**Tema 5: càmera 3 persona**

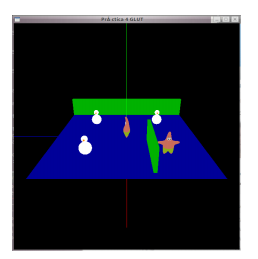
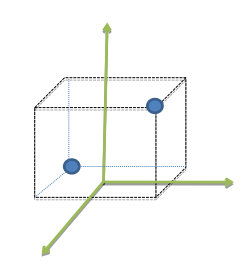
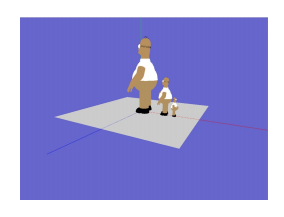
**Càmera en tercera persona**

Visualització inicial de l’escena tal que:

* Inclogui tota l’escena (no retalli cap objecte).
* Posició arbitrària de l’observador.
* Centrada en viewport.
* Optimitzant ocupació del viewport/vista.
* Sense deformació.

Dada: capsa mínima contenidora de l’escena:

cmin = (xmin, ymin, zmin) i cmax = (xmax, ymax, zmax)

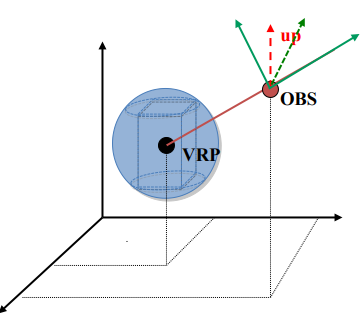


Inicialització posicionament amb OBS, VRP, up:

Centrat => VRP = centreEscena

Per assegurar que l’escena es veu sense retallar des d’una posició arbitrària cal que OBS estigui sempre fora de la capsa mínima contenidora; per tant, OBS ha de quedar fora de l’esfera englobant de la capsa.

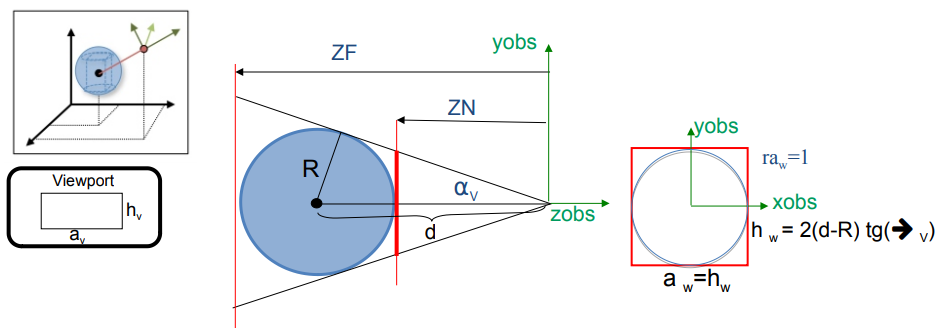
Distància “d” de l’OBS a VRP superior a R (radi) de l’esfera.



* **CapsaMinCont** (capsa mínima contenidora) = (xmin, ymin, zmin, xmax, ymax, zmax).
* **CentreEscena** = centre (CapsaMinCont) = ( (xmax + xmin) / 2, (ymax + ymin)/2, (zmax + zmin) / 2).
* **R** = dist ( (xmin, ymin, zmin), (xmax, ymax, zmax) ) / 2.
* **d > R** (per exemple, d = 2R).
* **OBS = VRP + d\*v** (v = vector normalitzat en qualsevol direcció);

**Up:** qualsevol que no sigui paral·lel a v; si volem escena vertical (eix Y es vegi vertical), up = (0, 1, 0).

Tota l’escena, sense deformar i òptica perspectiva:



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

Per tant, ZN ϵ [0, d – R] // ZF ϵ [d + R, …]

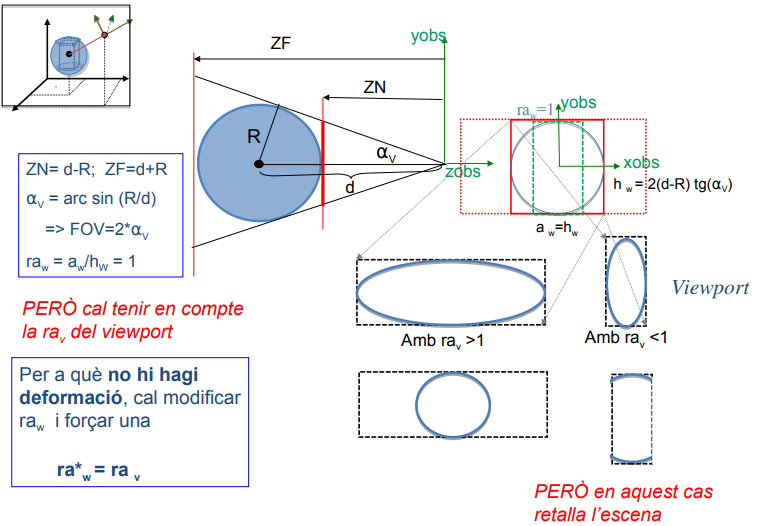
ZN = d – R; ZF = d + R per aprofitar la precisió en profunditat.

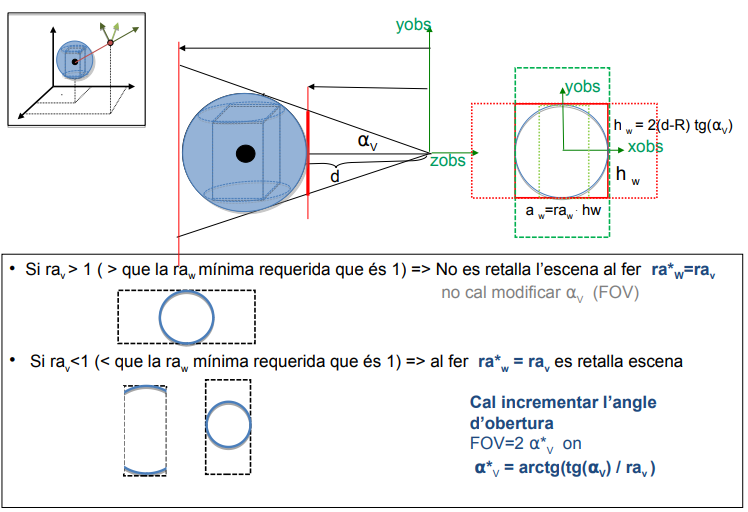
Per aprofitar al màxim la pantalla i el viewport, el window de la càmera s’ha d’ajustar per veure tota l’escena; una aproximació és ajustar el window per veure l’esfera engoblant.

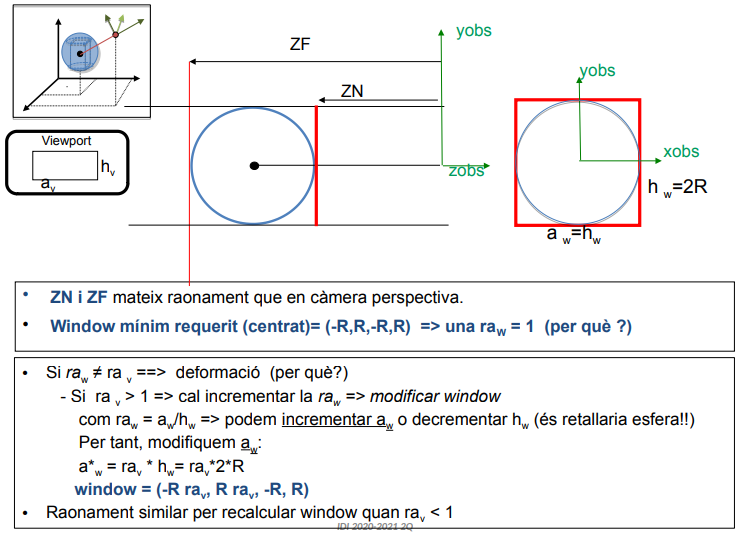
* R = d sin (αv); αv = arc sin (R / d) => FOV = 2\* αv
* Com el window està situat en ZN, αv determina que la seva alçada sigui:

hw = 2 (d – R) tg (αv)

raw  = aw / hw = 1 (αH hauria de ser igual a αV per assegurar que l’esfera no resulti retallada).





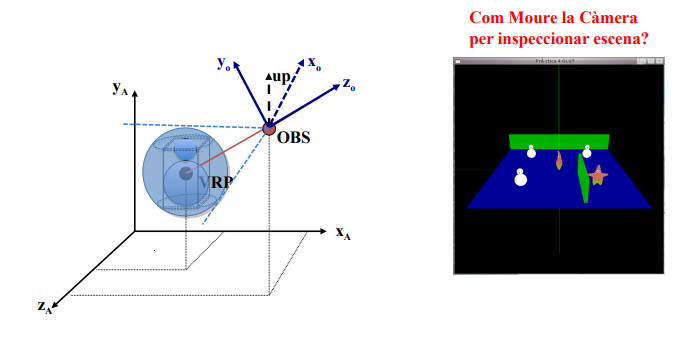


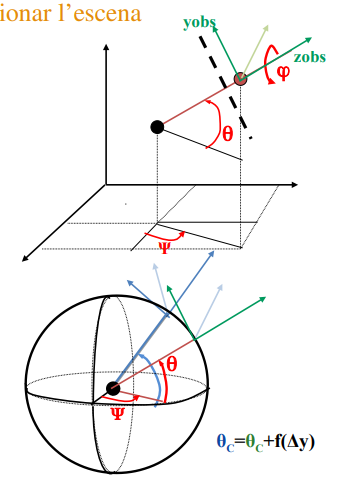
**Moure càmera (mode inspecció)**

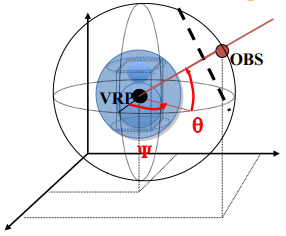
Posicionament: OBS, VRP, up => viewMatrix.

Òptica perspectiva: zN, zF, FOV, ra => projectMatrix.

Càmera en 3ra persona: posició inicial.



Moure la càmera per inspeccionar l’escena:



Els angles d’Euler determinen la posició d’un punt de l’esfera.

Des de la interfície d’usuari desplacem el cursor dreta/esquerra a (Ψ) i pujar/baixar (θ); per moure OBS sobre l’esfera.

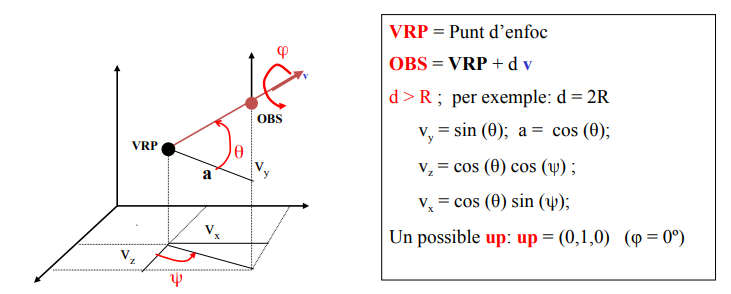
No cal canviar l’òptica.

Com calculem OBS, VRP, up?

VM = lookAt (OBS, VRP, up);

viewMatrix (VM);

Càlcul VRP, OBS a partir dels angles d’Euler:



L’òptica no cal modificar-la perquè ens movem sobre l’esfera que envolta l’escena de radi d.

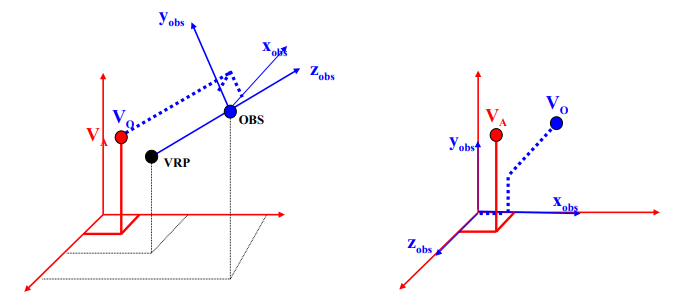
Es podria calcular la ViewMatrix directament a partir dels angles?

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

ψ en [-180,180], θ en [-90,90]

positius quan movem la càmera cap → i quan la movem cap ↑

**Càlcul de View Matrix amb càmera especificada amb angles d’Euler**

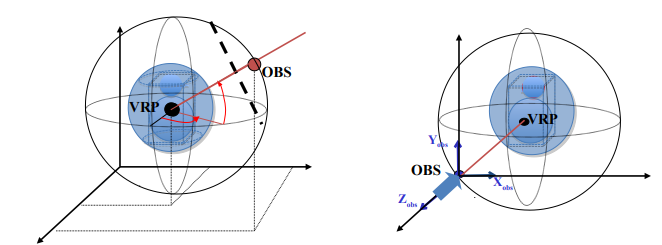
****

La viewMatrix serveix per tenir la posició dels vèrtexs respecte del SCO:

Vo = VM \* VA

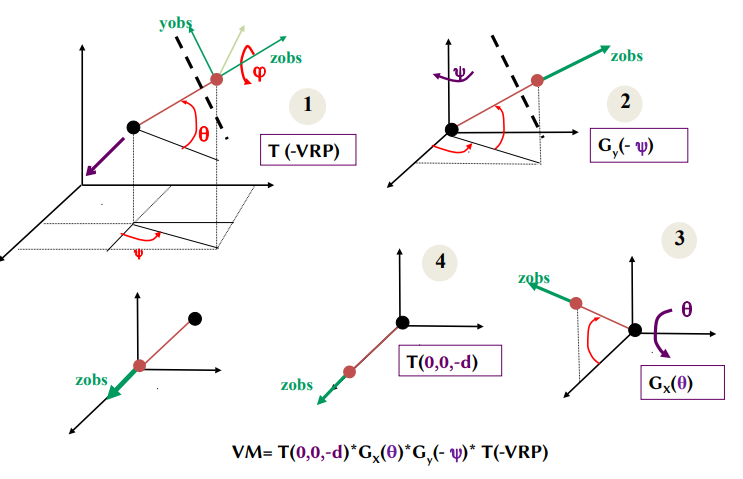
Es pot calcular la VM:

1. Pensant que movem la càmera (OBS, VRP, Up).
2. Pensat que tenim una càmera fixa al SCA i ubiquem l’escena respecte d’ella => realitzar una mateixa TGVM a tots els objectes. Si vèrtexs queden respecte a la càmera en la mateixa posició => TGVM serà igual a la VM calculada amb OBS, VRP i Up.

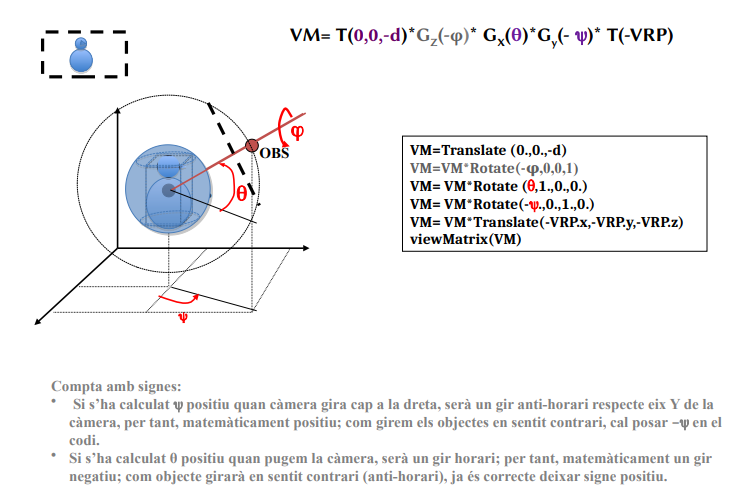


Ho podeu pensar com si girem l’esfera per a què la seva posició respecte la càmera per defecte sigui la mateixa. Agafar l’esfera i posicionar-la.

Noteu que zobs passarà a ser coincident amb zA (SCO i SCA coincidiran).

Pensarem el moviment tenint en compte que sabem calcular matriu de gir només si girem entorn d’eixos que passen per l’origen de coordenades.

Exemple: posicionament amb angles Euler:



Resum final: com visualitzar l’escena ?

