# CS 330 Reflection and Design Decisions.

## SFlaten

For this project I decided to use a Toblerone pyramid, a soccer ball sphere, and a chalk box that’s a cube. All these objects are very recognizable. The Toblerone has both rectangle and pyramid attributes with triangles. I chose the soccer ball because it could be broken down into different quadrants. For instance, there could be several ways to draw a soccer ball. You could use a for loop and draw the same one just increments location and size. But I could also import the standard library of glut and glue to draw. This project I reused the same shader and matrix. I was working with one pipeline for each aspect and those being passed one big matrix array object. There are different ways I could organize this. But due to time restraints I decided to go with the instructor’s code and refractor it. If I had to use multiple objects in the code in the future; I would’ve broken them down into separate files and called each function in the main(), as it’s own object like drawCube() drawTolberone() then at the end call render(). All the supplied sample code was mixed in with each other. It was too complex to call a render for each individual shape that way. However, I did try and it became too complex to even read the code. Another option for the plane, I didn’t have to use the matrix I could’ve just said draw plane and given it an x and y. But for simplicity I stuck to just one array. At the very end of the array I’m calling g.mesh to draw the cubes for the lights, rather than using the other way of index start and number. The soccer ball, I was wanting to try an edgeless sphere and started to with libraries, but realized a soccer ball has faces to it, and I prefer to draw the sphere but with faces. This way you know it’s a soccer ball.

The navigation part of the project was simple once I found the camera.H file. This kept the code in reusable modules. The file showed how the directions were being created. It took a little searching to figure out how to direction variables were being calculated. This was just a matter of reusing the same calculations for the up, down, left, right, forward, backward, pitch, yaw, worldUp. The only thing the cameras are doing is incrementing a variable within a timeframe. For using the mouse to look up down left and right I had to adjust the speed a little bit, which helped. For the mouse movement I chose the variables pitch and yaw because if you were standing in the middle of the scene the pitch would roll the scene up or down in the yard would roll of the whole scene left and right. For looking up I just use World-up because it was calculated in the camera.h

Customizations. I kept it modular and in best coding practices. I kept most of the customization in the camera.H. Although from switching to orthopedic to perspective I simply just made a new variable and called it in our logic statement with a bool to see if it was what it was or if it wasn’t what it was. I had use in-depth knowledge of C++ to figure out what type of variable I was calling and where it would be placed in the pipeline. When customizing the lights I simply just remade the variables and translate to place the new cube. I left the lamp orbiting for some reason. I re-assigned the variables in for the color. This way I could use the same main() shader. This could have been done differently for each light. Because the program reads, assigns and loads each second into the shader pipeline I just had to make a change of the global in that light section.

Here is the incremented mouse in the .cpp and .hText, letter

Description automatically generatedText, letter

Description automatically generatedText

Description automatically generated

A picture containing text, whiteboard

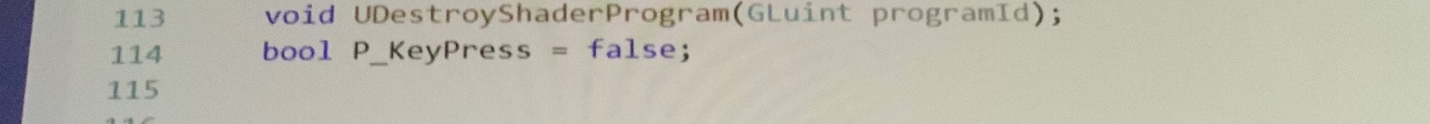
Description automatically generatedText, letter

Description automatically generated

Here are the global variables being redefined before sent to the shader pipeline to be rendered in the light cube #2.

This is how the texture is applied to a particular portion on the matrix

A picture containing text

Description automatically generatedThis is the switching between views. The projection variables is called and redefined upon switching.Text

Description automatically generatedText

Description automatically generated with low confidenceText, letter

Description automatically generated

Here I used a global to translate and scale the 2nd light

Here is how I loaded the textures, notice the const char has been declared, no need to do it again. Text

Description automatically generatedText, letter

Description automatically generatedGraphical user interface, text

Description automatically generated I think it shows a really good 3-D perspective. The projection variable is reassigned.

References

Vries J.(2014). *LearnOpenGl* Retrieved from URL: https://learnopengl.com/Introduction