

Graphics Programming Coursework

Michael Crainie S1828588



Session 2019/20 Trimester B

Computer GAMES: Software Development

Graphics Programming

Table of Contents

[**1.1 Declaration** 1](#_Toc40154360)

[**1.2 Shader Overview** 1](#_Toc40154361)

[**1.3 Custom Shader Overview** 1](#_Toc40154362)

[**1.4 Custom Shader (Fog)** 1](#_Toc40154363)

[**1.5 Custom Shader (Toon)** 2](#_Toc40154364)

[**1.6 Custom Shader (Combined)** 3](#_Toc40154365)

# **1.1 Declaration**

*I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any other award*.

*Michael Crainie*

# **1.2 Shader Overview**

A shader is a program that allows us to communicate with the computer’s GPU via the graphics pipeline. Shaders are typically written using GLSL which is a language very close to C but is specifically for use when programming graphics. Shaders written in GLSL primarily use in, out and uniform for their variable types in order to specify how each variable will be handled e.g. whether a variable should be passed from the frag shader to the vert or vice versa (ins and outs) or whether it will need to be manually set (uniforms). These variables will still require more traditional variable types such as floats, vec3s or vec4s.

# **1.3 Custom Shader Overview**

For this coursework, the custom shader being applied to one of the meshes is a combination of a fog shader and a toon shader. The toon shader is applied to the mesh and as the mesh moves further away more fog is applied to the object in order to make it darker in the distance. Before explaining how this effect is achieved it is best to explain how each of these shader techniques are achieved individually.

# **1.4 Custom Shader (Fog)**

First off in the toonFogCombine.vert we specify the version of glsl we would like to use which is version 400 in this case. A uniform of type mat4 named transform is then created which is the transform of whichever object this shader is being applied to. Next two outs are set one of type vec3 named v\_norm and one of type vec4 named v\_pos. These will be sent to the toonFogCombine.frag after being set. In the main function of the vert v\_norm is set to equal the VertexNormal along with v\_pos being calculated by multiplying the transform by the vertex position. Finally, gl\_position (position of the vertex) is also calculated using the same formula as the previous line. In the frag a out vec4 named FragColor is created to output the final colour the mesh the shader is being applied to, the two previous variables v\_norm and v\_pos are then received from the vert. Next three uniforms are created lightDir, maxDist and minDist these three will be set in the MainGame.cpp. In the main function of the frag a float named dist is created and the value is set to equal the absolute value of the vertex position on the Z axis in relation to the mesh. A second float named fogFactor is then created and calculated by (minDist – dist) / (minDist- maxDist). An interesting thing to note is the typical formula for this effect would actually be (maxDist – dist) / (maxDist – minDist) but this caused the adverse effect in this scenario and made the object appear lighter as it moved further away. The fogFactor is then clamped between values of 0 and 1 before another variable of type vec3 named lightColor is created and is then set to (0.1, 0.1, 0.1). Finally, a vec3 named color is created and is calculated by using the mix function on the fogColor, lightColor and fogFactor variables this is the colour that will be applied to the mesh’s vertices.

A screenshot of a cell phone

Description automatically generated

Figure 1 – Fog Frag Shader

A screen shot of a computer

Description automatically generated

Figure 2 – Fog Vert Shader

# **1.5 Custom Shader (Toon)**

In order to achieve the toon shading effect, the amount of light/intensity of light being applied to a mesh and its vertices needs to be calculated. In the vert for the toon shader a varying of type vec3 named normal is created along with a uniform of mat4 named transform. Transform works in the same way as in the fog shader and normal is made equal to VertexNormal. In the frag a uniform of type vec3 named lightDir is created and the normal from the vert is taken in. Inside the main function a float named intensity is created along with a vec4 named color. Intensity is then calculated by using the dot product on the lightDir and the surface normal. Dot(lightDir, normal). After this an else if statement is then created to simply change the color variable based on the light intensity. The model’s vertices are then set to the appropriate colour by setting gl\_FragColor to equal the color variable.

A screen shot of a social media post

Description automatically generated

Figure 3 - Toon Frag Shader

A screenshot of a cell phone

Description automatically generated

Figure 4 - Toon Vert Shader

# **1.6 Custom Shader (Combined)**

In order to combine both effects we simply need to move both effects into the one shader, create an additional variable in order to store both colours calculated by the individual effects separately and then finally multiply the two colours. After following these steps, the mesh will be toon shaded and apply fog to itself depending on its Z position. In this circumstance color is from the fog shader and color2 is from the toon shader (see figure 5).

A screenshot of a cell phone

Description automatically generated

Figure 5 - Custom Shader Frag

# 