Cours d'infographie

3ém année SI

Mr. Ghemougui Abdessatar

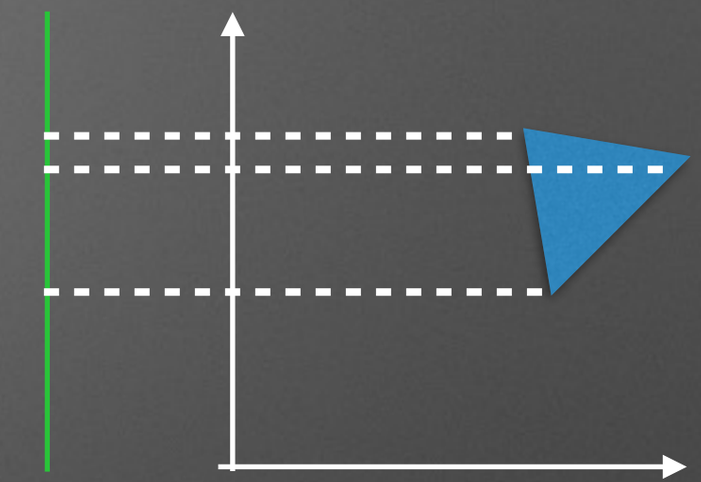
2016

Transformation de vue

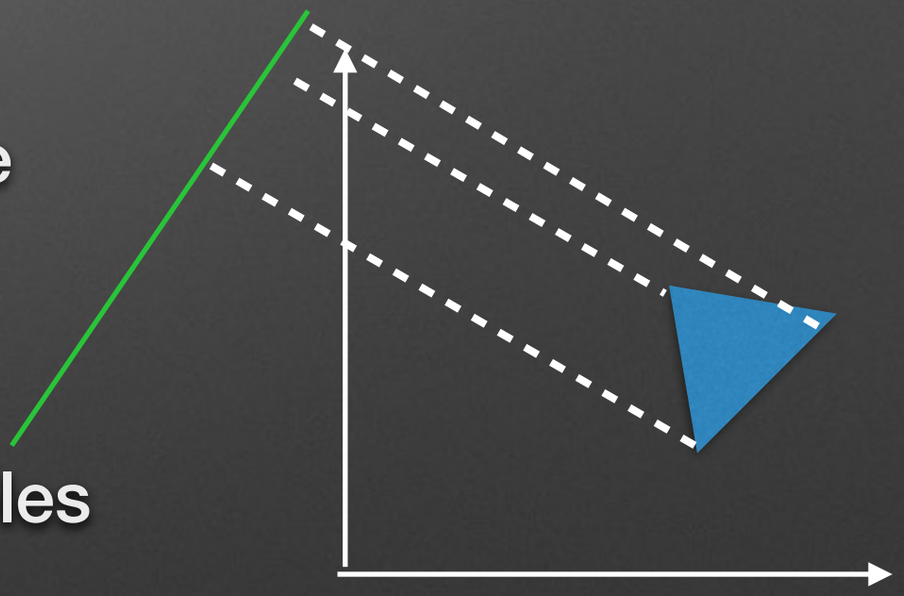
Projection

Projection parallelle

les points sont déplacés par des lignes parallèle dans la direction de projection jusqu'à le plan image



Si le plan image est perpendiculaire à la direction de projection donc on parle d’une projection **orthographique**, sinon on parle d’une projection **oblique.**
Ce type de projection :



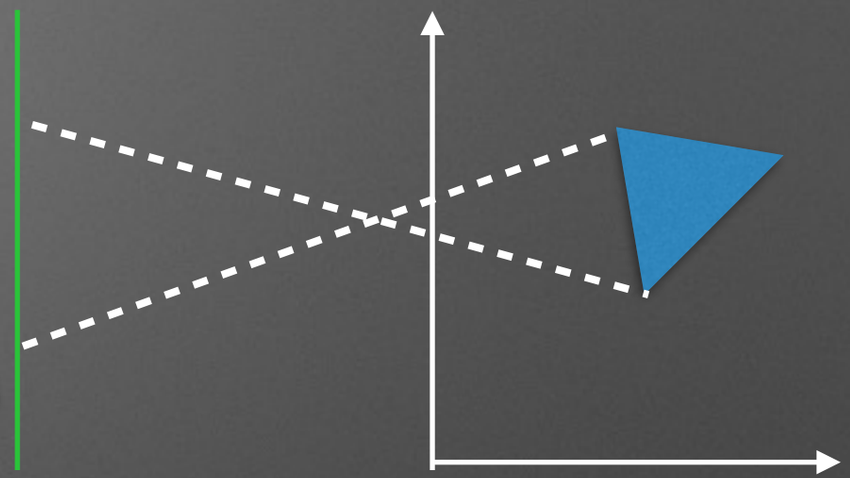
- Si deux lignes sont parallèles il restent parallèles après la projection.
- Ce type de projection préserve la taille des objets plantaires.

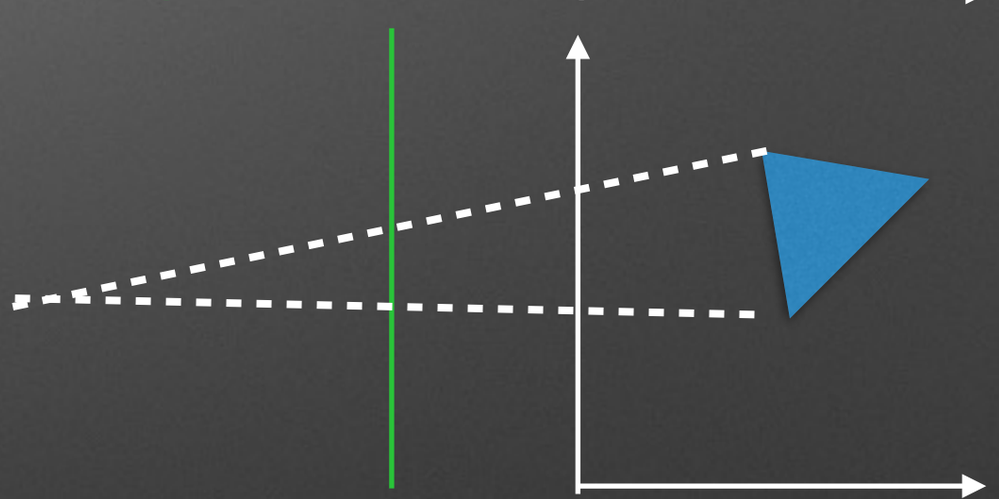
Projection

Projection perspective

les points sont déplacés par des lignes qui passent par un seul point dans la direction de projection jusqu'à le plan image

Les objets lointains de l’observateur devient plus petits.



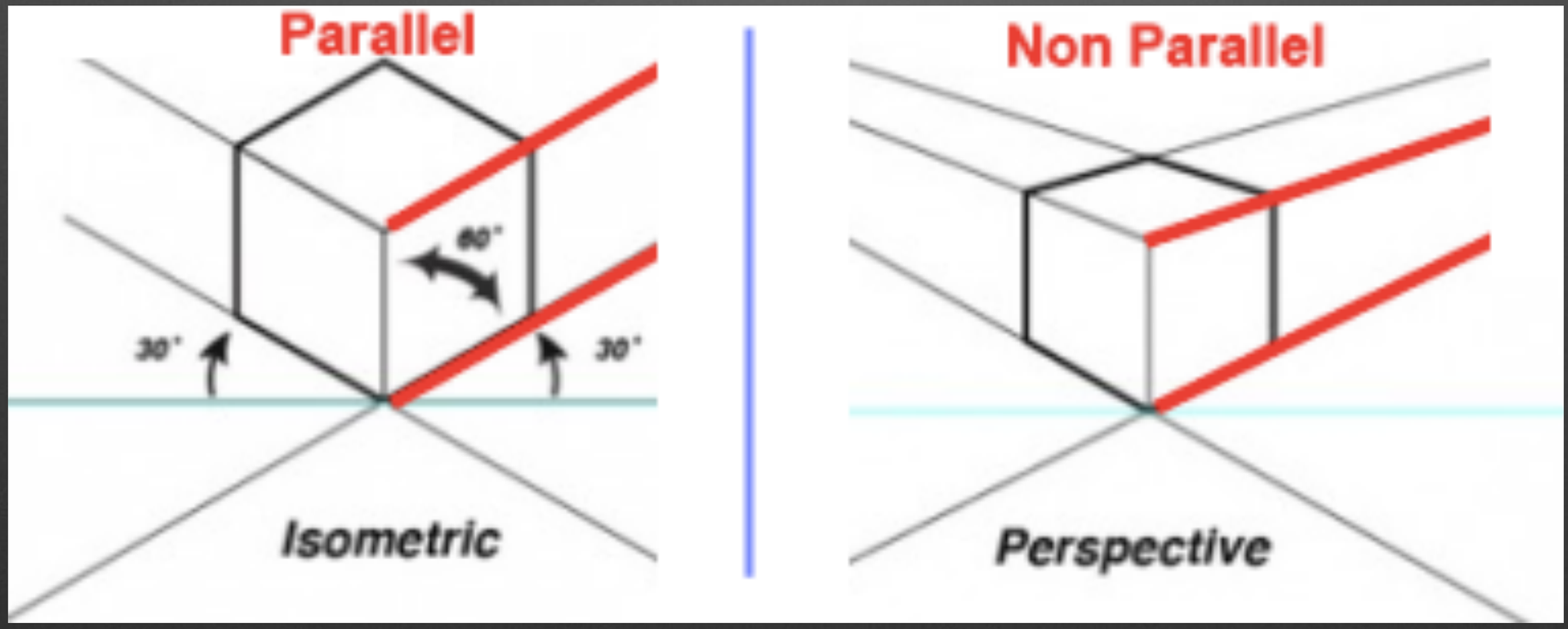


Projection

perspective vs. parallèle

Projection perspective est plus naturelle et réaliste

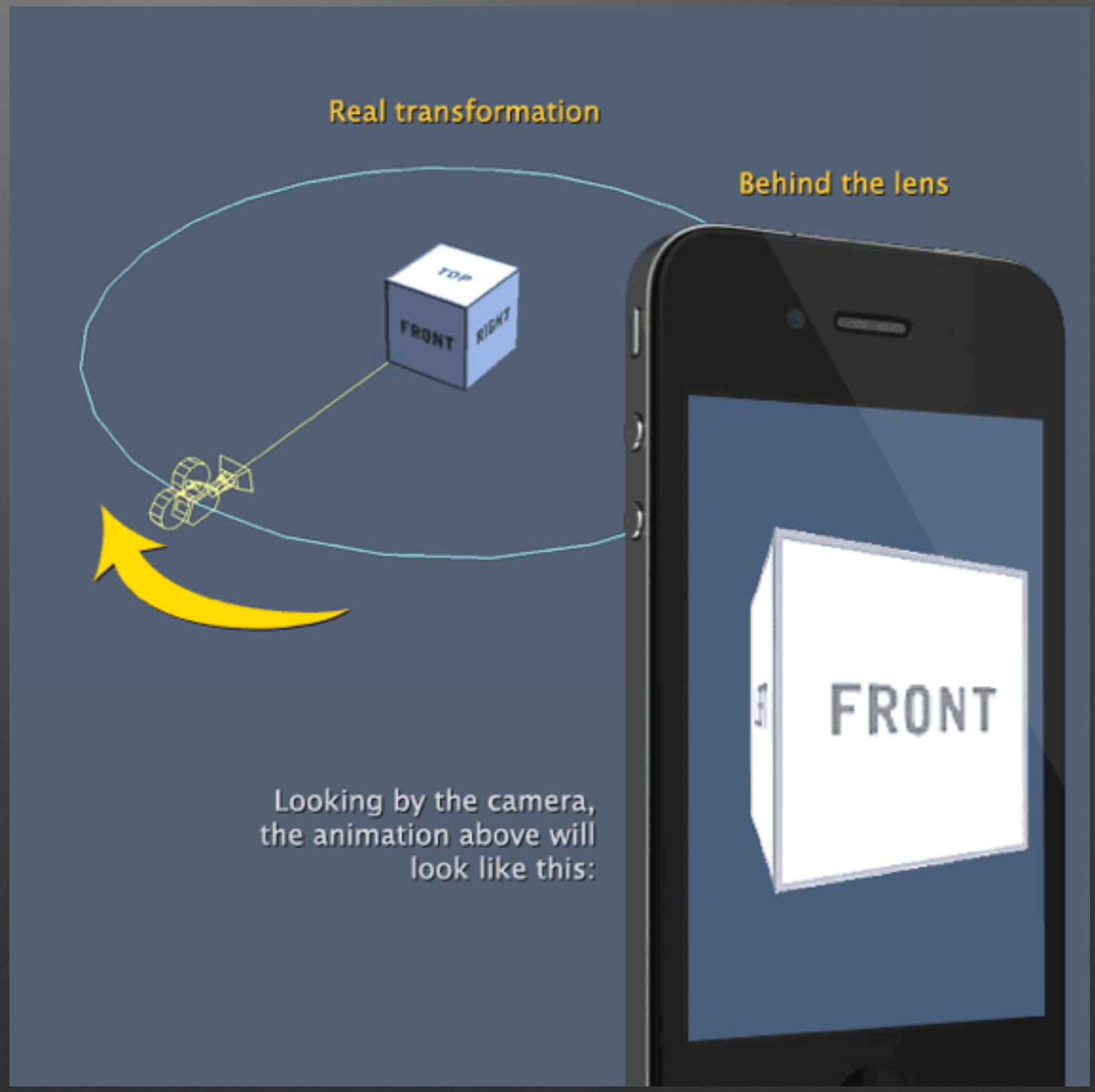
Projection parallèle généralement utilisée dans les application de la CAO et dans les dessin en 2D dans les platformes 3D.



Transformation de vue

L'objectif est de générer une image (2D) à partir d'une scène (3D),

Nous allons utiliser une caméra virtuelle



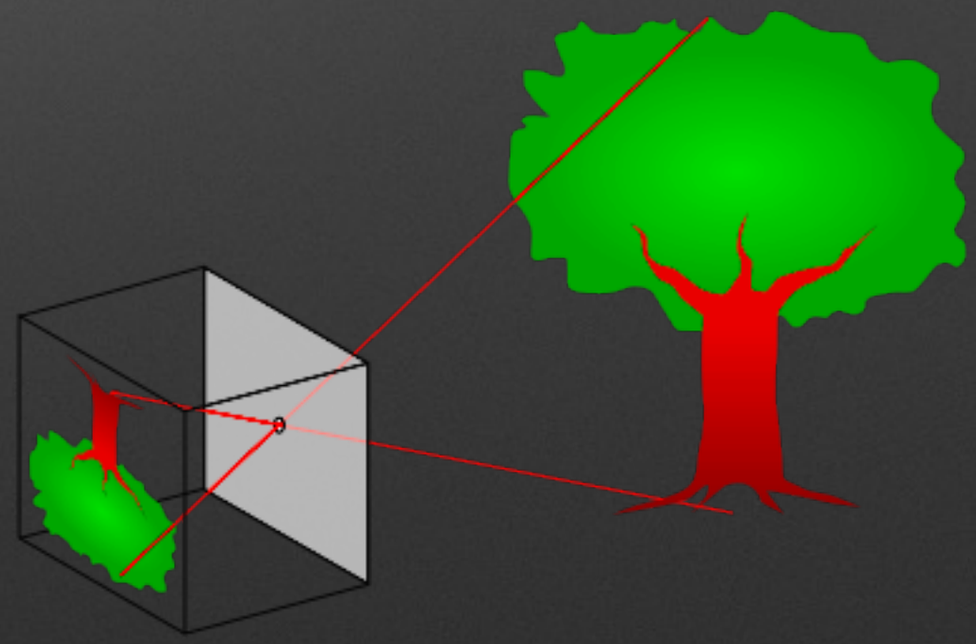
Modèle de caméra

Le modèle de caméra que nous allons utilisé est appelé pinhole camera (trou d'épingle)

Le caméra effectue une projection perspective

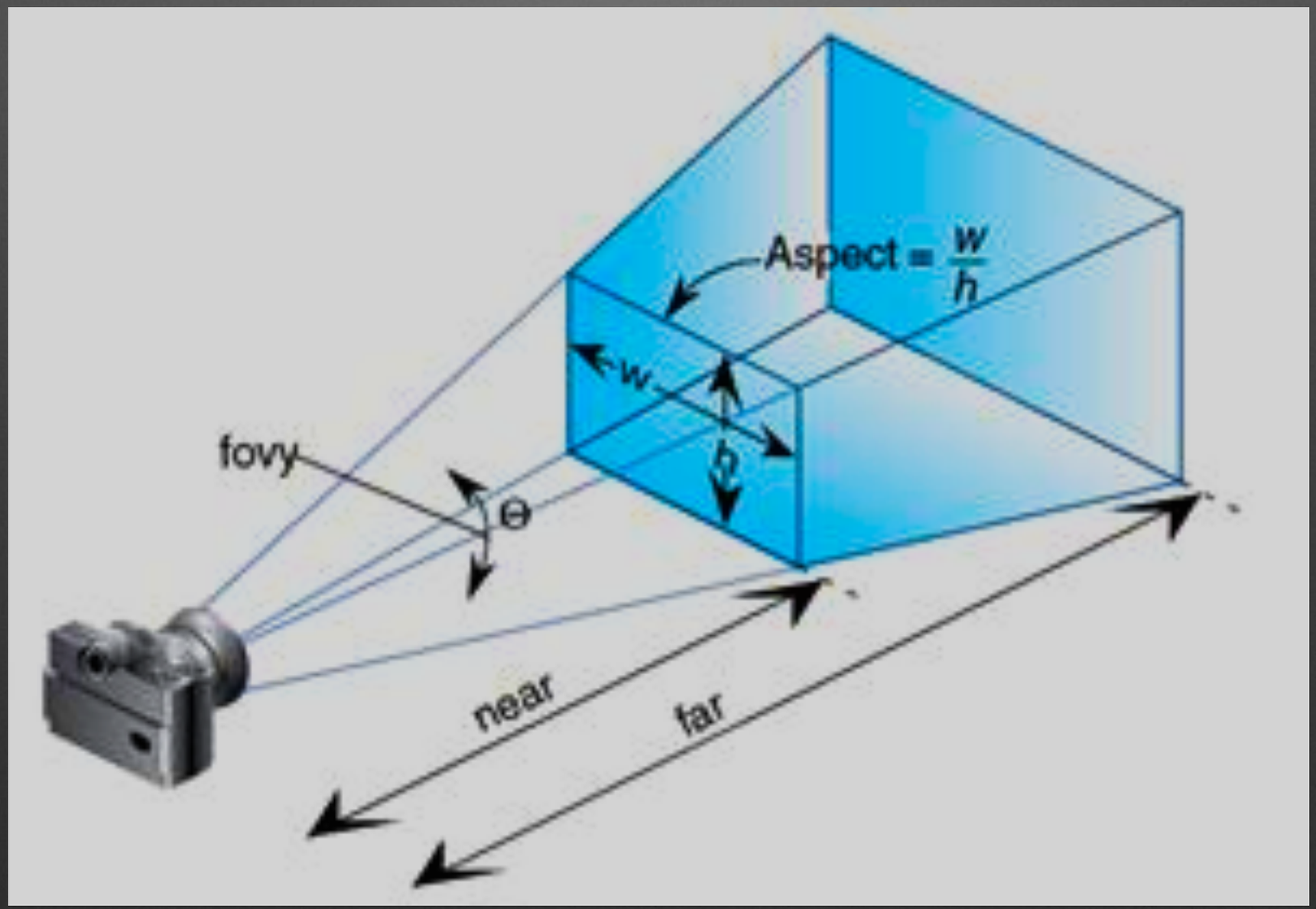
Le centre de projection est le pinhole appelé focal point.

L'image projetée est inversée (rotation avec 180°)

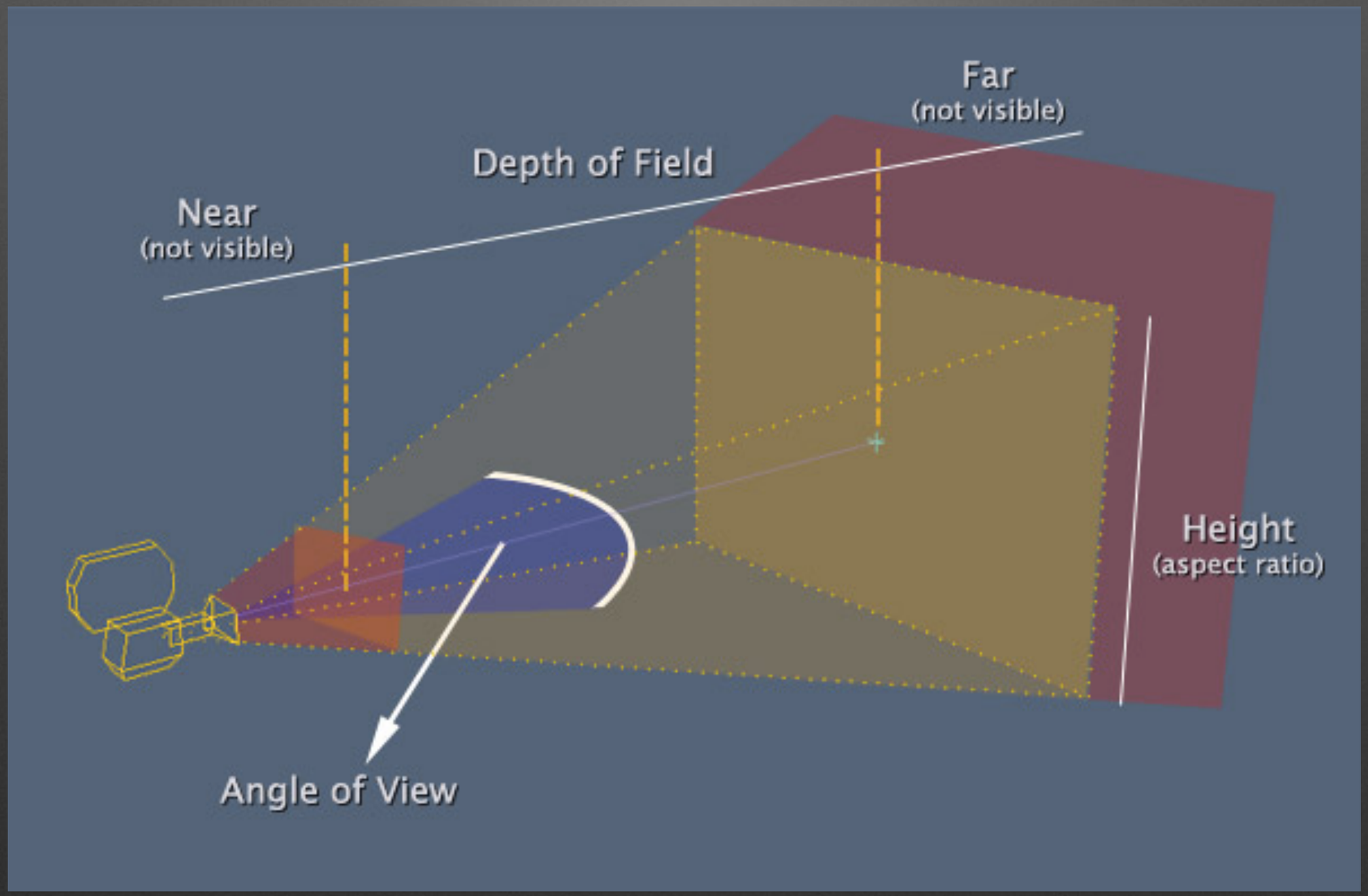


Modèle de caméra

On synthèse d'image le modèle de caméra est une version modifiée de celle de pinhole.



Modèle de caméra

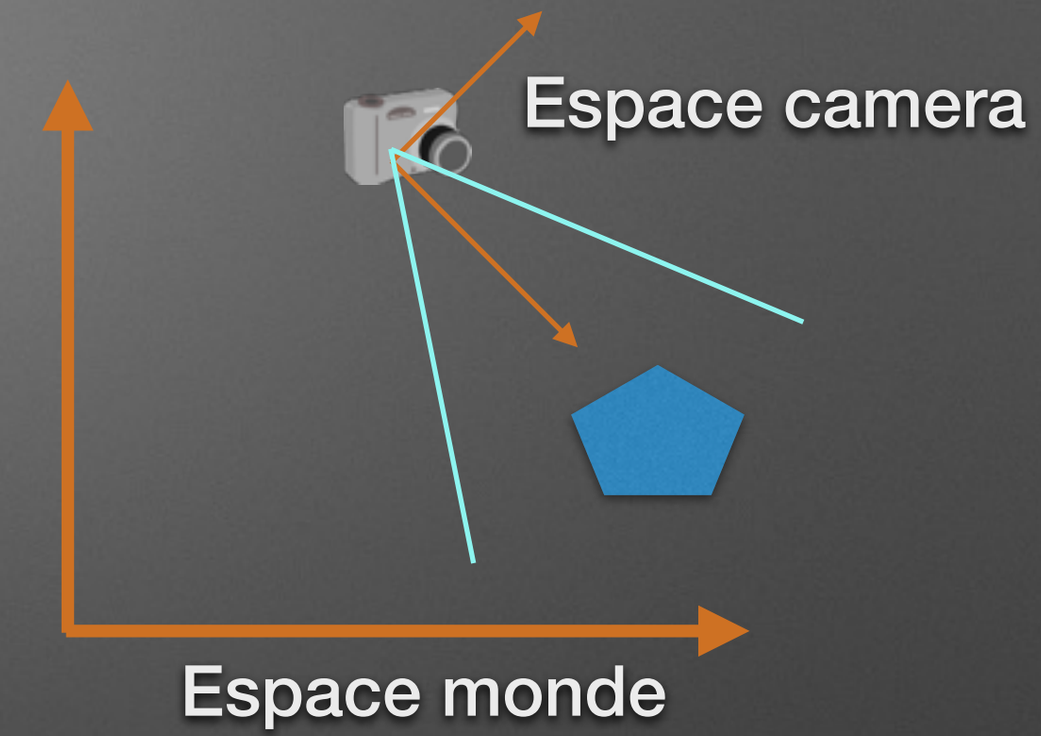


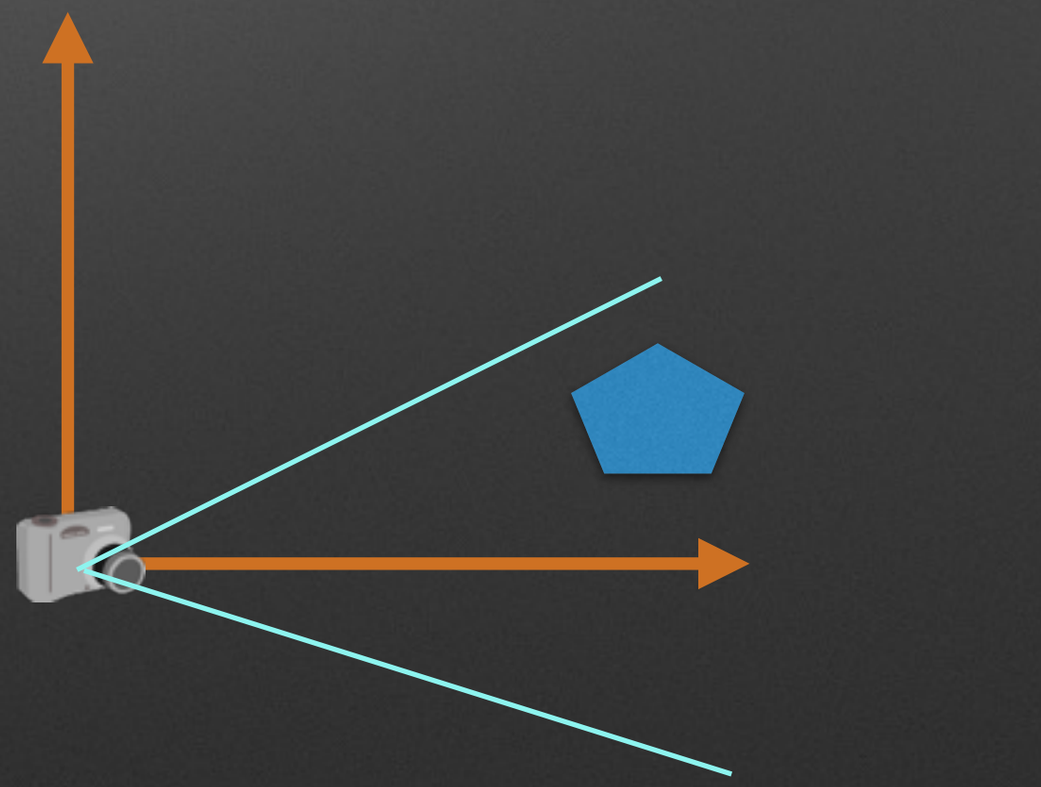
Modèle de caméra

Il est plus facile de faire la projection si la caméra est à l'origine

cela signifie de transformer tout les objets dans l'espace de caméra (view space)

C'est une opération de changement de système de coordonnées effectué par une simple multiplication de matrice

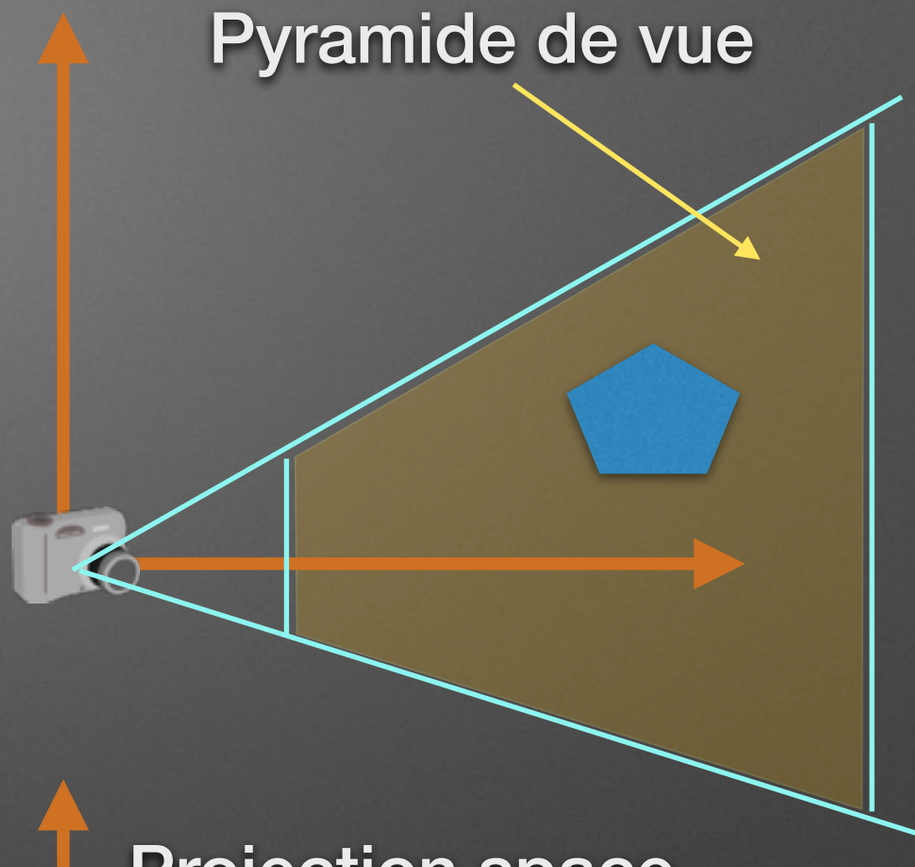


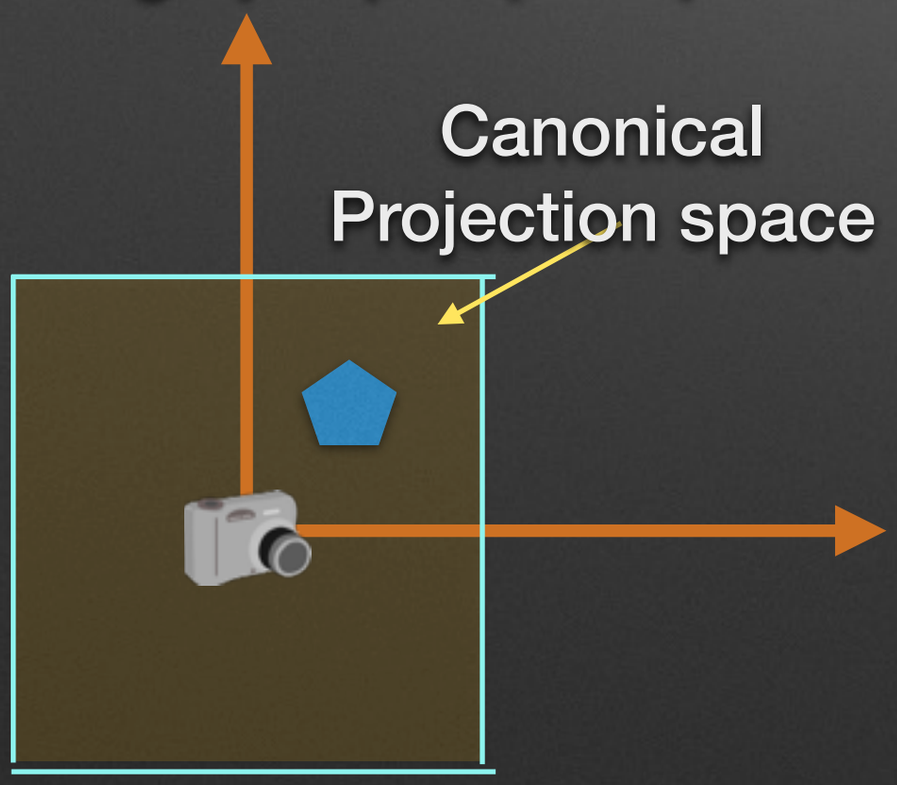


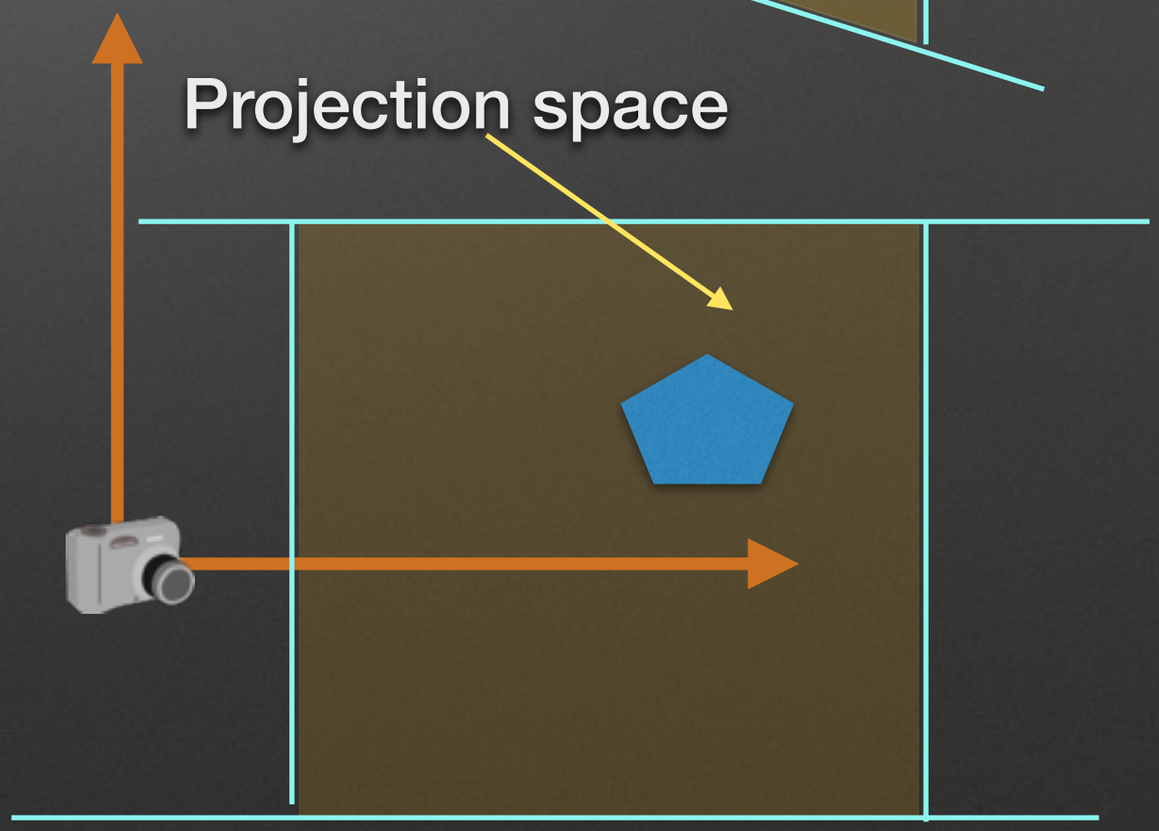
Modèle de caméra

Il est plus simple d'effectuer une projection parallèle à la place d'une projection perspective,

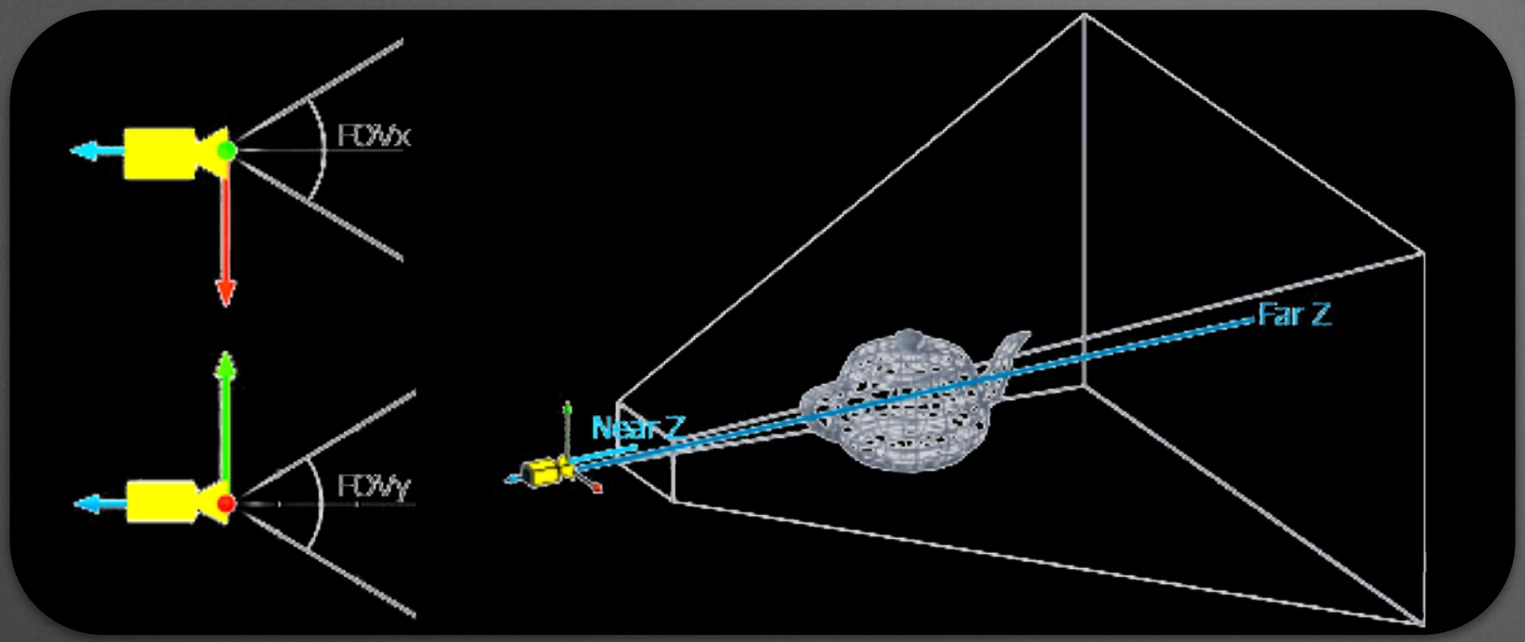
Cela peut être accomplit en transformant le pyramide de vue en un volume de vue orthographique. (cube)

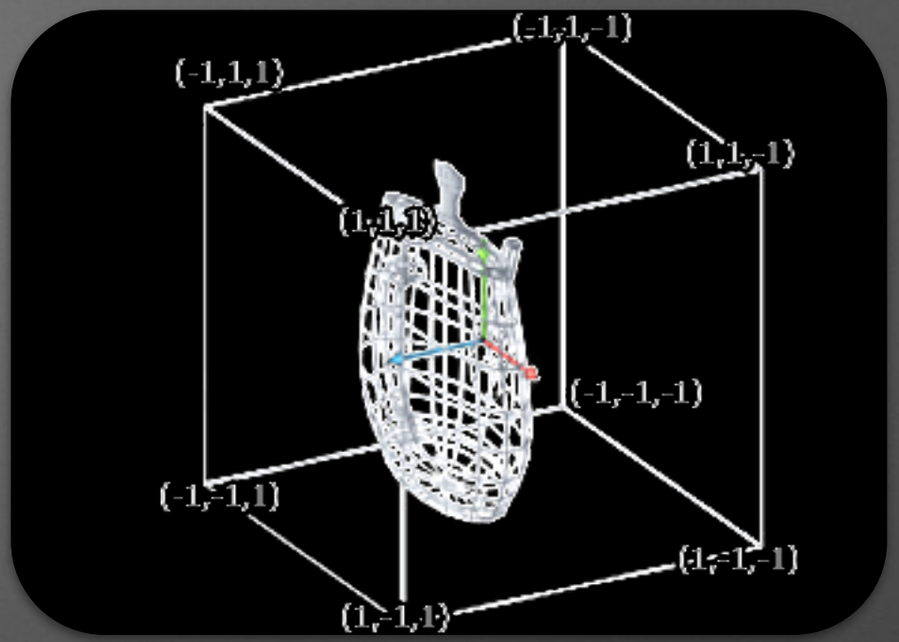






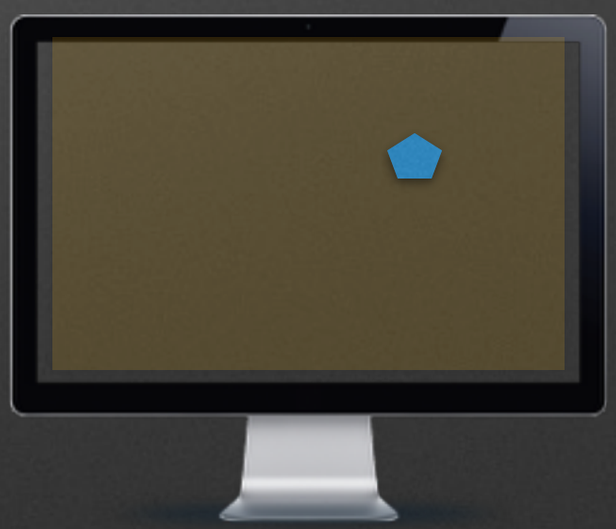
Modèle de caméra





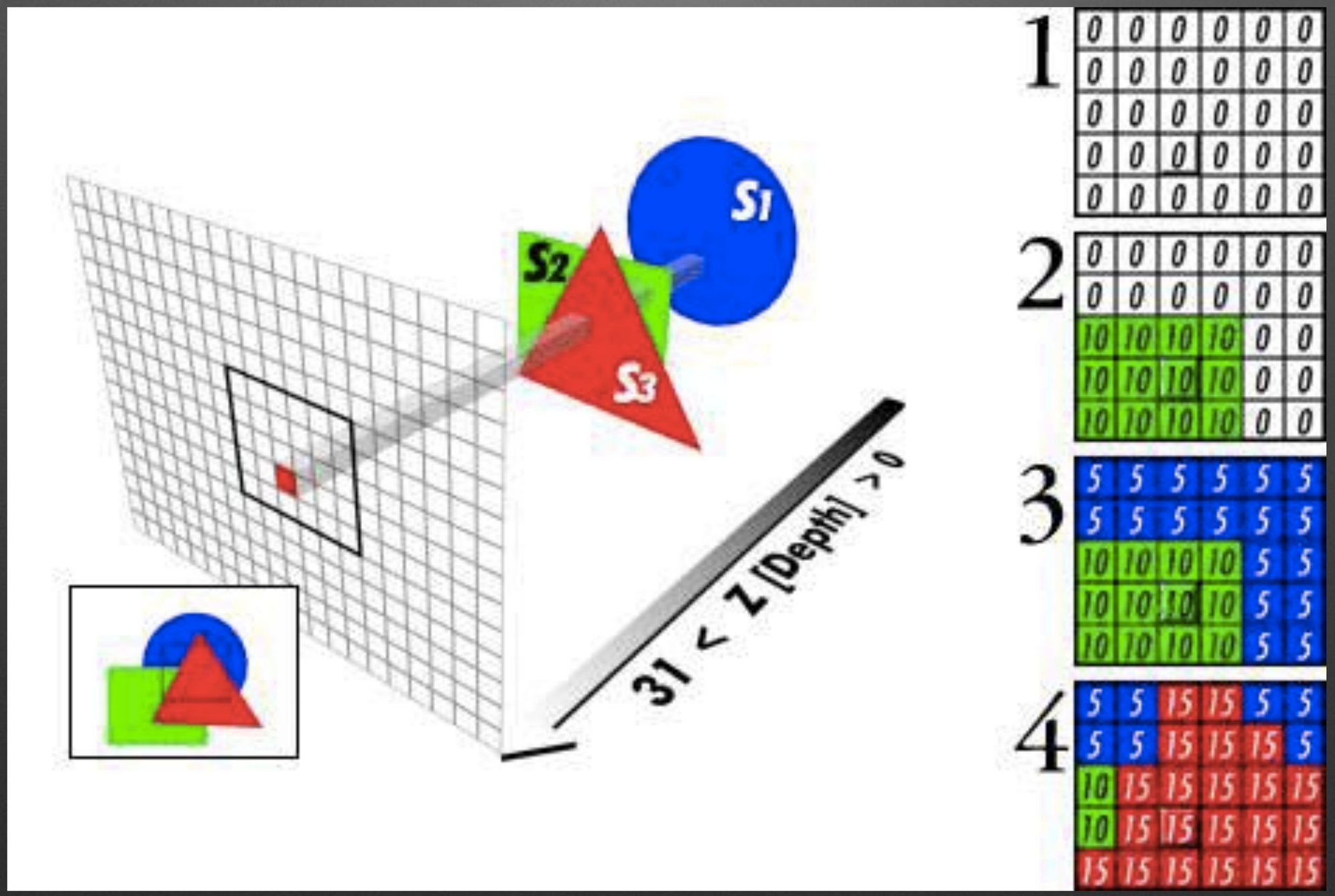
Espace caméra

Espace projection



Espace image

Modèle de caméra



API Graphique

Introduction à openGL

Introduction à openGL

OpenGL (Open Graphics Library)

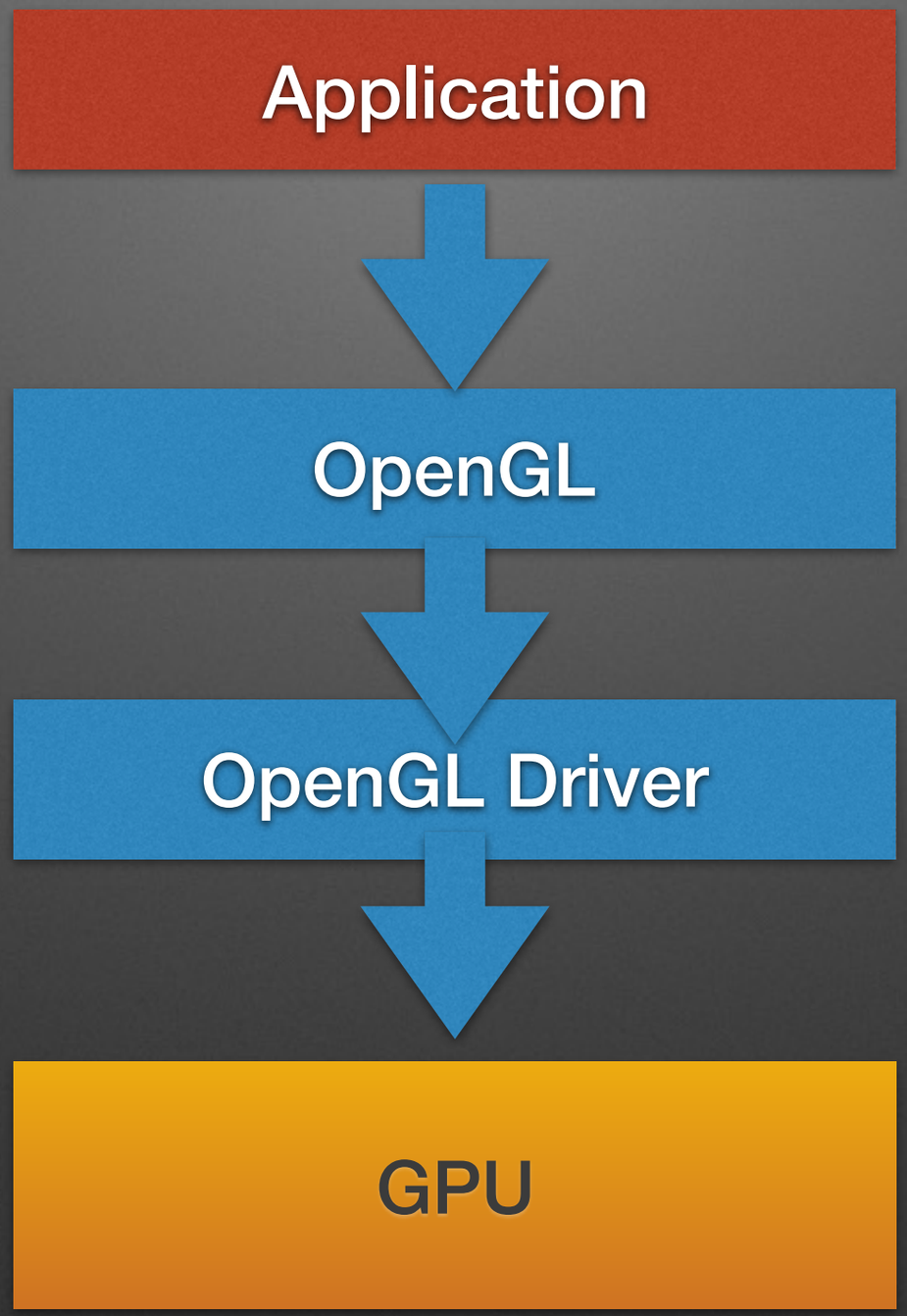
Un API pour le développement d'application graphique

Multi-platformes

En langage C

lancé par Silicon Graphics en 1992

Introduction à openGL



Introduction à openGL

OpenGL est une interface comprenant plus de 120 commandes, qui vont servir à décrire des objets (caméras, lampes, modèles 3D) et les opérations que l'on peut effectuer pour les manipuler

Elle s'accompagne de librairies complémentaires (GLU, GLUT, GLX, Open Inventor...)

Introduction à openGL

L'OpenGL Utility Library (GLU), surcouche à OpenGL, fournit des fonctions plus évoluées pour la gestion des caméras ou la création de NURBS. Fait partie d'OpenGL.

L'OpenGL Extension Library pour le système graphique X11 (GLX) : fournit les moyens de créer un contexte OpenGL et l'associer avec un drawable X11

Le GL Utility Toolkit (GLUT). Ecrite à l'origine pour simplifier les exemples de l'OpenGL Programming Guide (sous le nom de libaux). Fournit une interface simple, portable et indépendante de l'OS pour la GUI.

OpenInventor : librairie orientée objet de haut niveau, proche de VRML.

Introduction à openGL

```
#include <GL/gl.h>
#include <GL/glut.h>
main(int argc, char **argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (0, 0);
   glutCreateWindow (“Nom de fenetère“);
   // register callbacks
   glutDisplayFunc(fonction_affichage);
   glutKeyboardFunc(fonction_clavier);

   // enter GLUT event processing cycle
   glutMainLoop();
}
```

Introduction à openGL

```
#include <GL/gl.h>
#include <GL/glut.h>
main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (0, 0);
    glutCreateWindow (“Nom de fenetère“);
    // register callbacks
    glutDisplayFunc(fonction_affichage);
    glutKeyboardFunc(fonction_clavier);

    // enter GLUT event processing cycle
    glutMainLoop();
}
```

GLUT_DOUBLE	GLUT_RGBA
GLUT_SINGLE	GLUT_RGB
	GLUT_INDEX
GLUT_ACCUM	
GLUT_STENCIL	GLUT_DEPTH

Introduction à openGL

```
#include <GL/gl.h>
#include <GL/glut.h>
main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (0, 0);
    glutCreateWindow (“Nom de fenetère“);
    // register callbacks
    glutDisplayFunc(fonction_affichage);

    glutKeyboardFunc(fonction_clavier);
    // enter GLUT event processing cycle
    glutMainLoop();
}
```

Position, taille et nom de la fenêtre d’affichage

Introduction à openGL

```
#include <GL/gl.h>
#include <GL/glut.h>
main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (0, 0);
    glutCreateWindow (“Nom de fenetère“);
    // register callbacks
    glutDisplayFunc(fonction_affichage);
    glutKeyboardFunc(fonction_clavier);
    // enter GLUT event processing cycle
    glutMainLoop();
}
```

Fonction d’affichage

Introduction à openGL

```
#include <GL/gl.h>
#include <GL/glut.h>
main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (0, 0);
    glutCreateWindow (“Nom de fenetère“);
    // register callbacks
    glutDisplayFunc(fonction_affichage);
    glutKeyboardFunc(fonction_clavier);
    // enter GLUT event processing cycle
    glutMainLoop();
}
```

Fonction clavier

Introduction à openGL

```
#include <GL/gl.h>
#include <GL/glut.h>
main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (0, 0);
    glutCreateWindow (“Nom de fenetère“);
    // register callbacks
    glutDisplayFunc(fonction_affichage);
     glutKeyboardFunc(fonction_clavier);
    // enter GLUT event processing cycle
    glutMainLoop();
}
```

Boucle de message

Introduction à openGL

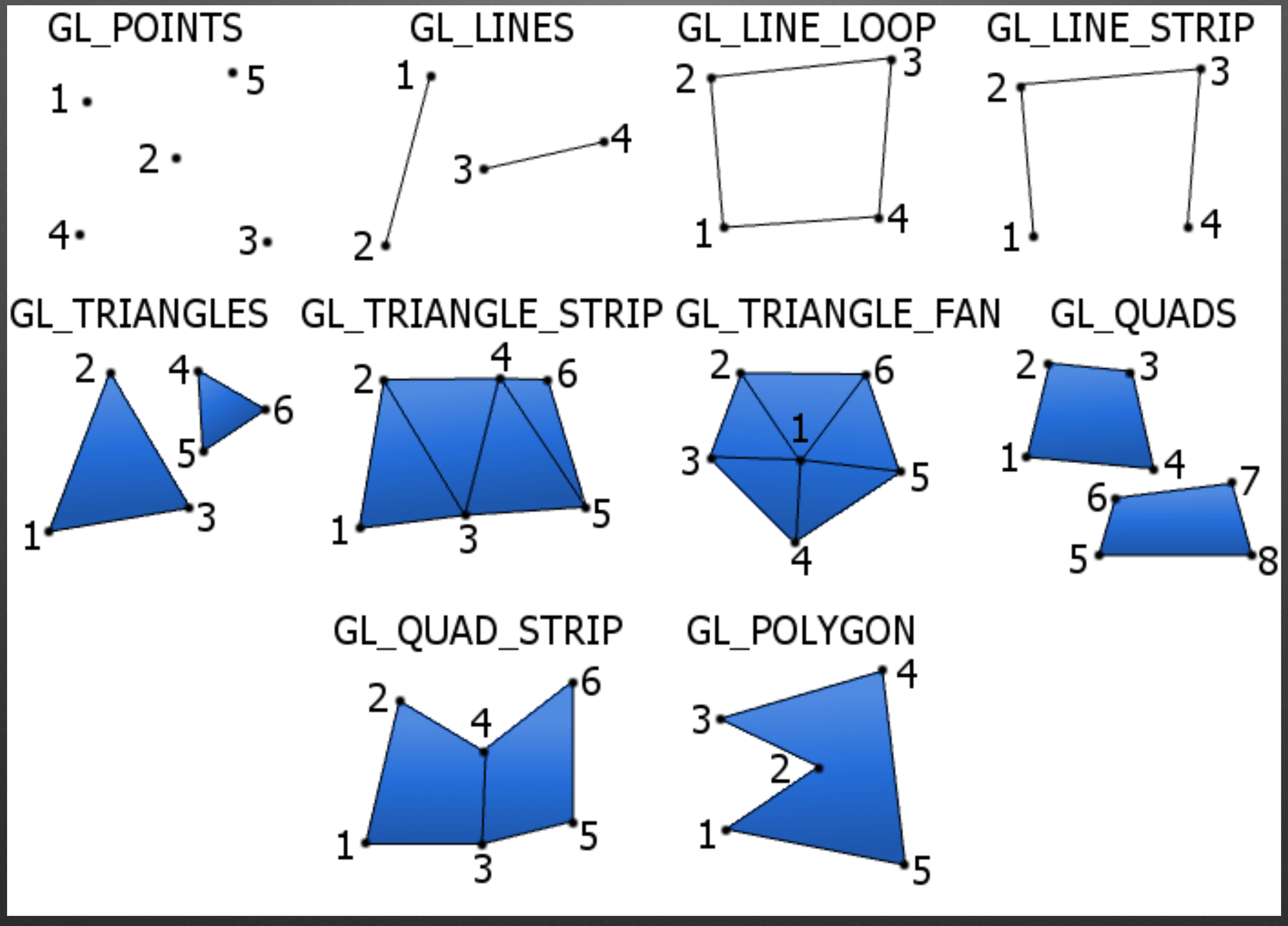
```
void foction_affichage(void) {

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glBegin(GL_TRIANGLES);
        glVertex3f(-0.5,-0.5,0.0);
        glVertex3f(0.5,0.0,0.0);
        glVertex3f(0.0,0.5,0.0);
    glEnd();

        glutSwapBuffers();
}
```

Effacer les buffers

Dessin d'un triangle

Introduction à openGL

Formes de base



Introduction à openGL
Vertex

les primitives sont formés de sommets qu'on appelle ***Vertex***

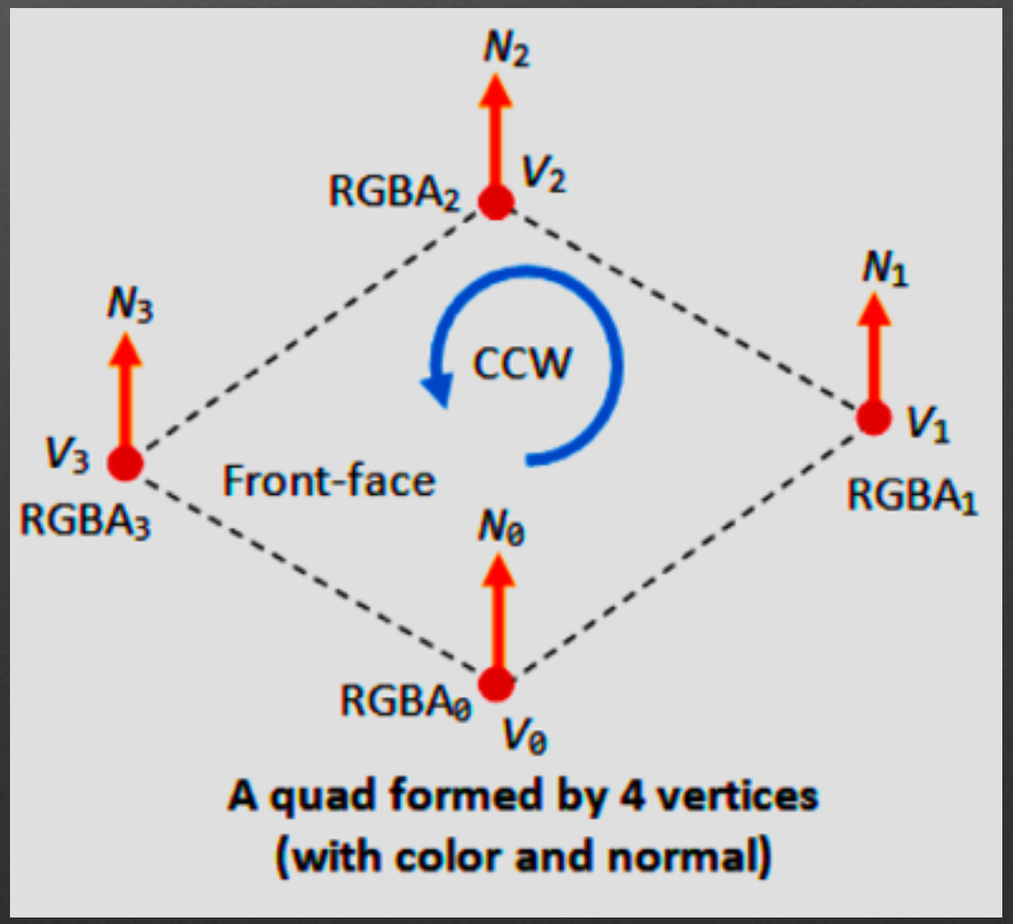
Un Vertex est défini par :

Une position dans l'espace 3D (x,y,z)

Une couleur exprimée en RGB ou RGBA

Un vecteur normal

Des coordonnées de texture (u,v)



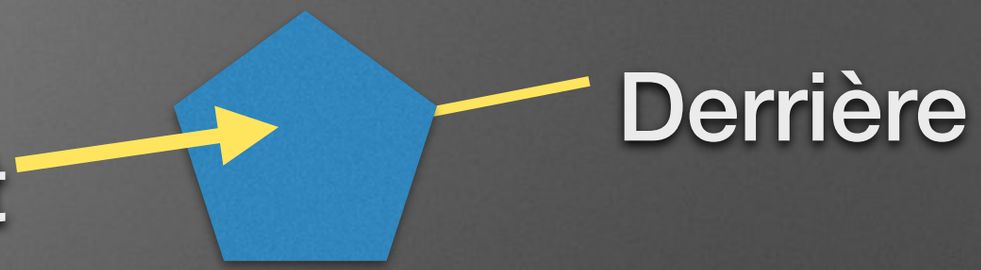
Introduction à openGL

Variantes de glVertex*()

	GLfloat	GLdouble	GLint
2D	glVertex2f()	glVertex2d()	glVertex2i()
3D	glVertex3f()	glVertex3d()	glVertex3i()
HOMOGÈNE	glVertex4f()	glVertex4d()	glVertex4i()

Introduction à openGL
Les polygones

Les polygones en 3D ont deux cotés : le devant et le derrière



glPolygonMode(GLenum face, GLenum mode);

GL_FRONT

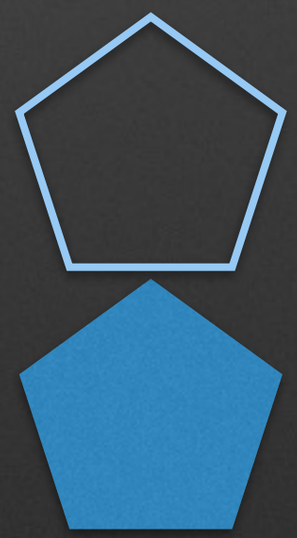
GL_BACK

GL_FRONT_AND_BACK

GL_POINT

GL_LINE

GL_FILL



Introduction à openGL
Les matrices OpenGL

OpenGL met à notre disposition trois matrices :

- Matrice de transformation-visualisation : **GL_MODELVIEW**
- matrice de projection : **GL_PROJECTION**
- matrice de texture : **GL_TEXTURE**

Unse seule matrice est active à la fois, nous pouvons changer la matrice active par la fonction **glMatrixMode**()

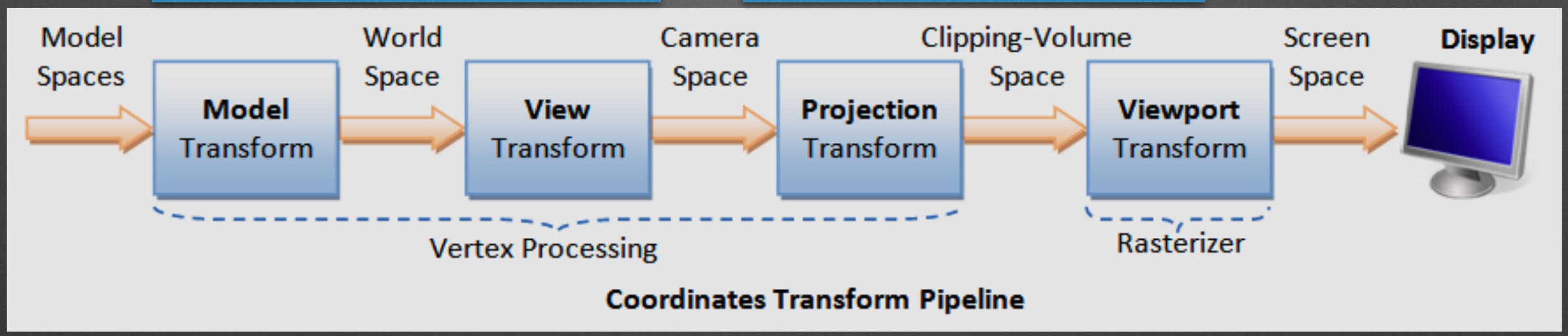
la fonction **glLoadIdentity()** permet d'initialiser la matrice active par la matrice d'identité

la fonction **glLoadMatrix()** permet d'initialiser la matrice active par des valeurs directement

Introduction à openGL
Les matrices OpenGL

GL_MODELVIEW

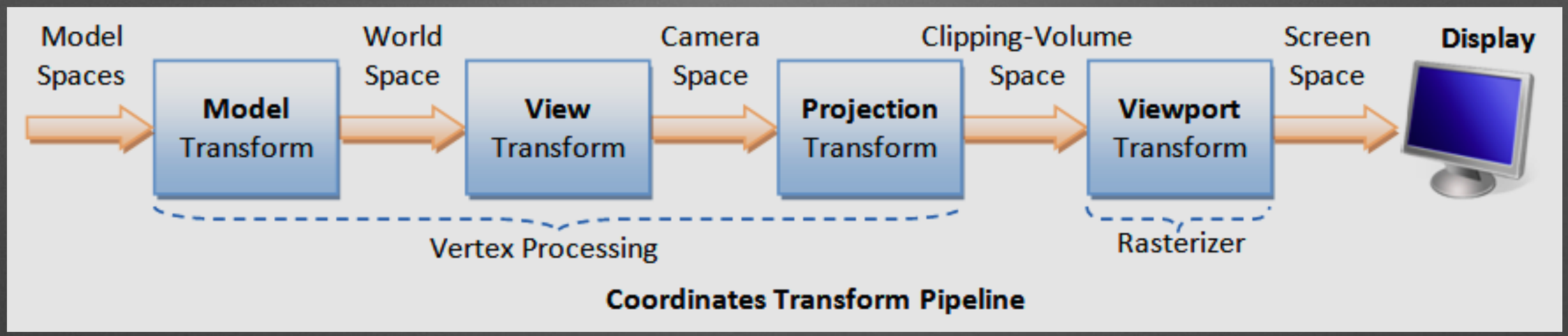
GL_PROJECTION



Coordinates Transform Pipeline

Introduction à openGL
Les matrices OpenGL

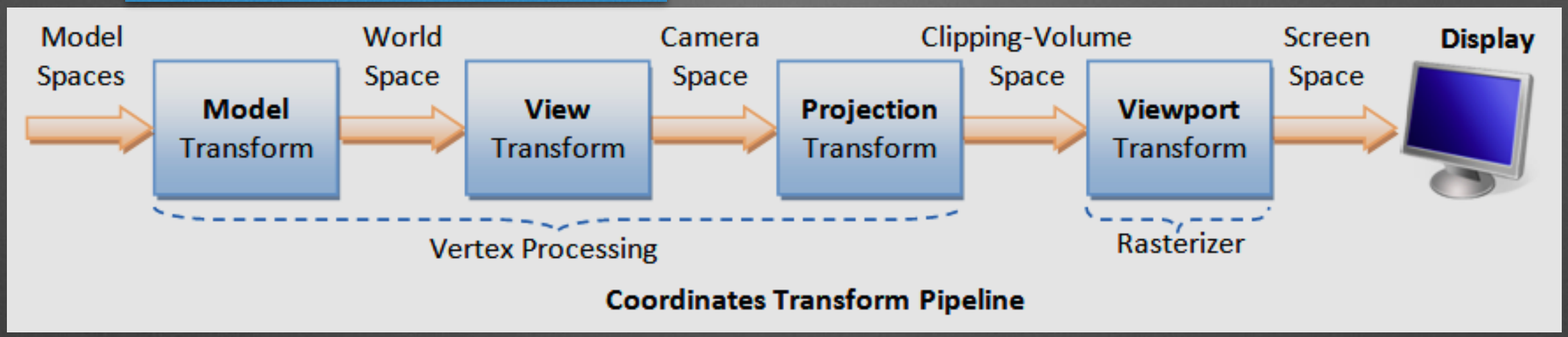
GL_MODELVIEW



La transformation du modèle : les objets graphiques sont définis dans un repère ou ils sont simples à concevoir, puis on transforme ces objets par des rotations, translations et changements d'échelle (affinités) pour les mettre dans le repère du monde. On fait subir ces transformations aux objets au moyen des fonctions glRotatef, glTranslatefet glScalef dans le mode GL_MODELVIEW

Introduction à openGL
Les matrices OpenGL

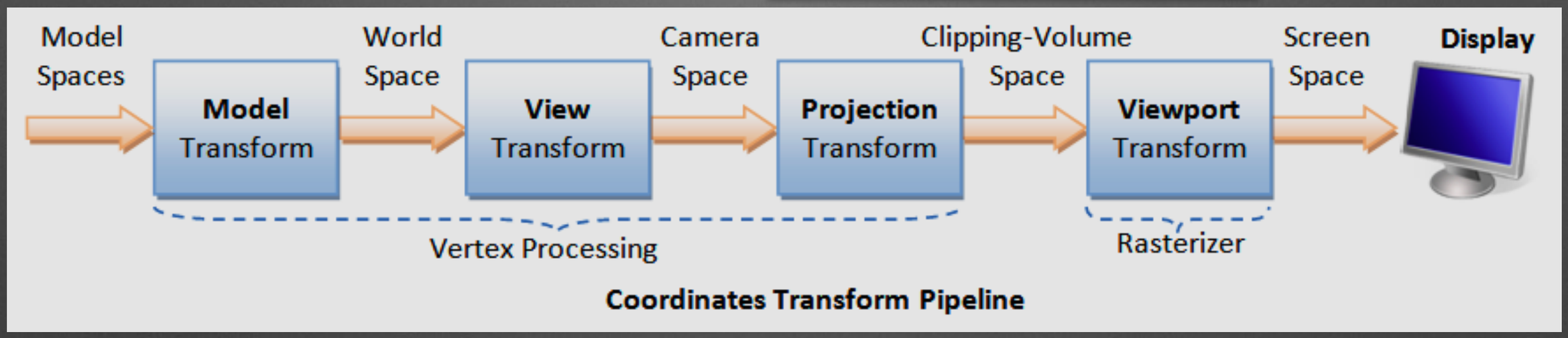
GL_MODELVIEW



La transformation de la caméra : une fois dans le repère du monde, tous les objets subissent un changement de repère vers le repère de la caméra. On fait subir cette transformation aux objets au moyen des fonctions ***gluLookAt***, ***glRotatef***, ***glTranslatef*** et ***glScalef*** dans le mode **GL_MODELVIEW**.

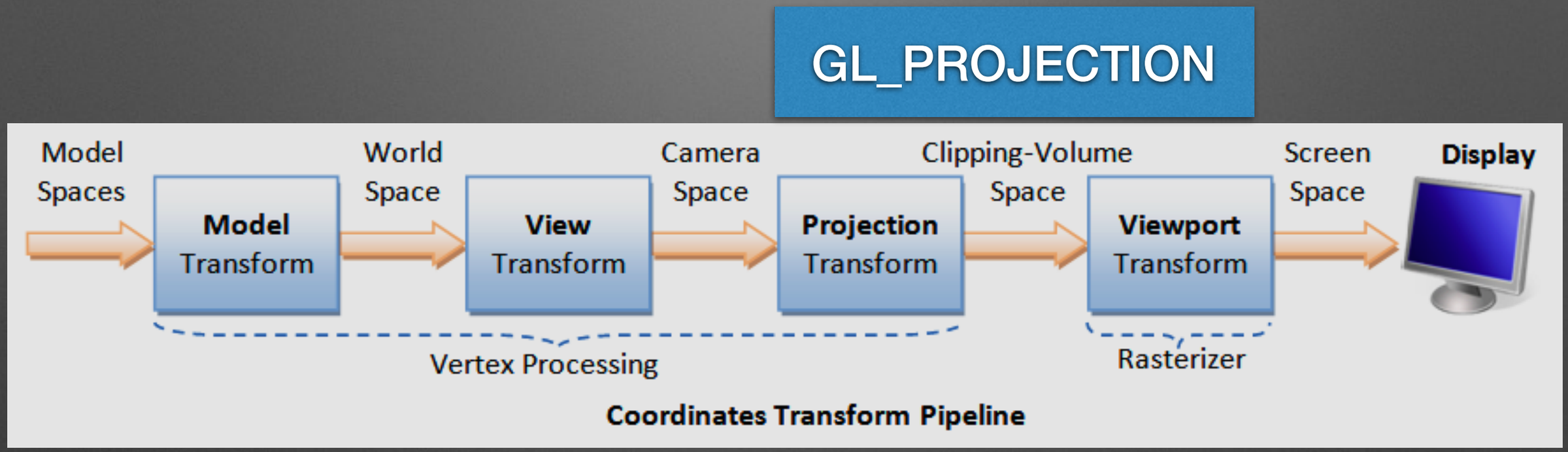
Introduction à openGL
Les matrices OpenGL

GL_PROJECTION



Projection : une fois dans le repère de la caméra, les objets subissent une projection vers une image 2D (dans un buffer OpenGL). Pour régler les paramètres de cette projection, on utilise la fonction ***gluPerspective*** dans le mode **GL_PROJECTION**.

Introduction à openGL
Les matrices OpenGL



Transformation 2D : une fois générée l'image 2D dans un buffer, elle subit des changement d'échelle sur les axes (affinités orthogonales) pour être ajustée aux dimensions de la fenêtre graphique à l'écran. Cette dernière transformation est paramétrée par la fonction ***glViewport***.

Introduction à openGL
Positioner la caméra

La fonction **gluLookAt** permet de modifier du GL_MODELVIEW pour régler la transformation de la caméra

void gluLookAt(GLdouble px, GLdouble py, GLdouble pz, GLdouble Cx, GLdouble Cy, GLdouble Cy, GLdouble Vx, GLdouble Vy, GLdouble Vz);

1. les coordonnées de la position (px_x, py_y, pz_z) de la caméra;

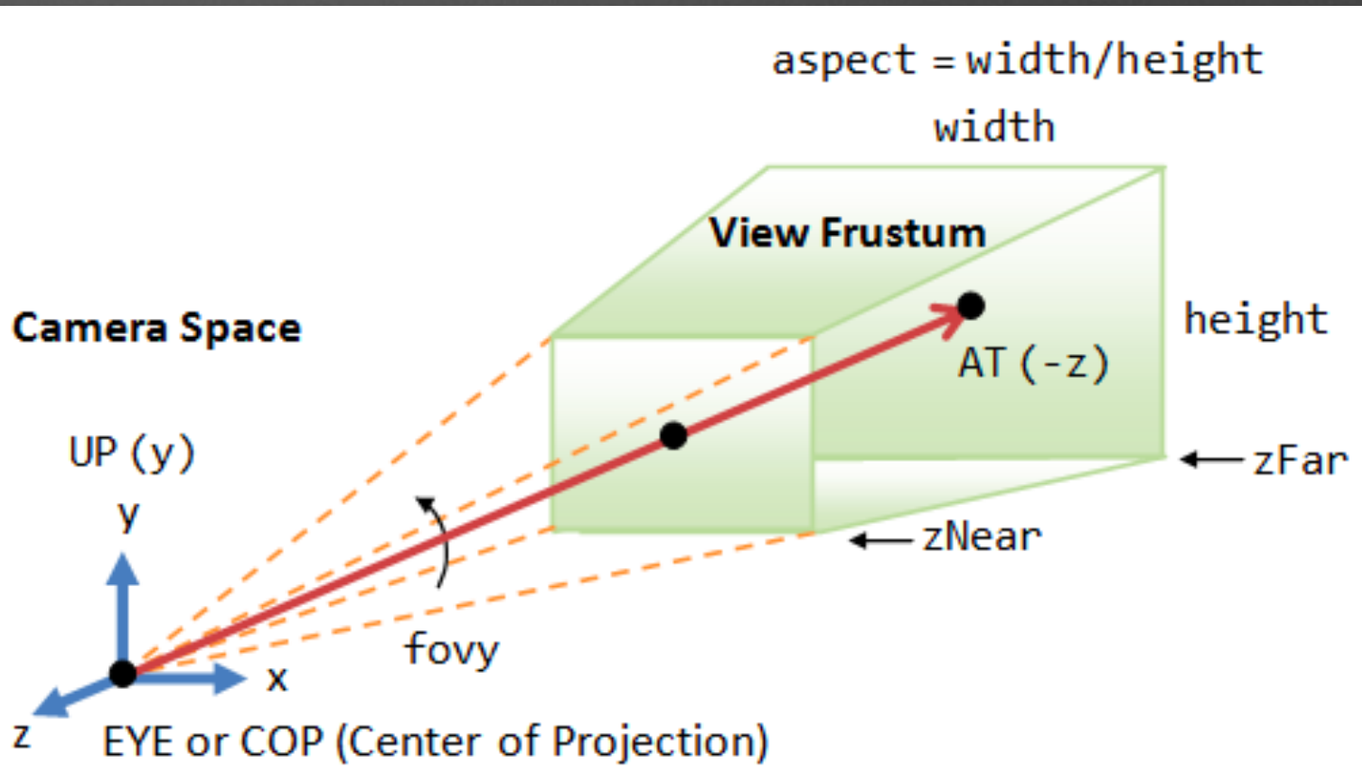
2. Les coordonnées (Cx_x,Cy_y,Cz_z) d'un point dans la direction de visée (ce point est aussi appelé le centre) ;

3. Les coordonnées du vecteur UP de la caméra.

Introduction à openGL
Paramètres de caméra

Projection perspective

```
void gluPerspective(GLdouble fovy,
    GLdouble aspect,
    GLdouble z_Near,
    GLdouble z_Far)
```



Perspective Projection: The camera's view frustum is specified via 4 view parameters: fovy, aspect, zNear and zFar.

Introduction à openGL
Paramètres de caméra

Projection perspective

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

gluPerspective(45.0,1.0, 0.1,1000.0);
```

Activer la matrice de projection

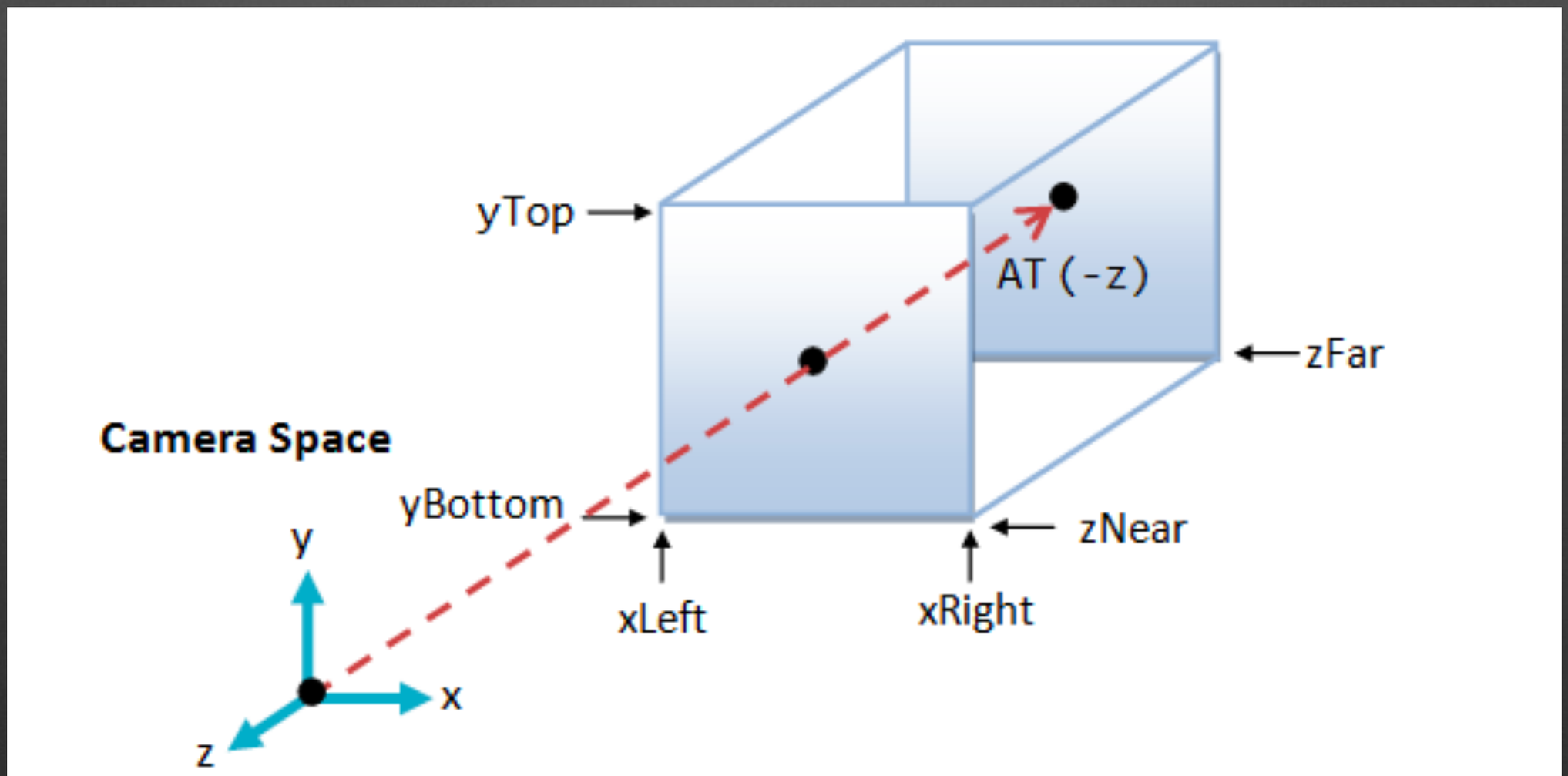
Réinitialiser la matrice

Paramétrer la caméra

Introduction à openGL
Paramètres de caméra

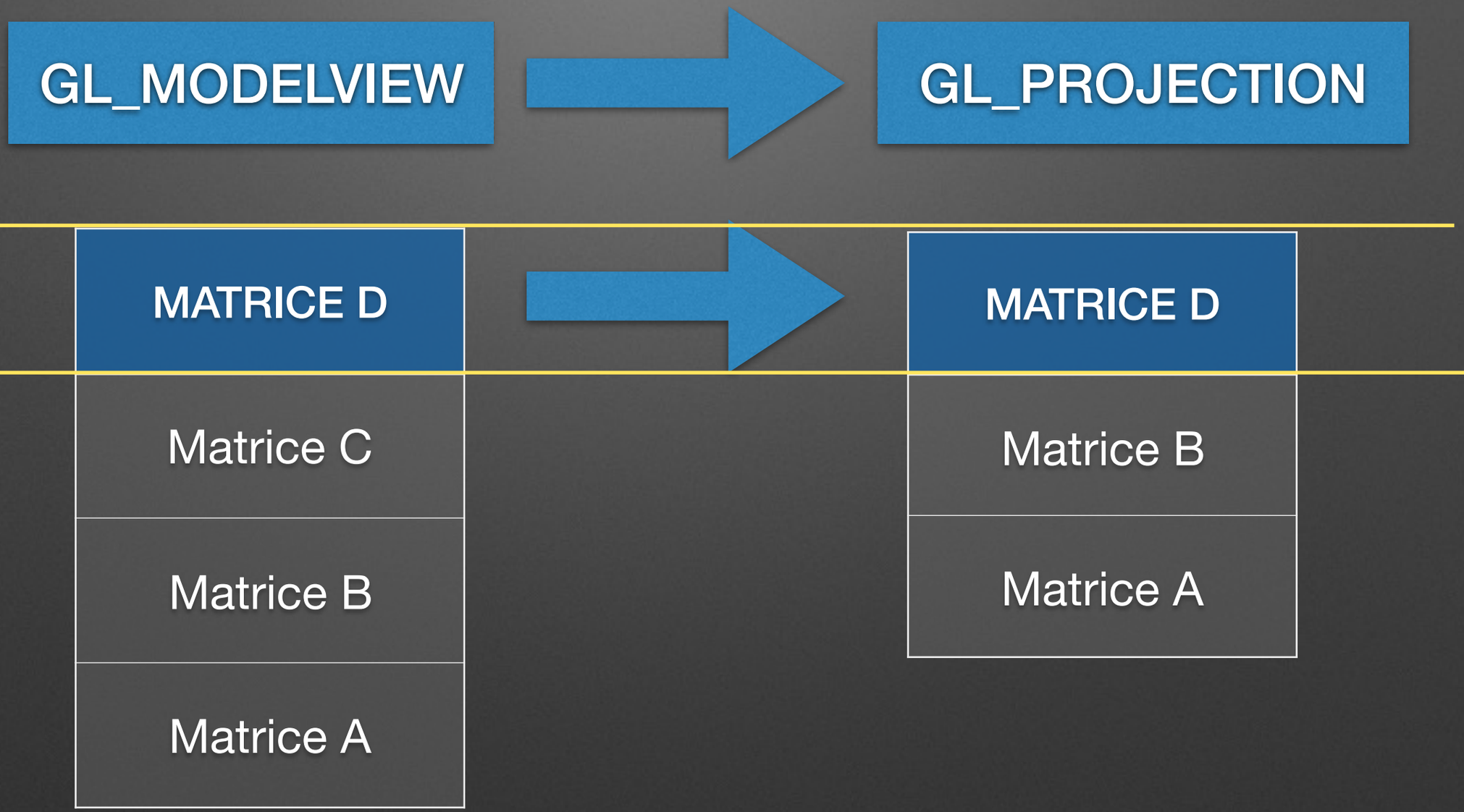
Projection orthographique

void **glOrtho**(GLdouble xLeft, GLdouble xRight, GLdouble yBottom, GLdouble yTop, GLdouble zNear, GLdouble zFar)



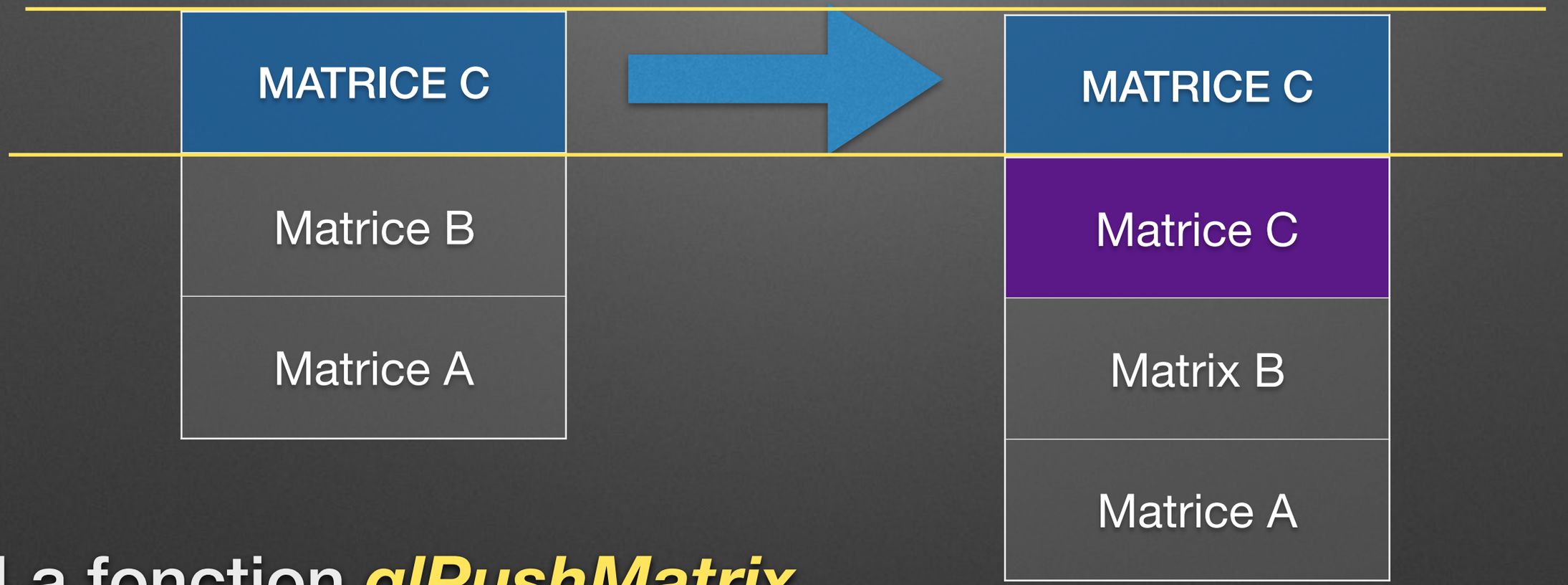
Orthographic Projection: Camera positioned infinitely far away at $z = \infty$

OpenGL
la pile des matrices



OpenGL
la pile des matrices

glPushMatrix()



La fonction ***glPushMatrix*** empile la matrice en cours en la ***dupliquant***

OpenGL
la pile des matrices

glPopMatrix()

La fonction ***glPopMatrix*** enlève la matrice dans la tète de la pile et la prochaine matrice dans la pile devient la matrice active

éclairage en infographie

Eclairage en infographie
Lighting

Eclairage en infographie
Lighting

Calculer l'interaction des objets avec la lumière selon un ***modèle d'éclairage***

Prendre en compte les variétés des sources de lumière et les propriétés des matériaux.

Garder ses calcules rapides pour les applications interactives.

Eclairage en infographie

Local

Uniquement prendre en compte :

Nombre de sources
positions des sources
leurs caractéristiques

La surface à éclairer
Sa direction (normale)

Global

prendre en compte :

Les reflexions multiples
Les ombres

Eclairage en infographie

Les composants d'un modèle d'éclairage

Lumière totale

=

Emissive

+

Ambiante

+

Reflective

Composante émissive

Relative uniquement à l'objet
N'influence pas les autres objets
la couleur est constante le long le la surface de l'objet.