Càmera i Visualització (2)

Càmera:
Posició, òptica

Il·luminació

Model
Geomètric
"Model Fronteres"
de sòlids

Visualització

MV → P_{obs}

OpenGL

Píxel(x,y,c)

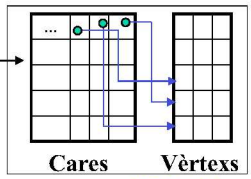
FrameBuffer

yD

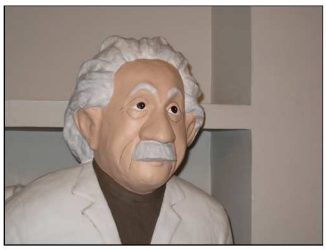
xD

taula [res_x][res_y] de color
color = (r,g,b) r,g,b en [0…1]

nom	TG/ param	model



Objectes en SCModel



1. Posició, orientació
2. Òptica

3. Fer la Foto

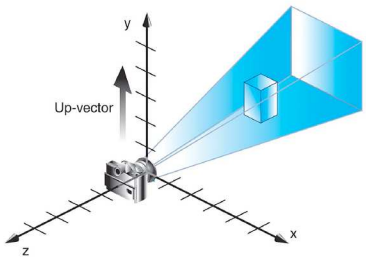
4. Emmarcar

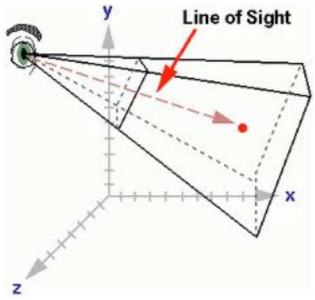
Finestra OpenGL
(Pantalla)

yD

Viewport/Vista

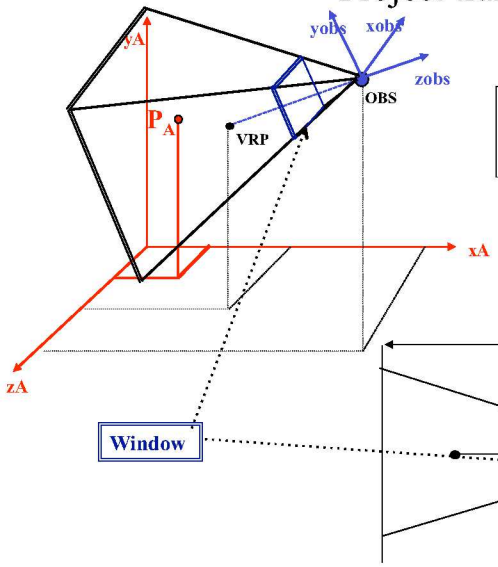
5





EL **VOLUM de VISIÓ -ÒPTICA- SEMPRE**
es defineix RESPECTE L'OBSERVADOR

Project transformation: òptica OpenGL



Tipus de càmera: Perspectiva

α_v (FOV= $2\alpha_v$), zNear, zFar, ra_w

gluPerspective (FOV, ra_w, zN, zF)

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

$a_w = h_w * ra_w$

zF

zN

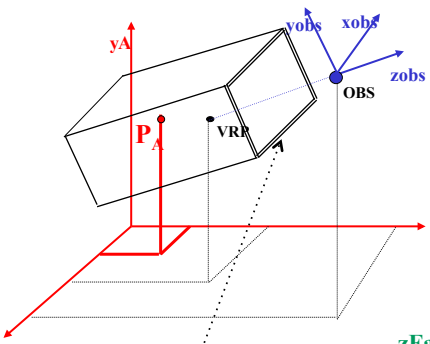
$ra_w = a_w / h_w$

α_v

OBS

h_w

a_w

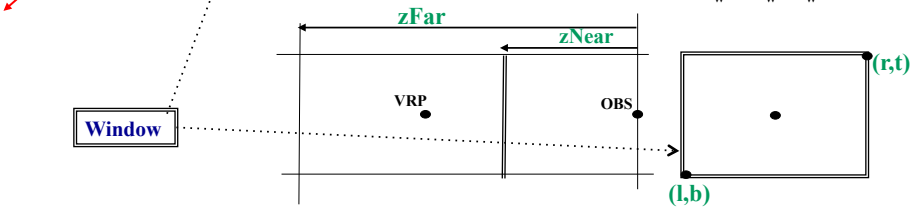


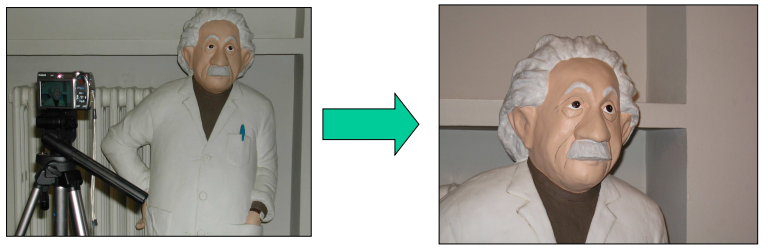
Tipus de càmera: Axonomètrica:

l,r,b,t, –window- , zNear, zFar

glOrtho (left, right, bottom, top, zN,zF)

$h_w = t - b$
$a_w = r - l$
$ra_w = a_w / h_w$



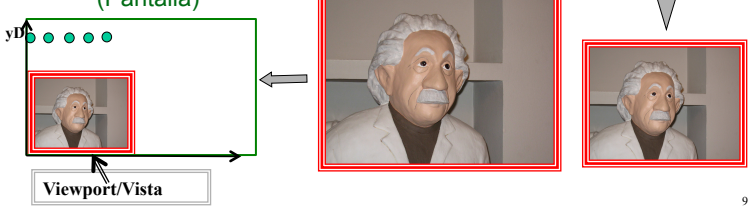


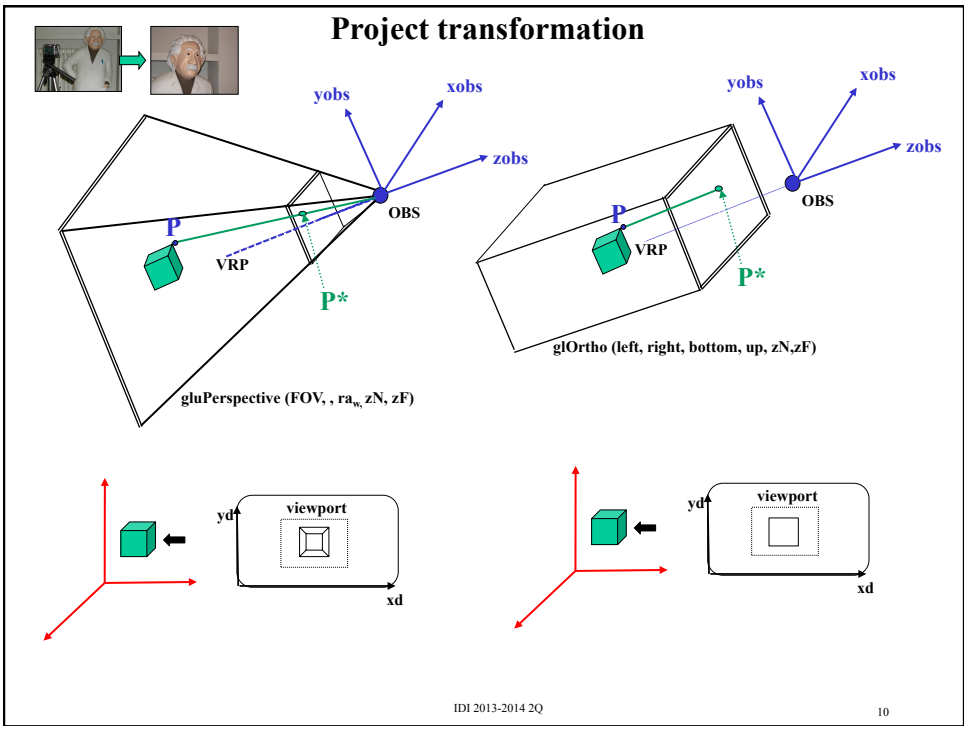
1. Posició, orientació
2. Òptica

3. Fer la Foto

4. Emmarcar

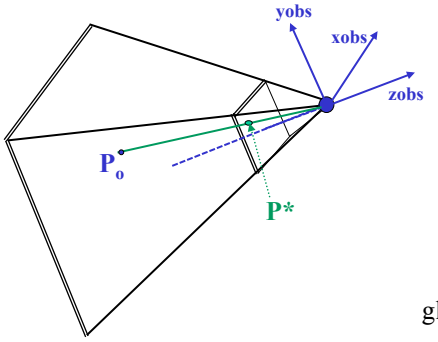
Finestra OpenGL
(Pantalla)



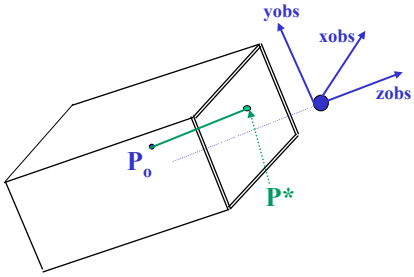

Project transformation
yobs
xobs
zobs
OBS
P
VRP
P*
gluPerspective (FOV, , ra_w, zN, zF)
yobs
xobs
zobs
OBS
P
VRP
P*
glOrtho (left, right, bottom, up, zN,zF)
yd
viewport
xd
yd
viewport
xd
IDI 2013-2014 2Q
10


IDI 2013-2014 2Q
11

Project transformations: càmera OpenGL

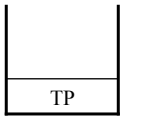


gluPerspective (FOV, ra$_{w,}$ zN, zF)



glOrtho (left, right, bottom, up, zN,zF)

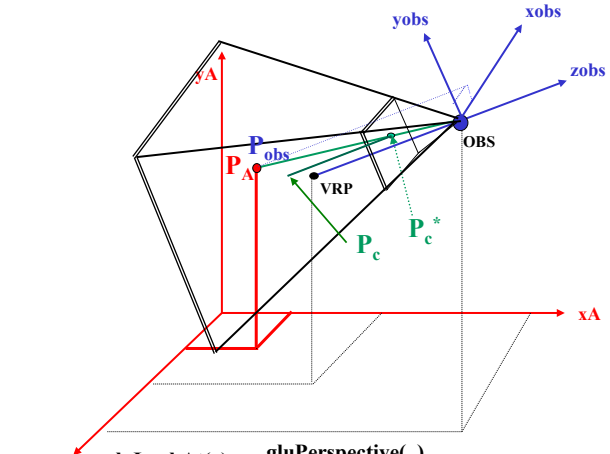
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
gluPerspective(....);

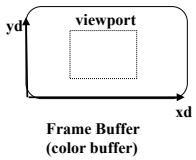


GL_PROJECTION

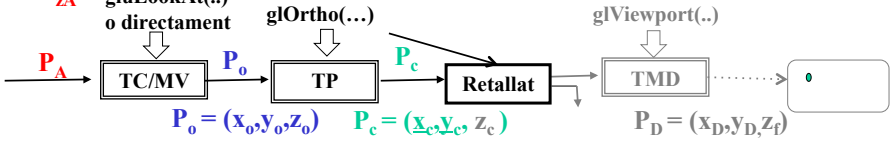
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho(....);

Procés de visualització OpenGL (2)

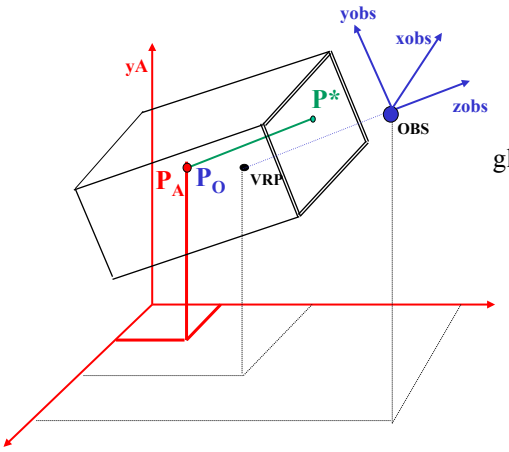




Frame Buffer
(color buffer)



$P_o = (x_o, y_o, z_o)$ $P_c = (x_c, y_c, z_c)$ $P_D = (x_D, y_D, z_f)$

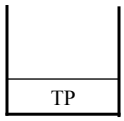


Tipus de càmera: Axonomètrica:

left,right,bottom, top , zNear, zFar

glOrtho (left, right, bottom, top, zN,zF)

TP



GL_PROJECTION

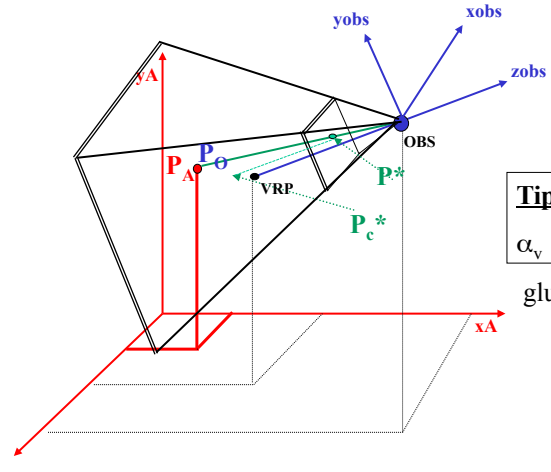
$P_c^* = TP\ P_O$

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

a=2/(r-l) b=2/(t-b)

c= 2 /(f-n)

d = (n+f)/ (f-n)

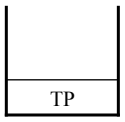


Tipus de càmera: Perspectiva:

α_v (FOV= $2\alpha_v$) , zNear, zFar, ra_w

gluPerspective (FOV, ra_w, zN, zF)

TP



GL_Projection

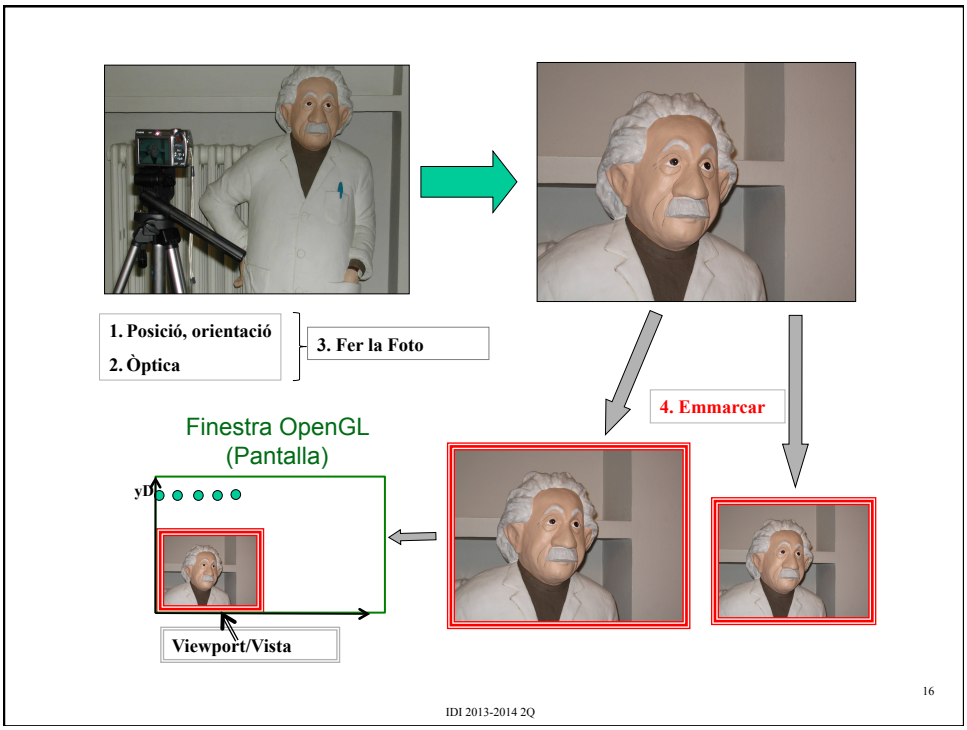
$P_c^* = TP\ P_O$

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

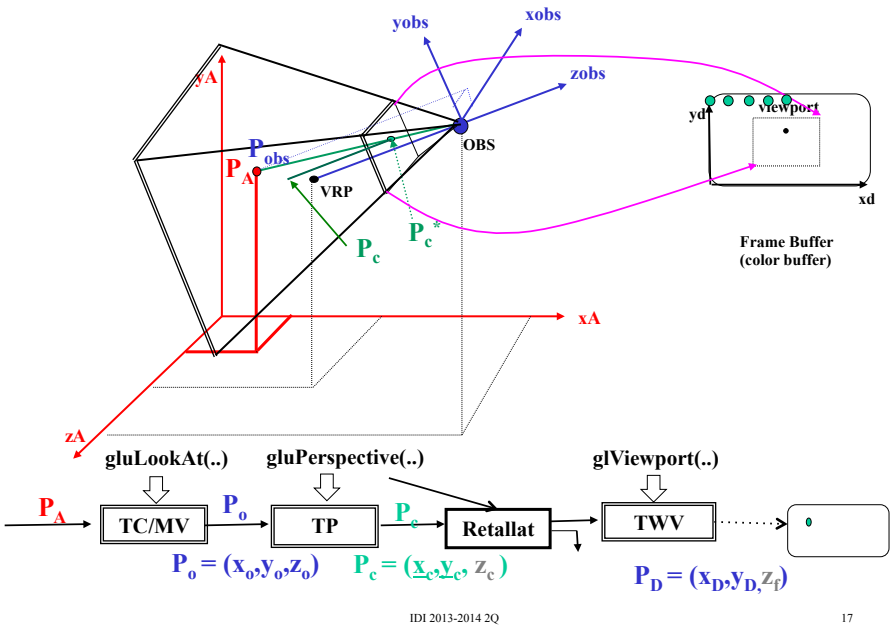
a = tg (FOV/2)

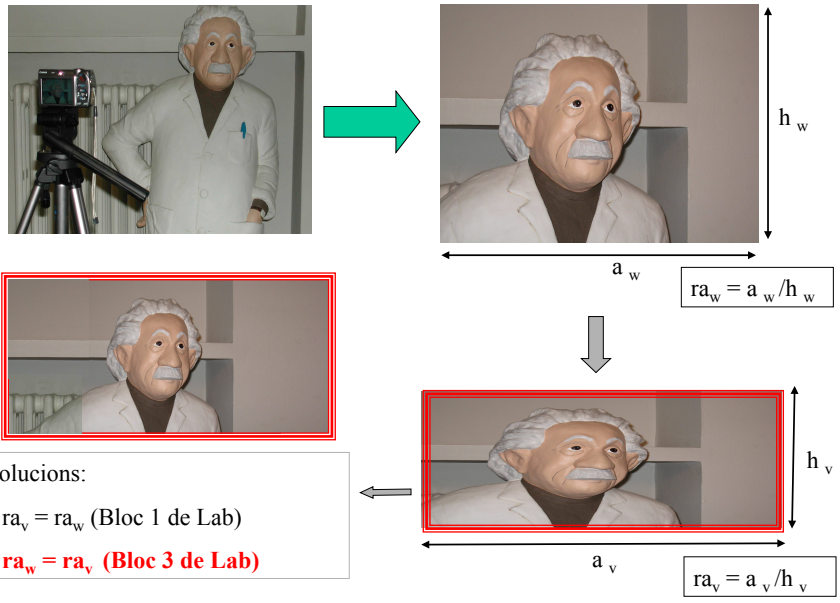
c = (f+n)/(n-f)

d = 2 n f /(n-f)



Procés de visualització OpenGL (3)

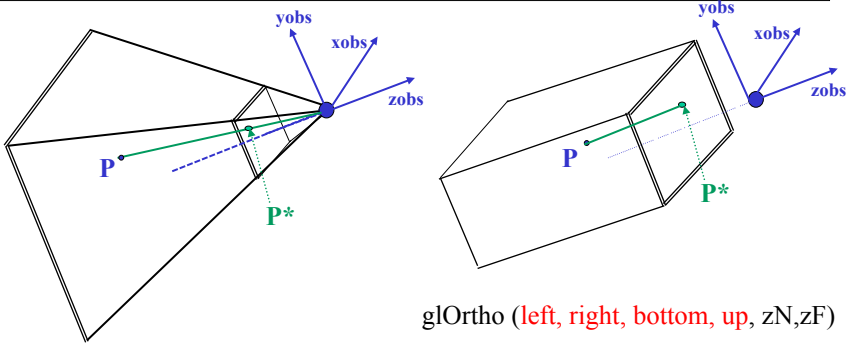




Solucions:

- $ra_v = ra_w$ (Bloc 1 de Lab)
- **$ra_w = ra_v$ (Bloc 3 de Lab)**

Project transformations: càmera OpenGL



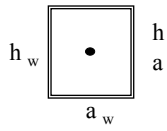
gluPerspective (FOV, ra_w, zN, zF)

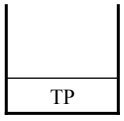
glOrtho (left, right, bottom, up, zN,zF)

$ra_w = a_w / h_w$

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

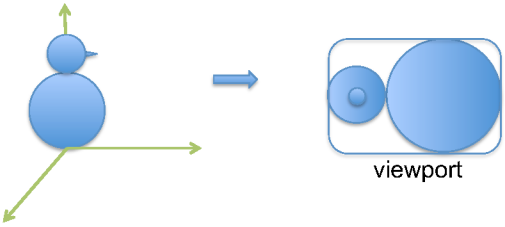
$a_w = h_w * ra_w$





GL_PROJECTION

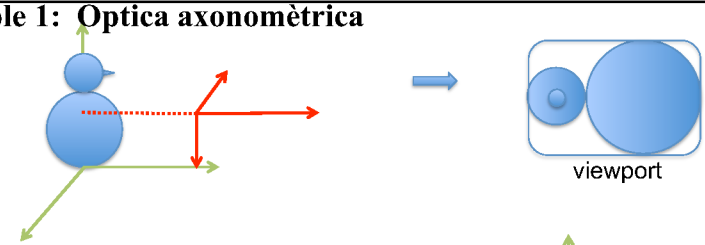
Exemple 1: Donada l'escena de la figura, formada per: una esfera de radi 10 i centre (0,10,0), altre esfera de radi 5 i centre (0,25,0), i un con de base centrada en (2.5, 25,0), r=2 i llargada 5 orientat segons l'eix X; iniciar tots els paràmetres d'una càmera ortogonal que permeti obtenir la image que s'indica en un viewport de 600x400.

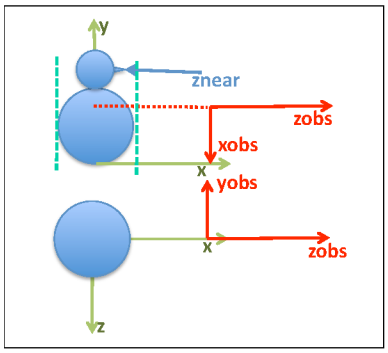


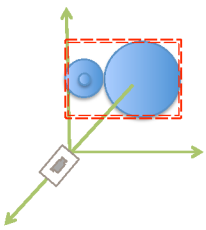

```
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glTranslatef (0.,0.,-30.);
glRotatef (90.,0.,0.,1.);
glRotatef (-90.,0.,1.,0.);
glTranslatef (0.,-15.,0.);
Pinta_ninot_de_neu; //
```

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

Exemple 1: Optica axonomètrica



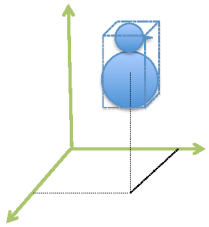




```
glViewport (0,0,600,400);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho(-15,15,-10,10,20,40);
```

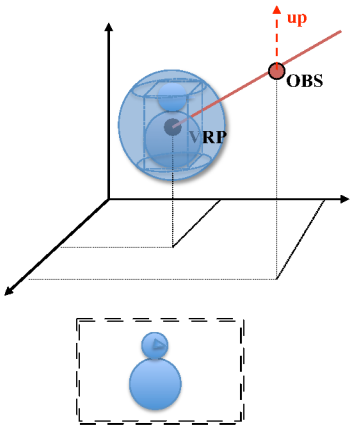
Exercici d'inicialització i moure càmera

- Veure escena sempre sense retallar i sense deformació (en pas a viewport)
- La imatge inicial volem que estigui centrada i ocupant raonablement el viewport i mostri escena vista des d'una posició arbitrària.
- Càmera perspectiva.
- Permetre modificació interactiva de punt de vista.



Imaginem aquesta escena

Exercici d'inicialització posicionament amb OBS, VRP, up



• Centrat => VRP=CentreEscena

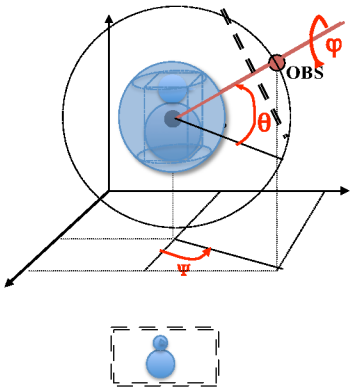
•Per assegurar que escena es veu sense retallar des d'una posició arbitrària CAL que OBS sempre fora capsa mínima contenidora en una posició qualsevol; per assegurar-ho CAL que OBS fora de l'esfera englobant de la capsa => distància "d" de l'OBS a VRP superior a R esfera.

-

CapsaMinCont=(xmin,ymin,zmin,xmax,ymax,zmax)
CentreEscena=Centre(CapsaMinCont) => ((xmax+xmin)2, ...)
R=dist((xmin,ymin,zmin),(xmax,ymax,zmax))/2
d>R; per exemple d=2R
OBS=VRP+ d* v ; v normalitzat en qualsevol direcció;
per exemple v= (1,1,1)/ ||(1,1,1)||

• up qualsevol que no sigui paral.lel a v; si volem ninot vertical (eix Y és vegi vertical) up=(0,1,0)

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



- Imaginem movem la càmera (OBS) sobre una esfera centrada en VRP de radi d.
  - VRP i d calculats com s'ha vist en exercici anterior.
  - Podem definir la posició amb angles Euler: ψ i θ. per exemple: ψ= 45º i θ=45º (o altres valors)
  - gir càmera sobre si mateixa: φ per exemple: 0º per veure eix Y vertical.
- Podem indicar càmera a OpenGL amb TG a objecte (més directe) o calculant VRP, OBS i up (veure transpa següent). Recordeu que transformacions "en codi" són als objectes (repasseu exercicis anteriors)

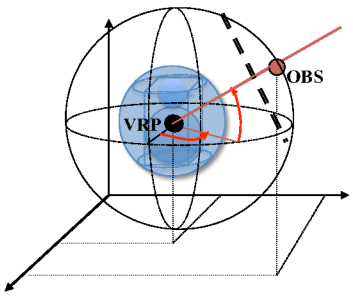
```
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glTranslatef (0.,0.,-d)
glRotatef (θ,1.,0.,0.)
glRotatef (-ψ.,0.,1.,0.)
glTranslatef (-VRP.x,-VRP.y,-VRP.z))
```

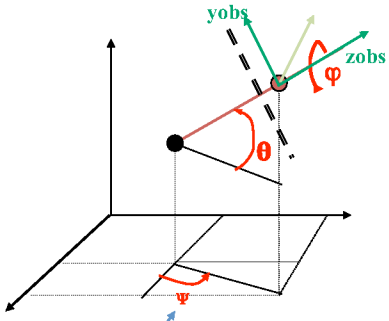
Ull amb signes:

- 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 −ψ 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 positu.

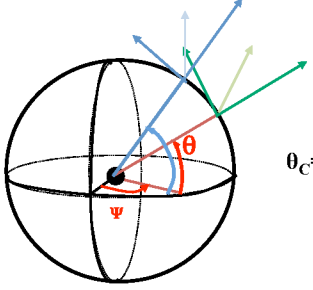
24

Moure la Càmera amb angles d'Euler





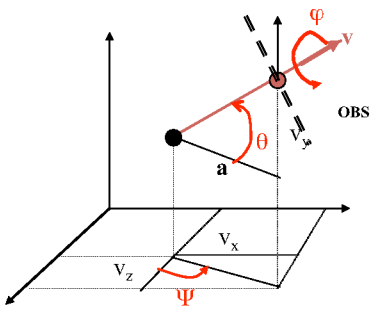
- Imaginem que en la interfície d'usuari estem ubicant la Càmera movent el cursor dreta/esquerra (Ψ) i pujar/baixar (θ). És com moure OBS sobre l'esfera i els angles d'Euler determinen un punt en esfera.
- També ho podeu pensar com si girem l'esfera per a què la seva posició respecte la càmera de defecte sigui la mateixa (agafar l'esfera amb la ma i girar-la).
- Codi per OpenGL directe a partir dels angles.



$\theta_C=\theta_C+\Delta ymov$

26

Càlcul VRP, OBS a partir angles Euler



VRP = Punt d'enfoc

OBS = **VRP** + d v

d > R ; per exemple: d = 2R

v_y = sin (θ); a = cos (θ);

v_z = cos (θ) cos (Ψ) ;

v_x = cos (θ) sin (Ψ);

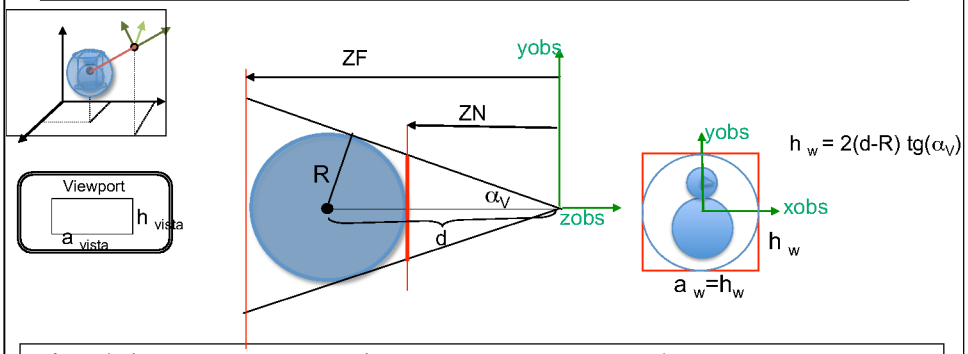
Un possible **up**: **up** = (0,1,0) (φ = 0º)

Noteu que "aquí" estem considerant els angles d'orientació de la càmera ==> Ψ, θ positius quan movem la càmera cap a la dreta i quan la pugem.

28

IDI 2013-2014 2Q

Tota escena en la vista, sense deformar i càmera perspectiva



- Si tota l'esfera englobant està dins la profunditat del camp de visió, no retallem l'escena.

  Per tant, ZN ∈] 0, d-R] ZF ∈] d+R, …];

  per a aprofitar precissió profunditat: ZN= d-R; ZF=d+R

- Per a aprofitar al màxim la pantalla (de fet el viewport), el window de la càmera s'ha d'ajustar a l'escena; una aproximació és ajustar el volum de visió (piràmide) de la càmera a l'esfera englobant.

  - R = d sin (α_V) ; α_V = arc sin (R/d) => FOV=2*α_V

  - com window està situat en ZN, α_V determina que la seva alçada sigui: $h_w = 2(d-R)\ tg(\alpha_V)$

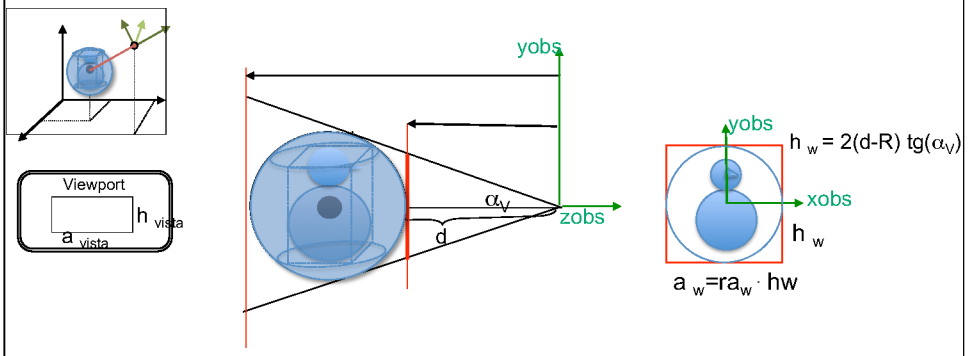
- $ra_w = a_w/h_W = 1$ (perque α_H hauria de ser igual a α_V per asegurar que esfera no retallada)

  - Però ULL!! per a què no hi hagi deformació, cal que ra_w sigui sempre igual a ra_v per tant, si no volem modificar el viewport:

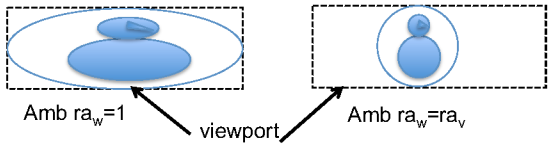
    $ra^*_w = ra_v$ amb aquesta nova ra^*_w es retallarà l'esfera? (continuarà tota dins volum de visió?)

29

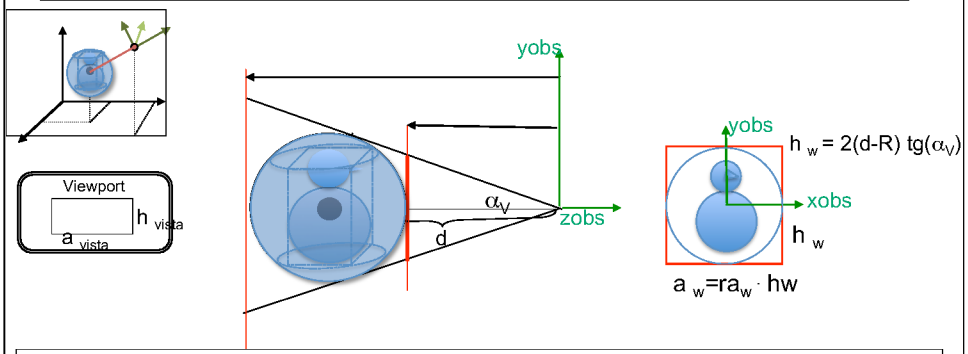
Tota escena en la vista, sense deformar i càmera perspectiva



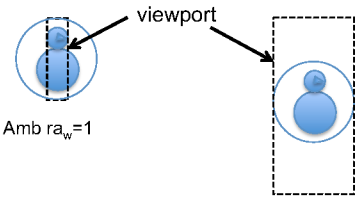
- Si $ra_v > 1$ (>ra_w mínima requerida 1) => No es retalla, no cal modificar α_V (FOV)



Tota escena en la vista, sense deformar i càmera perspectiva

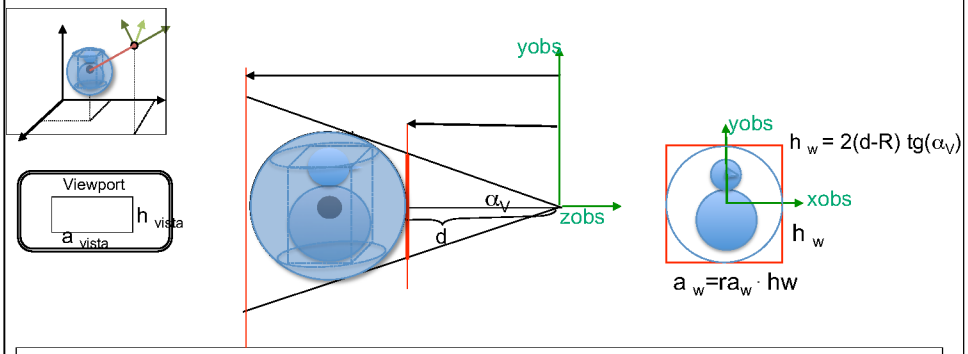


- Si $ra_v<1$ (<ra_w mínim requerida 1) => cal incrementar l'angle d'obertura (quedarà espai lliure a dalt i a baix)
  FOV=2 α^*_V on α^*_V = arctg(tg(α_V) / ra_v)



- Amb $ra_w=ra_v$ i nou FOV
- Sempre cal calcular el nou angle a partir de l'inicial (window quadrat). Penseu que pasaria si no ho feu i modifiqueu interactivament el viewport (finestra gràfica) fent-ho >1 i <1 molts cops seguits.

Tota escena en la vista, sense deformar i càmera perspectiva



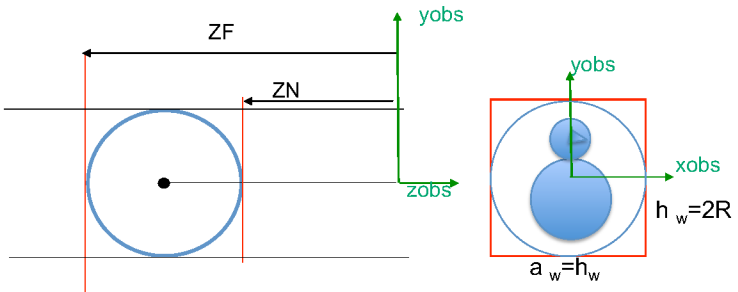
- Si $ra_v > 1$ (>ra_w mínima requerida 1) => No es retalla, no cal modificar α_V (FOV)
  Justificació: ra^*_w serà superior a 1; si no modifiquem l'angle FOV, h_w no canvia =>
  $a^*_w = ra^*_w \cdot h_w$ i com $ra^*_w > ra_w$ => $a^*_w > a_w$ i, per tant, serà
  més gran del necessari però es veurà tota l'esfera i quedarà espai pels laterals.

- Si $ra_v < 1$ (<ra_w mínim requerida 1) => cal incrementar l'angle d'obertura (quedarà espai lliure a dalt i a baix)
  FOV=2 α^*_V on $\alpha^*_V = arctg(tg(\alpha_V) / ra_v)$

  Justificació: *com* $a^*_w = ra^*_w \cdot h_w$, si no modifiquem angle, h_w no varia; com $ra^*_w < ra_w$ => $a^*_w < a_w$ i l'esfera quedaria retallada (en horitzontal). Per tant, cal incrementar l'angle α_V (i, per tant, h^*_w) per a garantir una amplada del window igual a la mínima requerida (igual que la h_w inicial).
  - sabem: $h^*_w = a_w / ra_v = (2(d-R)tg(\alpha_V)) / ra_v$ i per trigonometria $h^*_w = 2(d-R)\ tg(\alpha^*_V)$
  - per tant: $\alpha^*_V = arctg(tg(\alpha_V) / ra_v)$

32

Tota escena en la vista, sense deformar i càmera axonomètrica



- ZN i ZF mateix raonament que en càmera perspectiva.
- Window mínim requerit (centrat)= (-R,-R,R,R) => una $ra_W = 1$ (per què ?)
- Si $ra_w \neq ra_v$ ==> deformació (per què?)
  - Si $ra_v > 1$ => cal incrementar la ra_w => *modificar window*
    com $ra_w = a_w/h_w$ => podem incrementar a_w o decrementar h_w (és retallaria esfera!!)
    Per tant:
    $a^*_w = ra_v * h_w = ra_w * 2R$ => $inc\_a = a^*_w - a_w$
    window = (- (R+inc_a/2), R+inc_a/2, -R, R)= (-R ra_v, R ra_v, -R, R)

  - raonament similar per recalcular window quan $ra_v < 1$

33