Project Reflection:

The image that my scene is based on can be seen below -

A picture containing ground

Description automatically generated

The shapes that I used to develop this scene were planes, cylinders, cubes, and a pyramid. I used planes for the laptop case and table as these were flat surfaces. The laptop case flap is a cube along with the phone. By making these with cubes, I could adjust their height to give them a more realistic design. The keychain is made up of multiple cylinders and a pyramid. The key ring and the chain links were set to different sizes to account for their different uses. The cup and the straw were also made using a cylinder shape. I had to update the cylinder.cpp file to account for the fact that almost all my cylinders did not need a top or bottom cover. An extra function was added to include the bottom cover on the cup, but not the top. I ended up using textures for all my objects. For the table itself, I ended up using a screenshot of my reference image so that my rendered result would be more accurate. Other textures I chose included grey mesh fabric, pleather, a black background, plastic, an image of Thor’s hammer, stainless steel, and an outline image I made for the keychain myself. When selecting these textures, I made sure to use Google’s advanced search feature as instructed to make sure my images are large enough and have legal usage rights. There was a big learning curve when it came to creating multiple 3D objects. It was difficult at first to get a shape to generate how I would like and to update its scaling, translation, and rotation to get it in the right spot. For the laptop case, flap, phone, and keychain pyramid I had to create a set of vertices for each. Through a lot of trial and error, I developed a better understanding of how to update the x, y, and z values to properly extrude my shapes. Developing the rotation, scaling, and translation was a thorough process. In my development, I realized that scaling an object can have a large effect on how an object translates. I also recognized that rotating an object before translating it helped me personally with placing the object into its expected location.

The camera position I set up when you first load the project was developed to mimic the view of my reference photo. The camera position was an important part of how a user can experience my scene. Using input keys W, A, S, D, Q, E, mouse cursor, and mouse scroll I can position the camera to navigate myself through the scene. The W, A, S, and D keys were used for your typical forward, left, right, and backward motions. Q and E were used for going up and down. The mouse cursor could be used to look left, right, up, and down the environment by updating the camera's orientation. To zoom in and out a user can utilize the mouse scroll. To get these inputs, I had to use the glfwGetKey() function. By using this, I could capture what button was pressed. When a key press registers then the camera position could be updated according to the requirements as stated in our guidelines for this project. For example, the W key is used to move forward so when that key is pressed and registers through the processInput() function then the camera position will increase based on a set camera speed and the camera’s frontward orientation. The mouse scroll simply updates the field of vision. When the mouse cursor moves, the mouse\_callback() function updates the camera's front-facing orientation based on yaw and pitch values dictated by the position of the cursor.

As mentioned previously, one of the functions I created was called renderCup() for the cup object. I added this function to the cylinder.cpp class file and declared it in the staticMesh3D.h header file. By doing so, I could easily call the function to generate different objects using the same header file. This was necessary since the cup object needed a bottom cover while the straw and key links did not. The functions declared in my project were set in a specific order to avoid confusion during development. I had lighting established at the beginning through numerous functions such as URender(), UCreateMesh(), UDestroyMesh(), and a void main() function. Other activities were set in the order of vertices, textures, and object placement. These activities were all completed within the int main() function. At the bottom of the project's source file, are the functions responsible for user callback functions for use cases such as camera navigation. This includes processInput(), framebuffer\_size\_callback(), mouse\_callback(), and scroll\_callback(). Using openGL was a great opportunity to practice organization since it tends to take a lot of lines of code to create a scene. Keeping functionality close together was imperative for quick skimming back and forth during development. This is a skill I will take with me in my career moving forward.

**References**

*docs.gl*. (n.d.). https://docs.gl/

*GLFW: Input guide*. (n.d.). https://www.glfw.org/docs/latest/input\_guide.html

*LearnOpenGL - Hello Triangle*. (n.d.). https://learnopengl.com/Getting-started/Hello-Triangle