Concordia University

Report presented to

Dr. Tiberiu Popa

As a requirement for

Computer Graphics [COMP371 - W]

By

Alexandre Vallières (40157223)

Final project

Tuesday April 4th, 2023

Description

This project's main goal is to make a program that allows a user to load custom objects and shaders to observe the effects. The base code is from the lab capsule 3, which has been modified to allow for the other functionalities. This means that the same dependencies, namely GLW, GLEW and GLM, need to be installed. After implementing all the objectives, the stretch goal set was to implement a way to change the light and color properties to allow the user to visualize their object under different conditions.

Implementation

Objective 1: Create UI for user selection

Allowing the user to load files means that there is a need for a UI; this has been implemented with the help of the ImGui library, which has been installed statically into the project. The UI is separated into windows each about one aspect of the program the user can control. The user can use the scroll wheel while aiming the cursor on the desired window and holding the *Ctrl* key to resize the window. They can also drag it elsewhere on the screen. The user selection also includes the shininess of the object and one other custom variable inside the shader, which have been implemented as uniforms inside the shaders and the main code. The ImGui library renders its windows over the existing rendering and allows compatibility with multiple OS's.

Objective 2: Create a metallic shader

The metallic shader has been implemented using the Cook-Torrance algorithm using Perlin noise for the roughness calculations. Because of the specular nature of the Cook-Torrance algorithm, the shader only consists of a fragment shader which only shows the metallic effect from the specular light reflections. The custom property decides the roughness of the object by modifying its apparent granularity.

Objective 3: Create a toon shader

The toon shader consists of a customized step function that calculates the diffuse coefficient the same way Phong shading does, but splits it in different steps to distort the normal of the object and give a cartoon-like appearance. It does not use any specular reflection and also does not need any particular vertex shader to create the effect. It is important to note that this shader does not allow for the modification of a custom property, as it is a relatively simple shader with not many variables.

Objective 4: Create a balloon shader

The balloon shader uses a custom function to distort the position of the object based on its normal, which gives it a balloon-like appearance. The custom property changes the amount of distortion applied, from 0 to 0.1 times the normal of the object. While it could have been possible to use both types of shaders to create an even more realistic object, it was decided to only use a vertex shader in order to mix effect of other fragment shaders with it.

Objective 5: Allow user to load custom shaders and objects

To accomplish this objective the extended version of the Extended Native File Dialog has been used and implemented statically inside the project. This allows for native file exploring that make the program compatible with any OS. When opening the file browser with the corresponding button on the shader window, the first window opened will ask for a fragment shader that with the format **.fragment.shader.glsl* and open a second window for a vertex shader file with the format **.vertexshader.glsl* if a fragment shader has been chosen. Closing the window at any time cancels any action made. For the object selection, the file needs to be in the **.obj* format.

Bonus objective: Allow user to modify light and color properties

In order to properly investigate the effects of the implemented shaders onto the object, it was decided that moving the light and its settings would be beneficial to the program. Moreover, properties and the color of the light, as well as the object's color properties, have been implemented to allow the user to visualize their object as they want. These variables have all been implemented as uniforms in the shaders that are passed from the code.

Difficulties

There have been many difficulties throughout the project. However, the main problem was to understand how everything in the program fit together. After this was figured out, the ImGui library caused some problem as its implementation was thought to allow fixed UI components and the reorganisation of the overall rendering. However, it was simpler than that and once this misunderstanding was solved, implementing everything related to the UI was easy.

Regarding the shaders, some problem arose as to how exactly to create the proper effect. The toon shader was the easiest one to get a hold of, while the implementation of the balloon shader was somewhat slow due to how the normals of the platter are inverted in the *teapot1* ob ject that came with the initial lab code. However, the metallic shader was by far the hardest one to implement. At first, an attempt to incorporate textures was made, but that would have meant that the fragment shader would not have been compatible with other vertex shaders due to the fact that it needed customized inputs and outputs in its vertex shader.