

Tema 3. Procés de visualització projectiu

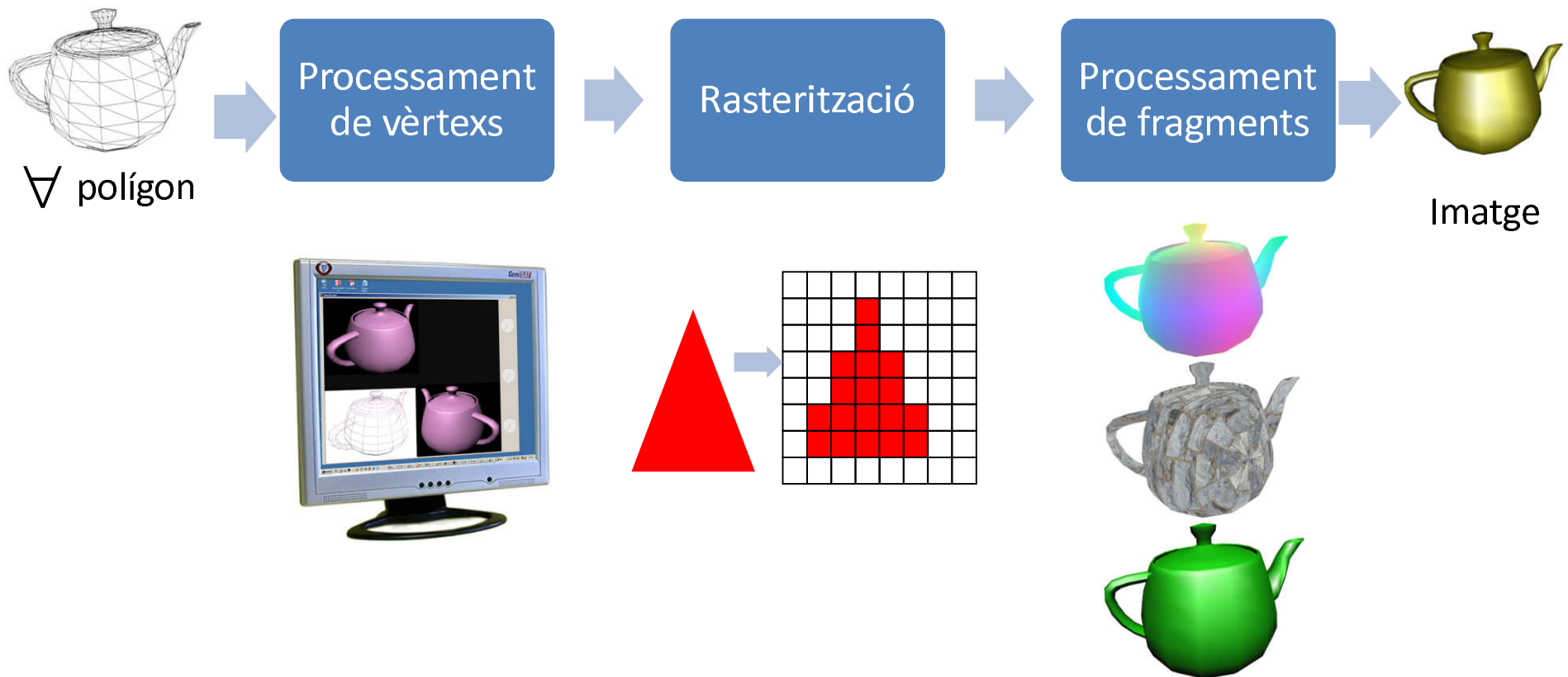
C. Andujar

Setembre 2013

SISTEMES DE COORDENADES I

TRANSFORMACIONS GEOMÈTRIQUES

Paradigma projectiu simplificat



Informació geomètrica al pipeline

```
void draw()
```

```
{
```

```
    GLfloat light_pos[] = { 1.0, 1.0, 1.0, 0.0 };
```

```
    glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
```

Posició d'una llum

```
    for (...) {
```

```
        glBegin(GL_POLYGON);
```

```
        glNormal3f(nx,ny,nz);
```

Components de la normal

```
        glVertex3f(x,y,z);
```

Coordenades d'un vèrtex

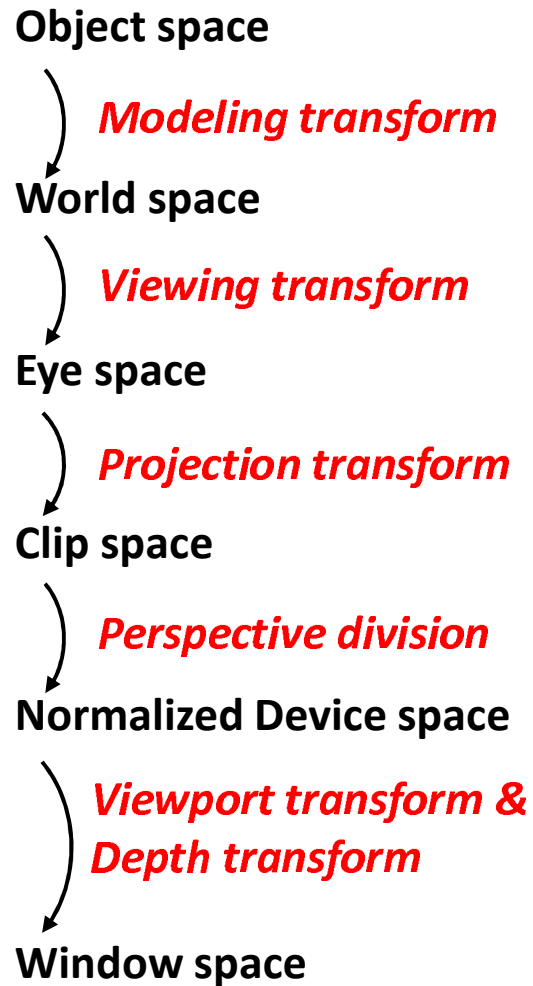
```
        ...
```

```
        glEnd();
```

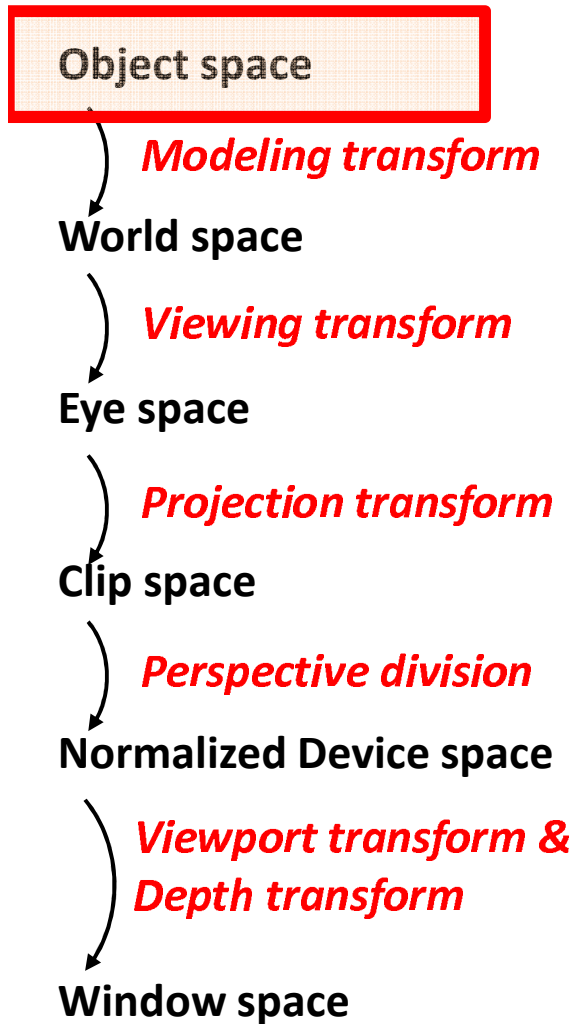
```
    }
```

```
}
```

Sistemas de coordenadas



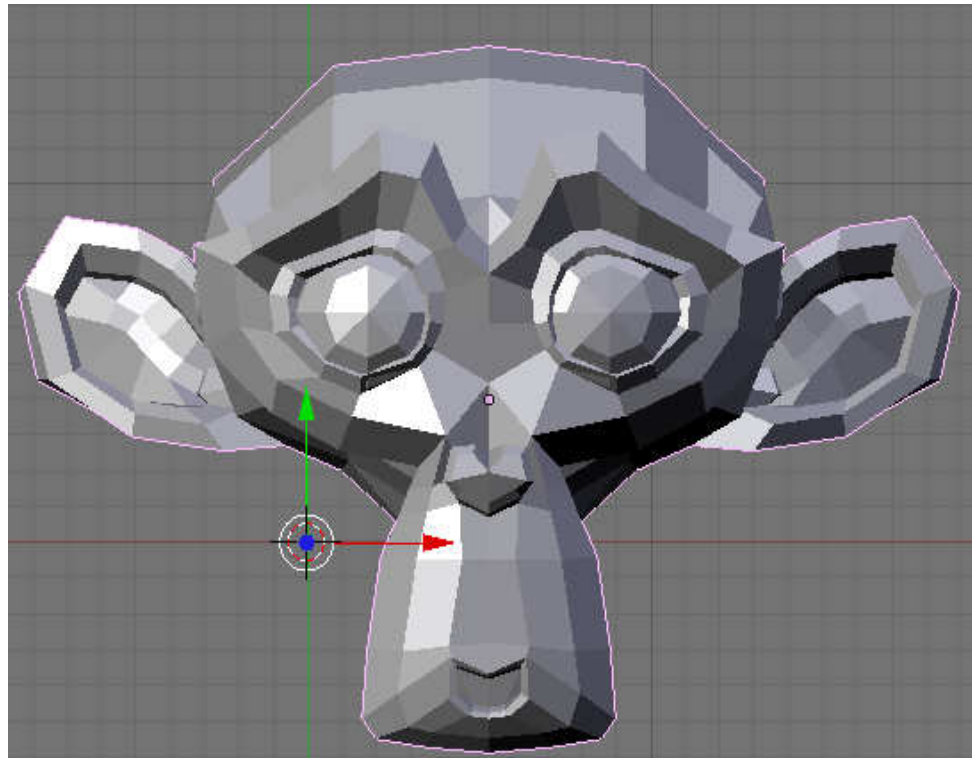
Sistemes de coordenades



Object space (x_m, y_m, z_m, w_m)

Model space, SC del model, SC de l'objecte

- *SC utilitzat per modelar l'objecte.*
- *w_m normalment serà 1.0 (punts) o 0.0 (vectors)*



Sistemes de coordenades

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

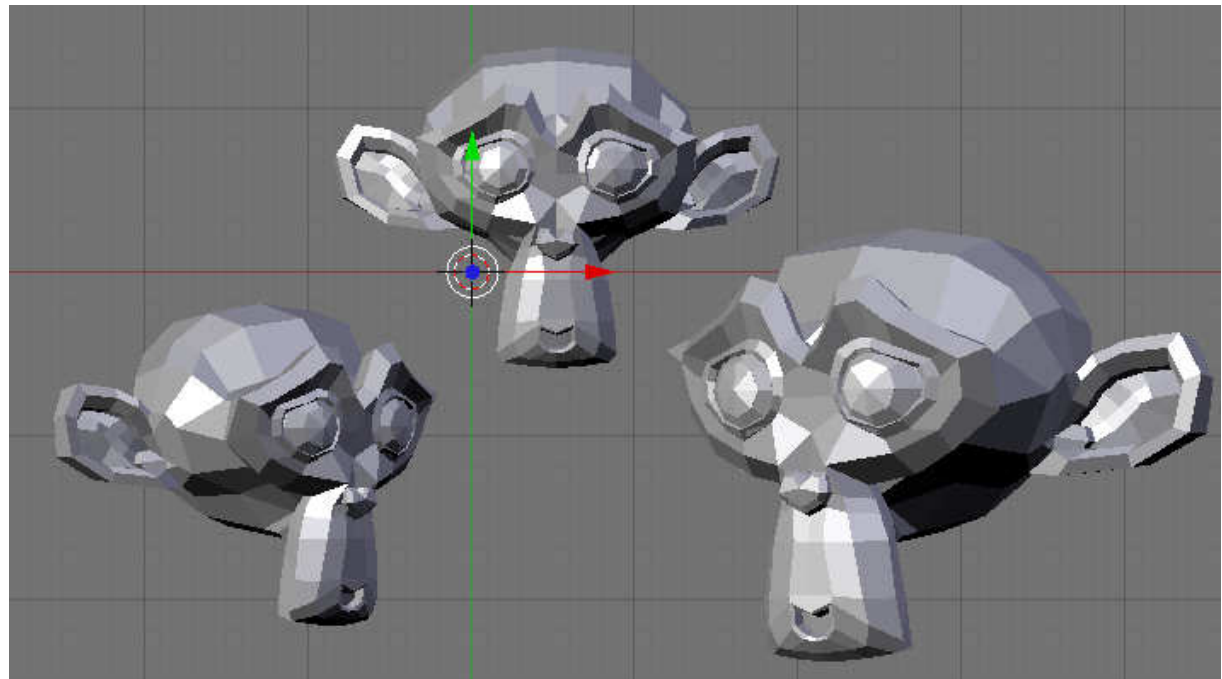
**Viewport transform &
Depth transform**

Window space

Modeling transform

Transformació de modelat

- Aquesta transformació situa cada instància d'un objecte en relació a l'escena.
- Sovint és la identitat, o una composició de translacions, rotacions i escalats (`glTranslatef`, `glRotatef`, `glScalef`)



Sistemes de coordenades

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

**Viewport transform &
Depth transform**

Window space

Modeling transform

Transformació de modelat

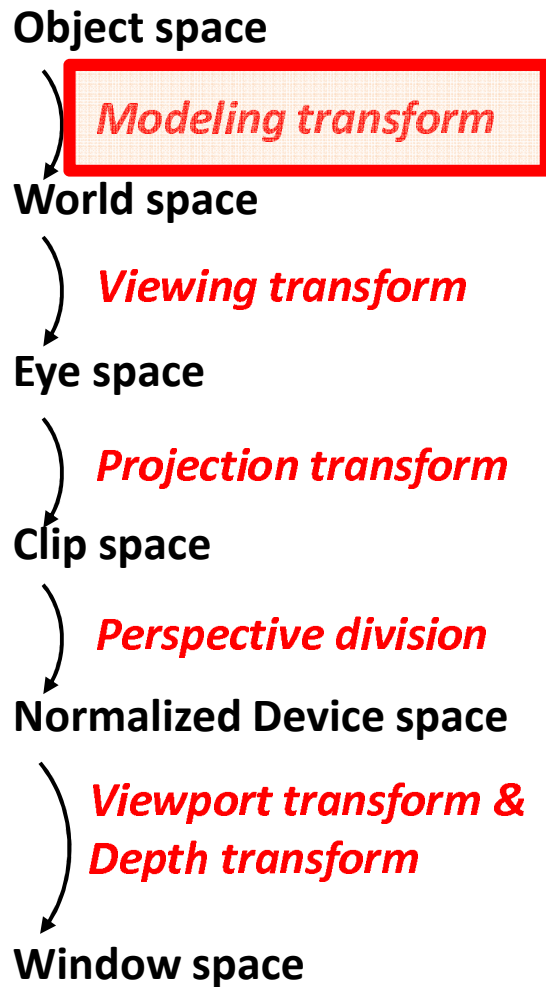
$$\text{glTranslatef}(t_x, t_y, t_z) \quad T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{glScalef}(s_x, s_y, s_z) \quad T = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{glRotatef}(a, x, y, z) \quad T = \begin{bmatrix} x^2d + c & xyd - zs & xzd + ys & 0 \\ yxd + zs & y^2d + c & yzd - xs & 0 \\ xzd - ys & yzd + xs & z^2d + c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$c = \cos(a)$, $s = \sin(a)$, $d = 1 - \cos(a)$

Sistemes de coordenades



Modeling transform

Transformació de modelat

- Punts, vectors: matriu 4x4 multiplicada pel punt:

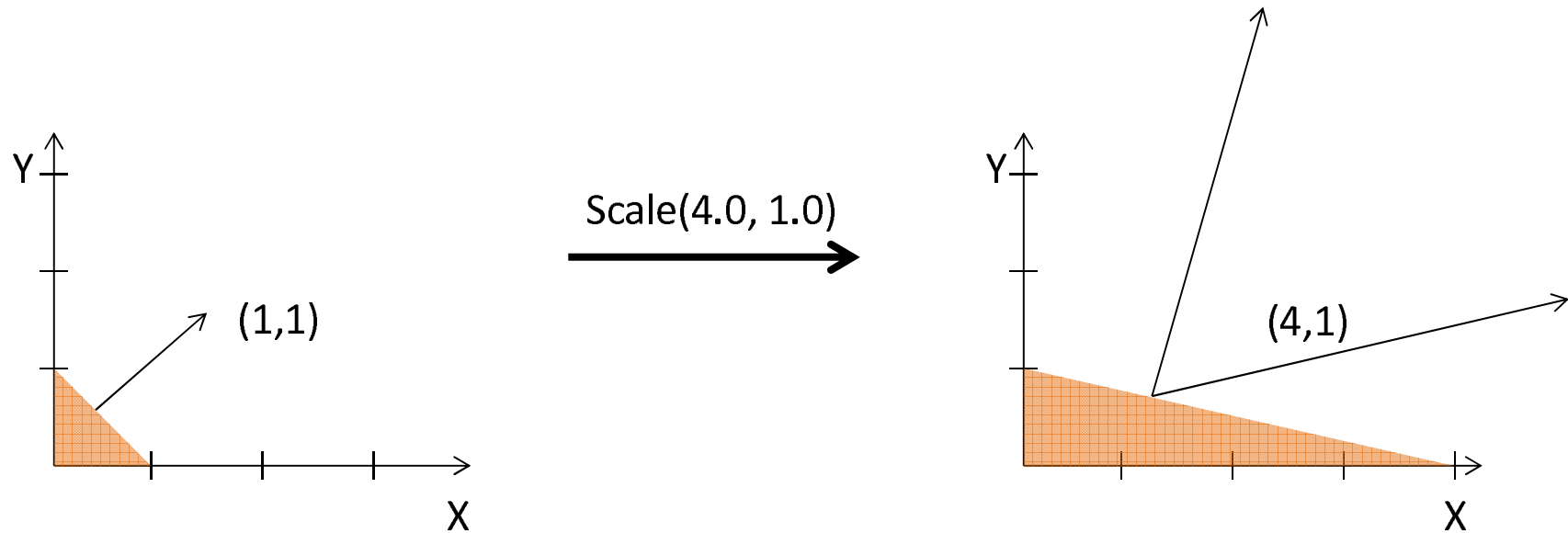
$$\begin{bmatrix} x_a \\ y_a \\ z_a \\ w_a \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix}$$

- Normals: cal la trasposta inversa de la submatriu 3x3

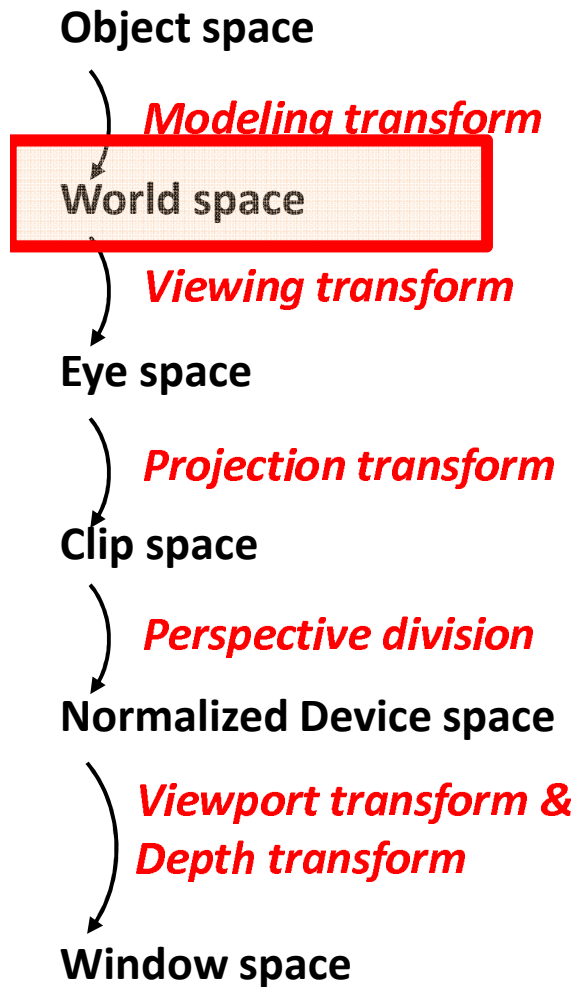
$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^{-T} \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix}$$

Normalment no us
caldrà calcular
aquesta inversa

Sistemas de coordenades



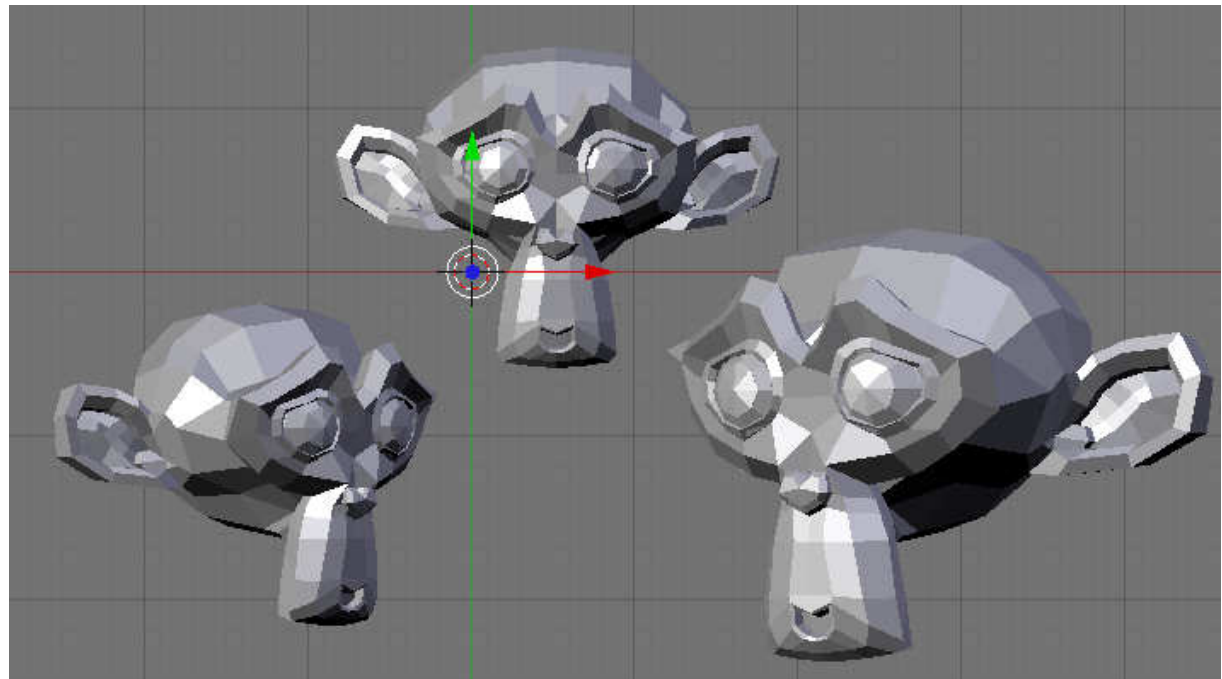
Sistemes de coordenades



World space (x_a, y_a, z_a, w_a)

SC de mon, SC de l'aplicació

- SC utilitzat per representar l'escena
- La transformació de modelat sovint preserva la component homogènia i per tant, w_a normalment serà 1.0 (punts) o 0.0 (vectors)



Sistemes de coordenades

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

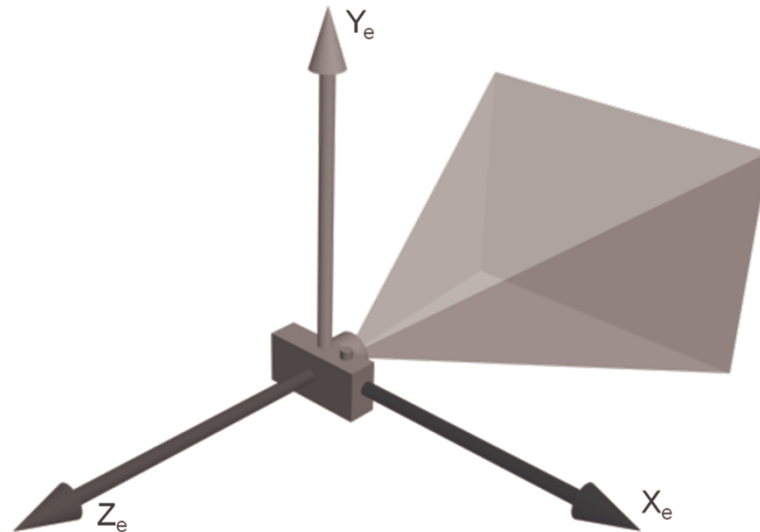
*Viewport transform &
Depth transform*

Window space

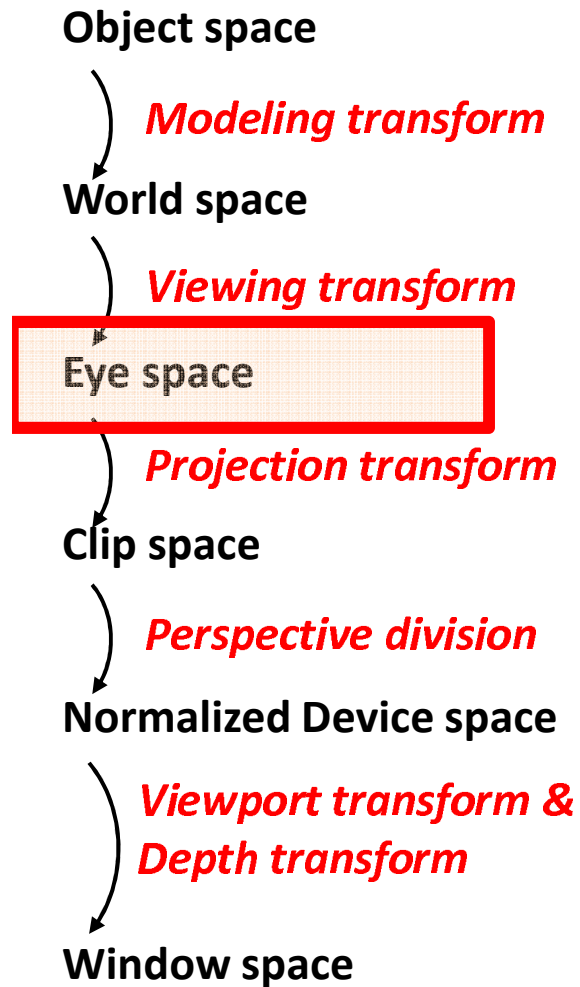
Viewing transform

Transformació de visualització, transformació de càmera

- Fa un canvi al sistema de referència de la càmera.
- Depèn de la posició i orientació de la càmera
- Sovint es defineix amb **gluLookAt**, o amb una composició de translacions i rotacions (**glTranslatef**, **glRotatef**).

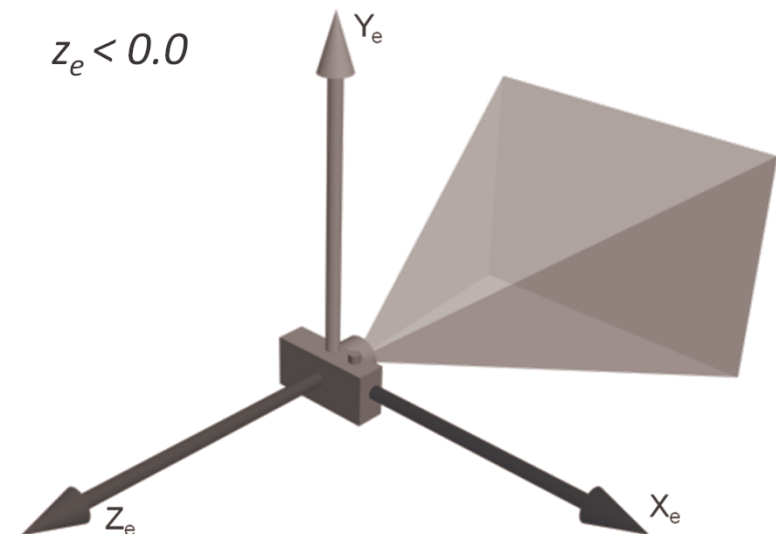


Sistemes de coordenades



Eye space (x_e, y_e, z_e, w_e)
SC de l'observador, SC de la càmera

- SC associat a la càmera
- La transformació de visualització sovint preserva la component homogènia i per tant, w_e normalment serà 1.0 (punts) o 0.0 (vectors)
- Si la càmera és perspectiva, per tal que el punt sigui visible...



Sistemes de coordenades

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

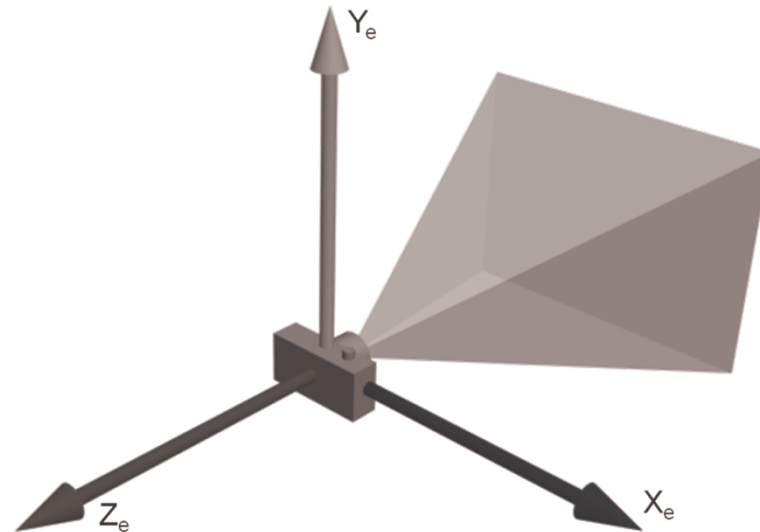
*Viewport transform &
Depth transform*

Window space

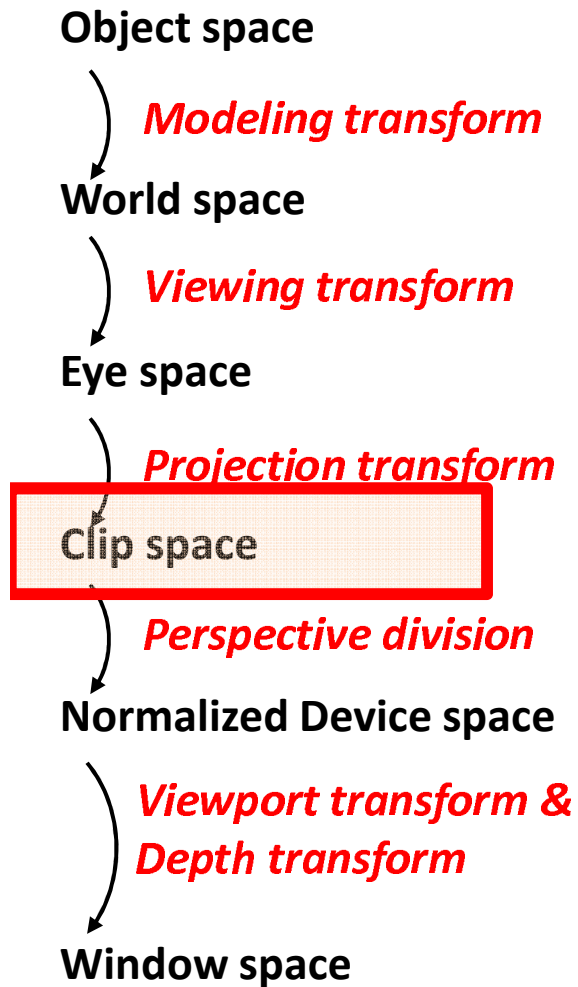
Projection transform

Transformació de projecció

- Depèn de la forma de la piràmide de visió i per tant del tipus de càmera (perspectiva, axonomètrica)
- Sovint es defineix amb ***gluPerspective***, ***glFrustum***, ***glOrtho***



Sistemes de coordenades



Clip space (x_c, y_c, z_c, w_c)

Coordenades de clipping, coordenades de retallat

- Si un punt és interior al frustum,

$$-w_c \leq x_c \leq w_c$$

$$-w_c \leq y_c \leq w_c$$

$$-w_c \leq z_c \leq w_c$$

- Si la càmera és perspectiva, llavors

$$w_c = -z_e$$

Sistemes de coordenades

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

*Viewport transform &
Depth transform*

Window space

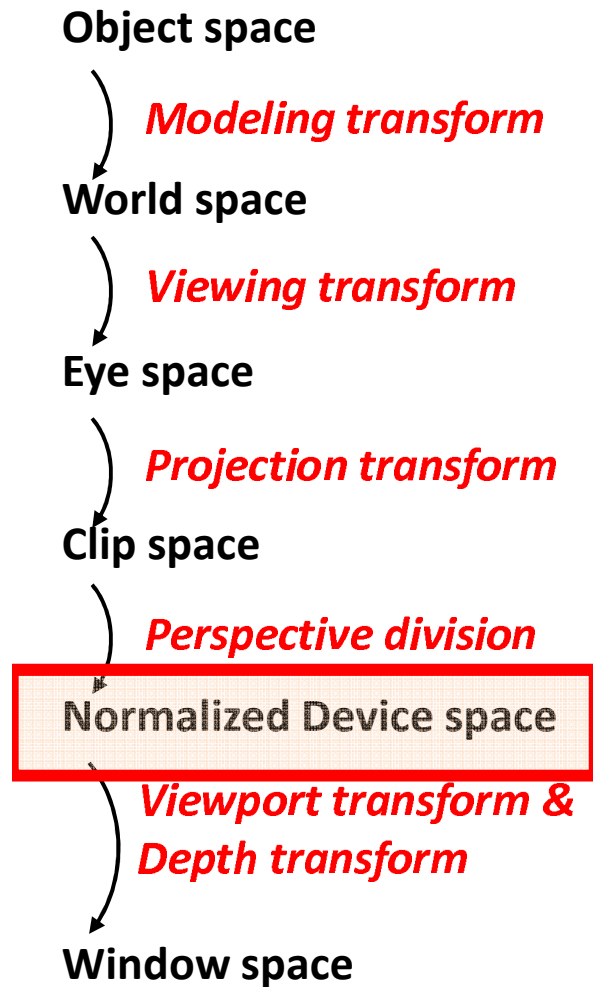
Perspective division

Divisió de perspectiva

- Simplement és el pas de coordenades homogènies a 3D
- Es divideix cada coordenada per la coord homogènia

$$(x_c, y_c, z_c, w_c) \rightarrow (x_c/w_c, y_c/w_c, z_c/w_c)$$

Sistemes de coordenades



Normalized device space (x_n, y_n, z_n)

NDC, cordenades normalitzades

- Si un punt és interior al frustum,

$$-1 \leq x_n \leq 1$$

$$-1 \leq y_n \leq 1$$

$$-1 \leq z_n \leq 1$$

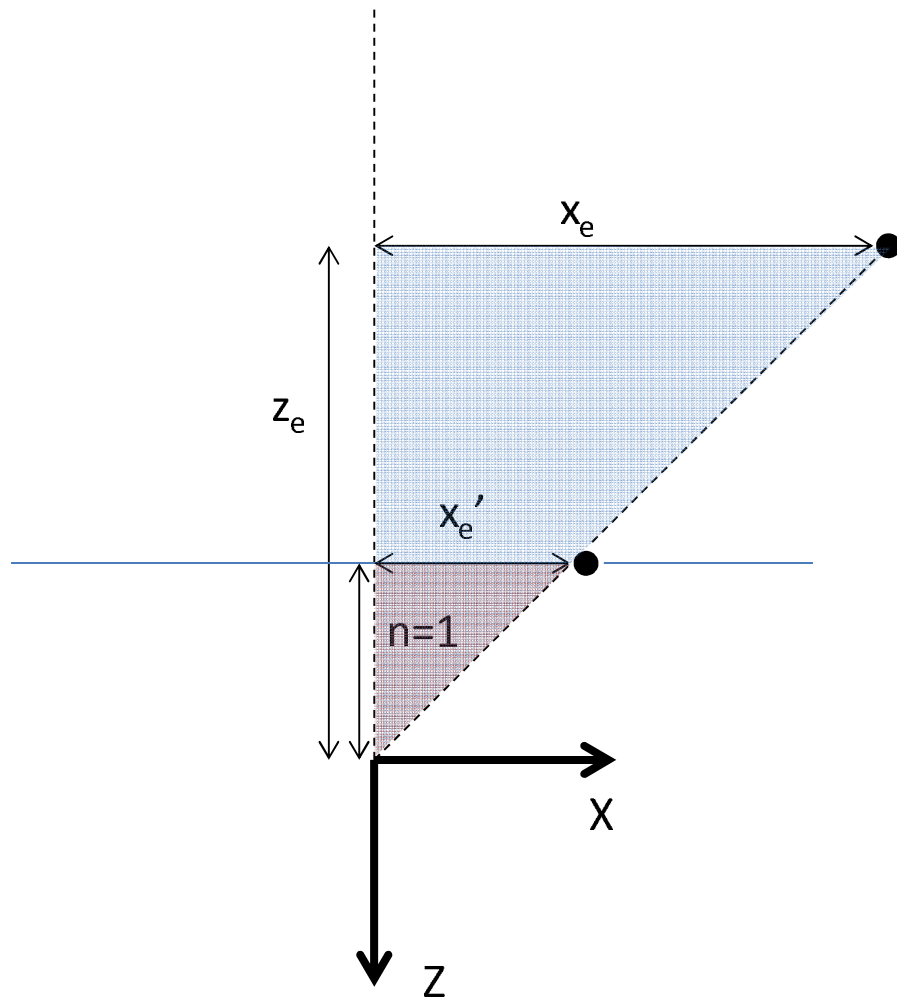
- Els punts situats sobre znear tenen

$$z_n = -1$$

- Els punts situats sobre zfar tenen

$$z_n = +1$$

Projecció i divisió per z_e



$$\frac{x_e'}{1} = \frac{x_e}{z_e}$$

gluPerspective(90, 1, 1, 11);

$$\begin{bmatrix} \cot \frac{fovy}{2} & 0 & 0 & 0 \\ \frac{aspect}{\cot \frac{fovy}{2}} & 0 & 0 & 0 \\ 0 & \cot \frac{fovy}{2} & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2*n*f}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix} \xrightarrow[\substack{n=1, f=11}]{fovy=90, aspect=1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1.2 & -2.2 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Pas eye space → clip space

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1.2 & -2.2 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} x_e \\ y_e \\ -1.2z_e - 2.2 \\ -z_e \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1.2 & -2.2 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_e \\ y_e \\ -1 \\ 1 \end{bmatrix}$$

Punt sobre el pla znear

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1.2 & -2.2 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ -11 \\ 1 \end{bmatrix} = \begin{bmatrix} x_e \\ y_e \\ 11 \\ 11 \end{bmatrix}$$

Punt sobre el pla zfar

Sistemes de coordenades

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

*Viewport transform &
Depth transform*

Window space

Viewport transformation

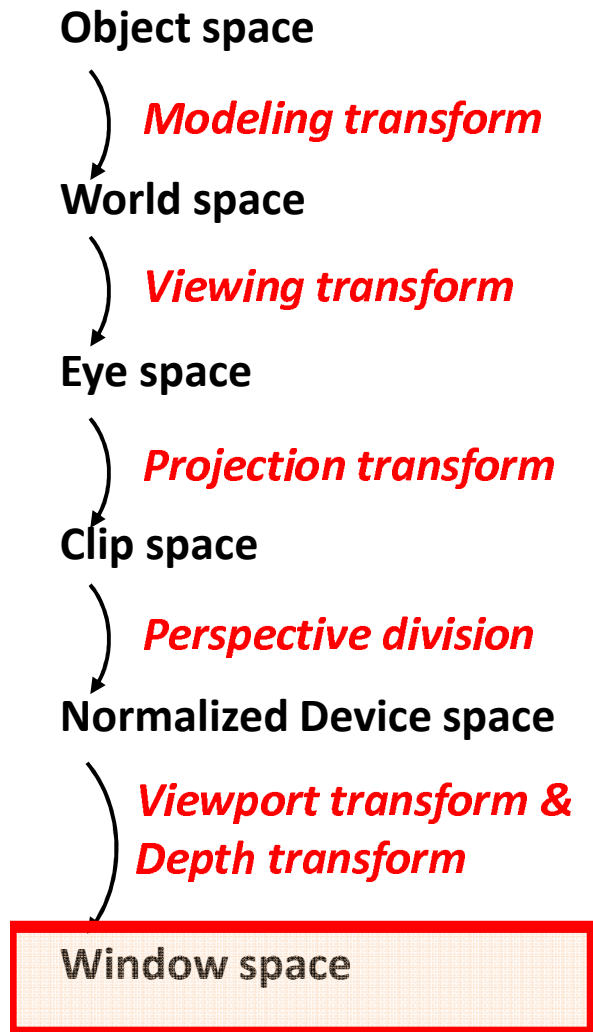
Transformació mon-dispositiu

- Depèn del viewport definit amb ***glViewport***

Depth range transformation

- Depèn de l'interval definit amb ***glDepthRange*** (per defecte $[0,1]$)

Sistemes de coordenades



Window space (x_d, y_d, z_d)

Cordenades de finestra, coordenades de dispositiu

- Si un punt és interior al frustum,

$$0 \leq x_d \leq w$$

$$0 \leq y_d \leq h$$

$$0 \leq z_d \leq 1$$

- Els punts situats sobre znear tenen

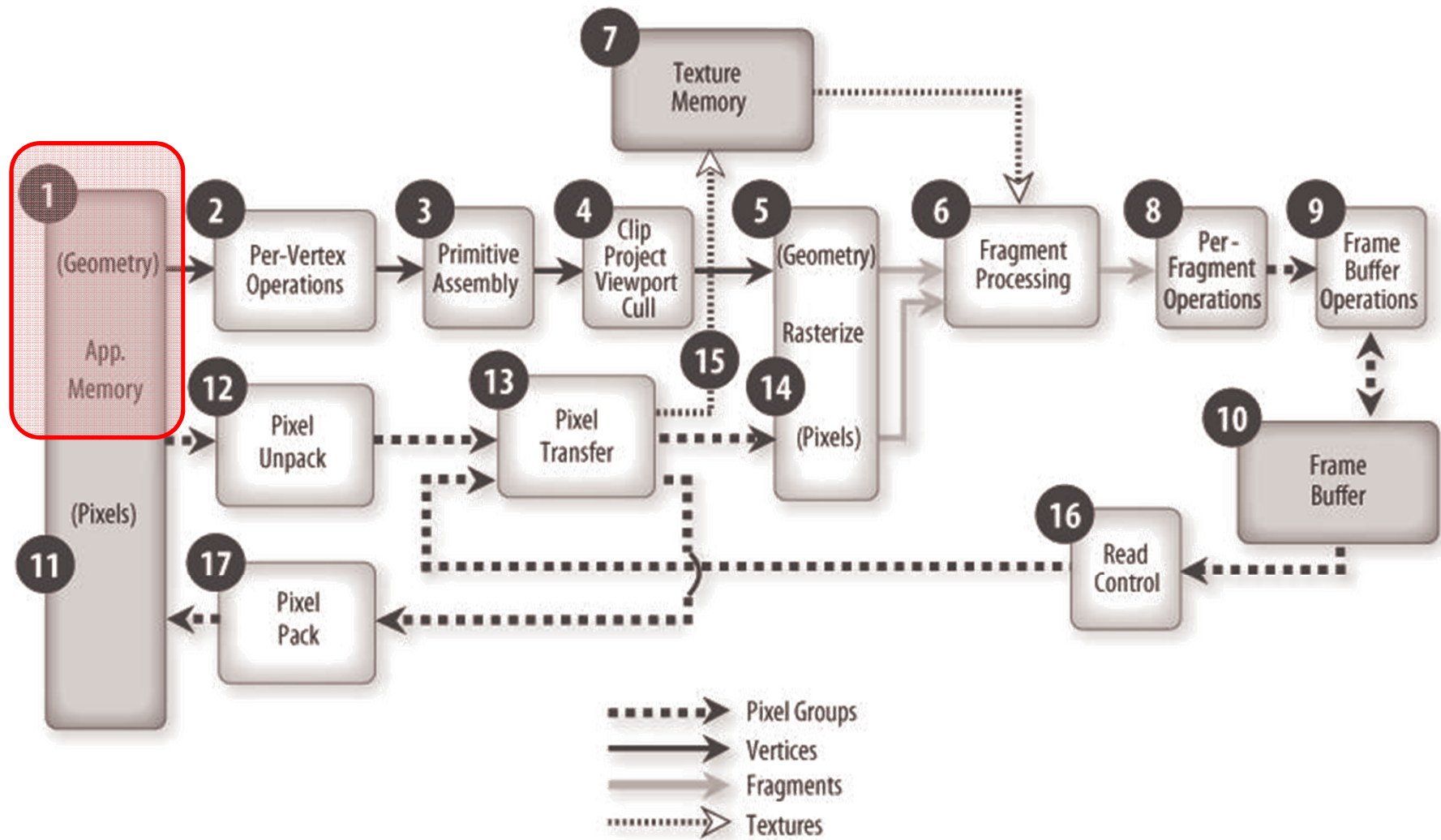
$$z_d = 0$$

- Els punts situats sobre zfar tenen

$$z_d = 1$$

PIPELINE OPENGL

Pipeline OpenGL



Pipeline OpenGL

1. Dibuix de primitives

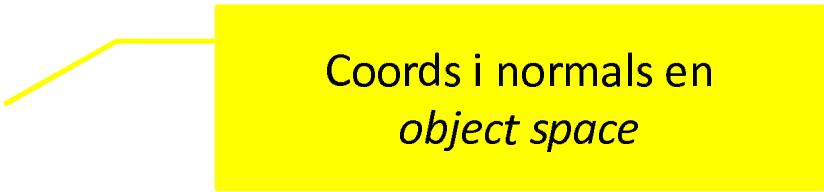
Les primitives gràfiques (punts, línies, polígons...) es poden enviar a OpenGL de diverses formes:

- Vèrtex a vèrtex: *glBegin, glVertex, glEnd*
- Vertex arrays: *glDrawArrays, glDrawElements...*
 - Vertex buffer object: vertex array emmagatzemat a la GPU
- Display lists: *glNewList, glCallList, glEndList*

Pipeline OpenGL

1. Dibuix de primitives

```
glBegin(GL_POLYGON);  
  glNormal3f(nx0, ny0, nz0);  
  glVertex3f(x0,y0,z0);  
  glNormal3f(nx1, ny1, nz1);  
  glVertex3f(x1,y1,z1);  
  ...  
glEnd();
```

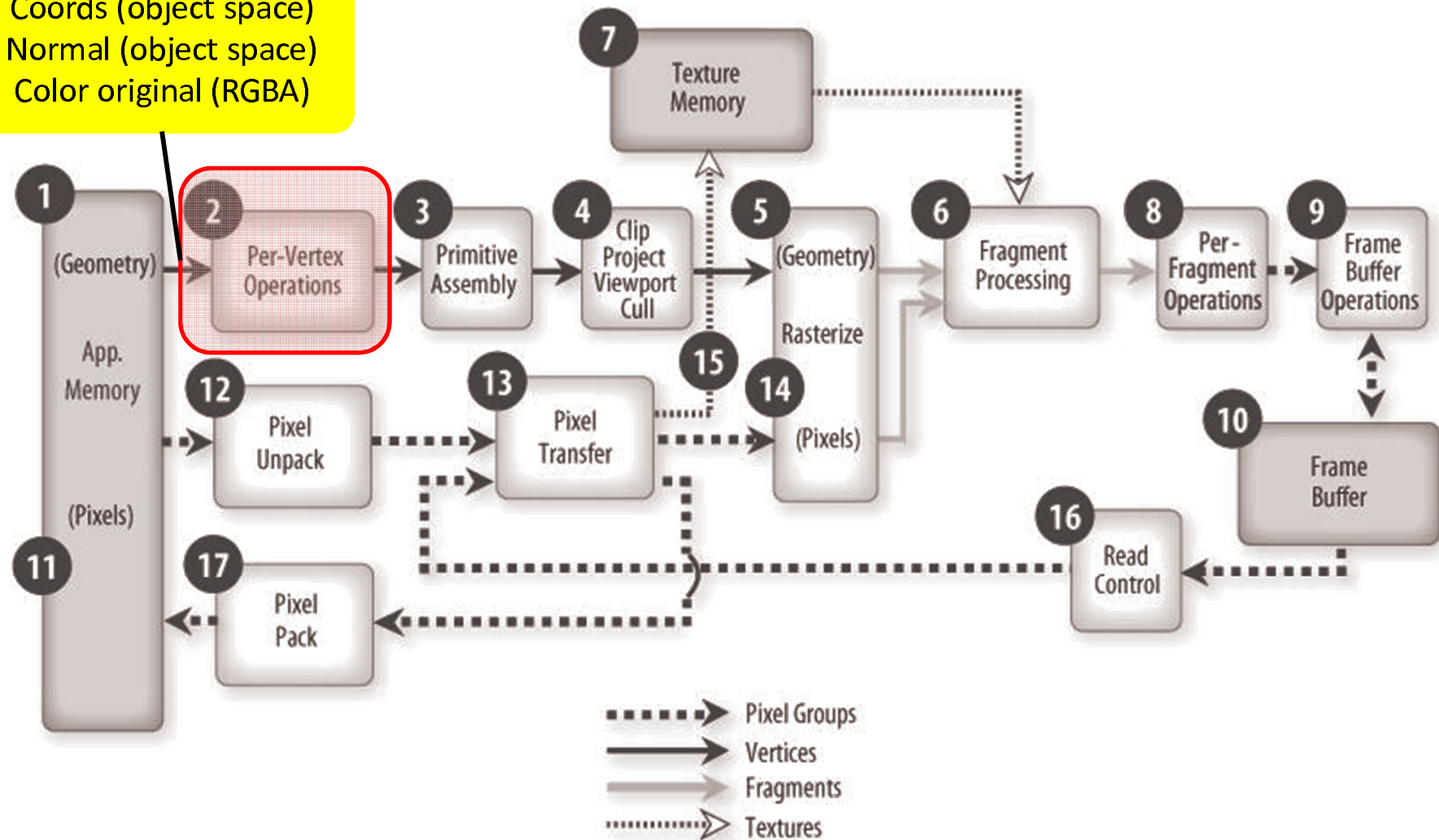


Coords i normals en
object space

Pipeline OpenGL

Vertices originals:

Coords (object space)
Normal (object space)
Color original (RGBA)

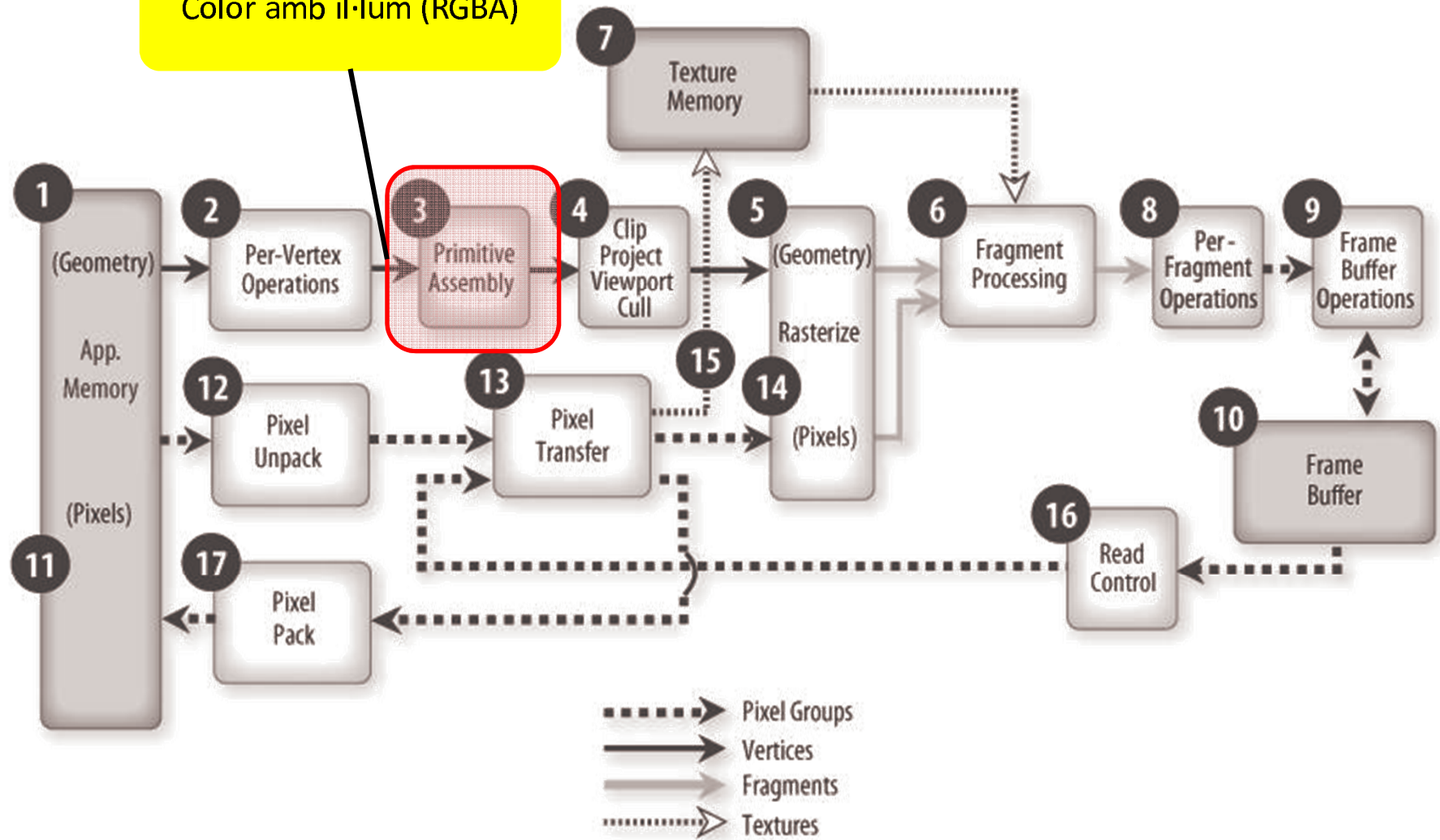


Pipeline OpenGL

2. Per-vertex operations

- Es transformen els vèrtexs (modelview i projection).
- Es transformen les normals (trasposta de l'inversa de la submatriu 3x3 de la modelview) i es calcula la il·luminació del vèrtex.
- Opcionalment:
 - Es generen coords de textura de forma automàtica.
 - Es transformen les coord de textura (texture matrix).

Vertexs transformats:
Coords (clip space)
Color amb il·lum (RGBA)

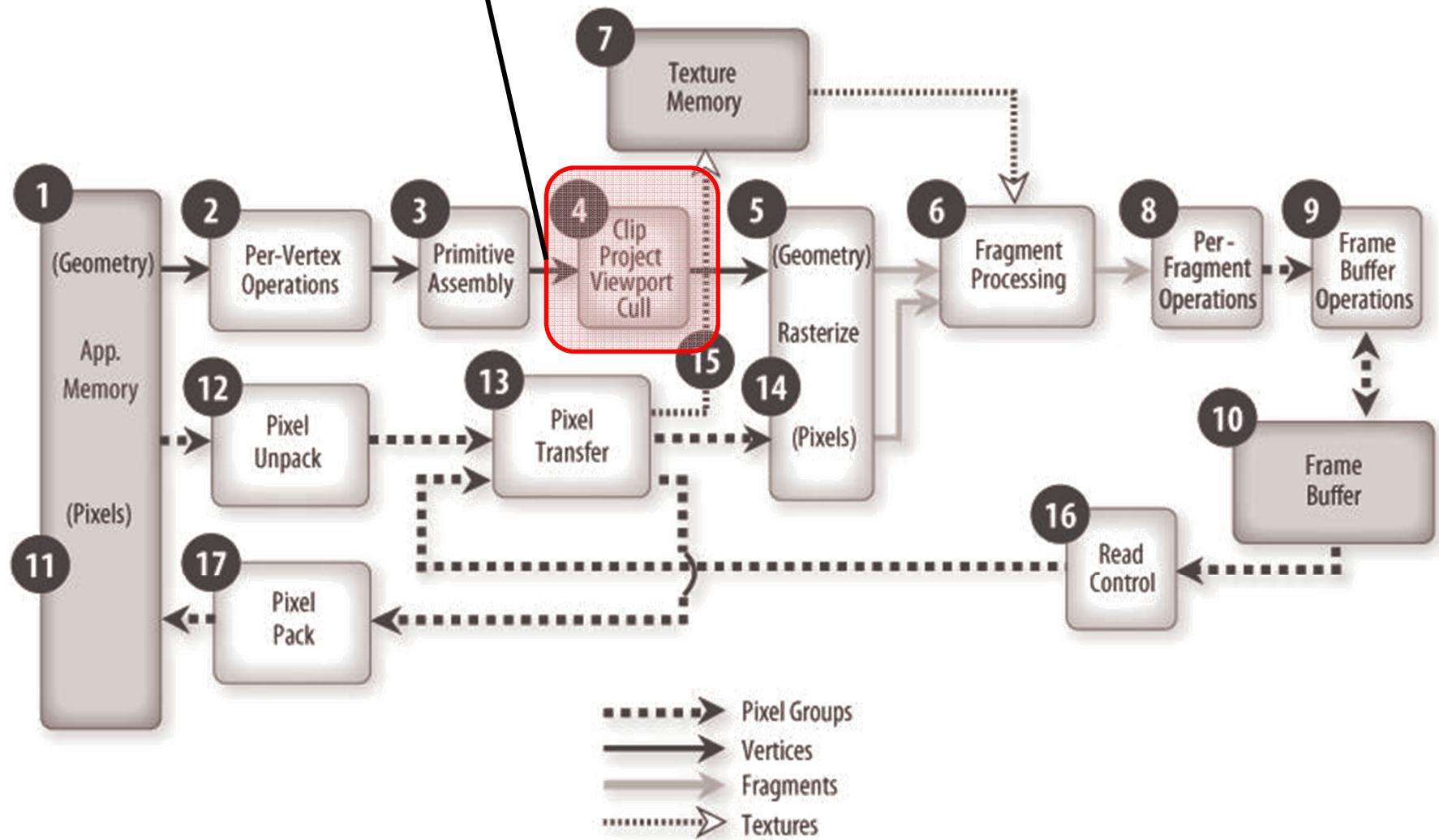


Pipeline OpenGL

3. Primitive assembly

- Els vèrtexs s'agrupen formant primitives.
- Cada primitiva GL_POINT, GL_LINES, GL_POLYGON requereix un clipping diferent.

**Primitives transformed:
(clip space) i il·luminades**

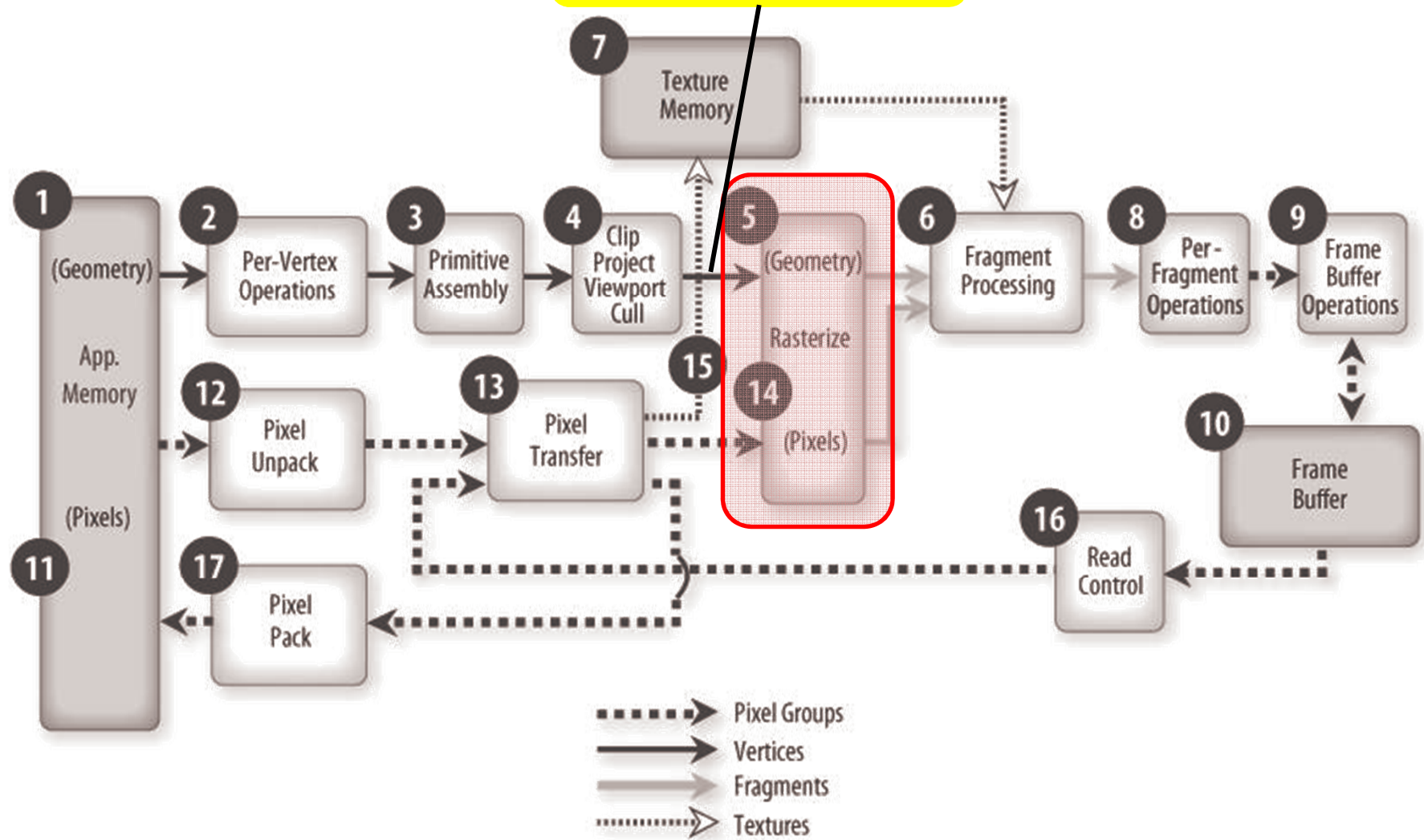


Pipeline OpenGL

4. Primitive processing

- Clipping (retallat) a la piràmide de visió.
- Divisió de perspectiva: es divideix (x,y,z) per w
- Viewport & depth transform → window coordinates
- Backface culling

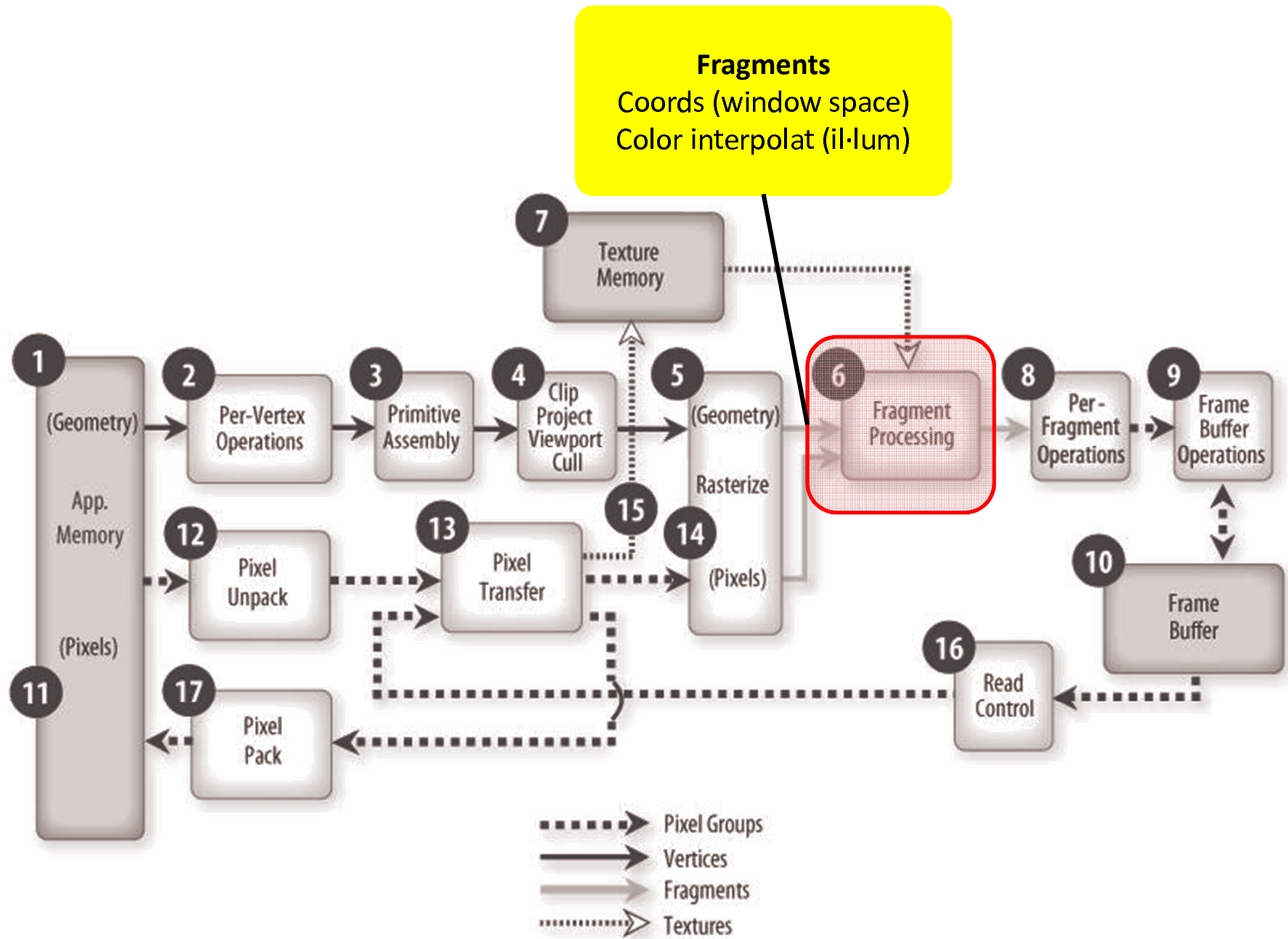
Primitives retallades
(window space)



Pipeline OpenGL

5. Rasterització

- Generació dels fragments corresponents a la primitiva retallada.
- Cada fragment té diversos atributs:
 - Coordenades (window space)
 - Color (interpolat si Gouraud)
 - Coordenades de textura (interpolades)



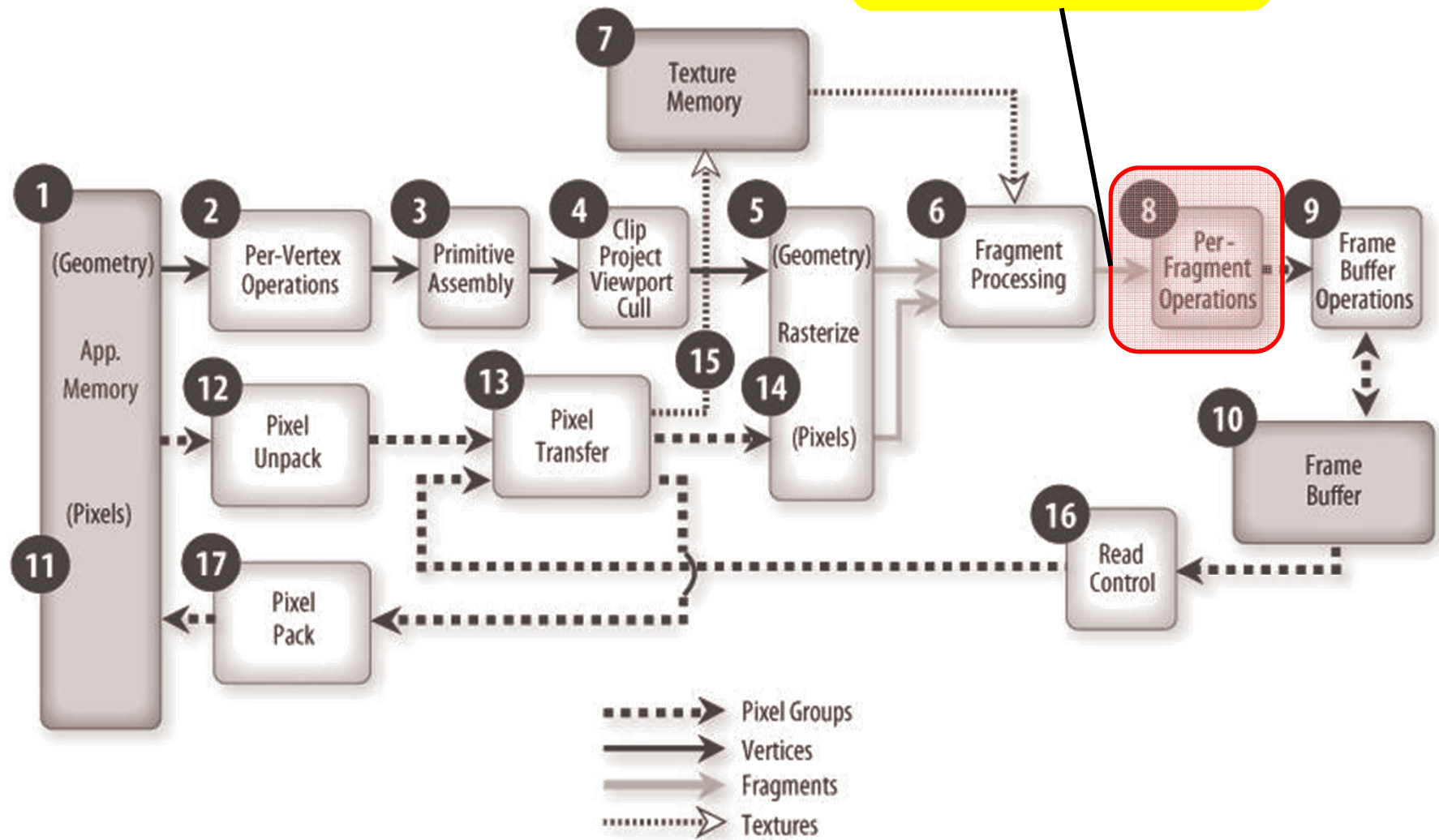
Pipeline OpenGL

6. Fragment processing (“shading”)

- Càlcul del color del fragment (texture mapping, boira, etc).

Fragments

Coords (window space)
Color (il·lum + textura)



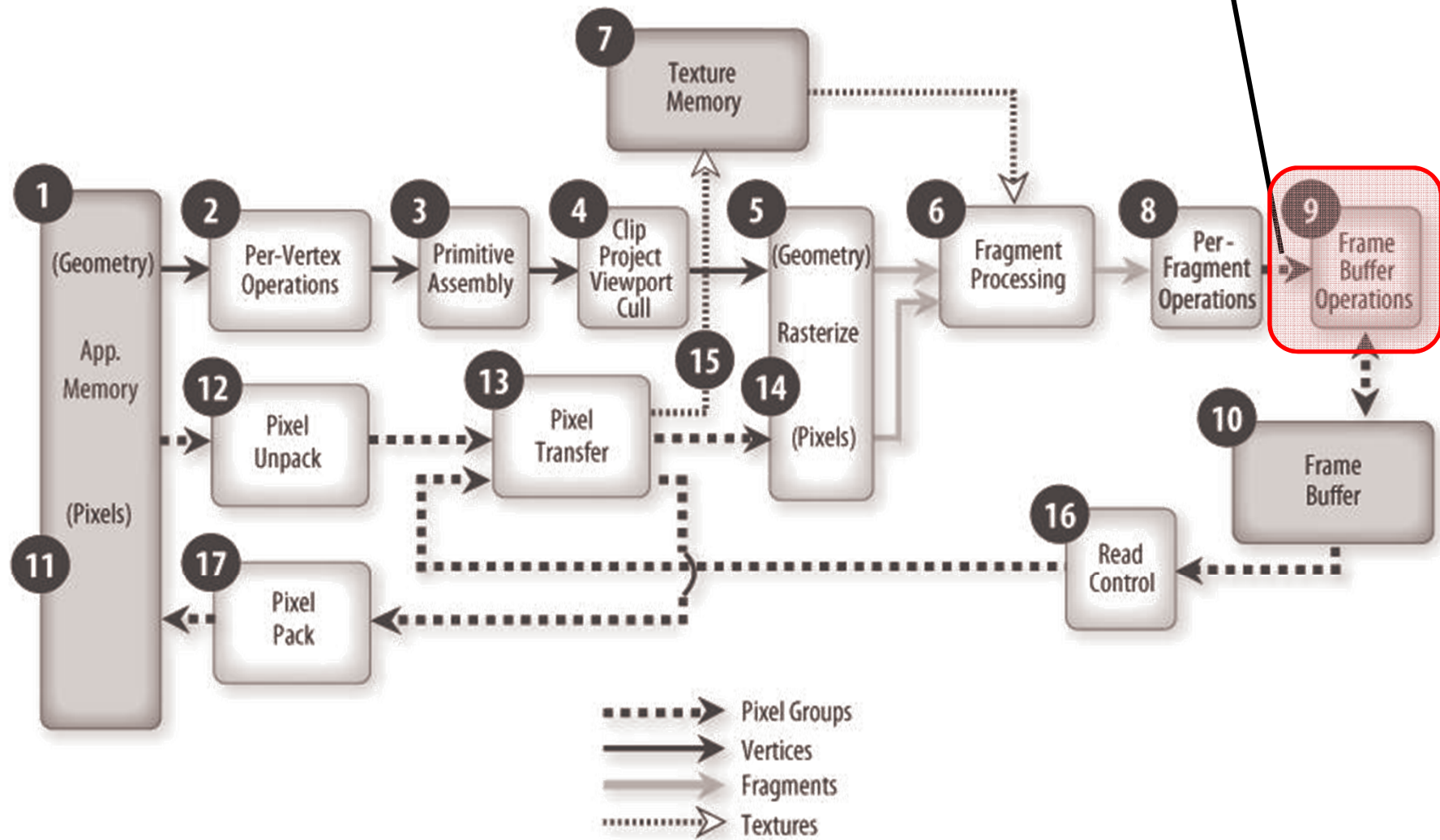
Pipeline OpenGL

8. Per-fragment operations (“raster operations”)

- Pixel ownership
- Scissor test
- Alpha test
- Stencil test
- Depth test (test Z-buffer)
- Blending
- Dithering
- Logical Ops (`glLogicOp`)

Fragments visibles

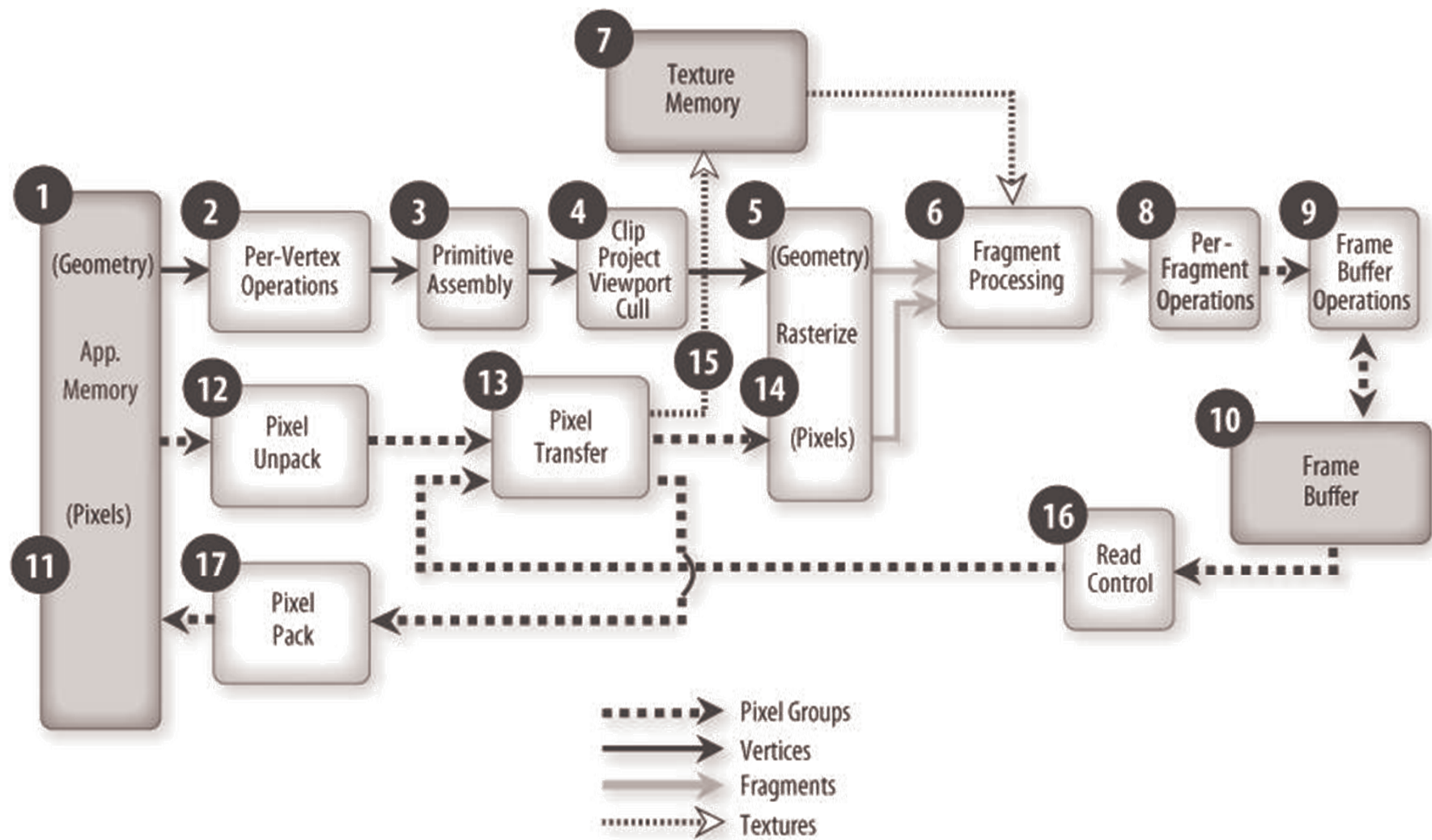
Coords (window space)
Color (il·lum + textura +
alpha blending)



Pipeline OpenGL

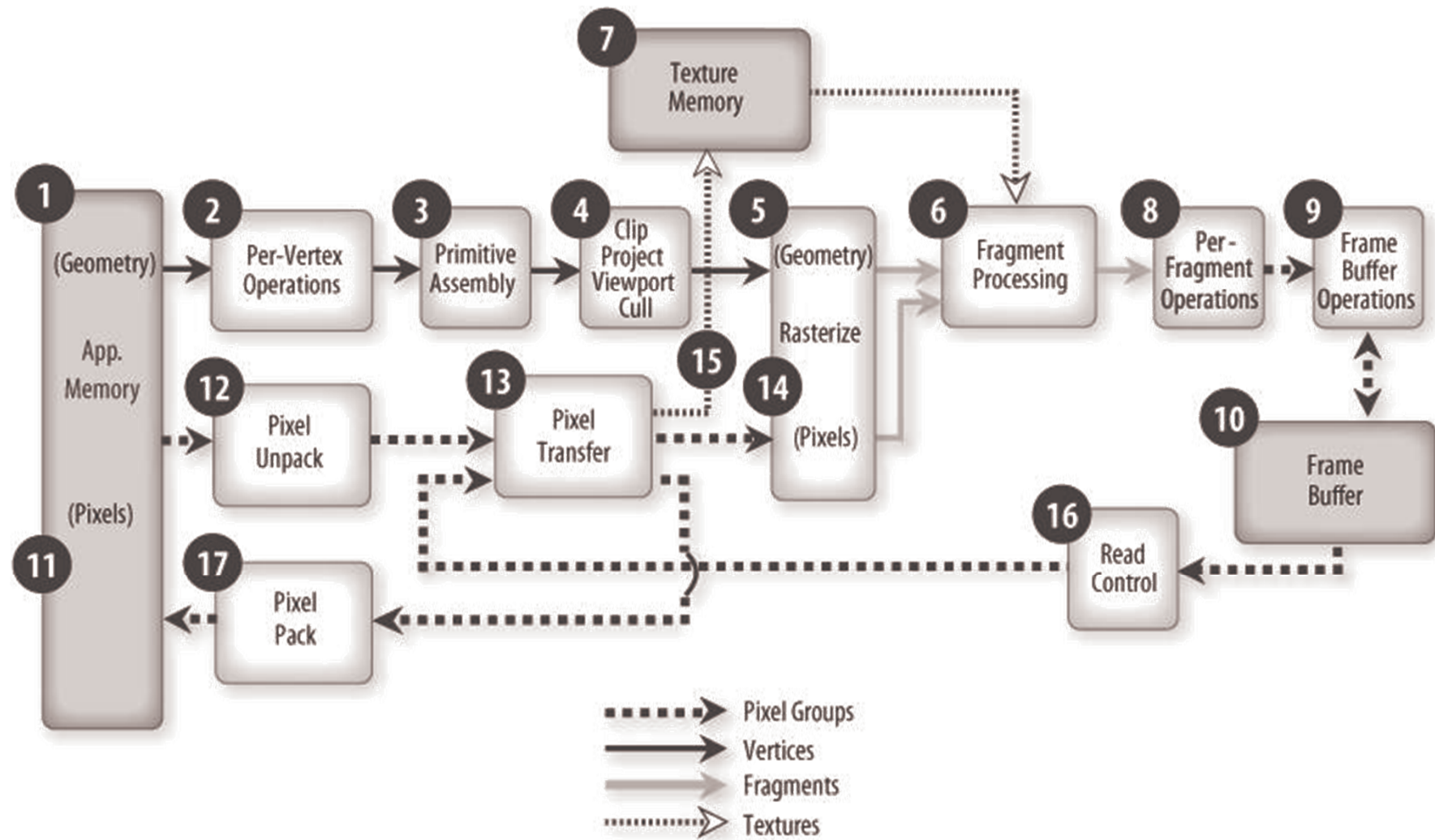
9. Frame buffer operations

- Es modifiquen els buffers que s'hagin escollit amb `glDrawBuffers`
- Es veu afectada per `glColorMask`, `glDepthMask`...

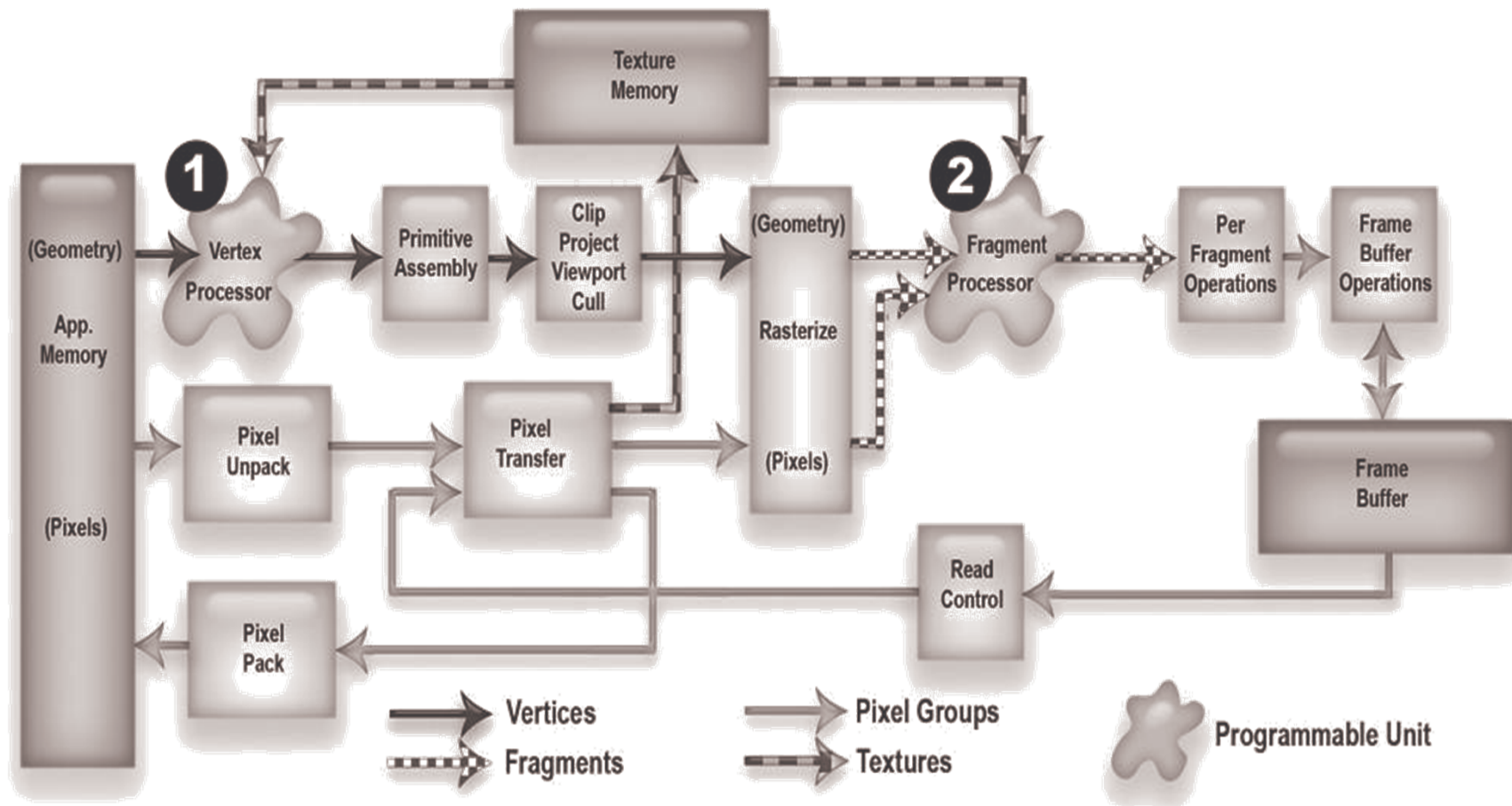


PIPELINE PROGRAMMABLE

Pipeline fix...



i pipeline programmable



Pipeline programable

- **Vertex processor:** part de la GPU capaç d'executar un programa per cada vèrtex.
- **Fragment processor:** part de la GPU capaç d'executar un programa per cada fragment.
- **Shader:** codi font d'un programa (o part) per la GPU
 - Vertex shader, fragment shader, geometry shader
- **Program:** executable d'un programa per la GPU
 - Vertex program, fragment program, geometry program

Pipeline programable

- Quan s'activa un **vertex program**, en lloc d'executar-se les operacions **per-vèrtex** prefixades d'OpenGL, s'executa el vertex program.
- Quan s'activa un **fragment program**, en lloc d'executar-se les operacions **per-fragment** prefixades d'OpenGL, s'executa el fragment program.
- Normalment el vertex/fragment program més senzill haurà de reproduir part de la funcionalitat fixa d'OpenGL.