Sham Hintolay
1618608

Homework 4 Report

1. Problem

In this Homework, we are tasked with recreating the following images. We need to recreate the following image using texture mapping in OpenGL.

I used the resources available at learnopengl.com to assist in this homework, specifically the Textures section.

Sham Hintolay
1618608

2. Method

In order to succeed in the assignment, we only made changes in the GetViewMatrix function within the Camera.h file, Initialized a projection matrix and set up the UV buffer and binded the texture within the main.cpp file, and filled the texture.frag and texture.vs files for the fragment shaders and vertex shaders.

In order to fill the GetViewMatrix() function, we initialized a view matrix of mat4 data type and used the glm:: LookAt function to get the matrix, then we returned it.

The projection matrix was initialized using the glm:: perspective function.

The UV buffer was set up by using the glGenBuffers and glBindBuffer functions, and the texture were binded by using glActiveTexture and glBindTexture functions.

In order to add the texture to the cube in the desired fashion, we need to set the positons in the texture.vs file, and attach the texture to the color of the vertex in the texture.frag file

3. Implementation Details

Sham Hintolay
1618608

In order to complete the GetViewMatrix() function we need to initialize our view matrix with the LookAt function by using the correct parameters. The paramters needed are the position vector, target vector, and the up vector. The Camera system that's already outlined already contains the variables we need to plug in. For the position parameter we simply plug in the camera's position, which is expressed in a vec3 datatype. The target parameter can be represented by the front relative to the cameras position, so we use position + front for target, then we simply plug in the camera's up vector for the up parameter. After doing all this we return the view matrix that was created.

Next in main.cpp we initialized the projection matrix. We used the perspective() function and filled the given parameters of field of view, aspect, near, and far, all of which are supposed to be float values. The field of view represents the width of the perspective frustum, and changing this value gives a view that zooms in and out, 45 is the default value so I plugged in 45. Aspect simply is the aspect ratio of the view, I used a default value of 800/600. Near and far are used to specify the near and far planes of the perspective, we plug in a minimal value for near and a arbitrary large number for far so that all the coordinates in between are drawn.

In order for us to set up the UV buffer, we need to use the glGenBuffers and glBindBiffer functions. We pass in the GL_ARRAY_BUFFER, and UVBO (UV buffer object) into them.

We also need to bind the textures, which are done using the glActiveTexture and glBindTexture functions. We pass in the given GL_TEXTURE0 and GL_TEXTURE_2D variables along with textureID.

Now we will examine how we successfully get the texture on to the cube. First, in the texture.vs file, in order to get the UV, to take the vertexUV inputted and invert it, or else we get a incorrect cube. We also need to take the inputted position and calculate the gl_position with respect to the model, view, and porjections matrixes. The UV outputted from the vertex shader in taken into the fragment shader, and using the texture sampler, we set the color for the vertex based on its positioning on the cube.

4. Results

This is the result :

Sham Hintolay
1618608