Final Project Design Decision Reflection

I went over each learning module before the class started to gain a comprehensive understanding of the project's requirements. At first, I intended to use tapered cylinders or cones to depict the trees (complex objects) in my scene, but assignment 1–4 feedback helped me choose the ideal shapes. I discovered that pyramids would more accurately depict the conical tree shapes and that shape selection must reflect geometric precision. The final scenario was more realistic and consistent thanks to this early modification.

I concentrated on making trees as complex objects first, and this milestone validated my choice to use pyramids for the trees. I used stacked pyramids and a cylinder for the trunk to create a custom function called RenderTree(). This design made the tree modular and reusable, which was helpful when adding other trees to the scene at various locations. My RenderScene() function remained clear and concise because this function encapsulates essential OpenGL functions like SetTransformations() and texture assignments.

I was informed that my orthographic views were not implemented appropriately after submitting Milestone Three. As a result, I gained a better understanding of how to set up g\_pCamera and utilize keyboard keys for projection switching. I was able to successfully construct front, side, and top orthographic views as well as a return to perspective view by going back and reviewing ProcessKeyboardEvents() in ViewManager.cpp. In the end, this enabled me to fine-tune object locations and proportions more precisely by enhancing my ability to examine scene layout and identify spatial flaws.

Adding textures was a fun step in the process. I had trouble at first getting the texture scale values right, but I soon discovered that it was just a matter of careful SetTextureUVScale() tweaks and trial and error. The DefineObjectMaterials() function defined and labeled my materials for reuse. This featured specially made cement, water, leaves, and wood. I discovered that, particularly in dynamic lighting, even little adjustments to ambient strength and shininess can have a significant impact on realism.

The lighting significantly improved the realistic appearance of my scene. In SetupSceneLights(), I put up two light sources to mimic a sunset: one that softens shadows by acting as an ambient fill and the other that acts as the primary light source that produces intense specular highlights on the pool and floatie. To create a soft, late-day mood, I paired these with warm orange hues and changed the focal length. Textures lacked dimensionality without lighting and material shaders; each object acquired depth and context after lighting was added.

Using modular code to include a table with a glass of orange juice was one of my last enhancements to the scene. A transparent cylinder encircling an opaque orange one was rendered to form the glass. To make the glass translucent and reflective, I applied alpha blending and material settings. SetShaderColor() and SetShaderMaterial() worked together to give the object a distinctive appearance. My code was legible and simple to debug since I used modular procedures like RenderTree(), RenderPool(), and RenderTableAndCup(). Every function demonstrated a consistent, well-organized structure by reusing the same transformation pipeline.

Using the mouse drag for rotation and the keys W, A, S, D, Q, and E for movement, the navigation controls I put in place enable complete movement around the 3D environment. The ViewManager.cpp functions ProcessKeyboardEvents() and Mouse\_Position\_Callback(), which interact with a Camera object to change view and projection matrices, handle this. In order to provide a reliable and user-friendly method of exploring the scene, I also added mouse scroll to adjust speed and projection toggling with number keys (1–4).