

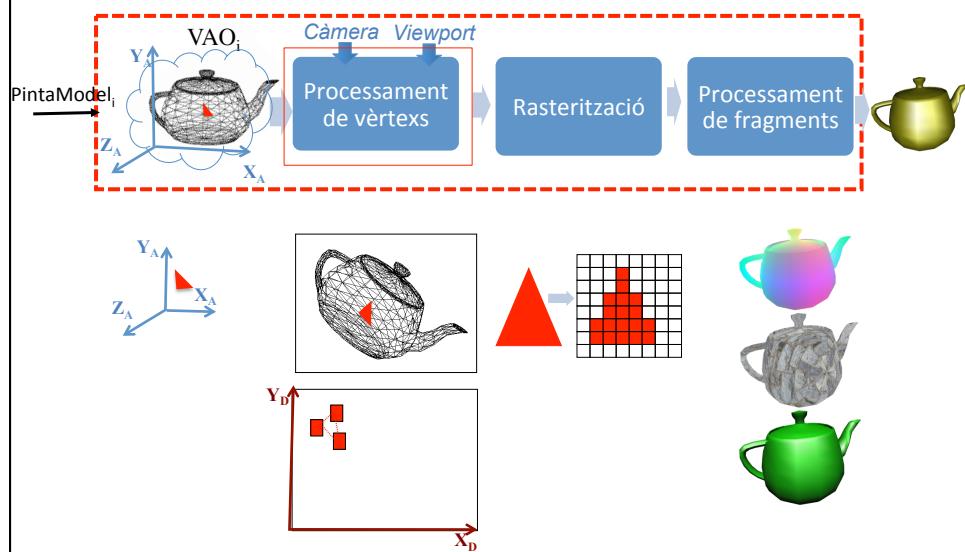
Classe 3: contingut

- Visualització: breu repàs.
- Càmera (1)
 - Posicionament: OBS, VRP, up
 - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs (2)
- Exemple definició de càmera

IDI Q1 2017-2018

1

Paradigma projectiu de visualització amb OpenGL 3.3

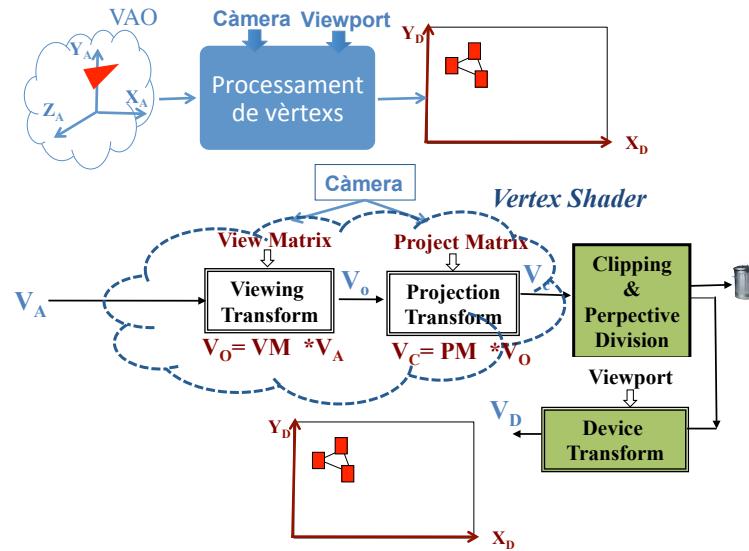


IDI Q1 2017-2018

2

1

Paradigma projectiu de visualització amb OpenGL 3.3



IDI Q1 2017-2018

3

Classe 3: contingut

- Visualització: breu repàs.
- **Càmera (1)**
 - Posicionament: OBS, VRP, up
 - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs (2)
- Exemple definició de càmera

IDI Q1 2017-2018

4

Posicionament de la càmera (1): OBS, VRP, up

OBS = Observador
VRP = View Reference Point
up = View Up Vector
up "indica" la direcció de l'eix vertical de la Càmera (inclinació)

Per a la determinació dels seus valors suposem que tots els objectes de l'escena estan al seu lloc → en SCA

IDI Q1 2017-2018

5

OBS, VRP, up → Càcul de la viewMatrix

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

$\mathbf{F} = \mathbf{OBS} - \mathbf{VRP} = (F.x, F.y, F.z) \quad \mathbf{F} = \mathbf{F} / \| \mathbf{F} \|$
 $\mathbf{s} = \mathbf{up} \times \mathbf{F} \quad \mathbf{s} = \mathbf{s} / \| \mathbf{s} \|$
 $\mathbf{w} = \mathbf{F} \times \mathbf{s}$

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

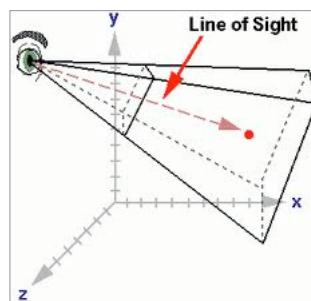
$V_A \xrightarrow{\text{viewMatrix}} \boxed{\text{Viewing Transform}} \xrightarrow{\mathbf{V}_o = VM^*V_A}$

IDI Q1 2017-2018

6



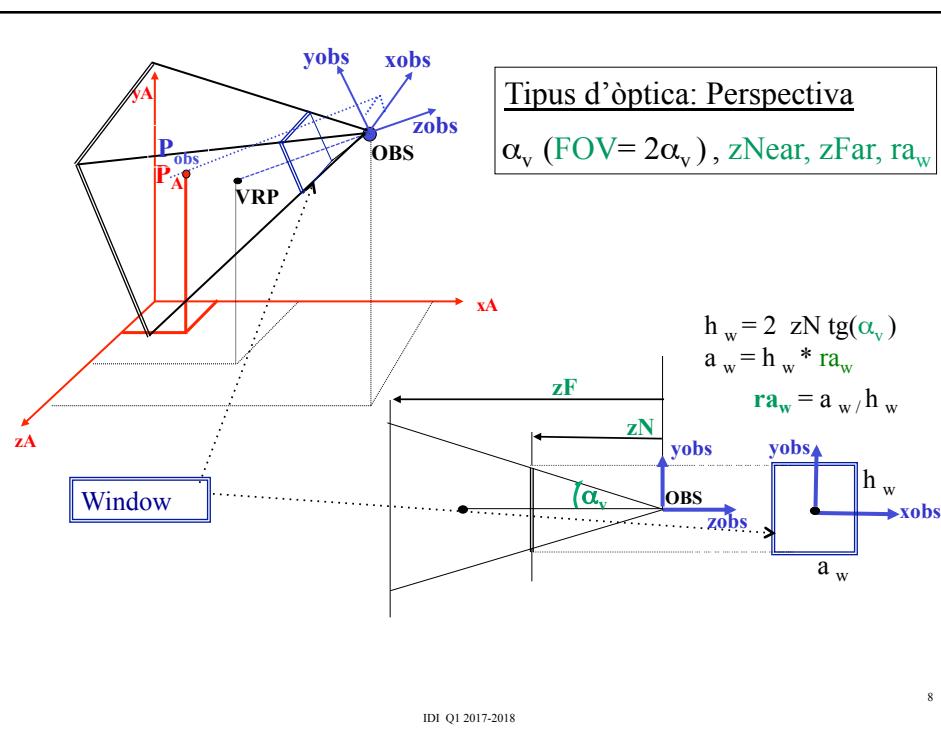
Òptica de la càmera (1): perspectiva



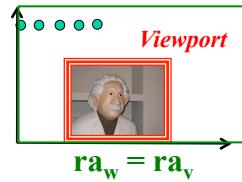
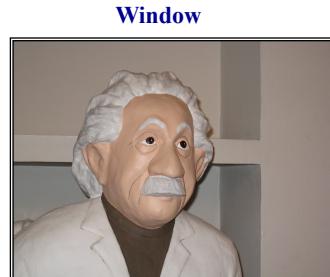
- Determina geometria potencialment visible
- **EL VOLUM de VISIÓ -ÒPTICA- SEMPRE es defineix RESPECTE L'OBSERVADOR**

IDI Q1 2017-2018

7



Sobre la relació d'aspecte del window i del viewport



Per a no tenir deformació en la imatge

$$ra_w = ra_v$$



IDI Q1 2017-2018

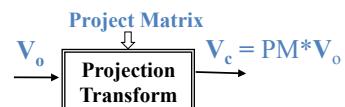
9

FOV, zNear, zFar, ra → Càcul de matriu de projecció (PM)

FOV, zNear, zFar, ra_w

$$PM = \begin{pmatrix} 1/ra*a & 0 & 0 & 0 \\ 0 & 1/a & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \begin{array}{l} a = \tan(FOV/2) \\ c = (f+n)/(n-f) \\ d = 2nf/(n-f) \end{array}$$

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

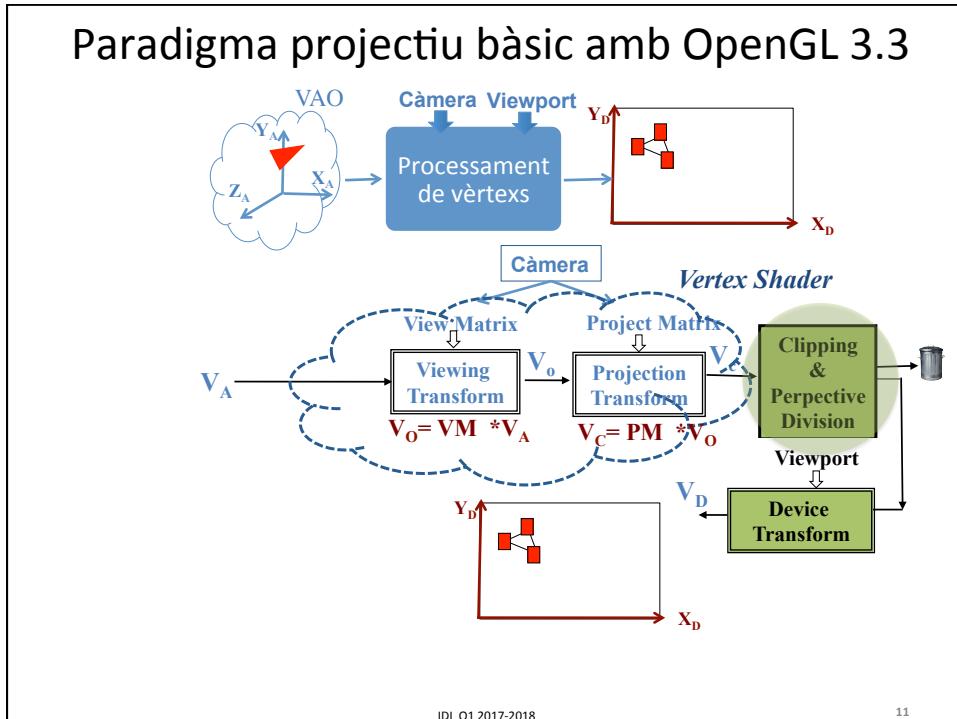


$$V_c = (x_c, y_c, z_c, w_c)_i$$

$$w_c = -z_o$$

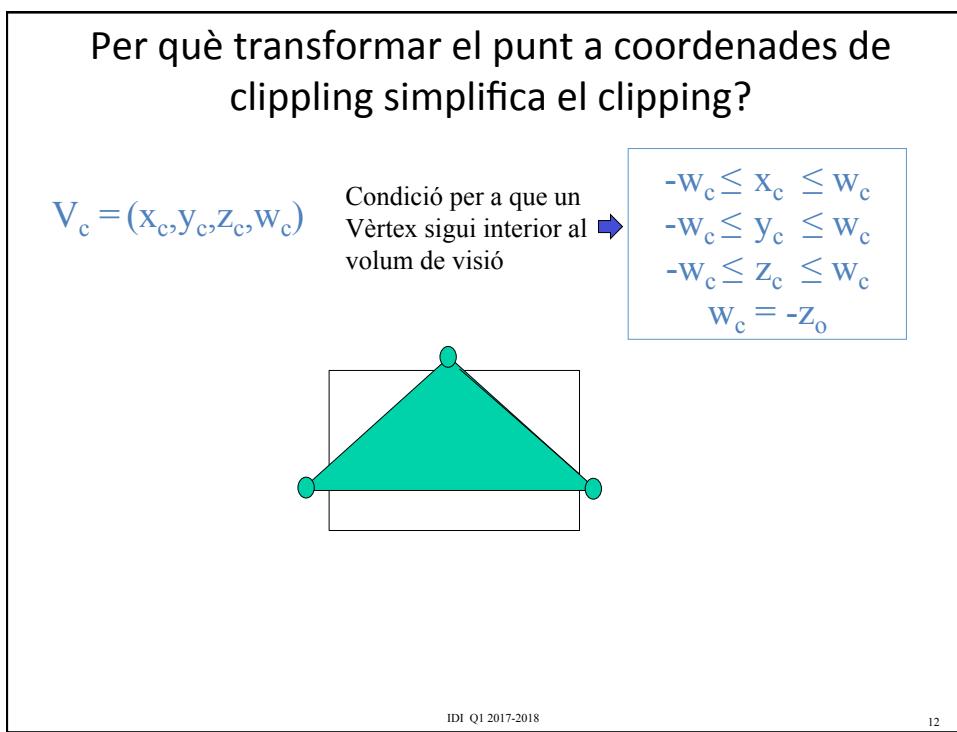
IDI Q1 2017-2018

10



IDI Q1 2017-2018

11



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

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

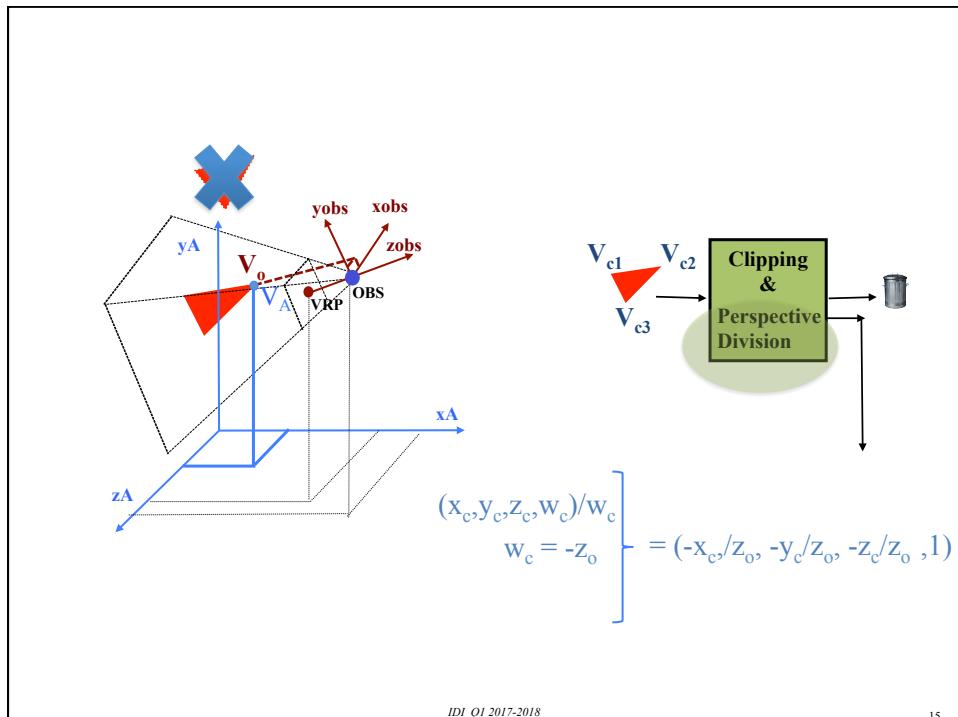
IDI Q1 2017-2018

13

Perspective division i Projeció

IDI Q1 2017-2018

14



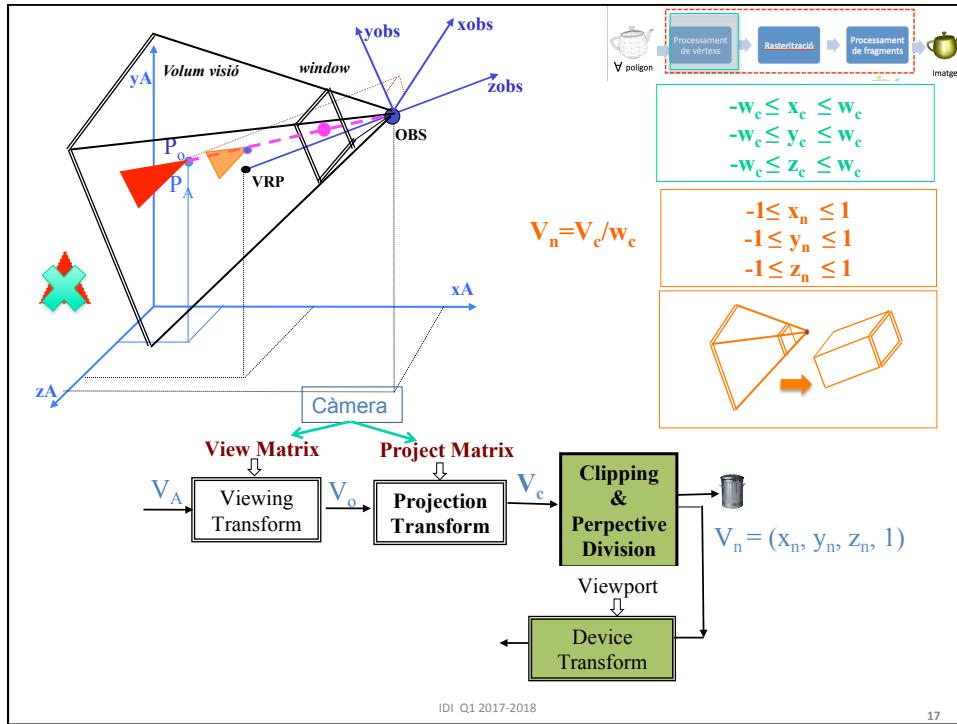
IDI Q1 2017-2018

15



IDI Q1 2017-2018

16



Given parameters: $\text{FOV} = 90^\circ$, $\text{ra} = 1$, $n = 1$, $f = 11$. The resulting projection matrix is:

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

The transformation equation is:

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

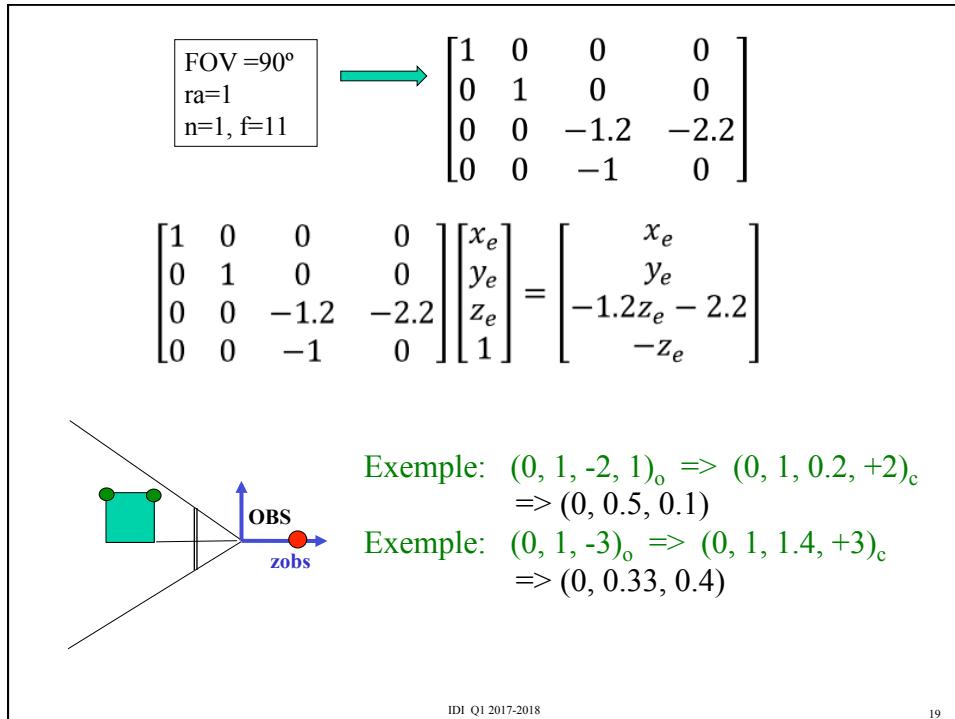
Below, a 3D diagram shows a point P_o being projected onto an observation plane (OBS) at z_{obs} .

Exemples:

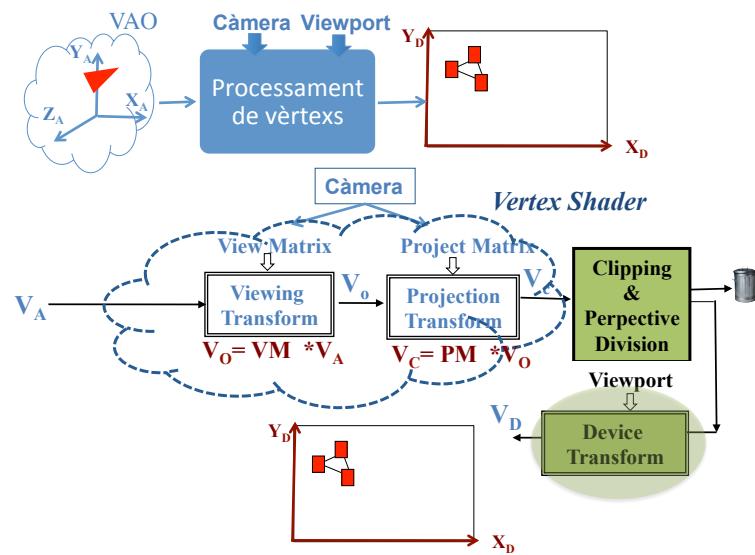
- $(0, 1, -2, 1)_o \Rightarrow (0, 1, 0.2, +2)_c$
- $(0, 1, -3, 1)_o \Rightarrow (0, 1, 1.4, +3)_c$
- $(0, 0, 1, 1)_o \Rightarrow (0, 0, -3.4, -1)_c$

At the bottom left: IDI Q1 2017-2018

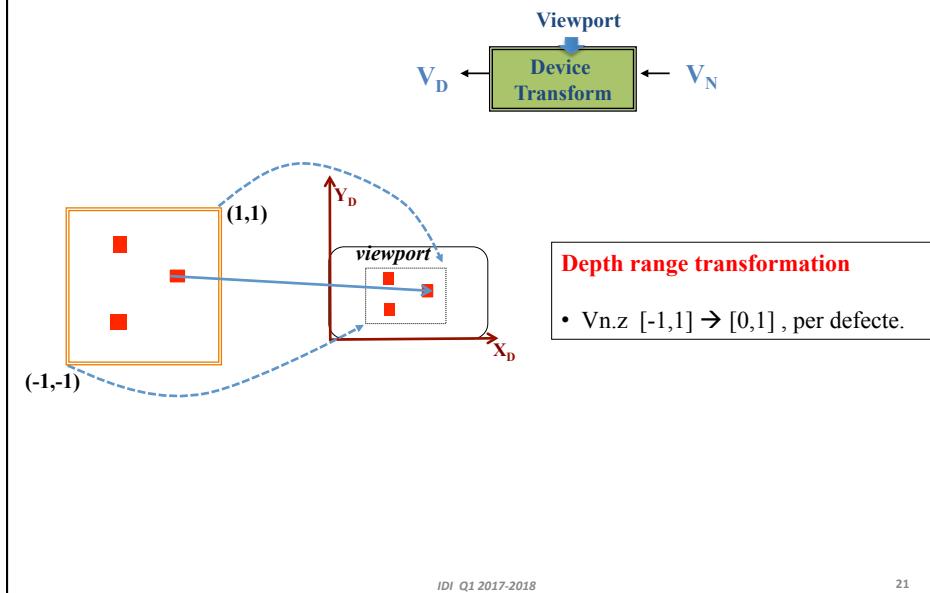
At the bottom right: 18



Paradigma projectiu bàsic amb OpenGL 3.3



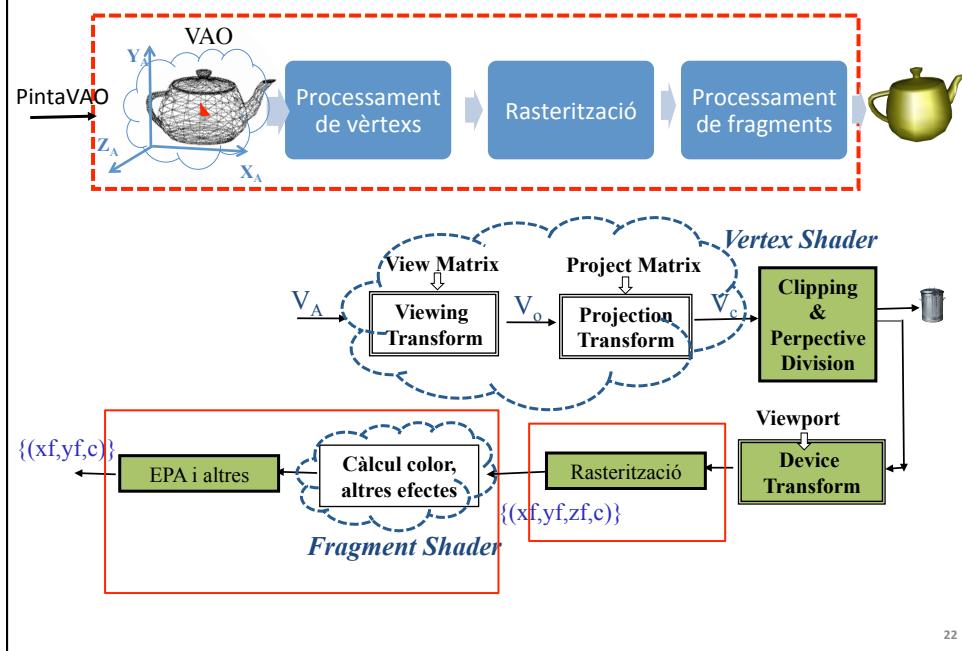
Paradigma projectiu bàsic amb OpenGL 3.3



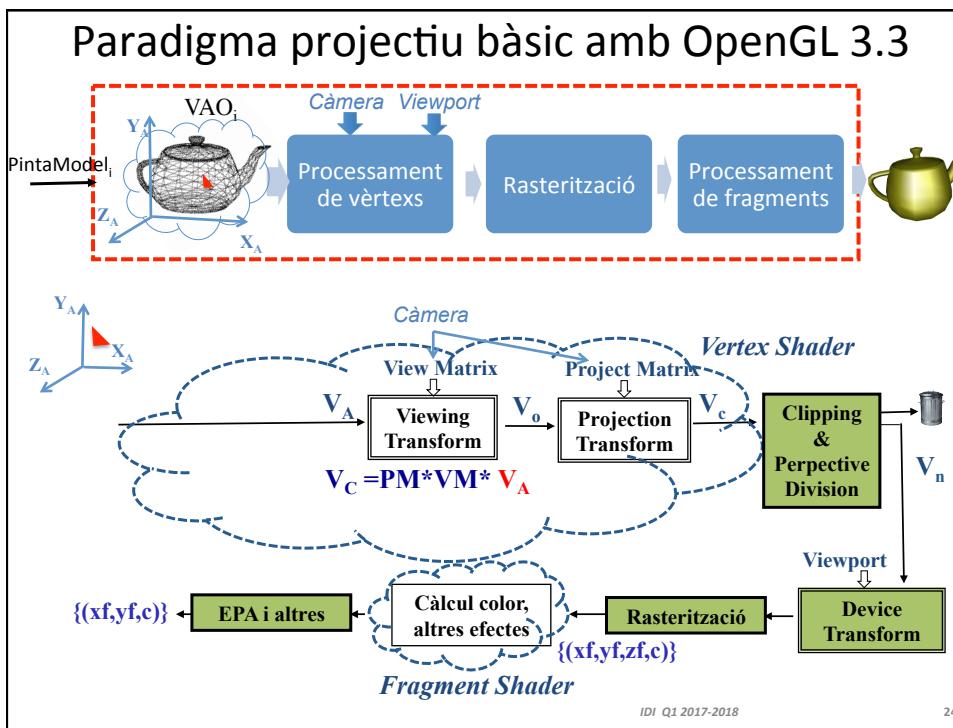
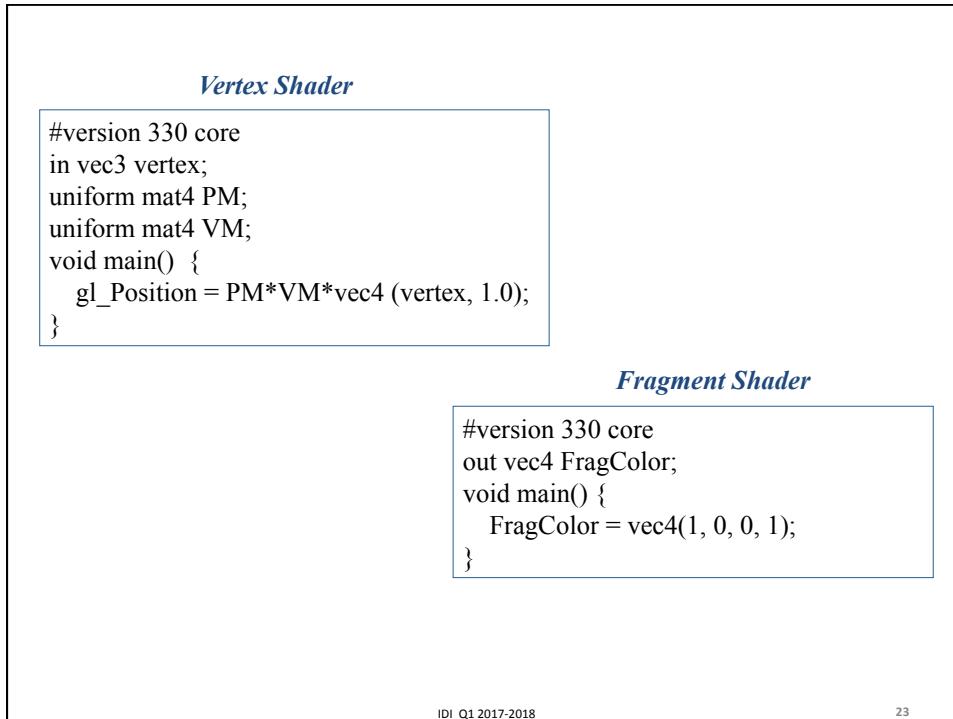
IDI Q3 2017-2018

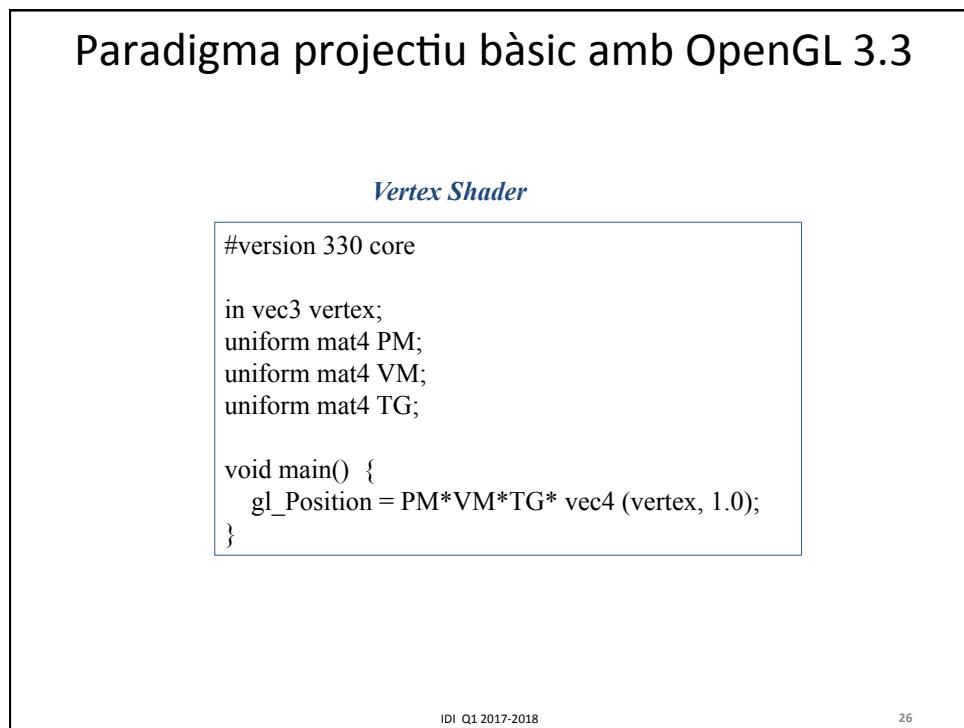
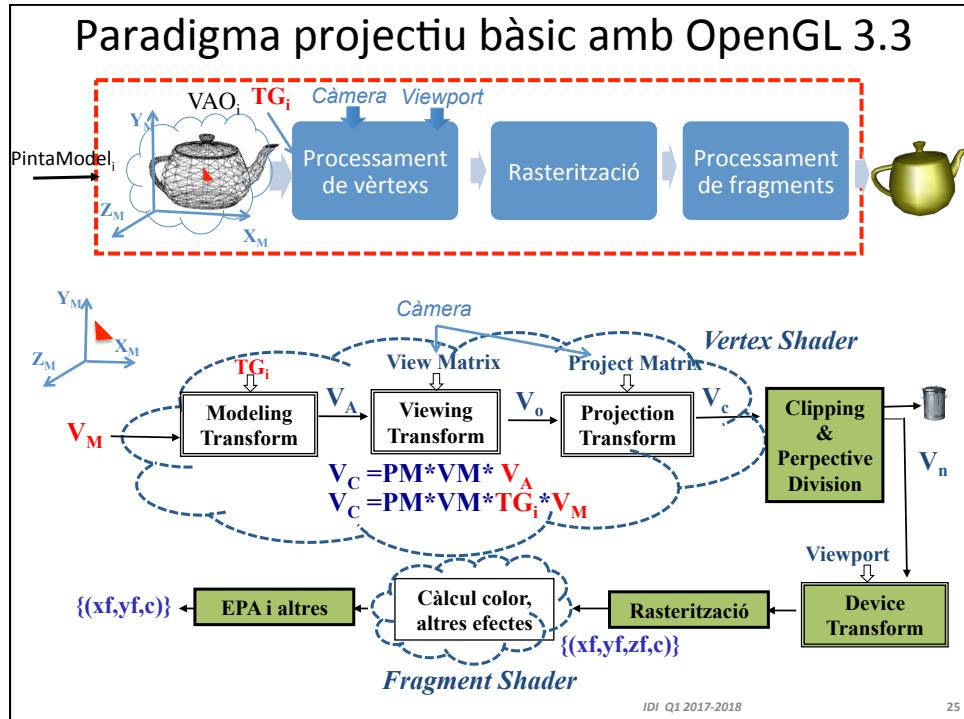
21

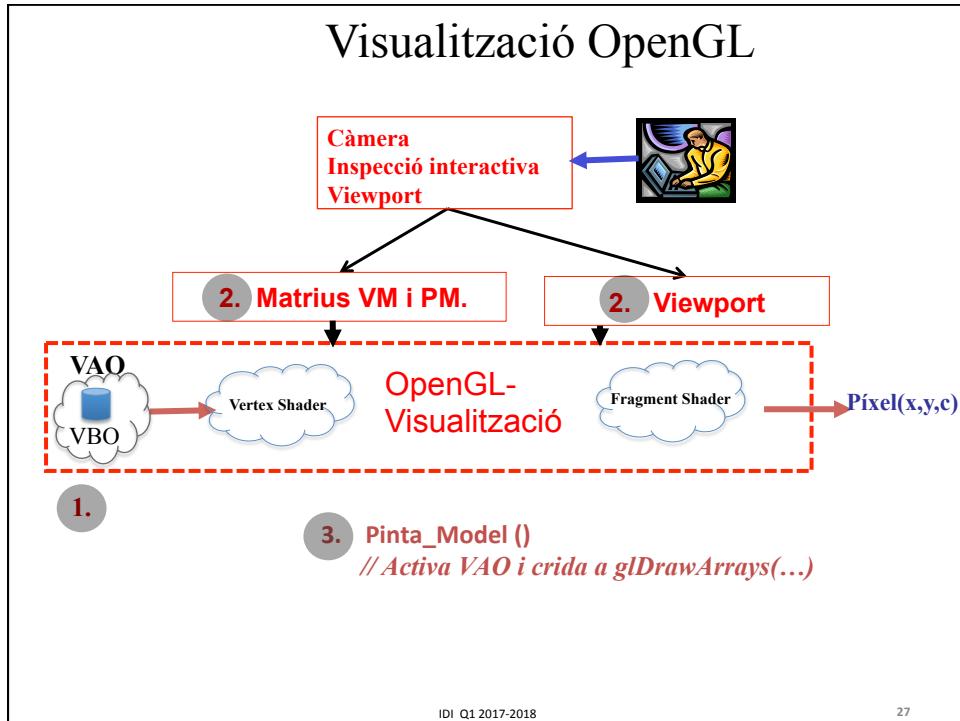
Paradigma projectiu bàsic amb OpenGL 3.3



22







IDI Q1 2017-2018

27

Classe 3: contingut

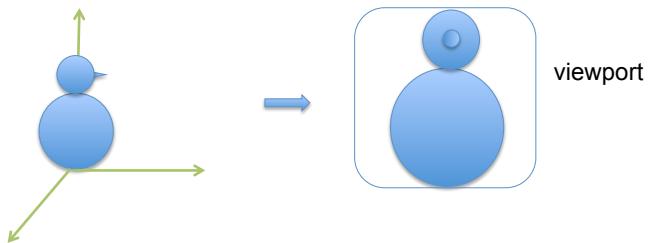
- Visualització: breu repàs.
- Càmera (1)
 - Posicionament: OBS, VRP, up
 - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs
- Exemple definició de càmera

IDI Q1 2017-2018

28

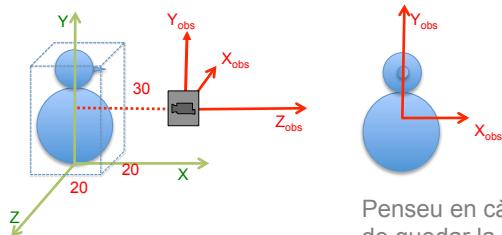
Exemple 1: Donada una funció `pinta_ninot()` que pinta un objecte com el de la figura, format per: una esfera de radi 10 i centre (0,10,0), una altra esfera de radi 5 i centre (0,25,0), i un con de base centrada en (2.5, 25,0), r=2 i llargada 5 orientat segons l'eix X⁺

- Indica tots els paràmetres d'una càmera que permeti obtenir la imatge similar a la que s'indica, en un viewport de 600x600 que ocupa tota la finestra gràfica.



IDI Q1 2017-2018

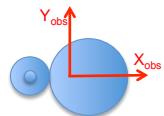
29



Penseu en càmera i com ha de quedar la imatge

```
VM = lookAt(OBS,VRP,up);
viewMatrix(VM);
pinta_ninot();
```

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



Quins paràmetres si volem que quedí així?

IDI Q1 2017-2018

30

Òptica de la càmera: Determina el Volum de Visió

1. Posició, orientació
2. Òptica

viewport

IDI Q1 2017-2018 31

Exemple 1: Òptica perspectiva

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

600
viewport

$z_N = 20; z_F = 40$
 $\alpha = \arctg(15/20) \rightarrow \alpha = 36,8^\circ$
 $ra_w = 20/30 = 0,66$
 Com $ra_v = 1 \rightarrow$ deformació
 Solució $ra_w = 1$

IDI Q1 2017-2018 32

```

/* CreateBuffers(); Crear VAO del model (un
cop)*/

...
/* calcular paràmetres càmera i
matrius cada cop que es
modifiquin */
VM = lookAt(OBS,VRP,UP);
viewMatrix(VM);
PM=perspective (FOV,ra,zN,ZF);
projectMatrix(PM);
glViewport (0,0,w.h);
...
/*PaintGL(); cada cop que es requerix
refresc*/
//Calcular TG, i passar a OpenGL
modelTransform_i(TG);
modelMatrix(TG);
pinta_ninot(VAO);

```

Vertex Shader

```

in vec3 vertex;
uniform mat4 TG, VM, PM;
void main ()
{
    gl_Position =
        PM*VM*TG*vec4(vertex,1.0);
}

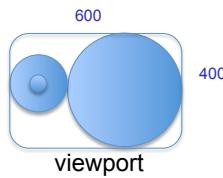
```

IDI Q1 2017-2018

33

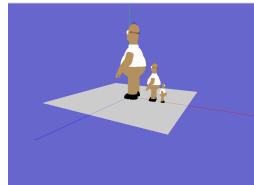
Per pensar...

- Com fer un zoom? Quins paràmetres de càmera modificaries?
- Quins paràmetres càmera per a obtenir:



IDI Q1 2017-2018

34



Per pensar...

Quins podrien ser uns **paràmetres de posició, orientació i òptica** per a una càmera que, donada una escena i coneguda la seva capsa mínima contenidora

(x_{\min} , y_{\min} , z_{\min}) - (x_{\max} , y_{\max} , z_{\max}), visualitzi una imatge que inclogui totalment l'escena, ocupant el màxim de la vista (viewport) i sense deformació?

Classe 3: contingut

- Visualització: breu repàs.
- **Càmera (1)**
 - Posicionament: OBS, VRP, up
 - Òptica perspectiva: definició i paràmetres
- Visualització: processat vèrtexs
- Exemple definició de càmera