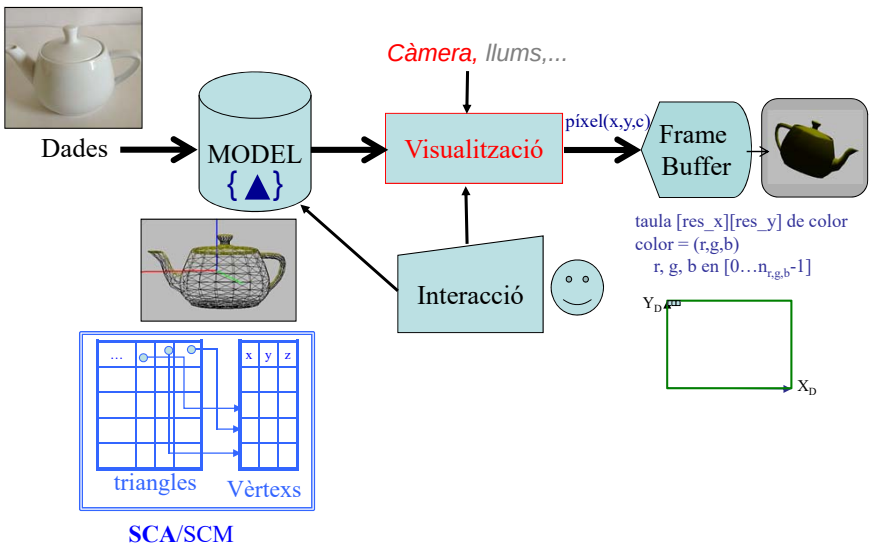
Classe 3: Contingut

- Especificació de càmera
- Exemple
- El procés de visualització projectiu
- Processament de vèrtexs:
  - Seqüència de processos/etapes
  - Matrius requerides
  - El vertex shader
- Processament de fragments:
  - El fragment shader

Visualització: Introducció

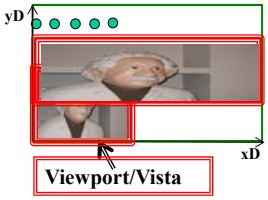


1. Posició, orientació
2. Òptica

3. Fer la Foto

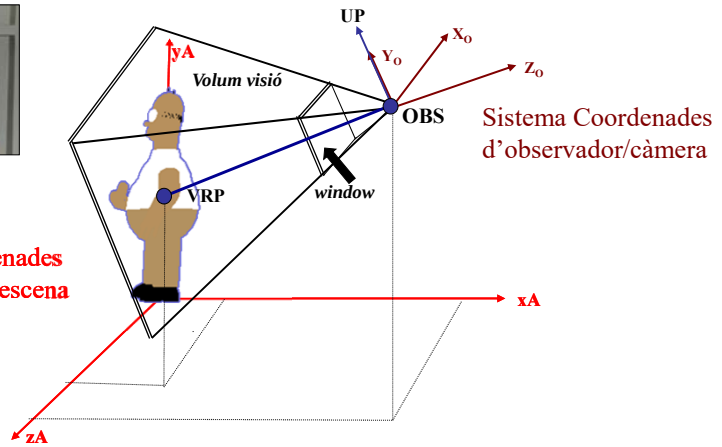
4. Emmarcar

Finestra OpenGL (Pantalla)



- Usualment, el viewport és tota la finestra OpenGL
- De moment, no ens preocuparem de si hi ha "deformacions"

Com especificar la càmera virtual?

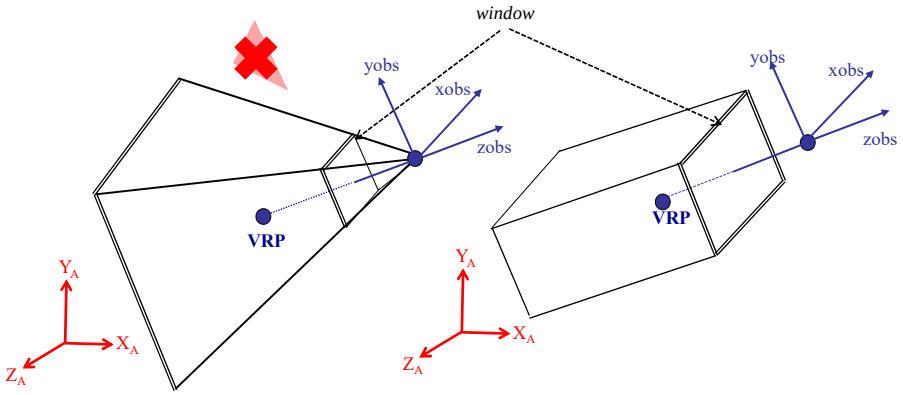


Sistema Coordenades d'observador/càmera

Sistema Coordenades Aplicació/món/escena

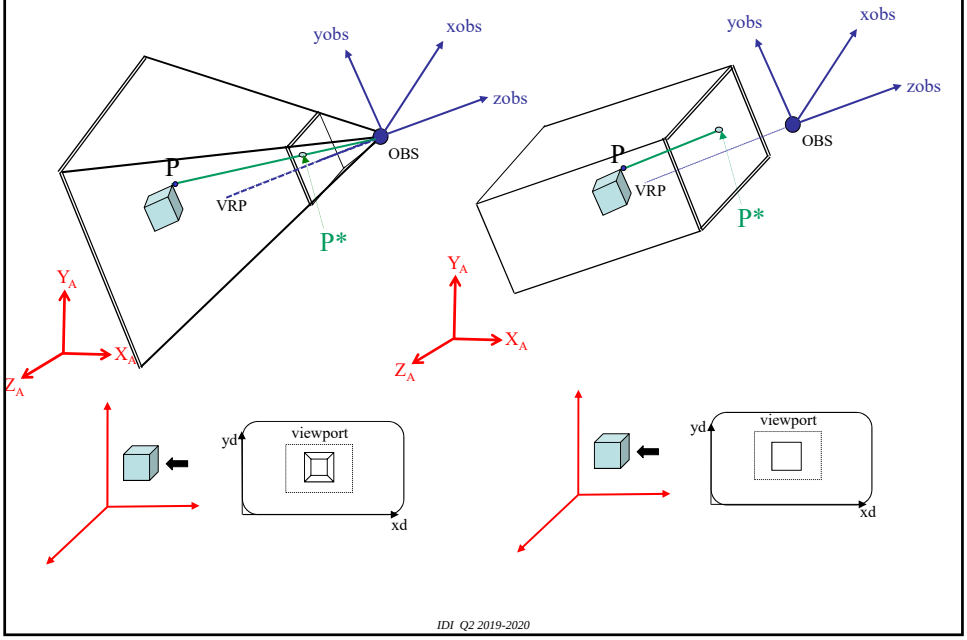
1. Ubicació càmera respecte SCA: obs, vrp, up
2. Definir òptica: Volum de Visió ➔ window, zNear, zFar

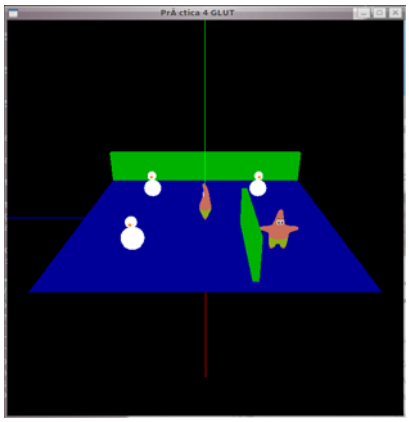
Òptica: perspectiva o ortogonal

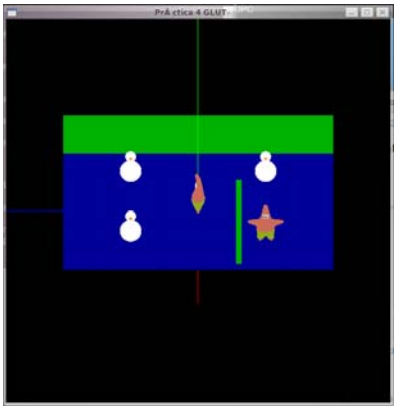


Definir òptica: volum de visió ➔(window, zNear, zFar)

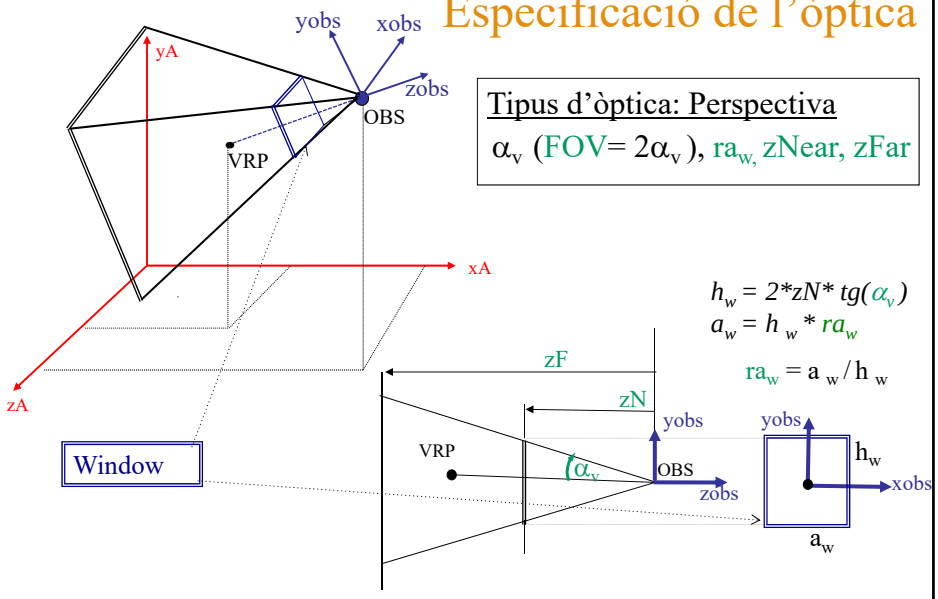
Projecció: perspectiva o ortogonal







Especificació de l'òptica



Tipus d'òptica: Perspectiva

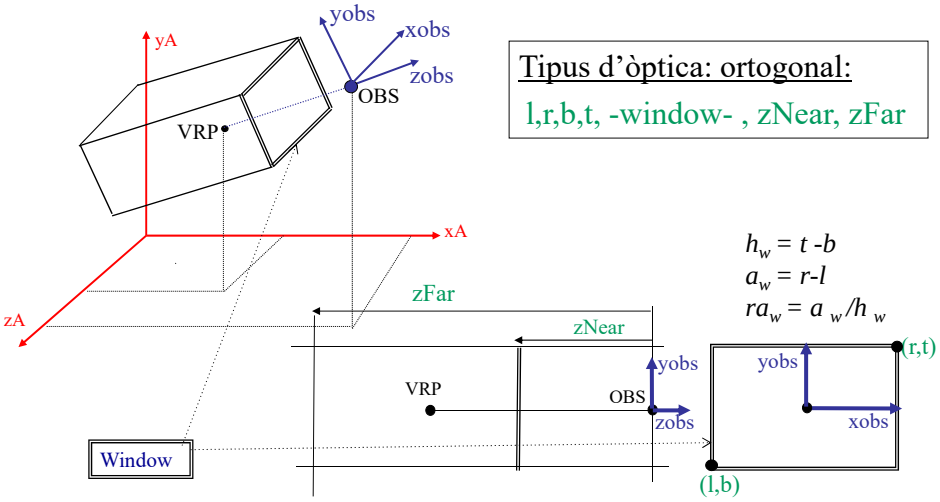
α_v (FOV= $2\alpha_v$), ra_w, zNear, zFar

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

$a_w = h_w * ra_w$

$ra_w = a_w / h_w$

Especificació de l'òptica



Tipus d'òptica: ortogonal:

l,r,b,t, -window- , zNear, zFar

$h_w = t - b$

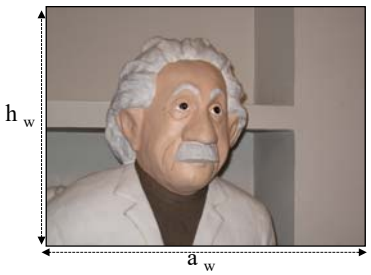
$a_w = r - l$

$ra_w = a_w / h_w$

IDI Q2 2019-2020

Sobre la relació d'aspecte del window i del viewport

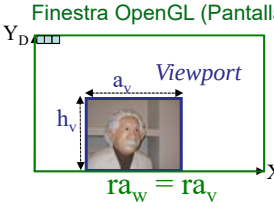
Window de l'òptica



Per a no tenir deformació en la imatge

$ra_W = ra_V$

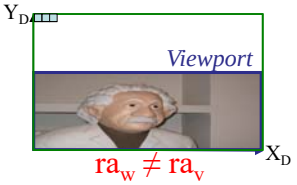
Finestra OpenGL (Pantalla)



$ra_V = a_V / h_V$

$ra_W = ra_V$

```
glViewport (ox, oy, av, hv)
```



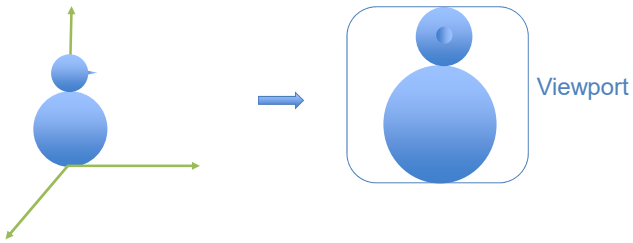
$ra_W \neq ra_V$

IDI Q2 2019-2020

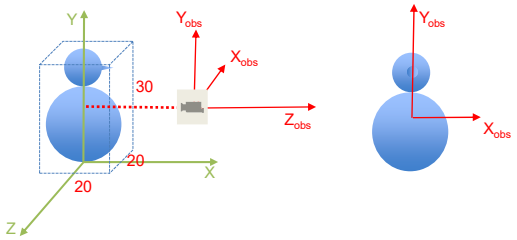
Classe 3: Contingut

- Especificació de càmera
- **Exemple**
- El procés de visualització projectiu
- Processament de vèrtexs:
  - Seqüència de processos/etapes
  - Matrius requerides
  - El vertex shader
- Processament de fragments:
  - El fragment shader

Exemple 1: Donada una funció `pinta_ninot()` que envia a visualitzar el VAO d'un objecte com el de la figura que està format per: una esfera de radi 10 amb centre (0,10,0), una altra esfera de radi 5 amb centre (0,25,0), i un con de base centrada en (2.5, 25,0), radi 2 i llargada 5 orientat segons l'eix X^+
Indica tots els paràmetres d'una càmera que permeti obtenir una imatge similar a la mostrada en l'esquema de la Figura. El viewport de 600x600 ocupa tota la finestra gràfica.

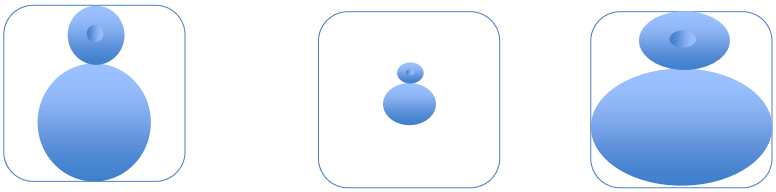


Exemple 1. Posició, orientació de la càmera



VRP=(0,15,0)
OBS=(30,15,0)
Up=(0,1,0)

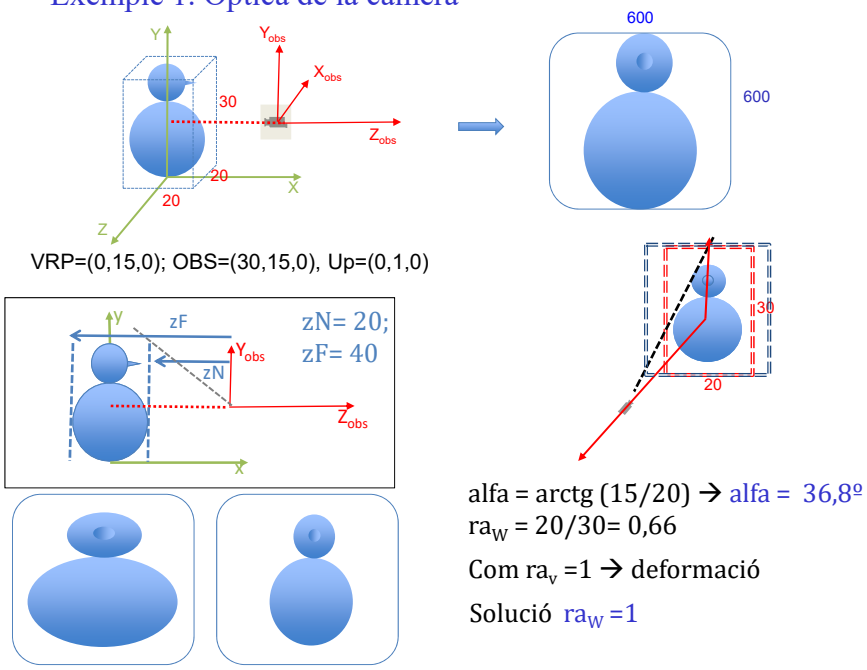
Exemple 1. Òptica de la càmera



Restriccions:

- Viewport ocupi tota la finestra gràfica 600x600
- Ninot centrat
- Ninot optimitzi espai en viewport
- Sense deformacions

Exemple 1. Òptica de la càmera



alfa = arctg (15/20) → alfa = 36,8º

ra_W = 20/30= 0,66

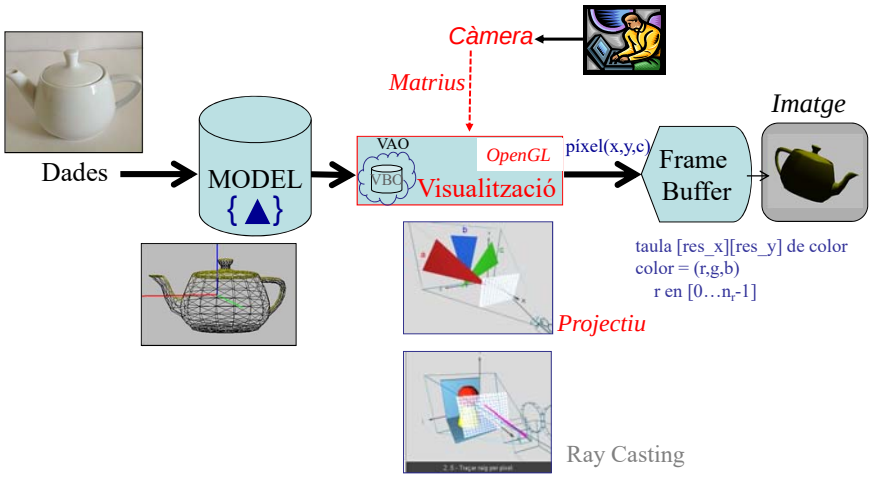
Com ra_v =1 → deformació

Solució ra_W =1

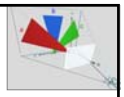
Classe 3: Contingut

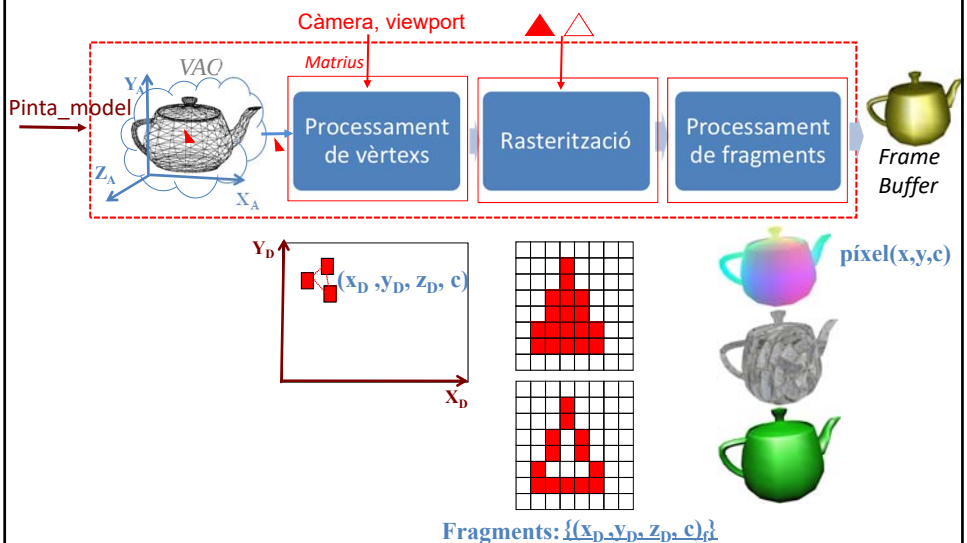
- Especificació de càmera
- Exemple
- **El procés de visualització projectiu**
- Processament de vèrtexs:
  - Seqüència de processos/etapes
  - Matrius requerides
  - El vertex shader
- Processament de fragments:
  - El fragment shader

Visualització: Introducció (2)

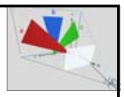


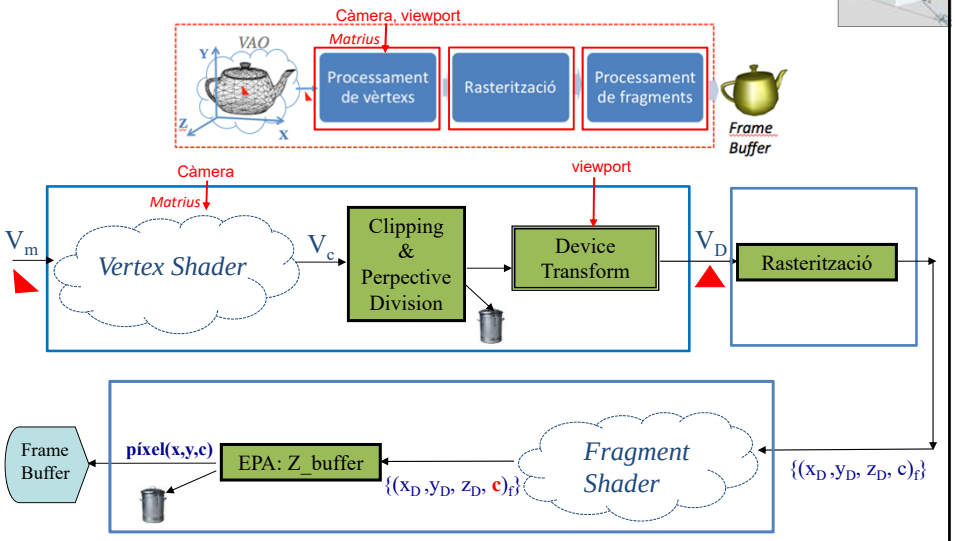
Pintar/visualitzar en OpenGL 3.3 (1)





Pintar/visualitzar en OpenGL 3.3 (2)

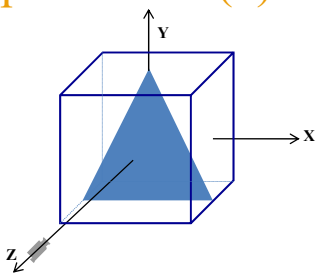




Pintar/visualitzar en OpenGL 3.3 (3)

Vertex Shader

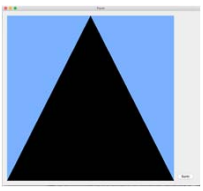
```
#version 330 core
in vec3 vertex;

void main()  {
 gl_Position = vec4 (vertex, 1.0);
}
```



Volum de Visió cub de (-1,-1,-1) a (1,1,1)

Fragment Shader

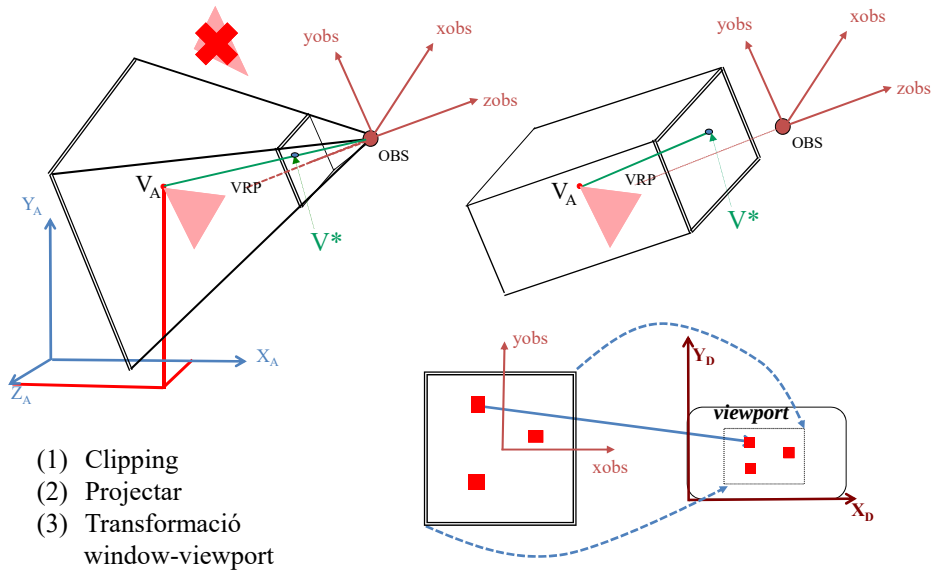
```
#version 330 core
out vec4 FragColor;

void main() {
 FragColor = vec4(0, 0, 0, 1);
}
```



Classe 3: Contingut

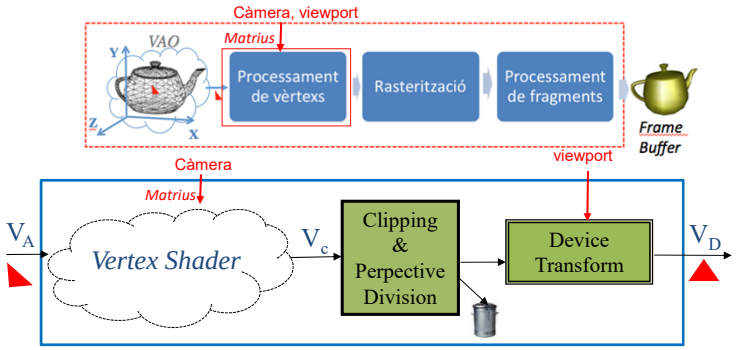
- Especificació de càmera
- Exemple
- El procés de visualització projectiu
- **Processament de vèrtexs:**
  - **Seqüència de processos/etapes**
  - **Matrius requerides**
  - **El vertex shader**
- Processament de fragments:
  - El fragment shader

Processament "manual" de vèrtexs



(1) Clipping
(2) Projectar
(3) Transformació window-viewport

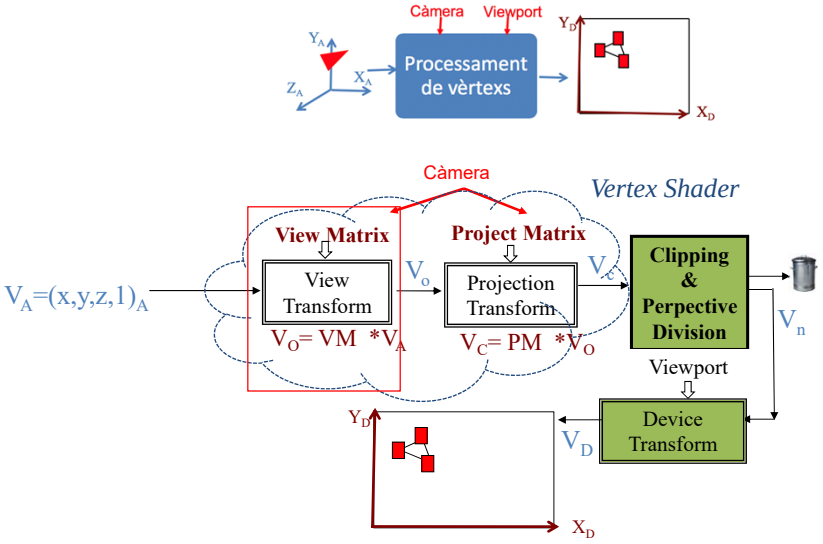
Processament de vèrtexs en OpenGL 3.3 (1)



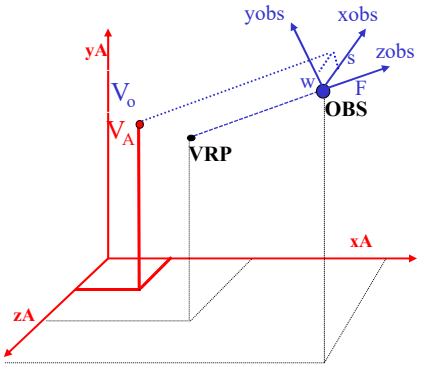
Condició per a que un vèrtex sigui interior al volum de visió

$$-w_c \leq x_c \leq w_c$$
$$-w_c \leq y_c \leq w_c$$
$$-w_c \leq z_c \leq w_c$$

Processament de vèrtexs en OpenGL 3.3 (2)



Pas 1: vèrtex en SCO. OBS, VRP, Up → Càlcul de la viewMatrix



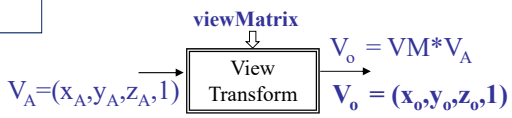
$$VM = \begin{bmatrix} s.x & s.y & s.z & 0 \\ w.x & w.y & w.z & 0 \\ F.x & F.y & F.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & -Obs.x \\ 0 & 1 & 0 & -Obs.y \\ 0 & 0 & 1 & -Obs.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

F = OBS - VRP = (F.x,F.y,F.z) F= F / || F ||

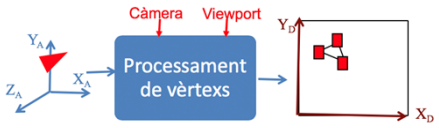
s = Up x F s = s / ||s||

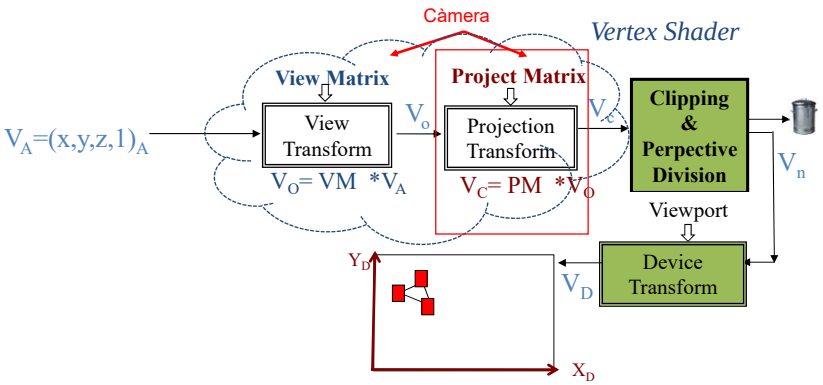
w = F x s

```
VM = lookAt(OBS,VRP,Up);
viewMatrix(VM);
```

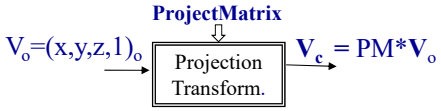


Pas 2: vèrtex en Coord Clipping. Òptica →Càlcul de la projectMatrix





Pas 2: La projectMatrix



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

a=2/(r-l) b=2/(t-b)
c= 2 /(zf-zn)
d = (zn+zf)/ (zf-zn)

Òptica Ortogonal

$V_c = (x_c, y_c, z_c, w_c)$ on $w_c=1$

```
PM=ortho(l,r,b,t,zN,ZF);
projectMatrix(PM);
```

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

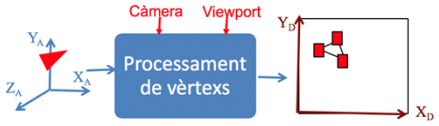
a = tg (FOV/2)
c = (zf+zn)/(zn-zf)
d = 2*zn *zf /(zn-zf)

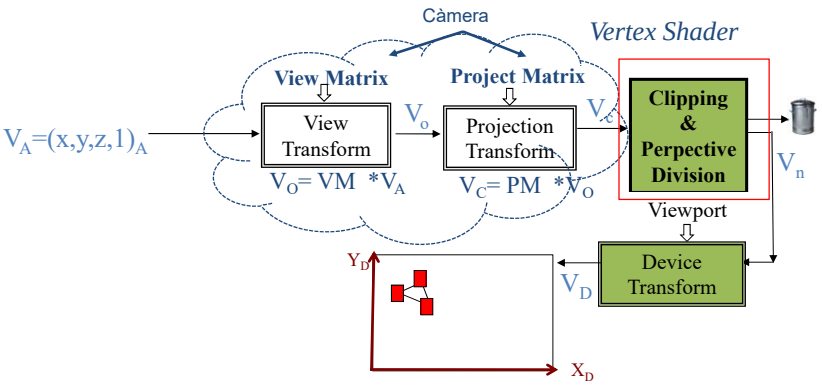
Òptica Perspectiva

$V_c = (x_c, y_c, z_c, w_c)$ on $w_c=-z_o$

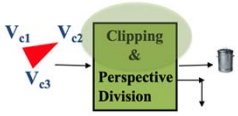
```
PM=perspective(FOV,ra,zN,ZF);
projectMatrix(PM);
```


Pas 3: Clipping i projecció





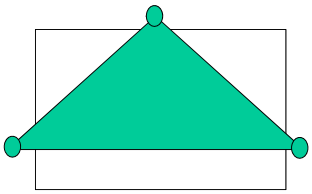
Pas 3.1 : Clipping

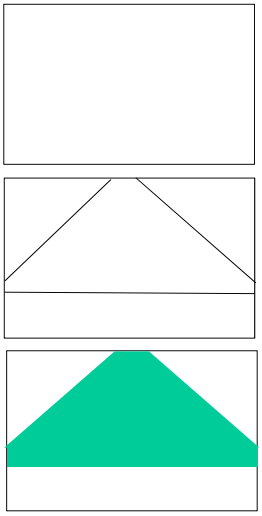


Condició per a que un Vèrtex sigui interior al volum de visió:

$$-w_c \leq x_c \leq w_c$$
$$-w_c \leq y_c \leq w_c$$
$$-w_c \leq z_c \leq w_c$$

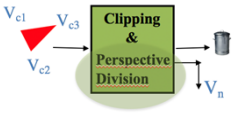
$V_c = (x_c, y_c, z_c, w_c)$ on $w_c=1$ en ortogonal
$V_c = (x_c, y_c, z_c, w_c)$ on $w_c=-z_o$ en perspectiva





IDI Q2 2019-2020

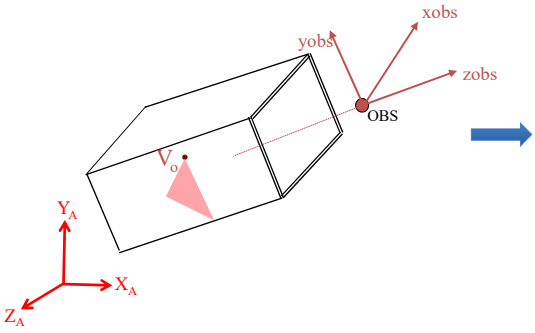
Pas 3.2 : Projecció. Òptica ortogonal (1)

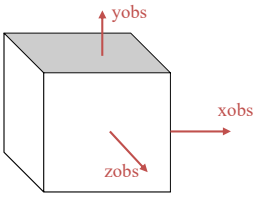


$$PM = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

a=2/(r-l) b=2/(t-b)
c= 2 /(zf-zn)
d = (zn+zf)/ (zf-zn)

$V_c = (x_c, y_c, z_c, w_c)$ on $w_c=1$

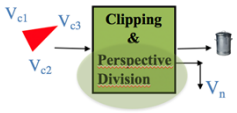


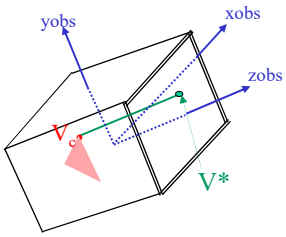


$$-1 \leq x_c \leq 1$$
$$-1 \leq y_c \leq 1$$
$$-1 \leq z_c \leq 1$$

IDI Q2 2019-2020

Pas 3.2 : Projecció. Òptica ortogonal (2)





$V_c = (x_c, y_c, z_c, w_c)$ on $w_c = 1$

$-1 \le x_c \le 1$
$-1 \le y_c \le 1$
$-1 \le z_c \le 1$

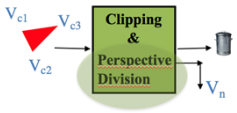
Vèrtex projectat:
$V_x^* = V_{cx}$ $V_y^* = V_{cy}$

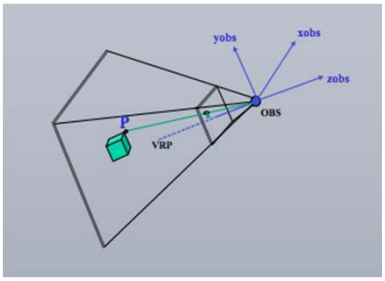
$V^* = V_c / w_c \rightarrow V_n$

$-1 \le x_n \le 1$
$-1 \le y_n \le 1$
$-1 \le z_n \le 1$

V_z^* per càlculs posteriors i indica distància a window

Pas 3.2 : Projecció. Òptica perspectiva



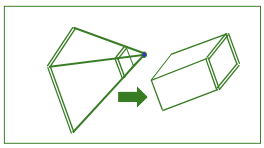


Vèrtex projectat:
$V^* = V_c / w_c = -V_c / z_o$
$x^* = -x_c / z_o$ $y^* = -y_c / z_o$ $z^* = -z_c / z_o$

Inversament proporcional a distància a observador ☺

$V_c = (x_c, y_c, z_c, w_c)$ on $w_c = -z_o$

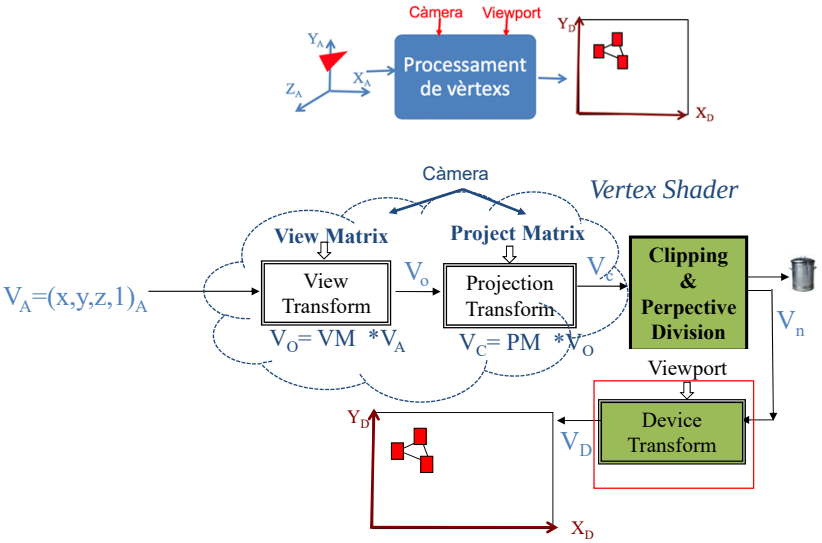
$-w_c \le x_c \le w_c$
$-w_c \le y_c \le w_c$
$-w_c \le z_c \le w_c$



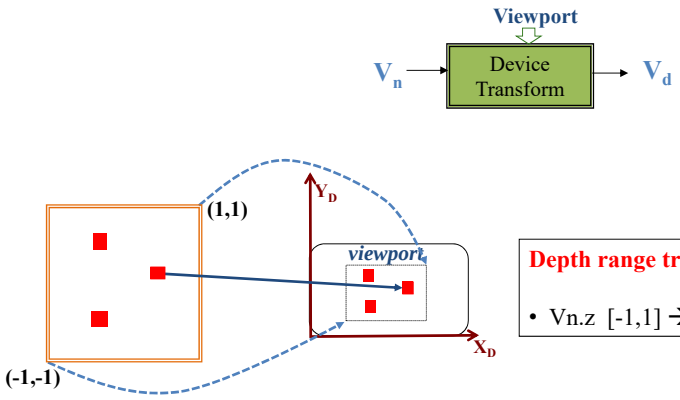
$V^* = V_c / w_c \rightarrow V_n$

$-1 \le x_n \le 1$
$-1 \le y_n \le 1$
$-1 \le z_n \le 1$

Pas 4: Transformació a coordenades de dispositiu



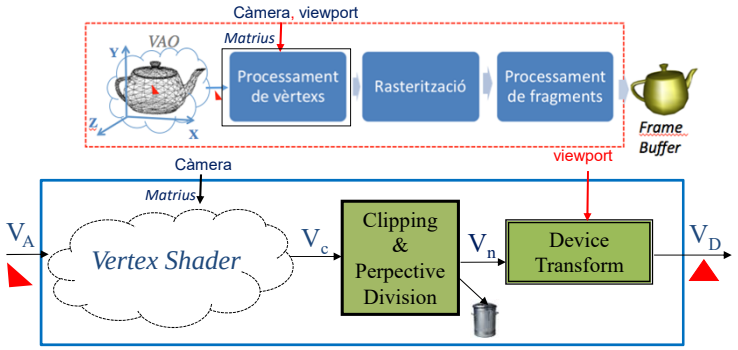
Pas 4: Transformació a coordenades de dispositiu



Depth range transformation

- Vn.z [-1,1] → [0,1], per defecte.

Processament de vèrtexs en OpenGL 3.3 (3)

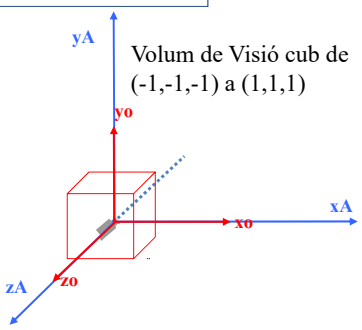


Processat de vèrtex: El vèrtex Shader

Vertex Shader

```
#version 330 core

in vec3 vertex;
uniform mat4 PM;
uniform mat4 VM;

void main()  {
   gl_Position = PM*VM*vec4 (vertex, 1.0);
}
```

```
#version 330 core
in vec3 vertex;
//uniform mat4 PM; equivalent a identitat
//uniform mat4 VM; equivalent a identitat
void main()  {
   //gl_Position = PM*VM*vec4 (vertex, 1.0);
   gl_Position = vec4 (vertex, 1.0);
}
```

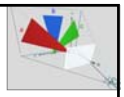


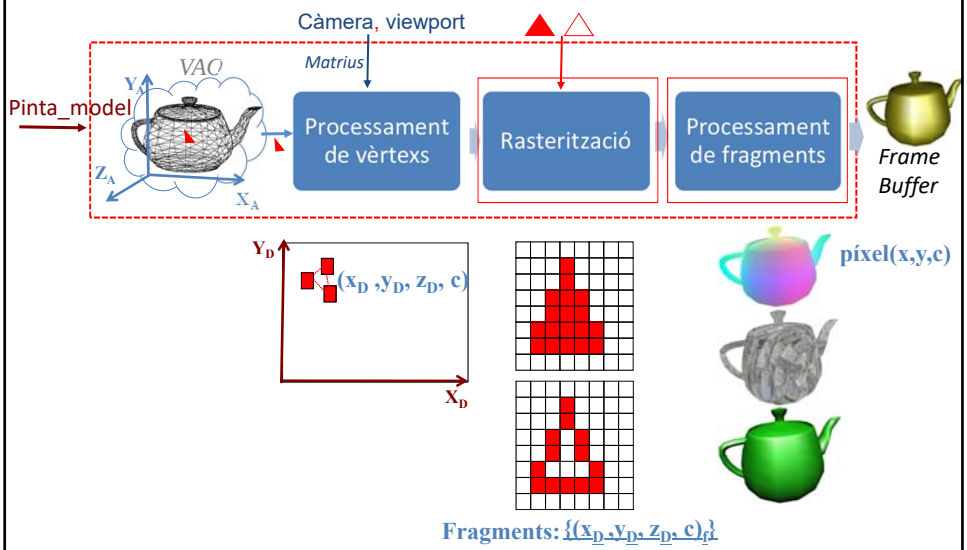
Volum de Visió cub de (-1,-1,-1) a (1,1,1)

Classe 3: Contingut

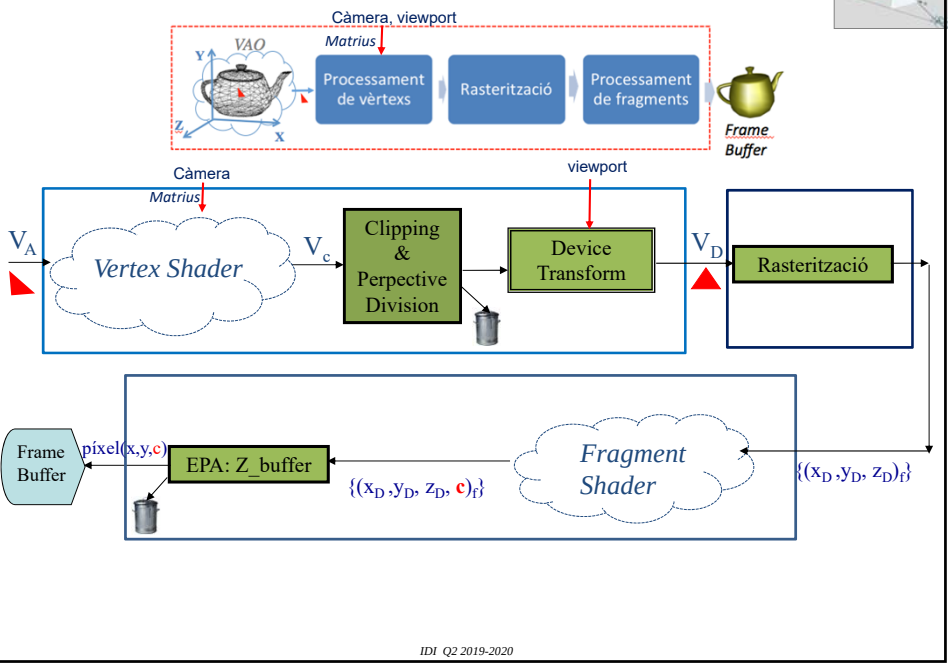
- Especificació de càmera
- Exemple
- El procés de visualització projectiu (breu repàs)
- Processament de vèrtexs:
  - Seqüència de processos/etapes
  - Matrius requerides
  - El vertex shader
- **Processament de fragments:**
  - **El fragment shader**

Pintar/visualitzar en OpenGL 3.3





Pintar/visualitzar en OpenGL 3.3

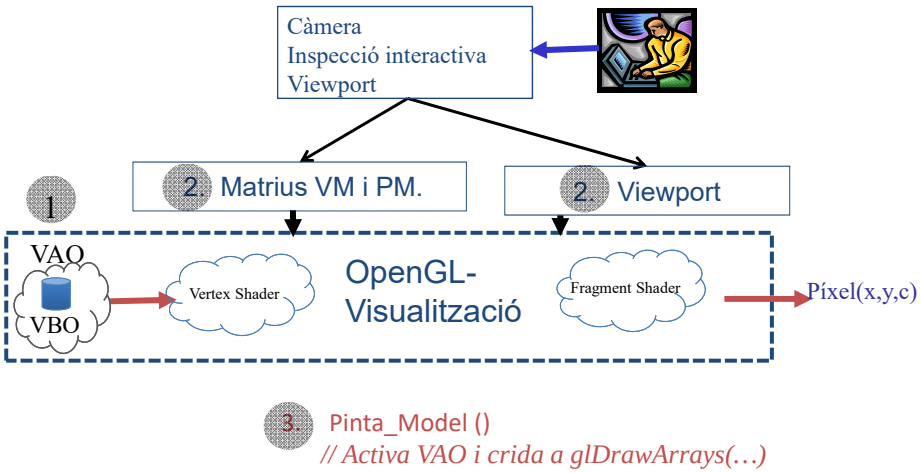


Processat de fragments: El fragment Shader

Fragment Shader

```
#version 330 core

out vec4 FragColor;

void main() {
   FragColor = vec4(0, 0, 0, 1);
}
```

Pintar/visualitzar en OpenGL 3.3 (resum)

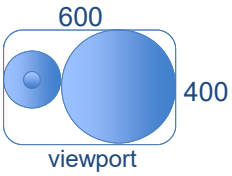


Classe 3: Conceptes i preguntes

- Paràmetres requerits per especificar posició i orientació de la càmera: VRP, OBS i Up. Entendre i saber utilitzar. Quin efecte té modificar un qualsevol dels paràmetres en la imatge final d'una escena?, pensar en cada paràmetre de manera independent.
- Pot tenir Up la direcció de visió (o sigui la direcció de la recta que uneix VRP amb OBS)?
- Paràmetres requerits per especificar l'òptica perspectiva i l'òptica ortogonal de la càmera. Entendre i saber utilitzar. Quin efecte té modificar Znear? i Zfar? i FOV? i window?
- Concepte de viewport/vista. Efecte de tenir relacions d'aspecte diferents en window i viewport.
- El procés de visualització projectiu: blocs funcionals que l'integren, ordre dels processos, sistemes de coordenades. Quantes matrius cal passar a la GPU? Com s'obtenen? Quina és la seva funció? Quan cal passar les matrius en el nostre codi?
- Si no es multiplica el vèrtex en el vertex shader per cap matriu, quina és la càmera?
- Diferència entre vèrtex i fragment.
- Què cal fer, com a mínim, en vertex shader i en fagment shader?
- Com s'obtenen les coordenades de clipping? Què son les coordenades normalitzades?

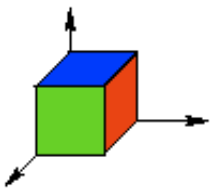
Per pensar…

- Quins paràmetres requiriria una òptica ortogonal per veure l'escena del primer exemple d'avui? Quin efecte tindria en la imatge generada?
- Quins paràmetres de posicionament de càmera per a obtenir:



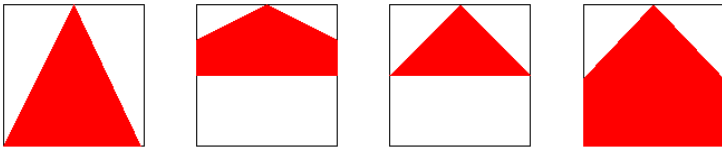
- Com fer un zoom? Quins paràmetres de la càmera modificaries?

Exemple 2. Tenim una escena amb un cub de costat 2 orientat amb els eixos i de manera que el seu vèrtex mínim està situat a l'origen de coordenades. La cara del cub que queda sobre el pla x=2 és de color vermell, la cara que queda sobre el pla z=2 és de color verd i la resta de cares són blaves.
a) Indica TOTS els paràmetres d'una càmera perspectiva que permeti veure completes a la vista només les cares vermella i verda. La relació d'aspecte del viewport (vista) és 2. Fes un dibuix indicant la imatge final que s'obtindria.
b) Quin efecte tindria en la imatge final modificar l'òptica ortogonal? Defineix la càmera ortogonal.



Exemple 3

Tenim una escena amb un triangle vermell amb vèrtexs V1=(-2,0,0), V2 = (2, 0, 0) i V3= (0, 1, 0). Suposant que tenim un viewport quadrat de 600x600 píxels, i que hem inicialitzat les matrius de càmera (view) i projecció (proj) a la matriu identitat, indica quina de les següents imatges és la que sortirà en un viewport de 600x600 (sabem que el Vertex Shader i el Fragment Shader estan correctament implementats):



a) **b)** **c)** **d)**