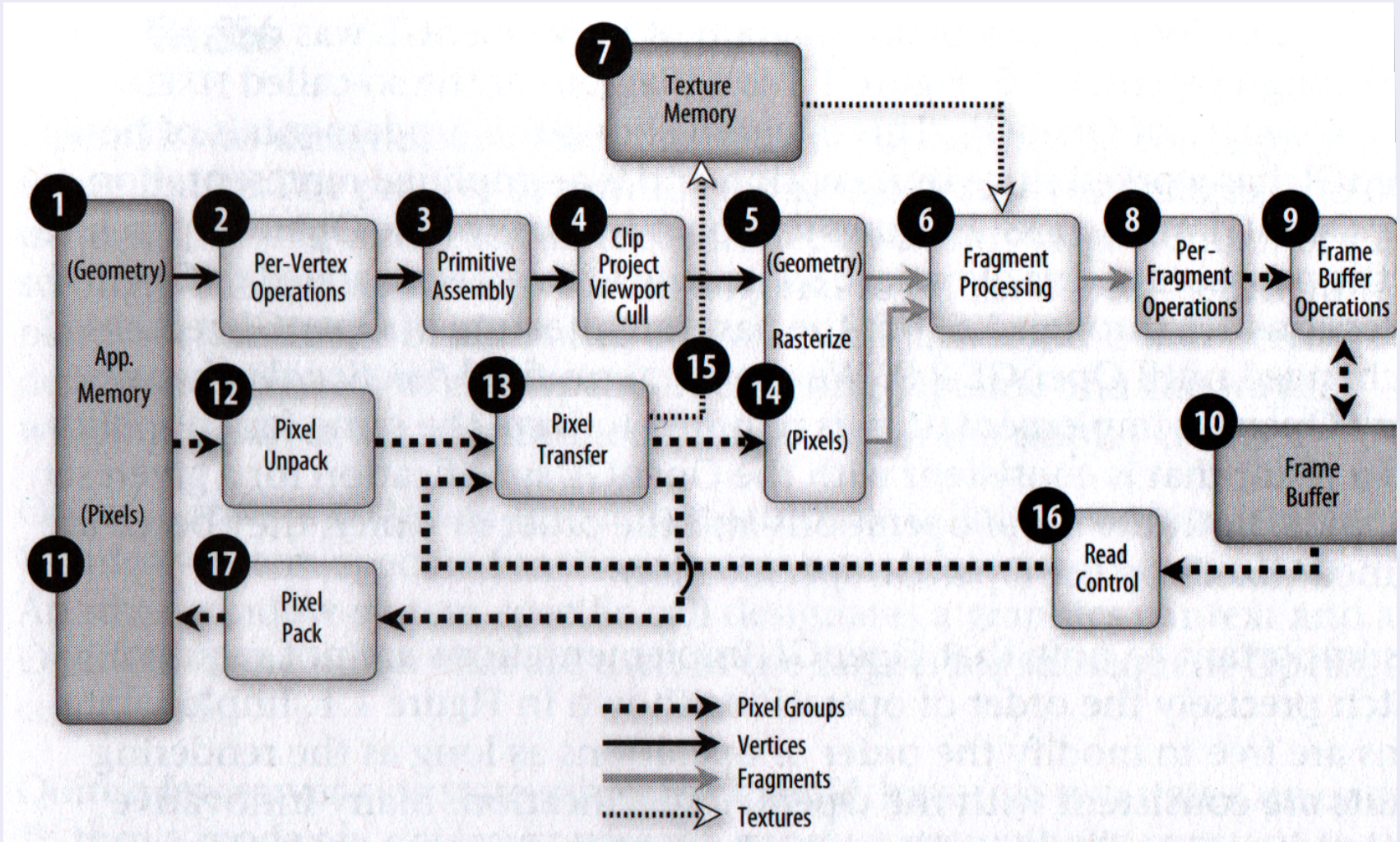
Pipeline gràfic programable

Professors de G

grup Moving

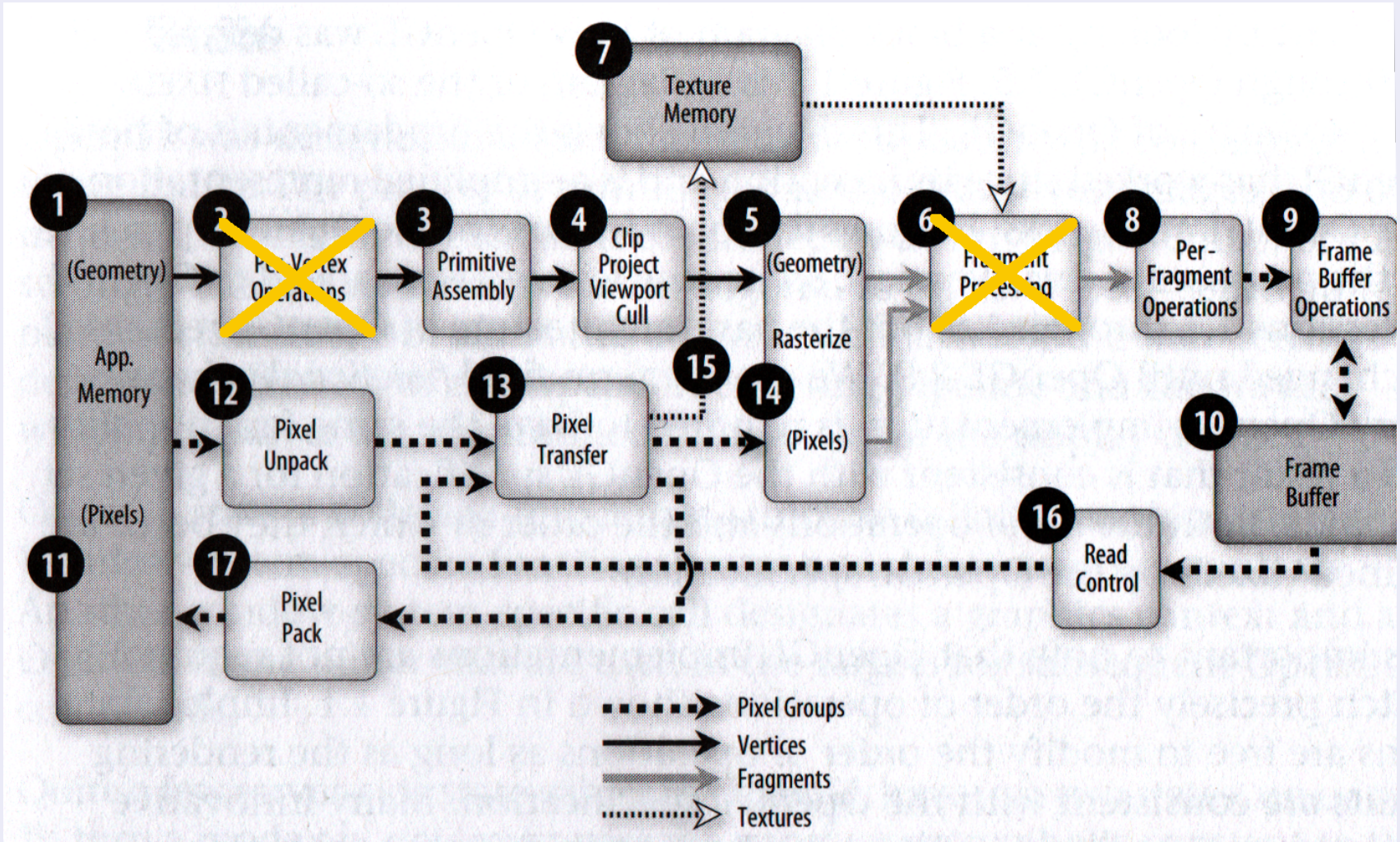
Classes de G, 1213Q2

El pipeline programable



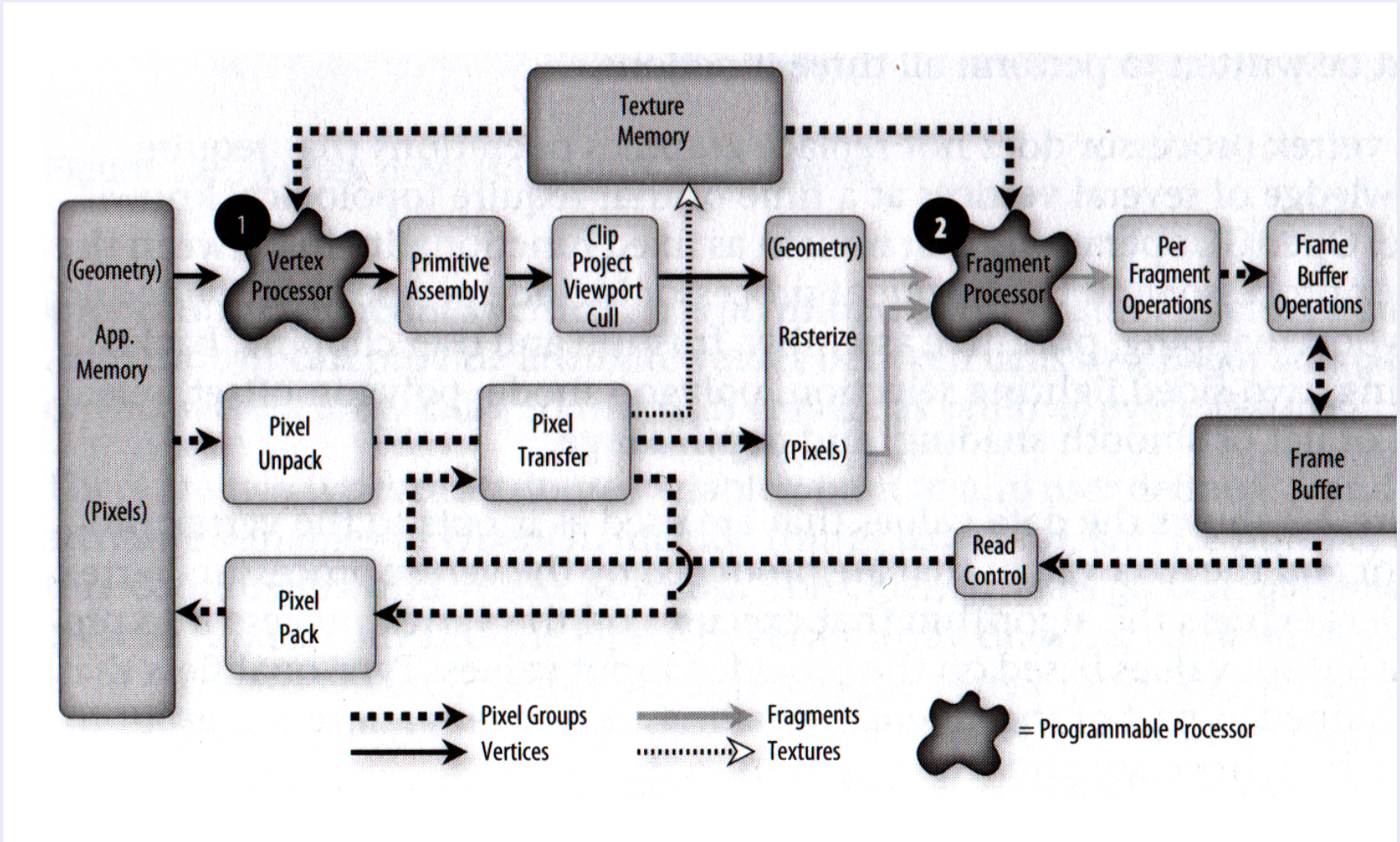
extreta de *OpenGL Shading Language*, R. J. Rost *et. al.*

El pipeline programable



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

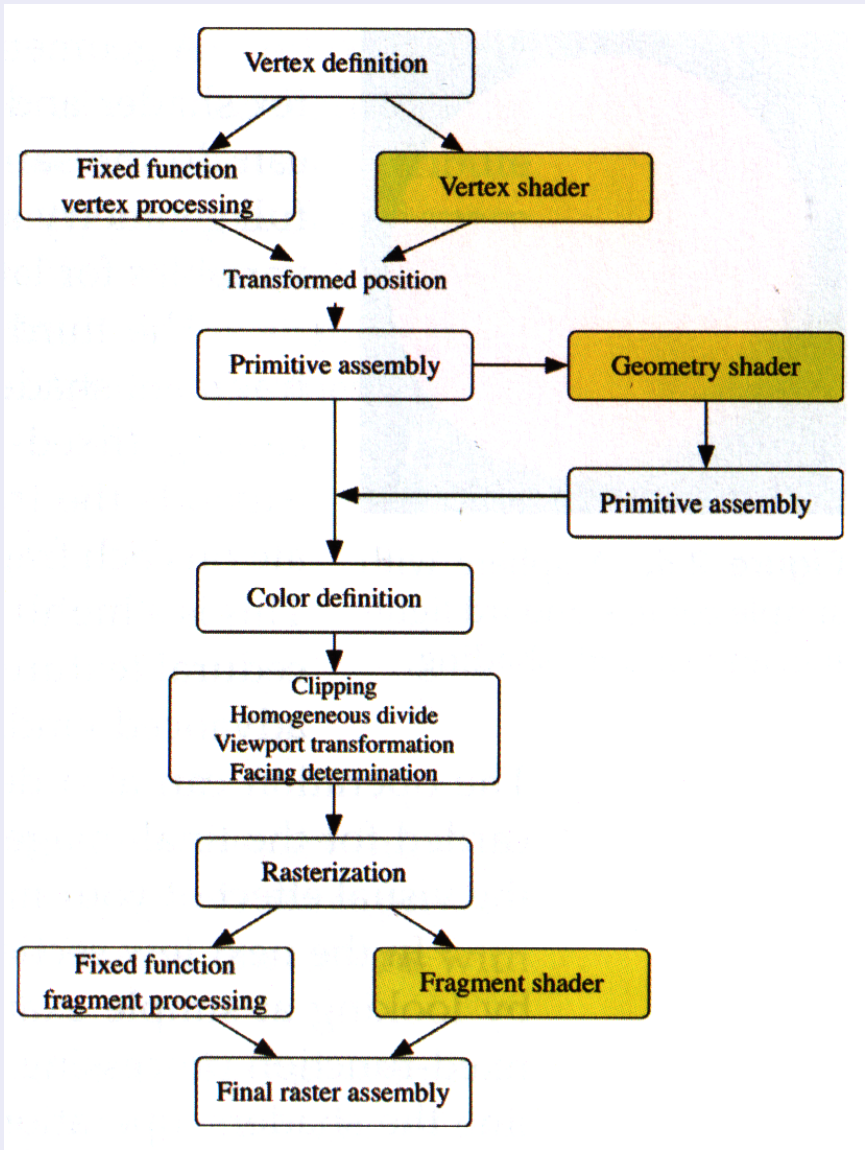
El pipeline programable



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

El pipeline programable

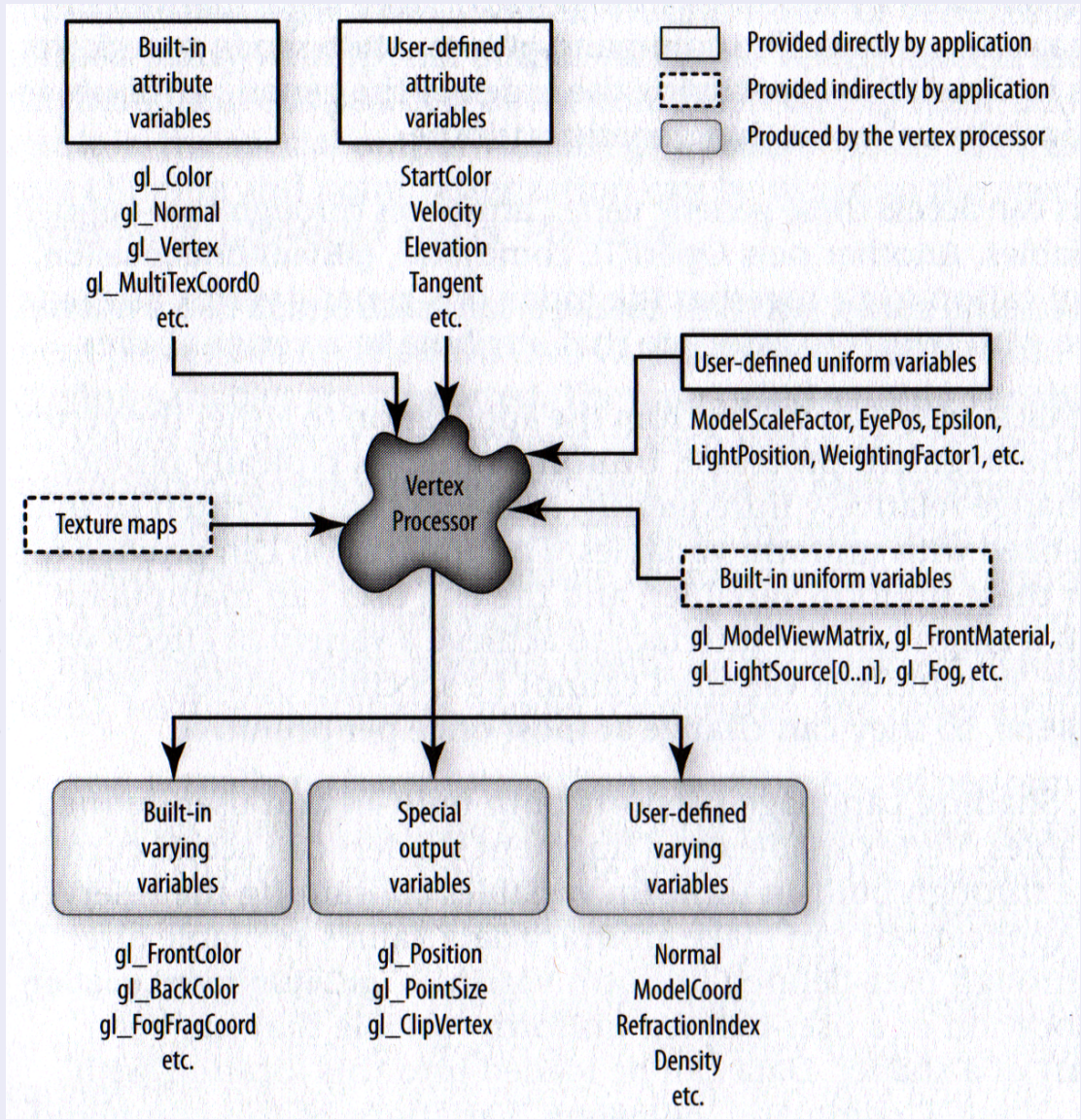
Funcionalitats substituïdes



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

El pipeline programable

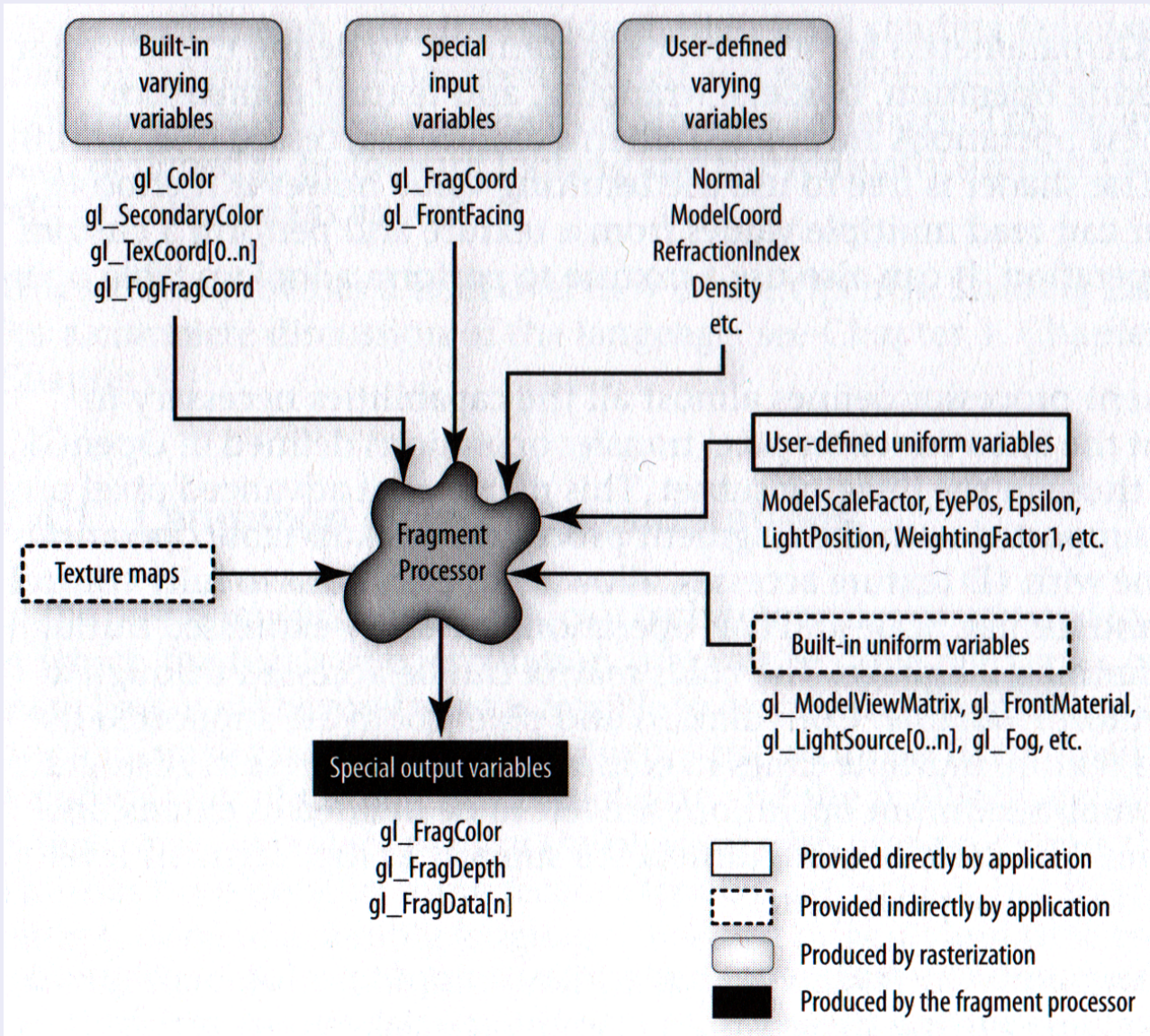
Fluxe d'informació als shaders



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

El pipeline programable

Fluxe 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 estandaritzat 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 estandaritzat pel OpenGL Architecture Board a partir del release 2.0.

Eines

FOSS

- BuGLe (`http://www.opengl.org/sdk/tools/BuGLe`)
- 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/`)
- gDEBugger (`http://www.gremedy.com/`)

Més moltes altres específiques d'alguna plataforma...

Evolució

Versions

Versió	Vers. OGL	data	incorpora
1.10	2.0	2004	vertex i fragment shaders
1.20	2.1	2006	
1.30	3.0	2008	Core and Compatibility profiles, in, out, inout
1.40	3.1	2009	
1.50	3.2	2009	geometry shaders
	3.3	2010	
	4.0	2010	tesselation shaders
	...		
	4.3	2012	compute shaders

Evolució

Versions

Versió	Vers. OGL	data	incorpora
1.10	2.0	2004	vertex i fragment shaders
1.20	2.1	2006	
1.30	3.0	2008	Core and Compatibility profiles, in, out, inout
1.40	3.1	2009	
1.50	3.2	2009	geometry shaders
→	3.3	2010	←
	4.0	2010	tesselation shaders
	...		
	4.3	2012	compute shaders

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 (*à la* 1.20)

```
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 (*à la* 1.20), 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 }
```

Introducció al GLSL

Un exemple una mica més complex (*à la* 3.30 Compatibility)

```
1  #version 330 Compatibility
2  out vec3 Normal;
3
4  void main(void) {
5    Normal = normalize(gl_NormalMatrix*
6                gl_Normal);
7    gl_Position =
8          gl_ModelViewProjectionMatrix*
9          gl_Vertex;
10 }
```

Introducció al GLSL

Un exemple una mica més complex (*à la* 3.30 Compatibility), 2.

```
1  #version 330 Compatibility
2  uniform vec3 DiffuseColor;
3  uniform vec3 PhongColor;
4  uniform float Edge;
5  uniform float Phong;
6  in vec3 Normal;
7
8  void main (void) {
9     vec3 color = DiffuseColor;
10    float f = dot(vec3(0,0,1),Normal);
11    if (abs(f)<Edge) color = vec3(0);
12    if (f>Phong) color = PhongColor;
13    gl_FragColor = vec4(color, 1);
14 }
```

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

Fluxe d'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