

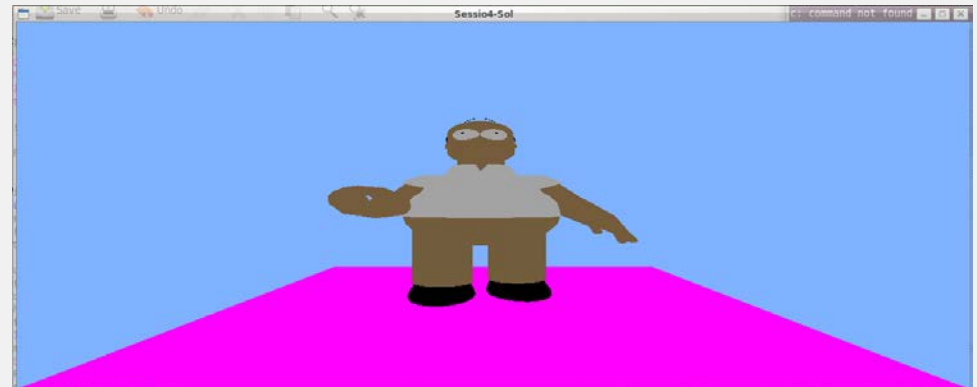
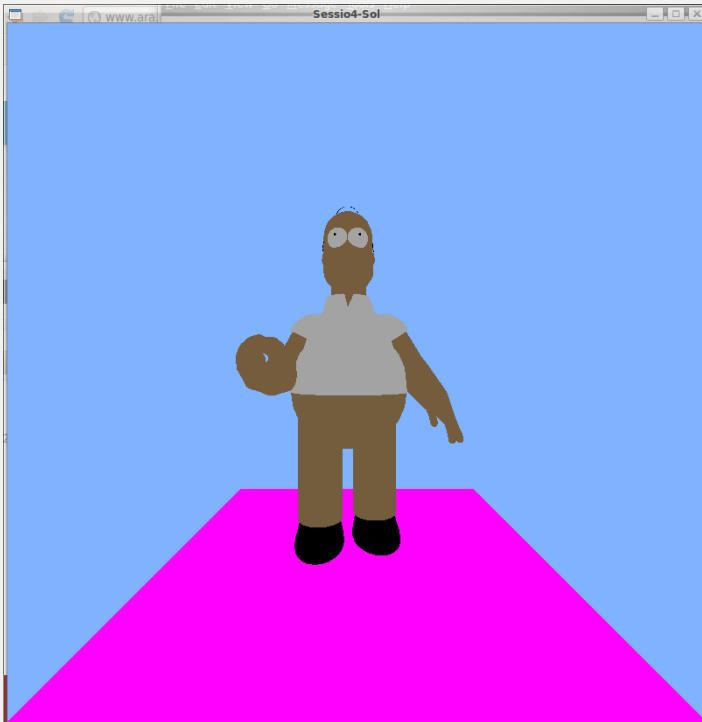
# Laboratori OpenGL – Sessió 5

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

# Redimensionat sense deformació

## (exercici 1)

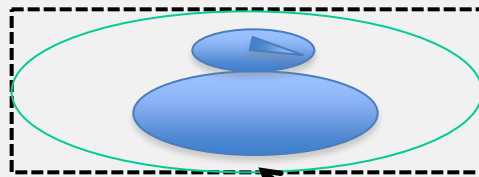
- Quan l'usuari redimensiona la finestra gràfica s'executa automàticament el mètode `resizeGL()`
- Si aquest mètode només modifica el *viewport*:



# Redimensionat sense deformació

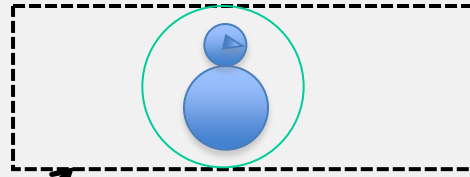
- La relació d'aspecte ( $ra$ ) del window ha de ser igual que la del viewport:  
 $ra_w = ra_v$
- Per tant si canvia la  $ra_v \rightarrow$  ha de canviar la  $ra_w \rightarrow$  refer perspective (...)

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



Amb  $ra_w=1$

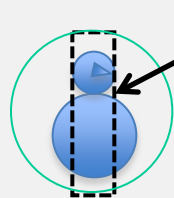
Viewport  
 $ra_v > 1$



Amb  $ra_w^*=ra_v$

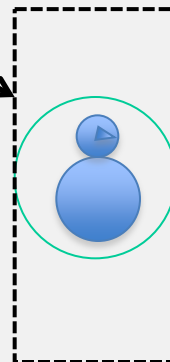
no cal modificar  $\alpha_v$  (FOV)

- 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

## (exercici 1)

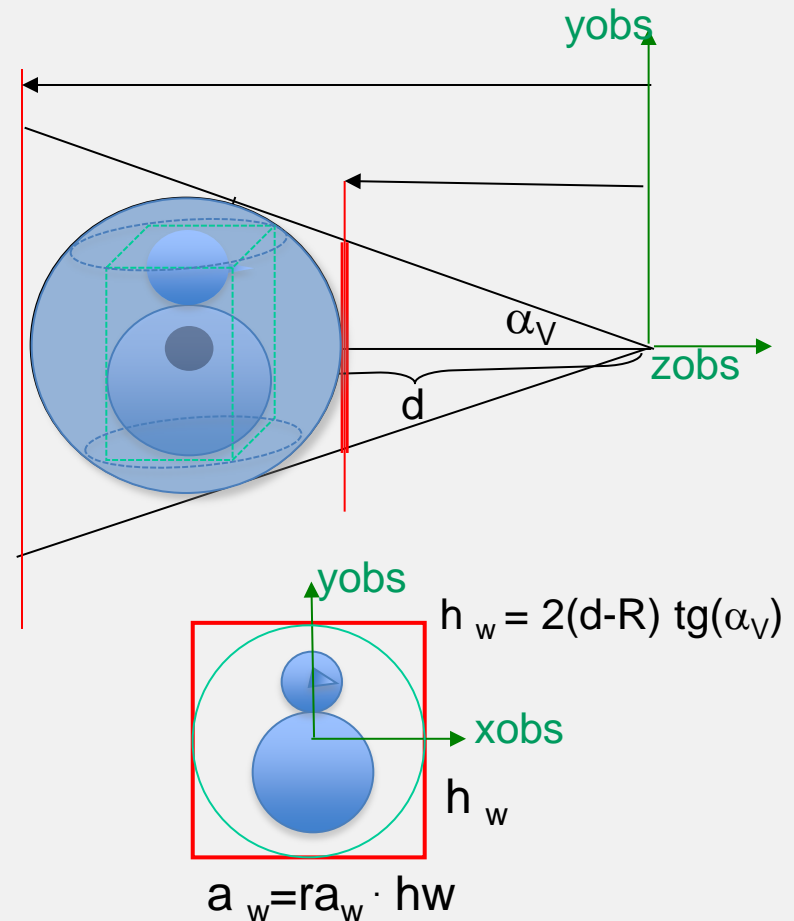
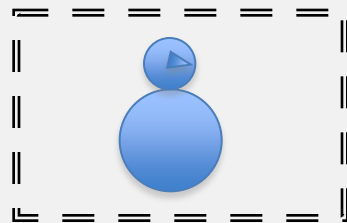
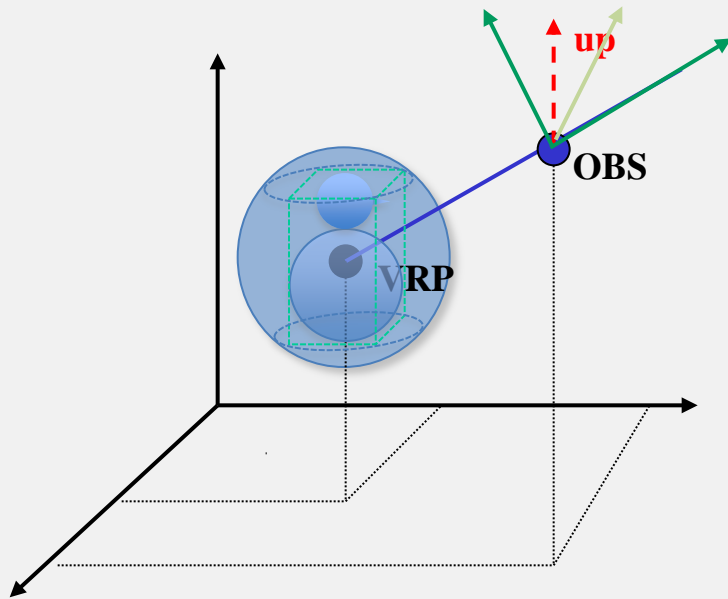
- El mètode `resizeGL` rep com a paràmetres l'amplada i alçada de la finestra gràfica
  - `void resizeGL (int width, int height);`
- Mètodes de `QOpenGLWidget` que ens poden ser útils:
  - `width ()` → retorna amplada de la finestra gràfica (int)
  - `height ()` → retorna alçada de la finestra gràfica (int)

// possible càlcul de la relació d'aspecte del viewport

`float ra = float (width ()) / float (height ());`

# Càmera en 3<sup>a</sup> persona (exercicis 2 i 3)

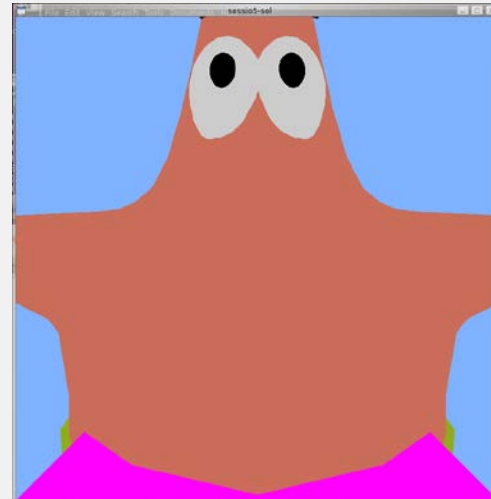
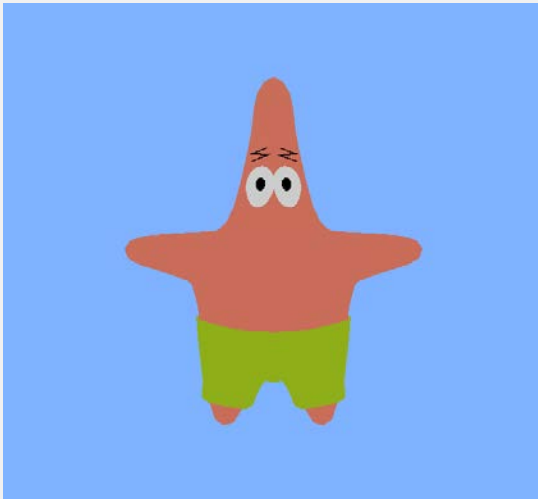
- Considerar la capsula (i esfera) mínima contenidora de l'escena
- Càlcul dels paràmetres de posició i orientació (OBS,VRP,Up)
- Càlcul dels paràmetres de l'òptica perspectiva (FOV, raw, ZN, ZF)



# Pintar objecte qualsevol

## (exercici 4)

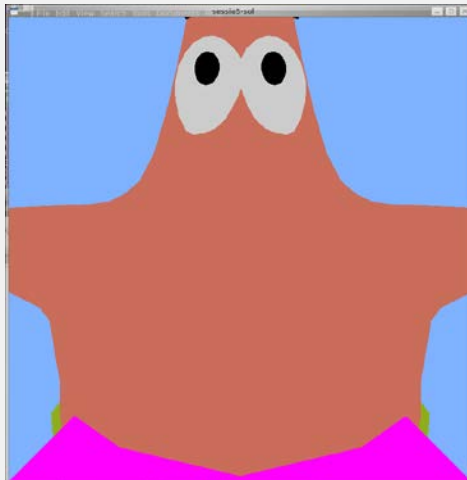
- Pintem el Patricio.obj
  - Model no centrat a l'origen i de mides no controlades (decisió del dissenyador del model)
  - Cal calcular la capsula contenidora del model
  - Es vol el model **sense escalar** i **centrat a l'origen** de coordenades
  - Cal afegir transformacions de model necessàries per a centrar el model



# Pintar objecte qualsevol

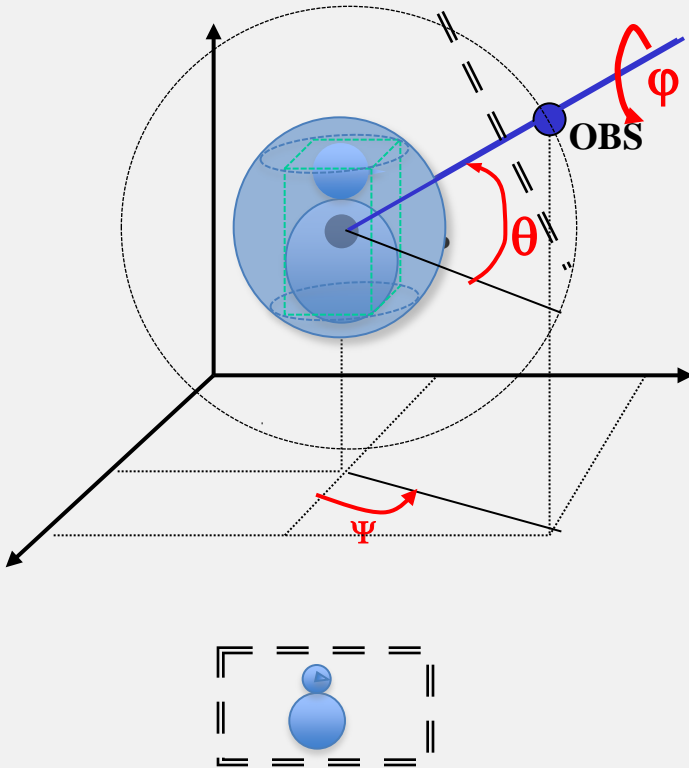
## (exercici 4)

- Recalculem càmera
  - Model del Patricio no hi cap 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*
    - tingueu en compte que la capsa de l'escena és ara la del Patricio però centrada a l'origen*
  - Què ha passat amb el terra?
  - No el pintem



# 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)
```

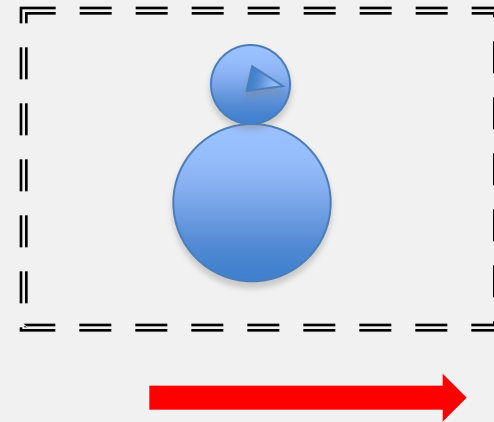
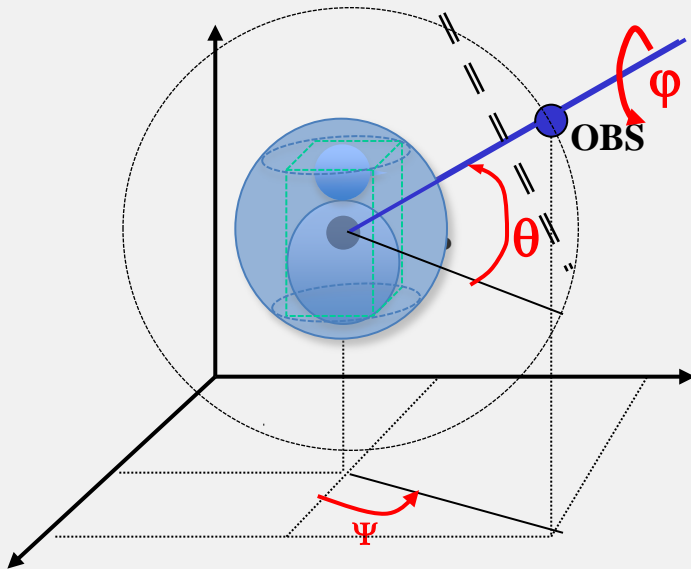
Ull amb signes:

- 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.

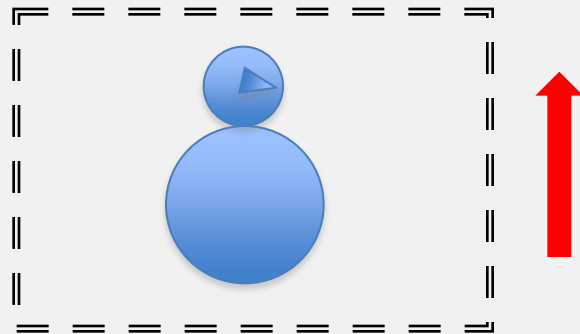


# Interacció amb angles d'Euler

(exercici 6)



Moviment del ratolí d'esquerra a dreta → increment angle  $\Psi$



Moviment del ratolí de baix a dalt → increment angle  $\theta$

# 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
```