Development Choices and Implementation

The 3D scene I developed represents a modern desktop computer setup, featuring dual monitors, a speaker, and a keyboard. This choice was driven by both simplicity but also because the props are so commonly found in video game graphics.

Development Choices:

The scenes composition was create to give the user multiple angles and positioning of geometry no matter what way they rotate or move the camera. The use of relevant textures allows you to get a clear idea of the concepts being represented in the 3D space.

The lighting system features both ambient and directional lighting. This was a bit confusing to get figured out but once it was working it was very satisfying to see the lighting be correctly applied to my materials. There is a white-ish ambient light and two flanking directional lights, one red, one blue.

The scene navigation has the given keyboard inputs with the addition of Q and E I’ve added to give more control over the camera, this helped during development but is still cool to use on the final result.

Code Organization

The program implements several custom functions that enhance code reusability and organization:

SetupLighting():  
This function encapsulates all lighting-related configurations, making it easy to modify light properties and add new light sources. It handles both directional and point lights, setting their positions, colors, and intensity values.

DefineMaterials():  
Creates and manages material definitions for different object types (matte, metal, plastic). Each material defines properties like:

* Diffuse color
* Specular color
* Shininess factor

These materials can be reused across different objects, ensuring consistent appearance and simplified maintenance. In my case I ended up liking the matte material on most objects with the lighting I had.

LoadSceneTextures():  
Manages texture loading and configuration, supporting up to 16 different textures per scene. It handles:

* Texture file loading
* Texture parameter configuration
* Texture slot management
* UV coordinate mapping

This structured approach makes the code maintainable and allows for easy scene modifications or expansions. The reusable functions reduce code duplication and provide consistent behavior across different objects in the scene.