

# Pipeline gràfic programable

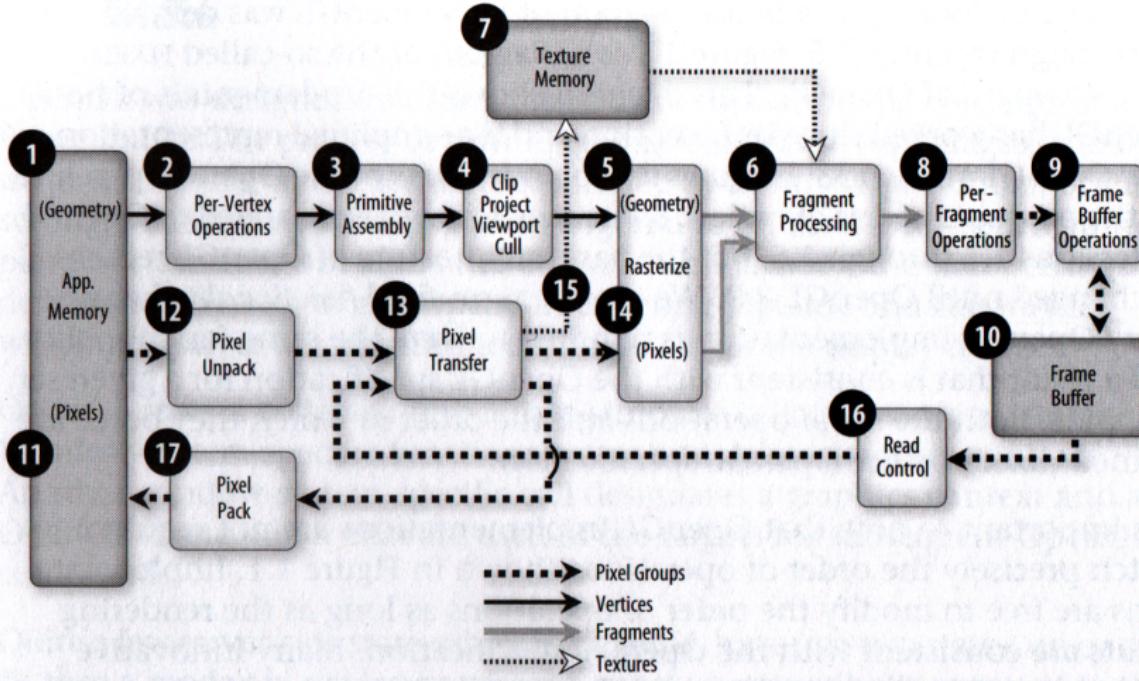
Professors de G

grup Moving

Classes de G, 1112Q2

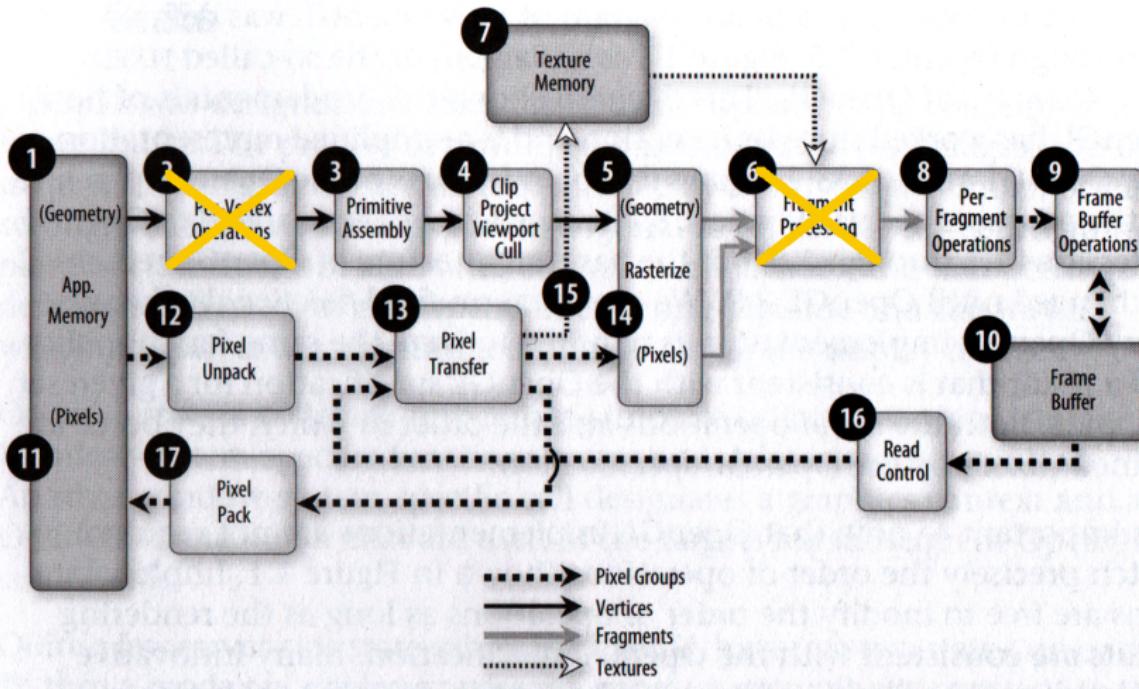
# El pipeline programable

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



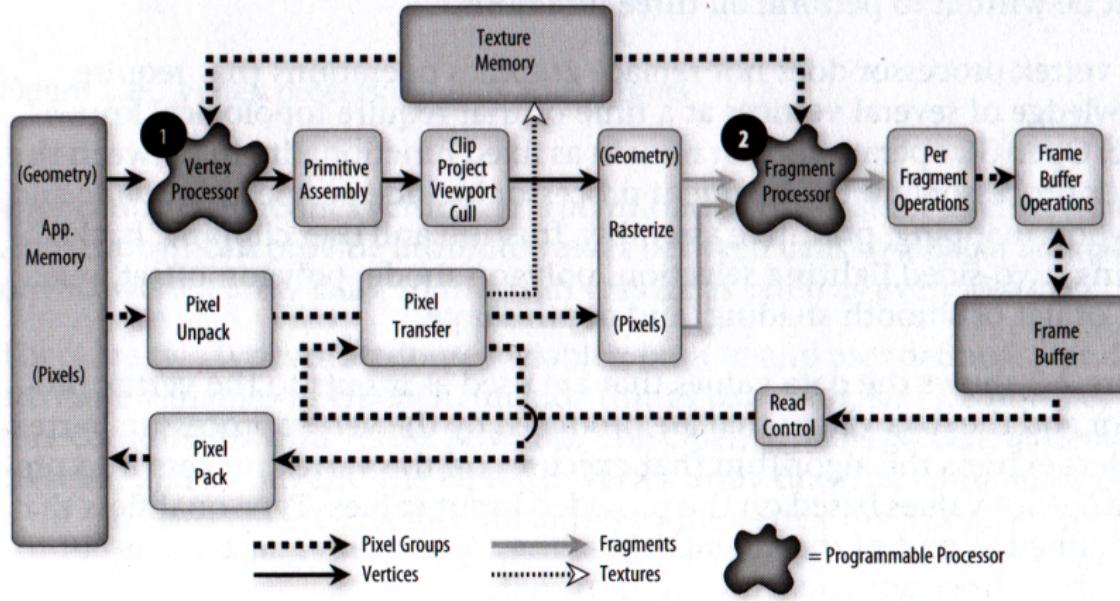
# El pipeline programable

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



# El pipeline programable

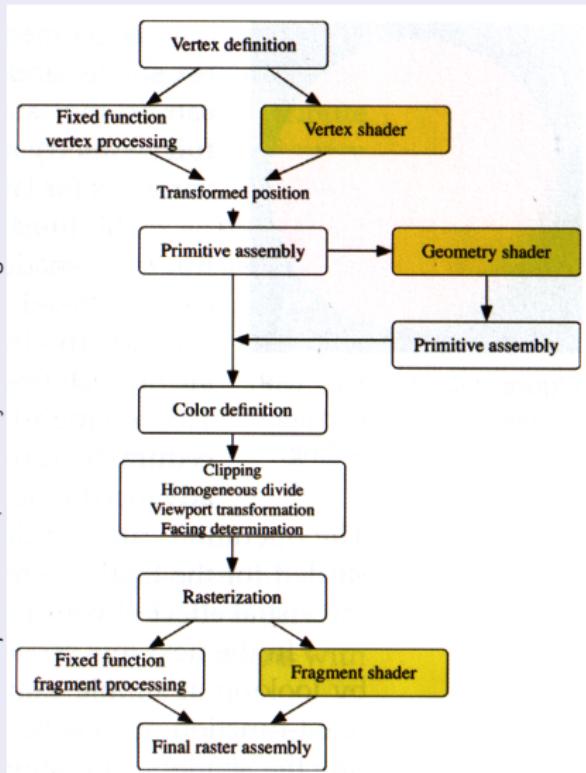
extreta de *OpenGL Shading Language*, R. J. Rost et. al.



# El pipeline programable

Funcionalitats substituïdes

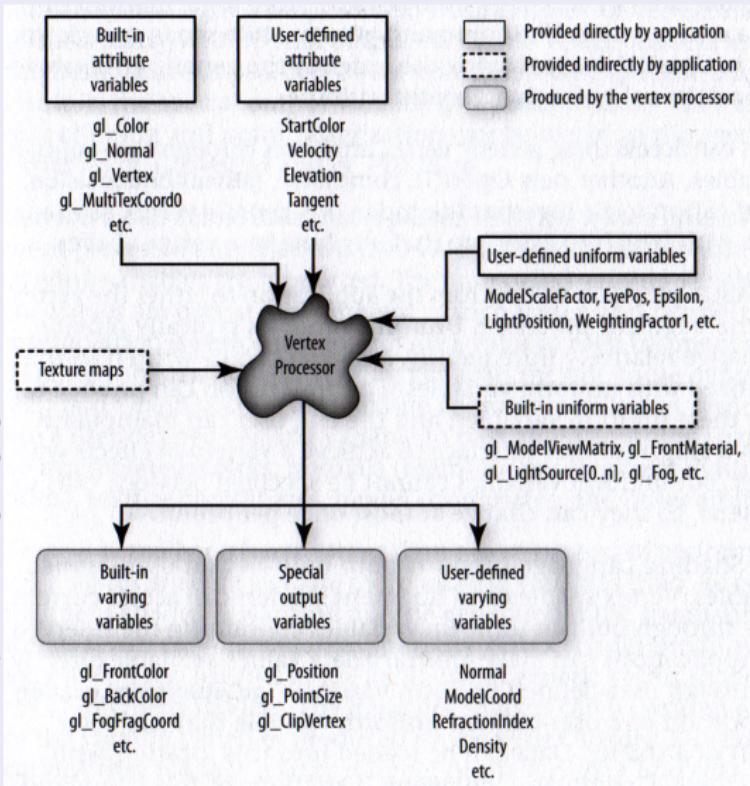
extreta de *Graphic Shaders*, M. Bailey & S. Cunningham



# El pipeline programable

## Flux d'informació als shaders

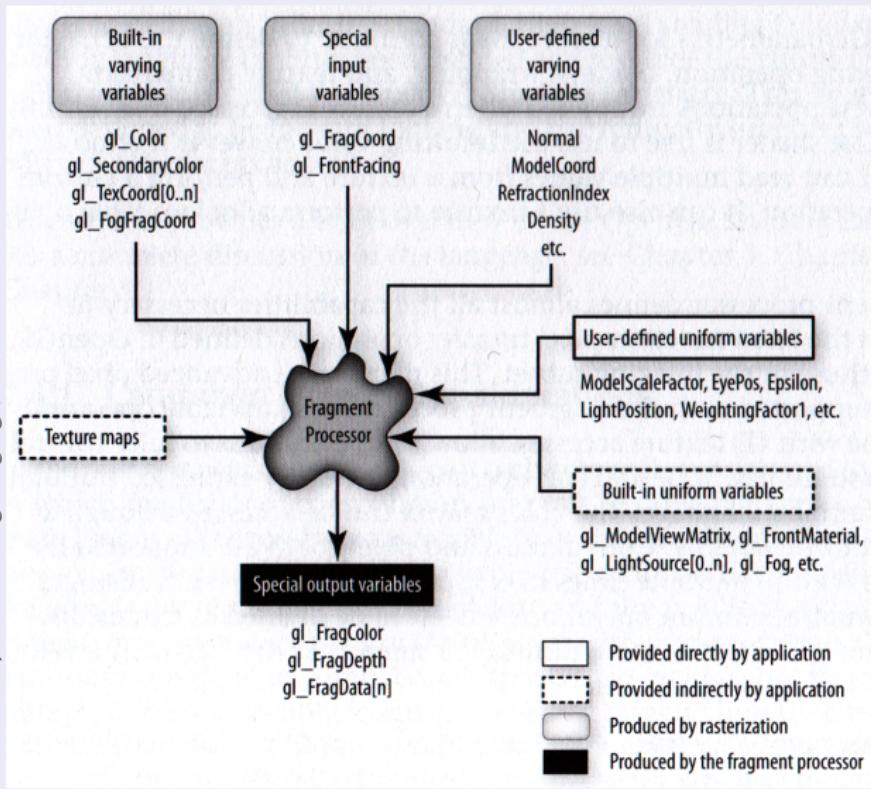
extreta de *OpenGL Shading Language*, R. J. Rost et al.



# El pipeline programable

## Flux d'informació als shaders

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



# Llenguatges de programació dels *shaders*

**Cg** (C per gràfics) Llenguatge desenvolupat per Nvidia.  
Col·laboració amb Microsoft. Basat en C.

**HLSL** (*High-Level Shader Language*) Llenguatge desenvolupat per Microsoft. Col·laboració amb Nvidia. Basat en C.

**GLSL** (*GL Shader Language*) Llenguatge estandarditzat pel OpenGL Architecture Board a partir del release 2.0.

# Llenguatges de programació dels *shaders*

**Cg** (C per gràfics) Llenguatge desenvolupat per Nvidia.  
Col·laboració amb Microsoft. Basat en C.

**HLSL** (*High-Level Shader Language*) Llenguatge desenvolupat per Microsoft. Col·laboració amb Nvidia. Basat en C.

**GLSL** (*GL Shader Language*) Llenguatge estandarditzat pel OpenGL Architecture Board a partir del release 2.0.

## Eines

### FOSS

- Lumina (<http://lumina.sourceforge.net/>)
- BuGLE (<http://www.opengl.org/sdk/tools/BuGLE>)
- Shader Maker  
([http://cg.in.tu-clausthal.de/publications.shtml#shader\\_maker](http://cg.in.tu-clausthal.de/publications.shtml#shader_maker))

### Lliure distribució

- ShaderDesigner  
(<http://www.opengl.org/sdk/tools/ShaderDesigner>)
- glsldevil (<http://www.vis.uni-stuttgart.de/glsldevil>)

### Eines comercials

- gDEBugger (<http://www.gremedy.com>)

Més moltes altres específiques d'alguna plataforma...

# Introducció al GLSL

## Exemple de *vertex shader*

```
1 void main()
2 {
3     gl_Position      = gl_ModelViewProjectionMatrix*
4                           gl_Vertex;
5     gl_FrontColor   = gl_Color;
6 }
```

## Exemple de *fragment shader*

```
1 void main()
2 {
3     gl_FragColor   = gl_Color;
4 }
```

# Introducció al GLSL

## Un exemple una mica més complex

```
1  varying vec3 Normal;  
2  
3  void main(void) {  
4      Normal = normalize(gl_NormalMatrix *  
5                          gl_Normal);  
6      gl_Position =  
7          gl_ModelViewProjectionMatrix *  
8          gl_Vertex;  
9 }
```

# Introducció al GLSL

Un exemple una mica més complex, 2.

```
1 uniform vec3 DiffuseColor;
2 uniform vec3 PhongColor;
3 uniform float Edge;
4 uniform float Phong;
5 varying vec3 Normal;
6
7 void main (void) {
8     vec3 color = DiffuseColor;
9     float f = dot(vec3(0,0,1),Normal);
10    if (abs(f)<Edge) color = vec3(0);
11    if (f>Phong) color = PhongColor;
12    gl_FragColor = vec4(color, 1);
13 }
```

# 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(), ...

# L'API d'OpenGL per a shaders

## Passos necessaris

- ① Crear *shader objects* amb `glCreateShader()`
- ② Assignar-los codi segons convingui amb `glShaderSource()`
- ③ Compilar cadascun amb `glCompileShader()`
- ④ Crear un programa (buit) amb `glCreateProgram()`
- ⑤ Incloure-hi els *shaders* que calgui amb `glAttachShader()`
- ⑥ *Linkar* el programa amb `glLinkProgram()`
- ⑦ Activar l'ús del programa amb `glUseProgram()`

Les crides `glGetShader()` i `glGetShaderInfoLog()` permeten comprovar el resultat i obtenir-ne informació adicional. També podem desfer el que hem fet amb `glDetachShader()`, `glDeleteShader()` i `glDeleteProgram()`.

# L'API d'OpenGL per a shaders

## Flux de informació

### Atributs

Podem afegir atributs segons sigui necessari amb `glBindAttribLocation()`/`glGetAttribLocation()`, usant `glVertexAttrib*`() entre `glBegin()` i `glEnd()`, tal com ho faríem amb atributs estàndard d'OpenGL.

### Uniforms

De forma semblant, disposem de `glGetUniformLocation()` per a obtenir el `GLuint` que identifica una variable d'aquest tipus, i podem ulteriorment donar-li valors amb `glUniform*`() i `glUniformMatrix*`()

# Un exemple més detallat

## Colorat de Phong

### Vertex shader

```
1     varying vec3 Vobs , Nobs ;  
2  
3     void main()  
4     {  
5         gl_Position = gl_ModelViewProjectionMatrix  
6                         * gl_Vertex ;  
7  
8         Vobs = vec3(gl_ModelViewMatrix * gl_Vertex);  
9  
10        Nobs = gl_NormalMatrix * gl_Normal;  
11    }
```

# Un exemple més detallat (II)

Colorat de Phong

## Fragment Shader (i)

```
1  varying vec3 Vobs, Nobs;
2  void main() {
3      vec3 L = gl_LightSource[0].position.xyz-Vobs;
4      L = normalize(L);
5      vec3 color = GetAmbient();
6      if (dot (L, Nobs) > 0.){
7          color += GetDiffuse (Nobs, L);
8          vec3 R = normalize (reflect (-L, Nobs));
9          vec3 V = normalize (- Vobs);
10         if (dot (R, V) > 0.)
11             color += GetSpecular (R,V);
12     }
13     gl_FragColor = vec4 (color.rgb, 1.);
14 }
```

# Un exemple més detallat (III)

Colorat de Phong

## Fragment shader (ii)

```
1  vec3 GetAmbient()
2  {
3      return (gl_LightSource[0].ambient.rgb
4              * gl_FrontMaterial.ambient.rgb);
5  }
6
7  vec3 GetDiffuse(vec3 N, vec3 L)
8  {
9      vec3 diff = (gl_LightSource[0].diffuse.xyz
10             * gl_FrontMaterial.diffuse.xyz);
11      diff = diff * max(dot(N,L), 0.0);
12      return diff;
13 }
```

# Un exemple més detallat (IV)

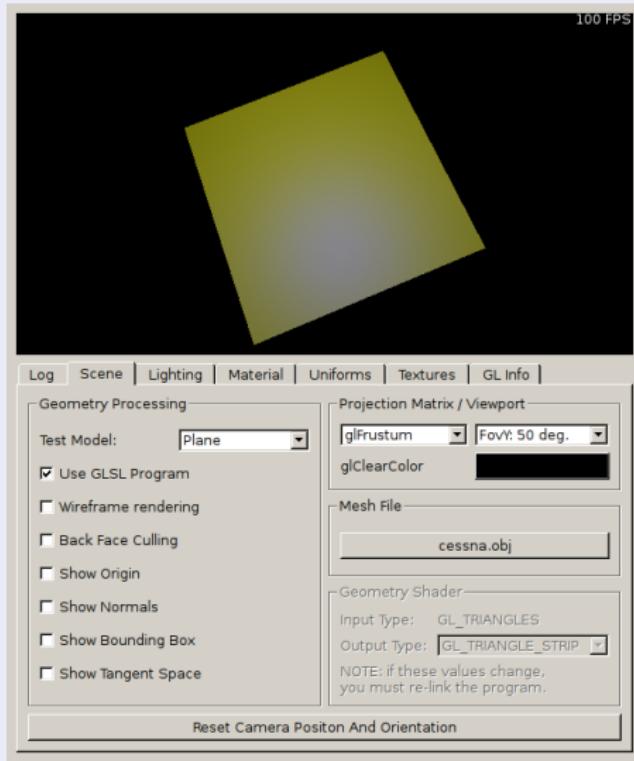
## Colorat de Phong

### Fragment shader (i iii)

```
1  vec3 GetSpecular(vec3 V, vec3 R)
2  {
3      vec3 spec = gl_LightSource[0].specular.xyz
4                  * gl_FrontMaterial.specular.rgb;
5      spec = spec * pow(max(dot(V,R),0.0),
6                          gl_FrontMaterial.shininess);
7      return spec;
8 }
```

# ShaderMaker

## Exemple d'una plataforma per a experimentar



The screenshot shows the ShaderMaker code editor. The tabs at the top are "Vertex Shader" and "Fragment Shader". The "Fragment Shader" tab is selected, showing the following GLSL code:

```
/* lighting.frag - per fragment lighting */

// switch between vertex and fragment lighting.
uniform bool disablePerFragmentLighting;

// use toon sahding
uniform bool useToonShading;

// wether the eye is located in the origin (true
// or at (0,0, +infinity) (false)
uniform bool eyeAtOrigin;

// input
varying vec3 normal; // fragment normal in eye
varying vec3 position; // fragment position in eye
```

At the bottom of the editor, there are two buttons: "Attach to program" and "Compile and Link (F5)".