**Christopher Lajoie**

**CS-330**

**Design Decisions for Final 3D Project Report**

**Development Choices**

The 3D scene I developed for this project recreates the 2D reference image shown here

A plate of food with a slice of cake and whipped cream

AI-generated content may be incorrect.

The image depicts a single slice of carrot cake with some nearby details. I decided to build up each object using as few polygons as possible, representing them with simple primitives such as prisms, cylinders, and spheres. This approach results in objects that are easily recognizable, yet light to render because they are comprised of triangles instead of more complex polygons. For instance, I created the basic shape of the cake slice by using three triangular prisms stacked on top of one another and then adding layers of thin triangular prisms in between to represent the frosting, I also added a thin rectangular frosting piece on the back side. The whipped cream was built from scaled spheres, and the decorations such as blueberries and strawberries were spheres. Combining these basic shapes allowed me to create a scene that directly resembled the provided image without overcomplicating the models.

I used AI generated high resolution textures that I created in third party tools for each of the objects in the scene. This allowed me to have complete control over the visual assets and ensure that they were also free of licensing issues. In the shader I then scaled and aligned each texture map so it was repeated without visible stretching across the object’s surface. This gives the models more realism when rendered. I set the cake texture to tile in order to repeat the crumb pattern for a consistent appearance, but for the fruit textures I had to use additional scaling and rotation to make the fruits look natural on their curved surfaces. Lighting provides the final touches in the scene. I used a neutral point light, a colored accent light, and a soft fill, then tuned ambient, diffuse, and specular to bring out the glossy fruit and soft frosting.

**User Navigation**

Navigation is intuitive and allows free movement in all directions. The WASD keys are used to move forward, backward, and strafe side to side along the horizontal plane, while the Q and E keys move the user’s viewpoint up and down in the vertical direction. These controls make it possible to explore the scene in three dimensions. I also added mouse support for a better user experience, with the cursor controlling the user’s pitch and yaw so they can smoothly rotate their view in any direction, and the scroll wheel to change movement speed. This allows the user to quickly switch between broad exploration and more focused, slower movement to examine details such as textures or lighting effects more closely.

Another feature I implemented is the ability to switch between perspective and orthographic projection using keyboard input. Perspective mode provides a realistic camera view, creating a natural sense of depth and three dimensionality in the scene. Orthographic mode flattens the view by projecting objects orthogonally to the screen, removing perspective distortion. This can be helpful for verifying layout and proportion without scale distortion. This feature gives users multiple ways to experience and visualize the same 3D scene, providing flexibility for visualization or evaluation purposes.

**Custom Functions and Modularity**

In the program I created custom functions to help keep the code modular and avoid repetition. A SetTransformations function can apply scaling, rotation, and translation to an object in a single call. This makes it easy to place multiple shapes in the same position or configuration consistently while also saving space in the code. Similarly, I created functions to set shader textures (SetShaderTexture) and materials (SetShaderMaterial) which can be applied to many different objects with a single call, rather than reentering the same information repeatedly.

I also modularized the lighting setup into a SetupSceneLights function, which initializes and positions all lights in the scene. This makes it easy to add or remove a light by changing a single piece of code. This approach leaves more room to expand upon the scene in future iterations if I choose to add more light sources. These functions help to reduce code duplication and make it more readable, following best practices for software development.

**Reflection**

Overall, this project helped me to understand how a combination of simple primitives can be assembled to approximate real world objects. I learned how to import and use textures in shaders, including some tips on aligning them properly on angled and curved surfaces. Another useful lesson was the value of modular code. By creating my own functions I could reduce code duplication and make the project easier to read and modify. This will be helpful for me in future assignments as well, when building more complex programs.

Navigation controls were also an important part of the project. By allowing movement in all three dimensions, adding mouse driven orientation, and supporting both perspective and orthographic projection modes, I provided users with multiple options to experience the final scene. Whether the intended use case is a client review, a prototype model, or a polished project, these features make the scene more usable and engaging.

If I were to extend this project in the future, I could turn this into a more interactive application by adding animation or even data driven elements. The primitives, textures, lighting, modular code, and user navigation I practiced here would all form the basis for more advanced graphics projects in future classes and the workplace.