Ryan Hoskins

CS-330

April 20th, 2024

Mr. Enkema

Module Seven – Reflection

Going into this project, I knew little to nothing of OpenGL. When considering options to use as a scene in this project, I wanted to choose something that utilized several shapes in interesting ways that would require me to combine ordinary shapes to complete them. For that reason, I decided to use the scene I based the project on. I was not sure whether to create the scene with only the perspective of the top of the island acting as the plane upon which everything else is created or to attempt recreating the island itself. Eventually, I decided to recreate the island, as I thought it would provide an interesting perspective and provide the opportunity to create flooring. The remaining objects presented a chance to create interesting combinations of shapes. For example, the ratan stand that the objects are placed onto could be created using a flatted cylinder with two Torus placed around the circumference of the cylinder’s top face. The vase was created using a sphere and cylinder, but due to the vase being of a glass material, I incorporated transparency into the coloring. The candle snuffer and candle provided a bit of a challenge, as they involved several parts that needed to be carefully placed, and the candle’s label was created by placing a cylinder without a top or bottom face. As an example of my approach to programming interesting shapes, the way I programmed the body of the glass vase can be seen here:  
A screenshot of a computer program

Description automatically generated

Together, the collection of objects presented an interesting opportunity for a challenging program that required a variety of requirements ranging from transparency, complex objects consisting of several other simple objects, and possessing the ability to manipulate basic shape meshes in a coordinated manner.

Composing the objects, establishing textures, and creating lighting conditions made up a large portion of this project. However, navigation also presented an essential component of the project. Providing the user with the ability to fully navigate the scene supports the 3D aspect of the scene. To provide a way for the user to navigate through the scene, it was important to incorporate input from the mouse itself as well as the keyboard. The mouse input was used to enable the user to change the direction they are viewing the scene while the scroll wheel provides the means to speed up or slow down navigating the camera through the scene. To accomplish this, two methods were created. The first, ProcessKeyboardEvents() within the ViewManager, and the second, MouseWheelCallback() within the ViewManager. As an example of incorporating keybord input to facilitate maneuvering through the scene, I utilized key bindings for specific keys:  
A screen shot of a computer program

Description automatically generated

Additionally, two other key bindings were included to provide the user with a quick way to view the scene in a Perspective and Orthogonal view. To do this, I established camera coordinates to set the positions, front, up, and zoom.   
A computer screen shot of a program code

Description automatically generated

This programming provides the users with a more immersive experience while navigating the 3D scene.

While the basic shapes used in the project were a considerable portion of the project, there were several other factors that presented challenging requirements, such as texture and lighting. To achieve a realistic scene, I utilized a combination of images that were used as the source material for wrapping the objects in a texture. The program utilized a LoadSceneTextures() method to streamline the process of loading these source images and apply a tag that could then be called upon for individual shapes. To further achieve a more realistic scene, I utilized the DefineObjectMaterials() method within the SceneManager to establish various shading attributes that could be applied to various objects. Lastly, the SetUpSceneLights method enables a modulate and organized way to create lighting in the program.

The program utilizes a wide variety of header files, structs, and classes with many objects that call upon these functions. To make the code more modular and organized, I called upon these functions. Each object needs to be scaled, positioned, transformed, textured, shaded, and drawn. Without the use of modular coding, the program would be of an enormous size and incredibly difficult to parse through. Each of the attributes I mentioned are established through the use of functions. As an example, this object was created as the label for the candle:A screenshot of a computer program

Description automatically generated

The position of the object is established using glm::vec3 followed by the 3D coordinates entered as floats. The texture is established by simply passing the tag that is used earlier in the program when loading the source image used as the texture. The shader material is also established by passing the tag used when defining the attributes for this material as well. Lastly, the object is actually drawn by calling on the DrawCylinderMesh function that quickly loads the pre-crafted mesh used for the cylinder.

To further illustrate how the program utilized modular code, the “parchment” material can be seen earlier in the program where it is defined and set:  
A screen shot of a computer

Description automatically generated

The “DefineObjectMaterial” method is a public struct used in the SceneManager header file that establishes ambientColor, ambientStrength, diffuseColor, specularColor, shininess, and the tag that can be used for referencing it. These help the program understand how to have lighting interact with any particular object with the material tag. By defining how the ambient, diffuse, and specular lighting should interact with the object, the program can establish realistic lighting conditions in the scene and 3D environment used. The tag streamlines the process of applying this material to the many objects drawn in the program.

This program seemed incredibly daunting and overwhelming at the start of the term, and I was honestly intimated by the thought of creating an immersive 3D scene. However, as the term progressed, the prospect seemed more and more within reach. This was a really challenging project that taught me a lot more than I thought it would be going into it, emphasized the importance of modular coding, and provided an incredibly rewarding result.

References

*LearnOpenGL - Materials*. (n.d.). LearnOpenGL. Retrieved April 21, 2024, from https://learnopengl.com/Lighting/Materials

*LearnOpenGL - Camera*. (n.d.). LearnOpenGL. Retrieved April 21, 2024, from https://learnopengl.com/Getting-started/Camera