

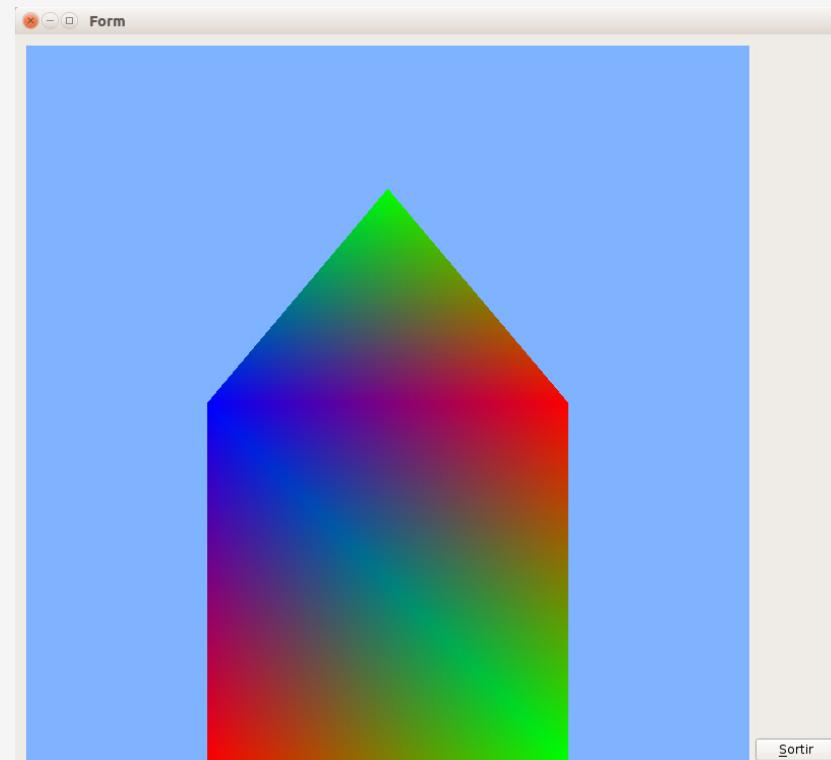
Laboratori OpenGL – Sessió 2.1

Bloc 2

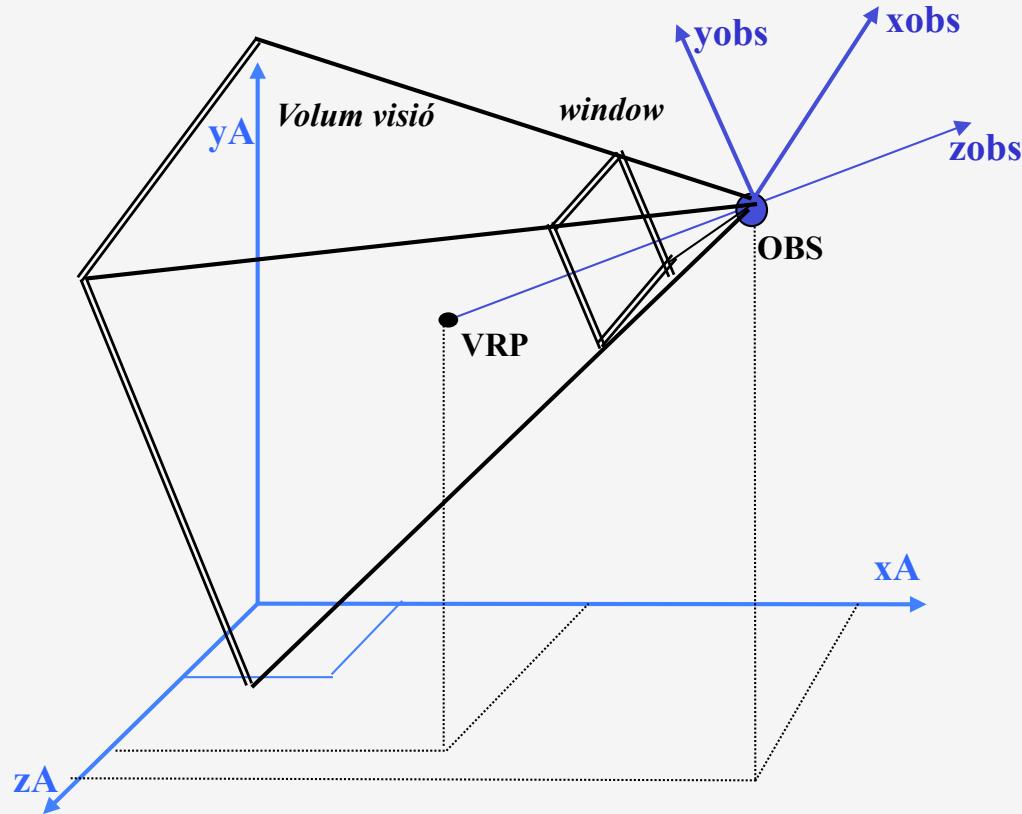
- Nou exemple de base
/assig/idi/blocs/bloc-2
- Transformacions de càmera amb glm
(view i projection)
- Classe Model – càrrega d'objectes OBJ
- Z-buffer

Nou exemple de base

- Pinta un objecte
- Inclou transformació de model
- Vertex i Fragment Shaders pinten amb color per vèrtex

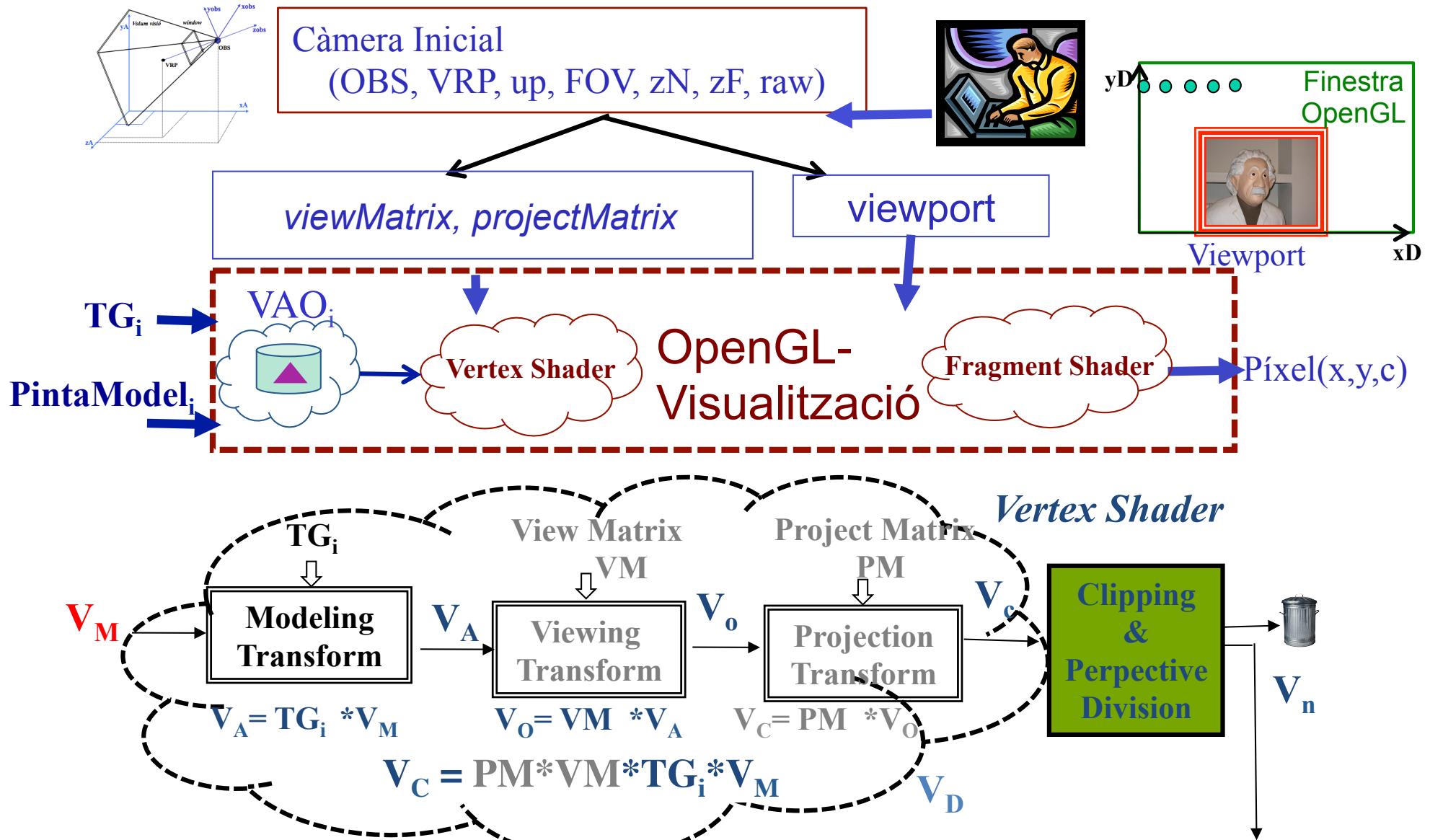


Càmera: OBS, VRP, up, zN, zF, FOV, raw



OBS, VRP, up, FOV, ra, zNear, zFar

Recordatori: Càmera i procés de visualització



Transformació de projecció

(exercici 1)

Definir una òptica perspectiva amb les paràmetres:

- $\text{FOV} = \text{M_PI}/2$.
- $\text{ra} = 1$
- $\text{znear} = 0.4$
- $\text{zfar} = 3$

Posició de la càmera de defecte (matriu identitat) →
OBS en origen de SCA, $\text{up} = (0, 1, 0)$, VRP sobre z-

Transformació de projecció

(exercici 1)

- Al codi .cpp de MyGLWidget:
 - Demanem un uniform location per al uniform de la matriu
`projLoc = glGetUniformLocation (program->programId(), "proj")`
 - Definim un mètode que ens calculi la transformació de projecció i enviï el uniform amb la matriu cap al vertex shader (cal que els paràmetres siguin floats)
`void MyGLWidget::projectTransform () {
 // glm::perspective (FOV en radians, ra window, znear, zfar)
 glm::mat4 Proj = glm::perspective ((float)M_PI/2.0f, 1.0f, 0.4f, 3.0f);
 glUniformMatrix4fv (projLoc, 1, GL_FALSE, &Proj[0][0]);
}`

Transformació de projecció (exercici 1)

- Al vertex shader (afegir):

...

```
uniform mat4 proj;
```

...

```
void main () {
```

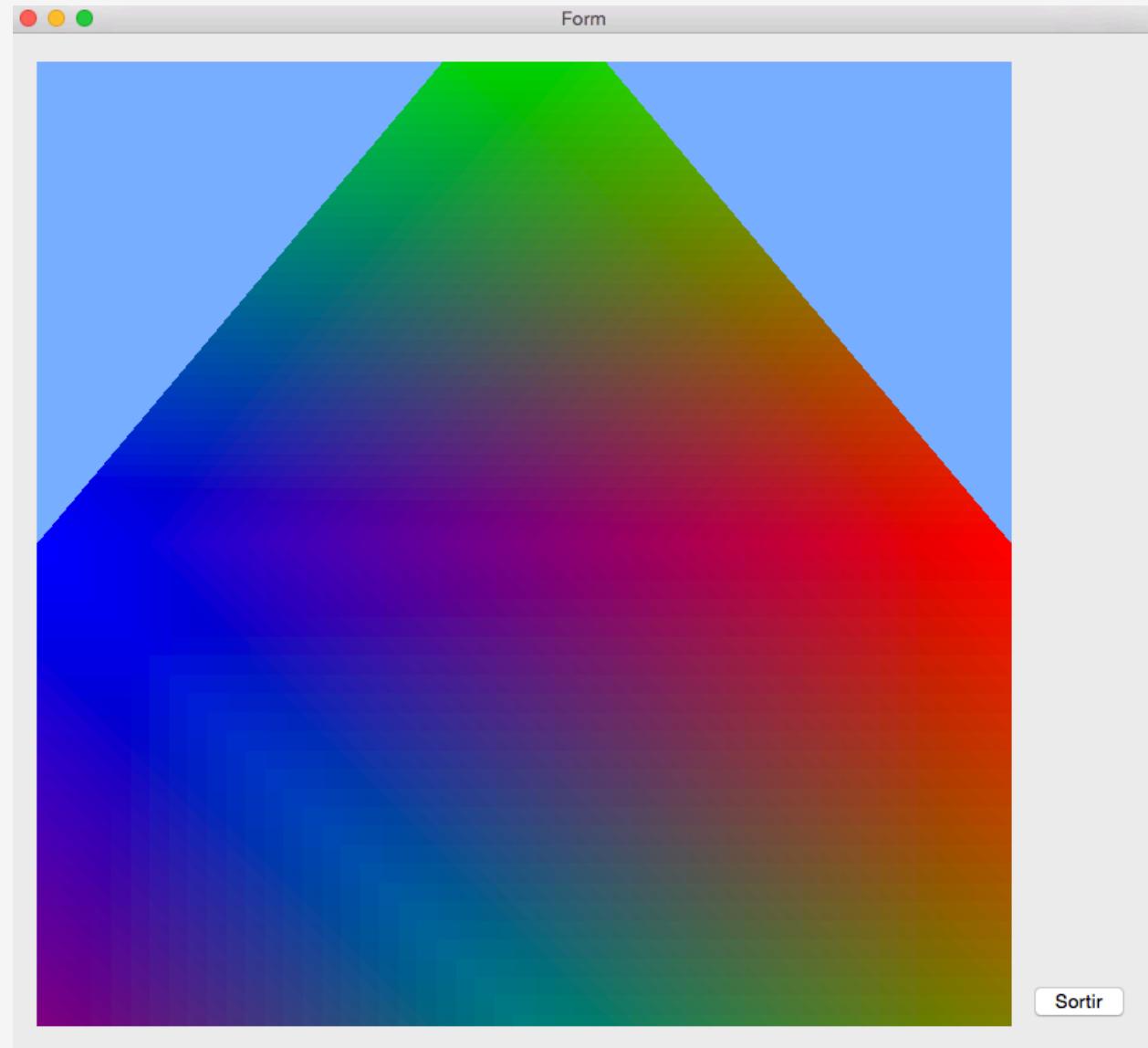
...

```
    gl_Position = proj * ... * vec4 (vertex, 1.0);
```

```
}
```

Transformació de projecció

(exercici 1)

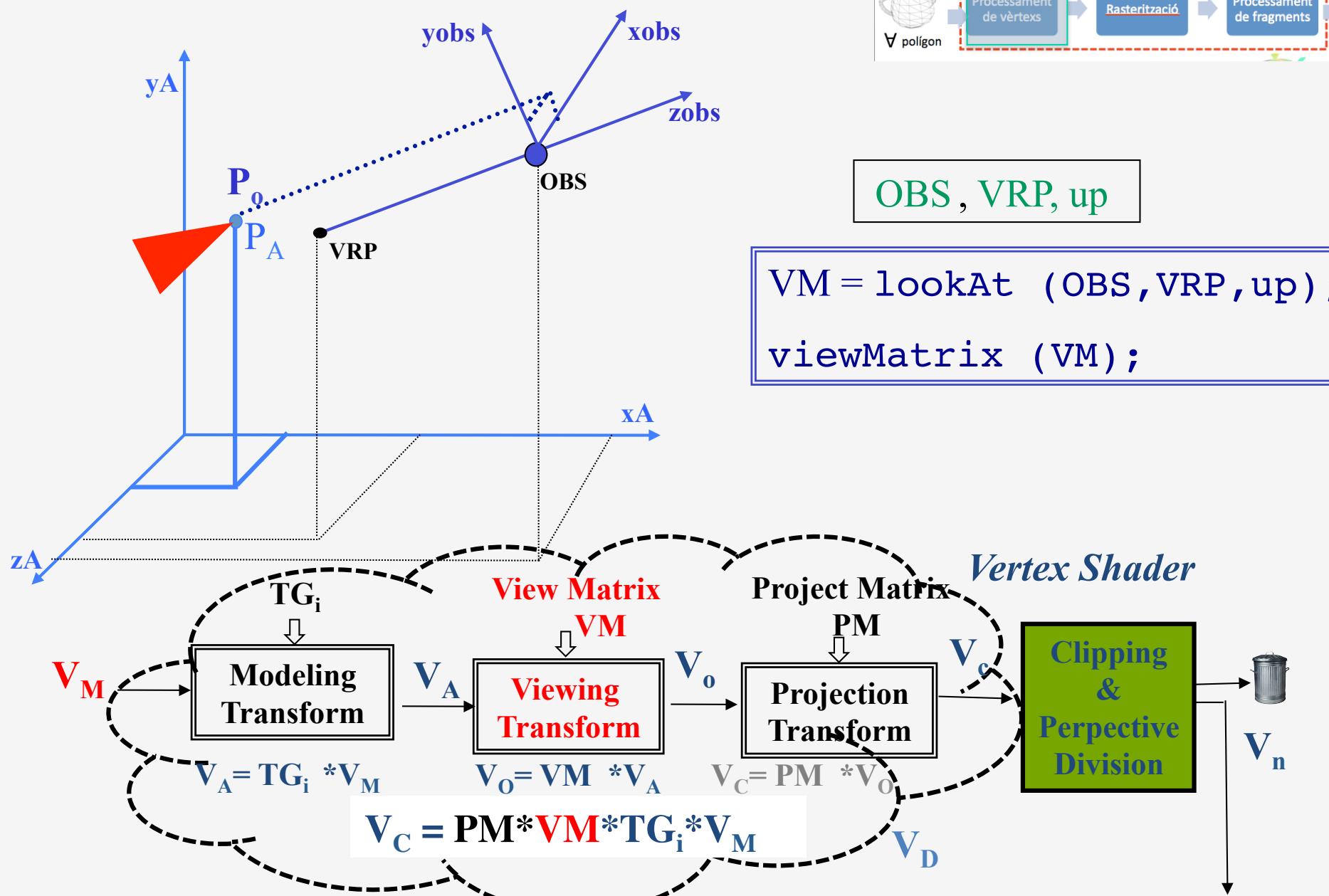


Transformació de punt de vista (exercici 2)

Definir una posició càmera amb les paràmetres:

- OBS= $(0,0,1)$
- VRP= $(0,0,0)$
- up = $(0,0,1)$

Transformació de punt de vista (view)



Transformació de punt de vista (view) (exercici 2)

- Al codi .cpp de MyGLWidget:
 - Demanem un uniform location per al uniform de la matriu
viewLoc = glGetUniformLocation (program->programId(), “view”)
 - Definim un mètode que ens calculi la transformació de punt de vista (view) i enviï el uniform amb la matriu cap al vertex shader
void MyGLWidget::viewTransform () {
 // glm::lookAt (OBS, VRP, UP)
 glm::mat4 View = glm::lookAt (glm::vec3(0,0,1),
 glm::vec3(0,0,0), glm::vec3(0,1,0));
 glUniformMatrix4fv (viewLoc, 1, GL_FALSE, &View[0][0]);
}

Transformació de punt de vista (view) (exercici 2)

- Al vertex shader (afegir):

...

```
uniform mat4 view;
```

...

```
void main () {
```

...

```
    gl_Position = proj * view * ... * vec4 (vertex, 1.0);
```

```
}
```

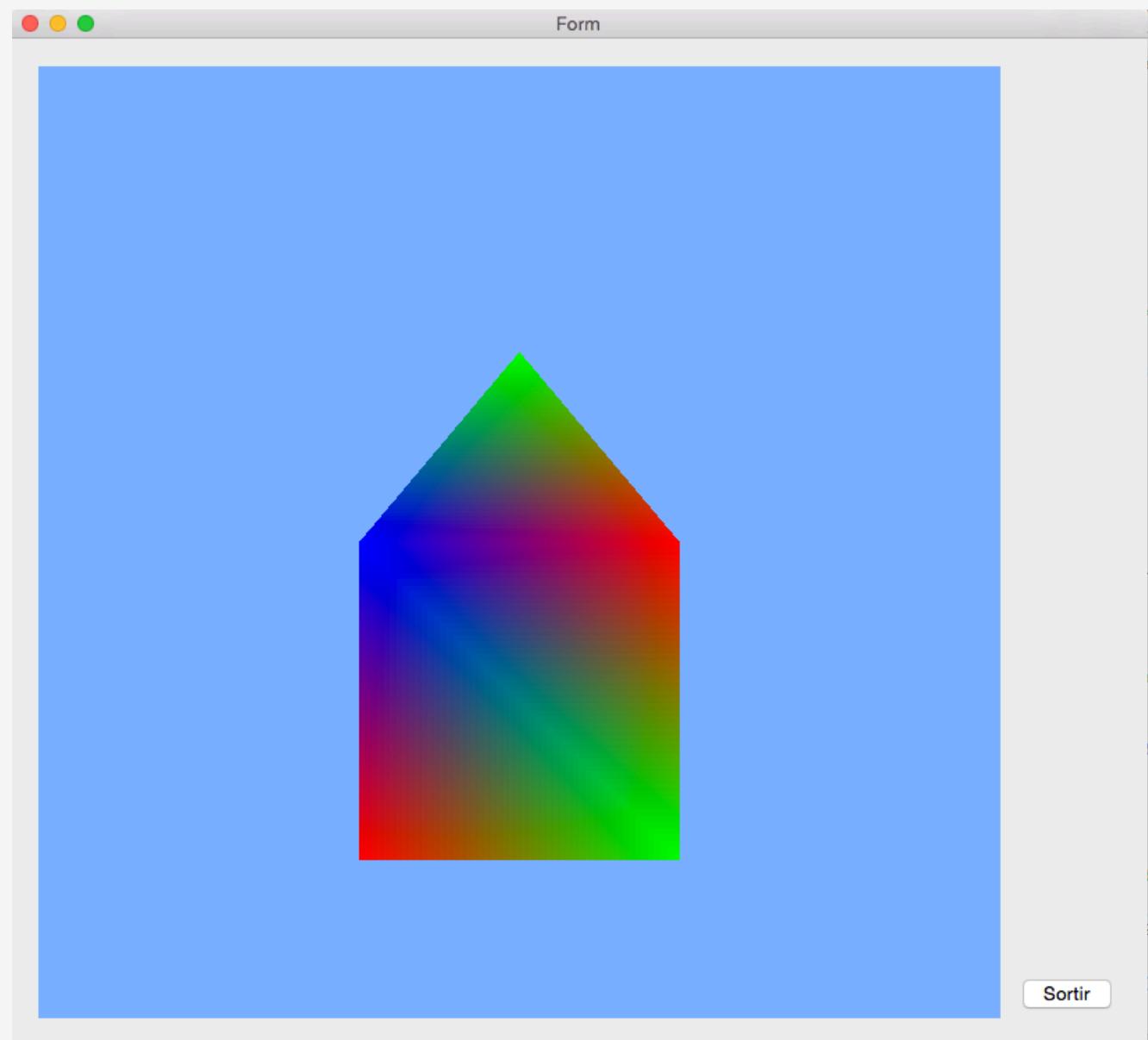
Transformació de punt de vista (view) (exercici 2)

Recomanació. Fer mètode ini_càmera() per:

- inicialitzar els paràmetres de la càmera
- Crida a viewTransform(), projectTransform()

(exercici 3)

- Modificar **up**

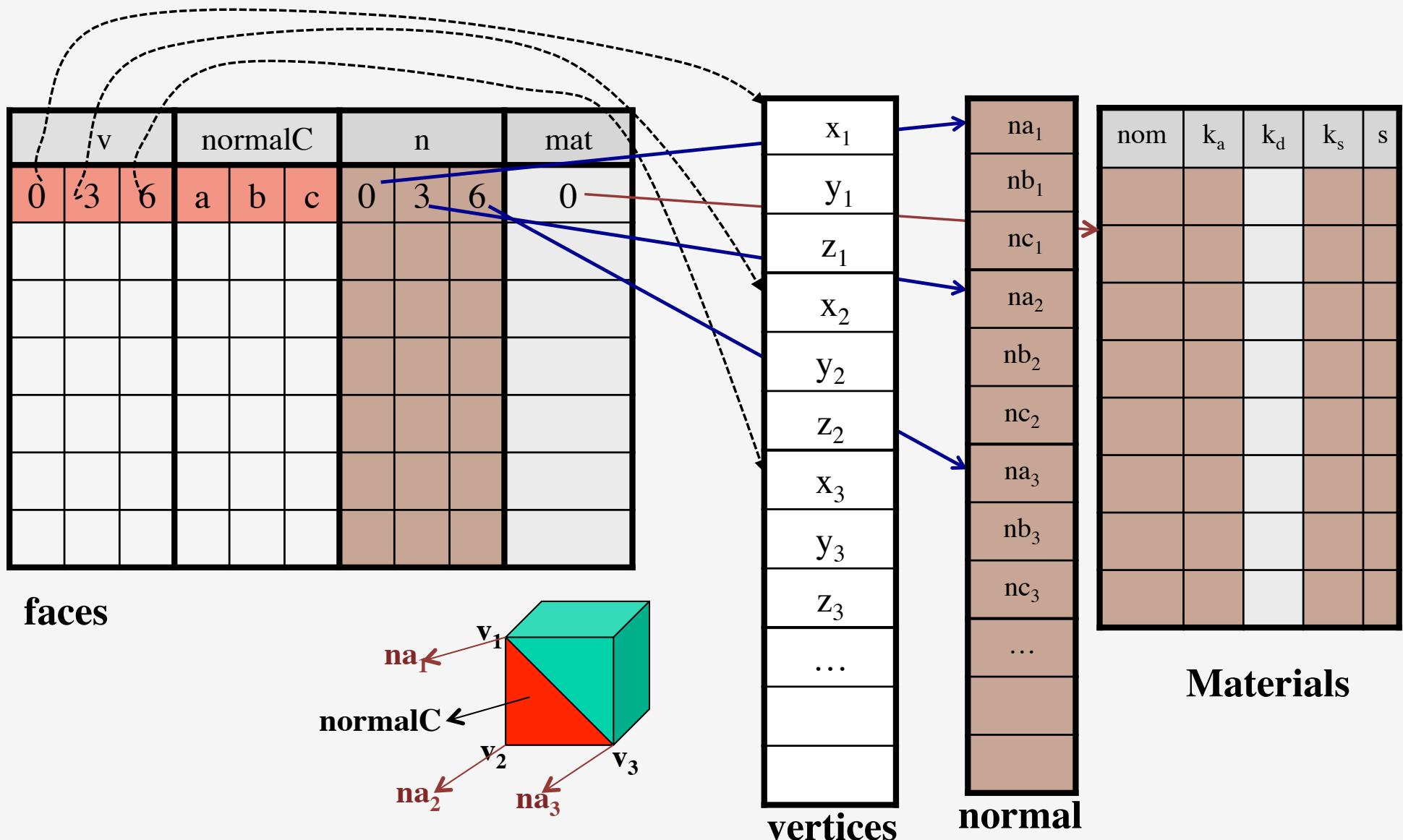


Càrrega de models OBJ (exercici 4)

- Classe Model: permet carregar *objecte.obj*
 - `/assig/idi/Model` (copieu-vos la carpeta en un directori vostre)
 - Analitzeu el `model.h` (classe Model)
 - Mètode **Model::load(std::string filename)**
Inicialitza les estructures de dades a partir d'un model en format OBJ-Wavefront en disc
- Modifiqueu el fitxer .pro afegint
 - `INCLUDEPATH += <el-vostre-directori>/Model`
 - `SOURCES += <el-vostre-directori>/Model /model.cpp`
- En `/assig/idi/models` trobareu models d'objectes.
 - Si els copieu a un directori local, per cada .obj copieu també (si existeix) el .mtl → definició dels materials corresponents.
 - Fins la propera sessió usarem el **HomerProves**
- Més models els podeu trobar a la xarxa.

```
TEMPLATE    = app
QT          += opengl
INCLUDEPATH += /usr/include/glm
INCLUDEPATH += ./Model
FORMS       += MyForm.ui
HEADERS     += MyForm.h MyGLWidget.h
SOURCES     += main.cpp MyForm.cpp \
               MyGLWidget.cpp ../Model/model.cpp
```

Representació classe Model



Analitzeu l'arxiu **model.h**

Compte!! amb el nom dels camps de Material que en l'esquema són simbòlics; p.e. k_d és `float diffuse[4]`

Representació auxiliar de la classe Model

x_1
y_1
z_1
x_2
y_2
z_2
x_3
y_3
z_3
...

VBO_vertices

nx_1
ny_1
nz_1
nx_2
ny_2
nz_2
nx_3
ny_3
nz_3
...

VBO_normals

r_1
g_1
b_1
r_2
g_2
b_2
r_3
g_3
b_3
...

VBO_matamb

r_1
g_1
b_1
r_2
g_2
b_2
r_3
g_3
b_3
...

VBO_matdiff

r_1
g_1
b_1
r_2
g_2
b_2
r_3
g_3
b_3
...

VBO_matspec

sh_1
sh_2
sh_3
...

VBO_matshin

Ús de la classe Model (exercici 4)

- Construcció d'un objecte de tipus Model (declaració)

Model m; // un únic model

Model vectorModels[3]; // array de 3 models

vector<Model> models; // vector stl de models

- Càrrega d'un arxiu (model) .obj

m.load (“../models/HomerProves.obj”);

- Accés als seus VBOs (els genera la propia classe Model)

glBufferData (..., m.VBO_vertices (), GL_STATIC_DRAW); // posició

glBufferData (..., m.VBO_matdiff (), GL_STATIC_DRAW); // color

- Per a saber el nombre de cares (totes les cares són triangles)

m.faces().size()

sizeof(GLfloat) * m.faces ().size () * 3 * 3 // nombre de bytes dels buffers

Exemples

- Pas de dades del buffer de posicions cap a la GPU

```
glBufferData(GL_ARRAY_BUFFER,  
             sizeof(GLfloat) * m.faces ().size () * 3 * 3,  
             m.VBO_vertices (), GL_STATIC_DRAW);
```

- Pintar l'objecte

```
glDrawArrays(GL_TRIANGLES, 0, m.faces ().size () * 3);
```

- Recorregut de la taula de vèrtexs

```
for (unsigned int i = 0; i < m.vertices().size(); i+=3) {  
    // escric per pantalla les coordenades del vèrtex  
    std::cout << "(x, y, z) = (" << m.vertices()[i] << ", "  
           << m.vertices()[i+1] << ", "  
           << m.vertices()[i+2] << ")" << std::endl;  
}
```

Z-buffer (exercici 4)

- Algorisme de Z-buffer:
 - Activar el z-buffer (només cal fer-ho un cop!)
glEnable (GL_DEPTH_TEST);
 - Esborrar el buffer de profunditats a la vegada que el frame buffer
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

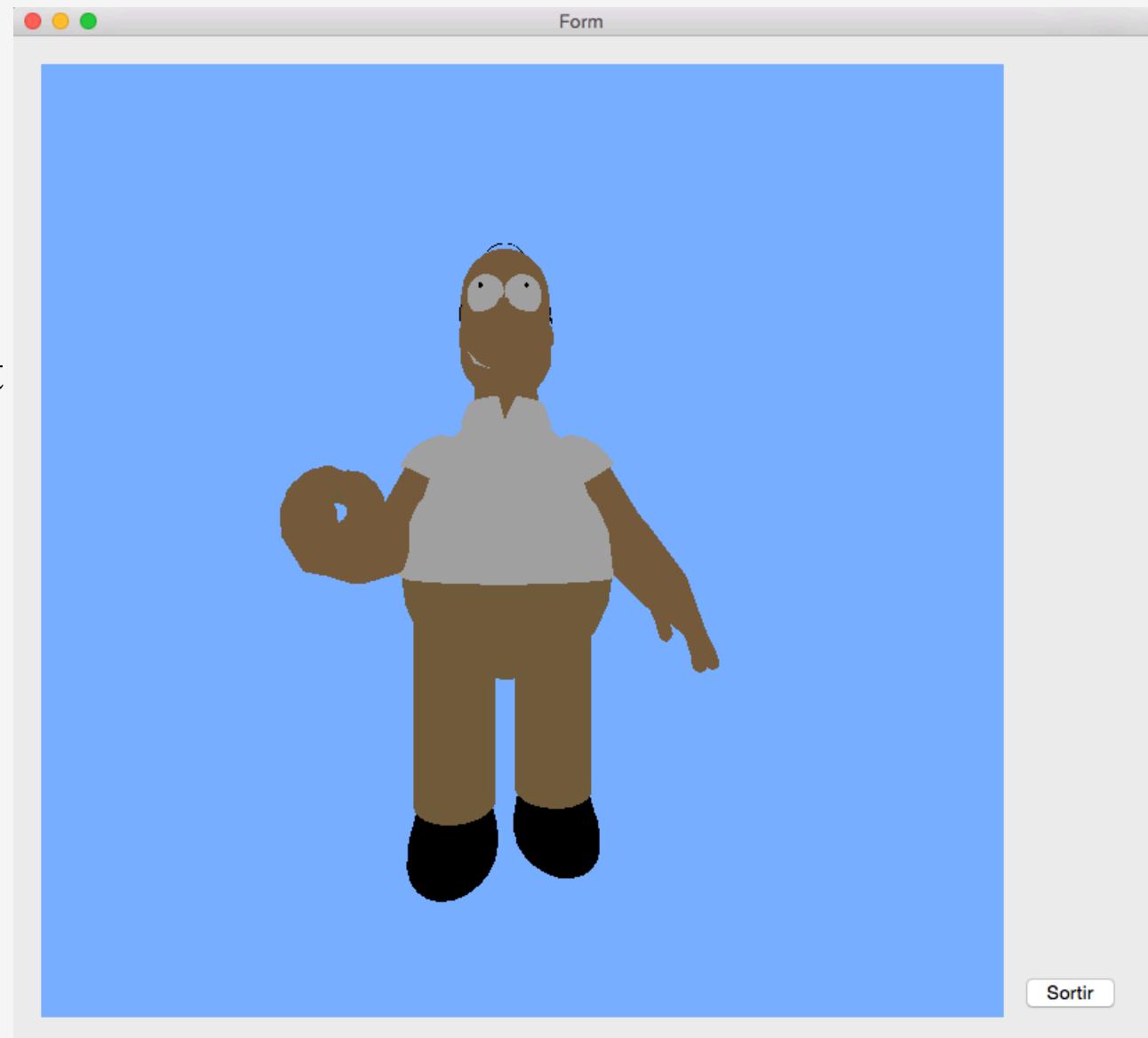
Càrrega i pintat del HomerProves (exercici 4)

(exercici 5)

Girar Homer amb teclat

(exercici 6)

Afegir un terra de 2x2
centrat en $(0, -1, 0)$



Exercicis sessió 2.1

El que cal que feu en aquesta sessió és:

- 1) **Mirar codi exemple bloc 2** (</assig/idi/blocs/bloc-2>) i entendre tot el que està programat.
- 2) **Feu TOTS els exercicis** que teniu al guió per a aquesta sessió. És important que els feu **en l'ordre** que es presenten.
 - Feu ús del que necessiteu del codi que s'ha presentat en aquestes transparències, però vigileu si feu *copy&paste* perquè copiar de pdf us pot portar problemes.