Laynie Tierney

CS-330-11374-M01 Comp Graphic and Visualization

22 June 2024

**7-1 Final Project Reflection: Design Decisions for the 3D Scene**

**Object Selection:**

The selection of objects in the 3D scene was based on creating a realistic and cohesive environment resembling a desk setup. Objects such as a plane, laptop, desk lamp, mug, and books were chosen to simulate a typical workspace. Each object was carefully selected to demonstrate different shapes (plane, box, cylinder, cone, sphere, torus) and materials (wood, glass/metal, grape/plastic texture) to showcase diversity and realism in the scene.

**Development Approach:**

During development, several approaches were considered for implementing the scene:

* **Object Loading and Placement:** Objects were loaded using basic mesh types (plane, box, cylinder, etc.) from a library to streamline rendering and ensure performance efficiency.
* **Texture and Material Application:** Textures and materials were applied using a shader manager, allowing for dynamic adjustments in lighting and surface appearance during runtime.
* **Lighting Setup:** Multiple light sources were configured using shader parameters to enhance realism and create a visually appealing scene with shadows and highlights.

**Navigation in the 3D Scene:**

Users can navigate the 3D scene using both keyboard and mouse inputs:

* **Keyboard Controls:** Users can use keys like W, S, A, D, Q, E to zoom, pan, and rotate the camera view within the scene. Different views (front, side, top, perspective) can also be toggled using numeric keys (1, 2, 3, 4).
* **Mouse Interaction:** Mouse movement allows users to interact with objects in the scene, adjusting the camera perspective based on cursor position changes.

**Camera Setup:**

The virtual camera setup was designed to offer flexibility and control over the scene's viewpoint:

* **Orthographic vs. Perspective Views:** The scene supports both orthographic and perspective projections, toggled based on user input. Orthographic views provide a flat representation useful for technical details, while perspective views offer realistic depth perception.
* **Dynamic Projection Matrix:** The projection matrix adjusts based on the selected view mode (orthographic or perspective), aspect ratio, and field of view (FOV), ensuring accurate rendering across different displays.

**Modular and Custom Functions:**

To maintain code modularity and organization, custom functions were implemented:

* **Shader Management:** A ShaderManager class centralized shader loading, compilation, and uniform setting, promoting code reusability across different objects and scenes.
* **Mesh Loading:** Basic mesh types (plane, box, cylinder, etc.) were encapsulated in a BasicMeshes class, simplifying object creation and ensuring efficient memory usage by sharing mesh data across multiple instances.
* **Transformation and Rendering:** Functions like SetTransformations and RenderScene in the SceneManager class were modularized to handle object transformations (scale, rotation, translation) and scene rendering, respectively, enhancing code clarity and maintainability.

**Conclusion:**

The design and implementation of the 3D scene were driven by considerations of realism, functionality, and user interaction. By carefully selecting objects, implementing versatile camera controls, and organizing code into modular functions, the scene not only meets visual expectations but also ensures scalability and ease of maintenance. These choices collectively contribute to a robust and immersive 3D environment suitable for various interactive applications.

References:  
*8.1 - Introduction to projections — LearnWebGL*. (n.d.). <https://learnwebgl.brown37.net/08_projections/projections_introduction.html#:~:text=An%20orthographic%20projection%20maintains%20parallel,are%20skewed%20toward%20vanishing%20points>.