Justify Development Choices

When choosing the original scene, I opted for colorful and fun while trying to keep the shapes a bit simpler in hopes of avoiding an overcomplicated project. As for shape choices, it was simple enough to create a complex shape which is just a cylinder and flattened sphere mesh joined together. After trial and error, even the simpler shapes were more complicated than originally anticipated. The pyramid mesh ended up being three separate color pyramids blended with a slight tweak in position to have a prominent color show on each face. The book ended up being three flattened boxes as well, one for the front and back cover, one for the pages in between, and another for the binding to match the rest of the object. Copying the same idea as the pyramid, the boxes are tweaked slightly to have prominent faces appear from the outside. In hindsight another option could have been creating a cover by using a plane mesh instead.

Textures paths were added to a LoadSceneTexture function to be used on shape meshes. This gave the code the ability to wrap objects in a texture or image from an outside source. Object materials such as a glass look were added to the DefineObjectMaterials method. This gave objects a textured shine depending on which material was chosen to help represent reflections of light. Light sources were added to the SetupSceneLights method which used two light sources with different colors to replicate sunshine from the windows. The PrepareScene method is where the majority of the code is pulled together. This is where we load all methods, meshes, lights, materials, and textures. The RenderScene method added shape instances to be loaded. After this each object is in its own method for rendering. This includes specifying objects, position, sizes, textures, and materials. The SceneManger header file was used to add add all textures, materials, lights, and shapes needed to be used in the SceneManager.cpp.

Explain User Navigation

Key Navigation Map

|  |  |  |
| --- | --- | --- |
| W = Zoom in | S = Zoom Out | A = Left |
| D = Right | Q = Up | E = Down |
| O = Orthographic view | P = Perspective Projection | Mouse = Change view |

In the ViewManager.cpp file, code was added to capture all mouse movements to include scrolling. In the ViewManager Header file I added a mouse scrolling callback function to help control the scrolling speed with a mouse. In the cpp file, a scroll tracker was added to the Mouse\_Position\_Callback method. Keyboard functions were implemented in the ProcessKeyboardEvents method which specified which action pressing each key would perform. A key was made for an orthographic view and another key was made for a projective view.

Custom Functions

The keyboard navigation is the most reusable portion of the code. The key choices are straightforward and could be used to navigate all types of 3D Scenes. The code has inline comments to explain each function and is clean and easy to read. Most people would be able to understand what it was doing. The light settings are modular and would be able to be used on other projects with tweaks to the values to get the desired look. The material sections can also be reused for other projects as they are named appropriately which helps explain what look that texture will achieve.