Final Milestone Reflection

The four objects that I chose for my project allow me to apply different methods and require the use of more complex objects from simple shapes. The first object I chose is the laptop. The laptop is the first complex object in my scene that requires multiple layers of the plane shape to simulate the backend, screen, and bezel of the top part of the laptop. Initially I was going to build every key using smaller square planes but that would take an immense amount of time, so I opted to use a keyboard texture on the bottom part of the laptop instead. I found a texture that fitted perfectly and didn’t repeat itself like most textures. The second object is the water bottle. This is the second complex object that requires a cone, and cylinders. Another reason I chose the water bottle is that it has a clear texture, so it allows me to mess around with alpha to reveal the water that’s inside. I simply changed the alpha from value 1 to 0.3. The third object is the air pods case. This object was the most complicated to build. The rounded corners and edges of the case required me to use more than just a rectangular box. I attempted to round the edges by using mini cylinders laying down, so this is my best approximation of an air pods case with rounded edges. The last object is a lunchbox. The actual lunchbox was the easiest as it is a large rectangular box with some plane shapes attached to it to show different textures like the gray patch on the original picture. The strap is made from flattened torus with a fabric texture. I realized too late that we can render a half torus, so the strap is made from a full-sized torus that hides half of itself within the lunchbox.

For the most part, implementing 3D camera navigation controls wasn’t too difficult although I did have a bit of trouble implementing the camera speed adjustment by using the mouse scroll wheel. Mainly three methods are used to control the camera which are “processkeyboardevents”, “Mouse\_Position\_Callback”, and “MouseScrollCallback”. The first method recognizes input from the keyboard via the event queue, which is one input recognition method. It will first call the glfwGetKey which receives an application window and a key as arguments and the return value is either “Key pressed” or “Key released”. If returned value is “key pressed” then the equal statement becomes true prompting an action tied to that specific key. For example, if the glfwGetKey returns true for the escape key it will then be compared with “GLFW\_PRESS” thus becoming a true statement and the following code block (application window shutting down) will execute. This is done for all required keys (A, S, D, etc.). The mouse position callback function works by capturing the first x and y position of the mouse and offsetting the position of the new coordinate based on how far the mouse position moved. Lastly the “MouseScrollCallback” function calls upon the ProcessMouseScroll function via the camera pointer to adjust the speed of the camera depending on if the scroll wheel goes up or down.

Modularity in a program is important; it allows for easy modification of parts of code without having to interact with multiple sections, apart from modifying the header files for declarations. We start by looking at the camera movement functions. The process keyboard function can be modified by itself without having to interact with any other part of the program. Within this function you can add or delete keys and their functionalities with ease. The mouse scroll back function can be removed without any issues if you wish to do so. When looking at the SceneManager cpp file, it’s simple to add and delete elements from your scene. The main function in the file is the creation of objects via simple shapes. Each shape can be easily added to the scene with a predefined signature format using certain attributes (color, position, scale, etc.). It’s important to follow this signature for all shapes, and if any stray from the format, then it can easily affect all other shapes rendered after. Lighting sources, textures, and materials can also be easily added, each with their own specific format.