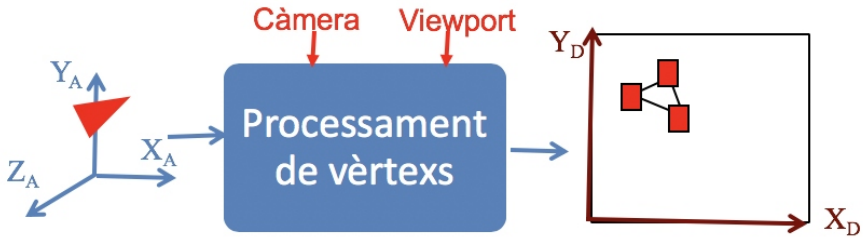
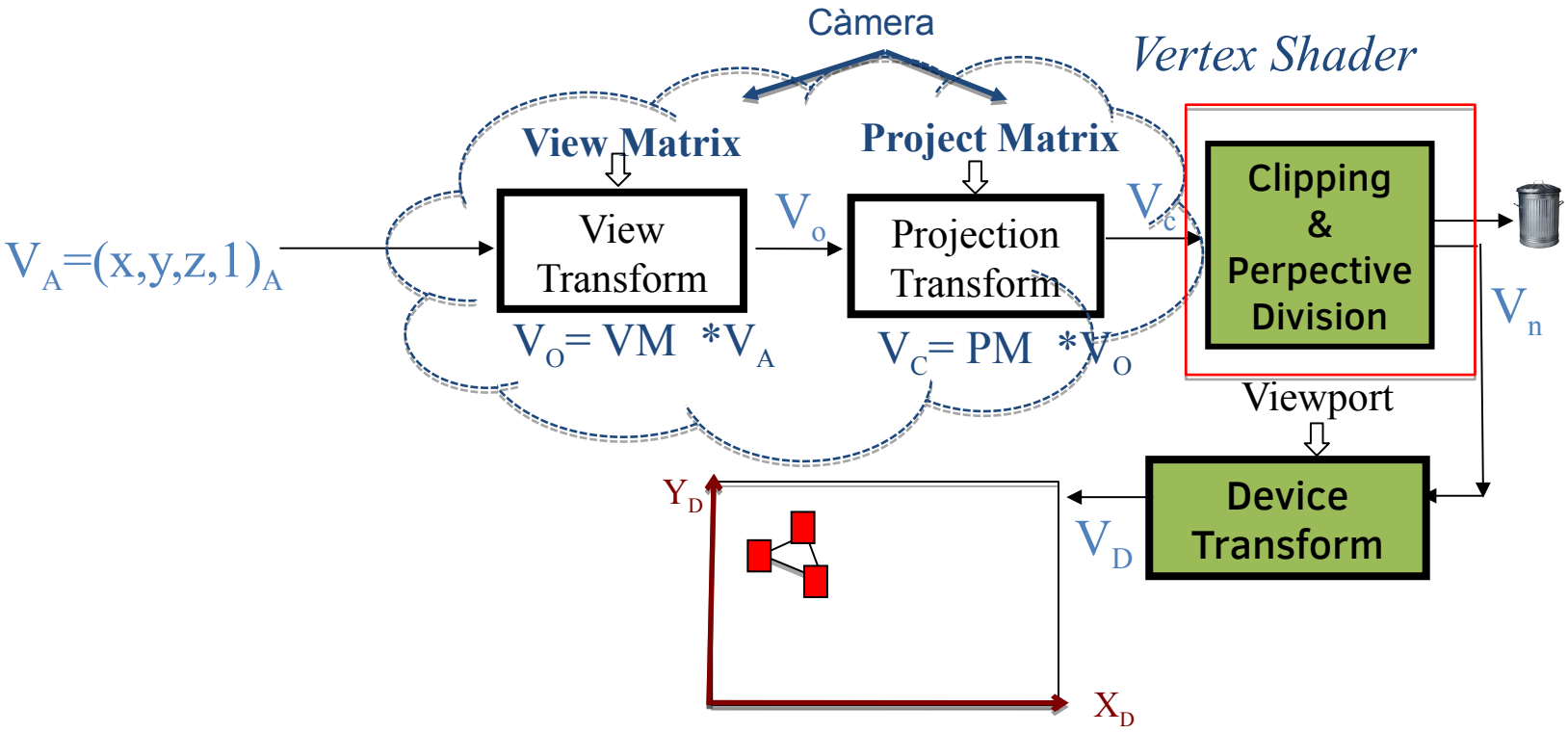
Classe 4: Contingut

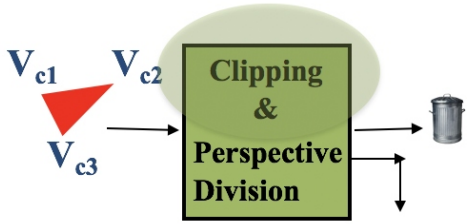
- Especificació de càmera
- El procés de visualització projectiu
- Processament dels vèrtexs
  - Retallat
  - De coordenades de clipping a coordenades de dispositiu
- Rasterització
- Processament dels fragments: el fragment shader

Pas 3: Clipping i projecció





Pas 3.1 : Clipping

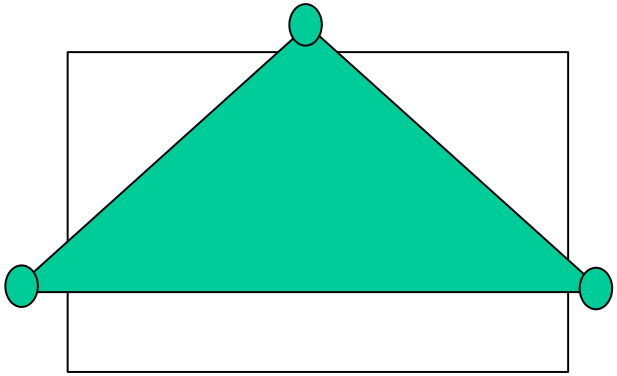


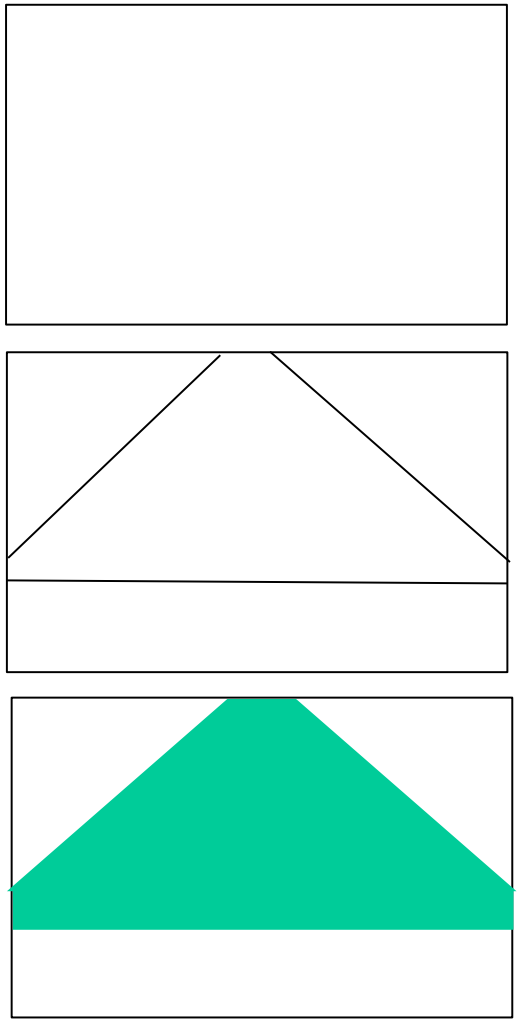
Condició per a que un Vèrtex sigui interior al volum de visió:

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

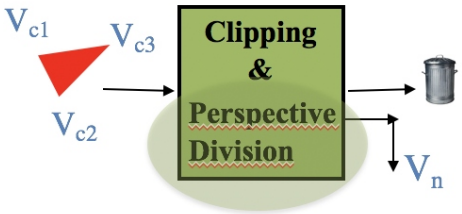
$V_c = (x_c, y_c, z_c, w_c)$ on $w_c = 1$ en ortogonal

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





Pas 3.2 : Projecció. Òptica ortogonal (1)



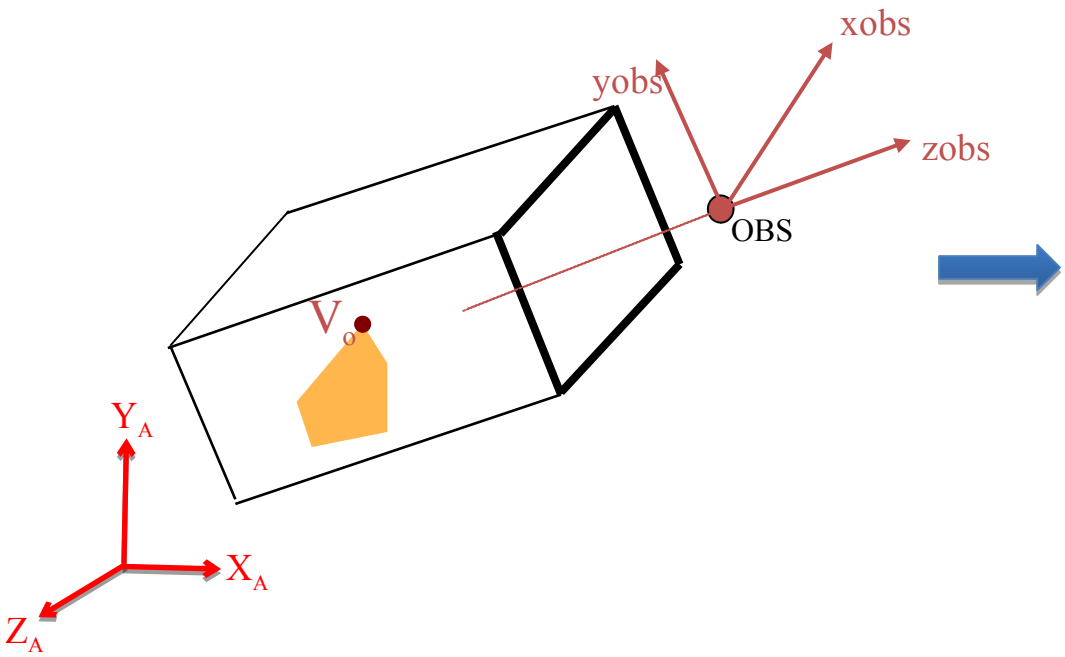
$$PM = \begin{pmatrix} a & 0 & 0 & e \\ 0 & b & 0 & f \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

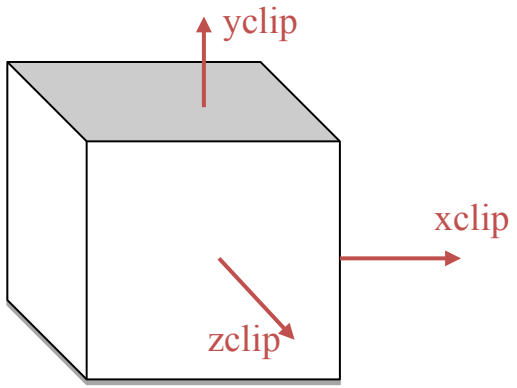
$a = 2/(r-l)$ $\quad e = (r+l)/(r-l)$

$b = 2/(t-b)$ $\quad f = (t+b)/(t-b)$

$c = 2/(zf-zn)$ $\quad d = (zn+zf)/(zf-zn)$

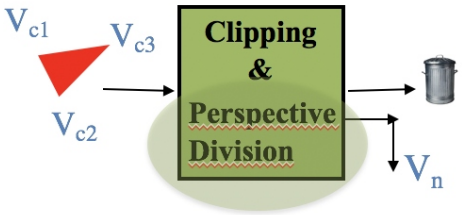
$$V_c = (x_c, y_c, z_c, w_c) \text{ on } w_c = 1$$

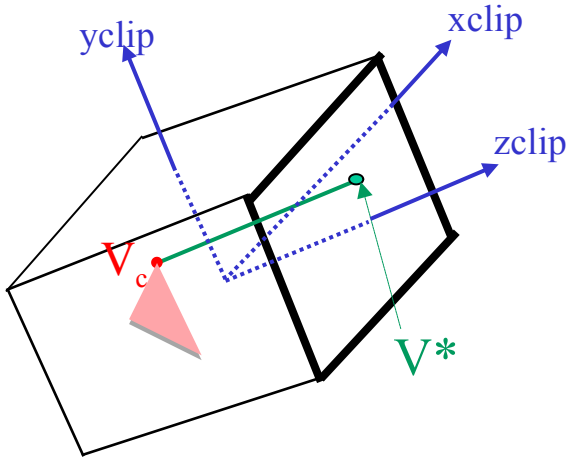




$$-1 \le x_c \le 1$$
$$-1 \le y_c \le 1$$
$$-1 \le z_c \le 1$$

Pas 3.2 : Projecció. Òptica ortogonal (2)





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

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

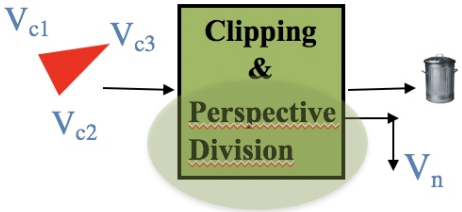
Vèrtex projectat:
$V_x^* = V_{cx}$ $V_y^* = V_{cy}$

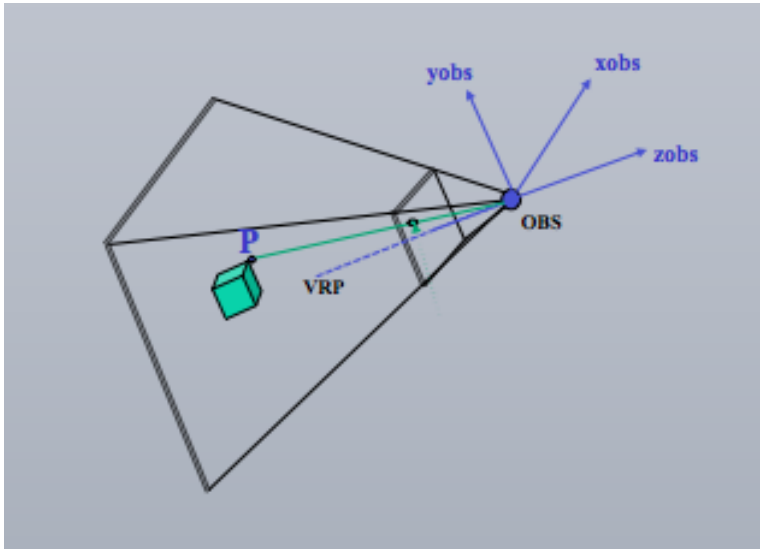
$$V^* = V_c / w_c \rightarrow V_n$$

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

V_z^* per càlculs posteriors i indica distància a window

Pas 3.2 : Projecció. Òptica perspectiva





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

$a = tg\ (FOV/2)$

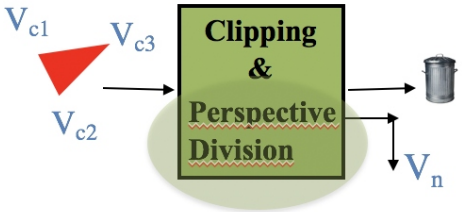
$c = (zf+zn)/(zn-zf)$

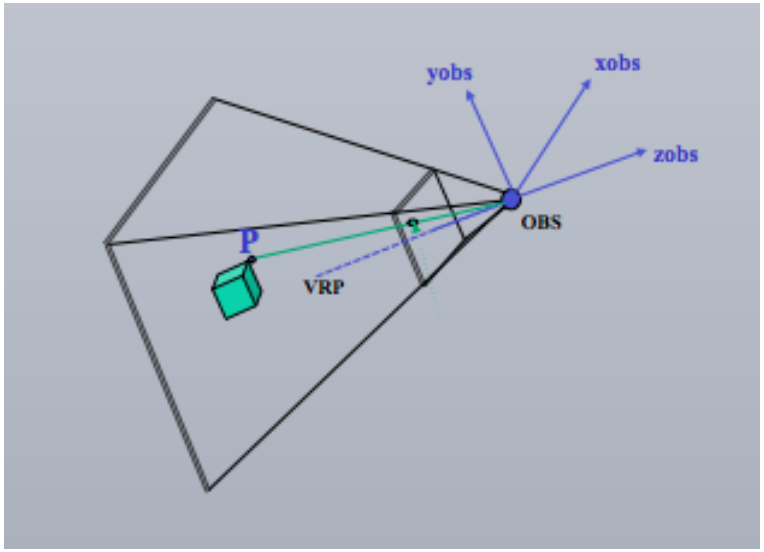
$d = 2*zn*zf/(zn-zf)$

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

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

Pas 3.2 : Projecció. Òptica perspectiva



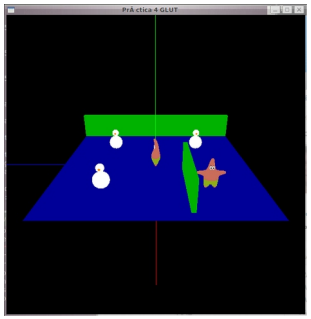


Vèrtex projectat:

$V^* = V_c / w_c = -V_c / z_o$

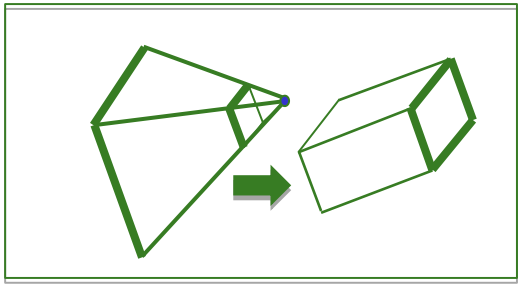
$x^* = -x_c / z_o \quad y^* = -y_c / z_o \quad z^* = -z_c / z_o$

Inversament proporcional a distància a observador



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

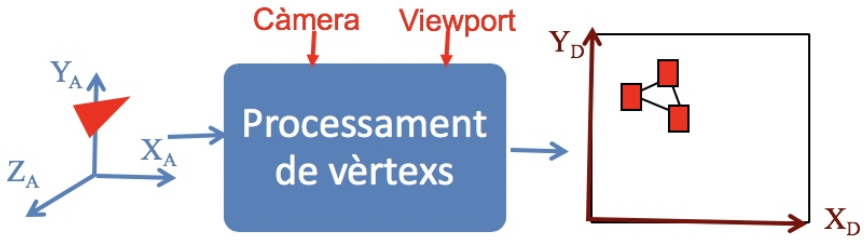
$$-w_c \leq x_c \leq w_c$$
$$-w_c \leq y_c \leq w_c$$
$$-w_c \leq z_c \leq w_c$$

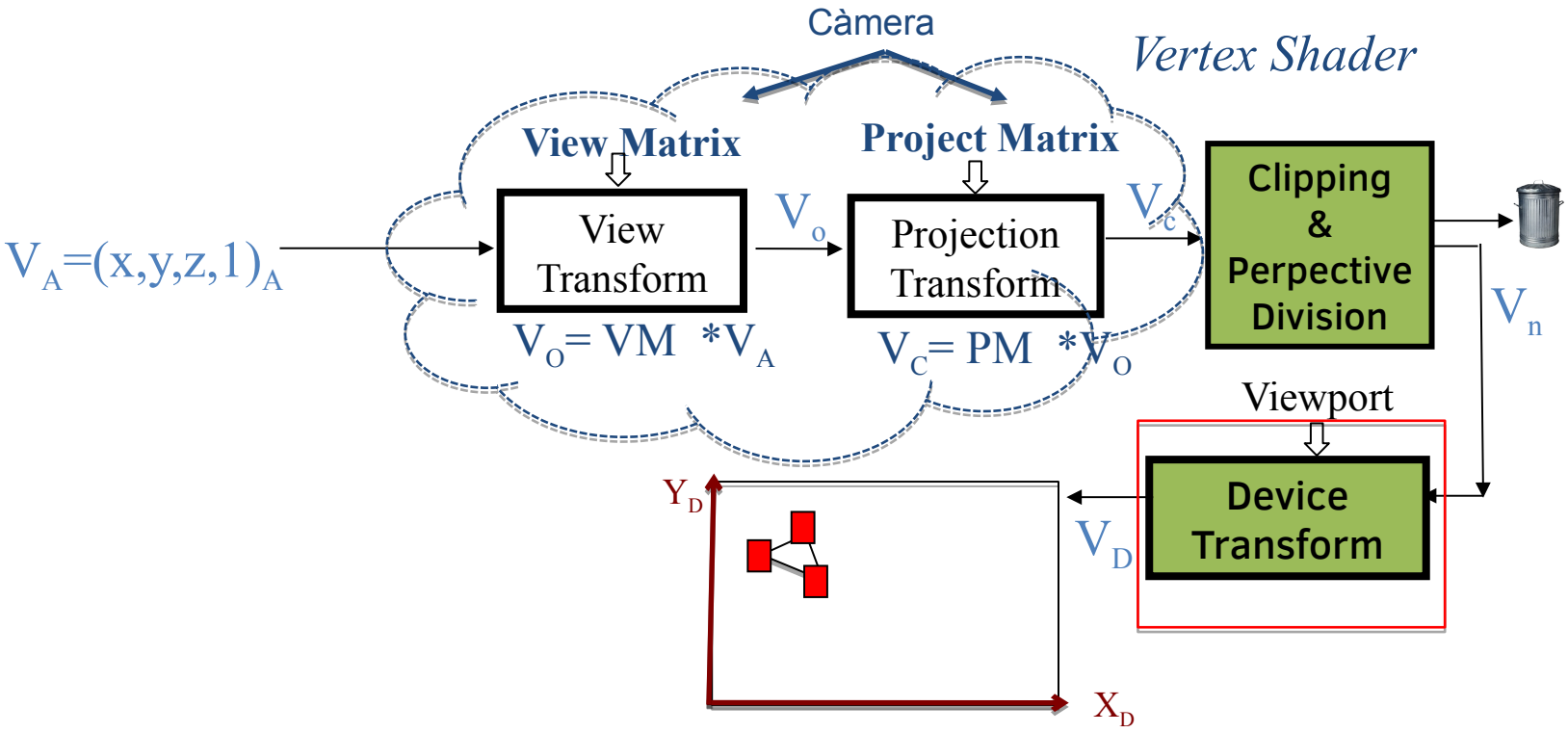


$$-1 \leq x_n \leq 1$$
$$-1 \leq y_n \leq 1$$
$$-1 \leq z_n \leq 1$$

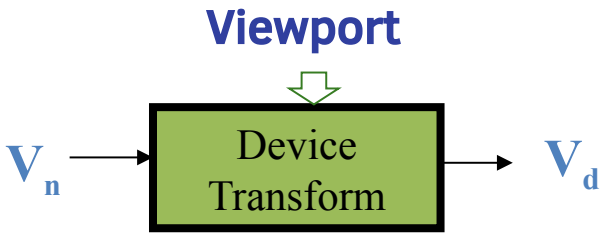
$V^* = V_c / w_c \rightarrow V_n$

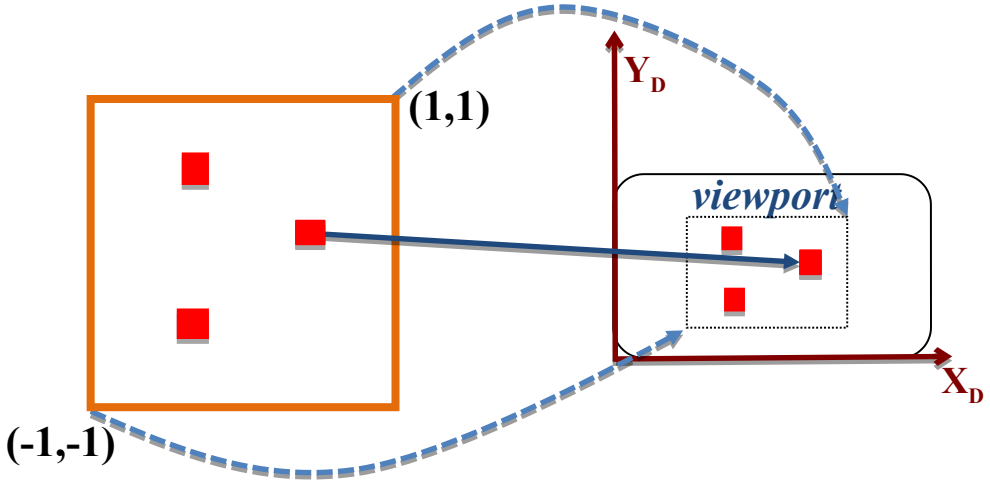
Pas 4: Transformació a coordenades de dispositiu





Pas 4: Transformació a coordenades de dispositiu

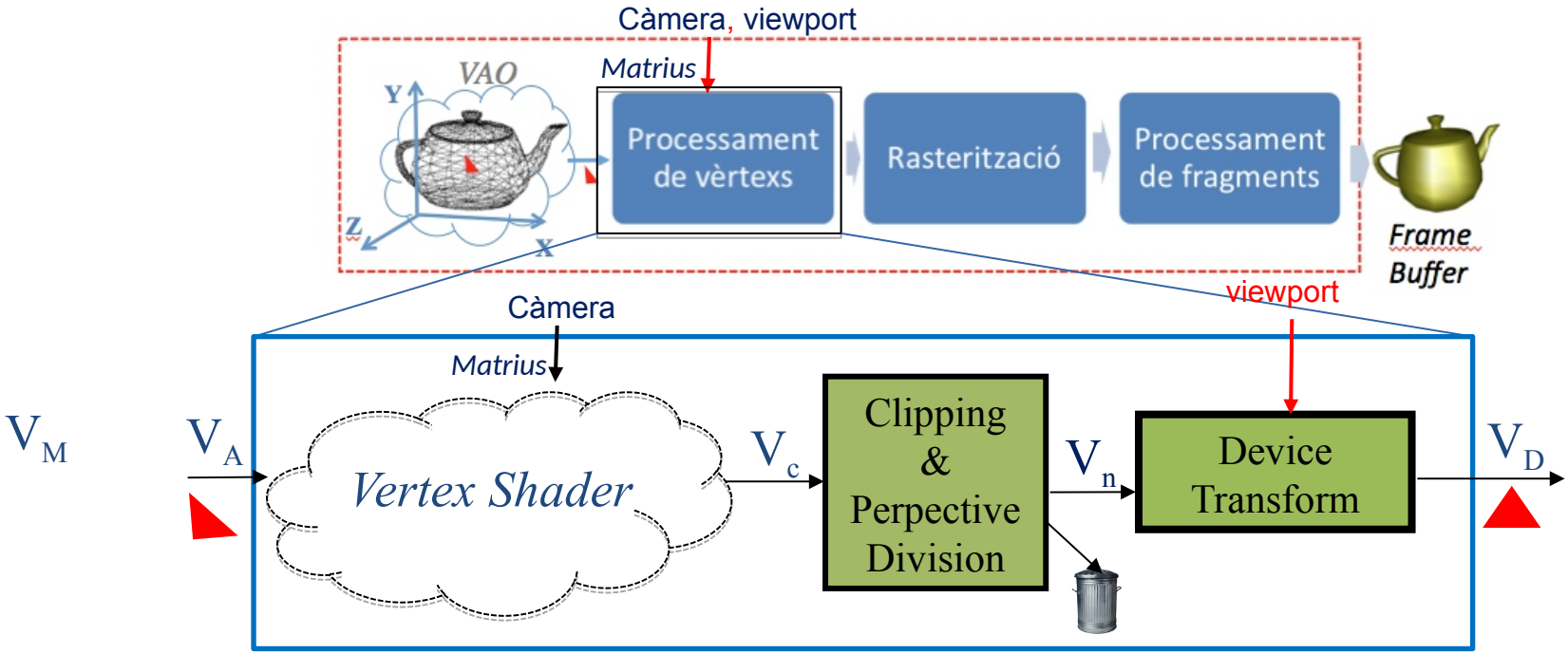




Depth range transformation

- Vn.z [-1,1] → [0,1], per defecte.

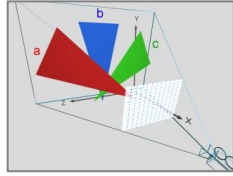
Processament de vèrtexs en OpenGL 3.3 (3)

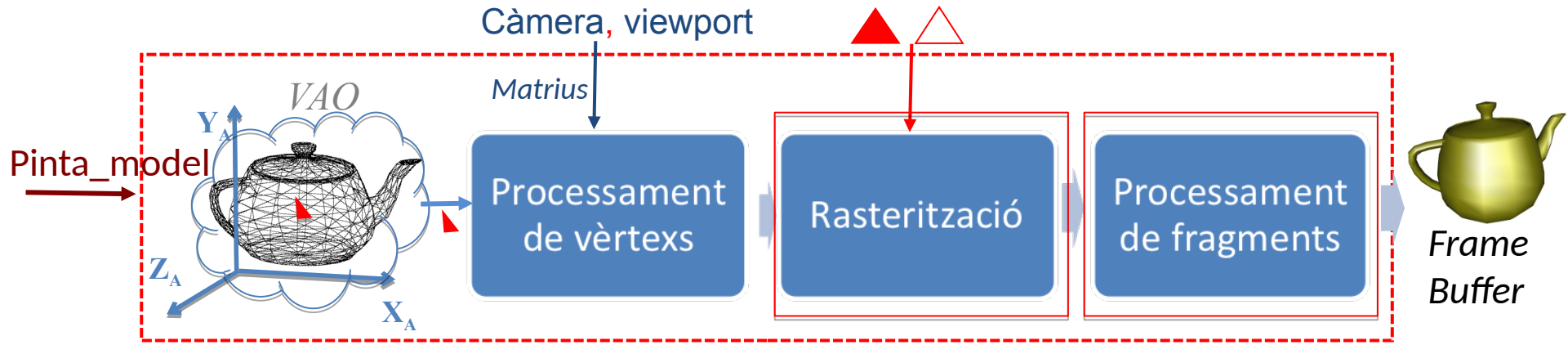


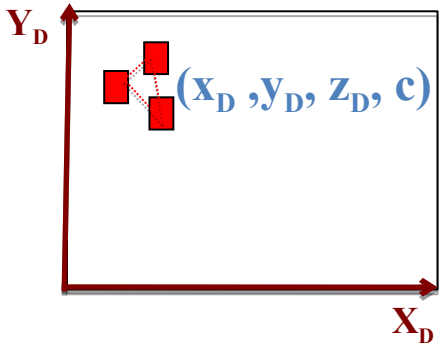
Classe 4: Contingut

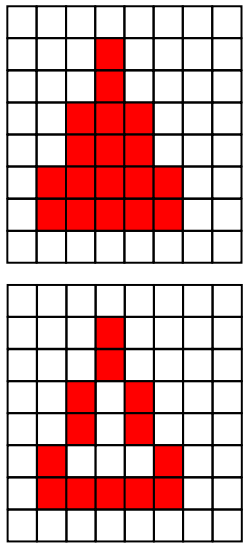
- Especificació de càmera
- El procés de visualització projectiu
- Processament dels vèrtexs
  - Retallat
  - De coordenades de clipping a coordenades de dispositiu
- Rasterització
- Processament dels fragments: el fragment shader

Pintar/visualitzar en OpenGL 3.3





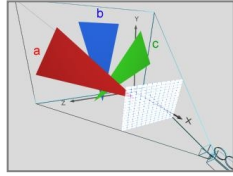


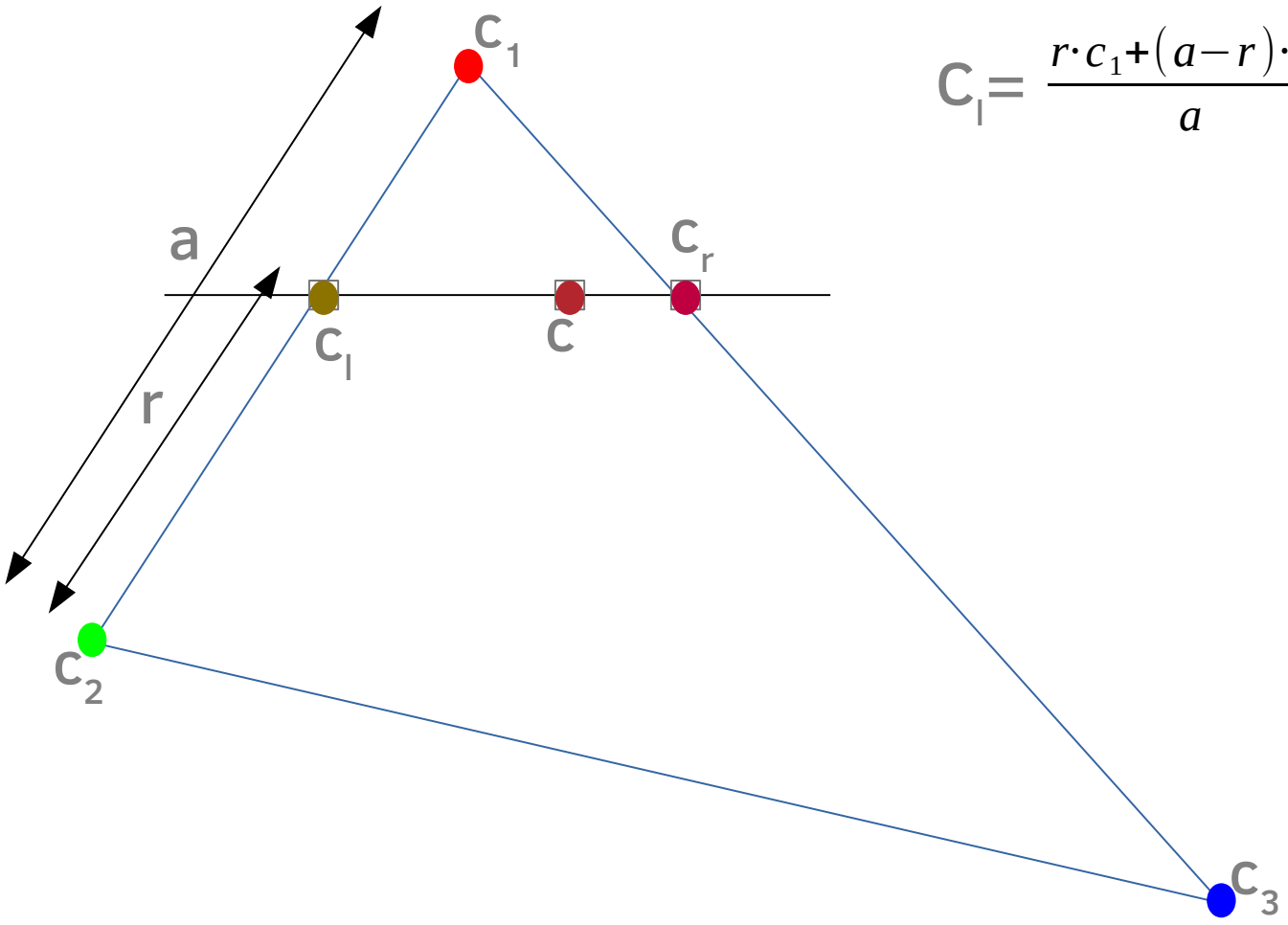


píxel(x,y,c)

Fragments: $\{(x_D, y_D, z_D, c)_f\}$

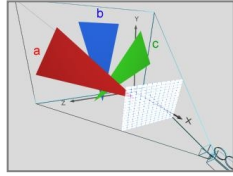
Pintar/visualitzar en OpenGL 3.3

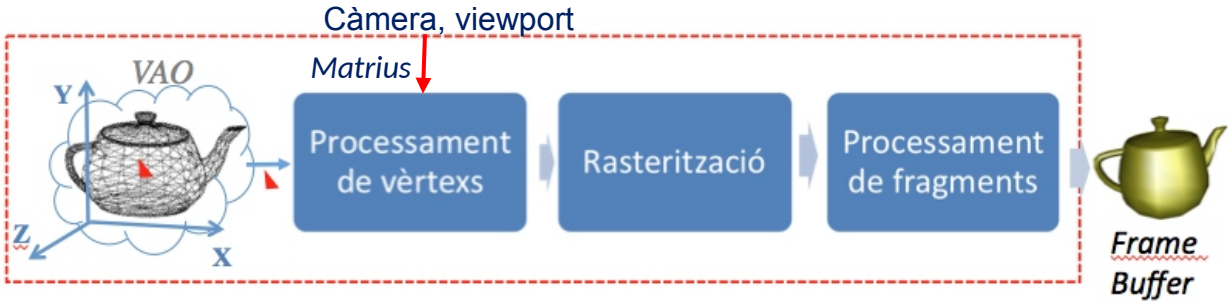


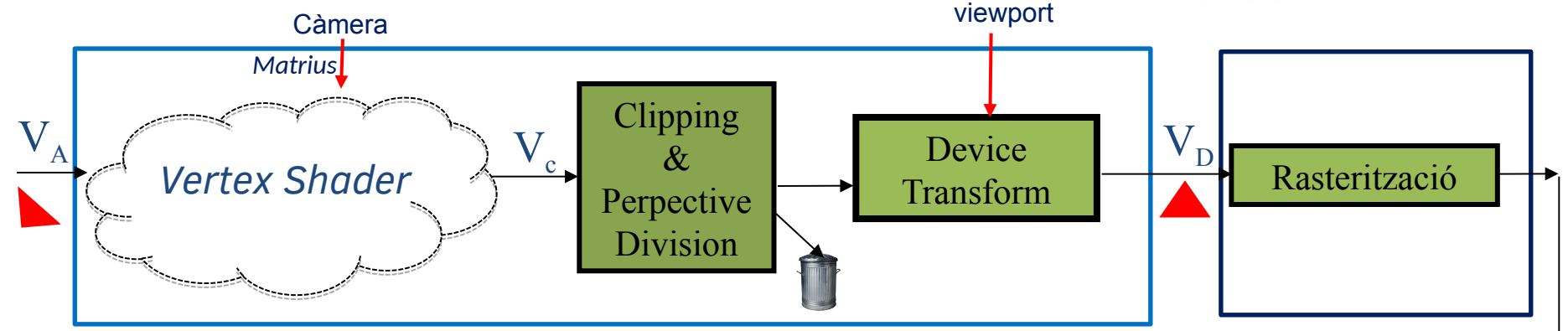


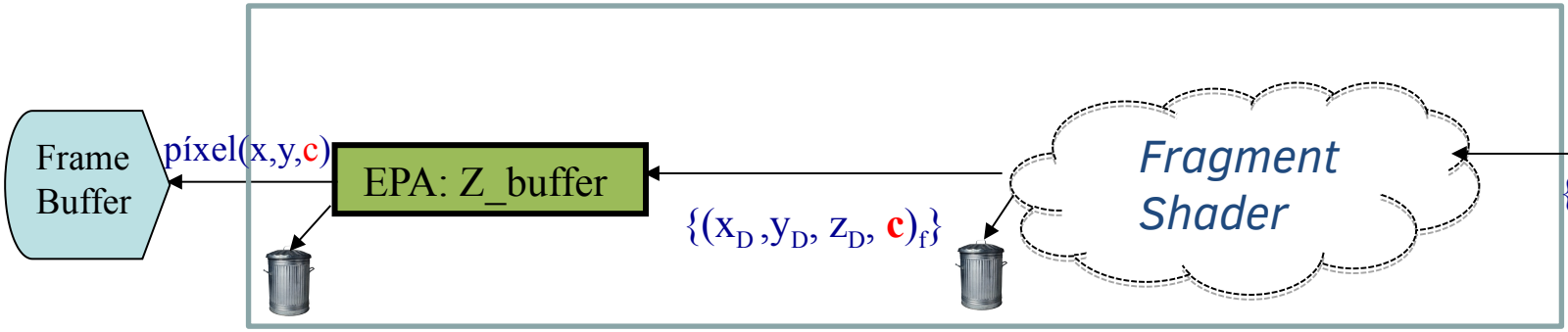
$$c_l = \frac{r \cdot c_1 + (a - r) \cdot c_2}{a}$$

Pintar/visualitzar en OpenGL 3.3









Processat de fragments: El fragment Shader

Fragment Shader

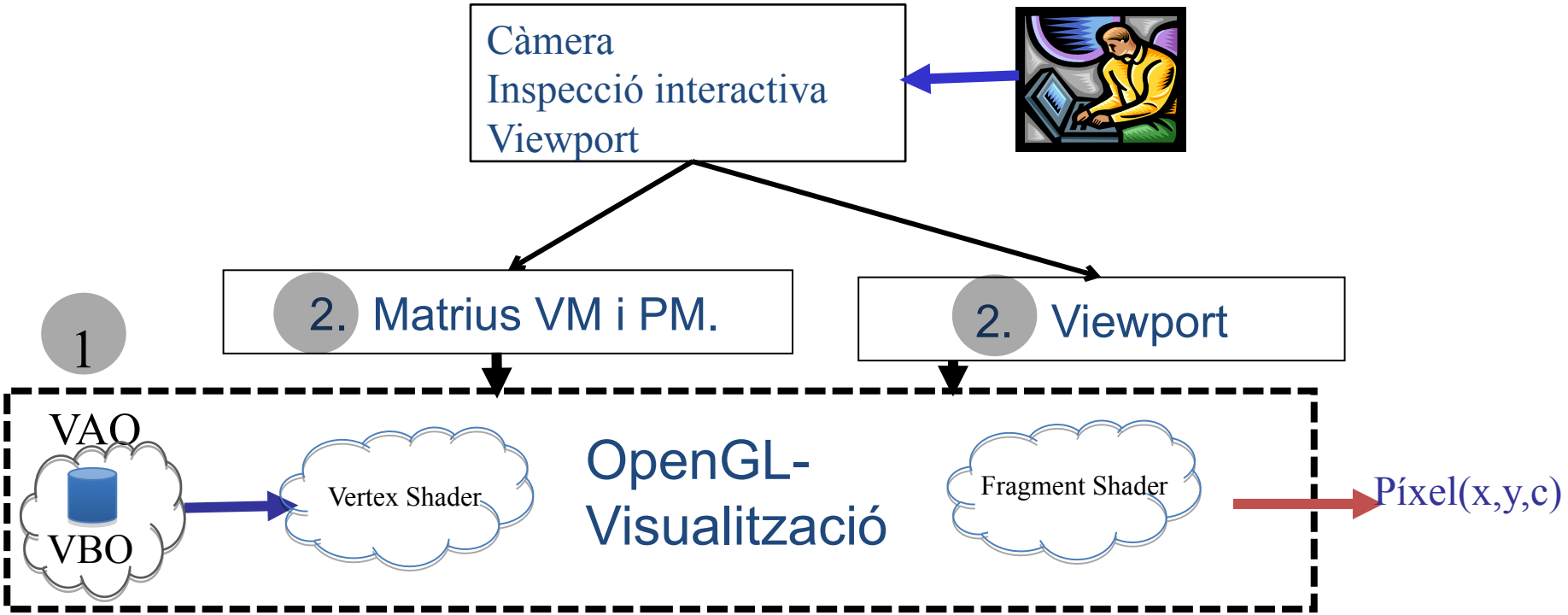
```
#version 330 core

in ...

out vec4 FragColor;

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

Pintar/visualitzar en OpenGL 3.3 (resum)



3. Pinta_Model()

// Activa VAO i crida a glDrawArrays(...)

Classe 4: Conceptes i preguntes

- El procés de visualització projectiu: blocs funcionals que l'integren, ordre dels processos, sistemes de coordenades.
- Diferència entre vèrtex i fragment.
- Què cal fer, com a mínim, en el fagment shader?
- Què son les coordenades normalitzades?
- Què és i com funciona el retallat? Per què cal?
- Com i quan es passa a coordenades de dispositiu? Què són aquestes coordenades exactament?
- Què passa amb els `out` addicionals que puguem afegir al vertex shader? Com arriba aquesta informació al fragment shader?