IMAT2908 CW No 2

|  |  |
| --- | --- |
| **Full Name: Tia Mulholland** | **P\_Number: p2433472** |
| **GitHub ID: Asia292** | **Word Count: 500** |

## Abstract

The goal of this was to render a Utah teapot with ambient, diffuse and specular lighting. Attenuation was also added to create more realistic lighting as well as a spotlight implemented.

## Introduction

Most of the work is done in the fragment shader which is one of the last stages of the rendering pipeline; it processes the output from rasterization into colours.[1] This is linked with the vertex shader, which handles the vertices, in order to create the desired shape.[2]

The principle of phong shading is that it is made up of three different forms of light – diffuse, ambient and specular – which when added together create a glossy appearance. The brightness of this decreases as the angle between the view direction and reflection increases, allowing the “shininess” of a material to be altered. [3]

## Method

Originally, specular, ambient and diffuse were calculated within the main function of the fragment shader. Ambient was calculated by multiplying the reflection coefficient (K) with light intensity (L); diffuse multiplied its own K and L with the cos of the angle between the light and the normal of the plane; specular also uses its own K and L but multiplied by the dot product between the reflection and view vector to the power of the shininess value.

These values are then clamped to avoid overshooting the vector, and added together to create the phong shader. Attenuation was then calculated by using the formula and applied to the specular and diffuse values.

To make everything more organised I then moved it into structs and functions. Each element of the light (diffuse, ambient, specular and attenuation) was given its own function, with structs implemented to hold their variables.

The spotlight is calculated by taking the dot product of the distance between the position of the spotlight and that of the vertex and the direction of the spotlight. The resulting angle is then compared to the defined cut off angle to determine if that area of the model falls within the spotlight. If it does full lighting is applied, if not than only ambient. I also designed a function to allow the user to turn the spotlight on and off which is controlled with a bool.

Camera rotation was implemented by adjusting the y and x axis by the yaw and pitch accordingly and roll by altering the z axis.

## Evaluation

I achieved everything I had aimed to do; however, the extra implementation of the spotlight switch doesn’t work as intended. The if statements that declare which form of L to use (choosing between spotlight and light position to use with vertex position) is never run, therefore L is never defined. Similarly, the if statement that sets up the lighting is also never run so it is reverting to its default. This leads me to believe that there is an issue with boolean if statements within openGL that is preventing them from running. If I was to redo this I would find an alternative way to implement the switching of the spotlight.

## References

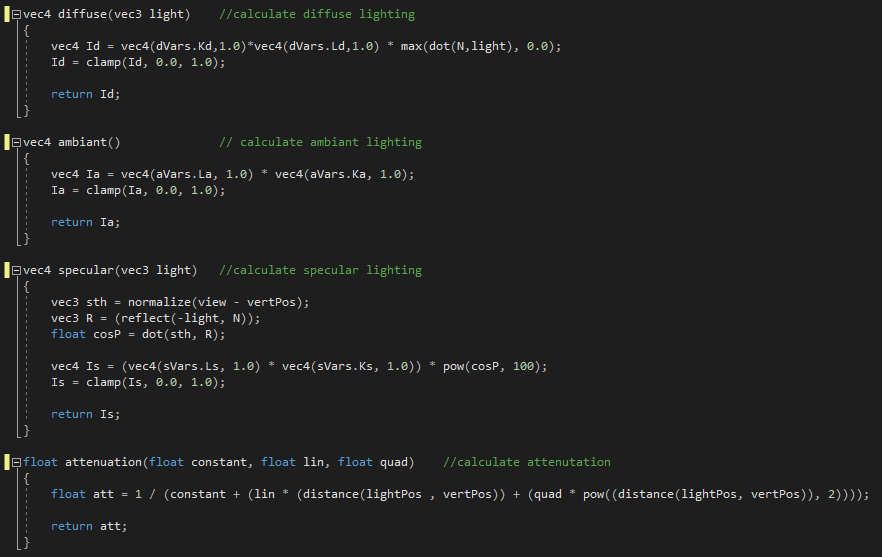
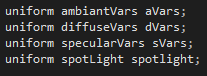
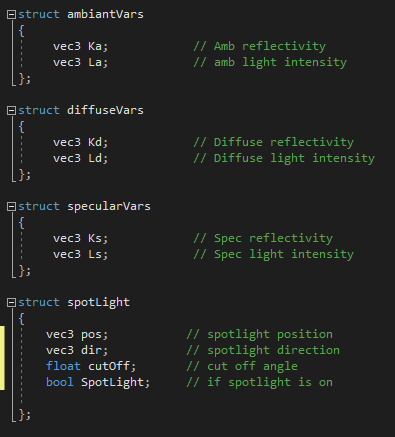
[1] OpenGL Wiki. (3 January 2018). Fragment Shader [Online]. Available: <https://www.khronos.org/opengl/wiki/Fragment_Shader>

[2] OpenGL Wiki. (10 November 2017). Vertex Shader [Online]. Available:

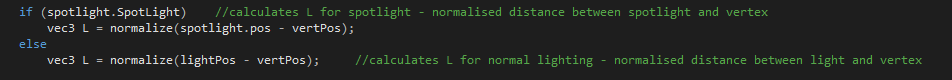
<https://www.khronos.org/opengl/wiki/Vertex_Shader>

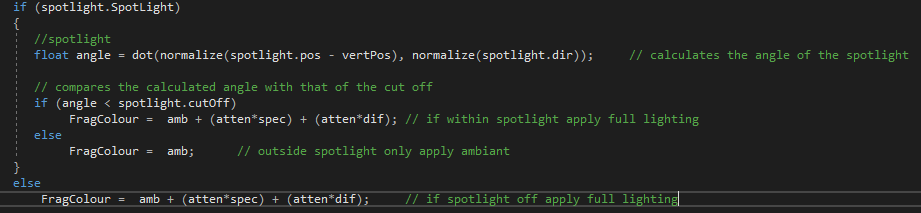
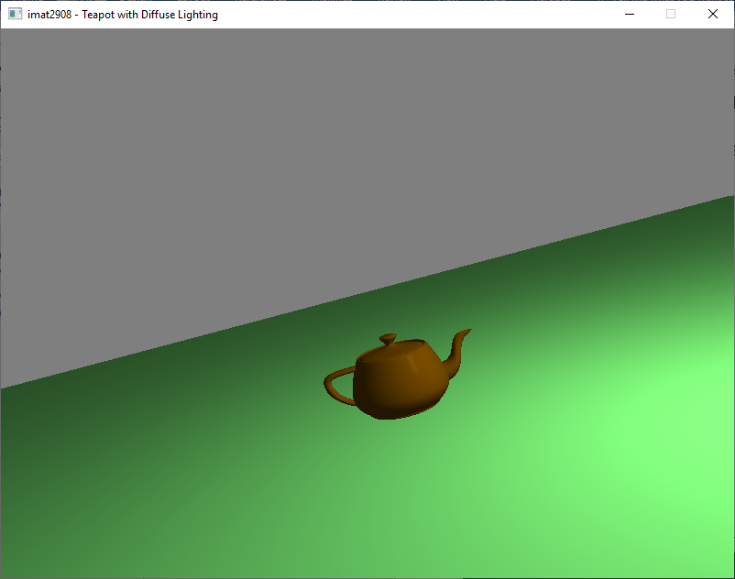
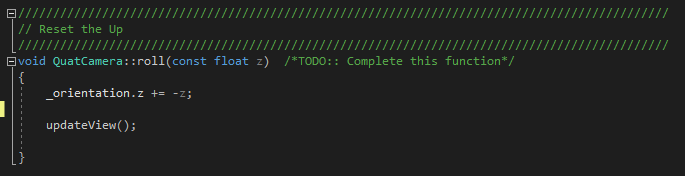
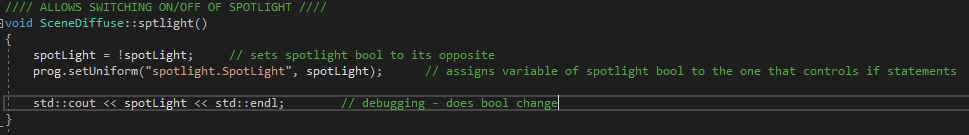
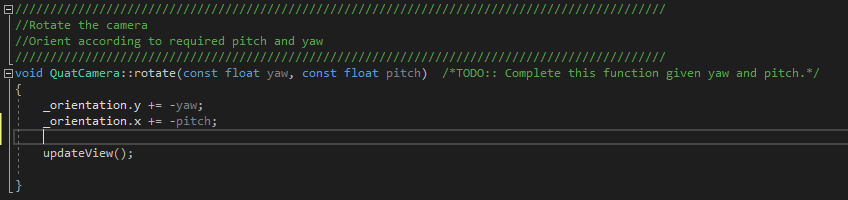
[3] Scratchapixel 2.0. (N/A). The Phong Model, Introduction to the Concepts of Shader, Reflection Models and BRDF [Online]. Available:

<https://www.scratchapixel.com/lessons/3d-basic-rendering/phong-shader-BRDF>



structs and functions for lighting

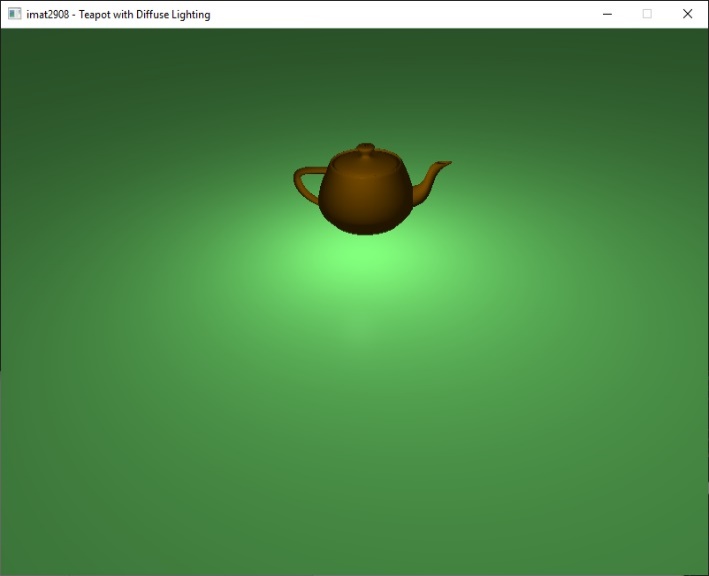




Spotlight implementation

Normal lighting implementation and camera movement

Switch between spotlight and normal light

Camera movement