Design Decisions – 7-1 Final Project

\*\*Name:\*\* Marissa Lanza  
\*\*Course:\*\* CS-330 Computational Graphics and Visualization  
\*\*Date:\*\* April 14, 2025

# 1. Design Choices and Object Creation

For my final 3D scene, I recreated a chocolate dessert object based on a 2D reference image. I used a combination of low-polygon primitives, including a box, cone, cylinder, plane, and a tapered cylinder. The scene includes:  
- A textured ground plane using a chocolate bar image  
- A vertical chocolate wall  
- A white base box to simulate a dessert plate  
- A tapered cylinder ring to mimic frosting  
- A marble-textured cylinder topped with a decorative cone  
  
The central object was built using multiple primitives stacked and scaled to simulate layers and toppings. I kept polygon counts low by sticking with basic meshes and used scaling, positioning, and texture UV mapping to refine their appearance.

# 2. Lighting Implementation

I used two point lights:  
- One warm key light on the right  
- One cool fill light from the left (bluish tone)  
  
Both were activated with ambient, diffuse, and specular values. The shader uses Phong shading, and the fragment shader logic supports toggling lighting and texture use independently.  
  
I also activated a directional light for overall soft ambient effect.

# 3. Texture Projection

Two textures were used:  
- A 1024x1024 image of chocolate for the wall, floor, and white base  
- A pink marble texture for the top of the dessert  
  
Textures were scaled using SetTextureUVScale() and toggled with bUseTexture. The bUseLighting flag was enabled on certain objects to show light reflection properly.

# 4. Camera Navigation and Controls

I implemented a navigable camera that moves with:  
- WASD for movement on the XZ plane  
- Q and E for vertical Y-axis movement  
- Mouse for pitch and yaw  
- Scroll wheel for speed adjustment  
  
I also added a keybind to toggle between perspective and orthographic projection, maintaining the same camera orientation.

# 5. Custom Code and Modularity

I created modular helper functions:  
- SetTransformations() – scales, rotates, and positions objects  
- SetShaderTexture() – binds texture ID by name  
- SetTextureUVScale() – adjusts tiling  
- m\_pShaderManager->setVec3Value() and similar methods for cleaner shader input passing  
  
These methods make the code cleaner and easier to expand in future projects.

# 6. Conclusion

This project taught me how to:  
- Apply lighting and shading techniques with GLSL  
- Work with textures and UV mapping  
- Control camera navigation with precision  
- Think like a graphics developer for client-based rendering needs  
  
I'm proud of the final scene and how it reflects both technical requirements and creative expression.