7-1 Final Project

Francisco Sousa

SNHU

CS 330

Eugenio Rodriguez

08/17/2025

## **Development Choices**

For my final project, I chose to replicate a castle-style scene based on my reference image. I broke the scene into simple primitive shapes to create an efficient, low-polygon 3D representation. The towers were built from cylinders with cone roofs and an added cylinder battlement ring. The garden topiary objects combined spheres with a cylinder trunk, which satisfied the requirement of creating an object from two or more primitive shapes. Additional details such as lamps were modeled from cylinders and spheres, while the ground was a large plane. This ensured that at least four different primitives (plane, cylinder, cone, sphere, box) appeared in my scene.

All geometry was kept under 1,000 triangles per object to ensure low polygon counts, organized mesh layout, and efficient rendering. Transformation functions were applied consistently to control scaling, rotation, and translation, which follows best practices for 3D rendering pipelines (LearnOpenGL, n.d.-a).

## **Textures**

I applied accurately projected textures to multiple objects. Examples include:

* **Ground plane:** stone pavers texture, UV-scaled to tile evenly.
* **Tower bases:** brick wall texture projected around the cylinder.
* **Roof cones:** a dark roof tile texture.
* **Garden topiary:** green foliage texture mapped to the spheres.
* **Door:** wood plank texture applied to a vertical plane.

Textures were implemented using OpenGL texture wrapping and filtering techniques to ensure realism and avoid stretching (LearnOpenGL, n.d.-b). These were sourced with relative paths (textures/...) to maintain portability, following best practices for cross-platform file handling (Khronos Group, n.d.).

## **Lighting**

I implemented two light sources to create a polished scene:

1. A **directional light** to replicate sunlight, with ambient, diffuse, and specular values configured to provide warm overall illumination.
2. A **point light** with a colored tint placed above the scene to mimic lantern glow.

Both lights used the Phong lighting model, incorporating ambient, diffuse, and specular components for realism (LearnOpenGL, n.d.-c; Shreiner et al., 2013). Positioning avoided dark shadow artifacts while still providing depth and realism.

## **Navigation**

The scene supports full camera navigation:

* **WASD keys** move forward, backward, left, and right.
* **Q/E keys** control vertical movement.
* The **mouse cursor** changes orientation.
* **Mouse scroll** adjusts movement speed.

These controls were coded based on the standard OpenGL camera implementation, which involves updating the view matrix with pitch, yaw, and position changes (LearnOpenGL, n.d.-d). A toggle function allows switching between perspective and orthographic projections, keeping the camera orientation intact while updating the projection matrix (LearnOpenGL, n.d.-e).

## **Custom Functions**

To maintain modularity and clarity, I created custom functions for repeated tasks:

* **RenderTower()** builds a tower from cylinder, cone, and battlement ring components.
* **RenderMainCastle()** assembles the central and side towers, along with the front door.
* **RenderFrontGarden()** draws topiary spheres and spiral shapes using a helper function.
* **RenderLamps()** draws lamp posts using a cylinder pole and sphere globe.
* **SetTransformations()** centralizes scaling, rotation, and translation of meshes.

This modular structure improved readability, maintained clean logic, and followed industry best practices for structured graphics code (Shreiner et al., 2013).

## **Reflection**

My development choices were guided by efficiency, accuracy, and best practices. I used low-poly primitives for performance, accurate UV-mapped textures for realism, and Phong lighting to highlight depth and detail. Navigation controls were carefully tuned to make the scene interactive and user-friendly. By following modular programming and graphics principles, I created a scene that balances technical requirements with creative design.

## **References**

Khronos Group. (n.d.). *OpenGL registry*. Retrieved from [https://registry.](https://registry.khronos.org/OpenGL/)khronos.org/OpenGL/

LearnOpenGL. (n.d.-a). *Transformations*. Retrieved from <https://learnopengl.com/Getting-started/Transformations>

LearnOpenGL. (n.d.-b). *Textures*. Retrieved from <https://learnopengl.com/Getting-started/Textures>

LearnOpenGL. (n.d.-c). *Lighting: Colors*. Retrieved from <https://learnopengl.com/Lighting/Colors>

LearnOpenGL. (n.d.-d). *Camera*. Retrieved from <https://learnopengl.com/Getting-started/Camera>

LearnOpenGL. (n.d.-e). *Coordinate systems*. Retrieved from <https://learnopengl.com/Getting-started/Coordinate-Systems>

Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). *OpenGL programming guide: The official guide to learning OpenGL, version 4.3* (8th ed.). Addison-Wesley.