GLSL Geometry Shaders

Mike Bailey
mjb@cs.oregonstate.edu
Oregon State University

Here's What We Know So Far

- Fixed Function
- Programmable

The Geometry Shader: Where Does it Fit in the Pipeline?

Geometry Shader: What Does it Do?

Your application generates these:
- Points, Lines, Line Strip, Line Loop, Lines with Adjacency, Line Strip with Adjacency, Triangles, Triangle Strip, Triangle Fan, Triangles with Adjacency, Triangle Strip with Adjacency

The driver translates these and feeds them one-at-a-time into the Geometry Shader.

The Geometry Shader generates (almost) as many of these as it wants:
- Points, Line Strips, Triangle Strips

There needn't be any correlation between Geometry Shader input type and Geometry Shader output type. Points can generate triangles, triangles can generate triangle strips, etc.

Additional Arguments Available for glBegin():

- GL_LINES_ADJACENCY
- GL_LINE_STRIP_ADJACENCY
- GL_TRIANGLES_ADJACENCY
- GL_TRIANGLE_STRIP_ADJACENCY
Adjacency Primitives (and what they do by default)

This is what Fixed-Function OpenGL expects these vertices to mean. In Shader World, they can mean whatever you want them to mean. In Shader World, it’s just a way to get multiple vertices into a Geometry Shader.

**Lines with Adjacency**

N vertices are given.

- Lines with Adjacency
- N+3 vertices are given

- A line segment is drawn between #1 and #2.
- Vertices #0 and #3 are there to provide adjacency information.
- A line segment is drawn between #1 and #2, #2 and #3, …, #N and #N+1.
- Vertices #0 and #N+2 are there to provide adjacency information.

**Triangles with Adjacency**

6N vertices are given.

- Triangles with Adjacency
- 4+2N vertices are given

- Points 0, 2, and 4 define the triangle.
- Points 1, 3, and 5 tell where adjacent triangles are.
- Points 0, 2, 4, 6, 8, 10, … define the triangles.
- Points 1, 3, 5, 7, 9, 11, … tell where adjacent triangles are.

**Example: A Bézier Curve**

\[ P(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3P_3 \]
Example: Expanding 4 Points into a Bezier Curve with a Variable Number of Line Segments

**bezier.curve.lib**

Vertex beziercurve.vert
Geometry beziercurve.geom
Fragment beziercurve.frag
Program BezierCurve uNum = <2 4 50>
LineWidth 3.
LinesAdjacency [0.0 0.0] [1.1 1.1] [2.2 2.2] [3.3 3.3]

**beziercurve.vert**

void main()
{
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}

**beziercurve.frag**

void main()
{
    gl_FragColor = vec4(0., 1., 0., 1.);
}

**bezier.curve.geom**

Note: these are used to define the storage

Note: It would have made no difference if the matrix transform had been done in the geometry shader instead

**beziercurve.geom**

uNum = 5
uNum = 25

Another Example: Shrinking Triangles

Example: Shrinking Triangles

CG = (P0 + P1 + P2) / 3;
P0’ = CG + uShrink * (P0 - CG)
P1’ = CG + uShrink * (P1 - CG)
P2’ = CG + uShrink * (P2 - CG)

Centroid = "CG"
#version 330 compatibility
#extension GL_EXT_gpu_shader4: enable
#extension GL_EXT_geometry_shader4: enable
layout( triangles )  in;
layout( triangle_strip, max_vertices=200 )  out;
uniform float uShrink;
in vec3 vNormal[3];
out float gLightIntensity;
const vec3 LIGHTPOS = vec3( 0., 10., 0. );
vec3 CG;
void
ProduceVertex( int v )
{
    gLightIntensity = dot( normalize(LIGHTPOS - V[v]), vNormal[v] );
gLightIntensity = abs( gLightIntensity );
    gl_Position = gl_ModelViewProjectionMatrix * vec4( CG + uShrink * ( V[v] - CG ), 1. );
    EmitVertex( );
}
void
main( )
{
    V[0]  =   gl_PositionIn[0].xyz;
    V[1]  =   gl_PositionIn[1].xyz;
    ProduceVertex( 0 );
    ProduceVertex( 1 );
    ProduceVertex( 2 );
}
```c
void main()
{
  V01 = (gl_PositionIn[1] - gl_PositionIn[0]).xyz;
  V02 = (gl_PositionIn[2] - gl_PositionIn[0]).xyz;
  V0 = gl_PositionIn[0].xyz;

  int numLayers = 1 << uLevel;
  float dt = 1. / float(numLayers);
  float t_top = 1.;

  for (int it = 0; it < numLayers; it++)
  {
    float t_bot = t_top - dt;
    float smax_top = 1. - t_top;
    float smax_bot = 1. - t_bot;
    int nums = it + 1;
    float ds_top = smax_top / float(nums - 1);
    float ds_bot = smax_bot / float(nums);

    float s_top = 0.;
    float s_bot = 0.;

    for (int is = 0; is < nums; is++)
    {
      ProduceVertex(s_bot, t_bot);
      ProduceVertex(s_top, t_top);
      s_top += ds_top;
      s_bot += ds_bot;
    }
    ProduceVertex(s_bot, t_bot);
    EmitVertex();
  }
  t_top = t_bot;
  t_bot -= dt;
}
```

Example: Sphere Subdivision

Example: Sphere Subdivision with One triangle

Example: Sphere Subdivision with the Whole Sphere (8 triangles)

Example: Explosion

Another Example: Explosion

خاریج: 1. Break the triangles into points  
2. Treat each point’s distance from the triangle’s CG as an initial velocity  
3. Follow the laws of projectile motion:

\[
X = X_0 + V_x t \\
Y = Y_0 + V_y t + \frac{1}{2} a_y t^2
\]
Example: Explosion

```cpp
void main()
{
    V01 = (gl_PositionIn[1] - gl_PositionIn[0]).xyz;
    V02 = (gl_PositionIn[2] - gl_PositionIn[0]).xyz;
    V0  =   gl_PositionIn[0].xyz;
    CG = (gl_PositionIn[0].xyz + gl_PositionIn[1].xyz + gl_PositionIn[2].xyz) / 3.;
    int numLayers = 1 << uLevel;
    float dt = 1. / float(numLayers);
    float t = 1.;
    for( int it = 0; it <= numLayers; it++ )
    {
        float smax = 1. - t;
        int nums = it + 1;
        float ds = smax / float(nums - 1);
        float s = 0.;
        for( int is = 0; is < nums; is++ )
        {
            ProduceVertex(s, t);
            s += ds;
        }
        t -= dt;
    }
}
```

Another Example: Silhouettes

1. Compute the normals of each of the four triangles
2. If there is a sign difference between the z component of the center triangle and the z component of an adjacent triangle, draw their common edge

Example: Silhouettes

```cpp
void main()
{
    gl_Position = gl_ModelViewMatrix * gl_Vertex;
}
```

uniform vec4 uColor;

```cpp
void main()
{
    gl_FragColor = vec4(uColor.rgb, 1.);
}
```

```cpp
#version 330 compatibility
#extension GL_EXT_gpu_shader4: enable
#extension GL_EXT_geometry_shader4: enable
layout( triangles_adjacency )  in;
layout( line_strip, max_vertices=200 )  out;
void main( )
{
    vec3 V0 = gl_PositionIn[0].xyz;
    vec3 V1 = gl_PositionIn[1].xyz;
    vec3 V2 = gl_PositionIn[2].xyz;
    vec3 V3 = gl_PositionIn[3].xyz;
    vec3 V4 = gl_PositionIn[4].xyz;
    vec3 V5 = gl_PositionIn[5].xyz;
    vec3 N042 = cross( V4-V0, V2-V0 ); // the center triangle's normal
    vec3 N021 = cross( V2-V0, V1-V0 );
    vec3 N243 = cross( V4-V2, V3-V2 );
    vec3 N405 = cross( V0-V4, V5-V4 );
    if( dot( N042, N021 ) < 0. ) // make sure each outer triangle's normal is in the same general direction
    N021 = vec3(0.,0.,0.) - N021;
    if( dot( N042, N243 ) < 0. )
    N243 = vec3(0.,0.,0.) - N243;
    if( dot( N042, N405 ) < 0. )
    N405 = vec3(0.,0.,0.) - N405;
    }
if( N042.z * N021.z <= 0. ) {
    gl_Position = gl_ProjectionMatrix * vec4( V0, 1. );
    EmitVertex( );
    gl_Position = gl_ProjectionMatrix * vec4( V2, 1. );
    EmitVertex( );
    EndPrimitive( );
}
if( N042.z * N243.z <= 0. ) {
    gl_Position = gl_ProjectionMatrix * vec4( V2, 1. );
    EmitVertex( );
    gl_Position = gl_ProjectionMatrix * vec4( V4, 1. );
    EmitVertex( );
    EndPrimitive( );
}
if( N042.z * N405.z <= 0. ) {
    gl_Position = gl_ProjectionMatrix * vec4( V4, 1. );
    EmitVertex( );
    gl_Position = gl_ProjectionMatrix * vec4( V0, 1. );
    EmitVertex( );
    EndPrimitive( );
}

Example: Silhouettes

Another Example: Hedgehog Plots

void main( ) {
    V0  =   gl_PositionIn[0];
    V01 = ( gl_PositionIn[1] - gl_PositionIn[0] );
    V02 = ( gl_PositionIn[2] - gl_PositionIn[0] );
    Norm[0] = vTnorm[0];
    Norm[1] = vTnorm[1];
    Norm[2] = vTnorm[2];
    if( dot( Norm[0], Norm[1] ) < 0. ) Norm[1] = -Norm[1];
    if( dot( Norm[0], Norm[2] ) < 0. ) Norm[2] = -Norm[2];
    N0   = normalize( Norm[0] );
    N01 = normalize( Norm[1] - Norm[0] );
    N02 = normalize( Norm[2] - Norm[0] );
    int numLayers = 1 << uDetail;
    float dt = 1. / float( numLayers );
    float t = 1.;
    for( int it = 0; it <= numLayers; it++ ) {
        float smax = 1. - t;
        float da = smax / float( numLayers - 1 );
        float s = 0.;
        for( int is = 0; is < numLayers; is++ ) {
            ProduceVertices( s, t );
            s += da;
        }
        t -= dt;
    }
}
A GLSL Built-in Variable for the Geometry Shaders

int gl_PrimitiveIDIn

- Tells the number of primitives processed since the last time glBegin() was called
- Calling a vertex array function counts as an implied glBegin()
- gl_PrimitiveIDIn is 0 for the first primitive after the glBegin()

Geometry shaders can set the built-in variable gl_PrimitiveID to send a primitive number to the fragment shader

What Happens if you Exceed the Maximum Allowed Emitted Vertices?

New in GLSL 4.x – you can loop back through the Geometry Shader multiple times