**Reflecting on My Journey: Creating a 3D Kitchen Mockup Scene from 2D into 3D**

As I reflect on the past few weeks, I am struck by the immense challenges and growth that I have experienced while working on my 3D kitchen mockup scene via C++ in Visual Studio 2022. This project was not merely an academic exercise; it was a deep dive into the world of computer graphics, an exploration of the intricate dance between objects, textures, lighting, and code. It tested my understanding of 3D transformations, my ability to manage complex codebases, and my perseverance in the face of seemingly insurmountable errors. The process of replicating a 2D-inspired scene in a 3D environment required careful consideration of each development choice, from object selection to the control of the virtual camera.

The foundation of any 3D scene lies in the careful selection and placement of objects, which together bring the environment to life. For my 3D kitchen mockup, I chose objects that not only served functional purposes but also contributed to the overall aesthetic of the scene. These objects included a cutting board (sphere and torus), a backsplash (plane), large mortar (half sphere), stacked bowls (tapered cylinders), oil along with vinegar bottles (cylinder, tapered cylinder, torus, and half torus), and a countertop (plane). Each object was chosen with the deliberate intent to mirror a realistic kitchen environment like the one chosen in my 2D mockup, while also allowing for the exploration of various texturing and mapping techniques.

One of the key challenges was ensuring that each object was textured correctly. This involved mapping textures like hardwood for the cutting board and damask for the backsplash. The texture mapping process required a detailed understanding of UV coordinates and how they interact with the object's geometry. For instance, the oil and vinegar bottles required tiling of the texture to create a repeating pattern. This level of detail was crucial in making the kitchen scene not only visually appealing but also realistic.

Lighting played a significant role in enhancing the realism of the scene. By strategically placing light sources, such as kitchen recess lights above morning into afternoon sun breaking through the windows from the left side. I was able to create shadows and highlights that added depth to the objects, making them appear more three-dimensional. The combination of ambient, diffuse, and specular lighting was carefully calibrated to mimic the natural lighting of a kitchen. This required a deep understanding of the Phong reflection model, which was implemented through the fragment and vertex shader programs to control the light interaction with different surfaces. (*LearnOpenGL - Textures*, n.d.)

The ***SceneManager.cpp***and ***ViewManager.cpp*** files were central to the development of this project. SceneManager was responsible for managing the rendering process, including the loading of textures, setting up shaders, and drawing objects in the scene. This file became the backbone of the project, ensuring that each object was rendered correctly with the appropriate textures and lighting effects. ViewManager, on the other hand, handled the interaction between the user and the scene, including camera control and event processing. The development of these files was an iterative process, requiring constant refinement to overcome errors and achieve the desired functionality. (*LearnOpenGL,* n.d.)

The ability for users to navigate the 3D scene was a critical aspect of the project. To achieve intuitive and responsive navigation, I implemented a virtual camera controlled through various input devices, including the keyboard, mouse, and scroll wheel. The camera's position, orientation, and zoom level were controlled using the ***WASDQE*** keys and mouse movements, allowing users to explore the kitchen scene from different angles and perspectives. The combination of these input devices provided a comprehensive and intuitive way for users to interact with the 3D scene. The design choices behind these controls were made with the user experience in mind, ensuring that navigation felt natural and fluid.

In the ViewManager.cpp file, the ProcessKeyboardEvents() function was responsible for handling keyboard inputs. This function allowed users to move the camera forward, backward, left, right, up, and down using the ***W, S, A, D, Q,*** and ***E*** keys, respectively. Each key press triggered the corresponding movement function in the Camera class, such as “W” when activated will cause ***ProcessKeyboard(FORWARD, gDeltaTime)*** to move forward. This modular approach ensured that the camera movements were smooth and responsive, providing users with a seamless navigation experience. Mouse movements were processed using the ***Mouse\_Position\_Callback()*** function, which calculated the offset of the mouse position and adjusted the camera's orientation accordingly. This allowed users to look around the scene by simply moving the mouse, adding to the immersive experience. The mouse scroll wheel controlled the camera's zoom level, enabling users to zoom in and out of the scene with ease. This functionality was implemented through the Camera class's ***ProcessMouseScroll()*** method, which adjusted the camera's field of view based on the scroll input. (*LearnOpenGL - Camera*, n.d.)

To manage the complexity of the project and ensure that the code was modular and organized, there were several developed and/or utilized to custom functions. These functions not only streamlined the code but also made it more reusable and maintainable. One such function was ***DrawPlaneMesh(),*** which was responsible for rendering the plane mesh that served as the backsplash in the kitchen scene. This function encapsulated the entire process of setting up the vertex data, binding the texture, and drawing the mesh, making it easy to reuse whenever a plane mesh needed to be rendered. Another custom function was ***LoadSceneTexture(),*** which handled the loading of texture files and their application to objects in the scene. This function abstracted the complexities of texture loading, such as generating texture IDs then assigning them to texture names, and setting texture parameters, into a single, reusable function. This not only reduced code duplication but also made it easier to manage texture-related errors, as any issues could be traced back to a single function.

In the ***SceneManager.cpp*** file, the RenderScene() function was central to the entire rendering process. It orchestrated the drawing of all objects in the scene, applying textures and lighting effects as needed. The modular structure of ***RenderScene()*** allowed for easy adjustments and expansions, such as adding new objects or changing textures, without disrupting the overall flow of the program. These custom functions were designed with reusability in mind, allowing them to be adapted to different parts of the project as needed. They also contributed to the overall organization of the code, making it easier to read, understand, adjust, add to, and debug.

The journey of developing this 3D kitchen mockup scene in C++ was both challenging and rewarding. It required careful consideration of every development choice, from object selection and texturing to camera control and function modularity. Each decision was made to create a realistic and immersive 3D environment that users could navigate intuitively. The experience not only deepened my understanding of computer graphics but also honed my skills in C++ programming, preparing me for the challenges that lie ahead in my journey toward earning my bachelor’s degree in computer science.

**Citations**

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

*LearnOpenGL .* (n.d.). <https://learnopengl.com/>

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