**Reflection**

The objects I selected for my scene originally were a campfire, two books, a copper sphere with a copper base and a Vitamin C container. Later during the development of the scene, I decided I wanted to incorporate a Rubik’s cube to increase the colors in the scene and to make it feel less empty.

These objects gave me a wide range of primitive shapes to practice with which include a cube, multiple spheres, multiple cylinders, a pyramid, and many planes. They also allowed me many options when recreating the objects in 3D. This is seen when rendering the campfire stones, I originally planed on using a torus but quickly found out it didn’t look as good as individual spheres and it was also more complicated to program.

Some of the required functionality was hard to program such as putting multiple textures on a single object, in many cases this was solved by using a plane to cover the desired section with a different texture, making the scene have more detail. This can be seen in the book stack and the vitamin container.

The book was the hardest to texturize, I gave it the base page texture, then put a plane on the top to simulate the book cover and a plane on the side to simulate the book spine.

A table with objects on it

Description automatically generated with low confidence

My scene can be navigated using a mouse and keyboard as input devices. When using the keyboard the user can go forward, back, left, and right with the ‘W’,’S’,’A’ and ‘D’ keys. They can go up with ‘Q’ and down with ‘E’. The mouse can also be used to move around the scene. The camera moves according to where the pointer is, and the scroll button can be used to accelerate and decelerate the camera’s speed.

I believe if needed I could also set up different input devices to navigate the scene such as a controller or a touch screen.

Attached is an image with the code for movement using both input devices.

Text

Description automatically generated

The functions I use such as mouse\_callback, frambuffer\_size\_callback, processInput, etc., allow me to keep the code organized and modular. They give me the option of having all the related code in one area instead of it being spread across the program and they make the code easier to understand. It is modular because if I needed, I could copy that function and use it in another program, only having to change some logic slightly to better fit the new project.

Another function I developed was the orthographic view. This would change the original 3D perspective view into a 2D orthographic view when the ‘P’ key was pressed. It was created by having a Boolean switch between true and false when the hotkey was pressed, this would then influence what view got displayed.

Finally, the main function in the program which contains all the vertex data, VAO’s, textures and render calls is organized in sections, making it easier to locate information. This also facilitates the creation of new objects by setting a clear picture of the needed steps to render an item on screen.

If we wanted to make the code even more organized, we could have modularized some of the code in the main function. Each object could have been an individual function and we could have used render loops for multiple similar objects.

Attached are images of the different functions and how they are used.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated