Dylan Clivio

Instructor: Sherin Aly

CS 330

Final Project

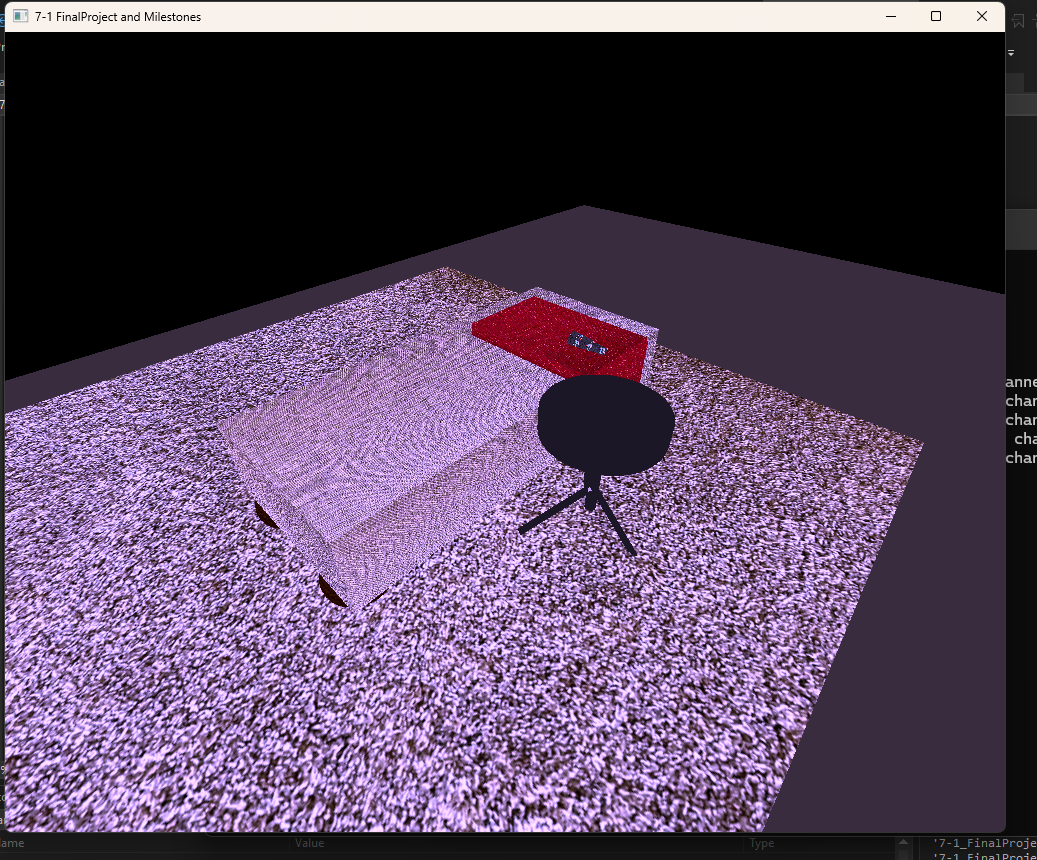
10/15/2024

Final Project Reflection

Reference Image:



Final 3D Render:



***Justify development choices for your 3D scene.*** *Think about why you chose your selected objects. Also consider how you were able to program for the required functionality.*

I chose the objects to render because they seemed to offer a lot of opportunity to familiarize myself with 3D rendering a diverse range of shapes. The footrest, remote, blanket, and stool offer completely different challenges when it comes to 3D rendering. The footrest is large and square with round and stubby wooden legs. The stool on the other hand, has thin and ridged lines making up its frame, and a round flat top making its seat. The remote and blanket made for additional complements to the scene by representing a more organic “lived-in” feel.

The footrest is primarily a large box. I added a box mesh to the SceneManager and then took a photo of the texture of the footrest in my house. I then used lightroom to convert the photo into a 1024x1024 jpg and placed it in the “Textures” folder. I was then able to apply the real texture to my 3D render. I did the same thing with the carpet in my 3D render. The legs of the footrest are very short and rounded off. I used Toruses to make them. I added the toruses and layed them flat on the plane. I then put a wood texture on them that I found online.

The blanket is made out of two boxes. I rendered one on top of the footrest and another hanging down the side. The remote is a tapered cylinder and I added a real image of a remote as the texture.

The stool is primarily made out of stretched boxes for the legs and frame. The seat is a short and wide cylinder and there are two screws represented with small grey cones on the frame.

I found the lighting portion of this course particularly challenging. I have two lights. One is the primary light that provides bright ambient lighting from above to completely light up the scene. The other light I added was supposed to be a spotlight but was challenging to implement. Mostly, all it does is add a tint of blue to the image.

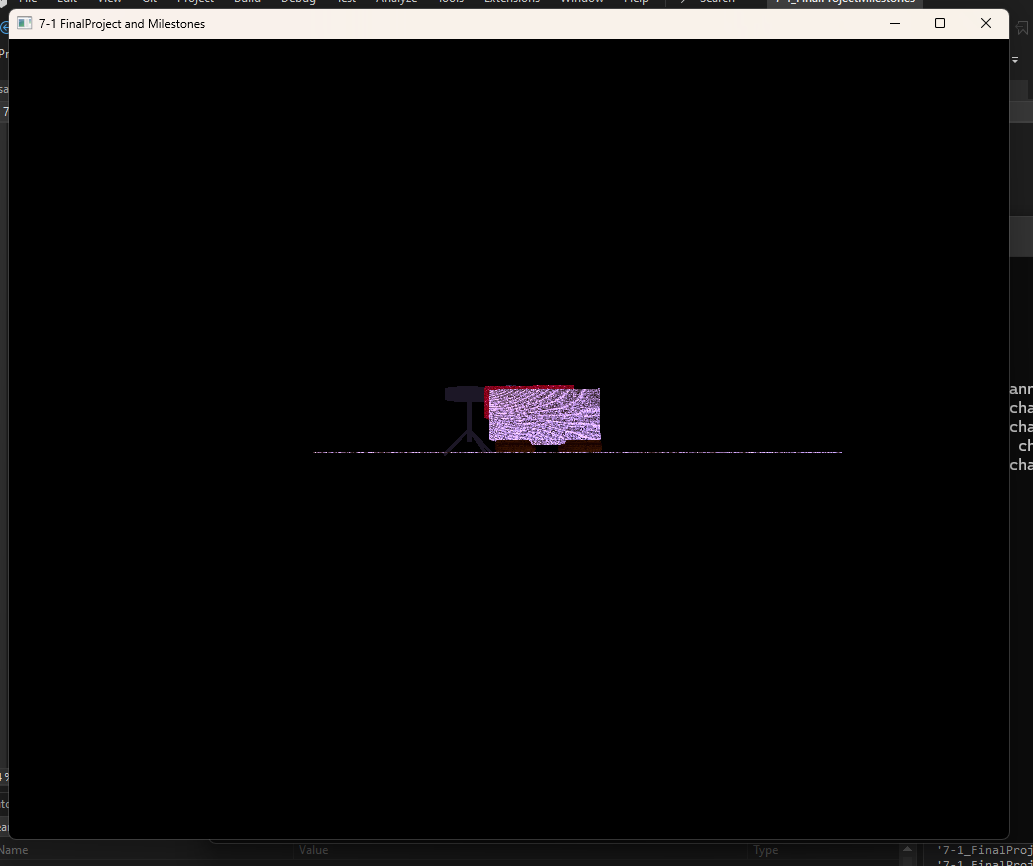
***Explain how a user can navigate your 3D scene****. Explain how you set up to control the virtual camera for your 3D scene using different input devices.*

The virtual camera can be controlled by the W,A,S,D,Q,E,O, and P keys.

The program registers the keys W, A, S, D in order to pan the camera around the scene. The speed of this panning can be altered by using the mouse wheel. The user can also use the Q and E keys to pan up or down.

Lastly, the user can press the “O” key for an orthographic view of the image, or “P” key for a normal perspective of the image.

Orthographic View:



***Explain the custom functions in your program that you are using to make your code more modular and organized****. Ask yourself, what does the function you developed do and how is it reusable?*

A couple things make my program more modular. For instance, adding a a texture to a mesh. I uploaded my own images for the project. All I have to do is make sure the file dimensions are correct and throw the images in the projects Textures folder. Then, when I want to use a texture, I just load up the texture in the program and save it as an easily referencable key. For instance, a texture of wood will be loaded into the program and then can be used by a mesh with the following code: SetShaderTexture("Blanket");. This makes it easy because I can use this code anytime I want to introduce a shape that utilizes that texture. I was able to take the same approach when adding object materials.

The type of material an object is made of impacts the way it reflects light. Because of this, I have multiple materials saved with various attributes like this one for wood: OBJECT\_MATERIAL woodMaterial;

woodMaterial.ambientColor = glm::vec3(0.1f, 0.1f, 0.1f);

woodMaterial.ambientStrength = 0.2f;

woodMaterial.diffuseColor = glm::vec3(0.3f, 0.3f, 0.3f);

woodMaterial.specularColor = glm::vec3(0.1f, 0.1f, 0.1f);

woodMaterial.shininess = 0.3;

woodMaterial.tag = "wood";

m\_objectMaterials.push\_back(woodMaterial);

Just like textures, I can reference this chunk of code by simply using the following when making my shape: SetShaderMaterial("wood");