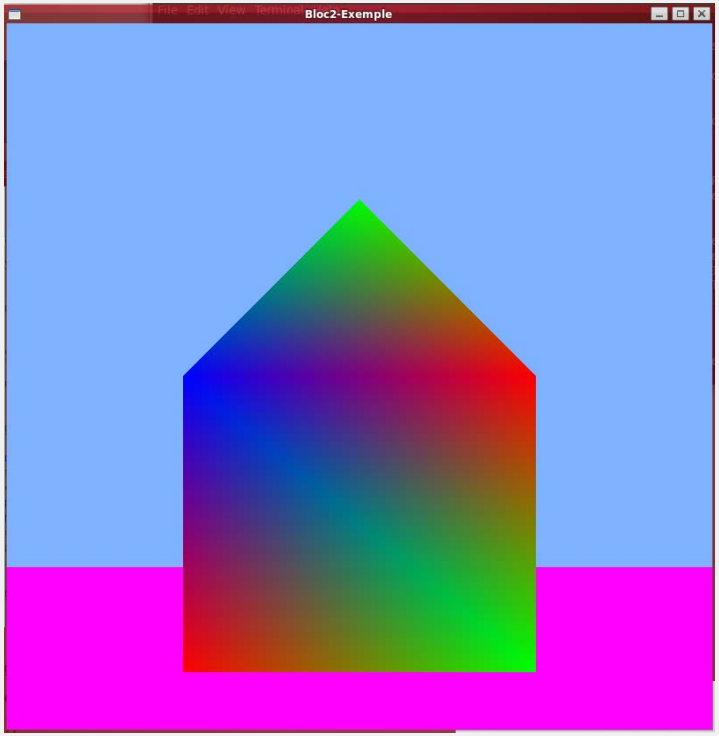
Laboratori OpenGL – Sessió 4
Bloc 2

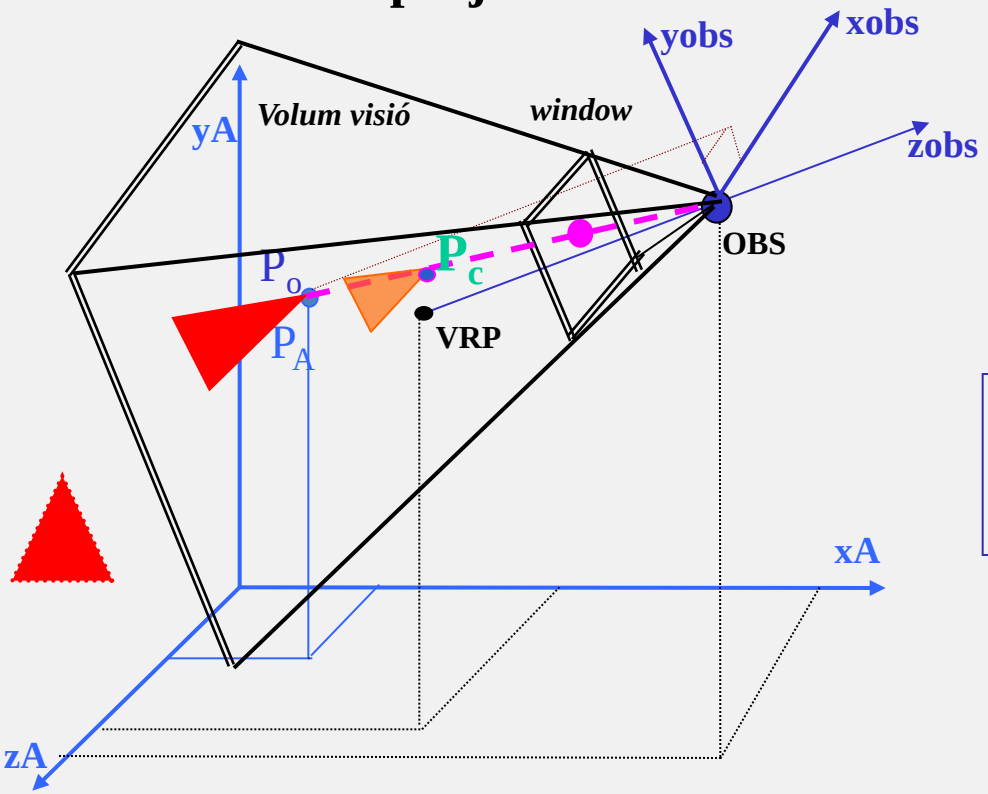
- Nou exemple de base
- Transformacions de càmera amb glm (view i projection)
- Z-buffer
- Classe Model – càrrega d'objectes OBJ

Nou exemple de base

- Pinta dos objectes
- Inclou transformació de model
- Vertex i Fragment Shaders pinten amb color per vèrtex



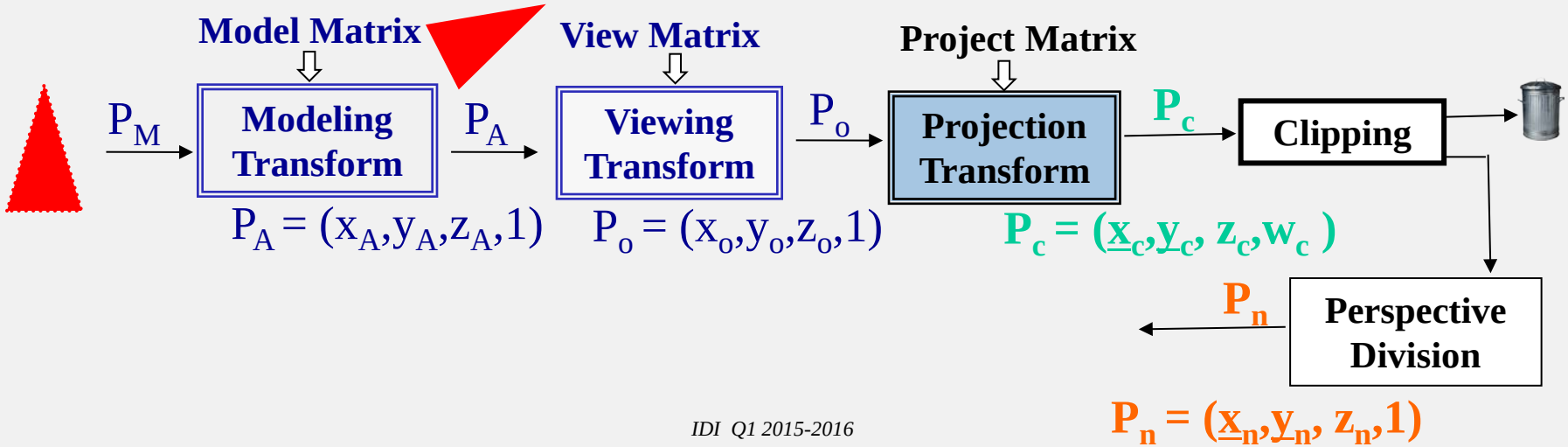
Transformació de projecció



FOV , zNear, zFar, ra_w

TP= Perspective (FOV, ra, zN, zF)

projectMatrix(TP);



Transformació de projecció

- Al codi cpp de QGLWidget:
  - 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

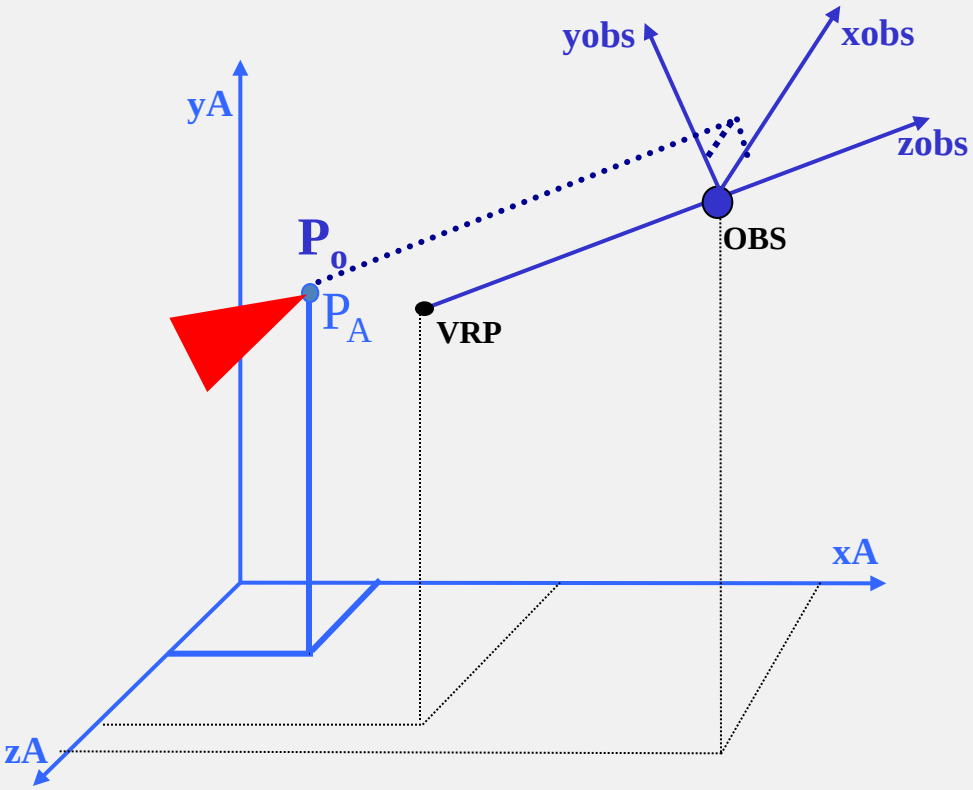
```
void MyGLWidget::projectTransform () {
    // glm::perspective (FOV en radians, ra window, znear, zfar)
    glm::mat4 Proj = glm::perspective (M_PI/2.0, 1.0, 1.0, 3.0);
    glUniformMatrix4fv (projLoc, 1, GL_FALSE, &Proj[0][0]);
}
```

Transformació de projecció

- Al vertex shader (afegir):

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

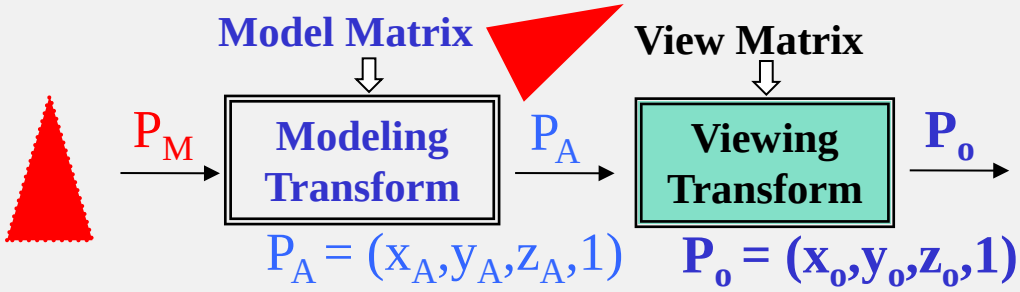
Transformació de punt de vista (view)



OBS , VRP, up

VM = LookAt (OBS,VRP,up);

viewMatrix (VM);



Transformació de punt de vista (view)

- Al codi cpp de QGLWidget:
  - 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(0,0,2),
                                  glm::vec3(0,0,0), glm::vec3(0,1,0));
    glUniformMatrix4fv (viewLoc, 1, GL_FALSE, &View[0][0]);
}
```

Transformació de punt de vista (view)

- Al vertex shader (afegir):

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

Z-buffer

- 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);
    ```

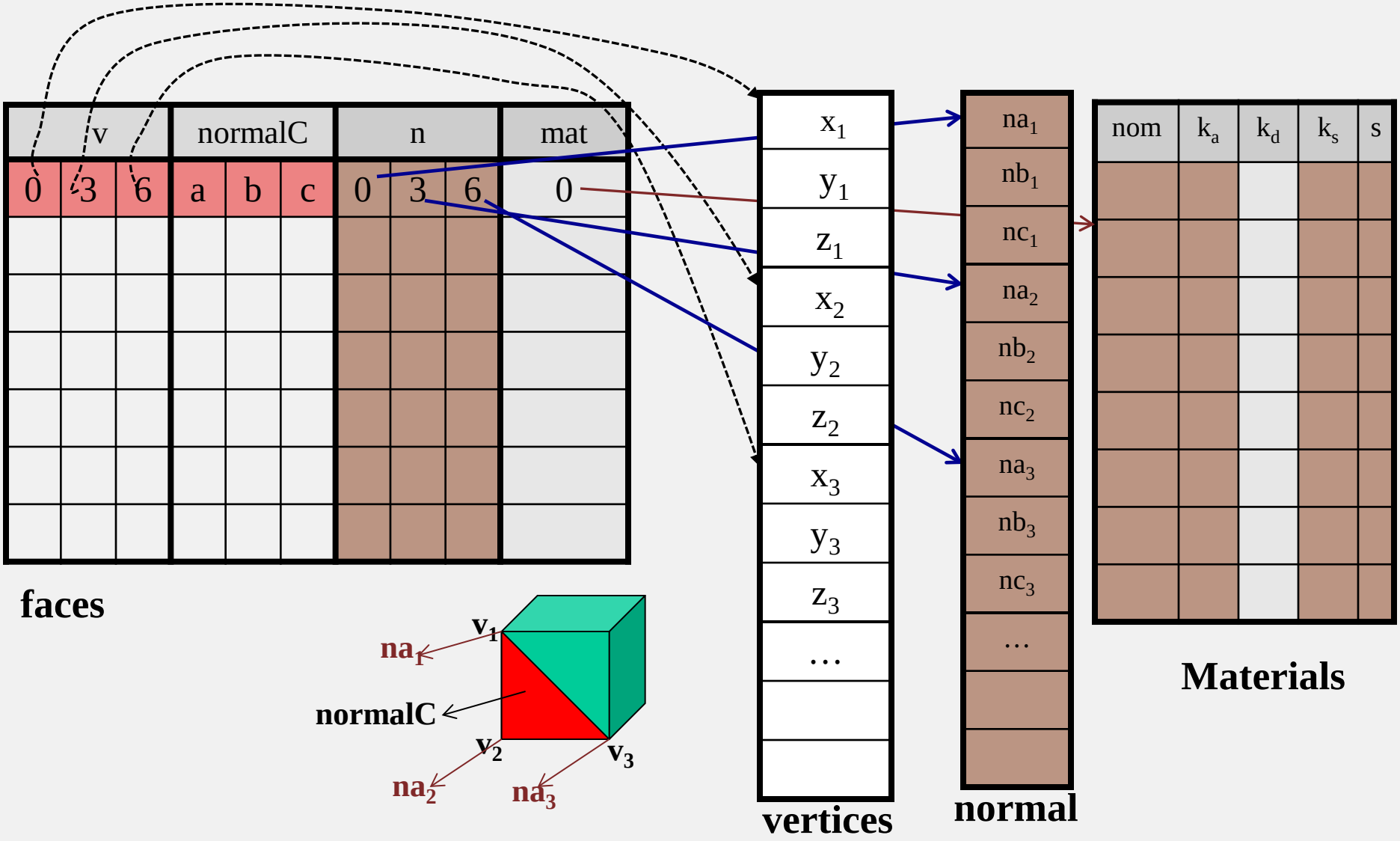
Càrrega de models OBJ

- 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.
- Més models els podeu trobar a la xarxa

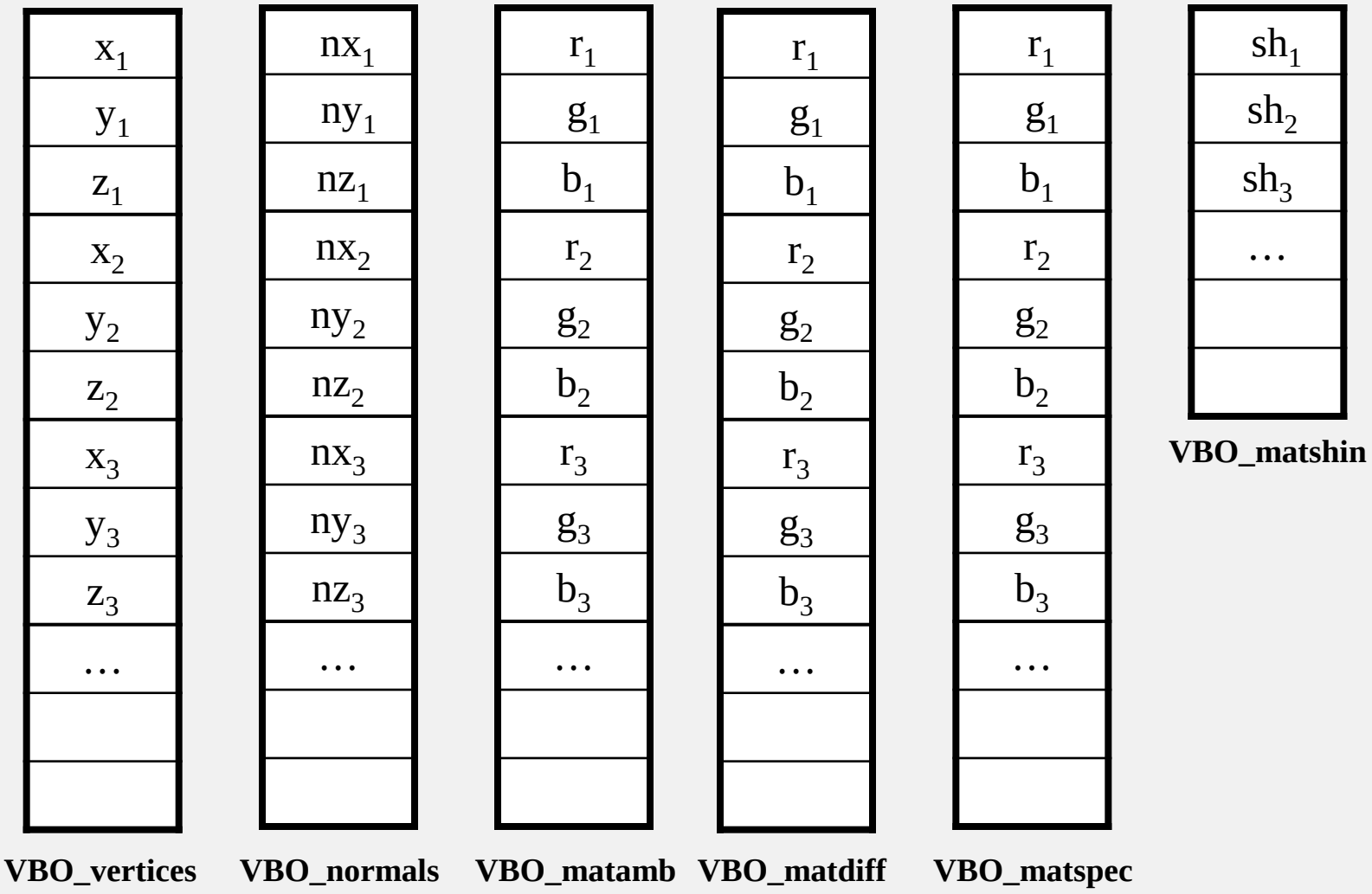
Representació classe Model



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 classe Model



Ús de la classe Model

- 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;
}
```

Exercicis sessió 4

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 els exercicis que teniu al guió per a aquesta sessió. És important que els feu **tots i 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.