



Introduction – Modélisation

Jean-Philippe Farrugia@univ-lyon1.fr

Présentation

- ⦿ Cours d'initiation à OpenGL ES
 - ⦿ API de synthèse d'images 3D.
 - ⦿ Dérivé mobile d'Open GL.
- ⦿ 8 heures, cours / TP.
- ⦿ Environnement Android,
 - ⦿ mais valable à 99% sur iOS.
- ⦿ Pas d'évaluation, pour votre culture ou vos projets...

Plan

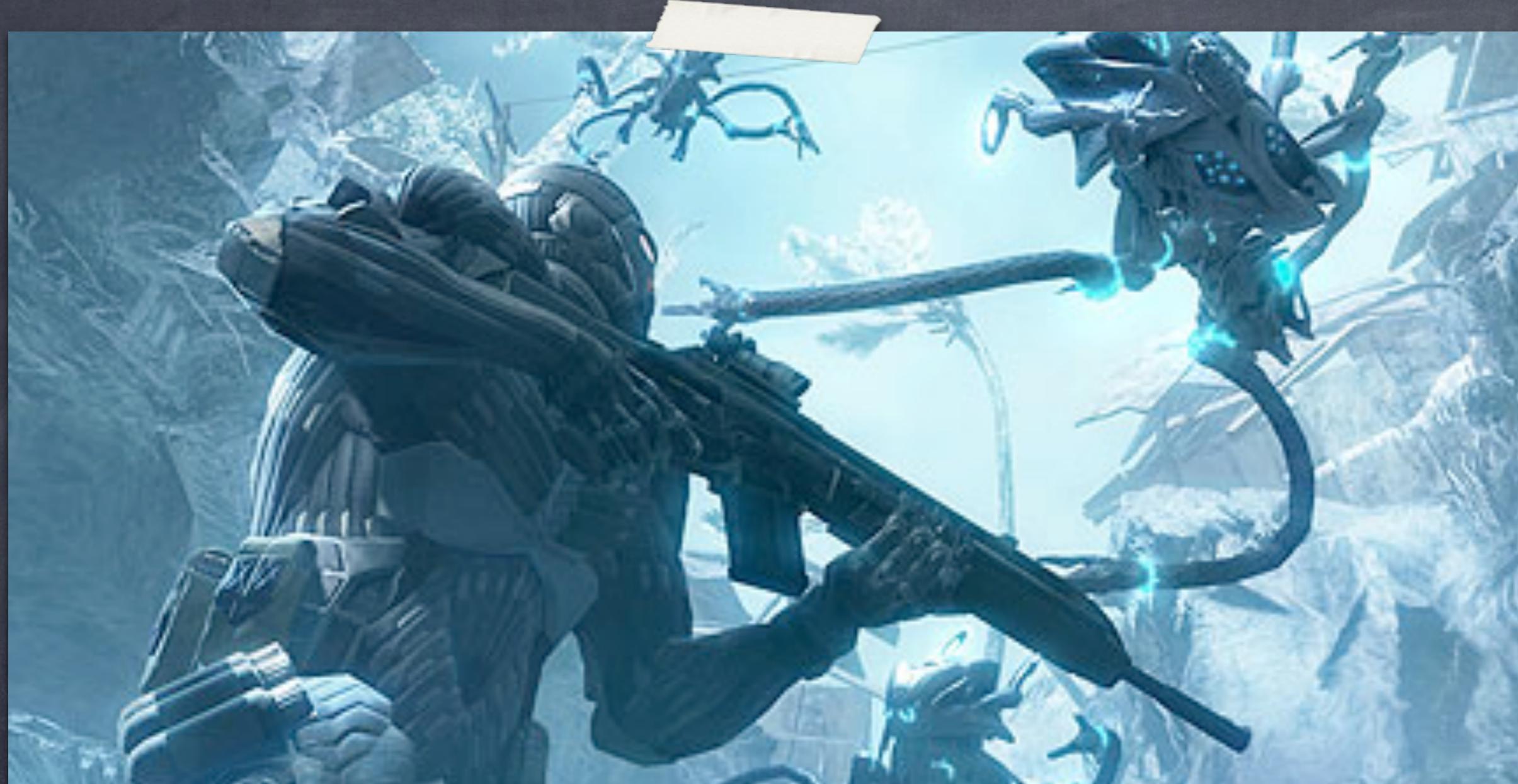
- ⦿ Notions de bases de synthèse d'images.
- ⦿ Le pipeline graphique et Open GL / GL ES.
- ⦿ Modélisation et transformations.
- ⦿ Eclairages et textures.
- ⦿ Open GL ES 2.0 / Shaders.

Synthèse d'images

- ⦿ Produire, à partir de données modélisées ou acquises, une nouvelle image.
- ⦿ Plusieurs types de résultats :
 - ⦿ 2D ou 3D.
 - ⦿ Plan ou stéréoscopique.
 - ⦿ Réaliste ou stylisé.
 - ⦿ Offline, interactif, temps réel.



Exemple



Exemple



Exemple



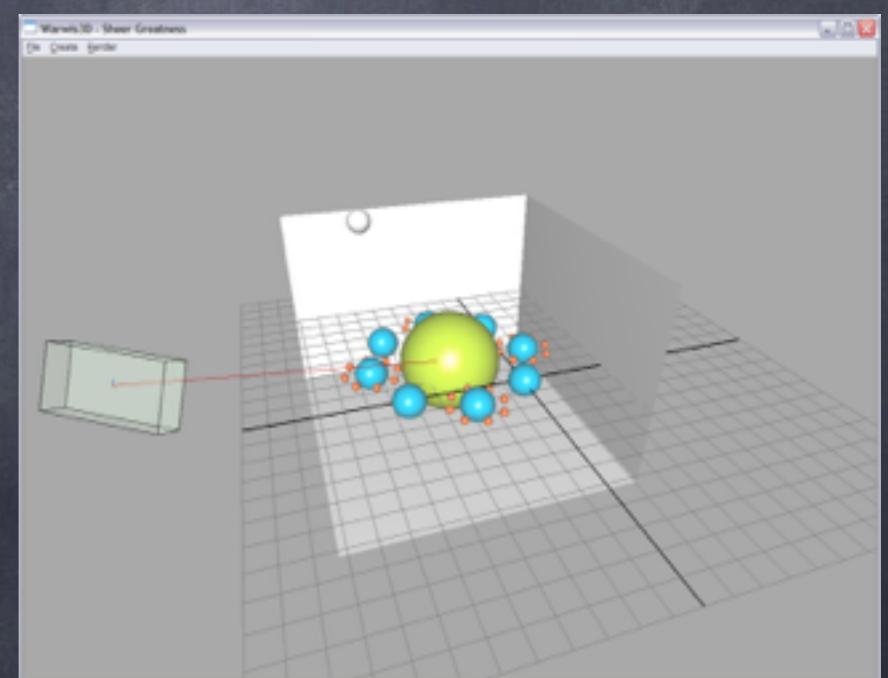
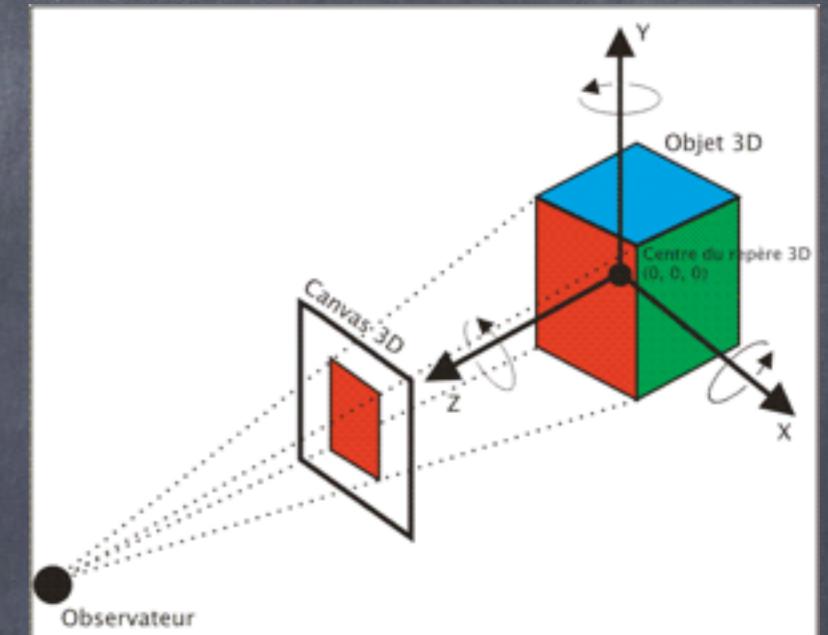
Exemple

Définitions

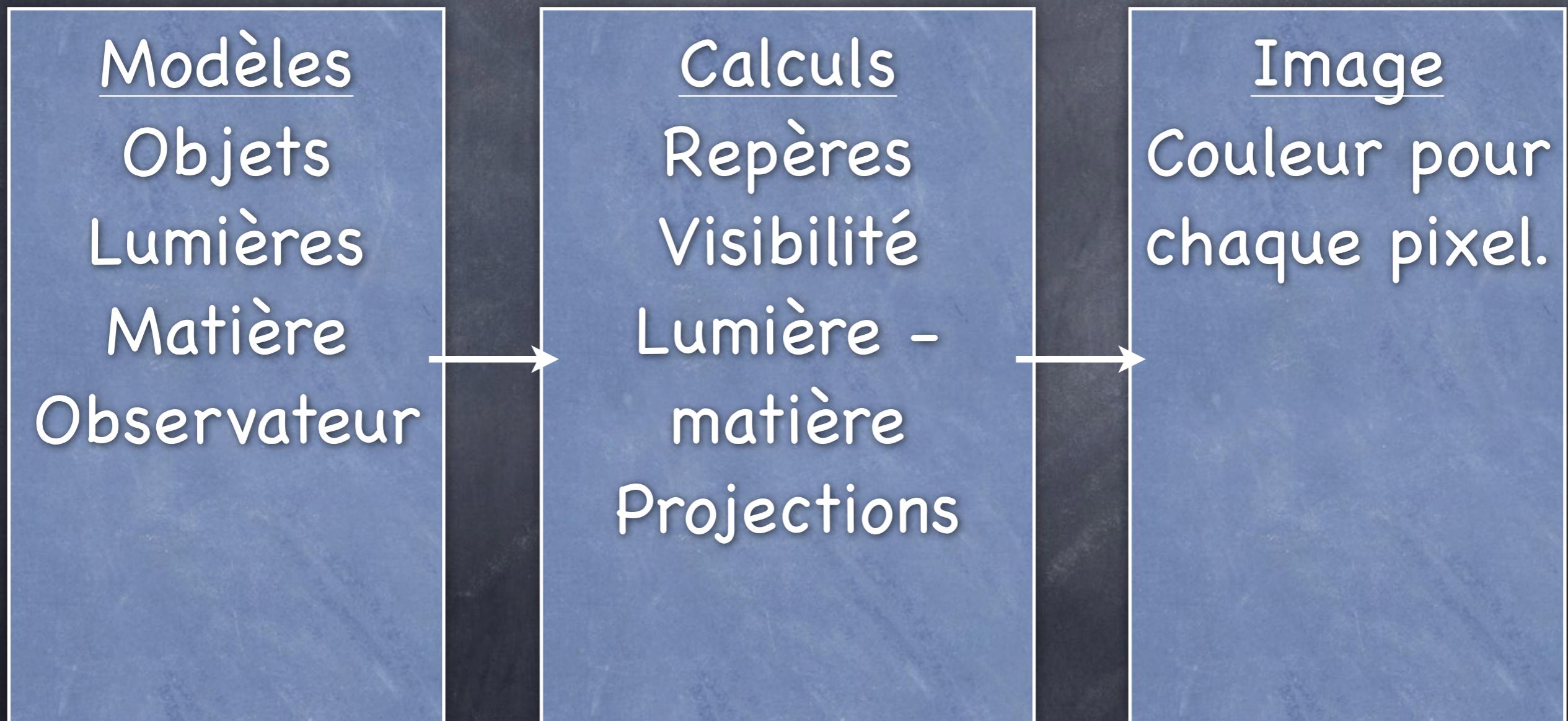
- ⦿ Une image : ensemble ordonné de pixels colorés.
 - ⦿ Nombre et taille des canaux.
 - ⦿ Dimensions.
- ⦿ Synthétiser une image, c'est :
- ⦿ Déterminer une couleur pour chaque pixel.

Définitions

- ➊ Une scène : ensemble d'éléments décrivant une situation «virtuelle».
- ➋ Éléments : Caméra, lumières, objets.
- ➌ Un modèle : une représentation numérique d'un élément de la scène.



Principe général

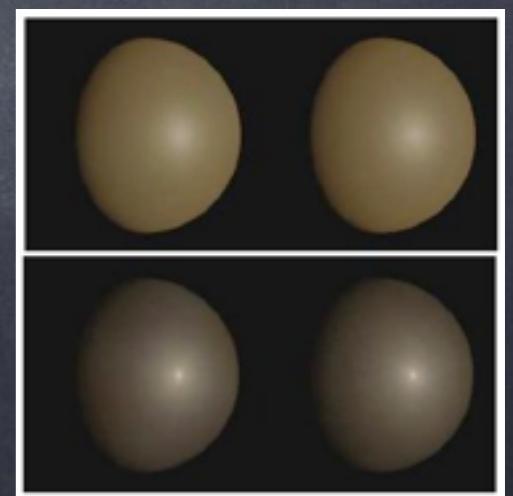
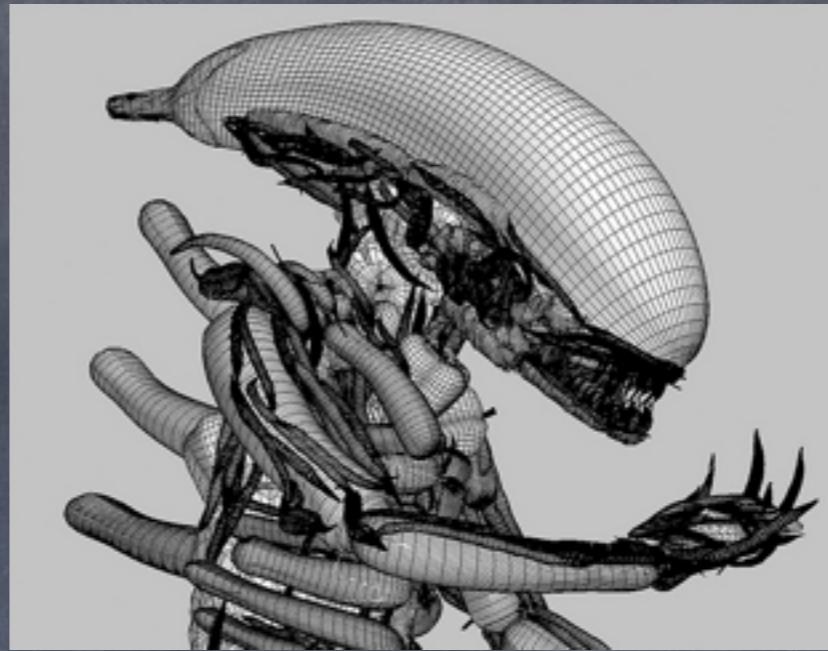


Modèles

- Un modèle pour chaque type d'élément de la scène.
 - Modèle d'objet.
 - Modèle de lumière, d'énergie, de couleur.
 - Modèle de l'observateur / caméra.

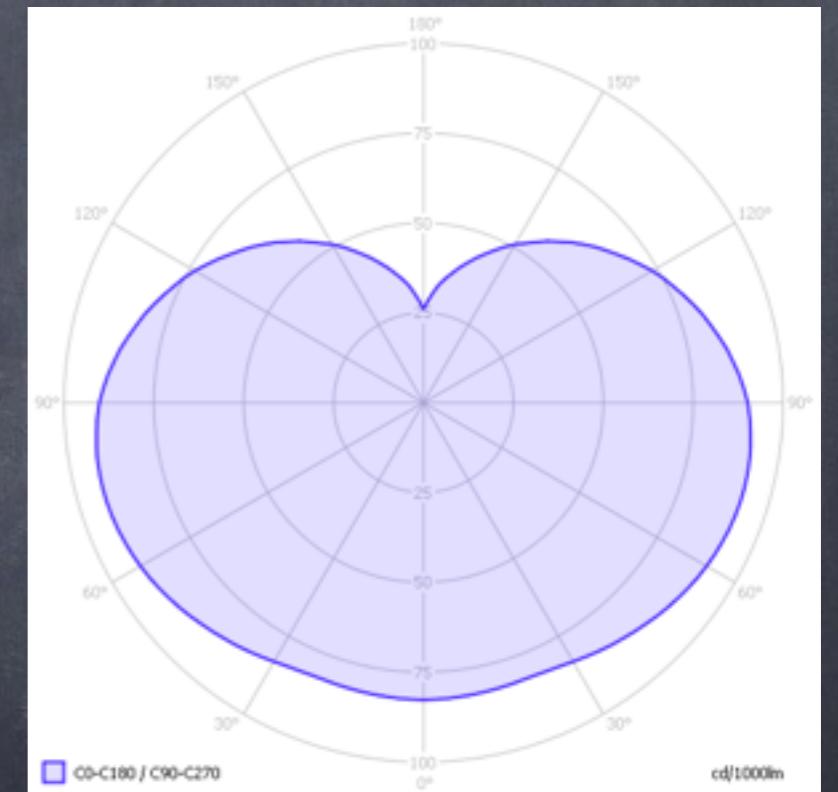
Modèle d'objet

- Modéliser :
 - Sa forme.
 - Sa matière, son aspect.
 - Sa position.
 - Ses mouvements.



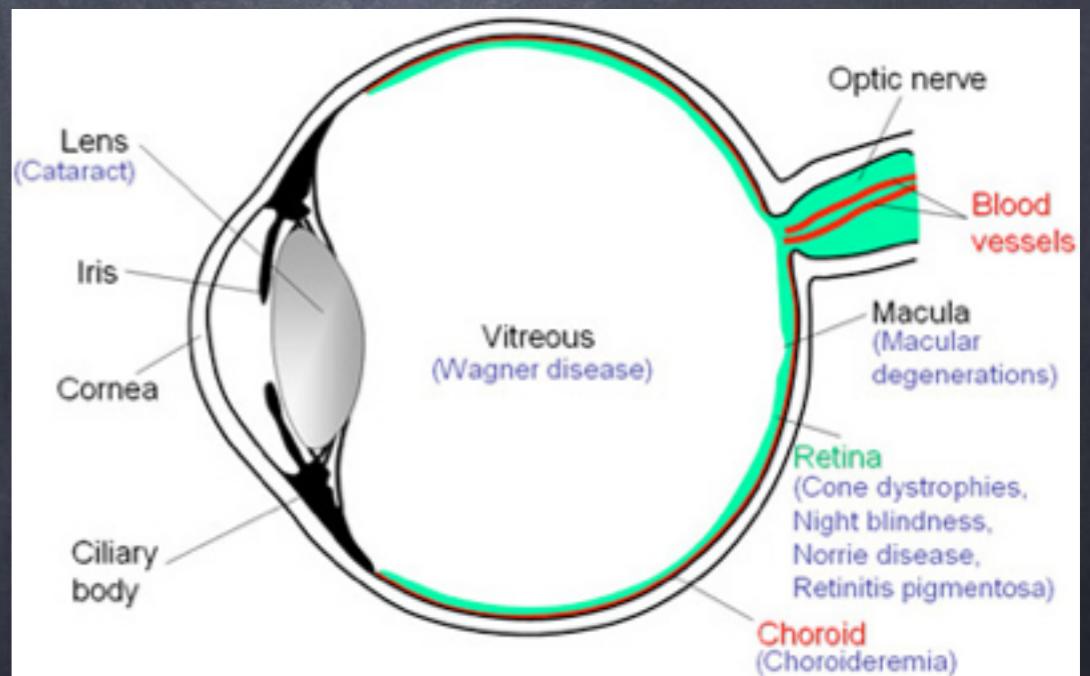
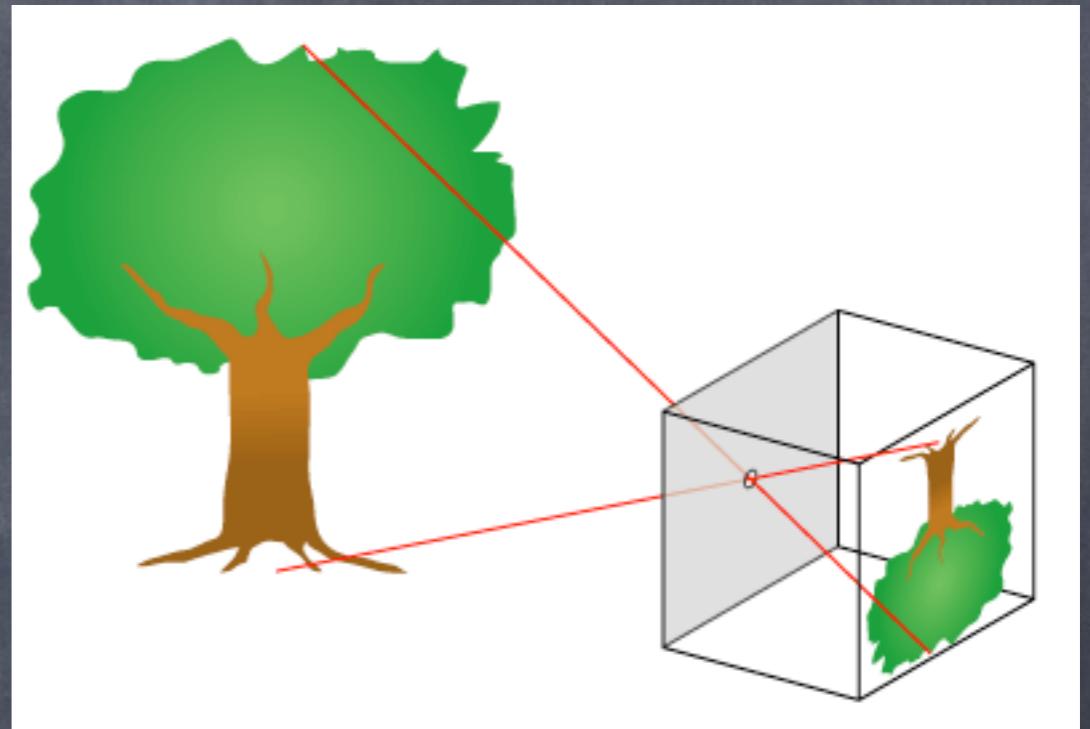
Modèle d'énergie

- Modélisation ?
- Perceptuelle ?
- Physique ?
- Rouge, vert bleu ?
- Sources ?
- Transport ?
- Rayons ?



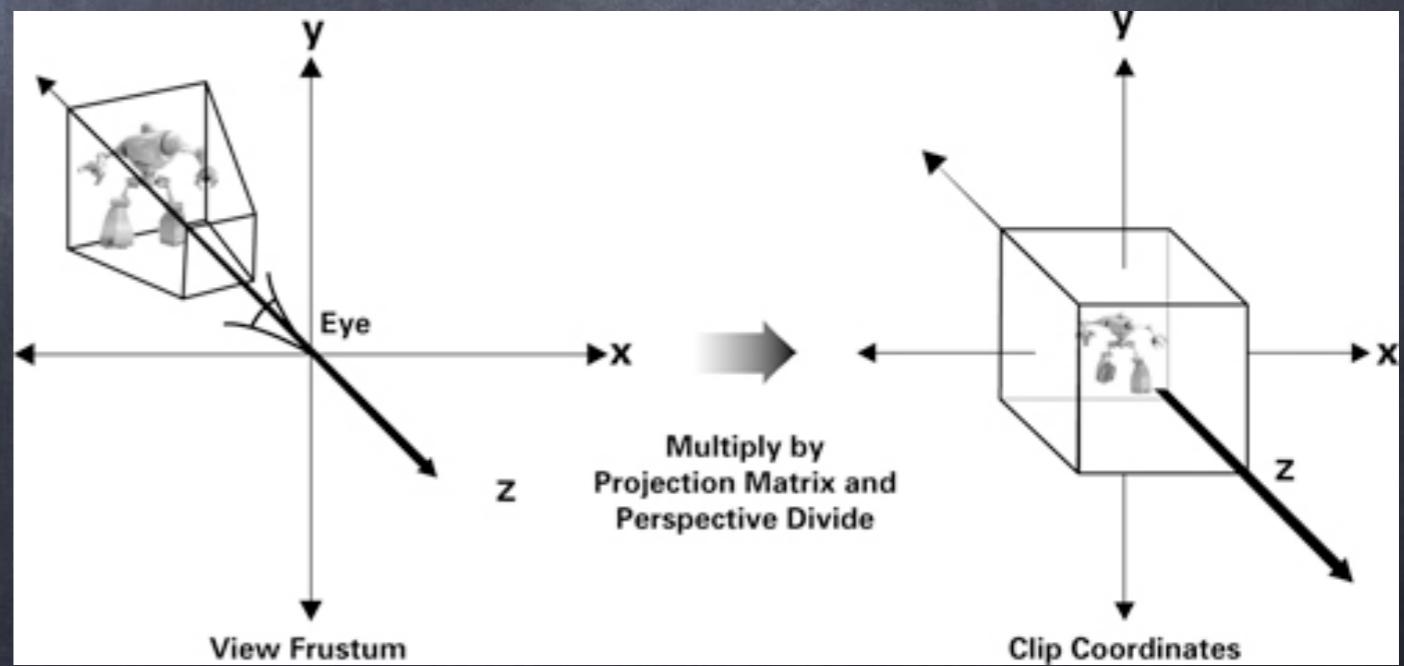
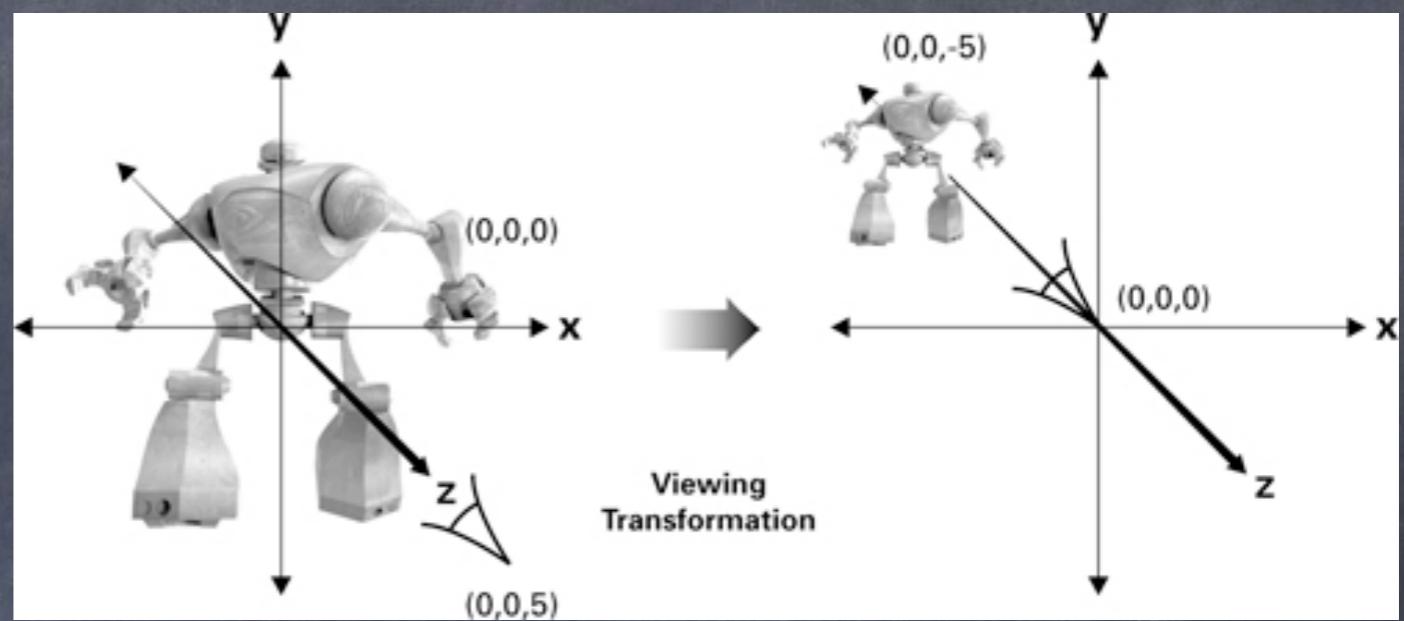
Modèle de l'observateur

- Ce qui regarde la scène :
- Objectif ?
- Oeil humain ?
- Autre (capteur) ?



Calculs

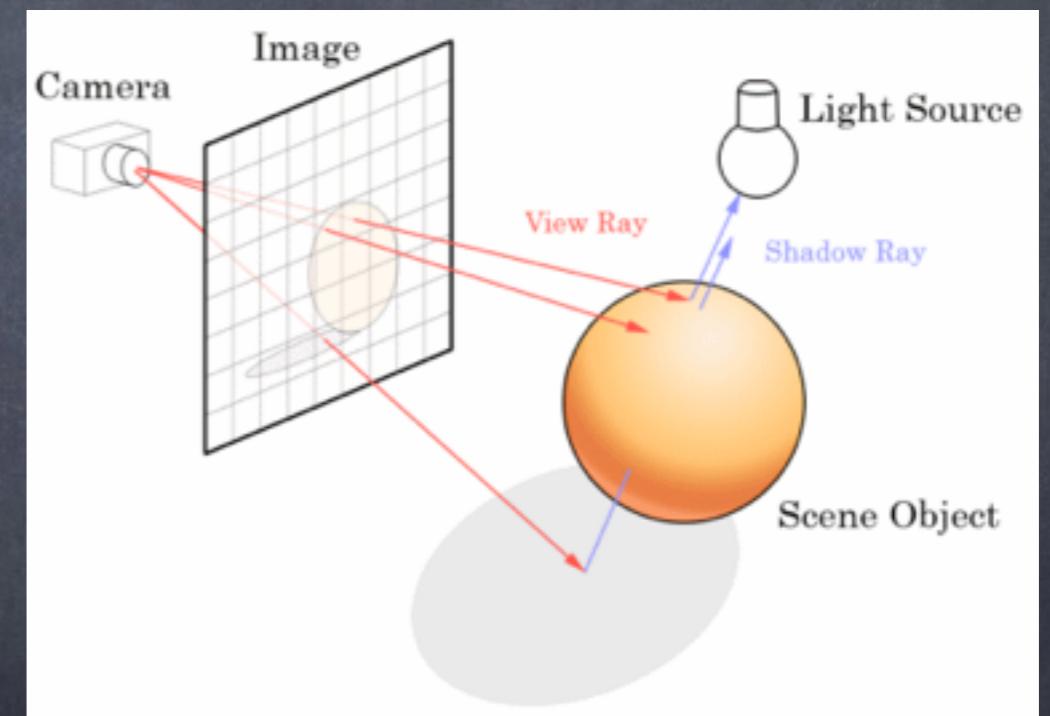
- Transformations et repères :
 - Paramètres de la caméra.
 - Placer les objets au sein de la scène.
 - Placer la caméra.



Calculs

- Interaction(s) lumière matière :

- Optique géométrique.
- Electromagnétisme.
- Thermique.
- Plus simple ?



Image

- Calculs physiques :
 - Valeur de luminance pour chaque pixel.
- Dispositif de reproduction ?
- Conversion en image affichable ?
 - Fonction gamma.
 - «Tone mapping».



Plan

- ⦿ Notions de bases de synthèse d'images.
- ⦿ Le pipeline graphique et Open GL / GL ES.
- ⦿ Modélisation et transformations.
- ⦿ Eclairages et textures.
- ⦿ Open GL ES 2.0 / Shaders.

Pipeline graphique

- Ensemble des processus qui mènent de la modélisation à l'image.

Modèles

Objets
Lumières
Matière
Observateur

Calculs

Repères
Visibilité
Lumière -
matière
Projections

Image

Couleur pour
chaque pixel.

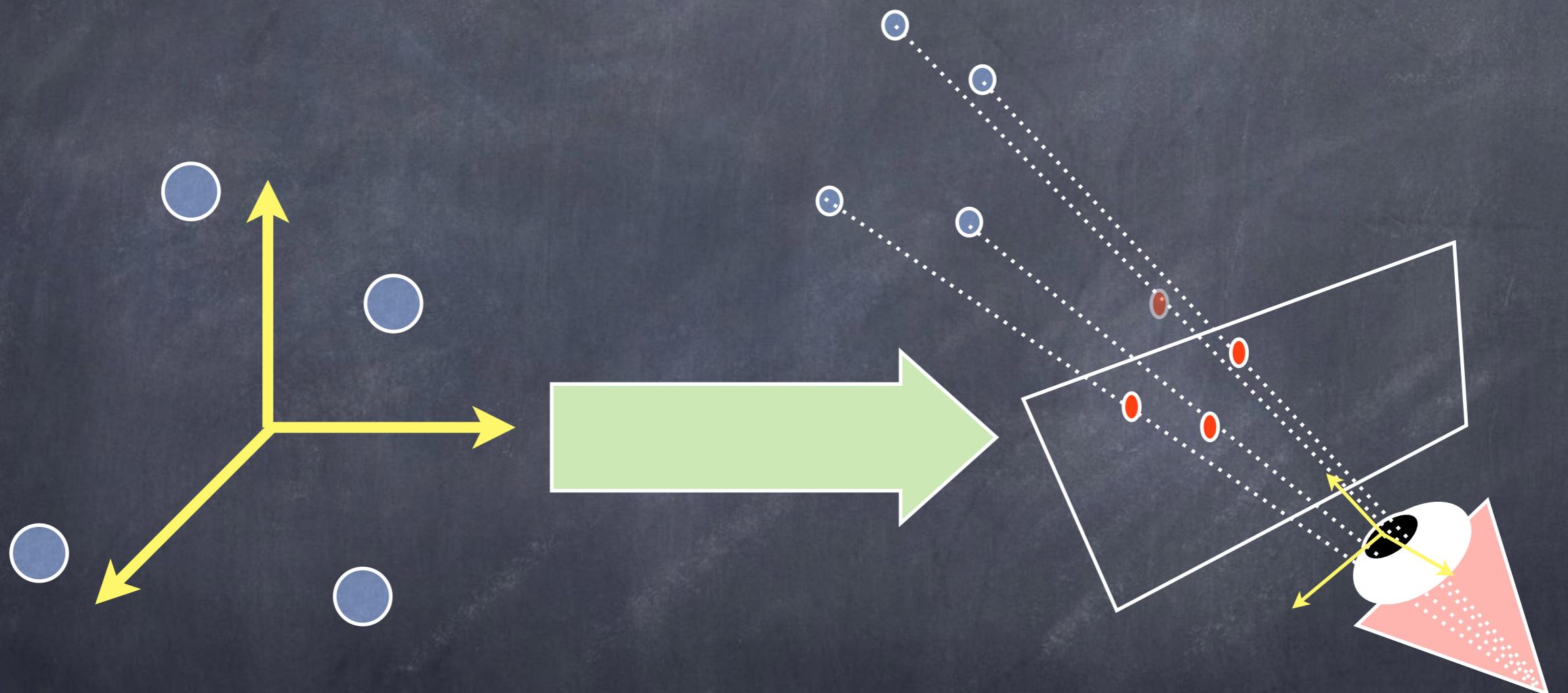
Pipeline graphique

- Suite d'opérations sur les modèles pour obtenir une image.
- Tâche globale du pipeline graphique : attribuer une couleur à chaque pixel de l'image.
- Processeur graphique (GPU) : implémentation hardware du pipeline graphique.

Le pipeline graphique

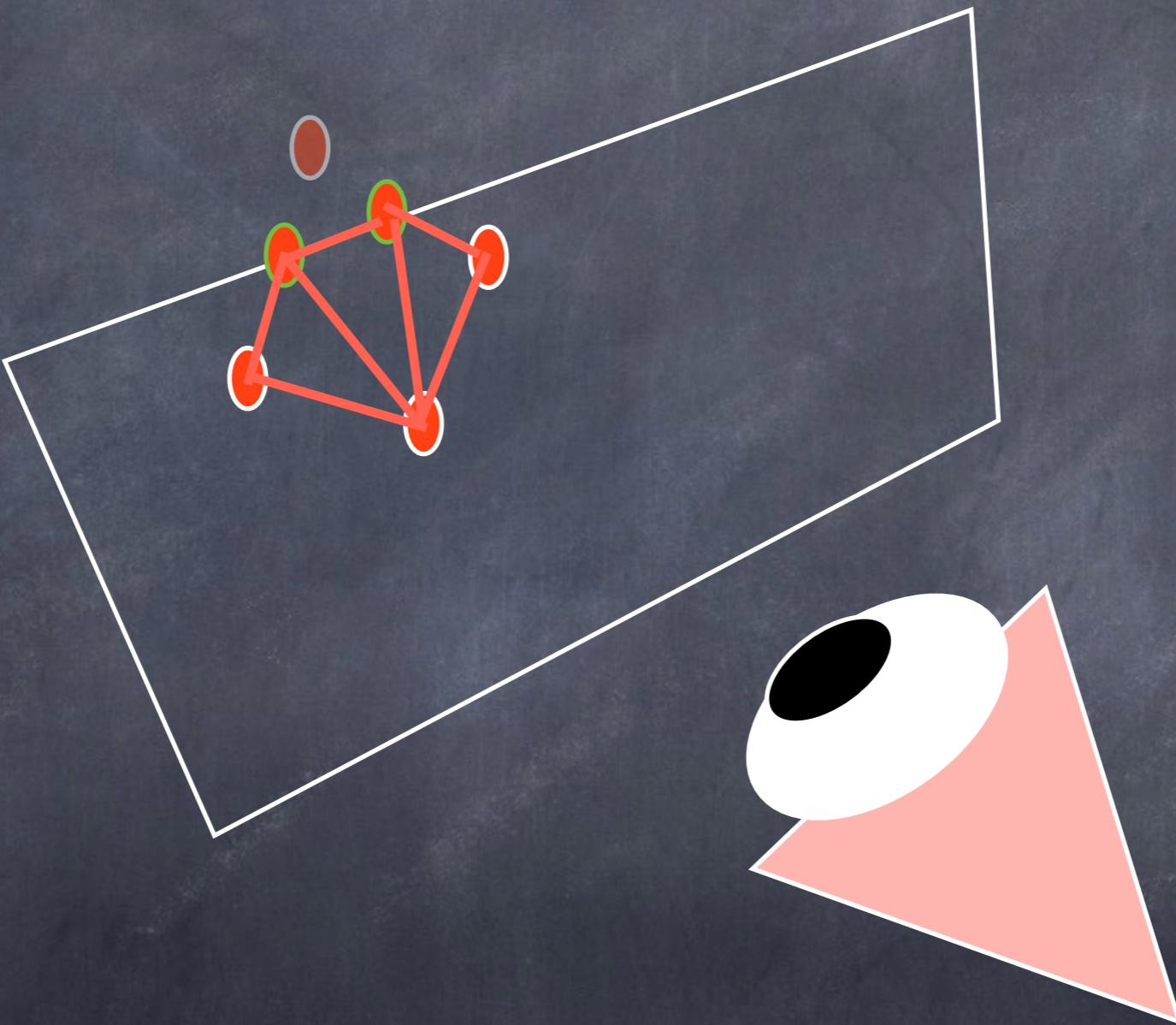
- ⦿ Principales étapes :
- ⦿ Transformations objet - monde - vue - projection.
- ⦿ Calcul éclairage.
- ⦿ Assemblage des primitives.
- ⦿ Rasterization (= discréétisation)
- ⦿ Calcul de la couleur des pixels.

Pipeline graphique



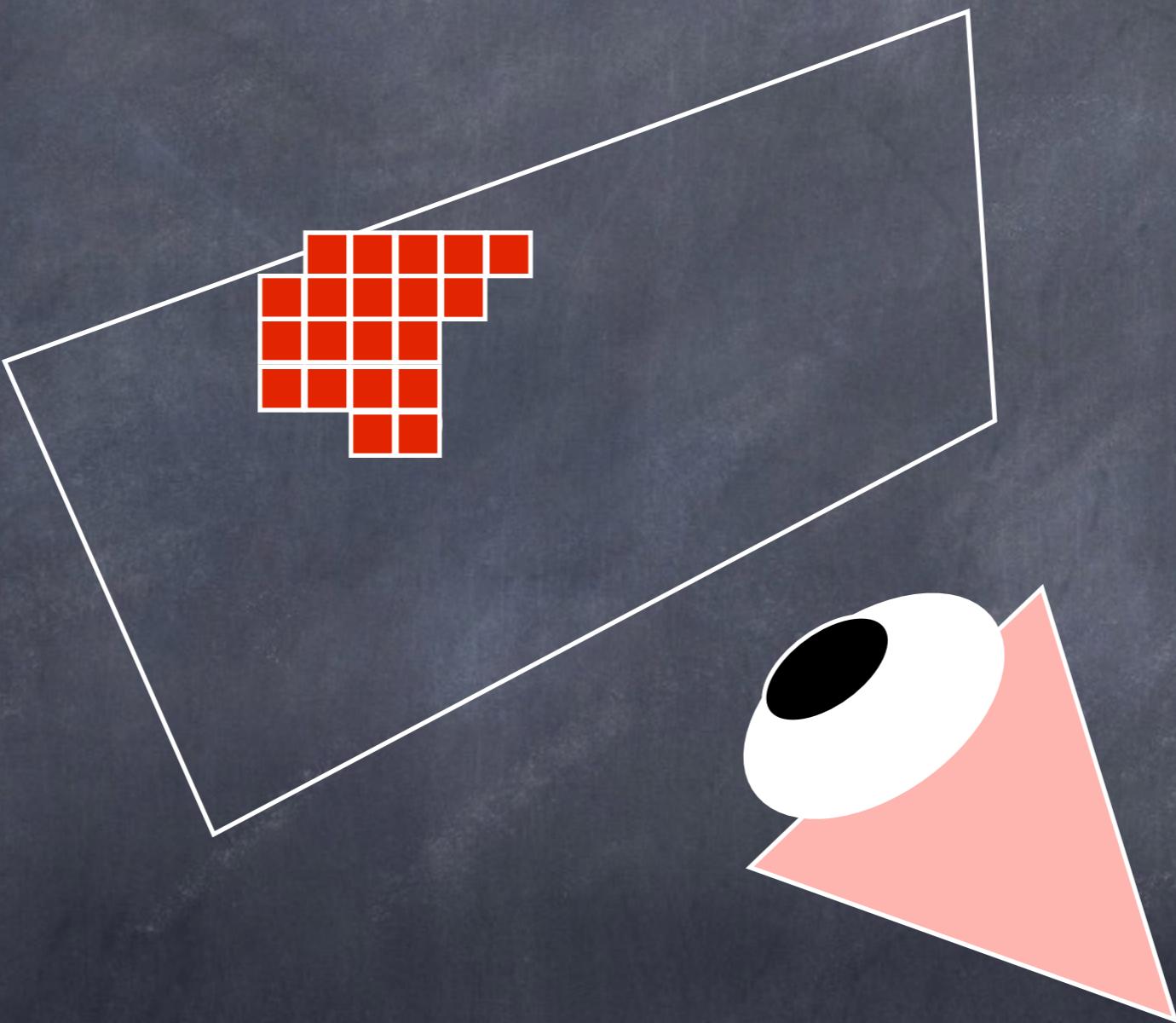
1) Transformations et éclairage

Pipeline graphique



2) Assemblage des primitives

Pipeline graphique



3) Rasterization et couleur

API Graphique

- ⦿ Bibliothèque permettant l'utilisation des capacités graphiques du système.
- ⦿ Permet de décrire les objets, les sources de lumières, les matières, la caméra...
- ⦿ Exécute le pipeline graphique sur ces données.
- ⦿ Eventuellement en temps réel.

API Graphique

- S'utilise avec un contexte :
- Stocke les paramètres de configuration et d'affichages.
- La création du contexte est la seule partie dépendante de l'OS.

API Graphique

- ⦿ Les plus courantes :
 - ⦿ OpenGL.
 - ⦿ Celle qui sera utilisée dans ce cours.
 - ⦿ Direct3D.

Direct3D

- ⦿ Créeée par Microsoft.
- ⦿ Version actuelle : Direct3D 11
- ⦿ Exclusivement sous Windows / Visual Studio.
- ⦿ Bibliothèque additionnelle : D3DX.
- ⦿ Très utilisée pour le jeu vidéo.
- ⦿ Version mobile : Direct 3D Mobile.
- ⦿ Windows Phone.

OpenGL

- ⦿ Crée par Silicon Graphics.
- ⦿ Version actuelle : OpenGL 4.4.
- ⦿ Multi-Plateforme : Windows, Linux, OS X...
- ⦿ Bibliothèques annexes : GLU, GLUT...
- ⦿ Nombreuses extensions.

OpenGL

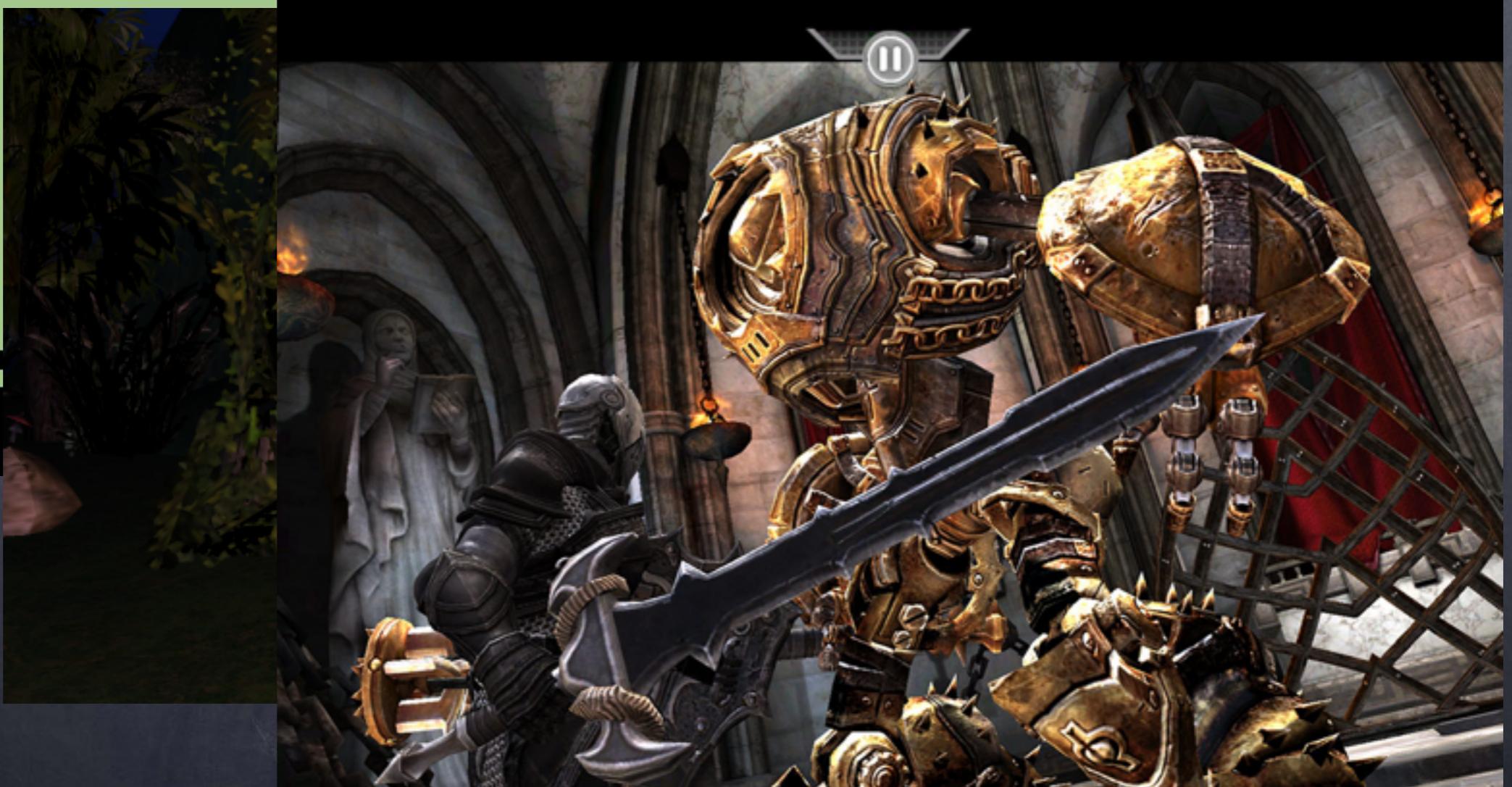
- ⦿ Se programme en C/C++, Python, Java, CAML...
- ⦿ Instructions préfixées par «gl».
- ⦿ Maintient des états.
 - ⦿ Transformations, couleur, faces cachées, textures...

OpenGL ES

- Version mobile d'Open GL
 - ES = «for Embedded Systems».
- Plus limitée qu'Open GL, mais plus légère :
 - Adaptée aux machines à capacités limitées.
 - Android, iOS, PS3, Symbian, WebGL, Blackberry...
- Version actuelle : 3.0.
- Documentation : <http://www.khronos.org/opengles/sdk/1.1/docs/man/>

Evolution...

0534



Options

OpenGL ES sur mobile

- ⦿ Dans tous les cas :
 - ⦿ Contexte OpenGL créé avec une vue particulière.
 - ⦿ Contenu de cette vue : le framebuffer.
 - ⦿ Zone qui contient l'image calculée, en mémoire vidéo.
 - ⦿ => On ne PEUT PAS intervenir directement sur son contenu.

Open GL ES sur Android

- Utilise les package javax.microeditions.khronos. et android.opengl.
- Instructions préfixées par gl.
- Création du contexte : 3 étapes.
 - Instanciation d'une GLSurfaceView.
 - Affectation d'un Renderer à cette vue.
 - Détails sur le slide suivant...
 - Affectation de la GLSurfaceView aux ressources.

Code principal

```
package my.testGL;

import android.app.Activity;
import android.os.Bundle;
import android.os.Bundle;
import android.opengl.GLSurfaceView;
import android.view.Window;
import android.view.WindowManager;

public class TestGLActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        // Pour une application plein écran :
        // On enlève le titre :
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
        // On paramètre la fenêtre principale en plein ecran :
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
        // Création de la glView :
        GLSurfaceView myGLView = new GLSurfaceView(this);
        // Affectation du renderer:
        myGLView.setRenderer(new OpenGLRenderer());
        // Exemple simple : une seule vue, la vue GL.
        setContentView(myGLView);
    }
}
```

Open GL ES sur Android

- ⦿ Le Renderer est la classe qui contient le code OpenGL.
- ⦿ Classe virtuelle pure, doit être dérivée pour être utilisée.
- ⦿ Implémentation obligatoire de 3 méthodes :
 - ⦿ onSurfaceCreated : initialisation.
 - ⦿ onDrawFrame : dessin proprement dit.
 - ⦿ onSurfaceChanged : explicite.

Code du Renderer

```
package my.testGL;

import javax.microedition.khronos.egl.*;
import javax.microedition.khronos.opengles.*;

import android.opengl.GLU;
import android.opengl.GLSurfaceView.Renderer;

public class OpenGLRenderer implements Renderer
{
    public void onSurfaceCreated(GL10 gl, EGLConfig config)
    {
        // Ici : code indépendant des dimensions de la vue
        gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        gl.glShadeModel(GL10.GL_SMOOTH);
        gl.glClearDepthf(1.0f);
        gl.glEnable(GL10.GL_DEPTH_TEST);
        gl.glDepthFunc(GL10.GL_EQUAL);
    }

    public void onDrawFrame(GL10 gl)
    {
        // Ici : code de dessin. Dans cet exemple simple, on ne fait qu'effacer l'écran.
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    }

    public void onSurfaceChanged(GL10 gl, int width, int height)
    {
        // Ici : code d'initialisation dépendant des dimensions
        // Cette méthode sera appelée en cas de redimensionnement.
        gl.glViewport(0, 0, width, height);
        gl.glMatrixMode(GL10.GL_PROJECTION);
        gl.glLoadIdentity();
        GLU.gluPerspective(gl, 45.0f, (float)width/(float)height, 1.0f, 100.0f);
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();
        gl.glTranslatef(0.0f, 0.0f, -4.0f);
    }
}
```

Exercice 1

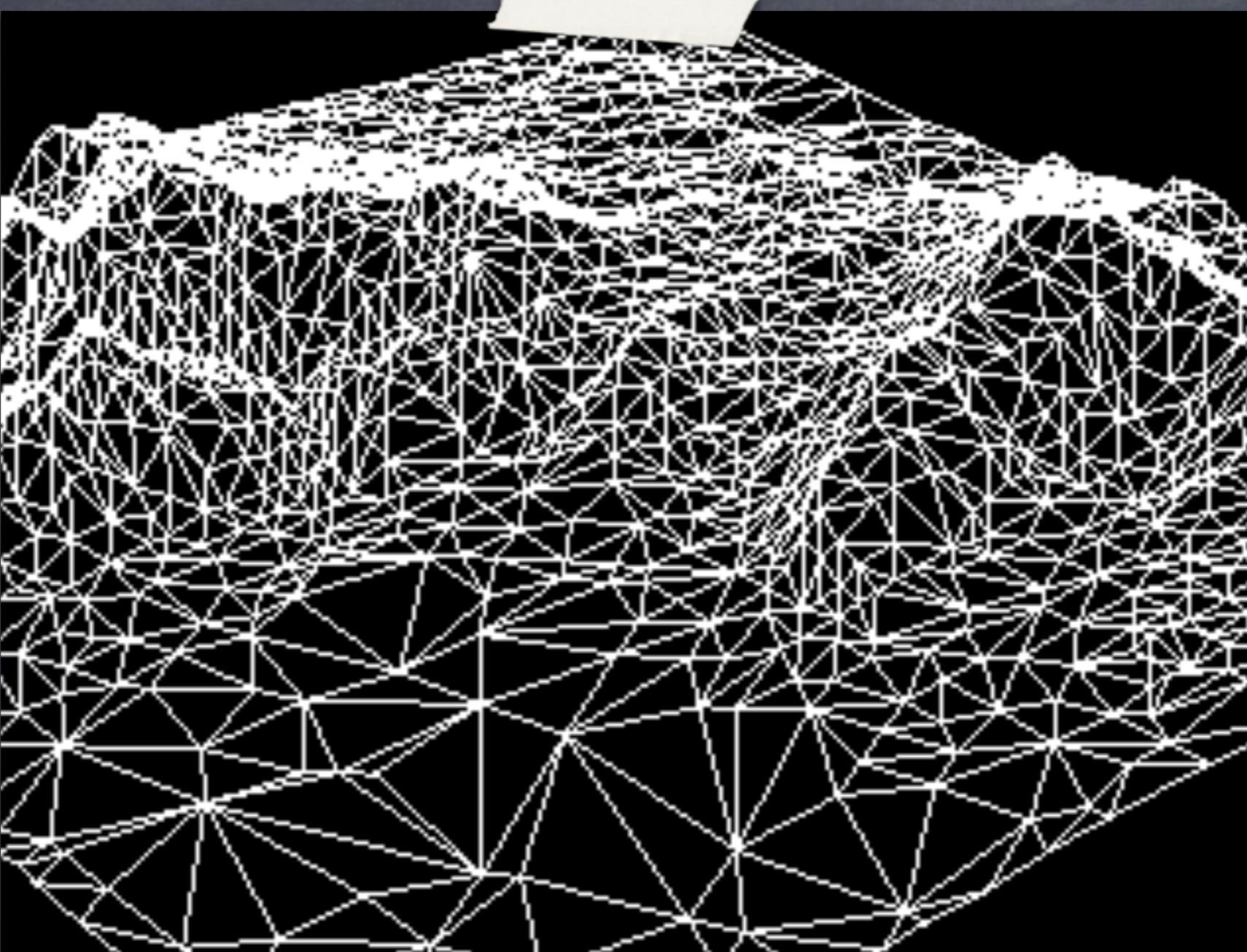
- ➊ Téléchargez et examinez le code exemple sur SPIRAL.
- ➋ Instructions données par l'intervenant.
- ➌ Identifiez chaque instruction GL. A quoi servent-elles ?
- ➍ Modifiez le programme pour changer la couleur de fond quand on tape sur l'écran.

Plan

- ⦿ Notions de bases de synthèse d'images.
- ⦿ Le pipeline graphique et Open GL / GL ES.
- ⦿ Modélisation et transformations.
- ⦿ Eclairages et textures.
- ⦿ Open GL ES 2.0 / Shaders.

Modéliser la géométrie

- ⦿ Représentation surfacique ou volumique.
- ⦿ Modélisation implicite.
- ⦿ Assemblage de primitives simples + opérations :
 - ⦿ Facettes (triangles, polygones).
 - ⦿ Surfaces (carreaux de Bézier, NURBS).
 - ⦿ Formes élémentaires : sphères, cylindres, cubes...
 - ⦿ Assemblage de «voxels».



Facettes

Modèle à facettes

- Définitions :
 - Un objet est un ensemble de faces.
 - Une face est un ensemble de sommets.
 - Un sommet est un ensemble d'attributs.

Description d'un sommet

- ⦿ Ensemble d'attributs :
 - ⦿ Position
 - ⦿ Couleur
 - ⦿ Normale
 - ⦿ Coordonnées de textures
 - ⦿ Autres ?

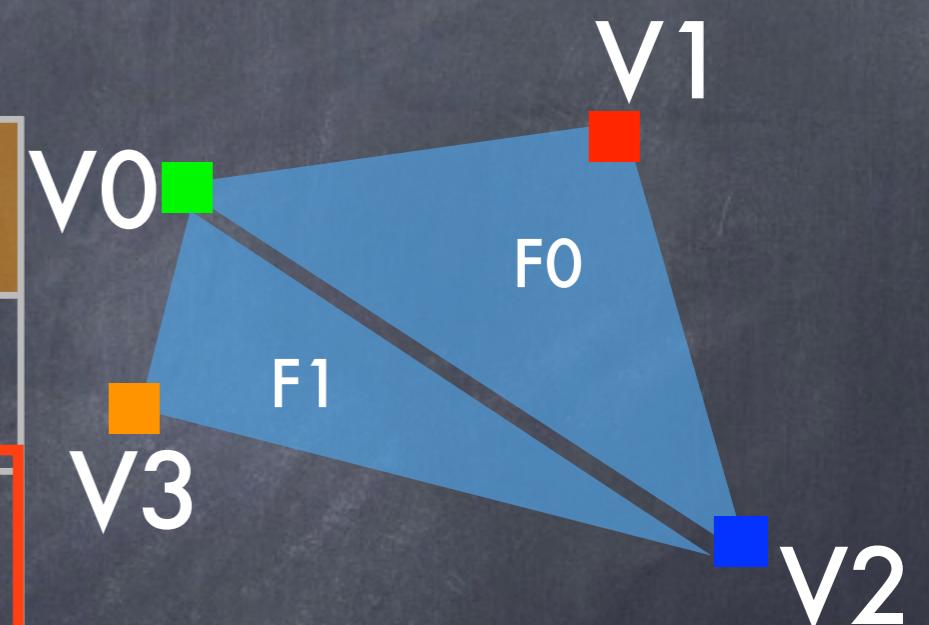
Description d'une face

- ⦿ Plusieurs méthodes :
 - ⦿ Liste de tous les sommets de la face.
 - ⦿ Liste d'indices + liste de sommets partagés.
- ⦿ Plusieurs stockages possibles :
 - ⦿ En mémoire centrale, coté client.
 - ⦿ En mémoire graphique, coté serveur.

Liste de sommets

Liste de faces

Faces	X0, Y0, Z0	X2, Y2, Z2	X1, Y1, Z1
F0	X0, Y0, Z0	X2, Y2, Z2	X1, Y1, Z1
F1	X0, Y0, Z0	X3, Y3, Z3	X2, Y2, Z2



Duplication de données...

Mode indexé

- Liste de sommets : pas très efficace.
- Ex avec un cube :
- Description par sommets : 24 positions.
 - Taille : $24 * 4 * \text{float}$
- Description indexée : 8 positions, 24 indices.
 - Taille : $24 * \text{uInt} + 8 * 4 * \text{float}$

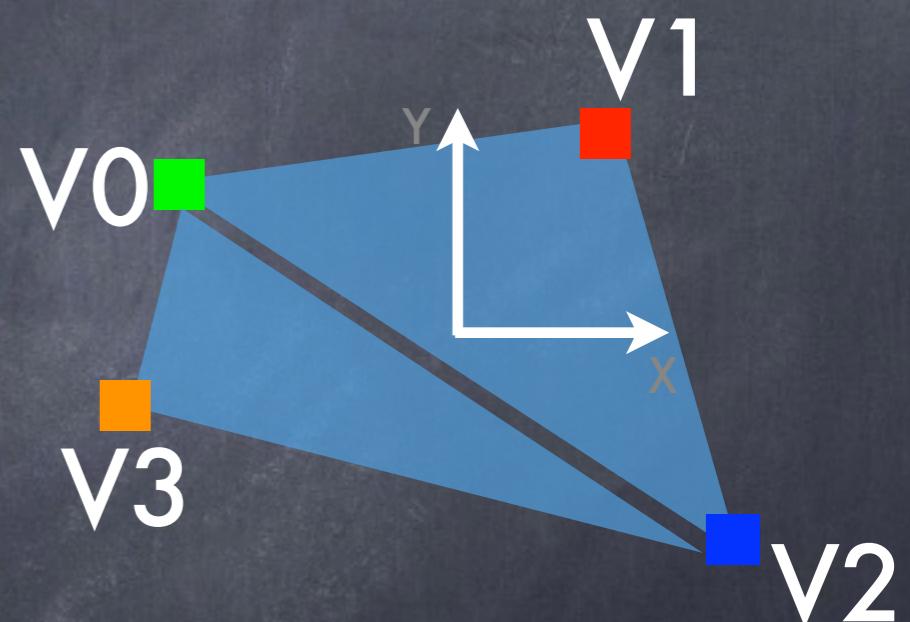
Méthode indexée

Liste de sommets

Vert.	0	1	2	3
X	-50	+50	+60	-60
Y	+50	+60	-50	-20
Z	0	+10	+20	+10

Liste d'indices

Faces			
F0	0	2	1
F1	0	3	2



Dans OpenGL

- Vertex Arrays :
 - Permettent le stockage des attributs en mémoire vidéo.
- Deux étapes :
 - Stocker les attributs dans un tableau.
 - Transférer ce tableau en mémoire vidéo.

Vertex Arrays

Stockage des coordonnées et des indices :

```
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import java.nio.ShortBuffer;

float[] vertices = {...};
float[] indices = {...};

ByteBuffer vertexBuffer;
ByteBuffer indexBuffer;

vertexBuffer = ByteBuffer.allocateDirect(vertices.length * 4);
vertexBuffer.order(ByteOrder.nativeOrder());
indexBuffer = ByteBuffer.allocateDirect(indices.length * 2);
indexBuffer.order(ByteOrder.nativeOrder());

// Buffer de coordonnées
vertexBuffer.asFloatBuffer().put(vertices);
vertexBuffer.position(0);

// Buffer d'indices
indexBuffer.asShortBuffer().put(indices);
indexBuffer.position(0);
```

Vertex Arrays

Transfert en mémoire vidéo et dessin

```
// Transfert des Coordonnées
gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
gl.glVertexPointer(3,
    GL10.GL_FLOAT,
    0,
vertexBuffer);

// Dessin proprement dit
gl.glDrawElements(GL10.GL_TRIANGLES,
    indices.length,
    GL10.GL_UNSIGNED_SHORT,
indexBuffer);

// Désactivation du vertex array
gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
```

Exercice 2

- ➊ Créez un programme pour afficher un rectangle avec les vertex arrays.
- ➋ Modifiez votre programme pour ajouter des couleurs à votre rectangle.
 - ➌ Une couleur par sommet.
 - ➍ Que donne le résultat ?
- ➎ Plus difficile : affichez un cube en utilisant les vertex arrays.
 - ➏ Pouvez-vous affecter une couleur par face ?