COSE436

Lecture 11: Advanced GLSL

Oct 22, 2024

Won-Ki Jeong

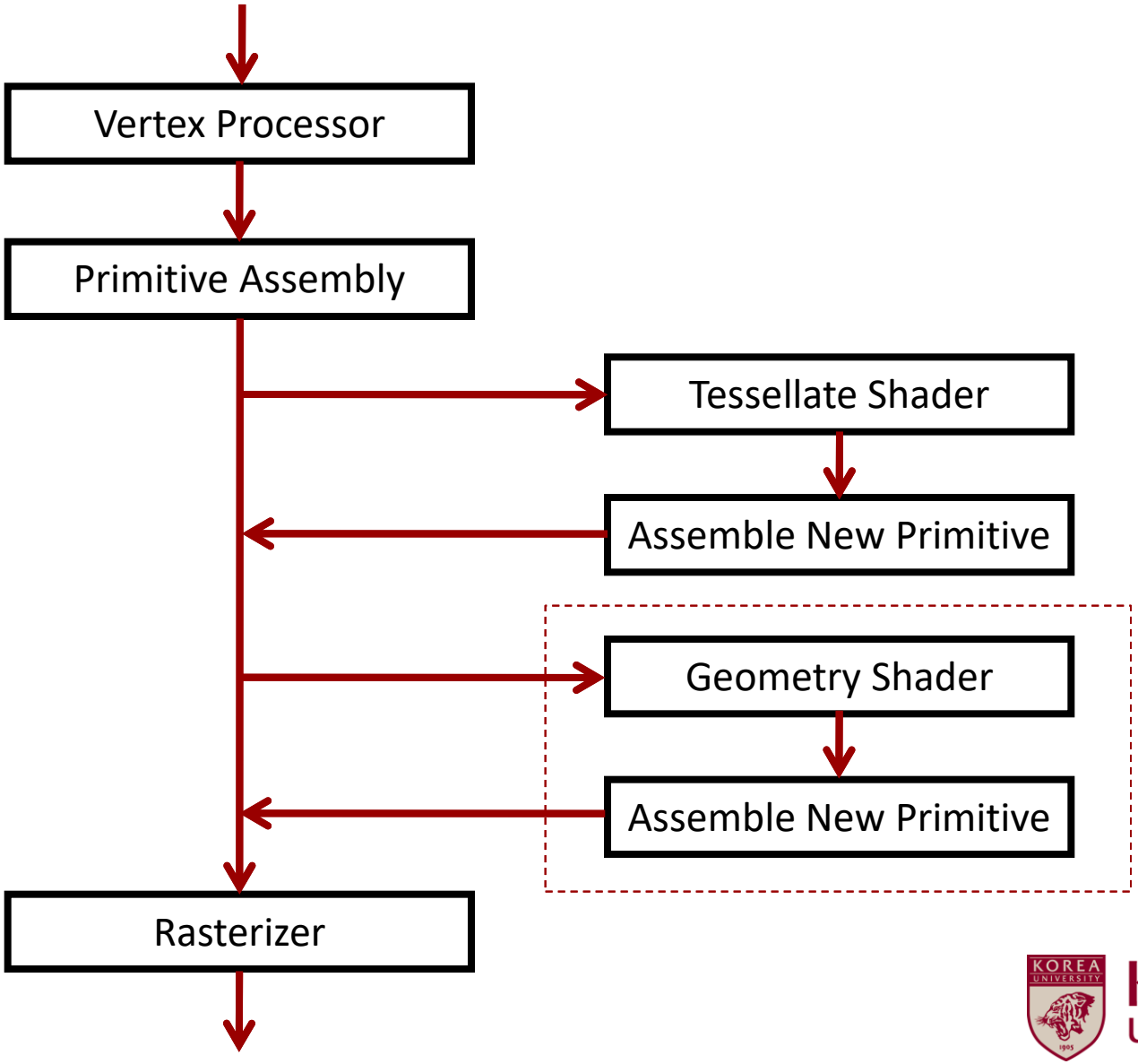
(wkjeong@korea.ac.kr)

KOREA UNIVERSITY

Outline

- Basic geometry shader setup
- Geometry shader examples
  - Bezier curve
  - Simple triangle
  - Subdivision using geometry shader

Geometry Shader

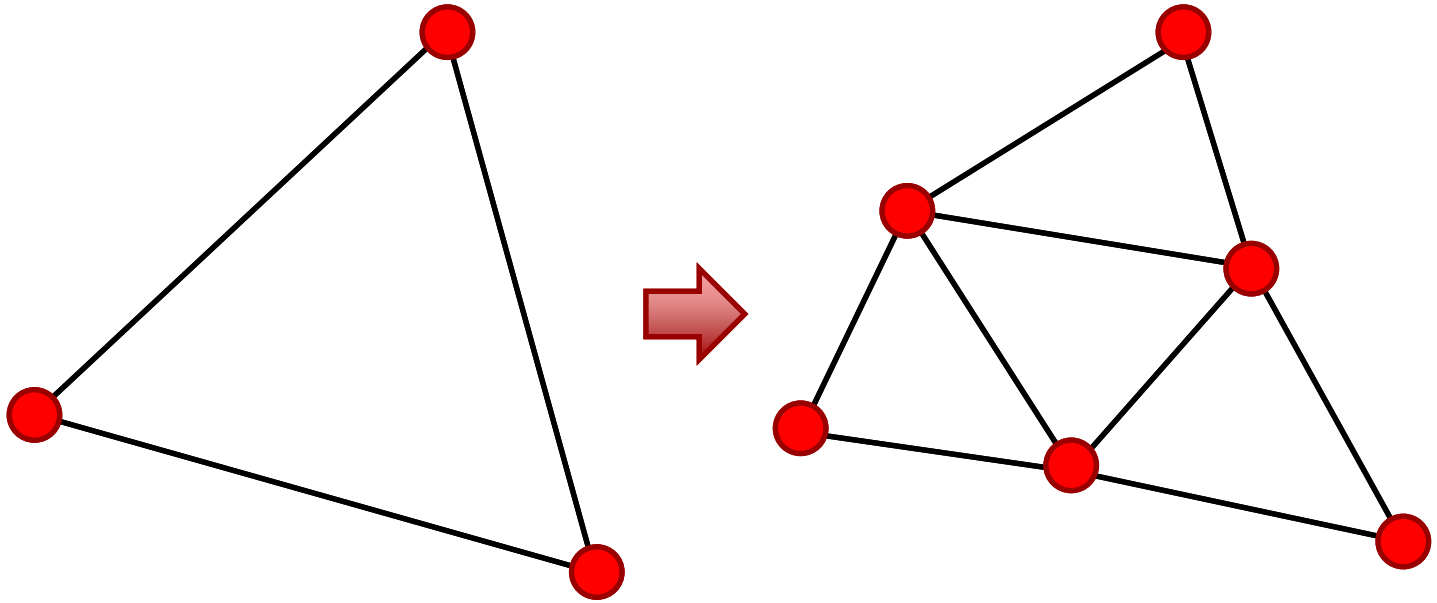


What Can Geometry Shader Do?

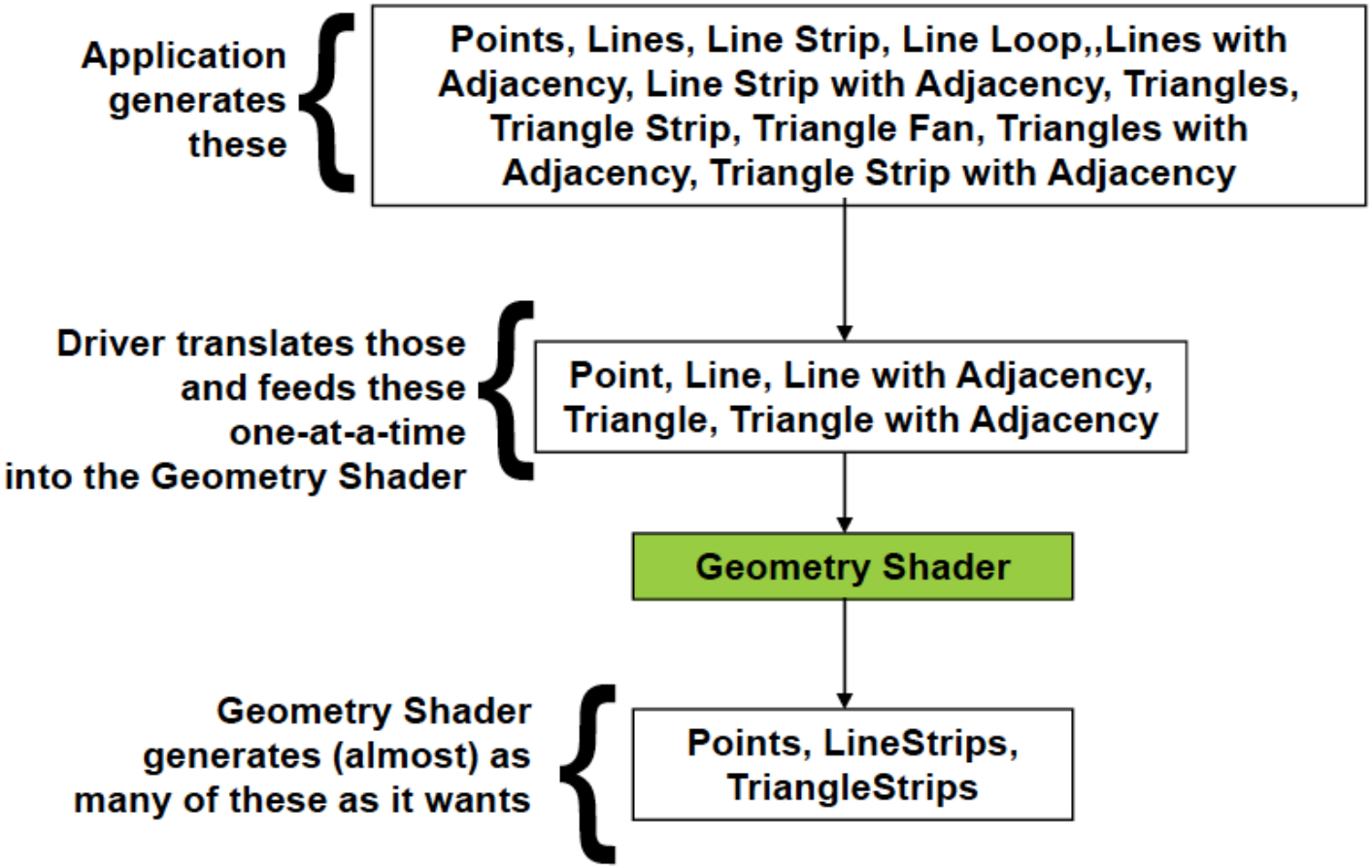
- Generate vertices
  - EmitVertex()
- Generate primitives
  - EndPrimitive()
- Apply transformations to vertices
- Assign vertex attributes
- Pass vertex attributes to rasterizer
  - Vertex normal, color, etc

Geometry Shader is...

- Per-primitive geometry modifier
  - Create / delete vertices
  - Create new primitives



Input / Output of Geom. Shader



Example: Triangle Strip

- Application point of view
  - n+2 vertices for n triangles
- Geometry shader point of view
  - n triangles, single triangle at a time
- Output of geometry shader
  - Points / line strips / triangle strips

Adjacency Primitives

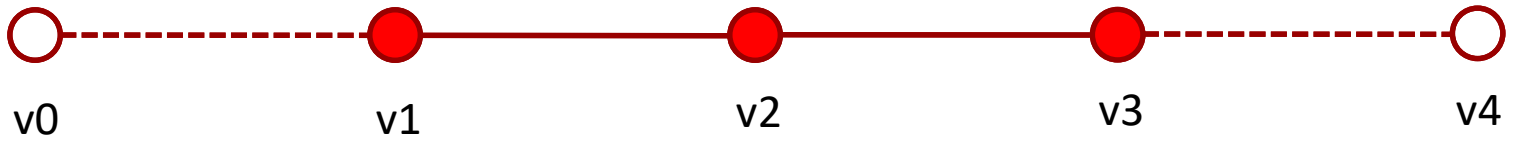
- Pass neighborhood information to geometry shaders
- Use as primitive type
  - `GL_LINE_ADJACENCY`
  - `GL_LINE_STRIP_ADJACENCY`
  - `GL_TRIANGLE_ADJACENCY`
  - `GL_TRIANGLE_STRIP_ADJACENCY`

Lines with Adjacency

- 4 vertices for a line segment
- GL will draw (v1,v2) only
- Geometry shader can access all four vertices

Line Strips with Adjacency

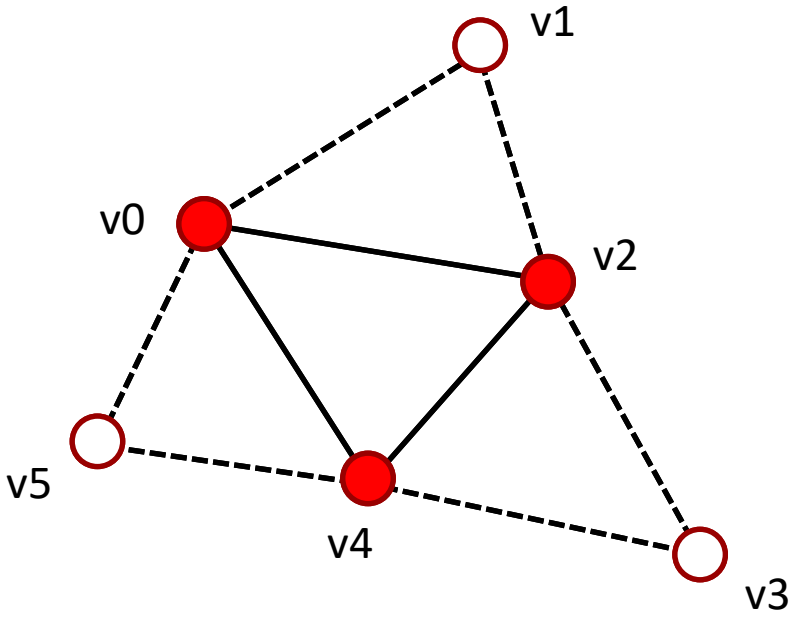
- n+3 vertices for n line segments
  - 0, 1, 2, …., n+2
- GL will draw (v1,v2), ..., (vn, vn+1) only
- Geometry shader can access all n+3 vertices



Example: n = 2

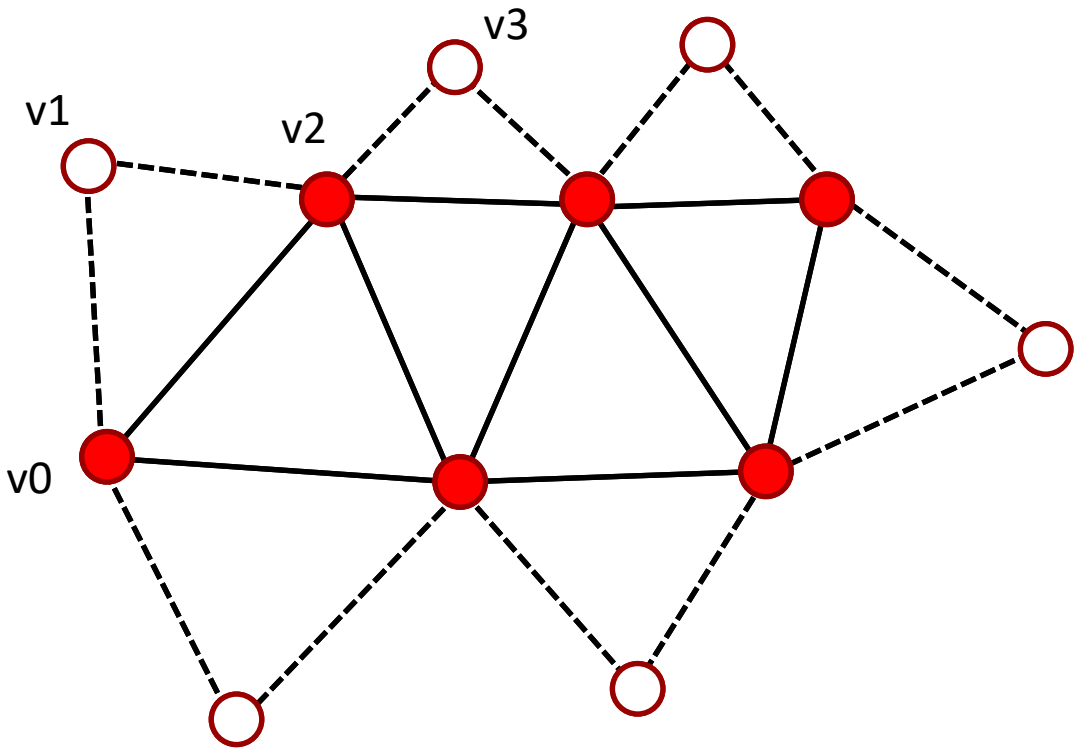
Triangles with Adjacency

- 6 vertices for a triangle
- v0, v2, v4 define a triangle
- v1, v3, v5 give neighborhood info



Triangle Strips with Adjacency

- 4 + 2n vertices for n triangles
- Even vertices (v0, v2, ...) define triangles
- Odd vertices (v1, v3, ...) define neighbor info



In/Out Types and Max Output

- Input / output / max vertex out can be set inside geometry shader code
  - GLSL 1.5 and above
- Input type: points, lines, lines_adjacency, triangles, triangles_adjacency
- Output: points, line_strip, triangle_strip

```
// Example: input is triangles, output is triangle
// strips, maximum number of output vertex is 6

layout(triangles) in;
layout(triangle_strip, max_vertices = 6) out;
```

Built-in Variable (GLSL 1.5~)

- `gl_in.length()`
  - # of vertices per primitive
- `gl_in[i]`
  - `.gl_Position`
  - `.gl_PointSize`
  - `.gl_ClipDistance[]`
- `gl_PrimitiveID`
- `gl_Layer`

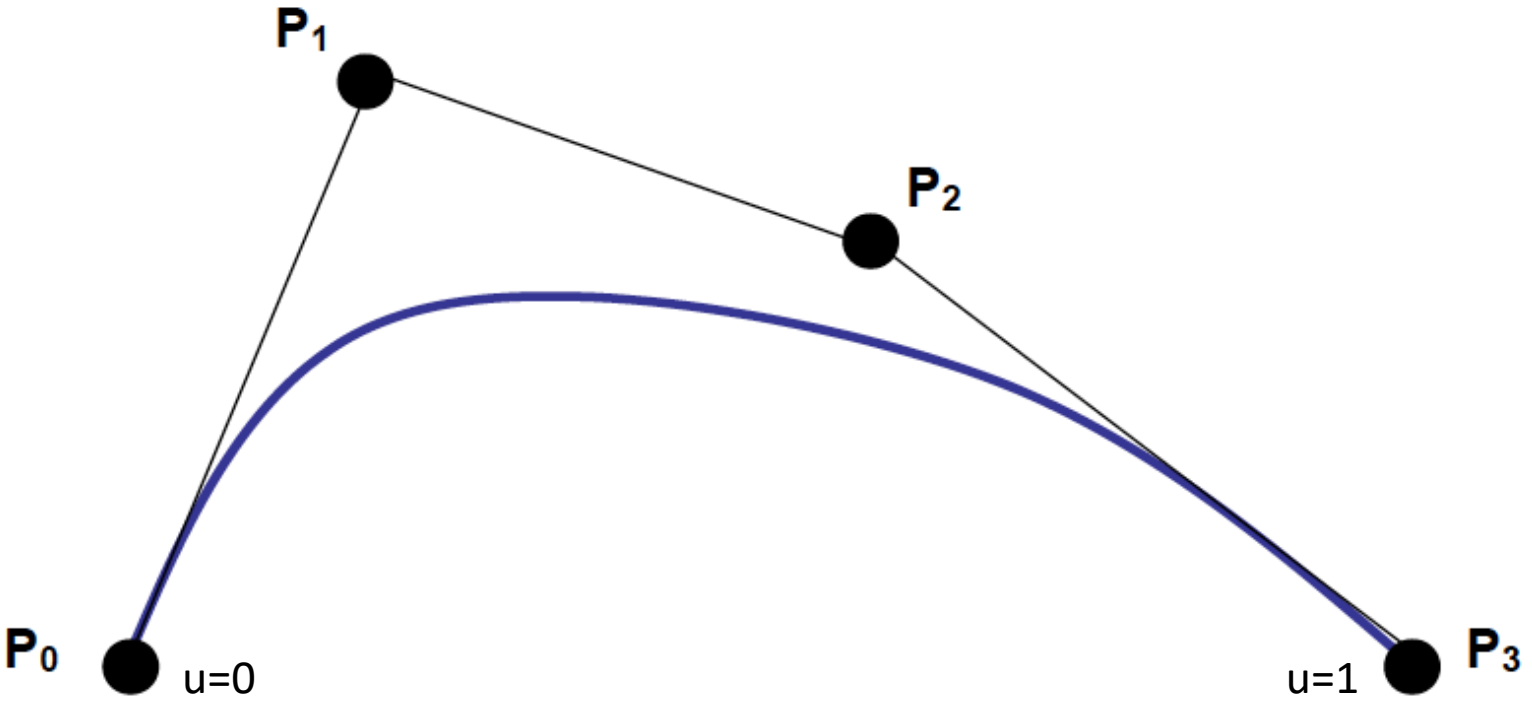
Note

- Vertex shader can pass the output to rasterizer if there is no geometry shader
- Geometry shader must be used with a vertex shader
- Input and output type for geometry shader can be *different*
  - Any primitive can be emitted from geometry shader

Outline

- Basic geometry shader setup
- Geometry shader examples
  - Bezier curve
  - Simple triangle
  - Subdivision using geometry shader

Bezier Curve



$$P(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3 P_3$$

Vertex / Fragment Shader

- Passing vertex / fragment

```
#version 140
#extension GL_ARB_compatibility: enable

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

```
#version 140
#extension GL_ARB_compatibility: enable
in vec3 perVertexColor;

void main()
{
    gl_FragColor.xyz = perVertexColor;
}
```

Geometry Shader

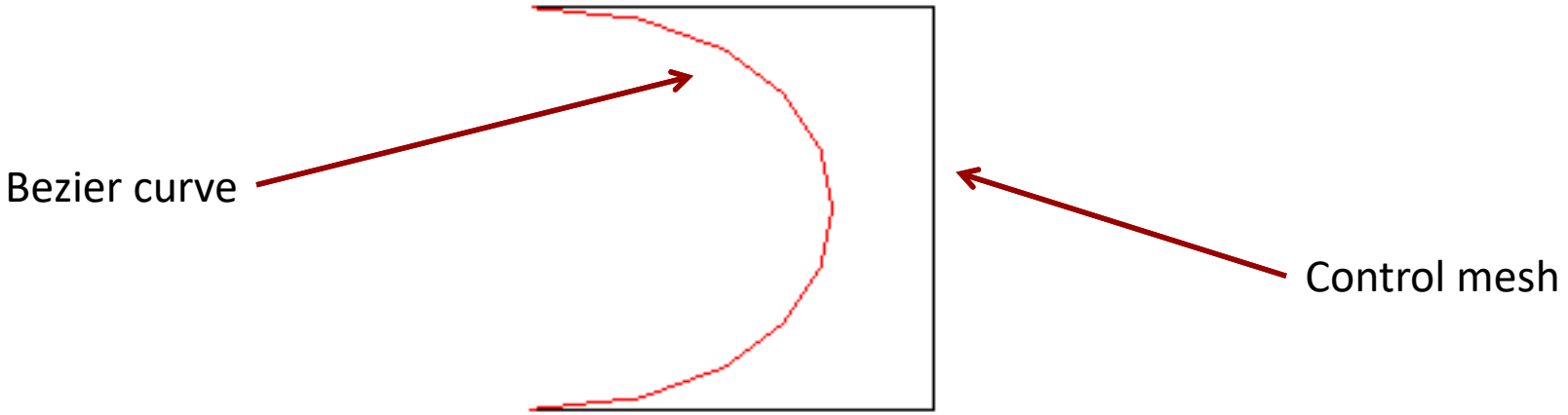
```
#version 150

layout(lines_adjacency) in;
layout(line_strip, max_vertices = 15) out;

out vec3 perVertexColor;
const int Num = 10;  // total # of output points : Num+1

void main()
{
  // Draw Control mesh : emit all four vertices
  for(int i = 0; i < gl_in.length(); i++)
  {
    gl_Position = gl_in[i].gl_Position;
    perVertexColor = vec3(0,0,0);
    EmitVertex();
  }
  EndPrimitive();
                                .....
```

Geometry Shader

```
                                    .....
  // Draw Bezier curve
  float dt = 1. / float(Num);
  float u = 0.0f;
  for( int i = 0; i <= Num; i++ ) // create Num+1 new vertices
  {
    float v = 1.0f - u;
    float v2 = v * v;
    float v3 = v * v2;
    float u2 = u * u;
    float u3 = u * u2;
    vec4 vtxPos = v3 * gl_in[0].gl_Position.xyzw +
                  3.0f * u * v2 * gl_in[1].gl_Position.xyzw +
                  3.0f * u2 * v * gl_in[2].gl_Position.xyzw +
                    u3 * gl_in[3].gl_Position.xyzw;
    gl_Position = vtxPos;
    perVertexColor = vec3(1,0,0);
    EmitVertex();
    u += dt;
  }
  EndPrimitive();
}
```

Result

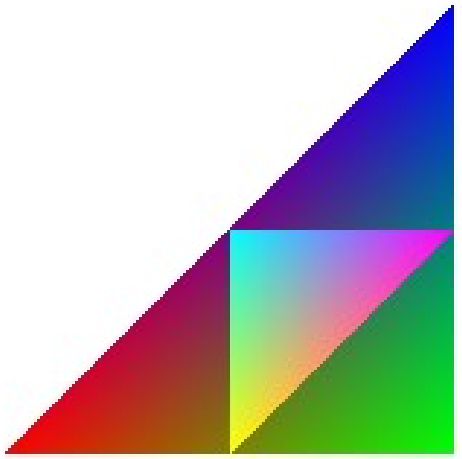
```
GLfloat vertices[] = {-1,-1,0, 1,-1,0, 1,1,0, -1,-1,0};
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3,GL_FLOAT,0,vertices);
glDrawArrays(GL_LINE_STRIP_ADJACENCY, 0, 4);
glDisableClientState(GL_VERTEX_ARRAY);
```



Simple Triangle Example

- Create a new triangle in geometry shader
  - Assign per-vertex color for interpolation

```
GLfloat vertices[] = {-1,-1,0, 1,-1,0, 1,1,0};
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3,GL_FLOAT,0,vertices);
glDrawArrays(GL_TRIANGLE, 0, 3);
glDisableClientState(GL_VERTEX_ARRAY);
```

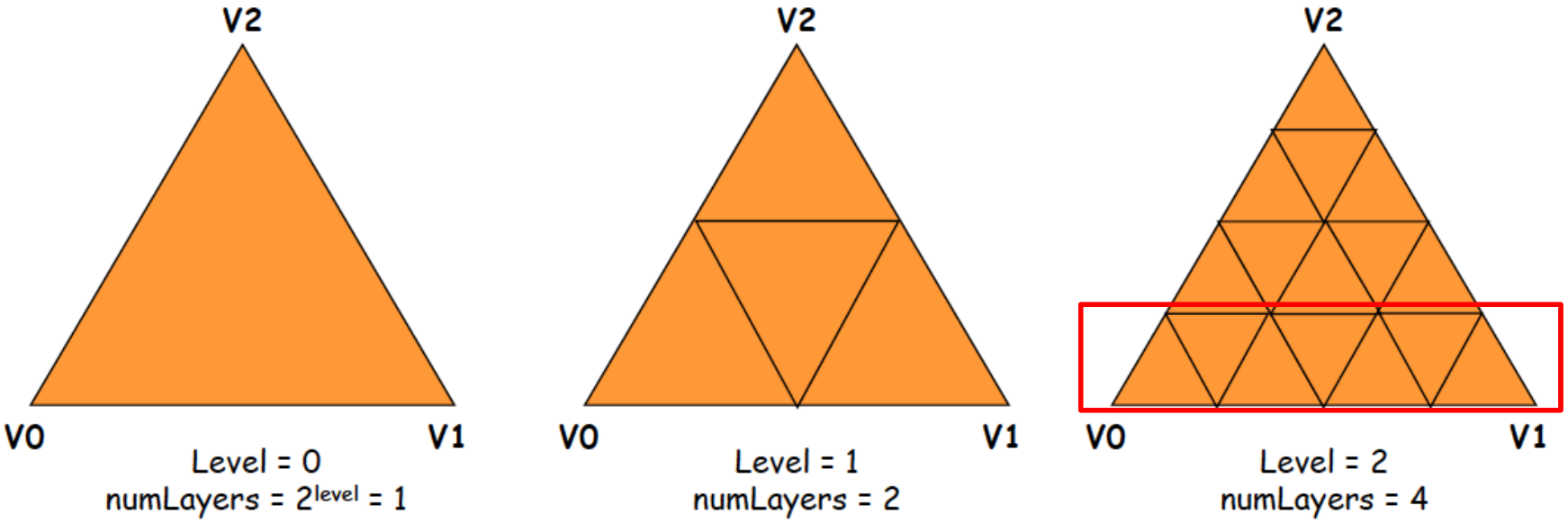


Generate a New Primitive

```
out vec3 perVertexColor;
layout(triangles) in;
layout(triangle_strip, max_vertices = 6) out;

const vec3 vtxCol[6] = vec3[6](vec3(1,0,0), vec3(0,1,0), vec3(0,0,1),
vec3(1,1,0), vec3(1,0,1), vec3(0,1,1));

void main() {
  for(int i = 0; i < gl_in.length(); i++) {
    gl_Position = gl_in[i].gl_Position;
    perVertexColor = vtxCol[i];
    EmitVertex();
  }
  EndPrimitive();  // passing input triangle as a new triangle strip

  for(int i = 0; i < gl_in.length(); i++) {
    if(i < 2)
      gl_Position = (gl_in[i].gl_Position + gl_in[i+1].gl_Position)/2.0 ;
    else
      gl_Position = (gl_in[2].gl_Position + gl_in[0].gl_Position)/2.0 ;
    perVertexColor = vtxCol[i+3];
    EmitVertex();
  }
  EndPrimitive();  // create a new triangle strip
}
```

Subdivision using Geometry Shader

- Each layer as a triangle strip along x-axis

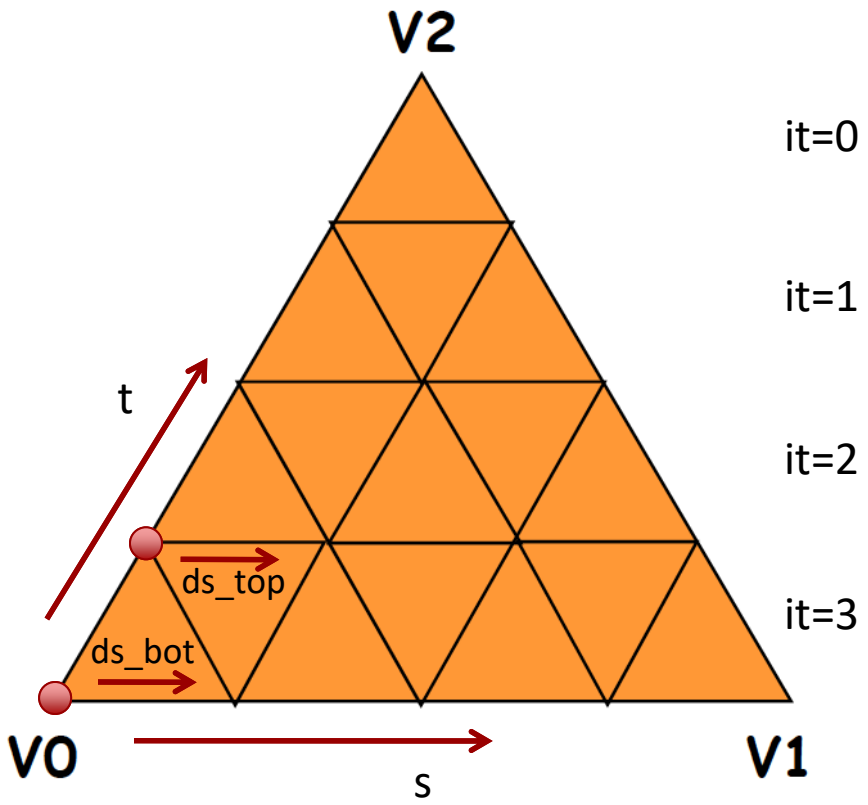


Geometry Shader

```
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 );
	EndPrimitive();

	t_top = t_bot;
	t_bot -= dt;
}
```

dt = 1/numLayer;

t_top = 1;



Questions?