

Laboratori de G

Professors de G

2021

Laboratori

- Primera meitat: shaders (GLSL)
- Segona meitat: shaders (GLSL) + aplicacions (C++, Qt, OpenGL)
- Usarem un visualitzador propi: viewer (aka GLArena)
- Metodologia bàsica:
 - Professor: explicacions (curtes excepte primera sessió de cada meitat)
 - Estudiants: implementació exercicis proposats (“obligatoris”/“opcionals”)
- Material: <https://www.cs.upc.edu/~virtual/G/index.php?dir=2.%20Laboratori/>
- Avaluació: preguntes als exàmens + dos controls de lab.
- Plagiarisme: a tots els actes evaluatius de G, el que lliureu (respostes, codi font...) ha de ser d'**autoria pròpia**. Altrament (còpia d'un altre estudiant, de repositoris externs, d'acadèmies, autoria compartida...) es considera frau.

Vertex shaders i Fragment shaders

Entorn per a desenvolupar shaders (viewer)

Professors de Gràfics

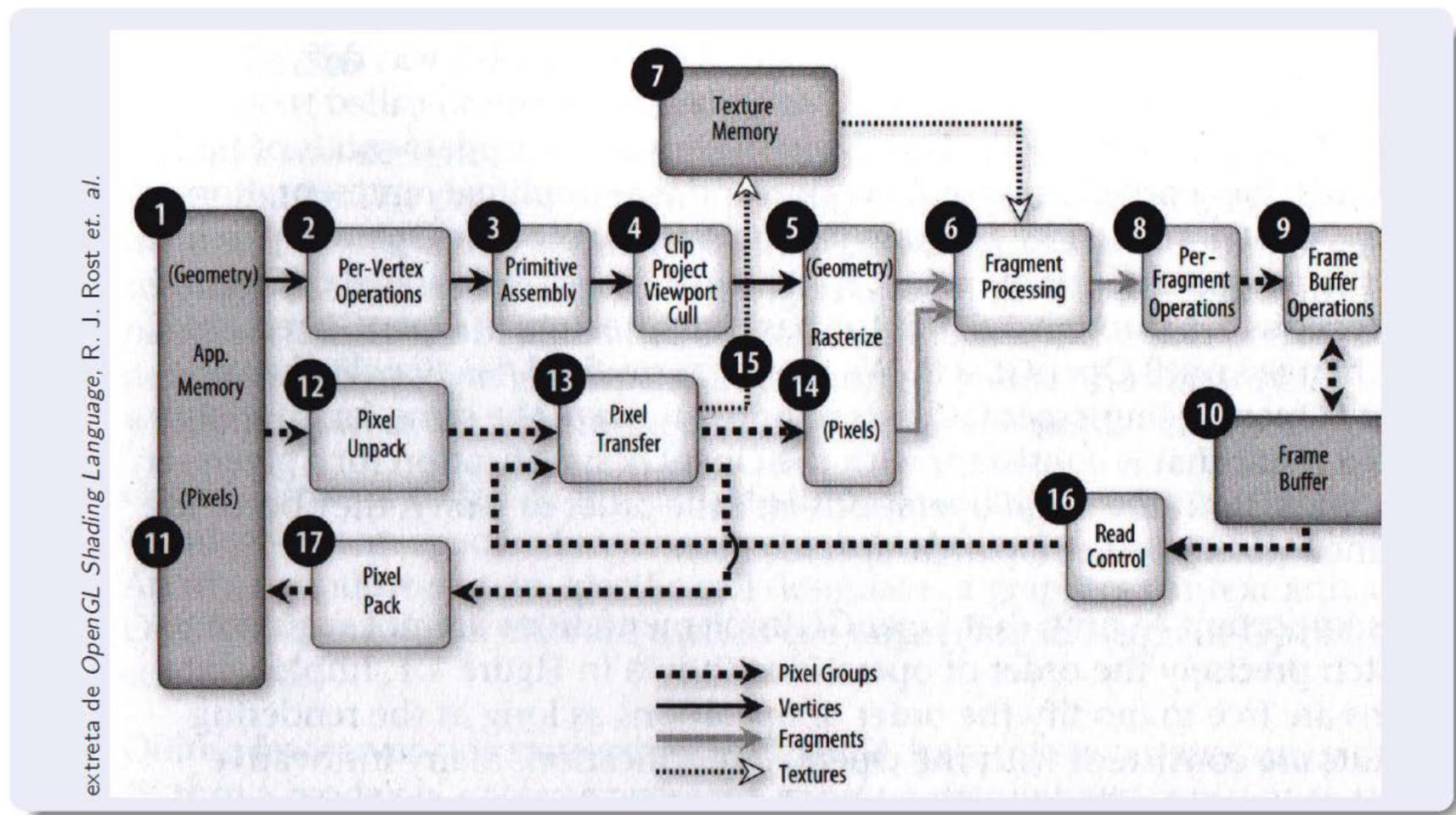
2021

Versions i pipelines d'OpenGL

OpenGL

1.0	(1992)	Primera versió
	...	
2.0	(2004)	VS + FS
	...	
3.0	(2008)	Core profile / Compatibility profile
3.1	(2009)	Millors GLSL
3.2	(2009)	Geometry shaders (GS)
3.3	(2010)	Usada al lab de gràfics
4.0	(2010)	Tessellation shaders (TCS + TES)
	...	
4.3	(2012)	Compute shaders (CS)
	...	
4.6	(2017)	Darrera versió publicada

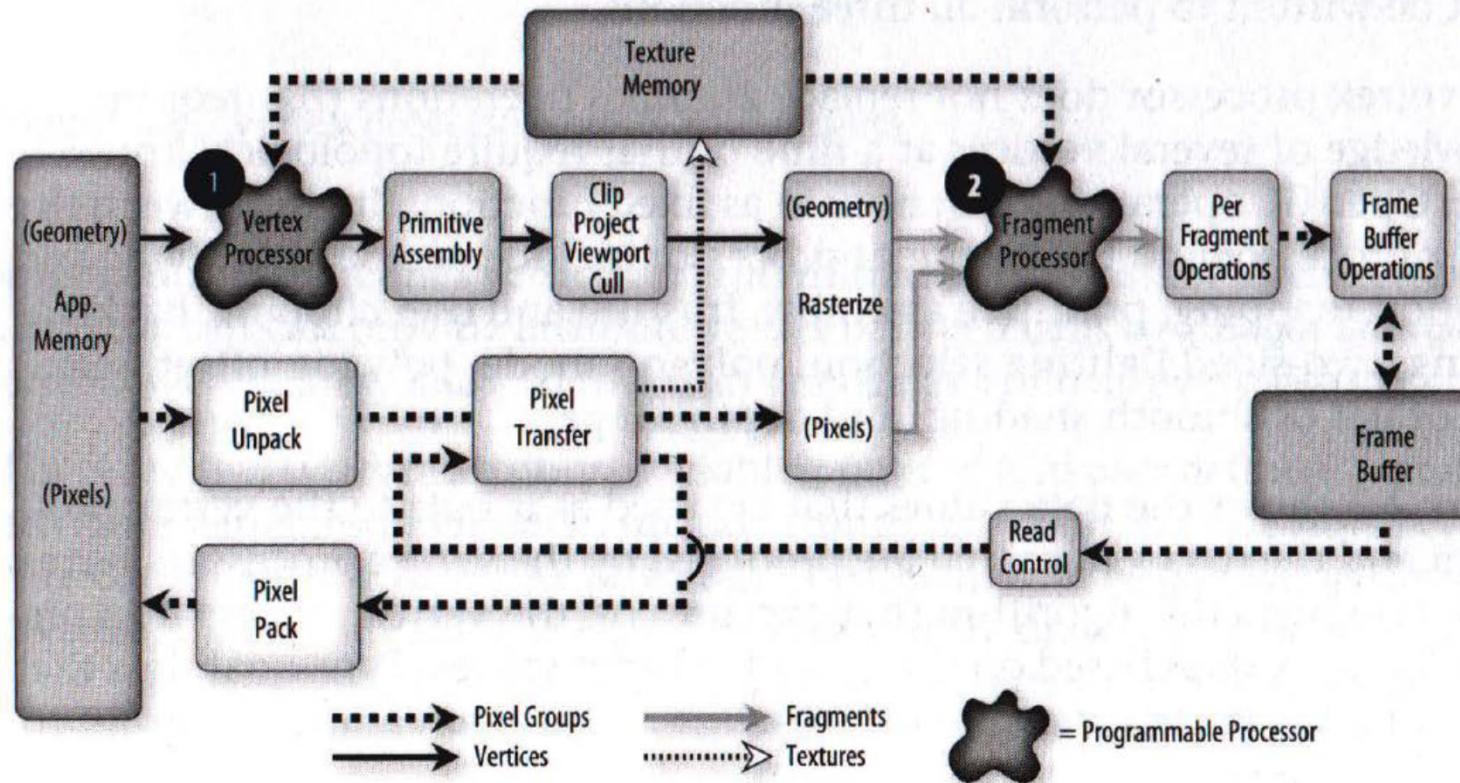
Pipeline fix



Pipeline programmable (VS + FS)

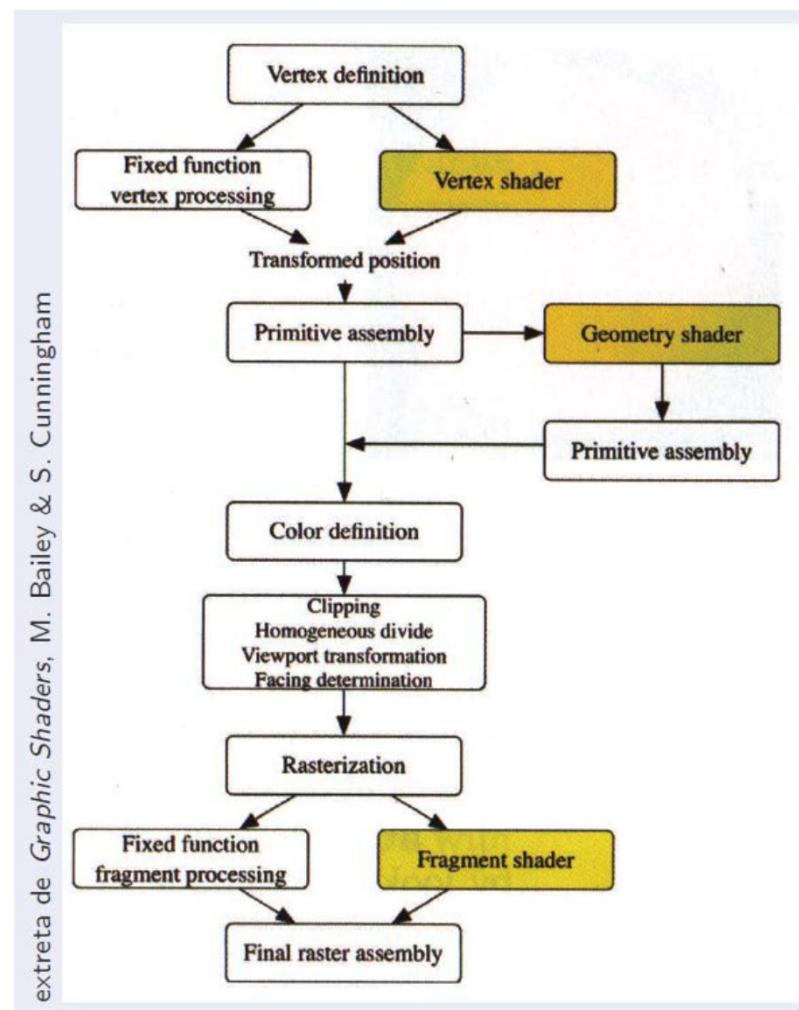
2.0+

extreta de OpenGL Shading Language, R. J. Rost et. al.

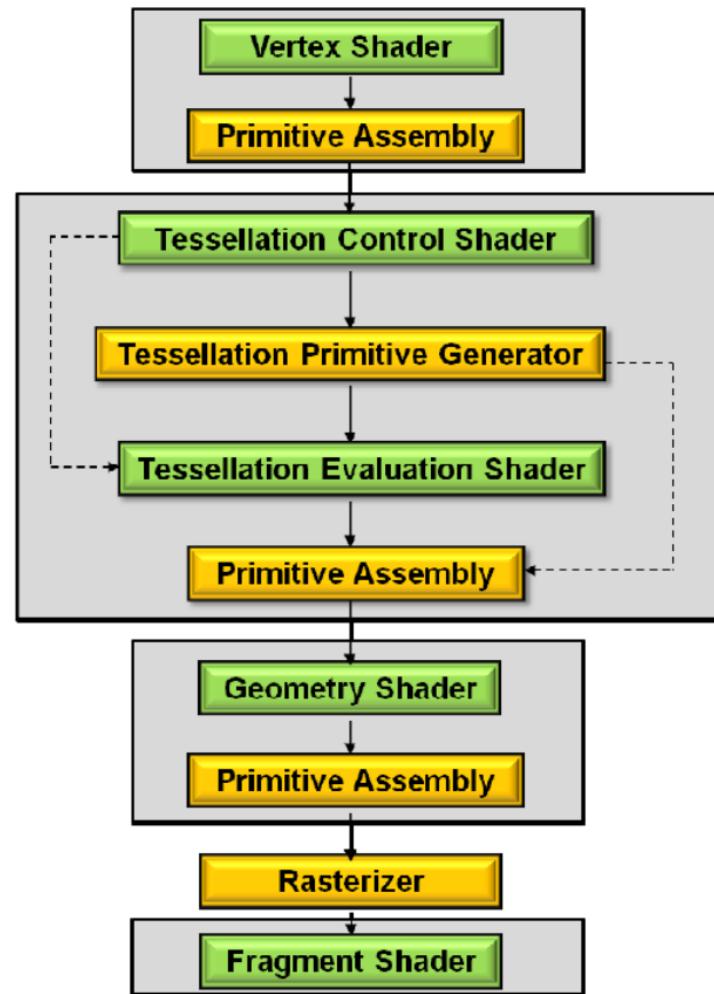


Pipeline programmable (VS + GS + FS)

3.2+

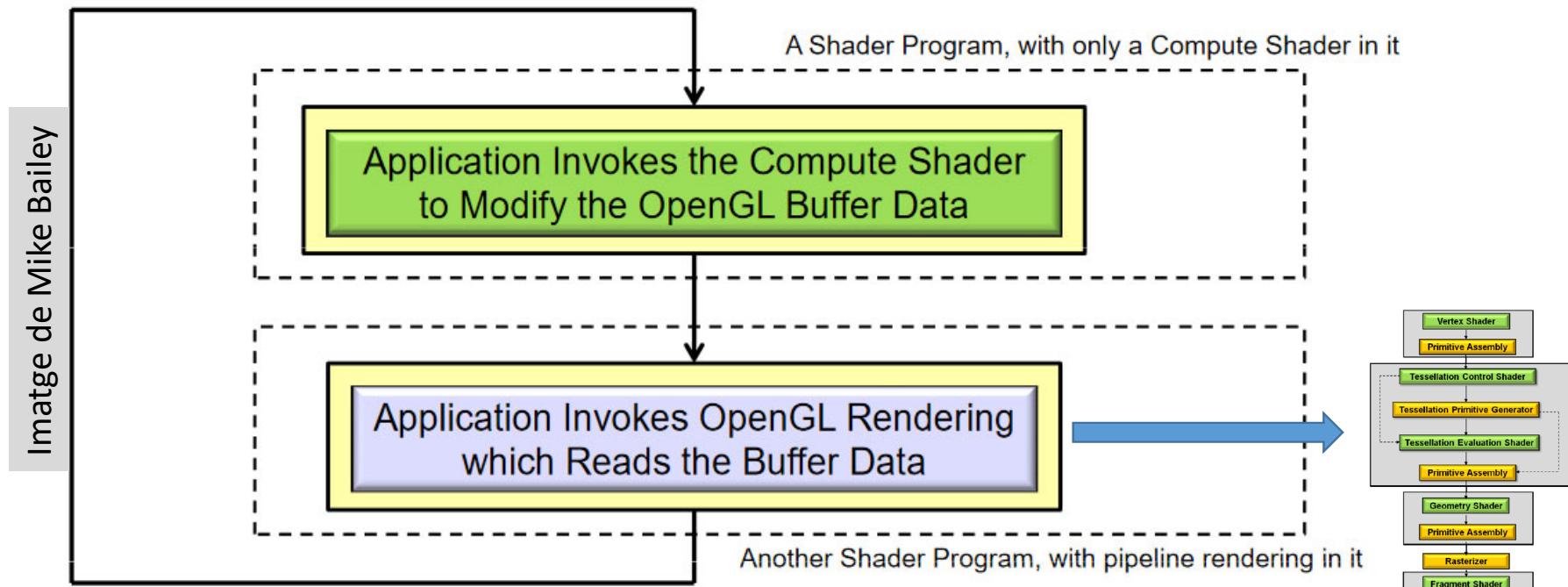


Pipeline programmable (VS + TCS + TES + GS + FS)



4.0+

Pipeline programmable (4.3+)

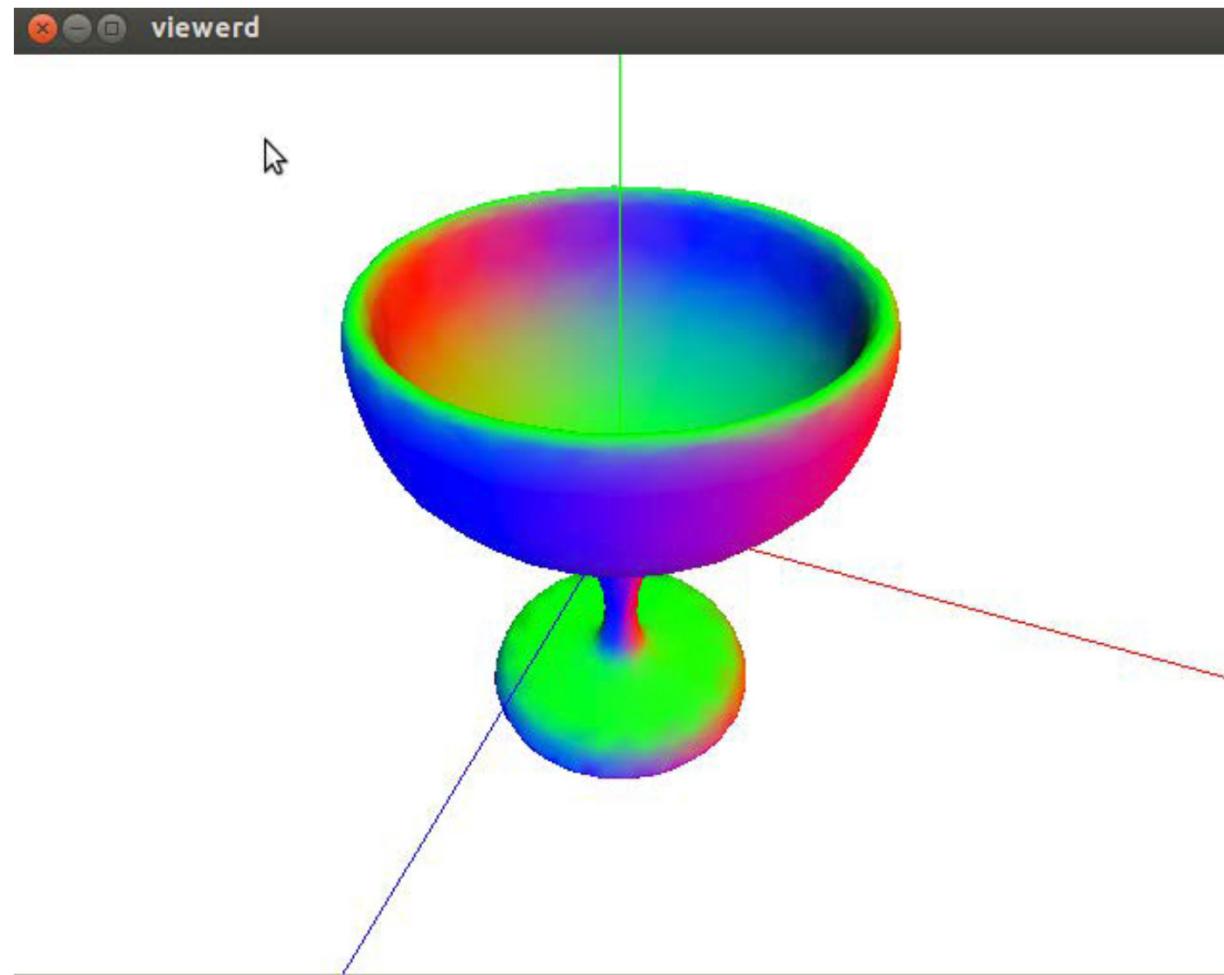


OpenGL

1.0	(1992)	Primera versió
	...	
2.0	(2004)	VS + FS
	...	
3.0	(2008)	Core profile / Compatibility profile
3.1	(2009)	Millors GLSL
3.2	(2009)	Geometry shaders (GS)
3.3	(2010)	Usada al lab de gràfics (VS+FS, VS+GS+FS)
4.0	(2010)	Tessellation shaders (TCS + TES)
	...	
4.3	(2012)	Compute shaders (CS)
	...	
4.6	(2017)	Darrera versió publicada

Viewer

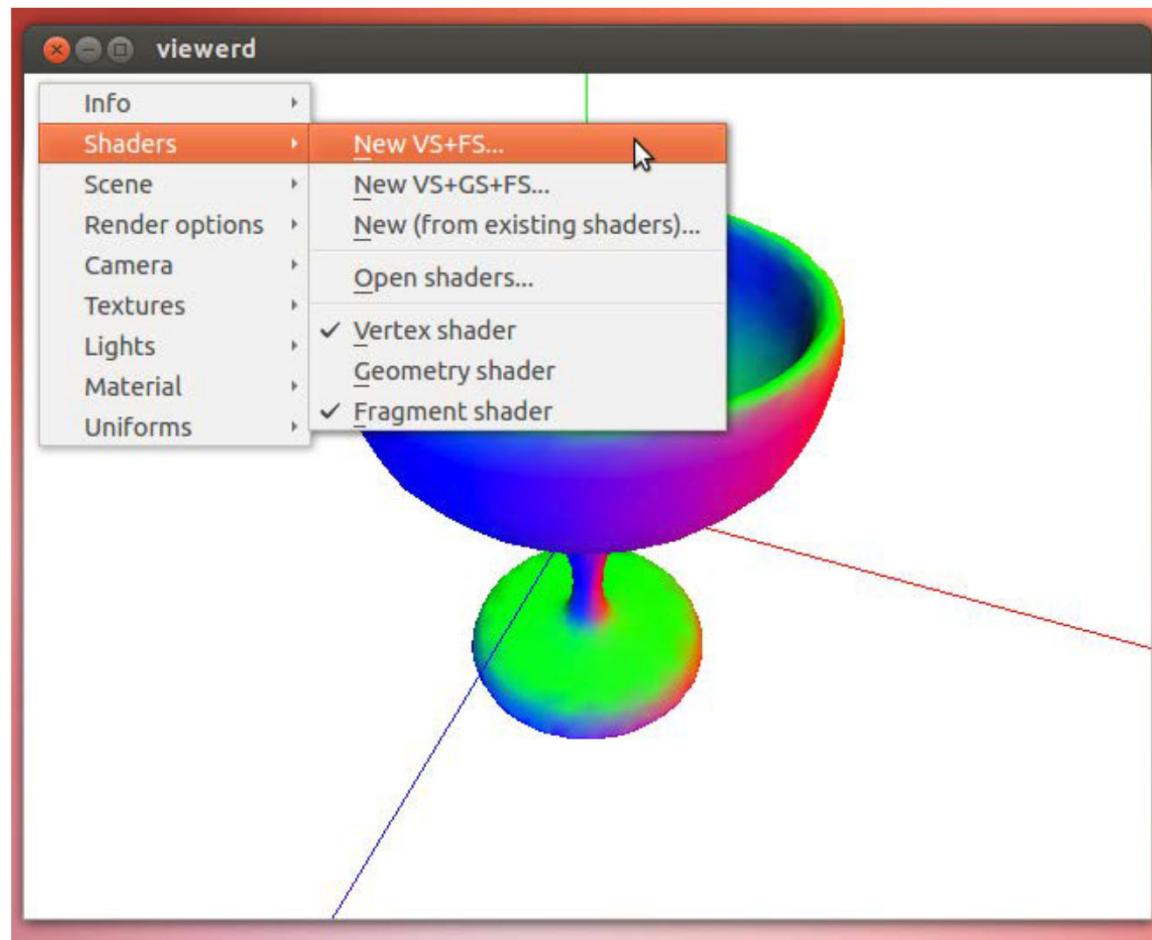
Viewer



Viewer als labs de la FIB

- Funciona en linux
- És recomanable crear una carpeta amb els shaders que anireu creant:
`mkdir shaders` (on vulgueu)
`cd shaders/`
`~/assig/grau-g/Viewer/GLarenaSL`
Premeu [SPACE] per accedir al menú

Viewer



Viewer

The image shows a dual-pane application interface for developing OpenGL shaders and viewing 3D scenes.

Left Panel (gedit): A code editor titled "myShader.vert (~/NewViewer) - gedit" containing GLSL 3.30 vertex shader code. The code defines attributes for position, normal, color, and texture coordinate, and varies frontColor and vtexCoord. It also includes uniforms for modelViewProjectionMatrix and normalMatrix, and a main() function that calculates the normalized normal vector, sets frontColor based on the z-component of the normal, sets vtexCoord to texCoord, and sets gl_Position to the transformed vertex position.

```
1 #version 330 core
2
3 layout (location = 0) in vec3 vertex;
4 layout (location = 1) in vec3 normal;
5 layout (location = 2) in vec3 color;
6 layout (location = 3) in vec2 texCoord;
7
8 out vec4 frontColor;
9 out vec2 vtexCoord;
10
11 uniform mat4 modelViewProjectionMatrix;
12 uniform mat3 normalMatrix;
13
14 void main()
15 {
16     vec3 N = normalize(normalMatrix * normal);
17     frontColor = vec4(color, 1.0) * N.z;
18     vtexCoord = texCoord;
19     gl_Position = modelViewProjectionMatrix * vec4(vertex.xyz, 1.0);
20 }
```

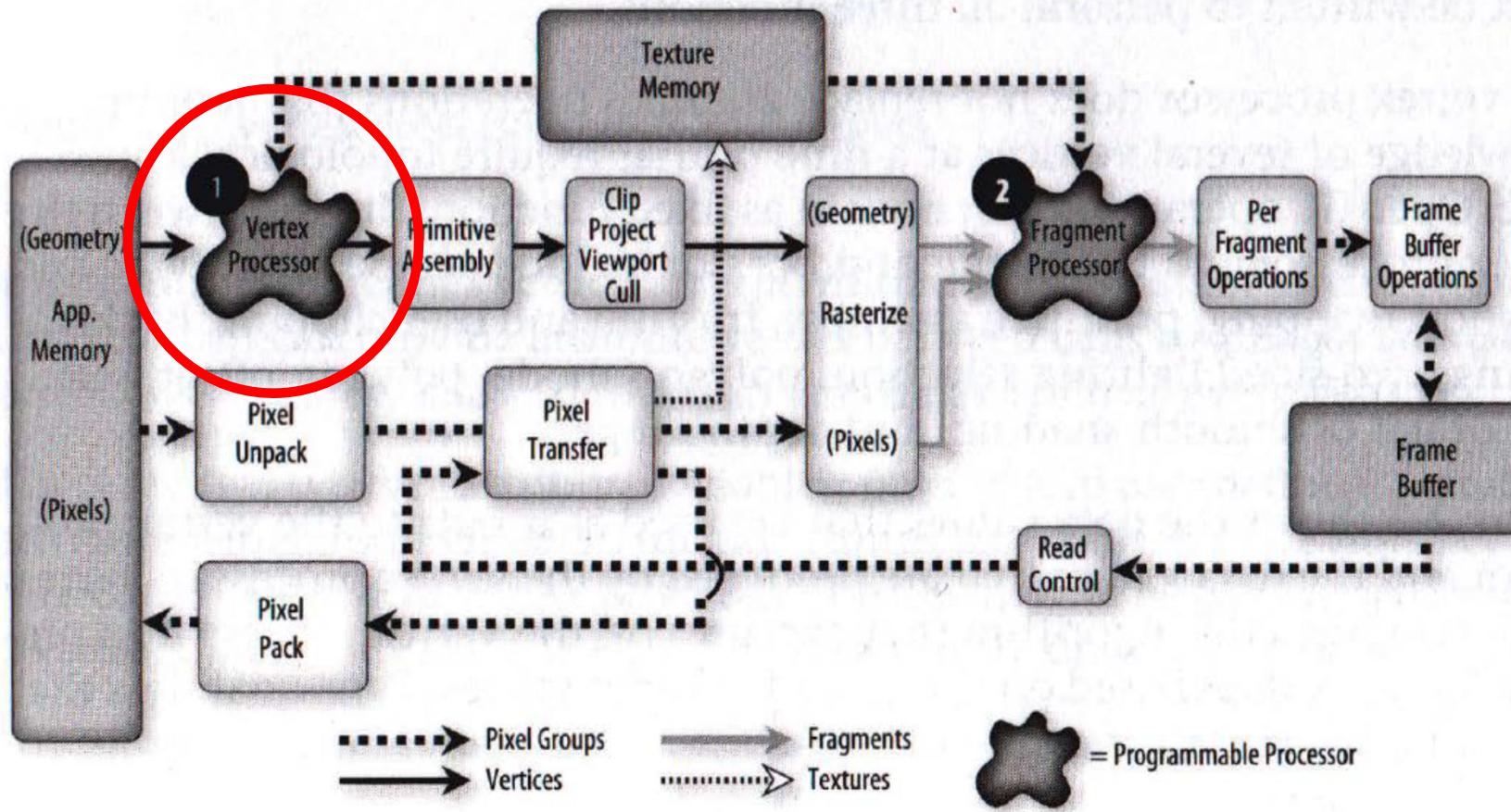
Bottom status bar: GLSL 3.30, Ancho de la tabulación: 4, Ln 1, Col 1, INS.

Right Panel (viewerd): A 3D rendering window titled "viewerd". The menu bar is open, showing "Info" as the active item, along with "Shaders", "Scene", "Render options", "Camera", "Textures", "Lights", "Material", and "Uniforms". A tooltip provides build information: Current path is /home/andujar/NewViewer, VS: myShader.vert [COMPILE: OK], GS: myShader.geom [DISABLED], FS: myShader.frag [COMPILE: OK], and PROGRAM LINK: OK. The 3D scene displays a teardrop-shaped object with a gradient color from purple at the top to green at the base, resting on a green circular plane.

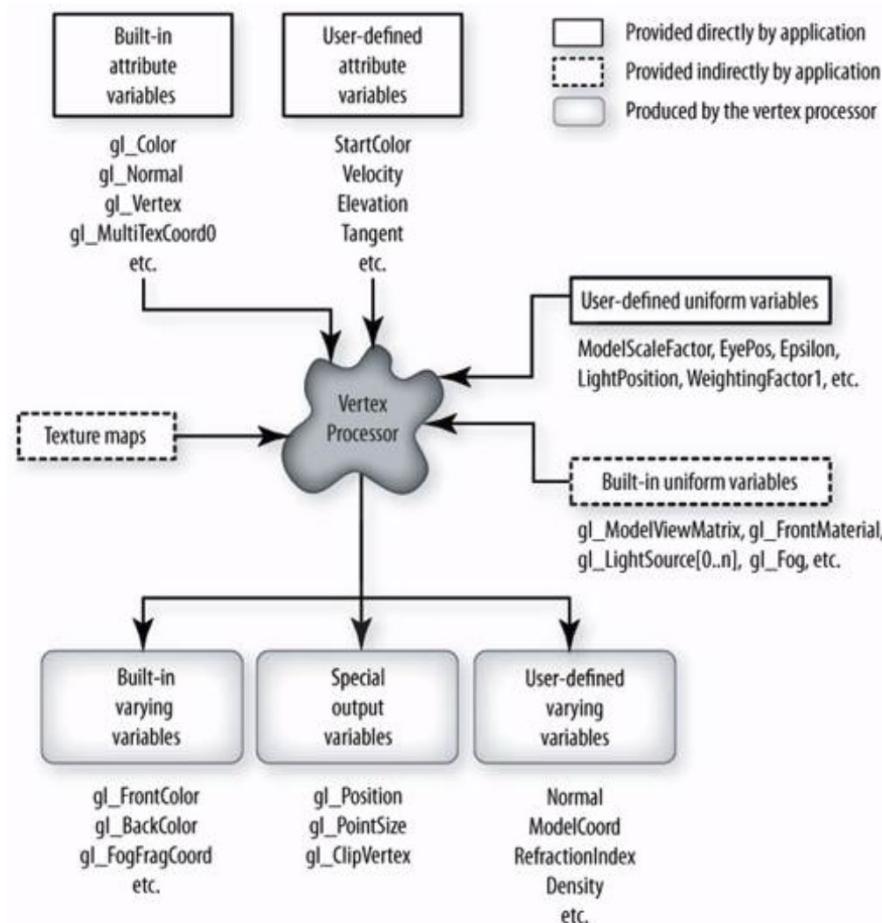
Configuració de gedit als labs de la FIB

- Activar syntax highlighting per GLSL 3.30:
`~/assig/grau-g/gedit-config`
- Activar el plugin “snippets” del gedit (**Preferences-Plugins-Snippets**)
Fa que **defs[TAB]** s’expandeixi a les declaracions de tots els uniforms que envia el viewer

Vertex shaders



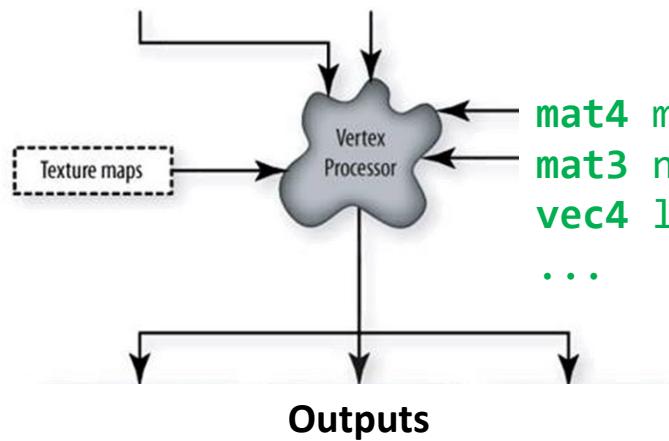
VS (compatibility profile)



VS (3.3 core profile)

Attributes (user-defined)

```
vec3 vertex;    // object space  
vec3 normal;   // object space  
vec3 color;  
vec2 texCoord;  
...  
...
```



Uniforms (user-defined, read-only)

```
mat4 modelViewMatrix;  
mat3 normalMatrix;  
vec4 lightAmbient;  
...  
...
```

Outputs

```
vec4 gl_Position; // predefined; usualment en clip space  
vec4 frontColor;
```

VS: entrades

Atributs definits pel viewer (cal declarar-los al VS):

```
layout(location= 0) in vec3 vertex;      // object space  
layout(location= 1) in vec3 normal;      // object space; unitària  
layout(location= 2) in vec3 color;        // color en RGB; valors en [0,1]  
layout(location= 3) in vec2 texCoord;     // coordenades de textura (s,t)
```

VS: uniforms

Variables uniform que envia el viewer (cal declarar-les):

```
uniform mat4 modelMatrix;  
uniform mat4 viewMatrix;  
uniform mat4 projectionMatrix;  
uniform mat4 modelViewMatrix;  
uniform mat4 modelViewProjectionMatrix;
```

```
uniform mat4 modelMatrixInverse;  
uniform mat4 viewMatrixInverse;  
uniform mat4 projectionMatrixInverse;  
uniform mat4 modelViewMatrixInverse;  
uniform mat4 modelViewProjectionMatrixInverse;  
  
uniform mat3 normalMatrix;
```

VS: uniforms

Variables uniform que envia el viewer (cal declarar-les):

uniform vec4 lightAmbient;

uniform vec4 lightDiffuse;

uniform vec4 lightSpecular;

uniform vec4 lightPosition; // (sempre estarà en eye space)

uniform vec4 matAmbient;

uniform vec4 matDiffuse;

uniform vec4 matSpecular;

uniform float matShininess;

VS: uniforms

Variables uniform que envia el viewer (cal declarar-les):

uniform vec3 boundingBoxMin; // cantonada de la capsa englobant

uniform vec3 boundingBoxMax; // cantonada de la capsa englobant

uniform vec2 mousePosition; // pos del cursor (window space); origen BL

uniform float time; // temps (segons) des de la darrera compilació

VS: sortides

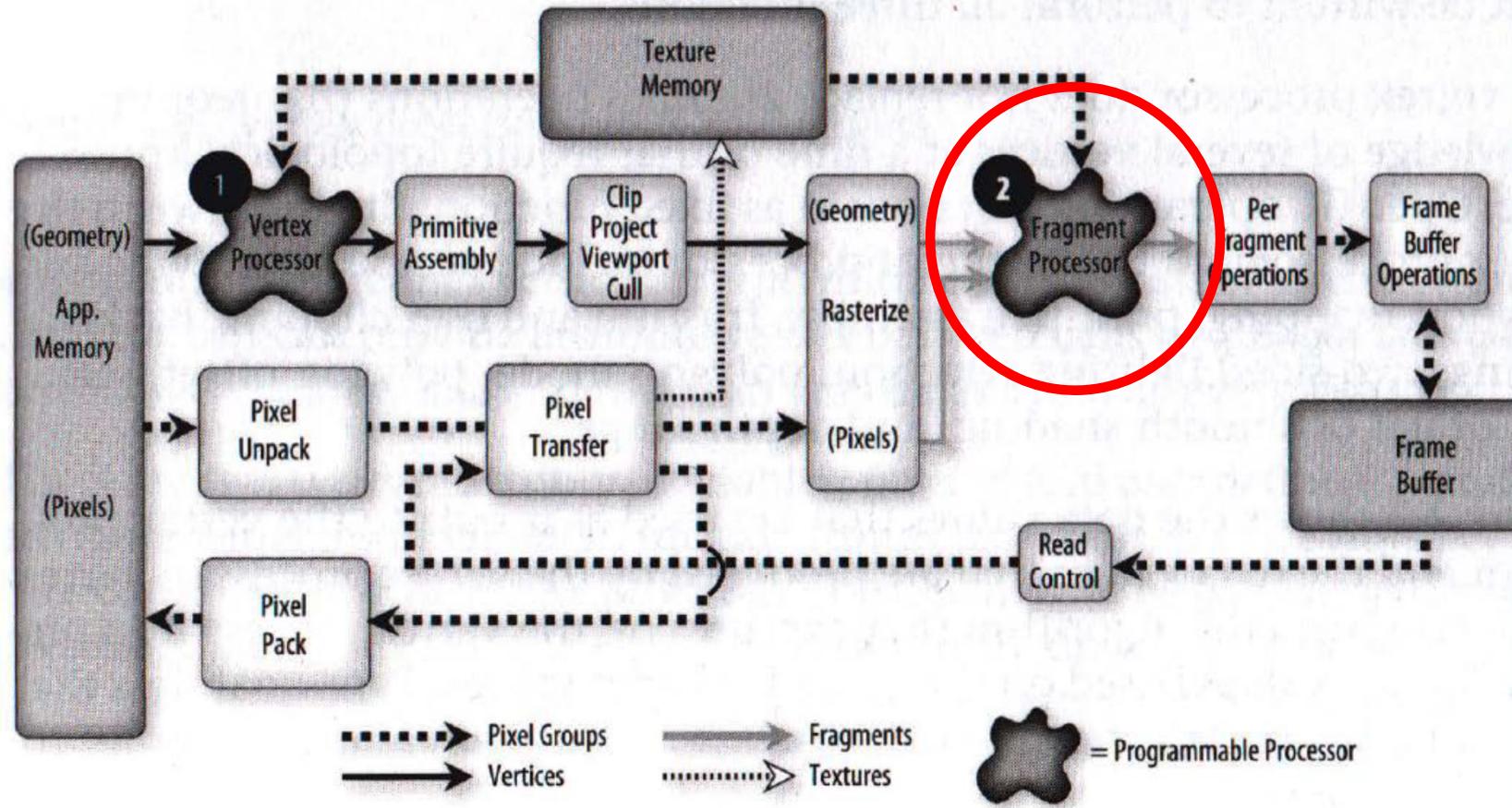
Output variables (pel VS són de sortida; pel FS són d'entrada):

- **out vec4 gl_Position** (predeclarada; habitualment en clip space)
- Qualsevol altre definida per l'usuari. Exemples: color, coordenades del vèrtex, normal, coordenades de textura...

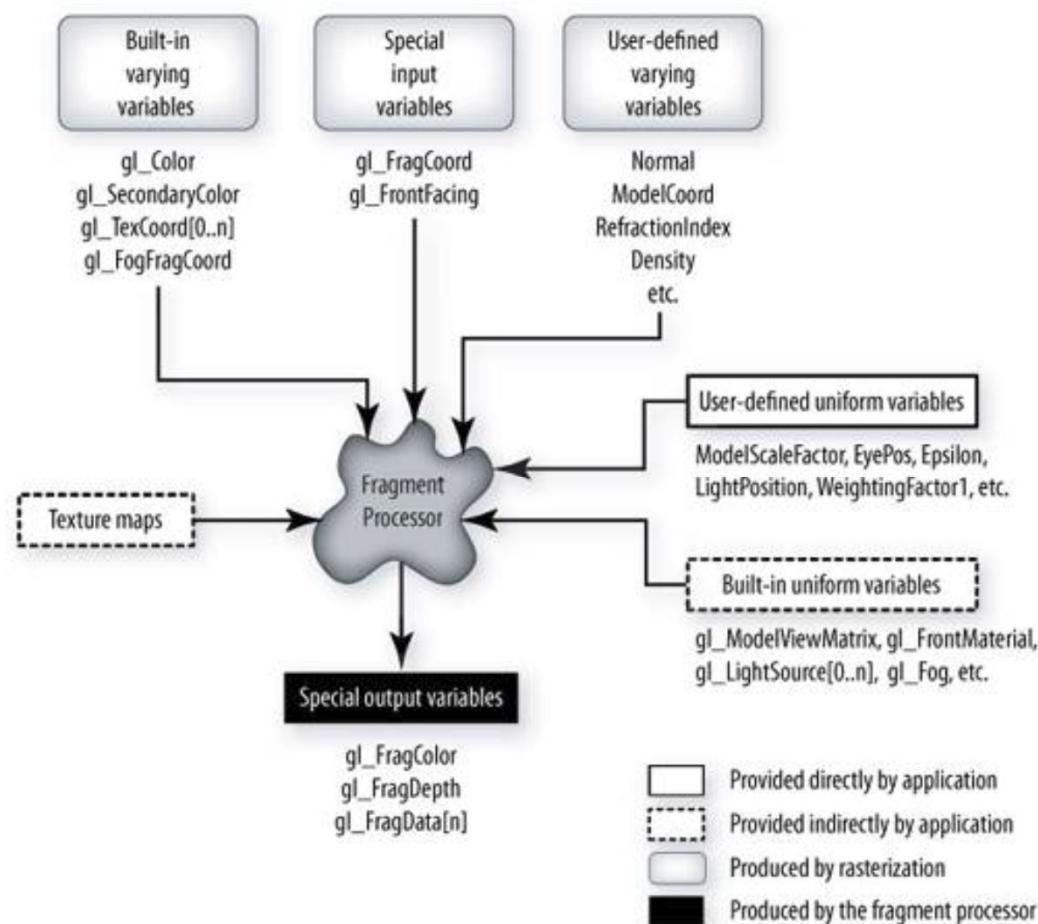
VS

- El VS s'executa per cada vèrtex que s'envia a OpenGL.
- Les tasques habituals d'un VS són:
 - Transformar el vèrtex (object space → clip space)
 - Transformar i normalitzar la normal (eye space)
 - Calcular la il·luminació del vèrtex
 - Generar o passar les coords de textura pel vèrtex

Fragment shaders



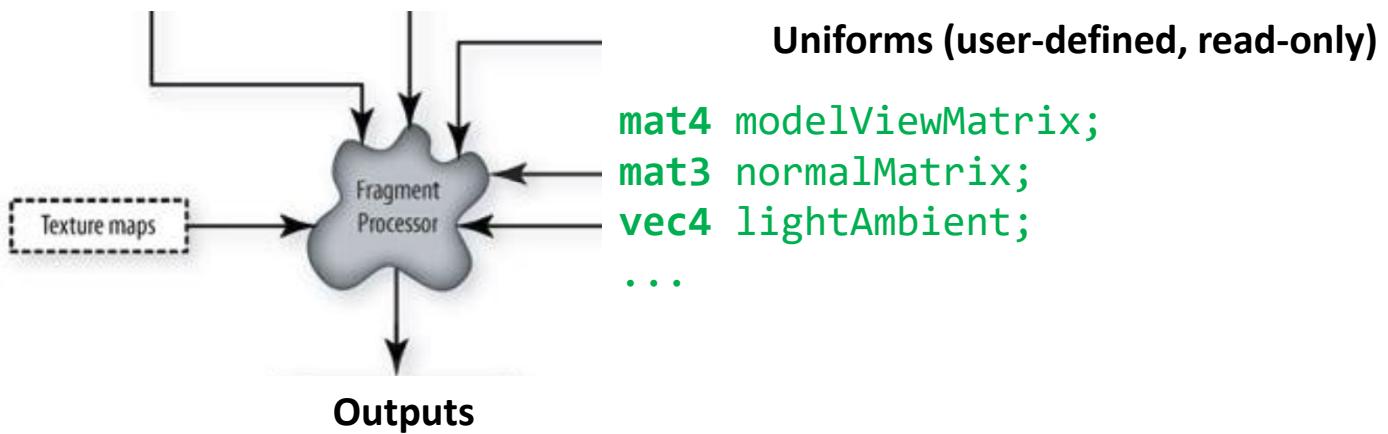
FS (compatibility profile)



FS (3.3 core profile)

Inputs

```
vec4 gl_FragCoord; // window space  
bool gl_FrontFacing; // el fragment l'ha generat una primitiva frontface?  
vec4 frontColor;  
...  
...
```



Outputs

```
float gl_FragDepth; // z en window space  
vec4 fragColor;
```

FS

- El FS s'executa per cada fragment que produeix una primitiva.
- Les tasques habituals d'un FS són:
 - Calcular la il·luminació
 - Usar textures per a afegir detall.
- I el que no pot fer un fragment shader:
 - Canviar les coordenades del fragment (sí pot canviar `gl_FragDepth`)
 - Accedir a informació d'altres fragments (tret de `dFdx`, `dFdy`)

Llenguatge GLSL

Elements del llenguatge

Tipus bàsics

Escalars

int, float, bool

Vectorials

vec2, vec3, vec4, mat2, mat3, mat4, ivec3, bvec4,...

Constructors

Hi ha *arrays*: mat2 mats[3];

i també *structs*:

```
1 struct light{  
2     vec3 color;  
3     vec3 pos;  
4 };
```

que defineixen implícitament constructors: light l1(col,p);

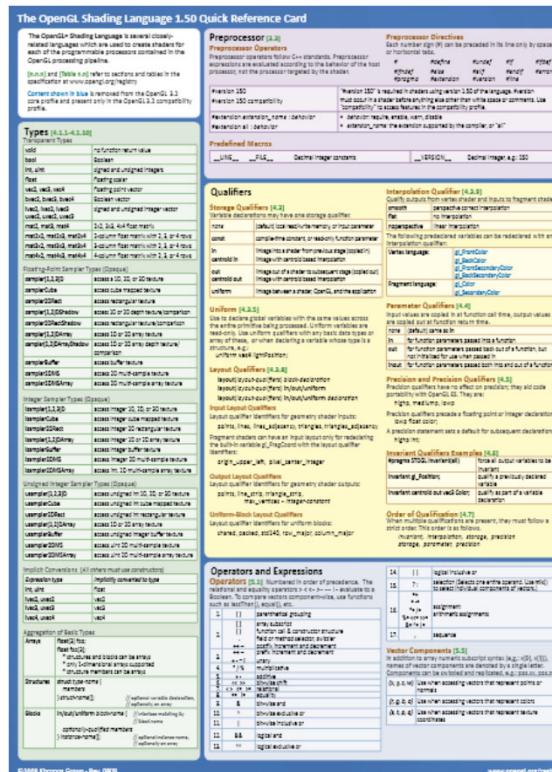
Elements del llenguatge

Funcions

N'hi ha moltes, especialment en les àrees que poden interessar quan tractem geometria o volem dibuixar. Per exemple, radians(), degrees(), sin(), cos(), tan(), asin(), acos(), atan() (amb un o amb dos paràmetres), pow(), log(), exp(), abs(), sign(), floor(), min(), max(), length(), distance(), dot(), cross(), normalize(), noise1(), noise2(), ...

OpenGL Quick Reference card

<https://www.khronos.org/files/opengl-quick-reference-card.pdf>



Exemple: Phong Shading

VS (1/3)

```
#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;

uniform mat4 modelViewProjectionMatrix, modelViewMatrix;
uniform mat3 normalMatrix;
uniform vec4 matAmbient, matDiffuse, matSpecular;
uniform float matShininess;
uniform vec4 lightAmbient, lightDiffuse, lightSpecular
uniform vec4 lightPosition;
```

VS (2/3)

```
vec4 light(vec3 N, vec3 V, vec3 L)
{
    vec3 R = normalize( 2.0*dot(N, L)*N - L );
    float NdotL= max( 0.0, dot(N, L) );
    float RdotV= max( 0.0, dot(R, V) );
    float Idiff= NdotL;
    float Ispec= 0;
    if (NdotL>0) Ispec=pow(RdotV, matShininess);
    return matAmbient * lightAmbient +
           matDiffuse * lightDiffuse * Idiff +
           matSpecular * lightSpecular * Ispec;
}
```

VS (3/3)

```
void main()
{
    vec3 P = (modelViewMatrix * vec4(vertex, 1.0)).xyz;
    vec3 N = normalize(normalMatrix* normal);
    vec3 V = normalize(-P);
    vec3 L = normalize(lightPosition.xyz - P);
    frontColor= light(N, V, L);
    gl_Position= modelViewProjectionMatrix * vec4(vertex, 1.);
}
```

FS

```
#version 330 core
in vec4 frontColor;
out vec4 fragColor;

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