

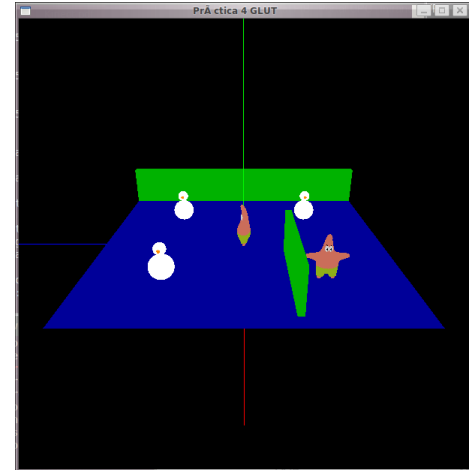
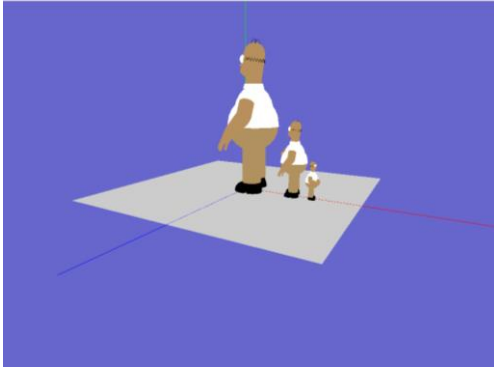
Classe 5: Càmera en tercera persona

- Càmera en tercera persona
- Moure càmera (mode inspecció)
- Càlcul de View Matrix amb càmera especificada amb angles d'Euler

Classe 5: Càmera en tercera persona

- **Càmera en tercera persona**
- Moure càmera (mode inspecció)
- Càlcul de View Matrix amb càmera especificada amb angles d'Euler

Càmera en 3a persona

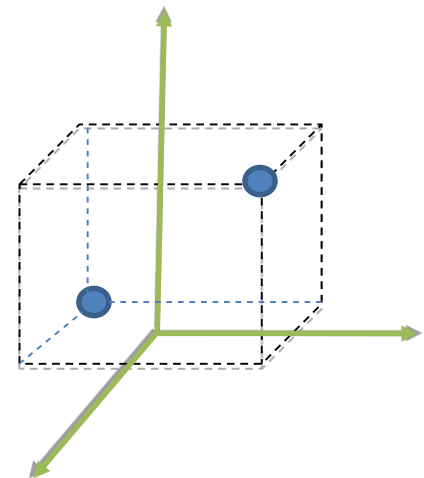


Visualització inicial de l'escena tal que:

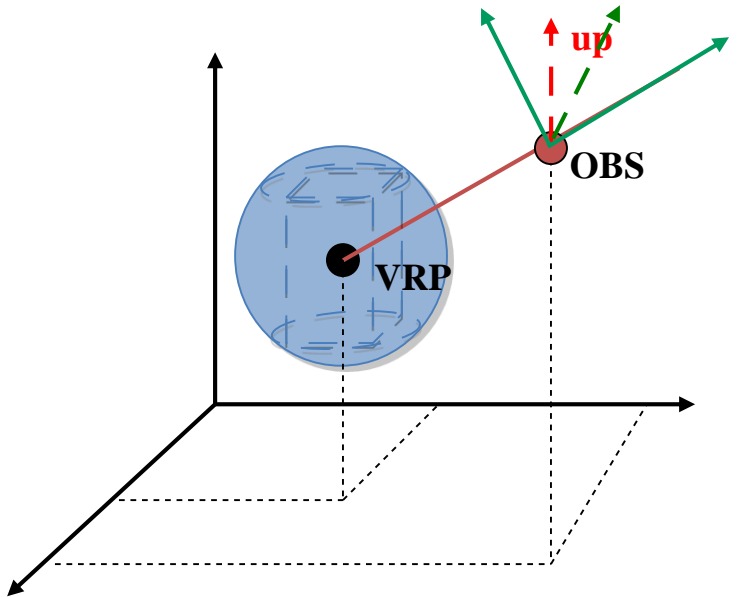
- inclogui tota l'escena (no retalli cap objecte)
- posició arbitrària de l'observador
- centrada en viewport
- optimitzant ocupació del viewport/vista
- sense deformació

Dada: capsa mínima contenidora de l'escena

$cmin=(xmin, ymin, zmin)$ i $cmax=(xmax, ymax, zmax)$

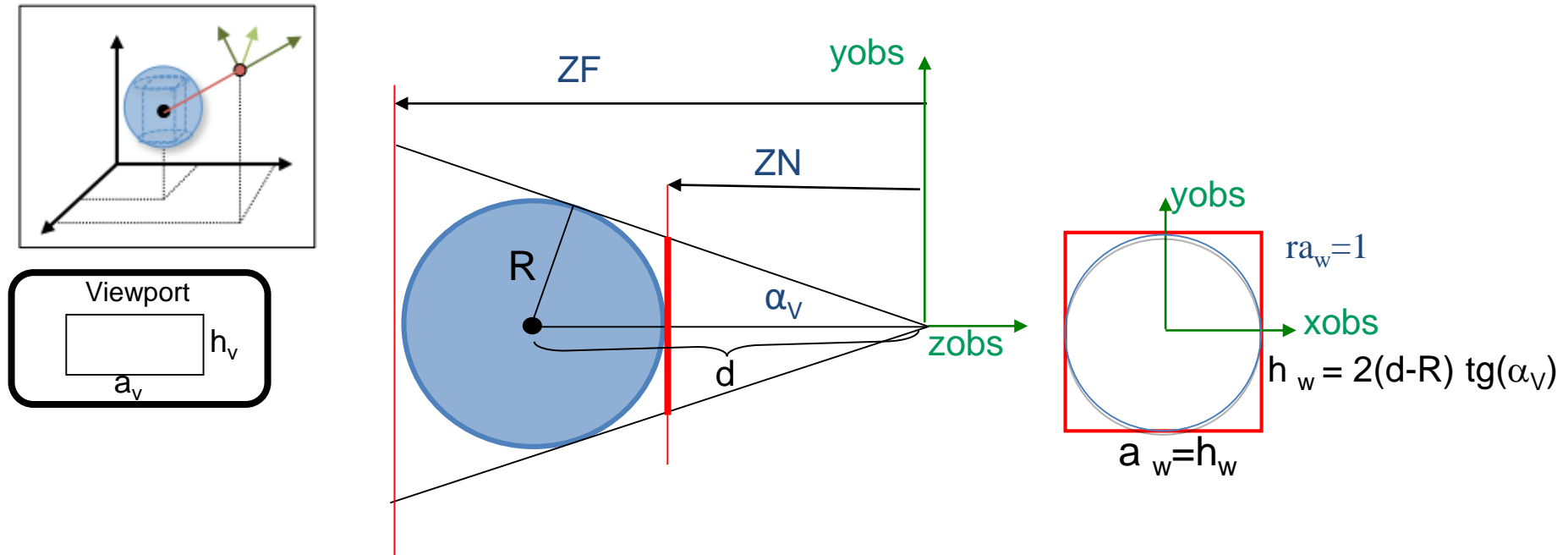


Càmera 3a persona (1): Inicialització posicionament amb OBS, VRP, up



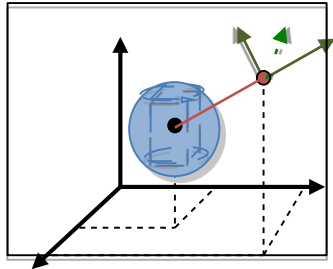
- Centrat => **VRP**=CentreEscena
- Per assegurar que l'escena es veu sense retallar des d'una posició arbitrària CAL que **OBS** sempre fora capsa mínima contenedora; per assegurar-ho CAL que **OBS** fora de l'esfera englobant de la capsa => distància "d" de l'**OBS** a **VRP** superior a R esfera.
 - CapsaMinCont=(xmin,ymin,zmin,xmax,ymax,zmax)
 - CentreEscena=Centre(CapsaMinCont) =
 $((x_{max}+x_{min})/2, (y_{max}+y_{min})/2, (z_{max}+z_{min})/2)$
 - $R = \text{dist}((x_{min}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{max}))/2$
 - $d > R$; per exemple $d = 2R$
 - **OBS**=**VRP**+ $d \cdot \mathbf{v}$; \mathbf{v} normalitzat en qualsevol direcció;
per exemple $\mathbf{v} = (1,1,1) / \|(1,1,1)\|$
- **up** qualsevol que no sigui paral·lel a \mathbf{v} ; si volem escena vertical (eix Y es vegi vertical) **up**=(0,1,0)

Càmera en 3a persona (2): tota l'escena, sense deformar i òptica perspectiva

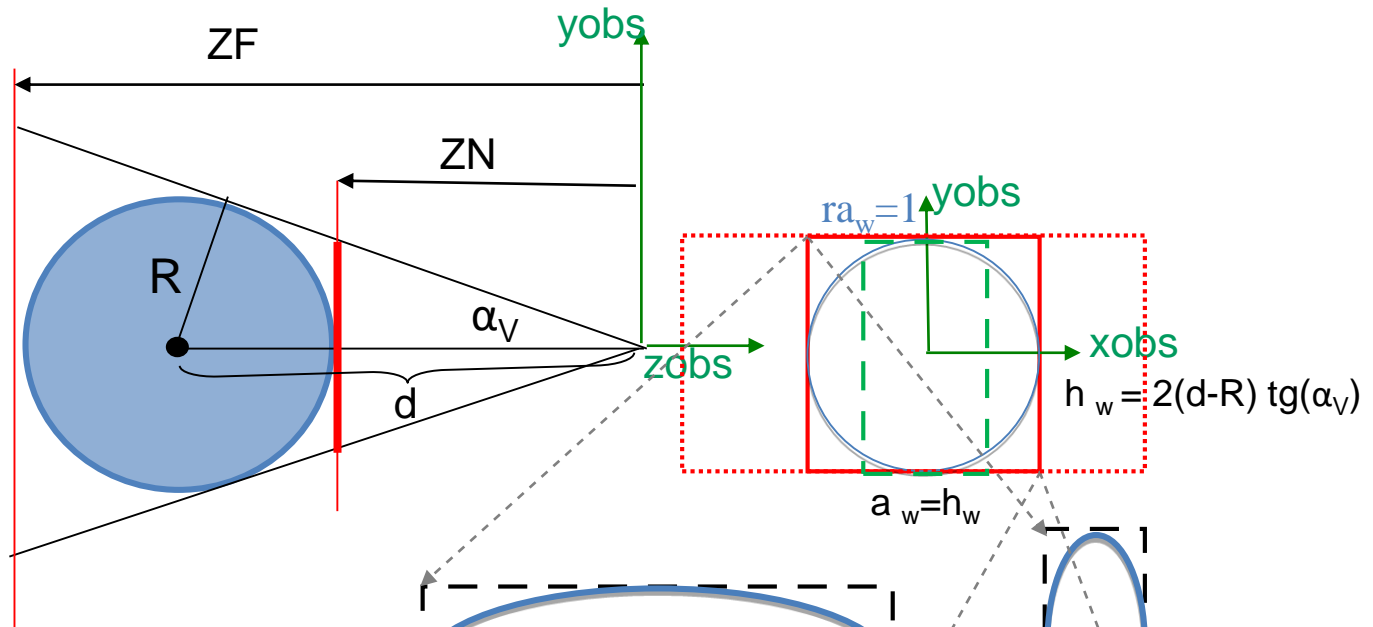


- Si tota l'esfera englobant està dins la profunditat del camp de visió, no retallem l'escena.
Per tant, $ZN \in]0, \underline{d-R}]$ $ZF \in [\underline{d+R}, \dots]$
 $ZN = d-R$; $ZF = d+R$ per aprofitar la precisió en profunditat
- Per a aprofitar al màxim la pantalla, el viewport, el window de la càmera s'ha d'ajustar per veure tota l'escena; una aproximació és ajustar el window per veure l'esfera englobant.
 - $R = d \sin(\alpha_v)$; **$\alpha_v = \text{arc sin}(R/d) \rightarrow \text{FOV} = 2 \cdot \alpha_v$**
 - com el window està situat en ZN , α_v determina que la seva alçada sigui:
 $h_w = 2(d-R) \text{tg}(\alpha_v)$
- **$ra_w = a_w/h_w = 1$** (α_H hauria de ser igual a α_v per assegurar que esfera no resulta retallada)

Càmera 3a persona (3): tota l'escena, sense deformar i òptica perspectiva



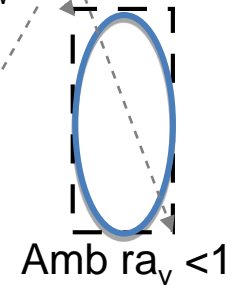
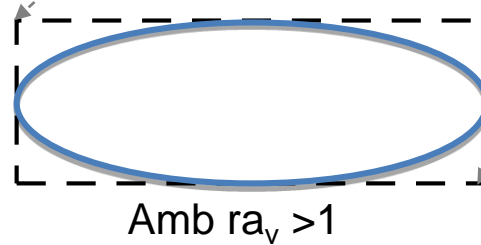
$$\begin{aligned} ZN &= d - R; \quad ZF = d + R \\ \alpha_v &= \arcsin(R/d) \\ \Rightarrow \text{FOV} &= 2 * \alpha_v \\ ra_w &= a_w / h_w = 1 \end{aligned}$$



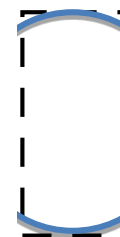
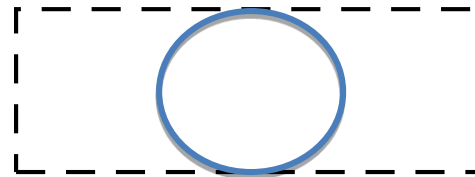
PERÒ cal tenir en compte la ra_v del viewport

Per a què **no hi hagi deformació**, cal modificar ra_w i forçar una

$$ra_w^* = ra_v$$

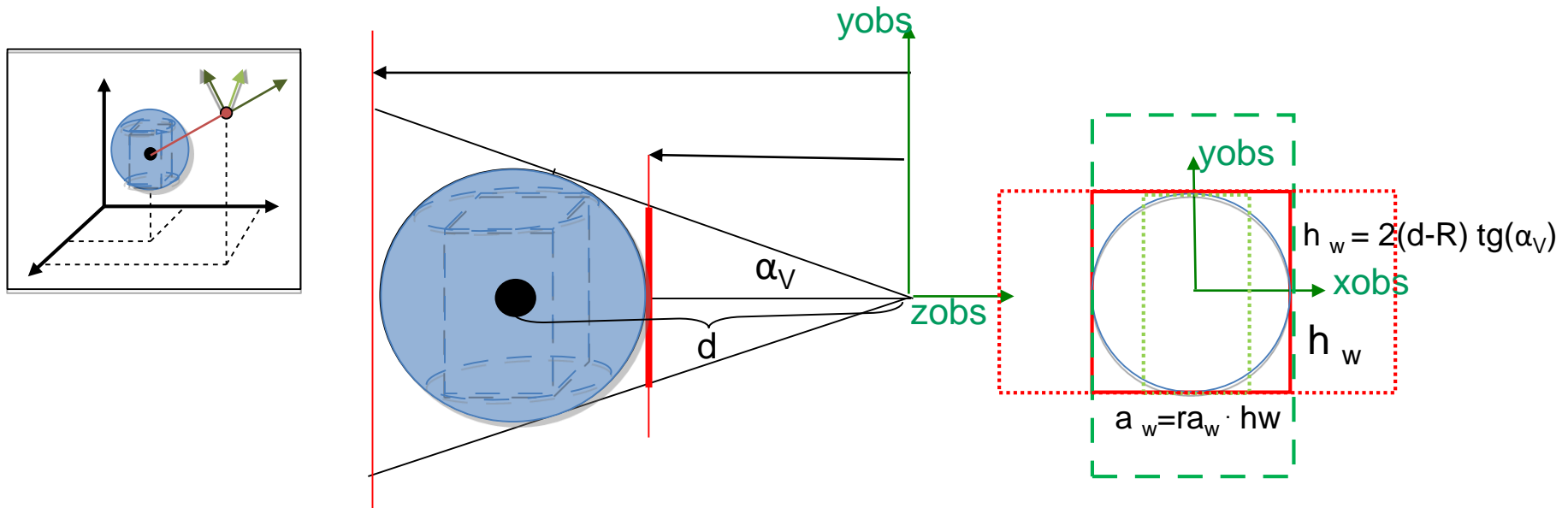


Viewport

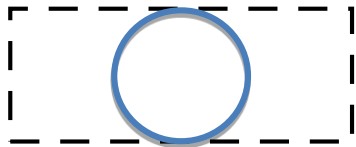


PERÒ en aquest cas retalla l'escena

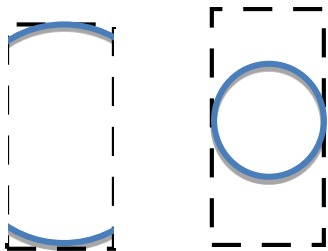
Càmera 3a persona (4): tota l'escena, sense deformar i òptica perspectiva



- Si $ra_v > 1$ ($>$ que la ra_w mínima requerida que és 1) \Rightarrow No es retalla l'escena al fer **$ra_w^* = ra_v$**
no cal modificar α_v (FOV)



- Si $ra_v < 1$ ($<$ que la ra_w mínima requerida que és 1) \Rightarrow al fer **$ra_w^* = ra_v$** es retalla escena

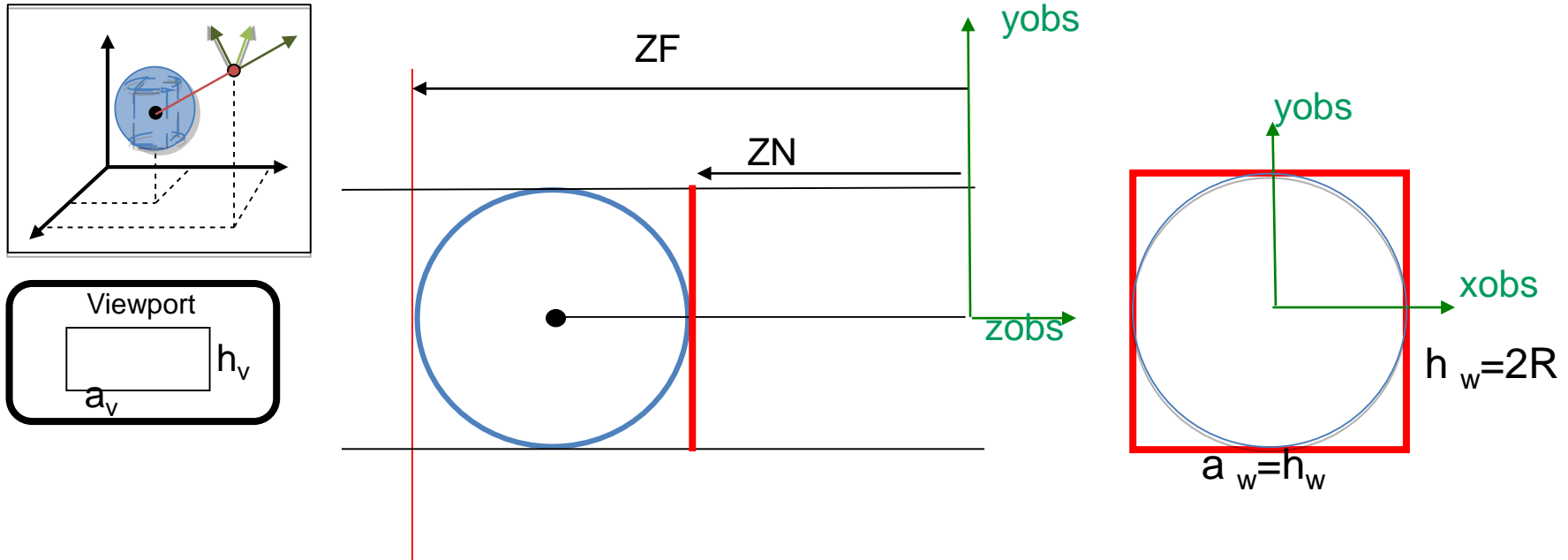


Cal incrementar l'angle d'obertura

FOV = $2 \alpha_v^*$ on

$$\alpha_v^* = \arctg(\tg(\alpha_v) / ra_v)$$

Càmera 3a persona (5): tota l'escena, sense deformar i òptica ortogonal



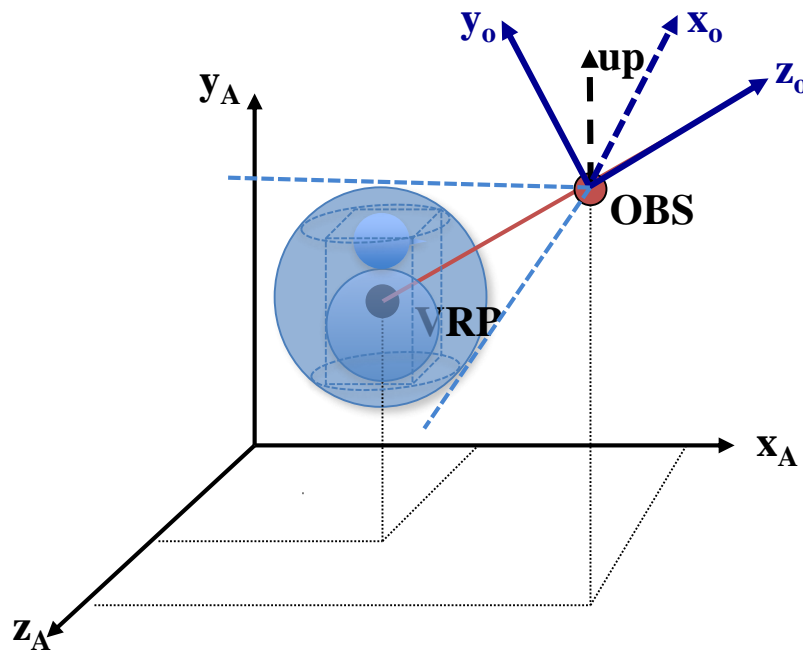
- **ZN i ZF** mateix raonament que en càmera perspectiva.
- **Window mínim requerit (centrat) = $(-R, R, -R, R)$ \Rightarrow una $ra_w = 1$ (per què ?)**
- Si $ra_w \neq ra_v \Rightarrow$ deformació (per què?)
 - Si $ra_v > 1 \Rightarrow$ cal incrementar la $ra_w \Rightarrow$ *modificar window*
 com $ra_w = a_w / h_w \Rightarrow$ podem incrementar a_w o decrementar h_w (és retallaria esfera!!)
 Per tant, modifiquem a_w :
 $a_w^* = ra_v * h_w = ra_v * 2 * R$
window = $(-R ra_v, R ra_v, -R, R)$
- Raonament similar per recalculer window quan $ra_v < 1$

Classe 5: Càmera en tercera persona

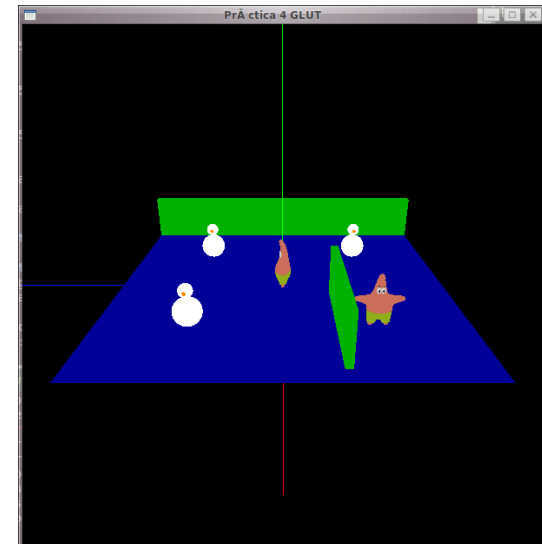
- Càmera en tercera persona
- **Moure càmera (mode inspecció)**
- Càlcul de View Matrix amb càmera especificada amb angles d'Euler

Vist...

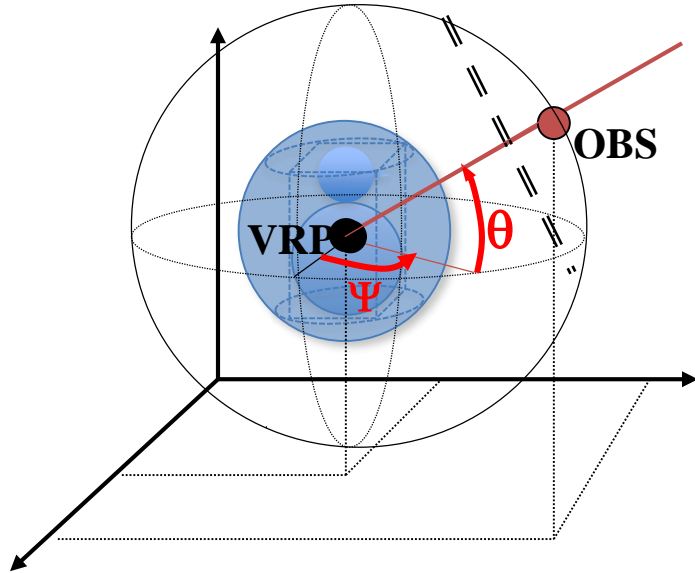
- Posicionament: OBS, VRP, up \rightarrow viewMatrix
- Òptica perspectiva: zN, zF, FOV, ra \rightarrow projectMatrix
- Càmera en 3a persona: posició inicial



**Com Moure la Càmera
per inspeccionar escena?**



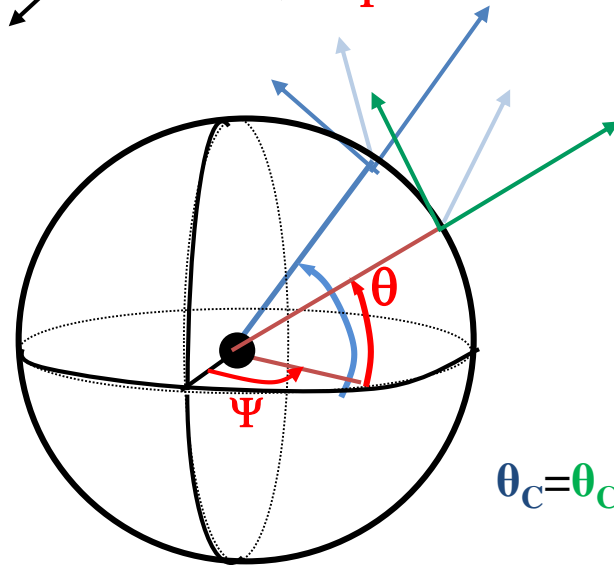
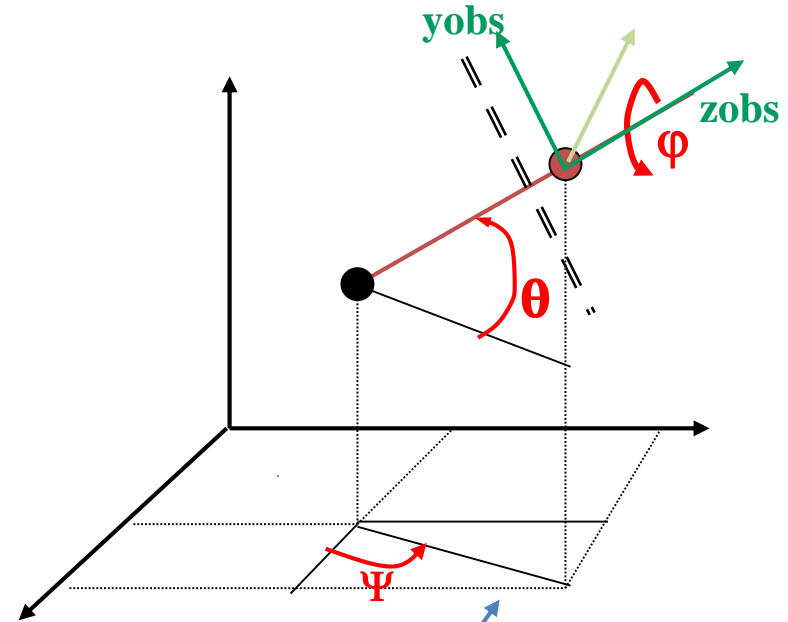
Moure la Càmera per inspeccionar l'escena



- Els angles (d'Euler) determinen la posició d'un punt en l'esfera
- Des de la interfície d'usuari desplacem el cursor dreta/esquerra (Ψ) i pujar/baixar (θ); per moure **OBS** sobre l'esfera
- No cal canviar l'òptica

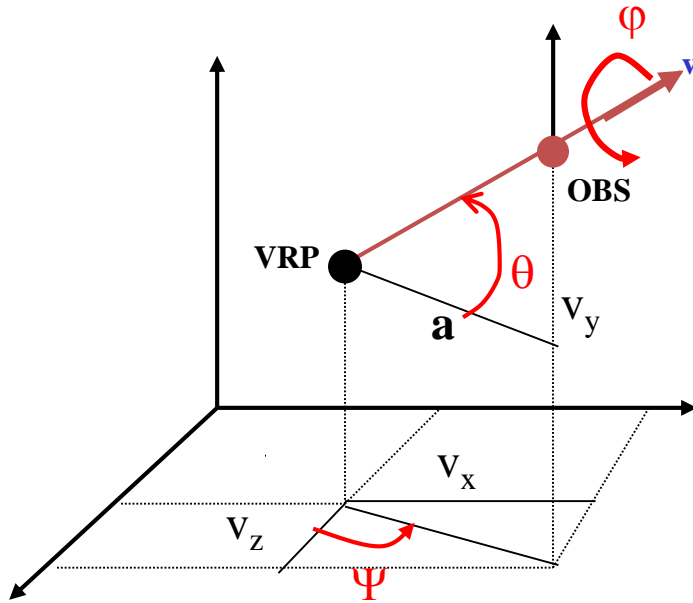
Com calculem **OBS**, **VRP**, up?

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



$$\theta_c = \theta_c + f(\Delta y)$$

Càlcul VRP, OBS a partir dels angles d'Euler



VRP = Punt d'enfoc

OBS = **VRP** + $d \mathbf{v}$

$d > R$; per exemple: $d = 2R$

$v_y = \sin(\theta)$; $a = \cos(\theta)$;

$v_z = \cos(\theta) \cos(\psi)$;

$v_x = \cos(\theta) \sin(\psi)$;

Un possible **up**: **up** = (0,1,0) ($\phi = 0^\circ$)

Noteu que l'òptica no cal modificar-la perquè ens movem sobre esfera que envolta l'escena de radi d .

Es podria calcular la ViewMatrix directament a partir dels angles?

Noteu que estem considerant els angles d'orientació de la càmera:

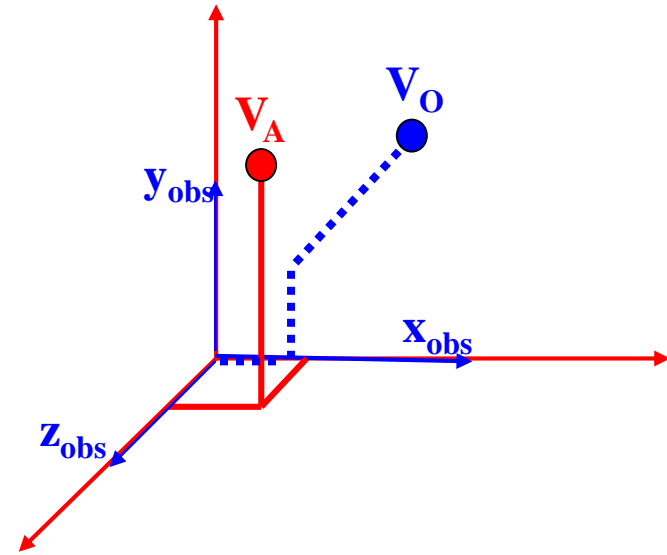
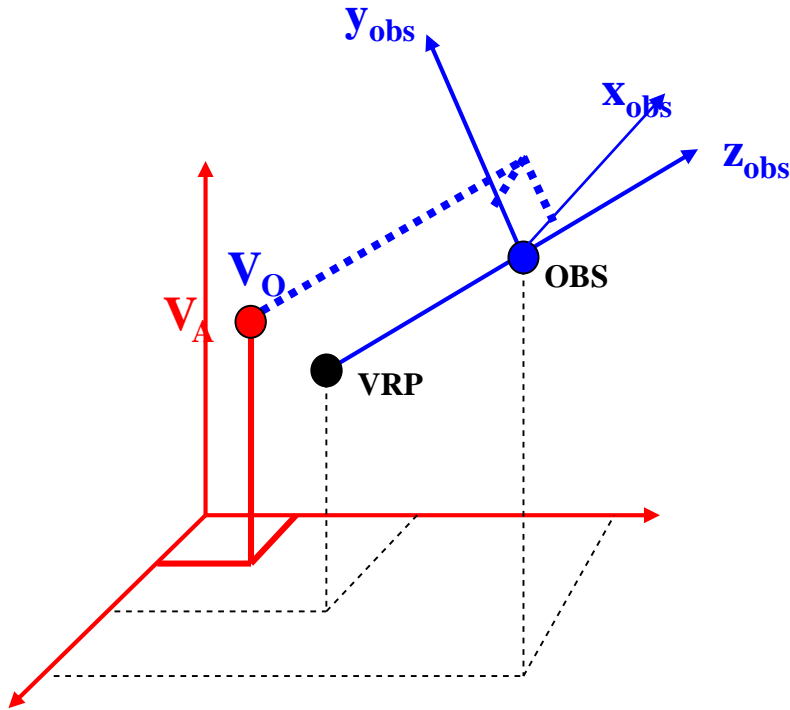
ψ en $[-180, 180]$, θ en $[-90, 90]$

positius quan movem la càmera cap \rightarrow i quan la movem cap \uparrow

Classe 5: Càmera en tercera persona

- Càmera en tercera persona
- Moure càmera (mode inspecció)
- **Càlcul de View Matrix amb càmera especificada amb angles d'Euler**

Càlcul viewMatrix directe a partir d'angles Euler, VRP i d (1)



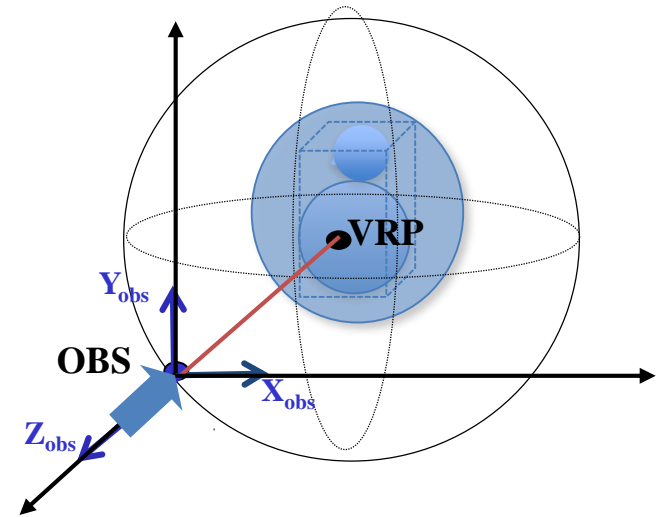
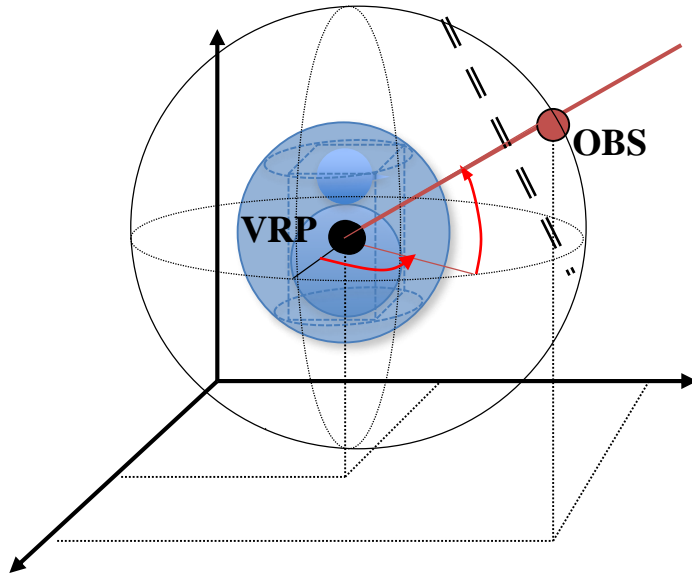
RECORDEU: La viewMatrix serveix per tenir la posició dels vèrtexs respecte del SCO

$$V_o = VM * V_A$$

Es pot calcular la VM:

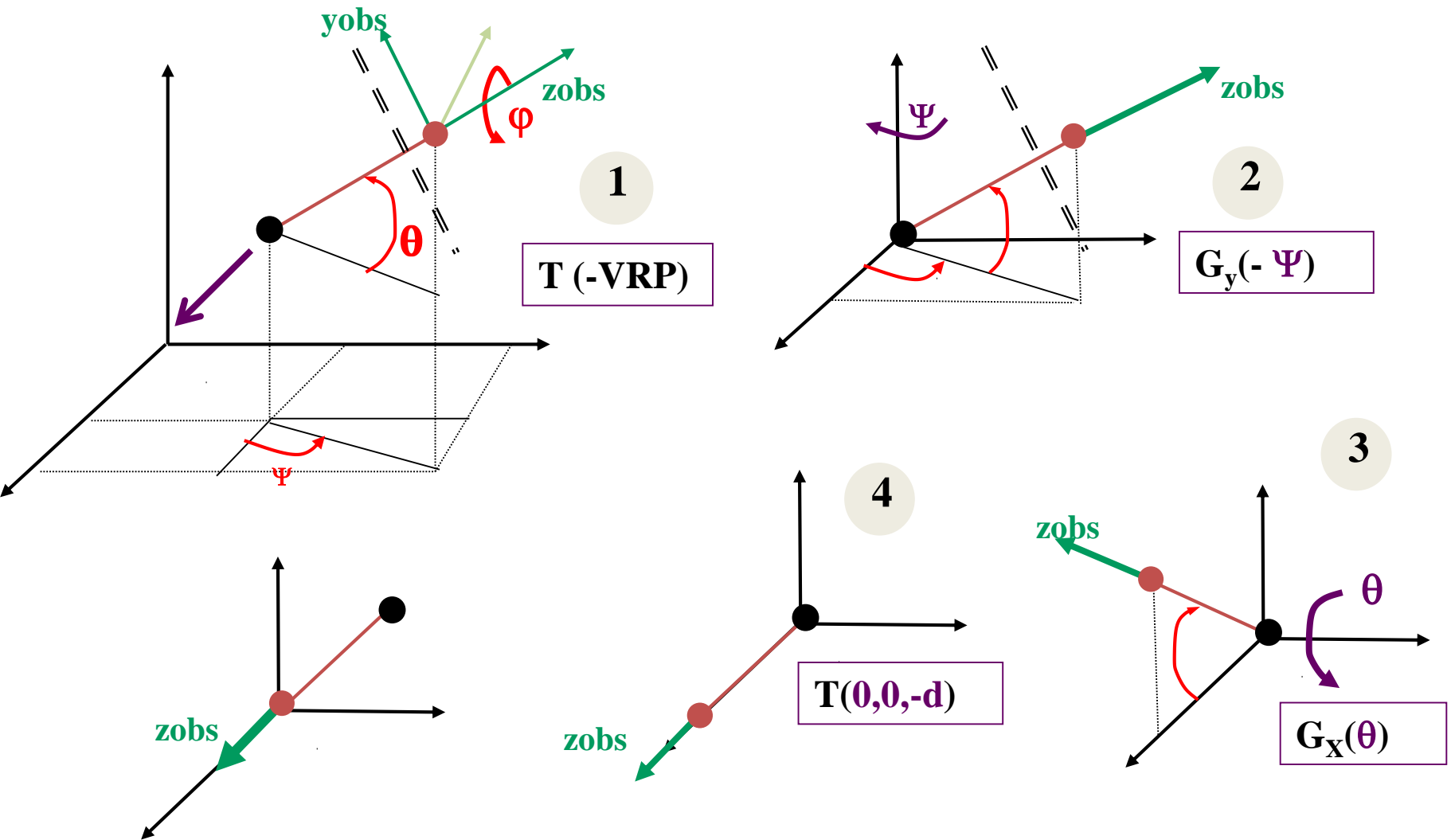
- Pensant que movem la càmera (OBS, VRP, Up)
- Pensant que tenim una càmera fixa al SCA i ubiquem tota l'escena respecte d'ella → realitzar una mateixa TG_{VM} a tots els objectes. Si vèrtexs queden respecte a la càmera en la mateixa posició → TG_{VM} serà igual a la VM calculada amb OBS, VRP, Up

Càlcul VM directe a partir d'angles Euler, VRP i d (2)



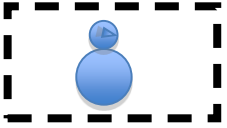
- Ho podeu pensar com si girem l'esfera per a què la seva posició respecte la càmera per defecte sigui la mateixa. Agafar l'esfera i posicionar-la.
- Noteu que z_{obs} passarà a ser coincident amb z_A (SCO i SCA coincidiran)
- Pensarem el moviment tenint en compte que sabem calcular matrius de gir només si girem entorn d'eixos que passen per origen de coordenades.

Càlcul VM directe a partir d'angles Euler, VRP i d (3)

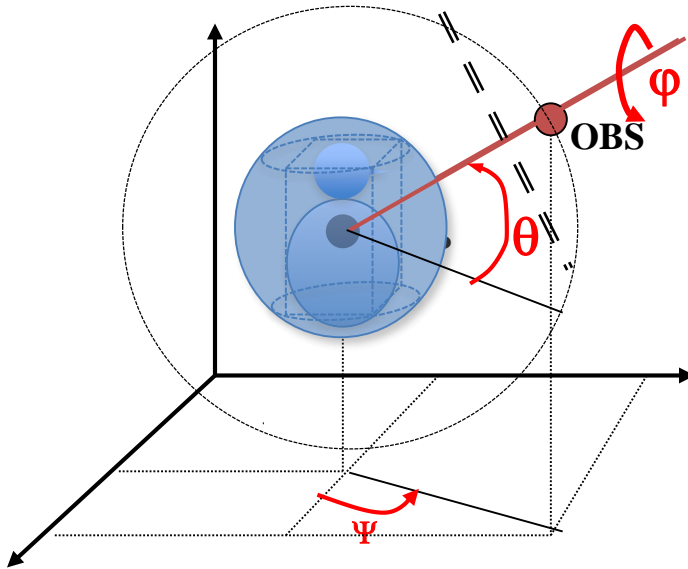


$$VM = T(0,0,-d) * G_x(\theta) * G_y(-\Psi) * T(-VRP)$$

Exemple: Posicionament amb angles Euler (4)



$$VM = T(0,0,-d) * G_z(-\phi) * G_x(\theta) * G_y(-\psi) * T(-VRP)$$

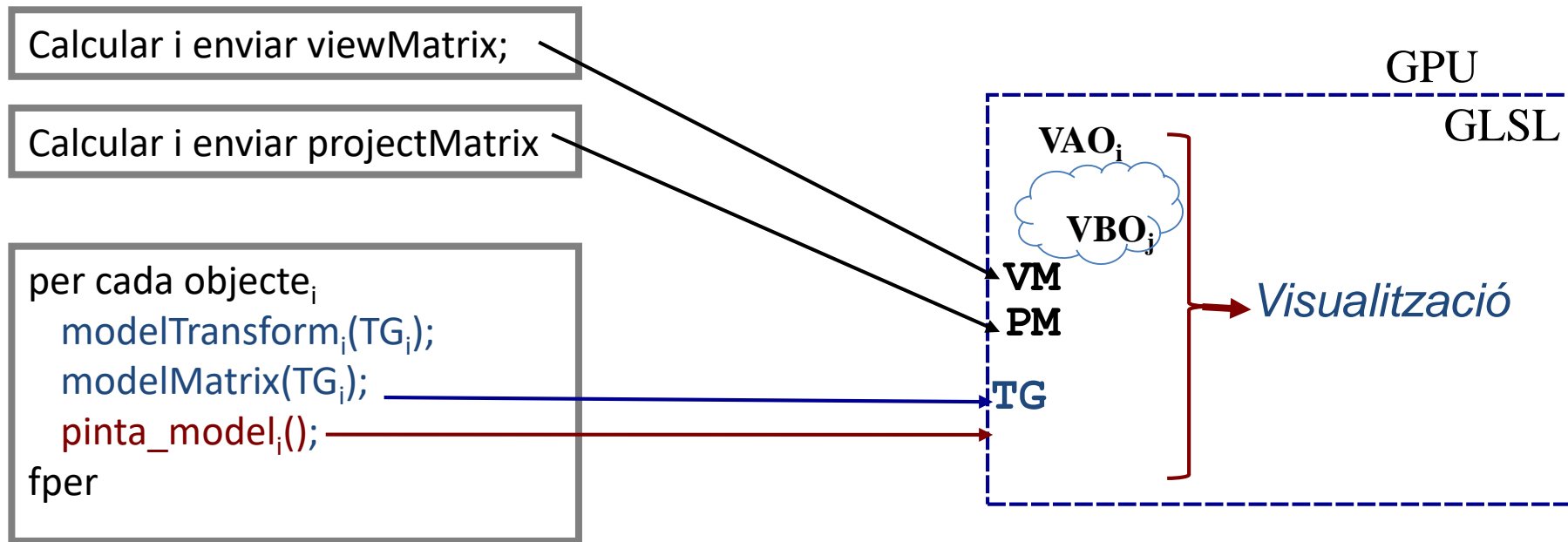


```
VM=Translate (0.,0.,-d)
VM=VM*Rotate(-\phi,0,0,1)
VM= VM*Rotate (\theta,1,0,0.)
VM= VM*Rotate(-\psi,0,1,0.)
VM= VM*Translate(-VRP.x,-VRP.y,-VRP.z)
viewMatrix(VM)
```

Compta amb signes:

- Si s'ha calculat ψ positiu quan càmera gira cap a la dreta, serà un gir anti-horari respecte eix Y de la càmera, per tant, matemàticament positiu; com girem els objectes en sentit contrari, cal posar $-\psi$ en el codi.
- Si s'ha calculat θ positiu quan pugem la càmera, serà un gir horari; per tant, matemàticament un gir negatiu; com objecte girarà en sentit contrari (anti-horari), ja és correcte deixar signe positiu.

Resum final: com visualitzar l'escena?



```
#version 330 core
```

*Vertex
Shader*

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

```
#version 330 core
```

*Fragment
Shader*

```
out vec4 FragColor;  
  
void main() {  
    FragColor = vec4(0, 0, 0, 1);  
}
```

Classe 3: Conceptes i preguntes

- Capsa i esfera contenidores de l'escena. Són les d'un model?
- Càlcul dels paràmetres d'una càmera en tercera persona.
- Què cal complir per a què no hi hagi deformació? I per poder veure sempre tota l'escena.
- Re-càlcul dels paràmetres de l'òptica en fer un “resize”. En quina part del codi els recalcularies?
- Metàfora d'inspecció d'una escena movent el cursor.
- Angles d'Euler.
- Càlcul de la viewMatrix (VM) concatenant transformacions geomètriques.
- Pseudocodi per crear la matriu anterior. Importa l'ordre de les transformacions?
- Inicialització d'angles per veure l'escena en planta i pseudo-codi de càlcul de la VM.
- Quines matrius cal tornar a enviar a la GPU si es fa un “resize”?
- Si per teclat indiquem que volem moure un objecte fent una translació de (5,0,0), què cal recalcular? Quina informació cal modificar de la que s'envia a la GPU?
- Exercicis proposats per donats uns OBS, VRP i up, calcular la viewMatrix amb transformacions geomètriques (i viceversa).