|  |
| --- |
| Graphics Programming |
| Coursework Document |
| William Donald  S1632091 |

*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*.

Contents

[1: Shader Design 2](#_Toc8298270)

[1.1: Colour effect 2](#_Toc8298271)

[1.2: ToonRim 2](#_Toc8298272)

[1.3: Potential improvements 2](#_Toc8298273)

[2: Code Description 3](#_Toc8298274)

[2.1: MainGame.cpp 3](#_Toc8298275)

[2.2: ColourChange.frag 3](#_Toc8298276)

[3: Figures 4](#_Toc8298277)

[3.1: Table of figures 4](#_Toc8298278)

[3.3: Hyperlinks used 4](#_Toc8298279)

# 1: Shader Design

## 1.1: Colour effect

The effect of the shader has been inspired by the colour altering paint that is sometimes applies to cars. This paint alters the colour of the car depending on the angle the observer is viewing from and the light source of the given area. The effect of the shader will mimic this by utilising the position and rotation of the object the shader will be applied to in order to achieve an effect which is similar to the colour altering effect of the paint in the real world.

Figure 1: Volkswagen Karmann Ghia with paint effect

Figure 2: TVR Tuscan Speed Six with paint effect

The above figures provide an example of the real-world effect that provided the inspiration for the shader used. However, unlike the effect of the paint in the real world the shader will be using the position and rotation of the model in order to determine the colours that will be seen and how they change.

## 1.2: ToonRim

For simplicity of implementation and to allow the details of the model to stand out more during runtime the toonRim shader that was provided as a lab exercise was used as a foundation. This allowed for the effect to be applied within the fragment shader of the toonRim shader. This method also provides a vertex shader which aid in the integration of the new shader into the code architecture.

While a toonRim shader was used for this implementation the effect could be applied to other shaders for example the reflection shader, which would better simulate the reflective painted metal that the effect was inspired by.

## 1.3: Potential improvements

A way in which this effect could be improved could as stated above to use a reflection shader as the foundation for the colour change to be added in order for the simulation of a reflective metal material. Another improvement could be to use the light source and the camera view to recreate the actual effect that exists in the real world instead of creating a similar effect inspired by the real-world effect. Anther improvement could be to allow for textures to be implemented alongside the colour changing effect. This would allow the effect to be applied to textured models for example cars.

# 2: Code Description

## 2.1: MainGame.cpp

Figure 3: Code from MainGame method SetColourChange

void MainGame::SetColourChange()

{

ColourChange.setVec3("p\_Pos", glm::vec3( transform.GetPos()->x, transform.GetPos()->y, transform.GetPos()->z));

ColourChange.setVec3("p\_Rot", glm::vec3(transform.GetRot()->x, transform.GetRot()->y, transform.GetRot()->z));

}

The above code is used to pass in the current position and rotation of the model to the fragment shader as the vec3 uniform variables p\_Pos and p\_Rot. These variables will then be used in order to modify the colour of each fragment used to draw the model.

## 2.2: ColourChange.frag

Figure 4: Code from ColourChange fragment shader

uniform vec3 p\_Pos;

uniform vec3 p\_Rot;

void main()

{

vec3 posRotColour;

posRotColour = vec3(colour.x, colour.y, colour.z) + ((p\_Pos + p\_Rot + p) \* vec3 (-0.1,0.4,-0.1));

fragcolour.rgb = vec3(smoothstep(0.5, 0.9, vdn)) + posRotColour;

}

The above code is used to implement the colour changing effect into the fragment shader.

The uniform variables p\_Pos and p\_Rot allow for the current position and rotation of the model to be passed in from the main game through the SetColourChange method. The vec3 posRotColour is used to calculate the colour of the fragment. The vec4 fragcolour is used to apply the desired effect to the colour of the fragments.

The equation used to determine the value of posRotColour is:

vec3(colour.x, colour.y, colour.z) + ((p\_Pos + p\_Rot + p) \* vec3 (-0.1,0.4,-0.1))

The vector 3 consisting of the x y and z components of the vec4 colour are used to apply the toon effect of the toonRim shader used as the foundation. The second part of the equation is used to create the colour changing effect by adding the objects position, rotation and its position in clip space. The result is then multiplied by another vector 3 in order to manipulate the colours the object will become should a different set of colours be desired. The result is added to the vector 3 used for the toon effect in order to obtain a final value for posRotColour for the current fragment. To obtain a final output for fragcolour.rgb posRotColour is added to a vector3 that is used to apply a rim effect to the model.

# 3: Figures

## 3.1: Table of figures

[Figure 1: Volkswagen Karmann Ghia with paint effect 2](file:///C:\Users\HP%20Lappie\Desktop\Graphics%20programming\Graphics%20programming%20coursework%20document.docx#_Toc8298211)

[Figure 2: TVR Tuscan Speed Six with paint effect 2](file:///C:\Users\HP%20Lappie\Desktop\Graphics%20programming\Graphics%20programming%20coursework%20document.docx#_Toc8298212)

[Figure 3: Code from MainGame method SetColourChange 3](#_Toc8298213)

[Figure 4: Code from ColourChange fragment shader 3](#_Toc8298214)

## 3.3: Hyperlinks used

Figure 1 Hyperlink:<https://volkswagenutah.wordpress.com/2013/01/03/chromaflair-volkswagen-style/3225184076_127130551f_b/>

Figure 2 Hyperlink: <https://en.wikipedia.org/wiki/ChromaFlair>