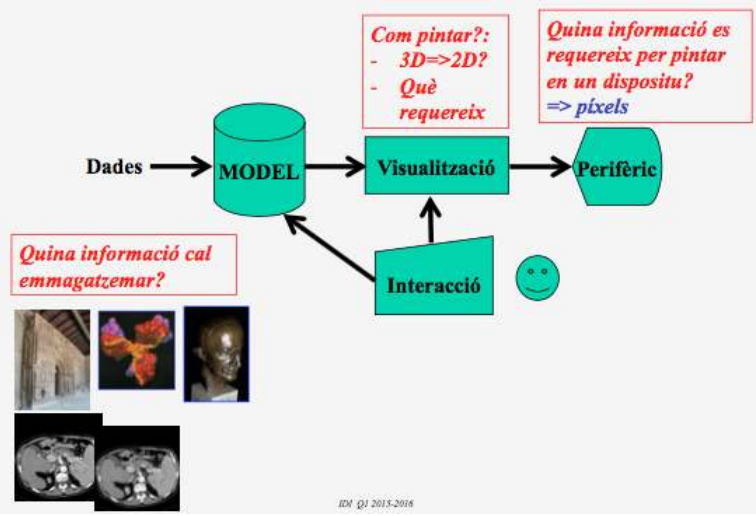
Primera part IDI

Interacció i Disseny d'Interfícies (Universitat Politècnica de Catalunya)

INTERACCIÓ I DISSENY D'INTERFÍCIES

ELEMENTS DEL SISTEMA GRÀFIC

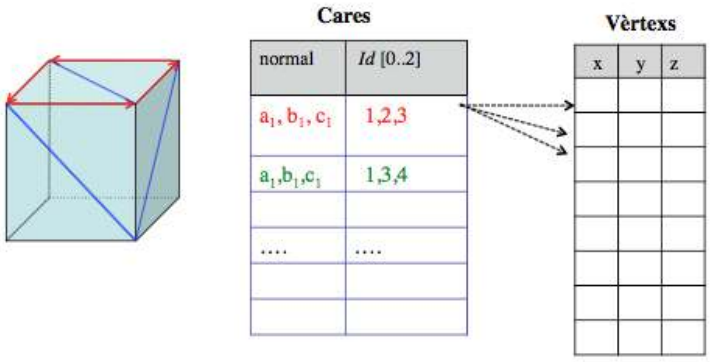


Models

Utilitzarem el model de fronteres i cares planes.

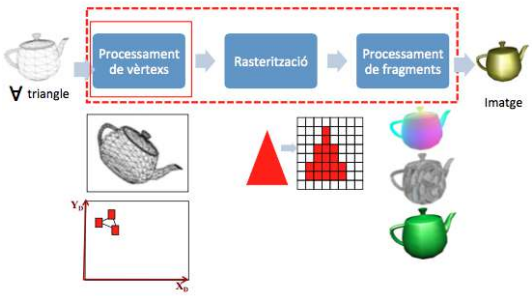
Guardem explícitament la informació de la superfície: cares (equació pla), arestes (equació recta) i punts (coordenades). Existeix, doncs, una relació d'adjacència entre els elements.
Per pintar les cares utlitzarem el model de cares planes (triangles).

Exemple:



Visualització.

Per indicar la càmara, definim: (1). l'ubicació respecte SCA (Sistema de Coordenades de l'Aplicació) i (2). el volum de visió. Així doncs, un cop determinades la posició, l'orientació i l'òptica, farem la foto i l'emmarcarem (viewport).
La visualització és el procés en que rebem una figura en un MG determinat i la convertim en imatge.



1. Processament de vèrtexs i sistemes de coordenades.

   Segueix les següents etapes:

   (a). World transformation space -> (viewing transform) -> Eye space.

   Transformació de visualització, transformació de càmera.

   -Requereix una matriu 4x4 per a fer canvi de sistema de coordenades.

   -Depèn de la posició i orientació de la càmera.

   (b). Eye space -> (projection transform) -> Clip space -> (clipping).

   Transformació de Projecció.

   - Facilita el clipping .

   - Requereix una matriu 4x4 per a fer canvi de sistema de coordenades.

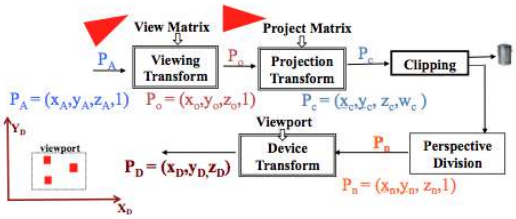
   - Depèn de l'òptica (volum de visió) de la càmera .

   (c). (clipping i Perspective division) -> Normalized Device space -> (viewport transform + depth transform) -> Devide space.

   Perspective Division

   - Facilita la transformació a coord. Dispositiu .

   - (x,y,z) del vèrtex interior a clipping entre [-1,1].



Viewport transformation

Transformació mon-dispositiu.

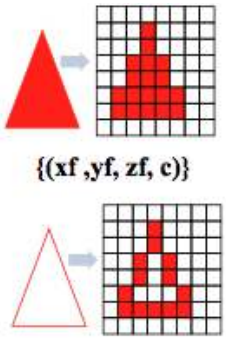
- Depèn del viewport definit amb glViewport.

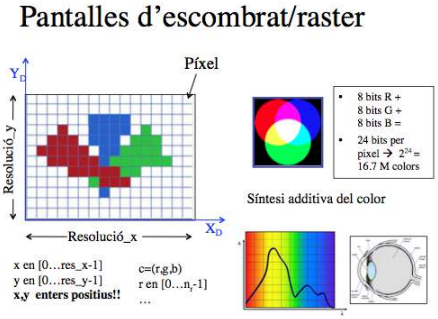
- (x,y) en coordenades de dispositiu.

Depth range transformation

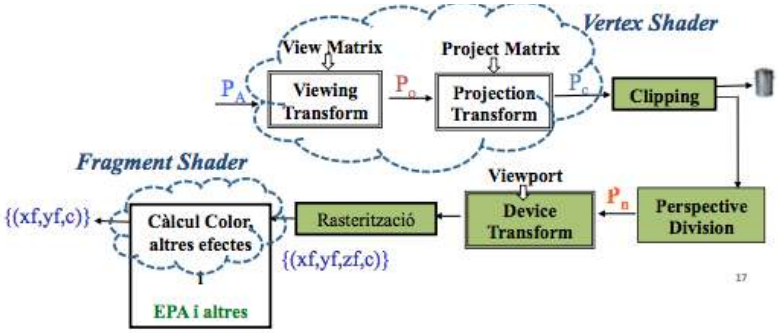
- la z per defecte [0,1].

2. Rasterització.



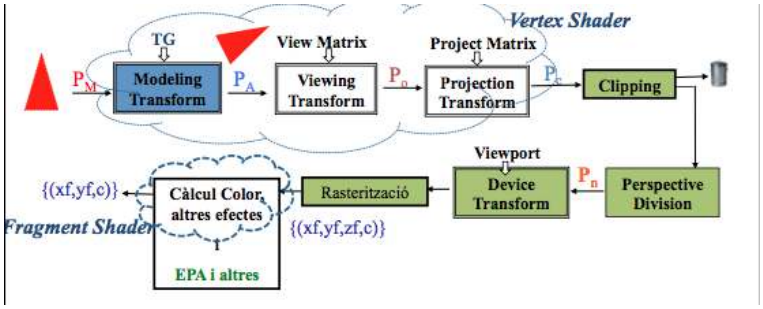


3. Processament de fragments.



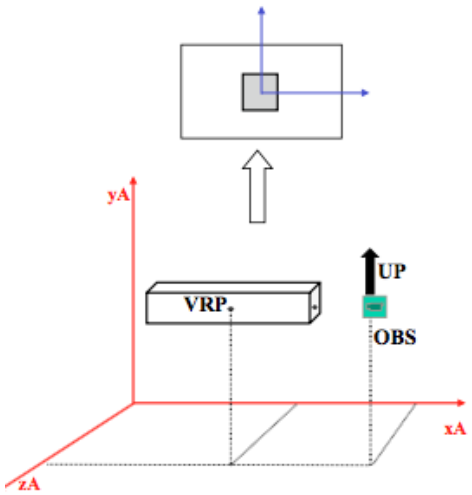
Tranformacions geomètriques del model.

Si, en una escena determinada, volem moure, girar, expandir o enpetitir, un dels objectes, li haurem d'aplicar una transformació geomètrica abans no es comenci el procés de visualització. Per tant, el paradigma projectiu que hem vist fins ara, s'ampliaria a:



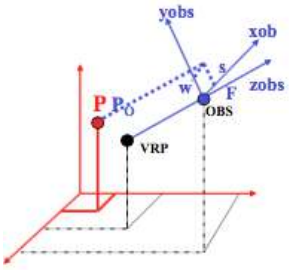
CÀMARA I VISUALITZACIÓ

Posicionament de la càmara: OBS, VRP, up.



VRP = View Reference Point.
OBS = Observador.
up = vector que indica la direcció de l'eix vertical de la càmara (inclinació).

Aquets tres paràmetres ens permeten realitzar el càlcul de la VIEW MATRIX:



$$VM = \begin{bmatrix} s.x & s.y & s.z & 0 \\ w.x & w.y & w.z & 0 \\ F.x & F.y & F.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \text{Trans}(-\mathbf{OBS})$$

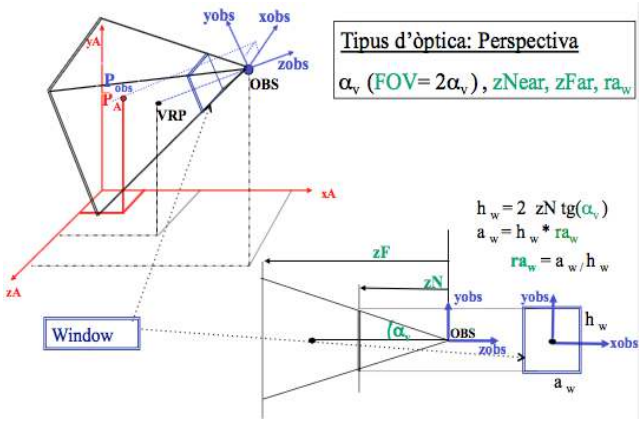
$$\mathbf{F = OBS - VRP} = (F.x, F.y, F.z) \qquad \mathbf{F = F / \| F \|}$$

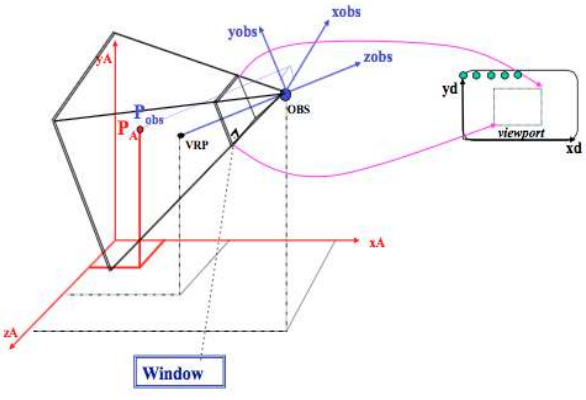
$$\mathbf{s = up \times F} \qquad \mathbf{s = s / \|s\|}$$

$$\mathbf{w = F \times s}$$

Òptica de la càmara: perspectiva.

Determina la geometria potencialment visible. El volum de visió (o òptica) es defineix sempre respecte l'OBS. Nota: si ra_w != ra_v es produeixen deformacions.





Gràcies als paràmetres de l'òptica perspectiva (FOV, zNear, zFar, ra_w, podem calcular la PROJECTION (TRANSFORM) MATRIX:

$$TP = \begin{pmatrix} 1/ra*a & 0 & 0 & 0 \\ 0 & 1/a & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \begin{matrix} a = tg\ (FOV/2) \\ c = (f+n)/(n-f) \\ d = 2\ n\ f\ /(n-f) \\ \\ \end{matrix}$$

L'òptica de la càmara, porta implícitca la definició d'un clip space per unes coordenades de clipping. Així doncs, per a què un vèrtex estigui dins del volum de visió, cal que les seves coordenades no superin les de clipping.

Exemple càlcul coordenades de clipping:

FOV =90°
ra=1
n=1, f=11

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1.2 & -2.2 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1.2 & -2.2 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} x_e \\ y_e \\ -1.2z_e - 2.2 \\ -z_e \end{bmatrix}$$

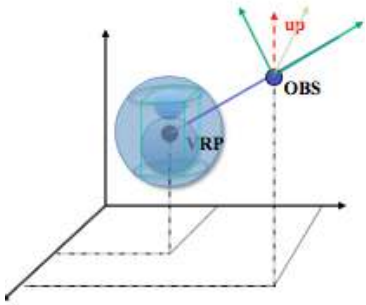
A partir d'aquí, es realitza la perspective division i la transformació a coordenades del viewport. Tancant així el pipeline de pricessament de vèrtexs.

Exercici exemple:

Exercici 18. Una escena està formada per dos cubs, un de costat 20 centrat al punt (0,0,0), i l'altre de costat 10 centrat al punt (15,0,0). Indiqueu TOTS els paràmetres d'una càmera que permeti veure a la vista dos quadrats, un damunt de l'altre (el més gran a sota), de manera que ocupin el màxim de la vista (*viewport*). Cal que indiqueu la posició i orientació de la càmera especificant **VRP**, **OBS** i **up**.

OBS	VRP	up
x = 5	x = 5	x = 1
y = 0	y = 0	y = 0
z = 20 (major que 10)	z = 0 (menor que 20)	z = 0

*La z està directament relacionada amb el FOV



Càmara en 3a persona (perspectiva).

Definim *capsa mínima contenidora*, com l'espai contigut dins de (x_{min}, y_{min}, z_{min}) - (x_{max}, y_{max}, z_{max}). L'esfera englobant de la cmc, és, lògicament, l'esfera mínima que engloba la cmc.

Incialització posicionament amb OBS, VRP, up (3a pers.):

- Objecte centrat: VRP al centre de l'escena (cmc).
- Per assegurar que veiem l'escena sense retallar des

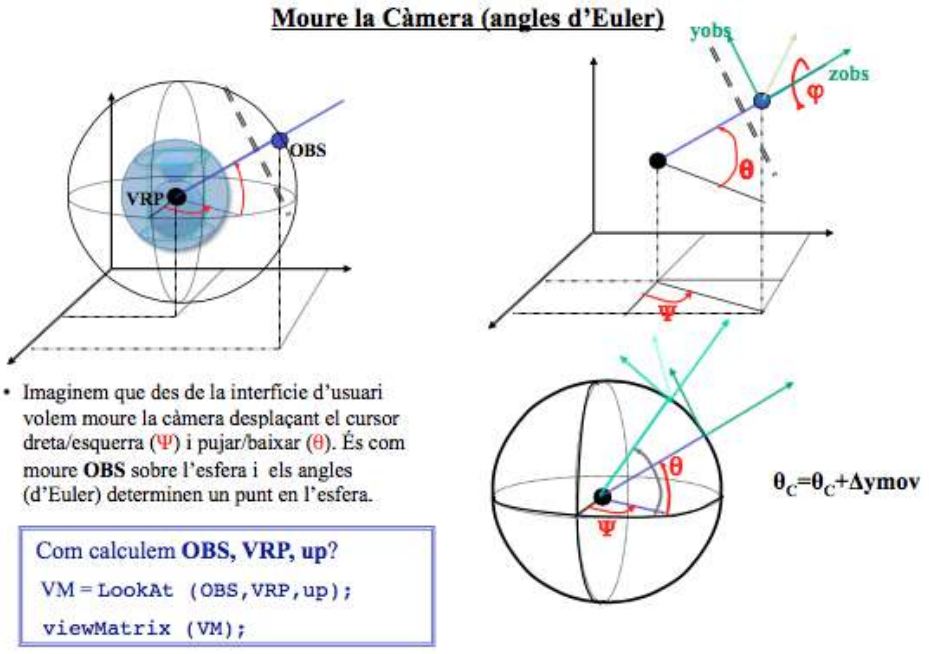
d'una posició arbitrària, cal que l'OBS es situi fora de la de l'esfera englobant. => la distància d de l'OBS al VRP ha de ser major que el radi r de l'esfera.

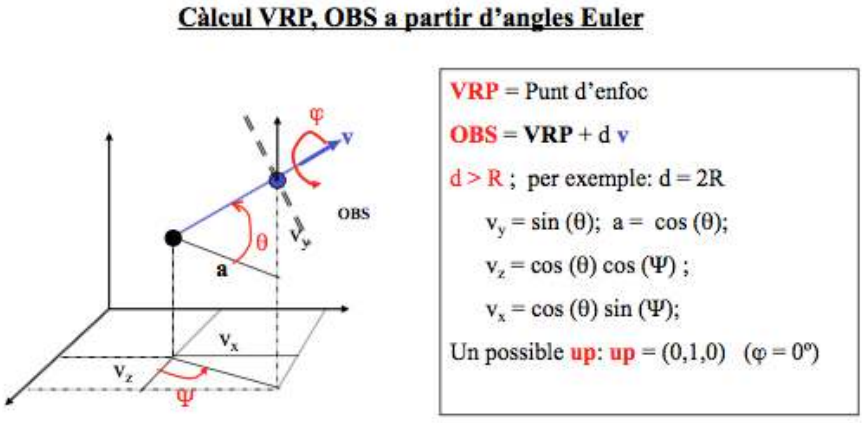
Així doncs,

CMC = (xmin, ymin, zmin, xmax, ymax, zmax), CentreEscena = Centre(CMC);
r = dist((xmin, ymin, zmin), (xmax, ymax, zmax)) / 2;
d > r, p.e., d = 2r;
OBS = VRP + d*v, on v és un vector normalitzat en qualsevol direcció, p.e.,
v = ||(1,1,1)||.

– up, qualsevol vector no paral·lel a v, p.e., si ho volem vertical: (0,1,0).

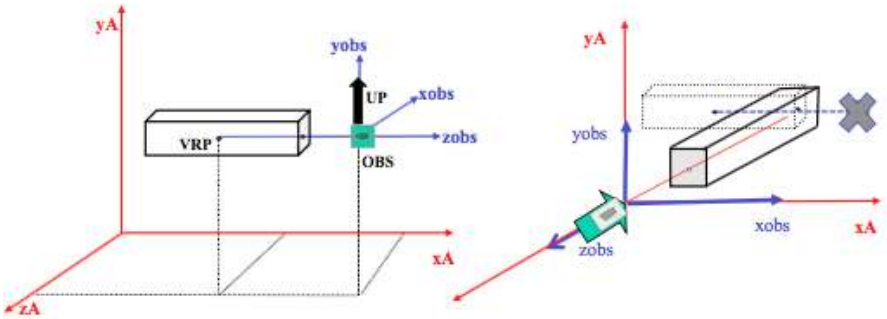
Moure la càmara: angles d'Euler.





I, amb els angles d'Euler, també podrem calcular la VM. A continuació, en veurem dos exemples, un de més senzill i un de més complexe.

Càlcul VM directe a partir d'angles Euler: consideracions i exemple senzill



1. Requerim VM per a referenciar (tenir posició relativa) escena respecte càmera.
2. És equivalent, per tenir mateixa posició relativa, moure la càmera en una direcció que escena en la contrària.
3. Si la posició relativa entre objecte/escena i càmera és la mateixa => mateixes coordenades dels punts respecte càmera (respecte SCO).
4. L'anomenada càmera de defecte d'OpenGL: coincideix amb SCA (OBS "mira" cap a Z^-)

Càlcul VM directe a partir d'angles Euler: exemple senzill

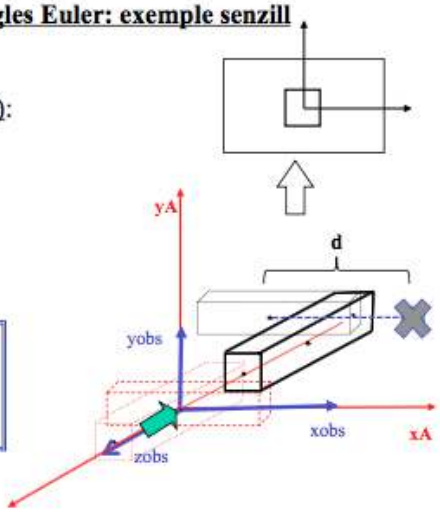
Moviment de l'objecte (una possibilitat):

1) T(**-VRP**)
2) Gy(-90)
3) T (0,0,-d)

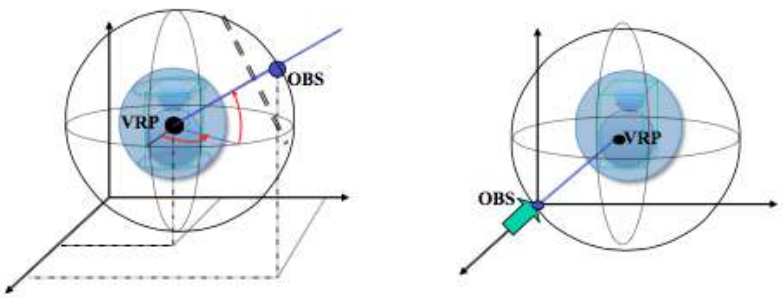
VM = T(0,0,-d) G_Y (-90) T(**-VRP**)

(angles positius => girs anti-horaris)

```
VM=Translate(0,0,-d);
VM= VM*Rotate(-90,0,1,0);
VM= VM*Translate(-VRP.x,-VRP.y,-VRP.z);
viewMatrix (VM);
```

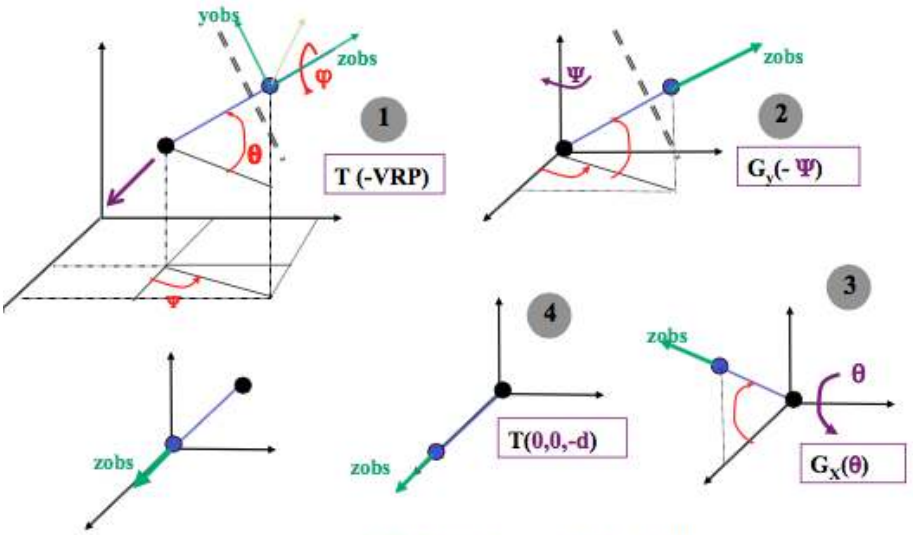


Càlcul VM directe a partir d'angles Euler: exemple més complexe



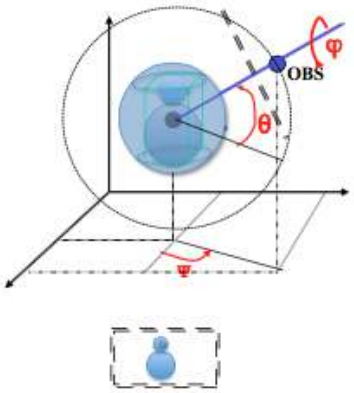
- Ho podeu pensar com si girem l'esfera per a què la seva posició respecte la càmera de defecte sigui la mateixa. Agafar l'esfera amb la mà i posicionar-la.
- Noteu que zobs passarà a ser coincident amb zA (SCO i SCA coincidiran)
- Pensarem el moviment tenint en compte que sabem calcular matrius de gir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

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



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

Finalment, veiem un exercici exemple d'inicialització de la càmara amb els angles d'Euler:



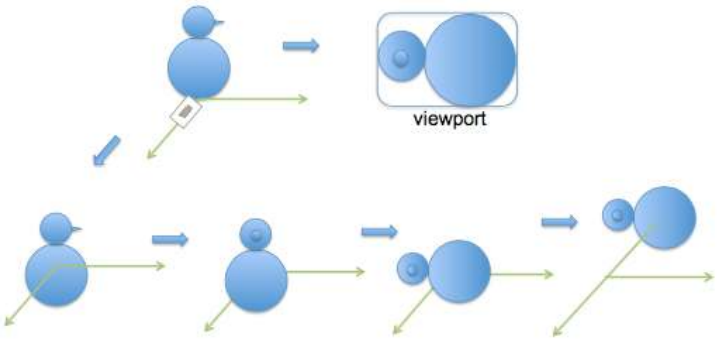
$$VM= T(0,0,-d)*G_Z(-\varphi)* G_X(\theta)*G_y(-\Psi)* T(-VRP)$$

```
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)
```

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.

Exemple Ninot: Càmera amb TG



$$TC=T(0,0,-30)G_z(90)G_Y(-90)T(0,-15.0)$$

```
VM= Translatef(0.,0.,-30.);
VM= VM*Rotate (90.,0.,0.,1.);
VM= VM*Rotate (-90.,0.,1.,0.);
VM= VM*Translate (0.,-15.,0.);
ViewMatrix(VM);
Pinta_Ninot();
```

Òptica per veure tota l'escena. Optimització del viewport. (perspectiva).

Si tota l'esfera englobant està dins del camp de visió, no retallem l'escena. Per tant, definirem

zN dins de [0, d-r], i

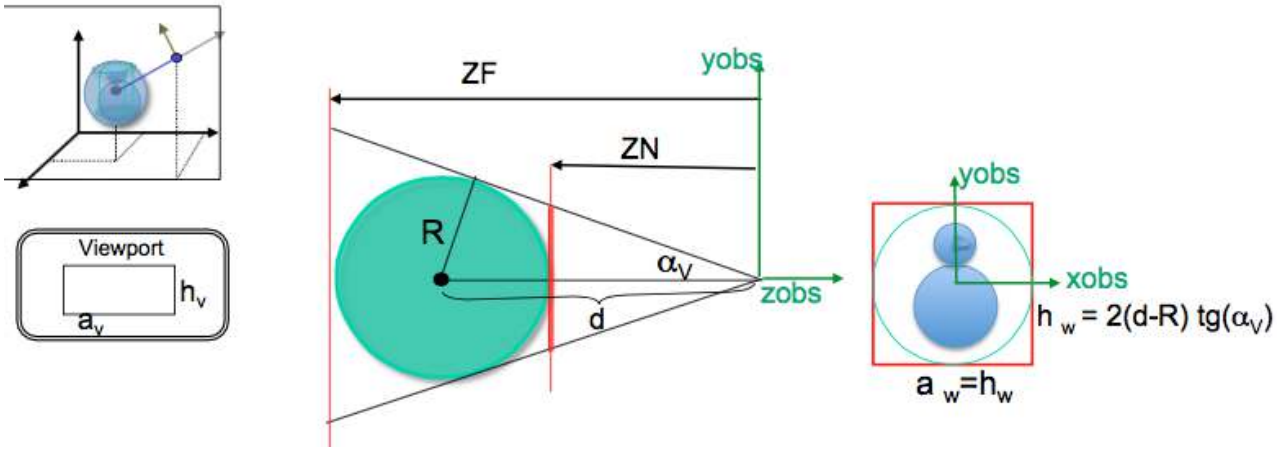
zF dins de [d+r, ...].

Per aprofitar al màxim el viewport, ajustarem el window de la càmara a l'escena. Una aproximació és ajustar el volum de visió (piramidal) a l'esfera englobant. Així,

$r = d \sin(\alpha_V)$; $\alpha_V = \text{arc sin}(r/d) \Rightarrow FOV = 2*\alpha_V$,

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

La ra_w serà 1, ja que α_H hauria de ser igual a α_V per assegurar-nos de que no retallem l'esfera. Però, què passa si ra_w != ra_v?



Si $ra_v > 1$, aleshores definim $ra_w = ra_v$ (a_w nova > a_w original) => NO ES RETALLA i no cal modificar el FOV.

Justificació: ra^*_w serà superior a 1; si no modifiquem l'angle FOV, h_w no canvia =>

$a^*_w = ra^*_w h_w$ 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 $ra_v < 1$, aleshores si definim $ra_w = ra_v$, ra_w nova < ra_w original => HI HA RETALLAT. Per evitar-ho, modificarem el FOV i deixarem espai lliure adalt i abaix:

$FOV = 2\alpha_V^*$, on $\alpha_V^* = \text{arctg}(tg(\alpha_V)/ra_v)$.

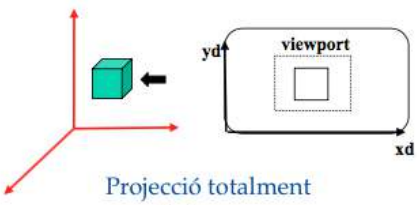
Justificació: *com* $a^*_w = ra^*_w h_w$, si no modifiquem angle, h_w no varia; com $ra^*_w < ra_w \Rightarrow a^*_w < a_w$ i l'esfera quedaria retallada (en horitzontal). Per tant, cal incrementar l'angle α_V (i, per tant, h^*_w) per a garantir una amplada del window igual a la mínima requerida (igual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

- sabem: $h^*_w = a_w / ra_v = (2(d-R)tg(\alpha_V)) / ra_v$ i per trigonometria $h^*_w = 2(d-R)\, tg(\alpha^*_V)$

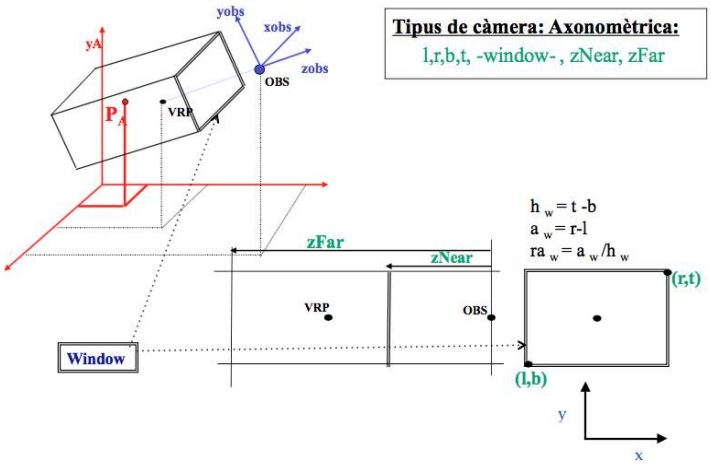
- per tant: $\alpha^*_V = \text{arctg}(tg(\alpha_V) / ra_v)$

Òptica axonomètrica: paràmetres.

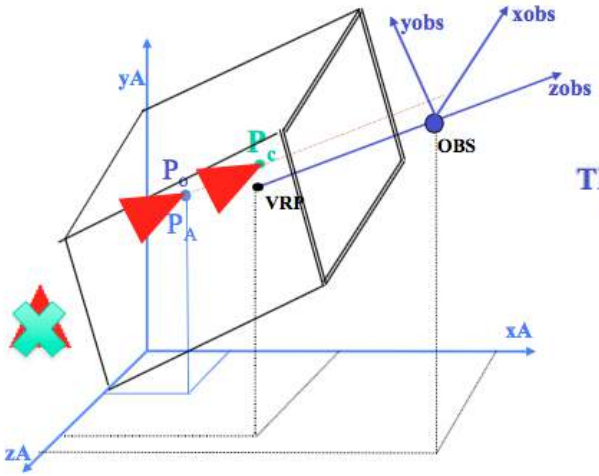
La projecció, ara, serà totalment paral·lela.



Projecció totalment paral.lela



Tipus de càmera: Axonomètrica:
l,r,b,t, -window-, zNear, zFar



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

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

c= 2 /(f-n)

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

Càmara en 3a persona (axonomètrica).

Igual que abans, definim *capsa mínima contenidora*, com l'espai contigut dins de (x_{min}, y_{min}, z_{min}) - (x_{max}, y_{max}, z_{max}). L'esfera englobant de la cmc, és, lògicament, l'esfera mínima que engloba la cmc.

Òptica per veure tota l'escena. Optimització del viewport. (axonomètrica).

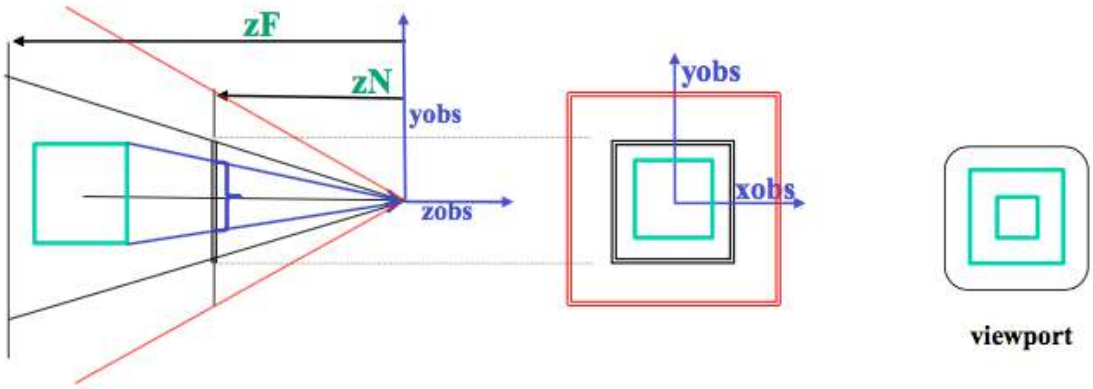
Els plans zN i zF segueixen el mateix raonament que en càmara perspectiva.
El window mínim requerit (centrat) és (-r,-r,r,r), per tant, $ra_w = 1$.
Com sempre, si ra_w != ra_v, tenim deformacions:

si $ra_v > 1$, $a_w^* = ra_v \cdot h_w = ra_w \cdot 2R$ => $\Delta a = a_w^* - a_w$; window = (- (R+ $\Delta a/2$), R+ $\Delta a/2$, -R, R)= (-R ra_v, R ra_v, -R, R).

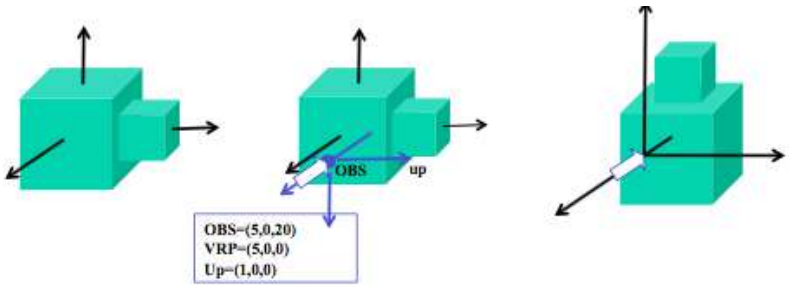
si $ra_v < 1$, ídem. perspectiva.

L'òptica i el zoom.

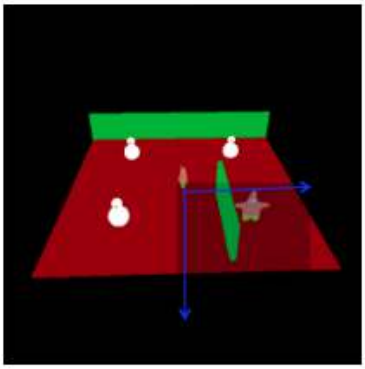
a) Modificar l'angle d'obertura (tot mantenint la *ra*) ;
Modificar window en axonomètrica
b) Modificar la distància de l'Obs al VRP
(modificant ZN i ZF adequadament)
c) Modificar Obs i VRP en la direcció –v → travelling



Exercici 18(bis). Una escena està formada per dos cubs, un de costat 20 centrat al punt (0,0,0), i l'altre de costat 10 centrat al punt (15,0,0). Indiqueu TOTS els paràmetres d'una càmera que permeti veure a la vista dos quadrats, un damunt de l'altre (el més gran a sota), de manera que ocupin el màxim de la vista (*viewport*). Cal que indiqueu l'òptica (perspectiva i axonomètrica) per veure l'escena optimitzant el viewport.

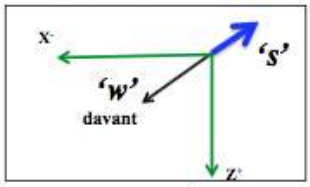


Exercicis de càmara en 1a persona.



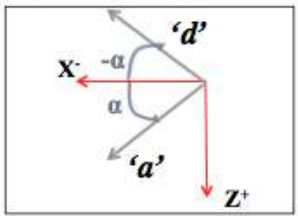
Moure el Patricio central

- Avançar/retrocedir (tecles 'w' i 's'):
  - modificar posició en la direcció del moviment "davant"
- Girar a la dreta/esquerra (tecles 'd' i 'a')
  - modificar "davant" (gir respecte eix Y).



OBS = (0,1,0)
VRP = (-10,1,0)
Up = (0,1,0)

zN = 0.1
zF = 15
FOV= 60º
ra = ra_v



Exercici 67: Disposem d'una càmera ortogonal amb els següents paràmetres:

OBS=(0.,0.,0.), VRP=(-1.,0.,0.), up=(0.,1.,0.), window de (-5,5) a (5,5), ra=1, zn=5, zf=10.

Indiqueu quin conjunt de paràmetres d'una càmera perspectiva defineix un volum de visió que conté l'anterior (és a dir, garanteix que es veurà, coma mínim, el mateix que amb la càmera axonomètrica):

a) FOV= 90, ra=1, zn= 5, zf=10
b) FOV= 60, ra=1, zn=5, zf=10
c) FOV= 60, ra= 2, zn=6, zf=11
d) FOV= 90, ra= 0.5, zn=5, zf=10

REALISME: ELIMINACIÓ DE PARTS OCULTES

Backface culling.

- Mètode EPA en espai objecte (a nivell de triangle).
- Requereix cares orientades, opaques, objectes

Càlcu

REALISME: IL·LUMINACIÓ

Els models d'il·luminació simulen el comportament de la llum per determinar el color d'un punt de l'escena.

- Permeten obtenir imatges molt més realistes que pintant cada objecte d'un color uniforme: