CNAM, spécialité Informatique
NSY116 - Multimédia & interaction humain-machine

L'image fixe (2)

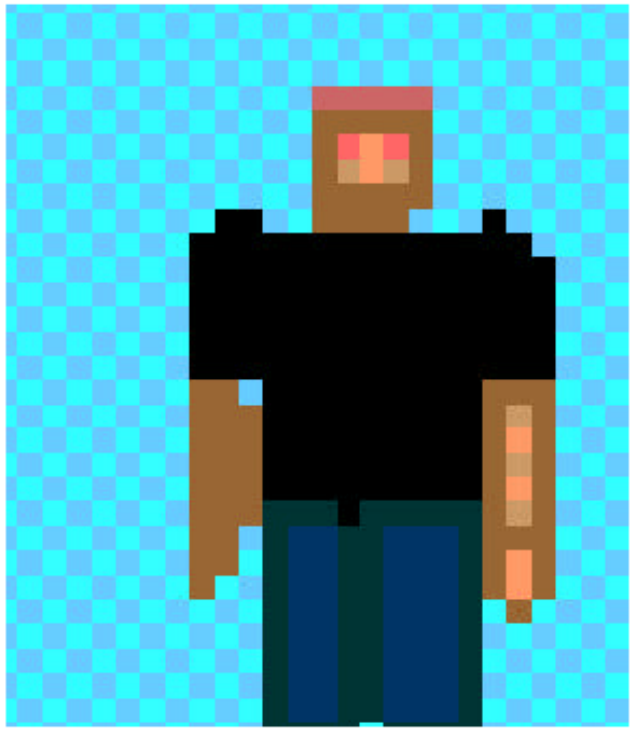
P. Cubaud <cubaud@cnam.fr>

1. L'image numérique
2. Traitement, analyse
3. Transport, compression
4. **Synthèse**

4. Synthèse d'image

- principes
- modélisation de la scène 3D
- le pipeline standard de rendu
- le lancé de rayon

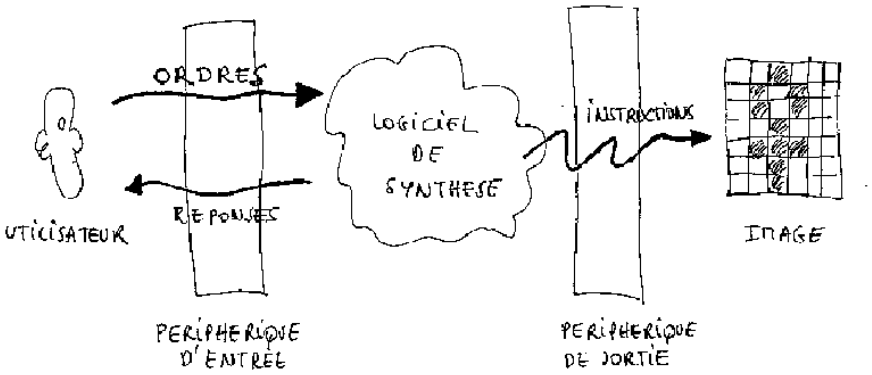
• Une image = une matrice de pixels



ici 29x25 pixels et 10 couleurs

• On veut obtenir cette image sans intervention directe sur les pixels : par programme

• Les ordres de l'utilisateur correspondent à la description de la scène que le programme doit dessiner sur l'écran



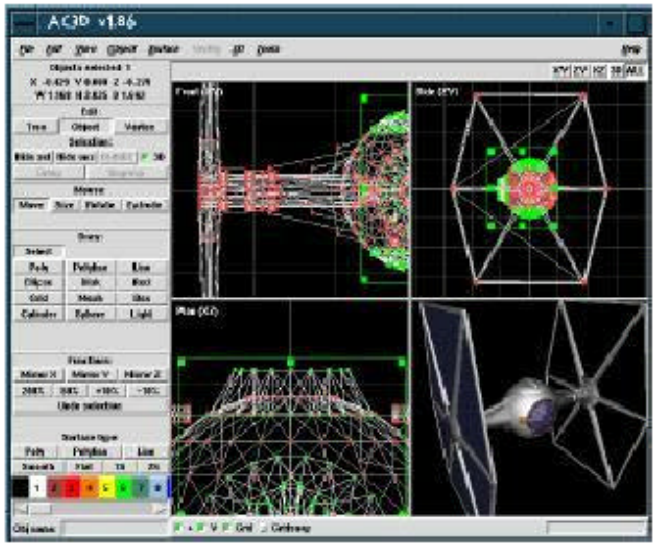
Divers degrés d'intervention de l'utilisateur :

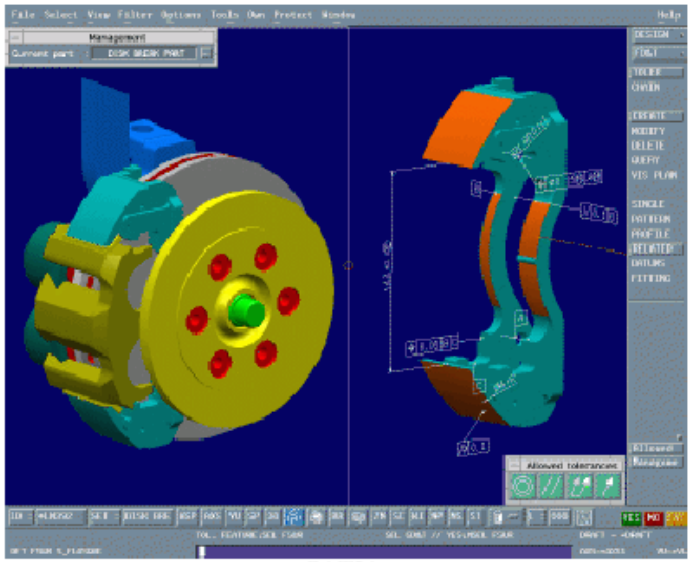
http://www.solo.com/studio/algorists.html

aucun

"je sème à tout vent" (Couchot, Bret, Tramus, 1990)

par commande





CATIA

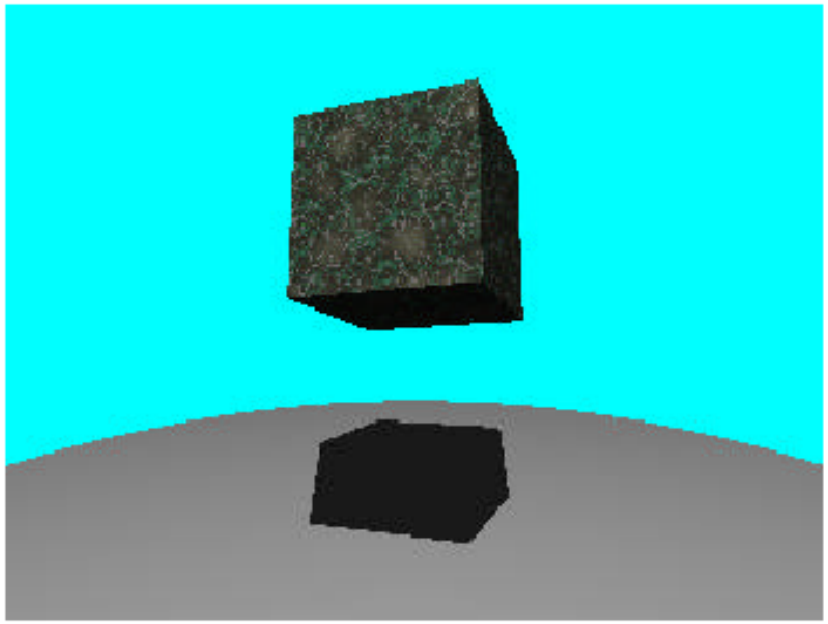
par modelage

- Par <u>langage spécialisé</u> : Description textuelle de la scène sans (trop) programmer.

  ex: VRML, POV, RayShade, Open Inventor…

```
#include "colors.inc"
#include "stones.inc"

  background { color Cyan }

  light_source { <0, 10, -10> color White}

  camera {
    location <0, 0, -7>
    look_at  <0, -1,  0>
  }

  box {
    <-1, -1,   -1>, < 1, 1, 1>
    texture {T_Stone25}
    rotate y*20
    rotate x*20
  }

  cylinder {
   <0,-5,0>, <0,-10,0>, 10
   pigment{White}
  }
```

Ex. de texte POV



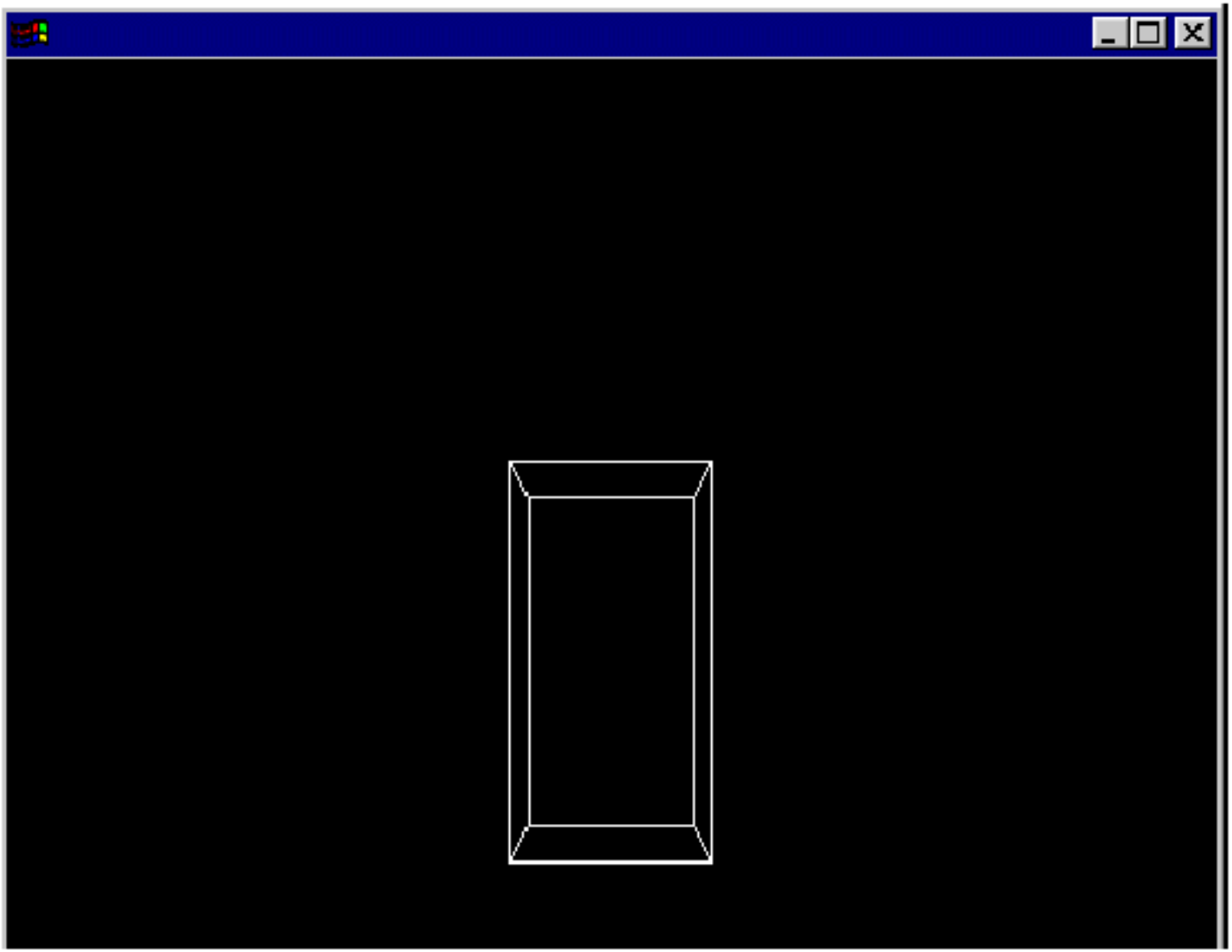
- Par programme : l'utilisateur écrit un programme décrivant une scène en s'appuyant sur une bibliothèque de fonctions 3D

  ex: openGL, Direct 3D, PHIGS, Java3D …

```
Extrait du "red book" d'OpenGL - fichier cube.c

#include <GL/glut.h>
#include <stdlib.h>

void init(void) {
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

void display(void) {
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glLoadIdentity ();/* clear the matrix */
   /* viewing transformation  */
   gluLookAt (0.0, 0.0, 5.0, 0.0,
              0.0, 0.0, 0.0, 1.0, 0.0);
   glScalef (1.0, 2.0, 1.0);
  /* modeling transformation */
   glutWireCube (1.0);
   glFlush ();
}
```

```
void reshape (int w, int h){
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
   glMatrixMode (GL_MODELVIEW);
}

void keyboard(unsigned char key, int x, int y){
   switch (key) { case 27: exit(0); break; }
}

int main(int argc, char** argv){
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutKeyboardFunc(keyboard);
   glutMainLoop();
   return 0;
}
```

Résultat …



Différentes techniques de rendu :

- Le filaire ("wireframe")
  - Monochrome
  - Coloré
  - Prise en compte de la distance à l'observateur ("depth cueing")

- Prise en compte des caractéristiques réflexives des surfaces
  - Coloriage uniforme
  - Une couleur par face ("flat shading")
  - Interpolation de Gouraud d'une face à l'autre
  - Interpolation de Phong (normale à la face)

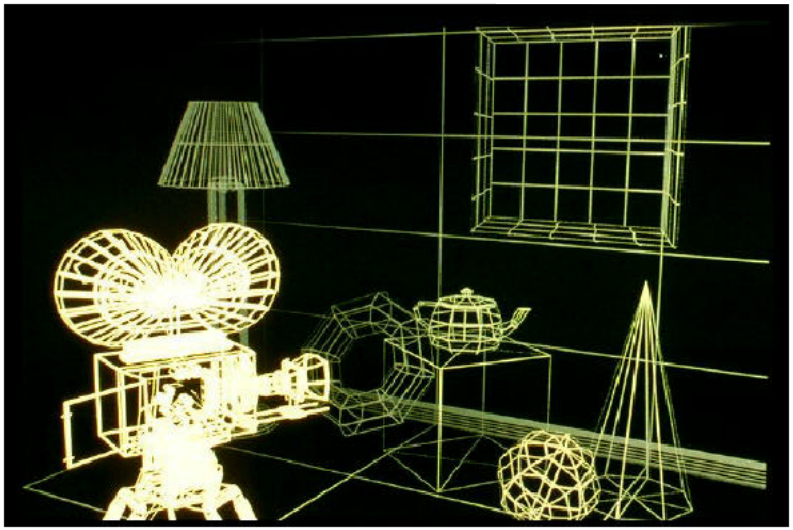
  la méthode la plus commune => logique câblée, temps réel

- Prise en compte des interactions entre objets
  - Tracé des ombres
  - Sources lumineuses non ponctuelle
  - Transparence de certains objets

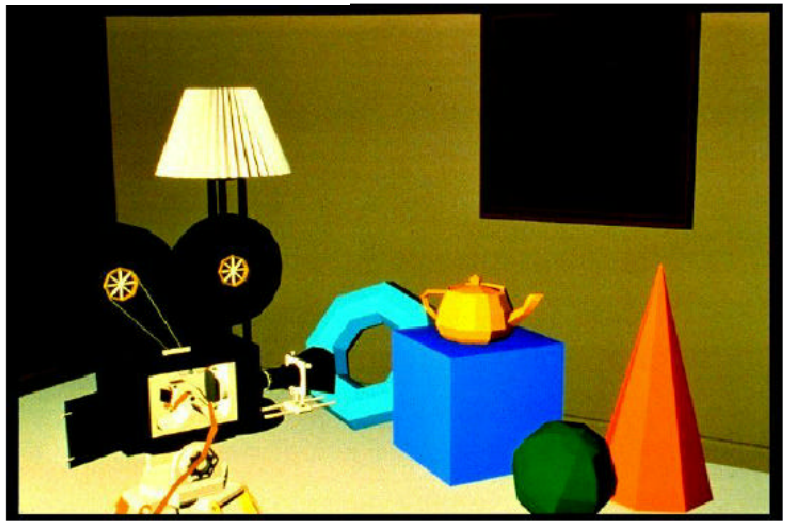
  suivi de rayons ("ray tracing"), radiosité, méthodes mixtes

  => encore coûteux

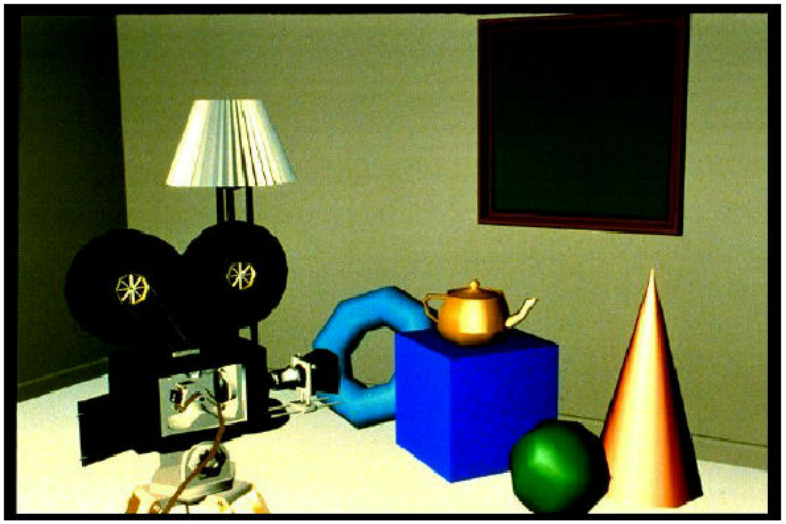
Série "Shutterburg" de PIXAR (sous Renderman)



filaire + depth cueing



coloriage à plat (flat shading)



lissage de Gouraud

plaqué de textures

lancé de rayon

Steve Anger "Philco 6Z4" 1993 avec POVray http://www.povray.org/

Radiosité - S. Feldman, J. Wallace Univ. Cornell

Modélisation de la scène 3D

- Quels objets veut-on représenter ?

  - des objets existants, dont on pourrait décrire toutes les caractéristiques de manière exhaustive

  - des objets nouveaux, dont les caractéristiques sont connues partiellement et proviennent d'un calcul (ou d'un modelage)

- Comment va-t'on décrire les objets ?

  Essentiellement par leur <u>forme</u> (volumes)

  * volume variable ou non ? solide □ liquide □ gaz

  * si solide : déformable ou non ?

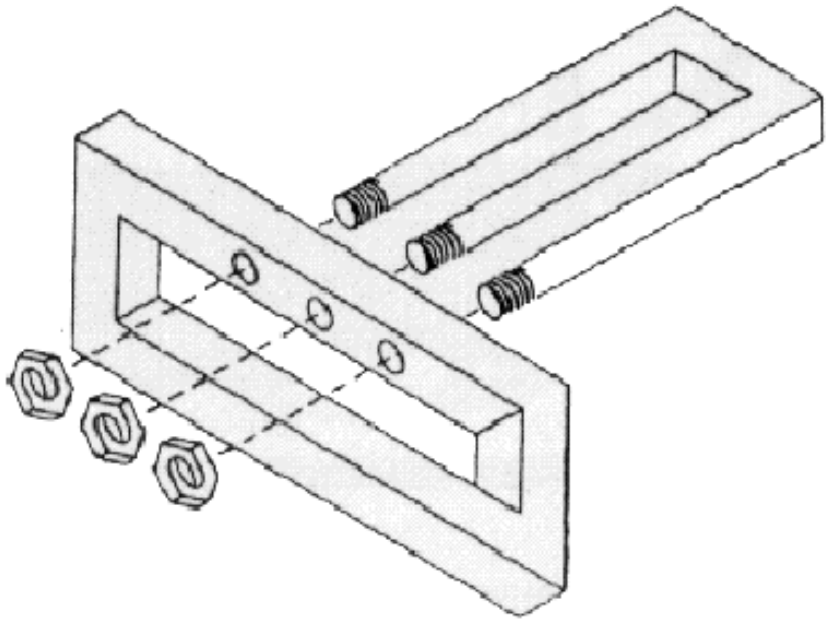
  cafetière □ étoffe □ visage □ tas de sable

  * si solide non déformable : décomposable en solides (ou surfaces) élementaires ?

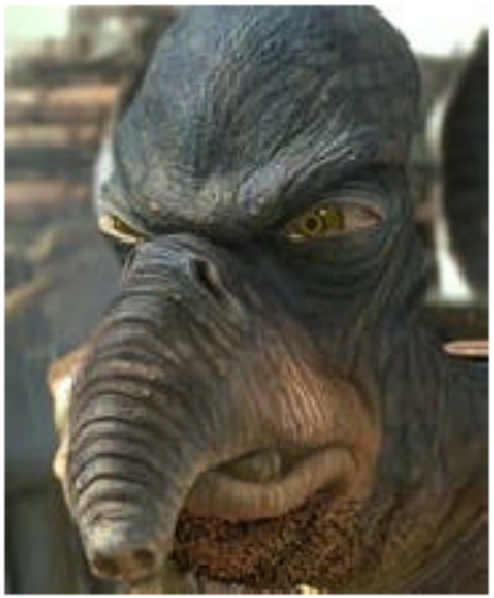
  forêt = ens. d'arbres, arbre = un tronc et des branches, tronc = un cylindre

  branche = un arbre…

• Les objets sont décrits par les frontières de leurs faces (supposées définies…)



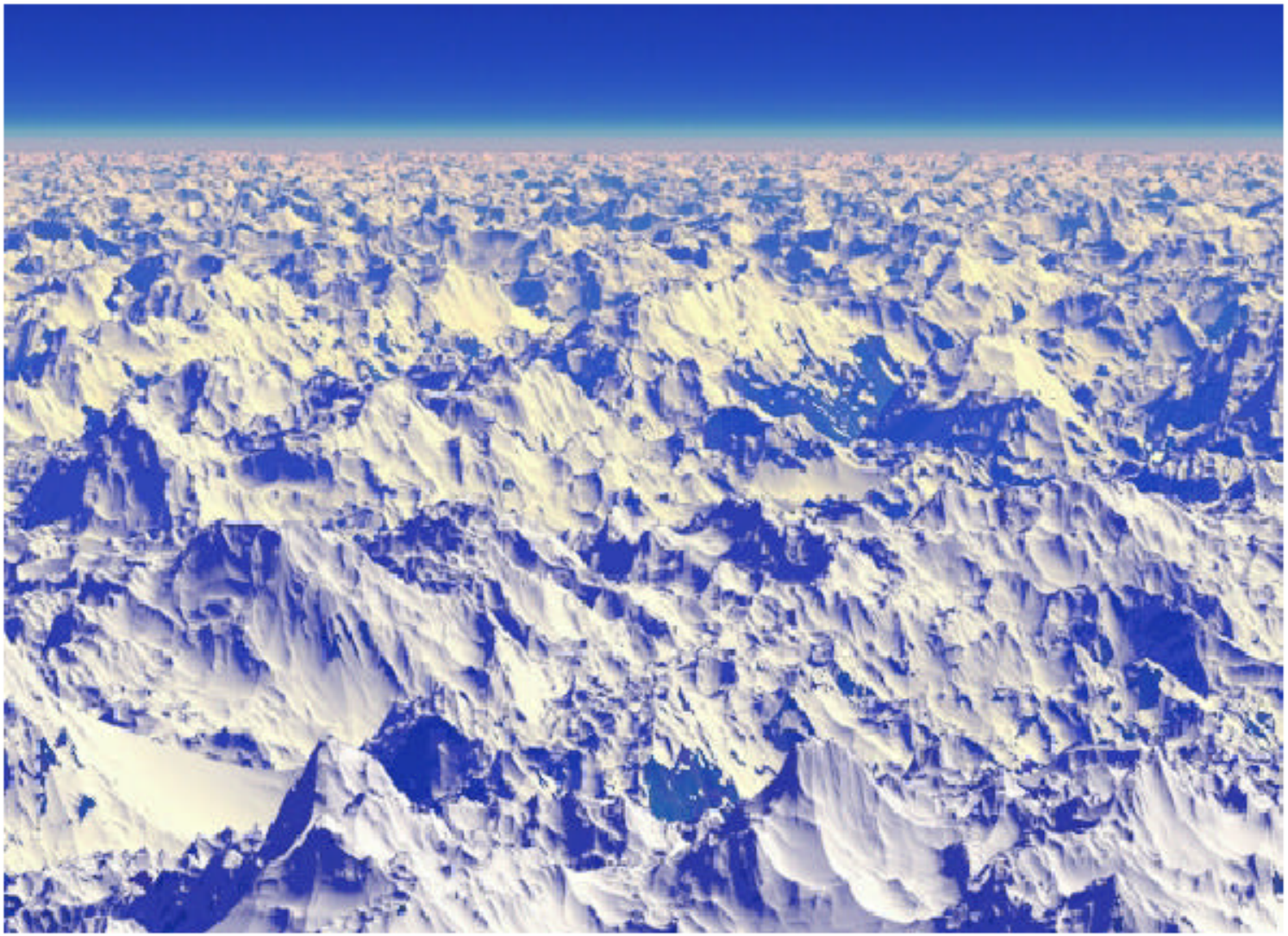
Don Mackey "Objet Impossible" in Skywriter 1966



En pratique : des modèles très restreints : Faces planes, frontières polygonales

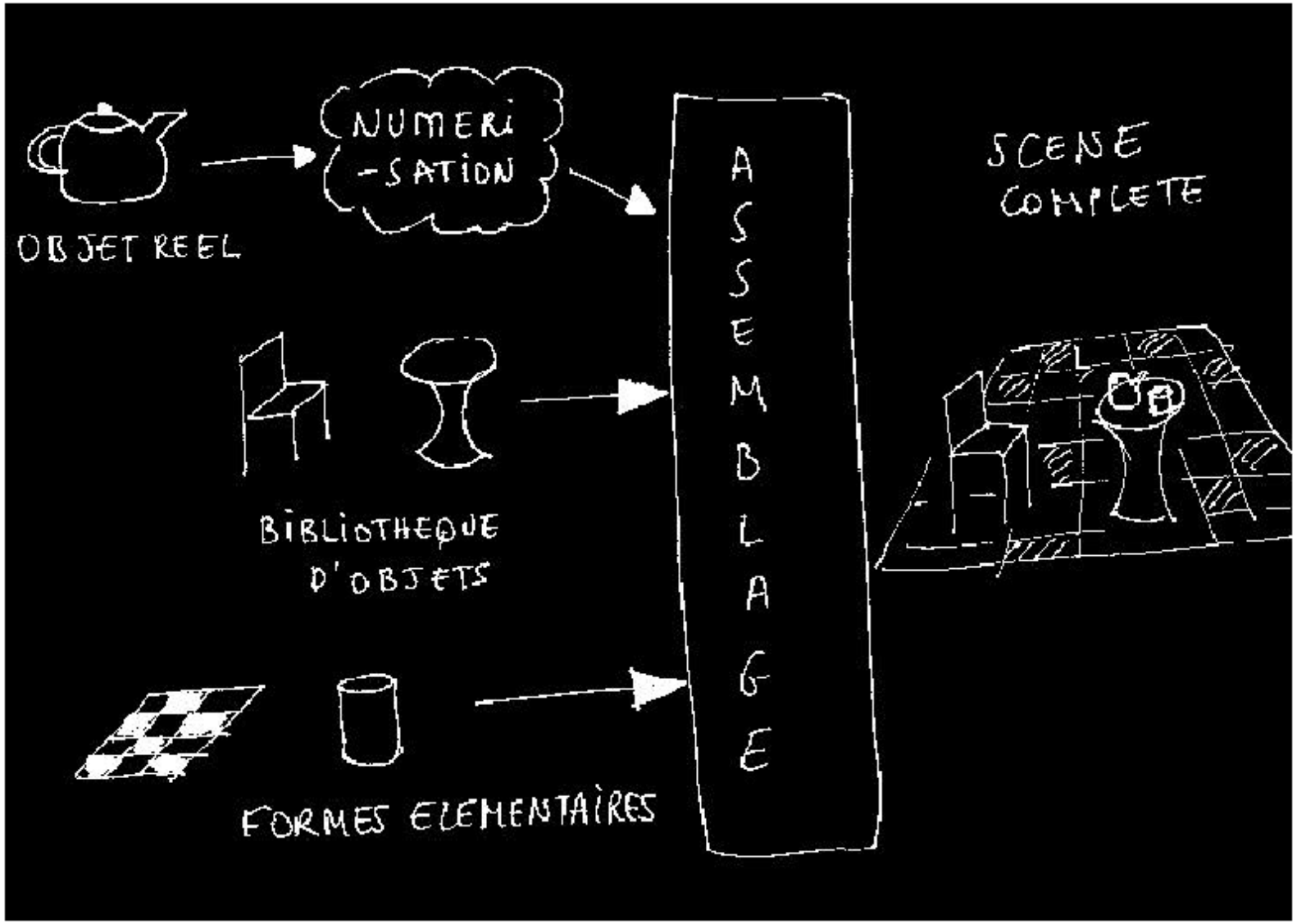
• On ajoute des "effets" pour prendre en compte les détails des objets et des conditions de visualisation

- Contre exemple : méthodes procédurales



F.K. Musgrave "Alps" in [Ebert et al.]

Assemblage de la scène 3D



- Description des surfaces élementaires : plusieurs méthodes

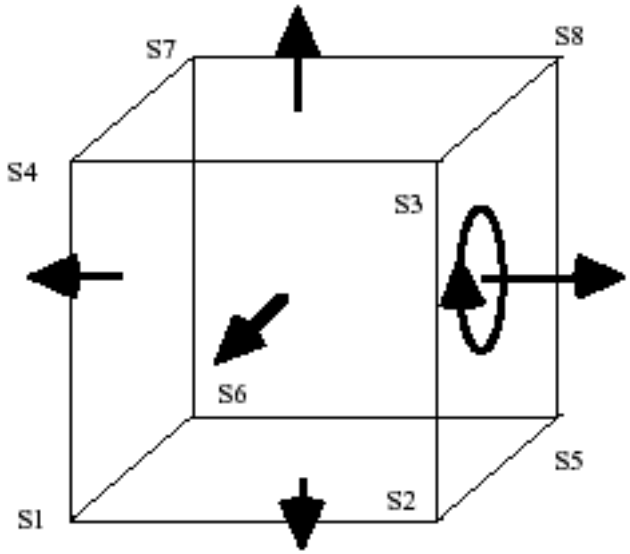
  - **explicite,** par liste de sommets (polyèdres)
  - description d'un **profil** (surfaces de révolution) ou d'une coupe (prismes, "sweeps")
  - par **points de contrôles** et un modèle de surface (Bézier, Coons, B-splines, …)
  - **implicite,** par algorithme (fractales, modèles stochastiques)

- Opérations d'assemblage

  - Transformations **géométriques** : translation, rotation, homothétie…
  - Opérateurs **booléen** : union, intersection, différence

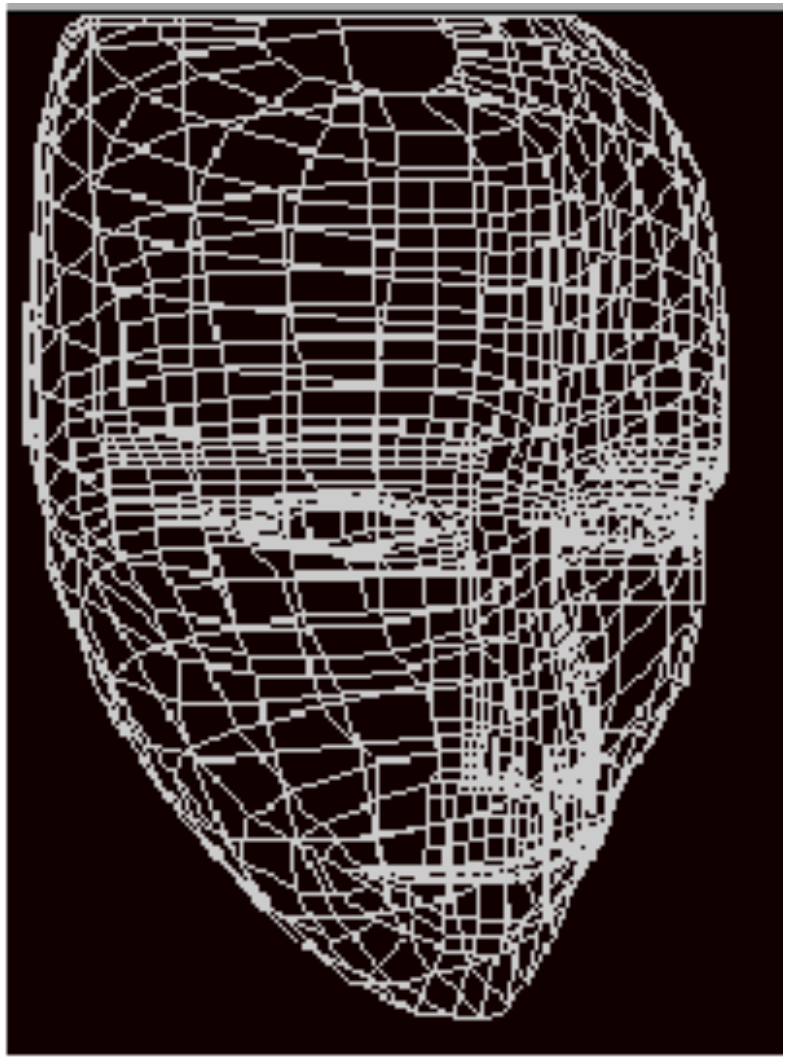
⇒ la scène 3D peut se décrire comme un **graphe**
(ou plus simplement comme un **arbre**) avec des nœuds opérateurs
et des racines primitives graphiques

Polyèdres



TS	X	Y	Z
1	0	0	0
2	0	1	0
3	1	1	0
4	1	0	0
5	0	1	1
6	0	0	1
7	1	0	1
8	1	1	2

TF	
1	1, 2, 3, 4
2	2, 3 8, 5
3	5, 8, 7, 6
4	6, 7, 4, 1
5	3, 4, 7, 8
6	1, 2, 5, 6

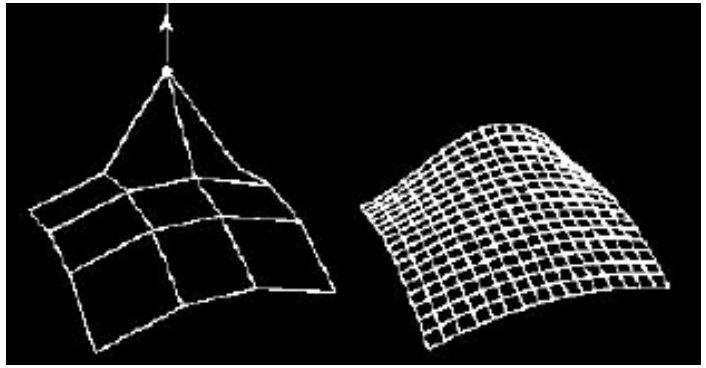


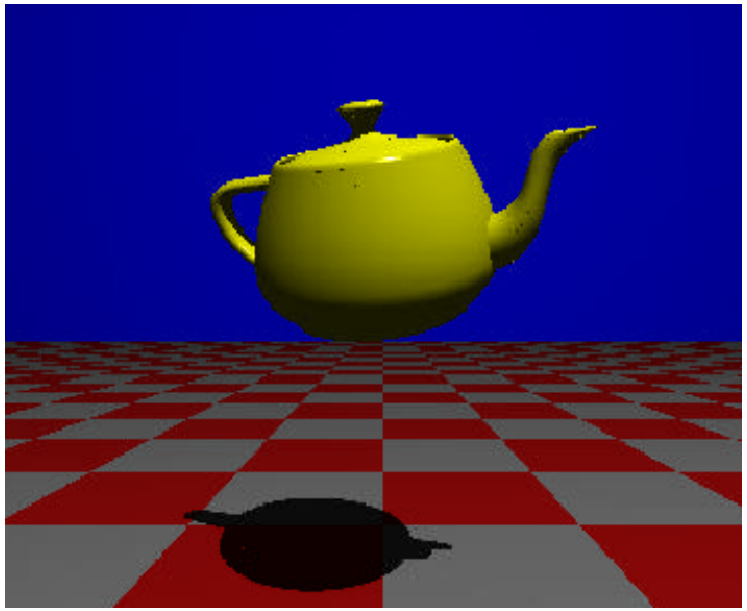
-choix du repère (main droite ou gauche)
-modèle plus simple (efficace) pour les bandes de triangles

Tesselation : ex. du patch de Bezier

16 points de contrôle

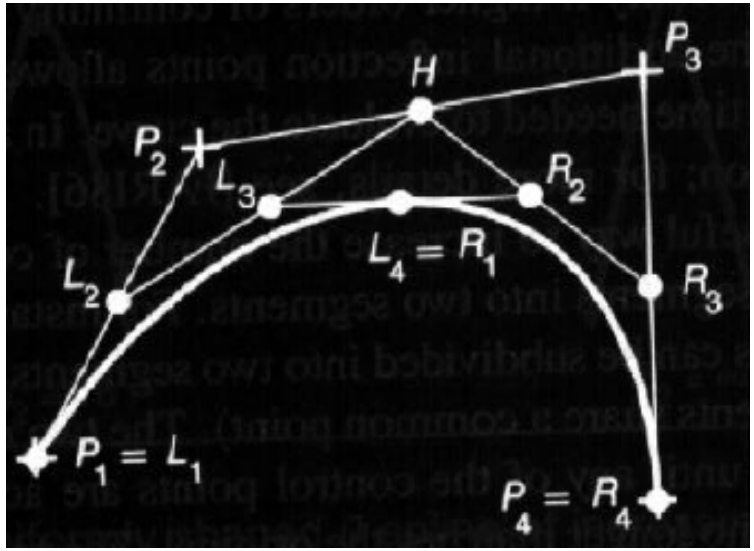
Polynôme de degré 3 en (x,y)





$$f(u,v) = (u^3 \;\; u^2 \;\; u \;\; 1) M_B \begin{pmatrix} \vec{P}_{00} & \cdot\cdot & \vec{P}_{03} \\ \cdot & \cdot\cdot & \cdot \\ \cdot & \cdot\cdot & \cdot \\ \vec{P}_{30} & \cdot\cdot & \vec{P}_{33} \end{pmatrix} M_B^T \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix}$$

$$\text{MB} = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$



tesselation rapide

Transformations géométriques usuelles

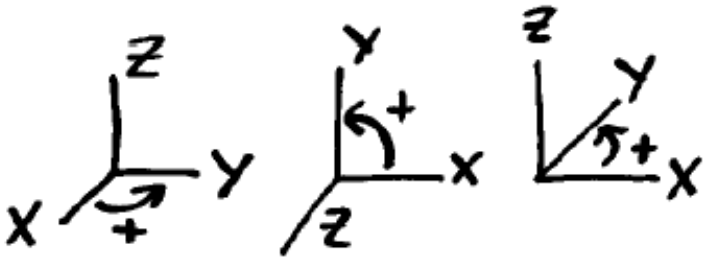
- utilisation des coodonnées homogènes (3D+échelle)

- Translation : $\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$

- Homothétie : $\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$

- Rotations par rapports aux axes X, Y et Z

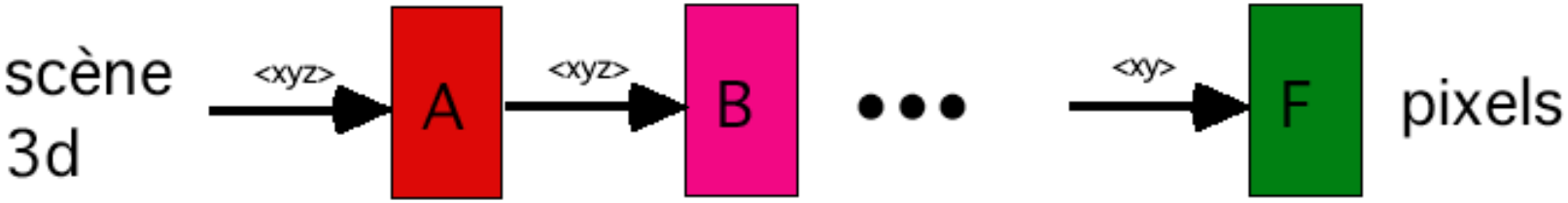
Si répère "main droite" !!! θ positif en allant de X à Y



$$R_Z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_X(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_Y(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Le pipeline standard de rendu



A- transformation des coordonnées locales en coordonnées de la scène (ou du"monde")

B- transformation en volume de vue canonique (cube 0-1)

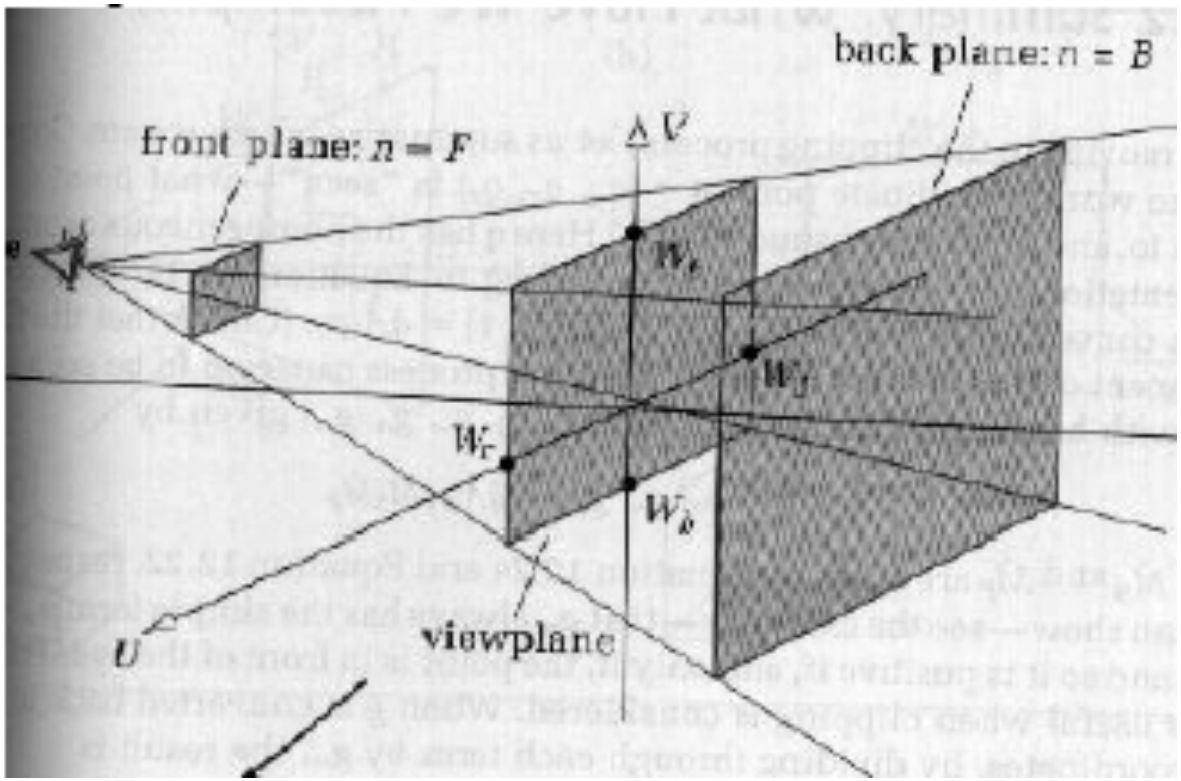
C- élimination des objets hors volume de vue et découpe (clipping) des objets à la frontière

D- élimination des faces cachées

E- projection en 2D

F- coloriage des faces

Volume de vue

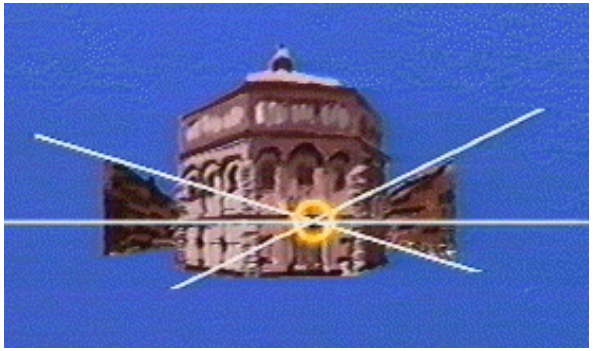


Nombreuses variantes. Un ex:

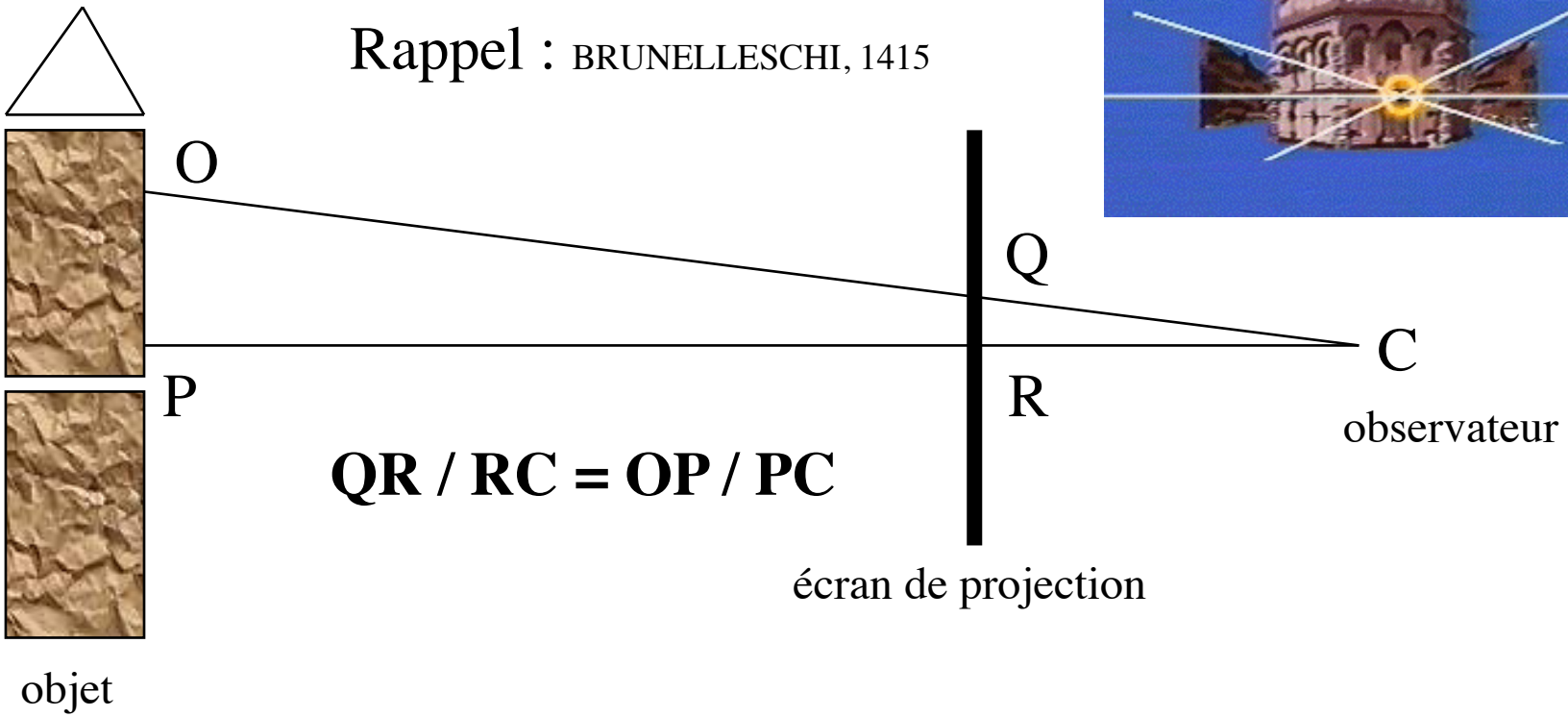
-vecteur de vue
-vecteur orientation
-pt de ref écran
-dim. scène écran
-distance oeil/écran
-avant et arrière plan

On calcule les coordonnées caméra des objets à partir de leurs coordonnées scène => matrice 4x4 de changement de repère

Projection : ex. de la perspective



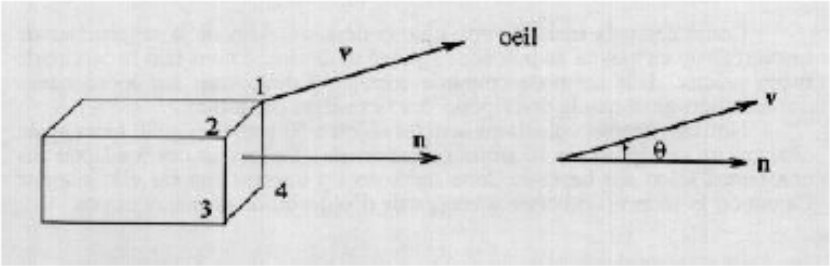
Rappel : BRUNELLESCHI, 1415



$$M_{per} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$$ avec d = distance(ecran, caméra) =RC

Elimination des faces cachées

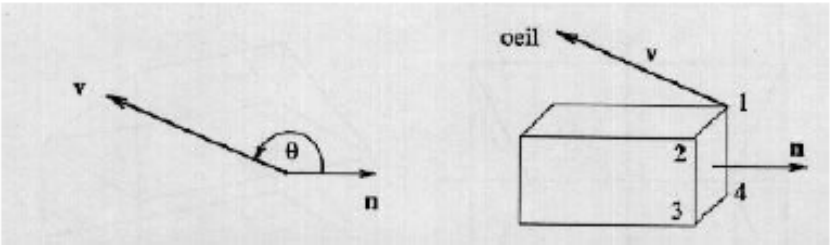
- Face visible



[DONY] p. 323

l'angle est aigu

- Face invisible



l'angle est obtu

- Test de visibilité

  - calculer le vecteur normal N à la face

  - déterminer le vecteur de vision V pour cette face

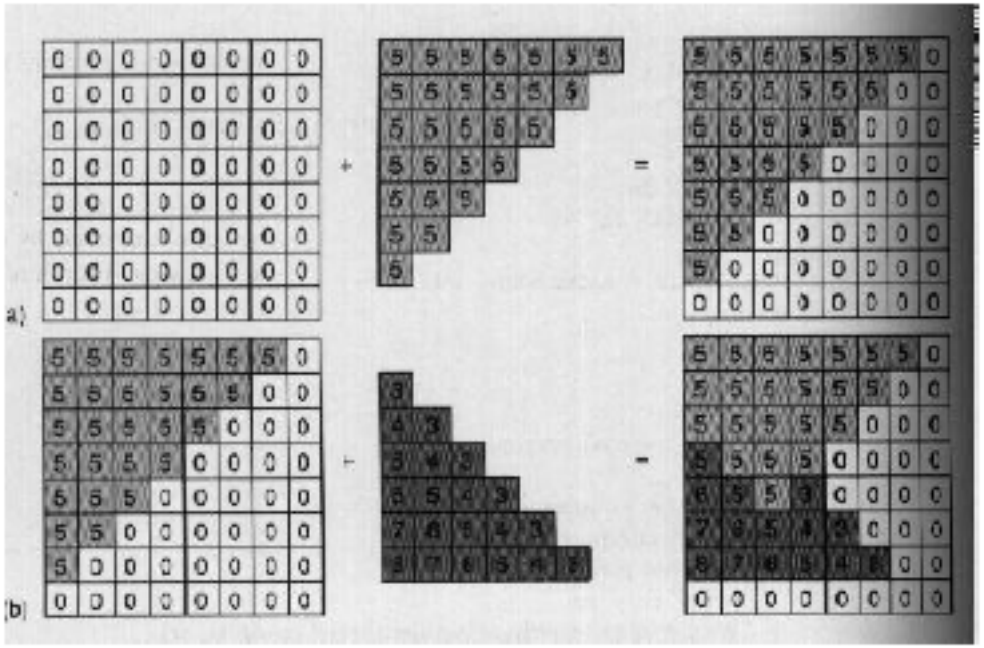
  - déterminer l'angle

  - s'il est obtu, la face est invisible

- A utiliser systématiquement avant tout algorithme d'élimination des surfaces cachées

Algorithme du Z-buffer (Catmull, 1974)

Suppose l'existence d'une matrice (Z-buffer) associant à chaque pixel de l'ecran, la profondeur (coordonnée Z) de la portion de facette concernée.



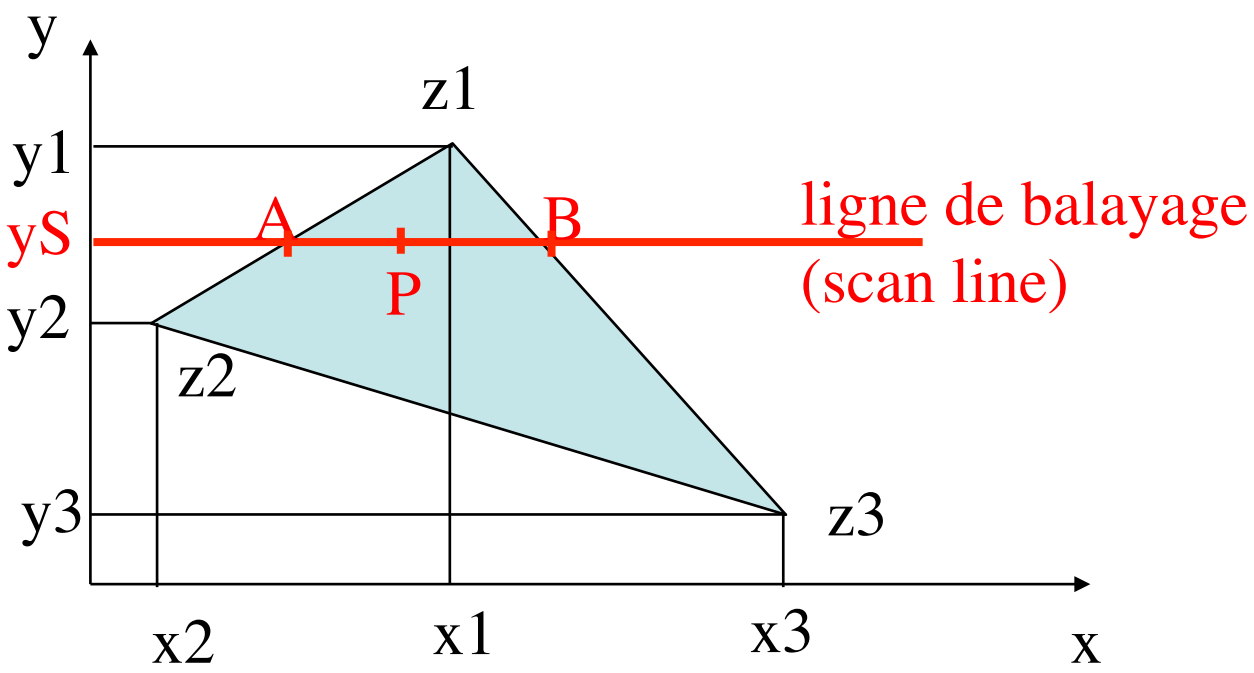
[FOLEY] p. 670

```
pour chaque face F faire
    pour chaque pixel (i,j) de F faire
        pz = valeur de Z pour la face F en (i,j)
        si pz < Zbuffer[i,j] alors
            Zbuffer[i,j] = pz
            pixel (i,j) = couleur de la face F
        fin si
    fin pour
fin pour
```

souvent : face = polygone (triangle)

***** attention à l'aliasing ....**

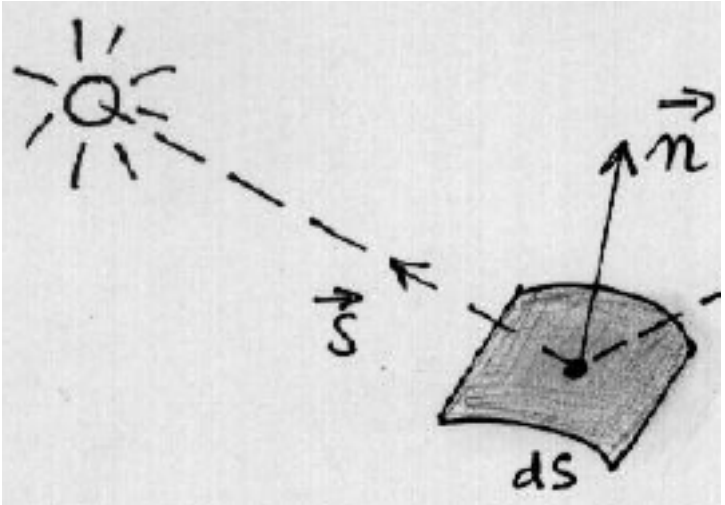
Interpolation des profondeurs

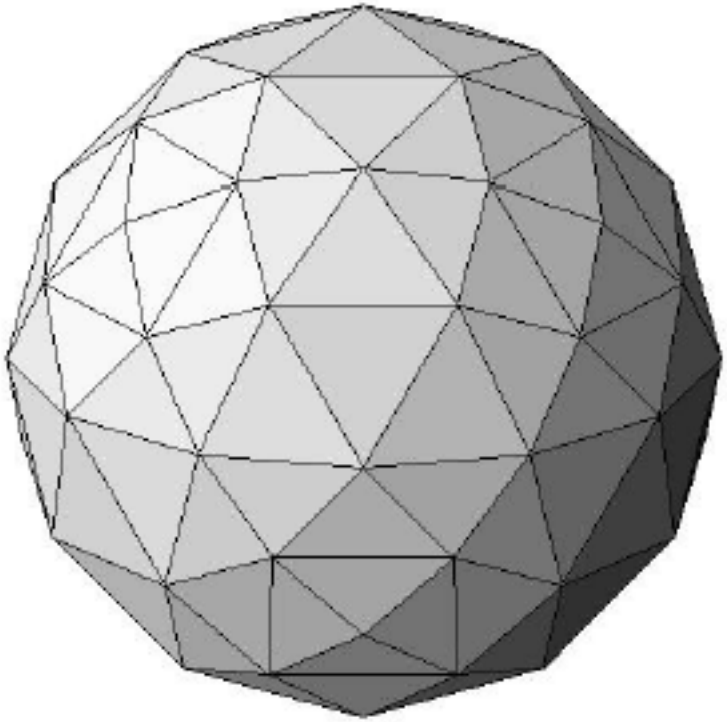


$$z_A = z_1 + (z_1 - z_2)\frac{y_1 - y_S}{y_1 - y_2} \qquad z_B = z_1 + (z_1 - z_3)\frac{y_1 - y_S}{y_1 - y_3}$$

$$z_P = z_A + (z_A - z_B)\frac{x_A - x_P}{x_A - y_B}$$

Coloriage : à plat (Bouknight, 1970)

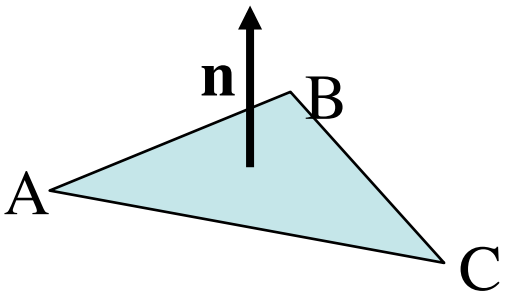




Utilisation de la loi de Lambert :

$$I_{diff} = I_P K_d (\vec{n}.\vec{s}) = I_P K_d \cos\theta$$

Calcul de la normale par produit vectoriel :



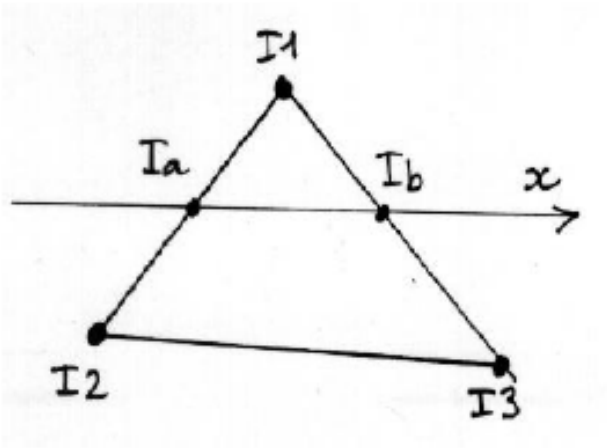
$$\vec{n} = \overrightarrow{AB} \circ \overrightarrow{AC}$$

$$\begin{cases} x_n = (yB - yA)(zC - zA) + (zB - zA)(yC - yA) \\ y_n = (zB - zA)(xC - xA) + (xB - xA)(zC - zA) \\ z_n = (xB - xA)(yC - yA) + (yB - yA)(xC - xA) \end{cases}$$

Coloriage de Gouraud (1971)

Principe : interpolation linéaire de l'intensité pour éliminer les discontinuités

Une technique similaire à l'interpolation de la profondeur pour le Z-Buffer :

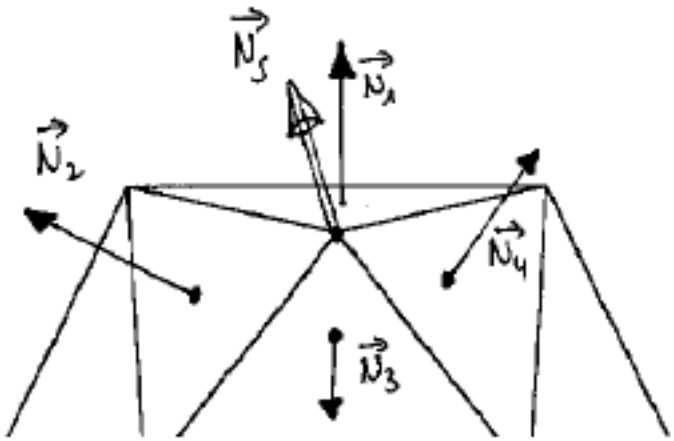


$$I_A = I_1 + (I_1 - I_2)\frac{y_1 - y_S}{y_1 - y_2}$$

$$I_B = I_1 + (I_1 - I_3)\frac{y_1 - y_S}{y_1 - y_3}$$

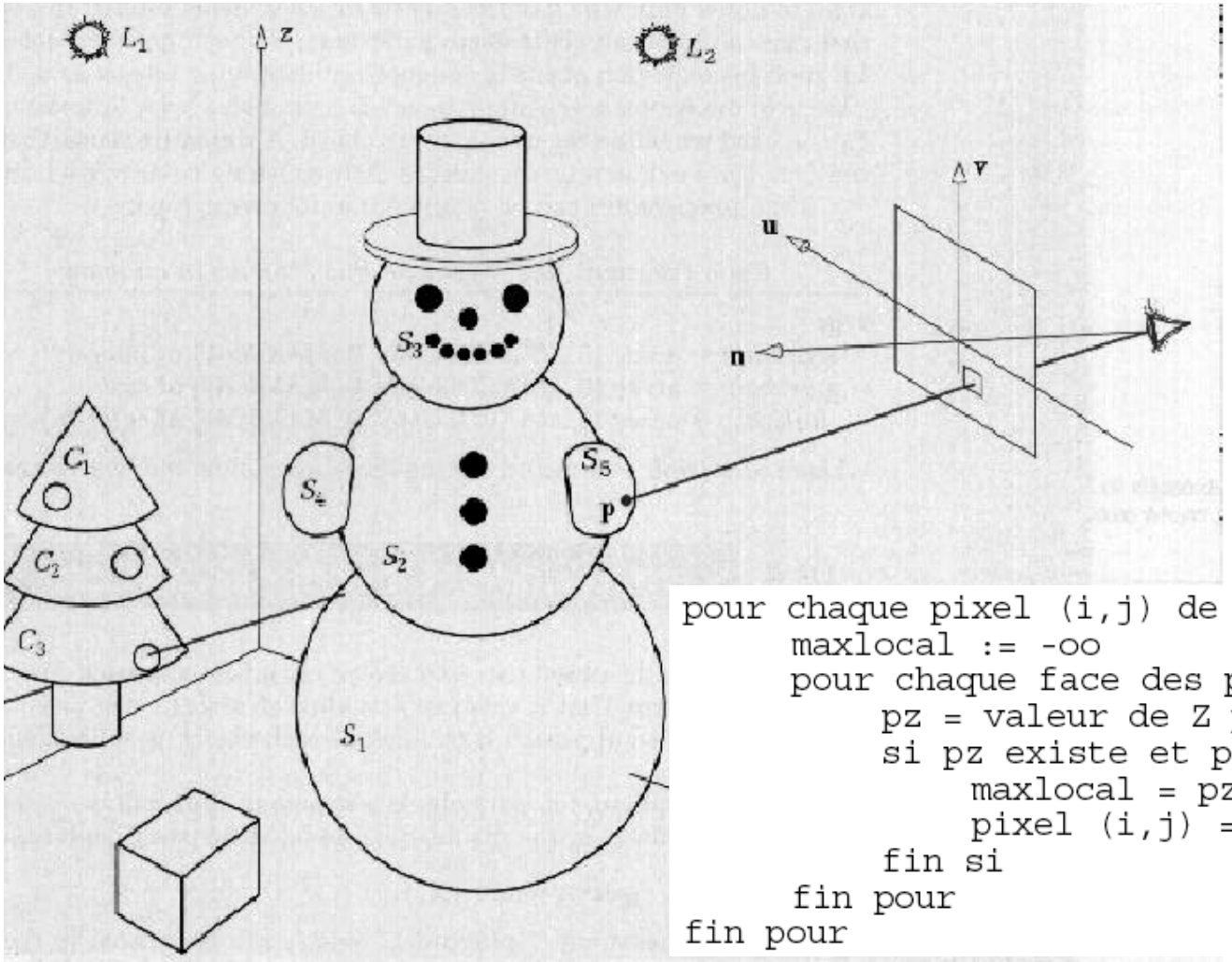
$$I_P = I_A + (I_A - I_B)\frac{x_A - x_P}{x_A - y_B}$$

L'intensité aux sommets est obtenue par interpolation des normales des faces adjacentes.



$$\vec{N}_S = \frac{\vec{N}_1 + \vec{N}_2 + \vec{N}_3 + \vec{N}_4}{\left\|\vec{N}_1 + \vec{N}_2 + \vec{N}_3 + \vec{N}_4\right\|}$$

Le lancé de rayons ("ray tracing", Appel, 1968)




```
pour chaque pixel (i,j) de l'écran faire
     maxlocal := -oo
     pour chaque face des polyèdres faire
          pz = valeur de Z pour la face en (i,j)
          si pz existe et pz >= maxlocal alors
               maxlocal = pz
               pixel (i,j) = couleur de la face
          fin si
     fin pour
fin pour
```

[HILL] p. 619

***** attention à l'aliasing .... (bis)**

On se ramène à des calculs d'intersection entre un "rayon lumineux" et les volumes de la scène

- Equation de la droite oeil - pixel

(x0,y0,z0) : coordonnées de l'oeil

(x1,y1,z1) : coordonnées 3D du centre du pixel (dépend du type de projection utilisée)

Représentation paramétrique :

$$\begin{cases} x = x_0 + t(x_1 - x_0) = x_0 + t\Delta_x \\ y = y_0 + t(y_1 - y_0) = y_0 + t\Delta_y \quad 0 \le t \le 1 \\ z = z_0 + t(z_1 - z_0) = z_0 + t\Delta_z \end{cases}$$

Exemple : intersection avec une sphère

Equation de la sphère de coord (a, b, c) et de rayon R :

$$(x-a)^2+(y-b)^2+(z-c)^2=R^2$$

Si une intersection existe, ses coordonnées vérifient les 2 équations (droite et sphère). On arrive à :

$$At^2+Bt+C=0 \quad \text{avec :}$$

$$A=\Delta_x^2+\Delta_y^2+\Delta_z^2$$

$$B=2\left(\Delta_x(x_0-a)+\Delta_y(y_0-b)+\Delta_z(z_0-c)\right)$$

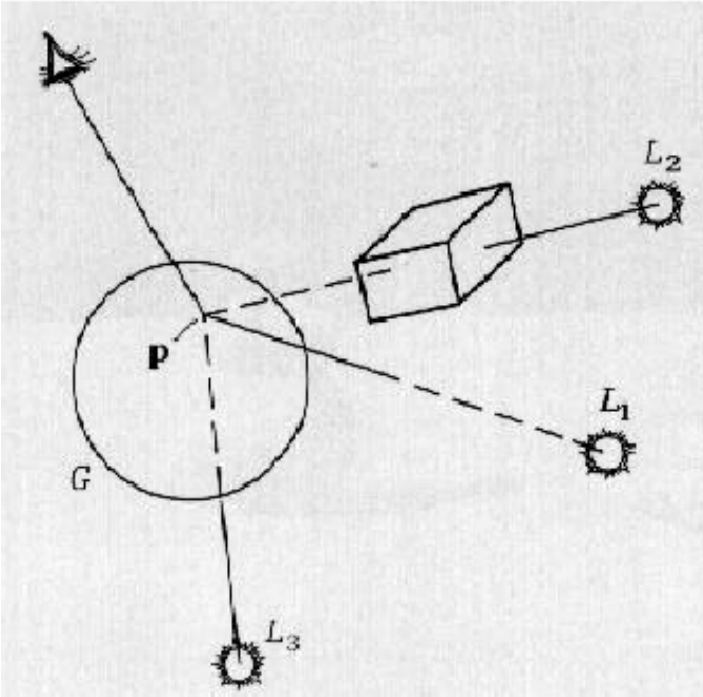
$$C=(x_0-a)^2+(y_0-b)^2+(z_0-c)^2-R^2$$

Si pas de solution : pas d'intersection

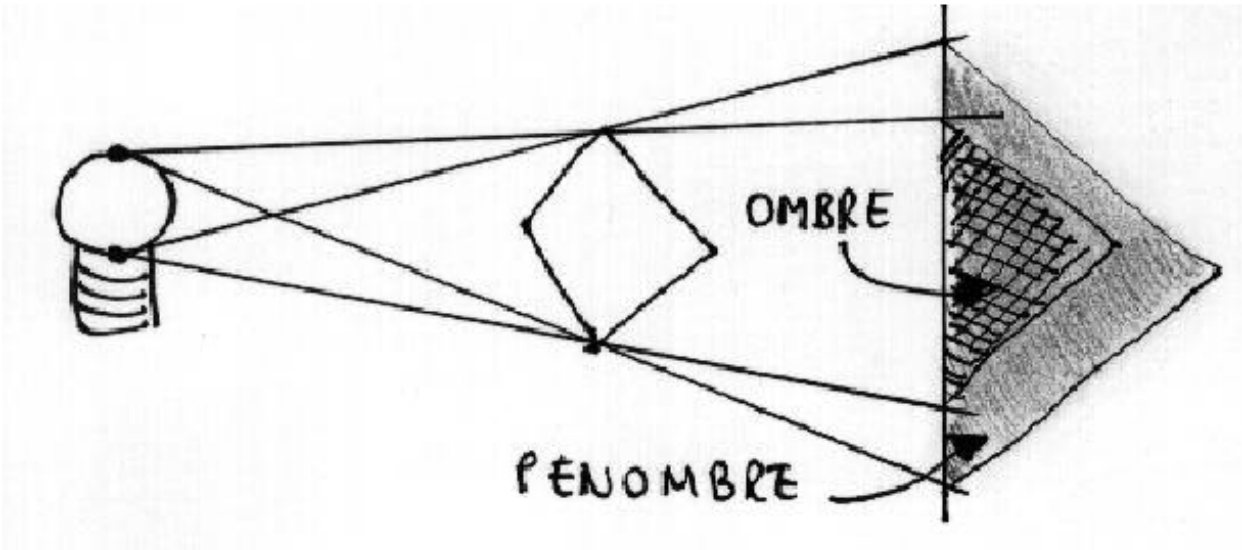
Si une seule : point tangent à la sphère

Sinon : les deux points sont trouvés (t mini est le 1er)

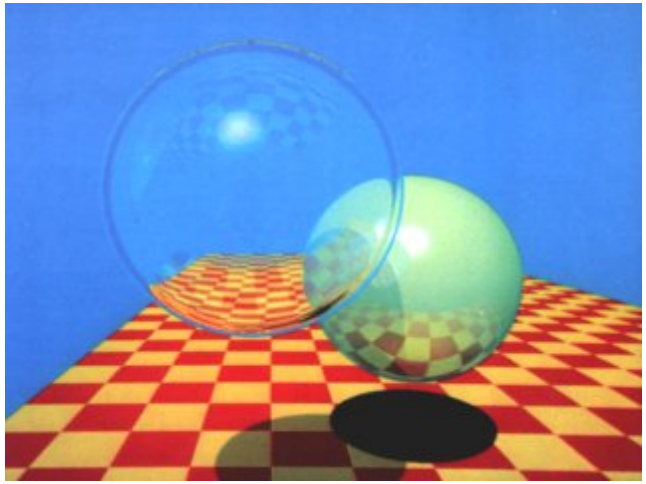
Coloriage et ombre

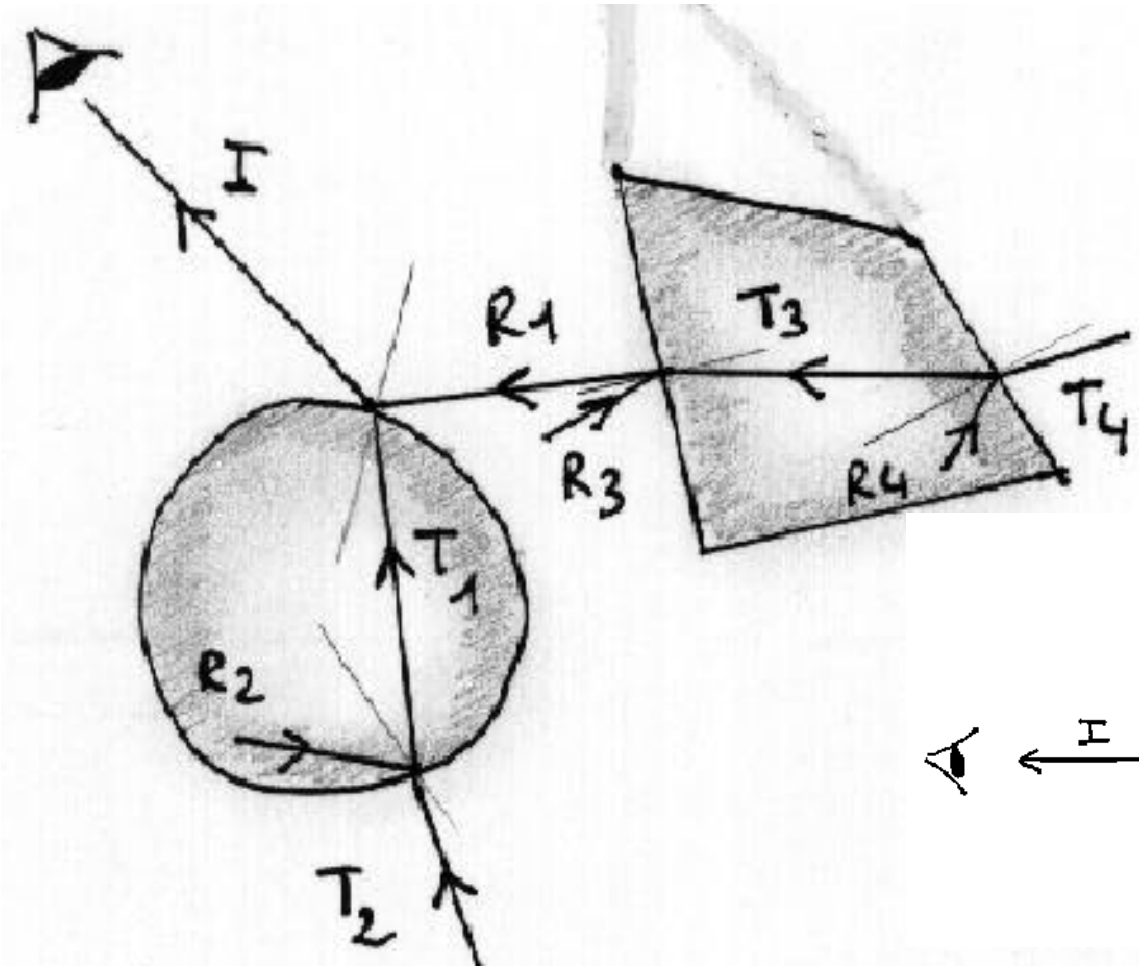


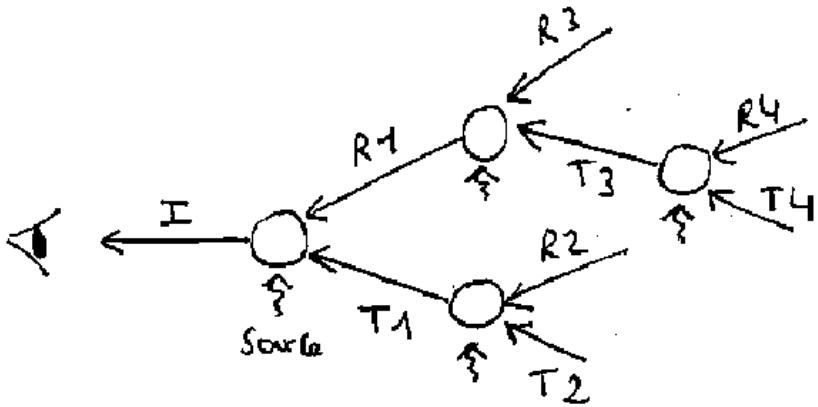
- On applique au point P un modèle de coloriage (flat shabing, Gouraud...) avec ou sans texture
- On "tire" un second rayon du point d'intersection vers la (les) source(s) lumineuse(s) (supposée ponctuelle)
- pour les points dans l'ombre, seule la lumière ambiente est prise en compte



Lancé de rayons récursif (Whitted, 1980)



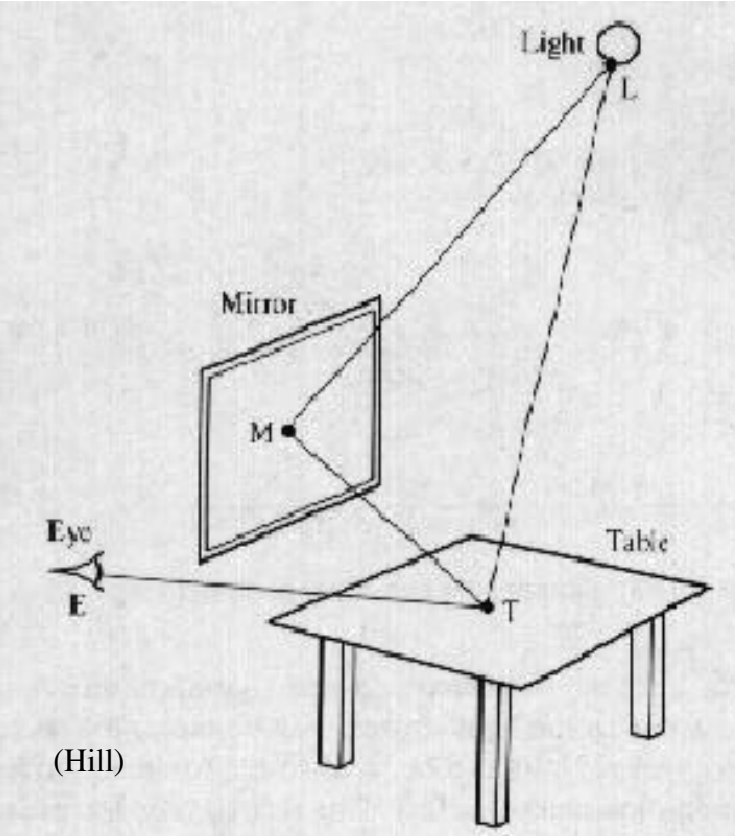




Dans le modèle de Whitted, les intensités R et T sont inversement proportionnelles aux longueurs des rayons.

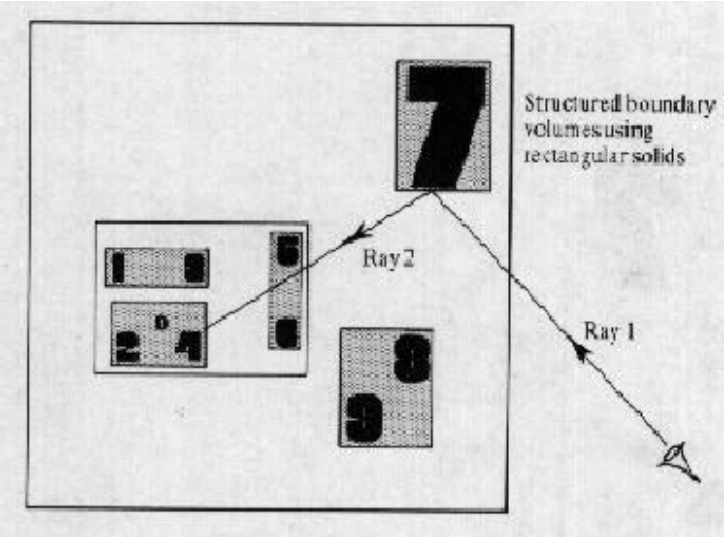
Conclusion sur le lancé de rayon

Ce n'est pas de la physique !!
par ex., on rate des rayons :



(Hill)

Très lent du fait des calculs d'intersection => nombreuses techniques d'optimisation (BSP, etc.)



(Watt & Watt)

+ Problème du niveau ambient