Laboratori INDI

Sessió 2.1

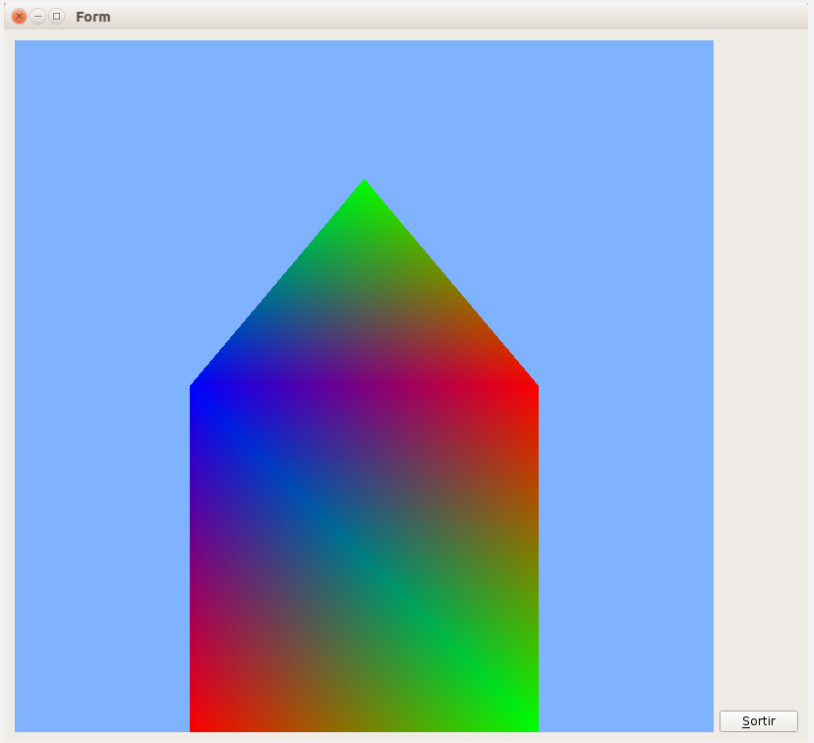
Taula de continguts

1. Nou exemple de base
2. Transformacions de càmera amb glm (view i projection)
3. Classe Model – càrrega d'objectes OBJ
4. Z-buffer
5. Exercicis

Taula de continguts

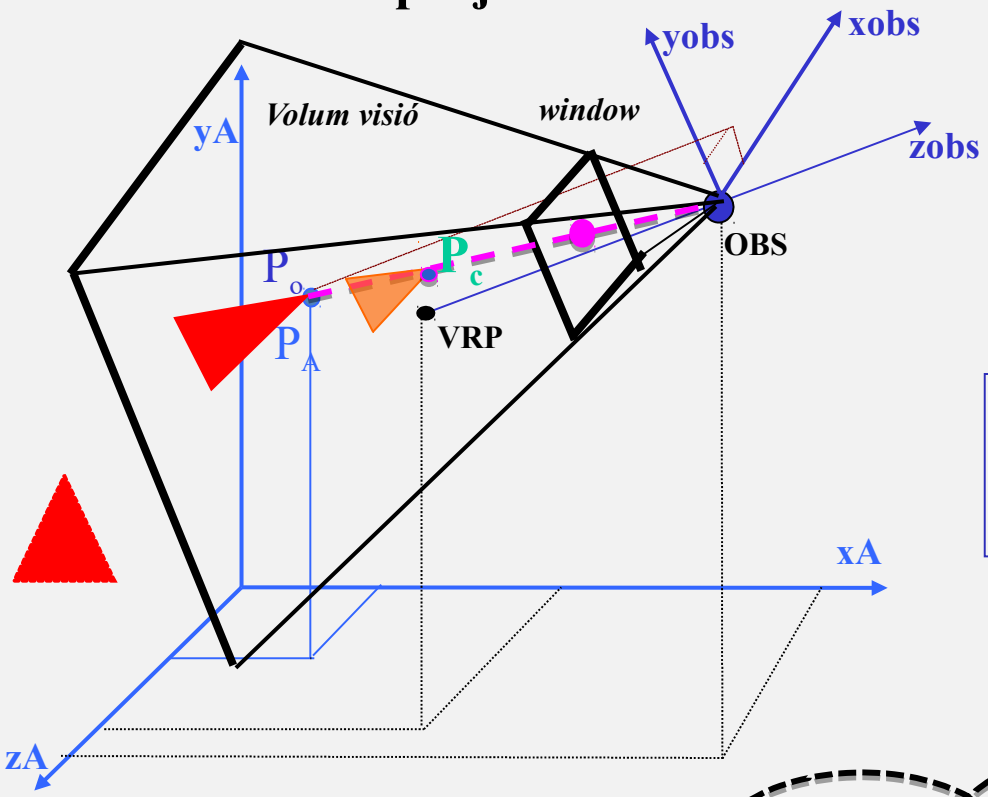
Nou exemple de base

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



Taula de continguts

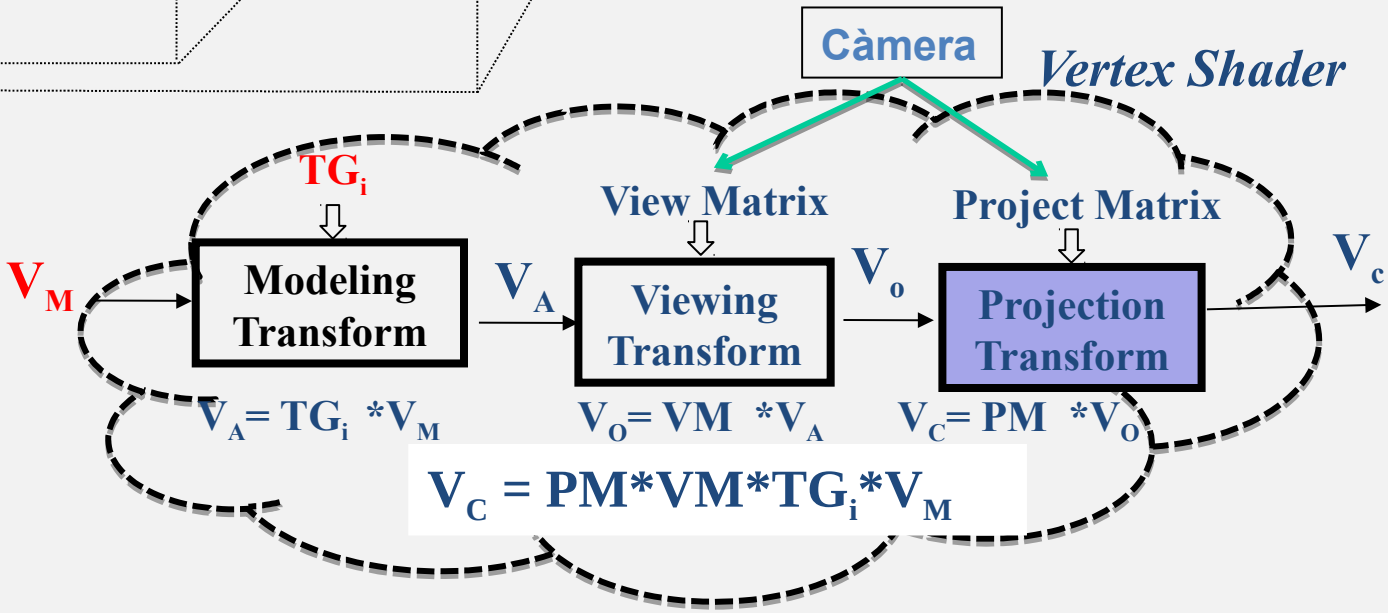
Transformació de projecció



FOV, ra_w, zNear, zFar

PM = perspective (FOV, ra_w, zN, zF)

projectMatrix(PM);



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)

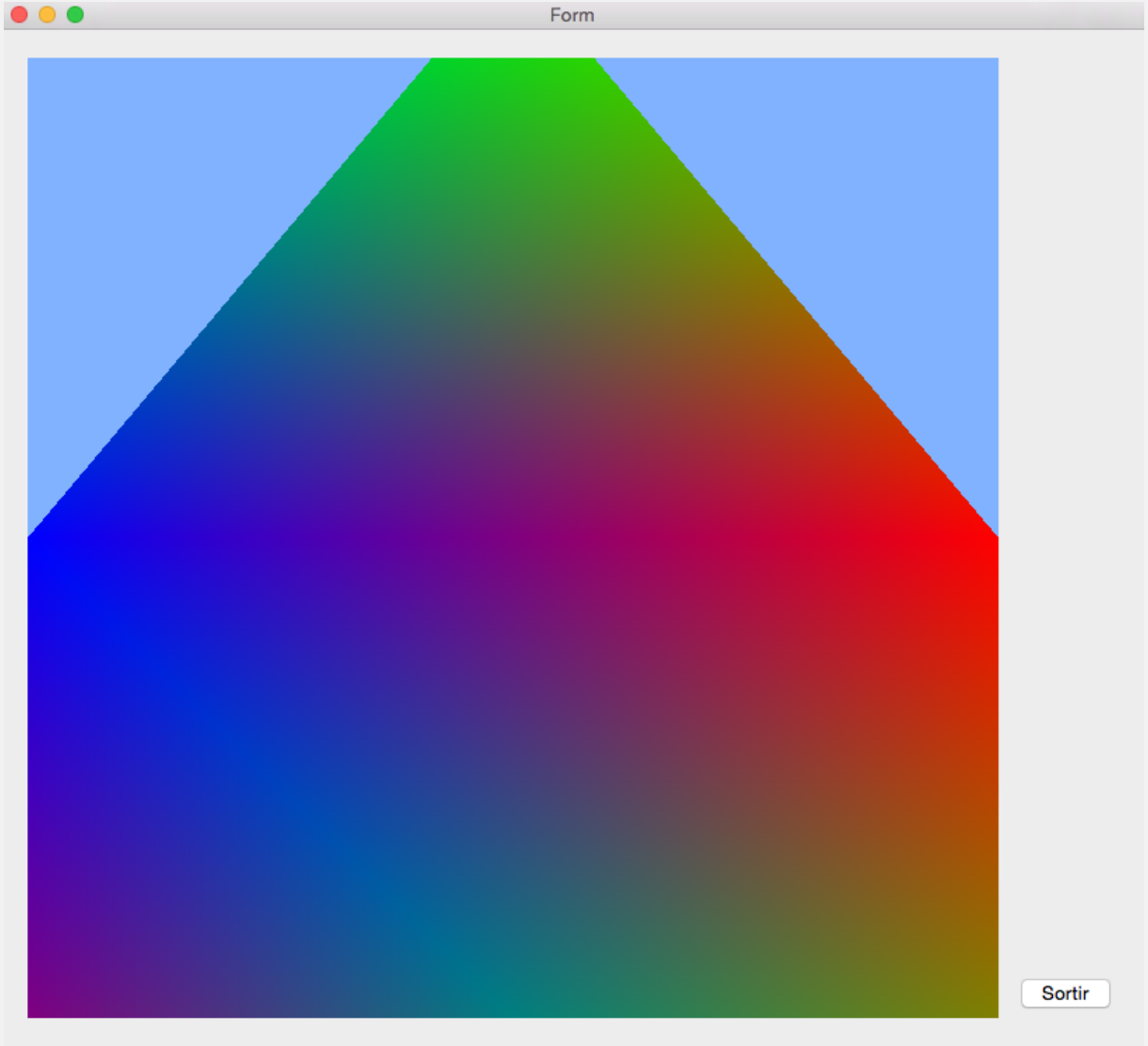
```
void MyGLWidget::projectTransform () {
    // glm::perspective (FOV en radians, ra window, znear, zfar)
    glm::mat4 Proj = glm::perspective ((float)M_PI/2.0f, 1.0f, 0.4f, 3.0f);
    glUniformMatrix4fv (projLoc, 1, GL_FALSE, &Proj[0][0]);
}
```

Transformació de projecció (exercici 1)

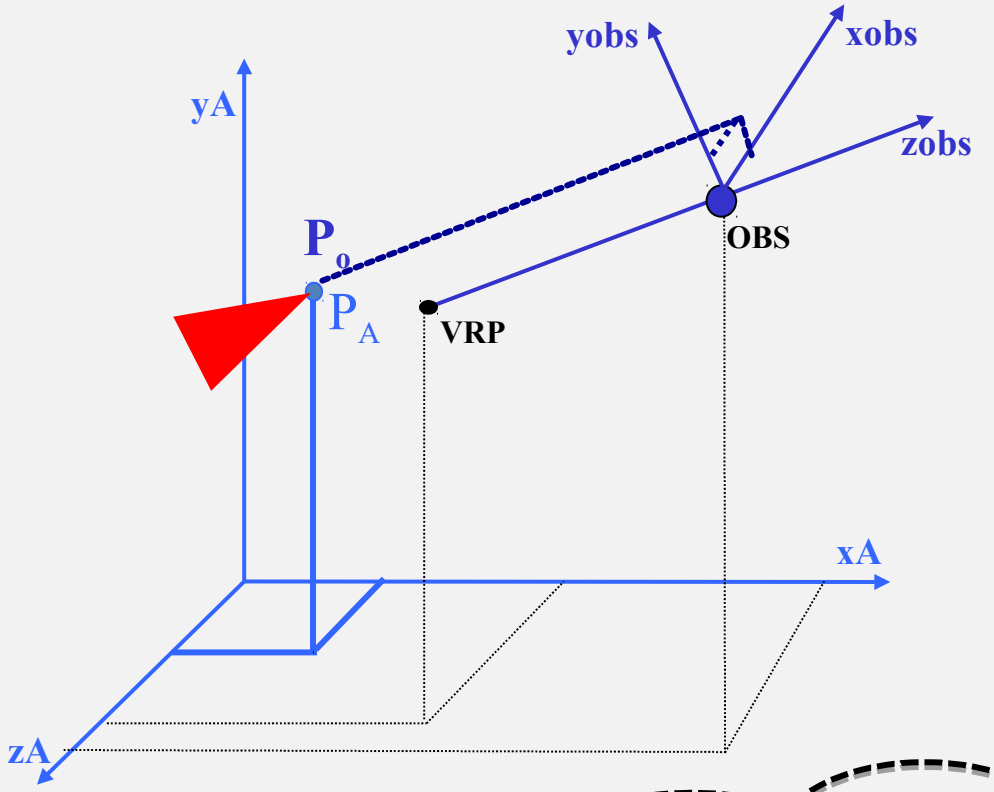
- Al vertex shader (afegir):

```
...
uniform mat4 proj;
...
void main () {
    ...
    gl_Position = proj * ... * vec4 (vertex, 1.0);
}
```

Transformació de projecció (exercici 1)

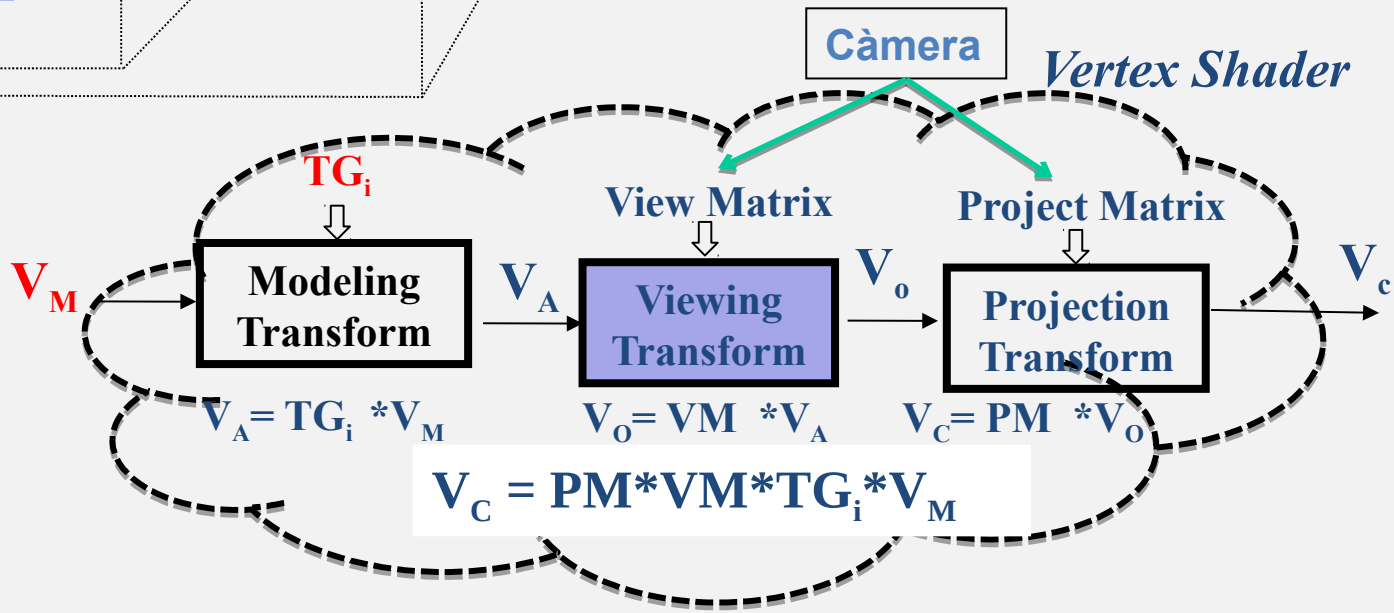


Transformació de punt de vista (view)



OBS , VRP, up

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



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]);
}
```

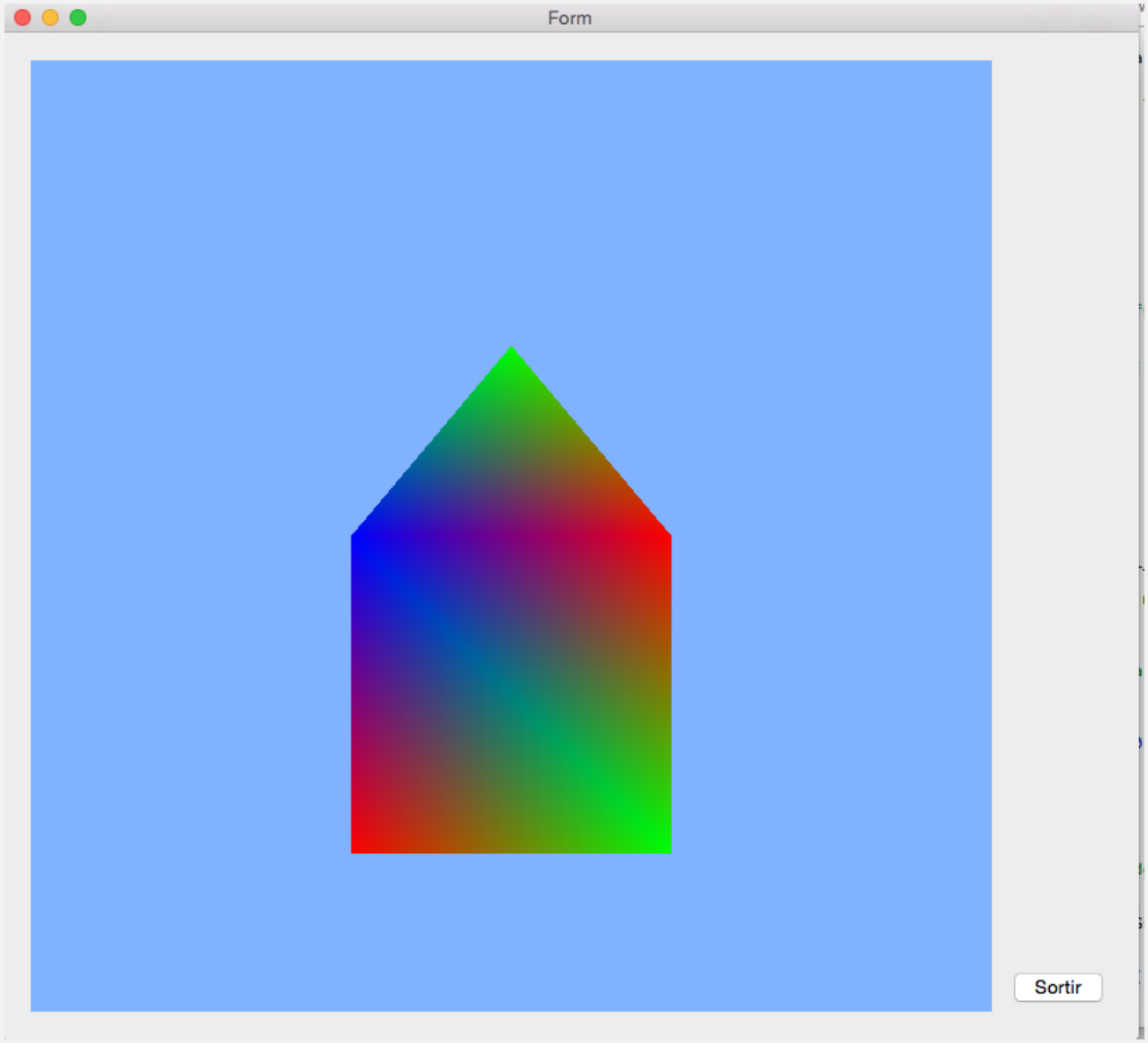
Transformació de punt de vista (view)
(exercici 2)

- 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)



Taula de continguts

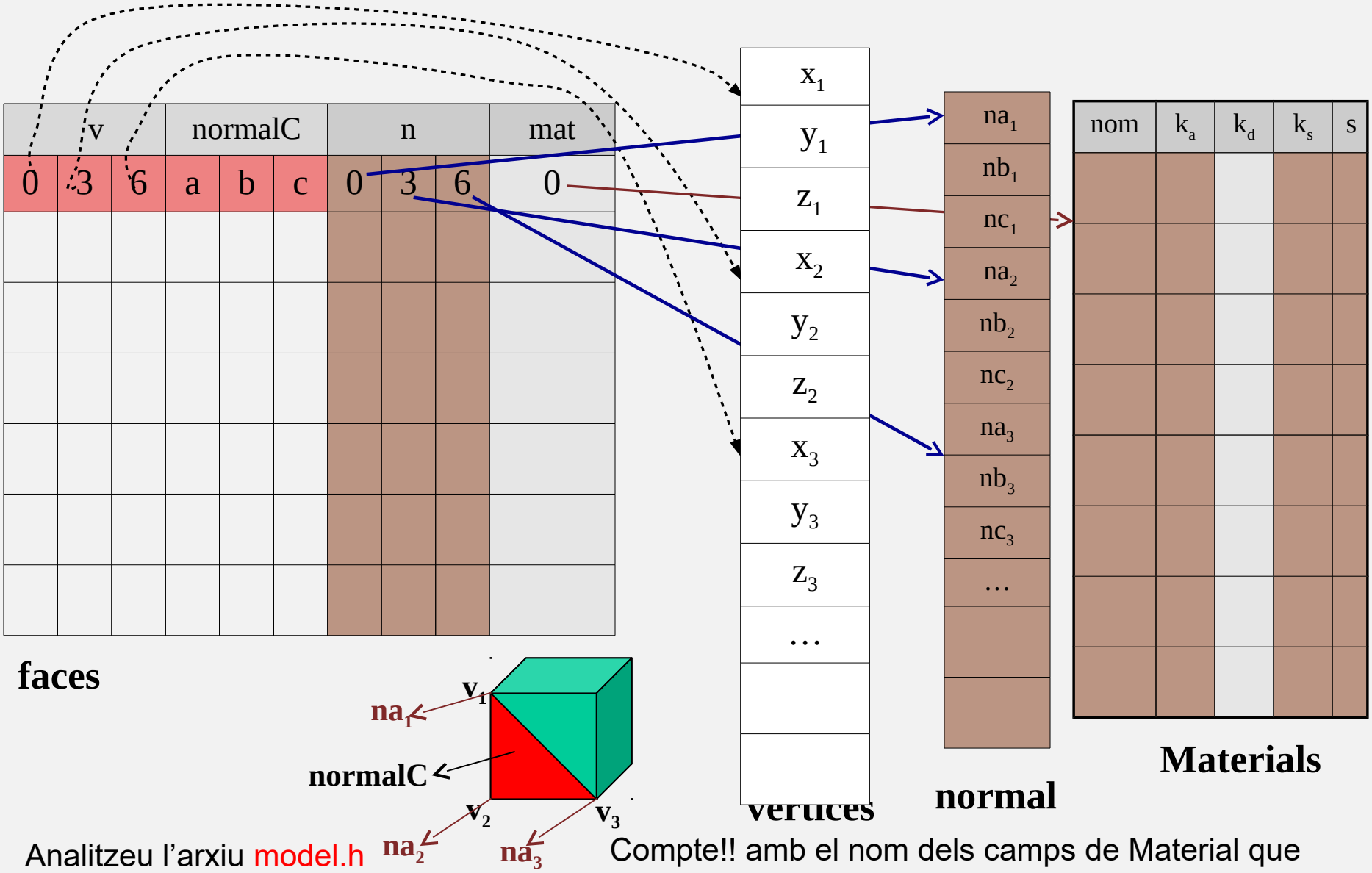
Càrrega de models OBJ (exercici 4)

- La Classe Model permet carregar un fitxer *.obj*.
  - */home/public/indi/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 */home/public/indi/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.

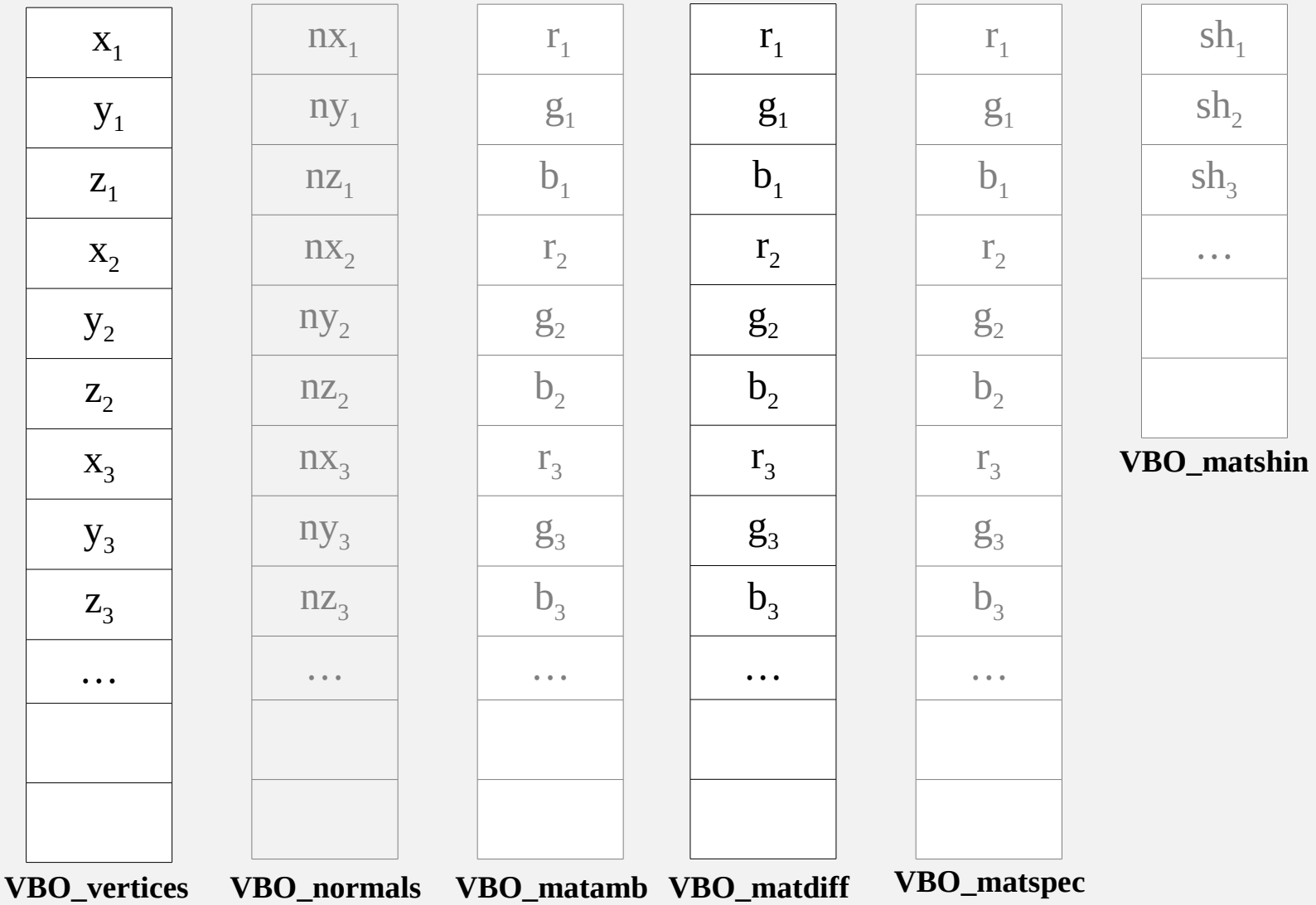
Representació classe Model



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



Ús de la classe Model (exercici 4)

- Cal fer include de la classe model en el fitxer MyGLWidget.h

```
#include “model.h”
```

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

```
Model m;  // un únic model
```

- Construcció de varis objectes tipus Model

```
Model vectorModels[3];  // array de 3 models
vector<Model> models;  // vector stl de models
```

Caldrà declarar la variable de tipus **Model** com a atribut de la classe, ja que serà necessari accedir a la informació del model des de diferents mètodes de la classe (per ex, des del paintGL).

Ús de la classe Model (exercici 4)

- Càrrega d'un arxiu .obj (model). La càrrega la farem generalment en el mètode *carregaBuffers*.

```
m.load ("../models/HomerProves.obj");
```

- La classe Model genera els arrays que utilitzarem per omplir els VBOs. Per accedir a aquests arrays cal fer:

```
m.VBO_vertices ()  // posició
m.VBO_matdiff ()    // color
```

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

```
m.faces().size()
```

- Per a saber el nombre de bytes que ocupa un array caldrà multiplicar: mida d'un float * nombre de cares * nombre de vèrtexs de cada cara * nombre de coordenades de cada vèrtex

```
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 (no és necessari per aquesta sessió)

```
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;
}
```

Taula de continguts

Z-buffer (exercici 4)

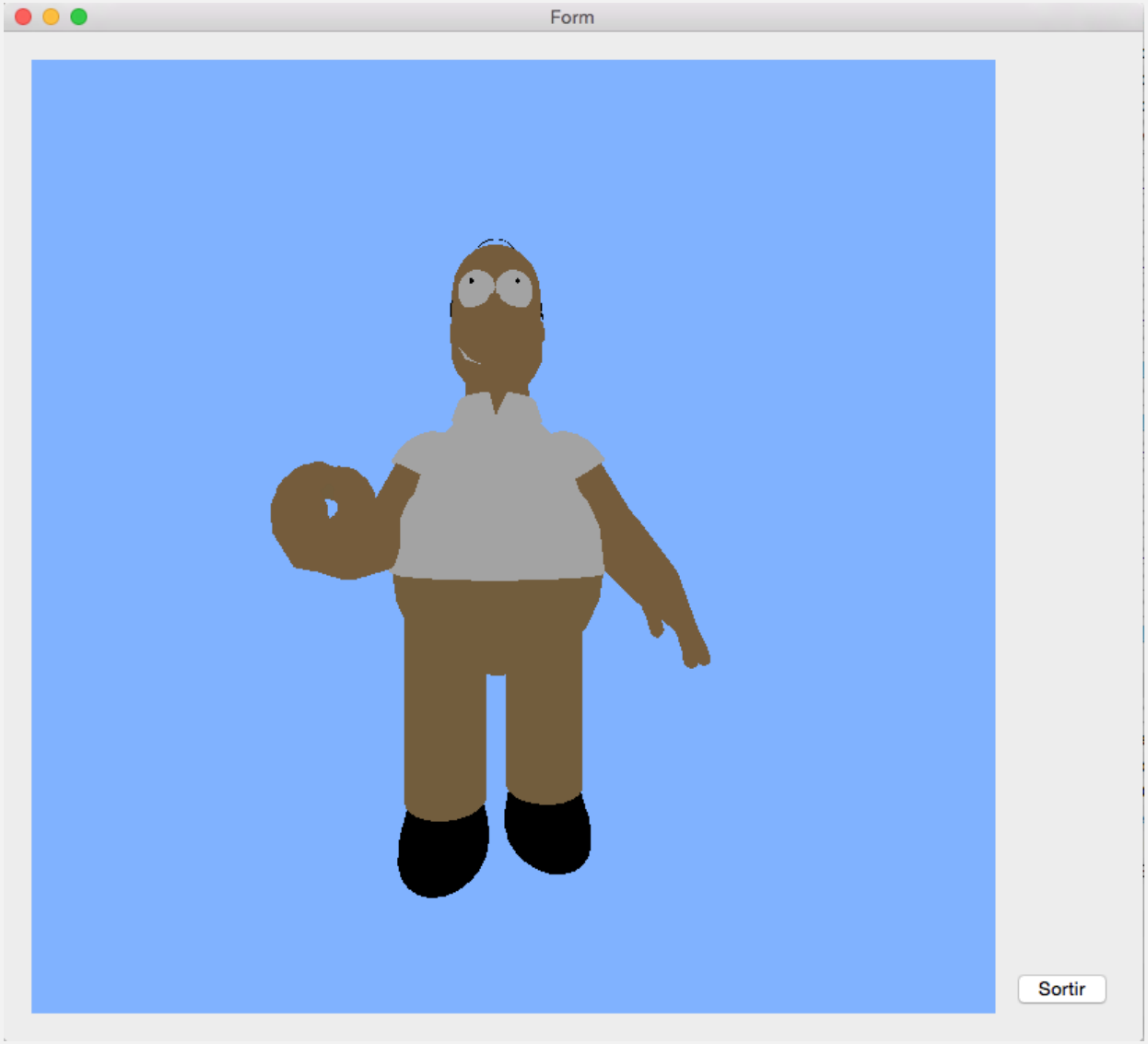
- Algorisme de Z-buffer:
  - Activar el z-buffer (només cal fer-ho un cop!). Ho farem a **initializeGL** després de cridar *initializeOpenGLFunctions*.

```
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)



Taula de continguts

Exercicis sessió 2.1

El que cal que feu en aquesta sessió és:

1) **Mirar codi exemple bloc 2** (el teniu disponible en el Campus digital) 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.