Nicholas Wyrwas

CS-330-13648-M01 Comp Graphic and Visualization 2025

Mr. Brian Battersby

03/01/2025

Final Project Reflection

When it came to justifying the choices, I made for my 3D scene. I wanted to ensure that the objects I chose represented the 3D objects as closely as possible from the image that I submitted earlier in the term. For example, with the desk I would use a box shaped mesh but elongate it to represent a desk. I used multiple cylinders for creating the water bottle, phone holder, and even the mouse. I programed for functionality by creating multiple different functions for each object within the SceneManager.cpp file. So that when I call the RenderScene() method it calls those functions individually. As well as, it makes the code more clean and easier to troubleshoot if an error ever arises. The SetupSceneLights function configures multiple light sources to light up a 3D scene through a shader manager. Directional light replicates sunlight by employing distinct ambient diffuse and specular characteristics. The scene features three point lights positioned strategically to deliver localized illumination since each light source possesses unique intensity and color settings. The backdrop receives illumination from one of the point lights in the scene. A spotlight is positioned above the scene’s center which directs its beam downward and features cutoff angles and attenuation to create directed light effects. The ShaderManager handles the transfer of light properties to shaders to create custom lighting effects that improve the scene's realism and visual depth. The function marks every light source as active to ensure proper rendering. Custom functions enhance code modularity and organization through task encapsulation while increasing reusability and maintainability. The SetTransformations() function consolidates the transformation logic which applies scale, rotation and position to meshes ensuring consistent application of these transformations while preventing code duplication. The SetShaderTexture(), SetShaderMaterial(), and SetShaderColor() functions enable developers to apply textures, materials, and colors to various objects while avoiding code duplication for each object's appearance changes. SetTextureUVScale() controls texture mapping on meshes while drawing functions such as m\_basicMeshes->DrawBoxMesh() and m\_basicMeshes->DrawCylinderMesh() manage specific shapes which enables mesh reuse throughout the scene. By centralizing repeated logic these functions make code management and expansion simpler as the project develops.