Name: Unna Pussayapaiboon (64462335) Course: COSC422 — Assignment 1

1. Description

Task 3 is chosen for this assignment. The program from Ex-05[1] was extended in order to approximate the surface of a cone. By applying an equation $z = \sqrt{x^2 + y^2}$, an approximated surface should look similar to a cone as shown in Figure 1. Tessellation level controller and lighting calculation are also applied to render the 3-D model.

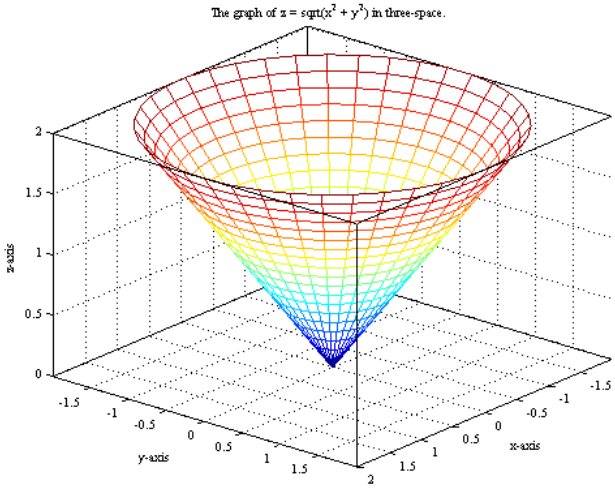

Figure 1: The Surface of a Cone [2]

2. Implemented Method

The program consists of seven files which are the main program (.cpp), vertex shader (.vert), control shader (.cont), evaluation shader (.eval), geometry shader (.geo), fragment shader (.frag), and the icosahedron vertex data header file (.h). The program flow is as shown in Figure 2. The vertex shader outputs the clip coordinates of the current vertex, and the position is passed to the control shader where the tessellation level is set. The primitive generator subdivides patches of vertex data into smaller primitives. Each of these contains a triangle mesh with vertices which refers to the tessellation coordinates. The patch vertices are fed into the tessellation evaluation shader. The evaluator then repositions each mesh vertex followed an approximation formula using patch vertices, and output them in clip coordinates. The output from evaluation shader is then sent to the geometry shader where the lighting calculation is done, and the resulting colour is sent to the fragment shader. The fragment shader sets the colour for each fragment.

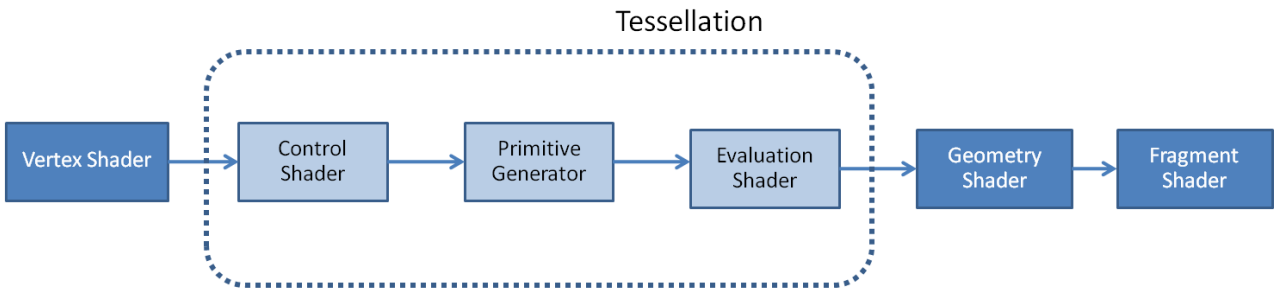

Figure 2: Program Flow

2.1) Cone Surface Approximation

glPatchParameteri is used to specify patches with 3 vertices in each patch. The tessellation is used to generate a mesh of triangles based on these vertices. The surface approximation is implemented in the tessellation evaluation shader. The interpolated position of the iscosahedron is given by the vector sum of the product between mesh vertex coordinates (gl_TessCoord) and barycentric coordinates (gl_in.gl_position).

An icosahedron from Ex-05 is used to approximate a sphere by normalising the interpolated position which will change the mesh vertex length from the centre to 1 unit. For the cone, the length of the mesh vertex will vary at each point along the z-axis. These lengths are radiuses of the cone along the z-axis. According to Figure 3, Pythagoras theorem can be applied to find the radius $r = \sqrt{x^2 + y^2}$. This means that $z = r = \sqrt{x^2 + y^2}$. Therefore, applying $z = \sqrt{x^2 + y^2}$ where the x and y coordinates are same as the sphere will give approximated surface of a cone in z-direction. The result surface is as shown in Figure 4 and Figure 5.

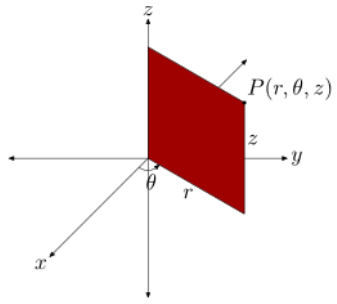

Figure 3: Radius Calculation [2]

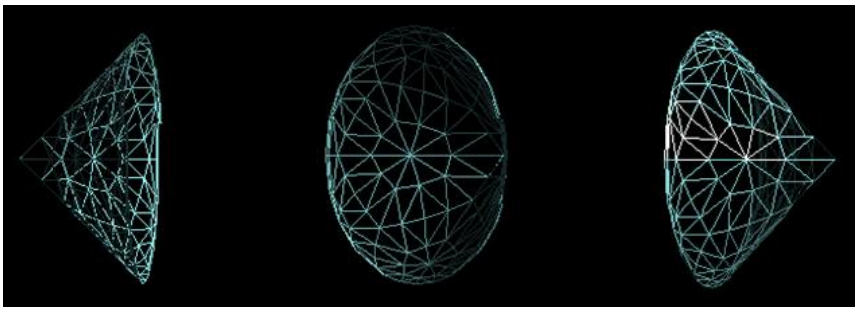

Figure 4: Cone Rendered (GL_LINE)

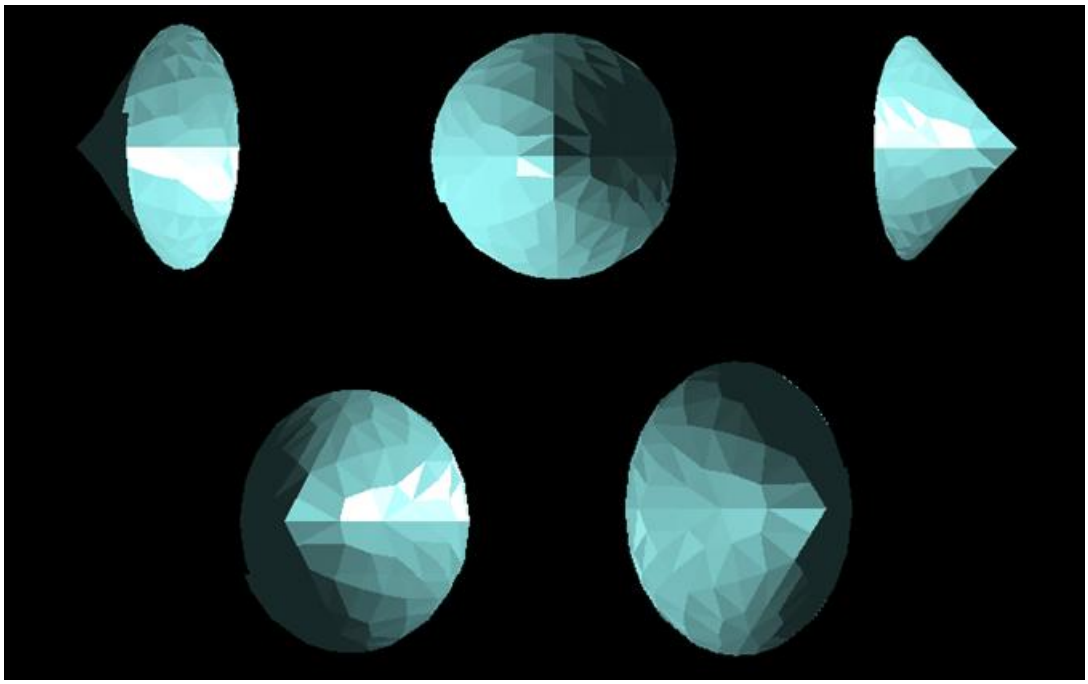

Figure 5: Cone Rendered (GL_FILL)

2.2) Tessellation level

Control shader is used to specify how much tessellation a particular patch gets. As the control shader is added, data from vertex shader will pass through the control shader before reach the evaluation shader. Patch vertices from the vertex shader are fed into the control shader where new patch vertices are generated according to the inner and outer tessellation level factor inside the control shader. The uniform float “tesslv” is defined, and used to vary the tessellation level in the control shader. The default values of inner and outer tessellation level are set at (4, 4, 4, 0) and (4, 0) respectively. The special keyboard function is created in the main program to support the zoom function. If the down button is pressed, zoomed factor increases by 2 and the tessellation level decreases. Increasing zoom factor will move the camera further which makes the cone look smaller. The tessellation level is set to decrease as the distance of the camera from the cone increases, vice versa. Since the special function is not inside the control shader, the function glUniform1f is used to specify the value of “tesslv” which is inside the control shader for the main program object. Figure 6 and Figure 7 shows the cone with minimum and maximum tessellation level factor respectively.

Figure 6: Cone with Minimum Tessellation Level Factor

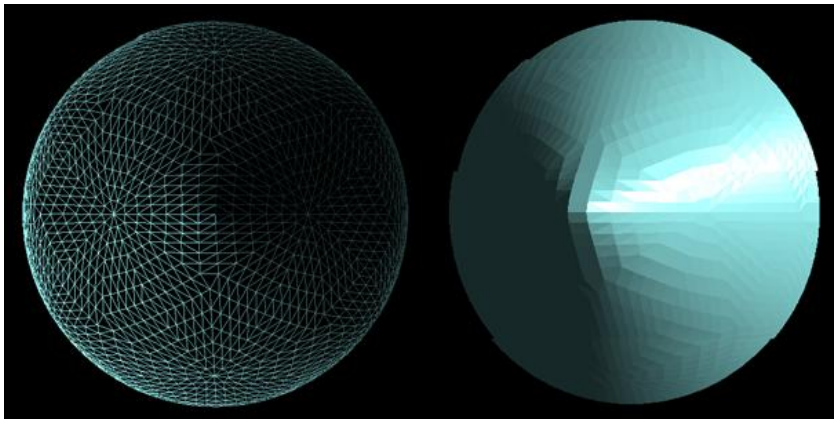

Figure 7: Cone with Maximum Tessellation Level Factor

2.3) Lighting Calculation

Lighting calculations for rendering the generated model are added by activating a geometry shader. The geometry shader receives a single primitive as input and may output zero or more primitives. It can also change the type of the input. At first, there was a problem in understanding how data passes through the geometry shader. If there is no tessellation shader, the geometry shader will take value straight from vertex shader. However, since the tessellation stage is used, the geometry shader receives the mesh vertex in clip coordinates from the tessellation evaluator. The geometry shader can process an entire primitive as it receives inputs in arrays with values corresponding to one whole primitive. Lighting calculations method from COSC422 lecture note [3] is applied to the geometry shader instead of the vertex shader, and the model-view projection matrix, the model-view matrix, the normal matrix, and the light position are specified as uniform variables. In this case, the new position which is obtained from the tessellation evaluator (gl_in) is used to calculate. Also, the transformed face normal is calculated by normalize the cross product of two vectors in the plane of the input triangle. These vectors are calculated by using Figure 8:

v1 = gl_in[2].gl_Position - gl_in[1].gl_Position

v2 = gl_in[0].gl_Position - gl_in[1].gl_Position

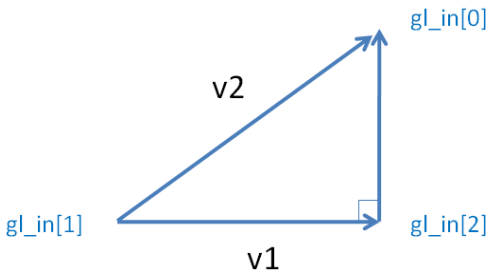

Figure 8: Transformed Face Normal Calculation

The colour from the geometry shader is emitted for each of the three gl_in vertices where gl_position is the product of mvp matrix and gl_in. These values are sent to fragment shader and displayed.

3. Keyboard Functions

- Up arrow key: Zoom in by factor of 2, and increase the tessellation level factor by 1. The maximum is set to 12.
- Down arrow key: Zoom out by factor of 2, and decrease the tessellation level factors by 1. The minimum is set to 0.
- Left arrow key: Toggle the display mode between GL_FILL and GL_LINE

4. References

[1] Ex-05 Surface Approximation

[2] Cone Equation http://msemac.redwoods.edu/~darnold/math50c/matlab/coordcyl/index.xhtml

[3] COSC422 Advanced Computer Graphics Lecture 1