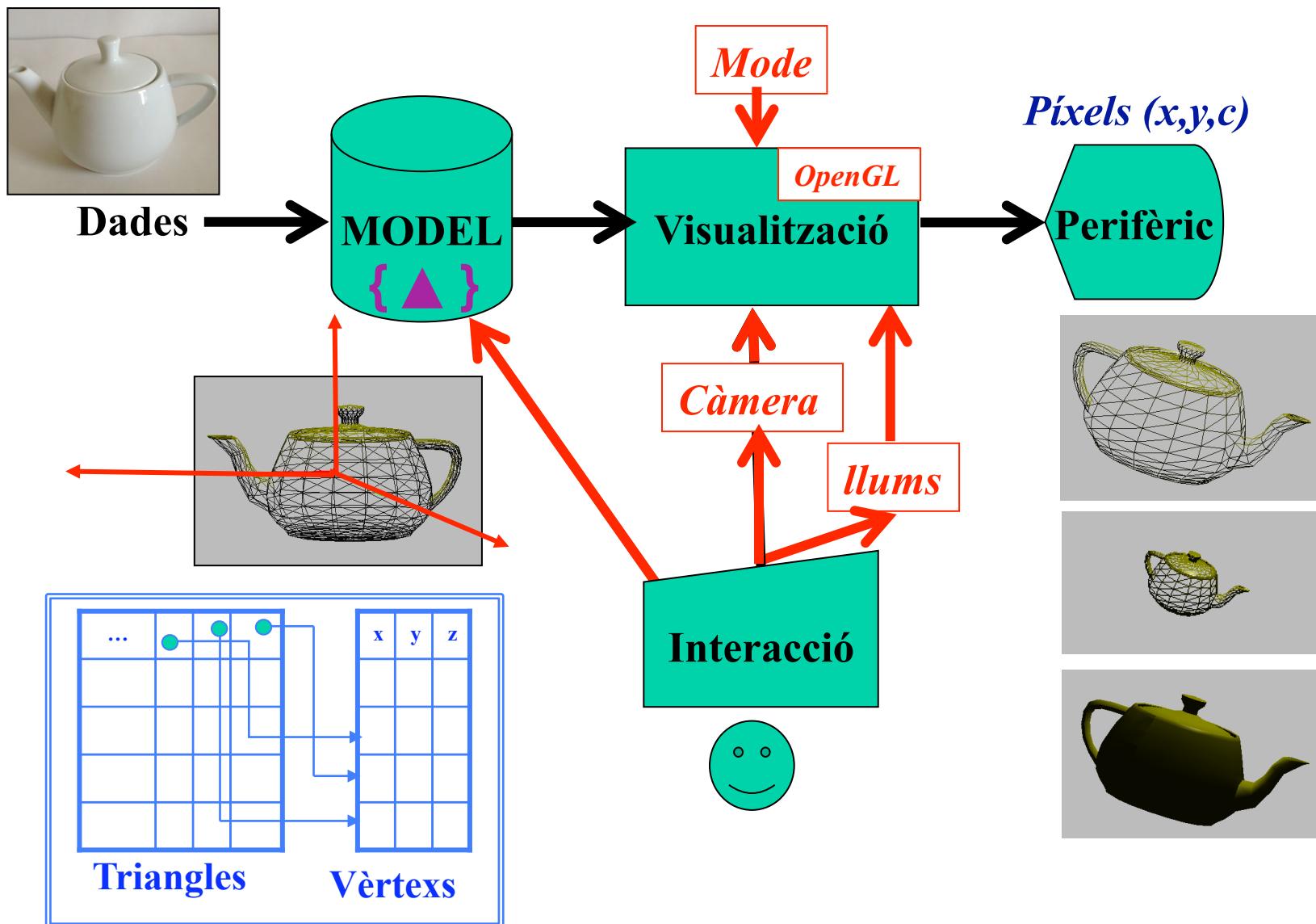


# Classe 2: contigut

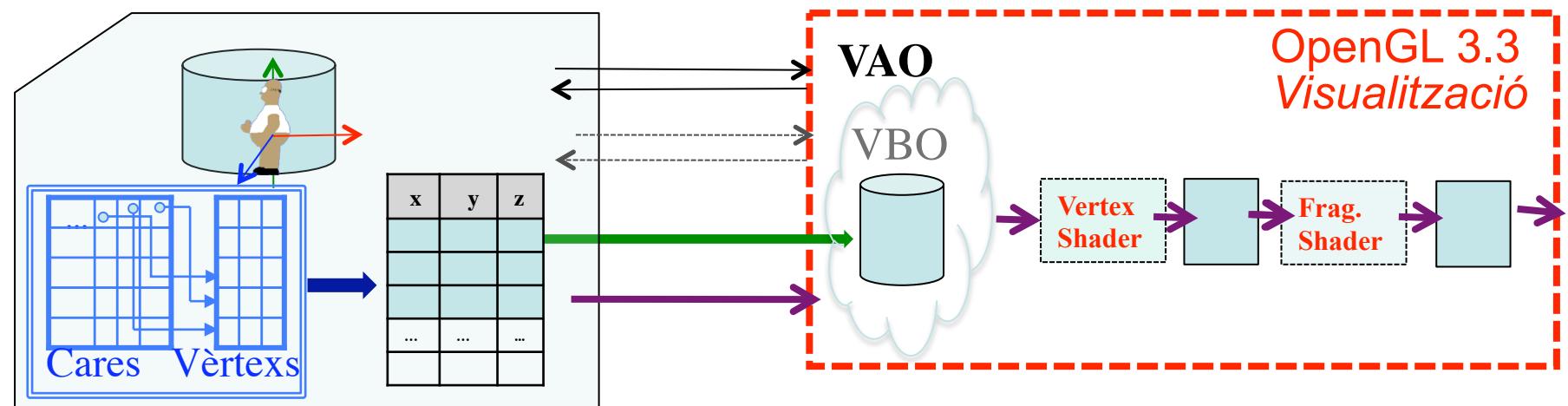
- Procés de Visualització (1)
- Models geomètrics (2): Escenes
- Breu repàs de TG i primers exercicis de TG

# Visualització (intro)



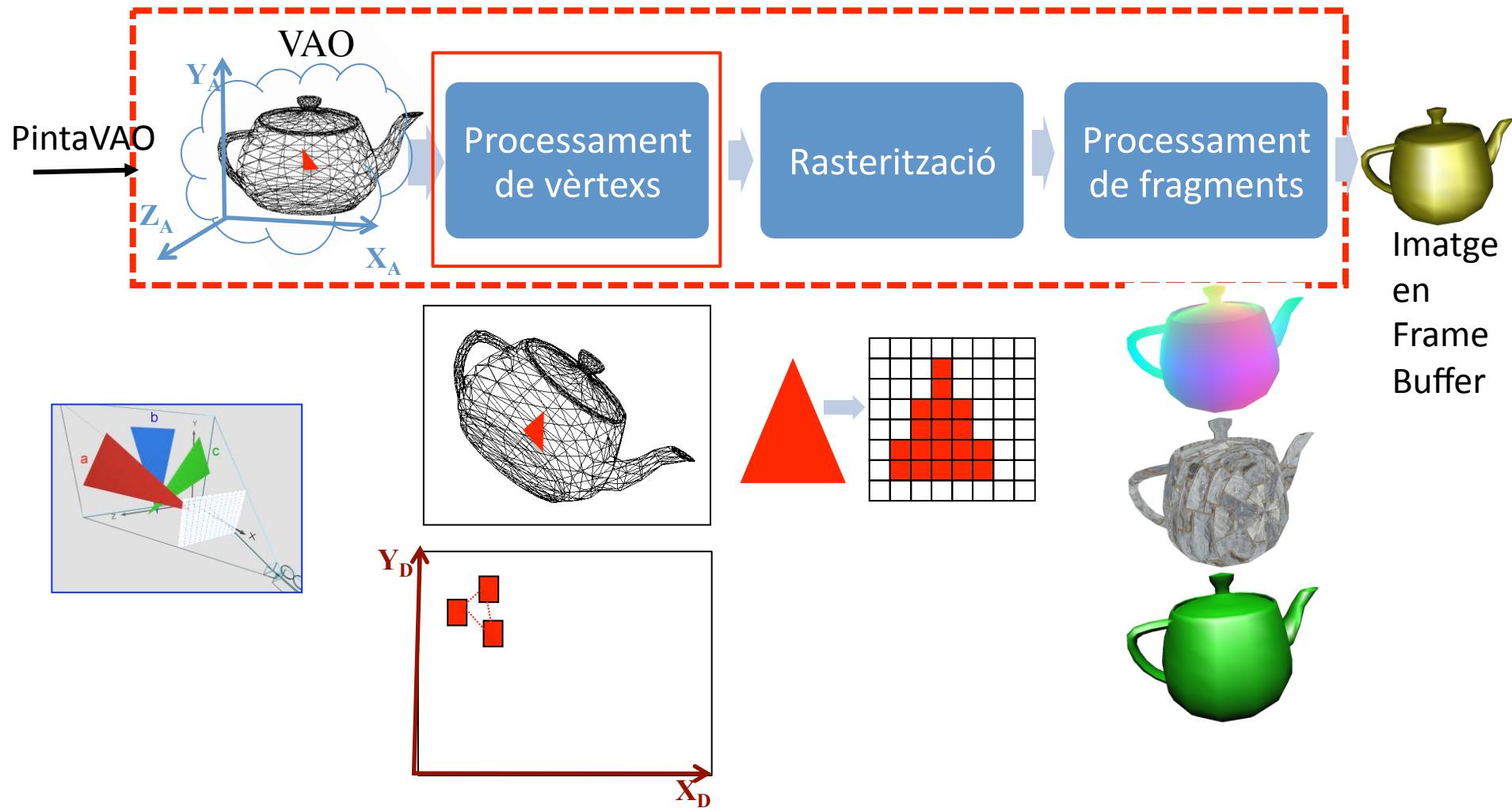
# Pintar en OpenGL 3.3: “core” mode

1. Crear en GPU/OpenGL un *VAO i VBOs*
2. Guardar llista de vèrtexs (amb repetició) en un *VBO*
3. Cada cop que es requereix pintar, indicar el *VAO* a pintar i dir que es pinta: *glDrawArrays(...)*. Acció **pinta\_model()** a teoria.



Aplicació. Model Geomètric

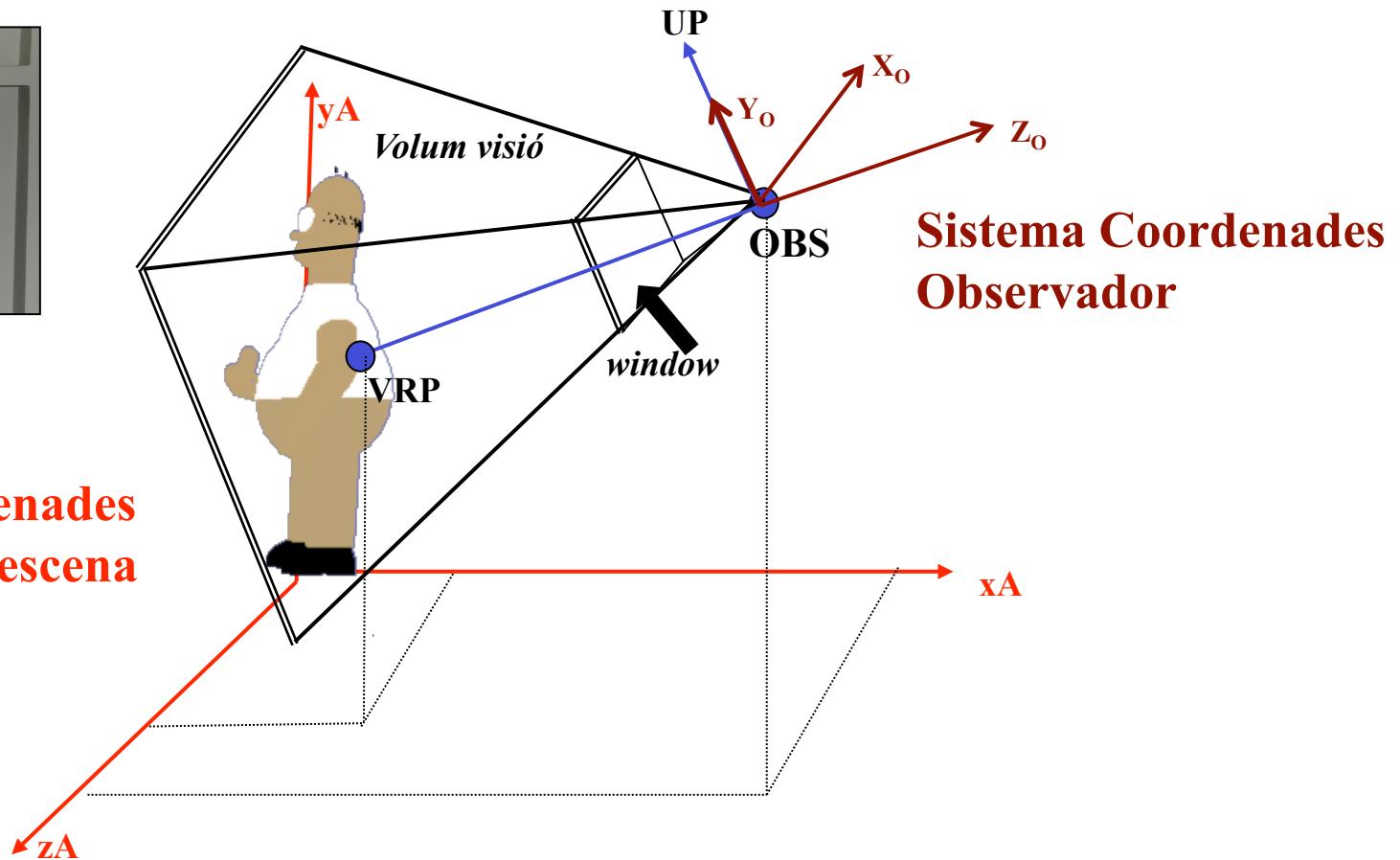
# Paradigma projectiu bàsic amb OpenGL 3.3



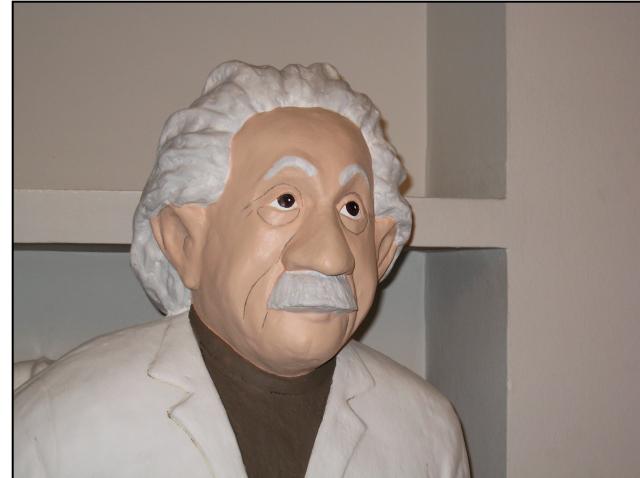
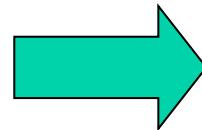
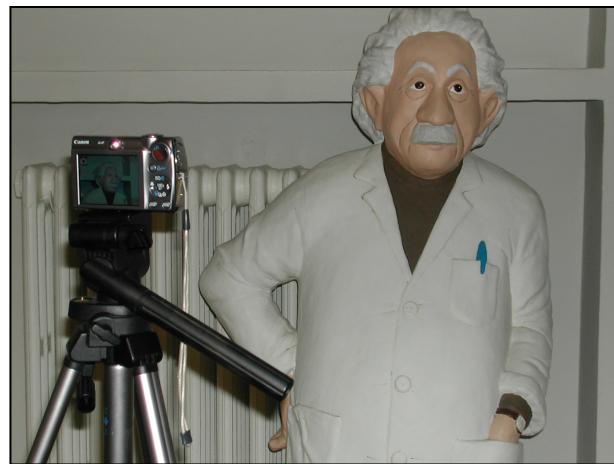
# Com indicar càmera?



Sistema Coordenades  
Aplicació/món/escena



1. Ubicació respecte SCA: obs, vrp, up
2. Definir Volum de Visió: òptica



1. Posició, orientació  
2. Òptica

3. Fer la Foto

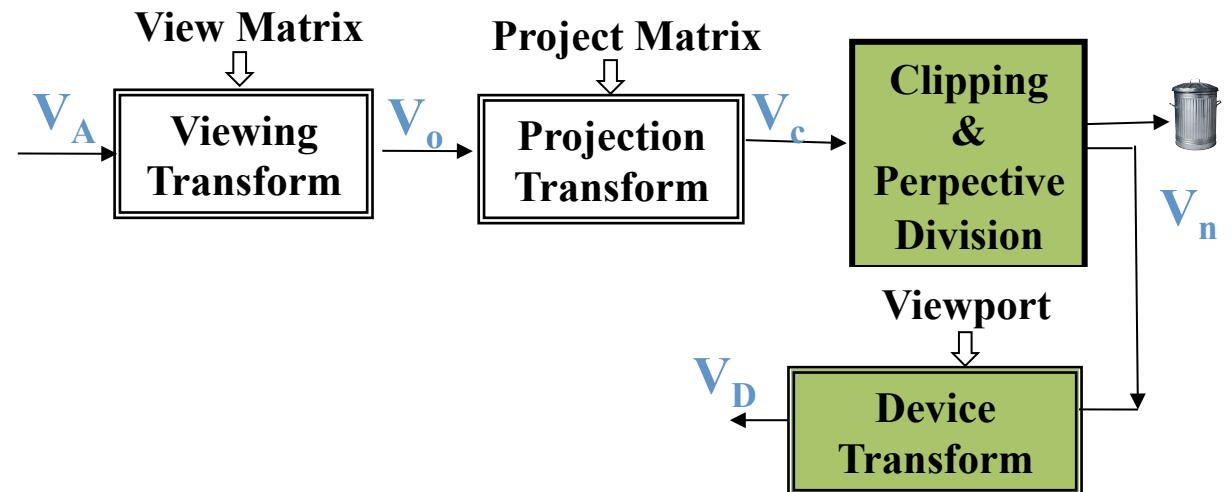
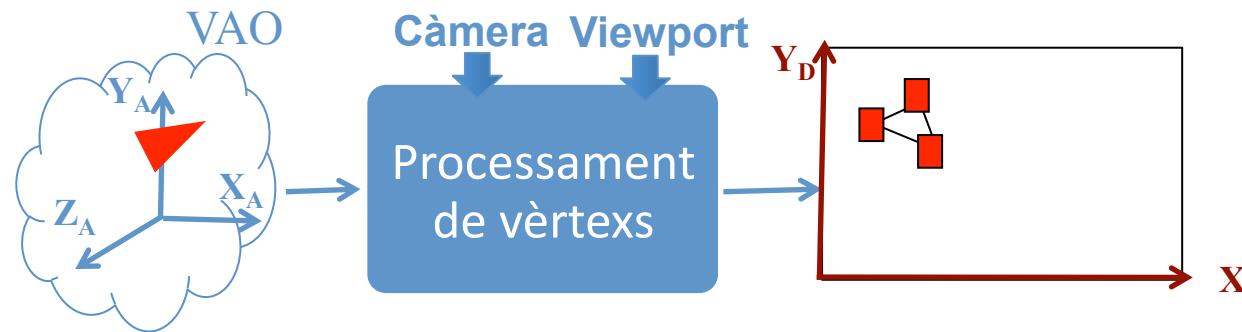
4. Emmarcar



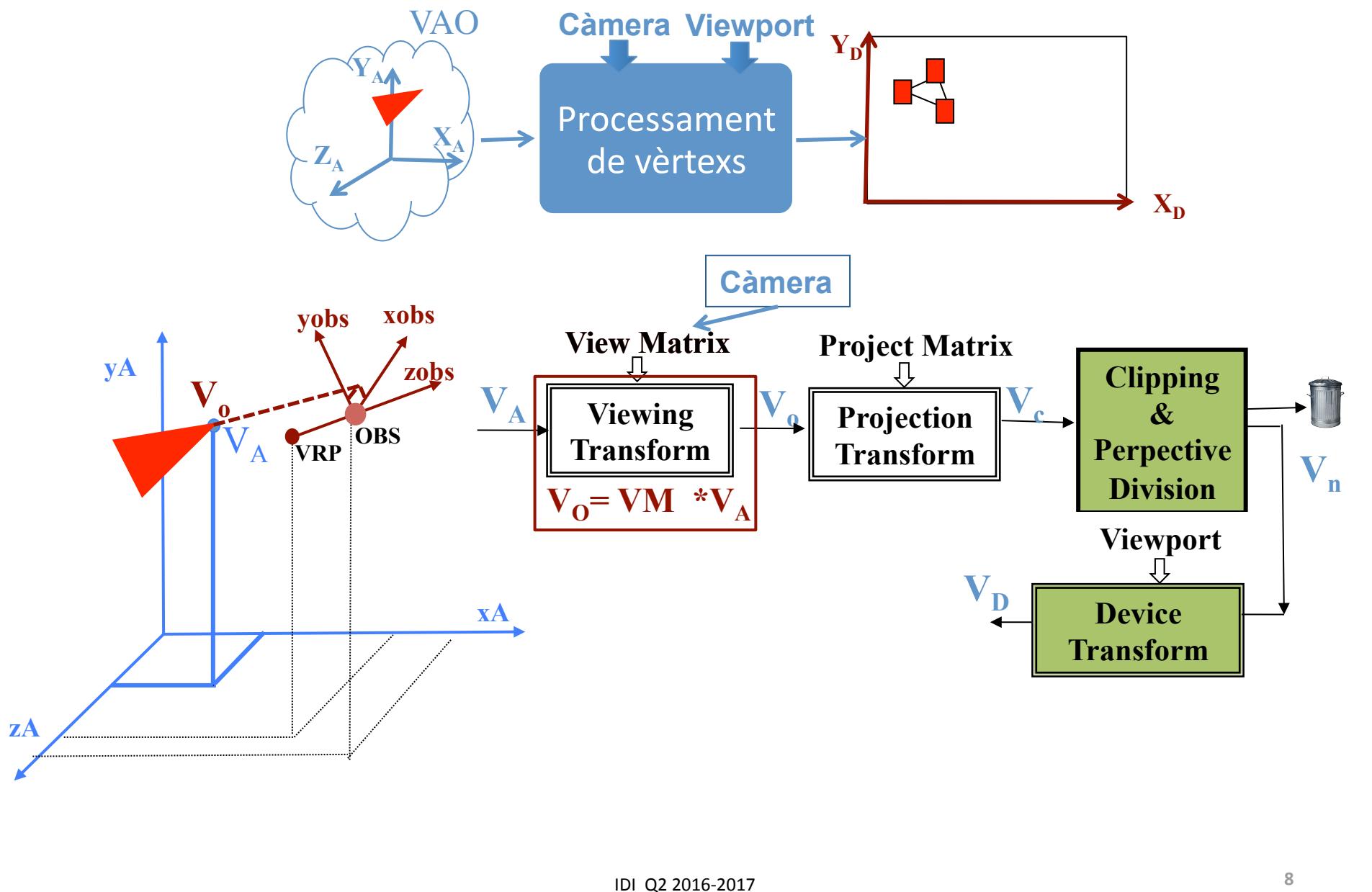
Usualment:

- El viewport és tota la finestra OpenGL
- De moment, no ens preocuparem de si hi ha “deformacions”

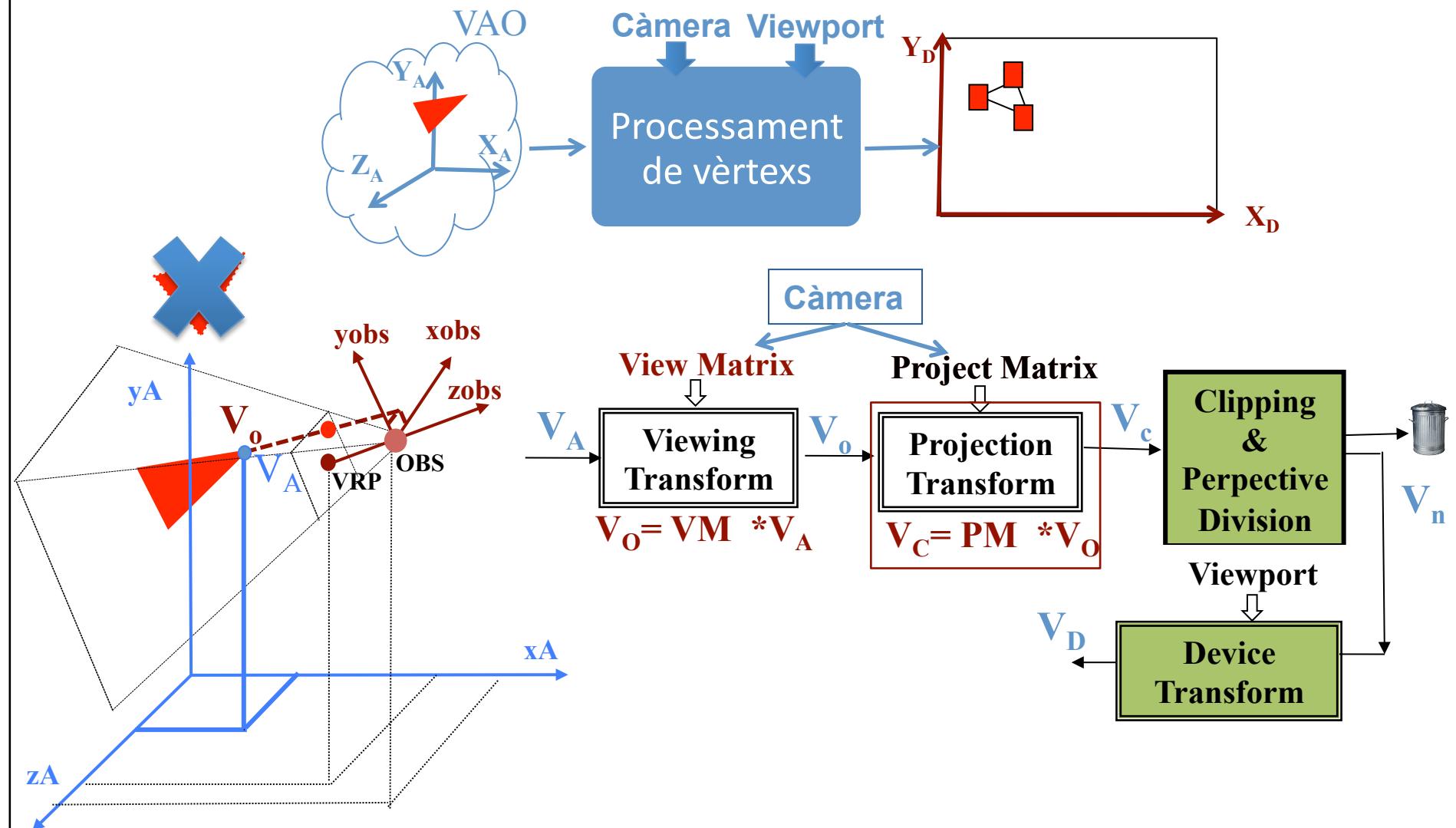
# Paradigma projectiu bàsic amb OpenGL 3.3



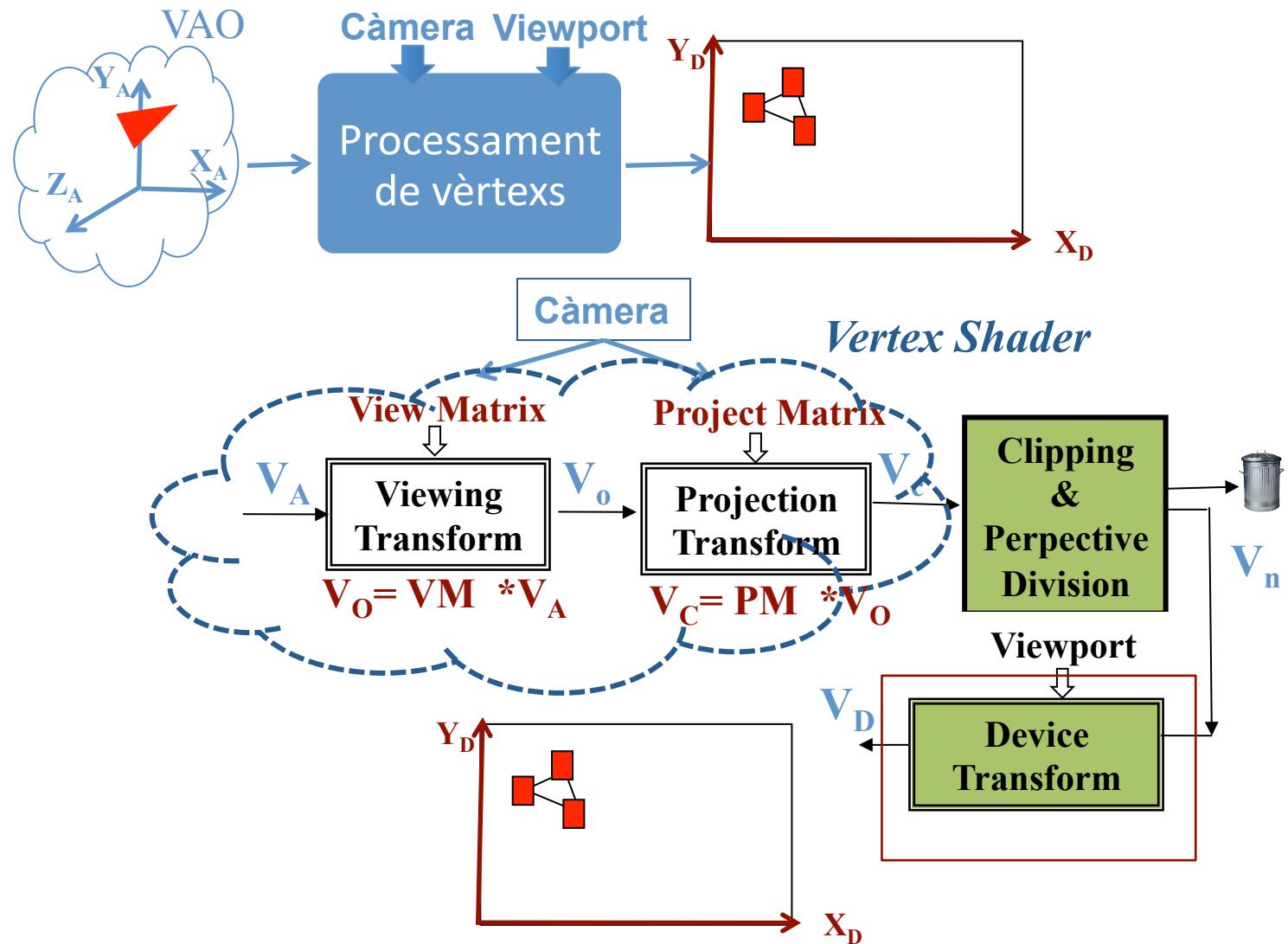
# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3

## *Vertex Shader*

```
#version 330 core

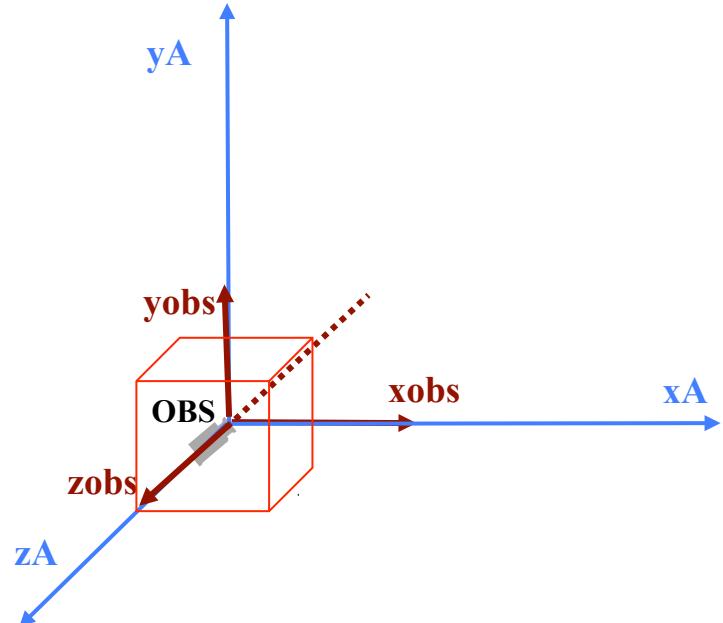
in vec3 vertex;
uniform mat4 PM;
uniform mat4 VM;

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

# Paradigma projectiu bàsic amb OpenGL 3.3

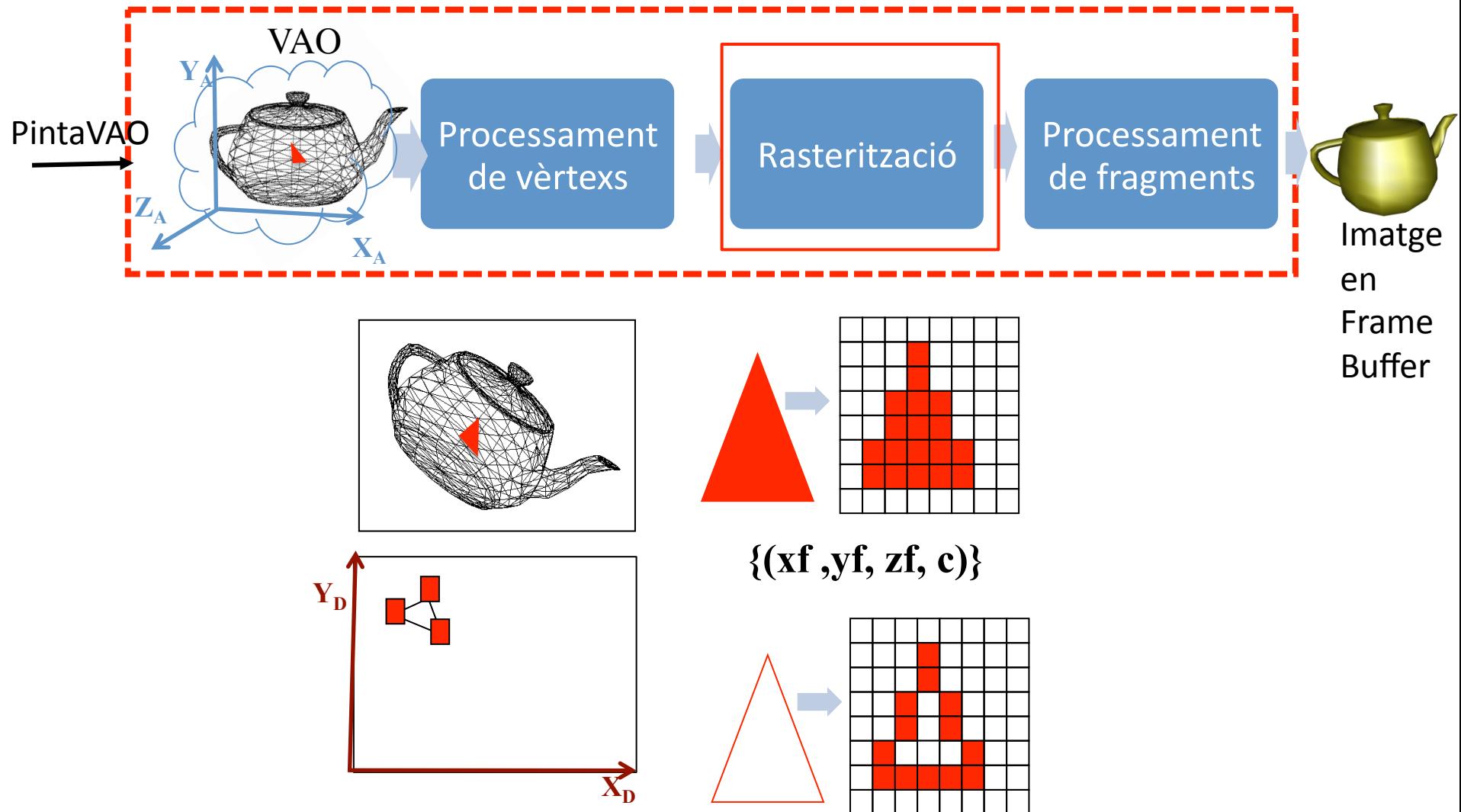
## *Vertex Shader*

```
#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);
}
```

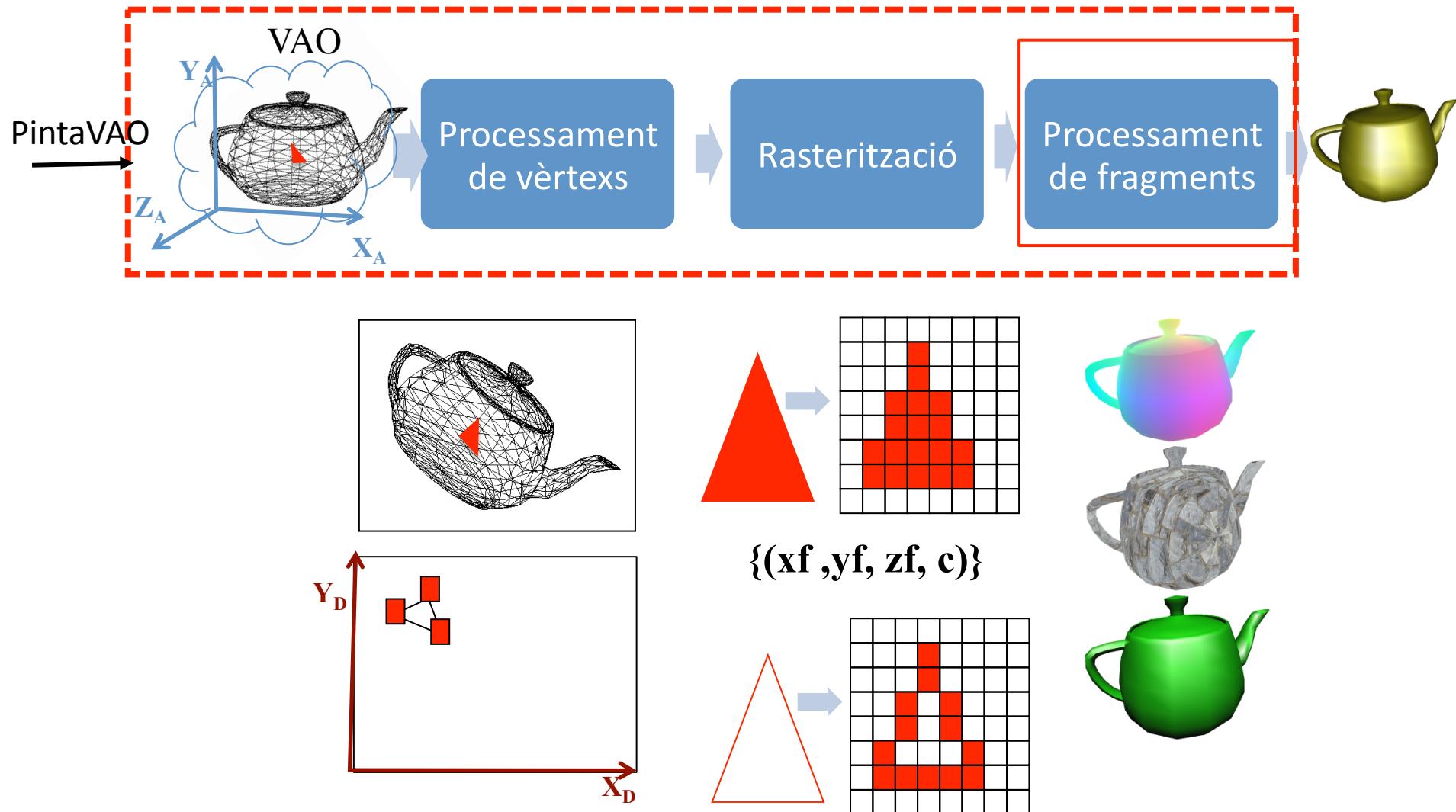


Volum de Visió cub de  $(-1, -1, -1)$  a  $(1, 1, 1)$

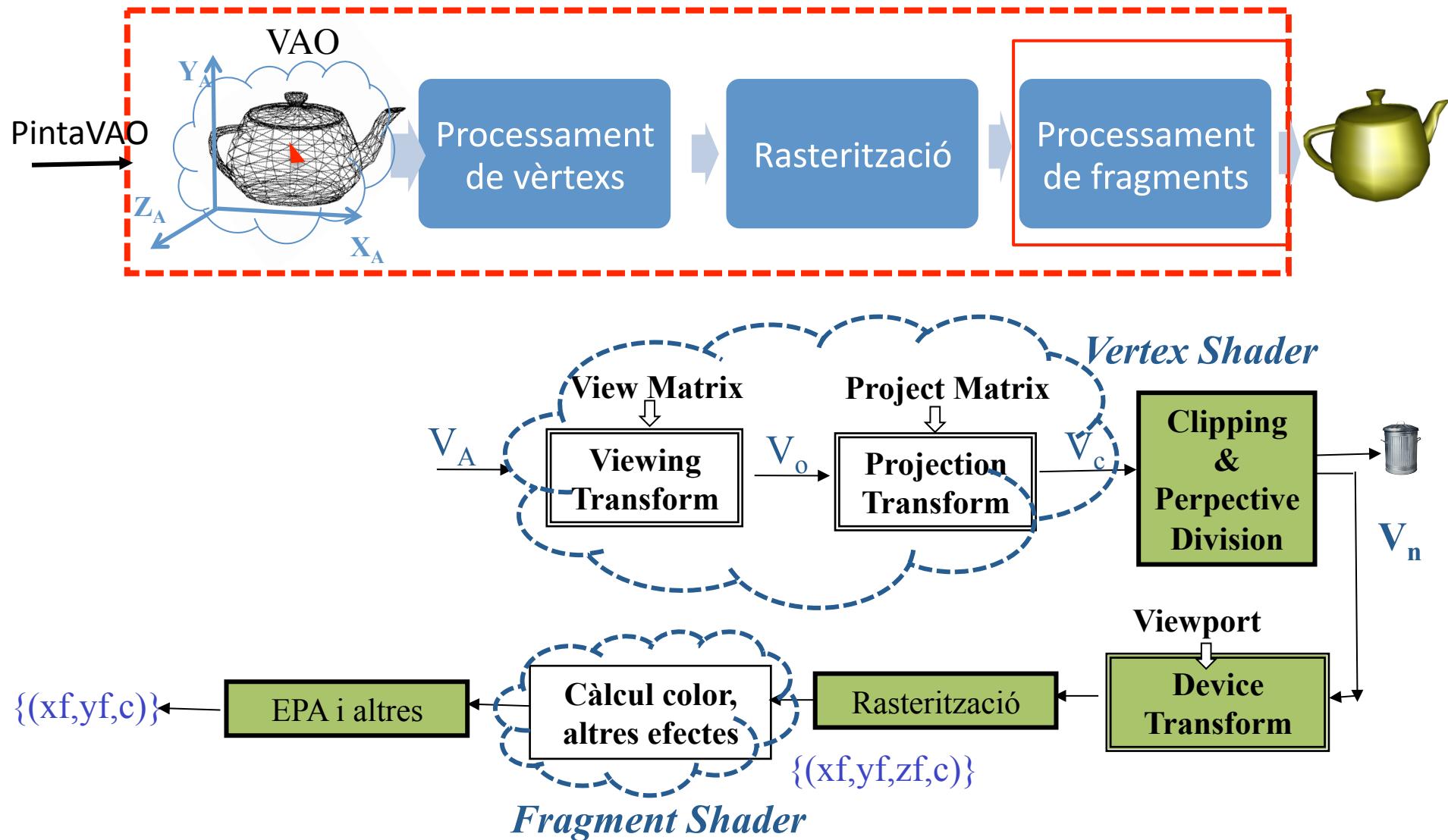
# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3



# Paradigma projectiu bàsic amb OpenGL 3.3



## *Fragment Shader*

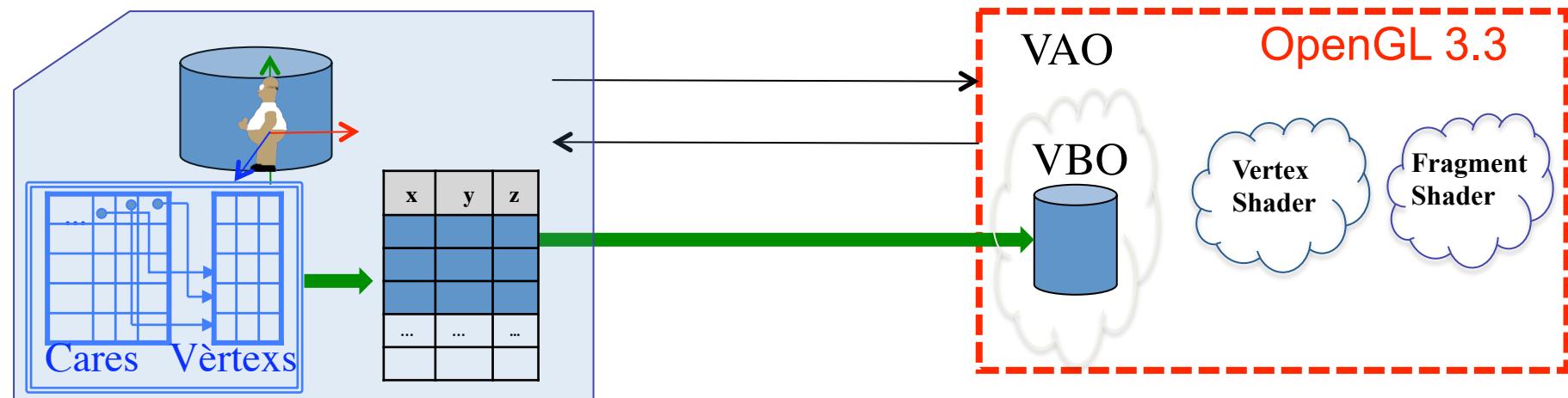
```
#version 330 core

out vec4 FragColor;

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

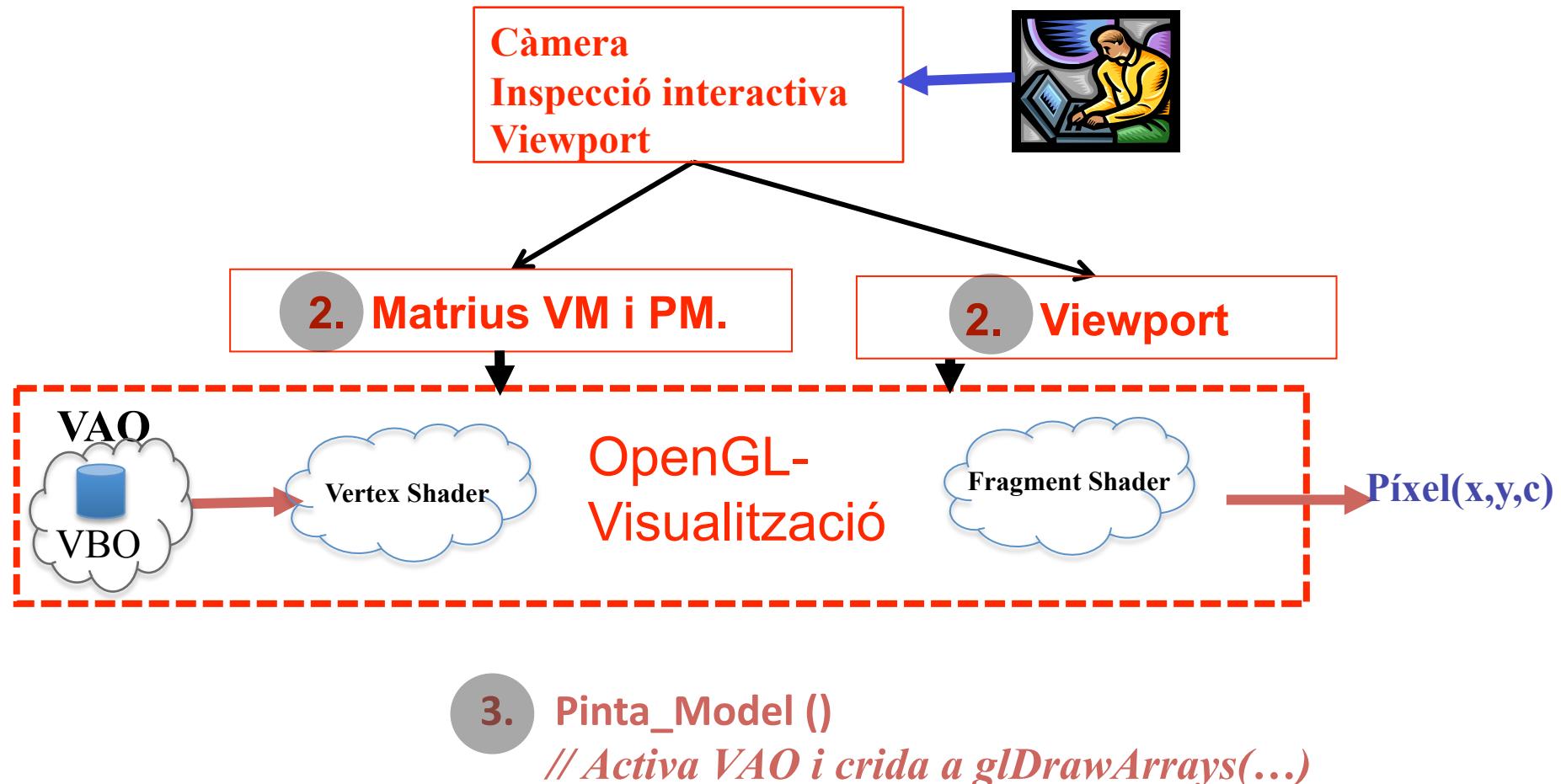
# Visualitzar en OpenGL 3.3: “core” mode

1. Generar VAO per cada model 3D que tinguem a memòria (NOMÉS cal fer-ho un cop si no modifiquem el model).



Aplicació. Model Geomètric

# Visualització OpenGL i assignatura



# Classe 2: contigut

- Visualització (1)
- Models geomètrics (2na part): Escenes
- Breu repàs de TG i primers exercicis de TG