Introduction à la Synthèse d’images

Christophe Renaud

Licence Informatique 3
Année universitaire 2023-2024

Version 1.2 du 05/02/2024

Objectifs du cours

- Acquérir des notions fondamentales en S.I.
  - Modélisation
  - Rendu
- Acquérir un savoir faire en Three.js
  - Modélisation d’objets
  - Transformations géométriques
  - Animation

Evaluation

- Note finale :
  - 50% examen
  - 50 % contrôle continu
- Examen :
  - Sur machine
  - Sur compte examen
  - Documentation (cours, TP) disponible sur le compte
  - Documents papier autorisés
- Contrôle continu
  - Évaluation individuelle des TPS

Cours & TPS : https ://www-lisic.univ-littoral.fr/~renaud

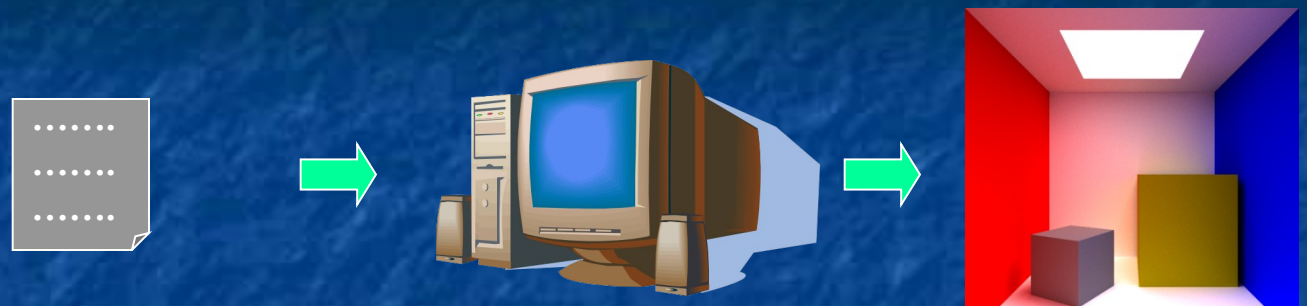
Plan du cours

1. Introduction
2. Modélisation d’objets 3D
3. Modèle d'éclairage local
4. Rendu temps réel
5. Introduction à Three.js

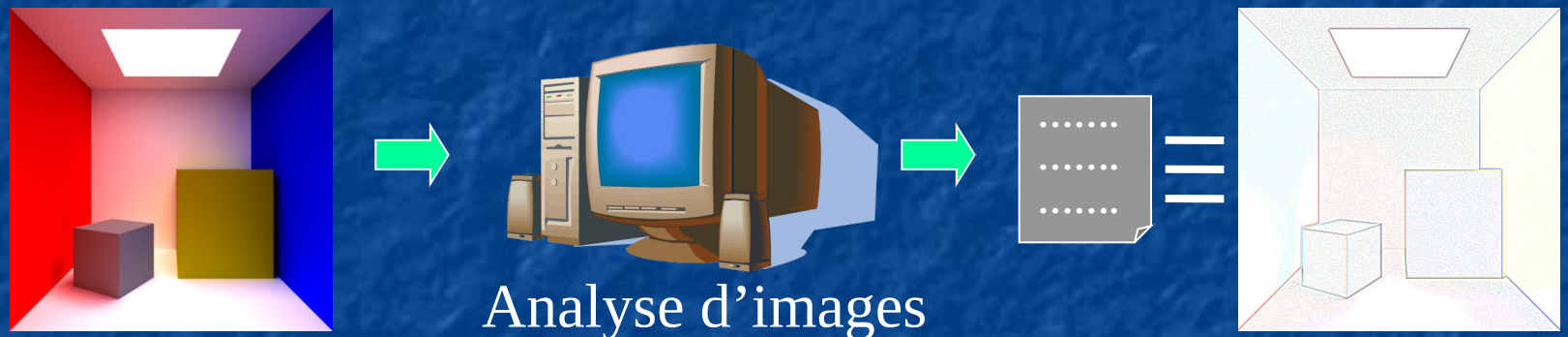
Plan du cours

1. Introduction
2. Modélisation d’objets 3D
3. Modèle d'éclairage local
4. Rendu temps réel
5. Introduction à Three.js

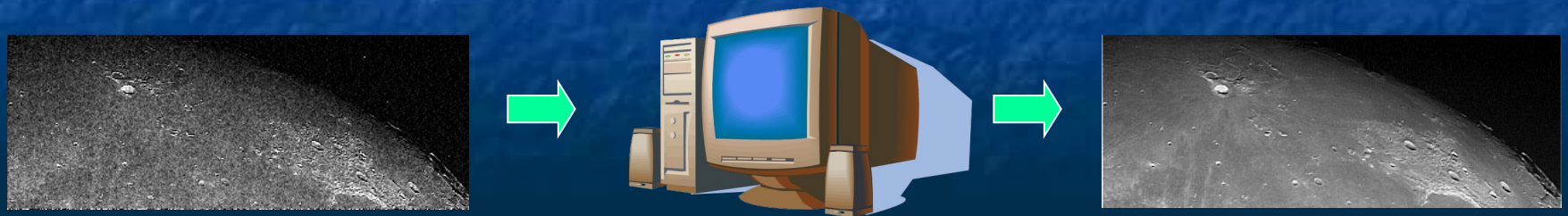
Définitions



Synthèse d'images



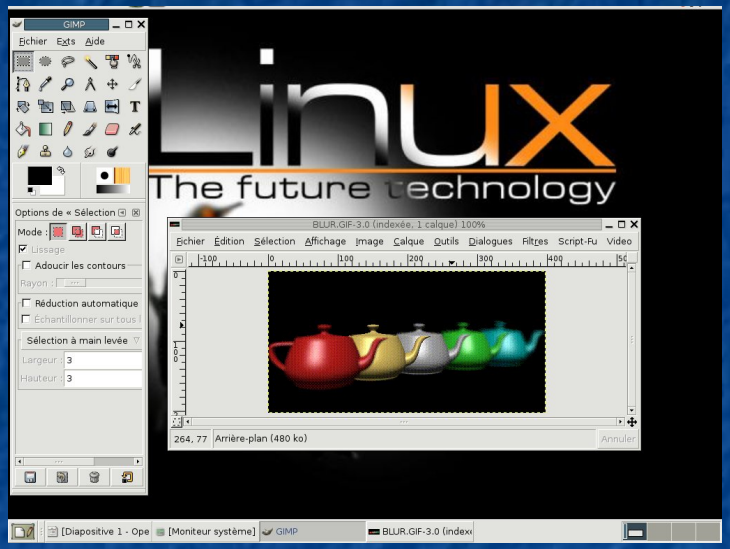
Analyse d'images

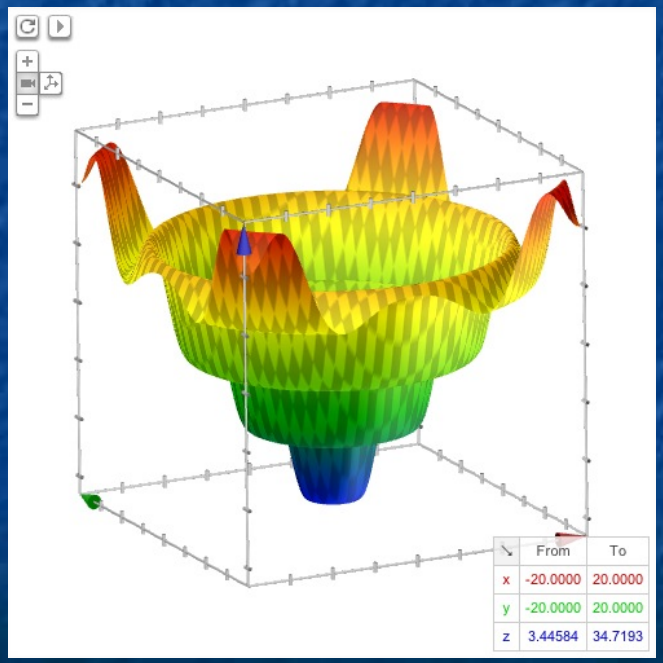


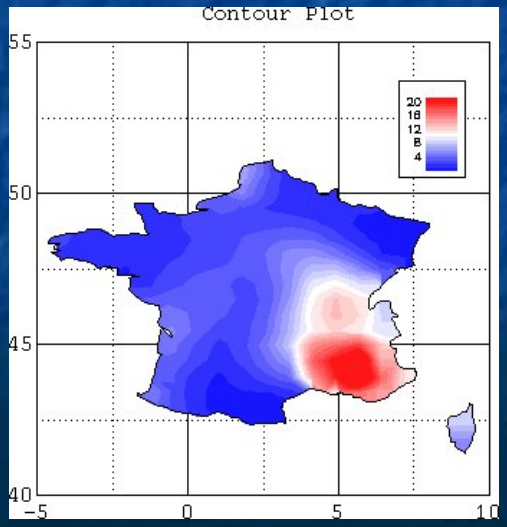
Traitement d'images

Applications de la synthèse d'images

- **Interfaces utilisateur**
- **Production de graphiques**

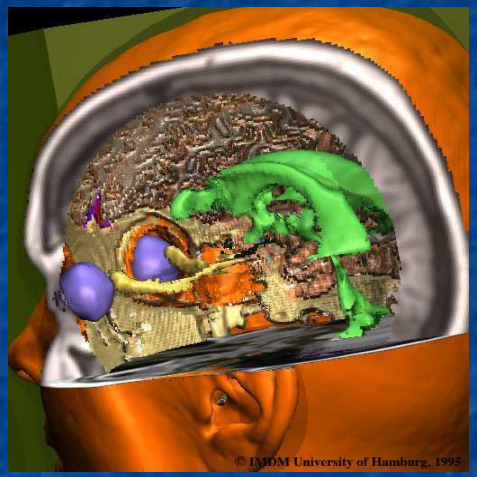


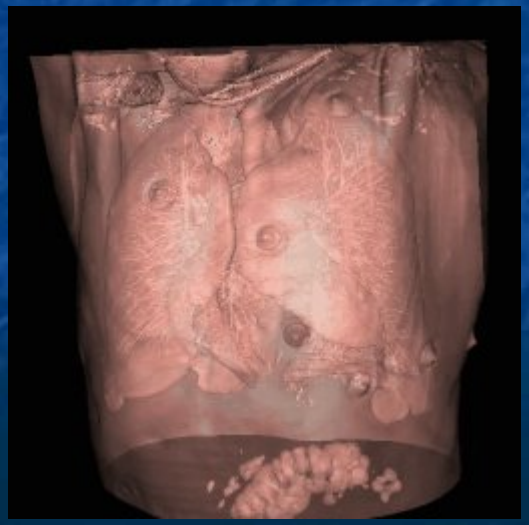




Applications de la synthèse d'images

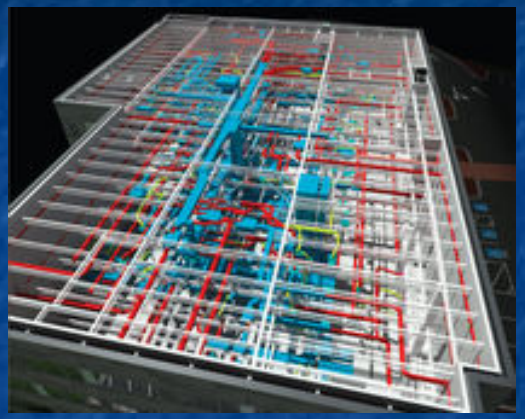
- Interfaces utilisateur
- Production de graphiques
- **Imagerie médicale**



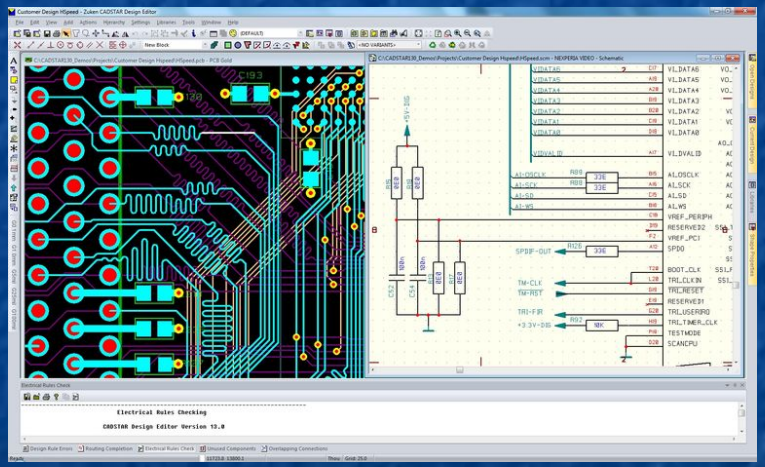


Applications de la synthèse d'images

- Interfaces utilisateur
- Production de graphiques
- Imagerie médicale
- **CAO**



Source : www.directindustry.fr - Autocad



Source : www.usinenouvelle.com - Zuken

Source : chromelight.fr

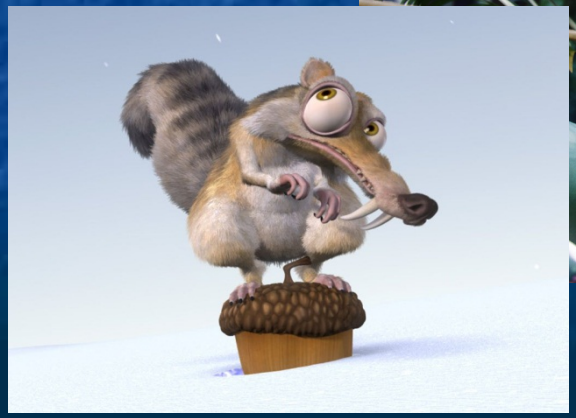
Applications de la synthèse d'images

- Interfaces utilisateur
- Production de graphiques
- Imagerie médicale
- CAO
- **Jeux & Vidéo**

Assassin's Creed Odyssey (c) Ubi Soft

Avatar 2 © 20th century fox

Mario Kart (c) Nintendo



Ice Age © Blue Sky Studios

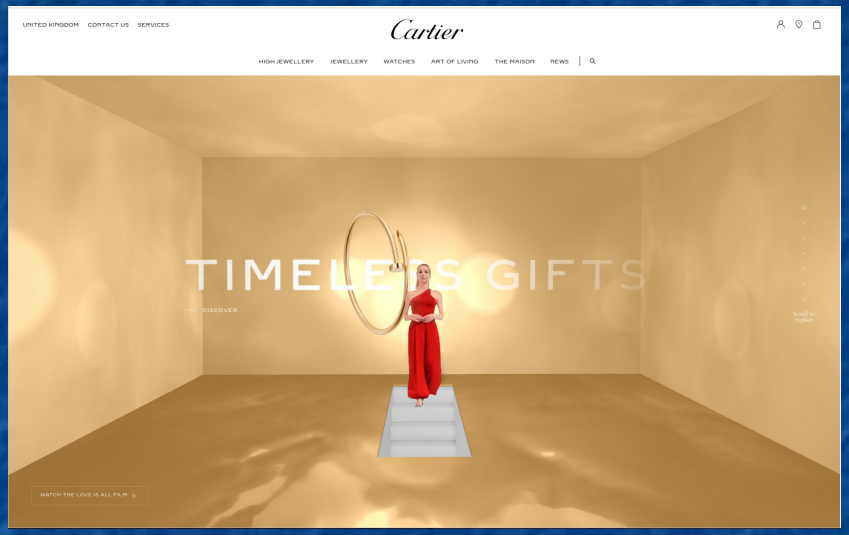
Applications de la synthèse d'images

- Interfaces utilisateur
- Production de graphiques
- Imagerie médicale
- CAO
- Jeux & Vidéo
- **Simulation / Réalité virtuelle**

Par Ethan Arnold, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=8864535

Applications de la synthèse d'images

- Interfaces utilisateur
- Production de graphiques
- Imagerie médicale
- CAO
- Jeux & Vidéo
- Simulation / Réalité virtuelle
- **Enrichissement sites web**

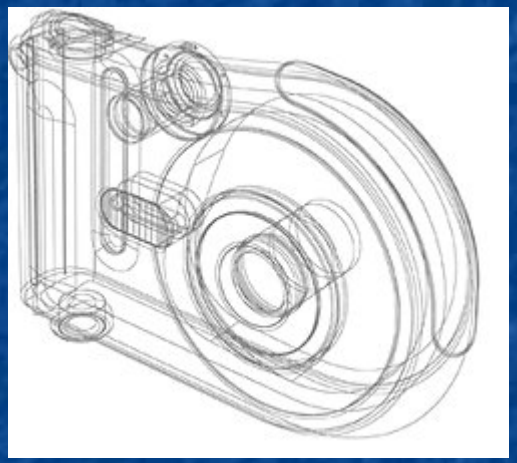


https://www.cartier.com/en-gb/love-is-all

https://sougen.co/

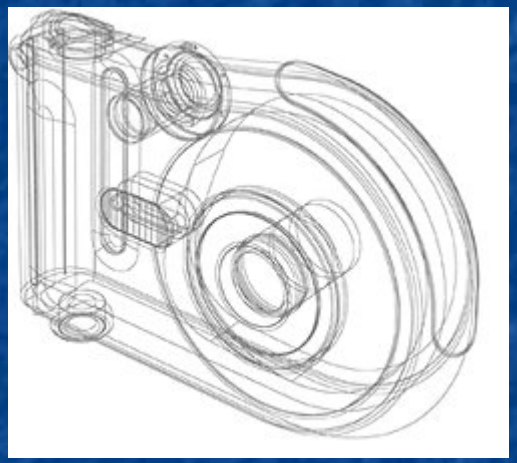
Notion de rendu

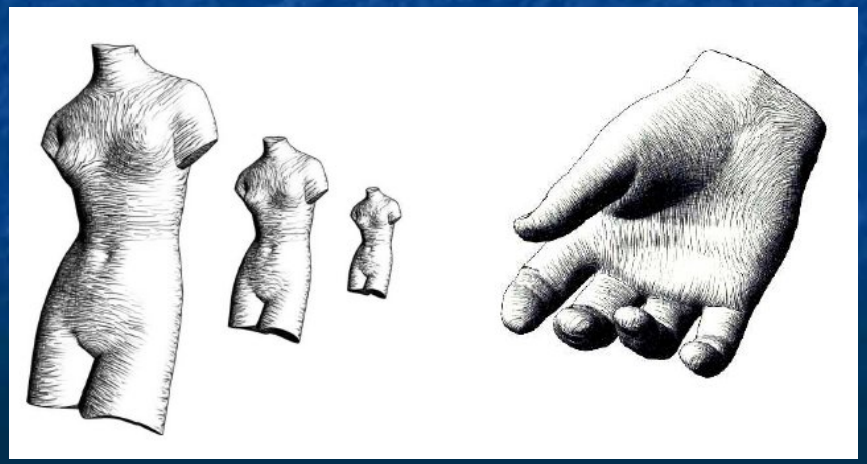
- Définition
  - Aspect final de l'image présentée à l'utilisateur
- Par extension :
  - Algorithme de calcul effectif de l'image finale
- Nombreux types de rendu
  - fil de fer



Notion de rendu

- Définition
  - phase finale de tout logiciel de synthèse d'images
  - permet le calcul effectif de l'image finale
- Nombreux types de rendu
  - Fil de fer
  - Expressif





Notion de rendu

- Définition
  - phase finale de tout logiciel de synthèse d'images
  - permet le calcul effectif de l'image finale
- Nombreux types de rendu
  - Fil de fer
  - Expressif
  - Réaliste
  - Etc …

Le choix dépend de l'application visée

Notion de rendu

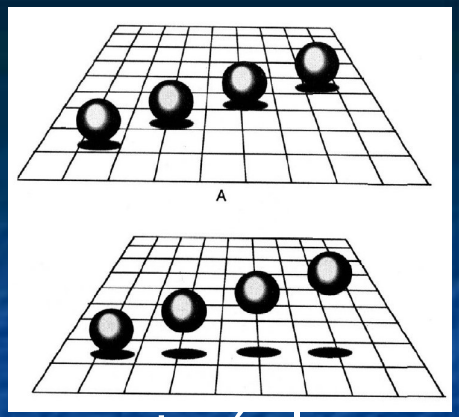
Deux grandes catégories d'algorithmes

- Rendu temps réel
  - Chaque image est calculée en moins de 1/25e de seconde
  - Utilisé pour les applications interactives (jeux vidéo, simulateurs, etc.)
  - Simplifications importantes
- Rendu différé
  - Pas de limite au temps de calcul d'une image
  - Utilisé pour les applications nécessitant précision et qualité (cinéma, architecture, etc)
  - Prise en compte de très nombreux phénomènes

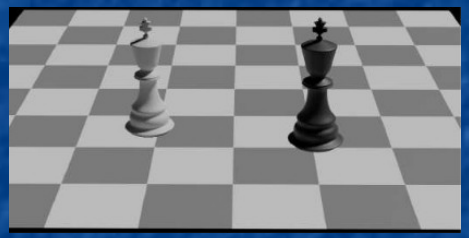
Dans les deux cas :

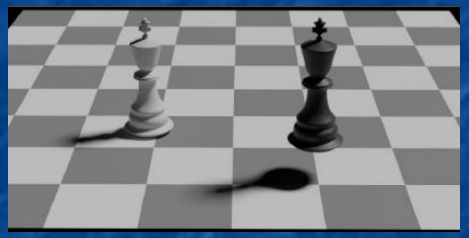
- importance de restituer certains phénomènes naturels

Les ombres

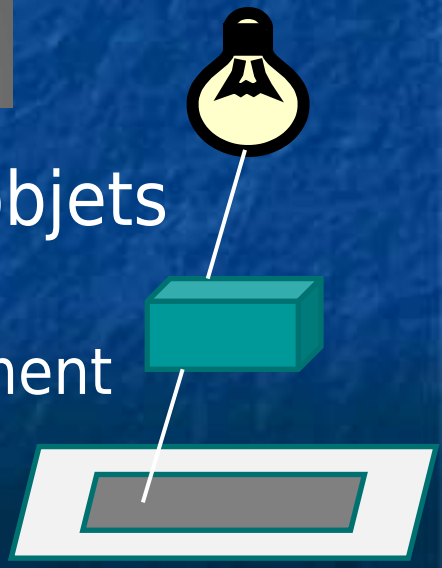


- Accroissent le réalisme de l'image
  - Présence systématique dans notre environnement réel
  - Important dans la perception :
    - Des distances, des volumes, des positions, de la géométrie

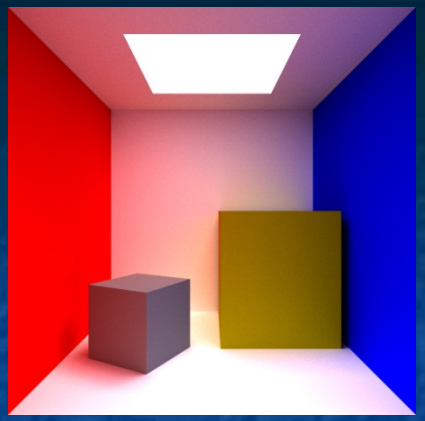




- Prise en compte des occlusions entre objets et sources
  - Rendu temps réel : difficiles à faire rapidement
  - Rendu différé : possibilité de calcul précis



Les reflets

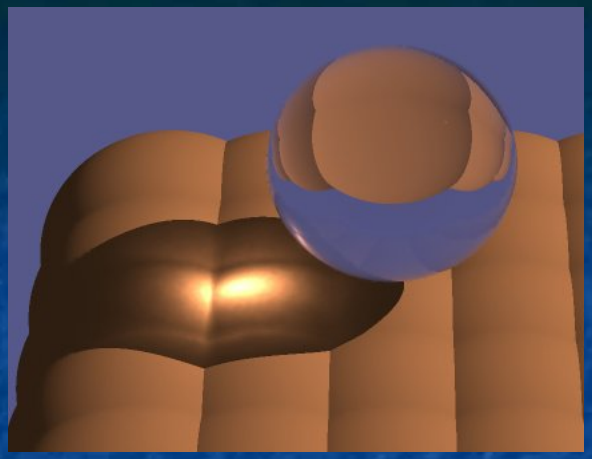


- Contribuent au réalisme des images
  - Éclairage indirect
    - Les zones non directement éclairées par les sources de lumière (soleil, néon, bougie, …) ne sont pas « noires »
  - Mélange de couleurs par réflexion
    - « color bleeding »
  - Qualité du rendu des matériaux
    - Modèle de réflexion précis => matériaux naturels

- Nécessitent de :
  - Modéliser les phénomènes de réflexion
  - De prendre en compte les interactions entre objets

- Rendu temps réel : difficile à faire rapidement

Les transparences

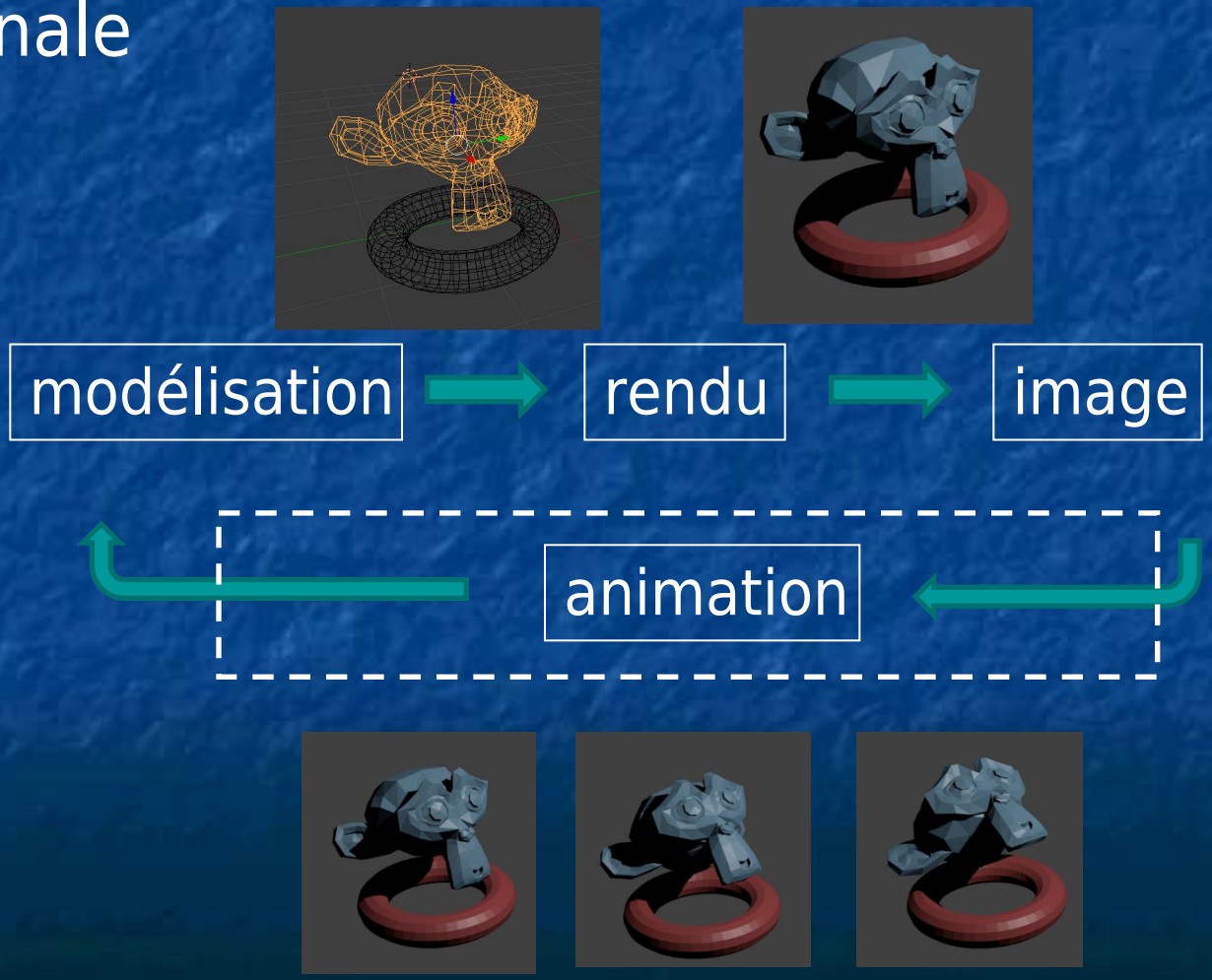


- propagation de la lumière au travers des matériaux
  - Lois de la réfraction
  - Apparition d'effets visuels plus ou moins complexes
    - Déformations
    - « Caustics »
    - Mirages, arcs-en-ciel, …

- Rendu temps réel : difficile à faire rapidement

La chaîne de synthèse

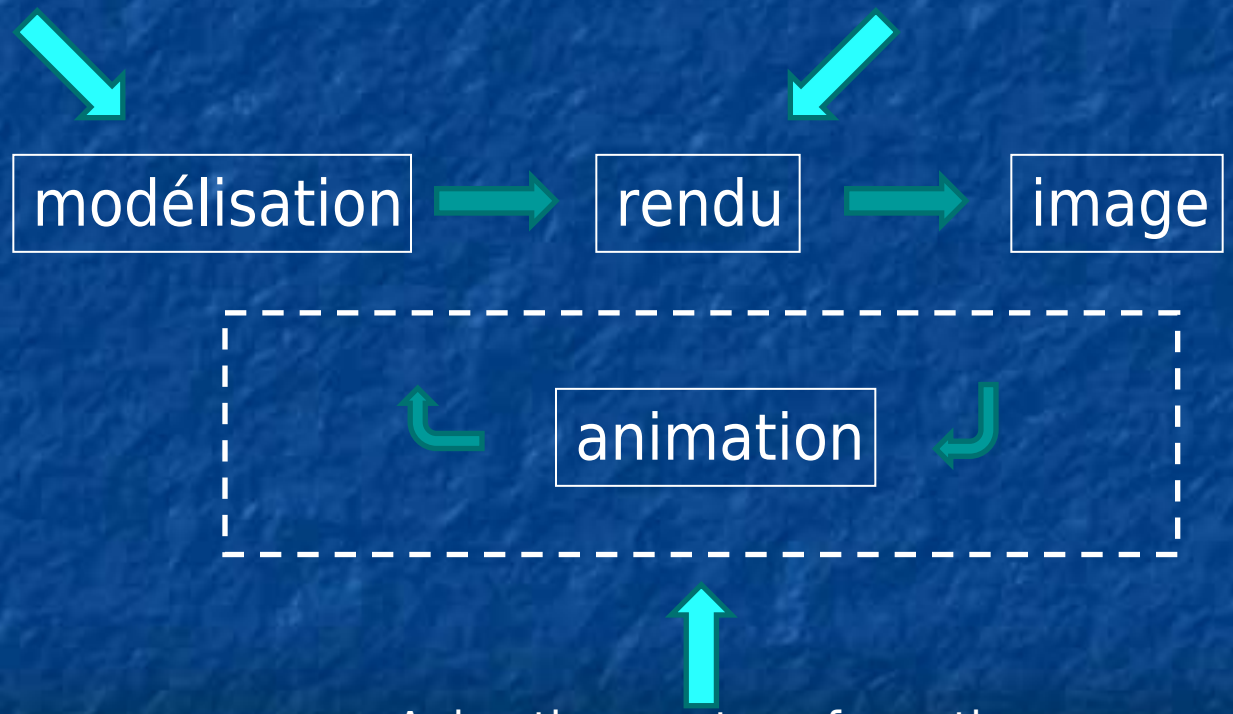
- Plusieurs étapes avant d'aboutir à l'image finale



La chaîne de synthèse (suite)

Aperçu de quelques techniques de base

Étude de deux « méthodes », l'une en rendu temps réel, l'autre en rendu différé



Animation par transformations géométrique classiques (TP)

Plan du cours

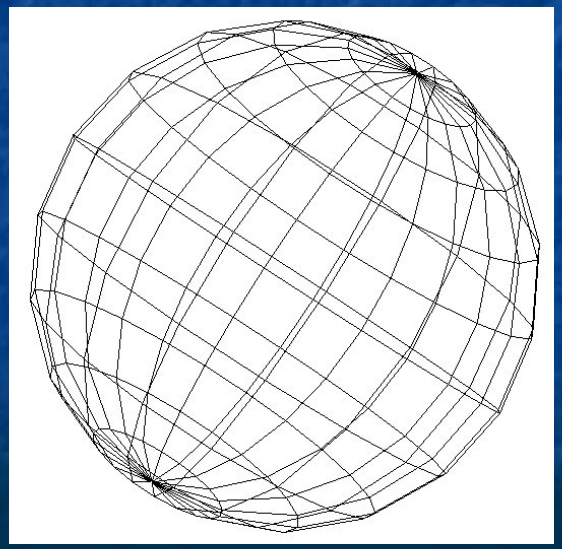
1. Introduction
2. Modélisation d’objets 3D
3. Modèle d'éclairage local
4. Rendu temps réel
5. Introduction à Three.js

Objectifs

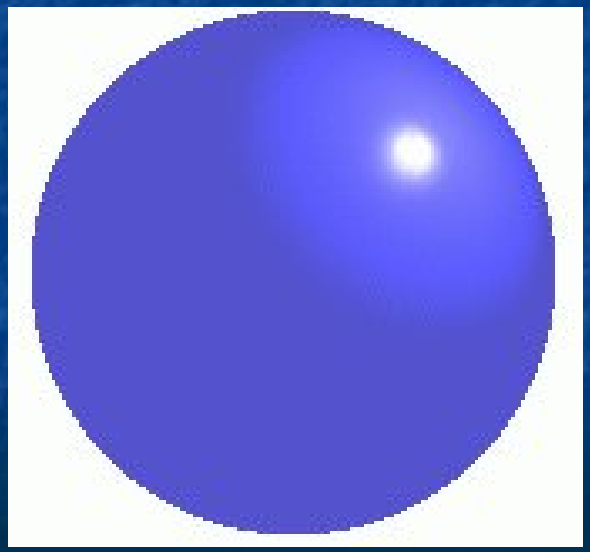
- Spécification de :
  - La géométrie des objets (quand elle existe)
  - De la position des objets
  - Des matériaux
    - Couleurs, Caractéristiques de réflexion, ...
- Étape initiale à tout algorithme de S.I.
- Pas de solution universelle :
    - nombreuses techniques différentes
    - dépendance à différents paramètres :
      - complexité de l'objet, animation, déformation
      - algorithme de rendu
      - périphérique d'entrée

Modèles polygonaux

- La surface d'un objet est décrite par un ensemble de polygones plans
  - Notion de facettes



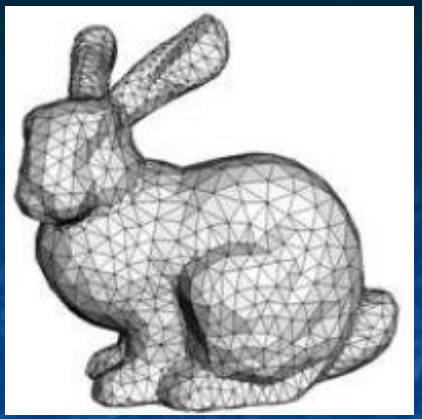
Algorithme de rendu

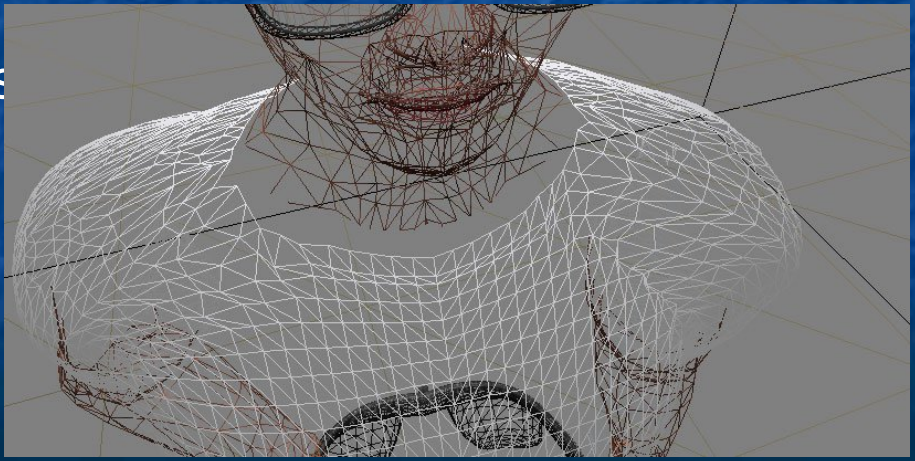


modèle

rendu

- Surfaces planes minimales :
  - triangles
- Modèle le plus répandu
  - Toute surface courbe peut être approchée par un nombre plus ou moins grand de facettes
- Utilisé par :
  - Les APIs graphiques
    - OpenGL, Direct3D
  - Les cartes 3D
    - Ne traitent que des triangles
  - Les modeleurs
    - Facettisation des objets

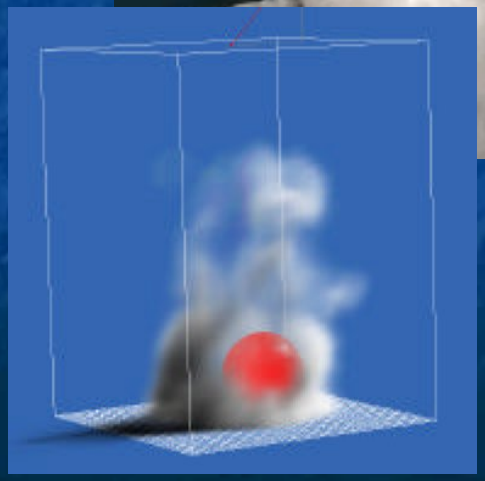




http://community.secondlife.com

Mais …

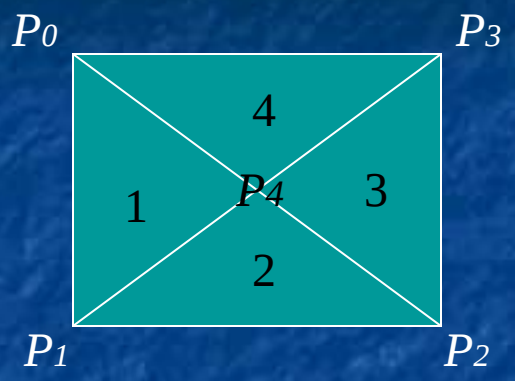
- Restent une approximation
  - Augmenter le nombre de facettes pour approximer correctement les rayons de courbure
- Mal adaptées aux objets complexes
  - Grand nombre de facettes
- Existence d’objets non surfaciques
  - Gaz, fumées, nuages, …

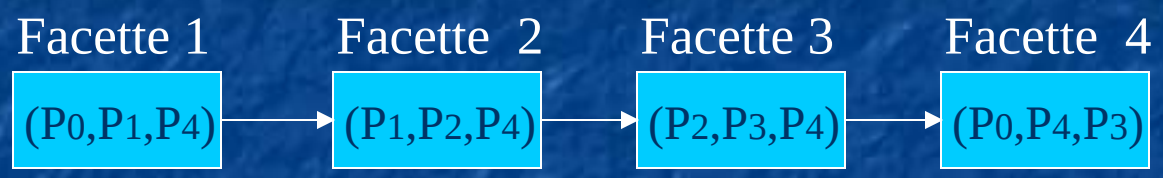


Représentation en mémoire

- Plusieurs représentations possibles selon les informations auxquelles on souhaite avoir accès
- Représentation de base :
  - une liste des facettes composant l'objet
  - chaque facette contient les coordonnées des points qui la composent

Exemple





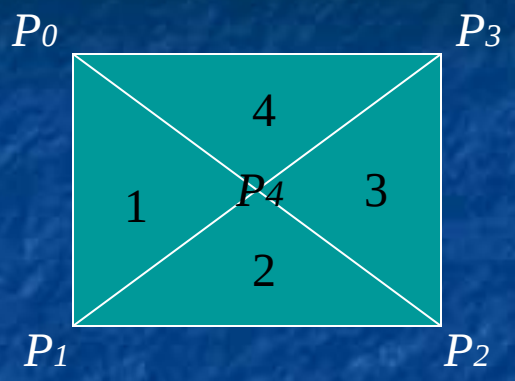

```
struct {
  float x, y ,z;
} point3D;

struct{
  point3D sommet[3];
  …
} facette;
```

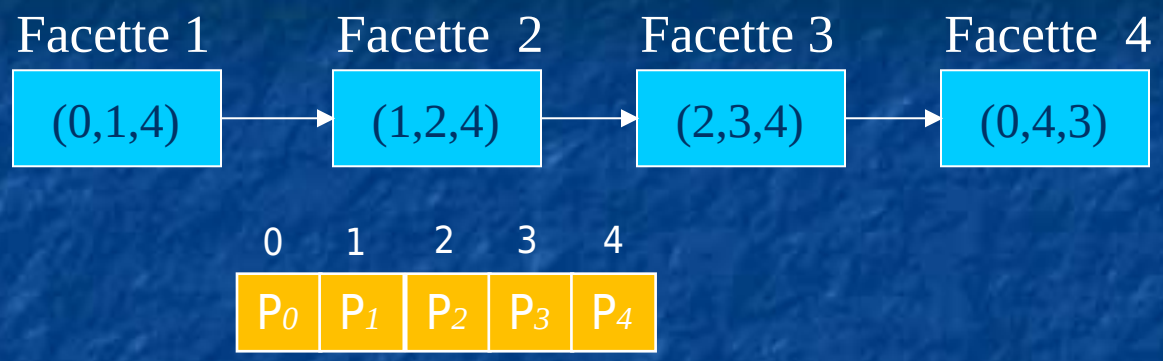
- Problèmes :
  - redondance d'informations : les points communs à plusieurs facettes sont répétés
  - perte de la notion de voisinage

Représentation classique

- Utilisée dans les différentes variantes de OPenGL et dans Three.js
  - Une liste de points (sommets)
  - Une liste de facettes
    - Chaque facette contient l'indice de ses sommets

Exemple





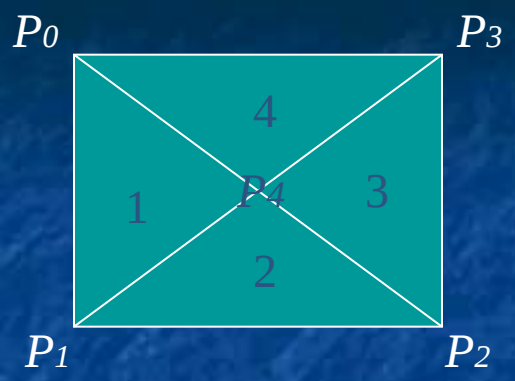

```
struct {
  float x, y ,z;
} point3D;

Point3D tabSommet[N];

struct{
  int sommet[3];
  …
} facette;
```

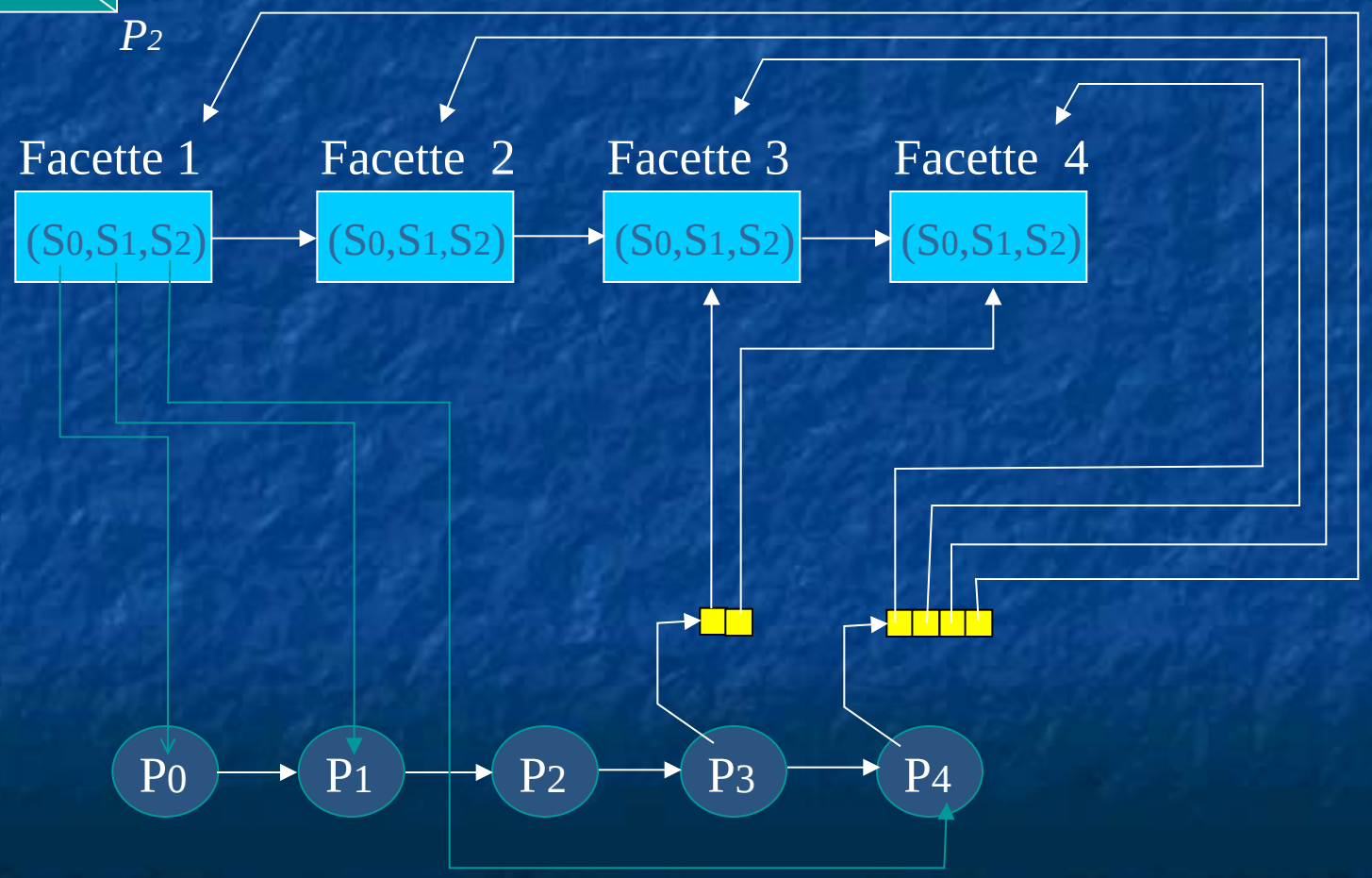
- Avantage :
  - Les sommets ne sont représentés qu'une seule fois
- Inconvénient :
  - perte de la notion de voisinage

Représentations plus complexes

- Gestion de 2 listes :
  - une liste de sommets et une liste de facettes
  - gestion des liens inter-liste
    - les facettes ont des pointeurs vers leurs sommets (indice, adresse)
    - les sommets ont des pointeurs vers leurs facettes (indice, adresse)
- Avantage :
    - Points et facettes ne sont représentés qu’une seule fois
- Inconvénients :
    - Gestion plus complexe

Exemple


P0
P3
4
1
P4
3
2
P1
P2


Facette 1
Facette 2
Facette 3
Facette 4
(S0,S1,S2)
(S0,S1,S2)
(S0,S1,S2)
(S0,S1,S2)
P0
P1
P2
P3
P4

Structures de données

```
struct{
  point3D *sommet[3];
  …
} facette;
```

Nombre de sommets connu

```
struct {
  float x, y ,z;
  maillon *debListeFacette;
} point3D;
```

Nombre de facettes auxquelles le sommet appartient inconnu.

Liste chaînée de « facettes »

```
struct{
  facette *fac;
  maillon *suiv;
} maillon;
```

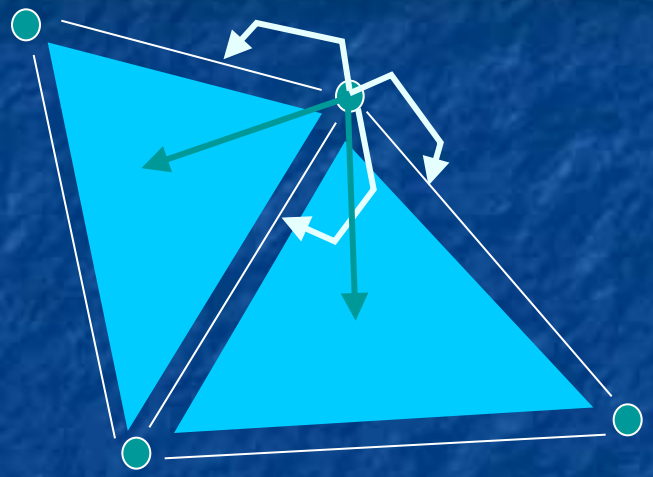
Pointeur vers la facette

Pointeur vers le maillon suivant de la liste

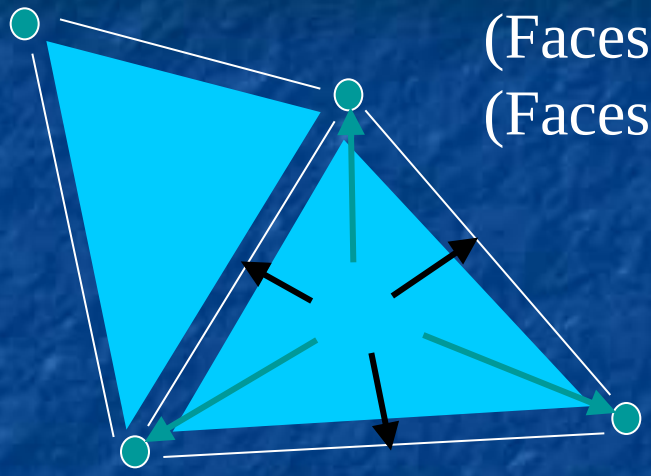
Représentations plus complexes

- Gestion de 3 listes
  - une liste de sommets
  - une liste de facettes
  - une liste d'arêtes
  - gestion des liens inter-liste

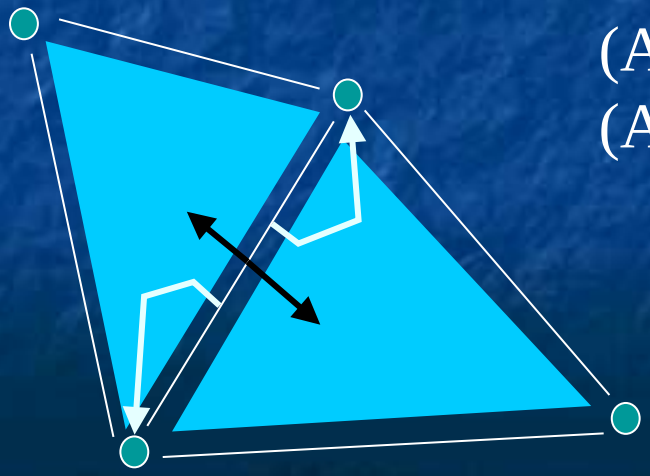
Exemple : Les winged-edge



(Sommet, faces)
(Sommet, arêtes)

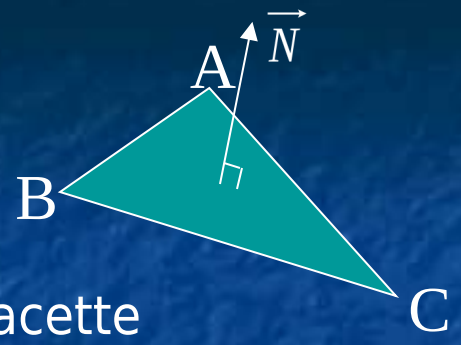


(Faces, sommets)
(Faces, arêtes)

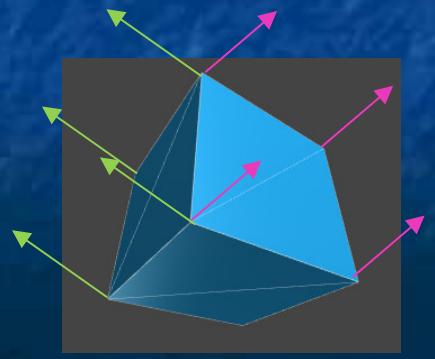


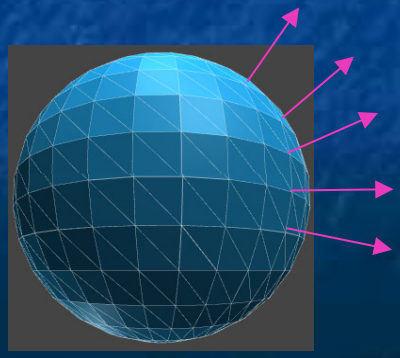
(Arêtes, faces)
(Arêtes, sommets)

Les normales (1)

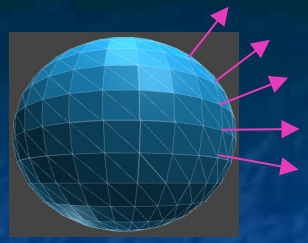


- Normale à une facette (plane) :
  - Vecteur perpendiculaire au plan de la facette
- Utilité :
  - Calculs de visibilité
  - Calculs d'éclairage
- En pratique : une normale pour chaque sommet
  - Surface plane : normale identique en chaque point
  - Surface courbe : vecteur perpendiculaire à la surface



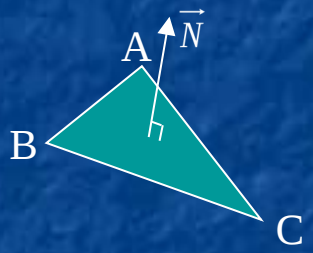


Les normales (2)

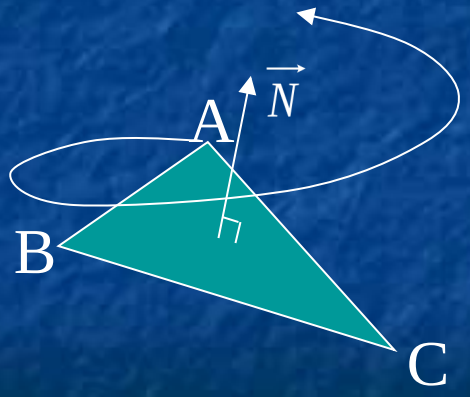


- Propriété à assurer :
  - La normale doit toujours être orientée vers l'extérieur d'un objet facettisé

- Calcul d'une normale : $\vec{N} = \overrightarrow{AB} \wedge \overrightarrow{AC}$

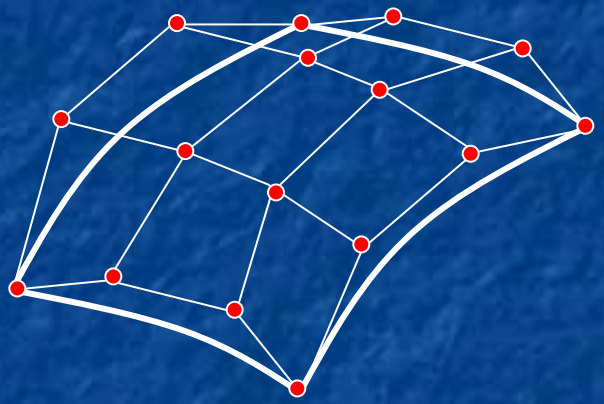


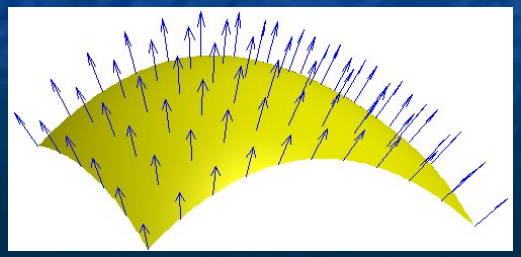
- Remarque : calcul automatique
  - Lors de la modélisation, les points A, B et C doivent être spécifiés dans l'ordre direct (règle du tire-bouchon ...)



Surfaces paramétrées

- Surfaces paramétrées par des polynômes en s et t :
  - $P(x,y,z) = (x(s,t), y(s,t), z(s,t))$
- Degré plus ou moins élevé selon la complexité de la surface
- Avantages
  - Meilleure représentation de la surface
  - Contrôle interactif de la forme de la surface (points de contrôle)
- Inconvénients
  - Rendu plus difficile
  - Souvent facettisées





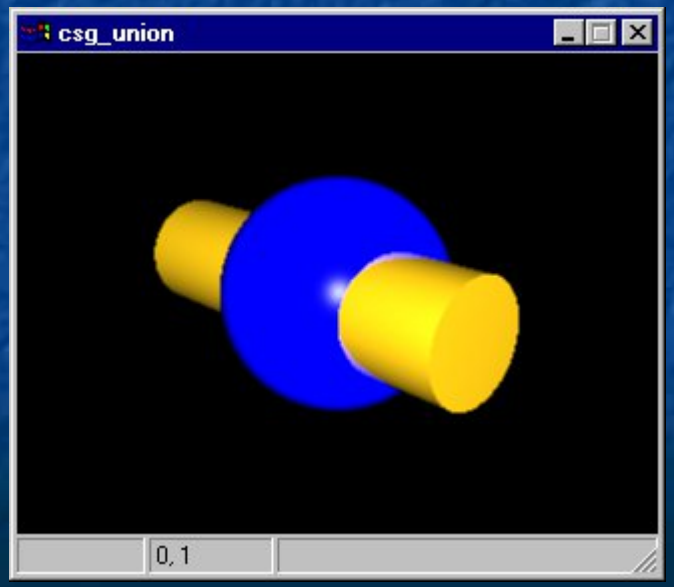
Source: wikipédia

Représentation CSG

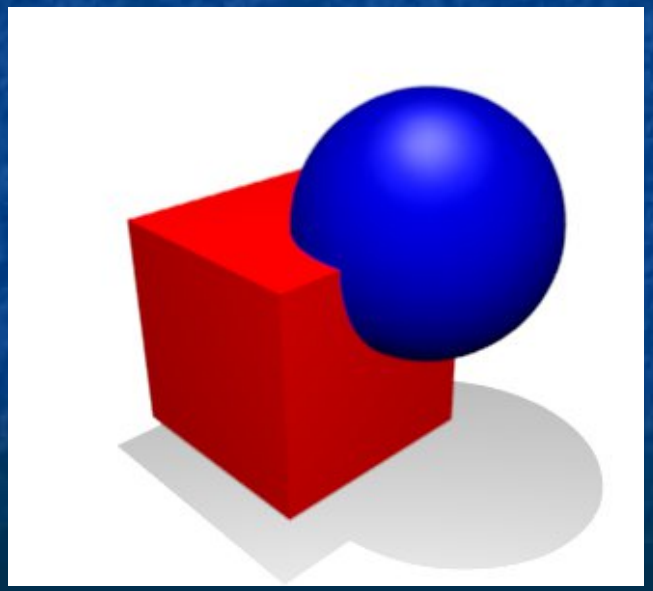
- CSG = Constructive Solid Geometry
- Principe :
  - objets simples = primitives volumiques simples
    - sphères, cubes, cônes, cylindres, ...
  - Objets complexes = ensembles d'objets simples
  - définition de relations entre les constituants d'un objet complexe :
    - opérateurs booléens

Opérateurs booléens

- L'union (∪) :
  - un objet complexe est constitué par le volume résultant de la réunion de 2 objets plus simples
  - exemples :



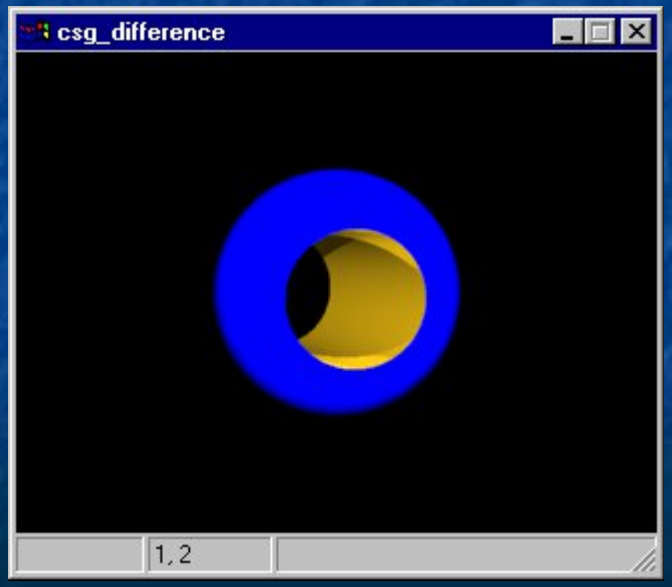
Sphere ∪ Cylindre



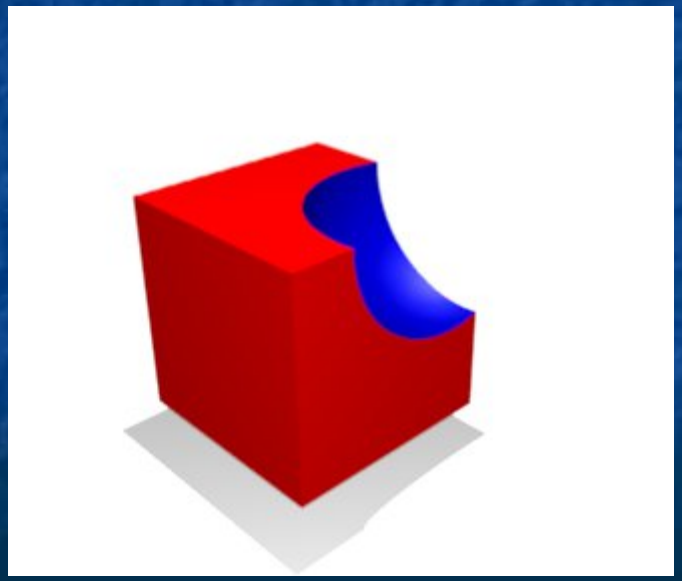
Cube ∪ Sphere

Opérateurs booléens

- La différence (-)
  - un objet complexe est constitué par le volume obtenu en soustrayant le second volume du premier
  - exemples :



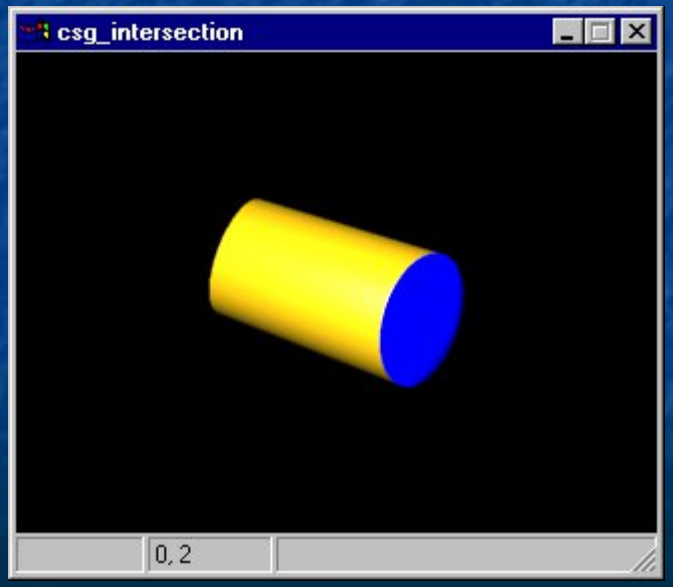
Sphère - Cylindre



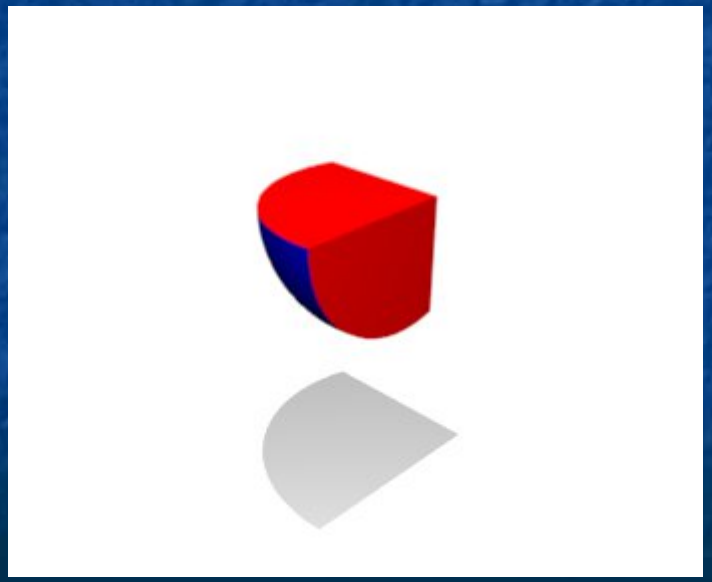
Cube - Sphère

Opérateurs booléens

- L'intersection (∩)
  - un objet complexe est constitué par le volume résultant de l'intersection de 2 objets plus simples
  - Exemples :



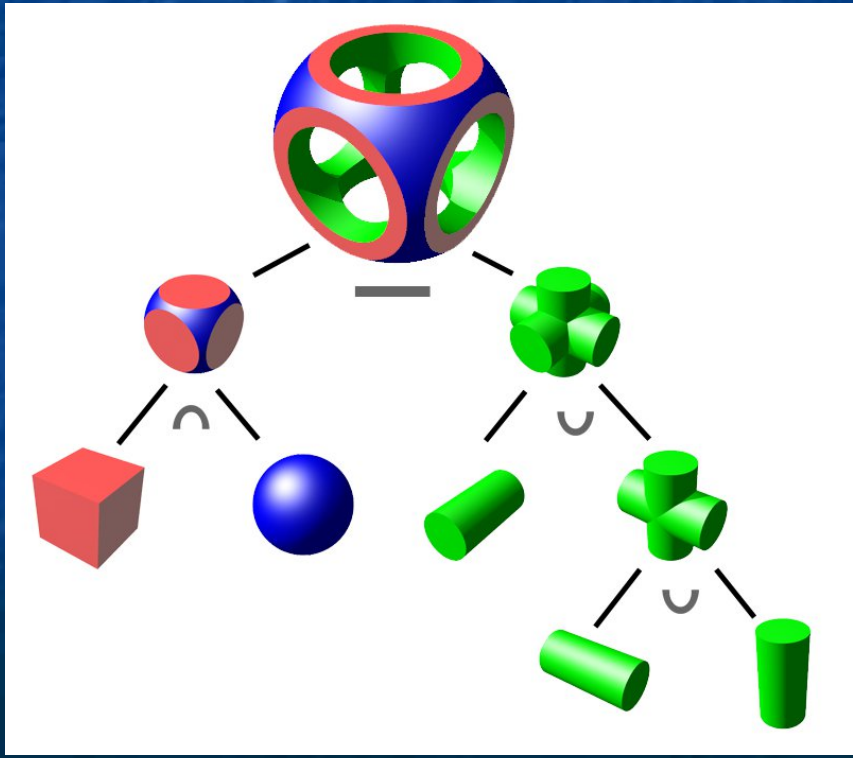
Sphère ∩ Cylindre



Cube ∩ Sphère

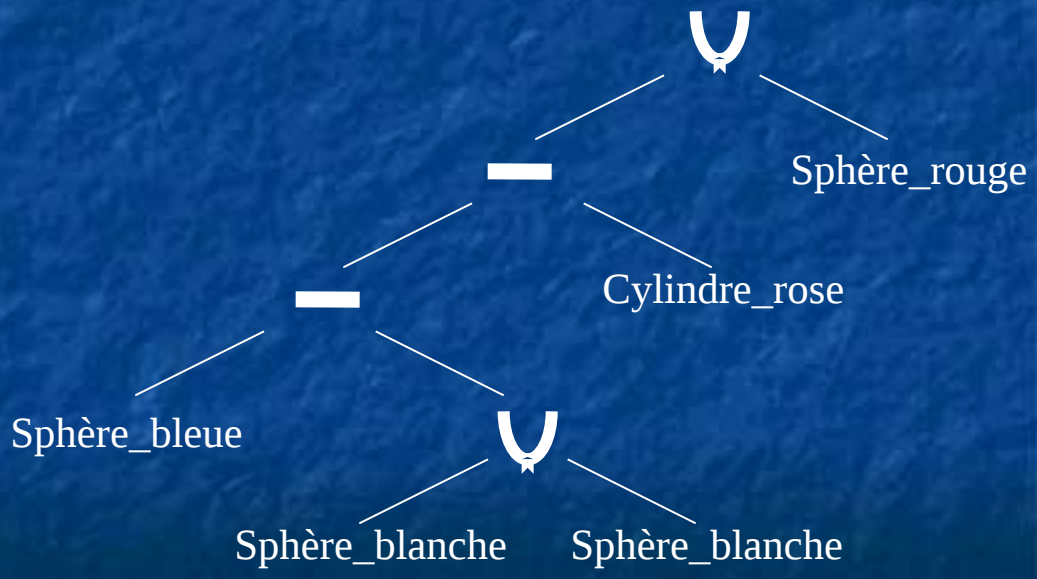
Généralisation

- Les opérandes d'un opérateur booléen peuvent eux-mêmes être issus d'une opération CSG
  - Arbre CSG



Exemple

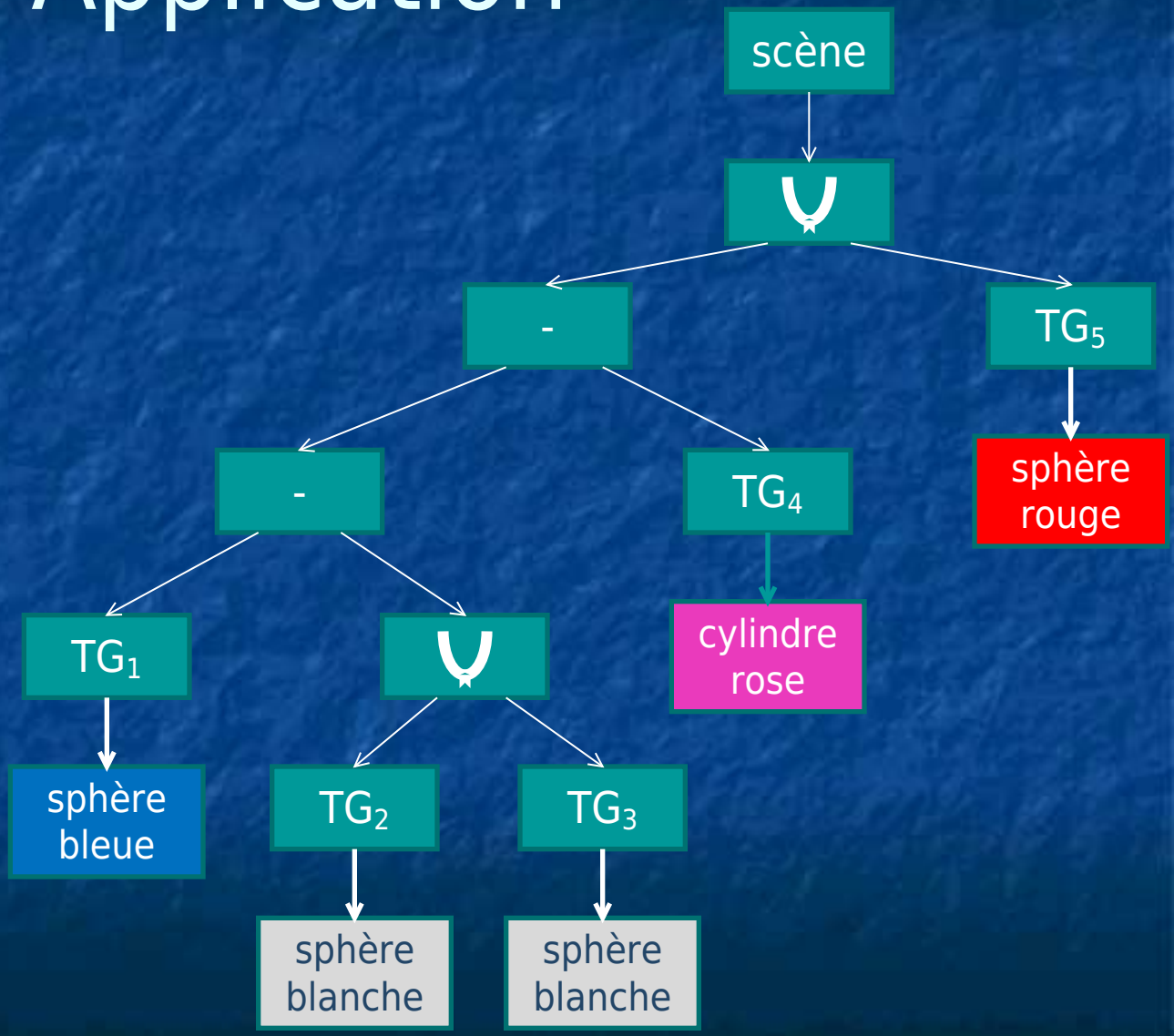
- Comment obtenir cet objet ?



Remarque

- Un nœud d’un arbre CSG peut contenir :
  - soit un opérateur booléen
  - soit une transformation géométrique à appliquer au sous arbre
    - translation
    - rotation
    - mise à l'échelle
    - …

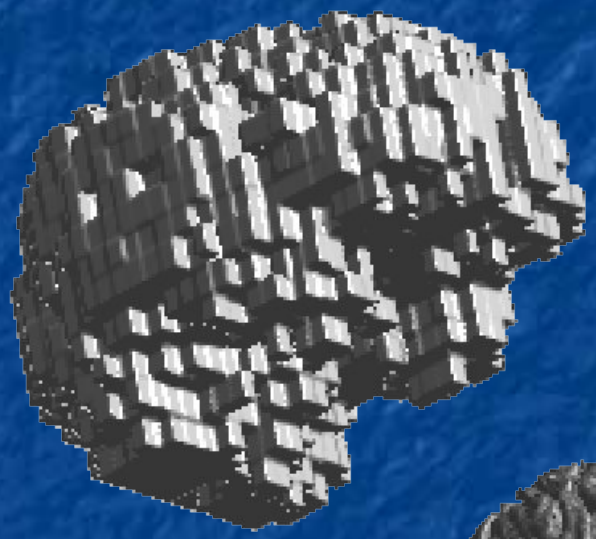
Application

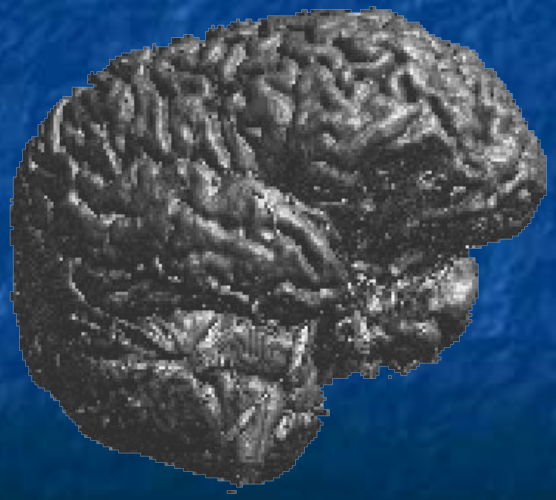


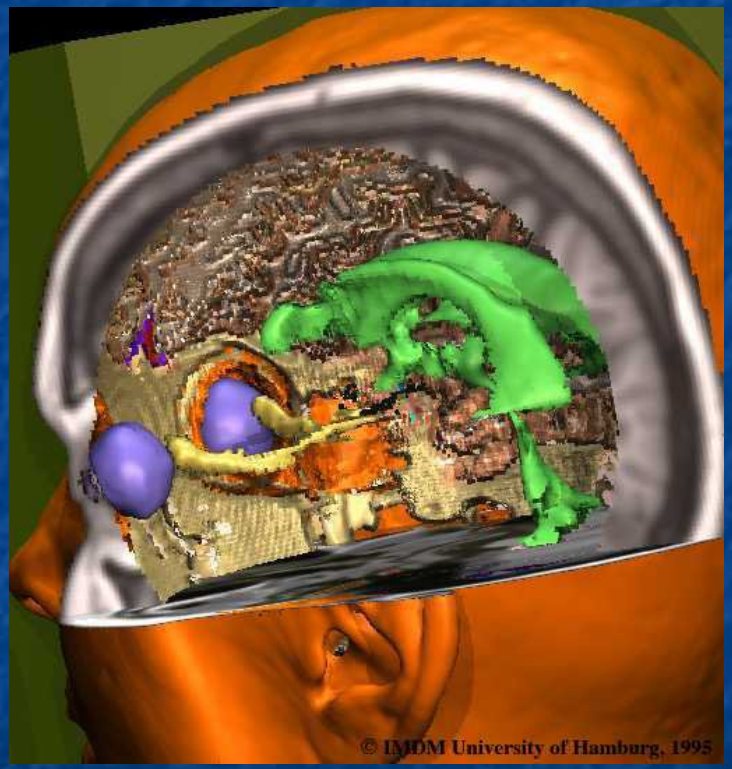
Le découpage spatial

- Principe :
  - l'espace modélisé est découpé en « cubes » élémentaires (voxels -*Volume Elements*)
  - chaque cube contient une indication sur la « matière » qu'il contient
  - l'objet final est celui qui est constitué de tous les voxels de même valeur
- très utilisé en imagerie médicale :
  - les scanners fournissent des « tranches » de pixels
  - un pixel = une densité de matière

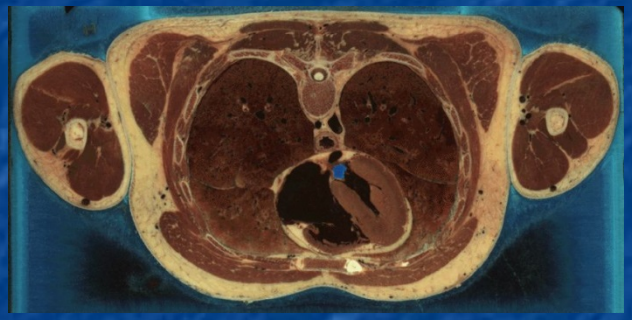
Exemples

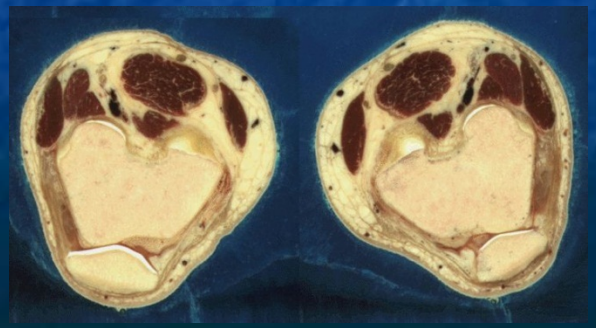






The visible human project

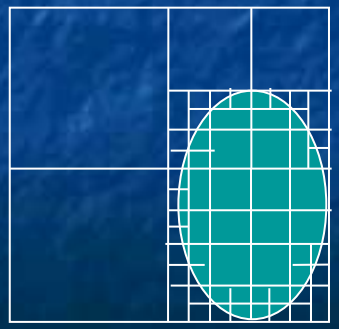




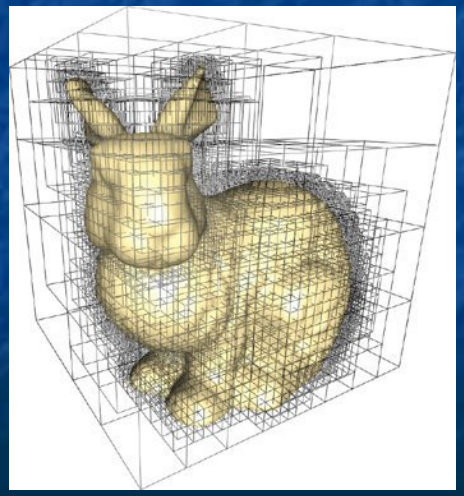
Sections de 1 mn pour le corps masculin, 1/3 mm pour le corps féminin

Les octrees

- Découpage homogène
  - nombreux voxels vides
    - Coût mémoire important
- Découpage non homogène :
  - subdivision récursive de l'espace en 8 sous voxels de même taille
  - arrêt quand le voxel est vide ou totalement plein, quelle que soit sa taille

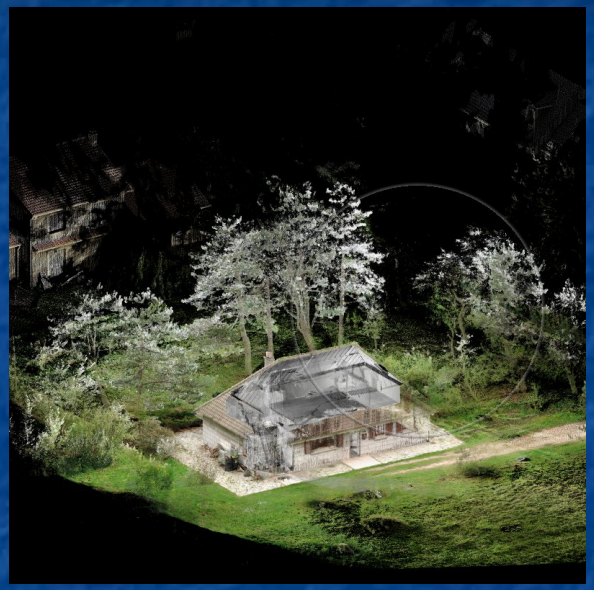


Ex en 2D (quadtree)



Ex 3D (octree)

Représentation par nuages de points



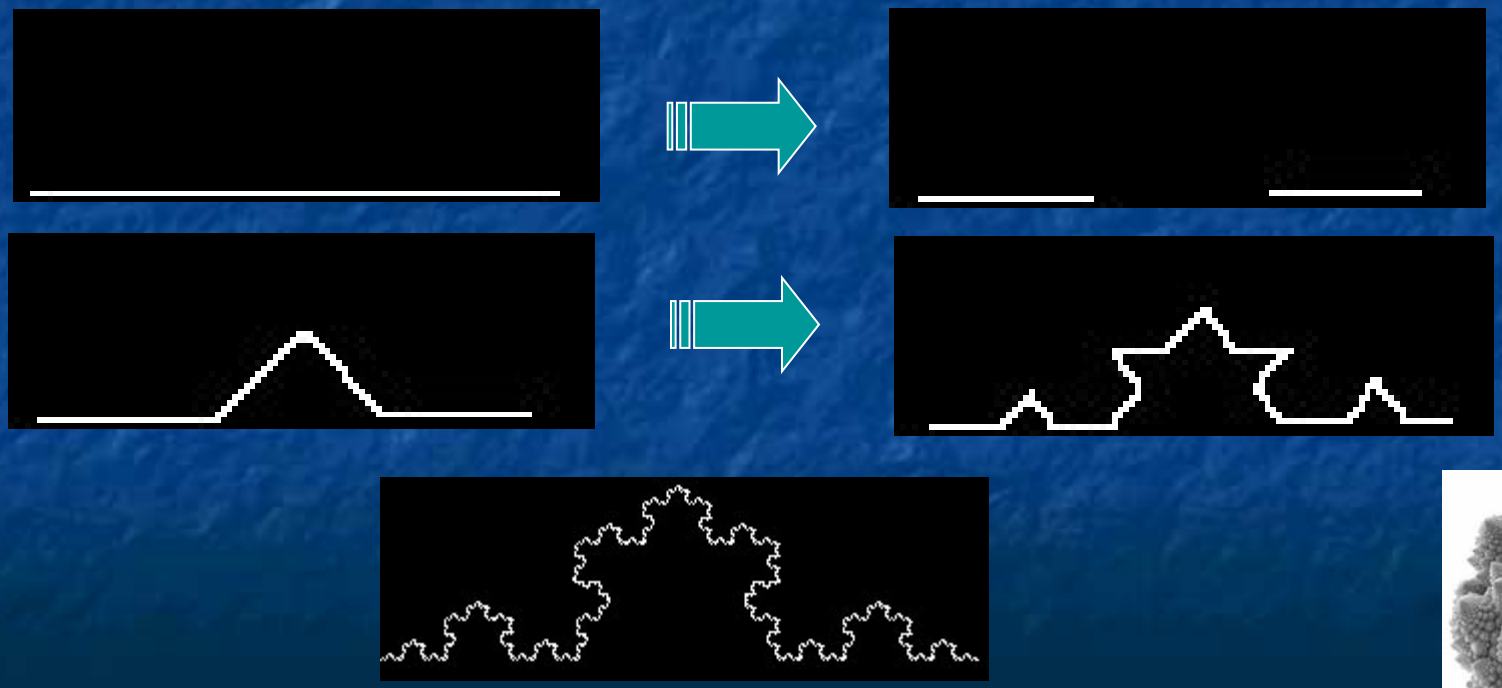
- Scanners 3D :
  - Fournissent des nuages de points
  - Informations diverses
    - Distances, couleurs
- Représentations mémoires
  - Facettisation
    - on reconstruit des facettes à partir des points
  - Ensemble de points
    - Necessite des algorithmes de visualisation spécifiques

Modélisation avancée

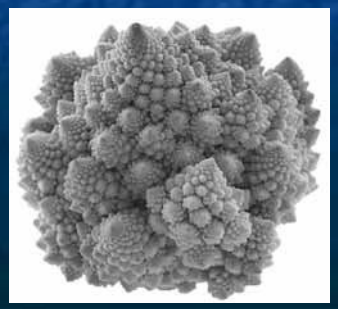
- Facettes mal adaptées à la modélisation de certains objets
  - arbres
  - Montagnes
- Facettes inadaptées à la modélisation de phénomènes naturels
  - fumées, gaz, nuages, etc ...

Les fractales

- Objet mathématique
  - dont les propriétés sont invariantes, quelle que soit l'échelle à laquelle il est regardé
  - Exemple : le flocon de Koch

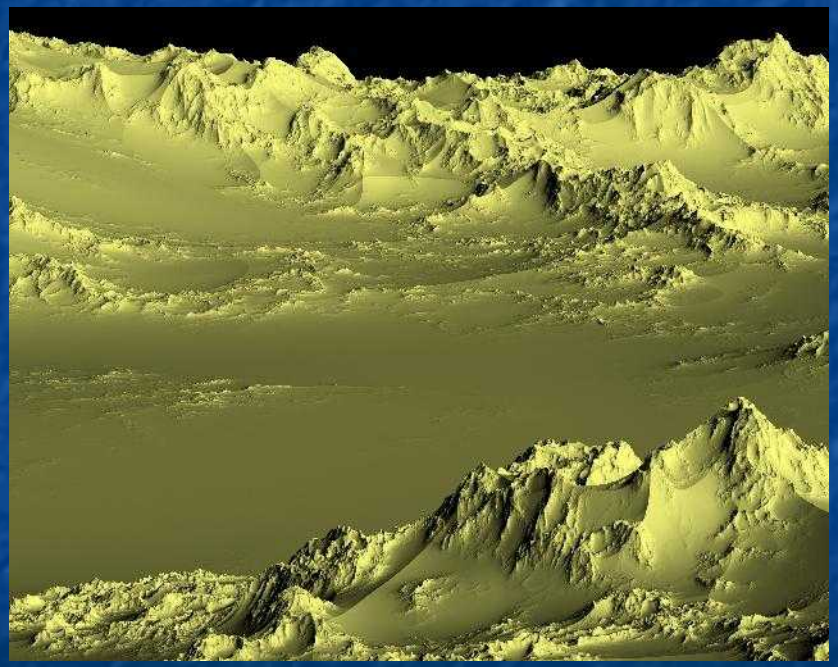


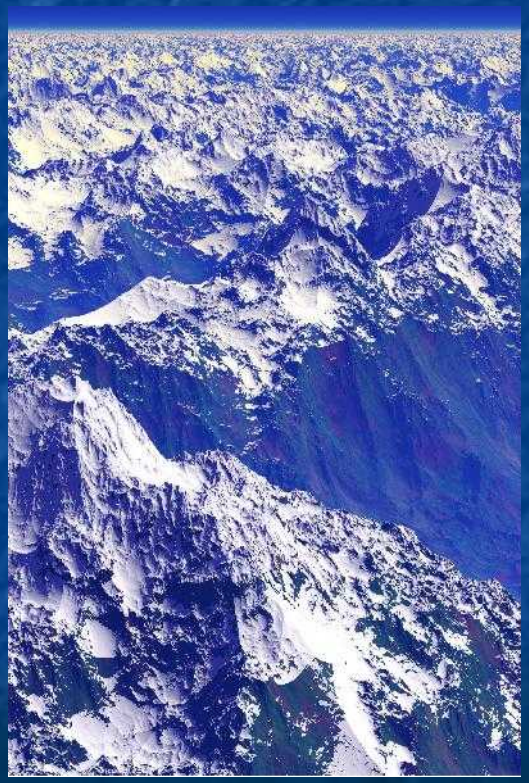
“Bien” adapté à la représentation d’objets naturels



Généralisation à la 3D

- Subdivision récursive d’un polygone de base
- Introduction de perturbations aléatoires sur la hauteur de chaque point



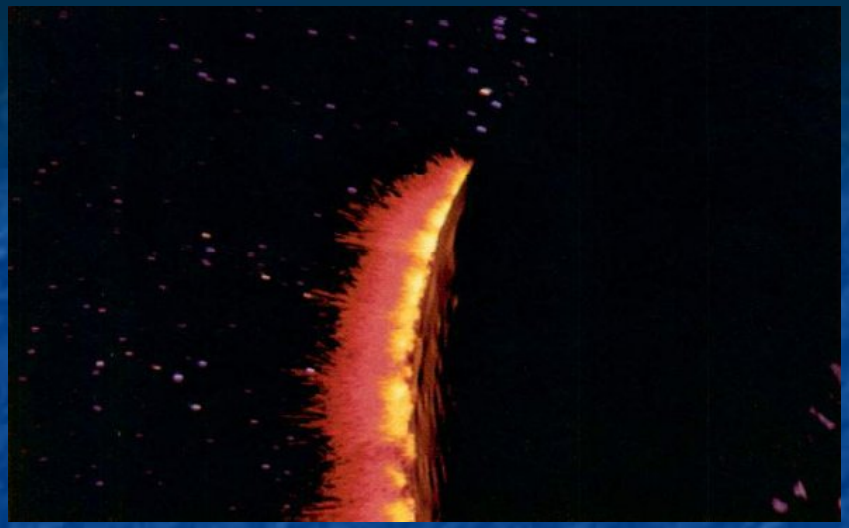


Mais rendu complexe du fait du nombre important de facettes obtenues

Les systèmes de particules

- Principe :
  - lancer des particules dont la trajectoire suit une certaine loi de distribution
  - chaque particule a une durée de vie et une trajectoire propre
  - à la mort d’une particule, celle-ci génère 0, 1 ou plusieurs nouvelles particules
  - l’objet est modélisé par l'ensemble des trajectoires suivies par les particules

Star Trek II - 1983

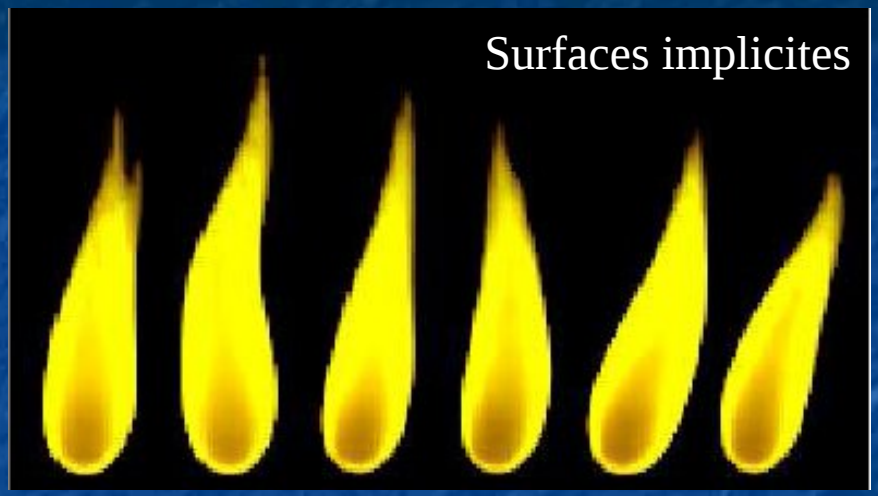


Feu d’artifice

Végétation

Quelques résultats

Un même phénomène – plusieurs modèles



Beaudouin et al - 2001

Wei et al - 2002

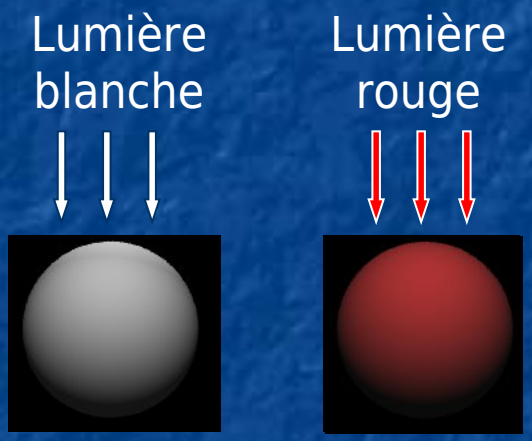
Bridault - 2004

Plan du cours

1. Introduction
2. Modélisation d’objets 3D
3. Modèle d'éclairage local
4. Rendu temps réel
5. Introduction à Three.js

Interactions lumineuses

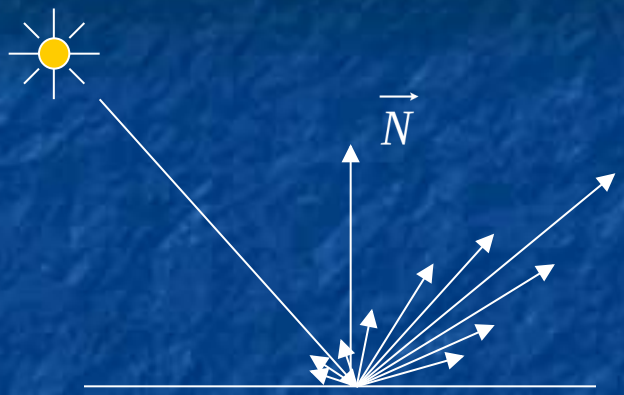
- Couleur observée dépend
  - Des sources de lumières
    - Distribution spectrale
    - Intensité
  - Des matériaux des objets
    - « couleur » de l'objet
    - Modes de réflexion
      - Mat (diffus)
      - Brillant (spéculaire)



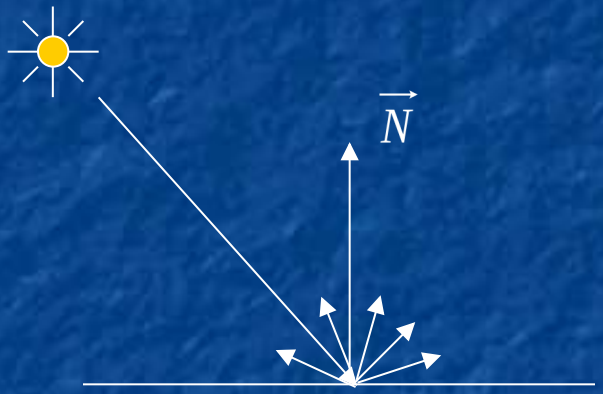
Sphères blanches

mat brillant

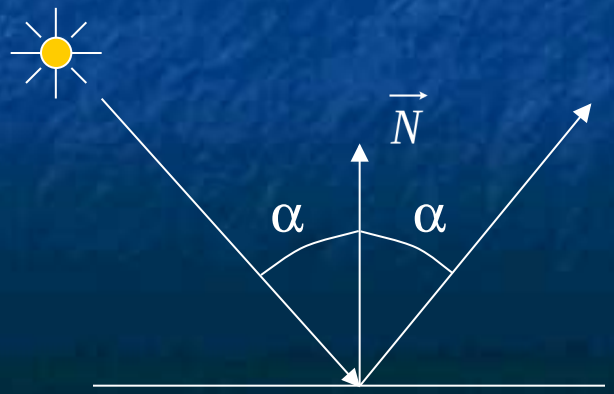
Modes de réflexion



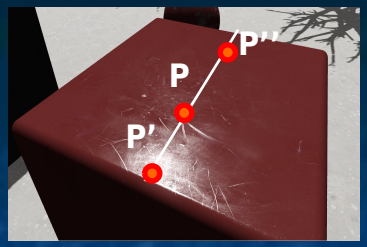
- Cas général :
  - Réflexion dans toutes les directions
  - Intensité différente

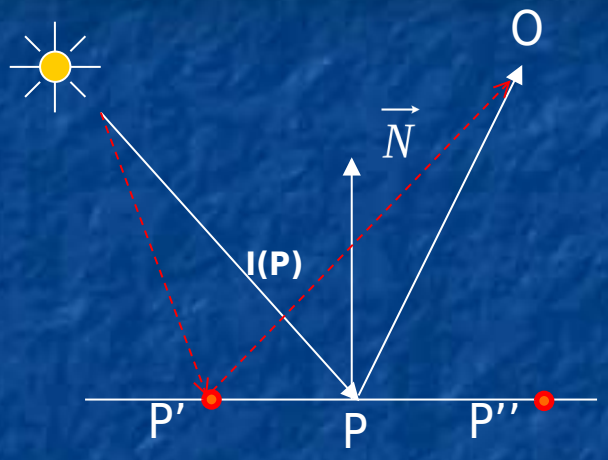


- Approximations :
  - Réflexion diffuse
    - Même intensité dans toutes les directions
    - Intensité indépendante du point de vue
  - Réflexion spéculaire
    - Une seule direction de réflexion
    - Intensité dépendante du point de vue

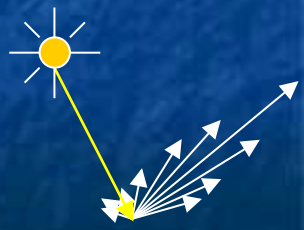


Calcul d'éclairage local



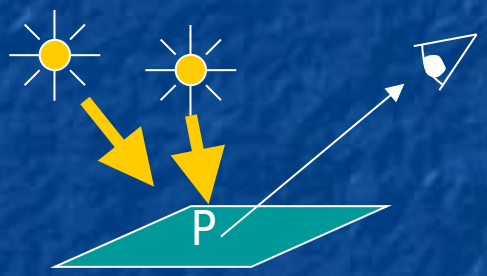


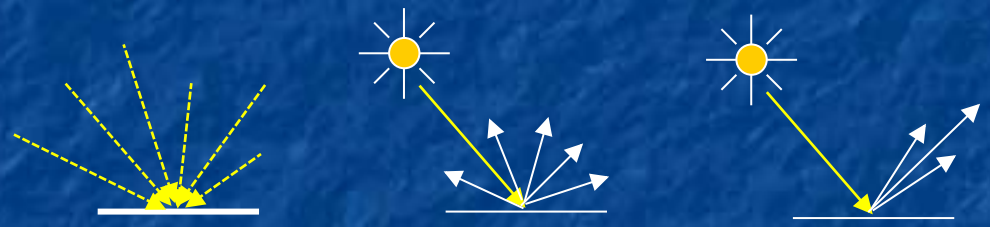
- Objectifs :
  - Calculer l'intensité de lumière I(P) reçue en un point P
  - Calculer l'intensité réfléchie vers l'observateur O
- Simplifications
  - Ne pas tenir compte des autres objets
    - Pas d'ombres ni de réflexions
  - Modèle de réflexion de la lumière simplifié
    - Différents modèles (Gouraud, Phong, Ward, …)



Le modèle d'éclairage de Phong

- Modèle empirique (1973)
  - Résultat visuellement convaincant
  - Coût de calcul limité - Utilisé par OpenGL & Three.js
- Expression
  - Soit I(P) l'intensité lumineuse réfléchie vers l'observateur au point P





$$I(P) = I_a(P) + I_d(P) + I_s(P)$$

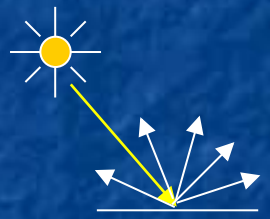
Intensité ambiante en P

Intensité réfléchie de manière diffuse en P

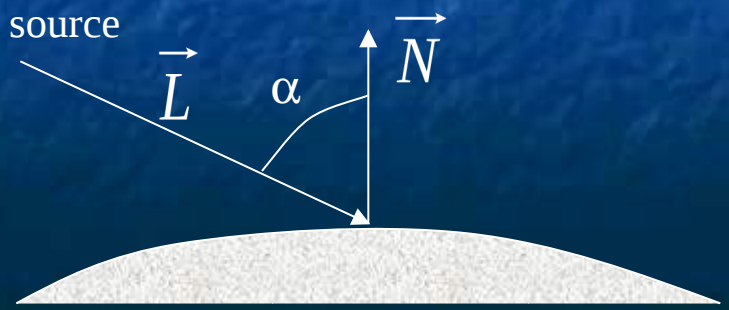
Intensité réfléchie
de manière spéculaire en P

Intensité diffuse

- Hypothèses
  - L'intensité reçue depuis une source varie avec l'angle d'incidence
    - Lumière rasante : intensité faible
    - Lumière zénithale : intensité maximale
  - La quantité de lumière réfléchie
    - Est indépendante de l'angle de réflexion (réflexion diffuse)
    - Dépend des propriétés de réflectance du matériau



On pose : $I_d(P) = K_d.I.(\vec{N}.\vec{L}) = K_d.I.\cos(\alpha)$ $avec\ 0 \leq \alpha \leq \frac{\pi}{2}$

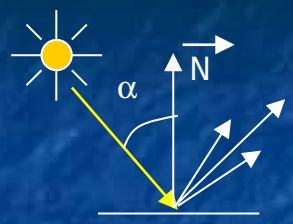


Coefficient diffus de l'objet (dans [0,1])

Intensité de la source

Intensité spéculaire

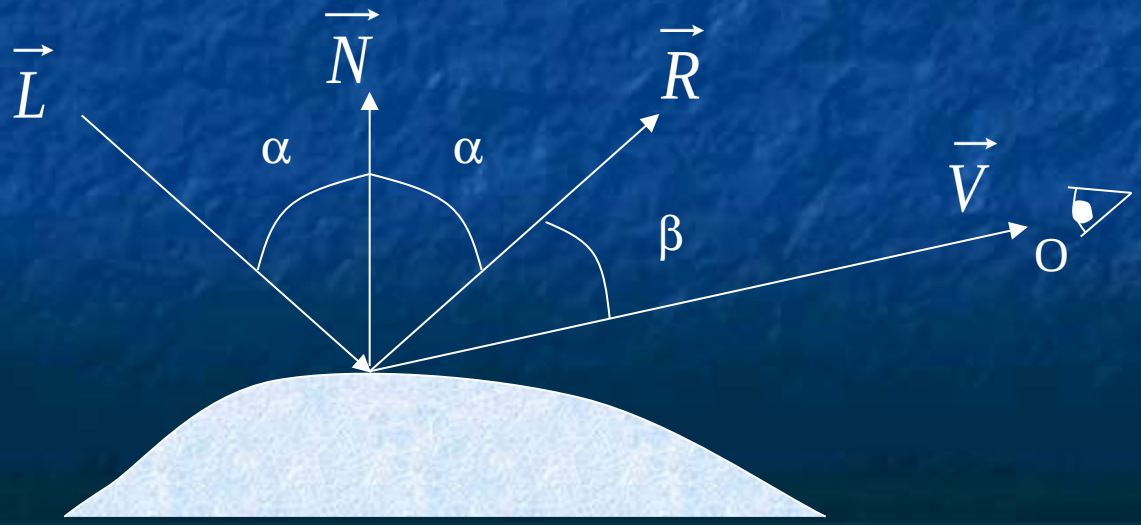
- Hypothèses :
  - La quantité de lumière réfléchie
    - dépend de l'angle de réflexion
    - Est maximale dans la direction spéculaire pure
    - S'atténue plus ou moins vite dans les autres directions en fonction du degré de brillance du matériau



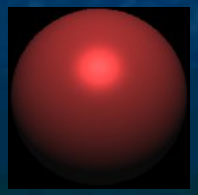
On pose : $I_s(P) = K_s . I.(\vec{V}.\vec{R})^s = K_s . I.\cos^s(\beta)$

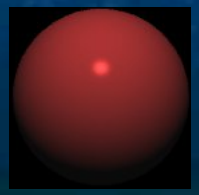
Coefficient spéculaire de l'objet (dans [0,1])

Intensité de la source



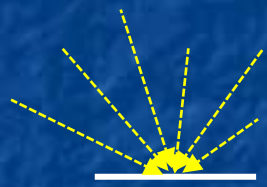
S = indice de brillance
Matériau terne : s petit (1-10)
Matériau brillant : s grand (10-100)





Intensité ambiante

- Points non directements éclairés : noirs !!!
- Dans la réalité :
  - Nombreuses réflexions de lumière éclairent ces points
  - Difficiles à prendre en compte
- Eclairage ambiant :
  - permet d'associer un éclairage « global » aux parties d'objets non directement éclairées par une source



On pose Ia(P) = Ka . Isa

Intensité de la source ambiante (constante commune à tous les objets)

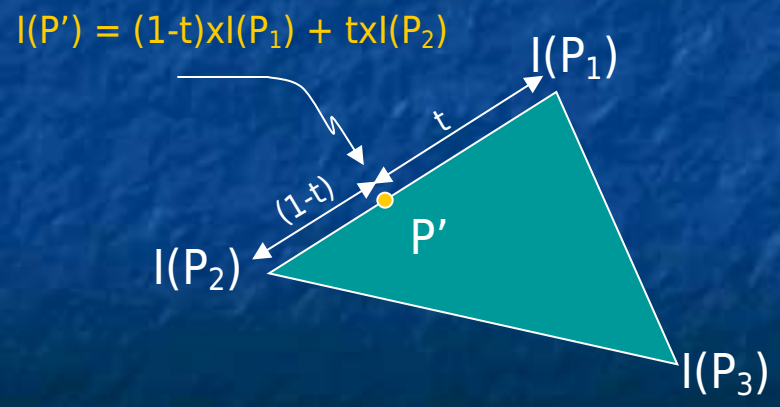
Coefficient ambiant de l'objet (dans [0,1])

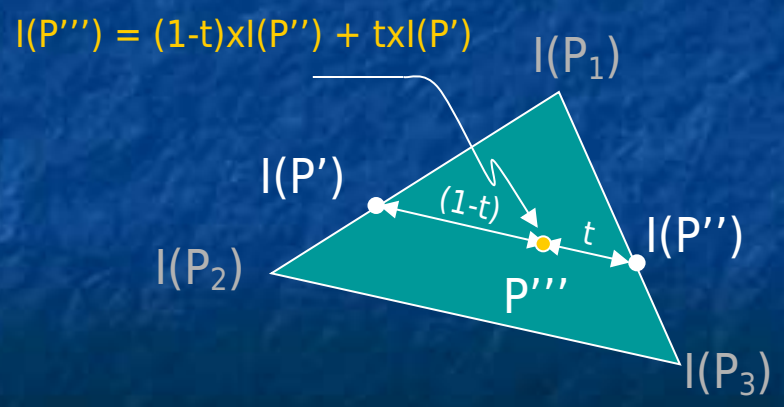
Méthodes d’interpolation

- Problématique
  - Calcul de I(P) en chaque point projeté sur l’écran
    - Calcul coûteux
    - Nombreux points invisibles
      - Calcul fait même pour les points qui ne seront pas visibles au final
  - Nécessité de réduire le coût de calcul
    - Faire un calcul exact au sommet des facettes
    - Interpoler les valeurs à l’intérieur des facettes

Interpolation de Gouraud (1)

- Principe
  - Calculer l'intensité au sommet des facettes
  - Interpoler les intensités à l'intérieur des facettes

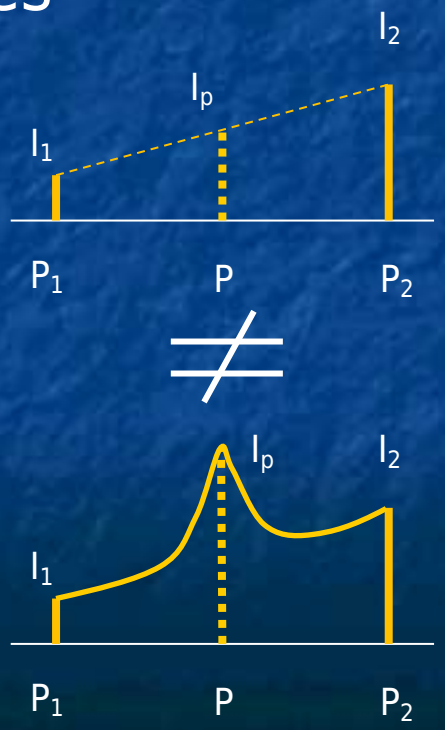




Interpolation de Gouraud (2)

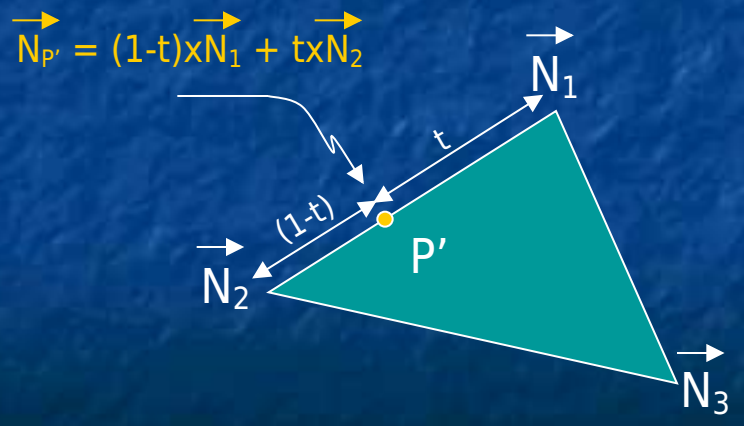
- Avantage :
  - Réduction importante du coût de calcul
- Inconvénients
  - interpolation linéaire sur les intensités

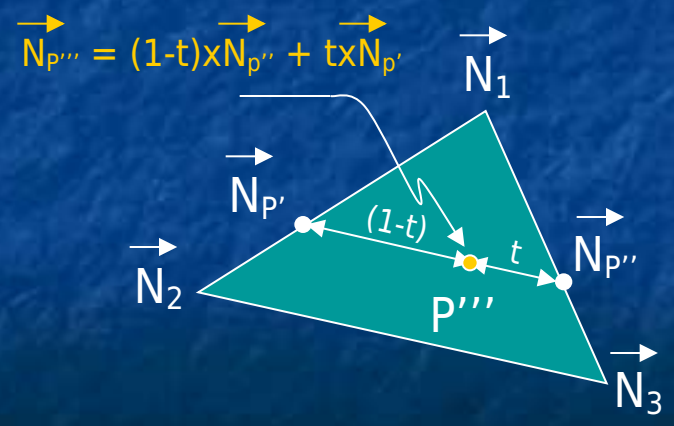
  => pas d'effets spéculaires
  - effets spéculaires :
    - effets non linéaires
    - localisés autour d'un point



Interpolation de Phong (1)

- Principe
  - Interpoler les normales
  - Effectuer le calcul exact en tout point





Interpolation de Phong (2)

- Avantage
  - Bonne représentation des réflexions spéculaires
- Inconvénient
  - Calcul beaucoup plus complexe
    - Évaluer $\cos^s(\beta)$ en tout point

OpenGL & Three.js : Interpolation de Gouraud

Plan du cours

1. Introduction
2. Modélisation d’objets 3D
3. Modèle d'éclairage local
4. Rendu temps réel
5. Introduction à Three.js
6. Le ray tracing

Rendu temps réel

- Affichage temps réel :
  - Fréquence d'affichage des images > 25 images par seconde
    - Limite perceptive pour la perception de saccades
- Rendu temps réel
  - Calcul minimum de 25 images par seconde
  - Réduction de la fatigue visuelle :
    - ~60 images par seconde
  - Utilisation du <u>pipeline de rendu</u>

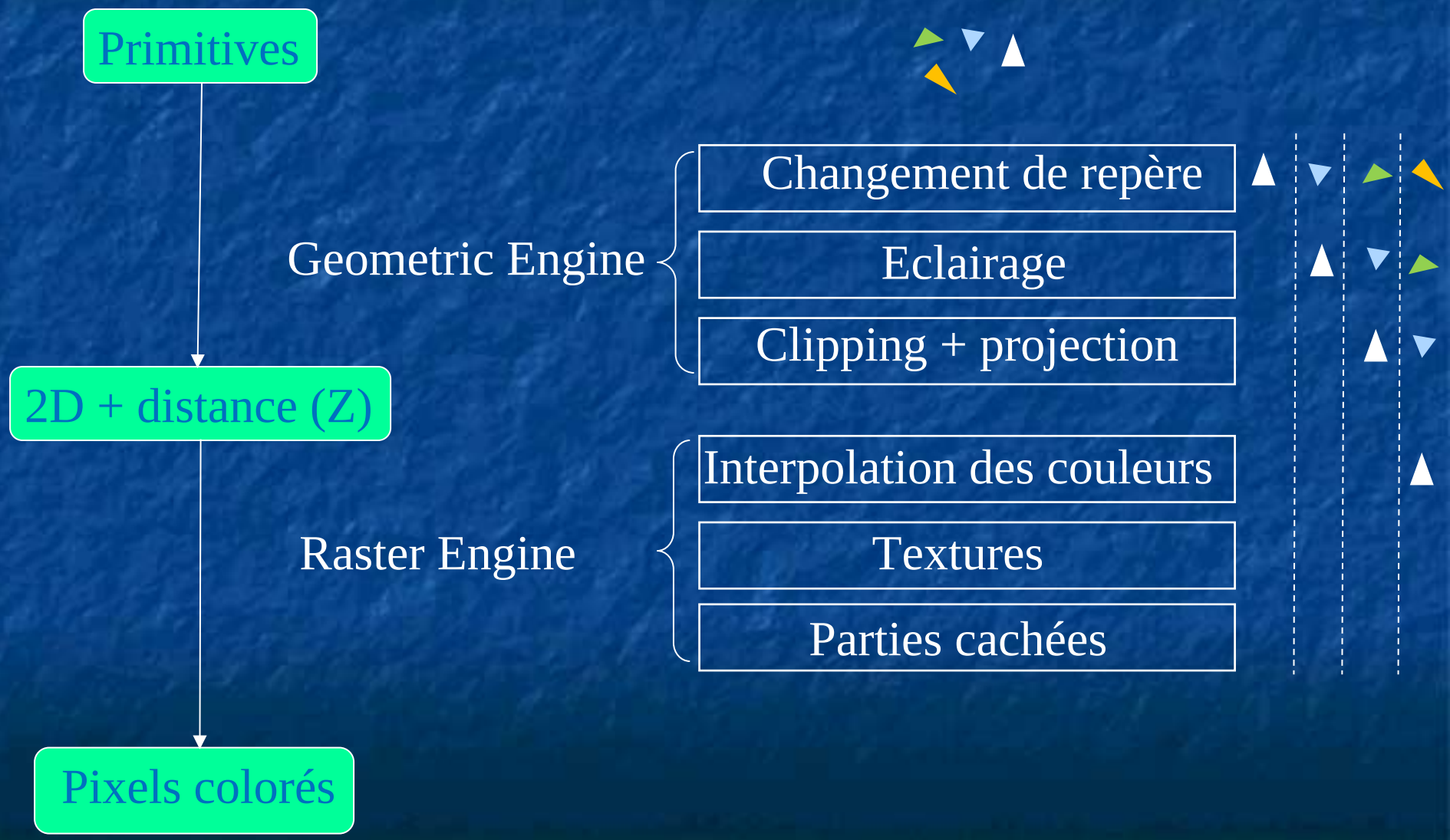
Le pipeline de rendu

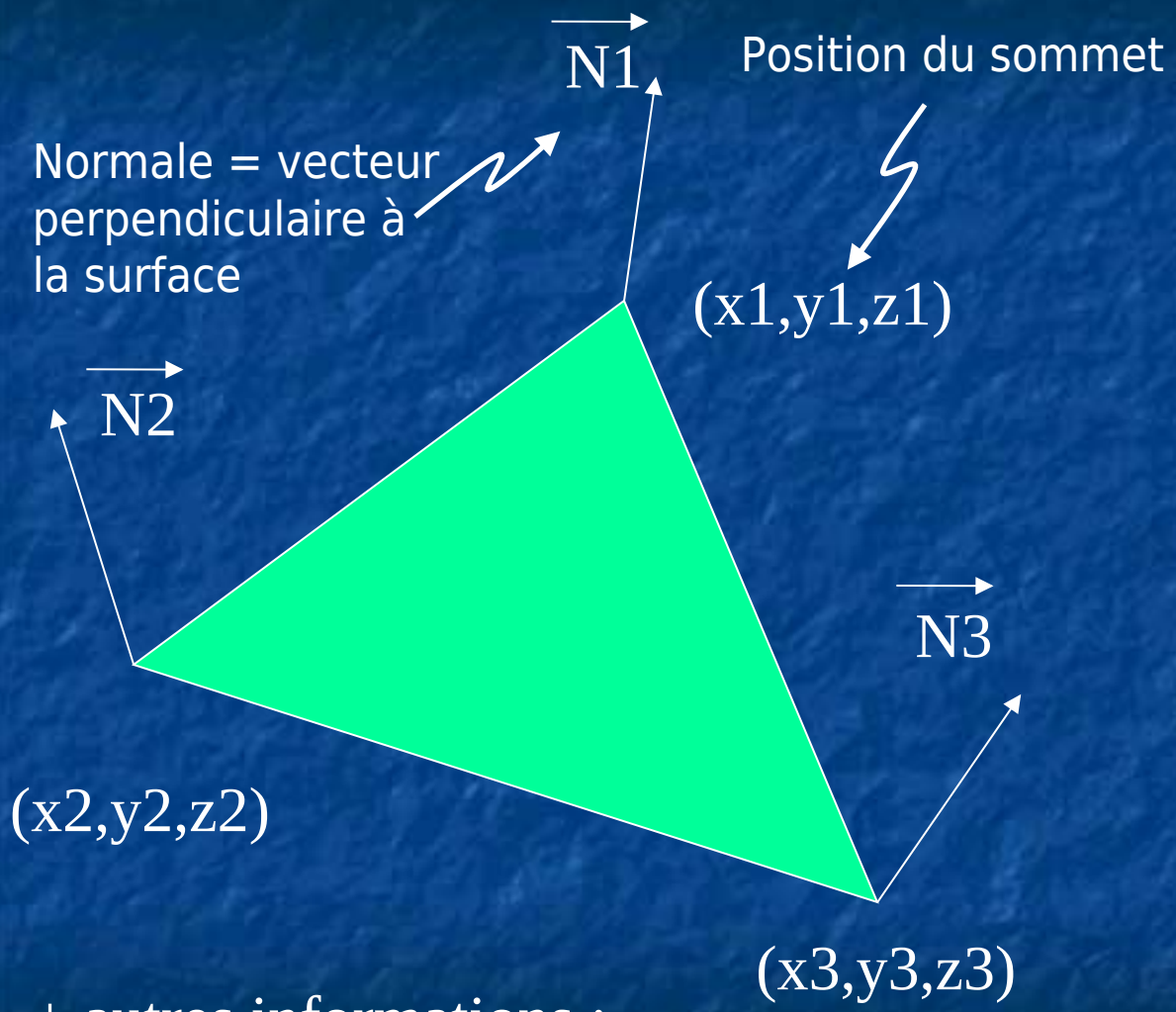
- Succession des étapes nécessaires à l'affichage de chaque facette sur la grille de pixels représentant l'écran

- Hypothèse :
  - scène = ensemble de facettes triangulaires

- Exécution
  - software :
    - versions Mesa d'OpenGL
    - Librairies OpenGL, WebGL, Direct3D, ...
  - Hardware :
    - cartes accélératrices 3D

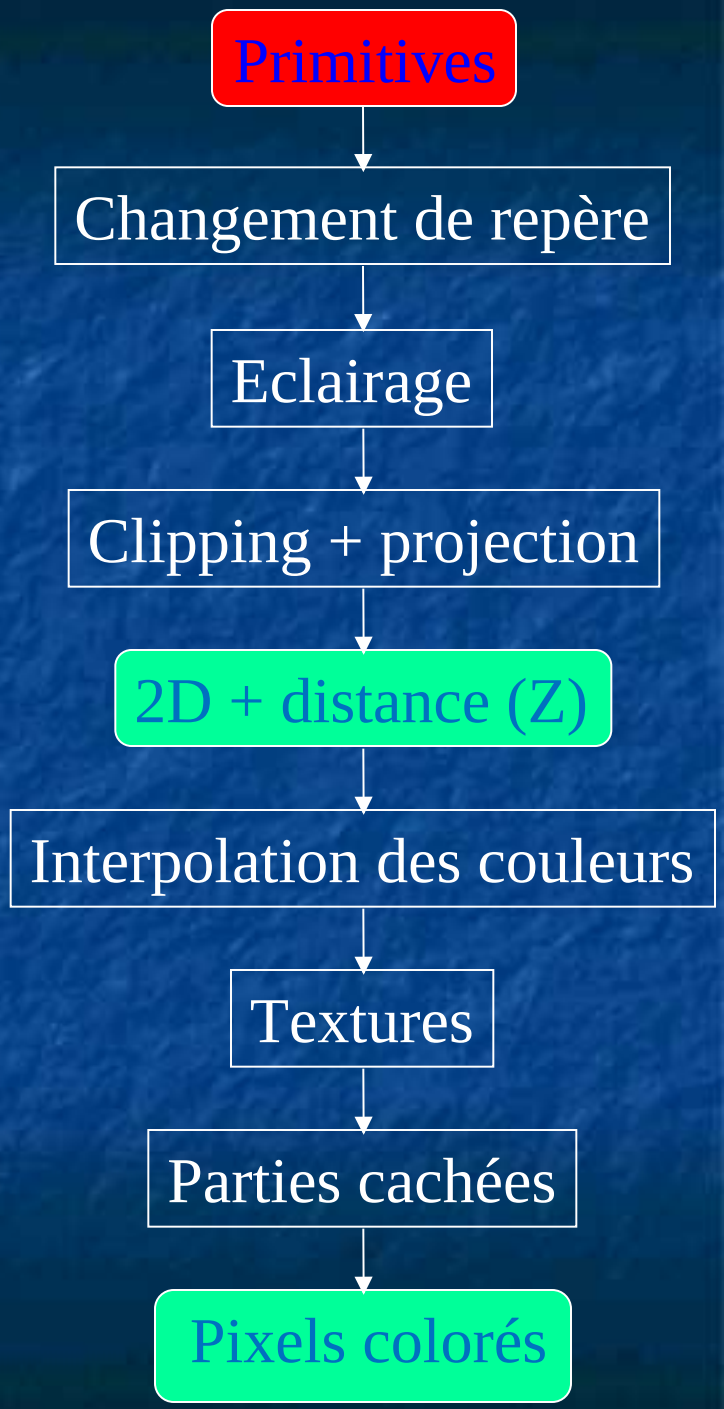
Le pipeline de rendu standard



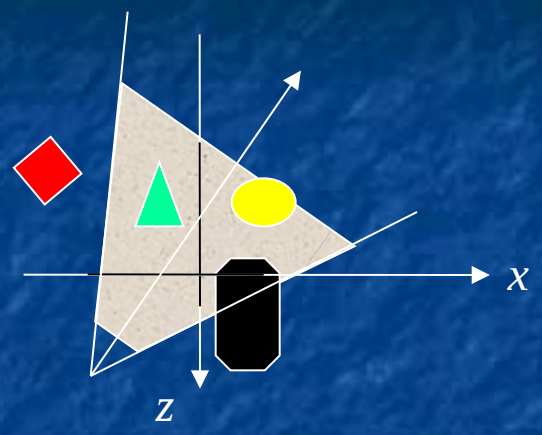


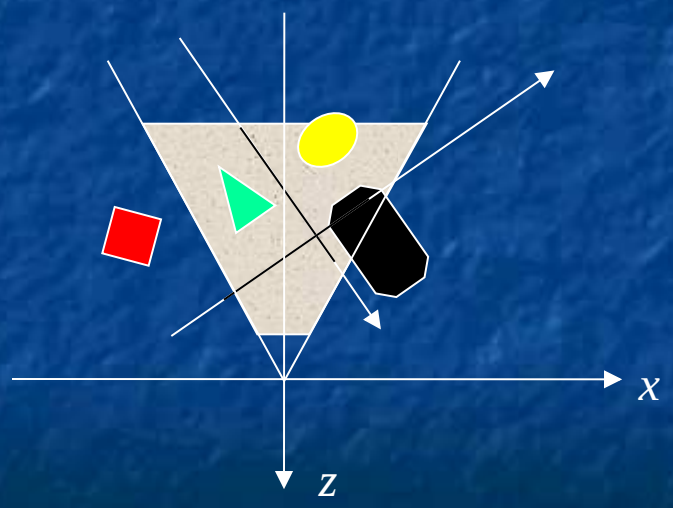
+ autres informations :

- caractéristiques du matériau
- coordonnées de texture
- …



Repère global





Repère observateur

Primitives

↓

Changement de repère

↓

Eclairage

↓

Clipping + projection

↓

2D + distance (Z)

↓

Interpolation des couleurs

↓

Textures

↓

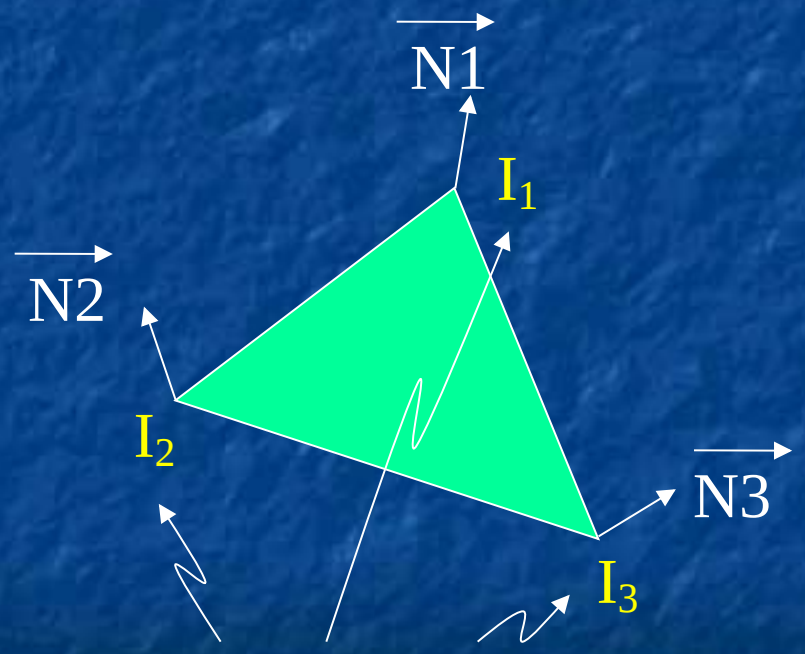
Parties cachées

↓

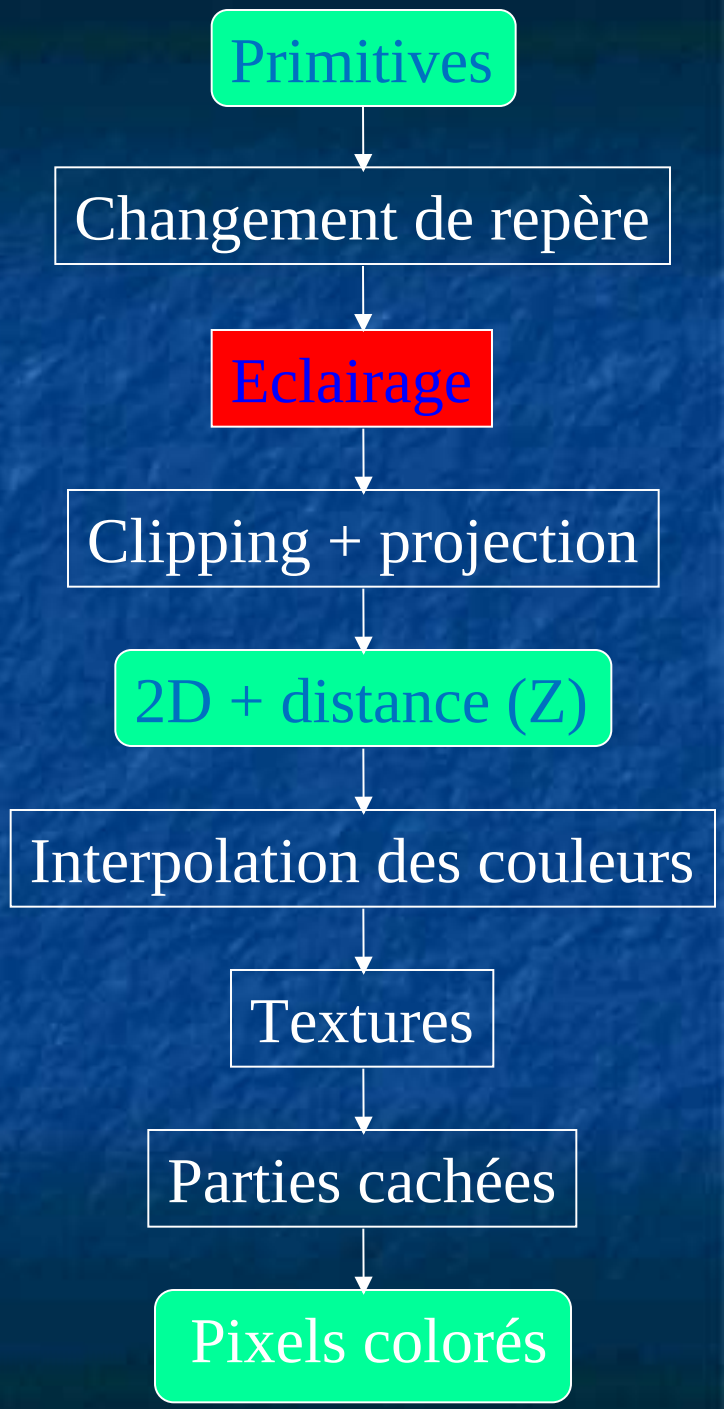
Pixels colorés

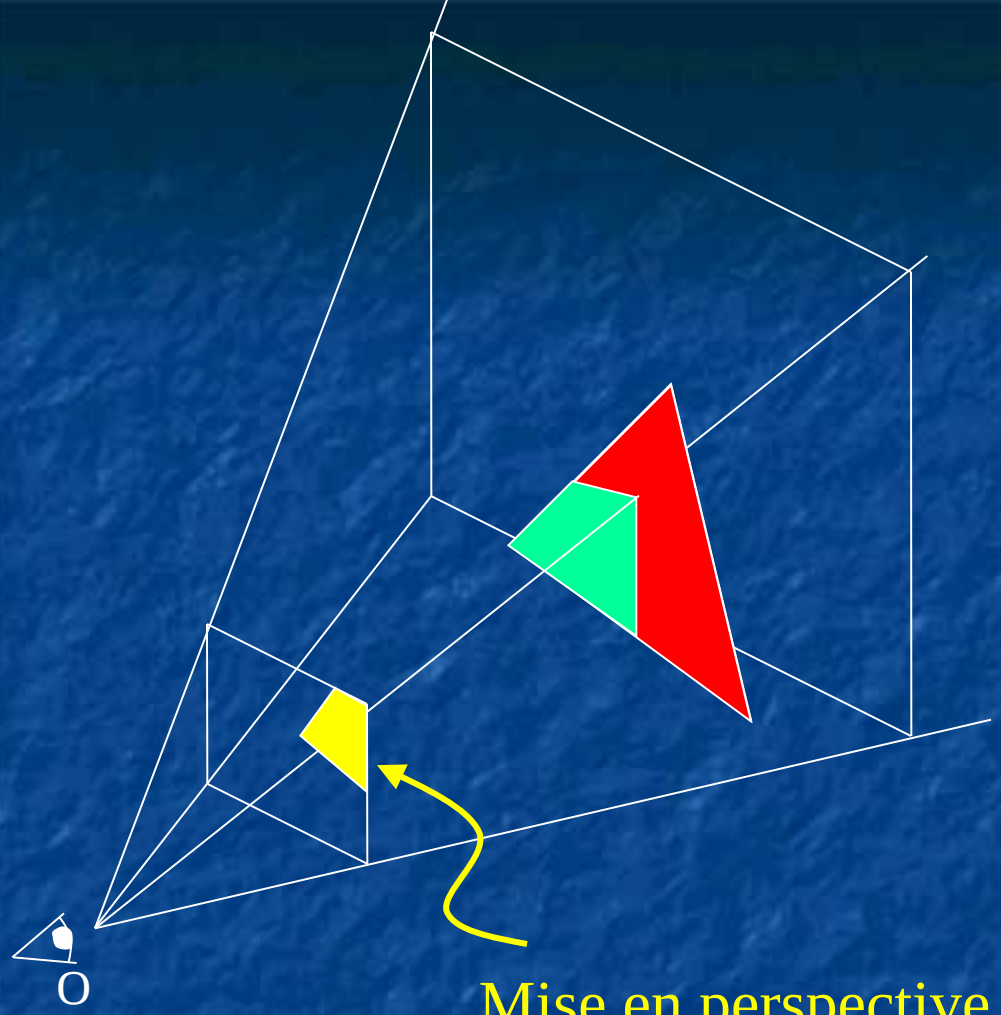
Calcul de l'effet lumineux produit sur chaque objet par chacune des sources présentes dans la scène

Calculs appliqués uniquement aux sommets de chaque triangle



Intensité obtenues par un modèle d'éclairage quelconque





Pyramide de vision = 6 plans de clipping

Primitives

↓

Changement de repère

↓

Eclairage

↓

Clipping + projection

↓

2D + distance (Z)

↓

Interpolation des couleurs

↓

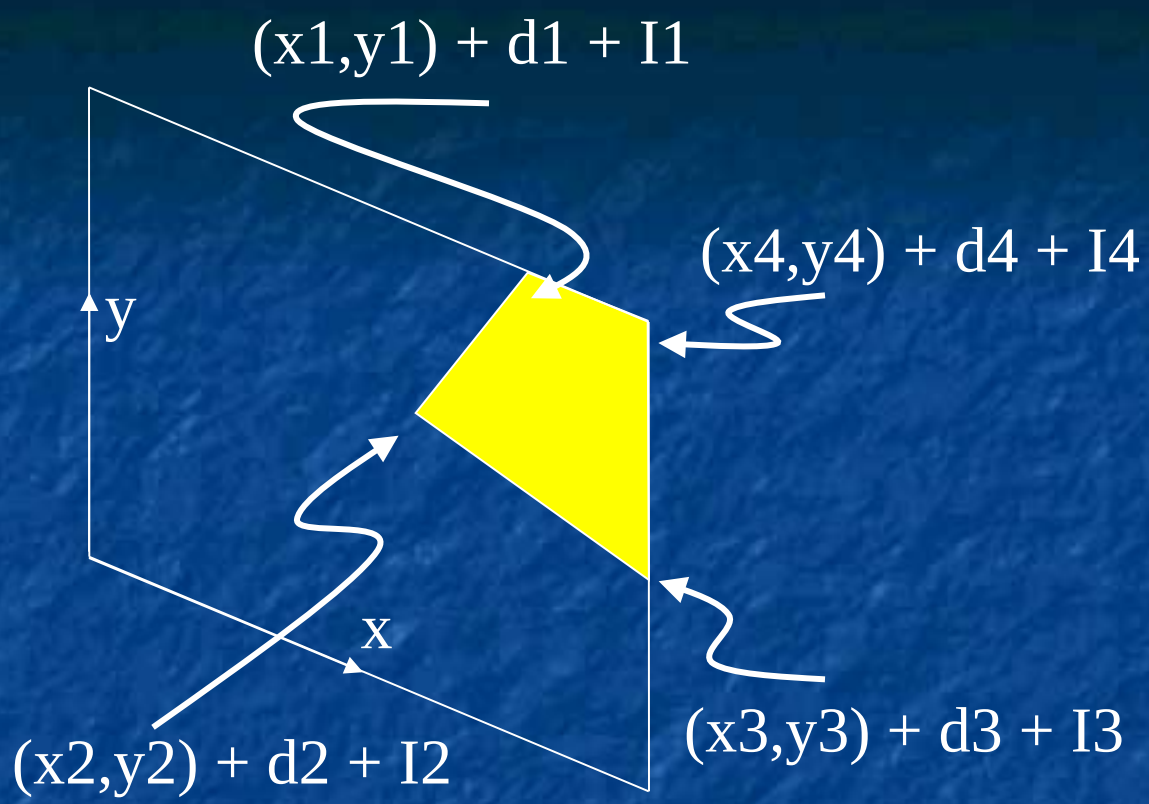
Textures

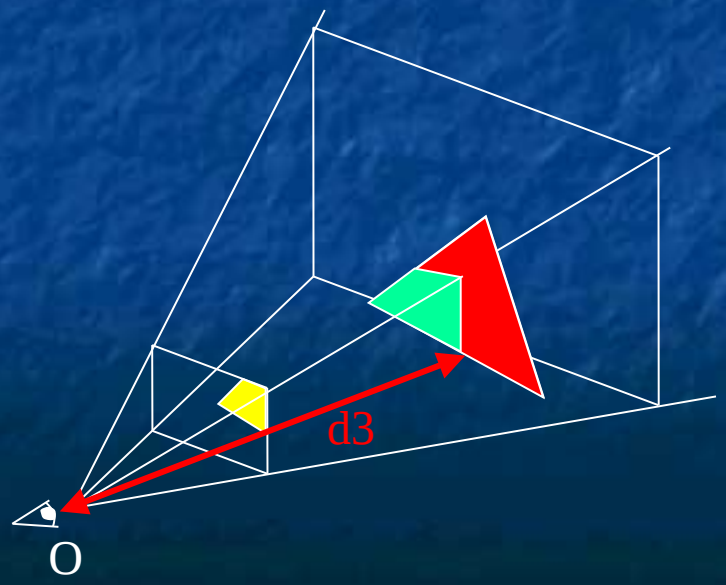
↓

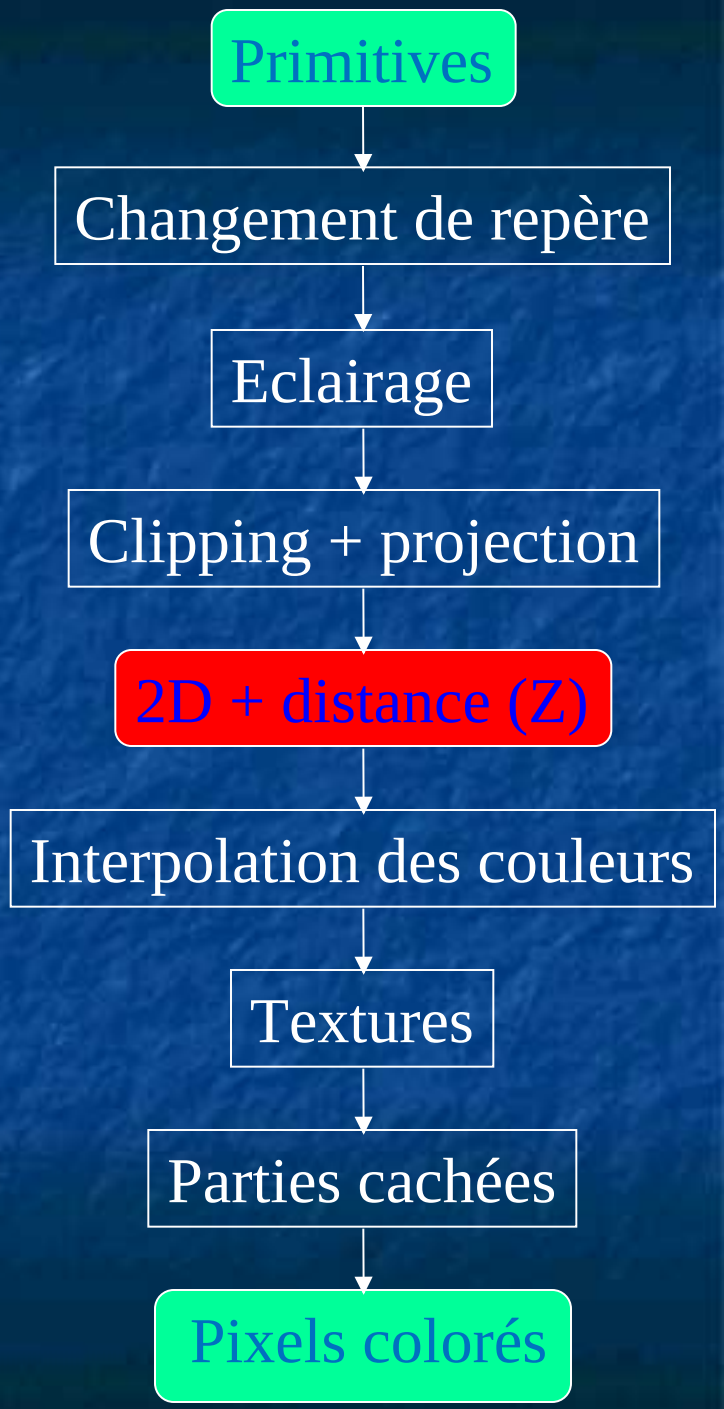
Parties cachées

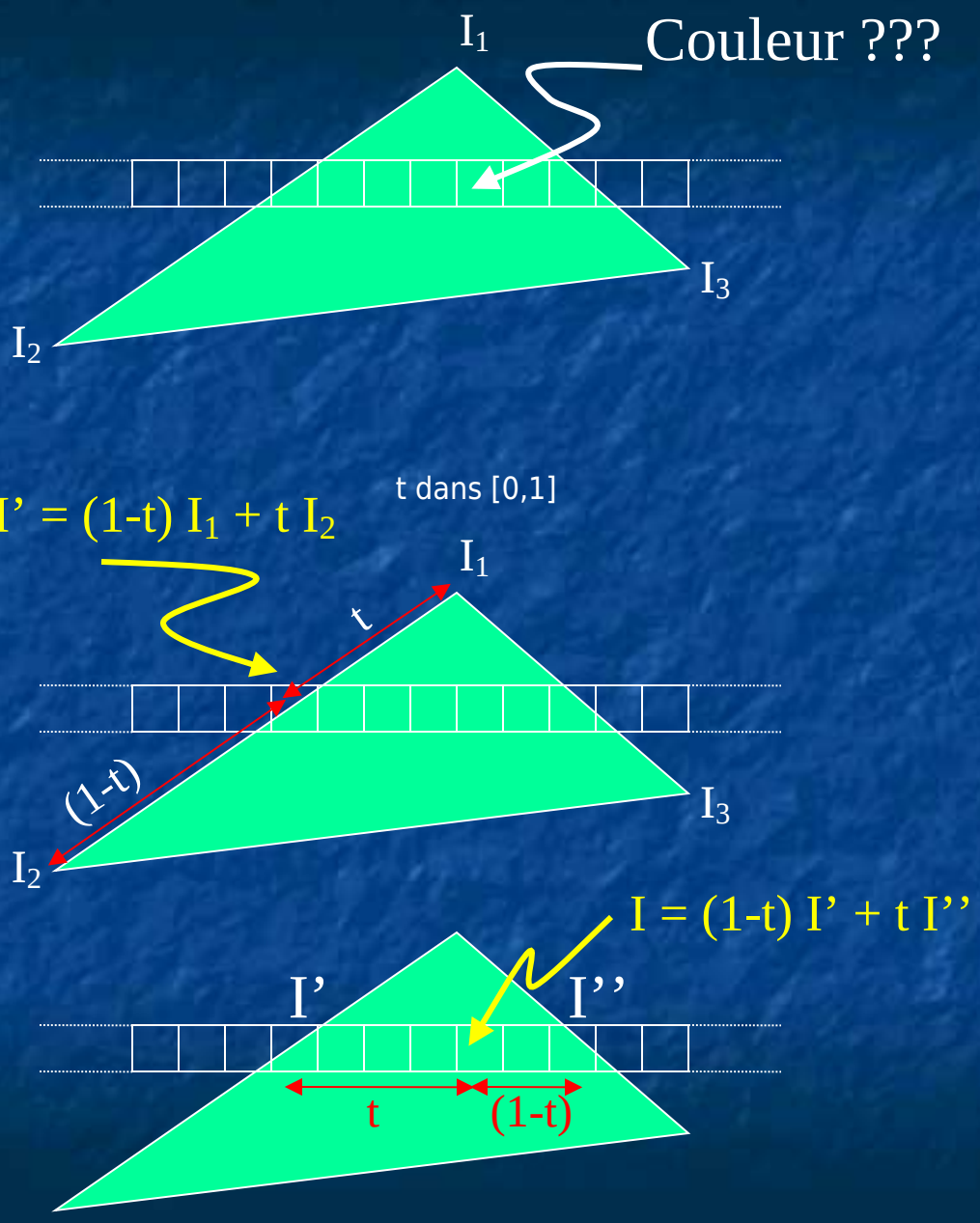
↓

Pixels colorés


(x1,y1) + d1 + I1
(x4,y4) + d4 + I4
y
x
(x2,y2) + d2 + I2
(x3,y3) + d3 + I3


d3
O


Primitives
Changement de repère
Eclairage
Clipping + projection
2D + distance (Z)
Interpolation des couleurs
Textures
Parties cachées
Pixels colorés



Primitives

↓

Changement de repère

↓

Eclairage

↓

Clipping + projection

↓

2D + distance (Z)

↓

Interpolation des couleurs

↓

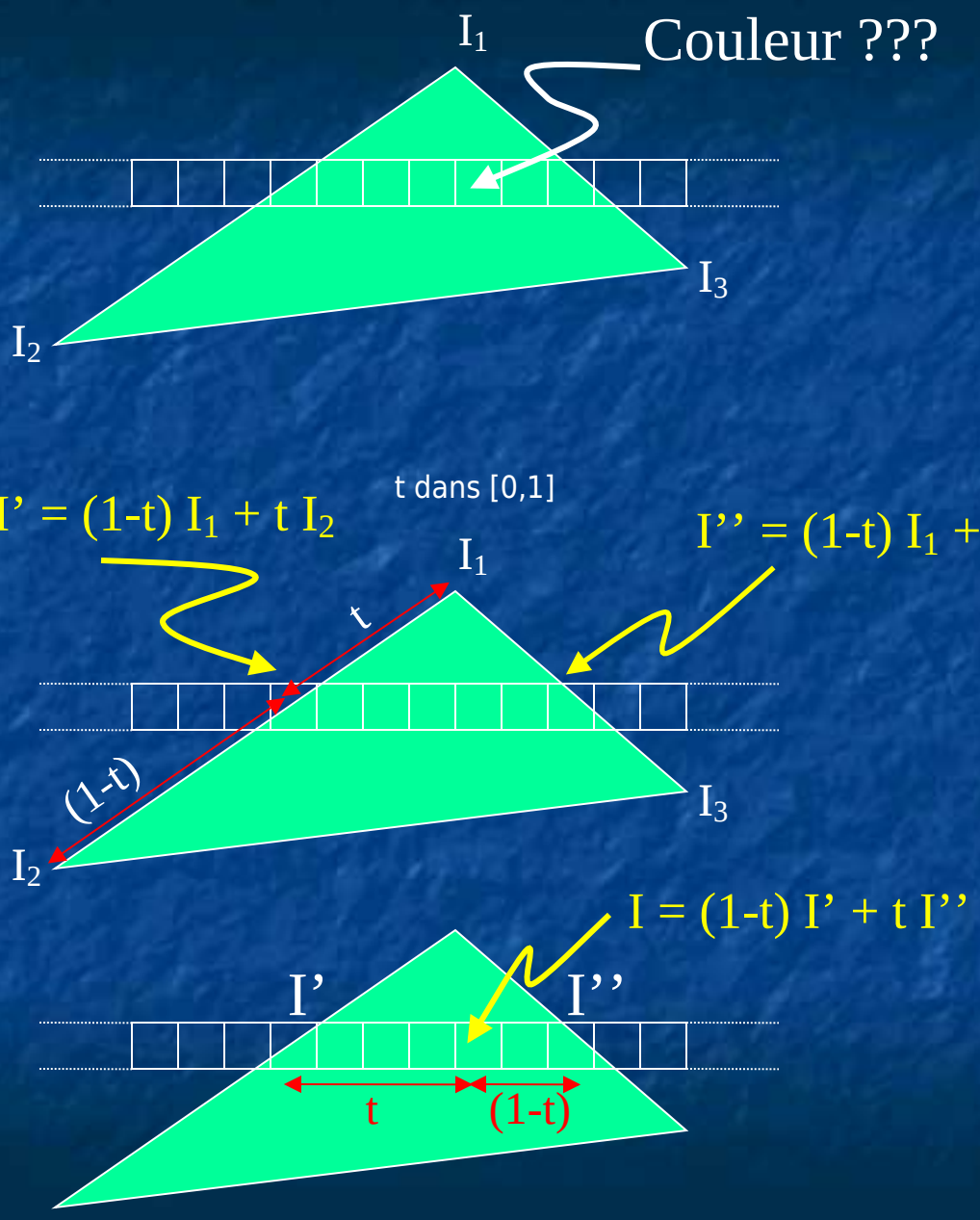
Textures

↓

Parties cachées

↓

Pixels colorés



Primitives

↓

Changement de repère

↓

Eclairage

↓

Clipping + projection

↓

2D + distance (Z)

↓

Interpolation des couleurs

↓

Textures

↓

Parties cachées

↓

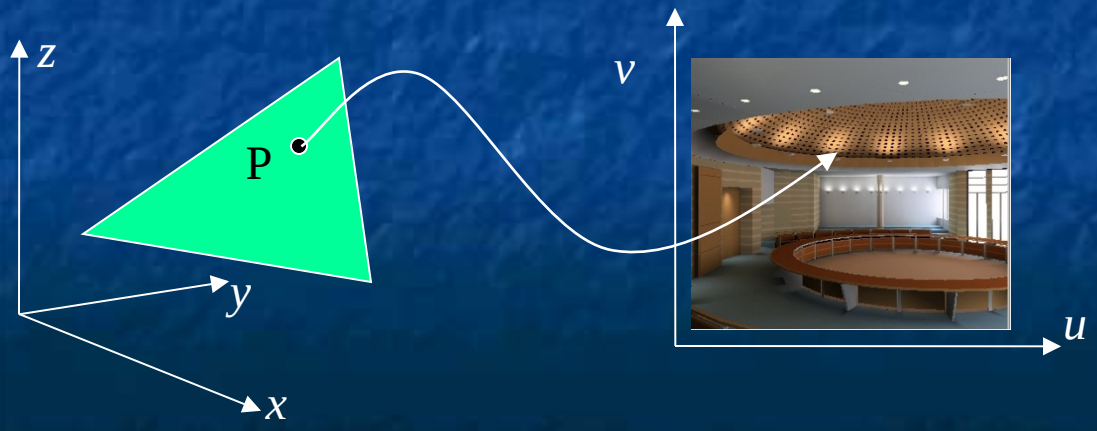
Pixels colorés

2D
mapping

3D
mapping

Texture 2D :

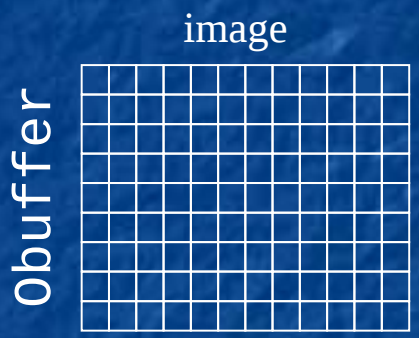
- trouver une transformation entre l'espace objet et l'espace des textures



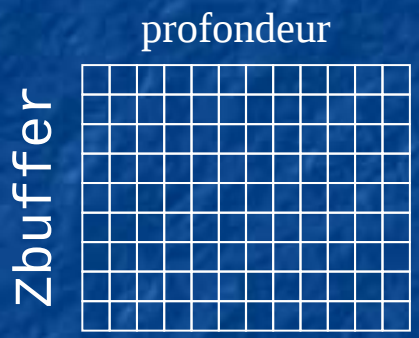
Primitives
↓
Changement de repère
↓
Eclairage
↓
Clipping
↓
2D + distance (Z)
↓
Interpolation des couleurs
↓
Textures
↓
Parties cachées
↓
Pixels colorés

Plusieurs objets peuvent se projeter sur le même pixel
Il faut garder le plus proche

Utilisation d'un tampon de profondeur (Z-buffer)



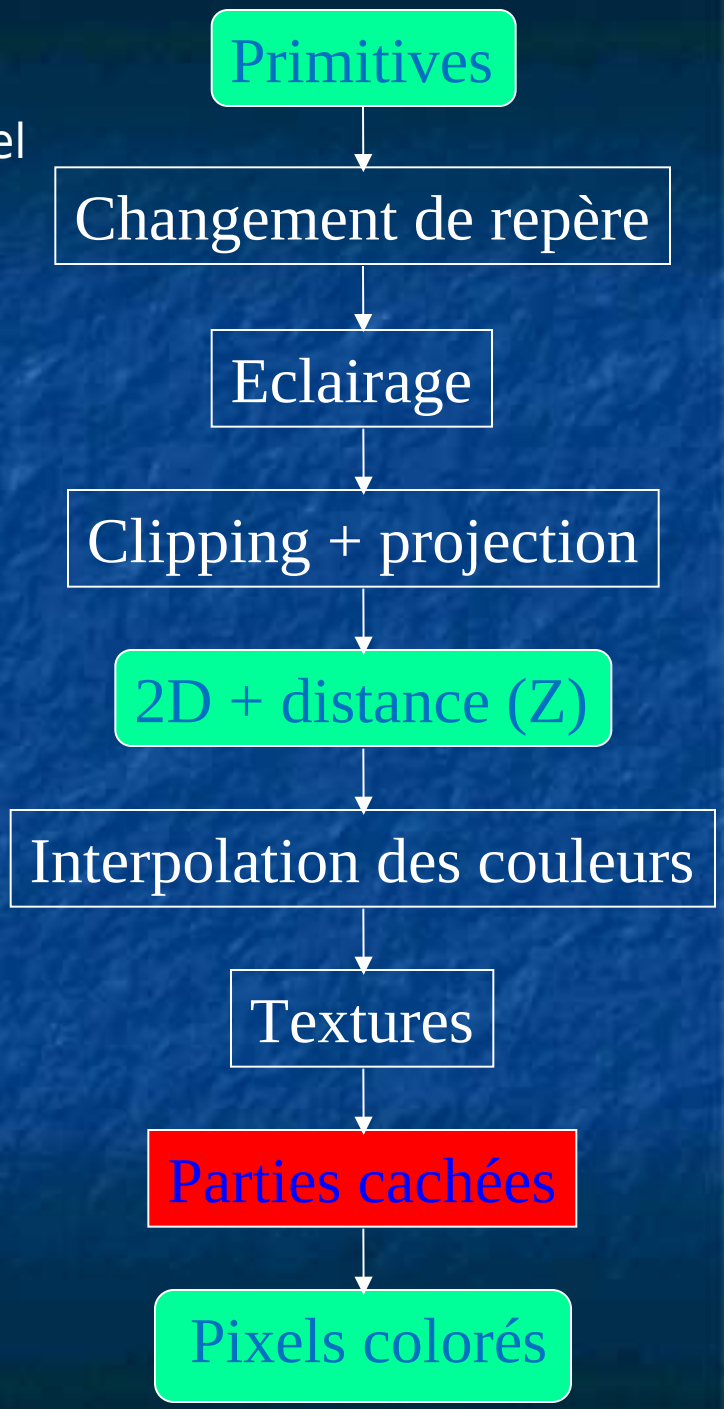
1 pixel = 1 couleur

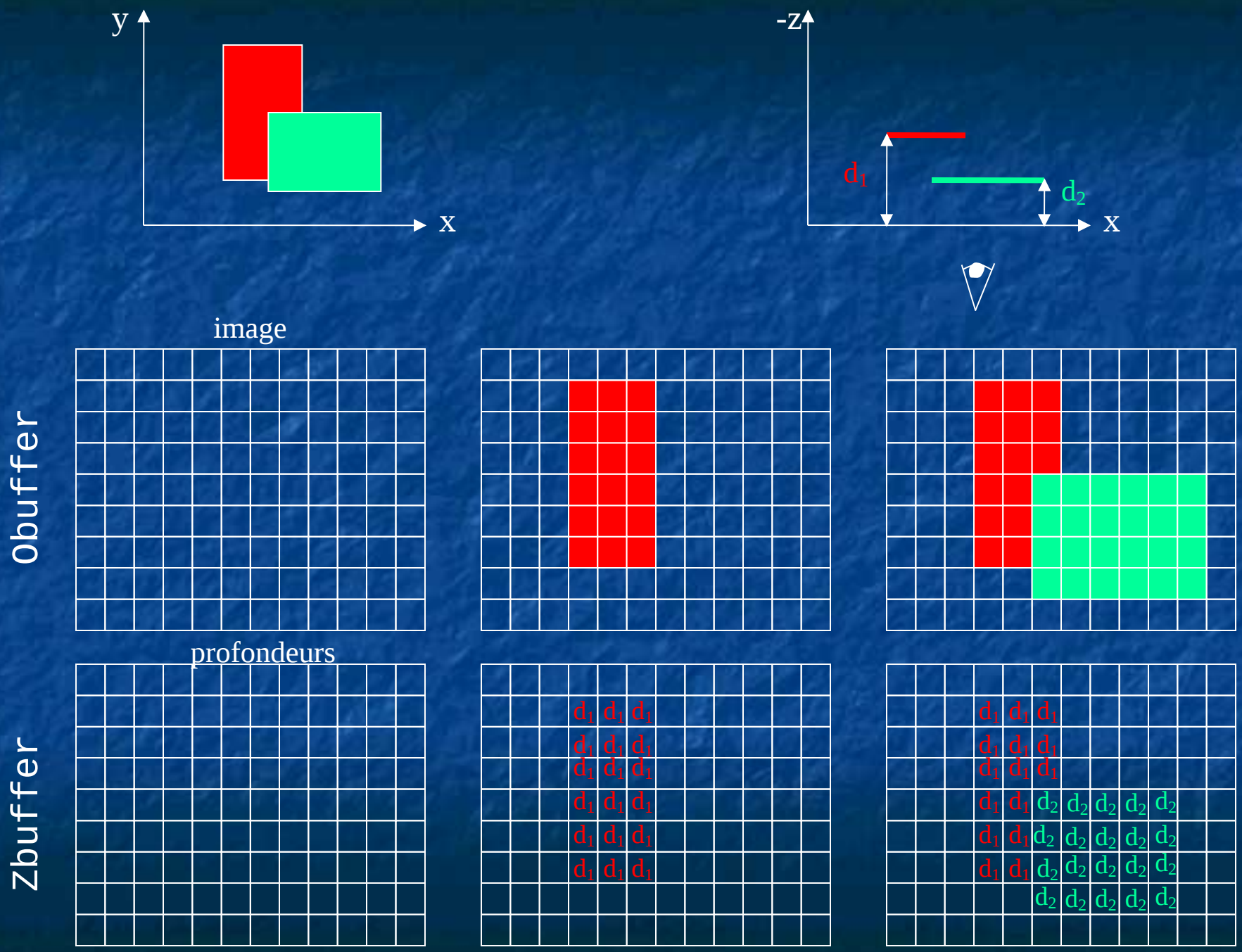


1 pixel = 1 distance

Mise à jour des buffers :

uniquement si la nouvelle primitive projetée est plus proche que celle actuellement mémorisée




y
x
-z
x
d_1
d_2
image
profondeurs
Obuffer
Zbuffer
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_1
d_1 d_1 d_2 d_2 d_2 d_2 d_2
d_1 d_1 d_2 d_2 d_2 d_2 d_2
d_1 d_1 d_2 d_2 d_2 d_2 d_2
d_2 d_2 d_2 d_2 d_2

Améliorations

- Cartes récentes offrent la possibilité de :
  - programmer l'étape d'éclairage (Vertex Shaders)
  - modifier la géométrie traitée (geometry Shaders)
  - manipuler des pixels via des programmes plus complexes (Fragment shaders)
- Caractéristiques :
  - programmation en assembleur dédié ou en langages de + haut niveau (ex. GLSL)
- Programmation d'autres "algorithmes"
  - Langage et librairie Cuda (Nvidia)

Plan du cours

1. Introduction
2. Modélisation d’objets 3D
3. Modèle d'éclairage local
4. Rendu temps réel
5. Introduction à Three.js
6. Le ray tracing

Introduction (1)

- Three.js c’est quoi ?
  - Interface de programmation (API)
    - Conception d’application 2D / 3D
    - S'appuie sur WebGL
      - version javascript d'OpenGL ES (*Embedded Systems*)
    - S'exécute en javascript dans le navigateur
      - Indépendante de la plateforme
      - Compatible avec la plupart des navigateurs
  - Affichage dans un canvas html 5
      - Compatible avec les canvas 2D sans WebGL

Introduction (2)

- Objectif :
  - Faciliter le développement d'applications 2D/3D
    - WebGL
      - puissant mais complexe à utiliser
      - Primitives graphiques de bas niveau
    - Three.js
      - Léger
      - Facile à utiliser
        - Nombreuses fonctionnalités de "haut niveau"
      - Rendu webgl, canvas ou svg

Introduction (3)

- Incorporation à une page web

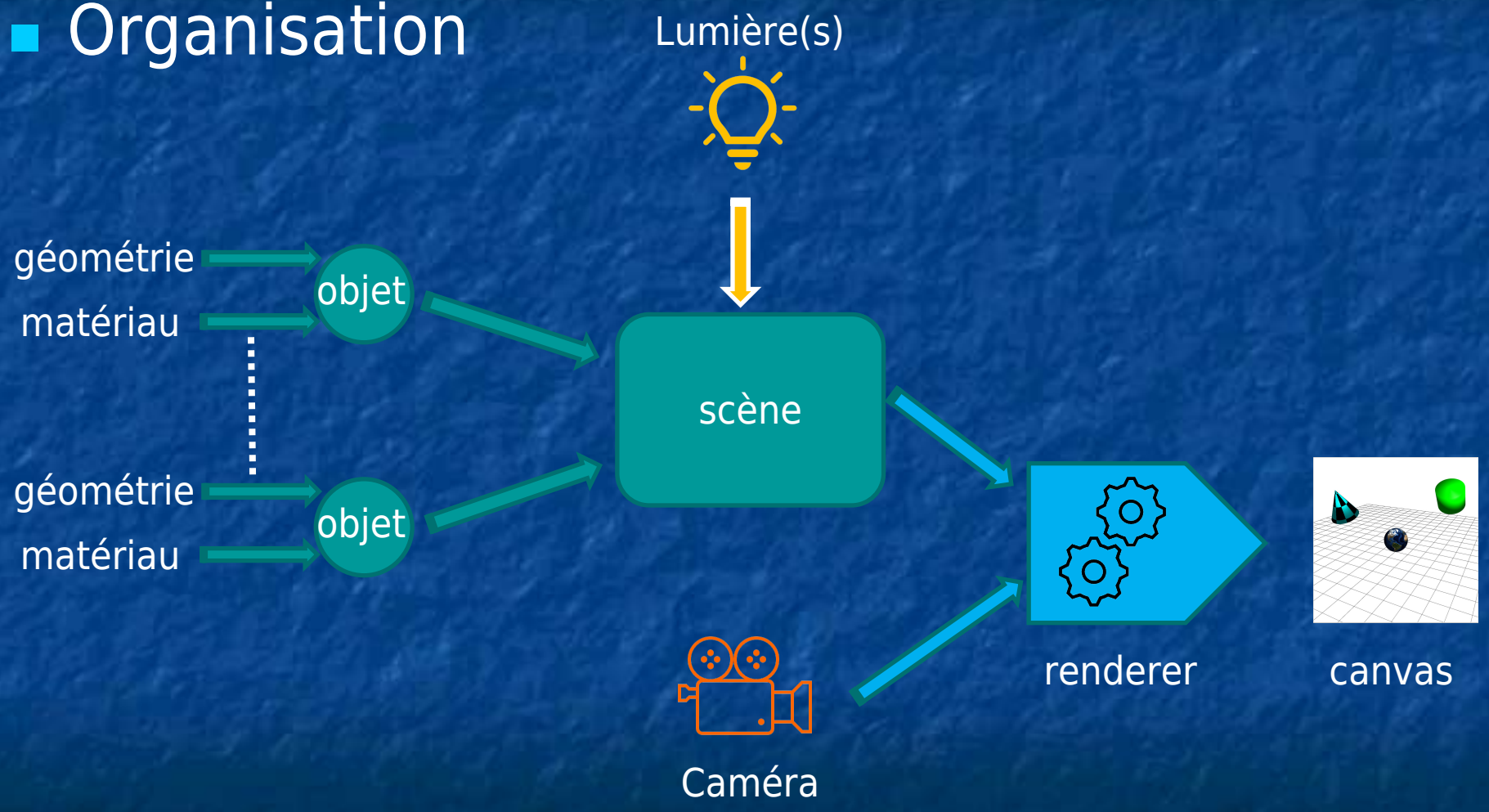
```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="utf-8">
    <title>Exemple basique</title>
  </head>

  <body>

    <script src="js/three.js"></script>

    <script>
      // développer le script Three.js ici ...
    </script>

  </body>
</html>
```

Inclure le script de la libraire

Développer son application

Introduction (4)

- Organisation



Introduction (5)

- Exemple

```
// création d'un objet
const geometrie = new THREE.BoxGeometry();
const materiel = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
const cube = new THREE.Mesh( geometry, material );
```

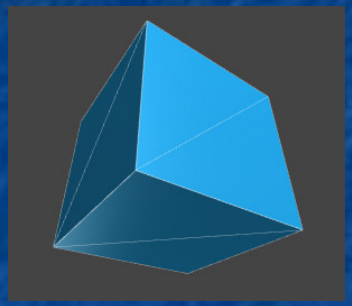
```
// création de la scène et ajout de l'objet
const scene = new THREE.Scene();
scene.add( cube );
```

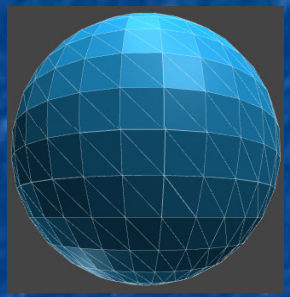
```
// Création d'une caméra
const camera = new THREE.PerspectiveCamera( 75, 1.0, 0.1, 1000 );
```

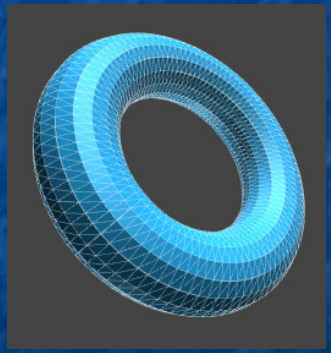
```
// Création et lancement du renderer
const renderer = new THREE.WebGLRenderer();
renderer.setSize( 500, 500 );
document.body.appendChild( renderer.domElement );
...
renderer.render( scene, camera );
```

Primitives géométriques

- Quelques objets prédéfinis
  - Parallélépipède (boite)
  - Sphère
  - Cône
  - Cylindre
  - Tore
  - …

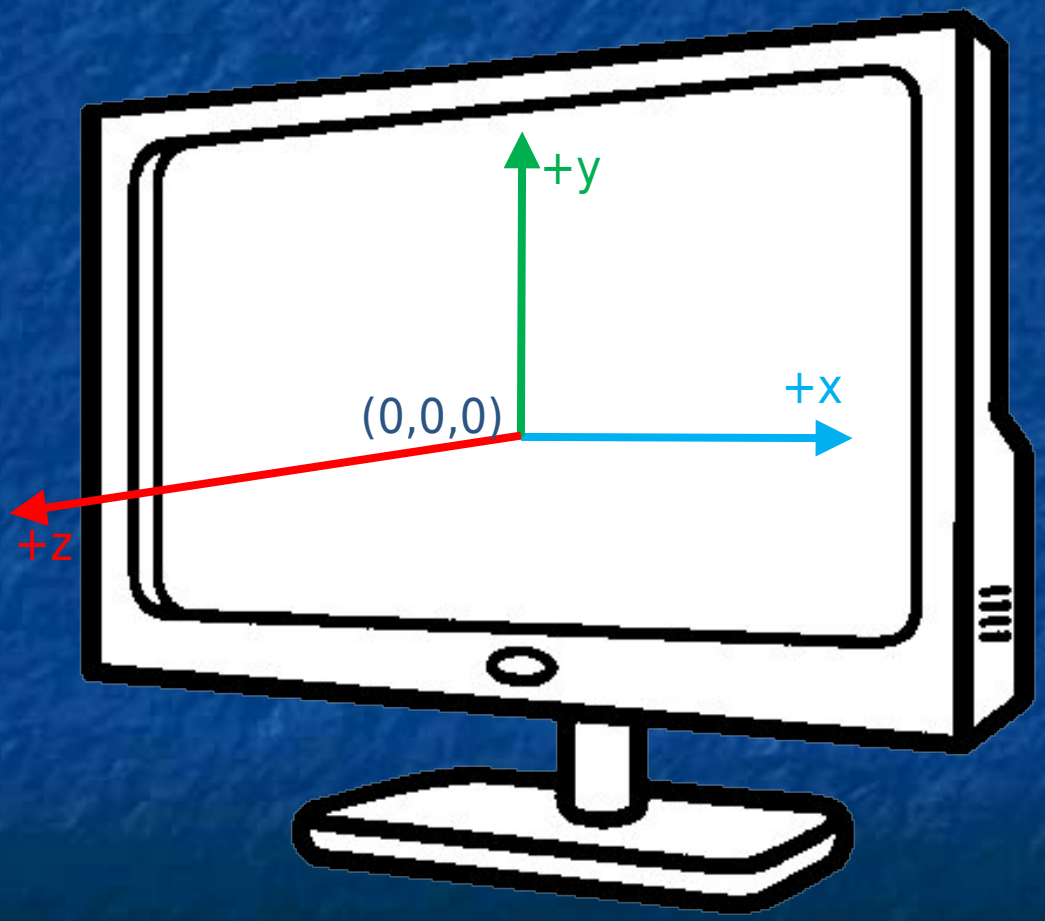




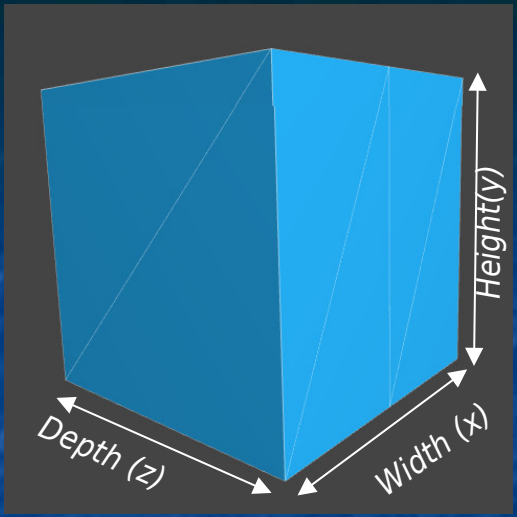


- Surfaces approchées par des triangles
- Positionnées à l'origine du repère global

Le repère de la scène 3D



La boite



- Syntaxe

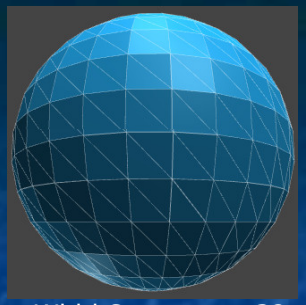
```
BoxGeometry(width : Float,
            height : Float,
            depth : Float,
            widthSegments : Integer,
            heightSegments : Integer,
            depthSegments : Integer)
```

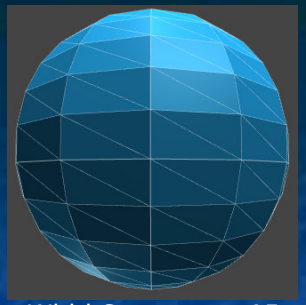
Dimensions en x, y et z

Nombre de découpes d'une arête en x, y et z

- Par défaut :
  - Centrée à l'origine
  - Côtés parallèles aux axes du repère
  - Valeur des paramètres : 1

La sphère


WidthSegments=32


WidthSegments=15

- Syntaxe

Rayon de la sphère

Nombre de segments en phi

Nombre de segments en thêta

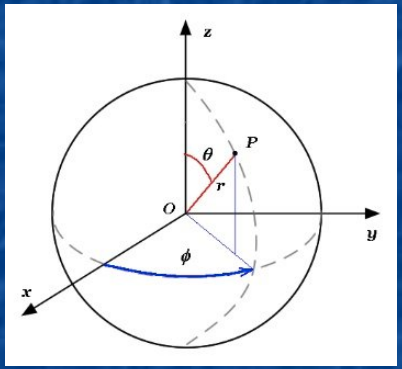
```
SphereGeometry(radius : Float,
               widthSegments : Integer,
               heightSegments : Integer,
               phiStart : Float,
               phiLength : Float,
               thetaStart : Float,
               thetaLength : Float)
```

Angle de départ en Φ

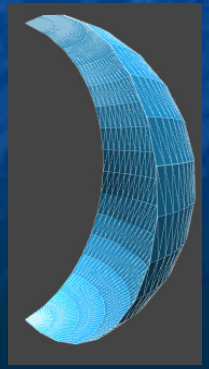
Longueur en Φ dans [0, 2Π]

Angle de départ en Θ

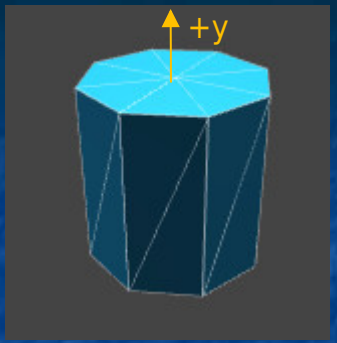
Longueur en Θ dans [0,Π]



- Par défaut :
  - Centrée à l'origine
  - Rayon 1
  - Découpage : 32 en Φ et 16 en Θ


phiLength = Π / 2

Le cylindre



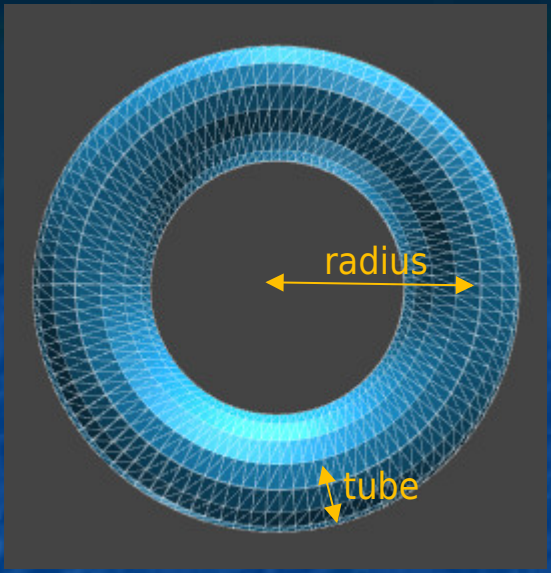
- Syntaxe

```
CylinderGeometry(radiusTop : Float,
                 radiusBottom : Float,
                 height : Float,
                 radialSegments : Integer,
                 heightSegments : Integer,
                 openEnded : Boolean,
                 thetaStart : Float,
                 thetaLength : Float)
```

- Rayon supérieur et inférieur
- Hauteur du cylindre
- Nombre de découpes
- Ouverture des extrémités
- Taille de la surface latérale à générer

- Par défaut :
  - Centré à l'origine, autour de l'axe Oy
  - Rayons à 1, hauteur à 1
  - OpenEnded à false (fermé)

Le tore



- Syntaxe

```
TorusGeometry(radius : Float,
              tube : Float,
              radialSegments : Integer,
              tubularSegments : Integer,
              arc : Float)
```

nombre de découpe de la circonférence du tore et de la circonférence du tube

Angle de génération pour la circonférence du tore

- Par défaut :
  - Centré à l'origine
  - Axe de symétrie Oz
  - Tore de rayon 1, tube de rayon 0.4

Remarque

- Objets définis par leur géométrie uniquement
  - Nécessité de leur associer un matériau

- Classe Mesh
  - Associe une géométrie à un matériau

Mesh(geometry : BufferGeometry,
material : Material)

la géométrie de l'objet

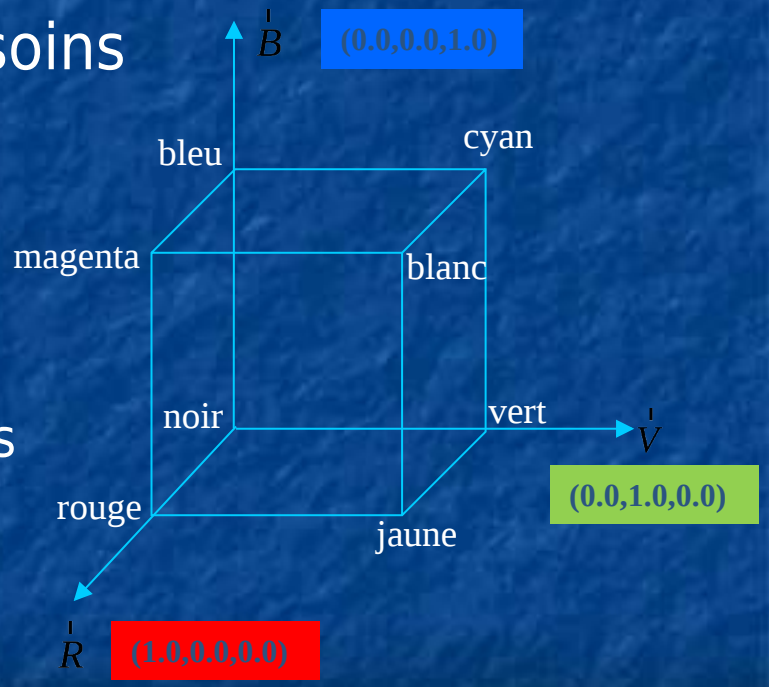
les caractéristiques de son matériau

```
const objet = new THREE.Mesh( ... , ...);
```

Matériaux (1)

- Nombreux matériaux disponibles
  - Héritent de la classe Material
  - Spécialisation en fonction des besoins
  - Nombreux attributs disponibles

- Couleurs
  - Définies dans le modèle (r,v,b)
    - Chaque valeur rvb est comprise dans
      - [0.0,1.0] valeurs réelles (pourcentage)
      - [0, 255] valeurs entières
      - [Ox00, 0xff] valeurs hexadécimales



(0, 255, 0) = (0.0, 1.0, 0.0) = 0x00ff00 = **vert**

Matériaux (2)

- Matériau basique
  - Hérite de la classe Material
  - N'est pas influencé par les sources de lumières

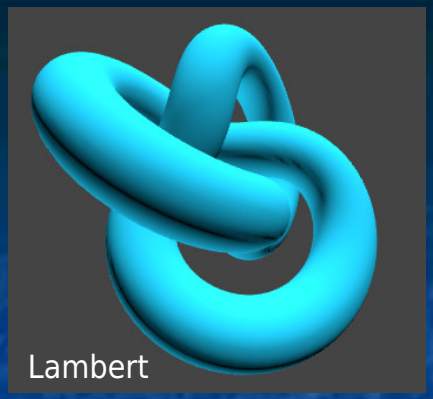
MeshBasicMaterial(parameters : Object)

les propriétés du matériau

- Propriété color
  - Fixe la couleur du matériau
  - Valeur par défaut : blanc (0xffffff))

```
const material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
```

Matériaux (3)



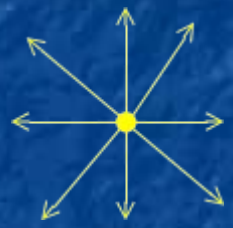
- Matériaux plus complexes
  - Le rendu est influencé par les sources de lumières
- Matériau diffus **MeshLambertMaterial(parameters : Object)**
  - Couleur de réflexion diffuse
- Matériau spéculaire **MeshPhongMaterial(parameters : Object)**
  - Modèle de Phong
  - Couleur de réflexion diffuse
  - Couleur de réflexion spéculaire
  - Coefficient de brillance

Les sources de lumière (1)

- Les sources :
  - Éclairent les objets de la scène
  - Différents types
    - Héritent de la classe Light
    - Disposent toutes :
      - D'une couleur (par défaut du blanc)
      - D'une intensité (par défaut 1.0)
    - Attributs additionnels selon le type de source

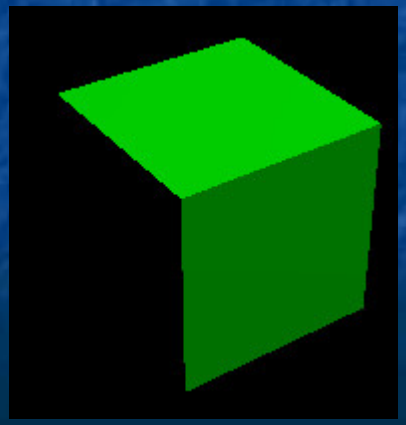
Les sources de lumière (2)

- Les sources ponctuelles :
  - Un point éclairant dans toutes les directions

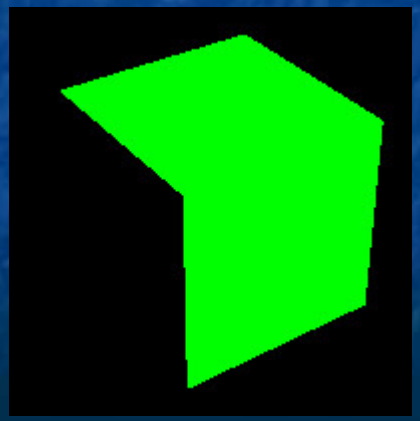


PointLight(color : Integer,
intensity : Float,
distance : Number,
decay : Float)

- Couleur de base de la source
- Puissance de la source
- Distance maximale d'effet de la source
- Décroissance de l'effet par rapport à la distance



Intensité = 1.0



Intensité = 10.0

Les sources de lumière (2)

- Les spots :
  - Sources ponctuelles
  - Éclairent dans un cône défini

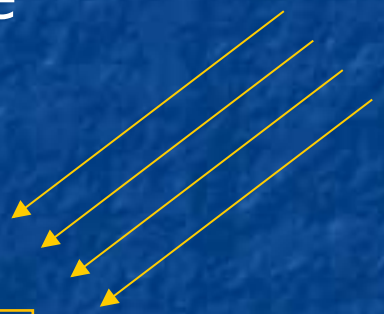
SpotLight(color : Integer,
intensity : Float,
distance : Float,
angle : Radians,
penumbra : Float,
decay : Float)

- Couleur de base de la source
- Puissance de la source
- Distance maximale d'effet de la source
- Demi-angle d'ouverture du spot
- Pourcentage de décroissance de l'intensité en s'éloignant de l'axe
- Décroissance de l'effet par rapport à la distance

- Attributs par défaut :
  - .position : (0, 1, 0)
  - .target : (0,0,0)

Les sources de lumière (3)

- Les sources directionnelles :
  - Éclairent la scène dans une direction donnée
    - Simulation d'une source à distance infinie



DirectionalLight(color : Integer,
intensity : Float)

Couleur de base de la source
Puissance de la source

- Attributs par défaut :
  - .position : (0, 1, 0)
  - .target : (0,0,0)

La scène

- Ensemble des informations nécessaires au rendu
  - Objets
  - Sources

Scene()

```
const scene = new THREE.Scene();
```

  - Ajout d'objets :

```
const objet = new THREE.Mesh(geometrie, materiau);
scene.add(objet);
```

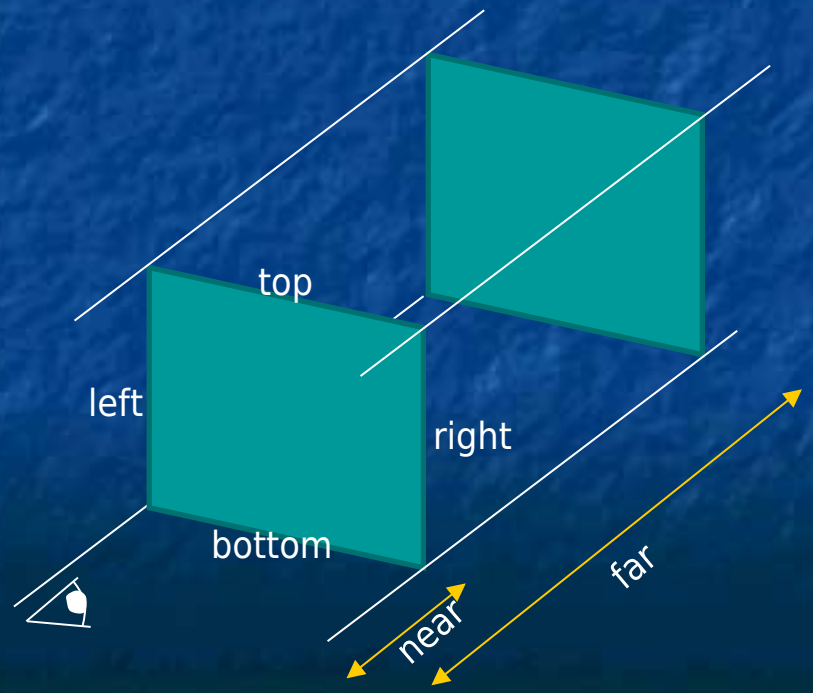
Les caméras (1)

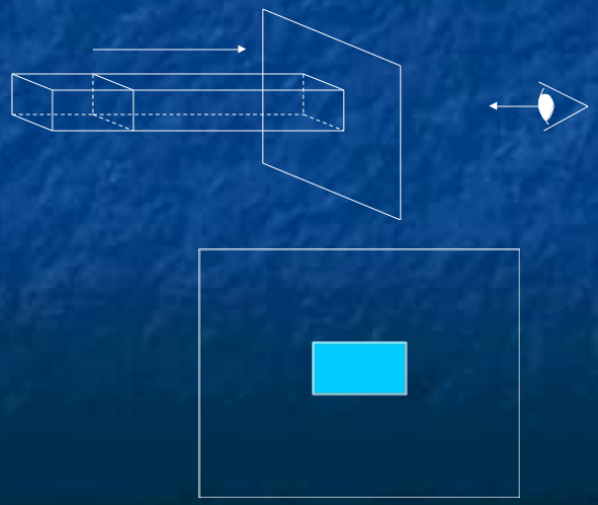
- Définissent les caractéristiques de projection de la scène sur l'image
  - Plusieurs types prédéfinis
  - 2 types principaux :
    - Projection orthographique
    - Projection perspective

Les caméras (2)

- Caméra orthographique

OrthographicCamera(left : Number,
right : Number,
top : Number,
bottom : Number,
near : Number,
far : Number)



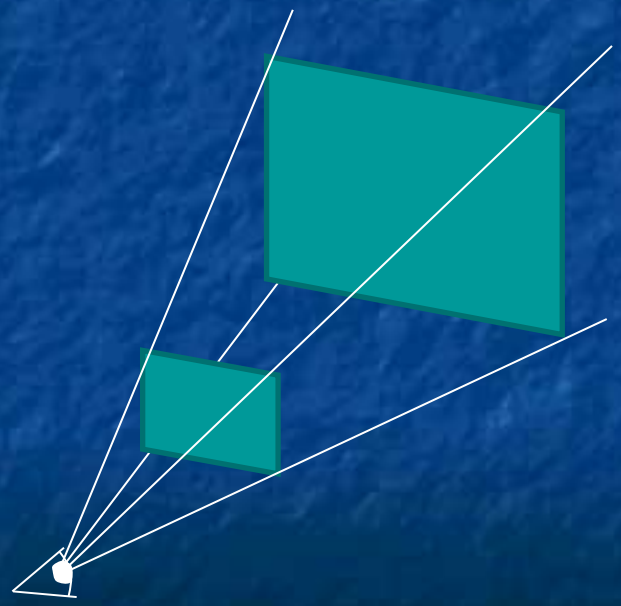


Les caméras (3)

- Caméra perspective

PerspectiveCamera(fov : Number,
aspect : Number,
near : Number,
far : Number)

fov	Angle d'ouverture
aspect	Rapport d'aspect largeur/hauteur
near	Distance du plan avant
far	Distance du plan arrière



- Valeurs par défaut :
  - Fov : 50
  - Aspect : 1
  - Near : 0.1
  - Far : 2000

Le "renderer" (1)

- C'est quoi ?
  - Objet en charge de dessiner une image
  - Plusieurs classes disponibles
    - Exemple :

WebGLRenderer(parameters : Object)

```
// Création du renderer
const renderer = new THREE.WebGLRenderer();
renderer.setSize( 500, 500 );
document.body.appendChild( renderer.domElement );
```

  - Ont besoin
    - D'une caméra
    - D'une scène
  - Passés à chaque nouveau rendu

```
renderer.render( scene, camera );
```

Le "renderer" (2)

- "animation"
  - Générer une nouvelle image tous les 1/60e de seconde
  - Fonction spéciale

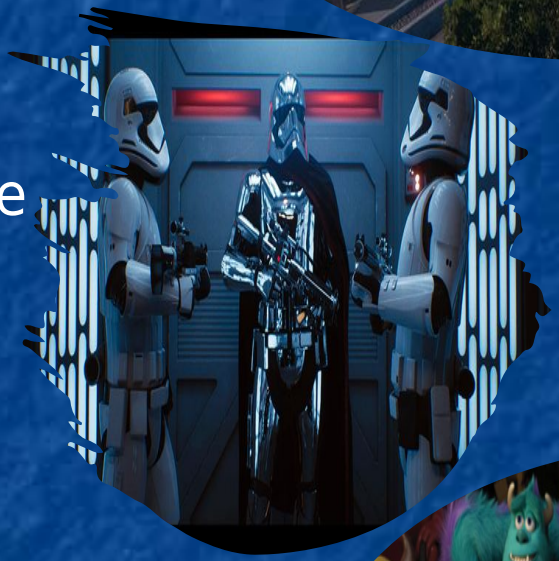
```
function animer () { // appelée à chaque besoin d'affichage-60fps)
    requestAnimationFrame( animer );

    .... // modification éventuelle de la scène

    renderer.render( scene, camera );
 };
```

Plan du cours

1. Introduction
2. Modélisation d’objets 3D
3. Modèle d'éclairage local
4. Rendu temps réel
5. Introduction à Three.js
6. Le ray tracing

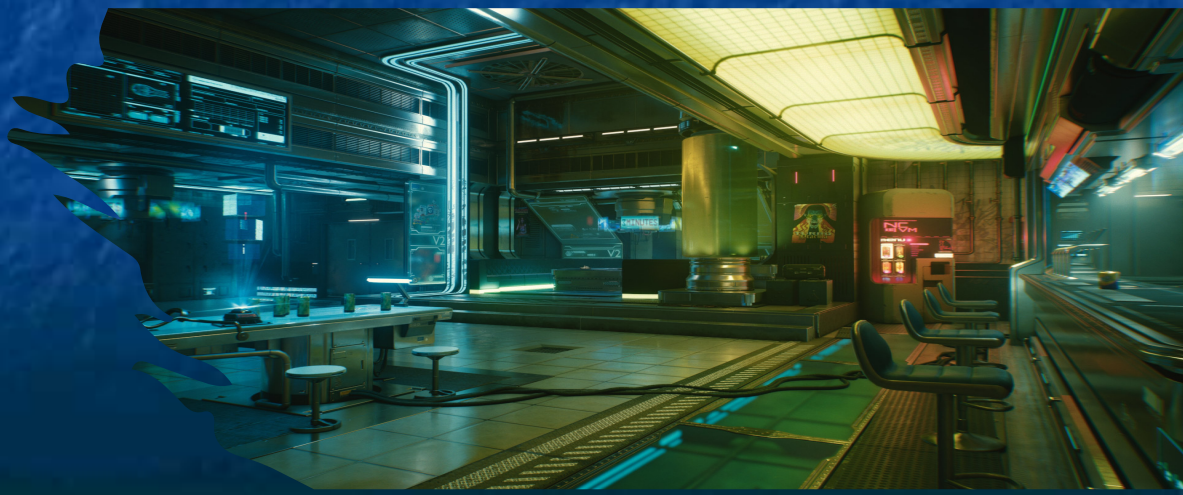
Le ray tracing

- Méthode de synthèse d'image "réaliste"
  - apparue en 1980
- Utilisée en production audio-visuelle
  - Monstre et compagnie 2 fin 2012
  - Rendu de la combinaison d'Iron Man
  - Plans SI derniers Star Wars
- Mais coûts de calcul très élevés
  - Plusieurs minutes à plusieurs heures



Le ray tracing

- Popularisation en cours via les jeux vidéo avec :
  - Du hardware spécialisé (Nvidia, AMD, ...) depuis 2019 (RTX 2000)
  - Des bibliothèques de développement (Nvidia Optix, Intel Embree, AMD FireRays, ...)
  - Moteurs de jeu standards (Unreal, Unity)



Principe (1/8)

Les mondes virtuels

Objets 3D

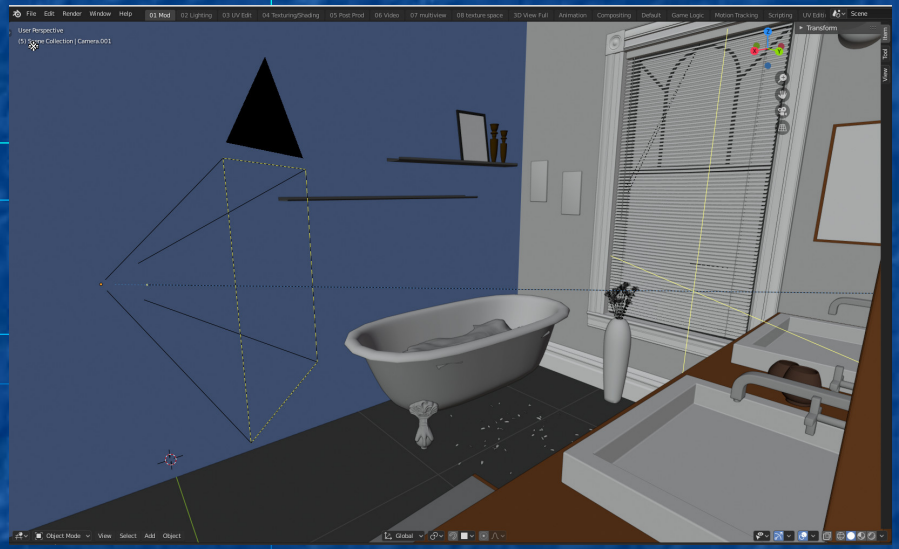
- Forme (géométrie)
- Apparence (matériaux)
- Position / animation

Lumières (sources)

- Forme / emission lumineuse
- Position / animation

Caméra(s)

- Résolution
- Ouverture
- Focus
- Sensibilité
- Etc.



Principe (2/8)

Vue du monde virtuel

Image

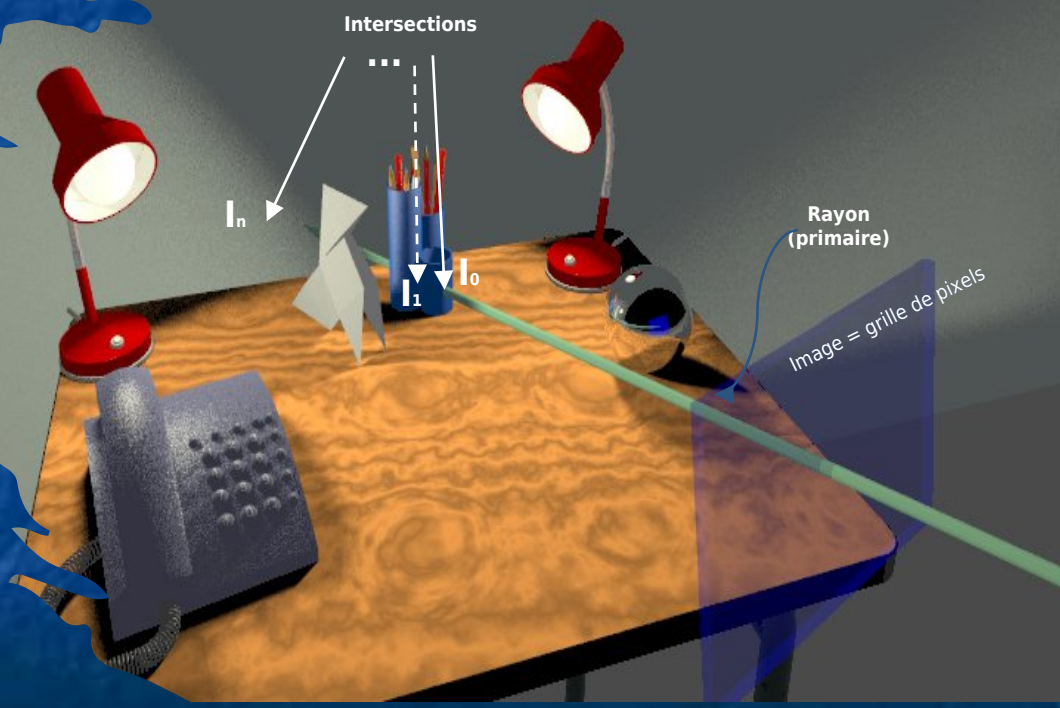
- Grille de pixel
- Dimensions = résolution de la caméra

Calcul de l'image

- Quel objet visible en chaque pixel ?
- Quelle couleur a cet objet ?

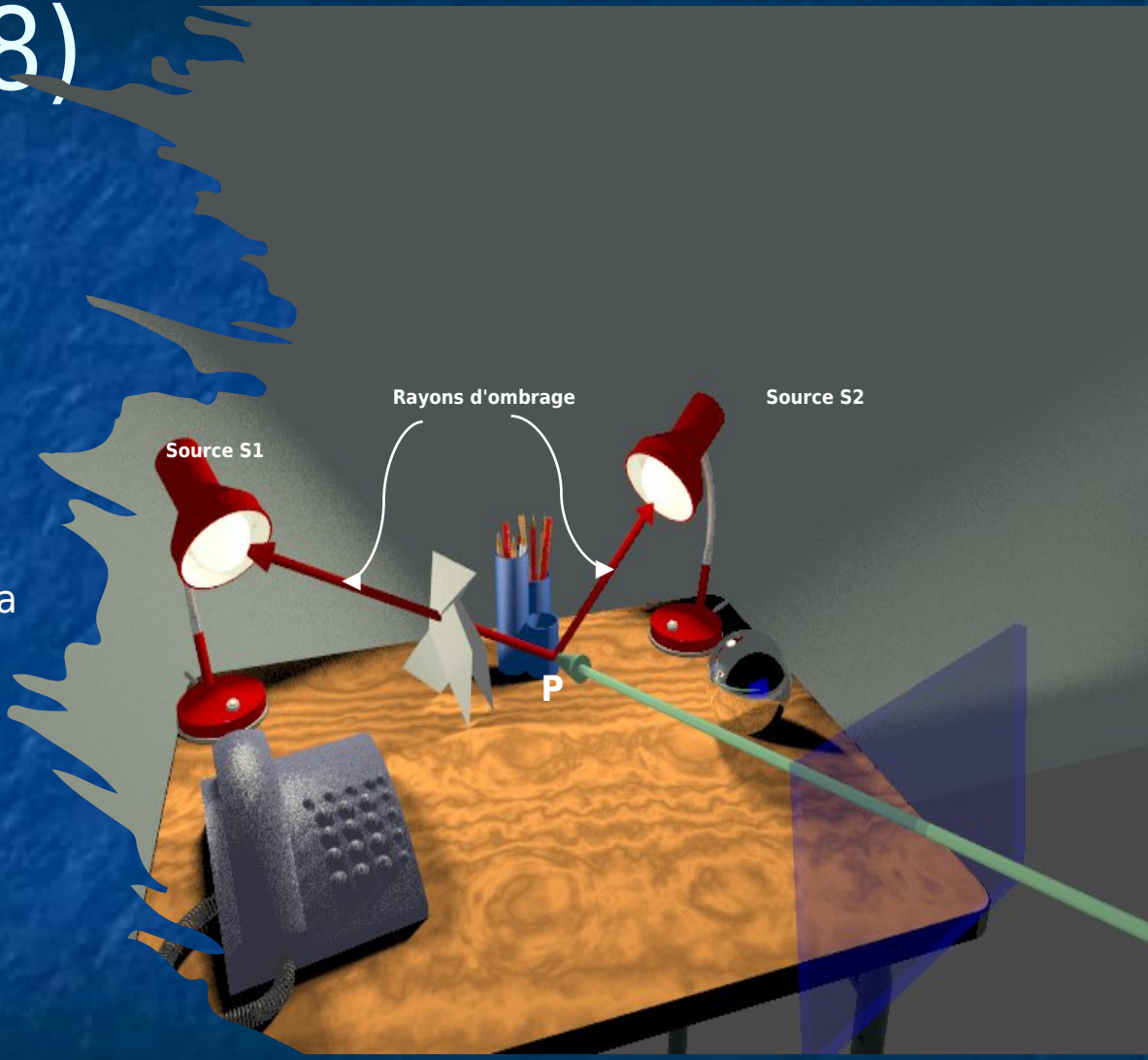
Principe (3/8)

- Objet visible ?
  - Lancer un rayon
    - depuis la caméra
    - à travers chaque pixel
  - Calculer les intersections avec chaque objet
  - Conserver l'intersection la plus proche
- Couleur du pixel ?
  - C(Pixel) = C(I_0)



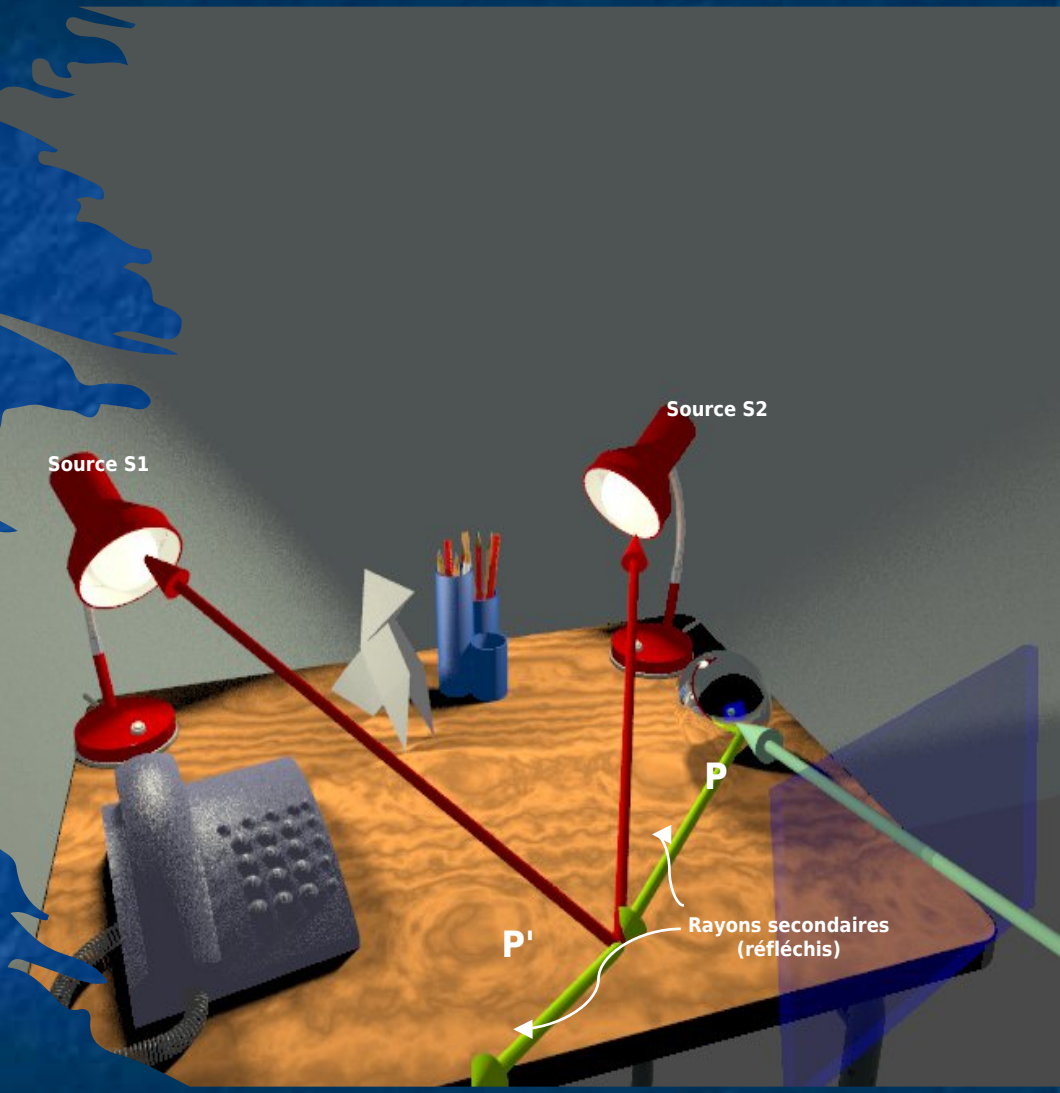
Principe (4/8)

- Eclairage direct du point visible
  - Rayon d'ombrage vers chacune des sources
  - Ajout de la contribution de la source si visible
  - Sinon ombre
- Couleur du pixel ?
  - $C(P) = \alpha.C(S_1) + \beta.C(S_2)$



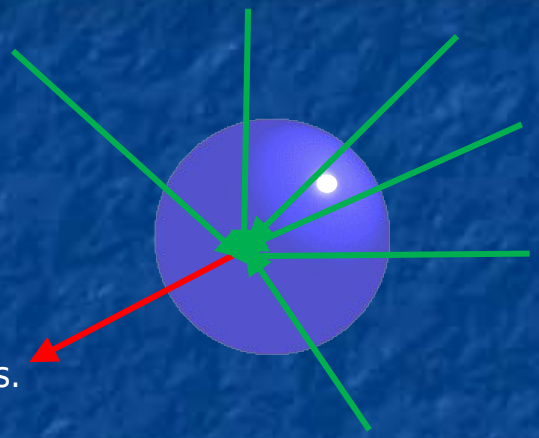
Principe (5/8)

- Eclairage indirect du point visible
  - Reflets ...
  - Envoi d'un rayon réfléchi
  - Évaluation de l'éclairage direct au point d'intersection trouvé
- Couleur du pixel ?
  - $C(P) = \alpha.C(S1)+\beta.C(S2)+ \gamma.C(P')$
  - $C(P') = \alpha'.C(S1)+\beta'.C(S2)+ ...$



Principe (6/8)

La réflexion



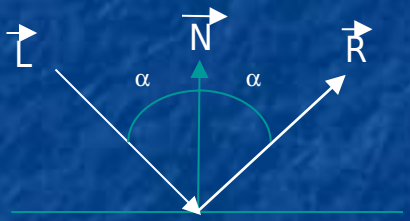
Objets brillants

- Réfléchissent la lumière de manière directionnelle
- Miroir, métaux, peintures brillantes, …
- Apparitions de reflets

$$\vec{R} = 2\vec{N}.(\vec{N}.\vec{L}) - \vec{L}$$

Quelle(s) direction(s) choisir ?

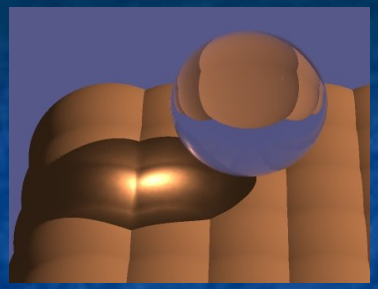
- Simplification : une seule direction
- Choix de la direction spéculaire parfaite



Matériaux mats :

- Pas de réflexion
- Estimation de l'ensemble de l'éclairage incident

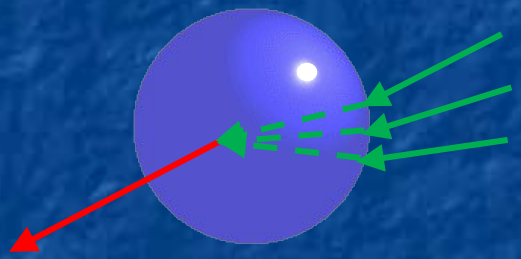
Principe (7/8)



La transparence

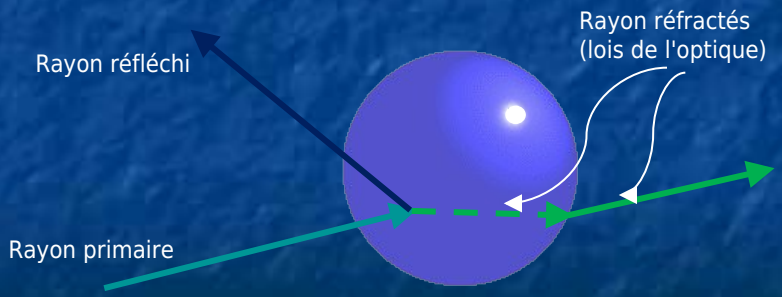
Objets non opaques

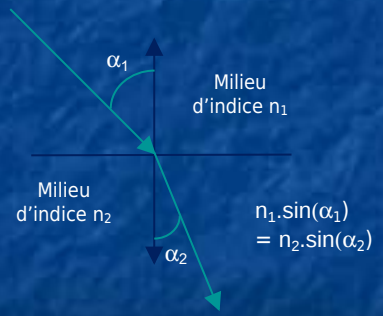
- De la lumière traverse l'objet
- Origines multiples (caustiques)



Quelle(s) direction(s) choisir ?

- Simplification : une seule direction
- Choix selon loi de la réfraction de Snell-Descartes





Principe (8/8)

Un processus récursif…

À chaque intersection

- Lancer de rayons d'ombrage
- Lancer d'un rayon réfléchi
- Lancer (éventuel) d'un rayon transmis

Arrêt du processus

- Matériau de l'objet trouvé mat
- Sinon
  - Profondeur maximale
  - Contribution lumineuse trop faible
  - Panachage des deux