

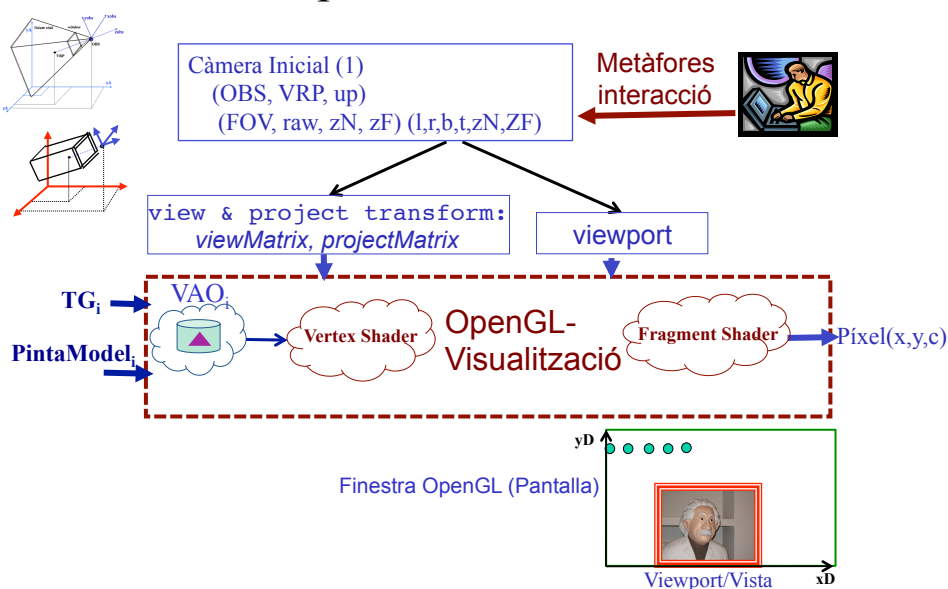
## Classe 4: contingut

- Càmera (2):
  - Recordatori
  - Exercici: càmera en 3ra persona
  - Moure Càmera i Angles Euler
- Alguns exercicis càmera

IDI 2017-2018 2Q

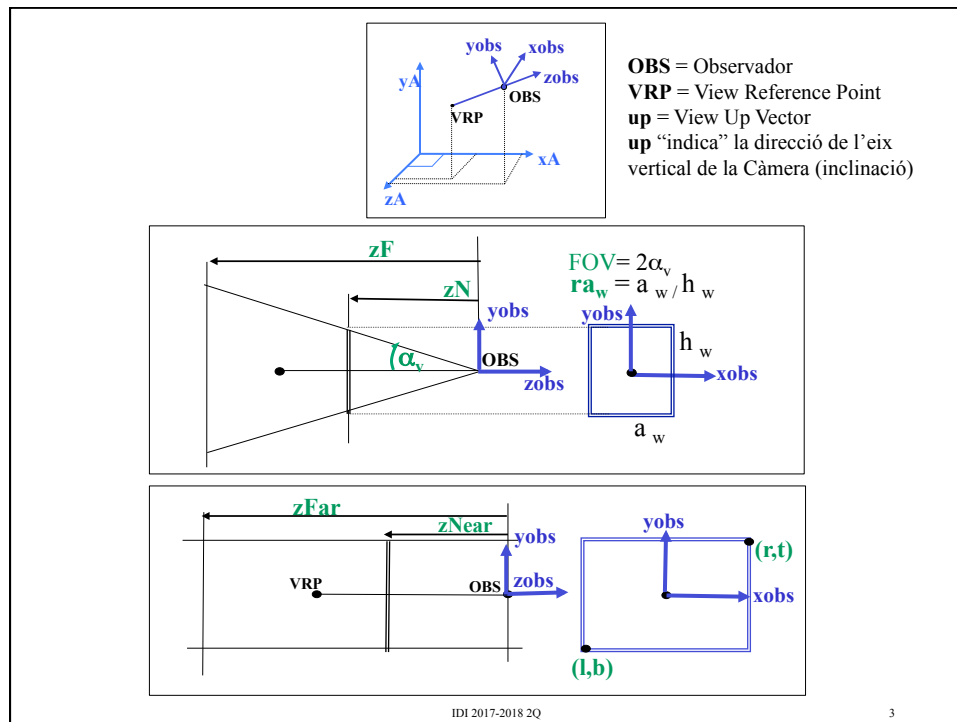
1

## Càmera i procés de visualització

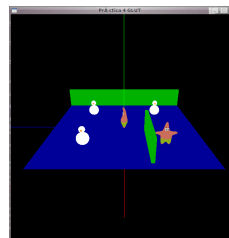
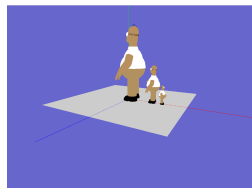


IDI 2017-2018 2Q

2



## Càmera 3ra persona



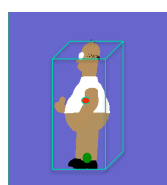
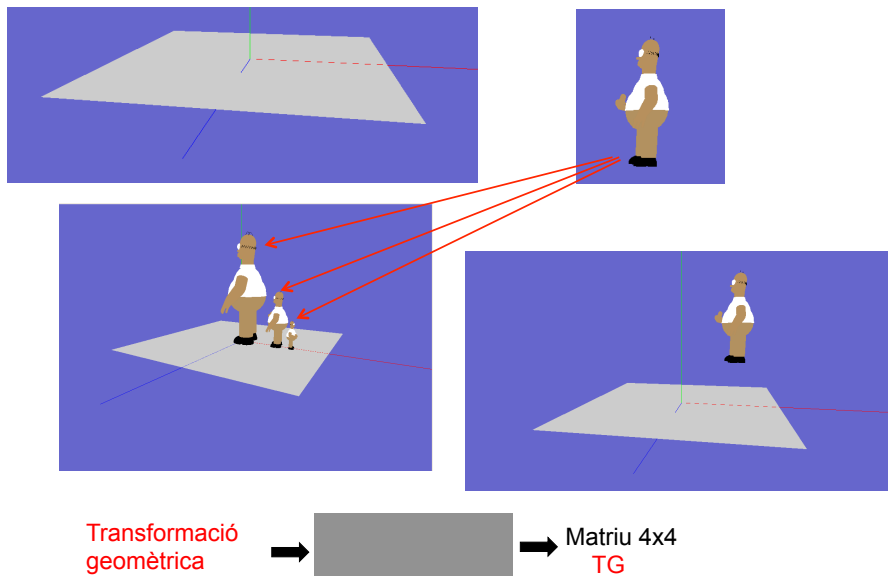
Quins paràmetres de posició, orientació i òptica per càmera en 3ra persona? →

- imatge inclogui tota l'escena (no retalli)
- centrada en viewport
- optimitzant ocupació del viewport
- sense deformació

Dada: caps mínima contenidora d'escena

(xmin, ymin, zmin) - (xmax, ymax, zmax)

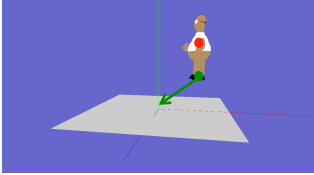
## Recordatori de com generem escena



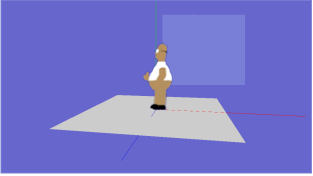
$CapsaMinCont = (xmin, ymin, zmin, xmax, ymax, zmax)$

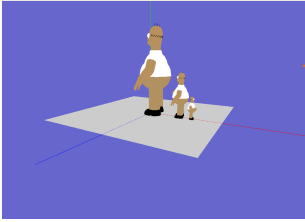
Mides  $\Rightarrow a = (xmax - xmin)$ ,  $h = (ymax - ymin)$ ,  $f = (zmax - zmin)$

$CentBaseCapsa = (cbx, cby, cbz) = (xmin + xmax)/2, ymin, (zmin + zmax)/2)$

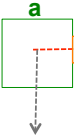


$TG_{H1} = \text{Trans}(t)$   
 $t = (-cbx, -cby, -cbz)$





$TG_{H2} = \text{Trans}(3a/4, 0, 0) S(1/2, 1/2, 1/2) \text{Trans}(t)$



$TG_{H3} = \text{Trans}(9a/8, 0, 0) S(1/4, 1/4, 1/4) R_y(-180) \text{Trans}(t)$   
 $TG_{H3} = \text{Trans}(9a/8, 0, 0) R_y(-180) S(1/4, 1/4, 1/4) \text{Trans}(t)$

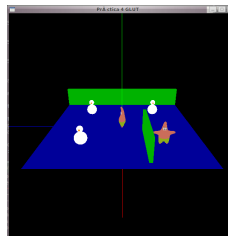
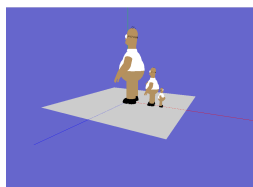
## Visualització OpenGL

per cada objecte,  
 //Càlcul  $TG_i$  i enviar a OpenGL  
 modelTransform( $i$ )  
 //pinta\_homer() o pinta\_terra()  
 pinta\_model( $i$ );  
 fper

$$TG_{H3} = \overset{4}{\text{Trans}(9a/8, 0, 0)} \overset{3}{S(1/4, 1/4, 1/4)} \overset{2}{R_y(-180^\circ)} \overset{1}{\text{Trans}(t)}$$

```
modelTransform_homer3(){
  TG=I;
  TG= TG*Translate(posx,posy,posz);
  TG= TG*Scale(s,s,s);
  TG= TG*Rotate (-180., (0.,1.,0.));
  TG= TG*Translate (-cb.x,-cb.y,-cb.z);
  modelMatrix(TG); //enviar uniform
}
```

## Càmera 3ra persona



Quins paràmetres de posició, orientació i òptica per càmera en 3ra persona? →

- imatge inclogui tota l'escena (no retalli)
- centrada en viewport
- optimitzant ocupació del viewport
- sense deformació

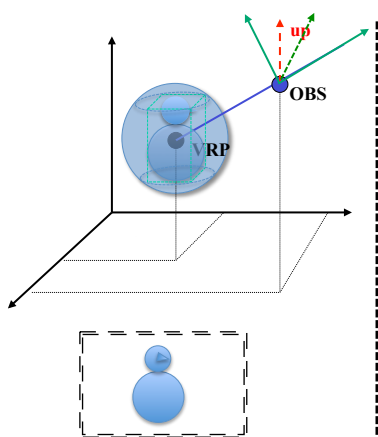
Dada: capsula mínima contenidora d'escena

(xmin, ymin, zmin) - (xmax, ymax, zmax)

IDI 2017-2018 2Q

9

### 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 capsula mínima contenidora; per assegurar-ho CAL que **OBS** fora de l'esfera englobant de la capsula => distància "d" de l'**OBS** a **VRP** superior a R esfera.
  - CapsaMinCont=(xmin,ymin,zmin,xmax,ymax,zmax)
  - CentreEscena=Centre(CapsaMinCont) =  
 $((xmax+xmin)/2, (ymax+ymin)/2, (zmax+zmin)/2)$
  - $R = \text{dist}((xmin,ymin,zmin), (xmax,ymax,zmax))/2$
  - $d > R$ ; 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 ninot vertical (eix Y es vegi vertical) **up**=(0,1,0)

IDI 2017-2018 2Q

10

### Tota escena en la vista, sense deformar i càmera perspectiva

Diagram illustrating the projection of a sphere of radius  $R$  at distance  $d$  from a camera. The camera's field of view is defined by angle  $\alpha_v$ . The scene is projected onto a viewport of width  $a_v$  and height  $h_v$ . The resulting image on the screen has width  $a_w$  and height  $h_w$ . The diagram includes labels for  $ZF$  (far plane),  $ZN$  (near plane),  $y_{obs}$ ,  $x_{obs}$ ,  $z_{obs}$ , and the viewport dimensions.

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

- **PERÒ** per a què no hi hagi deformació, cal que  $ra_w = ra_v$ , per tant, si no volem modificar el viewport cal forçar una  $ra_w^* = ra_v$

**Pregunta:** amb aquesta nova  $ra_w^*$  es retallarà l'esfera? (està tota l'esfera/escena dins del volum de visió?)

IDI 2017-2018 2Q

11

### Tota escena en la vista, sense deformar i càmera perspectiva

Diagram illustrating the projection of a sphere of radius  $R$  at distance  $d$  from a camera. The camera's field of view is defined by angle  $\alpha_v$ . The scene is projected onto a viewport of width  $a_v$  and height  $h_v$ . The resulting image on the screen has width  $a_w$  and height  $h_w$ . The diagram includes labels for  $y_{obs}$ ,  $x_{obs}$ ,  $z_{obs}$ , and the viewport dimensions.

Cas amb Viewport  $ra_v > 1$

Si només  $ra_w = ra_v \Rightarrow$  no modifiquem  $h_w$ , simplement es força una nova  $a_w^* > a_w$  mínima requerida, per tant, no es retalla  
no cal modificar  $\alpha_v$  (FOV)

Amb  $ra_w = 1$

Amb  $ra_w = ra_v$

IDI 2017-2018 2Q

12

### Tota escena en la vista, sense deformar i càmera perspectiva

Viewport:  $a_v$ ,  $h_v$ ,  $ra_v > 1$

$y_{obs}$ ,  $z_{obs}$ ,  $\alpha_v$ ,  $d$

$ra_w = 1$ ,  $h_w = 2(d-R) \tan(\alpha_v)$ ,  $a_w = ra_w \cdot h_w$

Cas amb Viewport  $ra_v < 1$

Si només  $ra_w = ra_v$  no toquem  $h_w \Rightarrow$  reduim  $a_w$

Amb  $ra_w = 1$

retallarà esfera!!!!

Per evitar-ho cal incrementar l'angle d'obertura, per incrementar proporcionalment l'amplada i englobar tota l'esfera. CAL  $ra_w^* = ra_v$  i nou FOV

$FOV = 2 \alpha_v^*$  on  $\alpha_v^* = \arctg(\tan(\alpha_v) / ra_v)$

• Sempre cal calcular el nou angle a partir de l'inicial (window quadrat).

13

### Tota escena en la vista, sense deformar i càmera perspectiva

Viewport:  $a_v$ ,  $h_v$

$y_{obs}$ ,  $z_{obs}$ ,  $\alpha_v$ ,  $d$

$h_w = 2(d-R) \tan(\alpha_v)$ ,  $a_w = ra_w \cdot h_w$

- Si  $ra_v > 1$  ( $> ra_w$  mínima requerida 1)  $\Rightarrow$  No es retalla, no cal modificar  $\alpha_v$  (FOV), només fer  $ra_w^* = ra_v$   
Justificació:  $ra_w^*$  serà superior a 1; si no modifiquem l'angle FOV,  $h_w$  no canvia  $\Rightarrow$   
 $a_w^* = ra_w^* \cdot h_w$  i com  $ra_w^* > ra_w \Rightarrow a_w^* > a_w$  i, per tant, serà més gran del necessari però es veurà tota l'esfera i quedarà espai pels laterals.
- Si  $ra_v < 1$  ( $< ra_w$  mínima requerida 1)  $\Rightarrow$  cal fer  $ra_w^* = ra_v$  i incrementar l'angle d'obertura  
 $FOV = 2 \alpha_v^*$  on  $\alpha_v^* = \arctg(\tan(\alpha_v) / ra_v)$

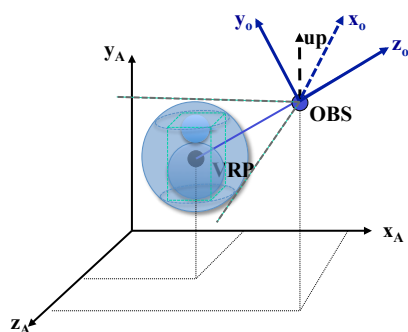
Justificació: com  $a_w^* = ra_w^* \cdot h_w$ , si no modifiquem angle,  $h_w$  no varia; com  $ra_w^* < ra_w \Rightarrow a_w^* < a_w$  i l'esfera quedaria retallada (en horitzontal). Per tant, cal incrementar l'angle  $\alpha_v$  (i, per tant,  $h_w^*$ ) per a garantir una amplada del window igual a la mínima requerida.

- Com  $h_w^* = a_w / ra_v$  i per trigonometria  $h_w^* = 2(d-R) \tan(\alpha_v^*)$ , igualant les equacions  
 $\alpha_v^* = \arctg(\tan(\alpha_v) / ra_v)$

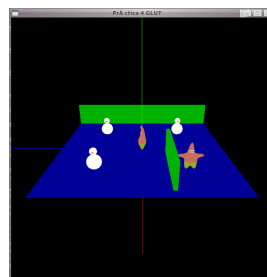
14

## Vist...

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



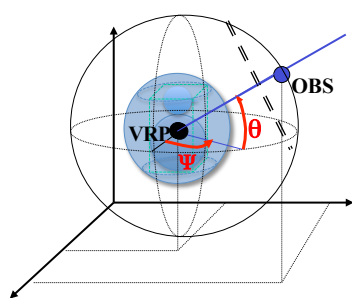
Com Moure la Càmera  
per inspeccionar escena?



IDI 2017-2018 2Q

15

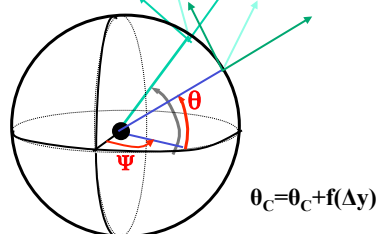
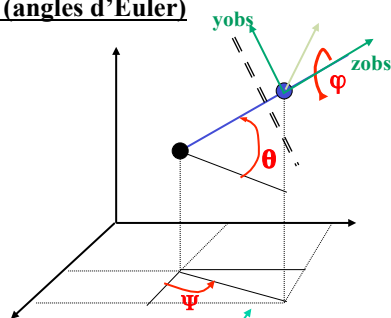
## Moure la Càmera (angles d'Euler)



- Els angles (d'Euler) determinen la posició d'un punt en l'esfera
- Des de la interfície d'usuari desplacem el cursor dreta/esquerra ( $\psi$ ) i pujar/baixar ( $\theta$ ); per moure OBS sobre l'esfera

Com calculem OBS, VRP, up?

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



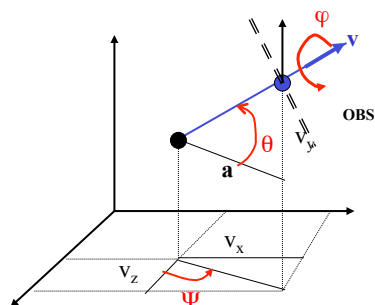
$$\theta_c = \theta_c + f(\Delta y)$$

IDI 2017-2018 2Q

16



### Càlcul VRP, OBS a partir dels angles d'Euler



**VRP** = Punt d'enfoc

**OBS** = VRP + d **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) ( $\varphi = 0^\circ$ )

Es podria calcular la View Matrix directament a partir dels angles?

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

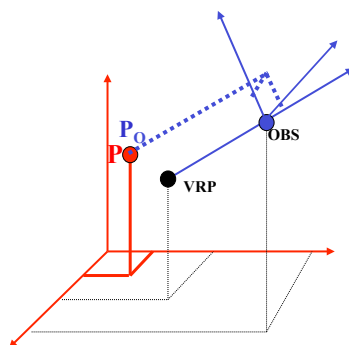
$\Psi$  en  $[-180, 180]$ ,  $\theta$  en  $[-90, 90]$

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

17

IDI 2017-2018 2Q

### Càlcul view Matrix directe a partir d'angles Euler, VRP i d



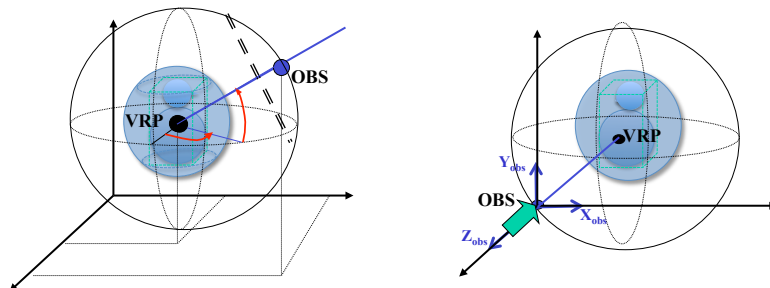
**RECORDEU:**

La viewMatrix serveix per tenir posició de punts respecte observador

18

IDI 2017-2018 2Q

### Càlcul VM directe a partir d'angles Euler, VRP i d

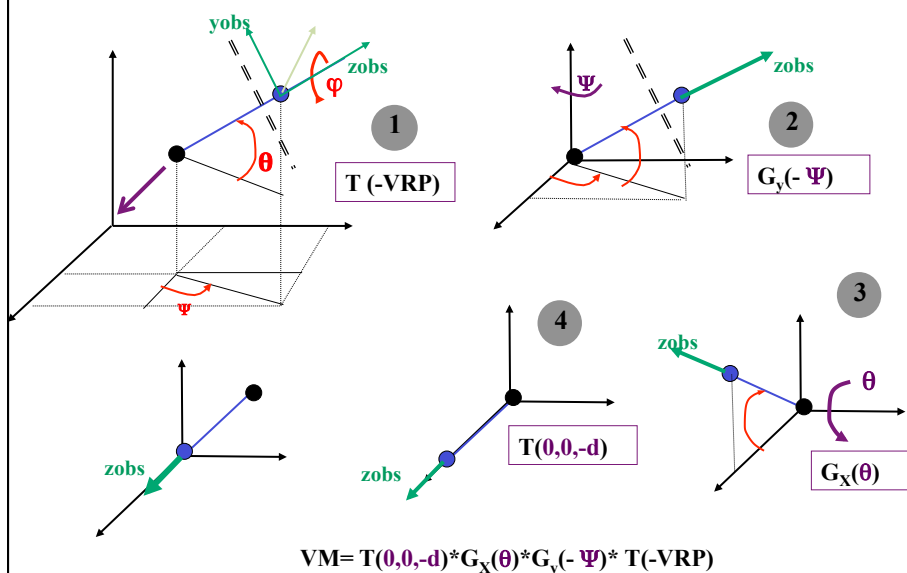


- Ho podeu pensar com si girem l'esfera per a què la seva posició respecte la càmera de defecte sigui la mateixa. Agafar l'esfera i posicionar-la.
- Noteu que zobs passarà a ser coincident amb zA (SCO i SCA coincidiràn)
- **Pensarem el moviment tenint en compte que sabem calcular matrius de gir només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

19

IDI 2017-2018 2Q

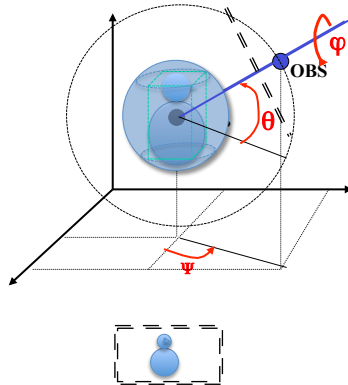
### Càlcul MV directe a partir d'angles Euler: exemple més complex



IDI 2017-2018 2Q

20

### Exercici d'inicialització càmera: Posicionament amb angles Euler (TG)



$$VM = T(0,0,-d) * G_Z(-\varphi) * G_X(\theta) * G_Y(-\psi) * T(-VRP)$$

```
VM=Translate (0,0,-d)
VM=VM*Rotate(-φ,0,0,1)
VM= VM*Rotate (θ,1,0,0.)
VM= VM*Rotate(-ψ,0.,1,0.)
VM= VM*Translate(-VRP.x,-VRP.y,-VRP.z)
viewMatrix(VM)
```

Compta amb signes:

- Si s'ha calculat  $\psi$  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  $\theta$  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.

IDI 2017-2018 2Q

21

```
/* CreateBuffers(); Crear VAO del model
(un cop)*/
...
/*IniCamera() calcular paràmetres càmera i
matrius cada cop que es
modifiquin */
/*viewTransform() amb lookAt o
Transformacions Euler*/
VM = lookAt(OBS,VRP,UP);
viewMatrix(VM);
//projectTransform()
PM=perspective (FOV,ra,zN,ZF);
projectMatrix(PM);
//resize(...)
glViewport (0,0,w,h);
/*PaintGL(); cada cop que es requereix
refresc*/
/*per cada model: modelTransform()
Calcula TG, i passar a OpenGL*/
modelTransform_i(TG);
modelMatrix(TG);
Pinta_model(VAO);
```

### Recordatori final de visualització

#### Vertex Shader

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

IDI 2017-2018 2Q

22