Càmera i Visualització

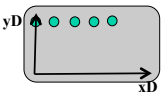
Càmera Il·luminació

Model Geomètric
"Model Fronteres"

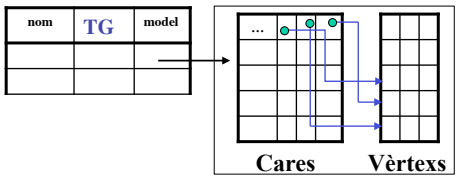
Visualització

OpenGL

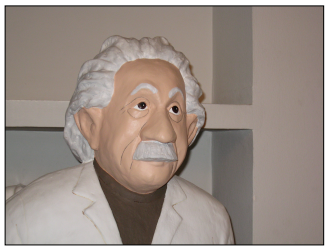
Píxel(x,y,c)



taula [res_x][res_y] de color
color = (r,g,b) r,g,b en [0…1]



Objectes en SCA o Model

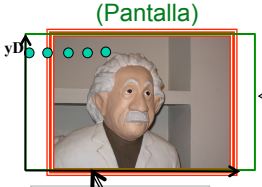


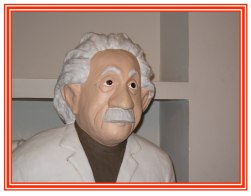
1. Posició, orientació
2. Òptica

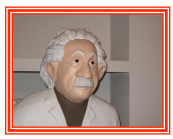
3. Fer la Foto

4. Emmarcar

Finestra OpenGL
(Pantalla)







Viewport/Vista

Visualització OpenGL

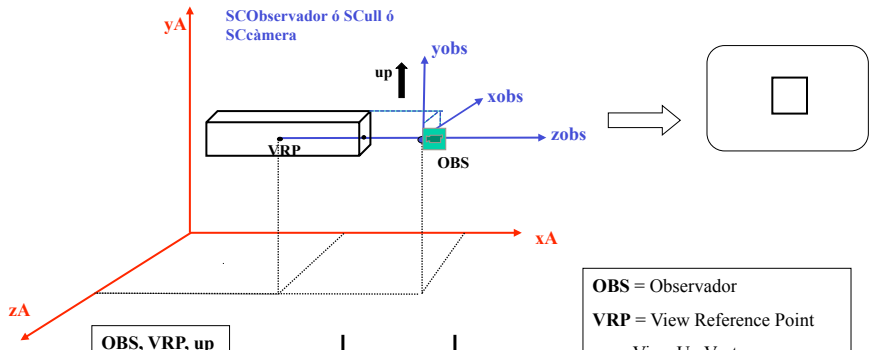
1. **Netejar finestra** → glClear(GL_COLOR_BUFFER_BIT | ...);

2. **Definir càmera i viewport**
   - 2a **Definir Vista/Viewport** → glViewport(…)
   - 2b **Posició i orientació**
     - ***Model-view Transformation***
   - 2c **Optica**
     - ***Projection Transformation***

3. **Recorregut de l'escena i pintar cares** →

```
glBegin (GL_POLYGON)
    glVertex (v1)
    ....
    glVertex (vn)
glEnd
```


View transformation: *posició+orientació*



OBS = Observador

VRP = View Reference Point

up = View Up Vector

up "indica" la direcció de l'eix vertical de la Càmera (inclinació)

OBS, VRP, up

↓

gluLookAt (**OBS**, **VRP, up**)

↓

TC/MV

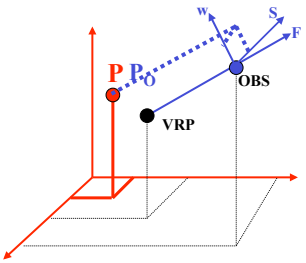
TC

GL_MODELVIEW

$P_O = TC\ P_A$

Com es calcula la matriu TC?

gluLookAt() → càlcul TC



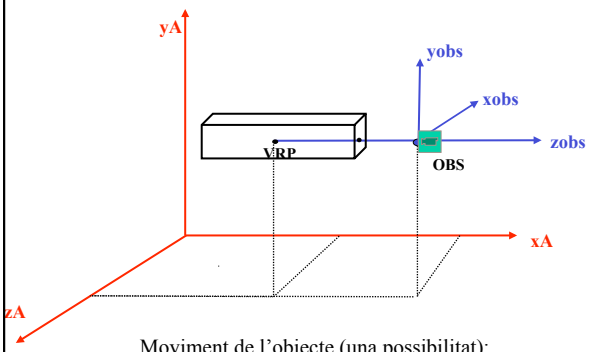
$$TC = \begin{bmatrix} s.x & s.y & s.z & 0 \\ w.x & w.y & w.z & 0 \\ F.x & F.y & F.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \text{Trans } (-\text{Obs})$$

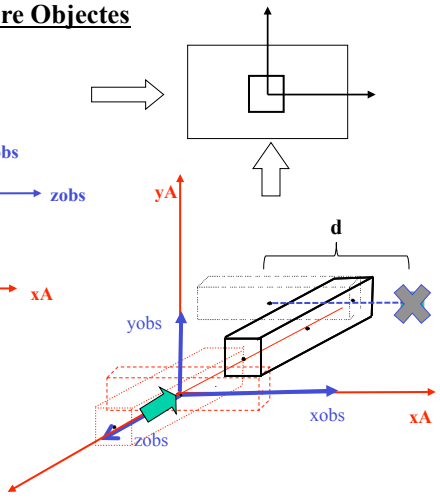
F = OBS - VRP = (F.x,F.y,F.z) **F= F / || F ||**

s = up x F **s = s / ||s||**

w = F x s

Posicionament de la Càmera: opció Moure Objectes





Moviment de l'objecte (una possibilitat):

1) Tr(**-VRP**)

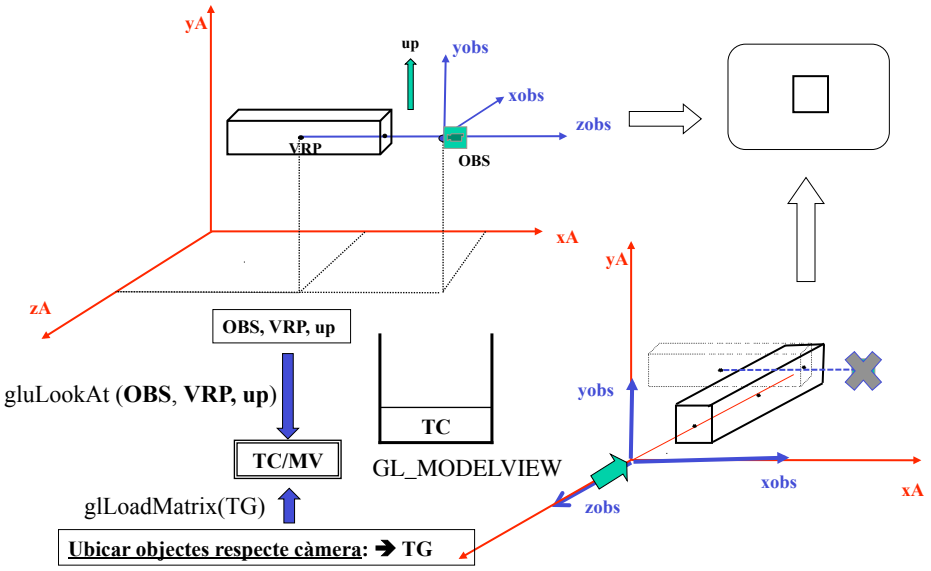
2) Gy(-90)

3) Tr (0,0,-d)

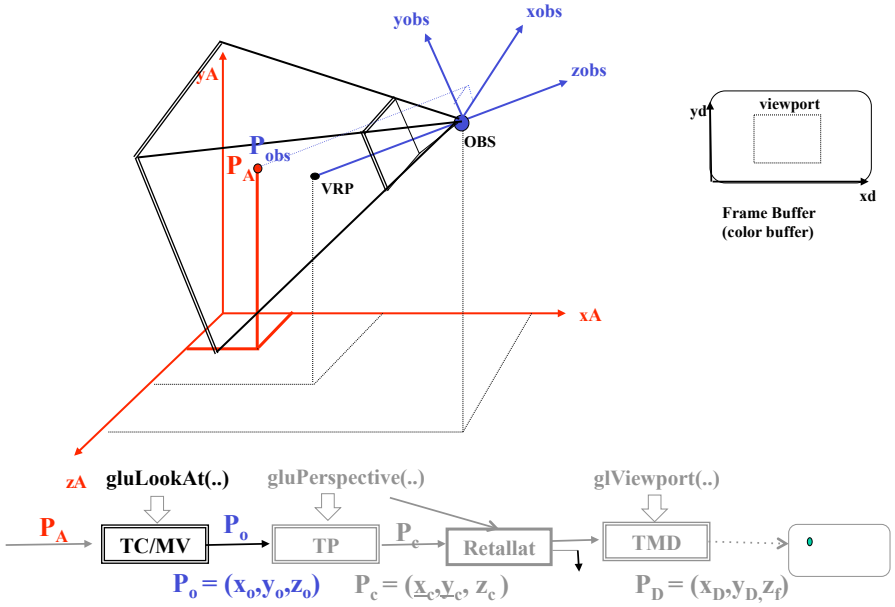
TC= T(0,0,-d) G_Y (-90) T(**-VRP**)

(angles positius => girs anti-horari)

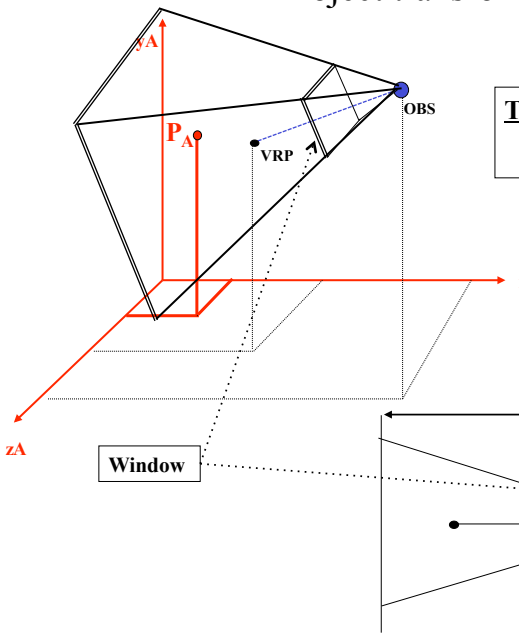
View transformation: *posició+orientació*



Visualització OpenGL: fonaments



Project transformations: òptica OpenGL



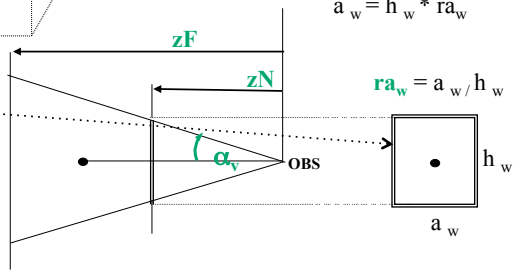
Tipus de camara: Perspectiva

α_v (FOV= $2\alpha_v$) , zNear, zFar, ra_w

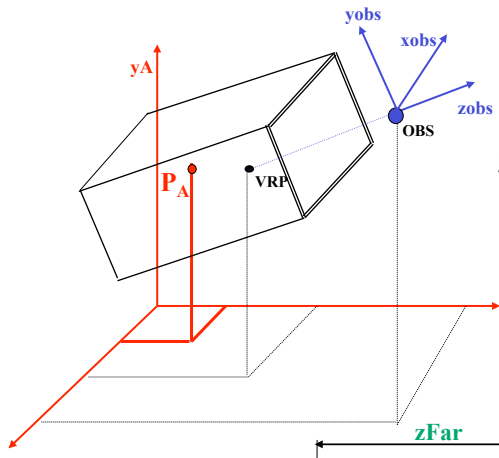
gluPerspective (FOV, ra_w, zN, zF)

$h_w = 2\ zN\ tg(\alpha_v)$

$a_w = h_w * ra_w$



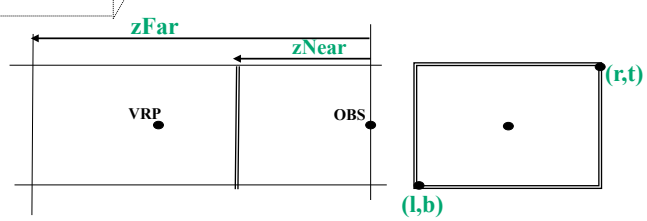
$ra_w = a_w / h_w$



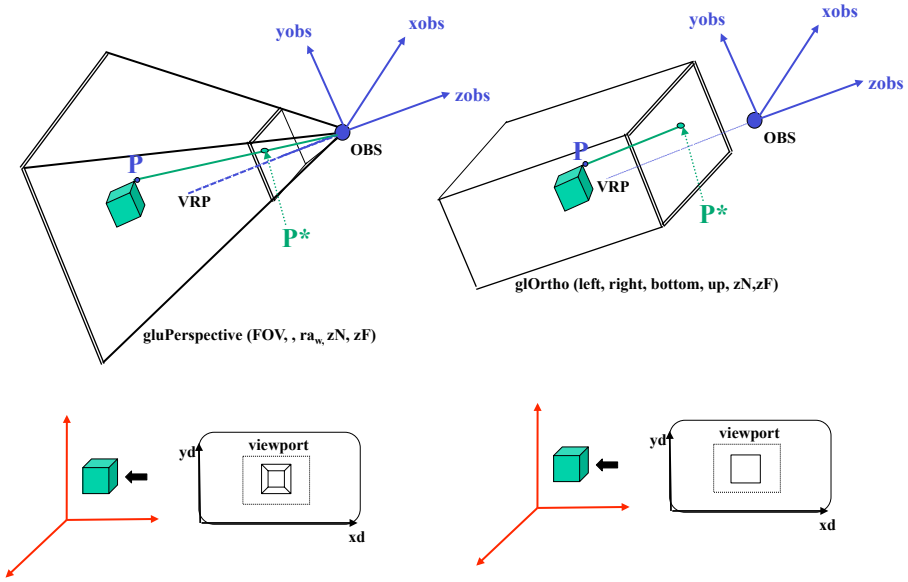
Tipus de camara: Axonomètrica:

window , zNear, zFar

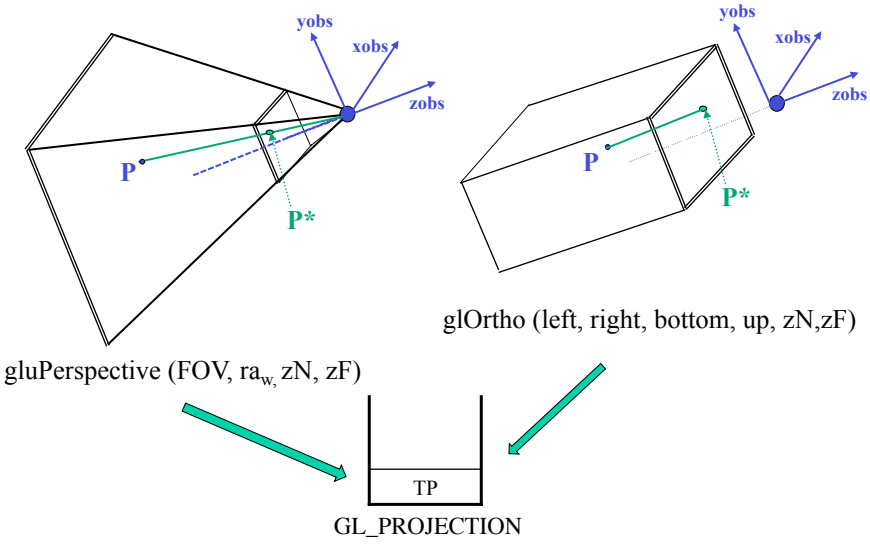
glOrtho (left, right, bottom, top, zN,zF)

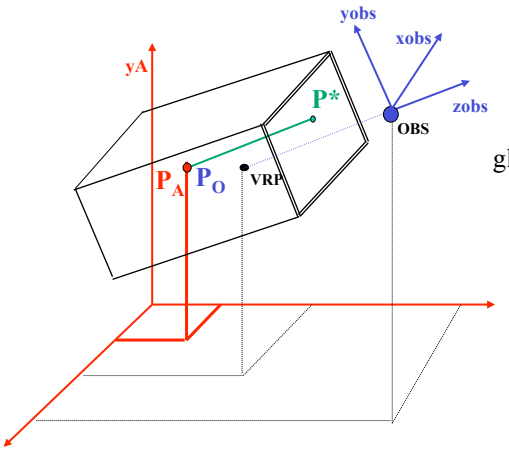


Project transformations



Project transformations: càmera OpenGL



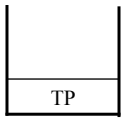


Tipus de camara: Axonomètrica:

window , zNear, zFar

glOrtho (left, right, bottom, top, zN,zF)

TP



GL_Projection

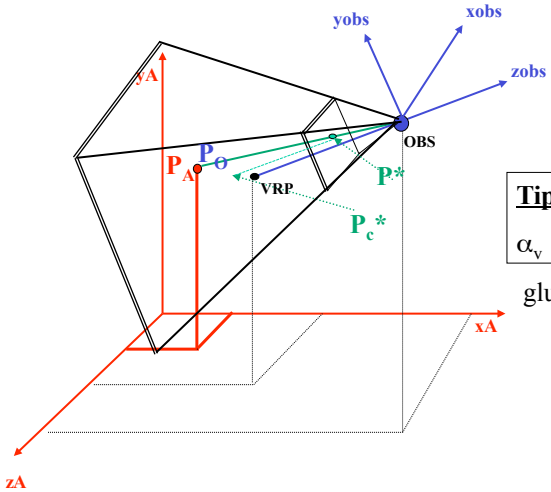
$P_c^* = TP\ (TC\ P_A) = TP\ P_O$

$$TP_o = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

a=2/(r-l) b=2/(t-b)

c= 2 /(f-n)

d = (n+f)/ (f-n)



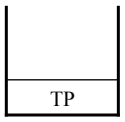
$ra_w = ra_v$ per no deformació

Tipus de camara: Perspectiva:

α_v (FOV= $2\alpha_v$) , zNear, zFar, ra_w

gluPerspective (FOV, ra_w, zN, zF)

TP



GL_Projection

$P_c^* = TP\ (TC\ P_A) = TP\ P_O$

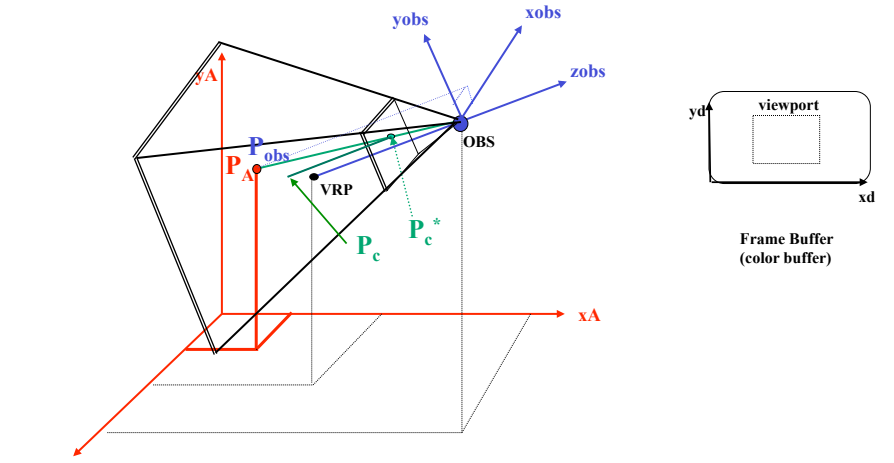
$$TP = \begin{pmatrix} 1/ra*a & 0 & 0 & 0 \\ 0 & 1/a & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

a = tg (FOV/2)

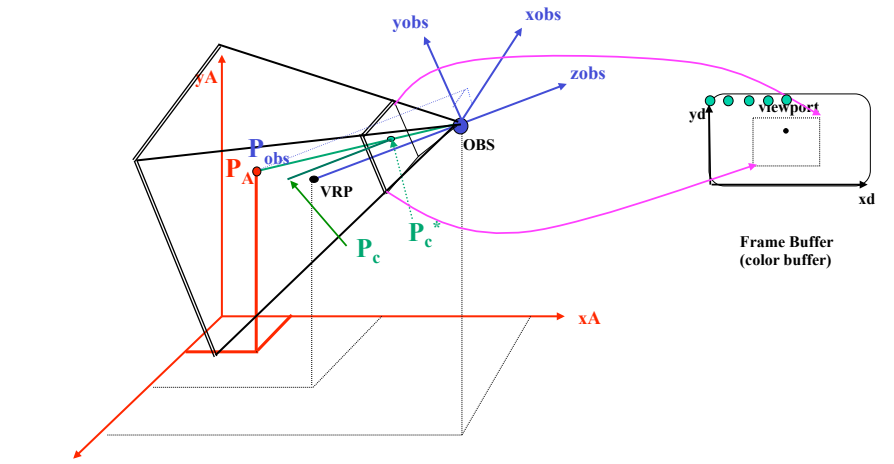
c = (f+n)/(n-f)

d = 2 n f /(n-f)

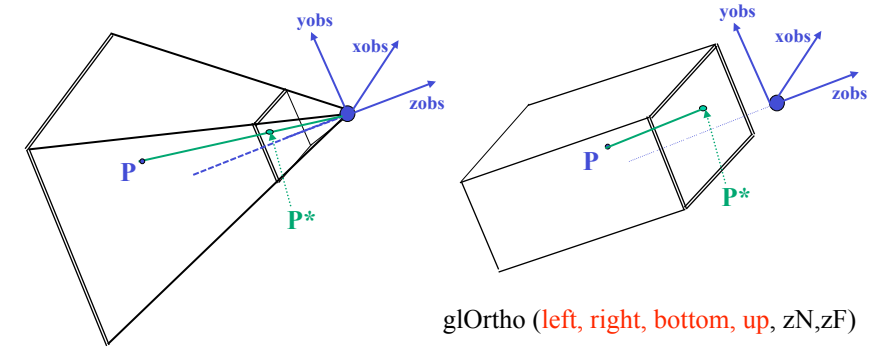
Visualització OpenGL: fonaments



Visualització OpenGL: fonaments



Project transformations: càmera OpenGL



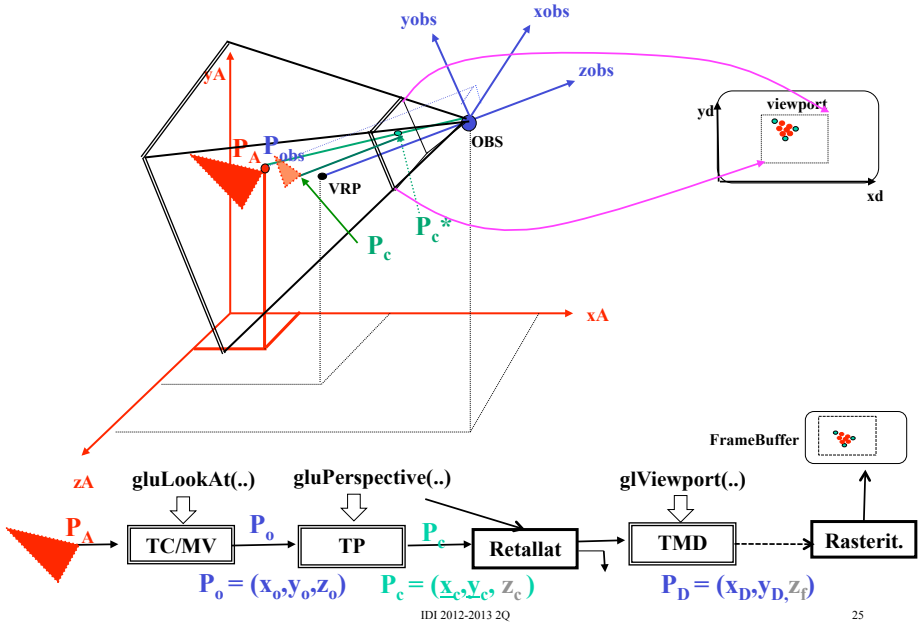
glOrtho (left, right, bottom, up, zN,zF)

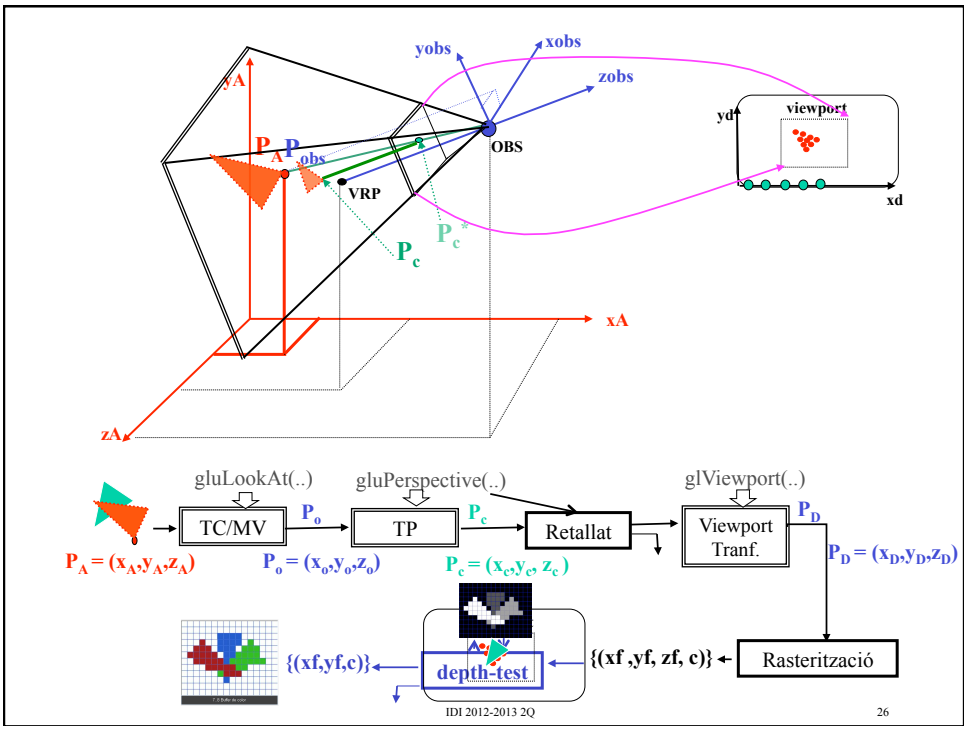
gluPerspective (FOV, ra_w, zN, zF)

TP

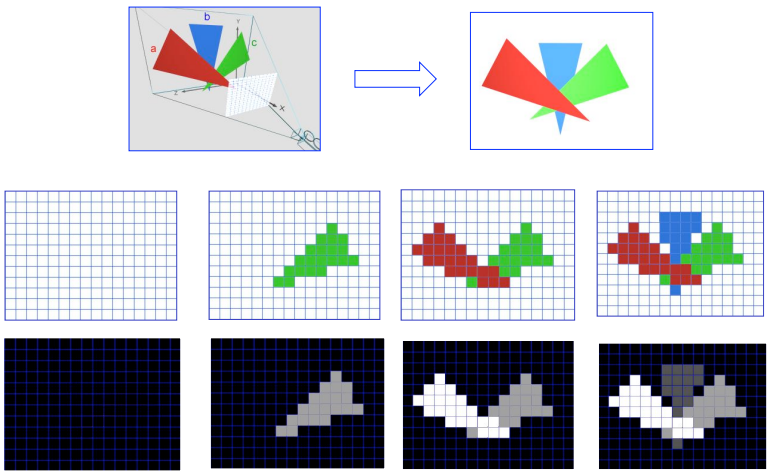
GL_PROJECTION

Visualització OpenGL: fonaments





Depth Buffer (z-buffer)



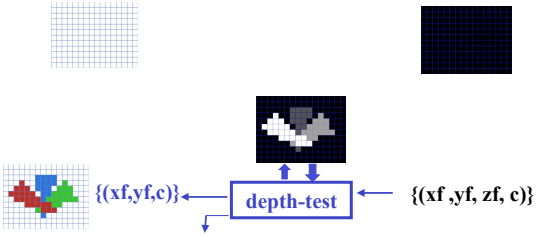
Depth Buffer (z-buffer)

- Dos buffers de la mateixa resolució que la pantalla

  Buffer color (frame_buffer) — (r, g, b)∈[0, 2^n-1]

  Buffer profunditats (depth_buffer) — z ∈ [0, 2^{nz}-1]

1. **Inicialitzar al color de fons**
1. **Inicialitzar al més lluny possible**

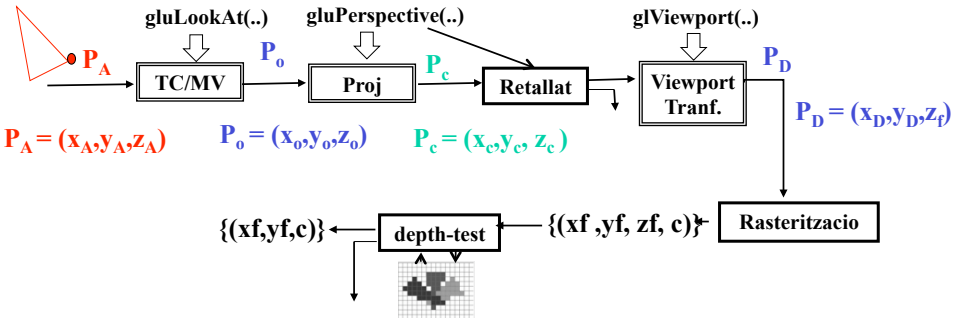



```
si zf < depth_buffer(xf,yf) llavors
    depth_buffer (xf,yf) := zf
    color_buffer (xf,yf) := c
fsi
```

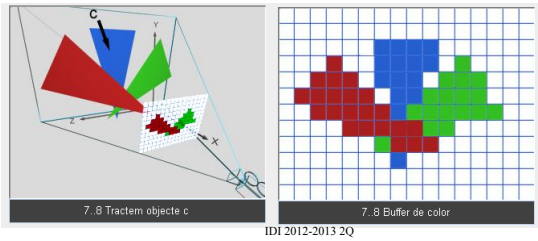

Depth Buffer (z-buffer)

- OpenGL
  - Activar z-buffer: glEnable(GL_DEPTH_TEST)
  - esborrar tots dos buffers
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

Visualització OpenGL: Pipeline bàsic



Cal conservar informació (una "z") per fer l'EPA!!!



Visualització OpenGL

1. Netejar finestra → glClear(GL_COLOR_BUFFER_BIT | ...);

2. Definir càmera I viewport

2a Definir Vista/Viewport → glViewport(…)

2b Posició i orientació

Model-view Transformation → gluLookAt (...) o

"movent" escena➔ donar la MV

2c Optica

Projection Transformation → gluPerspective(…) o glOrtho(…)

3. Recorregut de l'escena i pintar cares →

```
glBegin (GL_POLYGON)
    glVertex (v1)
    ....
    glVertex (vn)
glEnd
```

Pipeline d'OpenGL