)

A screen shot of a computer program

Description automatically generated

And finally, add flipActive under the right-click mouse button code and equal it to true.

(Line 514)

A screen shot of a computer

Description automatically generated

I also decided to add movement of the light and see if I could figure out a way to do so and I did! Here’s the code for that

A computer screen shot of a program code

Description automatically generated

**Artifact 3 – Database**

This one was a complete overhaul after instructor feedback. The plan was to store data of the object file in the database system. To start, I created a database called “obj\_data” and made tables of vertices, textures, normals, and faces. The vertices contain x, y, and z values and they were a *float* that way they could contain a decimal point. Textures had u and v as float. Normals had nx, ny, and nz as floats. Lastly, faces had indices of vertex, texture, and normals by using varchar or as in storing text at 255 for the maximum. After that, I created a DDL for the database –

A close-up of a computer code

Description automatically generated

The reason why I stored these in tables like this is because when you look into the obj file in Notepad++, you see the textures normals, etc. Here what I see (part of) for the lamp

A screenshot of a computer

Description automatically generated

Then after speaking to a tutor about what I needed to do next, I needed to parse it. I created an OBJLoader.cpp file with tutorials help from YouTube videos and from the tutor. I had to prefix the “v”, “vn”, etc.

A screen shot of a computer program

Description automatically generated

Then I created a OBJLoader.h file which I have done before. This way is to struct the values which defines them.

A screen shot of a computer program

Description automatically generated

Lastly, I created the OBJ\_Database.cpp to put it all together. The part that kind of concerns me is when the code connects to my IP address which if I had this into a cloud server, I would have the driver connected there instead.