**CS 330 Final Project: Design Decisions Document**

Nurlan Gasimzade

**Development Choices**

For my final project, I chose to replicate a 2D image of a table scene with two lamps, stacked books, a decorative jar, and a window. My primary goal was to use basic geometric shapes to closely approximate the real-world layout, while also applying materials, lighting, and textures to enhance realism.  
  
I built the two lamps using combinations of box, cone, sphere, and cylinder primitives. Each lamp includes a base, curved lower body, slender upper body, and a shade. These sections used multiple transformations to scale, rotate, and place each part precisely. The lamp design was a strong example of object construction using modular reusable shape logic. For example, the finial on top of each lamp reused a scaled sphere.  
  
The stacked books were constructed from textured box meshes and rotated slightly to add realism. The decorative jar involved a mix of cylinder and sphere primitives stacked and scaled appropriately to resemble a ginger jar. I positioned all objects based on careful visual comparison to the source image and fine-tuned their placement using X, Y, and Z coordinates.  
  
To match the realism of the source image, I applied textures to key objects. I used a high-resolution “wooden” texture for the lamp necks, a vase texture for the jar, a “book cover” texture for the books, and a table texture. These textures were mapped using SetShaderTexture() and properly scaled with SetTextureUVScale() to avoid stretching or blurring.  
  
Lighting setup included two point lights and a directional light. One point light was colored reddish to meet the rubric requirement. All lights were configured with ambient, diffuse, and specular values following the Phong model. This helped create shadows and highlights that changed dynamically with the camera's position.

**Navigation Implementation**

Navigation in the 3D scene allows full movement and orientation adjustment. Using WASD keys, the user can move forward, backward, and strafe left or right. The Q and E keys move the camera vertically up and down. Mouse movement allows users to look around freely, adjusting the yaw and pitch of the camera. Additionally, the mouse scroll controls zoom level (FOV), giving users control over movement speed.  
  
To allow different perspectives of the scene, I implemented a projection switch using keyboard keys. Pressing P enables perspective view, while O enables orthographic view. The camera’s position and orientation stay constant during this switch, giving users a choice of visual representation while maintaining usability.

**Modular Functions and Organization**

To keep the code organized and maintainable, I modularized tasks into custom functions. These include:  
- SetTransformations(): Combines scale, rotation, and translation into a model matrix for object placement.  
- SetShaderTexture() and SetShaderMaterial(): Apply specific textures or materials before rendering each mesh.  
- DefineObjectMaterials(): Centralizes material definitions using the Phong shading model.  
- LoadSceneTextures(): Loads and binds all textures once, improving performance and maintainability.  
- SetupSceneLights(): Encapsulates all light configuration logic.  
  
These functions made the rendering logic easier to read and reduced repetition. By calling the same transformation and material functions for all objects, I ensured consistent behavior and simplified updates throughout development.