Graphics Programming Documentation

Explanation of the code used to generate the additional graphical technique

Jakub Szulwinski

Computer Games (Software Development)

S1627131

*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*.

Jakub Szulwinski

**Table of Contents**

[**Introduction** 3](#_Toc8397446)

[**Shader Code Explanation** 4](#_Toc8397447)

[**Fragment Shader** 4](#_Toc8397448)

[**Uniforms** 4](#_Toc8397449)

[**Variables** 4](#_Toc8397450)

[**Main** 4](#_Toc8397451)

[**Vertex Shader** 5](#_Toc8397452)

[**Layouts** 5](#_Toc8397453)

[**Uniforms** 6](#_Toc8397454)

[**Variables** 6](#_Toc8397455)

[**Main** 6](#_Toc8397456)

[**Geometry Shader** 7](#_Toc8397457)

[**Layouts** 7](#_Toc8397458)

[**Uniforms** 8](#_Toc8397459)

[**Variables** 8](#_Toc8397460)

[**GeometryNormal** 8](#_Toc8397461)

[**Main** 8](#_Toc8397462)

# **Introduction**

This is a general overview and explanation of the code used to visualise normal with the use of the geometry shader. The geometry shader shows the position and direction of normals. The project was completed with the help of tutorials and lessons from the following websites and sources:

* <https://www.geeks3d.com/20111117/simple-introduction-to-geometry-shader-in-glsl-part-2/>
* <http://cpp0x.pl/kursy/Kurs-OpenGL-C++/GLSL/243>
* <https://learnopengl.com>
* [www.youtube.com/watch?v=C8FK9Xn1gUM&t](http://www.youtube.com/watch?v=C8FK9Xn1gUM&t) OpenGL Tutorial 49 Geometry Shaders
* <https://www.khronos.org/registry/OpenGL-Refpages>

# **Shader Code Explanation**

## **Fragment Shader**

This shader is used to generate a final output color to be used for the model.

*#version 400*

*in vec2 tex;*

*uniform sampler2D water;*

*out vec4 fragcolor;*

*void main()*

*{*

*fragcolor = vec4(1.0, 0, 0, 1.0);*

*}*

### **Uniforms**

Uniform sampler2D provides access to “water” texture for the model.

### **Variables**

Four-element vec4 fragcolor that holds the color data which is later defined in main(). Two-element tex variable is used to hold texture data.

### **Main**

The method sets the RGB value to yellow color by setting the R (red) value to max (1.0) and the B (blue) value to max (1.0).

## **Vertex Shader**

*#version 330 core*

*layout (location = 0) in vec3 aPos;*

*layout (location = 1) in vec2 aTexCoord;*

*layout (location = 2) in vec3 aNormal;*

*out vec2 TexCoord;*

*out vec3 v\_norm;*

*out vec2 v\_tex;*

*out vec4 v\_pos;*

*out VS\_OUT*

*{*

*vec3 normal;*

*}*

*vs\_out;*

*out vec3 Normal;*

*out vec3 Position;*

*uniform mat4 model;*

*uniform mat4 view;*

*uniform mat4 projection;*

*void main()*

*{*

*v\_norm = aNormal;*

*v\_tex = aTexCoord;*

*TexCoord = aTexCoord;*

*gl\_Position = model \* vec4(aPos, 1.0);*

*mat3 normalMatrix = mat3(transpose(inverse(view \* model)));*

*vs\_out.normal = normalize(vec3(projection \* vec4(normalMatrix \* aNormal, 0.0)));*

*Normal = mat3(transpose(inverse(model))) \* aNormal;*

*Position = vec3(model \* vec4(aPos, 1.0));*

*}*

### **Layouts**

Layouts are used to hold positions for data. aPos holds position of vertices, aTexCoord holds texture coordinates position and aNormal holds position of Normals.

### **Uniforms**

Matrices mat4 (4 columns, 4 rows) of floats are used to hold data for view matrix, model matrix and projection matrix, which are later used to calculate normals.

### **Variables**

*out vec2 TexCoord;* Two-element variable used to hold texture coordinates – going out

*out vec3 v\_norm;* Three-element variable used to hold normal data – going out

*out vec2 v\_tex;* Two-element variable used to hold texture data – going out

*out vec4 v\_pos;* Four-element variable used to hold position data – going out

*out vec3 Normal;* Three-element variable used to hold Normals – going out

*out vec3 Position;* Three-element variable used to hold Positions – going out

The interface block VS\_OUT outputs a three element normal variable.

### **Main**

First v\_norm and v\_tex are set respectively to data from uniforms aNormal and aTexCoord. Gl\_position of the model is calculated by multiplying model matrix with aPos. Normal matrix is calculated by transposing the inverse of the result of view times model matrix calculation. They are both 4x4 matrices so the result is 4x4 as well. Variable normal going out is a 4 element variable result of two normal matrices multiplying, later multiplied by the result of projection matrix and returning a three element variable which is normalized. Which means it returns a vector with the same direction as its parameter, but with length 1.

## **Geometry Shader**

*#version 330 core*

*layout (triangles) in;*

*layout (line\_strip, max\_vertices = 6) out;*

*uniform mat4 view;*

*uniform mat4 projection;*

*in vec2 TexCoord[];*

*in vec3 Normal[];*

*in VS\_OUT*

*{*

*vec3 normal;*

*}*

*gs\_in[];*

*in vec2 v\_tex[];*

*out vec2 TexCoords;*

*out vec2 tex;*

*const float SCALE = 0.03;*

*void GeometryNormal(int index)*

*{*

*gl\_Position = gl\_in[index].gl\_Position;*

*TexCoords = v\_tex[0];*

*EmitVertex();*

*gl\_Position = gl\_in[index].gl\_Position + vec4(gs\_in[index].normal, 0.0) \* SCALE;*

*EmitVertex();*

*EndPrimitive();*

*}*

*void main()*

*{*

*GeometryNormal(0);*

*}*

### **Layouts**

Layout triangles sets the geometry of the shape going in. The output of the shader is declared as line\_strip with the maximum amount of vertices as 6 which means 3 lines in a single strip.

### **Uniforms**

Matrices mat4 (4 columns, 4 rows) of floats are used to hold data for view matrix and projection matrix.

### **Variables**

*out vec2 TexCoord[];* Two-element array of variables used to hold texture coordinates – going in

*out vec3 tex;* Three-element variable used to hold texture data – going out

*out vec2 v\_tex[];* Two-element array of variables used to hold texture data – going in

*out vec4 v\_pos;* Four-element variable used to hold position data – going out

*out vec3 Normal[];* Three-element array of variables used to hold Normals – going in

Constant float SCALE used to set the size of normal visualised by the shader.

The interface block VS\_OUT outputs a three element normal variable.

### **GeometryNormal**

The position gl\_Position is defined as a current index value of the gl\_in.gl\_Position. Then it sets the TexCoords value to the v\_tex starting value. EmitVertex() emits the current values of output variables to the current output primitive. The gl\_Position then is updated by adding four-element variable gs\_in[index].normal and multiplying it by SCALE. The geometry shader then generates a normal vector for each vertex.

### **Main**

This method just initialises the GeometryNormal() method used to generate normal lines in geometry.