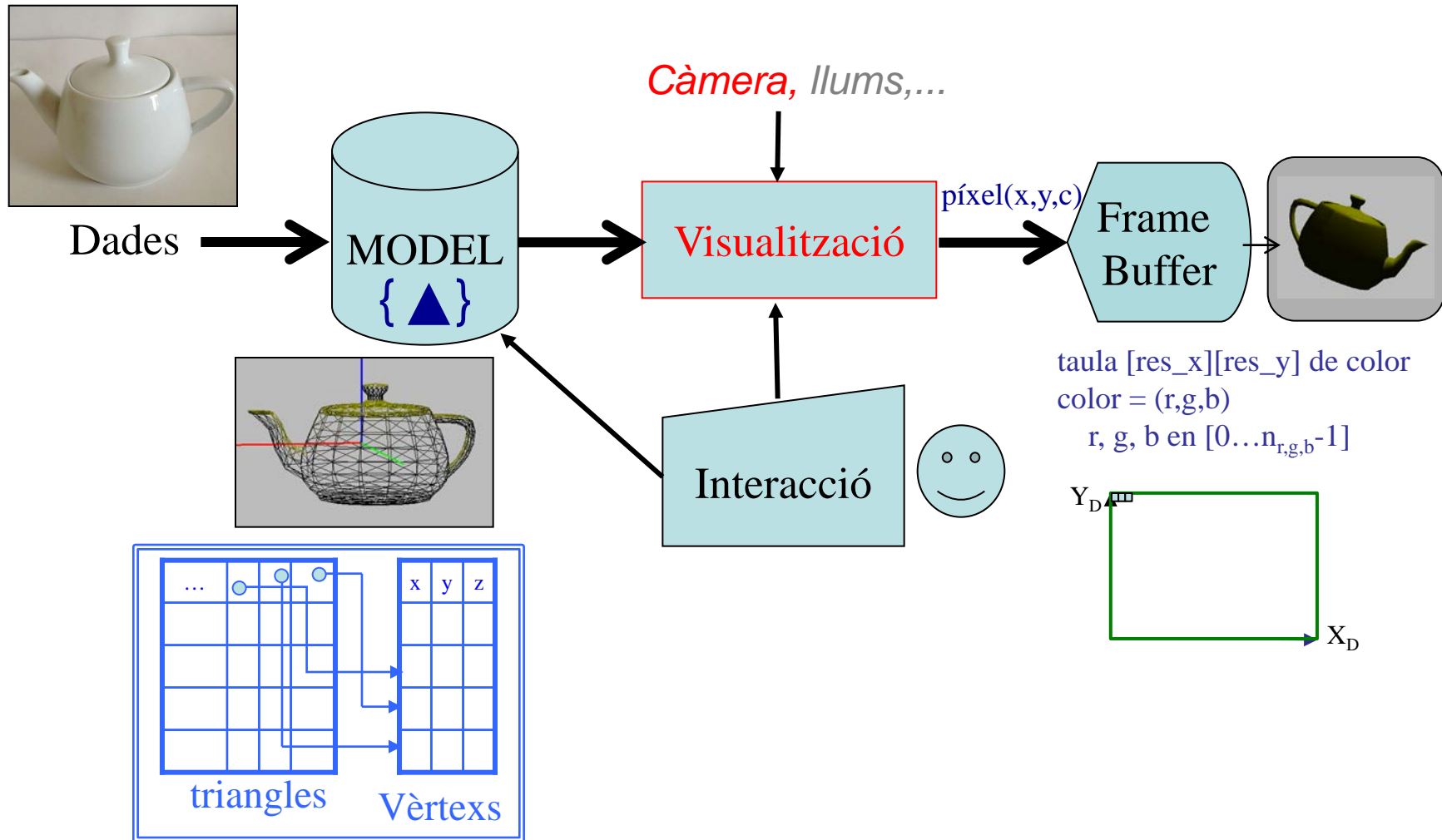
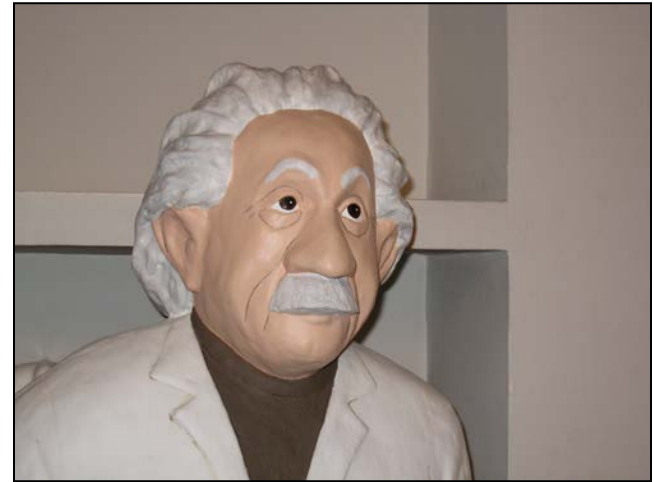
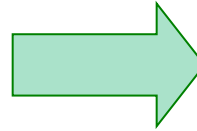


Classe 3: Especificació càmera

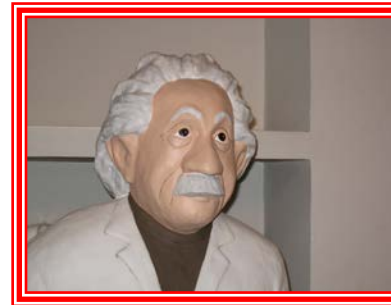
- **La càmera en el Procés de Visualització**
- Posició i orientació
- Òptica
 - Perspectiva
 - Ortogonal
- La càmera en el Vertex Shader
- Sistemes de coordenades – afegim TG
- Zoom
- Exemples

Aplicació gràfica





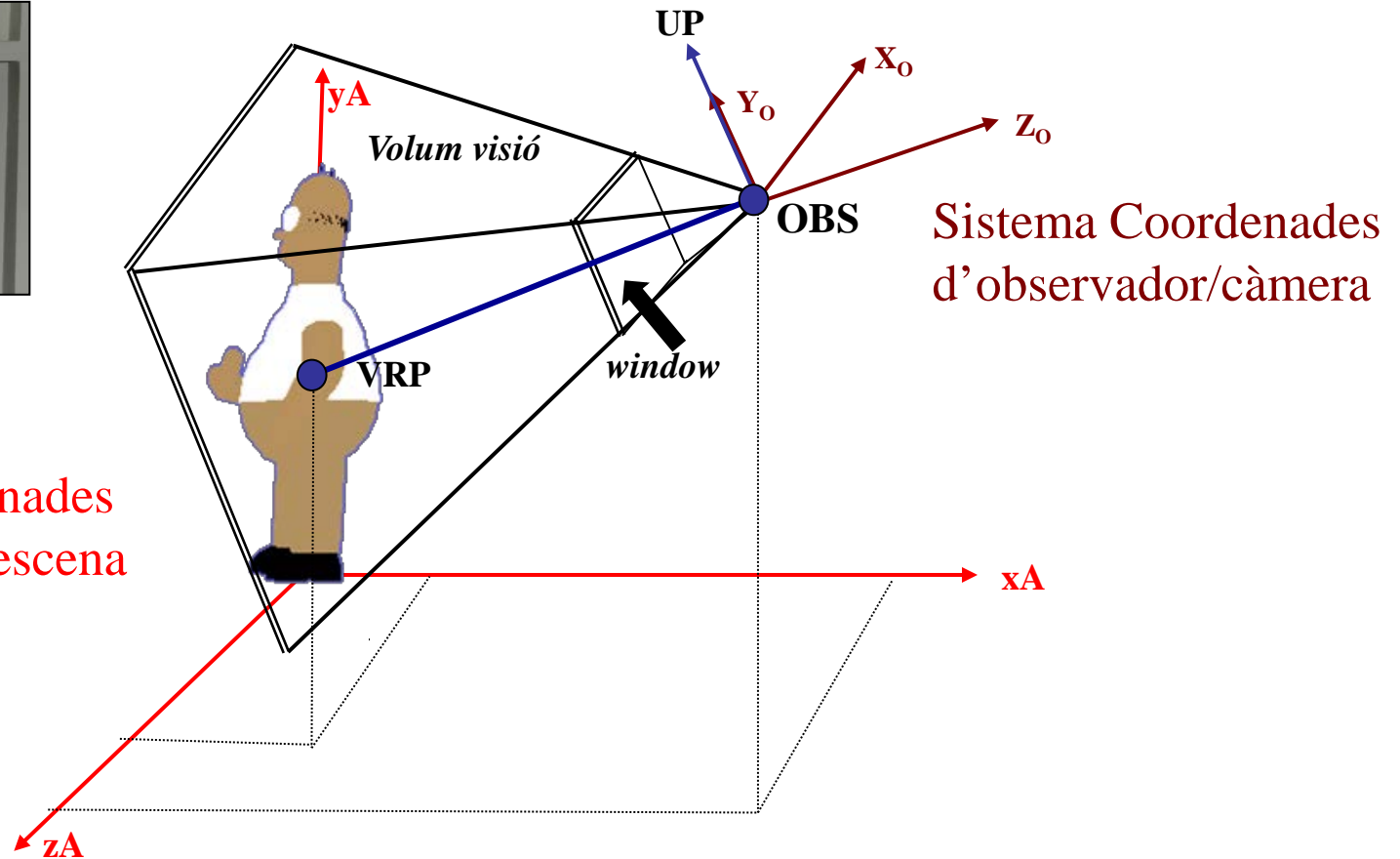
4. Emmarcar



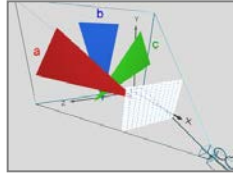
1. Posició, orientació
2. Òptica

3. Fer la Foto

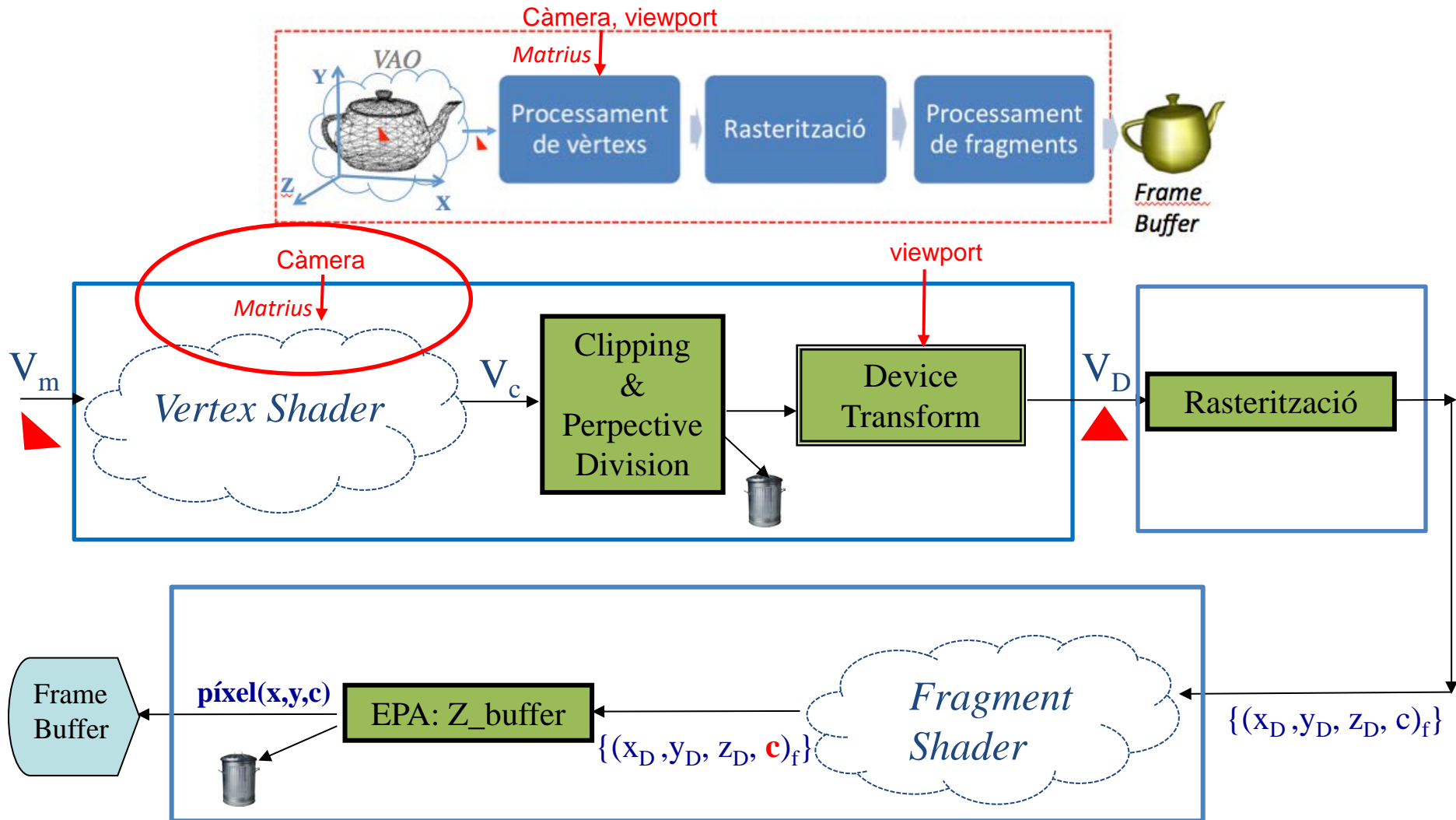
Com indicar la càmera?



1. Ubicació càmera respecte SCA: obs, vrp, up
2. Definir òptica: Volum de Visió → window, zNear, zFar

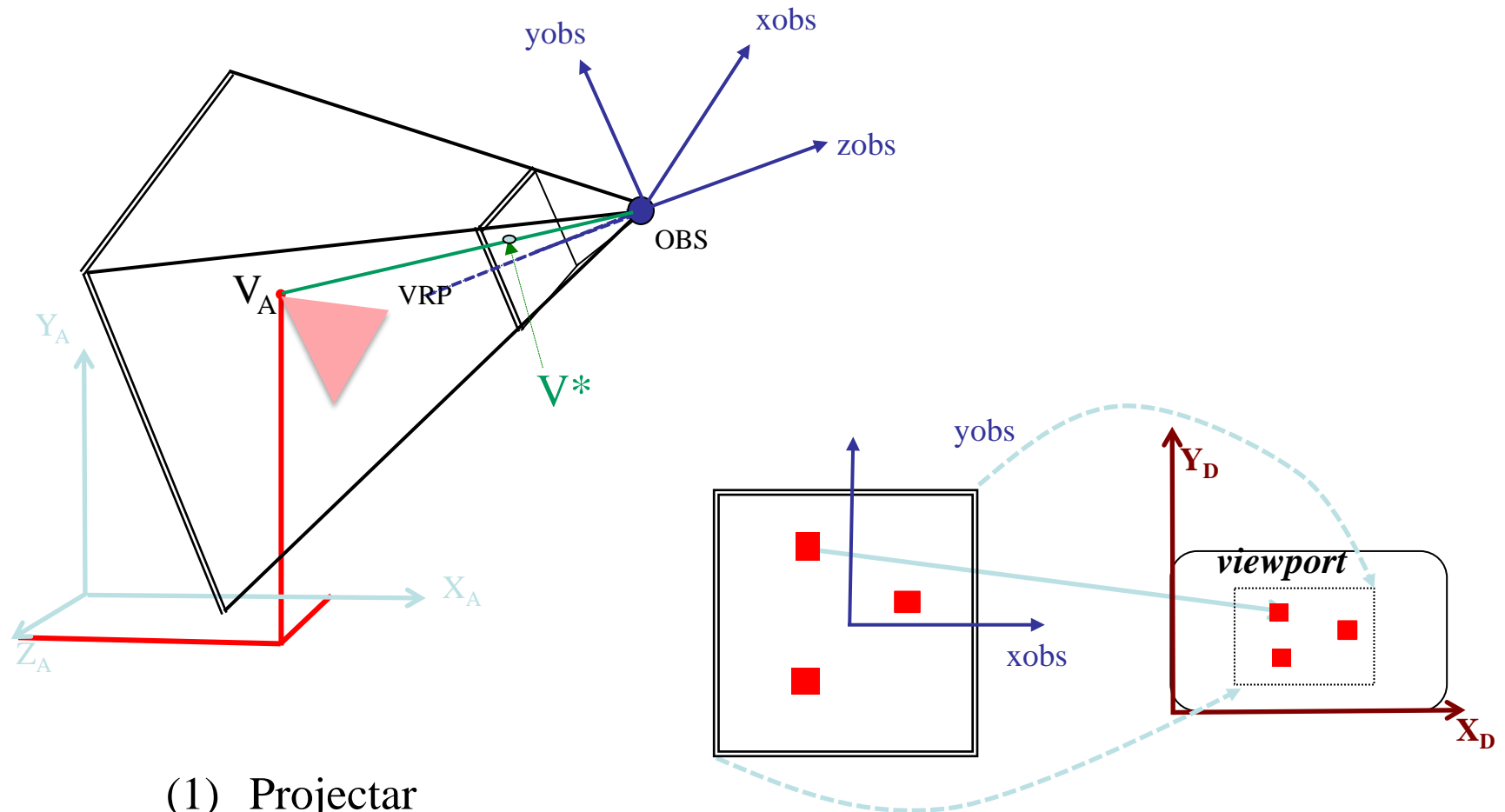


Procés de visualització d'OpenGL 3.3



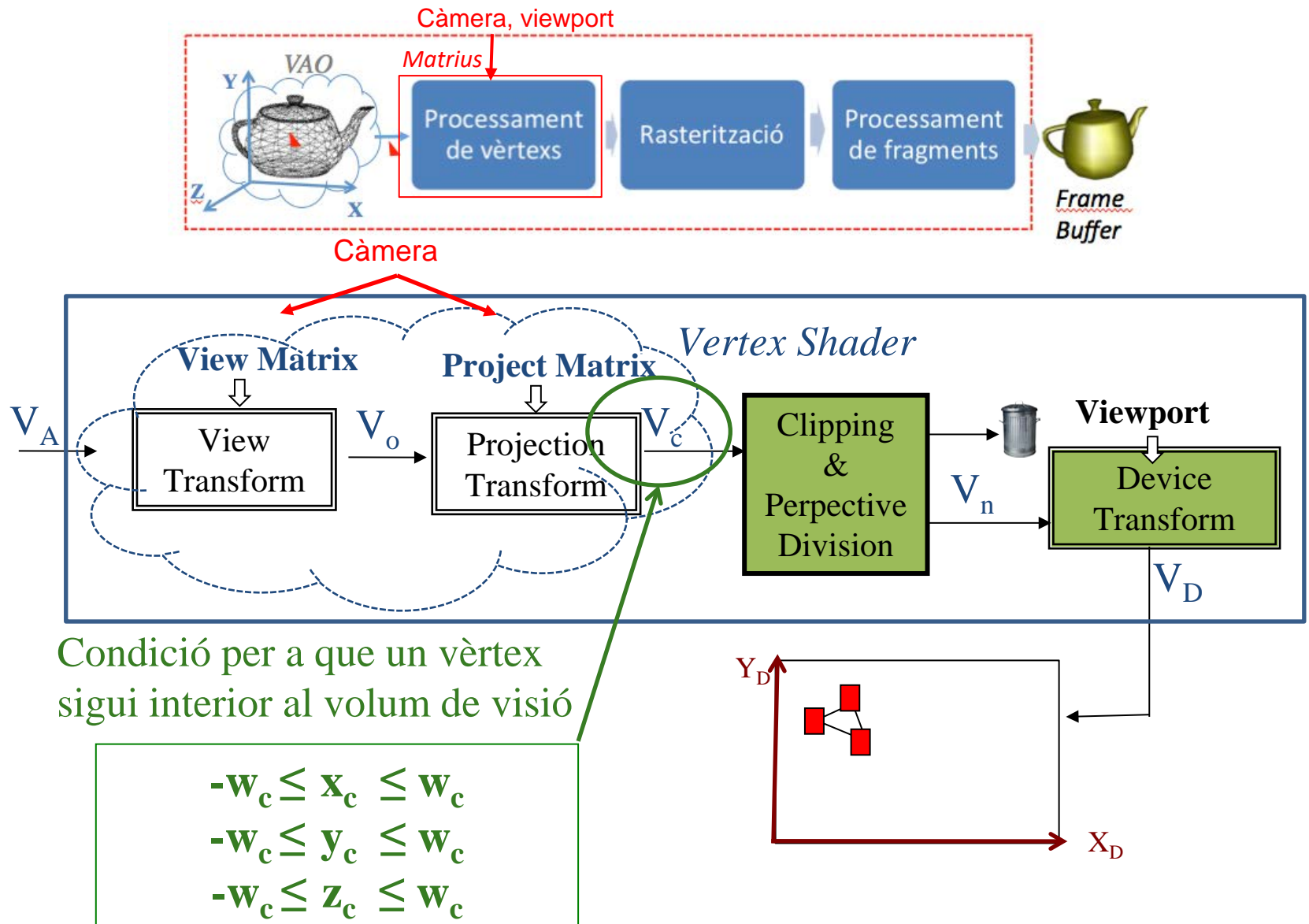


Processament “manual” de vèrtexs



- (1) Projectar
- (2) Transformació window-viewport

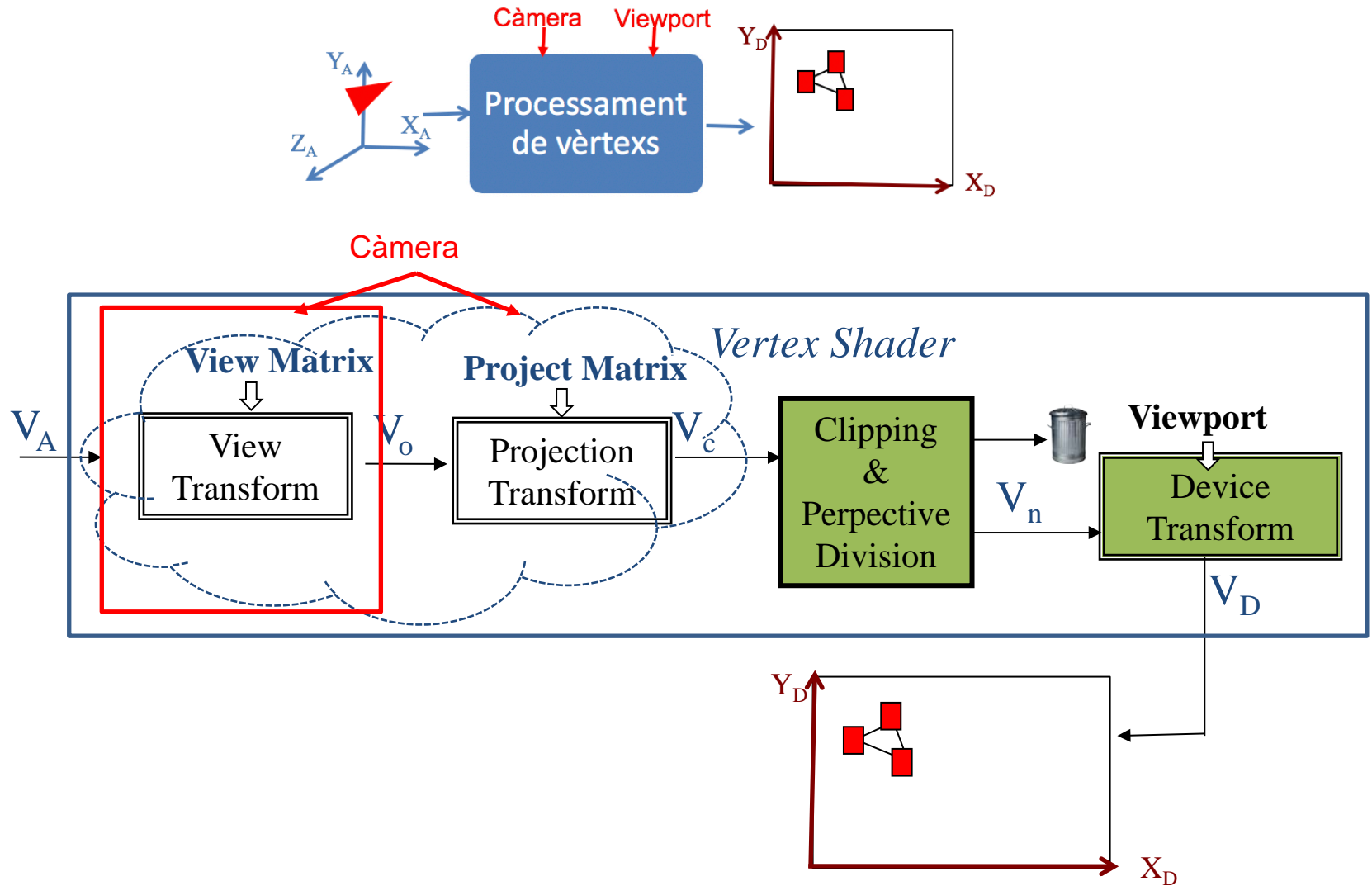
Processament de vèrtexs en OpenGL 3.3



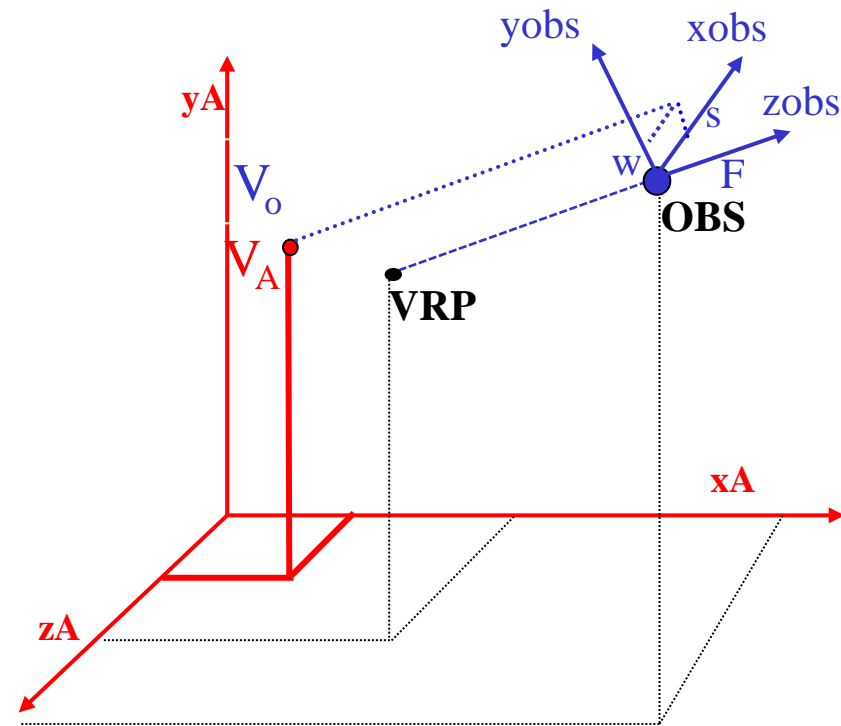
Classe 3: Especificació càmera

- La càmera en el Procés de Visualització
- **Posició i orientació**
- Òptica
 - Perspectiva
 - Ortogonal
- La càmera en el Vertex Shader
- Sistemes de coordenades – afegim TG
- Zoom
- Exemples

La càmera en el VS (1): View Matrix



Pas a SCO amb VRP, OBS i Up: Càlcul View Matrix



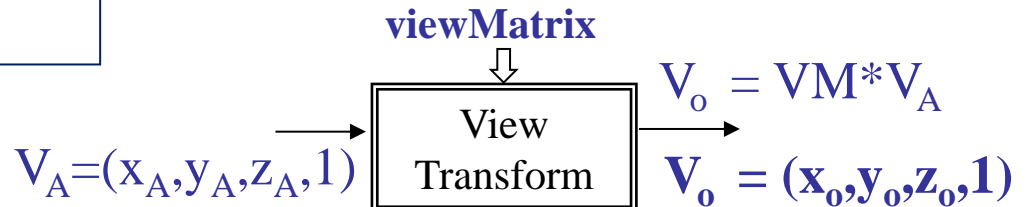
$$VM = \begin{bmatrix} s.x & s.y & s.z & 0 \\ w.x & w.y & w.z & 0 \\ F.x & F.y & F.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & -Obs.x \\ 0 & 1 & 0 & -Obs.y \\ 0 & 0 & 1 & -Obs.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$F = OBS - VRP = (F.x, F.y, F.z) \quad F = F / \|F\|$$

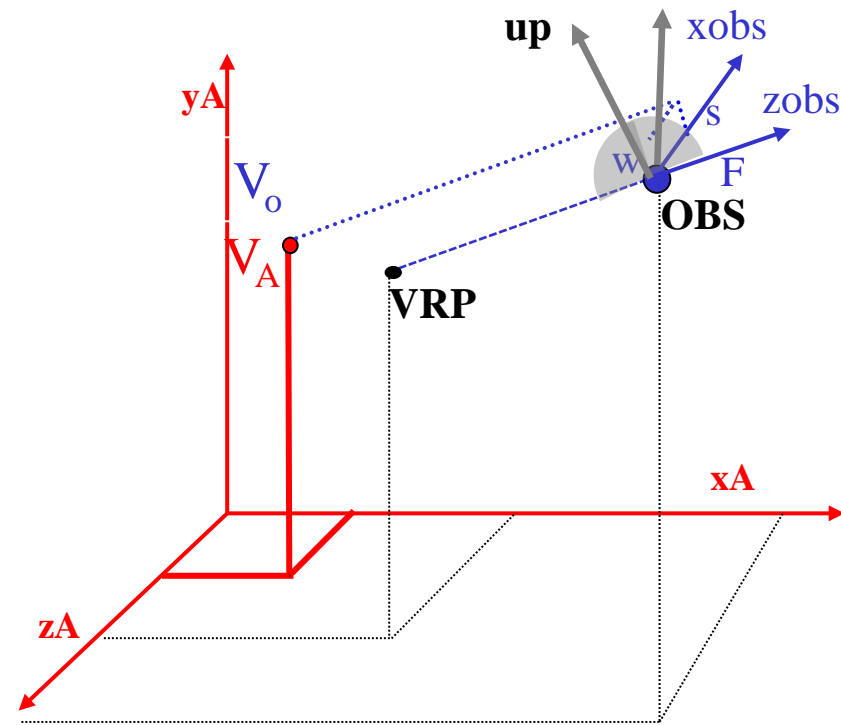
$$s = Up \times F \quad s = s / \|s\|$$

$$w = F \times s$$

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



Pas a SCO amb VRP, OBS i Up: Càlcul View Matrix



OBS = Observador

VRP = View Reference Point

up = View Up Vector

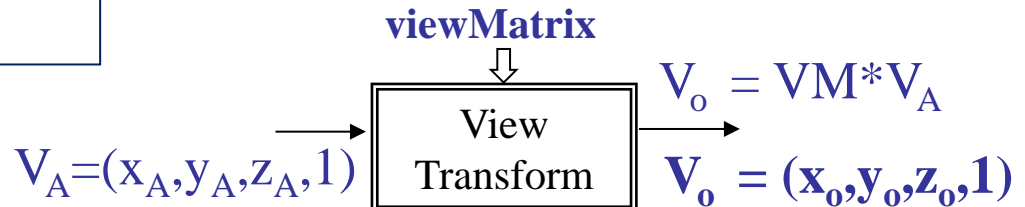
up “indica” la direcció de l’eix vertical de la Càmera

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

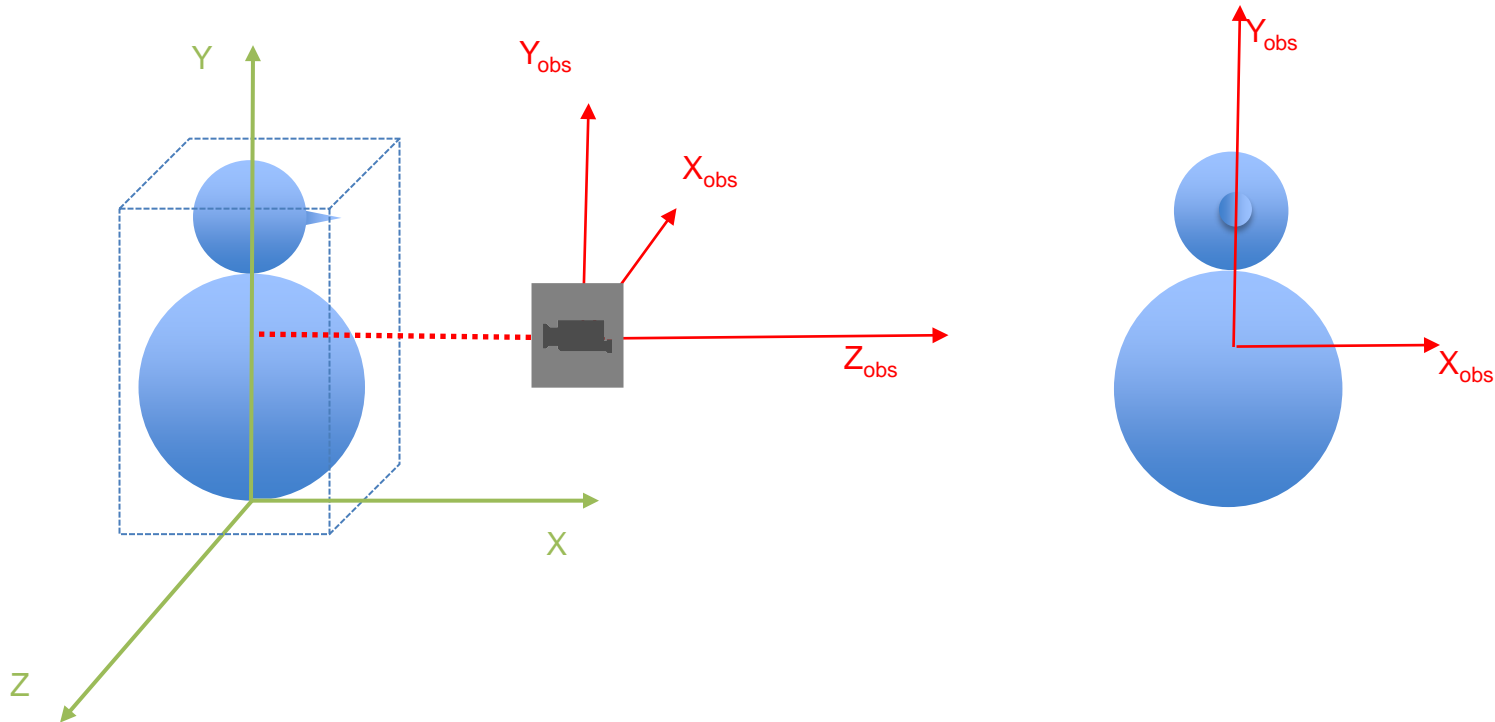
$$F = \text{OBS} - \text{VRP} = (F.x, F.y, F.z) \quad F = F / \|F\|$$

$$s = \text{Up} \times F \quad s = s / \|s\|$$

$$w = F \times s$$



Exemple (conceptual)



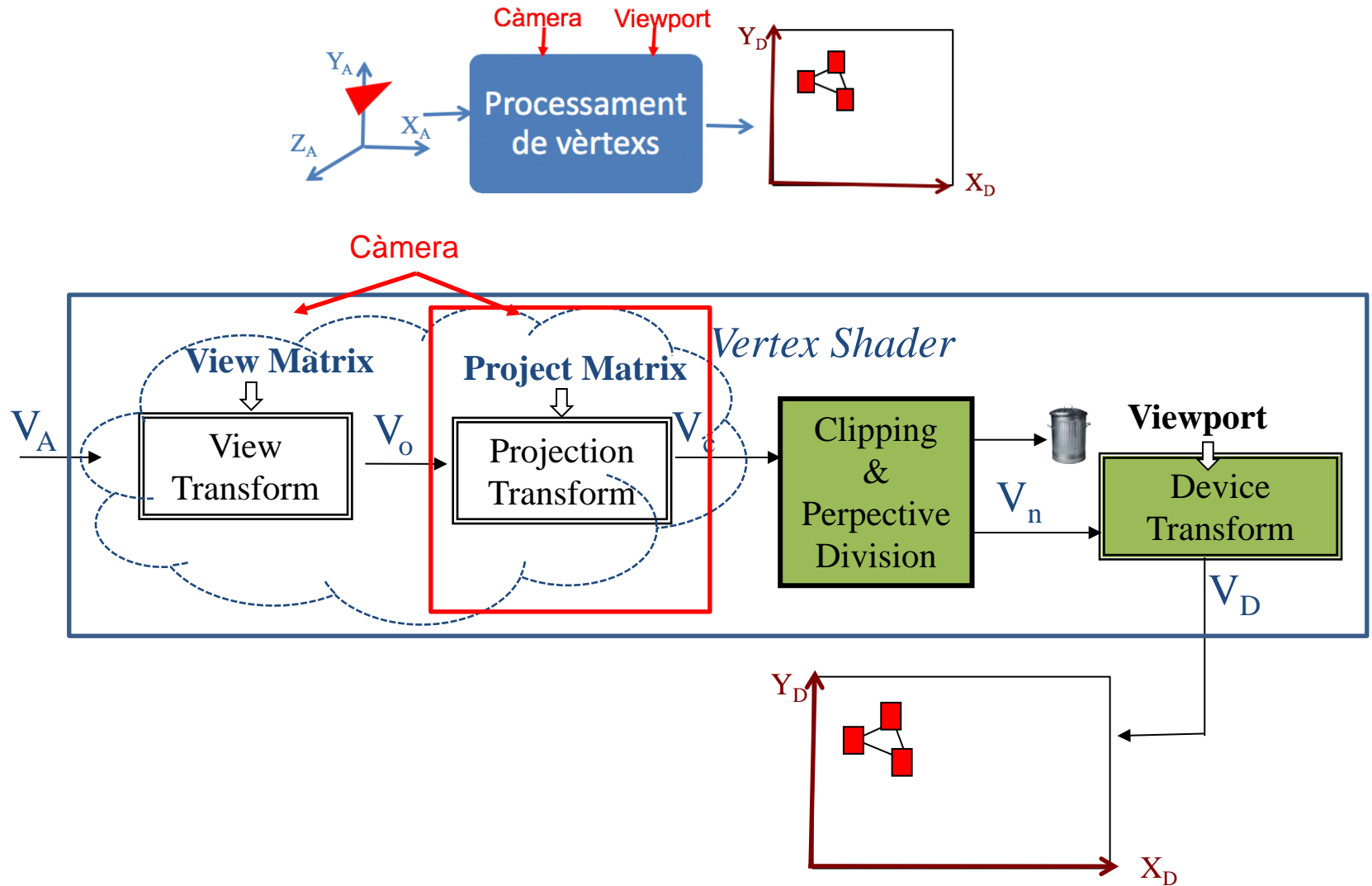
Canvi de punt de vista (canvi de Sistema de Coordenades)

Pas de SCA (Aplicació) a SCO (Observador)

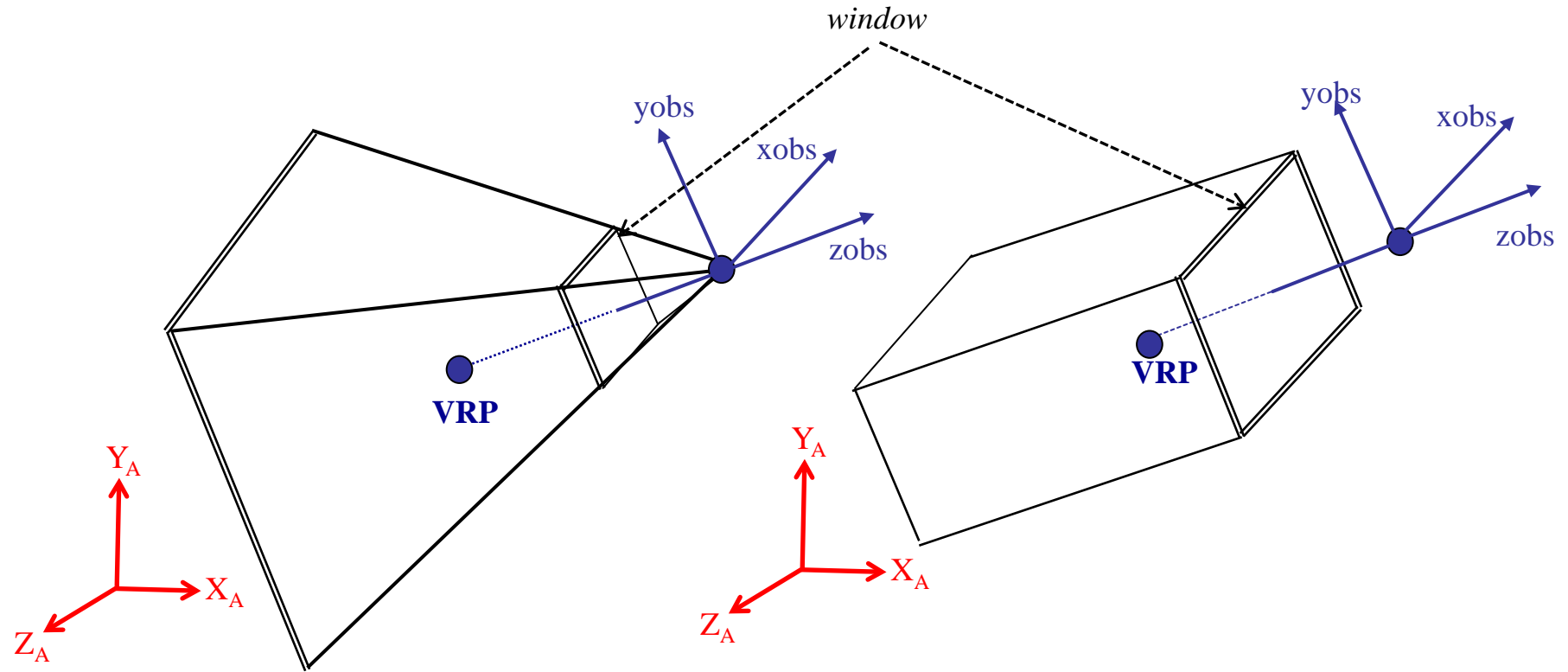
Classe 3: Especificació càmera

- La càmera en el Procés de Visualització
- Posició i orientació
- **Òptica**
 - **Perspectiva**
 - **Ortogonal**
- La càmera en el Vertex Shader
- Sistemes de coordenades – afegim TG
- Zoom
- Exemples

La càmera en el VS (2): Project Matrix

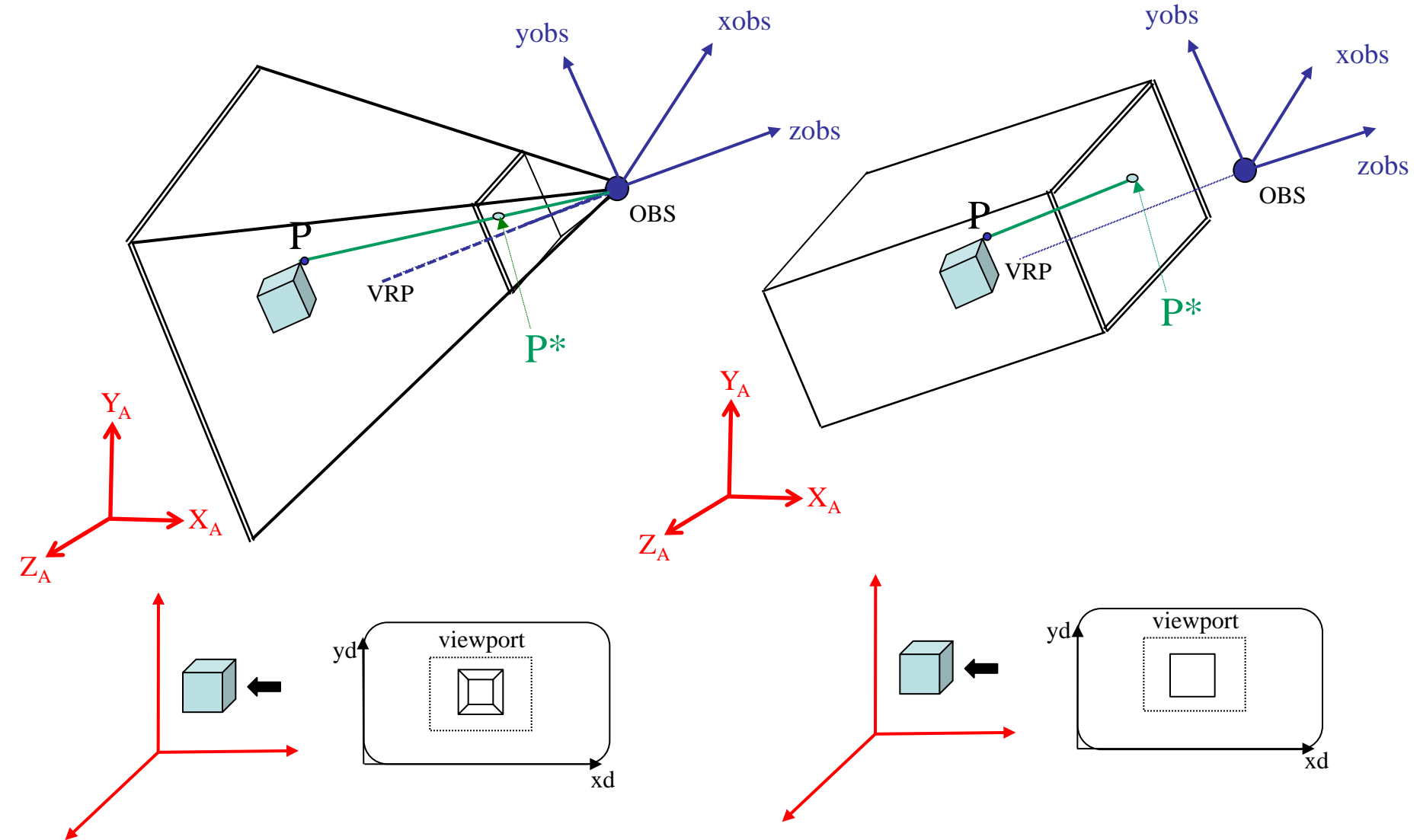


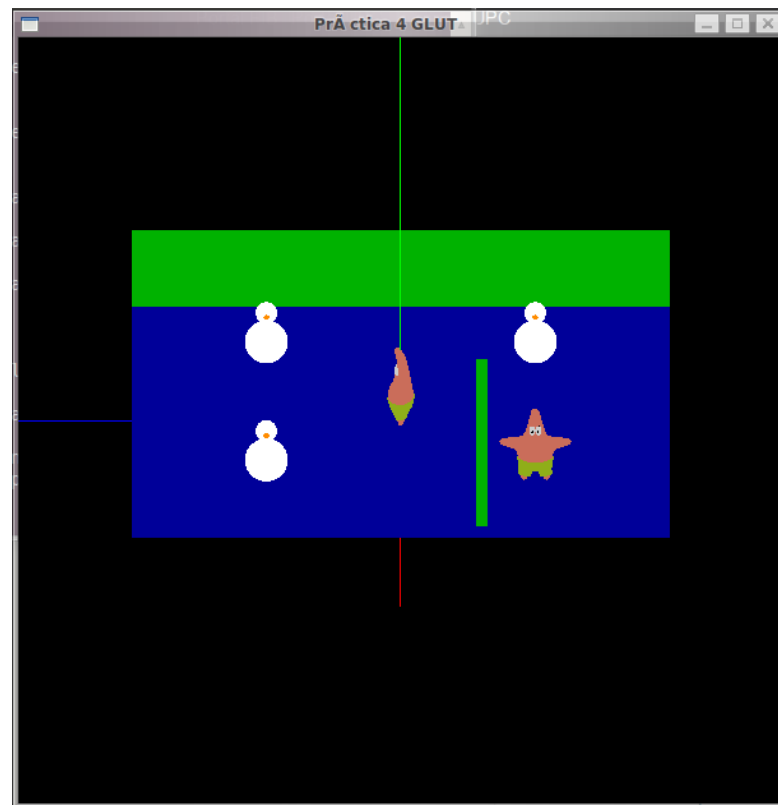
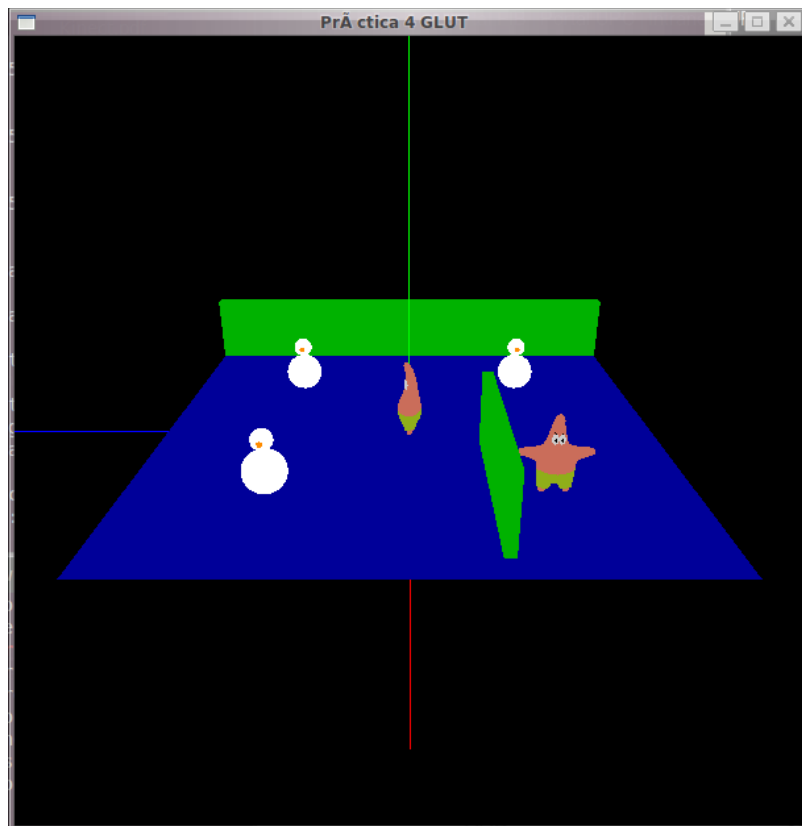
Òptica: Perspectiva o Ortogonal



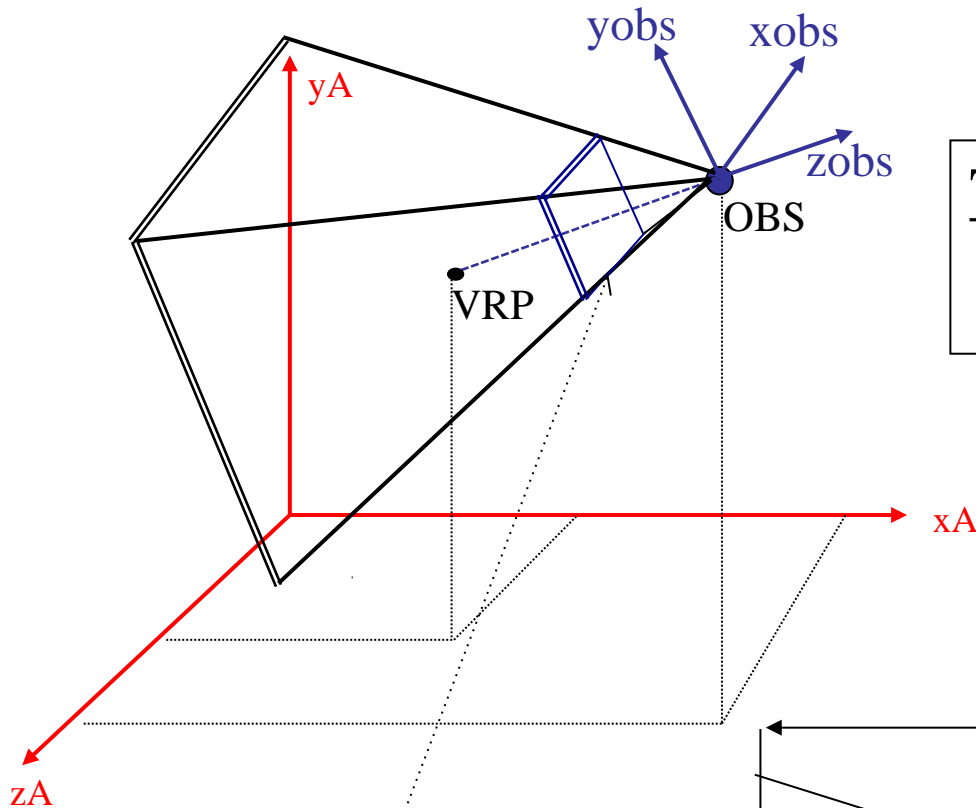
Definir òptica: volum de visió → (window, zNear, zFar)

Projecció: Perspectiva o Ortogonal





Òptica perspectiva: paràmetres



Tipus d'òptica: Perspectiva
 α_v ($FOV = 2\alpha_v$), ra_w , $zNear$, $zFar$

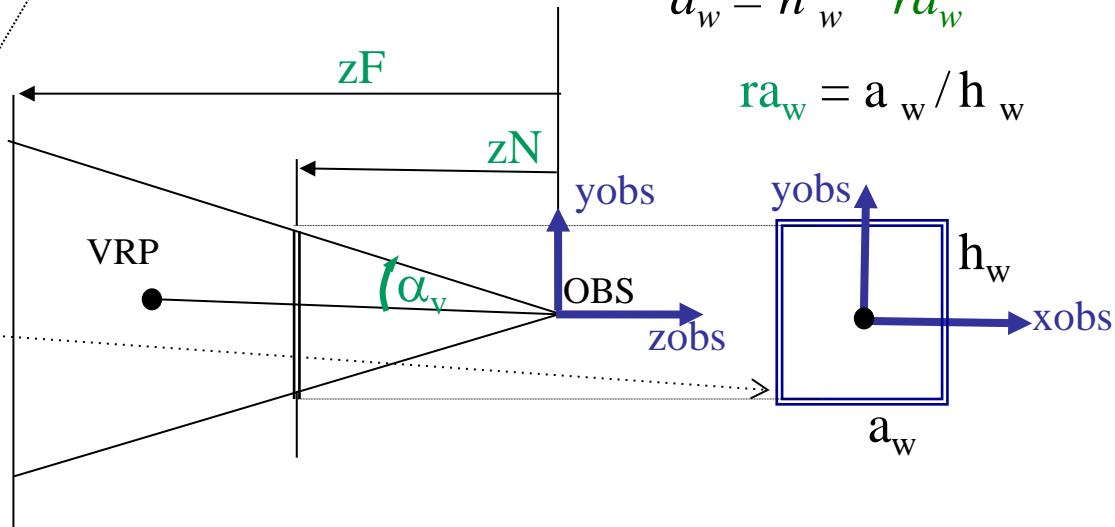
-window-

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

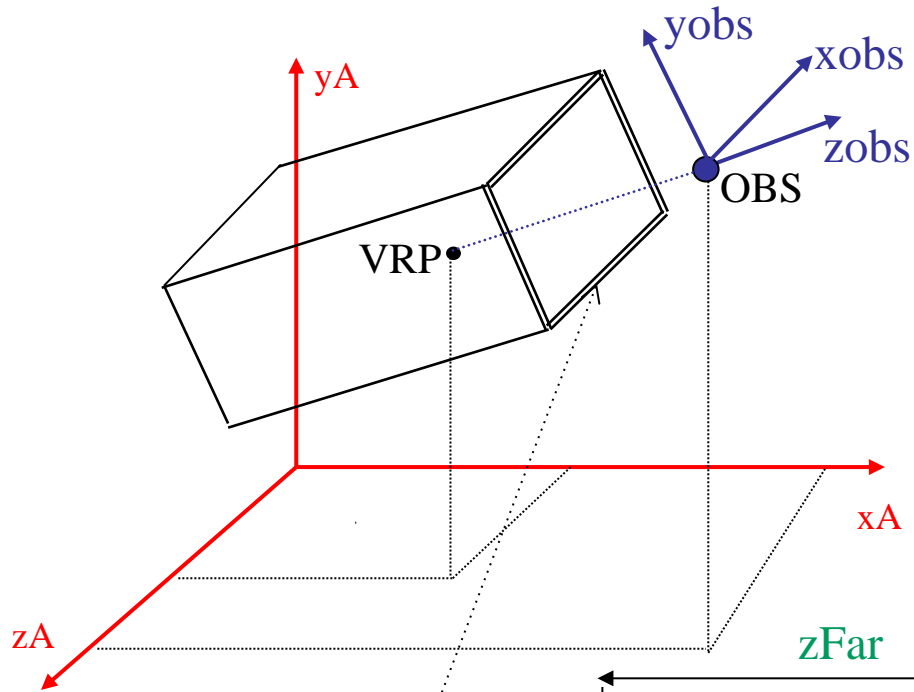
$$a_w = h_w * ra_w$$

$$ra_w = a_w / h_w$$

Window



Òptica ortogonal: paràmetres

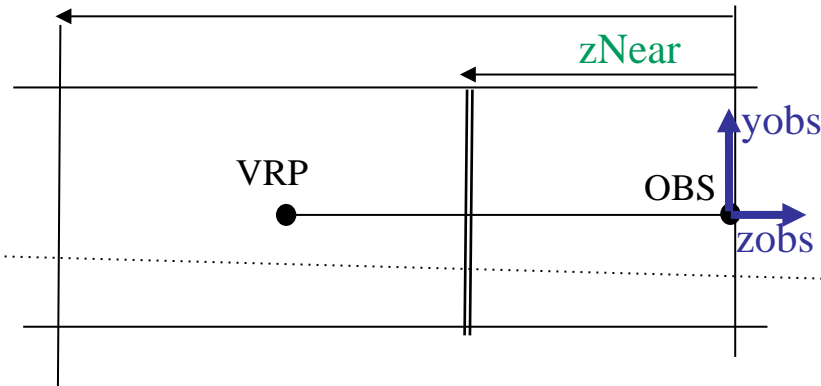


Tipus d'òptica: ortogonal

$l, r, b, t, z_{Near}, z_{Far}$

-window-

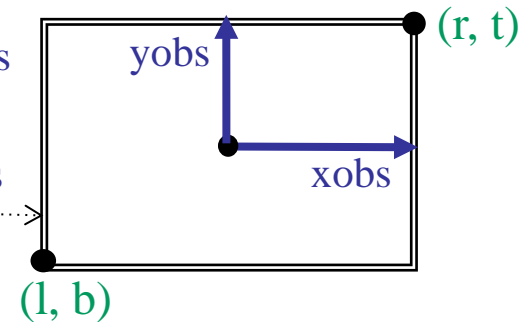
Window



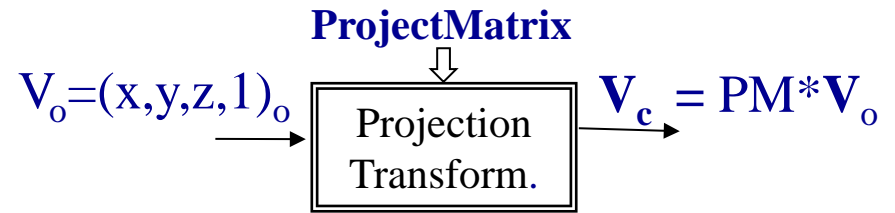
$$h_w = t - b$$

$$a_w = r - l$$

$$ra_w = a_w / h_w$$



Càlcul matriu Project Matrix



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

$a=2/(r-l)$ $b=2/(t-b)$
 $c=2/(zf-zn)$
 $d=(zn+zf)/(zf-zn)$

Òptica Ortogonal

$V_c = (x_c, y_c, z_c, w_c)$ on $w_c=1$

```

PM=ortho(l,r,b,t,zN,ZF);
projectMatrix(PM);
  
```

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

$a = \tan(FOV/2)$
 $c = (zf+zn)/(zn-zf)$
 $d = 2*zn*zf/(zn-zf)$

Òptica Perspectiva

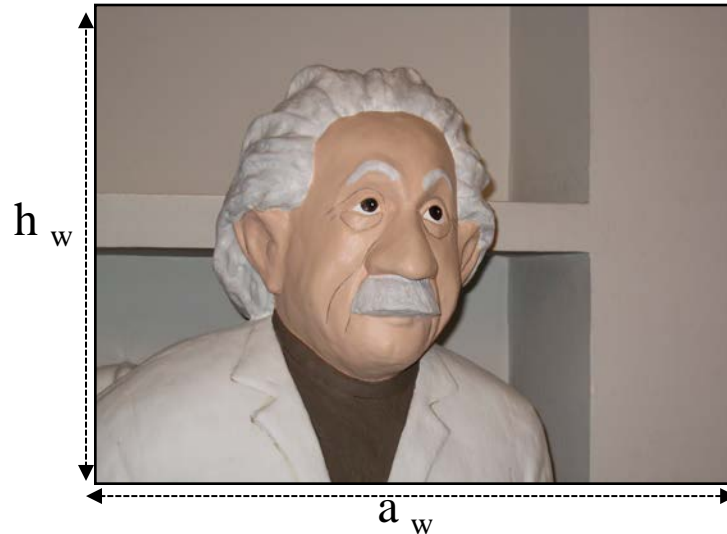
$V_c = (x_c, y_c, z_c, w_c)$ on $w_c=-z_o$

```

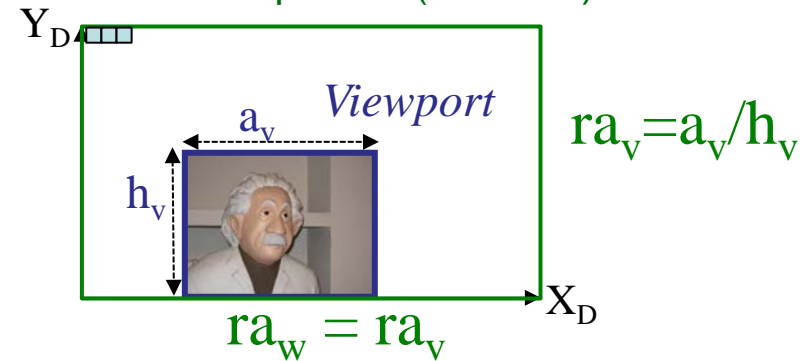
PM=perspective(FOV,ra,zN,ZF);
projectMatrix(PM);
  
```

Sobre la relació d'aspecte del window i del viewport

Window de l'òptica



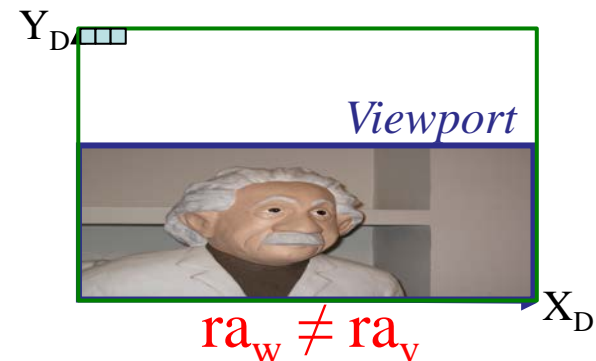
Finestra OpenGL (Pantalla)



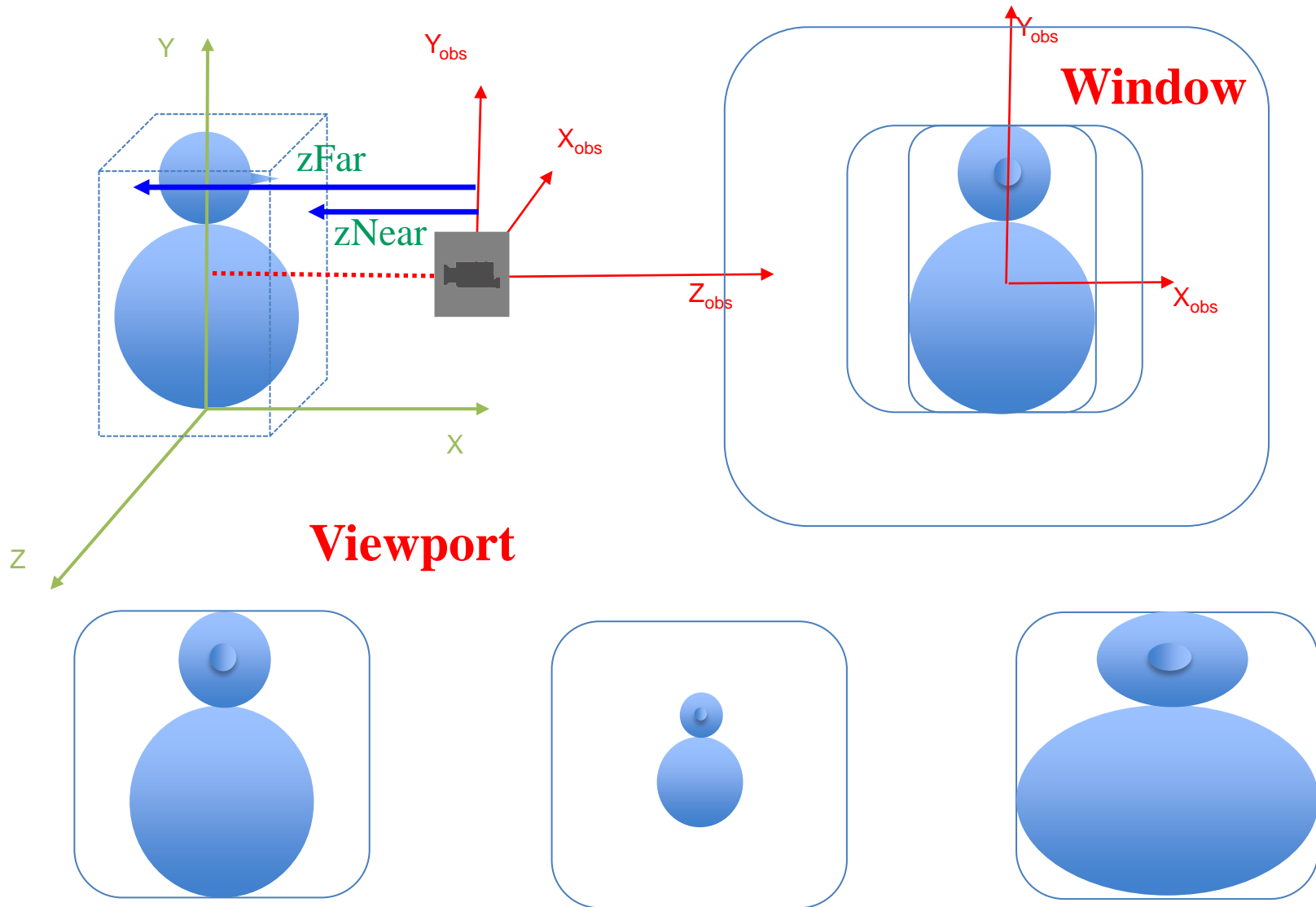
`glViewport (ox, oy, av, hv)`

Per a no tenir deformació en la imatge

$$ra_w = ra_v$$



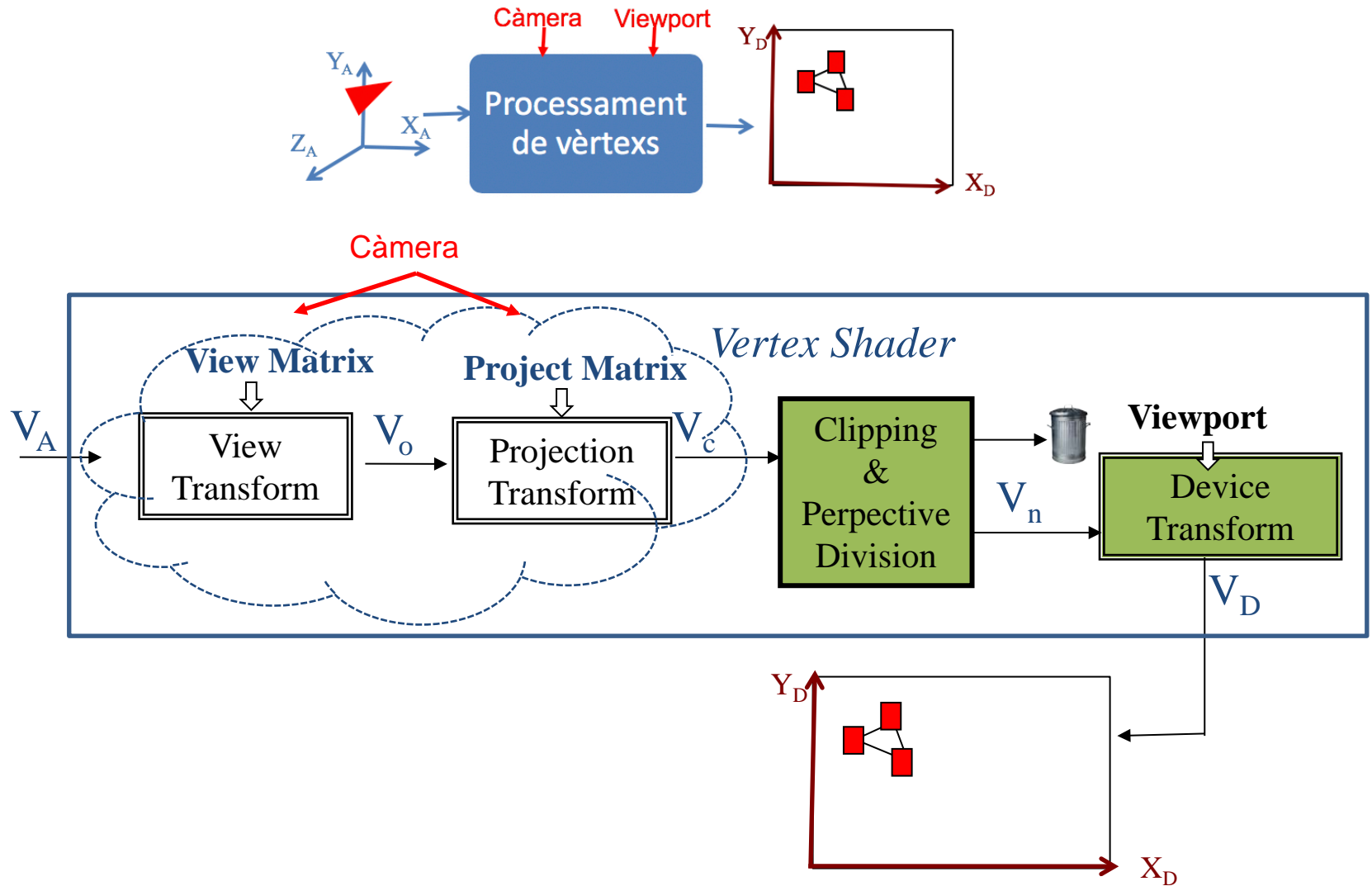
Example (conceptual)



Classe 3: Especificació càmera

- La càmera en el Procés de Visualització
- Posició i orientació
- Òptica
 - Perspectiva
 - Ortogonal
- **La càmera en el Vertex Shader**
- Sistemes de coordenades – afegim TG
- Zoom
- Exemples

La càmera en el VS (3): Implementació



El vèrtex Shader: Implementació

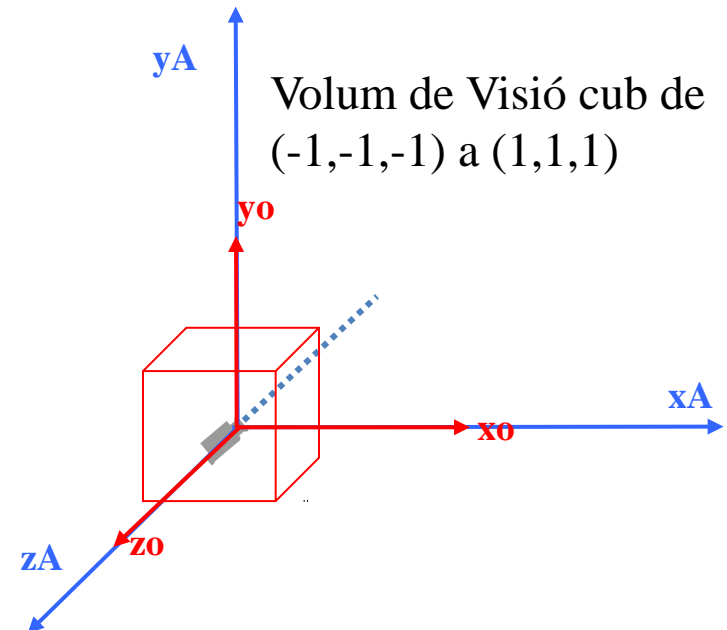
Vertex Shader

```
#version 330 core

in vec3 vertex;
uniform mat4 PM;
uniform mat4 VM;

void main() {
    gl_Position = PM*VM*vec4 (vertex, 1.0);
}
```

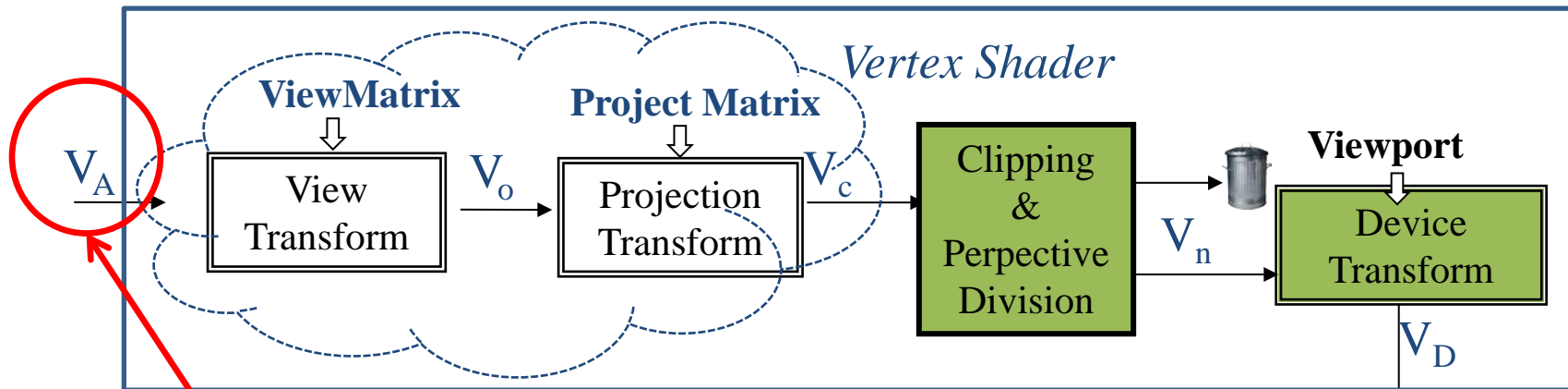
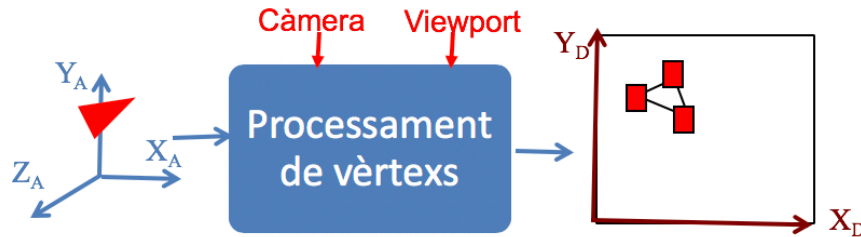
```
#version 330 core
in vec3 vertex;
//uniform mat4 PM; equivalent a identitat
//uniform mat4 VM; equivalent a identitat
void main() {
    //gl_Position = PM*VM*vec4 (vertex, 1.0);
    gl_Position = vec4 (vertex, 1.0);
}
```



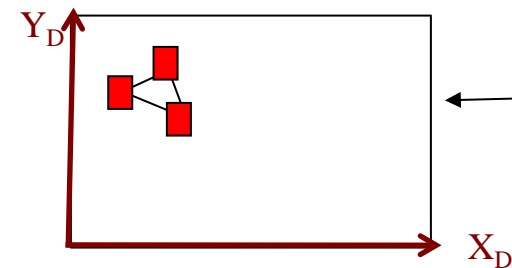
Classe 3: Especificació càmera

- La càmera en el Procés de Visualització
- Posició i orientació
- Òptica
 - Perspectiva
 - Ortogonal
- La càmera en el Vertex Shader
- **Sistemes de coordenades – afegim TG**
- Zoom
- Exemples

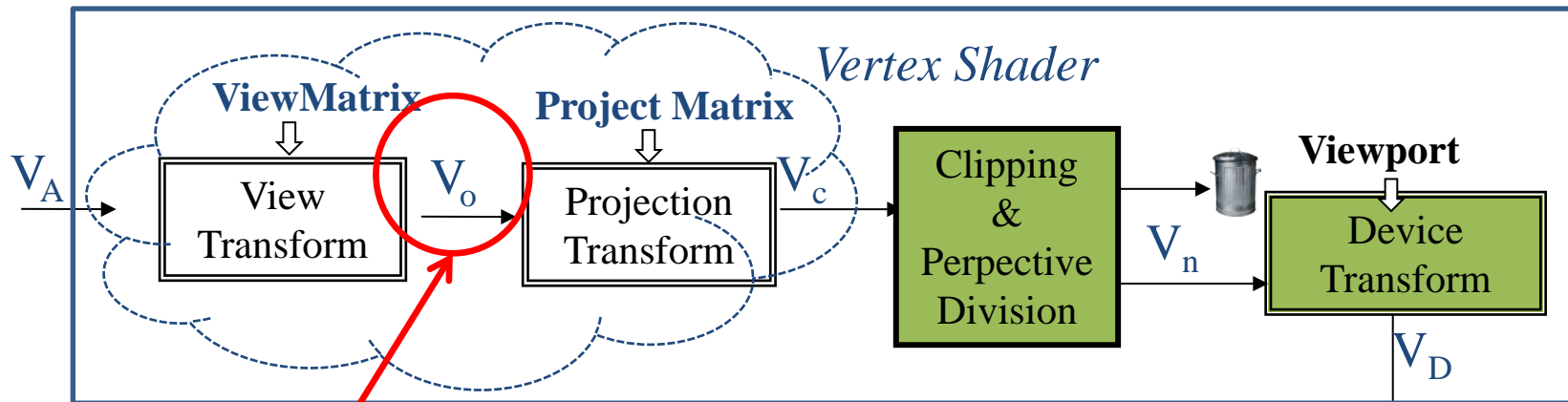
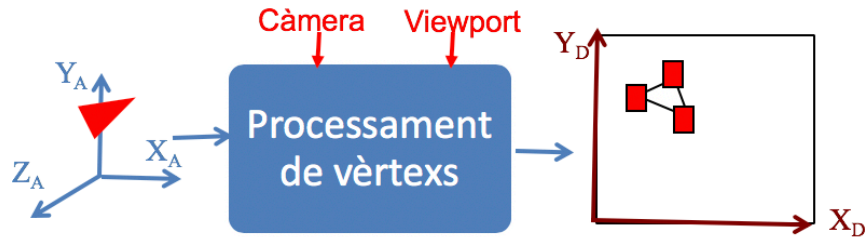
Sistemes de coordenades



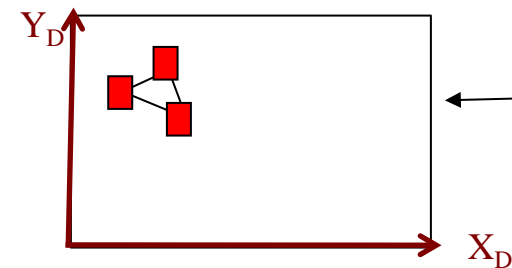
**Sistema de Coordenades
d'Aplicació (SCA)**



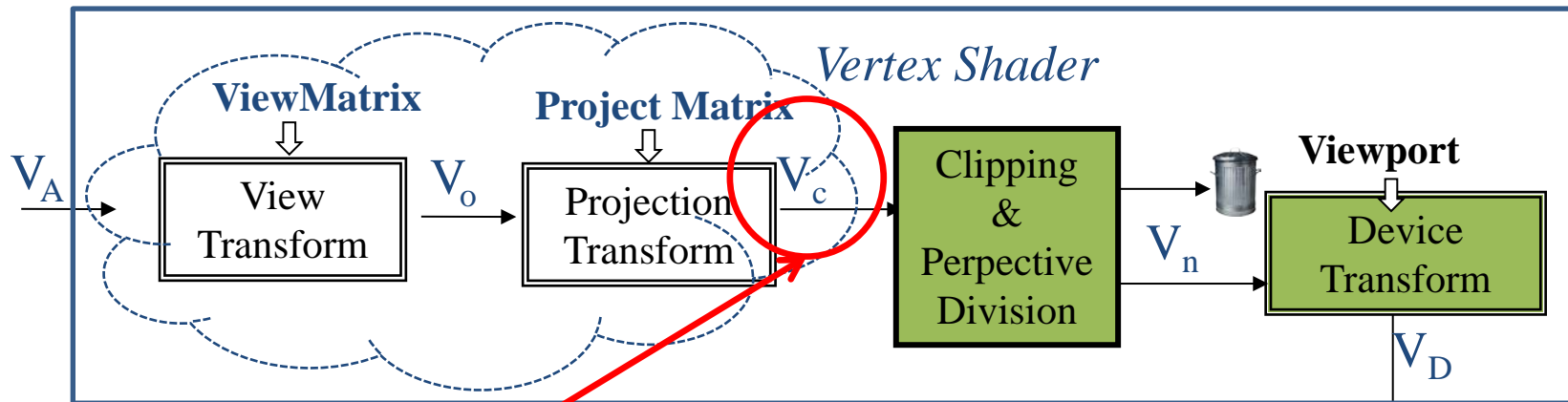
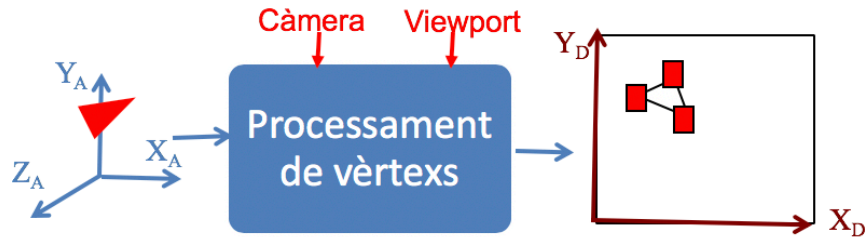
Sistemes de coordenades



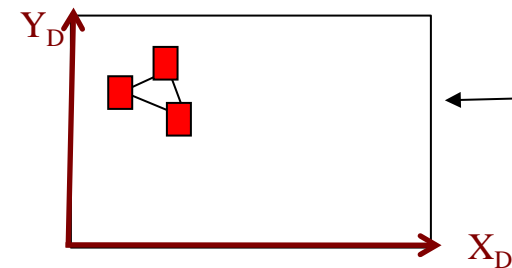
**Sistema de Coordenades
d'Observador(SCO)**



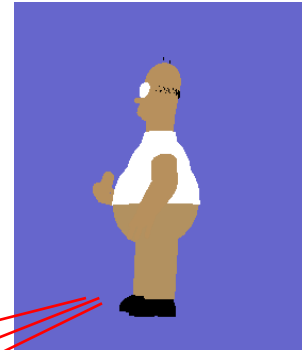
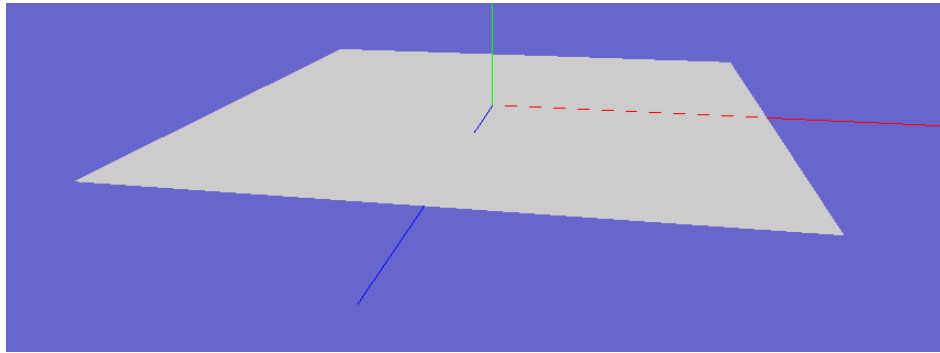
Sistemes de coordenades



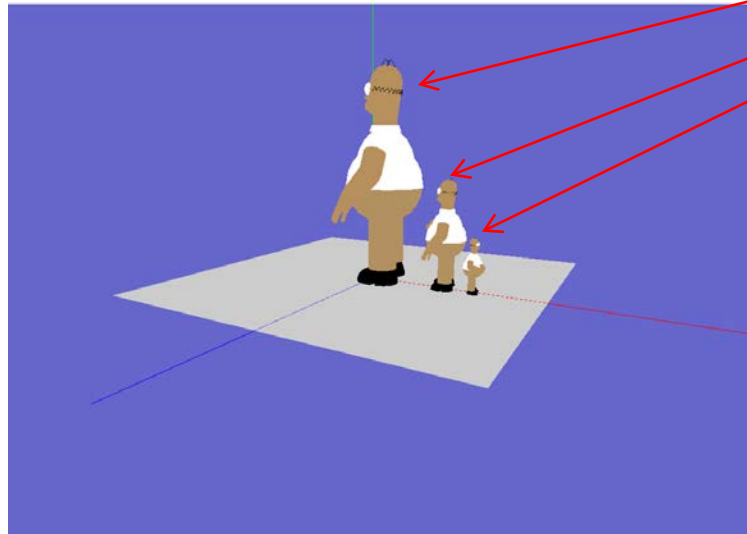
**Sistema de Coordenades
de Clipping (SCC)**



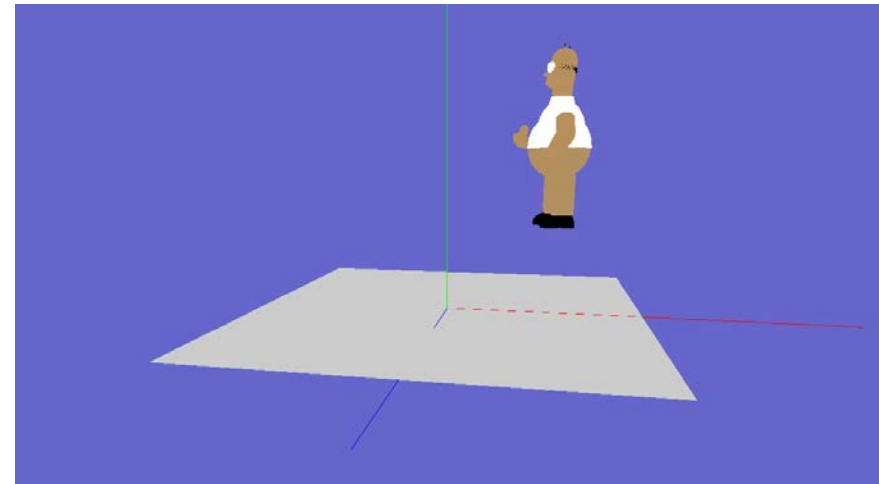
Recordem: Objectes en coordenades de model i ubicació en escena



TG_i



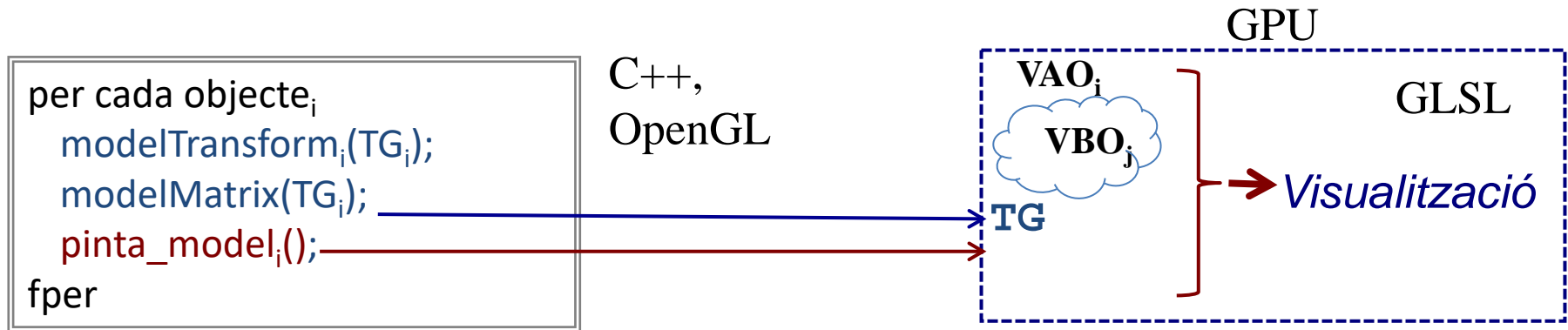
SCAplicació



SCModel

$$V_A = TG_i * V_m$$

Recordem: com visualitzar l'escena?



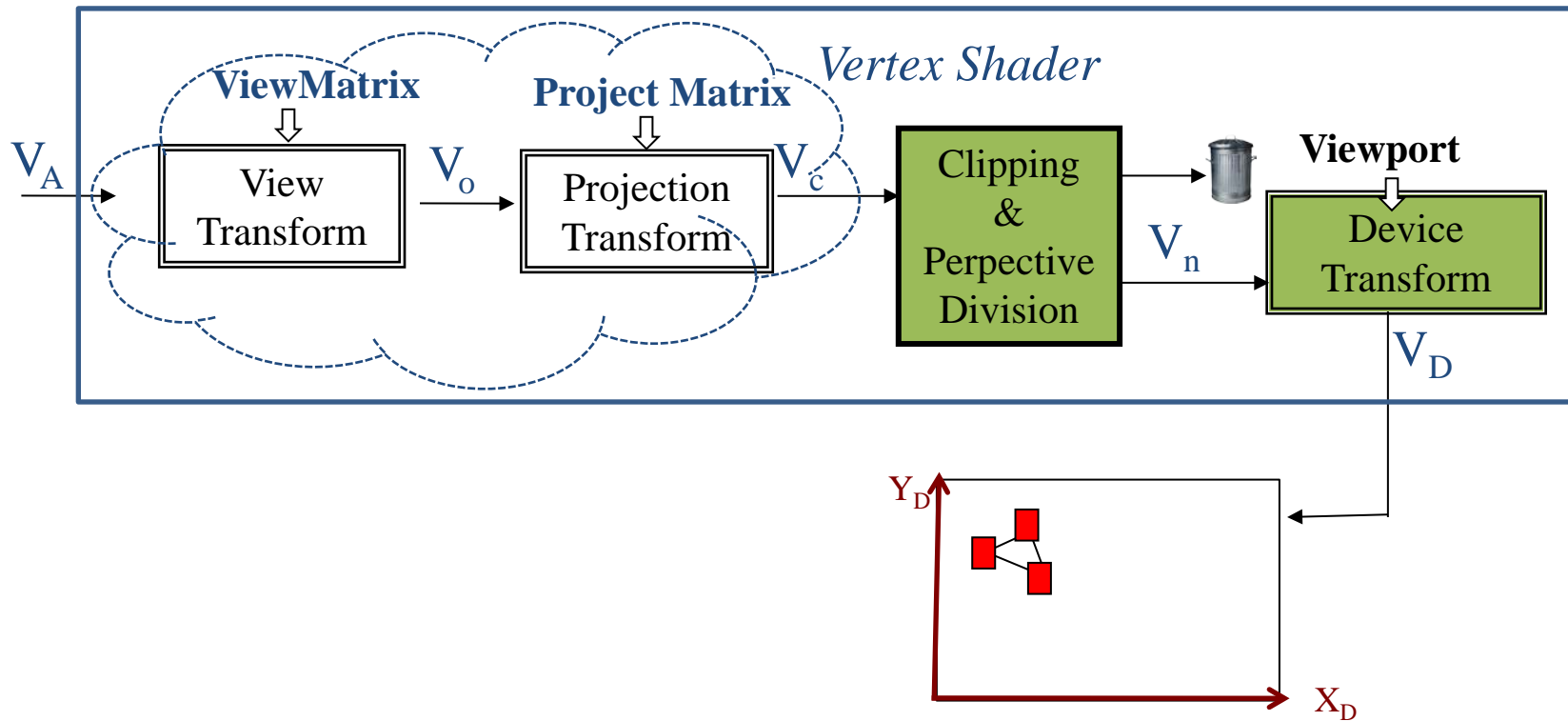
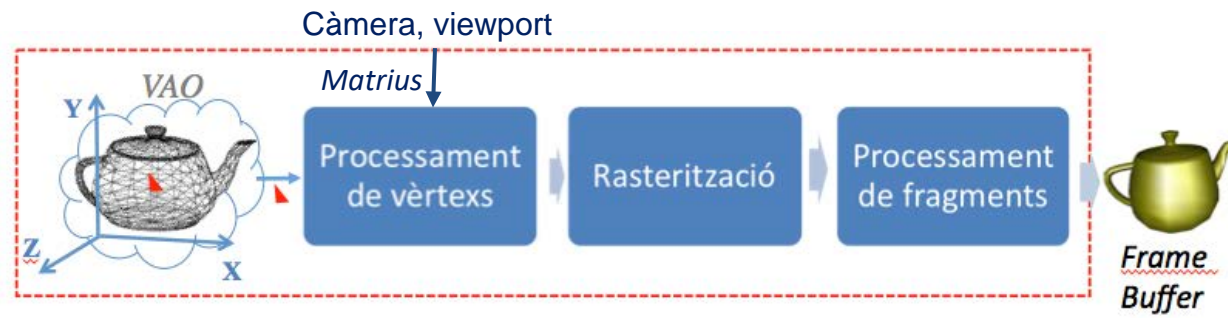
Vertex Shader

```
#version 330 core
in vec3 vertex;
uniform mat4 TG;

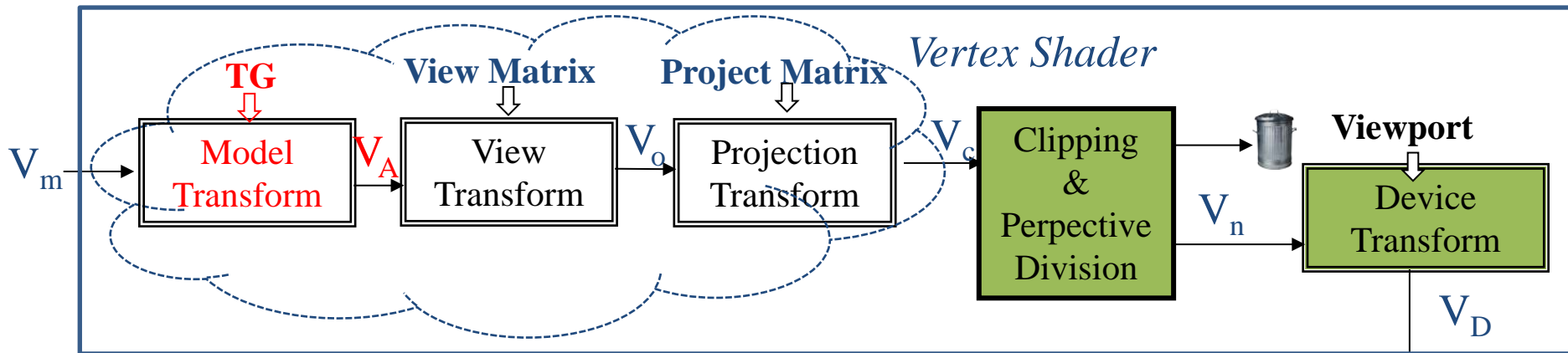
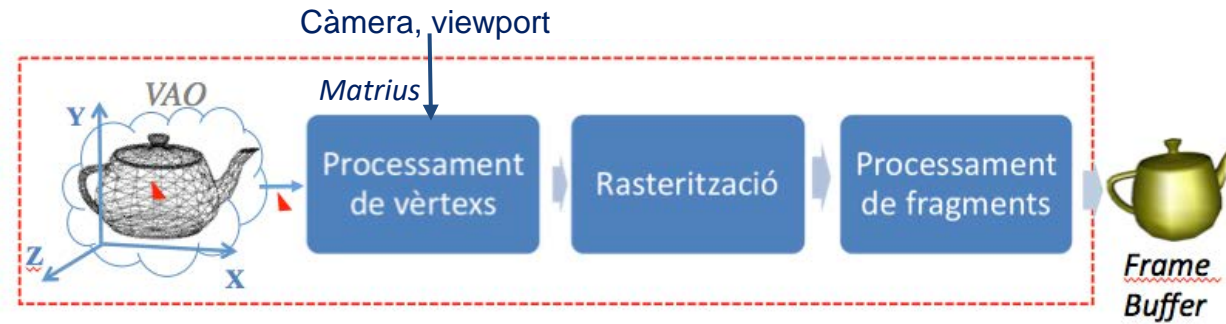
void main() {
    gl_Position = TG * vec4(vertex,1.0);
}
```

*Però no teníem en compte
la càmera...*

Vertex Shader – afegim TG



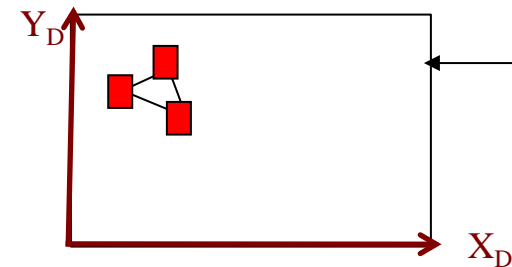
Vertex Shader – afegim TG



$$V_A = \text{TG} * V_m$$

$$V_O = VM * V_A = VM * TG * V_m$$

$$V_C = PM * V_O = PM * VM * TG * V_m$$



Programació bàsica del vertex shader

```
#version 330 core
```

```
in vec3 vertex;
```

```
uniform mat4 PM;
```

```
uniform mat4 VM;
```

```
uniform mat4 TG;
```

```
void main() {
```

```
    gl_Position = PM*VM*TG*vec4 (vertex, 1.0);
```

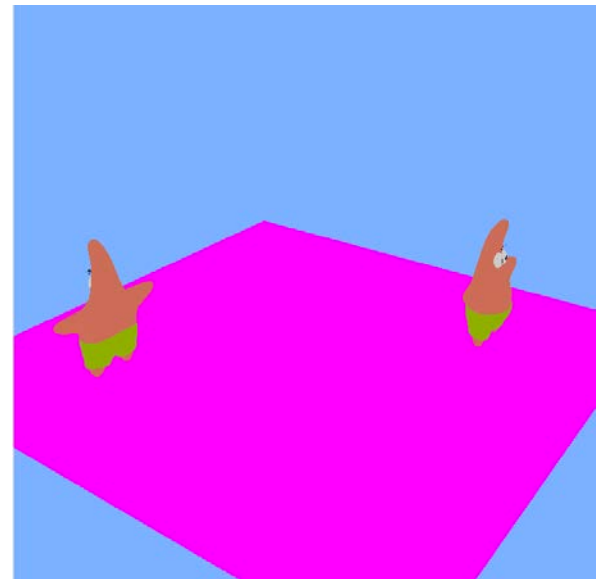
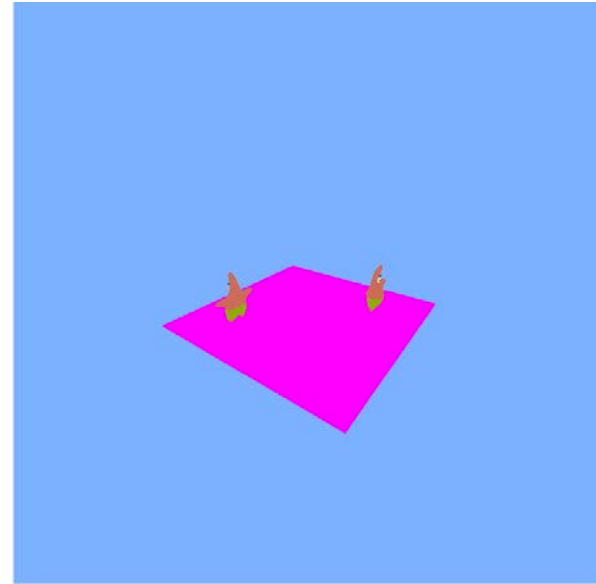
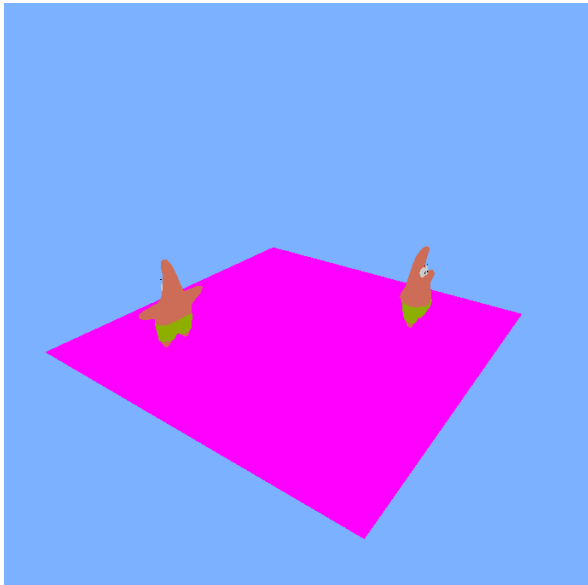
```
}
```

Vertex Shader

Classe 3: Especificació càmera

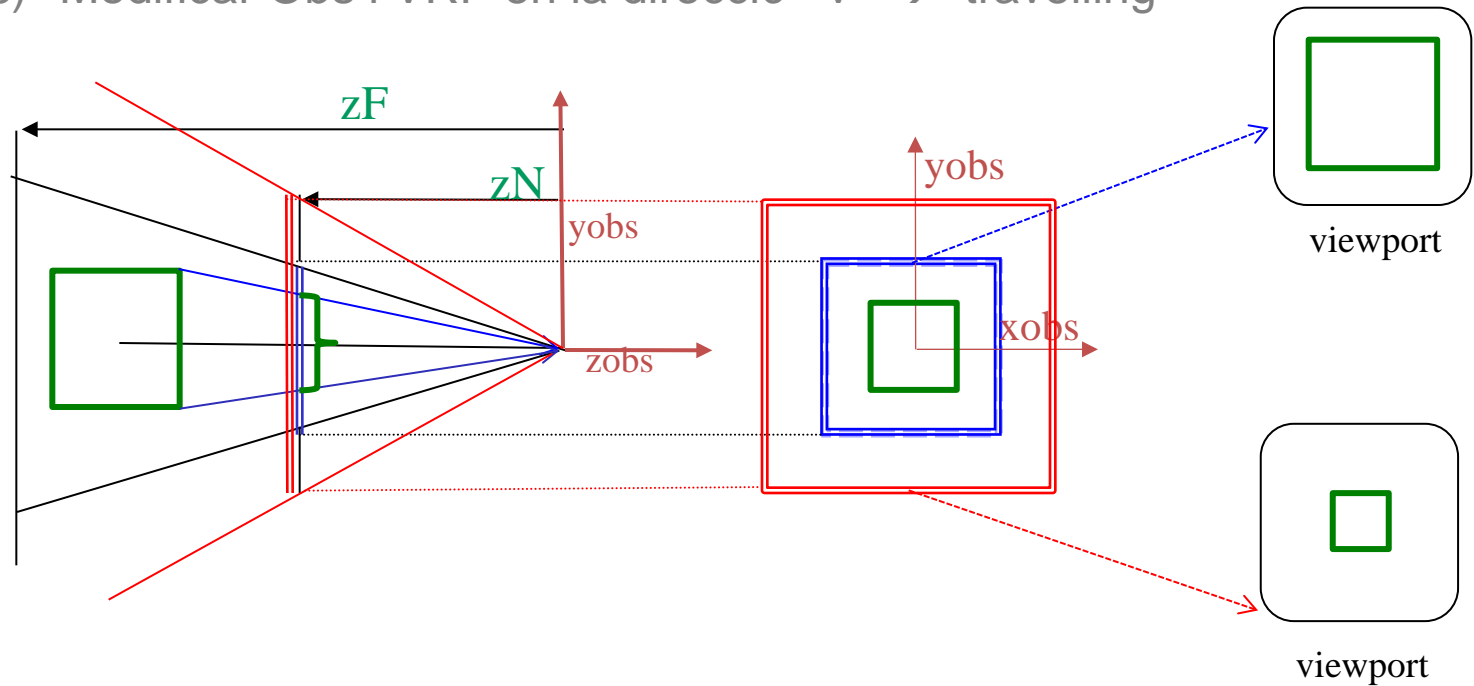
- La càmera en el Procés de Visualització
- Posició i orientació
- Òptica
 - Perspectiva
 - Ortogonal
- La càmera en el Vertex Shader
- Sistemes de coordenades – afegim TG
- **Zoom**
- Exemples

Zoom



L'òptica i el Zoom

- a) Modificar el window de la càmera:
 - L'angle d'obertura en òptica perspectiva (tot mantenint la ra)
 - Window en òptica ortogonal (tot mantenint la ra)
- b) Modificar la distància de l'OBS al VRP
(modificant zN i zF adequadament)
- c) Modificar Obs i VRP en la direcció $-v \rightarrow$ travelling



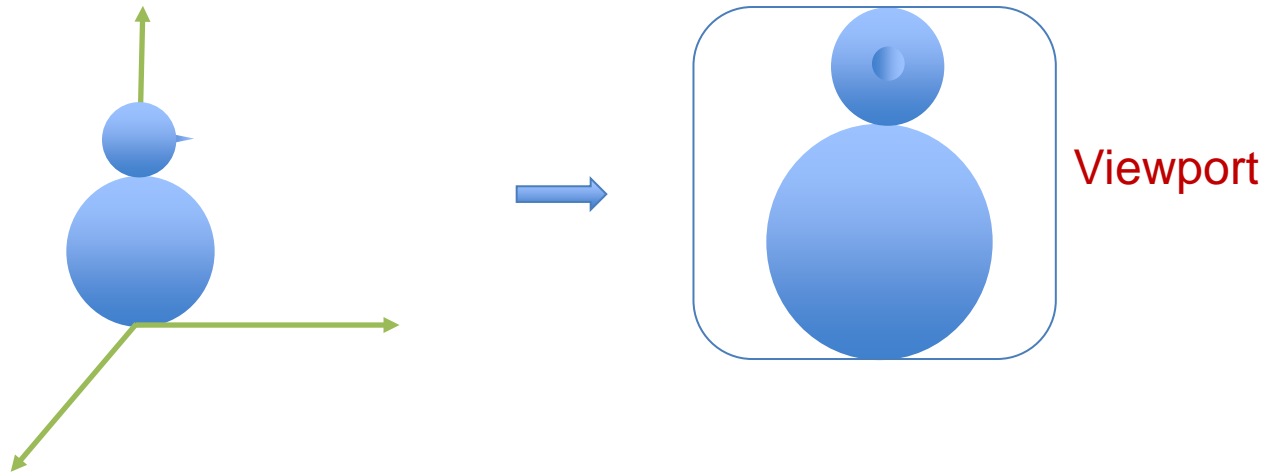
Classe 3: Especificació càmera

- La càmera en el Procés de Visualització
- Posició i orientació
- Òptica
 - Perspectiva
 - Ortogonal
- La càmera en el Vertex Shader
- Sistemes de coordenades – afegim TG
- Zoom
- **Exemples**

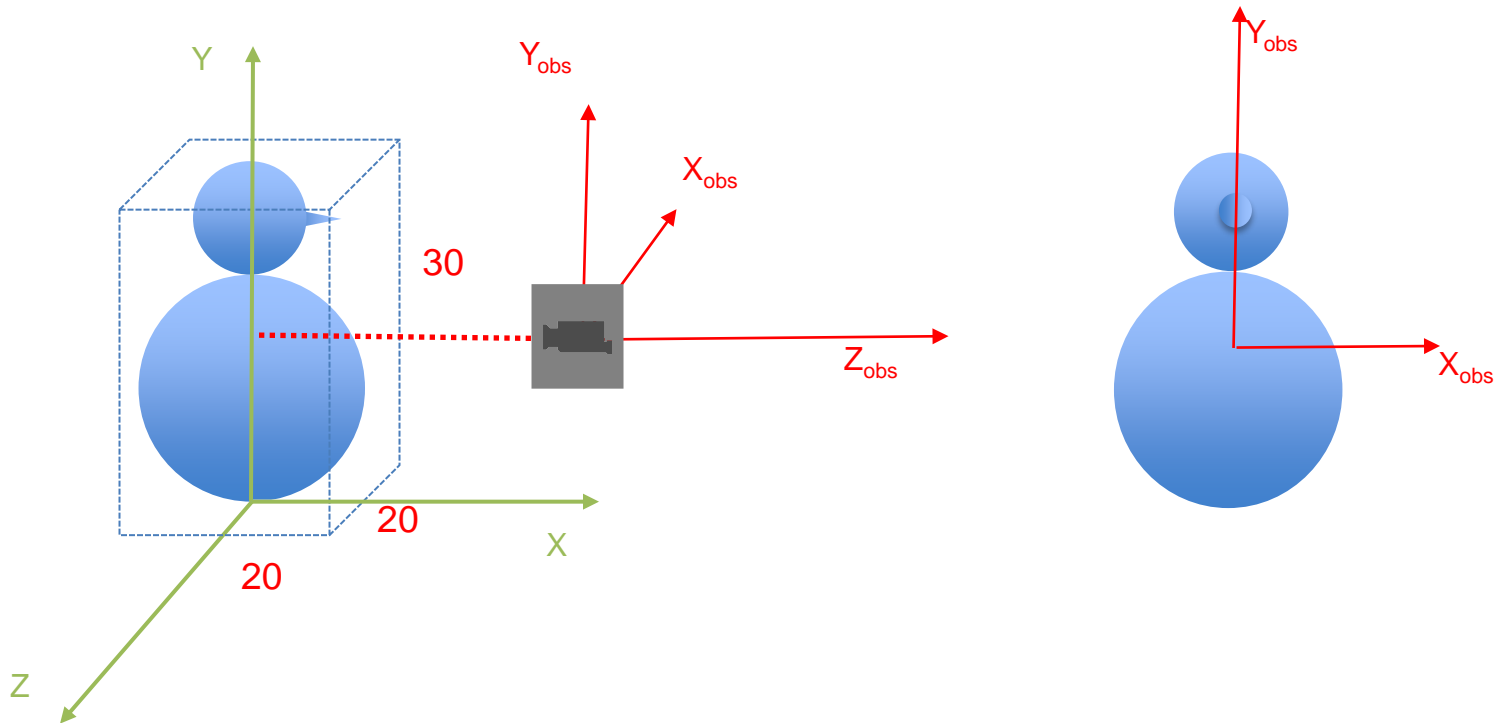
Exemple 1: Donada una funció `pinta_ninot()` que envia a visualitzar el VAO d'un objecte com el de la figura que està format per: una esfera de radi 10 amb centre $(0,10,0)$, una altra esfera de radi 5 amb centre $(0,25,0)$, i un con de base centrada en $(2.5, 25,0)$, radi 2 i llargada 5 orientat segons l'eix X^+

Indica tots els paràmetres d'una càmera que permeti obtenir una imatge similar a la mostrada en l'esquema de la Figura. El viewport és de 600x600 píxels.

- a) Indica els paràmetres per a una òptica perspectiva
- b) Indica els paràmetres per a una òptica ortogonal



Exemple 1: posició i orientació

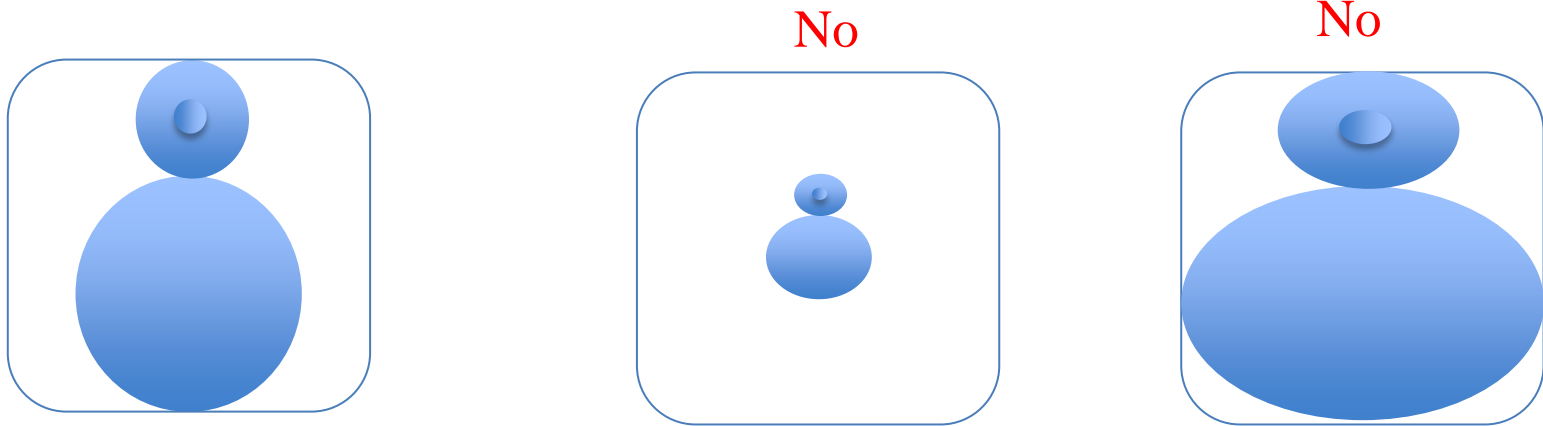


$$VRP = (0, 15, 0)$$

$$OBS = (30, 15, 0)$$

$$up = (0, 1, 0)$$

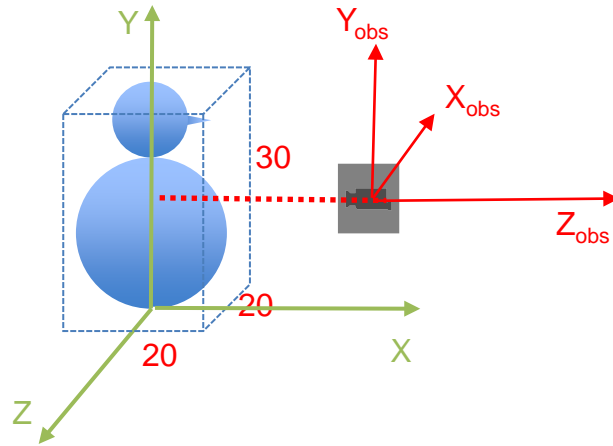
Exemple 1: òptica



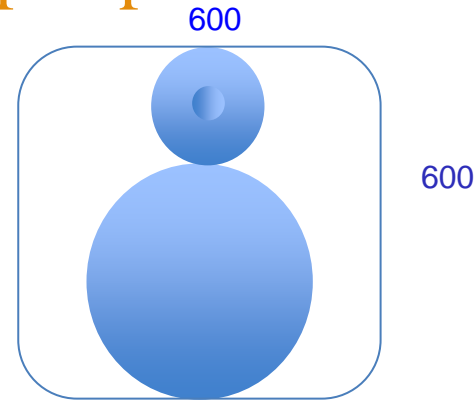
Restriccions:

- Viewport de 600x600
- Ninot centrat al viewport
- Ninot optimitzi espai en viewport
- Sense deformacions

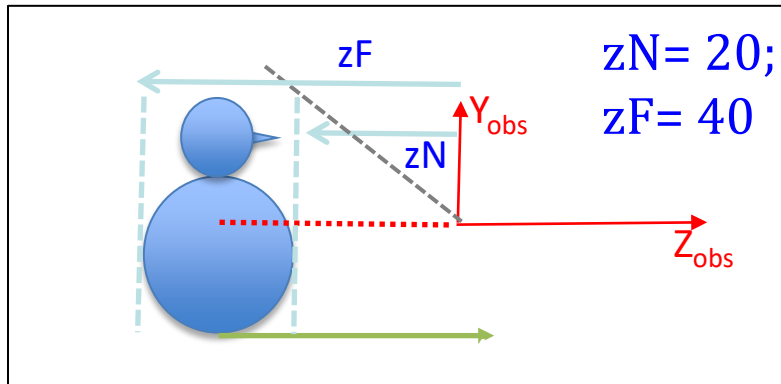
Exemple 1 a): òptica perspectiva



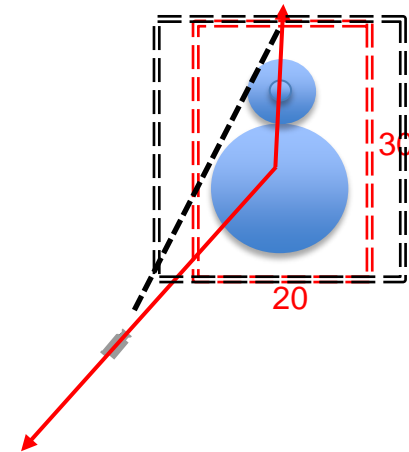
VRP=(0,15,0); OBS=(30,15,0), Up=(0,1,0)



FOV
ra_w
zNear
zFar



zN= 20;
zF= 40



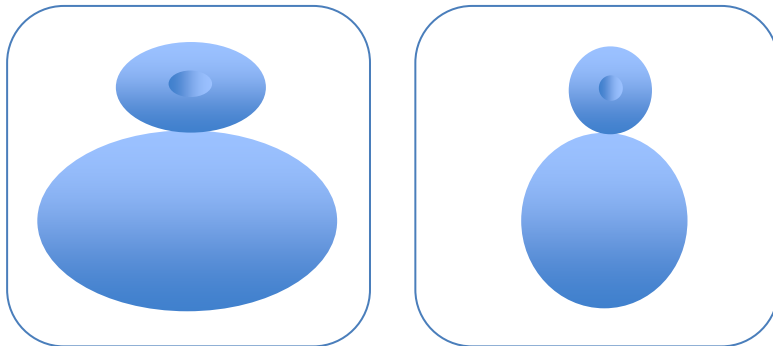
alfa = arctg (15/20) → alfa = 36,8°

FOV = 2 * alfa

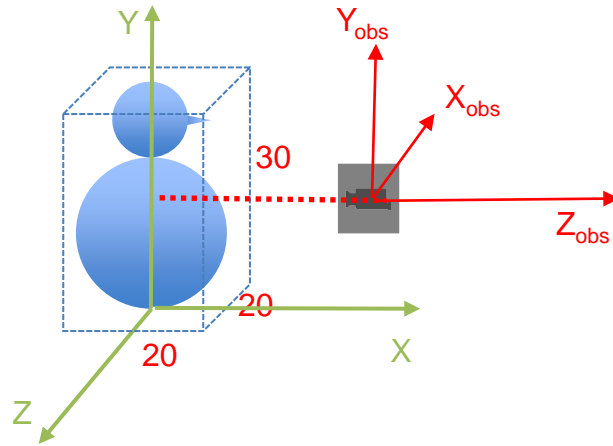
ra_w = 20/30= 0,66

Com ra_v = 1 → deformació

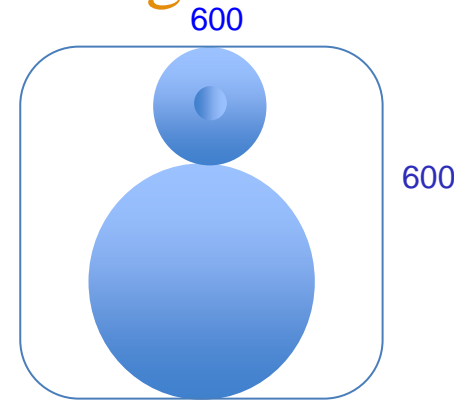
Solució ra_w = 1



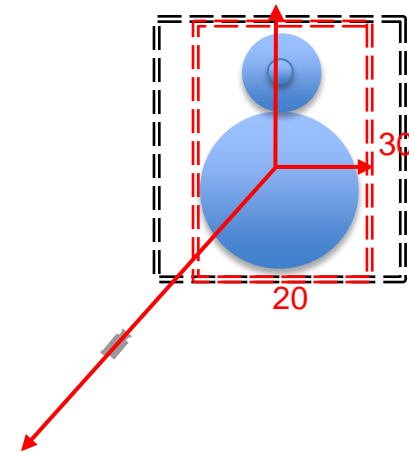
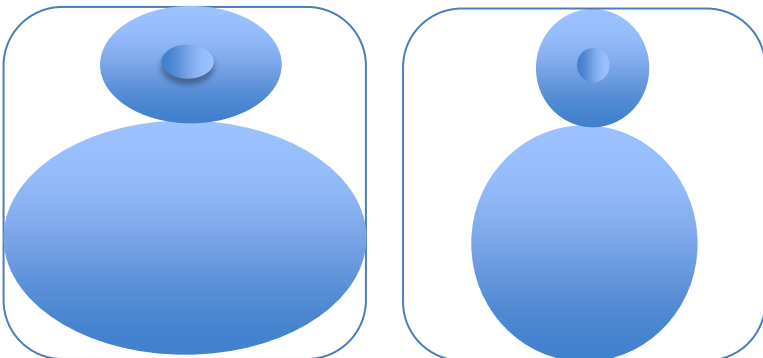
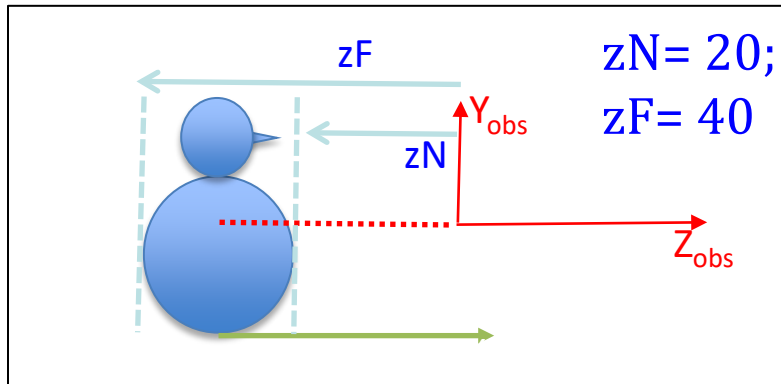
Exemple 1 b): òptica ortogonal



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



left, right
bottom, top
zNear
zFar



left = -10, right = 10

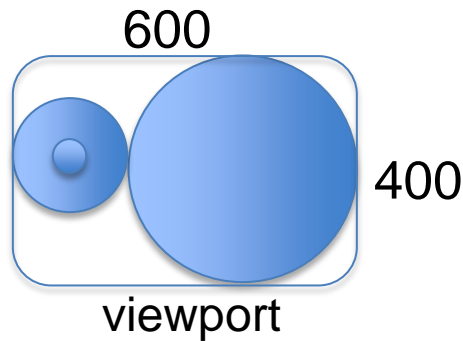
bottom = -15, top = 15

Com $ra_v = 1 \rightarrow$ deformació

Solució left = -15, right = 15
per tenir $ra_w = 1$

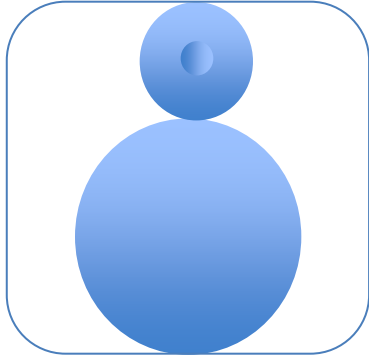
Exemple 1 bis

Què caldria canviar dels paràmetres de posicionament de càmera de l'exemple 1 per obtenir la imatge següent?



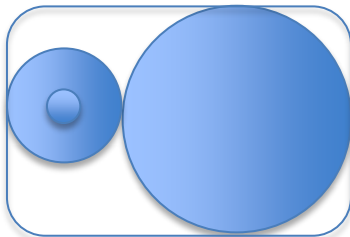
Exemple 1 bis

Tením



$VRP=(0,15,0)$; $OBS=(30,15,0)$, $up=(0,1,0)$

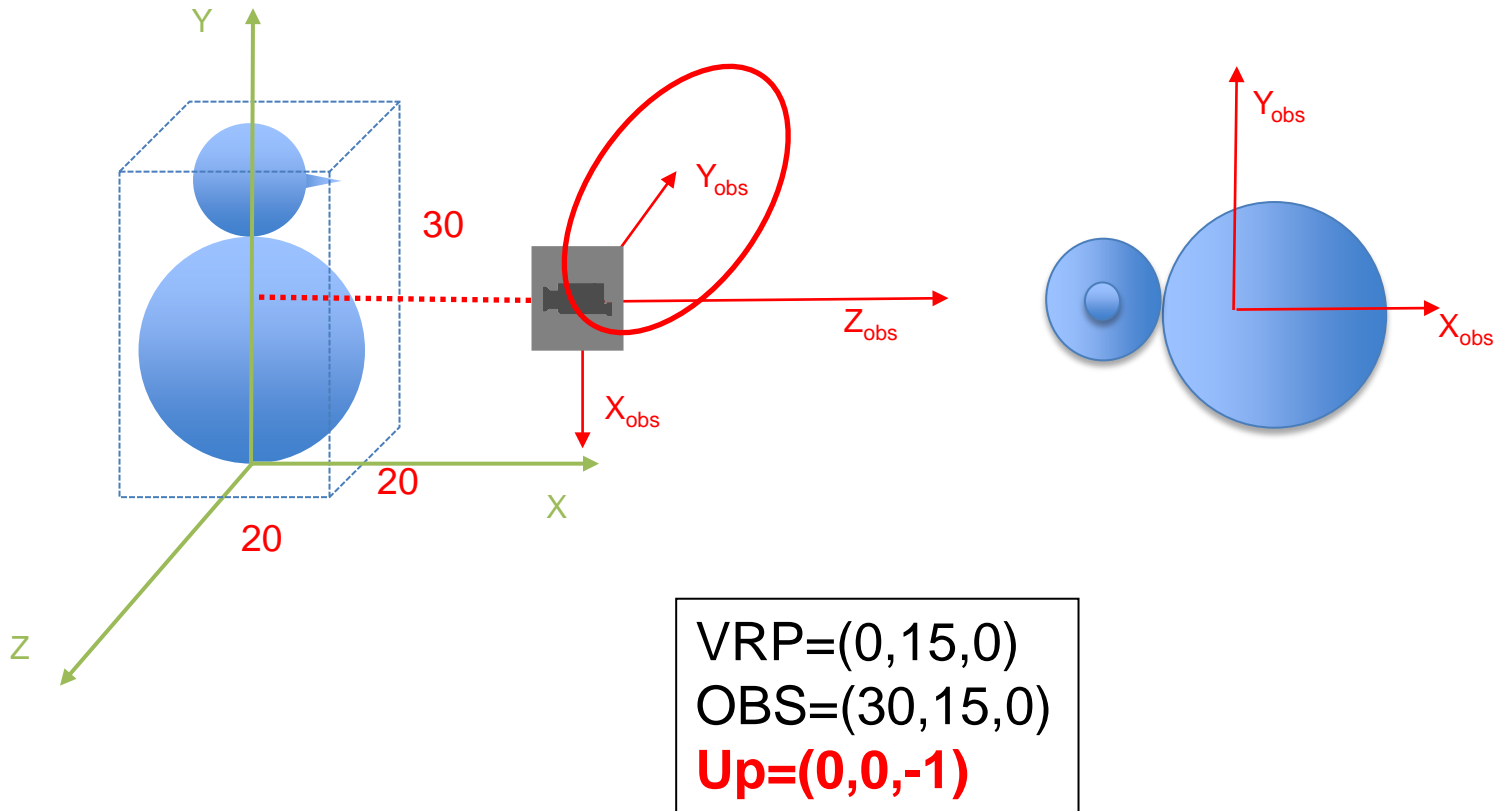
Volem



$VRP=(0,15,0)$ i $OBS=(30,15,0)$ **igual**

Canviarà up!

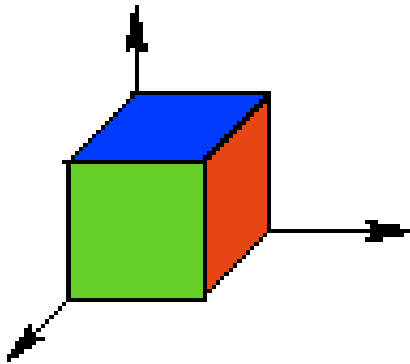
Exemple 1 bis



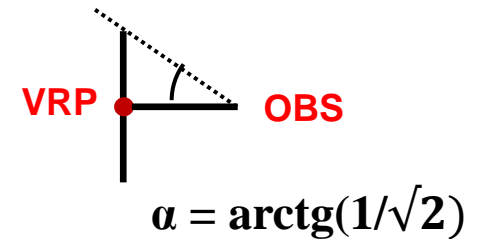
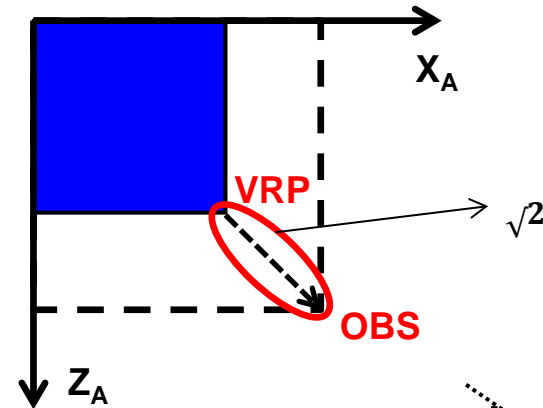
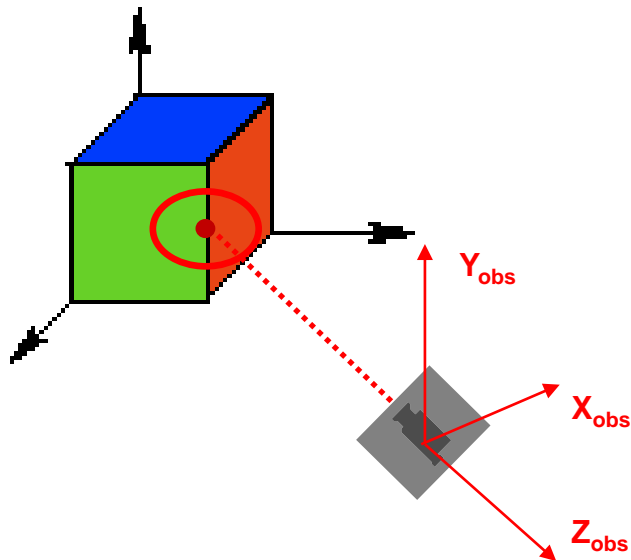
Exemple 2. Tenim una escena amb un cub de costat 2 orientat amb els eixos i de manera que el seu vèrtex mínim està situat a l'origen de coordenades. La cara del cub que queda sobre el pla $x=2$ és de color vermell, la cara que queda sobre el pla $z=2$ és de color verd i la resta de cares són blaves.

a) Indica TOTS els paràmetres d'una càmera perspectiva que permeti veure completes a la vista només les cares vermella i verda. La relació d'aspecte del viewport (vista) és 2. Fes un dibuix indicant la imatge final que s'obtingria.

b) Quin efecte tindria en la imatge final modificar l'òptica ortogonal? Defineix la càmera ortogonal.



Exemple 2 a)



Posició i orientació

VRP = (2,1,2)
OBS = (3,1,3)
Up = (0,1,0)

Òptica perspectiva

$0 < Z_{\text{near}} \leq \sqrt{2}$
 $Z_{\text{far}} \geq 3 * \sqrt{2}$
 $\text{FOV} = 2 * \text{arctg}(1/\sqrt{2})$
 $r_{a_w} = r_{a_v} = 2$

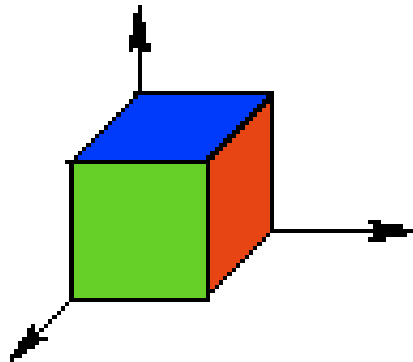
Imatge al viewport



Exemple 2. Tenim una escena amb un cub de costat 2 orientat amb els eixos i de manera que el seu vèrtex mínim està situat a l'origen de coordenades. La cara del cub que queda sobre el pla $x=2$ és de color vermell, la cara que queda sobre el pla $z=2$ és de color verd i la resta de cares són blaves.

a) Indica TOTS els paràmetres d'una càmera perspectiva que permeti veure completes a la vista només les cares vermella i verda. La relació d'aspecte del viewport (vista) és 2. Fes un dibuix indicant la imatge final que s'obtingria.

b) Quin efecte tindria en la imatge final modificar l'òptica ortogonal? Defineix la càmera ortogonal.



Penseu vosaltres l'apartat b)

Classe 3: Conceptes i preguntes

- Paràmetres requerits per posició i orientació de la càmera: VRP, OBS i Up. Quin efecte té modificar un qualsevol dels paràmetres en la imatge final?
- Pot tenir Up la direcció de visió (direcció de la recta que uneix VRP amb OBS)? Ha de coincidir amb l'eix Y del sistema de coordenades de l'observador?
- Paràmetres requerits per l'òptica perspectiva i ortogonal de la càmera. Quin efecte té modificar Znear i/o Zfar? i FOV? i window?
- Concepte de viewport/vista. Efecte de tenir relacions d'aspecte diferents en window i viewport.
- Càlcul dels paràmetres de càmera per crear una imatge determinada.
- Distingir els diferents sistemes de coordenades implicats en la definició de la càmera (SCA, SCO, SCC). Quantes matrius cal passar a la GPU? Com les obtenim?
- Si no es multiplica el vèrtex en el vertex shader per cap matriu, quina càmera estem usant?
- Zoom i modificació dels paràmetres que comporta.
- Com passar de coordenades de model (SCM) a aplicació (SCA) en el Vertex Shader.
- Si hi ha un objecte que ja està correctament posicionat en el seu VAO, cal passar una TG al shader? Suposant que l'escena té més d'un objecte.