

Geometry shaders (GLSL 1.20)

C. Andújar (*)

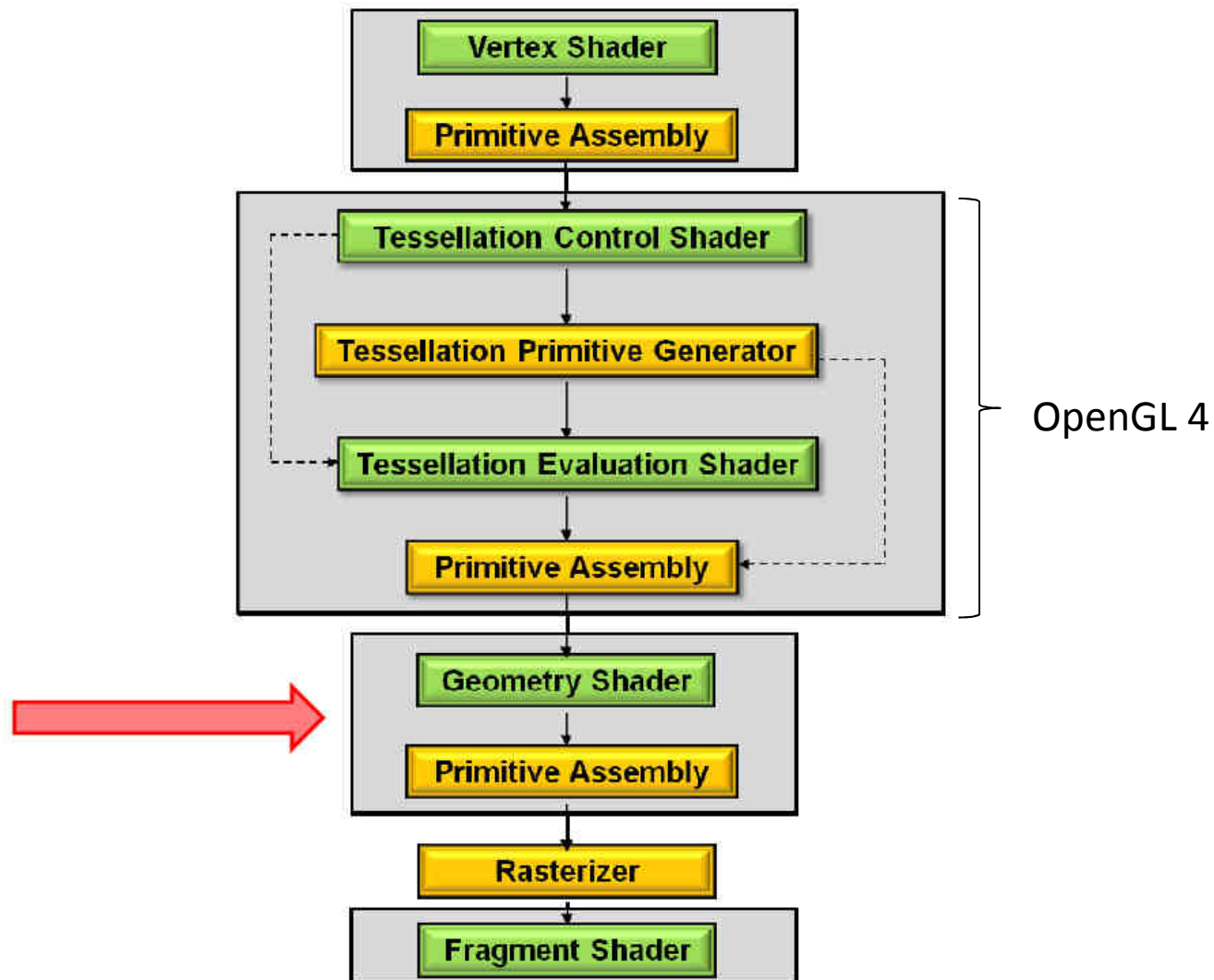
Abril 2012

(*) Basades en el material de Mike Bailey

Introducció

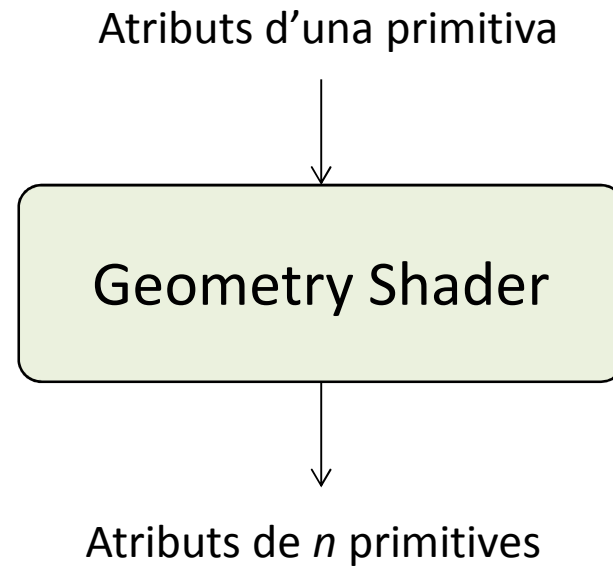
- Els GS processen **primitives** (punts, línies, triangles)
- Ofereixen la possibilitat de **crear noves primitives** i de canviar-ne la **topologia** (exemple: punt → triangle)
- Disponibles a partir d'OpenGL 2.1, GLSL 1.20.

Situació al pipeline

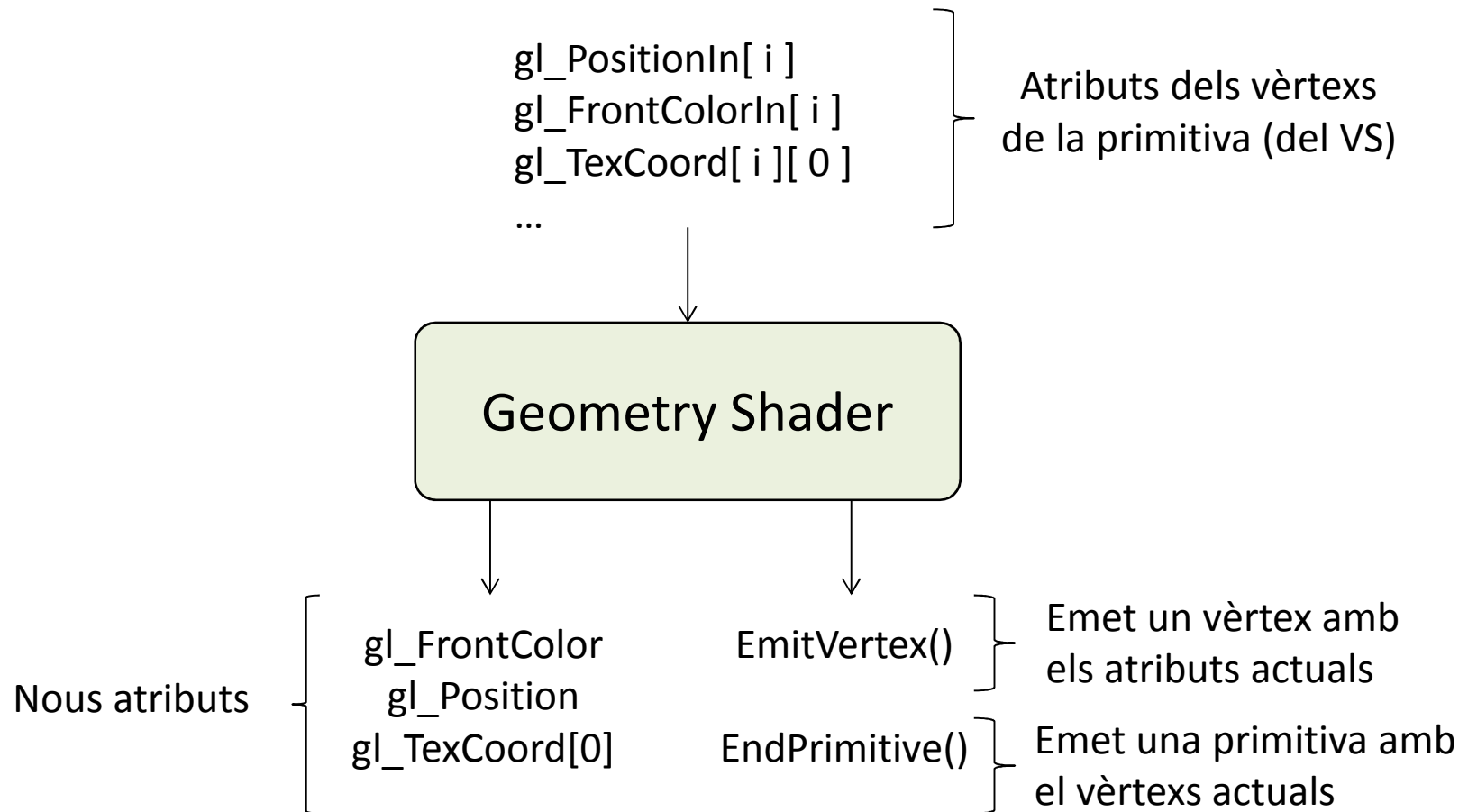


ENTORN D'EXECUCIÓ DEL GS

Entrades i sortides



Entrades i sortides



Reproducció del pipeline fix

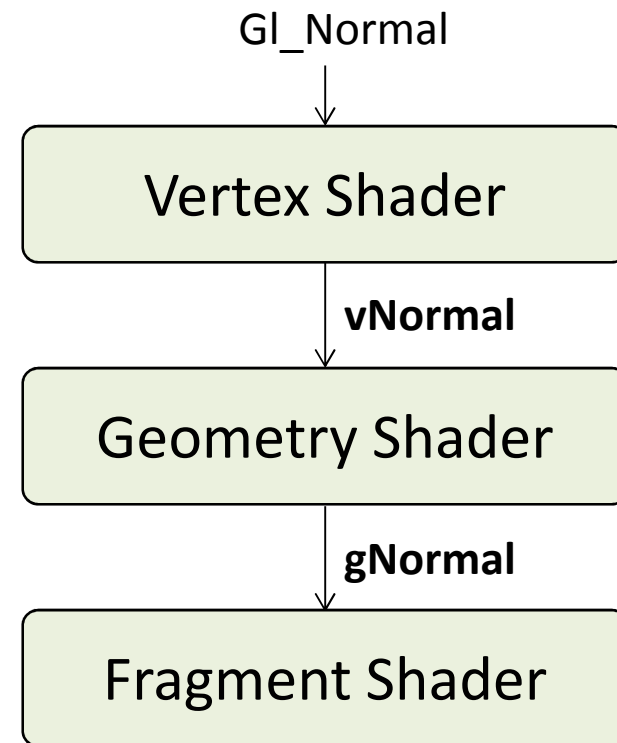
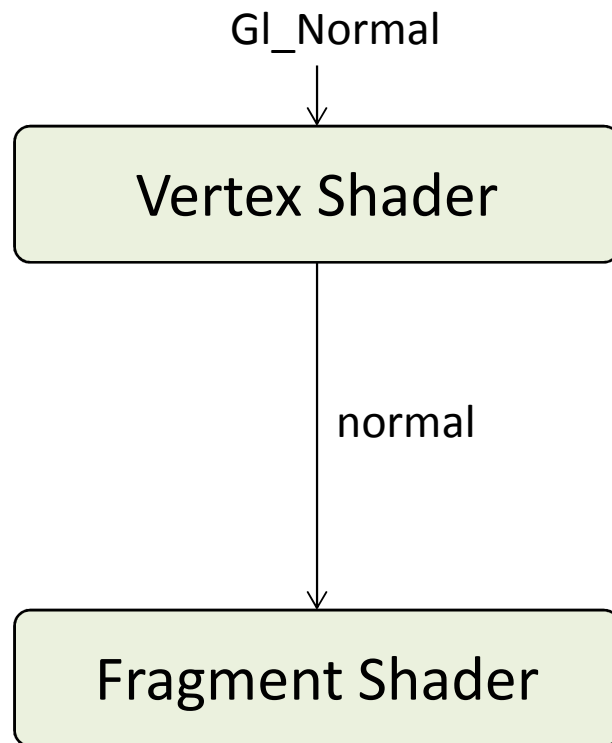
// these lines enable the geometry shader support.

#version 120

#extension GL_EXT_geometry_shader4 : enable

```
void main( void )  
{  
    for(int i=0; i < gl_VerticesIn; i++)  
    {  
        gl_FrontColor = gl_FrontColorIn[i];  
        gl_Position    = gl_PositionIn  [i];  
        gl_TexCoord[0] = gl_TexCoordIn  [i][0];  
        EmitVertex();  
    }  
    EndPrimitive(); // implicit  
}
```

Exemple: basic lighting v3 amb GS



Exemple: basic lighting v3 amb GS

// basic-lighting-2.vert

```
varying vec3 normal;

void main() {
    normal = gl_NormalMatrix * gl_Normal;
    gl_Position = gl_ModelViewProjectionMat...
    gl_FrontColor = gl_Color;
}
```

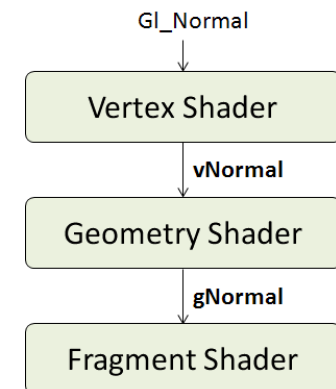
Del VS → FS

// basic-lighting-3.vert

```
varying vec3 vNormal;

void main() {
    vNormal = gl_NormalMatrix * gl_Normal;
    gl_Position = gl_ModelViewProjectionMat...
    gl_FrontColor = gl_Color;
}
```

Del VS → GS



Exemple: basic lighting v3 amb GS

// basic-lighting-2.geom

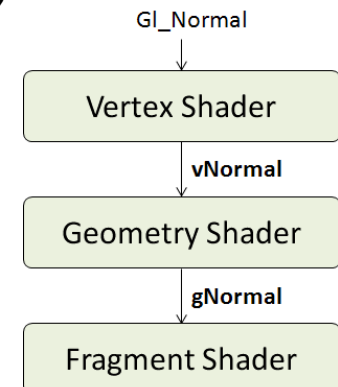
Del VS → GS

Del GS → FS

// basic-lighting-3.geom

```
#version 120
#extension GL_EXT_geometry_shader4 : enable
varying in vec3 vNormal[3];
varying out vec3 gNormal;

void main( void )
{
    for( int i = 0 ; i < gl_VerticesIn ; i++ )
    {
        gNormal = vNormal[i];
        gl_FrontColor = gl_FrontColorIn[i];
        gl_Position  = gl_PositionIn[i];
        EmitVertex();
    }
    EndPrimitive();
}
```



Exemple: basic lighting v3 amb GS

// basic-lighting-2.frag

varying vec3 **normal**;

void main()

{

gl_FragColor = **normal.z** * gl_Color;

}

// basic-lighting-3.frag

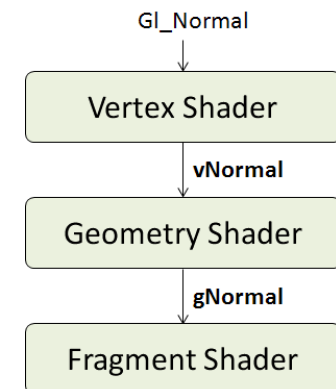
varying vec3 **gNormal**;

void main()

{

gl_FragColor = **gNormal.z** * gl_Color;

}



Observacions

- Si useu GS, els varying's del VS només arribaran al FS si el GS els ha copiat.
- No hi ha cap `BeginPrimitive()`; és implícit
- No cal cridar `EndPrimitive()` al final del GS; també és implícit.

Exemple: encogir els triangles

```
#version 120
```

```
#extension GL_EXT_geometry_shader4 : enable
```

```
in vec3 vNormal[];
```

```
varying vec3 gNormal;
```

```
uniform float shrinkFactor; // ex. 0.5
```

```
void main( void ) {
```

```
    vec4 center = (gl_PositionIn[0] + gl_PositionIn[1] + gl_PositionIn[2])/3.0;
```

```
    for( int i = 0 ; i < gl_VerticesIn ; i++ )
```

```
    {
```

```
        gNormal = vNormal[i];
```

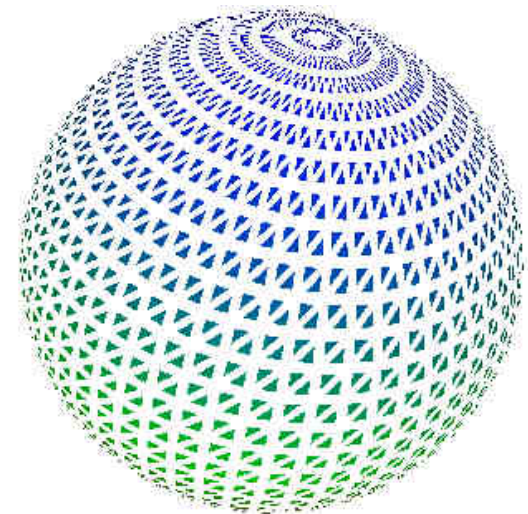
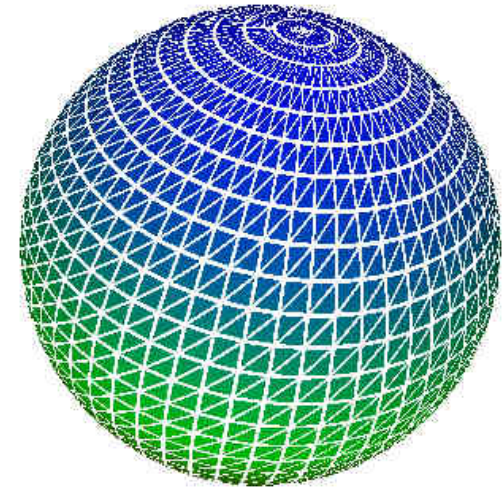
```
        gl_FrontColor = gl_FrontColorIn[i];
```

```
        gl_Position = mix(gl_PositionIn[i], center, shrinkFactor);
```

```
        EmitVertex();
```

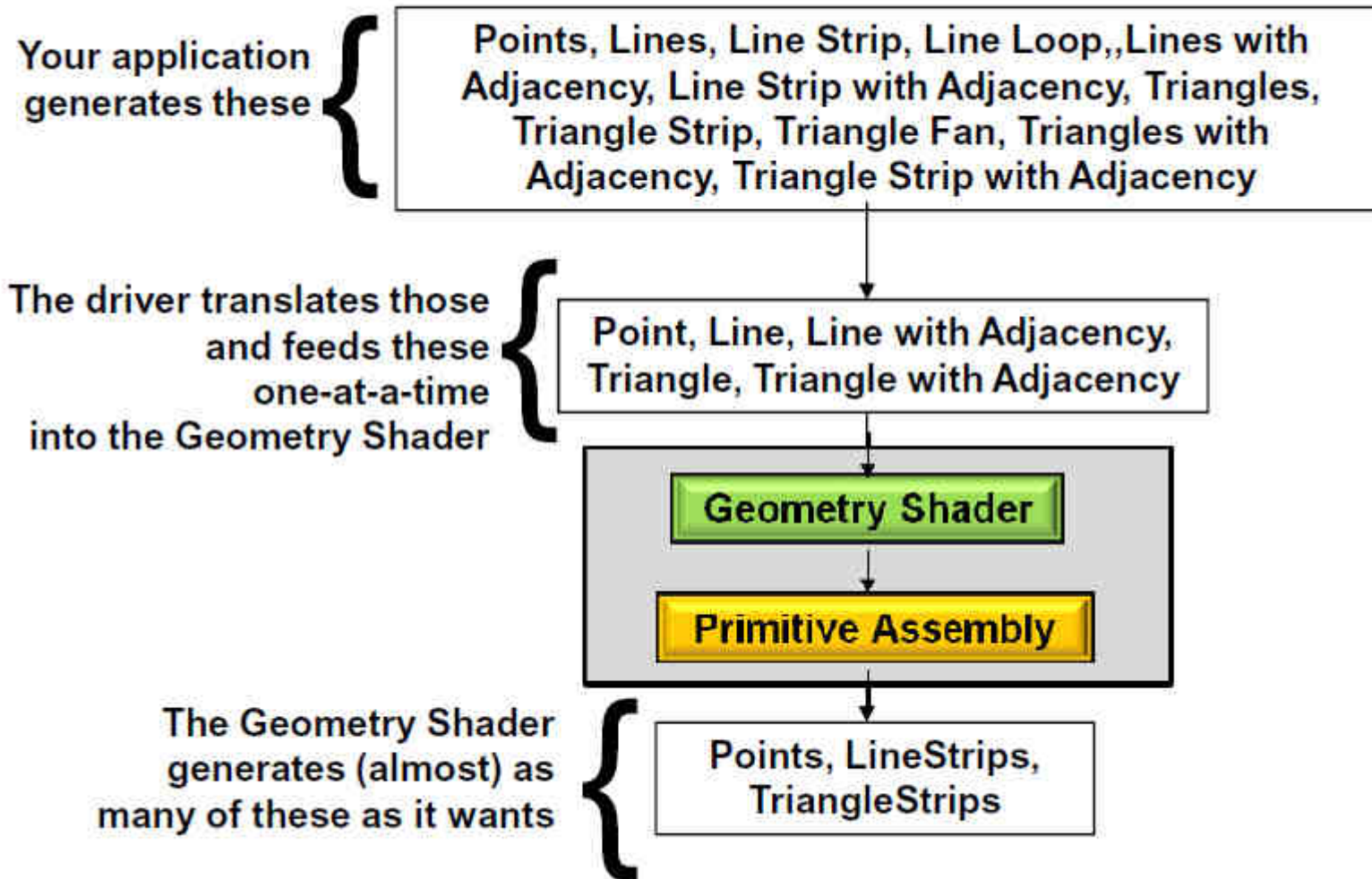
```
    }
```

```
}
```



TIPUS DE PRIMITIVES

Primitives



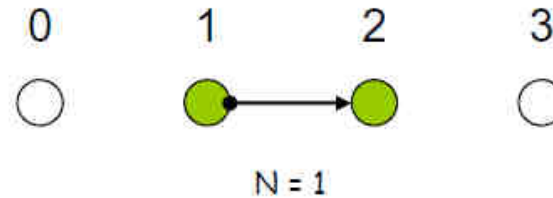
Primitives que envia l'aplicació

Primitives (glBegin...):

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

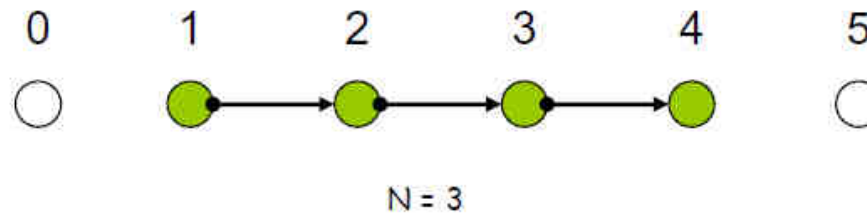
Adjacències - línies

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

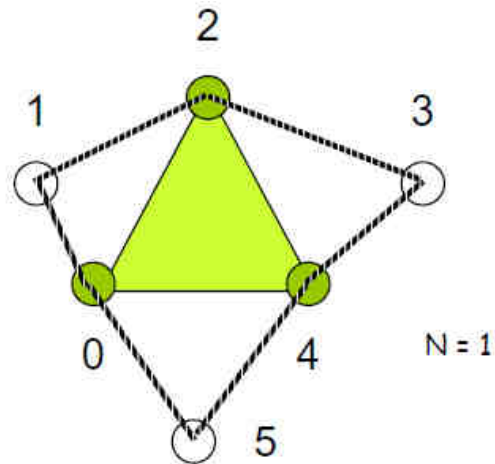


N+3 vertices are given
(where N is the number of line segments to draw).
A line segment is drawn between #1 and #2, #2 and #3, ..., #N and #N+1.
Vertices #0 and #N+2 are there to provide adjacency information.

Adjacències - triangles

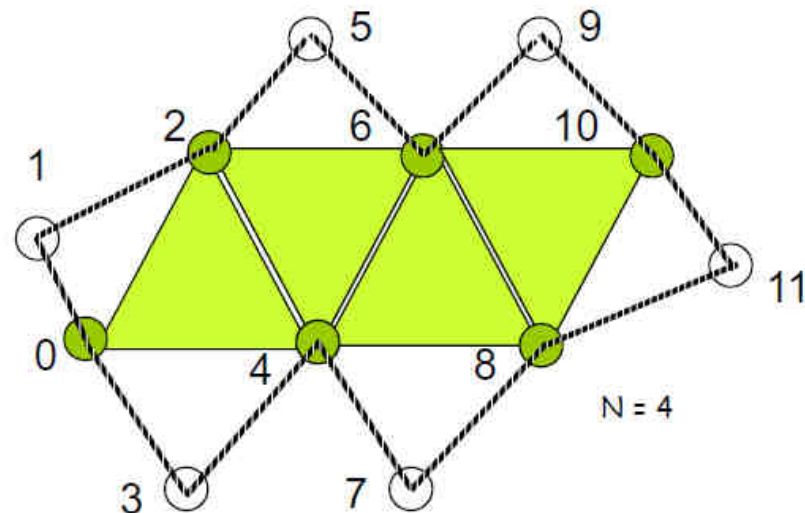
Triangles with Adjacency

$6N$ 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.



Adjacències – gl_VerticesIn

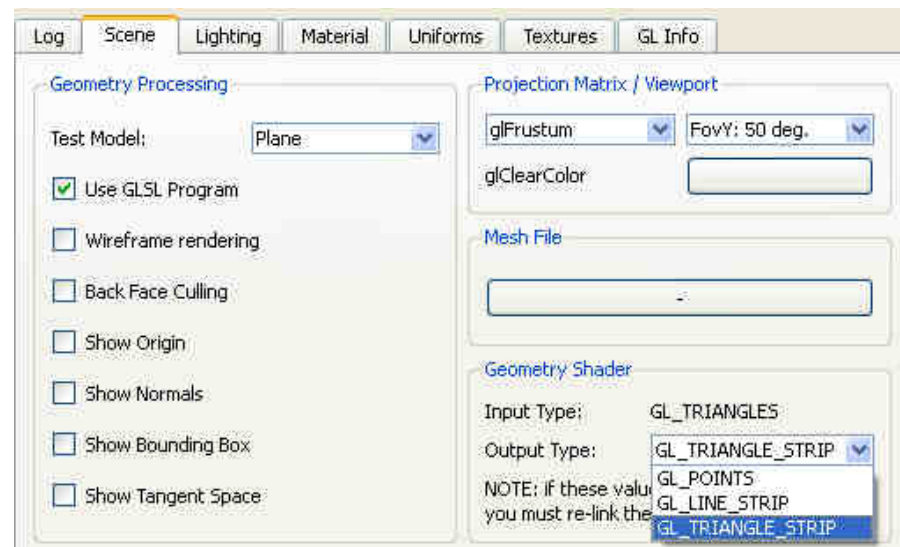
Número de vèrtexs que rep el GS:

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

Primitives que pot crear un GS

Un GS només pot generar:

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



Exemple: explosió

Idea:

- Aplicar una translació variable a les cares.
- Eüació bàsica MUA: $\mathbf{r} = \mathbf{r}_0 + \mathbf{v}_0 t + \frac{1}{2} \mathbf{a} t^2$
- \mathbf{r}_0 serà la posició del centre
- \mathbf{v}_0 serà el vector posició del centre

Exemple: explosió

```
#version 120
#extension GL_EXT_geometry_shader4 : enable
uniform float time;
const vec3 a= vec3(0.0, -9.8, 0.0); // gravity
uniform float speed; // eg. 2.5 (glass.obj)
void main( void ) {
    float t = mod(time, 3.0); // repeat every 3 seconds
    vec4 center = (gl_PositionIn[0] + gl_PositionIn[1] + gl_PositionIn[2])/3.0;
    vec3 v0 = speed * center.xyz;
    vec4 trans = vec4(v0*t + 0.5*a*t*t, 0.0);
    for( int i = 0 ; i < gl_VerticesIn ; i++ ) {
        gl_Position = gl_ModelViewProjectionMatrix * (gl_PositionIn[i] + trans);
        EmitVertex();
    }
}
```