7-1 Final Project

Duane Wegner

Southern New Hampshire University

CS-330 Comp Graphic and Visualization

**Brian Battersby (M.S.), Computer Science/STEM**

February 20, 2025

**Final Project Reflection**

**Proposed Desk Scene**

A computer on a desk

Description automatically generated(SNHU, 2025)

**Final Project Rendered Scene**

A computer on a marble desk

AI-generated content may be incorrect. A computer screen with a logo on it

AI-generated content may be incorrect.

**Development choices for my 3D scene**.

For my 3D scene, I made deliberate choices with the objects to highlight the contrast between simple and complex shapes, demonstrating how they can be combined to form realistic representations in a 3D environment.

* Cup: The cup was created using a tapered cylinder for the body, which mimics the structure of an actual cup, providing the right proportions and smooth curvature. The rim is modeled with a torus, which naturally matches the round shape of the cup's opening. For the handle, I used a half-torus, which allowed me to easily create the curved, ergonomic shape that fits a human hand.
* Monitor: The monitor consists of multiple boxes to represent different parts of the device, such as the base, screen, and stand. This choice was made because the box shape allows for easy scaling and texturing, which I used to add realism. By applying textures that simulate parts of the monitor, I was able to give the object a more lifelike appearance while keeping the model simple and easy to manipulate.
* Mouse: The mouse was made by combining a half-torus for the top, where the hand contacts, and a cylinder for the main body. The half-torus shape suited the ergonomic curve of a typical mouse, and the cylinder provided a smooth, consistent body shape. This combination allowed for a clean and recognizable design, reflecting the user interaction with the object.
* Keyboard: The keyboard was built using several boxes to represent the individual keys. I used overlays for the key labels, particularly for the Apple logo, which adds another layer of realism to the model. The choice of boxes for the keys is a practical one, as it allows me to easily align and position the keys while maintaining the basic grid structure of a standard keyboard.
* Lights, materials, and textures: To enhance the realism of the 3D scene, I incorporated lighting, textures, and materials to give depth and visual appeal to the objects. The lighting setup included ambient, directional, and point lights to simulate real-world lighting conditions, ensuring that the objects were properly illuminated from different angles and distances. This made the scene feel more dynamic and natural. Textures were applied to the surfaces of the objects, such as the monitor and keyboard, to replicate real-world details like screen surfaces and keys, adding visual complexity. Additionally, materials were assigned to objects, such as adjusting the shininess or roughness of the cup and keyboard, to give them a more lifelike appearance. The combination of realistic lighting, detailed textures, and well-defined materials helped to create a more immersive 3D environment, enhancing the user’s experience by making the objects feel tangible and true to life.

The combination of basic geometric shapes and texture mapping helped in achieving the desired level of detail and realism without overly complicating the models. I was able to focus on functionality by using textures for finer details and maintaining modularity in the structure, making it easy to modify or update parts of the scene as needed.

**Navigation of the 3D scene**.

In the 3D scene, the user can navigate using both the mouse and the keyboard. The mouse control allows the user to move the camera's viewpoint by detecting the relative movement of the cursor. This is accomplished by capturing mouse position changes and calculating the offset in both the x and y directions. The camera's orientation is adjusted based on these offsets, allowing the user to "look around" the scene. The movement is smooth and continuous, making it feel like a natural exploration of the virtual environment. Additionally, the first mouse movement event is used to initialize the mouse position, ensuring subsequent movements are calculated relative to this initial point.

For keyboard-based navigation, various keys are mapped to specific camera actions. The 'W' and 'S' keys control the camera's forward and backward movement, while the 'A' and 'D' keys allow for panning left and right. The 'Q' and 'E' keys provide vertical movement, enabling the camera to move up and down. The user can also switch between different predefined camera views using the 'O' key for an orthographic projection, cycling through front, side, and top views. Additionally, the 'P' key toggles to a perspective projection, offering flexibility in how the scene is viewed.

The mouse scroll in the 3D scene is used to adjust the camera's movement speed, providing a more intuitive way for users to navigate the environment. When the user scrolls the mouse wheel, the camera's movement speed either increases or decreases based on the direction of the scroll. A forward scroll, scrolling up, increases the movement speed, allowing the user to move faster through the scene, while a backward scroll, scrolling down, decreases the speed, allowing for more precise and controlled movements. This functionality allows for dynamic camera control, where the user can adjust their speed according to the needs of exploring or interacting with the scene.

These controls are integrated within the ViewManager class, which manages the camera's position, orientation, and movement based on user inputs. The system ensures that the camera behaves intuitively, providing a seamless and immersive navigation experience within the 3D environment. The use of both the mouse and keyboard provides multiple ways to interact with the scene, enhancing the overall user experience.

**Use of custom functions to promote modular and organized code.**

Custom functions significantly enhance modularity and organization by breaking down complex tasks into smaller, reusable pieces. For example, the createMaterial function simplifies the process of creating materials by setting properties like diffuseColor, specularColor, shininess, and tag. This function takes these parameters and returns a fully constructed material, which makes it versatile and reusable across different parts of the program. By using this function, you can easily create various materials, like plastic, metal, or wood, without duplicating the initialization code.

Another important function is addMaterialToList, which centralizes the process of adding materials to the m\_objectMaterials list. By abstracting away the direct interaction with the list, it keeps the code cleaner and easier to maintain. Whenever material needs to be added, you simply call this function, ensuring that all materials are handled consistently. Additionally, the setMaterialProperties function allows modifying the properties of an existing material, such as its color and shininess. This function makes it easy to update material properties dynamically during runtime, whether in response to user input or other changes, without directly modifying the material's properties in multiple places.

These custom functions promote code reuse, reduce duplication, and keep the program organized by isolating specific tasks into well-defined, single-responsibility functions. They also make the code more maintainable and adaptable for future changes, as any updates to the logic or functionality can be made in a single location. This modular approach not only improves the readability of the code but also allows for greater flexibility when expanding or modifying the program in the future.

**Reference**

SNHU. (2025, January 12). Final Project Review Guidelines and Rubric. Southern New Hampshire University. Retrieved from <https://learn.snhu.edu/content/enforced/1798879-CS-330-18454.202511-1/CS%20330%20Module%20One%20Project%20Review%20Image%205_20240118185604489.jpg?ou=1798879>