GLSL Geometry Shaders

Here’s What We Know So Far

One Vertex In

One Vertex Out

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

Can change # of vertices and/or topology

Last stop before the Rasterizer

Additional Arguments Available for glBegin():

GL_LINES_ADJACENCY

GL_LINE_STRIP_ADJACENCY

GL_TRIANGLES_ADJACENCY

GL_TRIANGLE_STRIP_ADJACENCY

Geometry Shader: What Does it Do?

Your application generates these

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

The Geometry Shader generates (almost as many of these as it wants

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.
Adjacency Primitives (and what they do when not using shaders)

In general, we will use the “with adjacency” primitives as a way of importing some number of vertices into the geometry shader. These are the most useful:

- GL_LINES_ADJACENCY: 4 vertices
- GL_TRIANGLES_ADJACENCY: 6 vertices

In the Geometry Shader, the dimensions indicated by $N$ are given by the variable gl_Position, although you will already know this by the type of geometry you are inputting:

In the Geometry Shader, the dimensions indicated by $N$ are given by the variable gl_Position, although you will already know this by the type of geometry you are inputting:

If you are using a Geometry Shader, then the GS must be used if you want to pass information from the Vertex Shader to the Fragment Shader. These are already declared for you:

- out vec4 gl_Position:
- vColor = gl_Color;

Plus any of your own variables that you have declared to be out:

- in vec4 gl_Position;
- out vec4 gColor;
- gColor = vColor[k];
Example: A Bézier Curve

\[ P(u) = (1-u)^3P_0 + 3u(1-u)^2P_1 + 3u^2(1-u)P_2 + u^3P_3 \]

Need to pass 4 points in to define the curve. Need to pass N points out to draw the curve.

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
Example: Shrinking Triangles

\[ CG = \frac{(P_0 + P_1 + P_2)}{3} \]

\[ P'_0 = CG + u_{\text{Shrink}} \cdot (P_0 - CG) \]

\[ P'_1 = CG + u_{\text{Shrink}} \cdot (P_1 - CG) \]

\[ P'_2 = CG + u_{\text{Shrink}} \cdot (P_2 - CG) \]

Another Example: Sphere Subdivision

It's often useful to be able to parameterize a triangle into \((s,t)\), like this:

\[ v(s,t) = V_0 + s(V_1 - V_0) + t(V_2 - V_0) \]

Note! There is no place in this triangle where \(S = T = 1\).
Example: Sphere Subdivision

```
spheresubd.geom

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

Example: Sphere Subdivision with One triangle

```
spheresubd.geom

Example: Sphere Subdivision
```

Another Example: Explosion

```
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

```glsl
#version 330 compatibility
#extension GL_EXT_gpu_shader4: enable
#extension GL_EXT_geometry_shader4: enable
layout( triangles ) in;
layout( points, max_vertices=200 ) out;
uniform int uLevel;
uniform float uGravity;
uniform float uTime;
uniform float uVelScale;
vec3 V0, V01, V02;
vec3 CG;

void ProduceVertex( float s, float t )
{
  vec3 v = V0 + s*V01 + t*V02;
  vec3 vel = uVelScale * ( v - CG );
  v = CG + vel*uTime + 0.5*vec3(0.,uGravity,0.)*uTime*uTime;
  gl_Position = gl_ProjectionMatrix * vec4( v, 1. );
  EmitVertex( );
}
```

```glsl```

Example: Explosion

```glsl
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 ) / 3.0;
  int numLayers = 1 << uLevel;
  float dt = 1.0 / float( numLayers );
  float t = 1.0;
  for( int it = 0; it <= numLayers; it++ ) {
    float smax = 1.0 - t;
    int nums = it + 1;
    float ds = smax / float( nums - 1 );
    float s = 0.0;
    for( int is = 0; is < nums; is++ ) {
      ProduceVertex( s, t );
      s += ds;
    }
    t -= dt;
  }
}
```

Another Example: Silhouettes

```glsl```
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's normal and the z component of an adjacent triangle's normal, draw their common edge
   i.e., you are looking for a crease.

```glsl```
Example: Silhouettes

```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: Bunny Silhouettes

![Bunny Silhouettes](image)

---

Another Example: Hedgehog Plots

```cpp
#version 330 compatibility
#extension GL_EXT_gpu_shader4: enable
#extension GL_EXT_geometry_shader4: enable
layout(triangles) in;
layout(line_strip, max_vertices=200) out;
uniform int uDetail;
uniform float uDroop;
uniform int uLength;
uniform float uStep;
in vec3    vTnorm[3];
in vec4 vColor[3];
out vec4 gColor;
int ILength;
vec3 Norm[3];
vec3 N0, N01, N02;
vec4 V0, V01, V02;
void ProduceVertices( float s, float t )
{
  vec4 v = V0 + s*V01 + t*V02;
  vec3 n = normalize( N0 + s*N01 + t*N02 );
  for( int i = 0; i <= uLength; i++ )
  {
    gl_Position = gl_ProjectionMatrix * v;
    gColor = vColor[0];
    EmitVertex( );
    v.xyz += uStep * n;
    v.y -= uDroop * float(i*i);
  }
  EndPrimitive( );
}
```

---

Example: Silhouettes (continued)

```cpp
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( );
}
```

---

Another Example: Hedgehog Plots (continued)

```cpp
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;
  for( int i = 0; i < numLayers; i++ )
  {
    gl_Position = gl_ProjectionMatrix * vec4( V0, 1. );
    EmitVertex( );
    v.xyz += uStep * Norm[0];
    v.y -= uDroop * float(i*i);
  }
  EndPrimitive( );
}
```

---

Example: Bunny Silhouettes (continued)

![Bunny Silhouettes](image)

---

Another Example: Hedgehog Plots (continued)

![Hedgehog Plots](image)
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++ )
    {
        ProduceVertices(s, t);
        s += ds;
    }
    t -= dt;
}