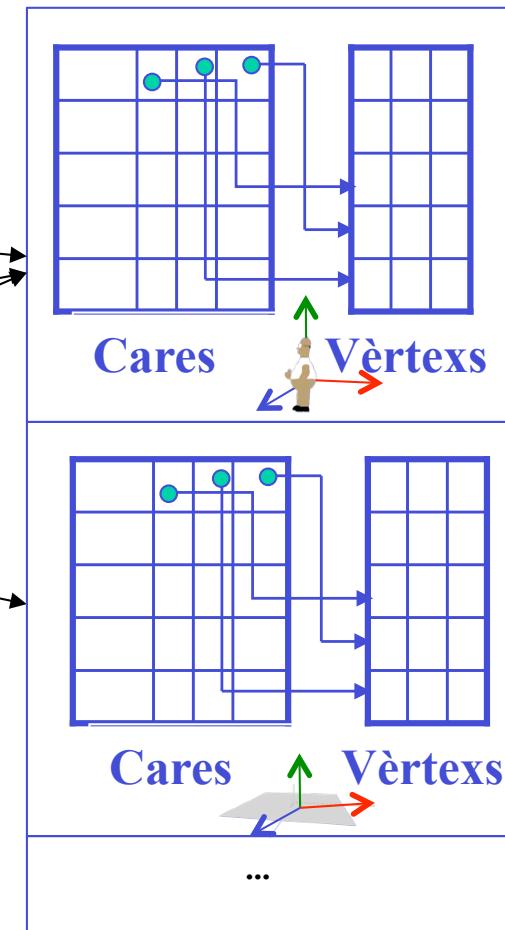
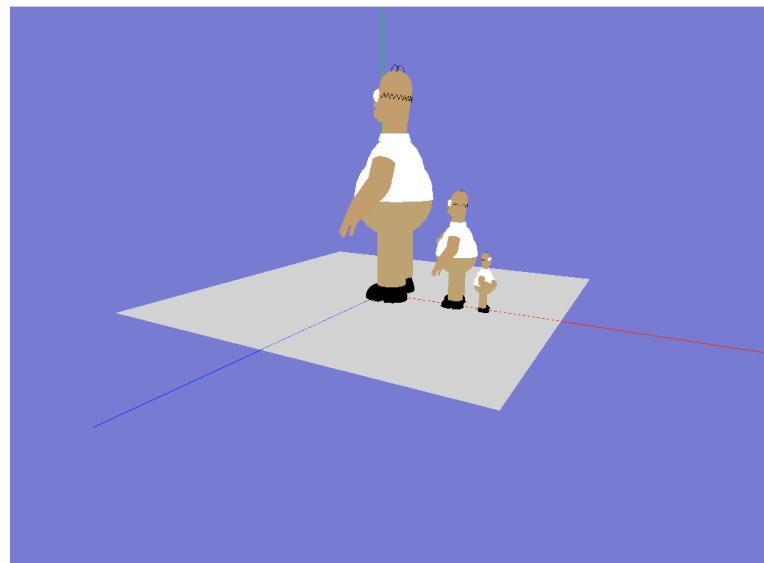


# Classe 3: contingut

- Visualització: breu repàs.
- Càmera (1)
  - Posicionament: OBS, VRP, up
  - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs (2 i fi)
- Exemple definició de càmera
- Exercici de TG pendent

# Escenes: Objectes en SCM. Com fem per pintar?

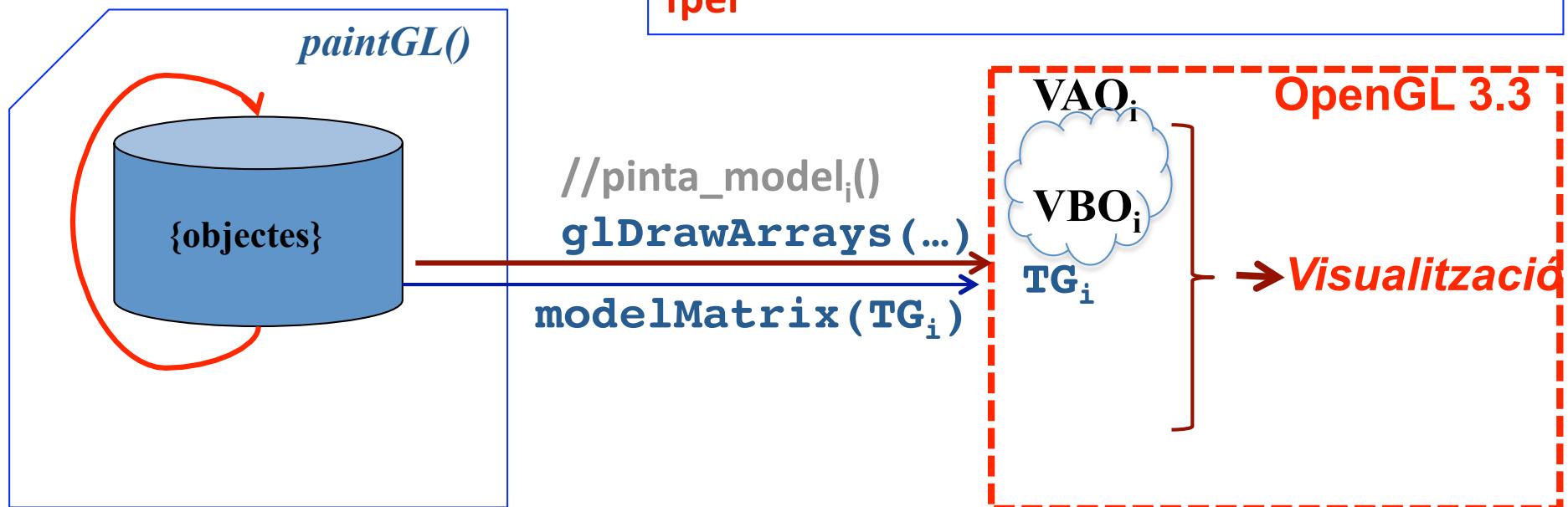
nom	...	Param per $TG_i$	model
h1			
h2			
h3			
terra			



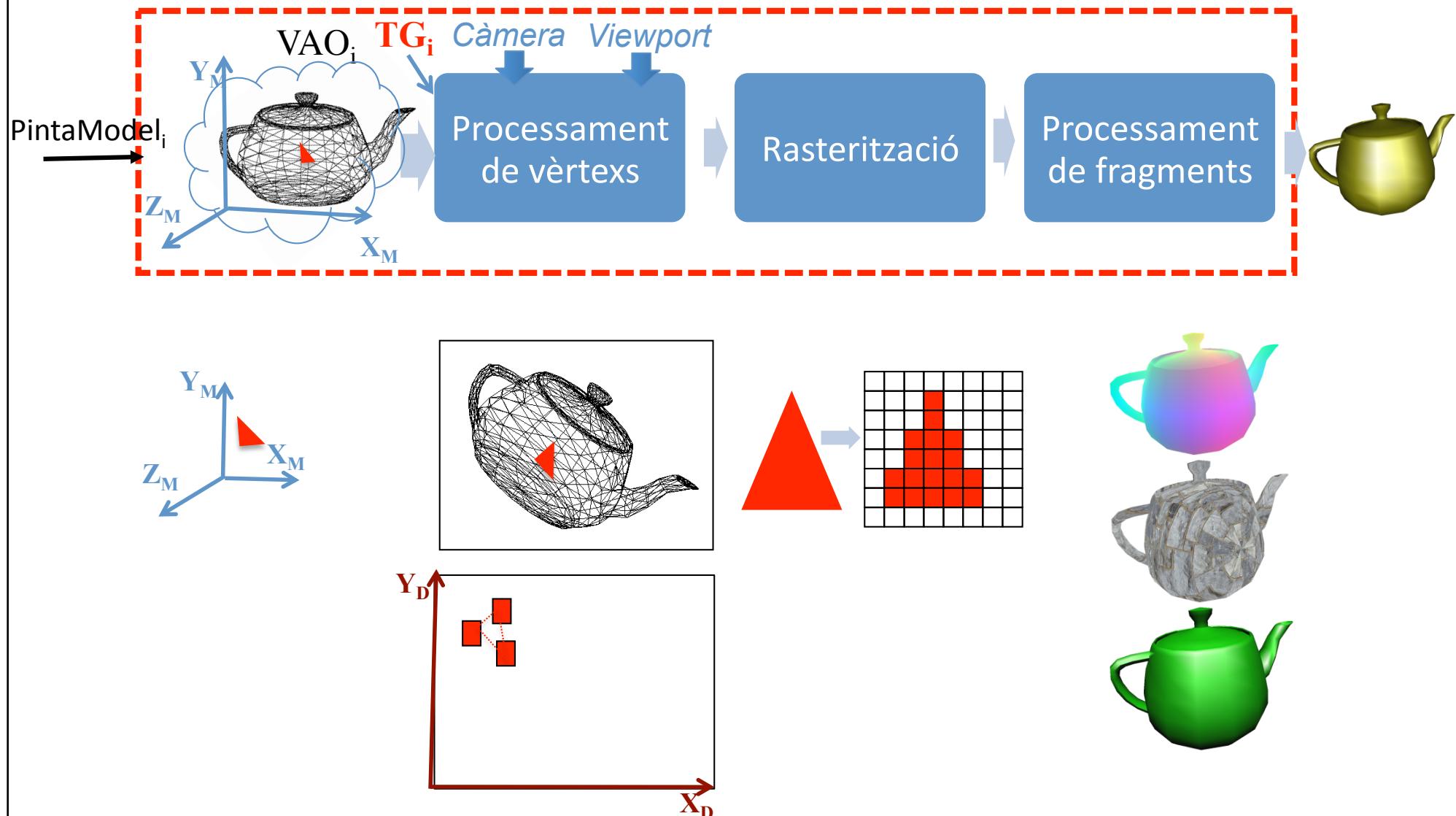
# Escenes: Objectes en SCM. Com fem per pintar?

```
// un únic VAO, per cada model  
// CreateBuffers();  
per cada model  
  crear i omplir VAOi,VBOi  
fper
```

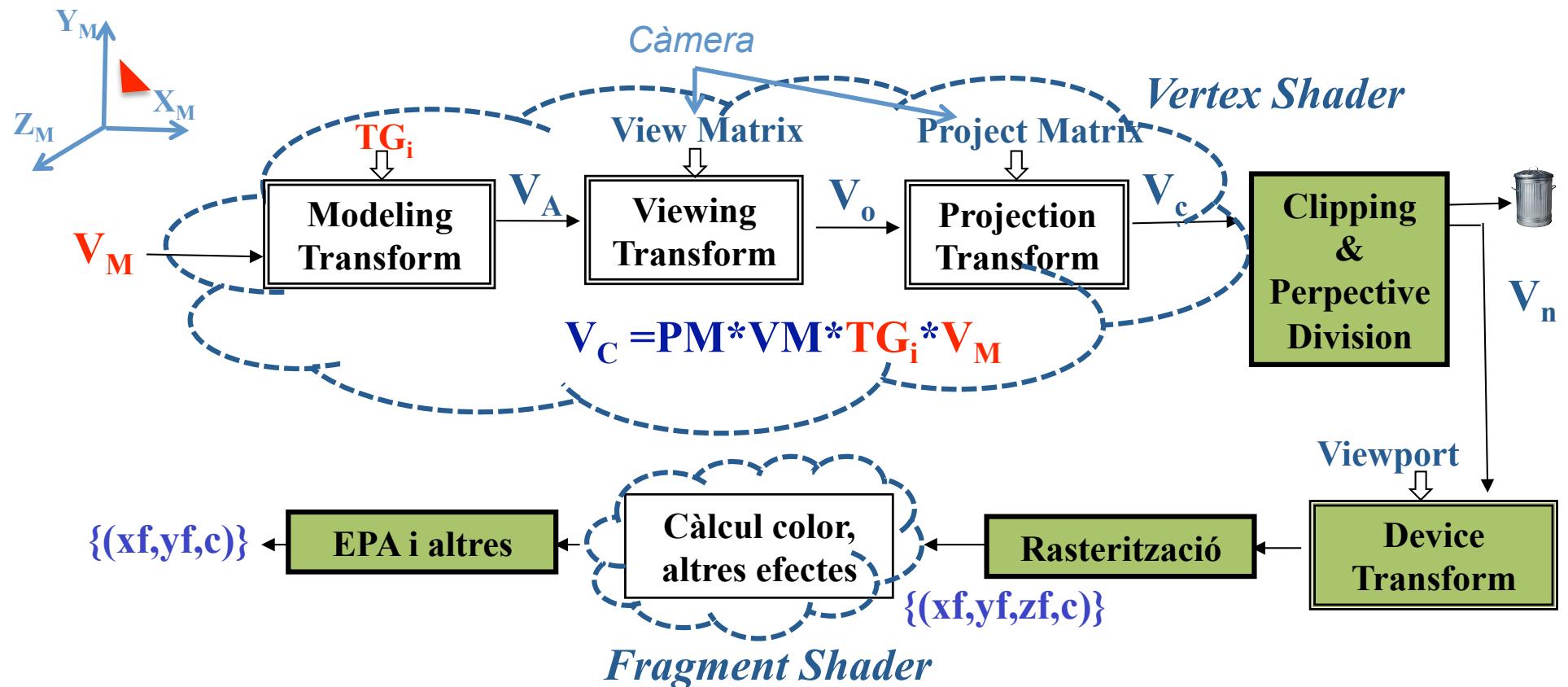
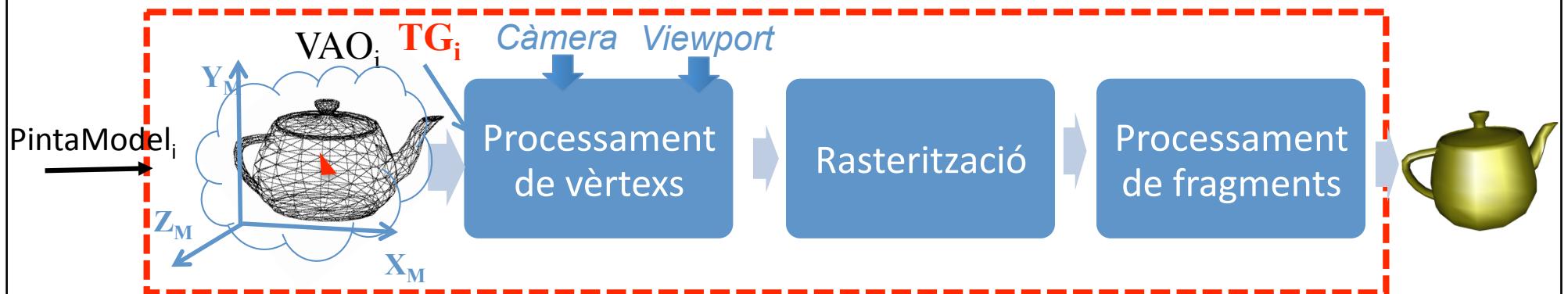
```
//paintGL ();  
per cada objectei  
  //Calcular la TGi a aplicar a model  
  modelTransformi(TGi);  
  // Indicar a OpenGL la TGi  
  modelMatrix(TGi); //envia uniform  
  pinta_modeli(); //el seu VAOi  
fper
```



# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3

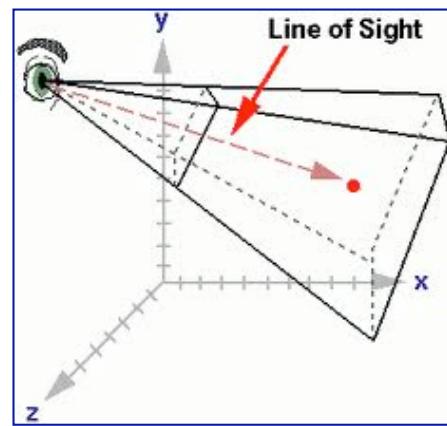


# Classe 3: contingut

- Visualització: breu repàs.
- **Càmera (1)**
  - Posicionament: OBS, VRP, up
  - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs (2 i fi)
- Exemple definició de càmera
- Exercici de TG pendent



# Posicionament de la càmera (1): OBS, VRP, up

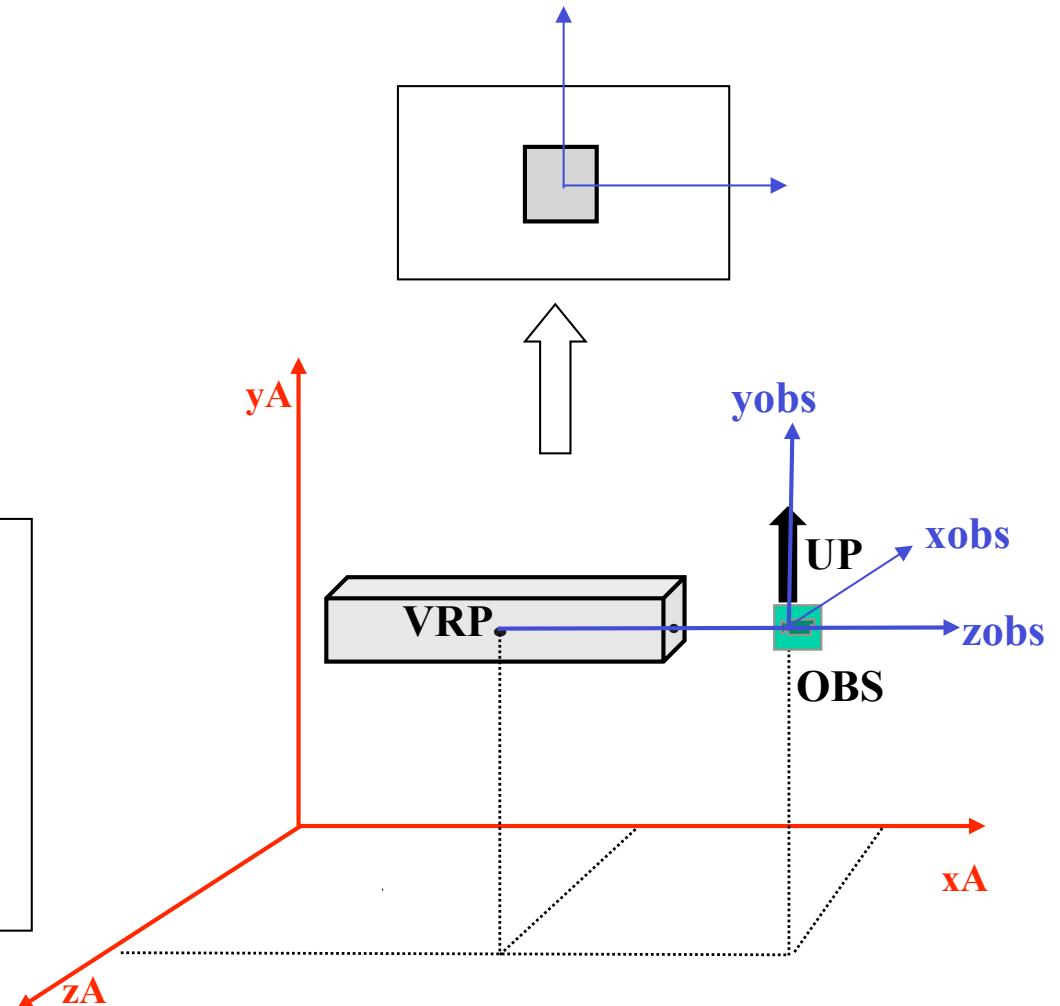


**OBS** = Observador

**VRP** = View Reference Point

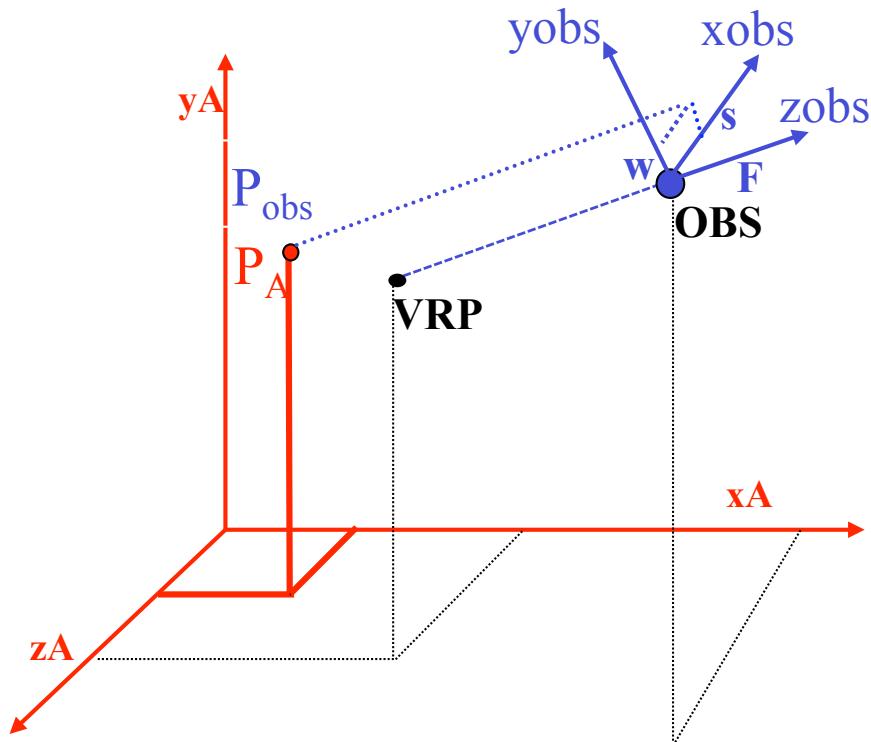
**up** = View Up Vector

**up** “indica” la direcció de l’eix vertical de la Càmera (inclinació)



*Per a la determinació dels seus valors suposem que tots els objectes de l’escena estan al seu lloc ➔ en SCA*

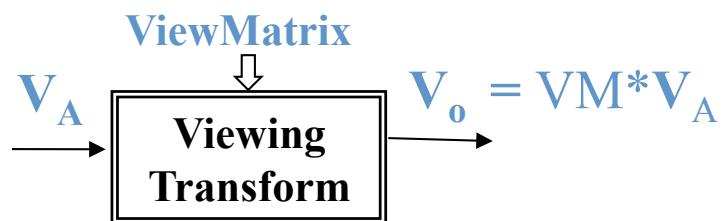
# OBS, VRP, up → Càcul de la View Matrix



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

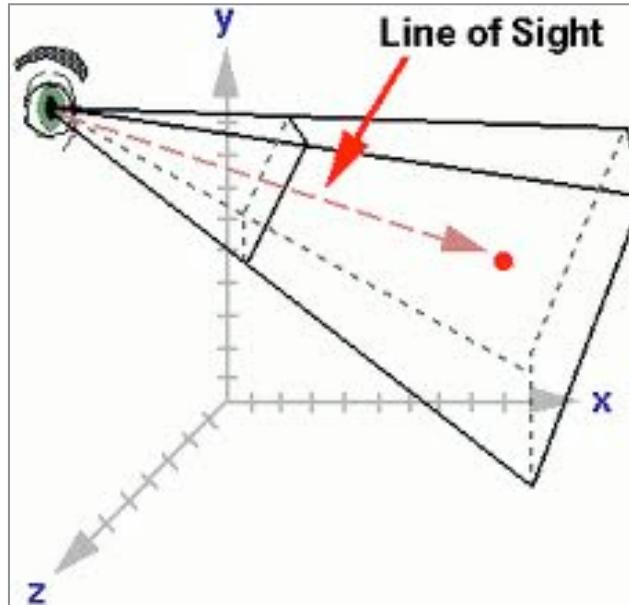
$$\begin{aligned} F &= OBS - VRP = (F.x, F.y, F.z) & F &= F / \| F \| \\ s &= up \times F & s &= s / \| s \| \\ w &= F \times s \end{aligned}$$

```
VM = lookAt(OBS, VRP, up);
viewMatrix(VM);
```

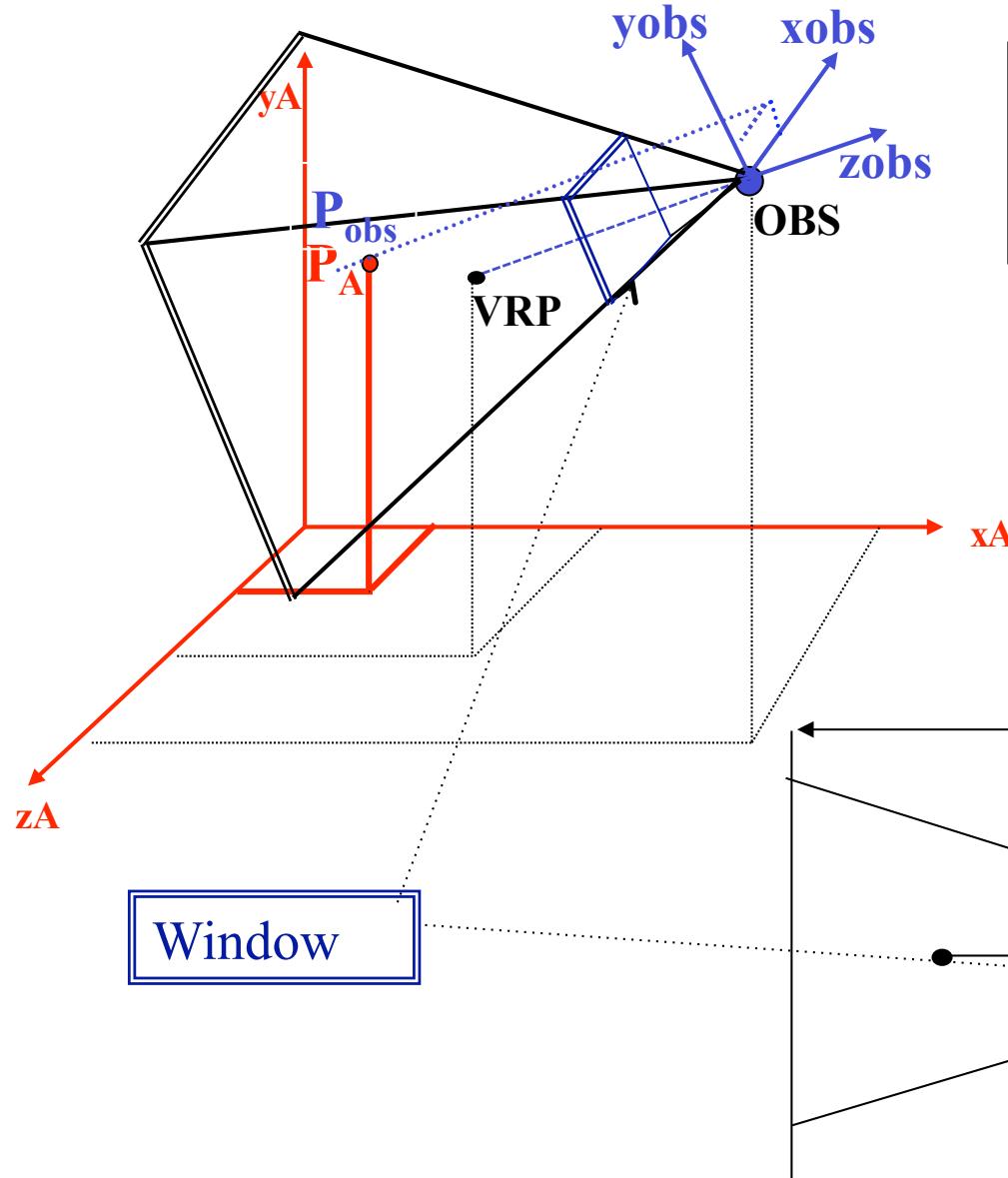




## Òptica de la càmera (1): perspectiva



- Determina geometria potencialment visible
- **EL VOLUM de VISIÓ -ÒPTICA- SEMPRE es defineix RESPECTE L'OBSERVADOR**



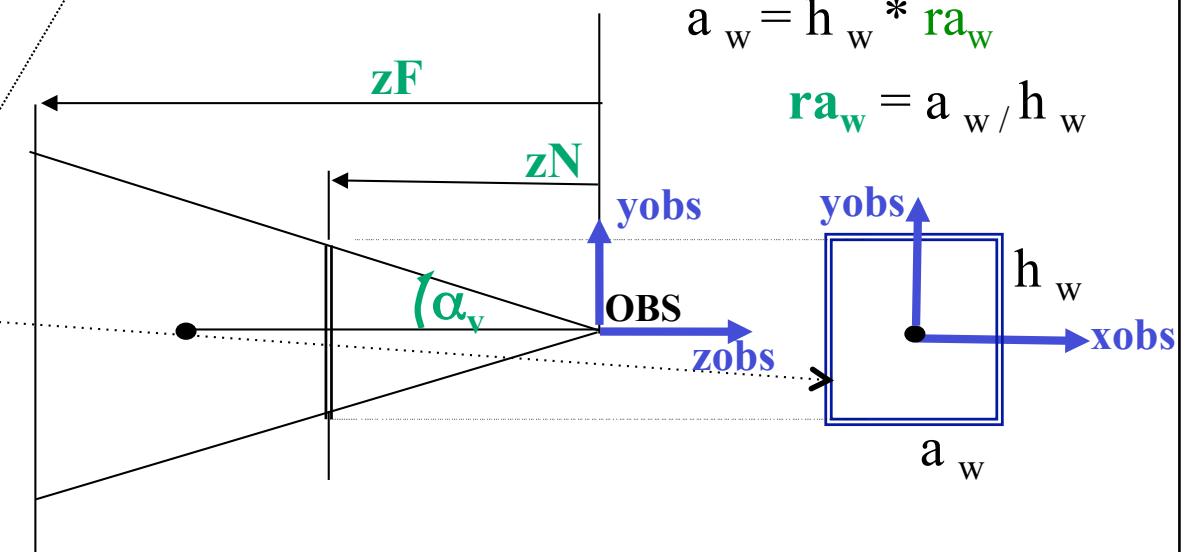
## Tipus d'òptica: Perspectiva

$\alpha_v$  ( $\text{FOV} = 2\alpha_v$ ),  $\text{zNear}$ ,  $\text{zFar}$ ,  $\text{ra}_w$

$$h_w = 2 \cdot zN \operatorname{tg}(\alpha_v)$$

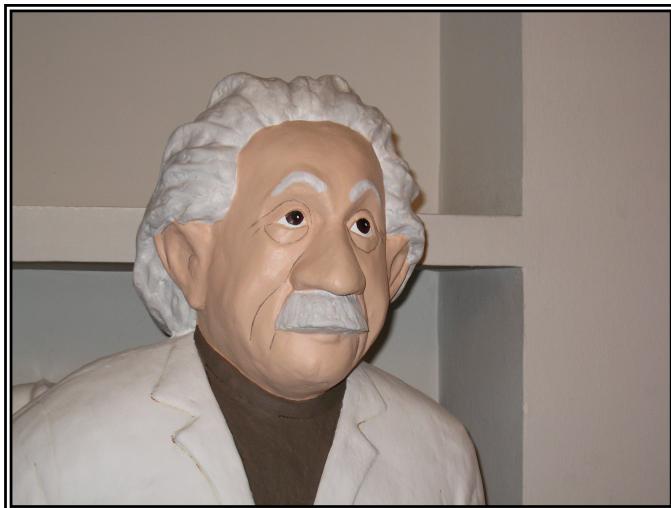
$$a_w = h_w * \text{ra}_w$$

$$\text{ra}_w = a_w / h_w$$



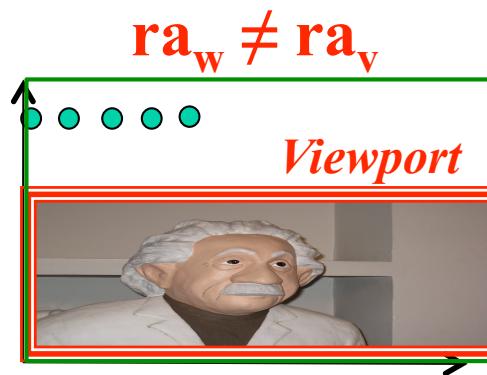
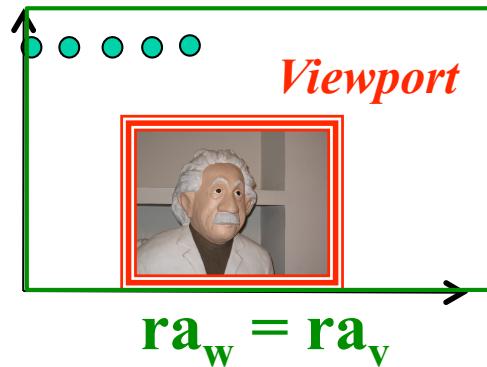
# Sobre la relació d'aspecte del *window* i del *viewport*

Window



Per a no tenir deformació en la imatge

$$ra_w = ra_v$$

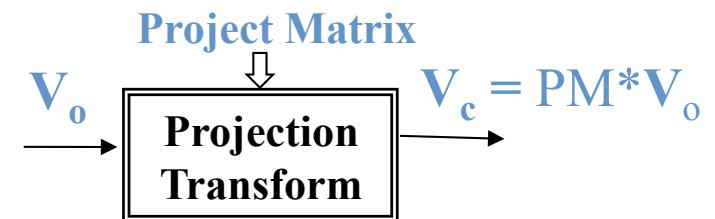


# FOV, zNear, zFar, ra → Càcul de matriu de projecció (PM)

FOV, zNear, zFar, ra<sub>w</sub>

$$PM = \begin{pmatrix} 1/ra*a & 0 & 0 & 0 \\ 0 & 1/a & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \begin{aligned} a &= \operatorname{tg}(FOV/2) \\ c &= (f+n)/(n-f) \\ d &= 2 n f / (n-f) \end{aligned}$$

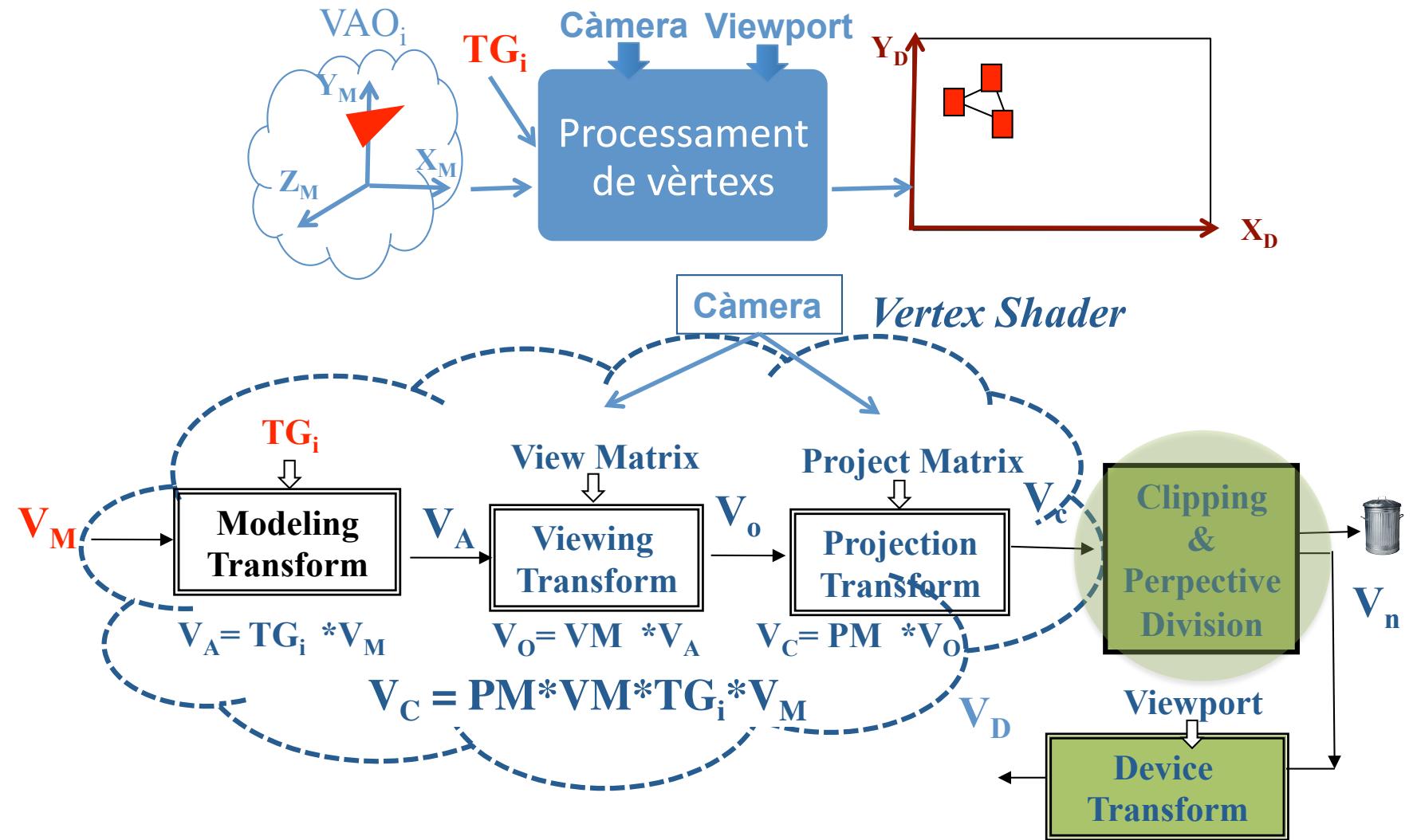
```
PM=perspective (FOV,ra,zN,ZF);
projectMatrix(PM);
```



$$V_c = (x_c, y_c, z_c, w_c)_i$$

$$w_c = -z_o$$

# Paradigma projectiu bàsic amb OpenGL 3.3

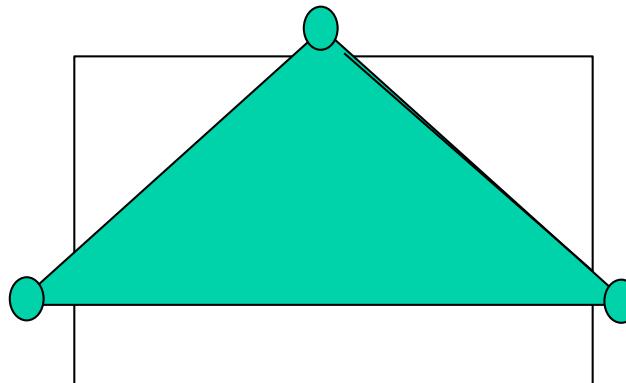


# Per què transformar el punt a coordenades de clipping simplifica el clipping?

$$V_c = (x_c, y_c, z_c, w_c)$$

Condició per a que un Vèrtex sigui interior al volum de visió

$$\begin{aligned} -w_c &\leq x_c \leq w_c \\ -w_c &\leq y_c \leq w_c \\ -w_c &\leq z_c \leq w_c \\ w_c &= -z_o \end{aligned}$$

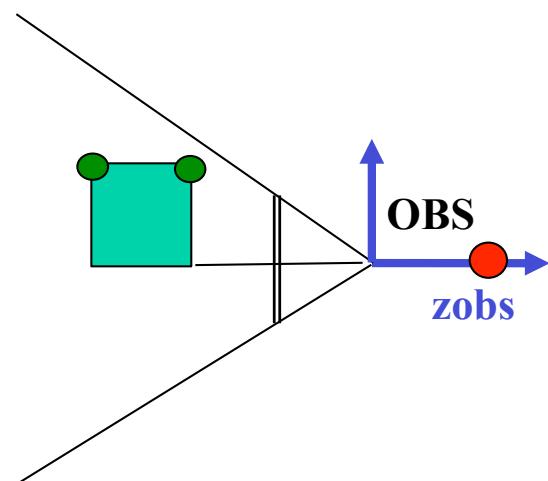


FOV = 90°  
 ra = 1  
 n = 1, f = 11

→

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

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



- Exemple:  $(0, 1, -2, 1)_o \Rightarrow (0, 1, 0.2, +2)_c$   
 Exemple:  $(0, 1, -3, 1)_o \Rightarrow (0, 1, 1.4, +3)_c$   
 Exemple:  $(0, 0, 1, 1)_o \Rightarrow (0, 0, -3.4, -1)_c$

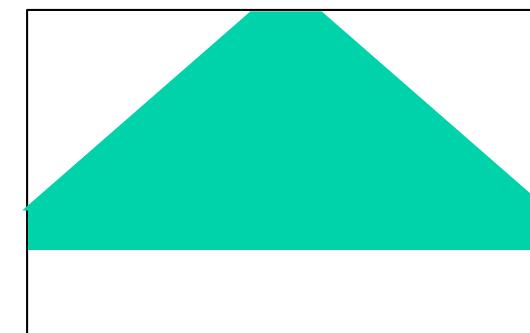
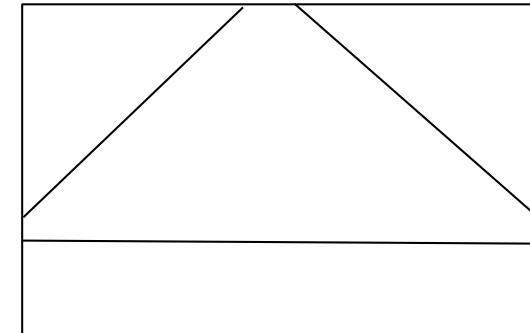
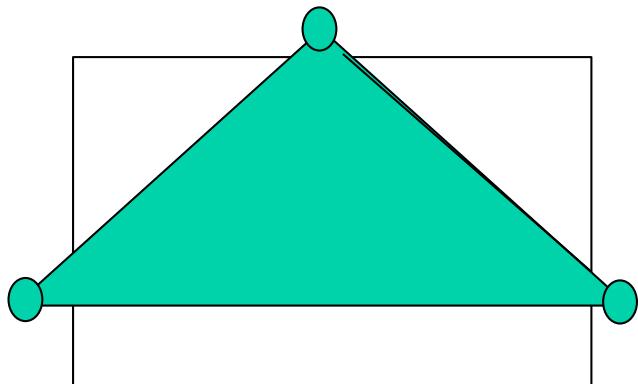
*Condició per a que un Vèrtex  
sigui interior al volum de visió:*

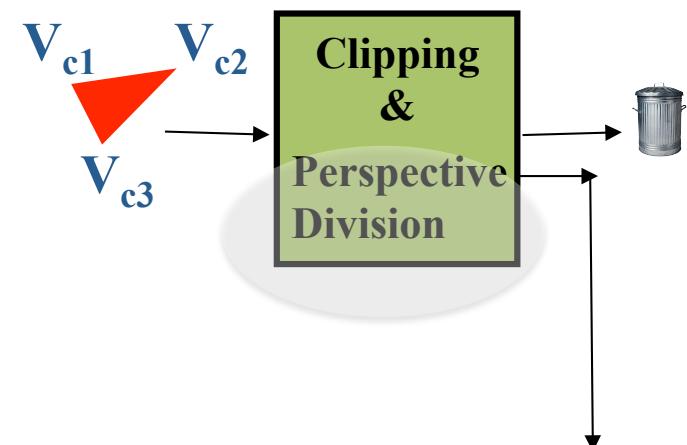
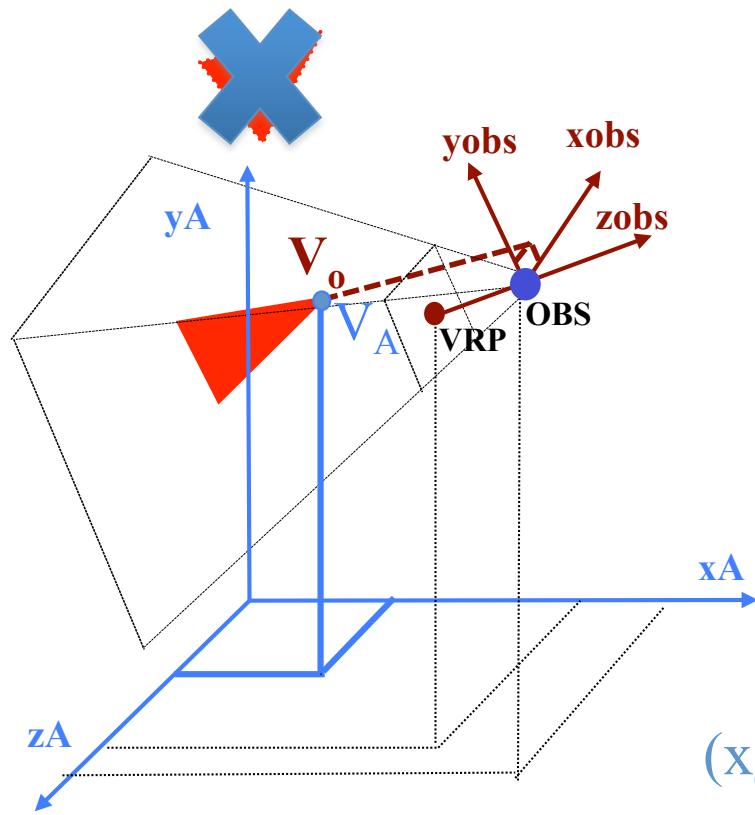
$$-W_c \leq x_c \leq W_c$$

$$-W_c \leq y_c \leq W_c$$

$$-W_c \leq z_c \leq W_c$$

$$W_c = -Z_o$$

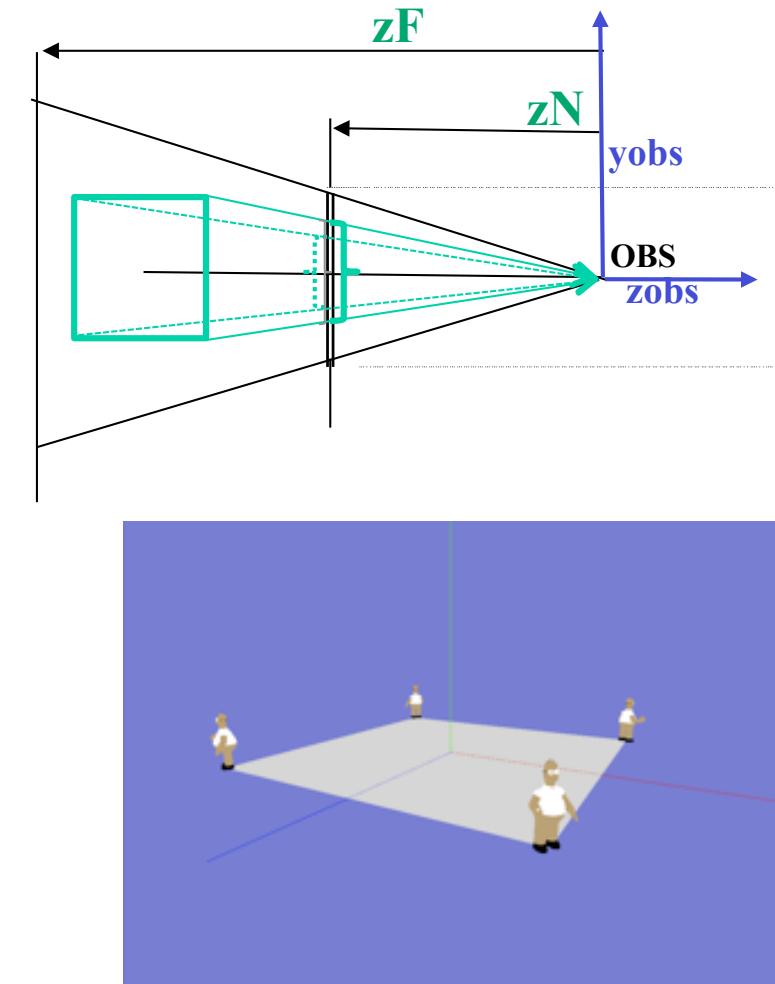
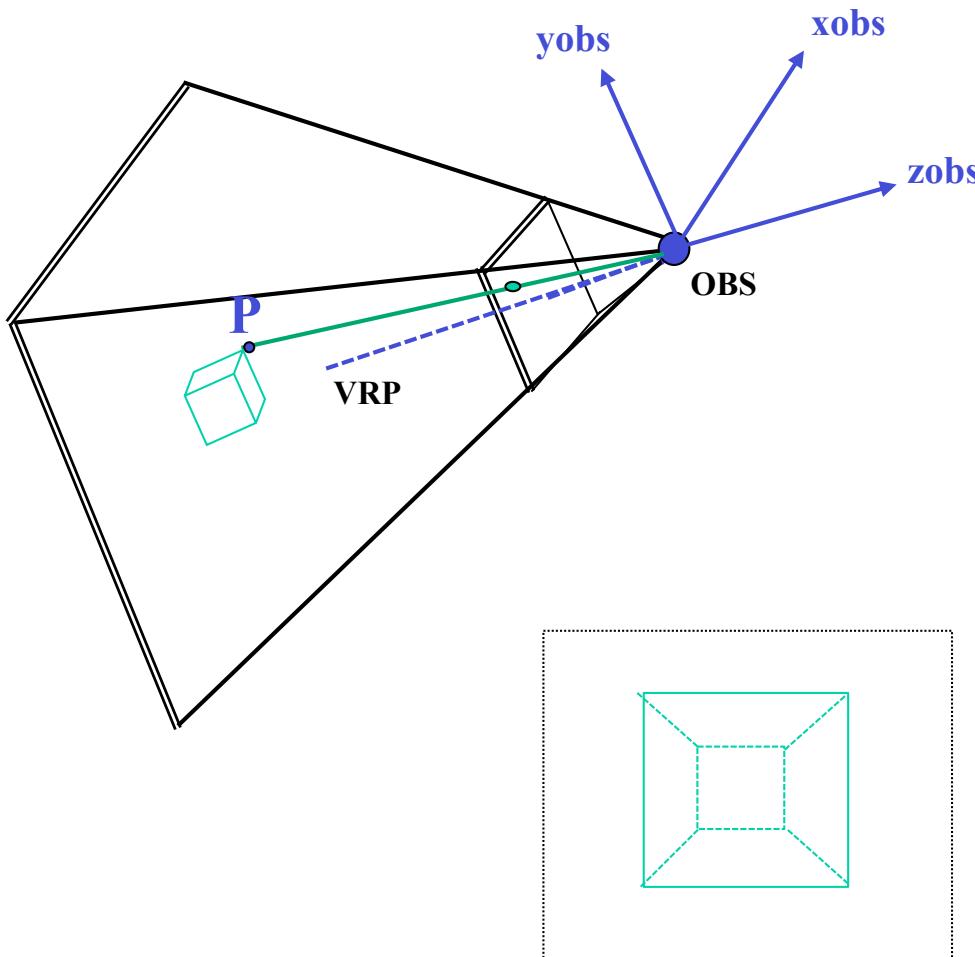


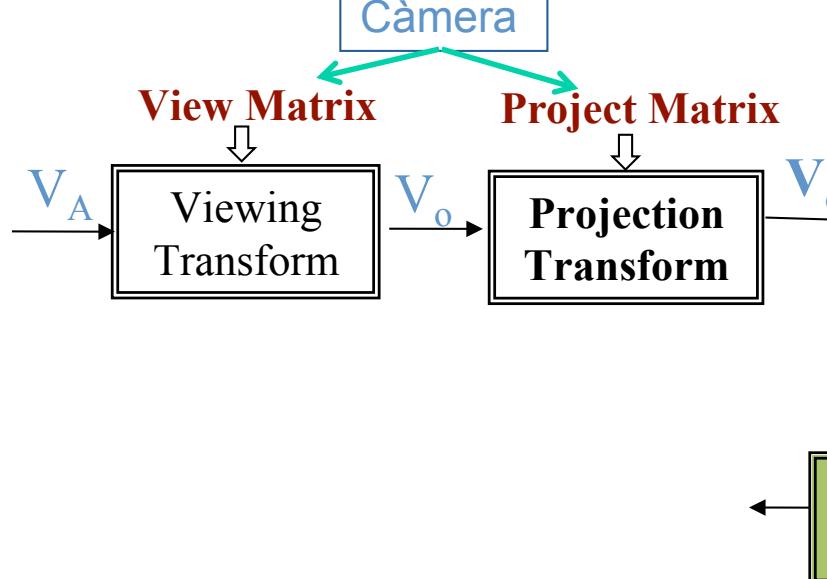
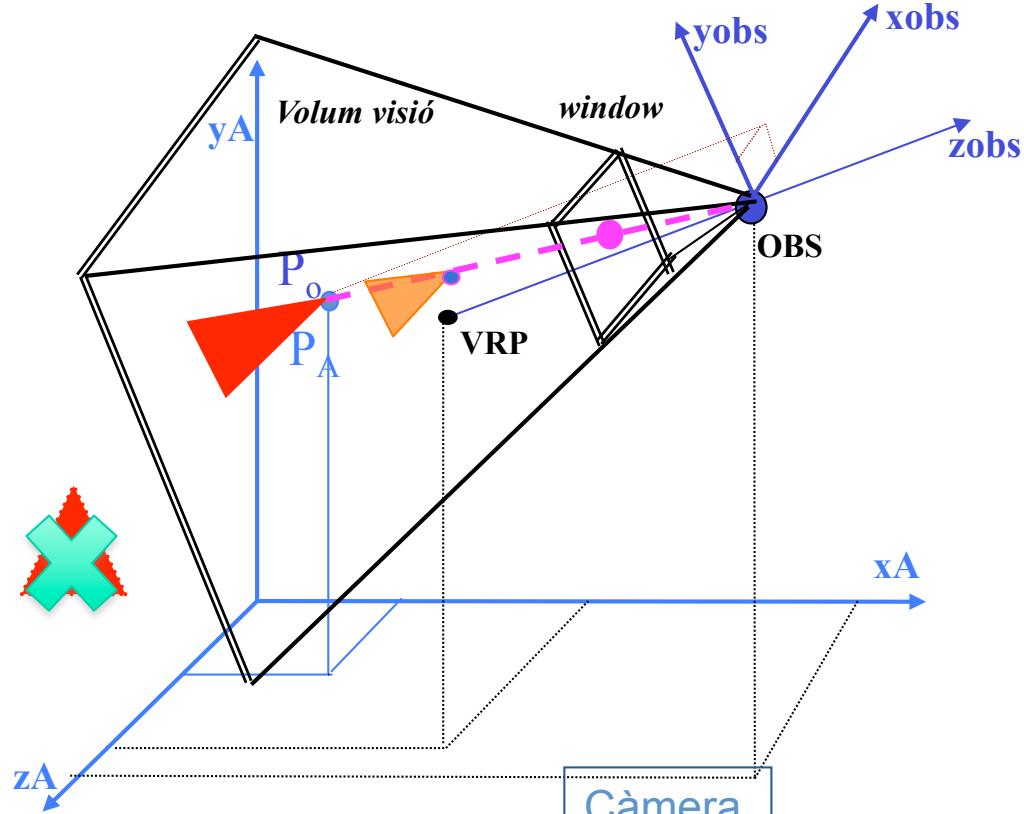


$$\left. \begin{aligned} & (x_c, y_c, z_c, w_c) / w_c \\ & w_c = -z_o \end{aligned} \right\} = (-x_c/z_o, -y_c/z_o, -z_c/z_o, 1)$$



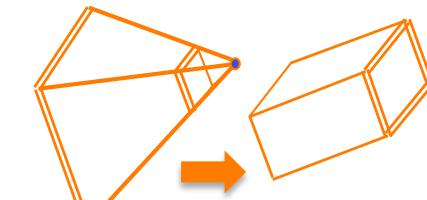
# Perspective division i Projecció





$$\begin{aligned} -w_c \leq x_c &\leq w_c \\ -w_c \leq y_c &\leq w_c \\ -w_c \leq z_c &\leq w_c \end{aligned}$$

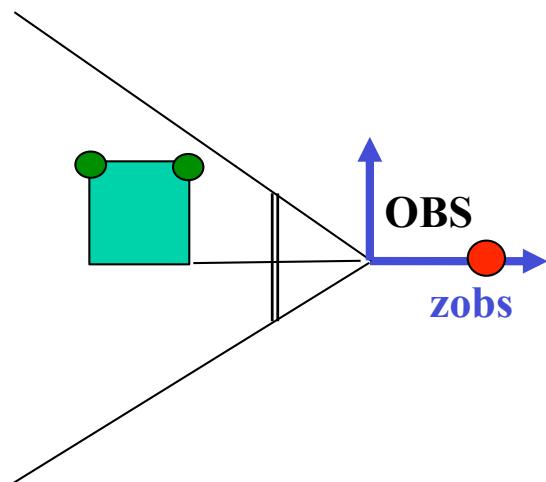
$$\begin{aligned} -1 \leq x_n &\leq 1 \\ -1 \leq y_n &\leq 1 \\ -1 \leq z_n &\leq 1 \end{aligned}$$



FOV = 90°  
 ra = 1  
 n = 1, f = 11

$$\xrightarrow{\text{ }} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1.2 & -2.2 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

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



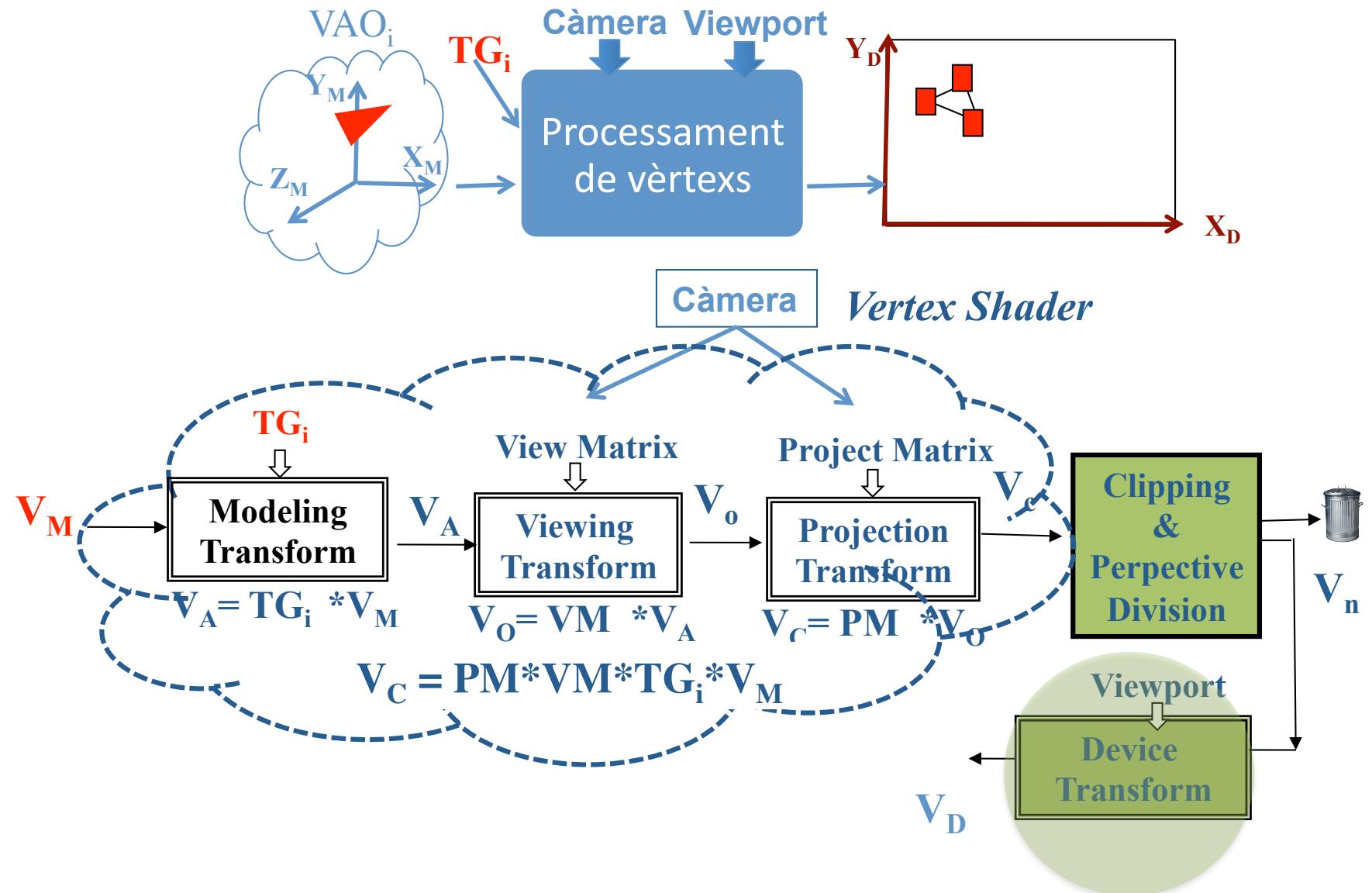
Exemple:  $(0, 1, -2, 1)_o \Rightarrow (0, 1, 0.2, +2)_c$

$$\Rightarrow (0, 0.5, 0.1)$$

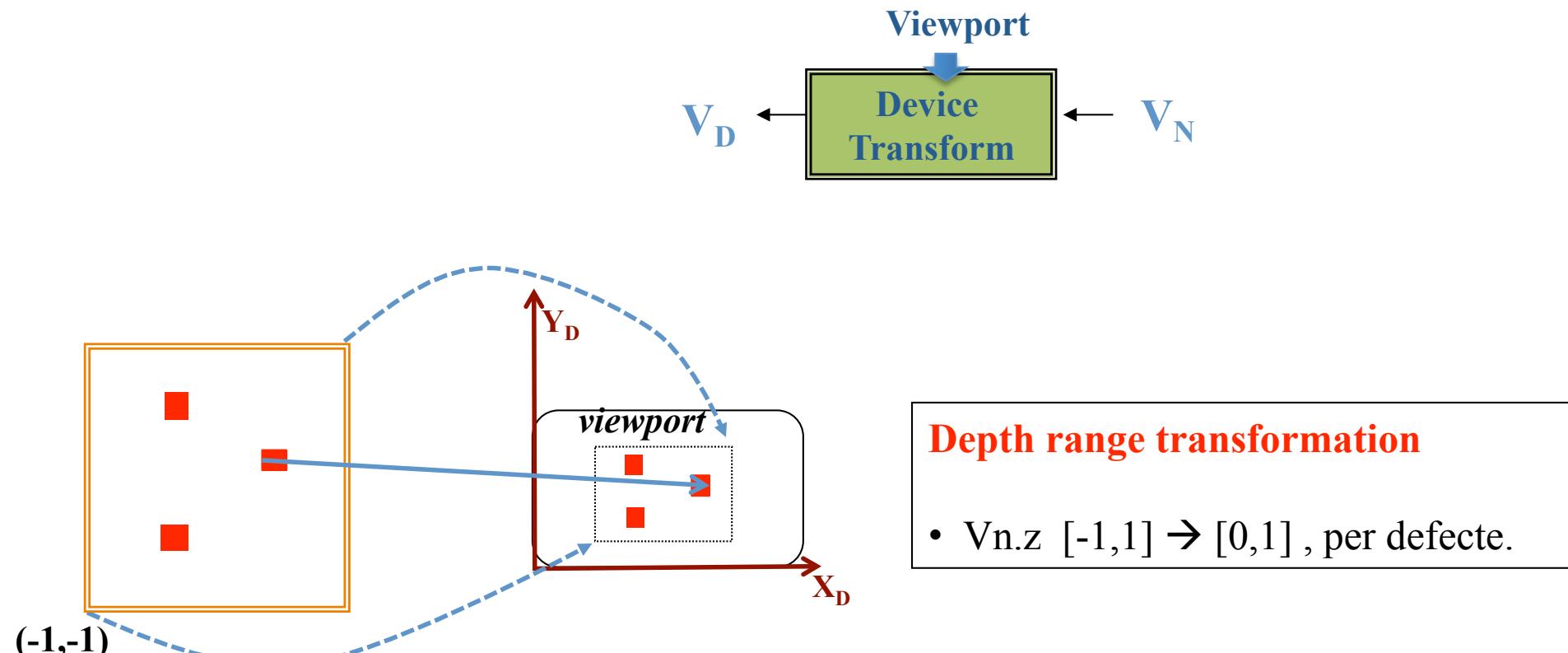
Exemple:  $(0, 1, -3)_o \Rightarrow (0, 1, 1.4, +3)_c$

$$\Rightarrow (0, 0.33, 0.4)$$

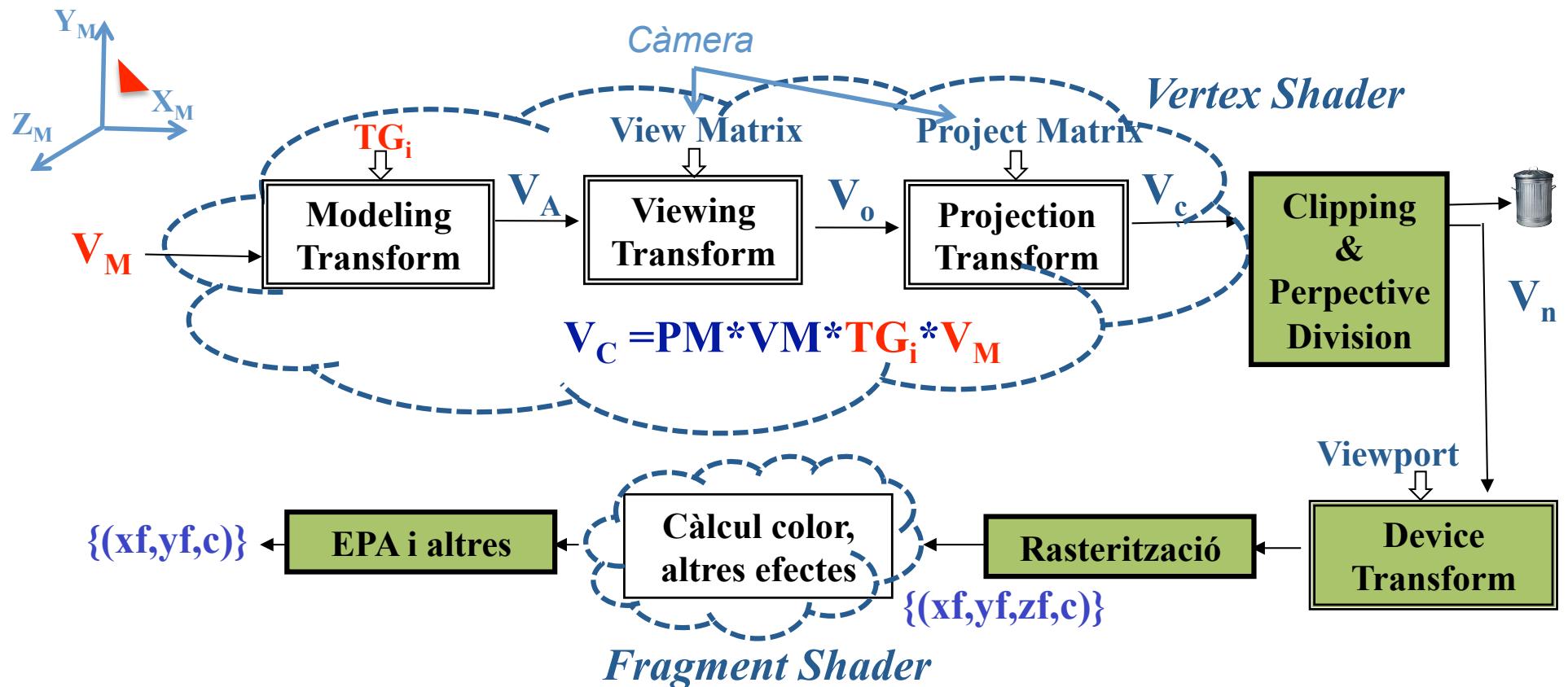
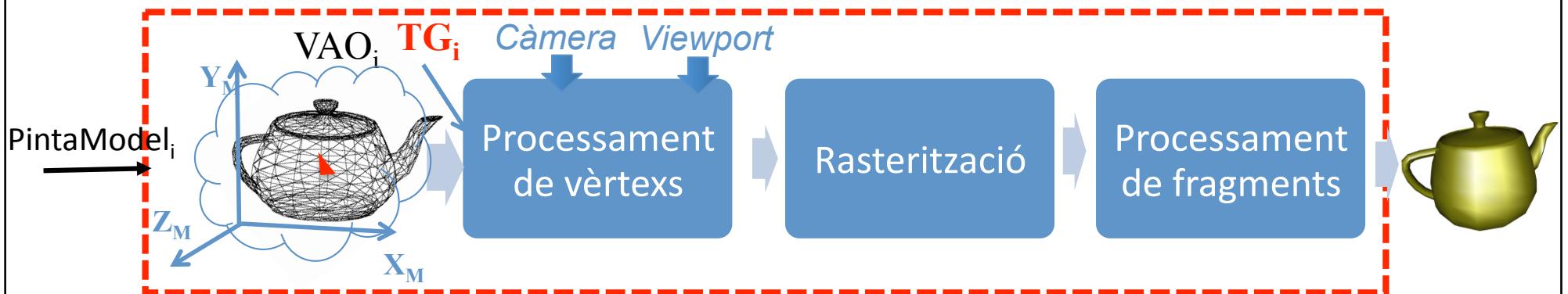
# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3

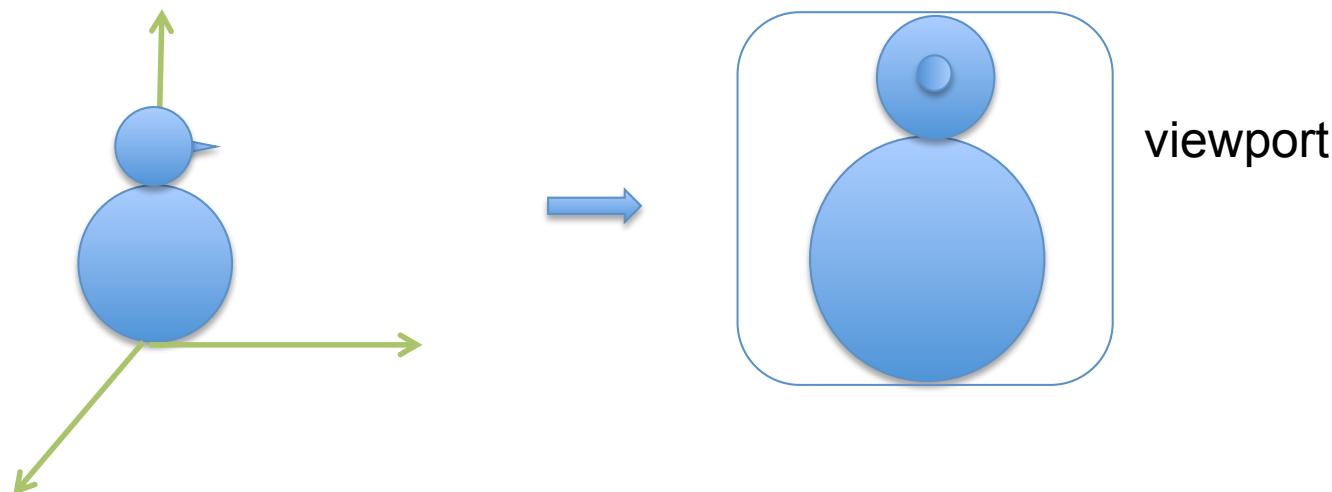


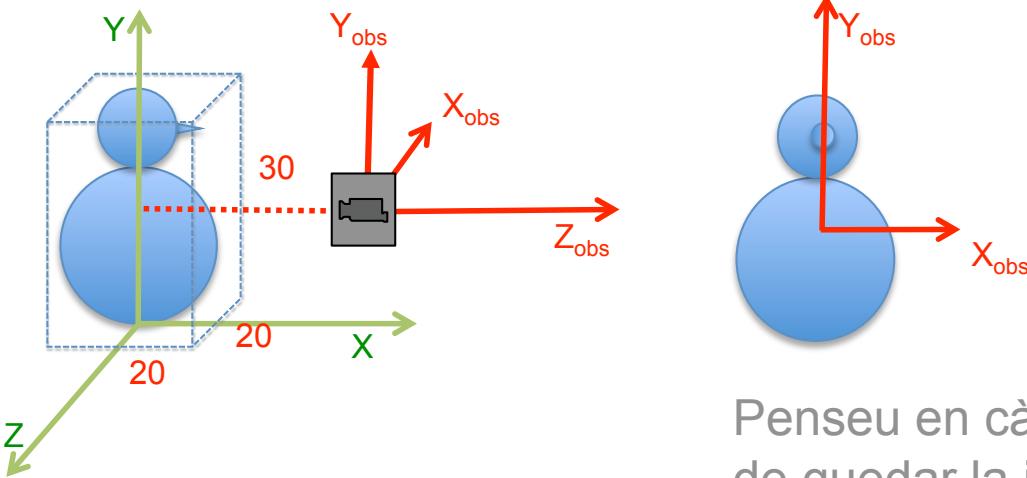
# Classe 3: contingut

- Visualització: breu repàs.
- Càmera (1)
  - Posicionament: OBS, VRP, up
  - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs (2 i fi)
- Exemple definició de càmera
- Exercici de TG pendent

**Exemple 1:** Indica:

- Donada una funció `pinta_ninot()` que pintaria un objecte com el de la figura, format per: una esfera de radi 10 i centre  $(0,10,0)$ , una altra esfera de radi 5 i centre  $(0,25,0)$ , i un con de base centrada en  $(2.5, 25, 0)$ ,  $r=2$  i llargada 5 orientat segons l'eix  $X^+$
- tots els paràmetres d'una càmera que permeti obtenir la imatge similar a la que s'indica, en un viewport de  $600 \times 600$  que ocupa tota la finestra gràfica.

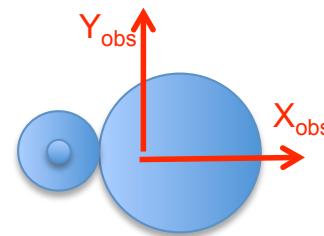




Penseu en càmera i com ha de quedar la imatge

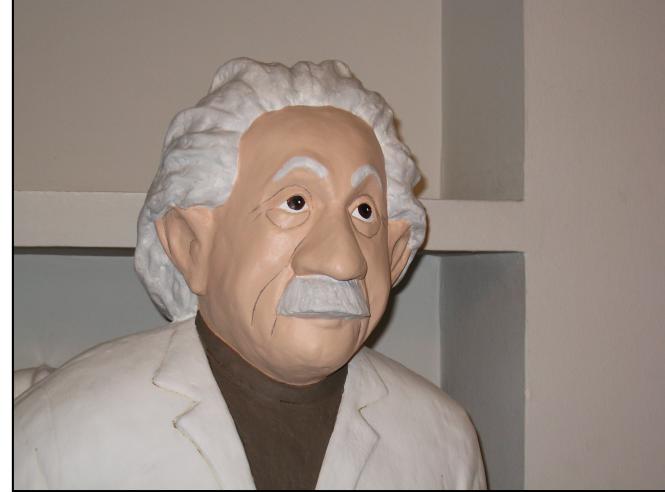
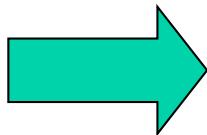
```
VM = lookAt(OBS, VRP, up);
viewMatrix(VM);
pinta_ninot();
```

$VRP=(0,15,0)$   
 $OBS=(30,15,0)$   
 $Up=(0,1,0)$

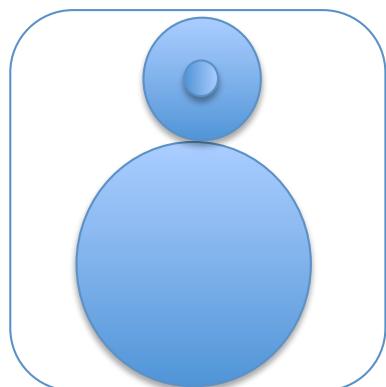


*Quins paràmetres si volem que quedí així?*

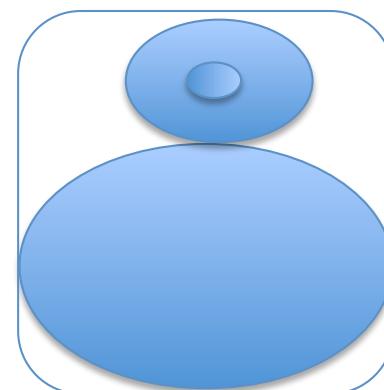
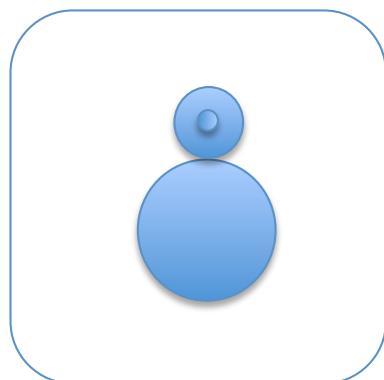
# Òptica de la càmera: Determina el Volum de Visió



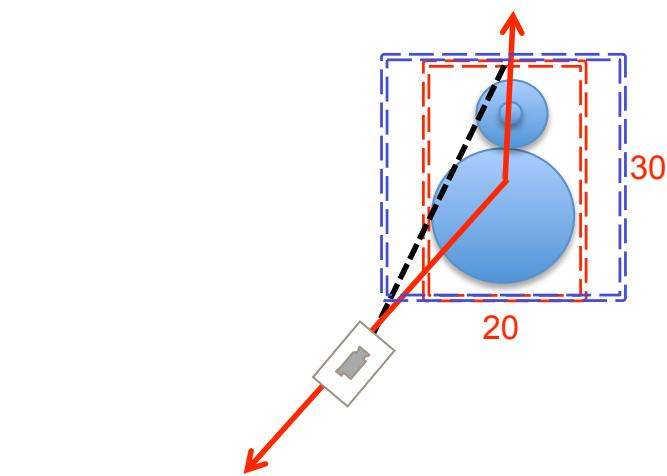
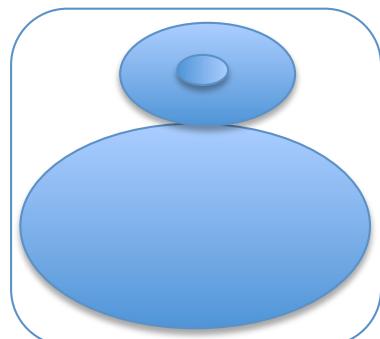
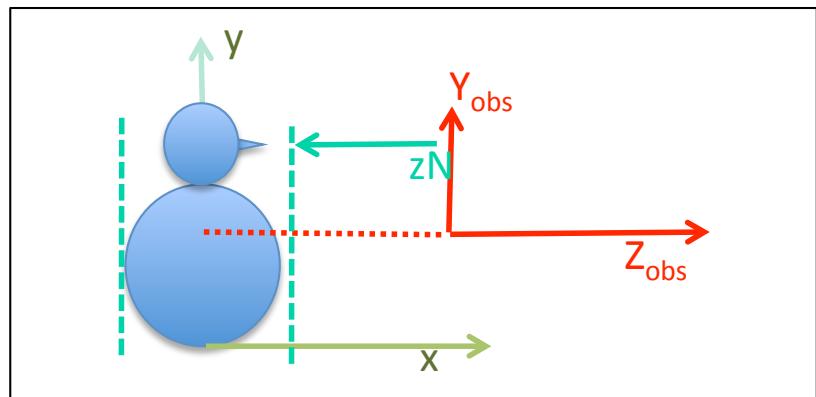
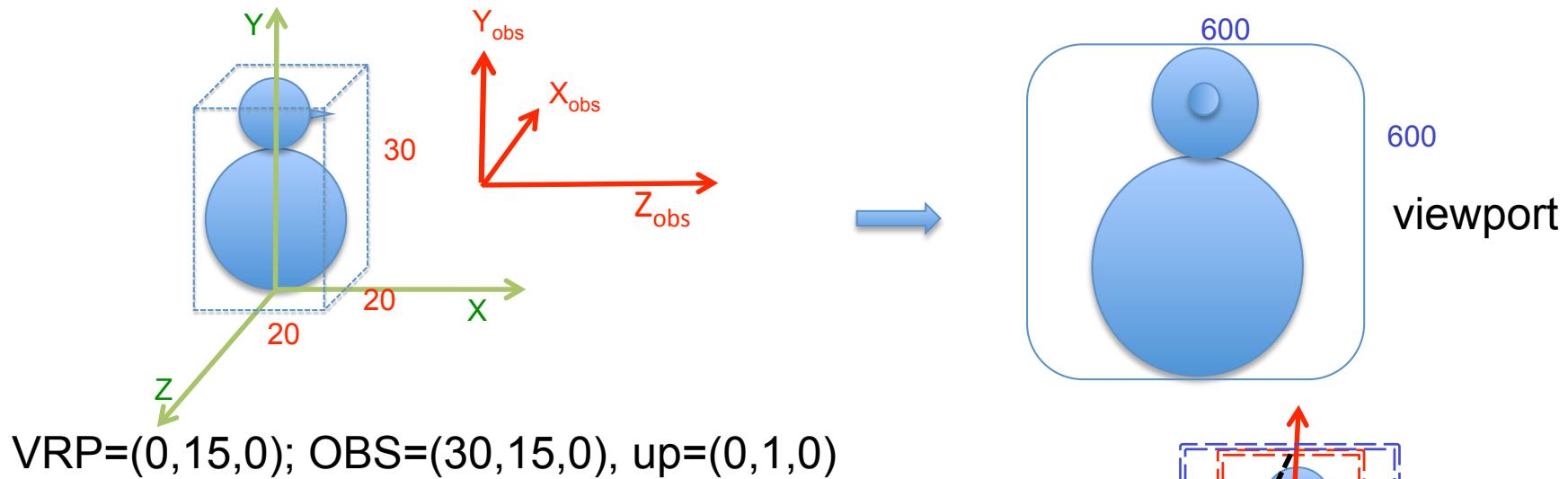
- 1. Posició, orientació**
- 2. Òptica**



viewport



# Exemple 1: Òptica perspectiva



$$zN = 20; zF = 40$$

$$\alpha = \arctg (15/20) \rightarrow \alpha = 36,8^\circ$$

$$ra_w = 20/30 = 0,66$$

Com  $ra_v = 1 \rightarrow$  deformació  
Solució  $ra_w = 1$

```

/* CreateBuffers(); Crear VAO del model (un
   cop)*/

...
/* calcular paràmetres càmera i
   matrius cada cop que es
   modifiquin */
VM = lookAt(OBS,VRP,UP);
viewMatrix(VM);
PM=perspective (FOV,ra,zN,ZF);
projectMatrix(PM);
glViewport (0,0,w.h);
....
/*PaintGL(); cada cop que es requerix
   refresc*/
//Calcular TG, i passar a OpenGL
modelTransform_i(TG);
modelMatrix(TG);
pinta_ninot(VAO);

```

### *Vertex Shader*

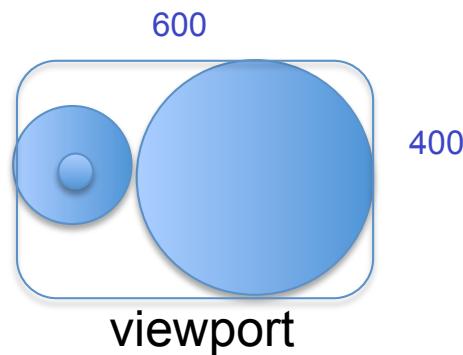
```

in vec3 vertex;
uniform mat4 TG, VM, PM;
void main ()
{
    gl_Position =
        PM*VM*TG*vec4(vertex,1.0);
}

```

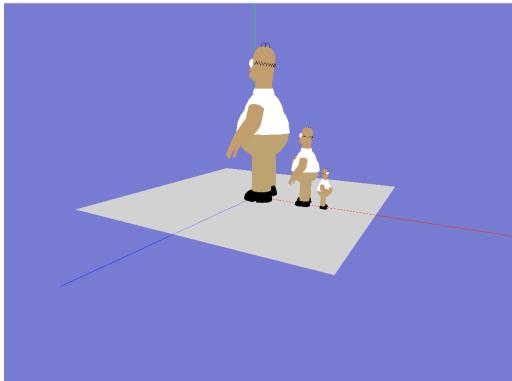
# Per pensar...

- Com fer un zoom? Quins paràmetres de càmera modificaries?
- Quins paràmetres càmera per a obtenir:



# Classe 3: contingut

- Visualització: breu repàs.
- **Càmera (1)**
  - Posicionament: OBS, VRP, up
  - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs (2 i fi)
- Exemple definició de càmera
- Exercici de TG pendent



## Per pensar...

Quins podrien ser uns **paràmetres de posició, orientació i òptica** per a una càmera que, donada una escena i coneguda la seva capsa mínima contenidora

$(x_{\min}, y_{\min}, z_{\min}) - (x_{\max}, y_{\max}, z_{\max})$ , visualitzi una imatge que inclogui totalment l'escena, ocupant el màxim de la vista (viewport) i sense deformació?

**Exemple 1:** Donades les funcions:

- pinta\_esfera( ) que envia a pintar una esfera de radi 1 i centre en (0,0,0)
- pinta\_con( ) que envia a pintar un con amb base centrada en origen i radi 1, i alçada 1 amb direcció del seu eix segons Z<sup>+</sup>

Indica:

- El pseudo-codi d'una funció pinta\_ninot( ) que pintaria un objecte com el de la figura, format per: una esfera de radi 10 i centre (0,10,0), una altre esfera de radi 5 i centre (0,25,0), i un con de base centrada en (2.5, 25,0), r=2 i llargada 5 orientat segons l'eix X<sup>+</sup>

