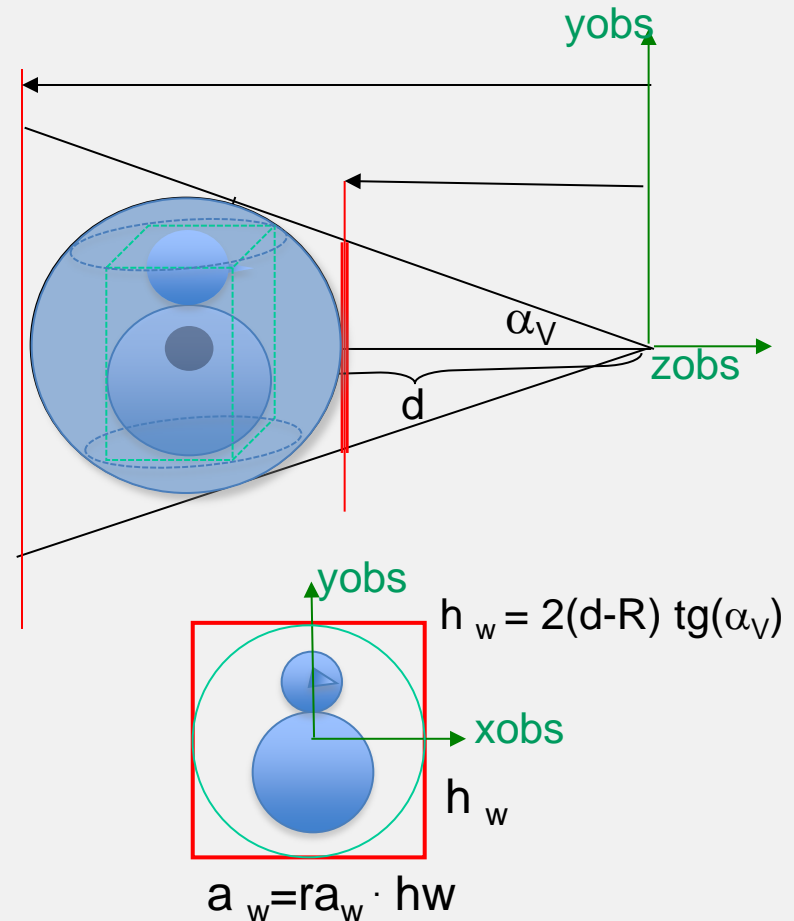
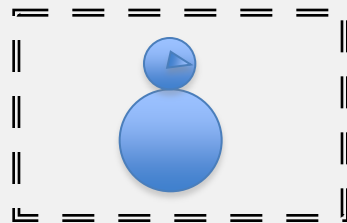
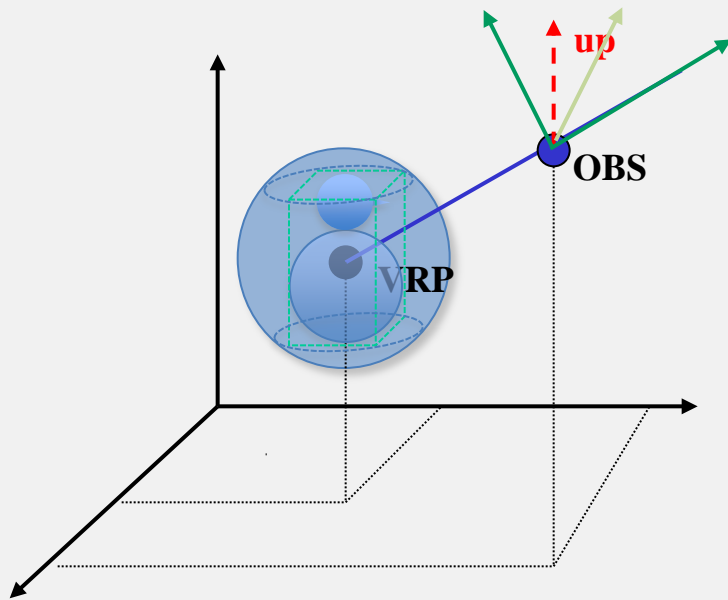


Laboratori OpenGL – Sessió 2.2

- Càlcul càmera per a visualitzar escena (càmera 3^a persona)
- Redimensionat finestra sense deformació ni retallat (resize)
- Visualitzar objecte qualsevol
- View Matrix amb angles d'Euler
- Interacció per inspecció (amb angles d'Euler)

Càmera en 3^a persona (exercicis 1 i 2)

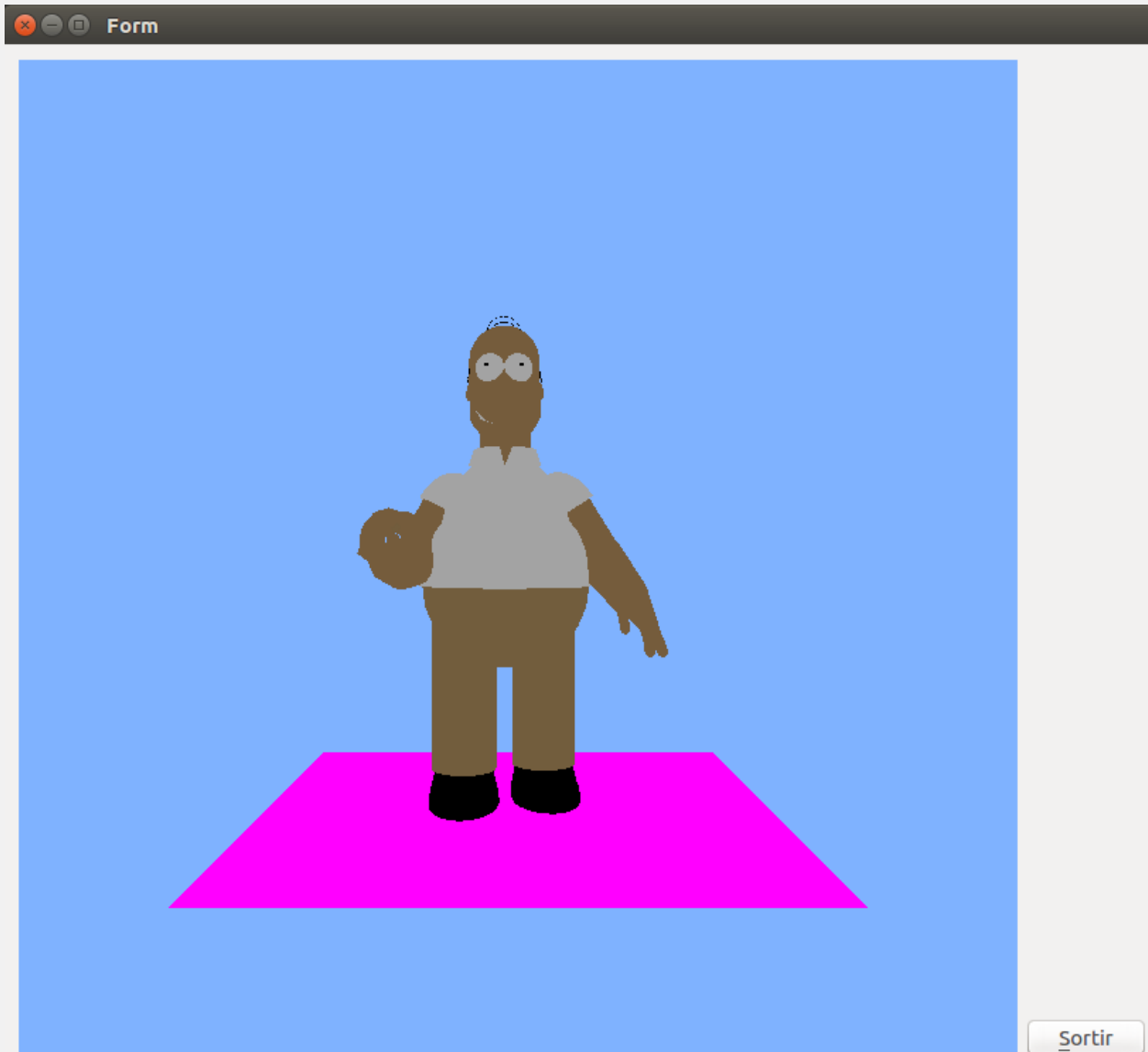
- Considerar la capsa (i esfera) mínima contenidora de l'escena
- Càlculer els paràmetres de posició i orientació (OBS,VRP,Up)
- Calcular els paràmetres de l'òptica perspectiva (FOV, raw, ZN, ZF)



Càmera en tercera persona

- Mètode per a calcular centre i radi d'escena: (exercici 1)
 - Donats punt mínim i màxim de la caixa contenidora
coneguts en la majoria de casos
- Usar centre i radi escena per a posar paràmetres càmera en tercera persona: (exercici 2)
 - Que es vegi escena centrada, sencera, sense retallar i ocupant màxim del viewport.

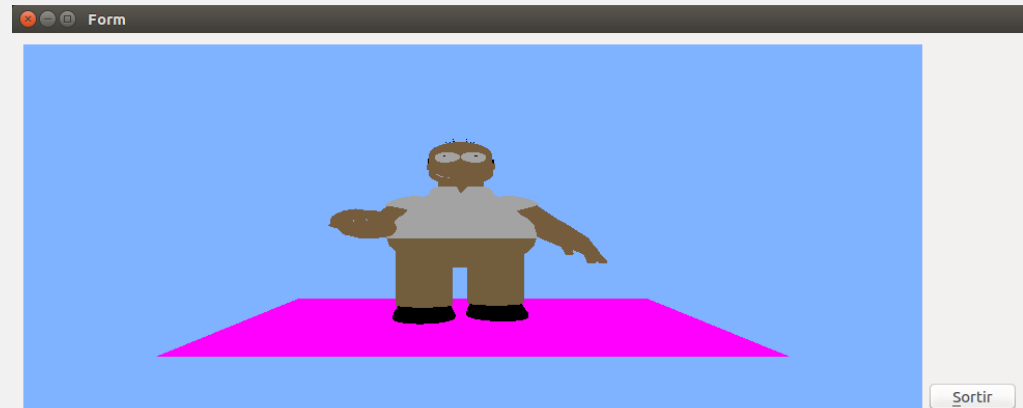
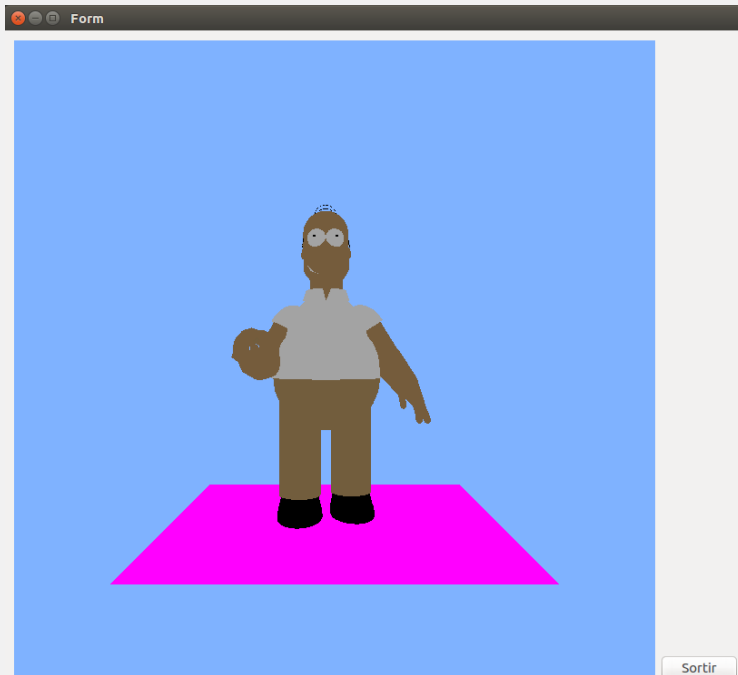
Càmera en tercera persona



Redimensionat sense deformació ni retallat

(exercici 3)

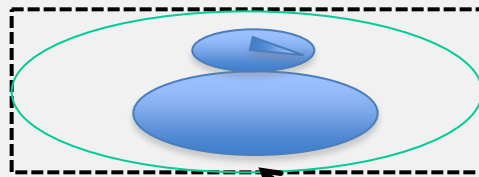
- Quan l'usuari redimensiona la finestra gràfica s'executa automàticament el mètode `resizeGL ()`
- Si aquest mètode no fa res:



Redimensionat sense deformació ni retallat

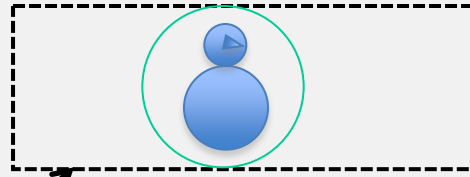
- La relació d'aspecte (ra) del window ha de ser igual que la del viewport:
 $ra_w = ra_v$
- Per tant si canvia la $ra_v \rightarrow$ ha de canviar la $ra_w \rightarrow$ refer perspective (...)

- Si $ra_v > 1$ i $ra_w = ra_v \Rightarrow$ la nova $a_w^* > a_w$ mínima requerida \Rightarrow No es retalla



Amb $ra_w=1$

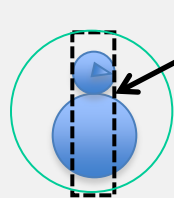
Viewport
 $ra_v > 1$



Amb $ra_w^*=ra_v$

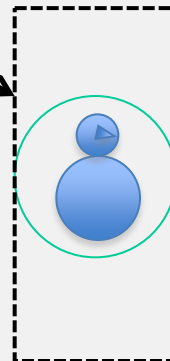
no cal modificar α_v (FOV)

- Si $ra_v < 1 \Rightarrow ra_w^* < ra_w \Rightarrow a_w^* < a_w \Rightarrow$ retallarà; per evitar-ho cal incrementar l'angle d'obertura (quedarà espai lliure a dalt i a baix)



Amb $ra_w=ra_v$

viewport



- Amb $ra_w = ra_v$ i nou FOV

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

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

Redimensionat sense deformació ni retallat

(exercici 3)

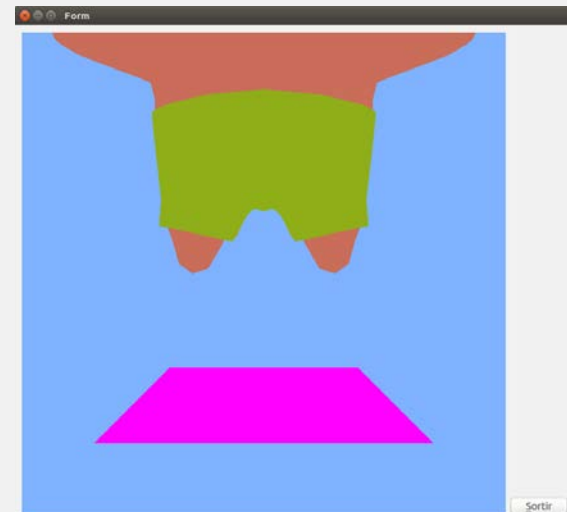
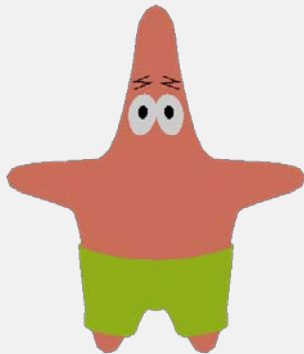
- El mètode `resizeGL` rep com a paràmetres l'amplada i alçada de la finestra gràfica
 - `void resizeGL (int w, int h);`

// possible càlcul de la relació d'aspecte del viewport
`float ra = float (w) / float (h);`
- Mètodes de `QOpenGLWidget` que ens poden ser útils:
 - `width ()` → retorna amplada de la finestra gràfica (int)
 - `height ()` → retorna alçada de la finestra gràfica (int)

Pintar objecte qualsevol

(exercici 4)

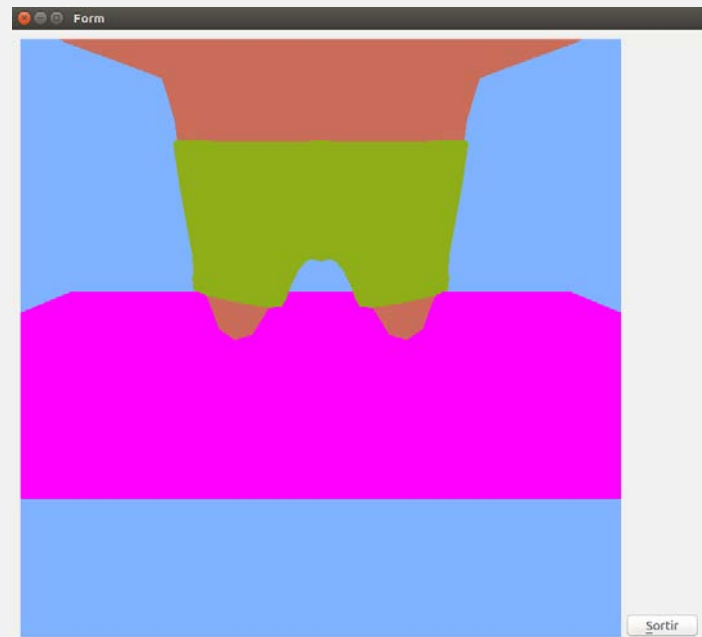
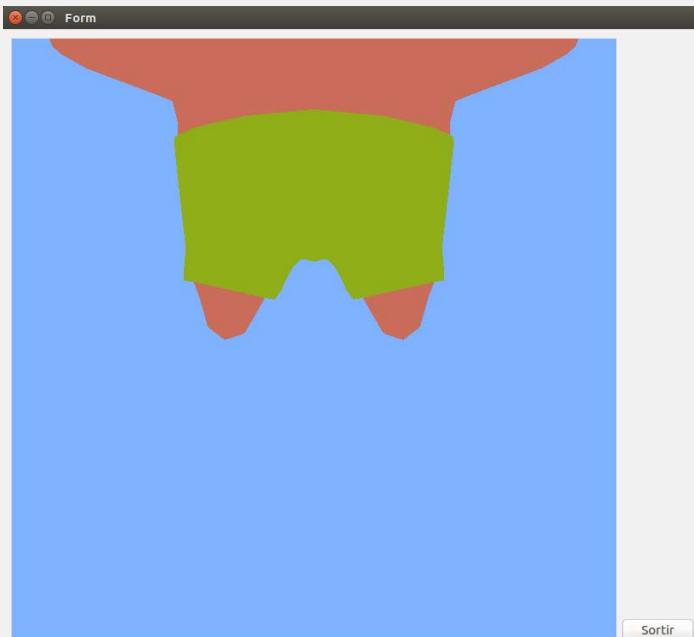
- Pintem el Patricio.obj
 - Model no centrat a l'origen i de mides no controlades (decisió del dissenyador del model)
 - Cal calcular la capsula contenidora del model
 - Es vol l'objecte **escalat per a que faci alçada 4 i amb la seva base centrada a l'origen** de coordenades
 - Cal afegir transformacions de model necessàries



Pintar objecte qualsevol

(exercici 4)

- Modifiquem el terra
 - Mida 5x5 i centrat a l'origen de coordenades
 - Canviem directament les coordenades dels vèrtexs
- Amb **obs-vrp** paral·lel a Z
- Amb **obs-vrp** no paral·lel a Z

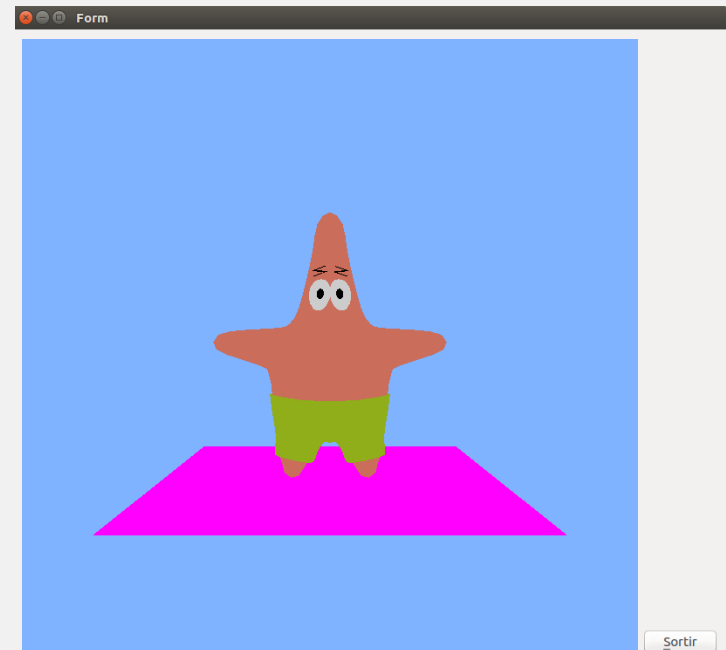
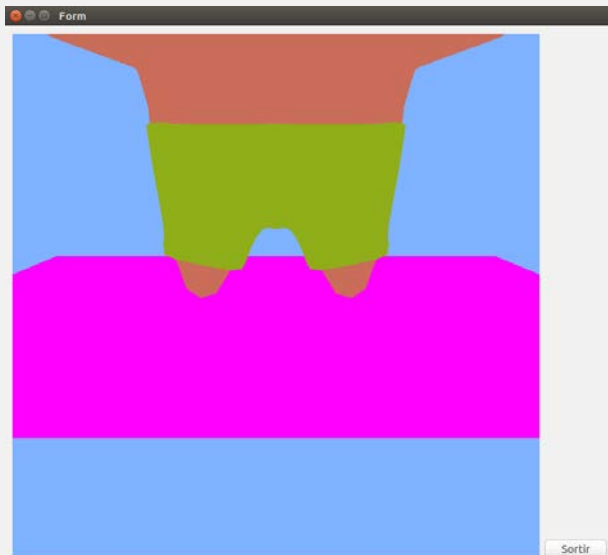


Pintar objecte qualsevol

(exercici 4)

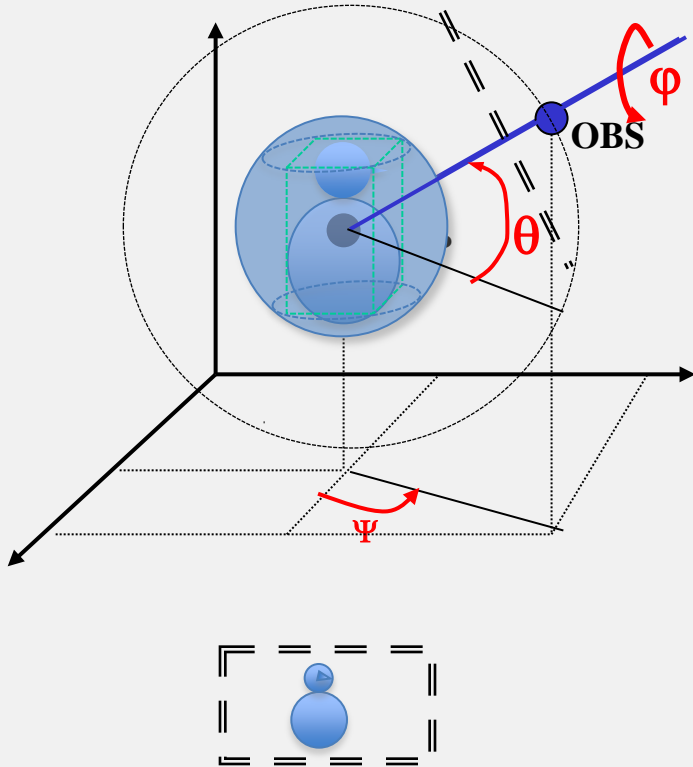
- Recalculem càmera
 - Patricio i terra no hi caben a la càmera que tenim
 - Cal recalcular els paràmetres (de posició i orientació i òptica) de la càmera perspectiva per a veure'l sencer i ocupant el màxim del *viewport*

la capsa de l'escena es calcula a partir e les dades del terra i del Patricio, que són conegudes.



Transf. *view* amb angles d'Euler

(exercici 5)



```
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)
```

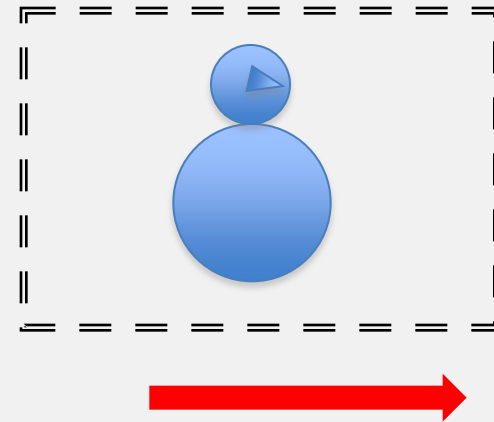
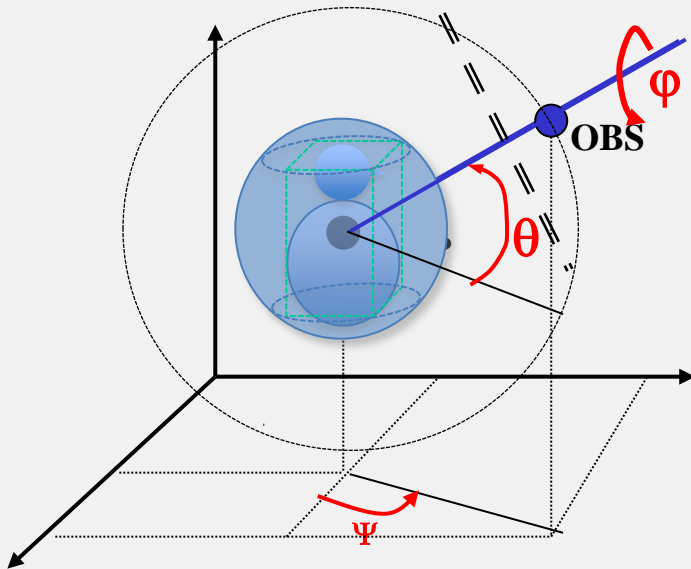
Atenció a l'ordre!

Compte amb signes:

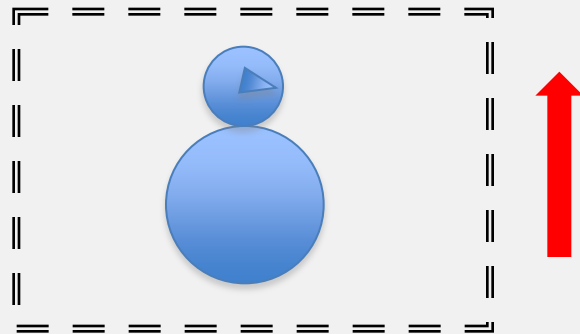
- Si s'ha calculat ψ positiu quan càmera gira cap a la dreta, serà un gir anti-horari respecte eix Y de la càmera, per tant, matemàticament positiu; com girem els objectes en sentit contrari, cal posar $-\psi$ en el codi.
- Si s'ha calculat θ positiu quan pugem la càmera, serà un gir horari; per tant,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

Interacció amb angles d'Euler

(exercici 6)



Moviment del ratolí d'esquerra a dreta → increment angle Ψ



Moviment del ratolí de baix a dalt → increment angle θ

Interacció amb angles d'Euler

(exercici 6)

Es vol que el moviment de càmera es faci prement el **botó esquerre** del ratolí, i no qualsevol.

- Si volem controlar el botó del ratolí que s'usa:

```
if ( e->buttons() == Qt::LeftButton ) // e és QMouseEvent
```

- Si volem controlar que a més no s'ha usat cap modificador (Shift, Ctrl, Alt):

```
if ( e->buttons() == Qt::LeftButton &&
```

```
    ! ( e->modifiers() &
```

```
        ( Qt::ShiftModifier | Qt::AltModifier | Qt::ControlModifier ) ) )
```

```
// controla que s'ha premut botó esquerre i cap modificador
```