### SNHU

### 7-1 Final Project: Reflection

Andreas Galatis

### Jeff Phillips

### CS 330 Computer Graphics and Visualization

August 14, 2022

**Final Project Reflection**

**Justify development choices for your 3D scene**

The 3D scene I selected (pictured below) consists of five objects: A tabletop, a small box, an eye-glass case, a candle, and a ball. These items were selected mainly to fulfill the

A picture containing indoor

Description automatically generated

requirements to select various primitive shapes that must appear at least once in the image. These shapes include a cube which is represented in the box, a plane, which is represented by the tabletop, an extended cube with a pyramid, which makes up the candle, a sphere, which is represented by the ball, and lastly a complex object consisting of two primitive shapes, a cube and a pyramid, is represented by the eyeglass case.

Each of these objects were drawn using multiple triangles which are drawn by identifying vertex positions. The cube for example has eight vertex positions, located on the end of each corner. By incorporating X, Y, and Z coordinates, these vertex positions were mapped into a three-dimensional space. The vertices were then connected to each other, three at a time, to create the eight faces of the cube. Each face had two triangles that made up the square, totaling 12 triangles to make a cube. In addition to creating the shape of the cube, this process also creates a mesh for the cube for various other elements, such as color and texture to be added to the cube.

The mesh, containing each vertex position, along with the color, and texture data is then stored in a vertex array, and then passed on to the vertex buffer, before being sent to a shader, which is the program that draws the object. Before rendering, however, a window must be created for the objects to be displayed in, and this is done by using the OpenGL function glfwCreateWindow and passing the width and height of the desired window.

**Explain how a user can navigate your 3D scene**

Some additional functions that enable the use of a mouse and keyboard to interact with the image were also added from the GLFW library. For the keyboard input, the function glfwGetKey was used to execute camera movements in conjunction with a separate Camera.h class that was downloaded from learnopengl.com (DeVriez, 2014). The “W” key is used to move the camera forward into the screen, “S” to move backwards, “A” to move to the left, “D” to move to the right, “E” to move up on the y-axis, “Q” to move down, and lastly the letter “P” to switch between an orthographic and perspective projection. An orthographic projection renders a 3D scene as if it was a 2D picture, meaning flat and not exactly true to 3D dimensional sizes when moving closer or further to them. A perspective projection, on the other hand, is a more realistic view of the scene, causing objects in the scene to change sizes as you move closer or further away from them, as well as navigate above, below and all around them, just like in real life. The Camera.h class also has a function to process mouse movements, which are assigned X and Y positions with offsets to record where the mouse is moving and have it registered in GLFW’s mouse call back functions to interact with the window.

**Explain the custom functions in your program that you are using to make your code more modular and organized.**

In addition to the downloaded Camera.h class mentioned above some other functions were also created to make the code easier to read as well as implement more objects in the future, or to reuse for another application altogether. Some of these functions include a function to process all the keyboard inputs by passing a parameter taken directly from the window output; a function to take in the mouse inputs, as well as the scroll and button click callbacks; two functions that handle all the rendering, one in orthographic projection and the other in perspective; several functions to create each of the unique meshes for each object, before passing them over to render; several functions to also bind textures to each object; a function to create the shader program; several functions to destroy the shaders, meshes and textures after use; and lastly a main function that consists of calls to load images and link them to the proper texture binds, initiating the shader programs for each object, all before calling the render functions and input processes located in a while loop, that exits only when the user closed the window. This enables continuous rendering and navigation inside the window, creating a real-time effect.

**References**

DeVriez, J. (2014). *Learn OpenGL, extensive tutorial resource for learning Modern OpenGL*. Https://Learnopengl.Com. Retrieved August 14, 2022, from https://learnopengl.com