Graphics

Kynan Justis

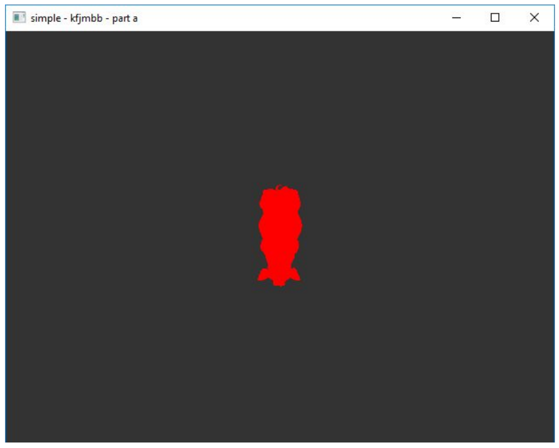
Assignment 3b

04/10/2018

Overview

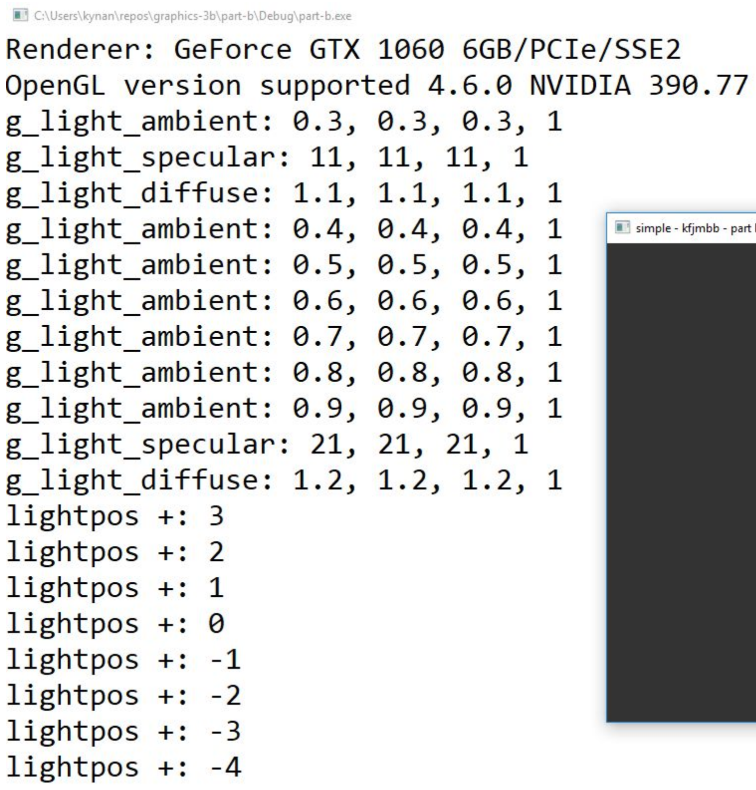
In this programming assignment, there were again three major parts. Part A involved integrating and compiling an Obj Parser library in order to facilitate loading models from a file. Once the program could successfully compile and display the pig, this section was complete. In Part B, the task was then to add a light source to the scene. From there, the lights were to be given interactive keyboard controls so they could be turned on and off. The RGB values of these light components needed to be modifiable as well. Additionally, the RGB values for the lighting materials needed to be modifiable as well using keyboard commands. Finally, in Part C, the task was to load a simple cube mesh and apply a 2D texture to it. This step also required loading and compiling an additional header file.

Part A



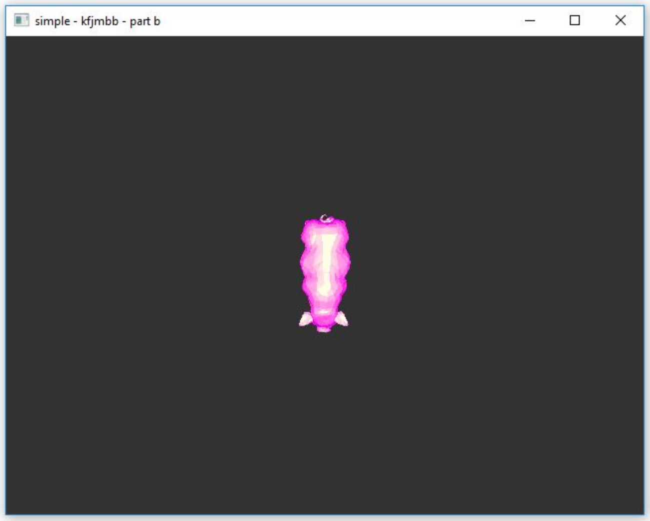
Much like the other homework assignments, this part was a simple compile-and-execute type of task. First, the Obj Parser had to be downloaded and included in the code project. Really, the only extra step was making sure the “pig.obj” file was referenced properly in the code as well as the project. Once those steps were complete, the program compiled and ran just fine.

Part B



C:\Users\kynan\repos\graphics-3b\part-b\Debug\part-b.exe

```
Renderer: GeForce GTX 1060 6GB/PCIe/SSE2
OpenGL version supported 4.6.0 NVIDIA 390.77
g_light_ambient: 0.3, 0.3, 0.3, 1
g_light_specular: 11, 11, 11, 1
g_light_diffuse: 1.1, 1.1, 1.1, 1
g_light_ambient: 0.4, 0.4, 0.4, 1
g_light_ambient: 0.5, 0.5, 0.5, 1
g_light_ambient: 0.6, 0.6, 0.6, 1
g_light_ambient: 0.7, 0.7, 0.7, 1
g_light_ambient: 0.8, 0.8, 0.8, 1
g_light_ambient: 0.9, 0.9, 0.9, 1
g_light_specular: 21, 21, 21, 1
g_light_diffuse: 1.2, 1.2, 1.2, 1
lightpos +: 3
lightpos +: 2
lightpos +: 1
lightpos +: 0
lightpos +: -1
lightpos +: -2
lightpos +: -3
lightpos +: -4
```



Part B was a lot more involved than the previous part of this assignment. In this section, the overall task was to implement a lighting model for the 3D environment. To accomplish this, a lot of code had to be added to the vertex and fragment shaders for the model. In the vertex shader, aside from the regular position and projection calculations, the ambient, diffuse, and specular reflections had to be calculated. Most of these calculations were provided from the slides. From there, the reflections were summed and passed out of the vertex shader as a color vector with an alpha value of 1.0. The fragment shader was much more simple, however, since the only change that had to be made was to feed the color calculated in the vertex shader to each respective color out value in the fragment shader. The ambient, diffuse, and specular settings for the scene had to be set in the main C++ code along with the ambient, diffuse, and specular material settings. Once all of these steps were complete, the pig object file began to render with lighting in the window.

From here, keyboard controls were added to handle changing various settings at run time. By pressing the UP and DOWN keys, the position of the lighting node can be increased and decreased along the Z-axis. To modify the RGB values of the ambient, diffuse, and specular reflections, the A, D, and S keys can be used respectively. Similarly, the RGB values of the ambient, diffuse, and specular materials can be altered using the Z, C, and X keys respectively. Finally, since so few keys were used, most of them only increased their initial starting values. To reset all of the values back to the default when the scene loads, the R key can be pressed. Debug prints are made to standard out in order to help show what modifications are happening during runtime.

Part C

Finally, Part C was probably the most difficult part of the whole assignment. My final version of the program doesn't compile, but I can still talk a little about the steps that I took to try and complete it. First, a new header file had to be added and successfully compiled in order to allow for loading the texture files. This was successfully completed at this hash value in my repository on GitHub: a1597e741c6a161dda7302c6a54414fe54fe7927. However, from this point on, things got confusing. I tried to refactor the cube project from assignment 3a since the only requirement was to texture a cube, but it was difficult to determine what properties needed to be altered in order for the texture to load properly instead of the colors. After setting up the "init" code in the C++, I thought that my VBA was setup correctly along with the separate VBOs. The VBO's consisted of arrays containing location points along with the texture data. With that portion complete, the only thing left was to write the vertex and fragment shaders to accept the texture coordinates and then interpolate the texture values across the surface of the cube. I setup the vertex shader to receive the vertex position values as an "in vec3" and a separate variable to send the texture coordinates back out. Unfortunately, I couldn't determine the use of the "in vec2 vt" variable that was included with the Q&A code, so I don't believe that I was sending data out of the vertex shader properly. The fragment shader seemed somewhat straightforward, but contained a new type of variable, the "uniform sampler2D basic_texture", which I wasn't sure how to use. I believe all that needs to be done is to set the "out vec4 frag_colour" to the same value as the color found at each given texel, but I couldn't test whether my code was successful or not. Whenever there was a problem within the shader code, the program would crash in Visual Studio without any meaningful errors. I would have needed more time to look into more advanced debugging tools for GLSL to know what was really wrong with my shader code. It's also possible that something in the C++ code just contained a logical error that wasn't caught by the debugger, but ultimately I couldn't get the texture mapping portion completed.

Note

The final version of the code, the full-color result images, and the Visual Studio build instructions can be found online at https://github.com/kfjustis/graphics-3b.