Final Project Justifications

Roger Fisher 10/18/2025

Developmental Justifications

In my final 3D scene, I created a realistic tabletop set which is based on my workspace. I chose items with a balance between geometric complexity and simplicity as it gave me an opportunity to work with several primitive shapes without having excessive polygons. All the objects were designed out of different combinations of the basic meshes including boxes, cylinders, tapered cylinders, and prisms to produce familiar items in the real world.

The koala box was made of six prisms that were put together and sized to create the angled edges of the box. Layered texturing, which is a method of producing depth and slight variation of color by performing several texture passes, was also used to make it look unique. This had enabled me to replicate the printed cardboard surface as well as the grassy look of the art. The cardboard surface also has a glossy reflection and provides the box an appearance that is closer to being real without having used complicated geometry. In the shader, the layering was made with alpha blending which contributed to making the printed details look as though they were a part of the surface and not just overlaid.

Cylinders were used to model the candles with the containers having larger cylinders while the lids had smaller cylinders to give the stratified appearance of the candle wax and glass. The vitamin bottle consists of a box used as the main body, a tapered cylinder used as the opening towards the top and a smaller cylinder for the cap. In the same vein, the water bottle employs similar methods by using a cylinder body, a tapered cylinder for the neck and a small cylinder for the cap.

The soda can was constructed by piled up cylinders to recreate the metal body, top and bottom curvatures. Although I was tempted to employ tapered cylinders to provide a slight can taper, I chose not to do so since that would entail having to remap the tapered cylinder function and could distort other pieces. As an unfortunate alternative, I used scaled down cylinders to refine the shape without making the silhouette inaccurate.

In the sand art, I have used two cylinders to produce the glass frame and the inner glass piece with a box to form the base of the art piece. I also used a sand texture to replicate the sand art on the inner cylinder to make it more like the original.

Lastly, I modeled a bag of truffles which consists of a large box for the main bag and a smaller box to simulate the opening of the bag. I also inserted a few small chocolate-colored spheres to represent single truffles falling out of the bag to give motion and naturalness to the picture.

Navigation

I made the scene in such a way that the users would be able to explore it freely from different perspectives. The camera system was designed based on a first-person style movement allowing it to move through the X, Y, and Z axis and allows the view to turn naturally with the mouse.

Movement controls are based on WASD keys, forward, backward, left and right movements while the Q and E keys are used to move up and down. These controls have been managed in my ProcessKeyboardEvents() function in my ViewManager.cpp. The camera rotation features of movement consist of manipulating the yaw and the pitch by the Mouse_Position_Callback() function where the camera adjusts according to the movement of the mouse so that the user can view the real time view of the surrounding environment.

Another feature that I have introduced is the scroll wheel which dynamically changes the speed of movement of the camera. This allows one to observe the scene without straining to examine large and small details. Last, I included the faculty to switch the perspective and orthographic projection views using the P and O keys. This provides the user with an option of using a realistic 3D depth view to a more diagrammatic 2D view of the arrangement of the objects.

Custom Functions

To make my program modular and structured, I separated my logic into several custom classes and helper functions. SceneManager class comprises configuring and rendering of all the objects in 3D, ViewManager class is devoted to camera logic, and ShaderManager is devoted to uniforms update and shader communication. This division facilitated a lot of easy troubleshooting and subsequent expansion of individual systems without interfering with the rest.

I developed the modular functions in SceneManager like the DefineObjectMaterials, SetShaderMaterial and SetShaderTexture to assign materials and texture to each object when I was inside SceneManager. Instead of coding surface properties on a case-by-case basis on a per-model basis, I created a reusable materials list that contained ambient, diffuse, and specular values together with shininess and transparency. It was now possible to apply the various material profiles to many objects at a fast rate with the same call of functions. The system is even flexible enough that I will be able to reuse the same material logic in subsequent OpenGL projects or scenes without writing any rendering code.

Besides the materials system, I also added some custom shader logic in order to enable layered texturing. This gave me the opportunity to combine several texture maps, including the base color and a gloss that is reflective, right into the fragment shader. Alpha blending of these layers allowed me to recreate more challenging surfaces such as a glassy surface on the Koala box. This type of shader can be reused wholesomely. I can readily add this shader to any future

application that requires either layered textures or variable transparency without having to re-write the underlying lighting or texture processing logic.

All camera behaviors like keyboard input, mouse orientation and projection selection are handled by ViewManager class. The PrepareSceneView() method then calculates the view and projection matrices each frame and sends them to the shader and makes sure that all the lighting and shading is calculated in the right way with respect to the point of view of the user. This structure in modules enables the program to remain organized, maintainable, and scalable to other scenes in future where more sophisticated interaction might be needed or more lighting systems.