IMAT2908 CW: Lighting

|  |  |
| --- | --- |
| **Full Name: Robert Quick** | **P\_Number: P2658455** |
| **GitHub Username: rbtq** | **Word Count:** |

[Introduction 3](#_Toc130405011)

[Brief History of shaders 3](#_Toc130405012)

[Pre-1980s 3](#_Toc130405013)

[1980s-2000 3](#_Toc130405014)

[Methodology 3](#_Toc130405015)

[Introduction 4](#_Toc130405016)

[phong.vert 4](#_Toc130405017)

[InputData 4](#_Toc130405018)

[ProcessData 5](#_Toc130405019)

[LightData 5](#_Toc130405020)

[Functions 5](#_Toc130405021)

[phong.frag 6](#_Toc130405022)

[LightData 6](#_Toc130405023)

[MaterialDefinition 7](#_Toc130405024)

[LightDefinition 7](#_Toc130405025)

[CameraDefinition 7](#_Toc130405026)

[Functions 8](#_Toc130405027)

[TeapotAD.cpp 9](#_Toc130405028)

[Functions 9](#_Toc130405029)

[SceneDiffuse(.h and .cpp) 10](#_Toc130405030)

[Enums 10](#_Toc130405031)

[Classes 11](#_Toc130405032)

[Structures 11](#_Toc130405033)

[Functions 12](#_Toc130405034)

[Phong shading 13](#_Toc130405035)

[Diffuse 13](#_Toc130405036)

[Ambient 13](#_Toc130405037)

[Specular 13](#_Toc130405038)

[Final Result 13](#_Toc130405039)

[Controls 14](#_Toc130405040)

[Results 14](#_Toc130405041)

[Conclusion 14](#_Toc130405042)

[References 14](#_Toc130405043)

## Introduction

A statement of the reports purpose

The purpose of this report is to show the different effects of attenuation factors, shininess, light intensity and material types on a teapot and a surface plane as well as provide a brief description of the shader used to create the result and the implementation process in which this shader was implemented.

## Brief History of shaders

### Pre-1980s

Gouraud shading was one of the first shading techniques published in 1971 by Henri Gouraud in which ‘a surface represented by a patch is approximated by polygonal planar facets. Gouraud computes information about the curvature of the surface at each vertex of each of these facets’. (Phong, 1975)

### 1980s-2000

In the 1980s, developing computer graphics was a pain; every hardware needed its own custom software(Hergaarden, 2011). This meant developing computer graphics took a lot of time and resulted in a program that could only run on specific hardware, limiting code re-useability.

An illustrated explanation of the theoretical principle of full Phong shading

History of shaders

History of phong shading

How to make a phong shader – formula’s

Real world examples

## Methodology

An annotated explanation of the sections of program code specifically needed to produce full Phong Shading including the structures and the functions

Talk about all code changes, structures and functions

### Introduction

Below is an explanation of all the structures and functions I have added to the program or were modified significantly, split into categories based on the file they are implemented in.

### phong.vert

Phong.vert calculates the model’s position on-screen using the values stores in the InputData structure below.

#### InputData

Text

Description automatically generated

This structure contains all the values that need be set in the main program for the vertex shader to work.

#### ProcessData

Text

Description automatically generated

This structure caches all the values that would be calculated more than once when the vertex shader is set up to reduce the number of operations.

#### LightData

Text

Description automatically generated

This structure contains all the data calculated from the input data in the vertex shader to be used for the fragment shader.

#### Functions

Text

Description automatically generated

What each function does:  
**calculateVM** – calculates the result of the model being translated by the camera’s view

**convertToHomogenous** – converts a vec3 to homogenous cords for matrix multiplication

**calculateVertexPosition** – calculates the vertex’s position on screen.

**calculateLightPosition** – calculates the light sources position on screen.

**calculateVertexNormal** – calculates the unit direction from the point to the light source

**setModelPositionOnScreen** – sets the model’s position on screen

Text

Description automatically generated

The main function puts all of the functions together to put the objects in the scene on screen.

### phong.frag

phong.frag calculates the total intensity of light using ambient, diffuse and specular light and the materials properties and produces a brightness value.

#### LightData

Text

Description automatically generated

This structure contains all of the data in the LightData structure in phong.vert

#### MaterialDefinition

Text

Description automatically generated

This structure contains data on a material with 3 different reflectivity values and a value for shinyness.

#### LightDefinition

Text

Description automatically generated

This structure contains data about the light source.

#### CameraDefinition

Text

Description automatically generated

This structure contains data about the camera.

#### Functions

Text

Description automatically generated

What each function does:

**calculateLightVector** – calculates the unit vector from the vertex to the source

**calculateAmbient** – calculates ambient light intensity based on the intensity of ambient light produced by the source and the ambient reflectiveness of the material.

**calculateDiffuse** – calculates diffuse light intensity based on the intensity of diffuse light produced by the source, the diffuse reflectiveness of the material and a positive angle between the material’s normal and the light source. This value is clamped between 0 and 1.

**calculateSpecular** – calculates specular light intensity based on the intensity of specular light produced by the source multiplied by the specular reflectiveness of the material further multiplied by (The angle between the camera angle and light reflected off the surface of the material) to the power of the material shininess value.

Text

Description automatically generated

The main function puts all of the above functions together and simply adds the intensity of each light type together to produce a final value which is used when rendering.

### TeapotAD.cpp

#### Functions

Text

Description automatically generated

This function prints the change in value that just occurred after +, -, \* or / was pressed on the keypad into console.

Graphical user interface, text

Description automatically generated

This function prints the value that would currently be changed if +, -, \* or / was pressed on the keypad into console. The value selected can be changed using 8 and 2 on the keypad. The switch statement will output a string based on what value was passed in.

Text

Description automatically generated

This function handles keypresses, I modified it so it enabled me to change values for all objects in the scene and view those values in console. Useful for testing purposes.

### SceneDiffuse(.h and .cpp)

#### Enums

Text

Description automatically generated

This Enum contains everything that can be changed about the scene

#### Classes

Text

Description automatically generated

The above class keeps track of what value is selected to be changed. The operator functions enable the value being selected to be changed. 22 is the last value in the enum.

#### Structures

Text

Description automatically generated

This structure can be used to define a light source’s intensity and enables the modification of these values.

Text

Description automatically generated

This structure can be used to define a materials reflectiveness and shininess factor and enables the modification of these values.

The default values are set up in the scene diffuse constructor function.

#### Methods

Text

Description automatically generated

The above method displays an object in the scene, used to display the plane and teapot and reduces code re-use.

A screenshot of a computer

Description automatically generated with medium confidence

The above method changes the value currently selected using the keypad by a specified amount.

Text

Description automatically generated

The above method gets the value currently selected using the keypad.

Text

Description automatically generated

The above method prints all of the values for everything in the scene into console.

### Results

#### Values

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Object | Specular Reflectiveness | | | Diffuse Reflectiveness | | | Ambient  Reflectiveness | | | Shininess  Factor |
| R | G | B | R | G | B | R | G | B |
| Teapot | 0.29 | 0.29 | 0.29 | 0.46 | 0.29 | 0 | 0.46 | 0.29 | 0 | 1 |
| Plane | 0.1 | 0.1 | 0.1 | 0.51 | 1 | 0.49 | 0.51 | 1 | 0.49 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Light Source | Ls | Ld | La |
| Default | 0.3 | 0.9 | 0.3 |

#### Diffuse

Scene with only diffuse lighting


Figure 1 Teapot with only diffuse lighting

Diffuse lighting was already implemented into the program, resulting in a teapot that looked like figure 1. Somewhat realistic, until you look to the horizon or at the dark side of the teapot.

#### Ambient

A picture containing text

Description automatically generated

Figure 2 Teapot with only ambient lighting

Ambient light alone resulted in a teapot that looked like figure 2. Dull and a single color.

#### Specular

A picture containing text

Description automatically generated

Figure 3 A teapot with only specular lighting

Specular light alone resulted in a teapot that looked like figure 3, very dark, but a little shiny.

#### Final Result

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

Figure 4b No shadow casted by the teapot on to the plane

Figure 5a A teapot with full phong shading

Figure 4 shows the final result of my phong shader, it looks realistic, apart from the fact there is no shadow casted by the teapot on to the plane.

### Controls

## Results

Output screen captures showing different effects of attenuation factors, shininess factor, light intensity and material types on the result. This should have a proper discussion and justification

Make a table

## Conclusion

Talk about how the project went

## References

Use IEEE style (does not count towards 2000 words limit)

[Illumination for computer generated pictures | Communications of the ACM](https://dl.acm.org/doi/abs/10.1145/360825.360839)  
  
[Continuous Shading of Curved Surfaces | IEEE Journals & Magazine | IEEE Xplore](https://ieeexplore.ieee.org/abstract/document/1671906)

[3388769.3407523 (acm.org)](https://dl.acm.org/doi/pdf/10.1145/3388769.3407523) - Physically Based Shading in eory and Practice

M. Hergaarden, “Literatuur shaders - CS.VU.NL,” *Graphics shaders*, Jan-2011. [Online]. Available: https://www.cs.vu.nl/~eliens/download/literatuur-shaders.pdf. [Accessed: 22-Mar-2023].

B. T. Phong, “Illumination for computer generated pictures,” *Communications of the ACM*, 01-Jun-1975. [Online]. Available: https://dl.acm.org/doi/10.1145/360825.360839. [Accessed: 22-Mar-2023].