Geometry shaders
(GLSL 3.30 core)

C. Andújar (*)

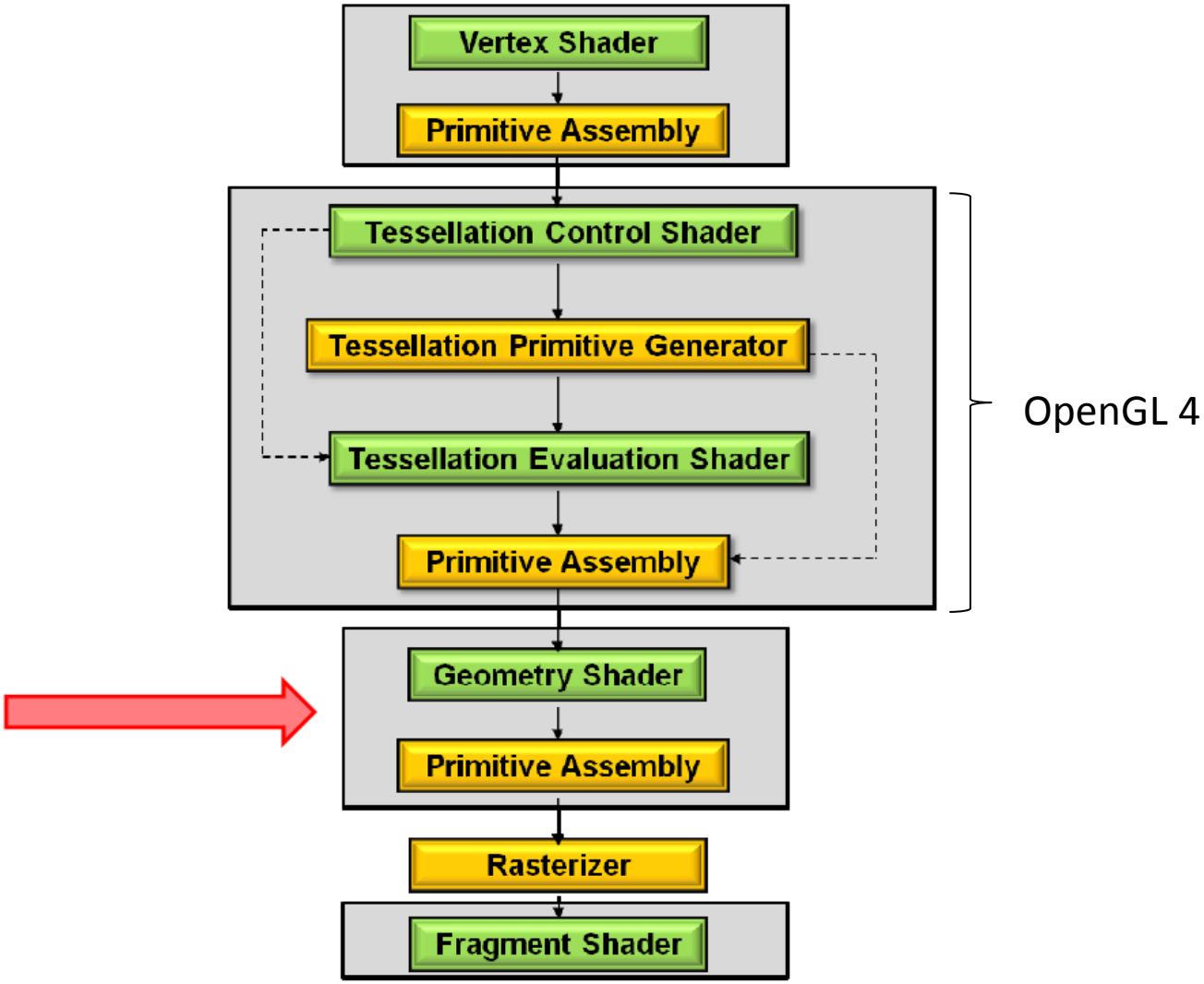
Nov 2015

(*) Basades en el material de Mike Bailey

Introducció

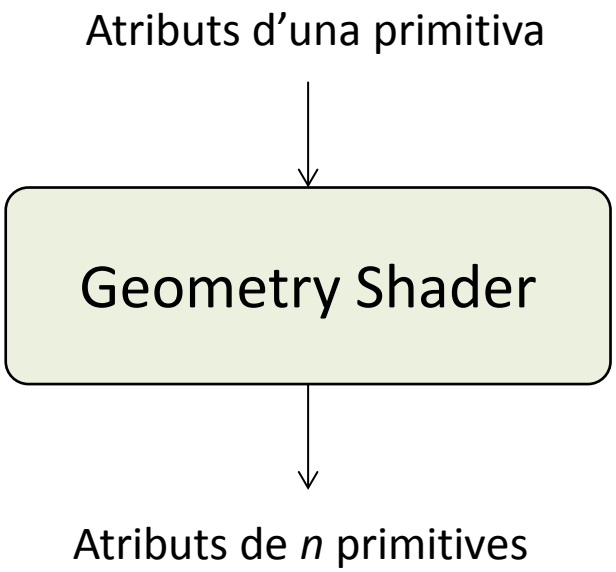
- Els GS processen **primitives** (punts, línies, triangles)
- Ofereixen la possibilitat de **crear noves primitives** i de canviar-ne la **topologia** (exemple: punt → triangle)
- Disponibles a partir d'OpenGL 2.1, GLSL 1.20.

Situació al pipeline



ENTORN D’EXECUCIÓ DEL GS

Entrades i sortides



Entrades i sortides

gl_in[i].gl_Position

in vec4 vfrontColor[]
in vec3 vnormal[]
in vec2 vtexCoord[]
…

Atributs dels vèrtexs de la primitiva (del VS)

Geometry Shader

Sortides per vertex

gl_Position

out vec4 gfrontColor;
out vec3 gnormal;
out vec2 gtexCoord;

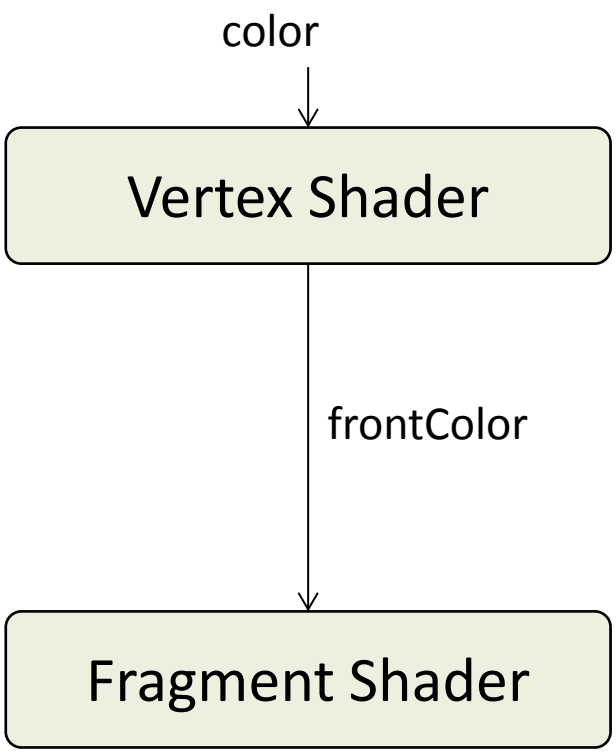
EmitVertex() — Emet un vèrtex amb els atributs actuals

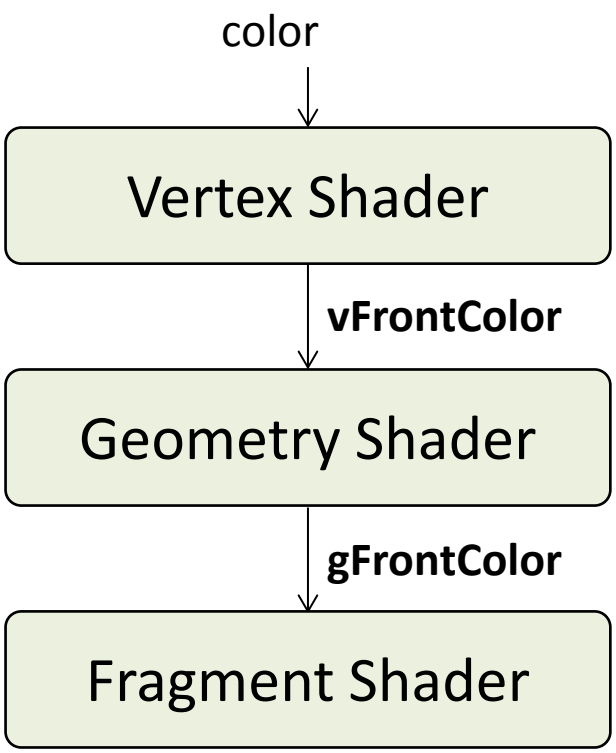
EndPrimitive() — Emet una primitiva amb el vèrtexs actuals

Exemple minimalista GS

```
#version 330 core
layout(triangles) in;
layout(triangle_strip, max_vertices = 36) out;

void main(void){
  for( int i = 0 ; i < 3 ; i++ )
  {
    gl_Position = gl_in[i].gl_Position;
    EmitVertex();
  }
  EndPrimitive();
}
```

Exemple


color
Vertex Shader
frontColor
Fragment Shader


color
Vertex Shader
vFrontColor
Geometry Shader
gFrontColor
Fragment Shader

Shaders per defecte: VS

```
// default.vert
#version 330 core
layout (location = 0) in vec3 vertex;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec3 color;
layout (location = 3) in vec2 texCoord;
out vec4 frontColor;
…
void main(){
  vec3 N = normalize(normalMatrix * normal);
  frontColor = vec4(color,1.0) * N.z;
  gl_Position = modelViewProjectionMatrix *
vec4(vertex.xyz, 1.0);
}
```

```
// default.vert
#version 330 core
layout (location = 0) in vec3 vertex;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec3 color;
layout (location = 3) in vec2 texCoord;
out vec4 vfrontColor;
…
void main(){
  vec3 N = normalize(normalMatrix * normal);
  vfrontColor = vec4(color,1.0) * N.z;
  gl_Position = modelViewProjectionMatrix *
vec4(vertex.xyz, 1.0);
}
```

Shaders per defecte: GS

```
// default.geom
```

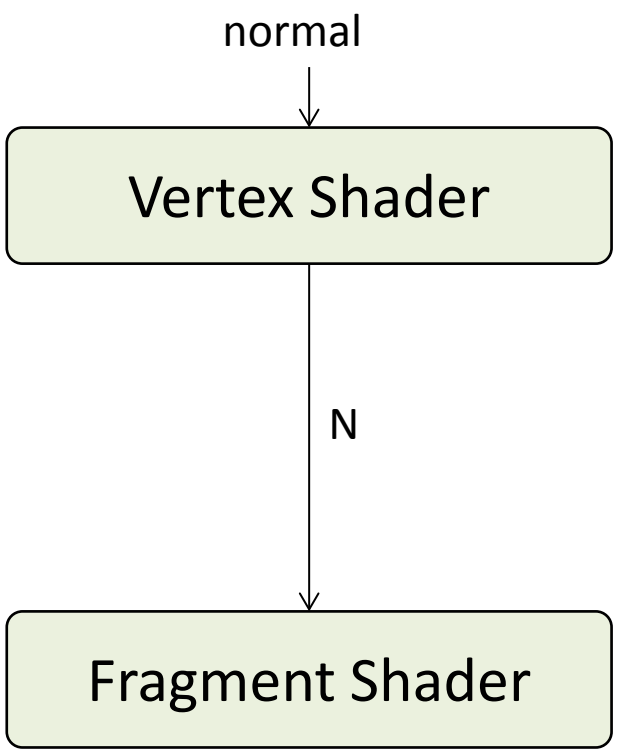
```
// default.geom
#version 330 core
layout(triangles) in;
layout(triangle_strip, max_vertices = 36) out;
in vec4 vfrontColor[];
out vec4 gfrontColor;
void main( void ){
  for( int i = 0 ; i < 3 ; i++ )
  {
    gfrontColor = vfrontColor[i];
    gl_Position = gl_in[i].gl_Position;
    EmitVertex();
  }
  EndPrimitive();
}
```

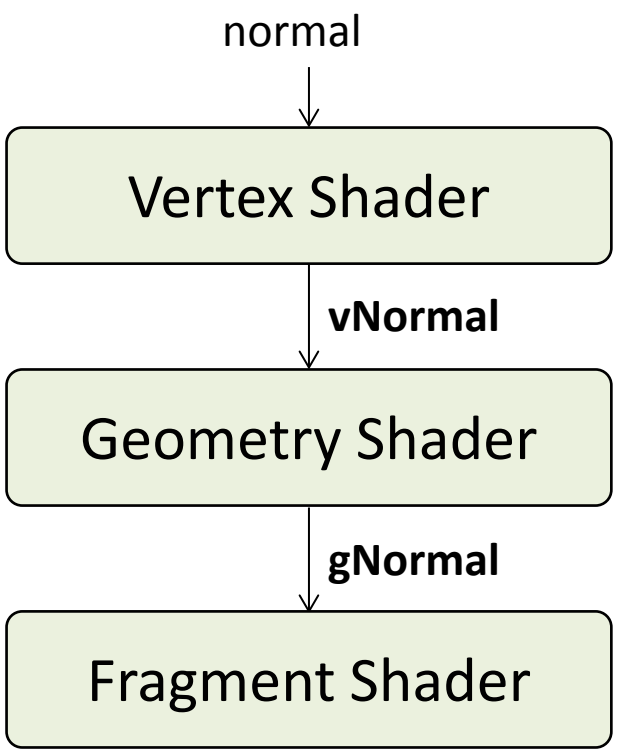
Shaders per defecte: FS

```
// default.frag
#version 330 core
in vec4 frontColor;
out vec4 fragColor;

void main()
{
  fragColor = frontColor;
}
```

```
// default.frag
#version 330 core
in vec4 gfrontColor;
out vec4 fragColor;

void main()
{
  fragColor = gfrontColor;
}
```

Il·luminació per fragment amb GS



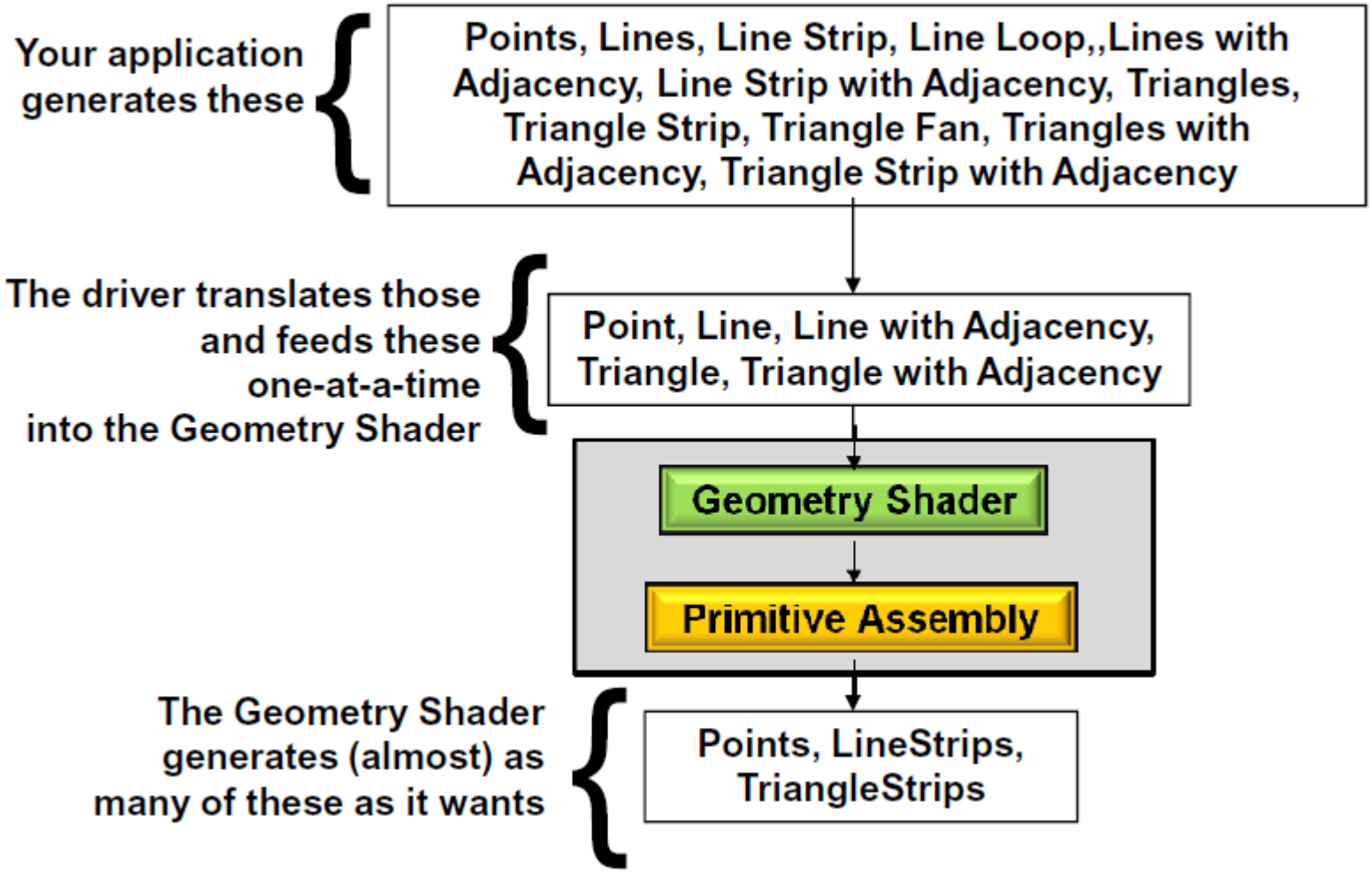


Observacions

- Si useu GS, els **out** del VS només arribaràn al FS si el GS els hi ha passat.
- No hi ha cap **BeginPrimitive**(); és implícit
- Es recomana cridar **EndPrimitive**() al final de cada primitiva (tot i que la darrera crida és implícita).

TIPUS DE PRIMITIVES

Primitives


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 those and feeds these one-at-a-time into the Geometry Shader
Point, Line, Line with Adjacency, Triangle, Triangle with Adjacency
Geometry Shader
Primitive Assembly
The Geometry Shader generates (almost) as many of these as it wants
Points, LineStrips, TriangleStrips

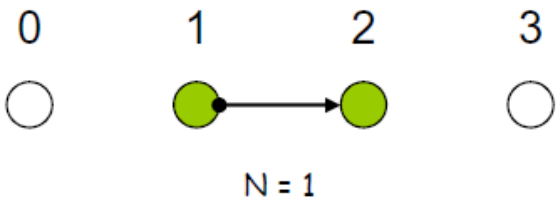
Primitives que envia l’aplicació

Primitives (glBegin…):

- GL_POINT
- GL_TRIANGLES
- …
- **GL_LINES_ADJACENCY**
- **GL_LINE_STRIP_ADJACENCY**
- **GL_TRIANGLES_ADJACENCY**
- **GL_TRIANGLE_STRIP_ADJECENCY**

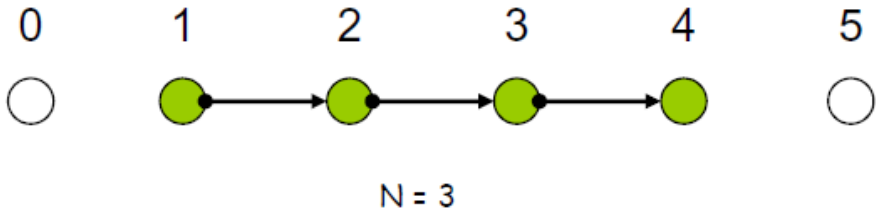
Adjacències - línies

Lines with Adjacency



4N vertices are given.
(where N is the number of line segments to draw).
A line segment is drawn between #1 and #2.
Vertices #0 and #3 are there to provide adjacency information.

Line Strip with Adjacency

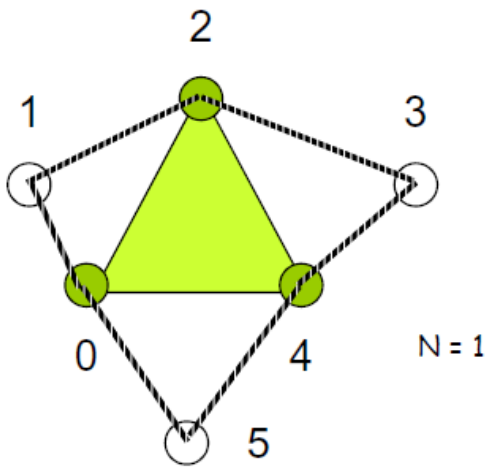


N+3 vertices are given
(where N is the number of line segments to draw).
A line segment is drawn between #1 and #2, #2 and #3, ..., #N and #N+1.
Vertices #0 and #N+2 are there to provide adjacency information.

Adjacències - triangles

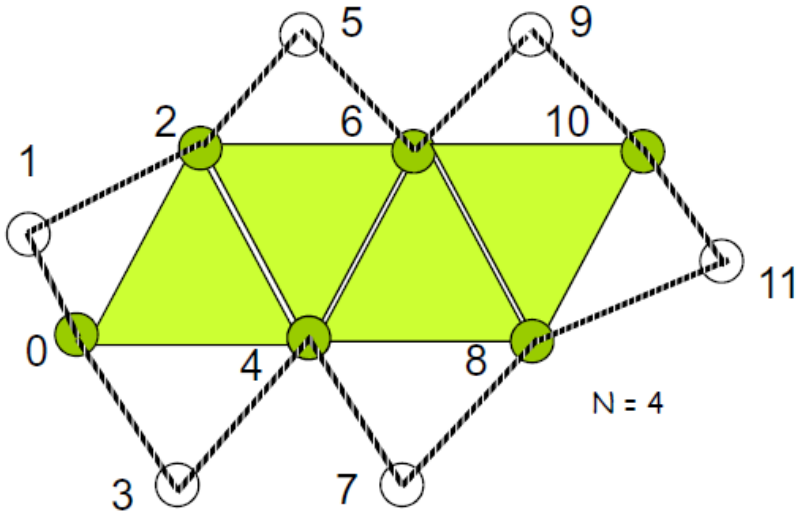
Triangles with Adjacency

6N vertices are given
(where N is the number of triangles to draw).
Points 0, 2, and 4 define the triangle.
Points 1, 3, and 5 tell where adjacent triangles are.



Triangle Strip with Adjacency

4+2N vertices are given
(where N is the number of triangles to draw).
Points 0, 2, 4, 6, 8, 10, …define the triangles.
Points 1, 3, 5, 7, 9, 11, … tell where adjacent triangles are.



Número de vèrtexs

Número de vèrtexs que rep el GS:

- GL_POINTS → 1
- GL_LINES → 2
- GL_TRIANGLES → 3
- **GL_LINES_ADJACENCY → 4**
- **GL_TRIANGLES_ADJACENCY → 6**

Geometry Language

in gl_PerVertex {
vec4 gl_Position;
float gl_PointSize;
float gl_ClipDistance[];
} gl_in[];
 	
in int gl_PrimitiveIDIn;
	out gl_PerVertex {
	vec4 gl_Position;
	float gl_PointSize;
	float gl_ClipDistance[];
	};
	
	out int gl_PrimitiveID;
	out int gl_Layer;

Primitives que pot crear un GS

Un GS només pot generar:

- Punts (GL_POINTS)
- Segments (GL_LINE_STRIP)
- Triangles (GL_TRIANGLE_STRIP)