

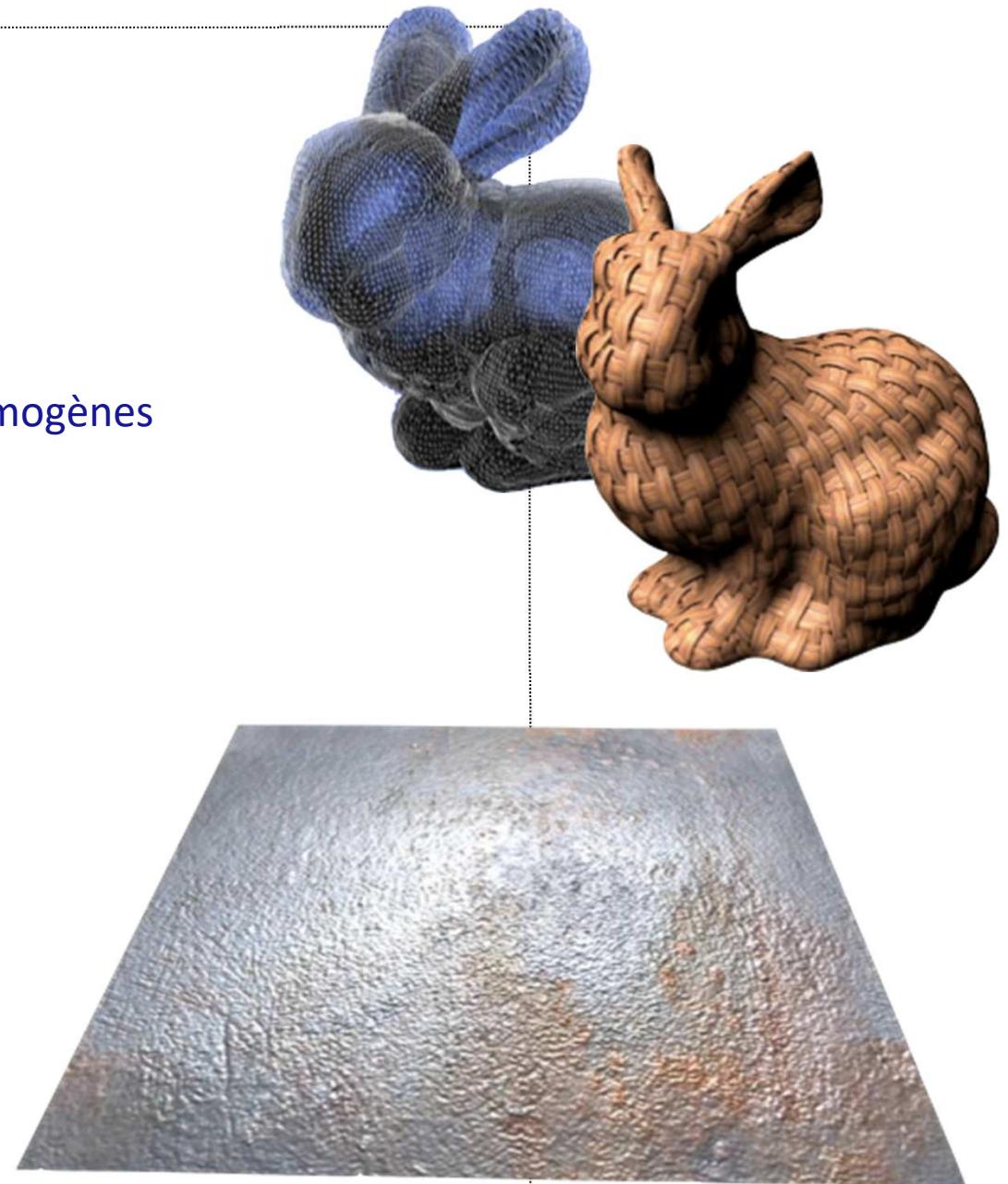
# Cours de Synthèse d'Images

M2 Informatique

Daniel Meneveaux

# Plan du cours

- **Introduction**
  - ▶ Principes généraux
  - ▶ Applications
- **Modélisation géométrique**
  - ▶ Repère 3D et objets 0D,1D,2D,3D
  - ▶ Transformations et coordonnées homogènes
- **Visualisation**
  - ▶ Le “tampon de profondeur”
  - ▶ Le “lancer de rayons”
- **Rendu réaliste**
  - ▶ La lumière, les ombres
  - ▶ Les matériaux, textures
- **Problèmes**
  - ▶ Simulation d'éclairage
  - ▶ Aliassage, etc.



# Introduction - principes

---

## ● Construction d'une scène 3D

- ▶ Modélisation géométrique (forme des objets)
- ▶ Modélisation radiométrique (sources lumineuses)
- ▶ Modélisation photométrique (matériaux des objets)

## ● Animation

- ▶ Modèles physiques
- ▶ Gestion du mouvement

## ● Calculs de simulation d'éclairage

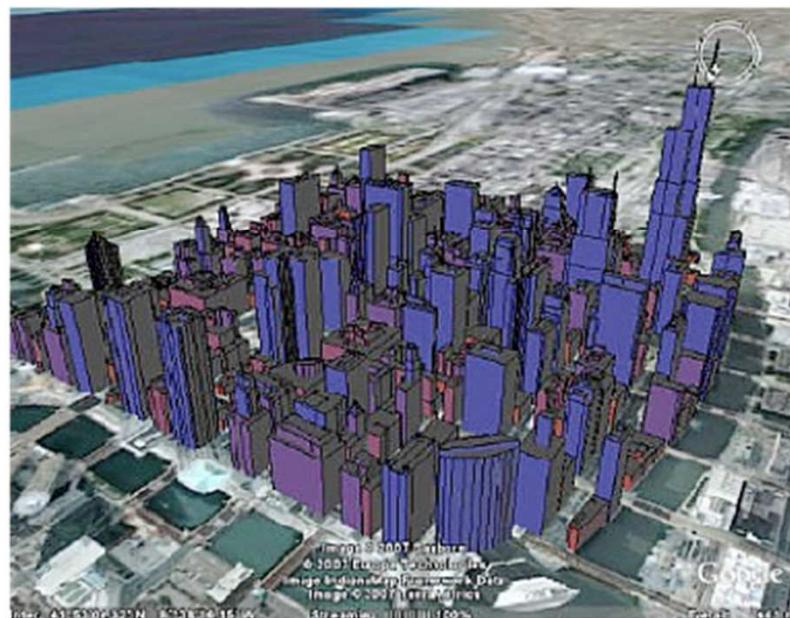
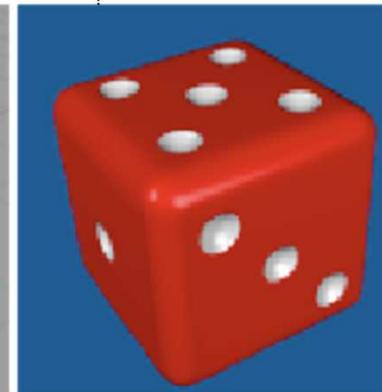
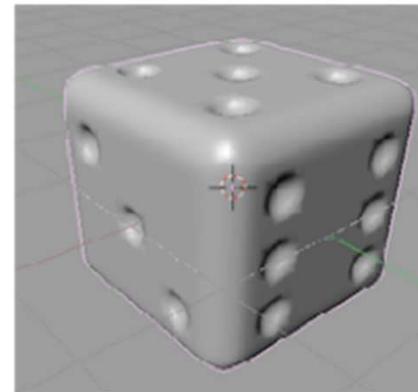
- ▶ Inter-réflexions lumineuses

## ● Construction d'une image, deux grandes familles

- ▶ Tampon de profondeur
- ▶ Lancé de rayons

# Modélisation géométrique

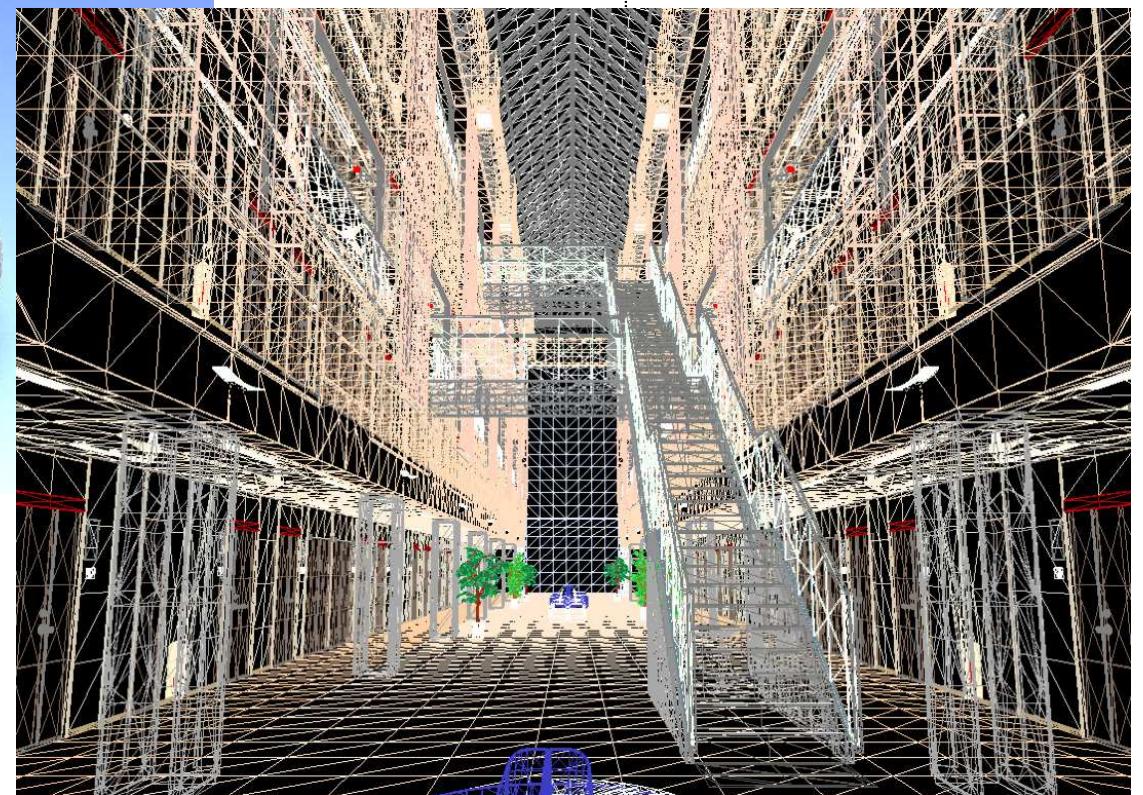
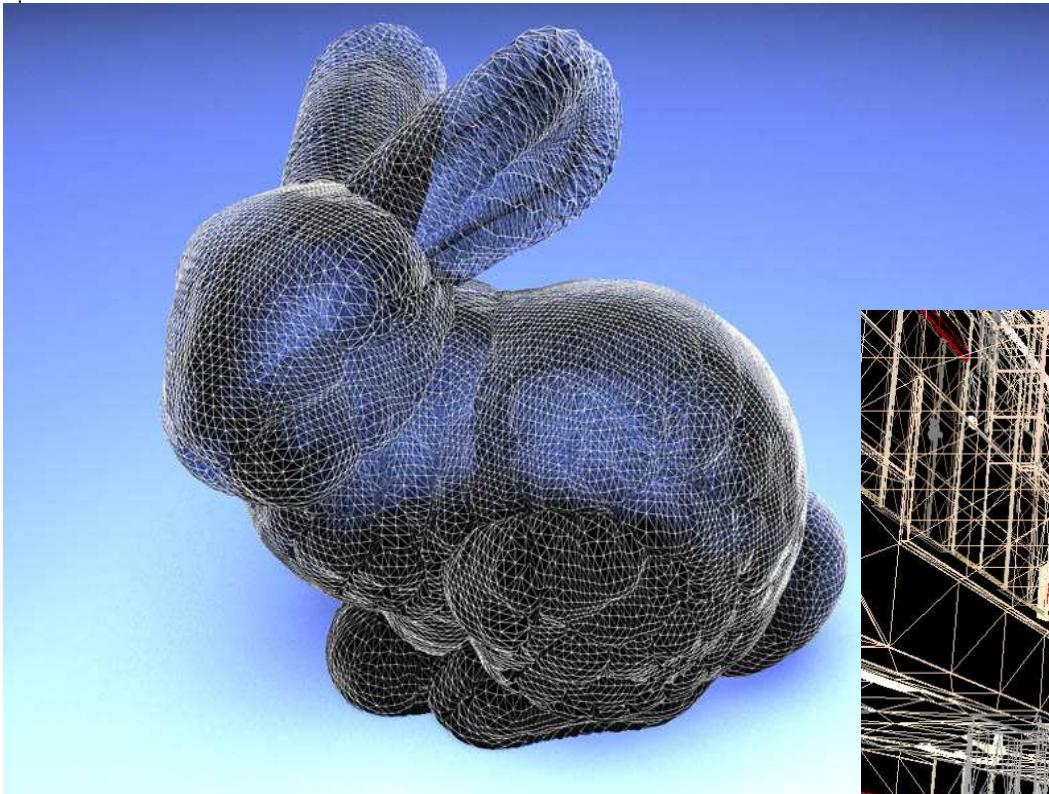
## ○ Primitives opérations de construction / assemblage



# Mais...

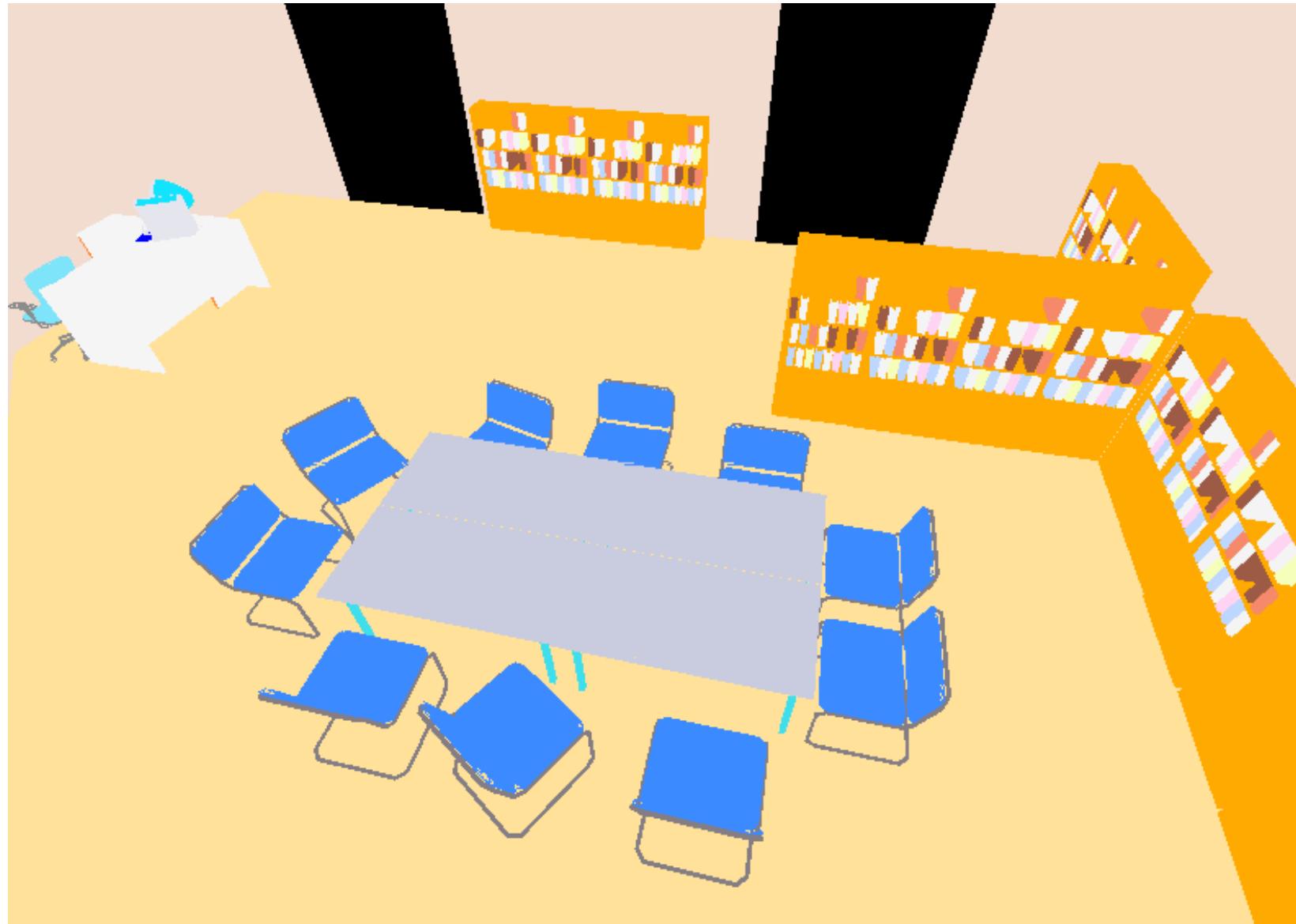


## Seulement la géométrie



# La “couleur” (?) seule

---



# Et il faudrait plutôt ça

---



## ○ Associer à chaque objet

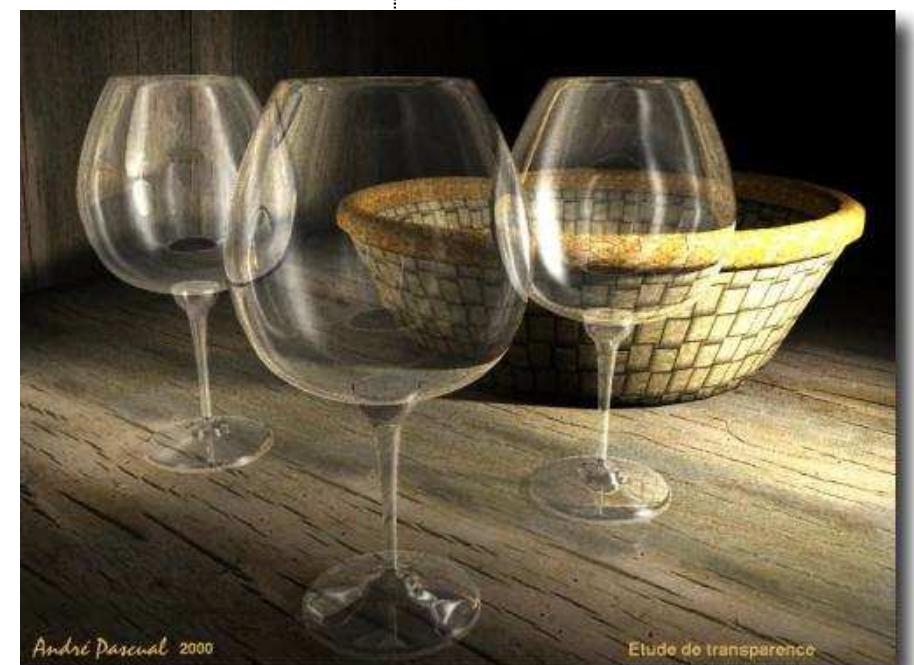
- ▶ une... “couleur” ???
- ▶ une... “texture” ???
- ▶ À quoi cela correspond-il ?

## ○ Effets lumineux

- ▶ Surfaces transparentes
- ▶ Surfaces spéculaires (brillantes)
- ▶ Milieux semi-transparents
- ▶ Objets anisotropes (tissus)
- ▶ etc.

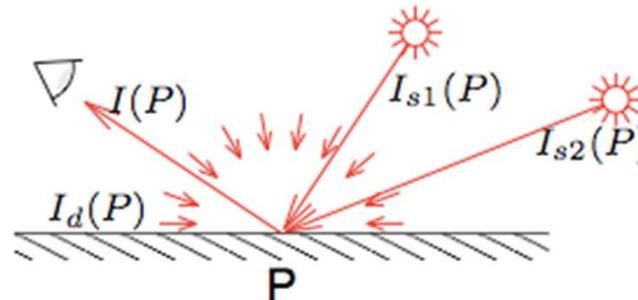
## ○ Donc définir

- ▶ des sources de lumière
- ▶ les matériaux des objets



## ● Eclairement "direct"

- ▶ Directement des sources lumineuses



## ● Eclairement indirect

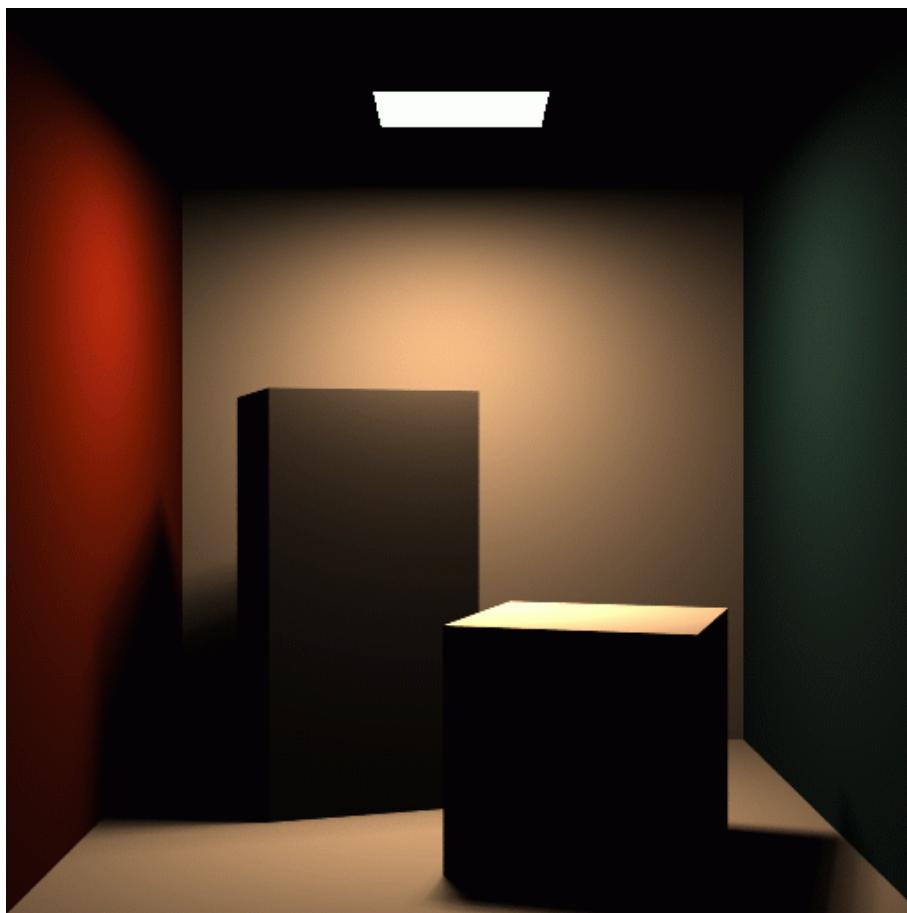
- ▶ inter-réflexions lumineuses
- ▶ spécularité
- ▶ transparence
- ▶ simulation d'éclairage...

Nécessite encore d'autres calculs !

Nous en verrons une partie

# Direct et indirect

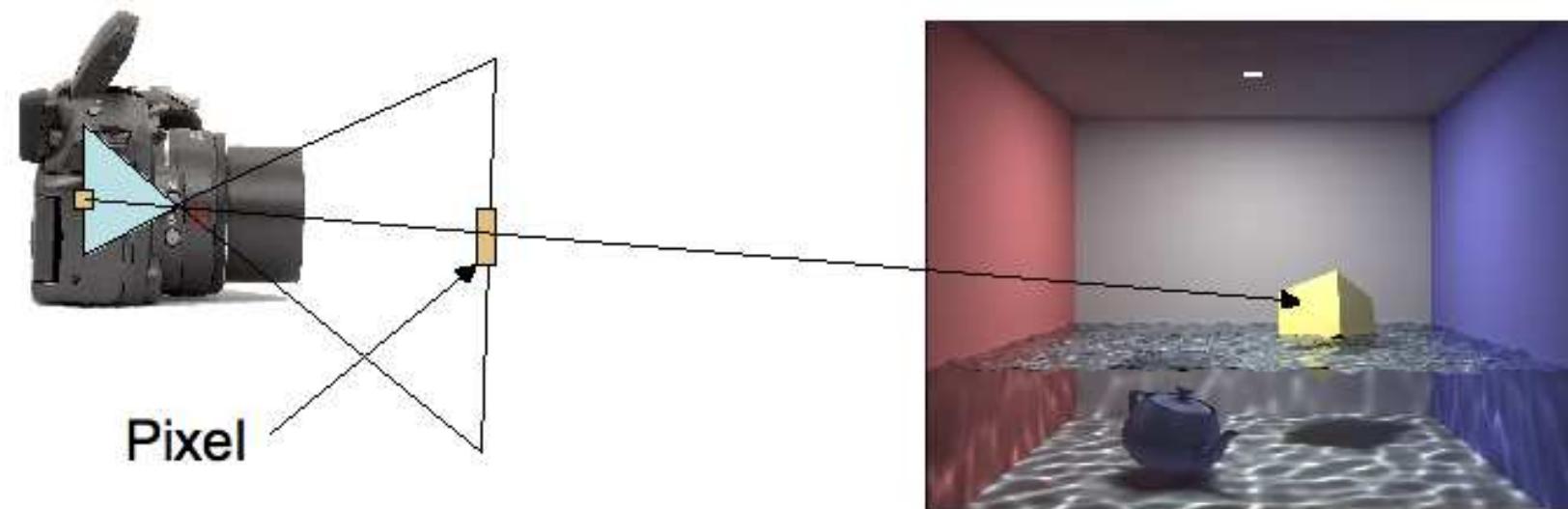
- Direct seulement (gauche)
- Direct et indirect (droite)



# Et les images ?

- ➊ Placer une caméra (virtuelle) dans la scène (virtuelle)
- ➋ Calculer la valeur RVB de chaque pixel
  - ▶ Projeter la géométrie
  - ▶ Prendre en compte les transferts lumineux

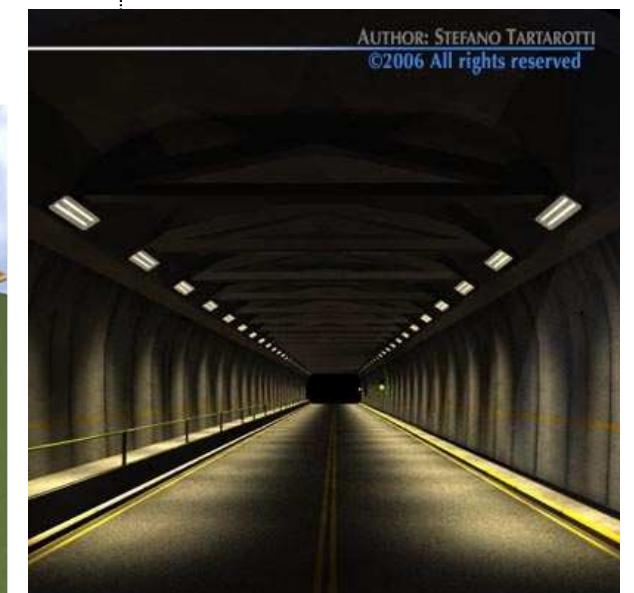
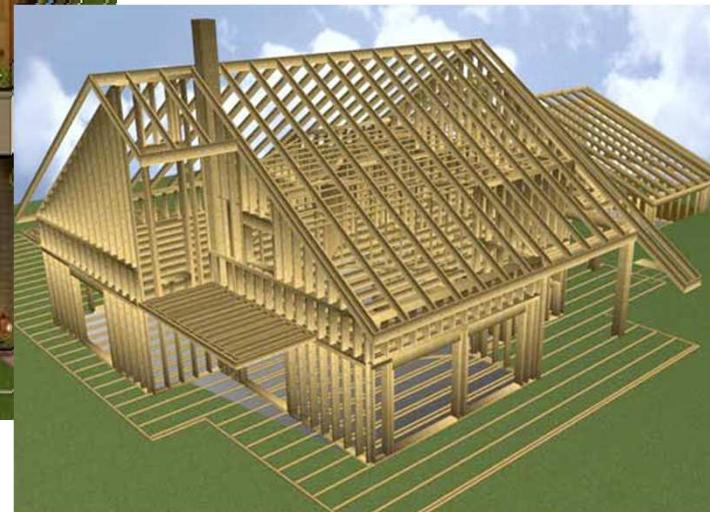
L'image n'est PAS (!!!) virtuelle...



# Applications : Architecture

## Architecture

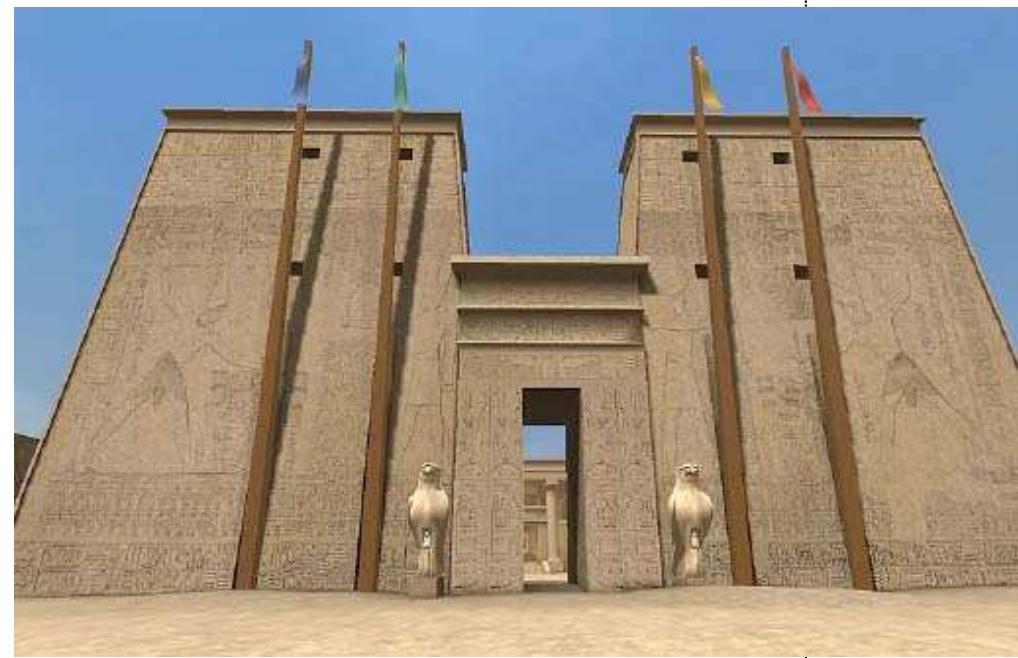
- ▶ Modélisation avant la construction
- ▶ Prévoir le placement de sources lumineuses
- ▶ Intérieur : bâtiments, tunnels, etc.
- ▶ Extérieur : environnements urbains



# Applications : Archéologie

## ○ Archéologie

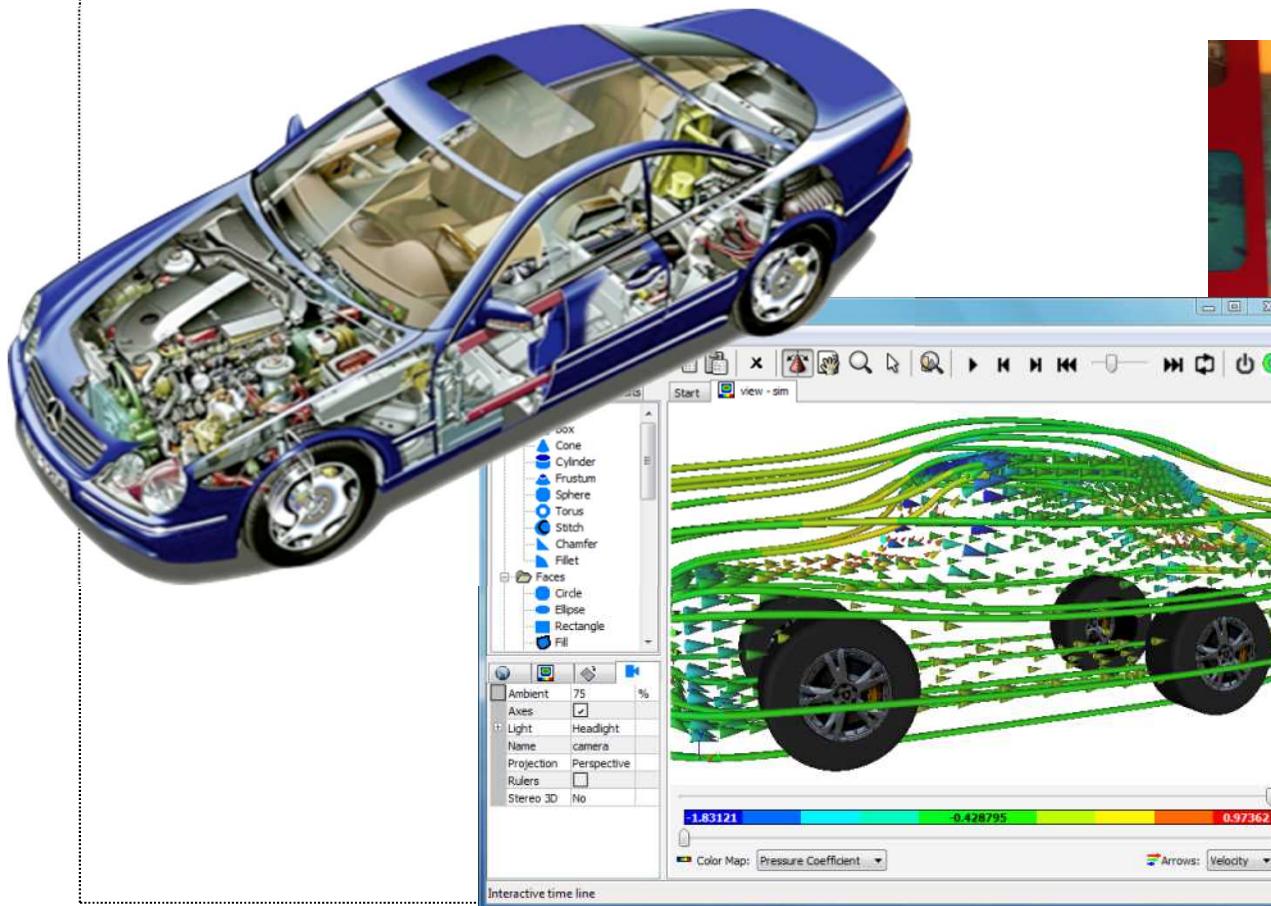
- ▶ Reconstruction de sites
- ▶ Hypothèses historiques à vérifier
- ▶ Éclairage des édifices (lumière naturelle)



# Applications : Automobile

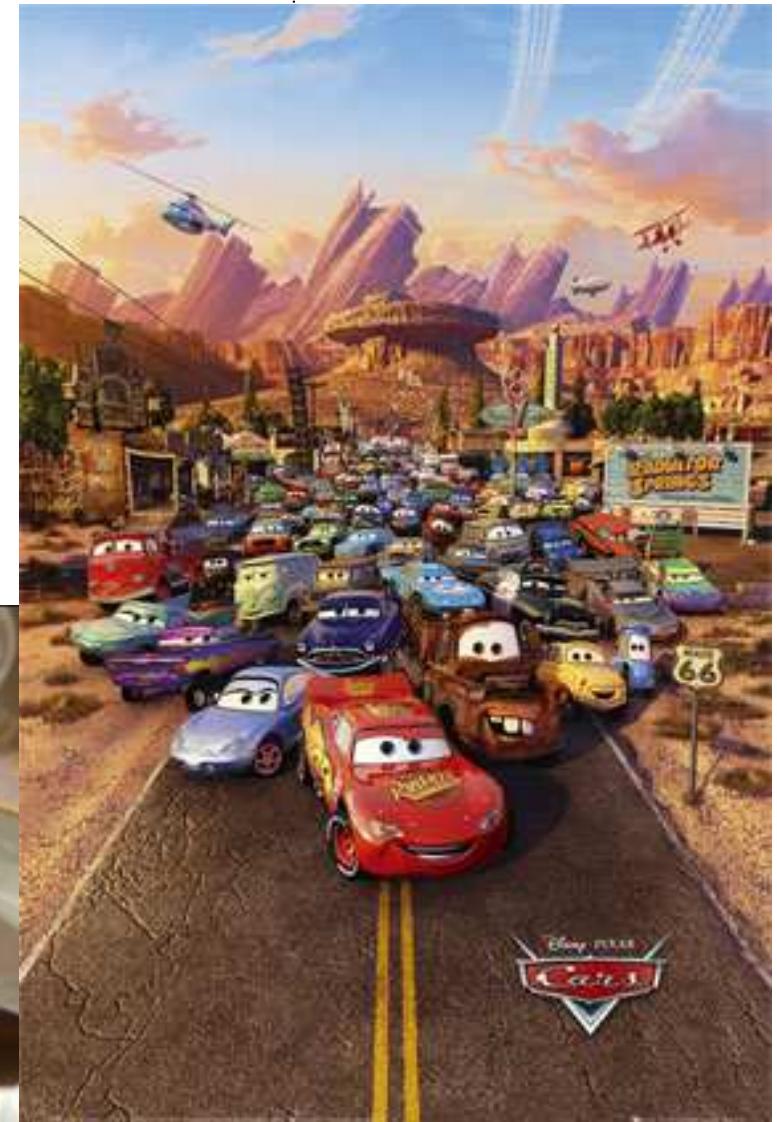
## ○ Modélisation 3D, rendu réaliste

- ▶ carrosseries de voitures
- ▶ peintures
- ▶ environnements extérieurs, simulateurs, pluie, etc.



# Et le reste...

- Simulateurs : auto, avion, etc.
- Jeux vidéo 3D (cartes graphiques)
- Films d'animation
- Publicité, communication



## ● Modèles 3D et opérations

- ▶ Représentation des objets
- ▶ Transformations de base

## ● Modèles de matériaux

- ▶ Diffus, spéculaires
- ▶ Quelques modèles simples (Lambert, Phong)

## ● Construction d'une image : les bases

- ▶ Principe du tampon de profondeur
- ▶ Principe du tracé de rayons

## ● Quelques exercices

- ▶ Au fur et à mesure
- ▶ N'hésitez pas à poser des questions !!!

- Introduction
  - ▶ Principes généraux
  - ▶ Applications
- **Modélisation géométrique**
  - ▶ Repère 3D et objets 0D,1D,2D,3D
  - ▶ Transformations et coordonnées homogènes
- Visualisation
  - ▶ Le “tampon de profondeur”
  - ▶ Le “lancé de rayons”
- Rendu réaliste
  - ▶ La lumière, les ombres
  - ▶ Les matériaux, textures
- Aller plus loin...
  - ▶ Simulation d'éclairage
  - ▶ Aliassage, etc.

# Modélisation géométrique

## ① Définir des objets dans l'espace

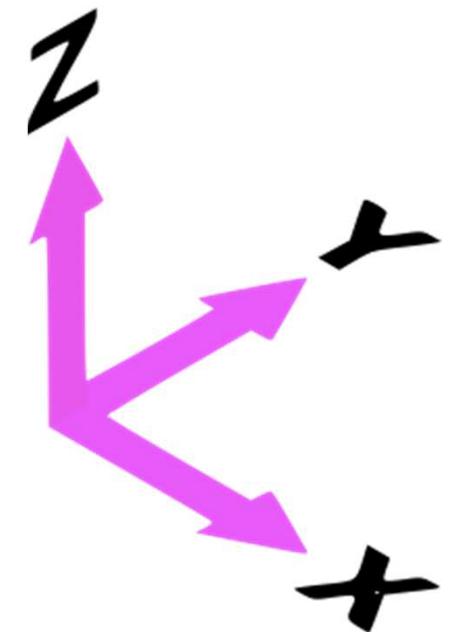
- ▶ représente la “scène” ou l’“environnement”
- ▶ définition d'un repère global

## ② Types d'objets

- ▶ 0D (sommets)
- ▶ 1D (arêtes, lignes brisées, cercles, etc.)
- ▶ 2D (triangles, polygones, disques, sphères, etc.)
- ▶ 3D (cônes, boules, cubes, etc.)

## ③ Placement dans l'espace

- ▶ coordonnées 3D ( $x,y,z$ )
- ▶ par convention dans ce cours :  $z$  est l'altitude
- ▶ unité pour ce cours : le mètre



# Avantages/inconvénients

---

## ● Maillages

- ▶ représentation uniforme (tous les objets pareils)
- ▶ mais pas toujours pratiques à manipuler
- ▶ objets complexes => beaucoup d'éléments

## ● Surfaces “courbes”

- ▶ représentation plus douce des courbes
- ▶ mais pas toujours pratiques à manipuler !
- ▶ affichage => maillage

## ● Objets de base 3D (sphères, cones, cylindres)

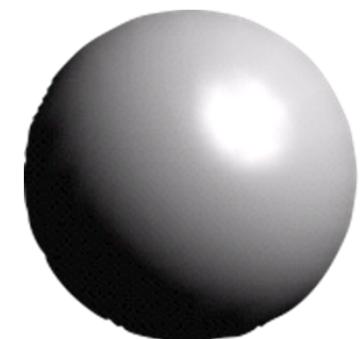
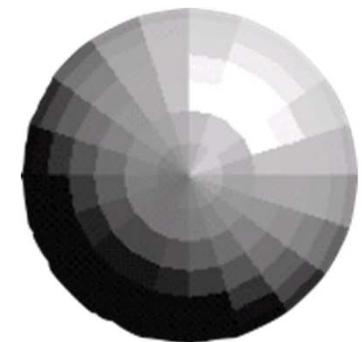
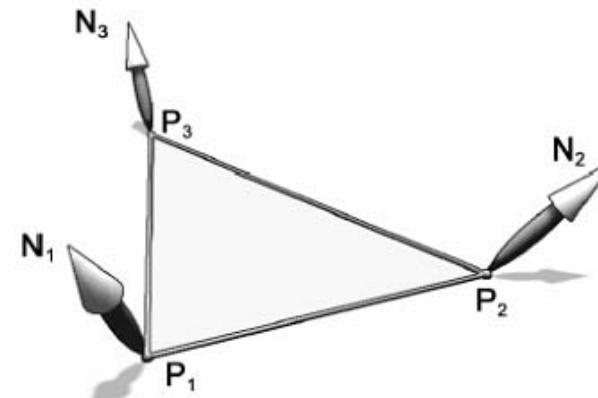
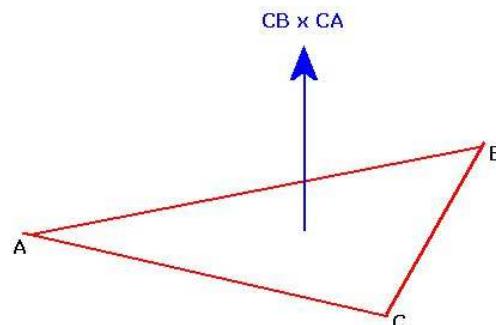
- ▶ pas très souvent dans la nature
- ▶ pratiques pour faire des tests
- ▶ finalement assez peu utilisés

En général on a surtout des maillages pour la visualisation

# Orientation des objets

## ○ Les objets sont orientés

- ▶ normale à la surface



## ○ Exercice (traduction orale nécessaire !) :

- ▶ Calculez la normale d'un triangle
- ▶ Calculez la normale d'une sphère

## ○ Rappels sur les produits entre vecteurs

- ▶ Produit scalaire
- ▶ Produit vectoriel
- ▶ Produit par un scalaire
- ▶ etc.

Tableau

## ● Besoins de placer les objets dans l'environnement

- ▶ déplacement : translation
- ▶ orientation dans l'espace : rotation
- ▶ taille de l'objet : homothétie

## ● Pour la translation

- ▶ déplacer un sommet ?
- ▶ déplacer une sphère ?  
vecteur de translation

Tableau

## ● Pour la rotation

- ▶ autour d'un axe Ox, Oy ou Oz
- ▶ déplacer un sommet  
angle(s) de rotation

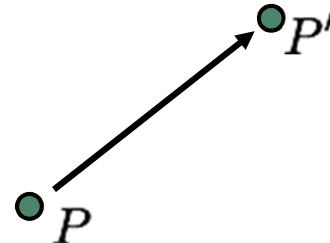
## ● Pour l'homothétie

- ▶ facteur d'homothétie
- ▶ attention aux effets de bord !

## ○ Vecteur de translation

$$T \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix}$$

$$P' = P + T = \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} + T \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} = \begin{pmatrix} x_p + x_t \\ y_p + y_t \\ z_p + z_t \end{pmatrix}$$



## ○ Exercices :

- ▶ Déplacer le point (0,0,0) le long du vecteur (1,1,1)
- ▶ Déplacer le point (1,1,1) le long du vecteur (-1,-1,-1)
- ▶ idem avec P(1.3,2.0,1.7) et v(0.2,0.5,0.3)
  
- ▶ Comment appliquer une translation à une sphère
- ▶ Idem pour un cube ?

Tableau

Tableau

- Pour changer la taille d'un objet
- Exemple de la sphère : appliquer au rayon
- Pour des maillages : à chaque sommet
- Attention :

- ▶ homothétie d'un sommet seul ?  $P' = h.P$
- pas vraiment de sens
- ▶ pour un maillage, on a pour chaque sommet P :

$$P' = h.P = h \cdot \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} h.x_p \\ h.y_p \\ h.z_p \end{pmatrix}$$

- Exercices :
  - ▶ homothétie de facteur 2 à un carré en 2D, centré en 0
  - ▶ idem pour une boîte min/max
  - ▶ Quel est le problème ?

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}$$

Tableau

# Homothétie (suite)

## Questions

- ▶ Que se passe-t-il pour les maillages centrés en (0,0,0)
- ▶ Et les autres ?
- ▶ Comment faire pour que le centre gravité reste au même endroit ?

$$\begin{pmatrix} x_1 \\ x_1 \\ x_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ x_2 \\ x_2 \end{pmatrix}, \dots, \begin{pmatrix} x_n \\ x_n \\ x_n \end{pmatrix}$$

Centre de gravité G

$$G = \begin{pmatrix} x_g \\ y_g \\ z_g \end{pmatrix} = \begin{pmatrix} (x_1 + x_2 + \dots + x_n)/n \\ (y_1 + y_2 + \dots + y_n)/n \\ (z_1 + z_2 + \dots + z_n)/n \end{pmatrix}$$

Tableau

# Homothétie (encore)

## ➊ Pour éviter que l'objet se “déplace”

- ▶ Appliquer une translation pour le centrer en (0,0,0)
- ▶ Réaliser l'homothétie
- ▶ Appliquer la translation inverse pour le replacer

$$P' = T^{-1} \cdot H \cdot T(P)$$

- ▶ Avec :

$$T = (-x_g, -y_g, -z_g)$$

$$T^{-1} = (x_g, y_g, z_g)$$

$H$  : facteur  $\lambda$

## ➋ Exercice :

- ▶ développez les calculs
- ▶ donnez la formule complète en fonction de  $\lambda$  et  $G$

## ➌ Et si on a un facteur par coordonnée ?

Tableau

Tableau

● Rotation d'angle  $\alpha$  autour d'un axe

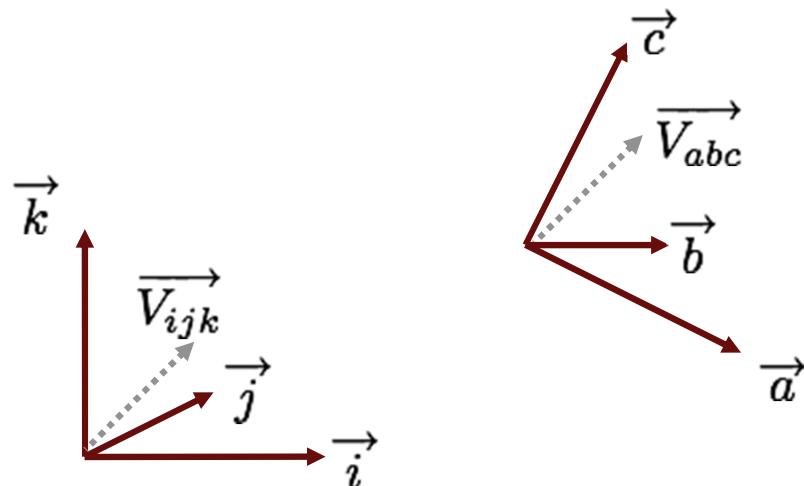
- ▶ Se trouve partout sur internet, cours de Licence
- ▶ On ne l'utilise pas ici pour le cours

● Plutôt besoin de changements de repères

- ▶ Pour la visualisation
- ▶ Rotation de la caméra

● Trouver la matrice M telle que  $M \cdot \overrightarrow{V_{ijk}} \rightarrow \overrightarrow{V_{abc}}$

- ▶ Très facile !
- ▶ Si on connaît les vecteurs  $\vec{a}$   $\vec{b}$   $\vec{c}$



Tableau

## ① Les 3 vecteurs (base orthonormée)

$$\vec{a} = \begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} \quad \vec{b} = \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} \quad \vec{c} = \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix}$$

$$M_r = \begin{pmatrix} x_a & x_b & x_c \\ y_a & y_b & y_c \\ z_a & z_b & z_c \end{pmatrix}$$

## ② Encore mieux

$$M^{-1} = M^t$$

$$M^{-1} = \begin{pmatrix} x_a & y_a & z_a \\ x_b & y_b & z_b \\ x_c & y_c & z_c \end{pmatrix}$$

Tableau

Note : cela correspond aussi exactement à une rotation pour faire tourner le vecteur !

# Rotation (encore)

## Pour vérifier

- ▶ appliquez la matrice M aux 3 vecteurs

$$\vec{i} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{j} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ et } \vec{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

- ▶ En principe

$$M \cdot \vec{i} = \vec{a}$$

$$M \cdot \vec{j} = \vec{b}$$

$$M \cdot \vec{k} = \vec{c}$$

Tableau

- ▶ Et idem dans l'autre sens !

$$M^{-1} \cdot \vec{a} = \vec{i}$$

$$M^{-1} \cdot \vec{b} = \vec{j}$$

$$M^{-1} \cdot \vec{c} = \vec{k}$$

Tableau

# Rotation (toujours)

## ➊ Attention néanmoins

- ▶ La trigo reste intéressante dans certains cas
- ▶ Vecteurs à partir de directions
- ▶ Quelques bases (le reste vient tout seul)

$$z_v = \cos(\theta)$$

$$x_v = \cos(\phi).[\sin(\theta)]$$

$$y_v = \sin(\phi).[\sin(\theta)]$$

Tableau

# Homogénéisation

---

● **Pas très pratique à combiner...**

- ▶ Translation : vecteur
- ▶ Homothétie : facteur
- ▶ Rotation : matrice

● **Peut-on tout mettre sous l'une seule des 3 formes ?**

- ▶ Faciliter les combinaisons
- ▶ Eventuellement précalculer la transformation
- ▶ L'appliquer à plusieurs objets à la fois ( $M=M.N.P.Q$ )

● **Seule solution : les matrices... (il y a un 'mais')**

● **Exercices**

- ▶ Décrivez une translation à l'aide d'une matrice  $3 \times 3$
- ▶ Qu'est-ce qui cloche ?
- ▶ Idem pour l'homothétie
- ▶ Que faudrait-il ?

# Homogénéisation (suite)

## ➊ Solution : les matrices 4x4 (avec des vecteurs 4D)

### ➋ Exercices

- ▶ Trouvez une matrice de translation M (4x4) telle que :

$$M \cdot \begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix} = \begin{pmatrix} x_v + x_t \\ y_v + y_t \\ z_v + z_t \\ 1 \end{pmatrix}$$

- ▶ Idem pour une matrice de rotation
- ▶ Idem pour une matrice d'homothétie (facteur h)

ATTENTION ! La matrice ne doit pas contenir les termes

$x_v \ y_v \ z_v$

- ▶ Généralisez la méthode (facile, maintenant)

# Rappels sur les matrices

## ○ Homogénéisation

- ▶ vecteurs 3D => coordonnées homogènes 4D

## ○ Produit matrice - vecteur

- ▶ 3x3, puis 4x4

Tableau

## ○ Produit matrice - matrice

- ▶ 3x3, puis 4x4

## ○ Produit scalaire

- ▶ 3D, puis 4D

## ○ Produit vectoriel

- ▶ 3D

## ○ Scalaire - vecteur

Coordonnées homogènes toujours ok ?

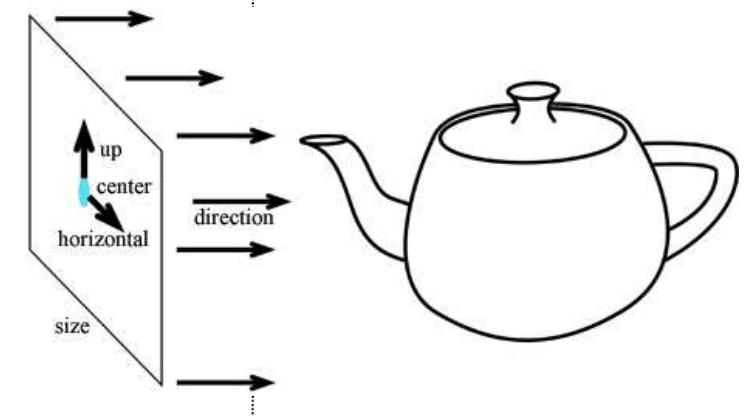
- Introduction
  - ▶ Principes généraux
  - ▶ Applications
- Modélisation géométrique
  - ▶ Repère 3D et objets 0D,1D,2D,3D
  - ▶ Transformations et coordonnées homogènes
- **Visualisation**
  - ▶ Le “tampon de profondeur”
  - ▶ Le “lancé de rayons”
- Rendu réaliste
  - ▶ La lumière, les ombres
  - ▶ Les matériaux, textures
- Aller plus loin...
  - ▶ Simulation d'éclairage
  - ▶ Aliassage, etc.

# Systèmes de projection

## ○ Projection orthographique

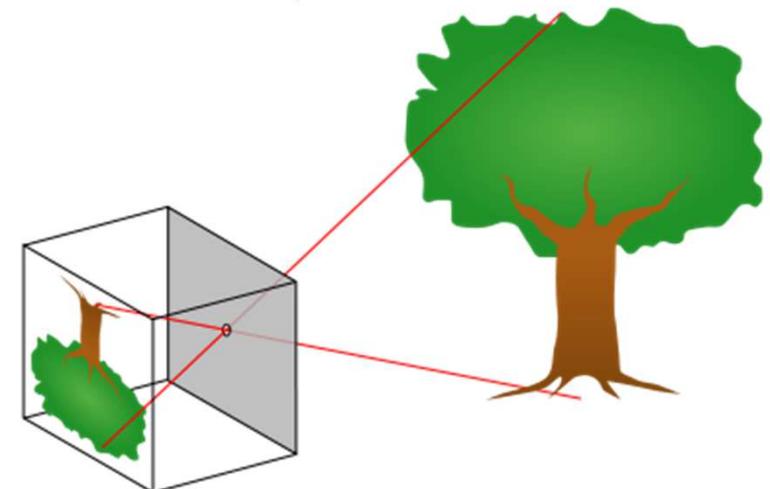
- ▶ pour la modélisation  
coupes, déplacer les objets, etc.
- ▶ pour la visualisation de plans  
vues de dessus, de profil, etc.
- ▶ mais déformation des objets  
par rapport à ce que nous voyons

Tableau



## ○ Projection perspective

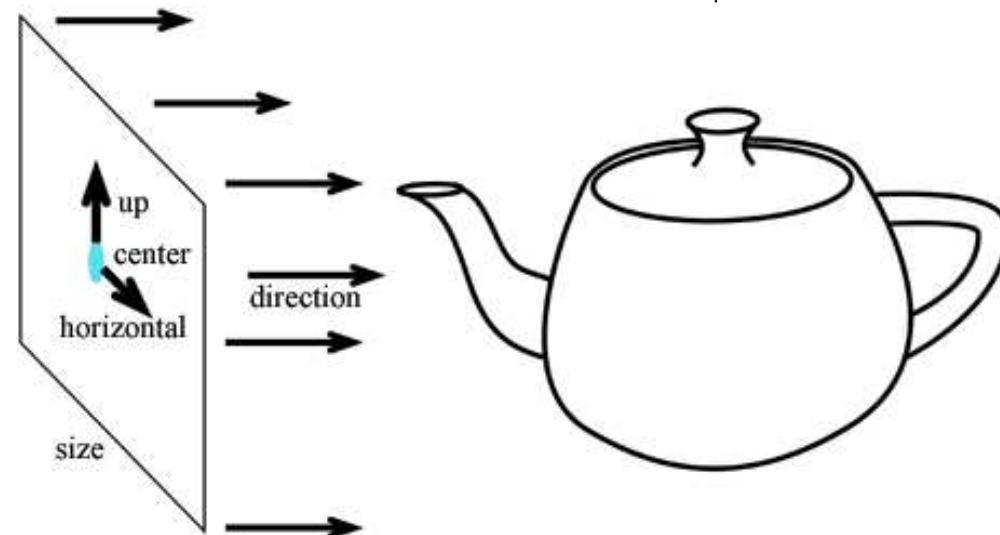
- ▶ correspond à notre vision (perspective)
- ▶ projection photographique
- ▶ principe de la caméra "trou d'aiguille"



# Projection orthographique

## ○ Projection parallèle

- ▶ direction perpendiculaire au plan image



## ○ Exercice

- ▶ plan projectif placé en 0, direction y
- ▶ définissez la matrice de projection  
pour cela, utilisez un point  $x,y,z$   
donnez leur projection sur le plan  
déduisez la matrice

Tableau

# Projection perspective

## ○ Projection “trou d'aiguille”

- ▶ modèle sténopé
- ▶ la lumière passe seulement par un petit trou

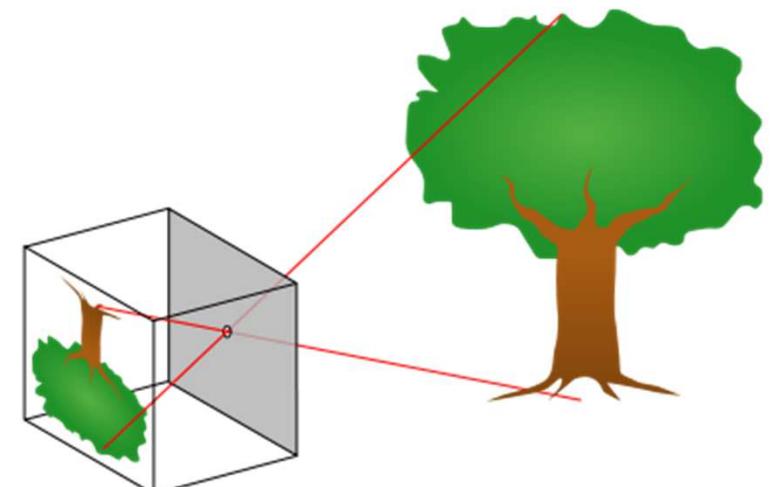
## ○ Plusieurs paramètres

- ▶ distance entre le trou et le capteur (focale)
- ▶ taille du trou (diaphragme)
- ▶ temps d'exposition (temps de pose)

## ○ Appareils-photos

- ▶ principe identique (lentilles)
- ▶ trou plus large (profondeur de champ)
- ▶ pellicule (sensibilité)
- ▶ capteurs numériques (idem)

Tableau

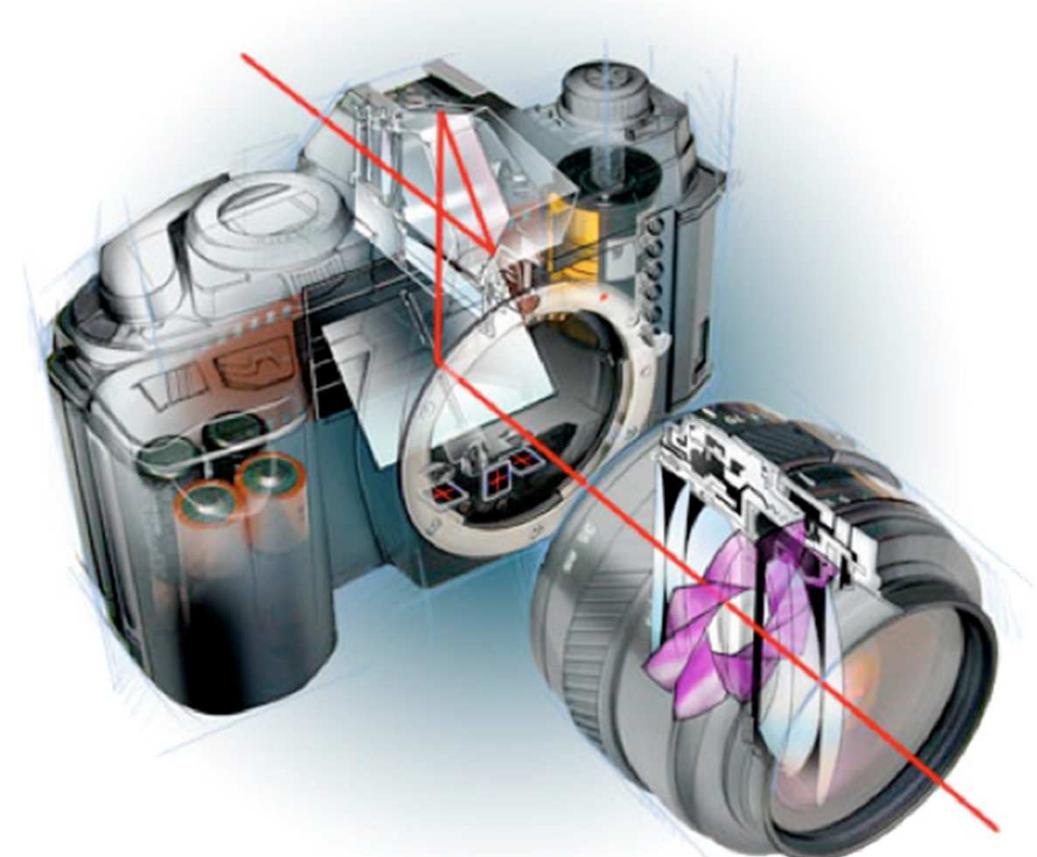


## ① Paramétrage

- ▶ distance focale (zoom)
- ▶ ouverture/diaphragme (“flou” selon la distance)
- ▶ temps de pose (exposition à la lumière)
- ▶ mise au point (choix du plan de netteté)

## ② Et le reste ? (prix de l'appareil)

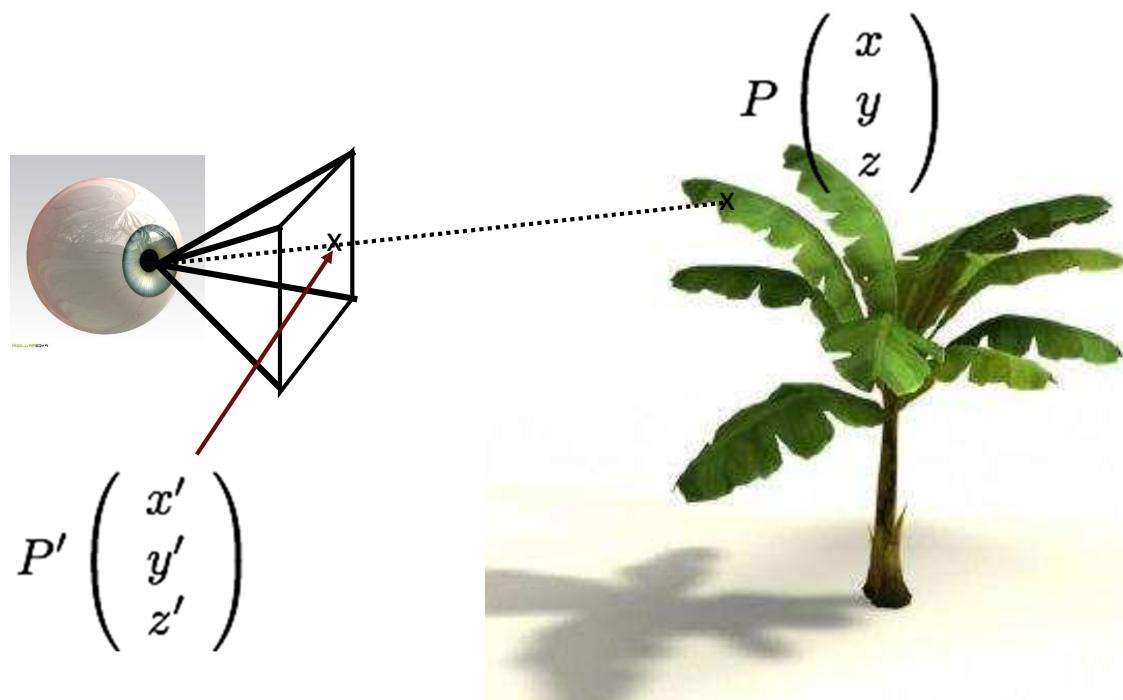
- ▶ des programmes pré-réglés
- ▶ traitement d'images
- ▶ qualité de la compression
- ▶ qualités des objectifs
- ▶ rapidité de la prise de vue
- ▶ etc.



# Projection perspective (suite)

## Vue simplifiée

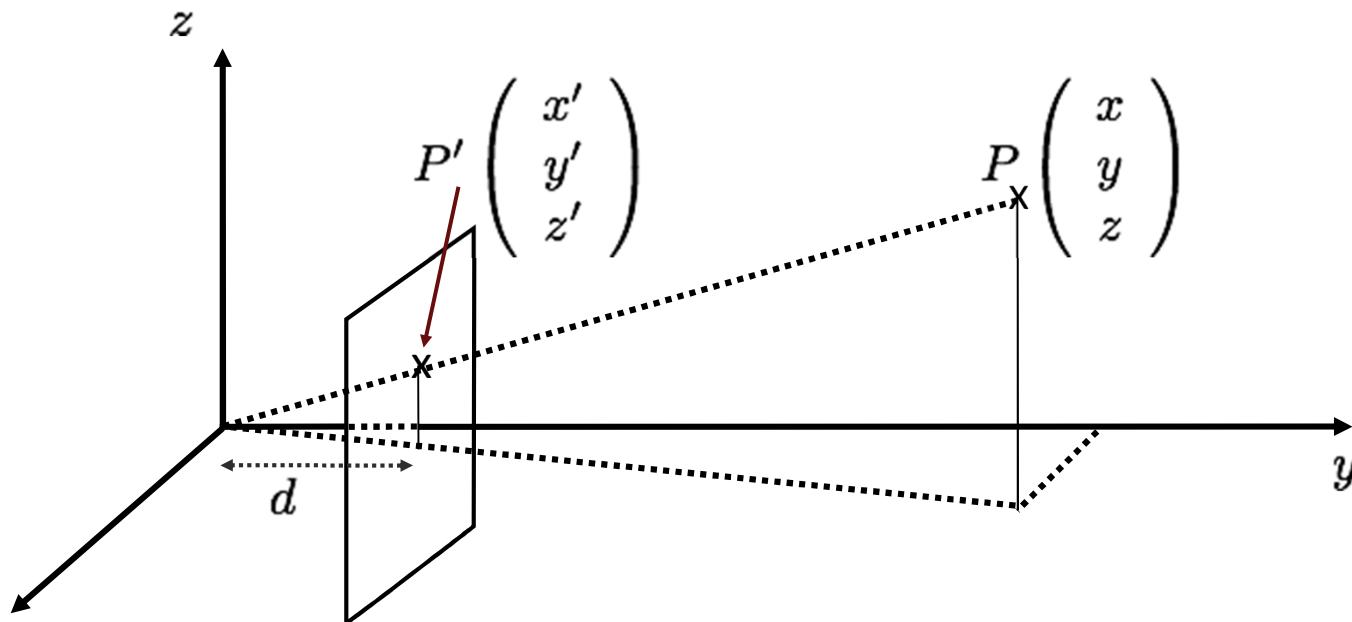
- ▶ caméra trou d'aiguille (tout est net)
- ▶ oeil-écran (pas d'inversion de l'image)
- ▶ projection de chaque point P de la scène



# Perspective (encore)

## Exercice

- œil placé en  $(0,0,0)$ , orienté suivant  $Oy$
- quelles sont les coordonnées de  $P'$  ?



Tableau

Thalès !!!

## ① Matrices (exercice)

- ▶ déduisez la matrice de projection
- ▶ vérifiez qu'elle fonctionne bien pour un point

Tableau

## ② Gestion d'objets plus complexes (exercice)

- ▶ on sait donc projeter... un point !
- ▶ comment faire pour afficher tout un objet ?

Tableau

## ③ Déplacements de caméra (exercice)

- ▶ comment organiser les matrices
- ▶ peut-on faire les calculs avec une seule matrice ?

Tableau

# Tampon de profondeur

## ○ Matrice de projection

- ▶ combinaison : translation/rotation/projection
- ▶ 1 seule matrice !

## ○ Affichage des triangles

- ▶ parcourir les pixels des arêtes
- ▶ remplissage du triangle sur l'écran

## ○ Gestion de la profondeur

- ▶ interpolation de la profondeur aux sommets
- ▶ conserver la profondeur pour le plus proche objet

## ○ Gestion R,V,B

- ▶ interpolation des sommets (normale, R,V,B, autre ?)
- ▶ ou calcul explicite (notion de “shader”)

Tableau

Tableau

C'est ce qui est fait par votre carte graphique !

# Tracé de droites

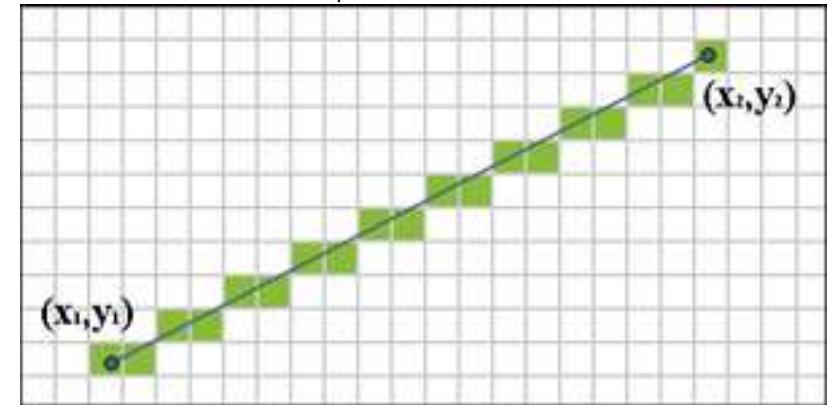
## ➊ Première méthode : algorithme de Bresenham (1963)

- ▶ Aller de  $P_1(x_1, y_1)$  à  $P_2(x_2, y_2)$
- ▶ Plan discret : pixels

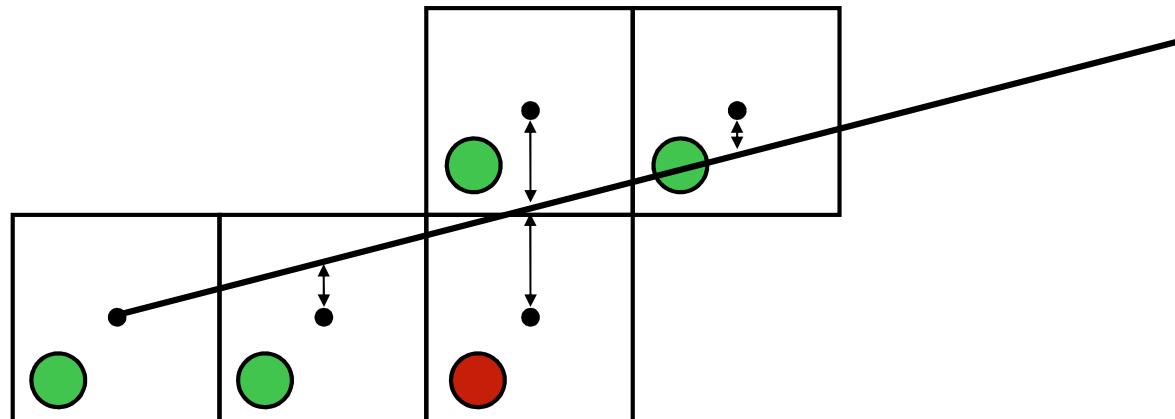
## ➋ Objectif : "allumer les pixels de la droite"

## ➌ Raisonnement sur le premier octant

- ▶  $Dx = x_2 - x_1$  et  $Dy = y_2 - y_1$
- ▶ premier octant :  $|Dx| > |Dy|$  et  $Dy > 0$
- ▶ de façon à avancer  $x$  systématiquement de 1
- ▶ pour les autres cas : raisonnement par symétrie



- ➊ Pente de la droite :  $Dy/Dx$ 
  - ▶ Donc si  $x$  augmente de 1,  $y$  augmente de  $Dy/Dx$
- ➋  $x, y$  représente le point au centre du pixel
- ➌ Si on considère une hauteur  $H$  associée à chaque  $y$  :
  - ▶ à l'intérieur d'un pixel,  $H$  varie de -0.5 à +0.5



```
procédure tracerSegment(entier  $x_1$ , entier  $y_1$ , entier  $x_2$ , entier  $y_2$ ) est
    déclarer entier  $x, y, dx, dy$  ;
    déclarer rationnel  $H, decal$  ;
     $dy \leftarrow y_2 - y_1$  ;
     $dx \leftarrow x_2 - x_1$  ;
     $y \leftarrow y_1$ ; // rangée initiale
     $H \leftarrow 0.0$ ; // valeur d'erreur initiale
     $decal \leftarrow dy / dx$  ;
    pour  $x$  variant de  $x_1$  jusqu'à  $x_2$  par incrément de 1 faire
        tracerPixel( $x, y$ ) ;
        si ( $H \leftarrow H + decal \geq 0.5$ ) alors // pour le pixel suivant
             $y \leftarrow y + 1$ ; // choisir plutôt le pixel suivant sur la ligne supérieure
             $H \leftarrow H - 1.0$ ; // ajuste la hauteur
        fin si ;
    fin pour ;
fin procédure ;
```

● Plusieurs optimisations sont encore possibles

- ▶ réduction des erreurs numériques
- ▶ calcul en entier

● Il reste maintenant à généraliser la méthode

- ▶ en regardant sur les valeurs de Dx et Dy

...

# Principe d'affichage

## ① Affichage en fil de fer à partir d'un maillage 3D

- ▶ Projeter tous les points des sommets
- ▶ Tracer toutes les arêtes !

On sait maintenant faire !

## ② Attention tout de même

- ▶ Placement de la caméra (translation/rotation)
- ▶ Matrice de projection

## ③ Maintenant

- ▶ Comment remplir ?
- ▶ Comment gérer la profondeur ?

# Affichage solide

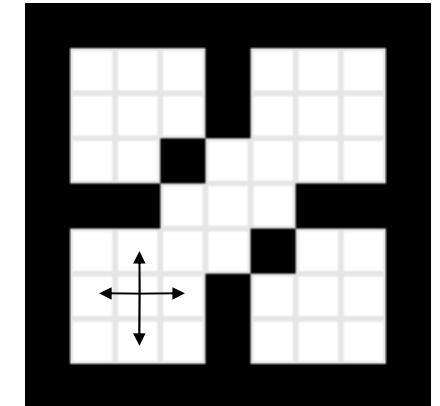
- ➊ L'affichage par des droites n'est pas toujours suffisant  
Il existe donc des algorithmes de remplissage
  
- ➋ Deux principales familles
  - ▶ Par germe (ou par diffusion)
  - ▶ Par lignes de balayage

Nous allons voir les idées de ces méthodes  
(sans tous les détails de mise en oeuvre)

# Remplissage par diffusion

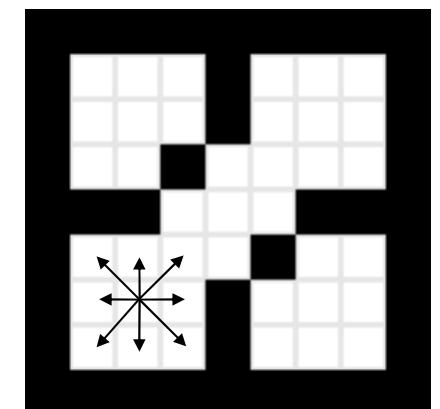
## ○ Principe

- ▶ connaître un pixel à l'intérieur du polygone
- ▶ diffuser le remplissage de proche en proche



## ○ Algorithme récursif 4-connexe (idem 8-connexe)

```
remplissage4(pixel, colcible, colrep)
début
    si couleur(pixel) = colcible
        alors
            couleur(pixel) = colrep
            remplissage4(pixel au nord, colcible, colrep)
            remplissage4(pixel au sud, colcible, colrep)
            remplissage4(pixel à l'est, colcible, colrep)
            remplissage4(pixel à l'ouest, colcible, colrep)
        finsi
    fin
```



# Remplissage par diffusion (3)

## ① Algorithme 4-connexe

```
remplissage4(pixel, colcible, colrep)
début
    Soit  $P$  une pile vide
    si couleur( $pixel$ ) !=  $colcible$  alors sortir
    finsi
    Empiler  $pixel$  sur  $P$ 
    Tant que  $P$  non vide
        faire
            Dépiler  $n$  de  $P$ 
            couleur( $n$ ) :=  $colrep$ 
            si couleur( $n$  nord) =  $colcible$  alors Empiler  $n$  nord sur  $P$  finsi
            si couleur( $n$  sud) =  $colcible$  alors Empiler  $n$  sud sur  $P$  finsi
            si couleur( $n$  est) =  $colcible$  alors Empiler  $n$  est sur  $P$  finsi
            si couleur( $n$  ouest) =  $colcible$  alors Empiler  $n$  ouest sur  $P$  finsi
        fintantque
    fin
```

## ② Optimisations possibles

- ▶ par exemple par propagation bidirectionnelle...

# Remplissage par lignes de balayage

## Remplissage par diffusion

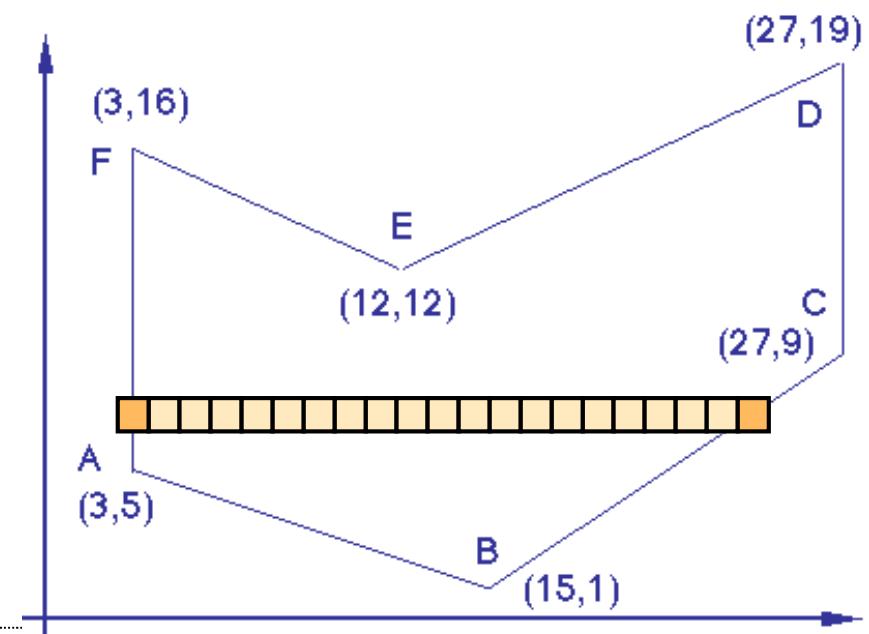
- ▶ Plusieurs tests pour le même pixel
  - ▶ Moins efficace
  - ▶ Parcours moins organisé
  - ▶ Mais permet de remplir n'importe quoi

## ① Seconde solution : lignes de balayage

- ▶ Nécessite de connaître les arêtes des polygones
  - ▶ Parcours / remplissage ligne par ligne

# Principe

- ▶ Trier les sommets
  - ▶ Partir d'une extrémité
  - ▶ Parcourir le polygone ligne par ligne

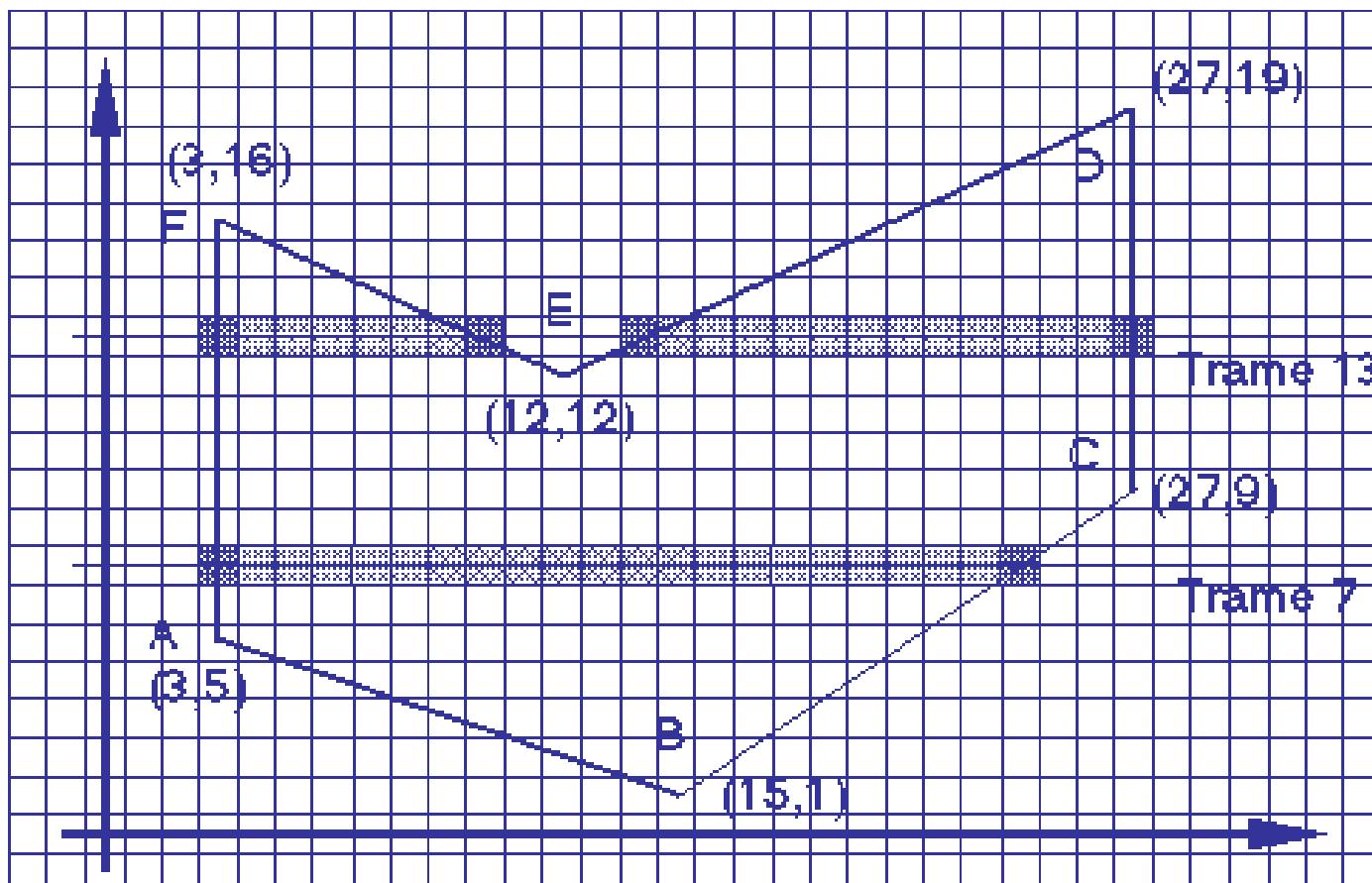


# Remplissage par lignes de balayage

- Pour cela :

- ▶ Savoir parcourir les droites ligne après ligne
- ▶ Remplir d'une extrémité à l'autre

- Attention aux polygones non convexes



# Remplissage par lignes de balayage

## ● Mise en oeuvre

- ▶ Mémoriser tous les changements de configuration
- ▶ Cela correspond aux passages aux sommets
- ▶ Et produit des changements d'arêtes

## ● Comment procéder ?

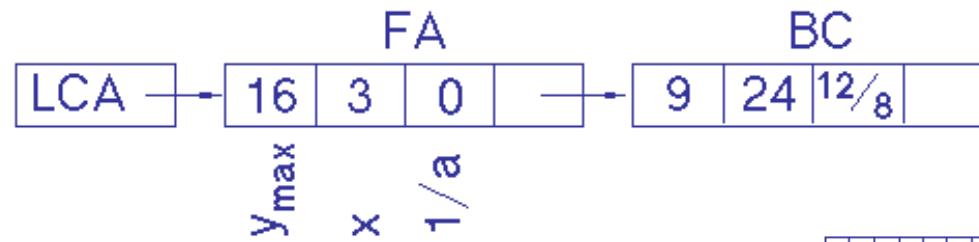
- ▶ Précalcul des coordonnées des sommets
- ▶ Indication des nouvelles arêtes
- ▶ Attention, il peut y en avoir plusieurs !

## ● Utilisation d'une LCA (liste des côtés actifs)

- ▶ Mise à jour au fur et à mesure de l'avancement
- ▶ Lorsque les sommets sont atteints

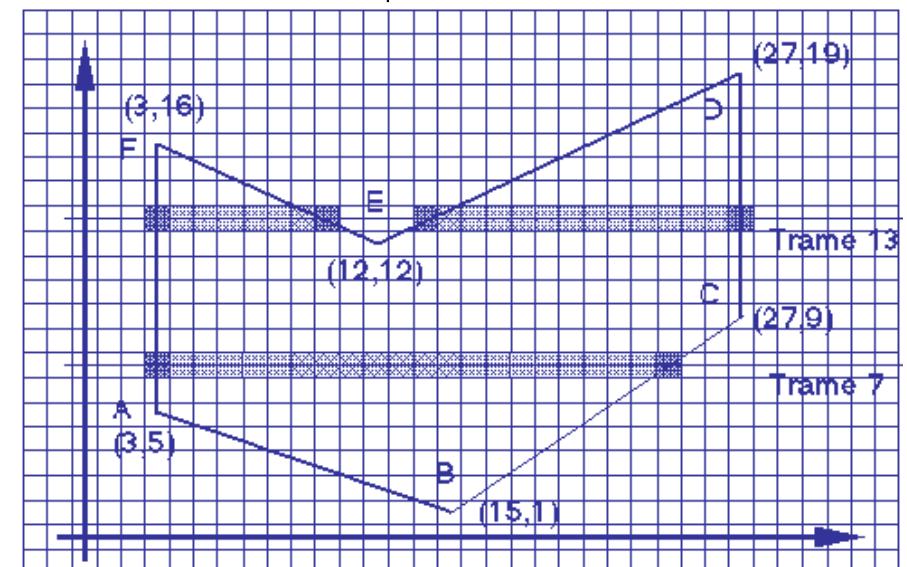
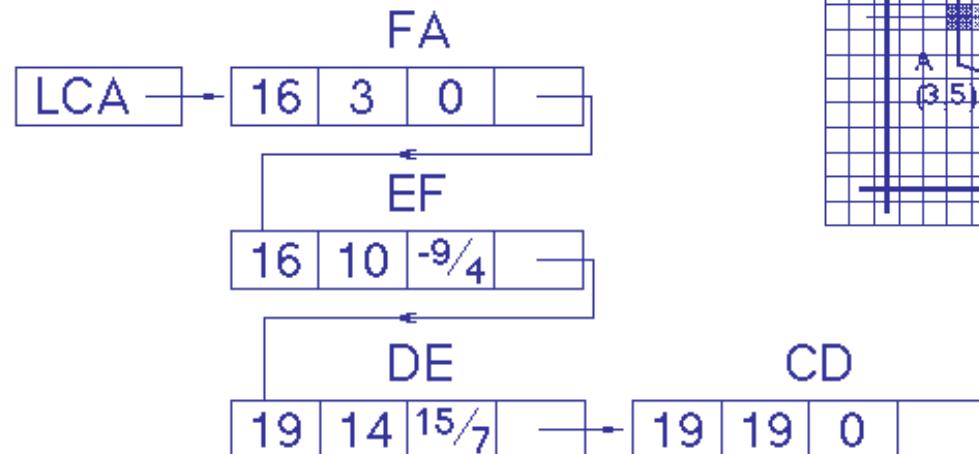
# Remplissage par lignes de balayage

## ● Pour la trame 7



- ▶  $1/a$ , avec 'a' la pente =  $dy/dx$

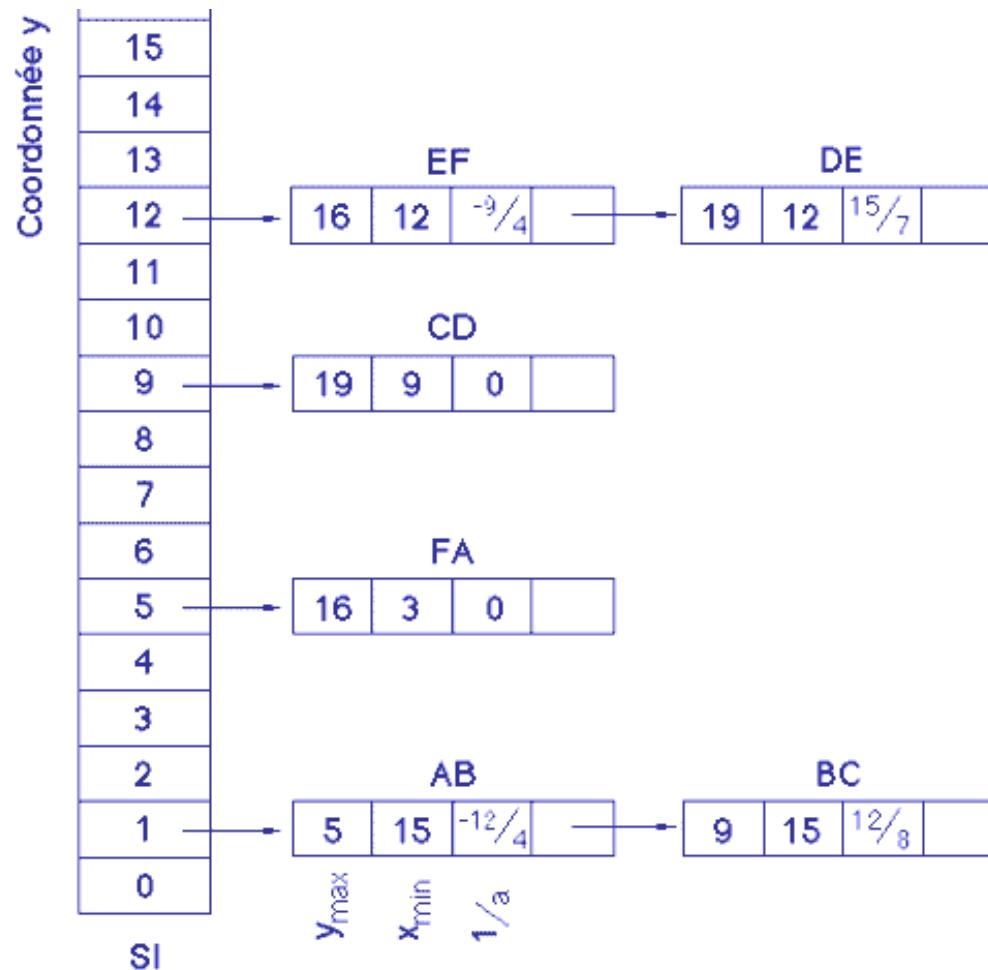
## ● Pour la trame 13



# Remplissage par lignes de balayage

➊ Pour éviter de savoir quelles nouvelles arêtes :

- ▶ Structure intermédiaire (SI)
- ▶ Indique pour chaque ligne quelles sont les arêtes



# Remplissage par lignes de balayage

## Algorithm

```
Procédure remplir_polygone(p:polygone) ;  
début  
    créer la structure SI  
    initialiser la structure LCA à vide  
    pour chaque ligne du polygone faire  
        mettre à jour les entrées de LCA à partir de SI  
        mettre à jour les sorties de LCA à partir de SI  
        afficher tous les morceaux de trames  
            décrits dans LCA (parcours de droites)  
    fin  
fin
```

## Parcours des droites

- ▶ Soit en déterminant (calculant)  $x$  à partir de l'équation  $y = ax+b$   
 $\Leftrightarrow x = (y-b)/a$  (attention à  $a=0$ )
- ▶ Soit de manière incrémentale (adapter Bresenham !)

Tableau

# Gestion de la profondeur

## ➊ Le remplissage seul ne suffit pas

Un polygone plus éloigné de la caméra peut être caché par un autre

## ➋ Méthode 1 (Algorithme du peintre)

- ▶ Trier les polygones lors de l'affichage
- ▶ Attention aux polygones qui se coupent
- ▶ Peu utilisé et non détaillé ici

## ➌ Méthode 2 (Tampon de profondeur ou depth-buffer)

- ▶ Le tampon de profondeur (ou Z-buffer/depth-buffer)
- ▶ Mémorise la profondeur associée à chaque pixel
- ▶ Au moment de l'affichage
- ▶ Nécessite de stocker une image de profondeur
- ▶ Comment estimer la profondeur ?

# Calcul de profondeur

---

## ● Principe

- ▶ Calculer/récupérer la profondeur
- ▶ Pour chaque sommet
- ▶ Interpoler lors du remplissage

## ● Profondeur

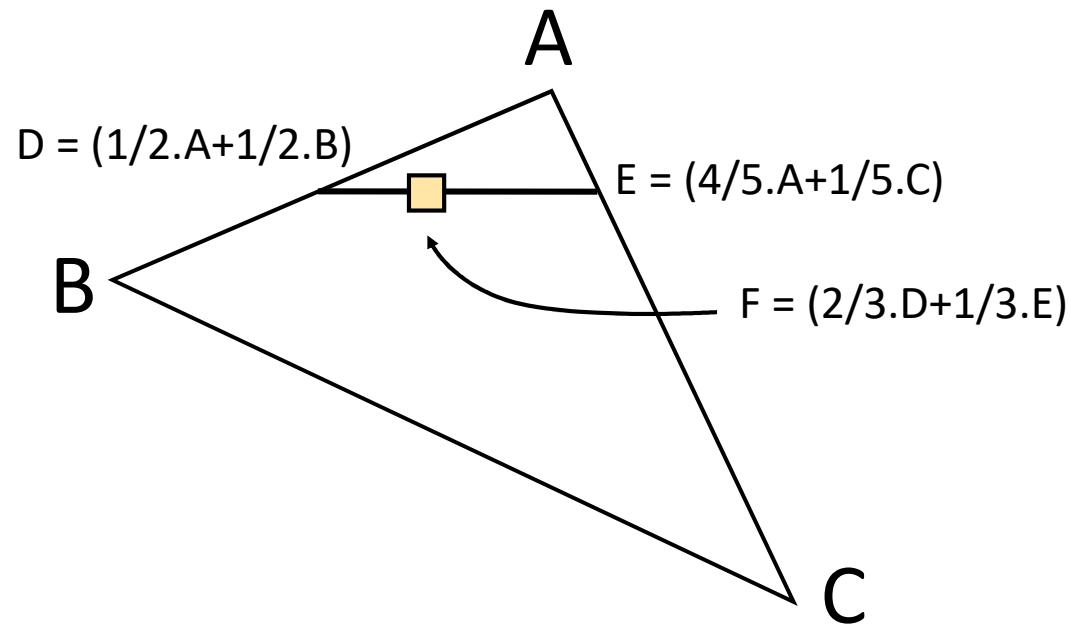
- ▶ Donnée par la valeur "Z" (ou Y pour nous)
- ▶ Juste avant la projection

## ● Interpolation

- ▶ Méthode classique "bilinéaire"
- ▶ Réalisée pour chaque pixel du remplissage

# Interpolation bilinéaire

- ➊ Une valeur à chaque sommet (A,B et C)

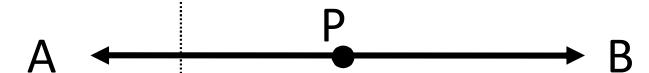


- ➋ Les valeurs A,B et C peuvent être
  - ▶ De la couleur
  - ▶ De la profondeur
  - ▶ Ou autre chose (coordonnées de textures...)

# Interpolation linéaire

## Exemple pour le milieu

$$P = 0.5A + 0.5B$$



## Pour tout autre point

$$P = tA + (1 - t)B$$

$$t = (P - B)/(A - B)$$

$$t = (x_P - x_B)/(x_A - x_B)$$

$$t = (y_P - y_B)/(y_A - y_B) \quad \text{Si la droite est verticale, car } x_A - x_B = 0$$

## 't' donne la pondération à associer à P pour interpoler

- ▶ si  $P=A$  alors  $t=1$
- ▶ si  $P=B$  alors  $t=0$

# Interpolation Linéaire : exemple

## ● Avec de la couleur

- ▶ Si A a pour valeur (Ra,Va,Ba) et B (Rb,Vb,Bb)
- ▶ Alors on aura pour P :

$$Rp = tRa + (1 - t)Rb$$

$$Vp = tVa + (1 - t)Vb$$

$$Bp = tBa + (1 - t)Bb$$

- ▶ si P=A alors ( $Rp=Ra$ ,  $Vp=Va$ ,  $Bp=Ba$ )
- ▶ si P=B alors ( $Rp=Rb$ ,  $Vp=Vb$ ,  $Bp=Bb$ )

## ● Avec une profondeur Za pour A et Zb pour B

- ▶  $Zp = tZa + (1-t)Zb$
- ▶ Si P=A, on a bien  $Zp=Za$
- ▶ Si P=B, on a bien  $Zp=Zb$

# Placement de caméra

○ La scène est localisée n'importe où dans l'espace

○ Exercice

- ▶ plaçons la caméra en 0,0,0
- ▶ avec une orientation suivant Oy

Faites un dessin

Construisez la matrice de projection

Décrivez l'algorithme d'affichage en fil de fer

- ▶ plaçons la caméra au point  $X_c, Y_c, Z_c$

Comment écrire le nouvel algorithme ?

Comment le rendre le plus efficace possible ?

- ▶ Comment définir une autre orientation ?

Comment représenter la rotation

Comment appliquer toutes les opérations

Est-ce possible d'utiliser une seule matrice ?

Maintenant, définissez l'algorithme complet !

Tableau

Tableau

Tableau

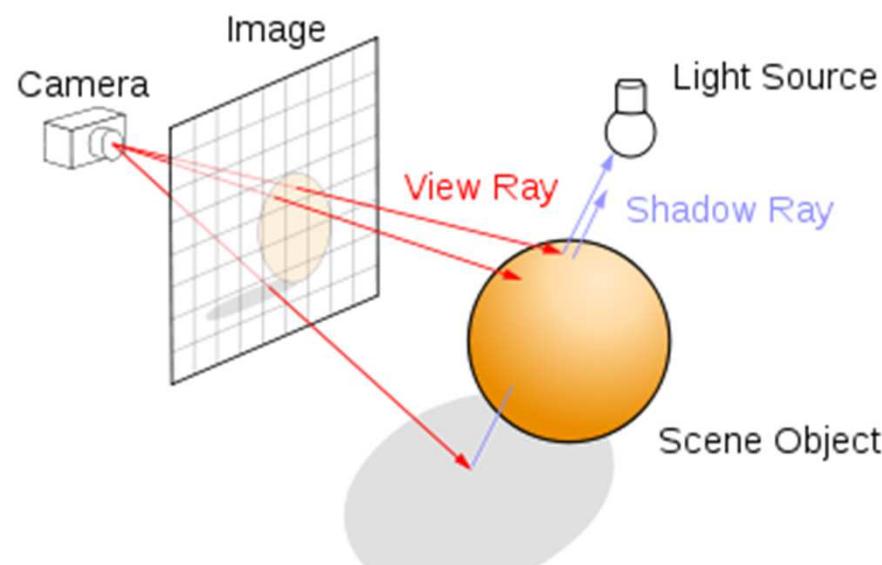
# Tracé de rayons

## ● Méthode générique

- ▶ trouver un point d'intersection
- ▶ entre un rayon (lumineux) et un objet

## ● Utilisée pour

- ▶ le lancer de rayons
- suivi du trajet de la lumière
- inverse : depuis la caméra jusqu'aux sources
- ▶ d'autres méthodes (tracé de photons, visibilité, etc.)



# Rayons : paramétrisation

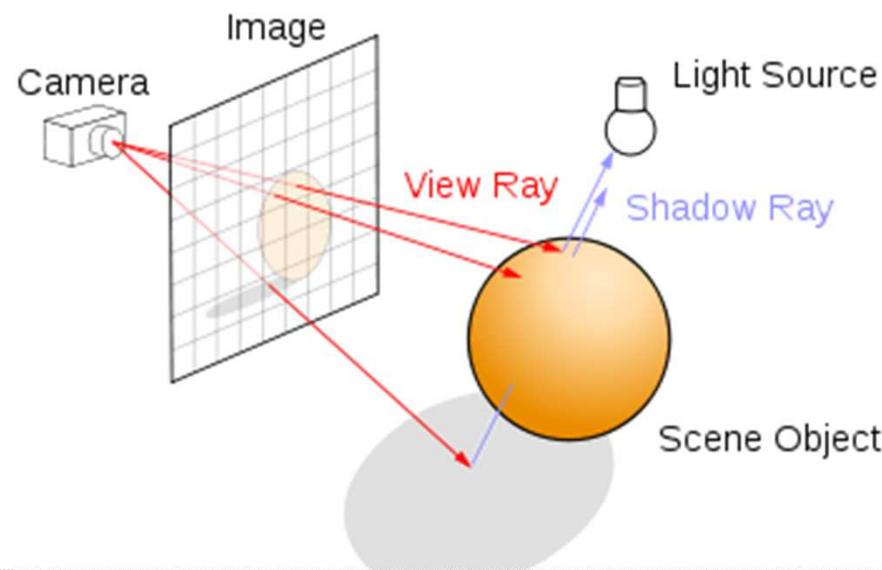
Tableau

## ○ Un rayon est une droite

- ▶ Comment la représenter ?
- ▶ Quelle est la façon la plus simple ?
- ▶ Intersection entre un rayon et un objet ?

## ○ Comment faire une image ?

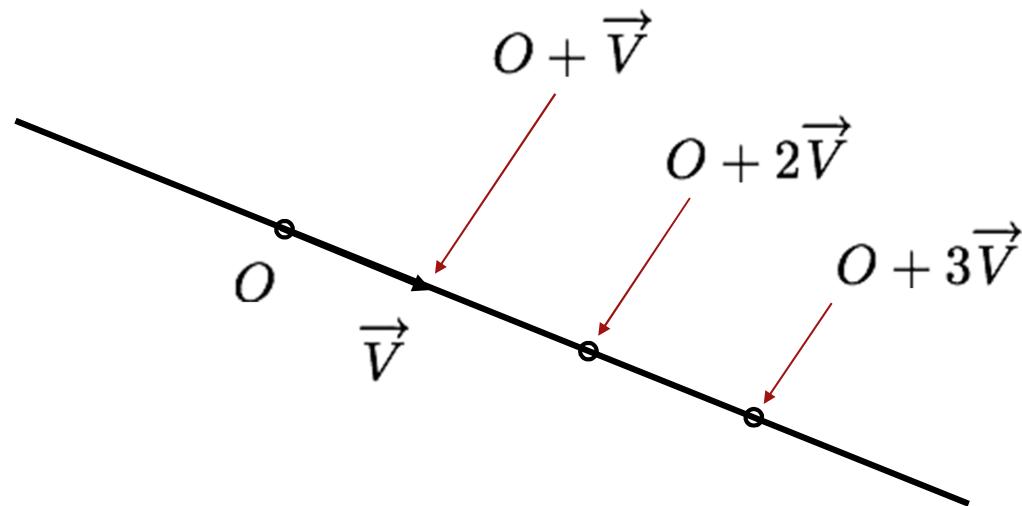
- ▶ Pour chaque pixel
- ▶ Rayon passant par le centre de projection et le pixel



# Rayon : paramétrisation

- Solution habituelle :

$$O + t\vec{V}$$



- le paramètre  $t$  est aussi la distance (parfois négative)
  - ▶ entre  $O$
  - ▶ et un point de la droite
- Si on a plusieurs objets atteints
  - ▶ le plus proche a une valeur de  $t$  plus petite ( $>0$ )

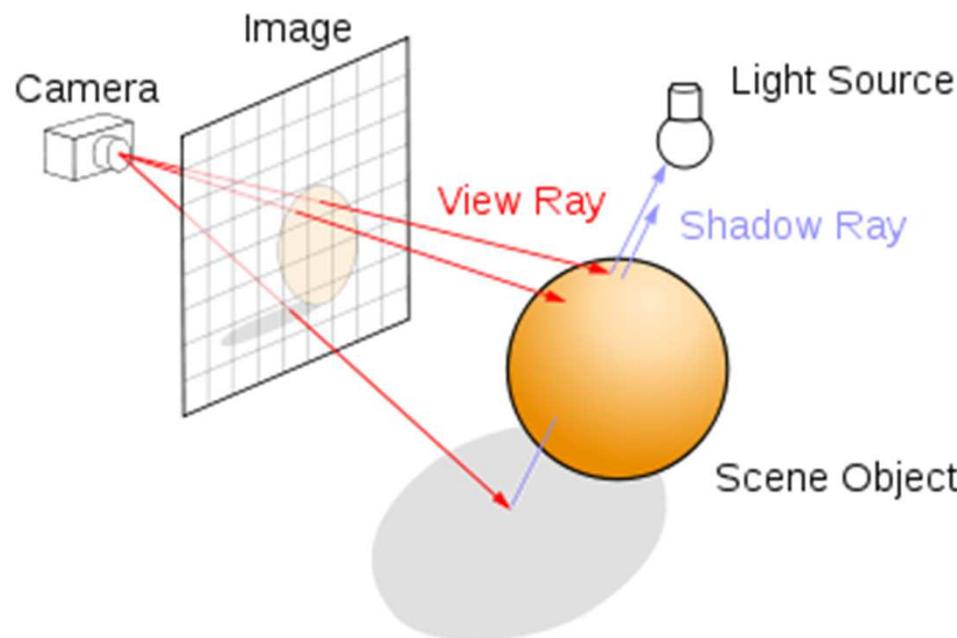
# Calcul d'une image

## ● Pour chaque pixel

- ▶ créer le rayon correspondant
- ▶ calculer l'intersection avec tous les objets
- ▶ conserver celle la plus proche ( $t > 0$  et  $t$  minimal)
- ▶ estimer la "couleur" de l'objet en ce point

## ● Projection automatique ! (gérée par pixel)

Tableau



## ○ Difficulté : comment calculer les intersections ?

## ○ Problème de complexité

- ▶ pour chaque pixel (beaucoup de pixels !)
- ▶ pour chaque objet (si beaucoup d'objets ?)
- ▶ à voir plus tard...

## ○ Exercice

▶ Étant donné un plan  $ax + by + cz + d = 0$

▶ Calculez l'intersection avec un rayon

▶ Étant donné une sphère  $x^2 + y^2 + z^2 = r^2$

▶ Calculez l'intersection avec un rayon

▶ Idem avec une sphère centrée en  $(x_o, y_o, z_o)$

▶ et de rayon  $r$

Tableau

Tableau

Tableau

# Intersections (suite)

## ① Même question pour un triangle

- ▶ définissez le rayon
- ▶ définissez le plan et calculer l'intersection
- ▶ voir si elle est bien dans le triangle

Tableau

## ② Intersection avec une boîte min/max

- ▶ faites d'abord un dessin en 2D avec un carré
- ▶ calcul des intersections avec chaque arête
- ▶ trouvez une relation
- ▶ faites la généralisation pour la boîte 3D

Tableau

## ● La caméra est définie

- ▶ par un point (le centre de projection)
- ▶ une direction
- ▶ une distance focale
- ▶ une résolution (nb de pixels)
- ▶ une taille de capteur (24 mm x 36 mm)

## ● Exercice

- ▶ la caméra est placée en  $(0,0,0)$
- ▶ la direction visée est  $(0,1,0)$
- ▶ une focale  $f$
- ▶ quel est l'angle d'observation selon  $x$  et selon  $z$  ?
- ▶ quel est l'angle de la diagonale ?

Tableau

# Caméra et rayons (suite)

## Exercice

- ▶ avec la même caméra
- ▶ on définit la résolution  $R_x \times R_y$  ( $R_x \times R_z$ )
- ▶ quelle est la position du centre du “premier” pixel ?
- ▶ quelle est la taille d'un pixel ?

## Exercice

- ▶ écrivez un algorithme qui parcourt tous les pixels
- ▶ et qui construit chaque rayon

## Exercice

- ▶ définissez les rayons si la caméra est en  $(X_c, Y_c, Z_c)$

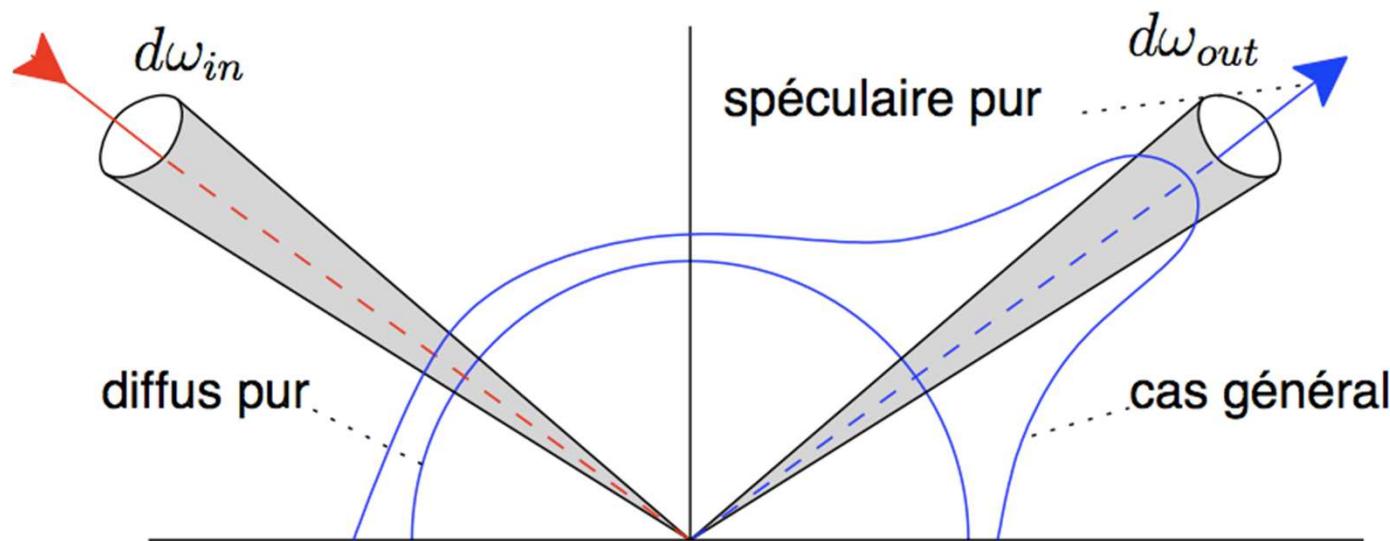
## Exercice

- ▶ idem si la direction visée n'est pas

$$(0, 1, 0) \longrightarrow (V_x, V_y, V_z)$$

- Introduction
  - ▶ Principes généraux
  - ▶ Applications
- Modélisation géométrique
  - ▶ Repère 3D et objets 0D,1D,2D,3D
  - ▶ Transformations et coordonnées homogènes
- Visualisation
  - ▶ Le “tampon de profondeur”
  - ▶ Le “lancé de rayons”
- **Rendu réaliste**
  - ▶ La lumière, les ombres
  - ▶ Les matériaux, textures
- Aller plus loin...
  - ▶ Simulation d'éclairage
  - ▶ Aliassage, etc.

- ➊ Trouver un bon modèle...
  - ▶ pour faire du bois, du métal, du marbre
  - ▶ avec de la cire, du vernis ?
- ➋ Pas si simple,
  - ▶ mais on va faire les premiers : diffus / spéculaire
  - ▶ formules et calculs très faciles
- ➌ Modèle de Lambert (surfaces diffuses)
- ➍ Modèle de Phong (diffuses et reflets spéculaires)

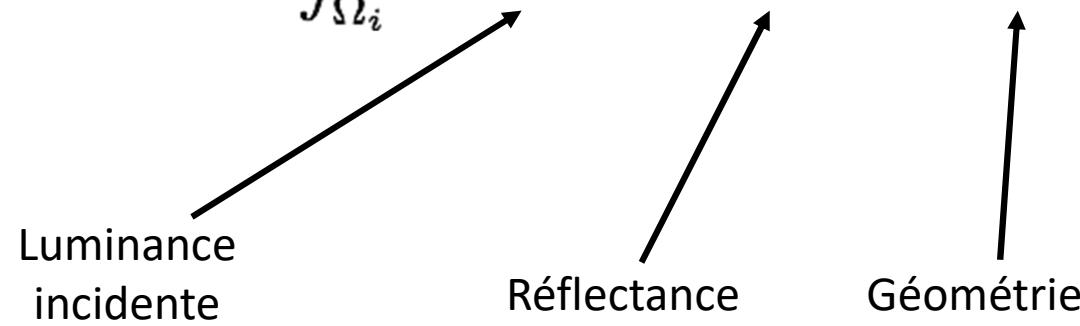


Tableau

## ● Lumière réfléchie

- ▶ identique dans toutes les directions
- ▶ terme constant !

$$L_o(x, \vec{\omega}_o) = \int_{\Omega_i} L(x, \vec{\omega}_i) \cdot f(x, \omega_i, \omega_o) \cdot \cos \theta_i \cdot d\omega_i$$



Tableau

● Pour ce cas :  $f(x, \omega_i, \omega_o) = C_{ste}$

● Pour une seule source lumineuse ponctuelle :

$$L_o(x) = L_i(x) \cdot C_{ste}(x) \cdot \cos \theta$$

- ➊ Très facile à programmer :

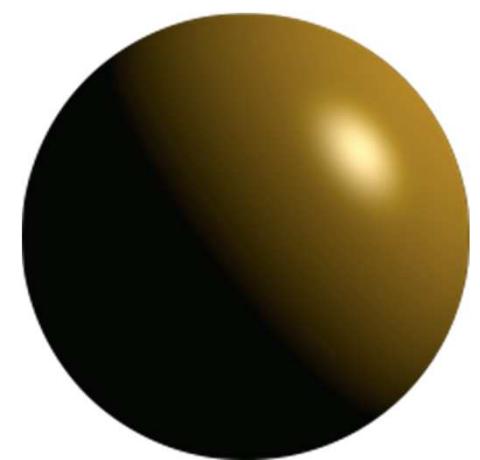
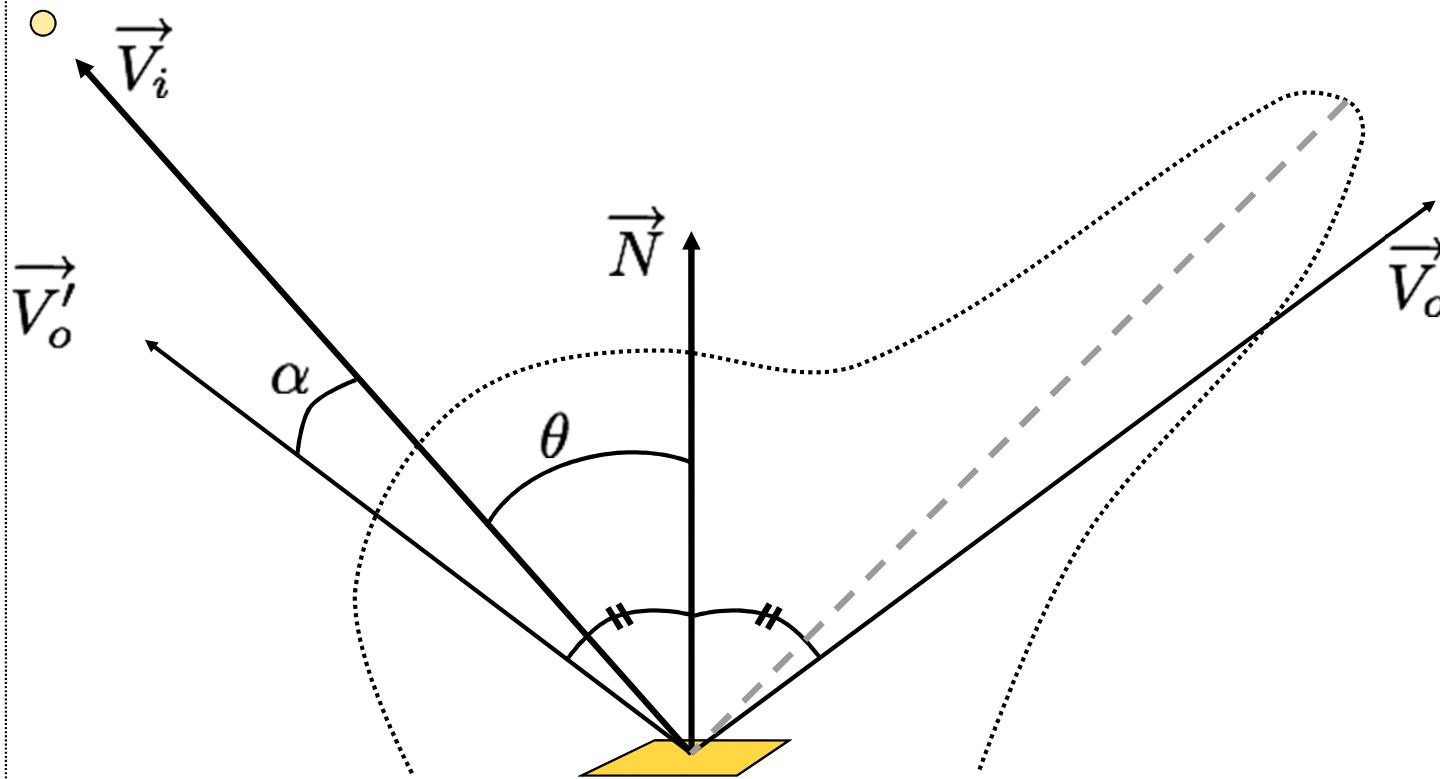
$$L_o(x) = L_i(x).C_{ste}(x).\cos \theta$$

$$L_o(x) = L_i(x).C_{ste}(x).(\vec{N} \cdot \vec{V}_i)$$

Tableau

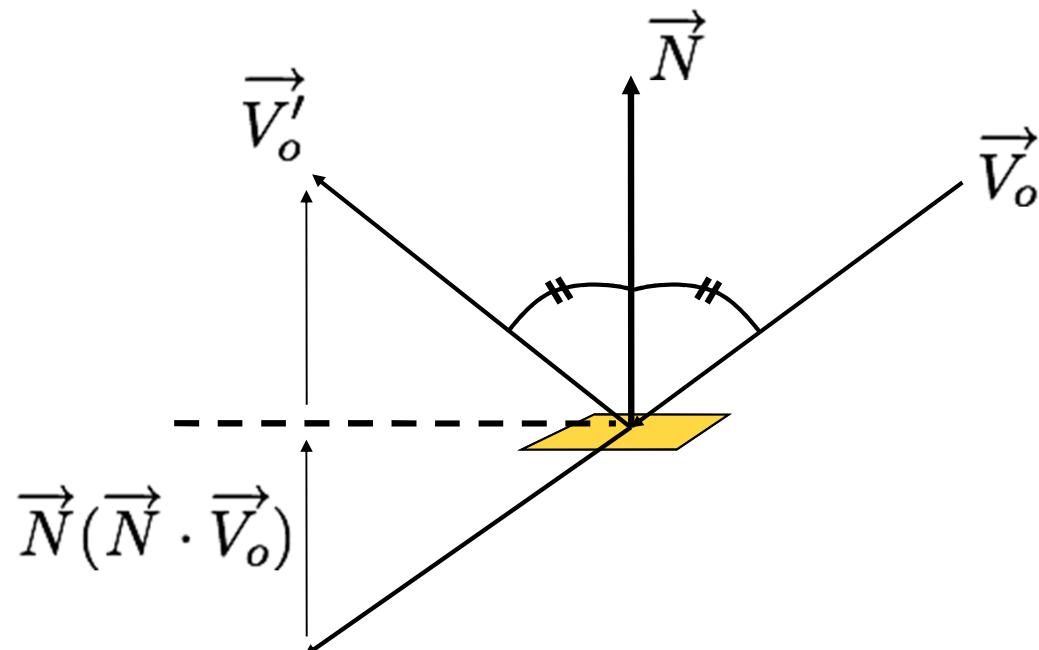
- ➊ Ajouter une partie spéculaire
- ➋ Sur la base du modèle de Lambert

$$L_o(x, \vec{V}_o) = L_i(x, \vec{V}_i) \cdot (K_d \cdot \cos \theta + K_s \cdot \cos^n \alpha)$$



Comment calculer  $\vec{V}'_o$  ?

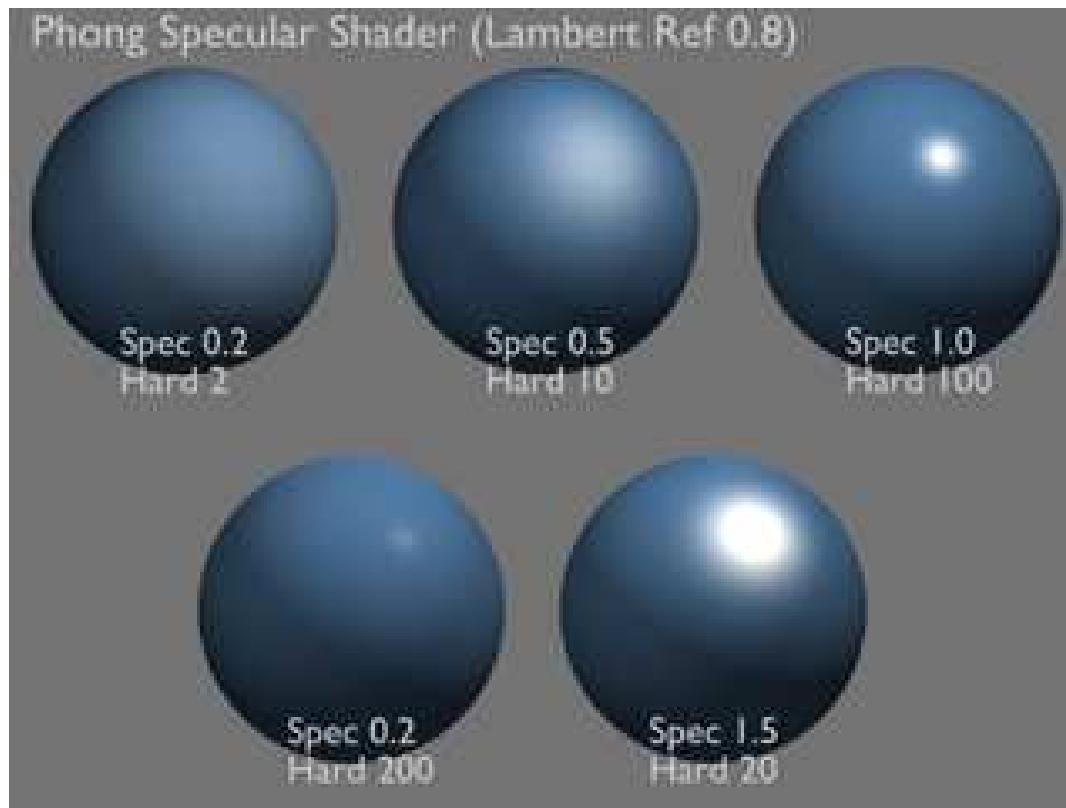
$$\vec{V}'_o = \vec{V}_o - 2\vec{N}(\vec{N} \cdot \vec{V}_o)$$



# Phong (encore)

● Paramètres :  $K_s, K_d, n$

- ▶  $K_d \Rightarrow$  aspect diffus
- ▶  $K_s \Rightarrow$  aspect brillant
- ▶  $n \Rightarrow$  taille du “lobe”



# Phong (toujours)

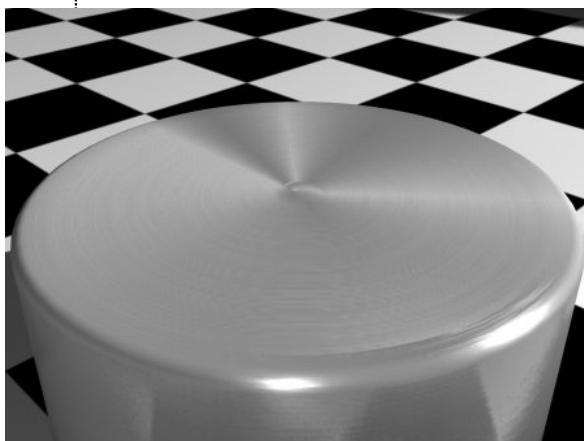
- Problèmes avec Phong (conservation de l'énergie)
- Heureusement : Phong modifié

$$\left( \frac{1}{\pi r^2} \cdot K_d + \frac{(n+2)}{2\pi r^2} \cdot K_s \cdot \cos^n \alpha \right)$$

- ▶ pas plus compliqué
- ▶ pas vraiment plus lent

- Et c'est tout pour les modèles pour le moment...

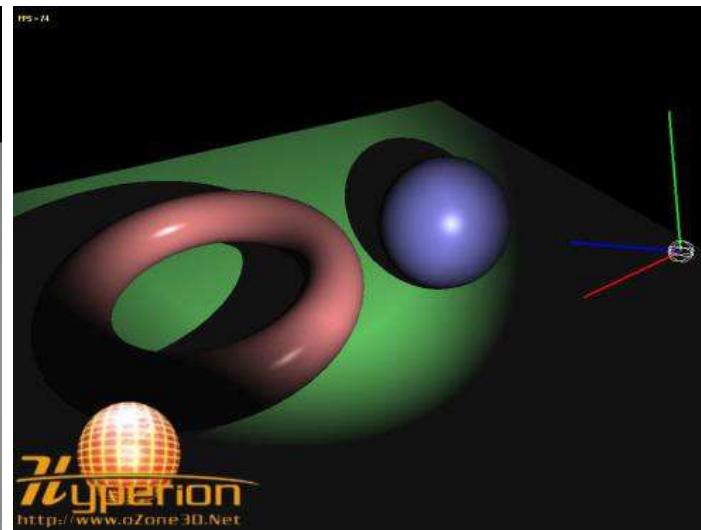
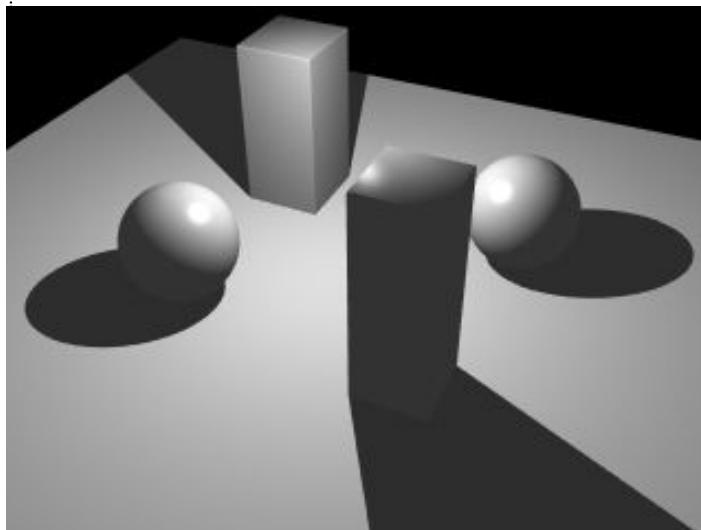
On ne pourra donc pas faire ça :



# Sources lumineuses

## ○ En général :

- ▶ sources ponctuelles (omnidirectionnelles)
- ▶ spots (directionnels)
- ▶ sources surfaciques



# Source ponctuelle

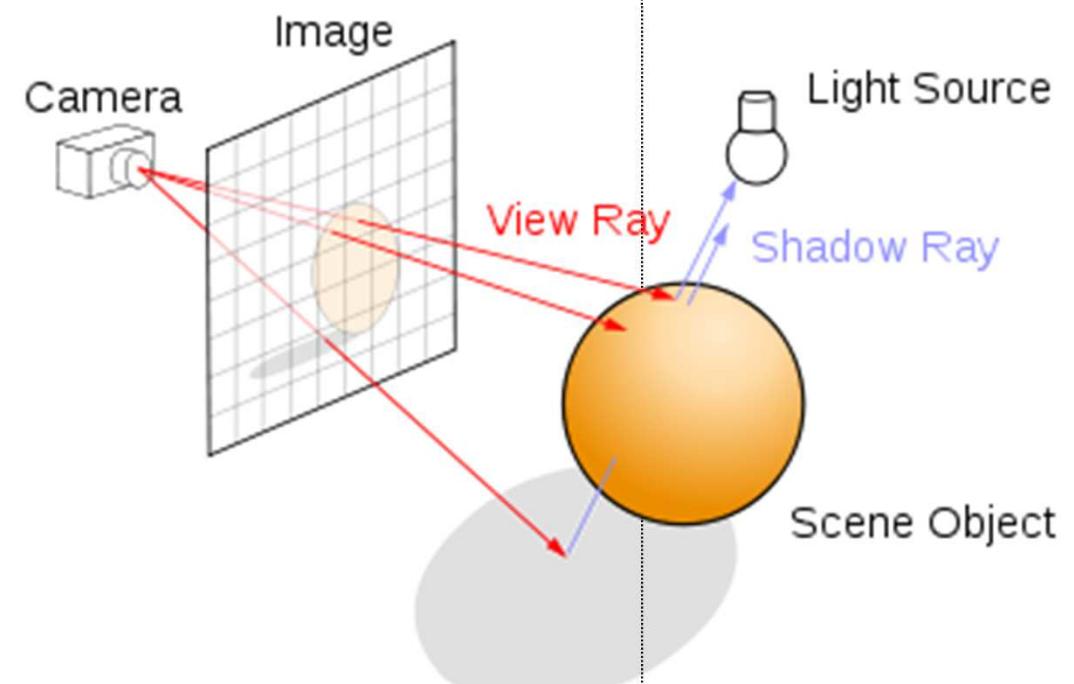
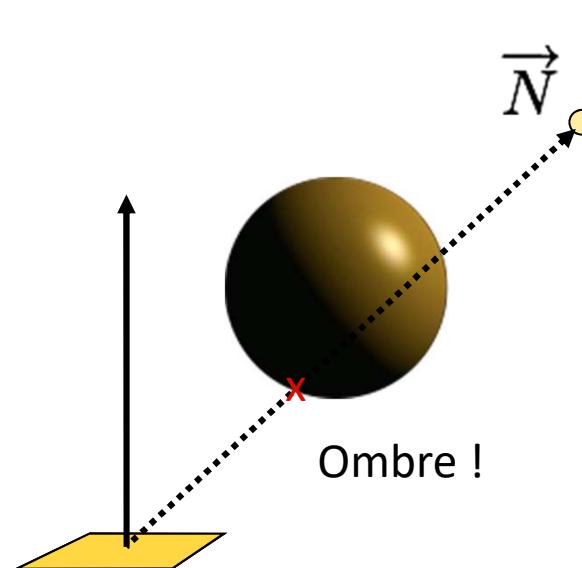
## ○ Relancer un rayon

- ▶ depuis le point d'impact (objet)
- ▶ vers la source

Tableau

## ○ S'il y a un objet avant la source => ombre

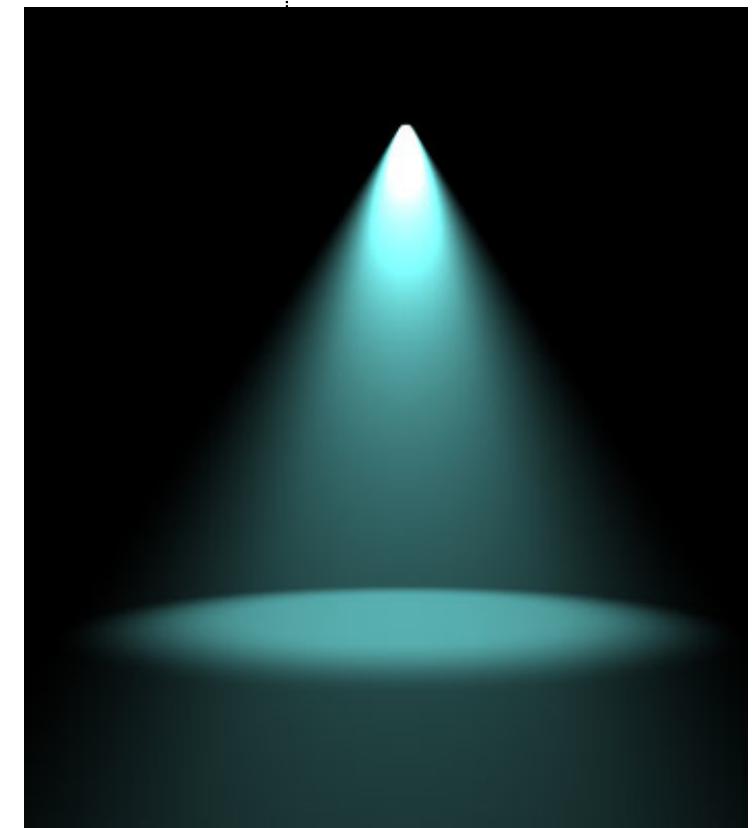
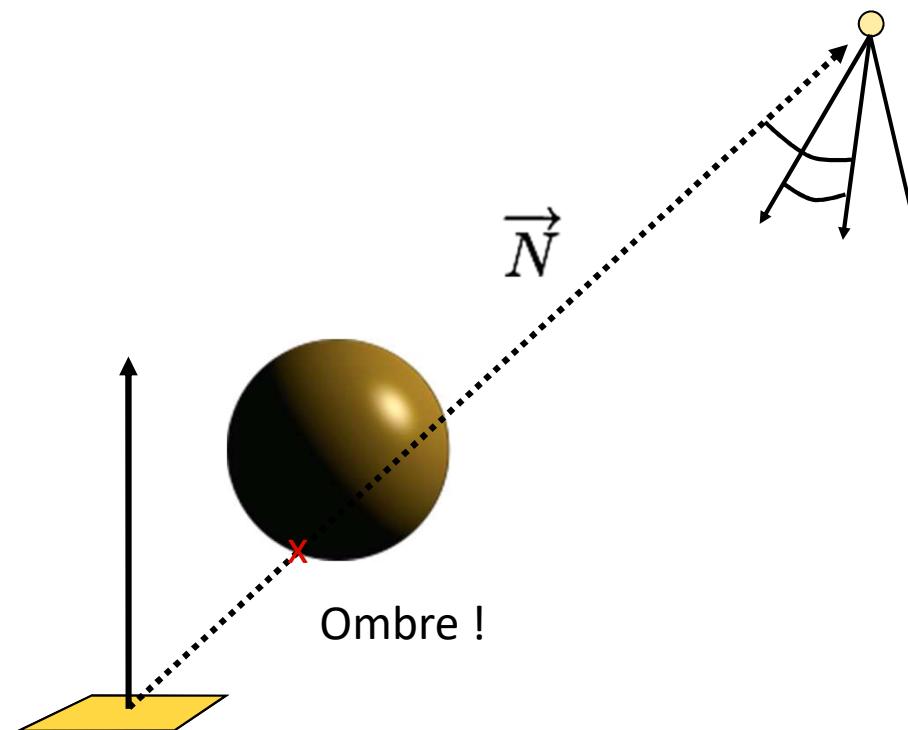
## ○ Sinon, calcul (Lambert/Phong ou autre)



## ○ Source ponctuelle, directionnelle

- ▶ définie par un point
- ▶ un vecteur directeur
- ▶ un angle (ou son cosinus)

Tableau



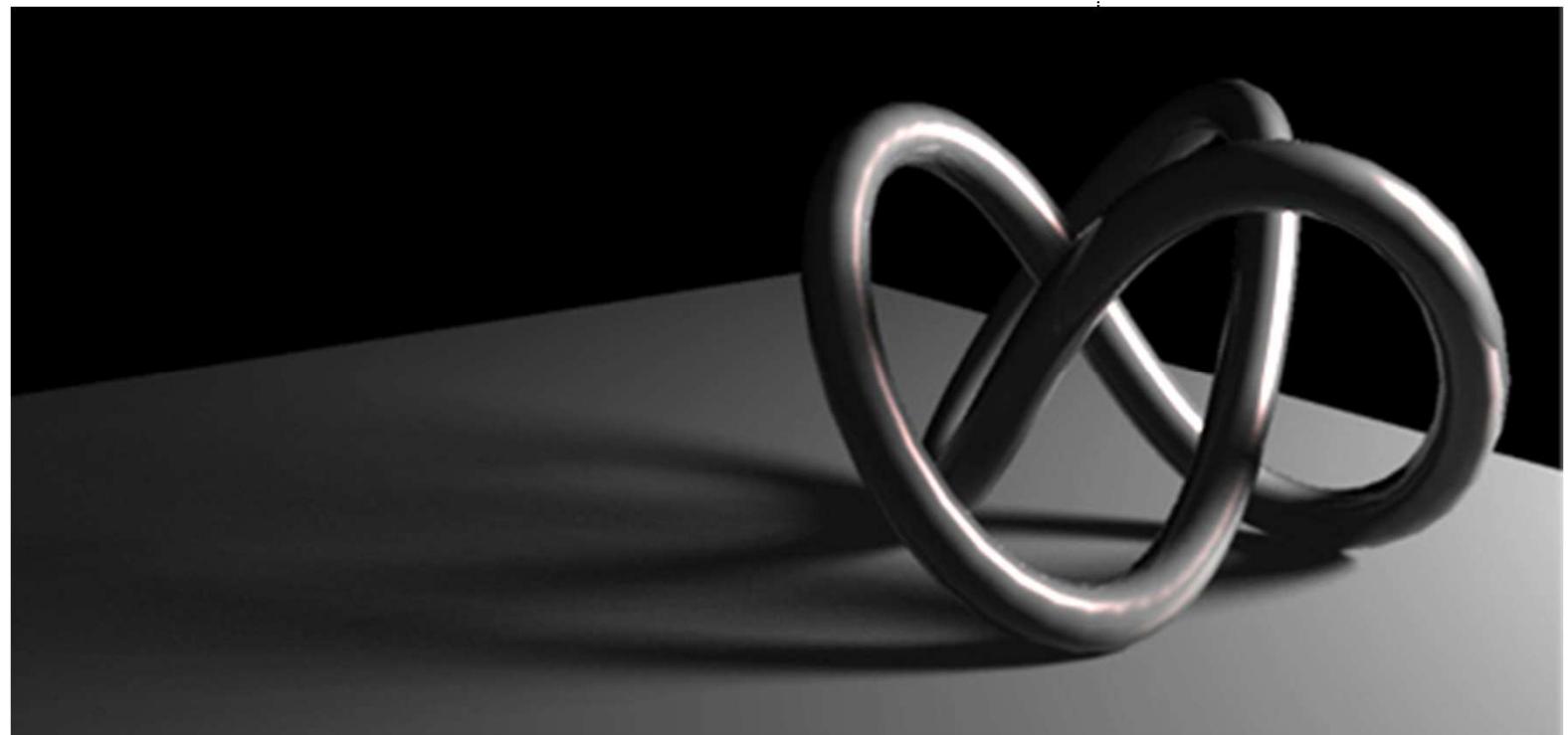
# Source surfacique

- **La source n'est plus un point, mais une surface**

- ▶ ombres douces
- ▶ rendu plus réaliste

- **Comment faire ?**

- ▶ échantillonner la source
- ▶ relancer des rayons (coûteux...)



- Introduction
  - ▶ Principes généraux
  - ▶ Applications
- Modélisation géométrique
  - ▶ Repère 3D et objets 0D,1D,2D,3D
  - ▶ Transformations et coordonnées homogènes
- Visualisation
  - ▶ Le “tampon de profondeur”
  - ▶ Le “lancé de rayons”
- Rendu réaliste
  - ▶ La lumière, les ombres
  - ▶ Les matériaux, textures
- **Aller plus loin...**
  - ▶ Simulation d'éclairage
  - ▶ Aliassage, etc.

● **Les méthodes vues permettent**

- ▶ de produire une (belle ou très belle) image
- ▶ de traiter correctement quelques effets lumineux
- ▶ d'en approcher d'autres

● **Il reste encore beaucoup d'autres phénomènes**

- ▶ matériaux des objets
- ▶ diffusion lumineuse sous-surfaciques
- ▶ simulation des inter-réflexions lumineuses
- ▶ etc.

● **Et surtout d'autres choses !**

- ▶ opérations de modélisation géométrique
- ▶ animation des objets
- ▶ etc.

# Réflexions lumineuses multiples

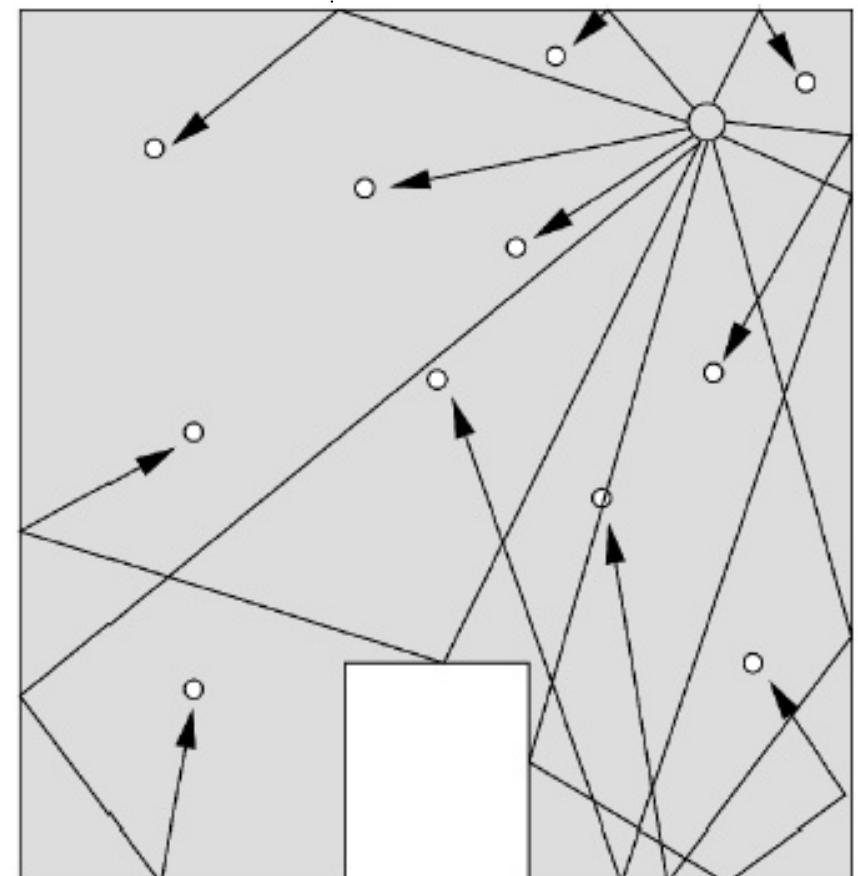
## ○ Source lumineuse

- ▶ lumière réfléchie de nombreuses fois (une infinité).
- ▶ dans toutes les directions (une infinité)
- ▶ pour chaque source...

Tableau

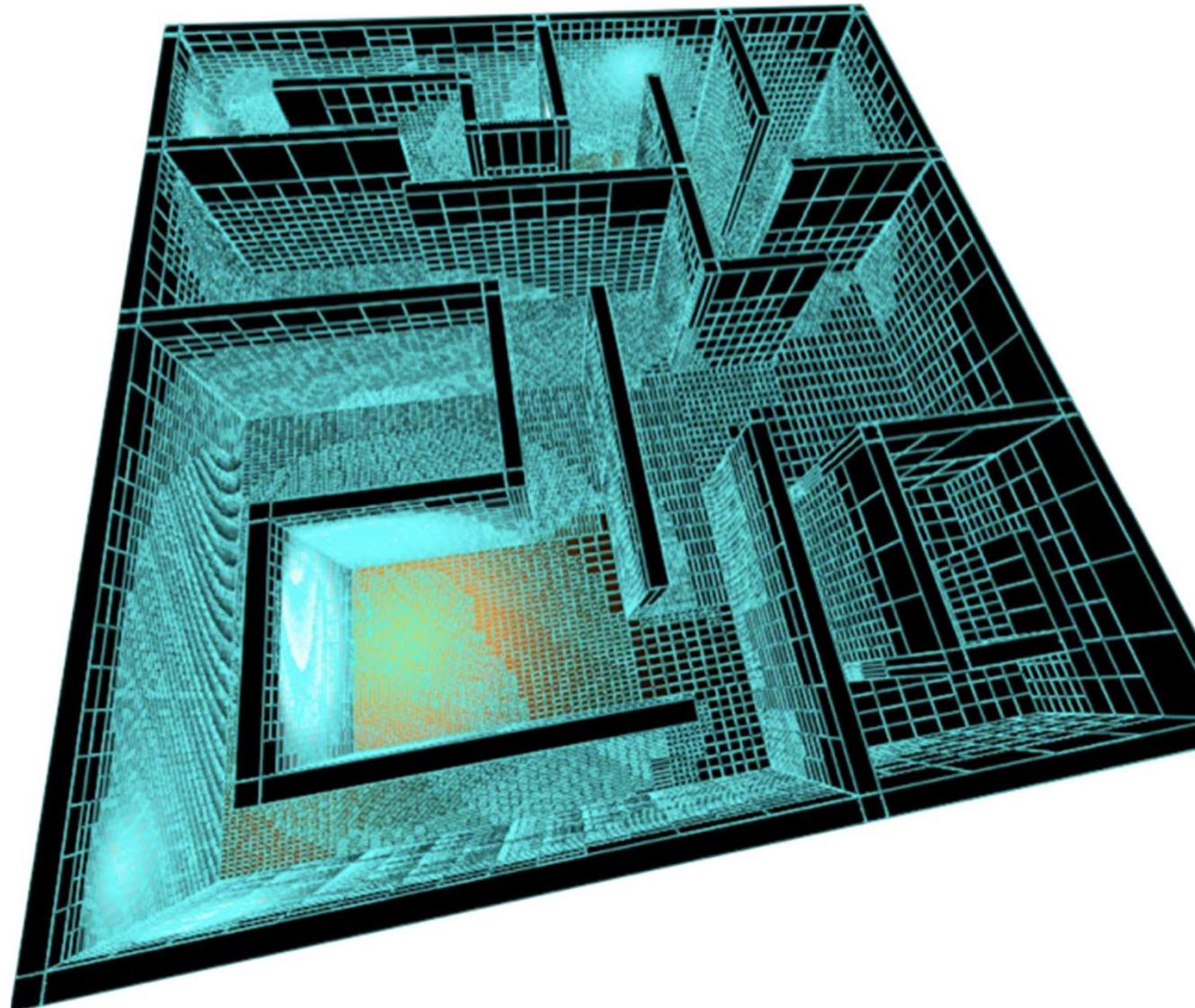
## ○ Comment simuler cela ?

- ▶ radiosité
- ▶ tracé de photons
- ▶ caches d'éclairement
- ▶ tracé de chemins (bidirectionnel)



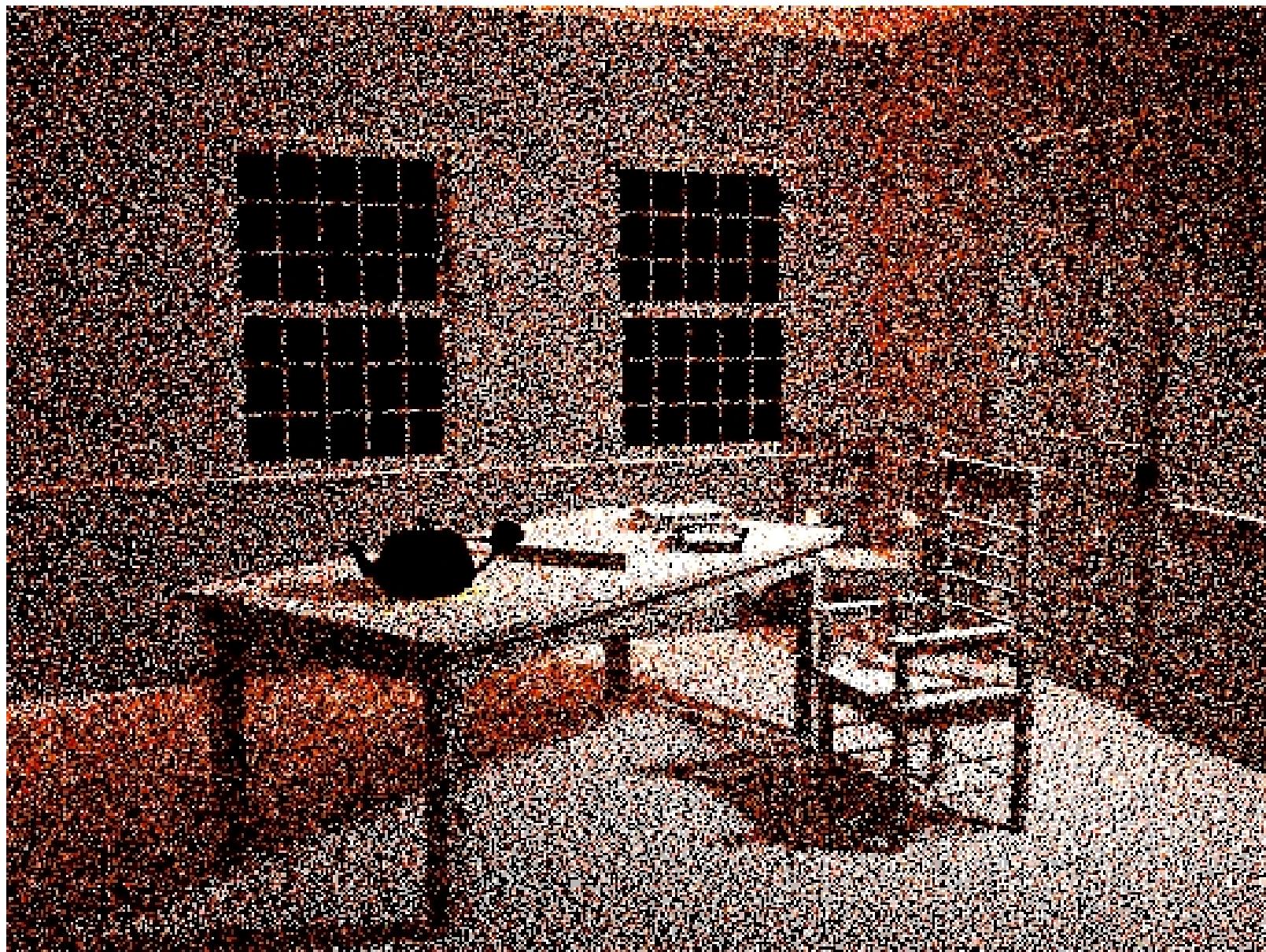
# Radiosité

---

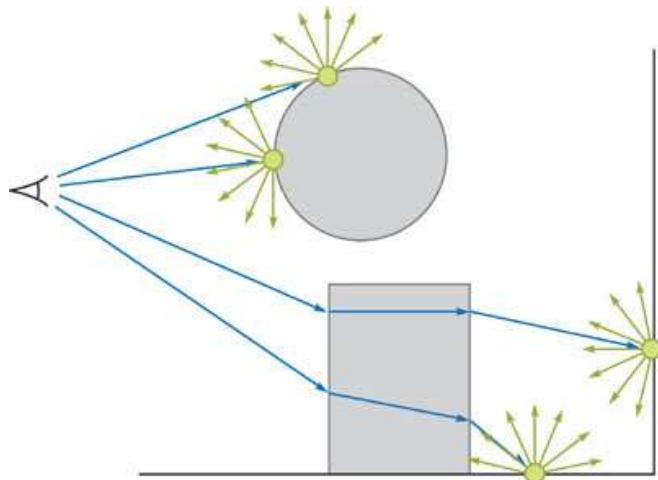


# Tracé de photons

---



# Collecte



Tracé de chemins (bidirectionnel)  
Caches d'éclairement  
Avec radiosité / tracé de photons (final gather)

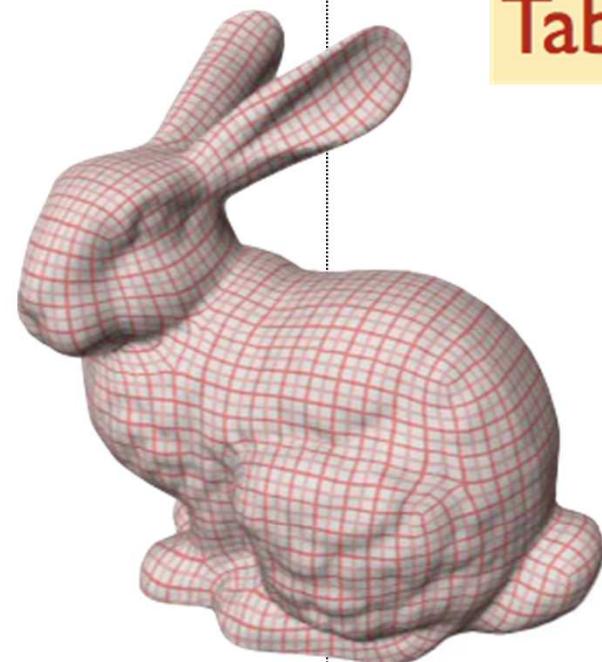


# Et les textures ?

## ① Textures images

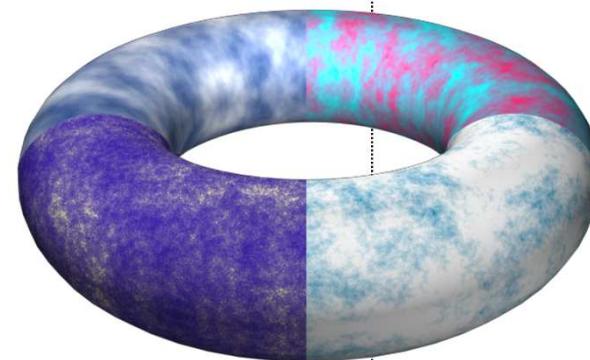
- ▶ définir un changement de repère : objet / image
- ▶ coordonnées de textures pour chaque sommet  
pas pratique à modéliser
- ▶ retrouver le texel pour un point donné
- ▶ RVB ???  
ce n'est pas une réflectance !

Tableau



## ② Textures procédurales (bruit de Perlin)

- ▶ fonctions dans le plan ou dans l'espace
- ▶ plus facile à appliquer aux objets  
plus coûteux en temps de calcul



## ○ Problème de précision des calculs

- ▶ le signal lumineux mal échantillonné par les rayons  
Plusieurs rayons par pixel (coûteux)

Tableau



Aliati-A



Aliasing



Anti-Aliasing