Ben Ivory (S1621251)

Computer Games (Software Development)

**Graphics Programming**

(M3I622944)

*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*.

Ben Ivory

Contents

[**1.0** **Introduction** 3](#_Toc8131310)

[**2.0** **Overview of additional graphical technique** 4](#_Toc8131311)

[**2.1** **shaderSpikeBall.vert** 4](#_Toc8131312)

[2.1.1 Layouts 4](#_Toc8131313)

[2.1.2 Uniforms 4](#_Toc8131314)

[2.1.3 Variables - Out 4](#_Toc8131315)

[2.1.4 Vertex Shader 4](#_Toc8131316)

[**2.2** **shaderSpikeBall.geom** 5](#_Toc8131317)

[2.2.1 Layouts 5](#_Toc8131318)

[2.2.2 Uniforms 5](#_Toc8131319)

[2.2.3 Variables – In 5](#_Toc8131320)

[2.2.4 Variables – Out 5](#_Toc8131321)

[2.2.5 Geometry Shader 5](#_Toc8131322)

[**2.3** **shaderSpikeBall.frag** 6](#_Toc8131323)

[2.3.1 Uniforms 6](#_Toc8131324)

[2.3.2 Variables – In 6](#_Toc8131325)

[2.3.3 Variables – Out 6](#_Toc8131326)

[2.3.4 Fragment Shader 6](#_Toc8131327)

1. **Introduction**

In this document the code used to produce the effect of spikes grown from added geometry will be explained and discussed.

1. **Overview of additional graphical technique**
   1. **shaderSpikeBall.vert**
      1. Layouts

A vec3 “aPos” to hold the position of each vertex.

A vec2 “aTexCoord” to hold the position of the texture coordinates.

A vec3 “aNormal” to each vertex normal.

* + 1. Uniforms

A mat4 “model” which holds the model matrix.

A mat4 “view” which holds the view matrix.

A mat4 “projection” which holds the projection matrix.

* + 1. Variables - Out

A vec2 “TexCoord” to hold the new texture coordinates data.

A vec3 “Normal” to hold the new normal data.

* + 1. Vertex Shader

The uniforms containing the matrices, that make up the model view projection matrix when combined, are used to transport the data, first from local space to world space using the model matrix, by from world to camera space utilising the view matrix and finally from camera to screen space with the projection matrix. The texture coordinates for each vertex are assigned from the data gathered from the appropriate layout and the normal data for each is moved into world space, and to allow the ability to stretch and warp the model, by multiplying the normal data by the normal matrix which is the transpose of the inverse of the model matrix. The “gl\_position” of each vertex is calculated using the model matrix.

* 1. **shaderSpikeBall.geom**
     1. Layouts

A layout with the parameter “triangles” and this dictates to the geometry the type of data it will be processing, in this case 3 points.

A layout containing the parameters “triangle\_strip, max\_vertices = 9”, this dictates the output of the geometry shader and as such outputs 3 (9 vertices) triangles, in a strip.

* + 1. Uniforms

A mat4 “view” which holds the view matrix.

A mat4 “projection” which holds the projection matrix.

* + 1. Variables – In

A vec2 “TexCoord[]” to hold the new array of texture coordinates data.

A vec3 “Normal[]” to hold the new array of normal data.

* + 1. Variables – Out

A vec2 “tex” containing the texture coordinates.

* + 1. Geometry Shader

The geometry shader takes in a triangle per invocation and then this is used to calculate a number of things. The “centerPos” (the center of each triangle) is calculated by adding the position of each point in the triangle together and dividing by the number of points. The “centerTexCoord” (the texture coordinates at the middle of each triangle) is calculated in a similar way, using the texture coordinates of each point. The “centerNormal” (the normal of the center point of the triangle) is calculated by using two vec3’s, both emerging from the same point on the triangle in this case the second and third point take away the first. These are fed into a cross product calculation and normalised, returning the “centerNormal”. The center of the new spike (“spikeCenter”) is calculated by using the centerPos and the center normal, plus the height of the new spike. Then the “gl\_Position” is multiplied by the projection and view matrix (previously multiplied by the model matrix in the vertex shader.) The texture coordinates are assigned for the relevant vertex and the new triangles are created.

* 1. **shaderSpikeBall.frag**
     1. Uniforms

A sampler2D “bricks” that holds the texture data.

* + 1. Variables – In

A vec2 “tex” containing the texture coordinates.

* + 1. Variables – Out

A vec4 “FragColour” that holds the colour of the fragment.

* + 1. Fragment Shader

The fragment shader determines the colour of each fragment, using the texture from the sampler2D and the texture coordinates.