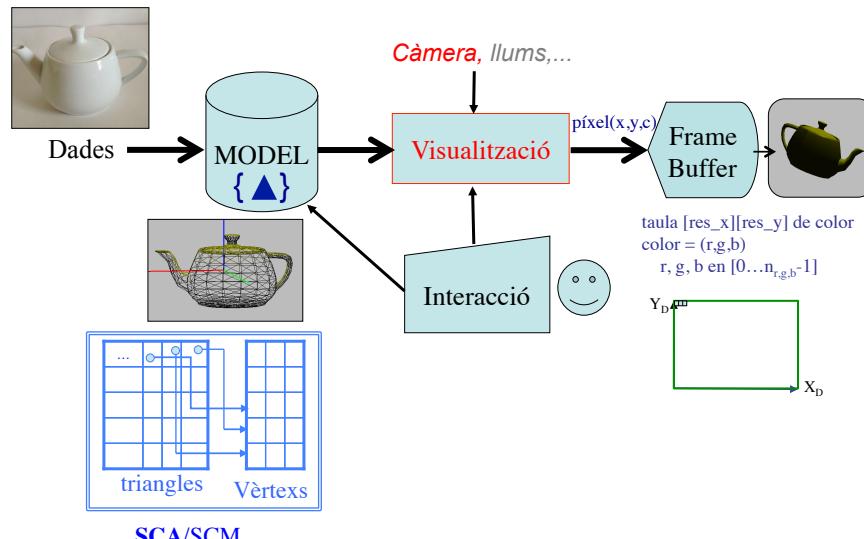


Classe 3: Contingut

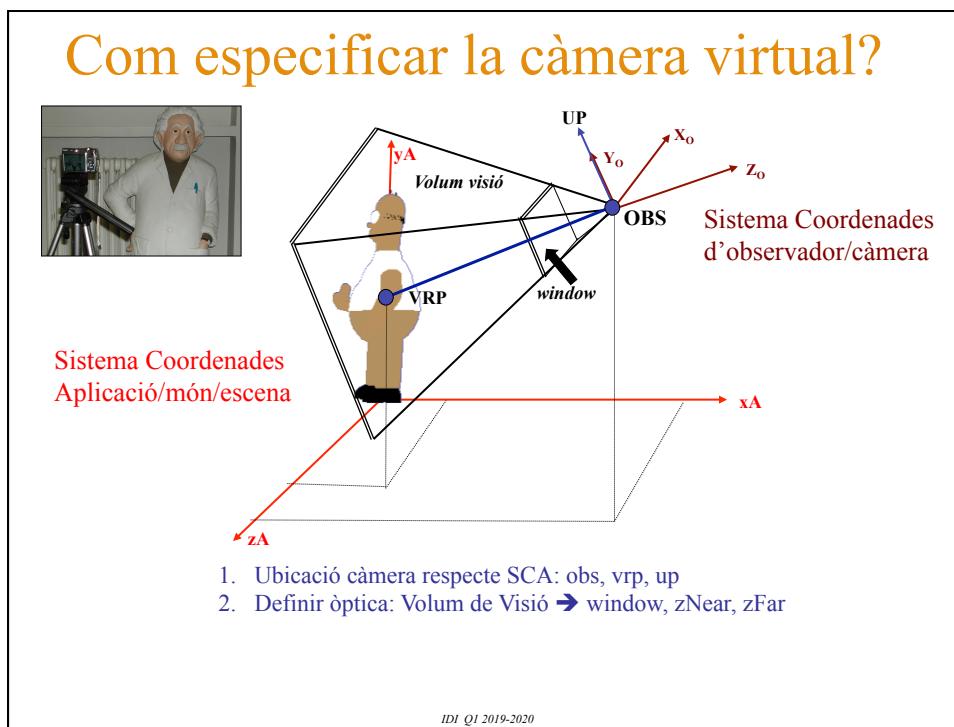
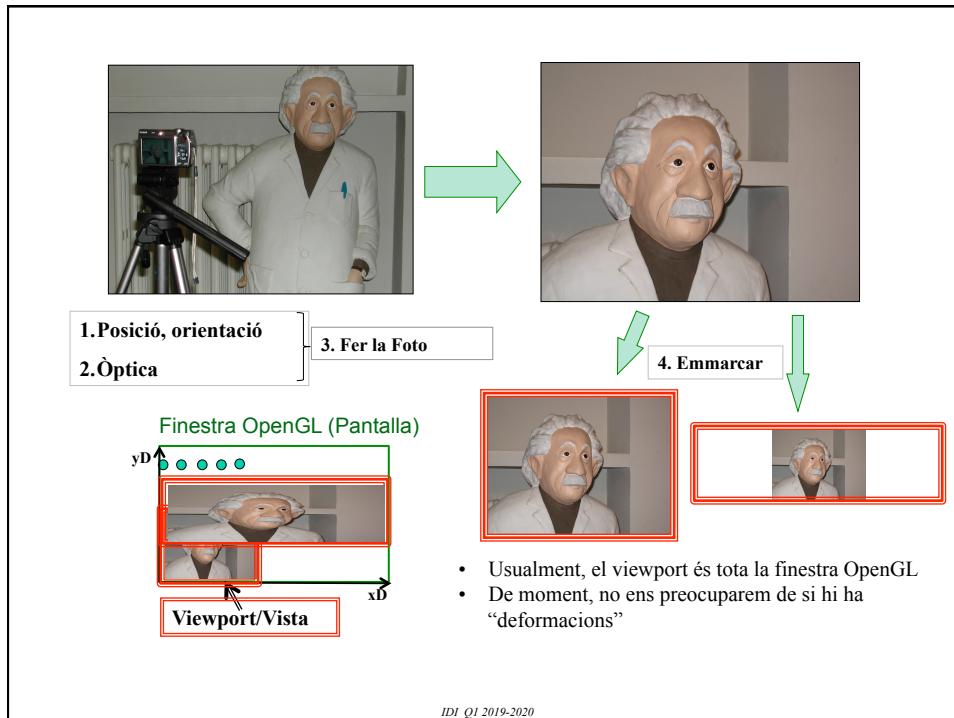
- Especificació de càmera
- Exemple
- El procés de visualització projectiu
- Processament de vèrtexs:
 - Seqüència de processos/etapes
 - Matrius requerides
 - El vertex shader
- Processament de fragments:
 - El fragment shader

IDI Q1 2019-2020

Visualització: Introducció

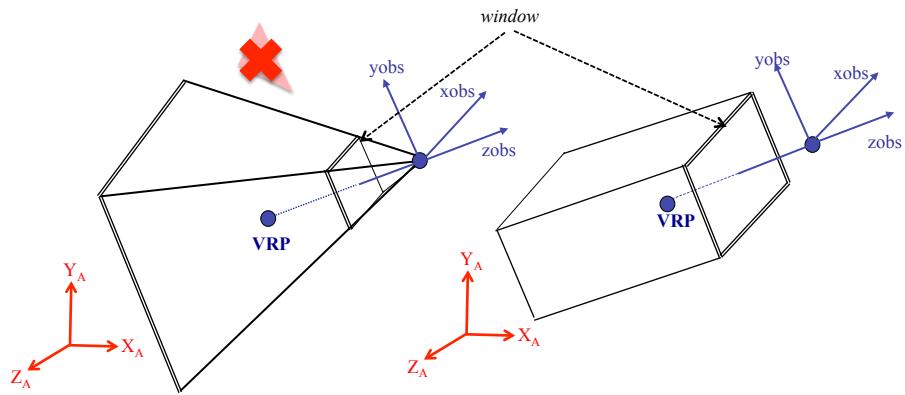


IDI Q1 2019-2020





Òptica: perspectiva o ortogonal

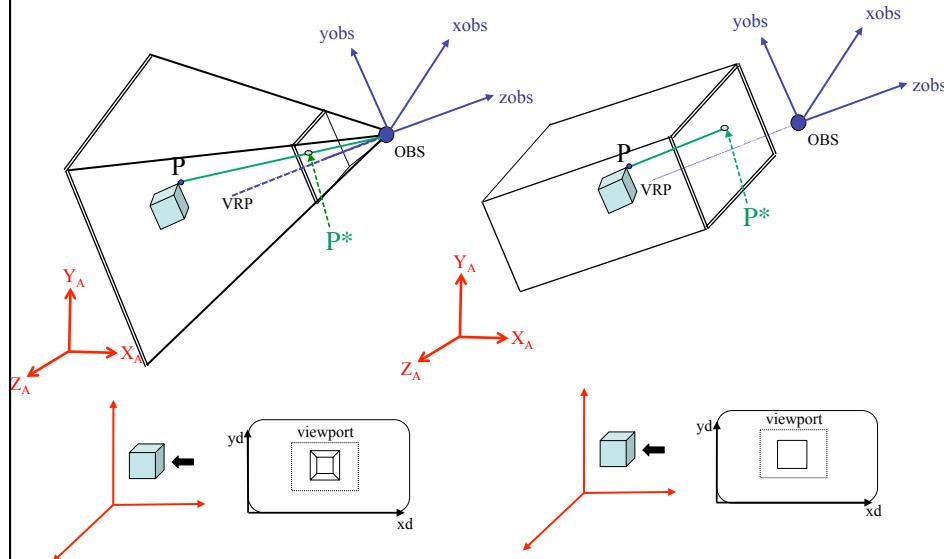


Definir òptica: volum de visió \rightarrow (window, zNear, zFar)

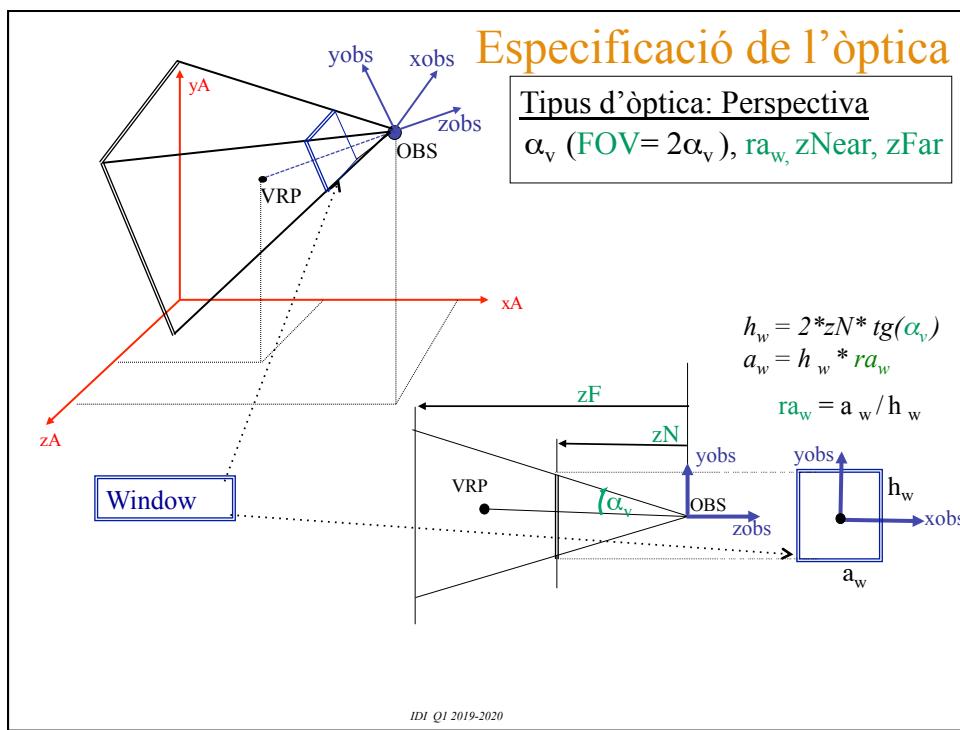
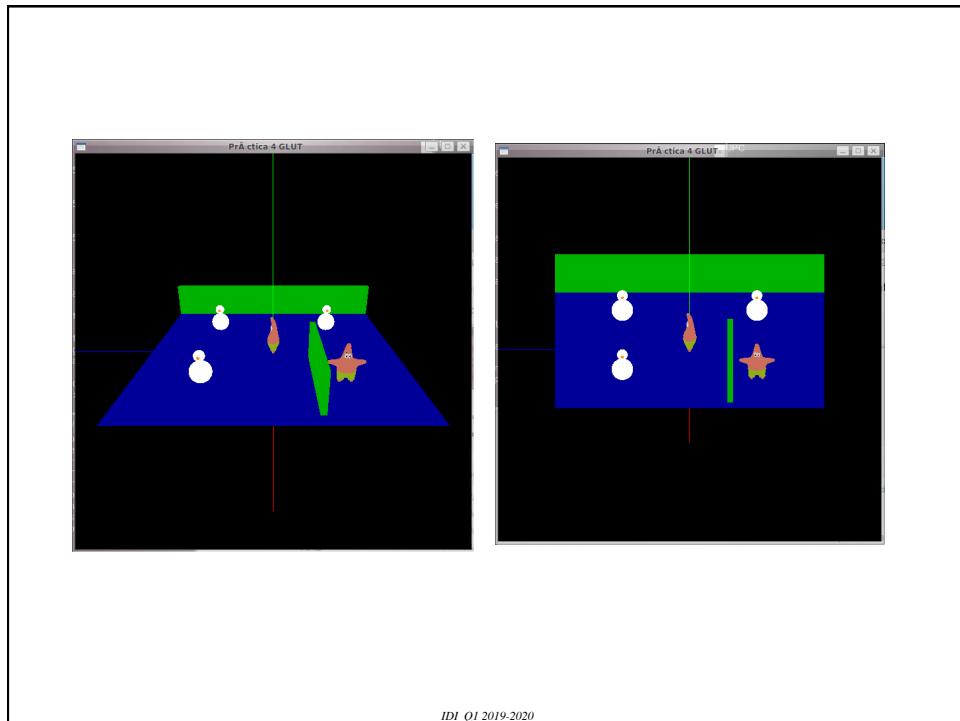
IDI Q1 2019-2020

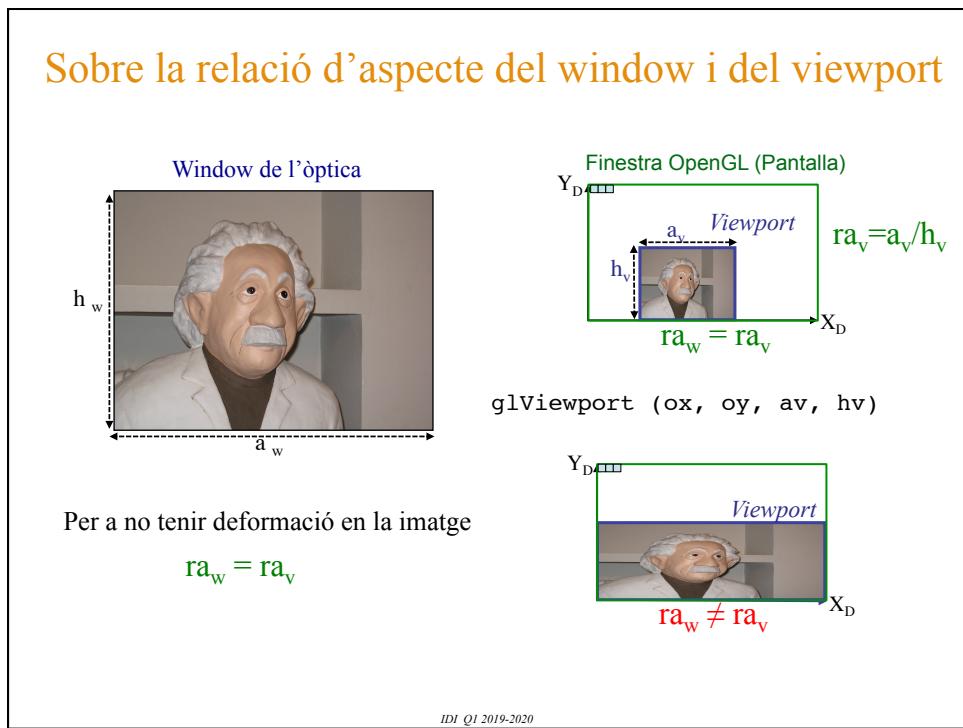
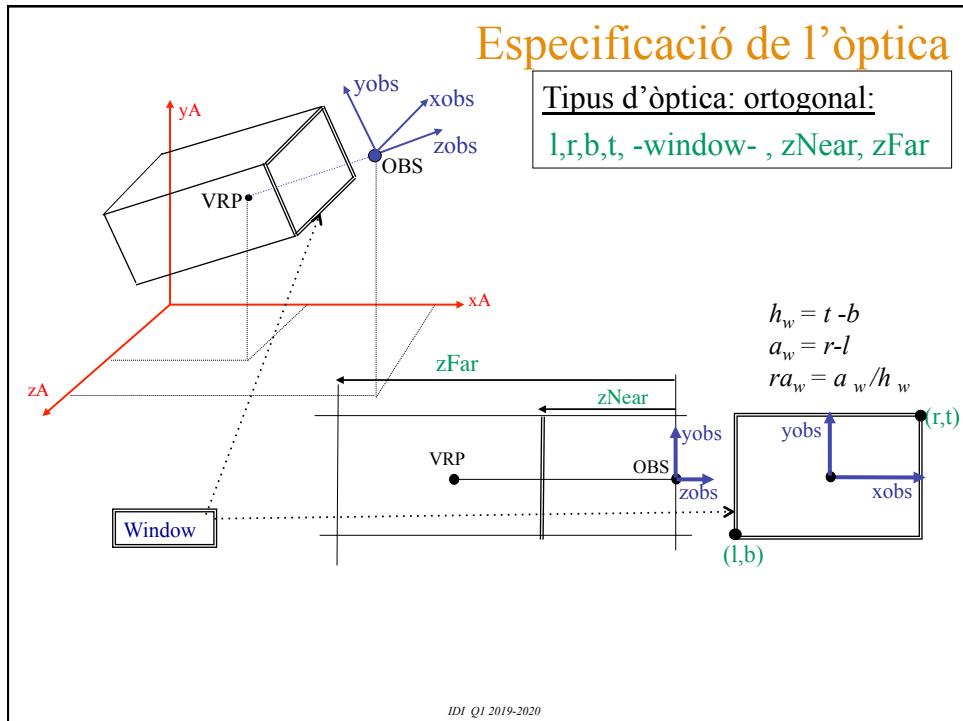


Projecció: perspectiva o ortogonal



IDI Q1 2019-2020





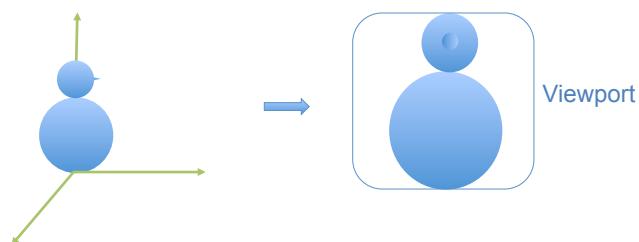
Classe 3: Contingut

- 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

IDI Q1 2019-2020

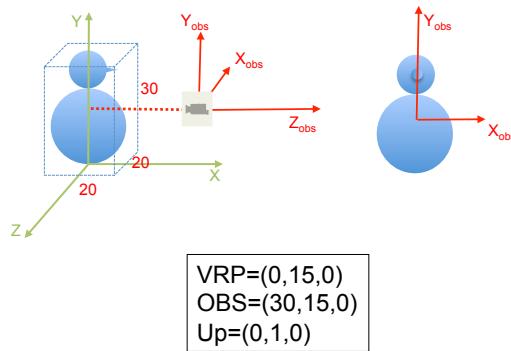
Exemple 1: Donada una funció `pinta_ninot()` que envia a visualitzar el VAO d'un objecte com el de la figura que està format per: una esfera de radi 10 amb centre (0,10,0), una altra esfera de radi 5 amb centre (0,25,0), i un con de base centrada en (2.5, 25,0), radi 2 i llargada 5 orientat segons l'eix 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.



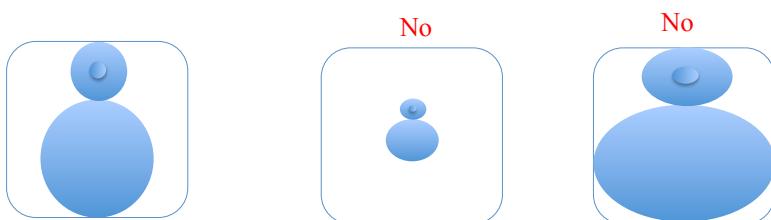
IDI Q1 2019-2020

Exemple 1. Posició, orientació de la càmera



IDI Q1 2019-2020

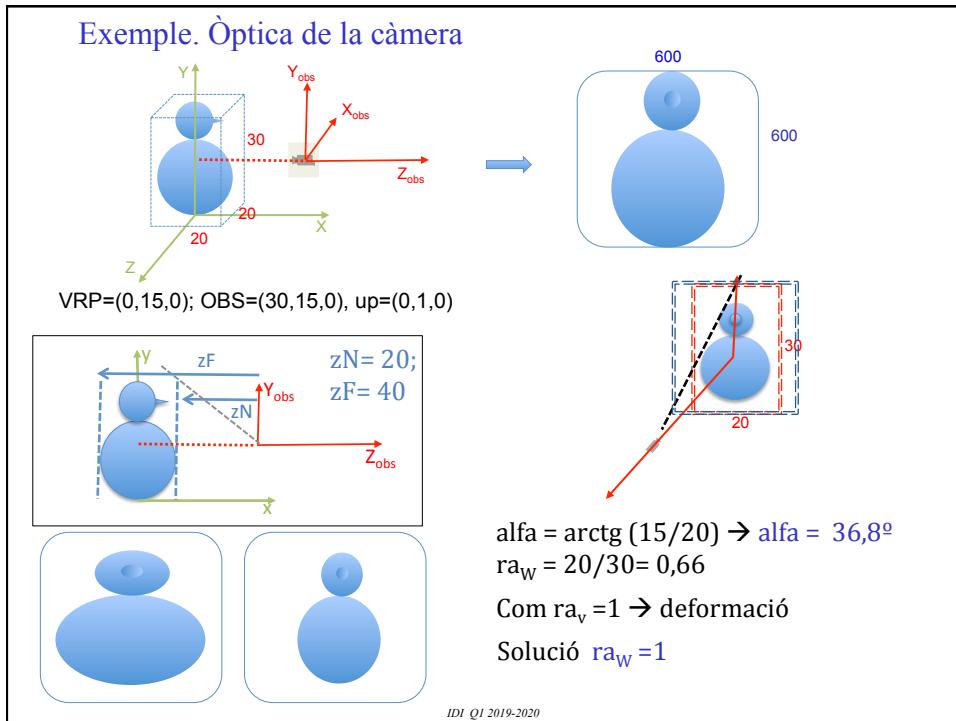
Exemple 1. Òptica de la càmera



Restriccions:

- Viewport ocupa tota la finestra gràfica
600x600
- Ninot centralitzat
- Ninot optimitza l'espai en viewport
- Sense deformacions

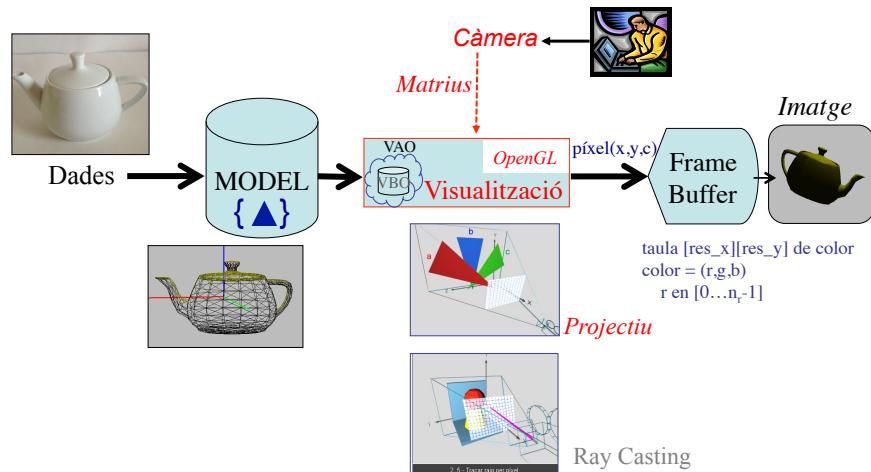
IDI Q1 2019-2020



Classe 3: Contingut

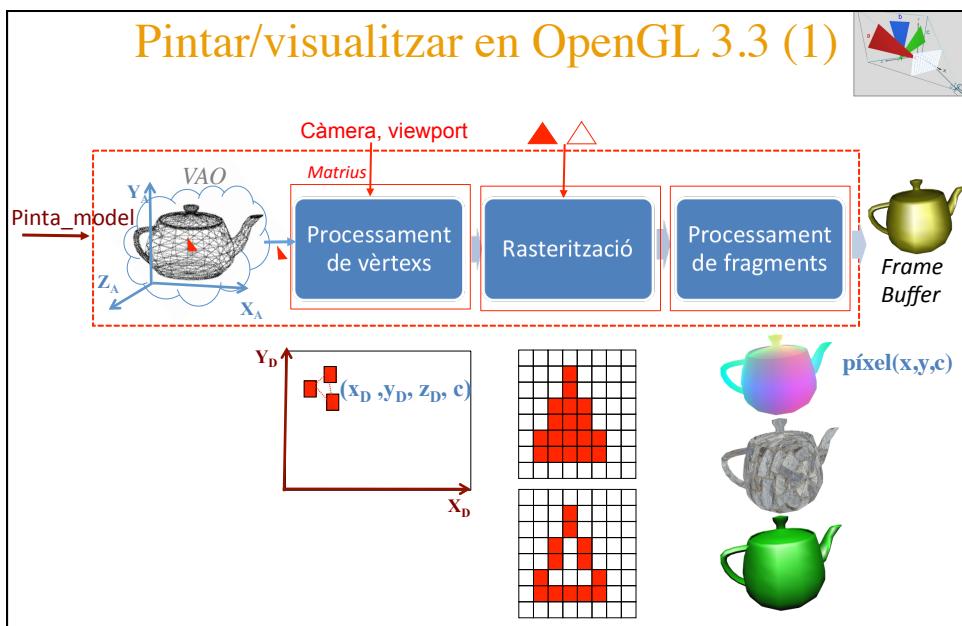
- Especificació de càmera
- Exemple
- **El procés de visualització projectiu**
- Processament de vèrtexs:
 - Seqüència de processos/etapes
 - Matrius requerides
 - El vertex shader
- Processament de fragments:
 - El fragment shader

Visualització: Introducció (2)

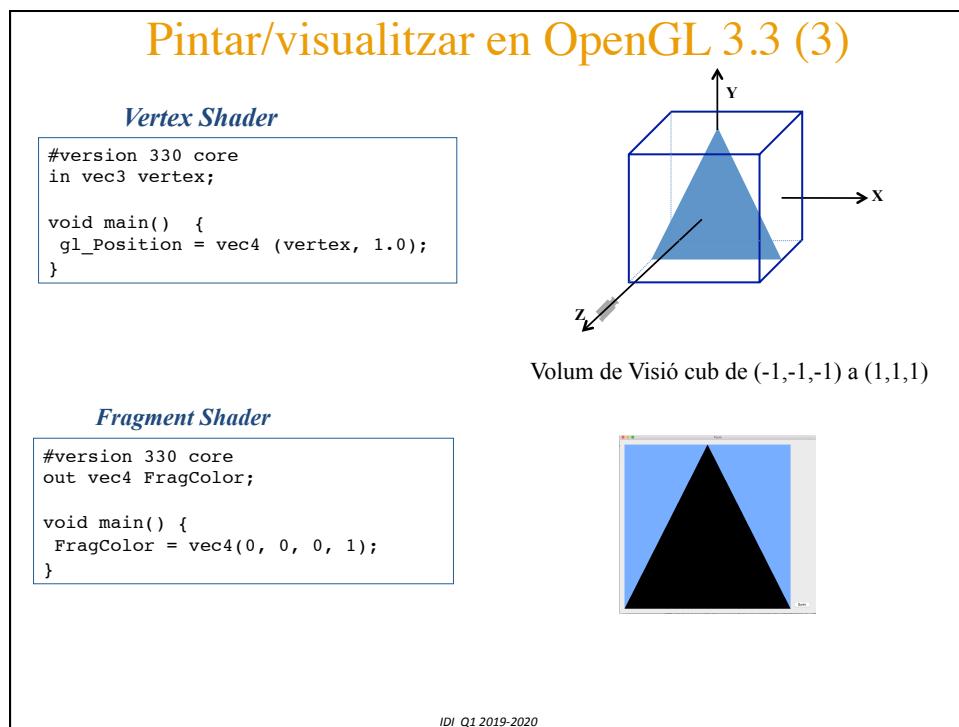
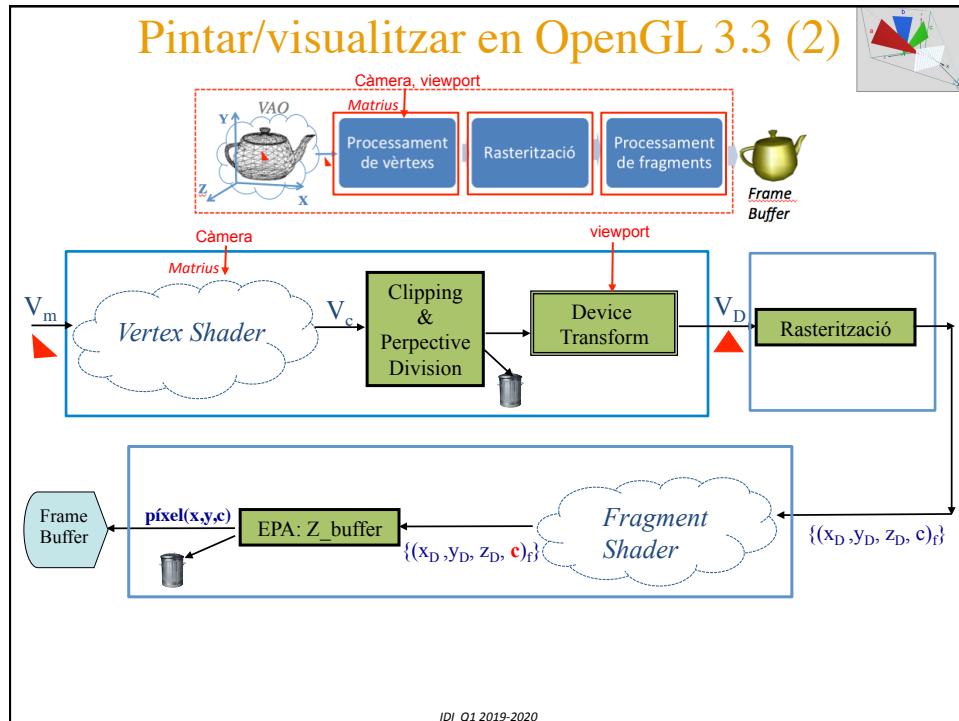


IDI Q1 2019-2020

Pintar/visualitzar en OpenGL 3.3 (1)



IDI Q1 2019-2020



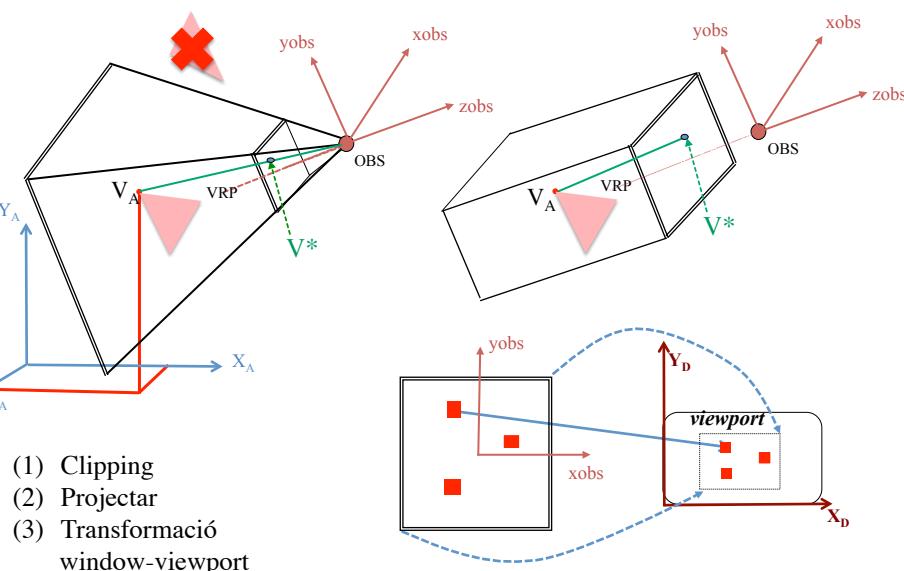
Classe 3: Contingut

- 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



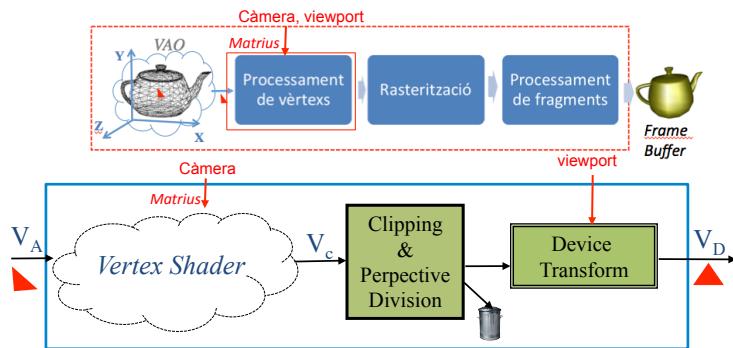
IDI Q1 2019-2020

Processament “manual” de vèrtexs



IDI Q1 2019-2020

Processament de vèrtexs en OpenGL 3.3 (1)

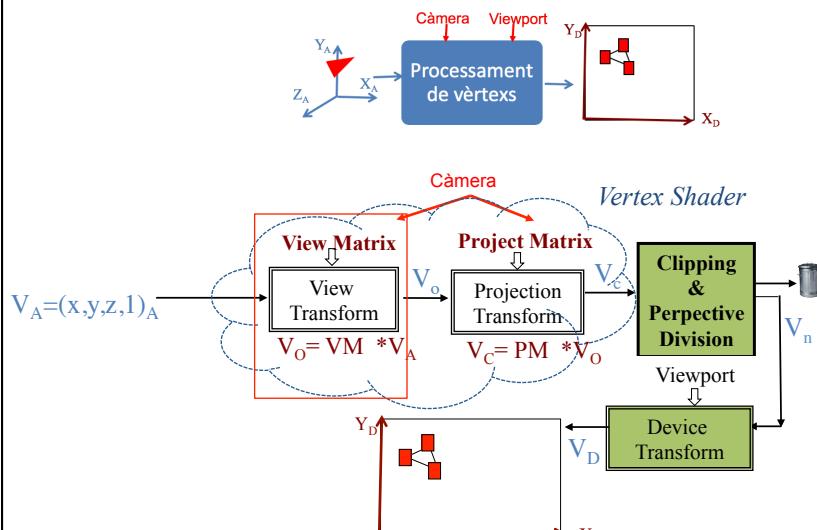


Condició per a que un Vèrtex sigui interior al volum de visió

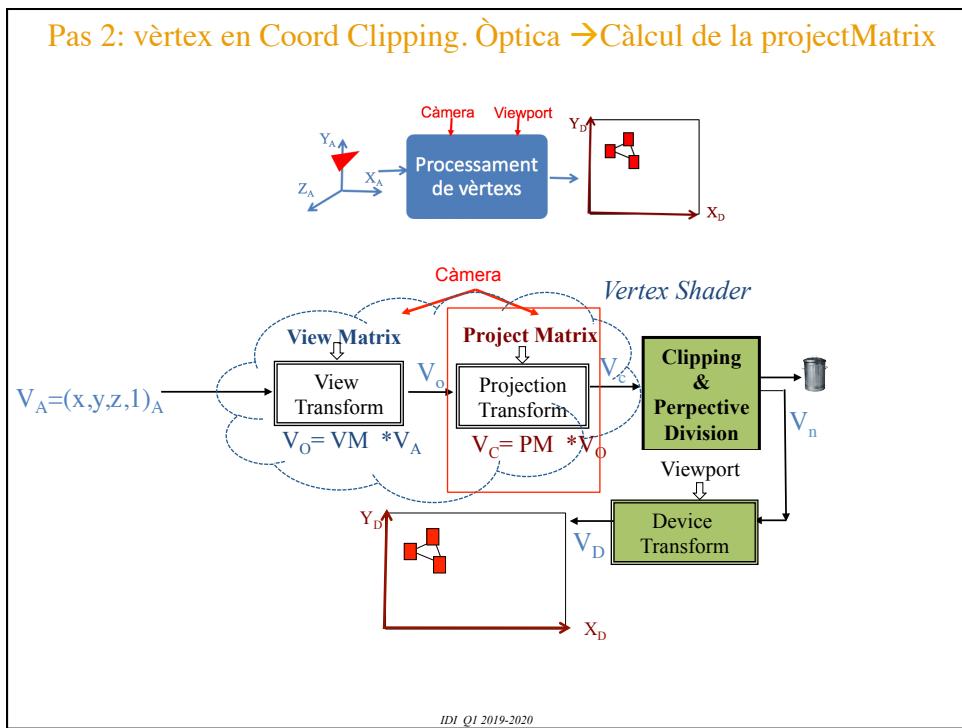
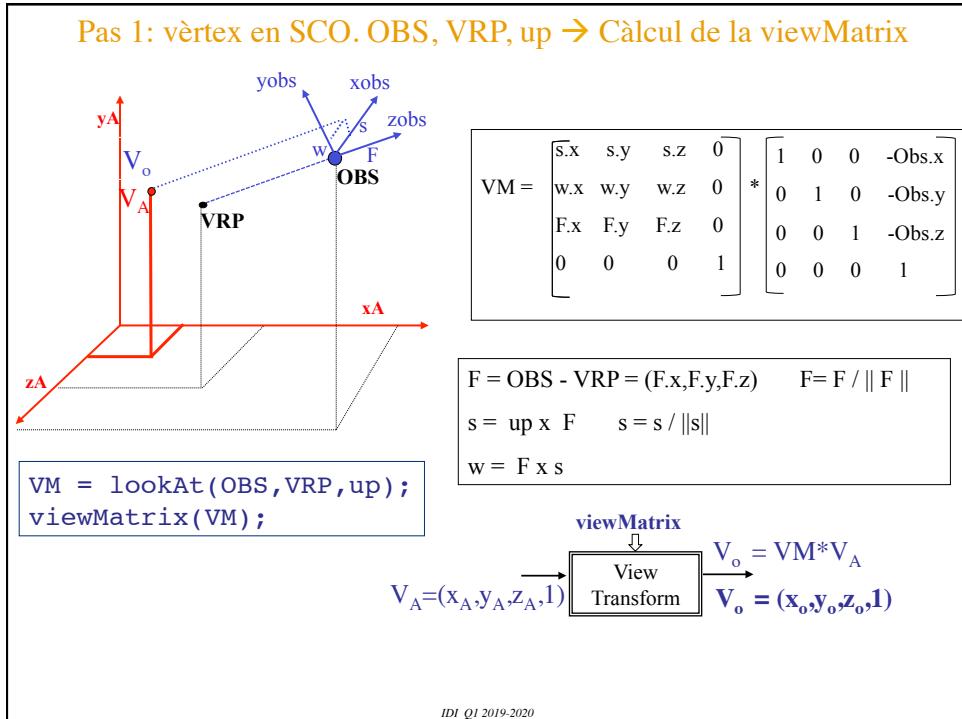
$$\begin{aligned} -w_c \leq x_c &\leq w_c \\ -w_c \leq y_c &\leq w_c \\ -w_c \leq z_c &\leq w_c \end{aligned}$$

IDI Q1 2019-2020

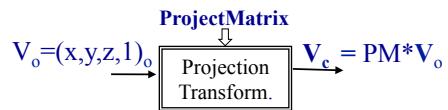
Processament de vèrtexs en OpenGL 3.3 (2)



IDI Q1 2019-2020



Pas 2: La projectMatrix



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

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

Óptica 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);`

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

$a = \tan(\text{FOV}/2)$
 $c = (zf+zn)/(zn-zf)$
 $d = 2*zn*zf/(zn-zf)$

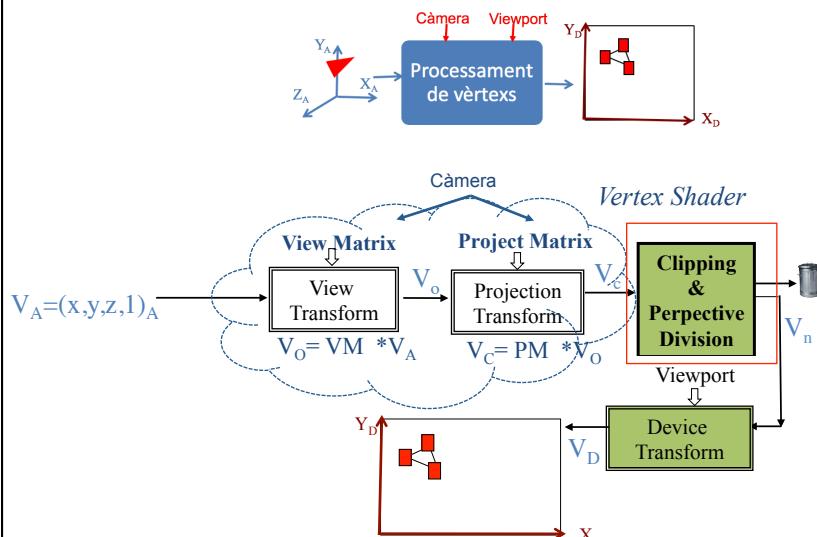
Óptica Perspectiva

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

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

IDI Q1 2019-2020

Pas 3: Clipping i projecció



IDI Q1 2019-2020

Pas 3.1 : Clipping

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

$$\begin{aligned} -W_c \leq X_c \leq W_c \\ -W_c \leq Y_c \leq W_c \\ -W_c \leq Z_c \leq W_c \end{aligned}$$

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

IDI Q1 2019-2020

Pas 3.2 : Projecció. Òptica ortogonal (1)

$$PM = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{aligned} a &= 2/(r-l) & b &= 2/(t-b) \\ c &= 2/(zf-zn) & d &= (zn+zf)/(zf-zn) \end{aligned}$$

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

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

IDI Q1 2019-2020

Pas 3.2 : Projeció. Òptica ortogonal (2)

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

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

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

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

$-1 \leq x_n \leq 1$
 $-1 \leq y_n \leq 1$
 $-1 \leq z_n \leq 1$

V_z^* per càlculs posteriors i indica distància a window

IDI Q1 2019-2020

Pas 3.2 : Projeció. Òptica perspectiva

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

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

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

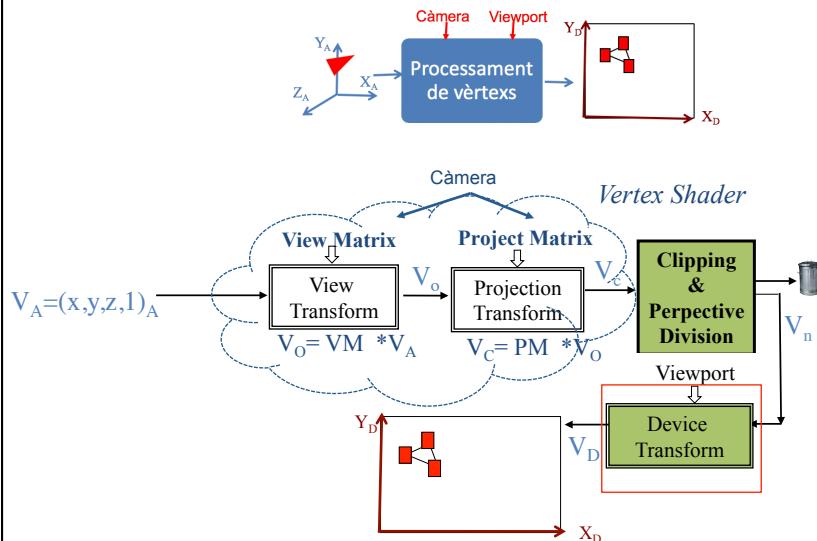
Inversament proporcional a distància a observador ☺

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

$-1 \leq x_n \leq 1$
 $-1 \leq y_n \leq 1$
 $-1 \leq z_n \leq 1$

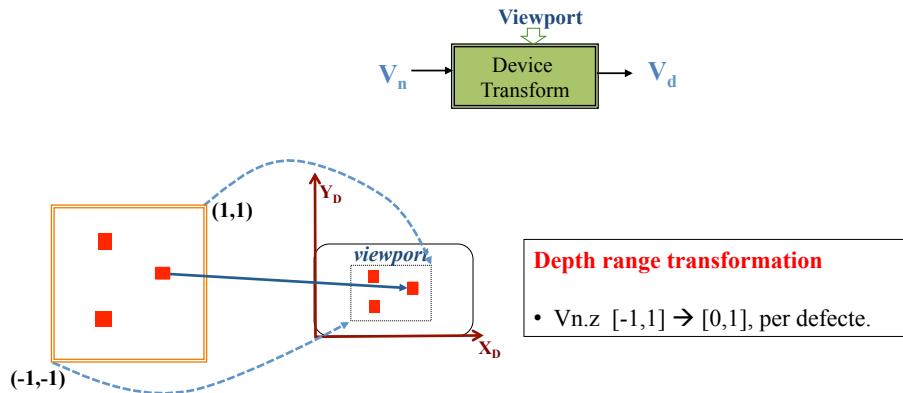
IDI Q1 2019-2020

Pas 4: Transformació a coordenades de dispositiu



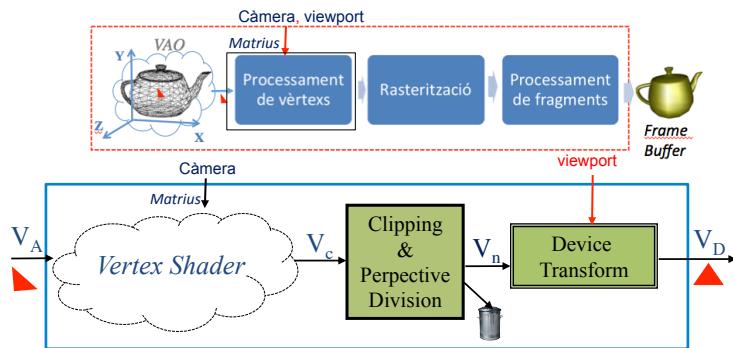
IDI Q1 2019-2020

Pas 4: Transformació a coordenades de dispositiu



IDI Q1 2019-2020

Processament de vèrtexs en OpenGL 3.3 (3)



IDI_Q1_2019-2020

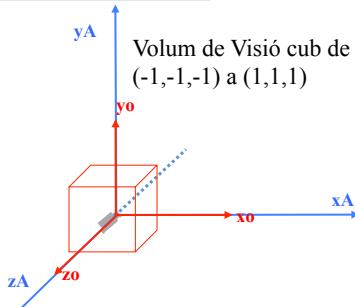
Processat de vèrtex: El vèrtex Shader

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);
}
```

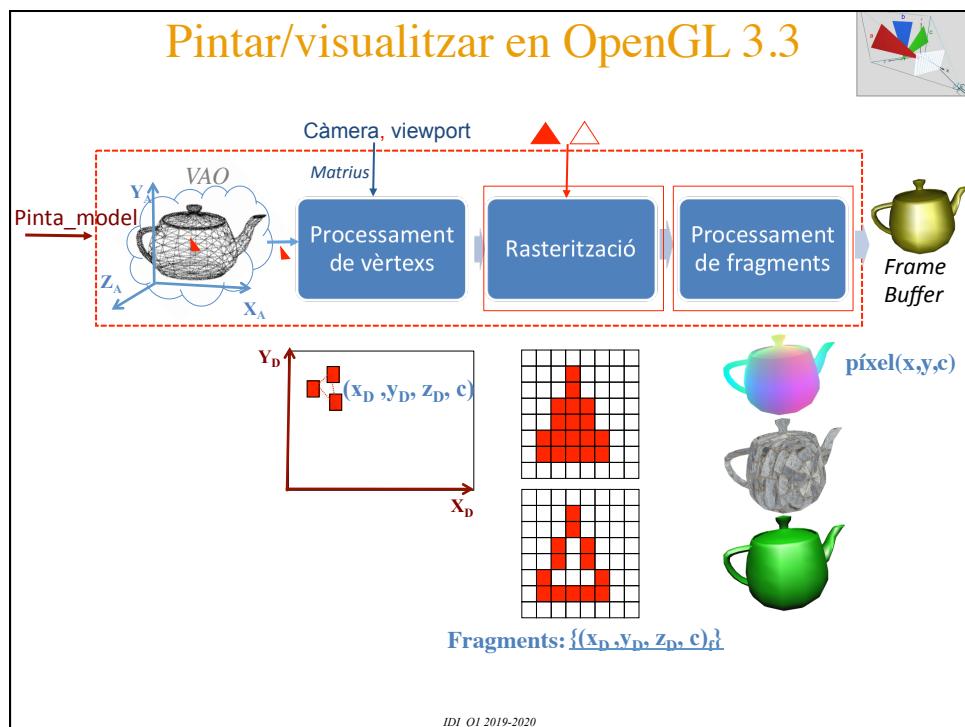


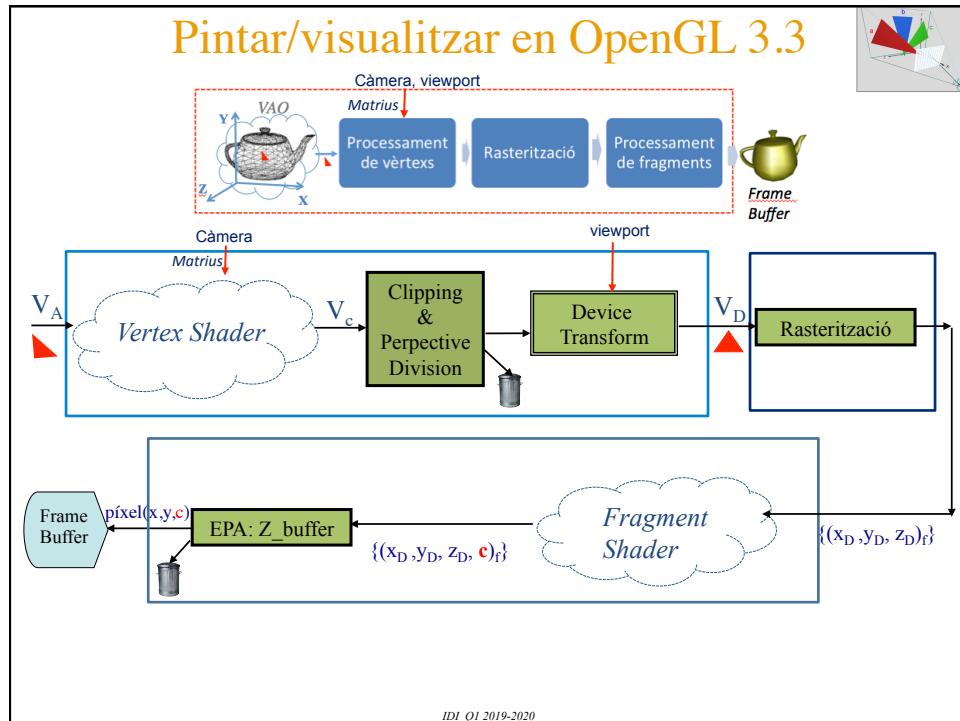
IDI_Q1_2019-2020

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**

IDI Q1 2019-2020





Processat de fragments: El fragment Shader

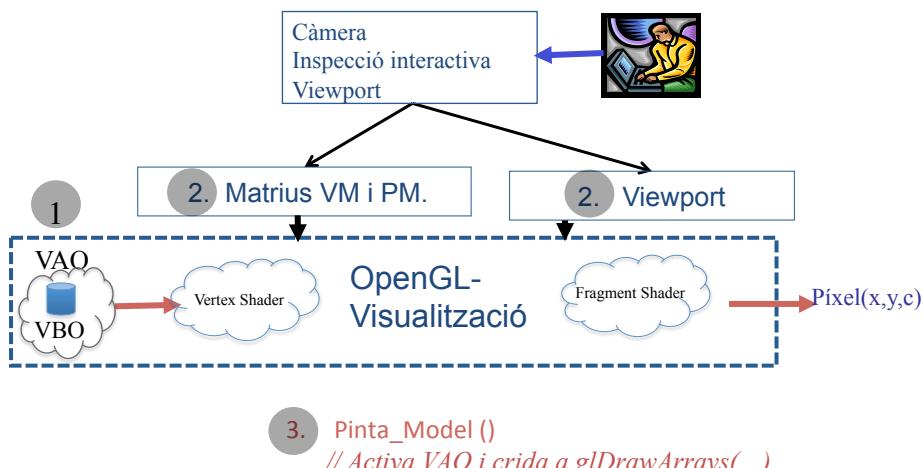
Fragment Shader

```
#version 330 core

out vec4 FragColor;

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

Pintar/visualitzar en OpenGL 3.3 (resum)



IDI Q1 2019-2020

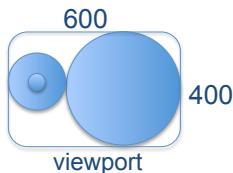
Classe 3: Conceptes i preguntes

- 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 fragment shader?
- Com s'obtenen les coordenades de clipping? Què son les coordenades normalitzades?

IDI Q1 2019-2020

Per pensar...

- Quins paràmetres requiria una àptica 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:



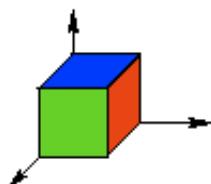
- Com fer un zoom? Quins paràmetres de la càmera modificaries?

IDI Q1 2019-2020

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.

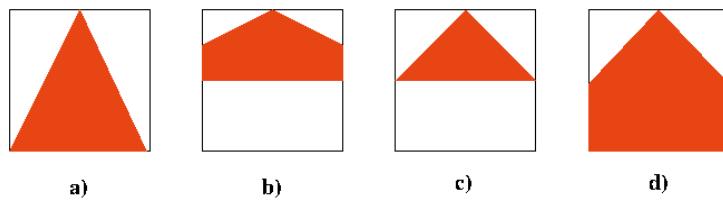
b) Quin efecte tindria en la imatge final modificar l'àptica ortogonal? Defineix la càmera ortogonal.



IDI Q1 2019-2020

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 pixels, i que hem inicialitzat les matrius de càmera (view) i projecció (proj) a la matríu 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):



IDI Q1 2019-2020