Osinor Igwemoh

Report for Homework 4.

Problem Description

This assignment involves textures, we are tasked with implementing texture mapping in OpenGL, the starter code already provides the uv data and the vertex data for us and we have to use our knowledge of openGL and its very helpful libraries to try to complete the model and give it the specified UV and texture loaded from the DDs file provided for us. We are also tasked with writing the code to bind the texture and the correct shader to draw the texture

Algorithm / Method/ Implementation

The first task for this assignment was getting the model to show up and this was done by modifying the shader programs so I could view the object. I updated the gl_position to be the "projection * view * model * the position vector" and set the UV variable to vertexUV so it can be passed to the fragment shader program. In the fragment shader program I used GSLS's built in texture function and passed it the arguments of the TextureSampler and the UV coordinates so It can render both the texture and the UV to the cube every time it is called. After I finished with the shader files and got the cube I had to do some binding in the main file so that the appropriate textures and uv data could be sent to the shader programs at runtime for rendering.

The first task in the main function was to setup the UV buffer. This included Generating the buffer, Then using openGL's function to bind the buffer, then I fill the buffer with the uv data by specifying the uv data array and its size to the openGL corresponding function. I also set the first two arguments of the VertexAttribPointer function to be 1 and 2 respectively and then called on the glEnableVertexAttribArray function passing it a value of 1 so it can differentiate from the vertex. This finally got me the uv data displayed on my cube and for the texture I had to bind it in the main while loop of the model. The starter code provided for us had already created the texture variable so I just passed it to the openGl's bind function and this created the output I saw that was required by the assignment.

My final result was a cube with just two sides of numbers being displayed from the DDs file and after search on the internet about this problem I discovered that the texture coordinate modification is not implemented in the source code so openGL does not read from the DDS file as it should. So because we are using the compressed textures in the dds file I had to invert the y component of my texture function in the fragment shader so as to read from the dds file correctly and display the correct image.