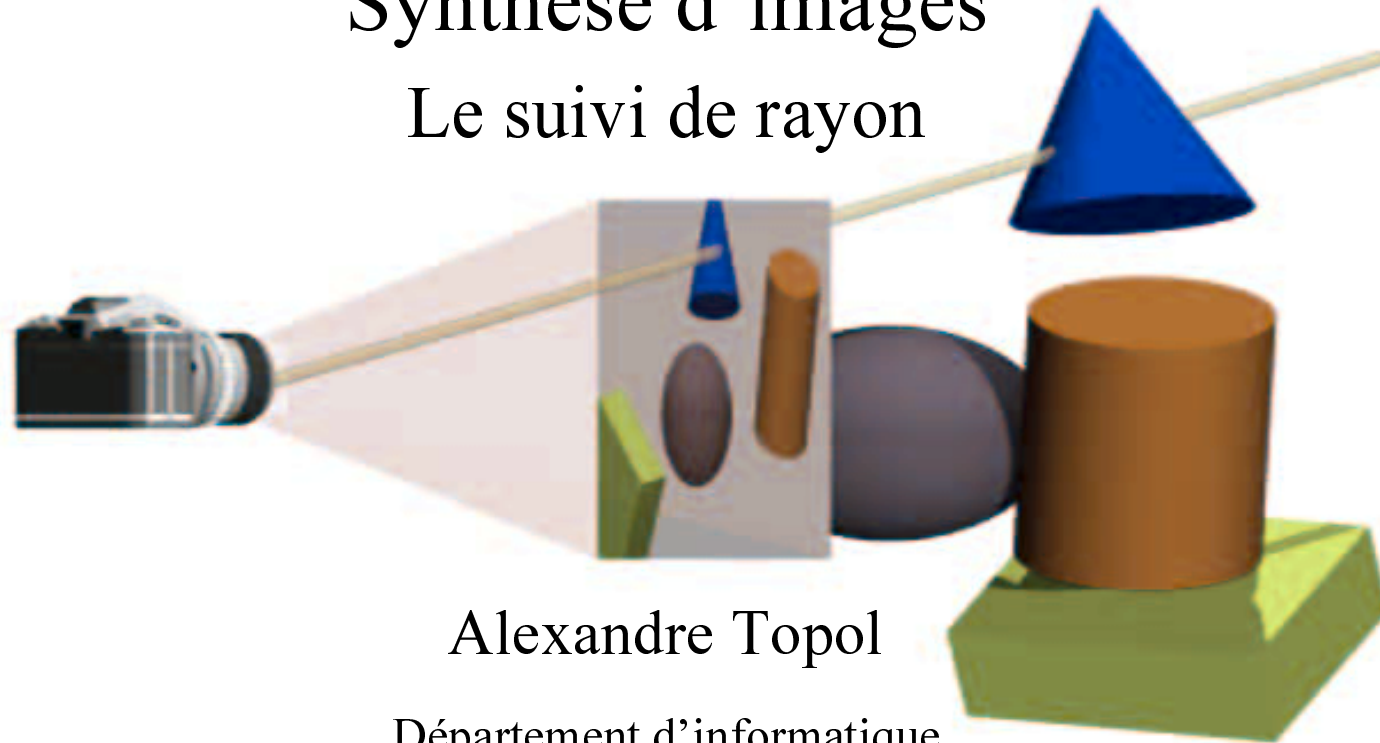


Valeur C et DEA  
*Conception d'applications multimédia*

# Synthèse d'images

## Le suivi de rayon



Alexandre Topol

Département d'informatique  
Conservatoire National des Arts et Métiers

2001-2002

## 9. Le suivi de rayon

---

9.1. Algorithme de base

9.2. Algorithme récursif

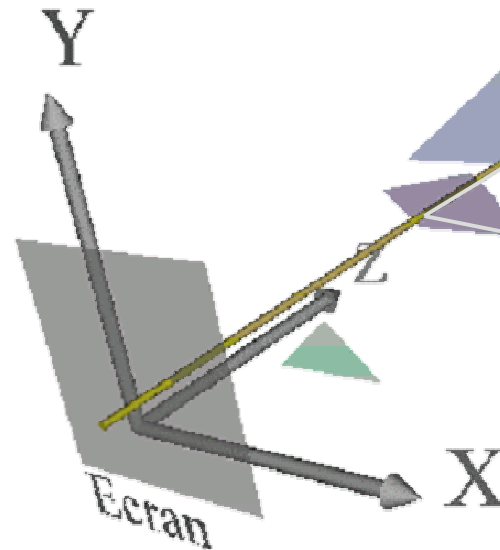
9.3. Optimisations

## 9.1. Algorithme de base

- **Rappel : l'algorithme du Z-buffer**

```

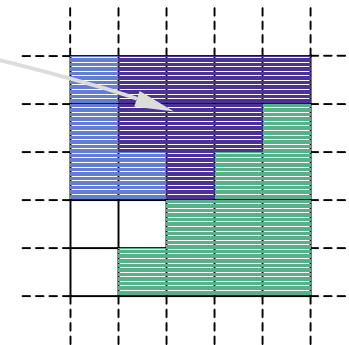
pour chaque face
  pour chaque pixel (i,j) de la face projetée
    pz=valeur de Z de la face en (i,j)
    si Zbuffer[i,j]<=pz alors
      Zbuffer[i,j]=pz
      FrameBuffer[i,j]=couleur pixel
    fin si
  fin pour
fin pour
  
```



**Z-Buffer**

-∞	-4	-4	-4	-4
-7	-7	-4	-4	-1
-∞	-7	-7	-1	-1
-∞	-∞	-1	-1	-1
-∞	-1	-1	-1	-1

**Frame-Buffer**

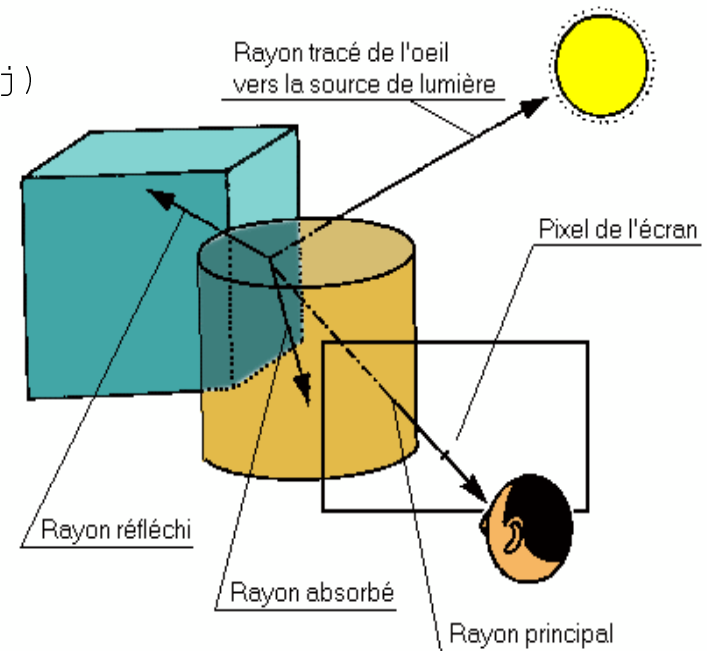


## 9.1. Algorithme de base

- Une autre manière de faire

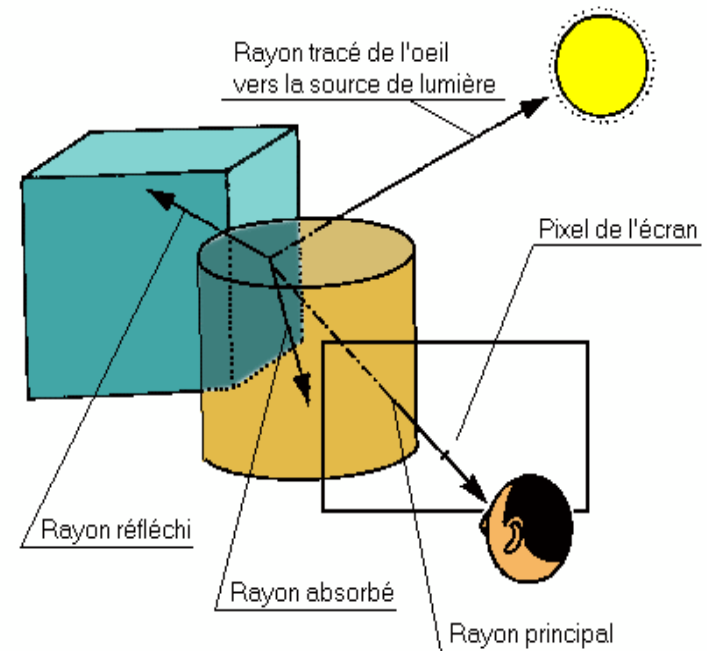
### A. Appel, Spring Joint Computer Conf. 1968

```
pour chaque pixel (i,j) de l'écran faire  
  maxlocal := -∞  
  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
```



## 9.1. Algorithme de base

- Le Z-buffer devient inutile !
- Mais coût *a priori* plus élevé...
- On se ramène à des calculs d'intersection entre un "rayon lumineux" et les volumes de la scène

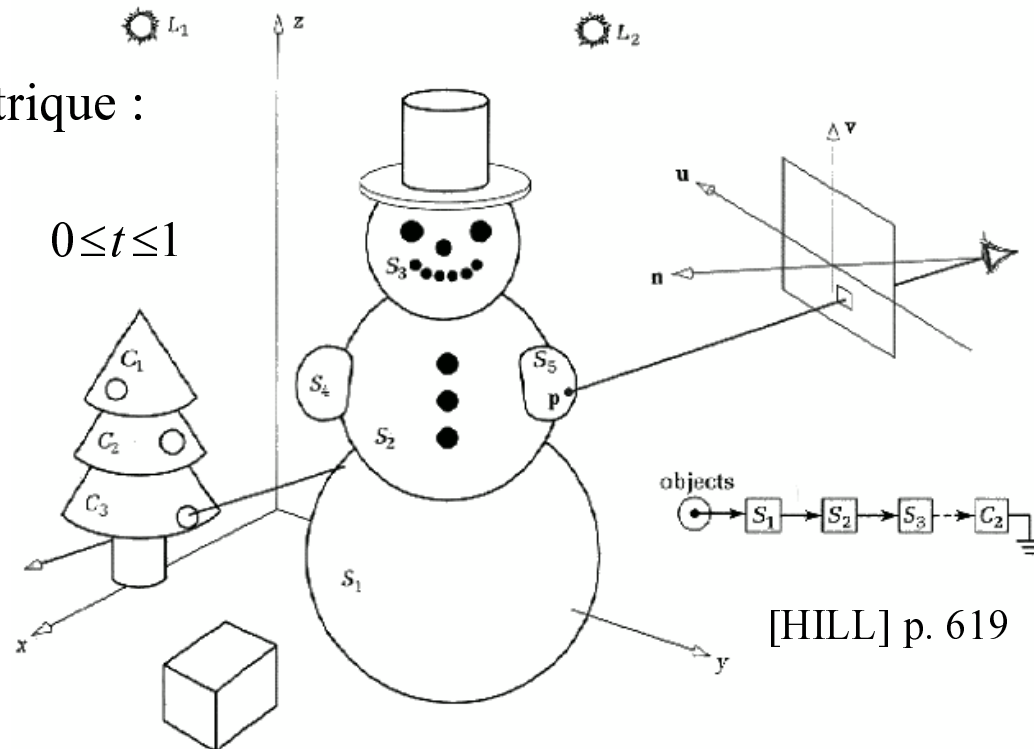


# 9.1. Algorithme de base Calcul d'intersection

- Équation de la droite œil - pixel  
 ( $x_0, y_0, z_0$ ) : coordonnées de l'œil  
 ( $x_1, y_1, z_1$ ) : coordonnées 3D du centre du pixel (dépend du type de projection utilisée)

Représentation paramétrique :

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



## 9.1. Algorithme de base Calcul d'intersection

- Exemple d'intersection avec une sphère

- Équation de la sphère de centre (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 :

$$At^2 + Bt + C = 0 \quad \text{avec}$$
$$\begin{cases} A = \Delta_x^2 + \Delta_y^2 + \Delta_z^2 \\ B = 2(\Delta_x(x_0 - a) + \Delta_y(y_0 - b) + \Delta_z(z_0 - c)) \\ C = (x_0 - a)^2 + (y_0 - b)^2 + (z_0 - c)^2 - R^2 \end{cases}$$

- 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)

## 9.1. Algorithme de base Calcul d'intersection

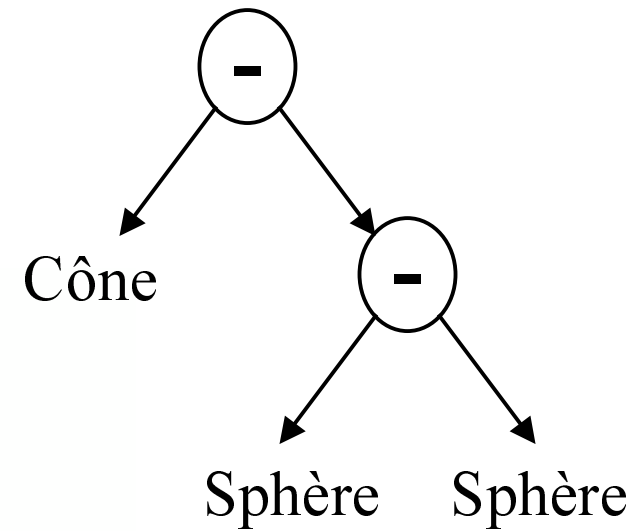
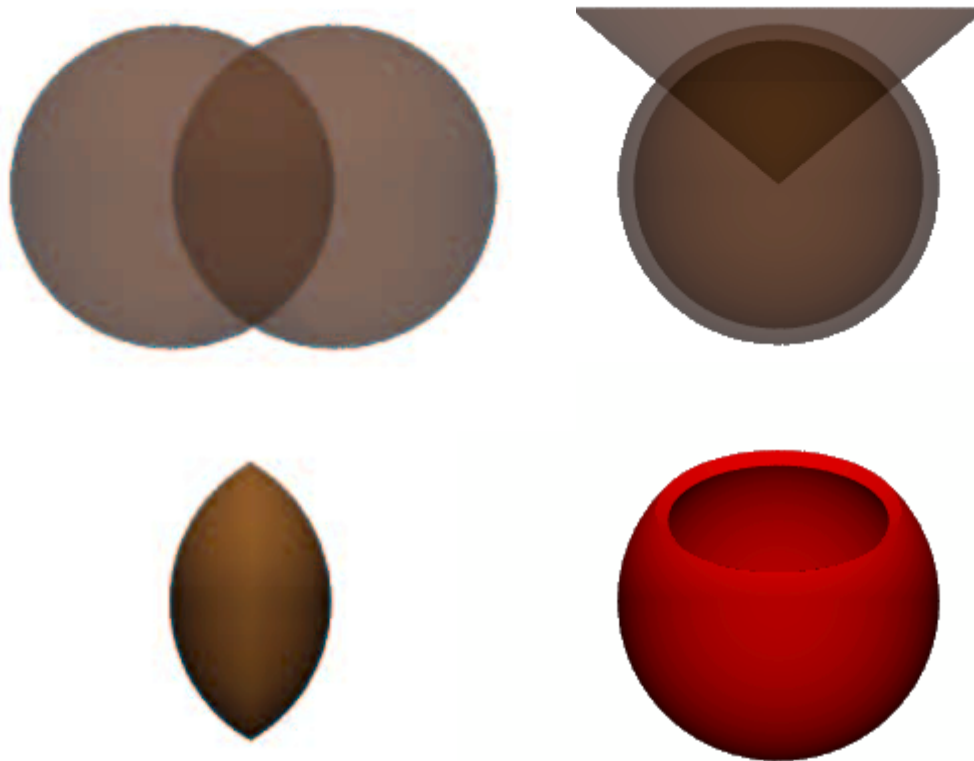
---

- Exemple d'intersection avec un polygone
  - Équation du plan contenant le polygone :
$$x \cdot n_x + y \cdot n_y + z \cdot n_z = cste = d$$
  - Condition d'intersection :
$$N = d - (n_x x_0 + n_y y_0 + n_z z_0)$$
$$D = n_x \Delta_x + n_y \Delta_y + n_z \Delta_z$$
  - Si  $D=0$  et  $N>0$  : rayon inclus dans le plan
  - Si  $D=0$  et  $N<0$  : rayon parallèle au plan
  - Si  $N \neq 0$  Point d'intersection :  $t = N/D$
  - Il faut ensuite tester si le point d'intersection est à l'intérieur du polygone convexe :
    - méthode des angles (coûteux)
    - méthode du balayage : si intérieur : nombre pair d'intersections

## 9.1. Algorithme de base

### Solides définis par opérateurs booléens

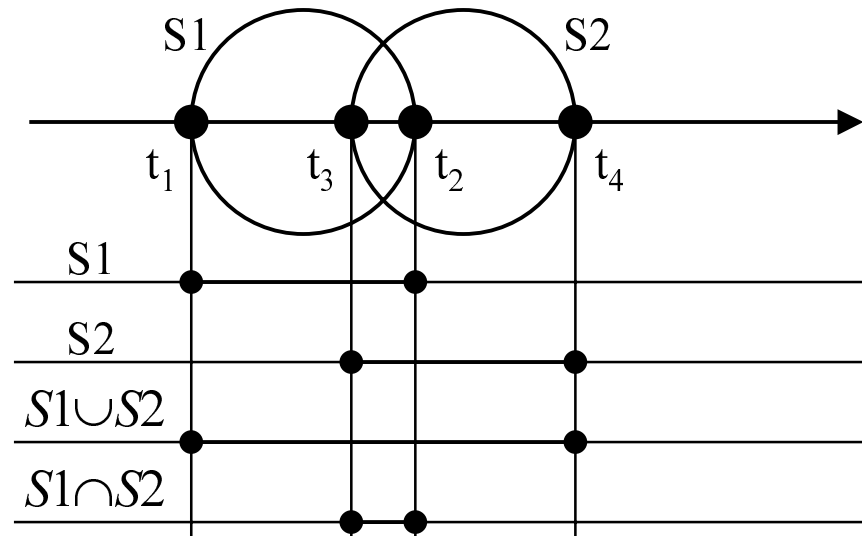
- Rappels chapitre 4 :



## 9.1. Algorithme de base

### Solides définis par opérateurs booléens

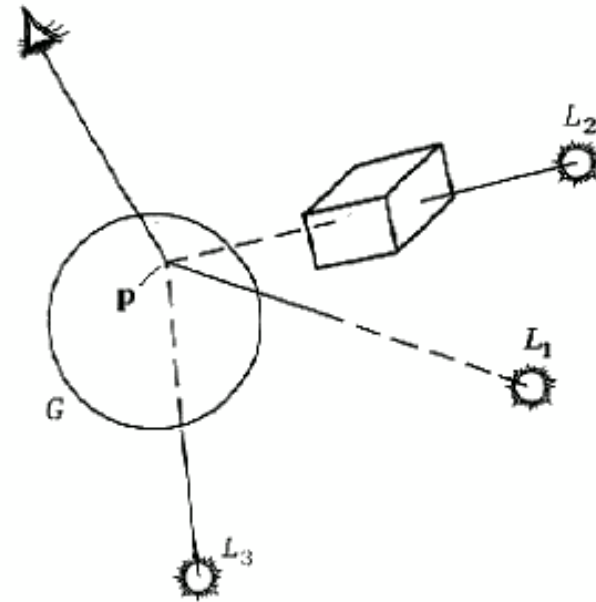
- Algorithme de GoldStein et Nagel (1971) :
  - On se ramène à un problème en 1D :



- Idem opération de différence
- Algorithme (récuratif) de parcourt d'arbre binaire

## 9.1. Algorithme de base Ombre et coloriage

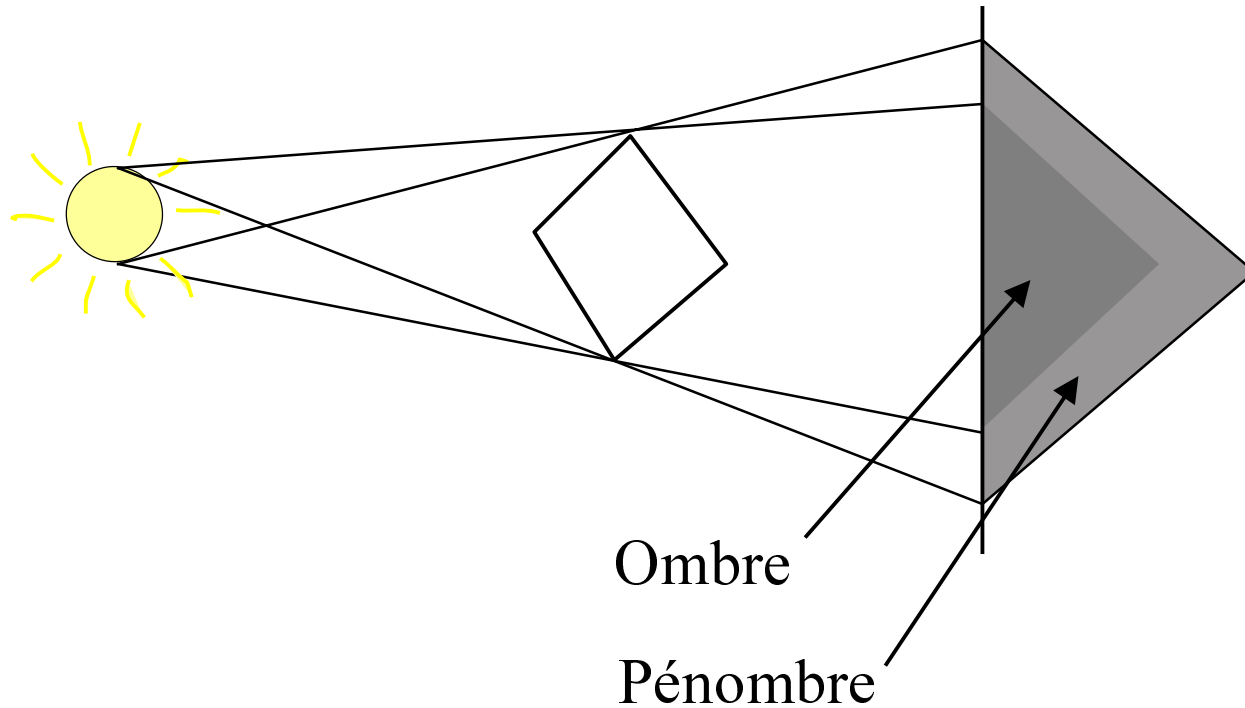
- On "tire" un second rayon du point d'intersection vers la (les) source(s) lumineuse(s) (supposée ponctuelle)
- On applique au point P un des modèles de coloriage vu au chap. 7 (Phong, ...), avec ou sans textures.
- Pour les points dans l'ombre, seule l'intensité de la lumière ambiante est prise en compte.



[HILL] p. 648

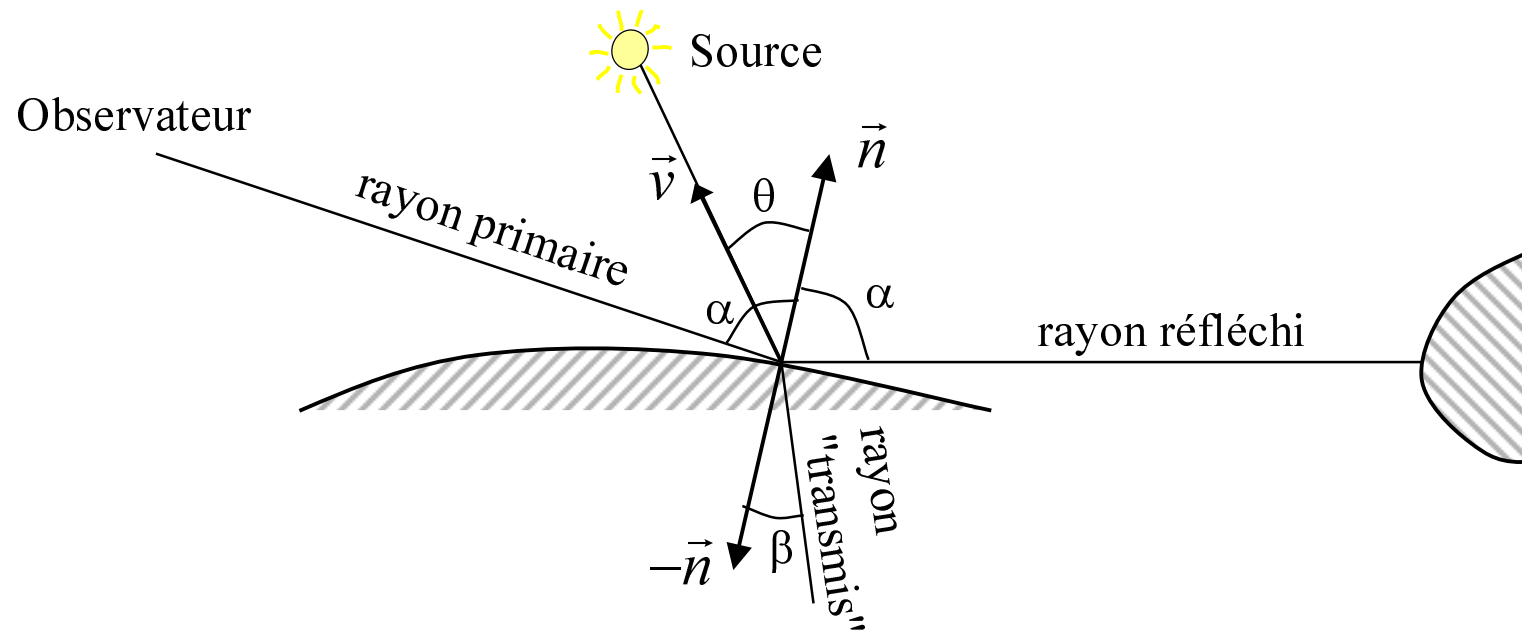
## 9.1. Algorithme de base Ombre et coloriage

- Source non ponctuelle : la pénombre



## 9.2. Algorithme récursif

- Whitted (1980) : prise en compte des autres interactions entre objets de la scène :
  - Réfraction dans les objets transparents
  - Réflexion de la lumière d'un objet à l'autre (miroirs)



## 9.2. Algorithme récursif

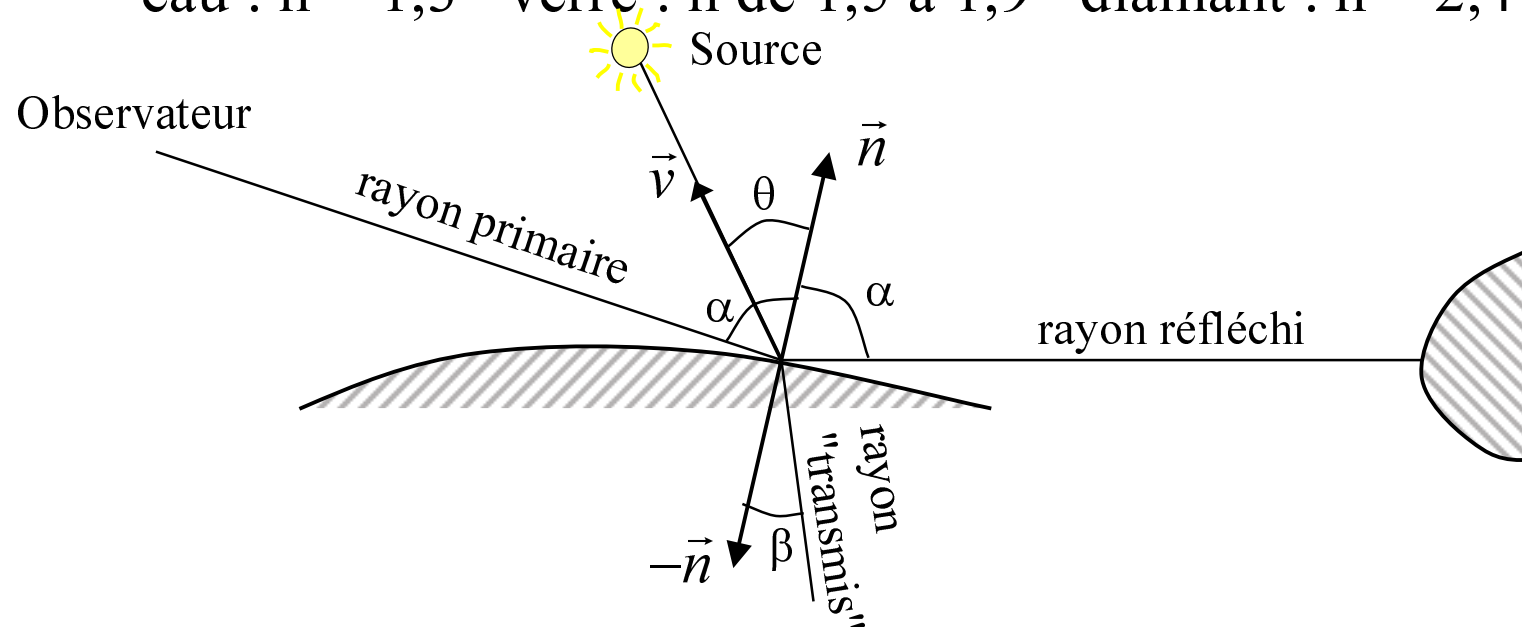
- Les intensités s'ajoutent à celles dues à la source S

$$I_{\text{total}} = I_{\text{amb}} + I_{\text{diff}} + I_{\text{spec}} + I_{\text{refl}} + I_{\text{refr}}$$

- Direction des rayons transmis :

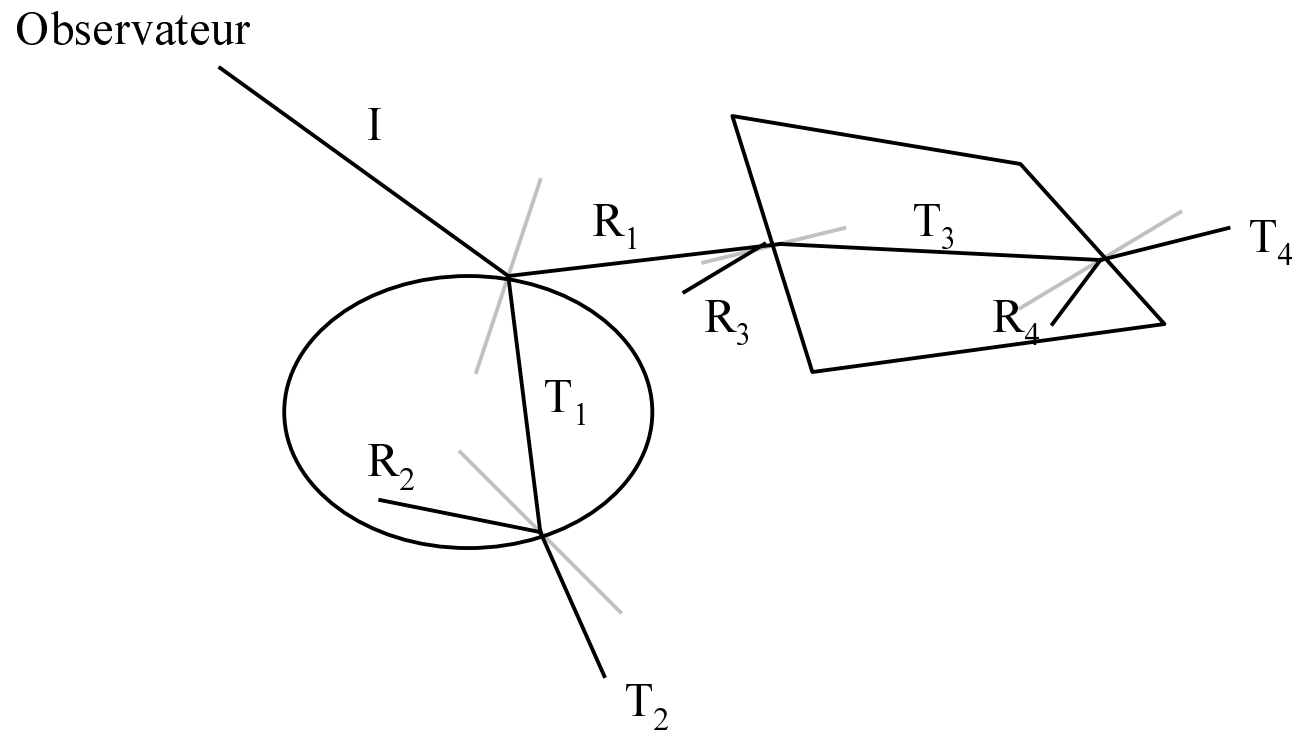
– loi de Descartes :  $\frac{\sin \alpha}{c_{\text{ext}}} = \frac{\sin \beta}{c_{\text{obj}}} \Rightarrow \sin \beta = n_{\text{obj}} \cdot \sin \alpha$

– eau :  $n = 1,3$  verre :  $n$  de 1,5 à 1,9 diamant :  $n = 2,4$



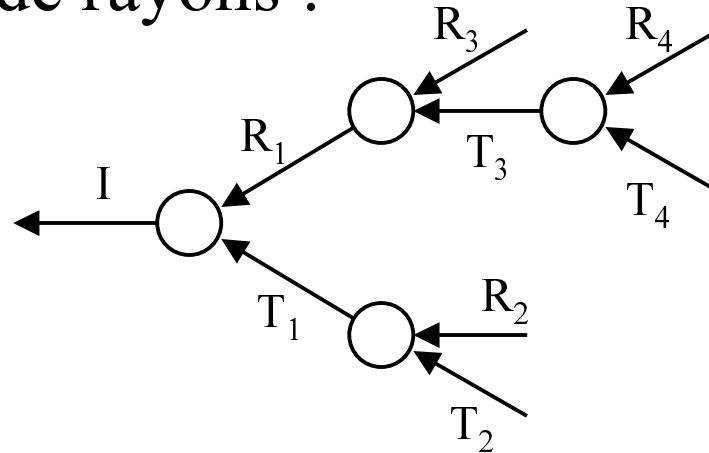
## 9.2. Algorithme récursif

- Arbre de rayons :



## 9.2. Algorithme récursif

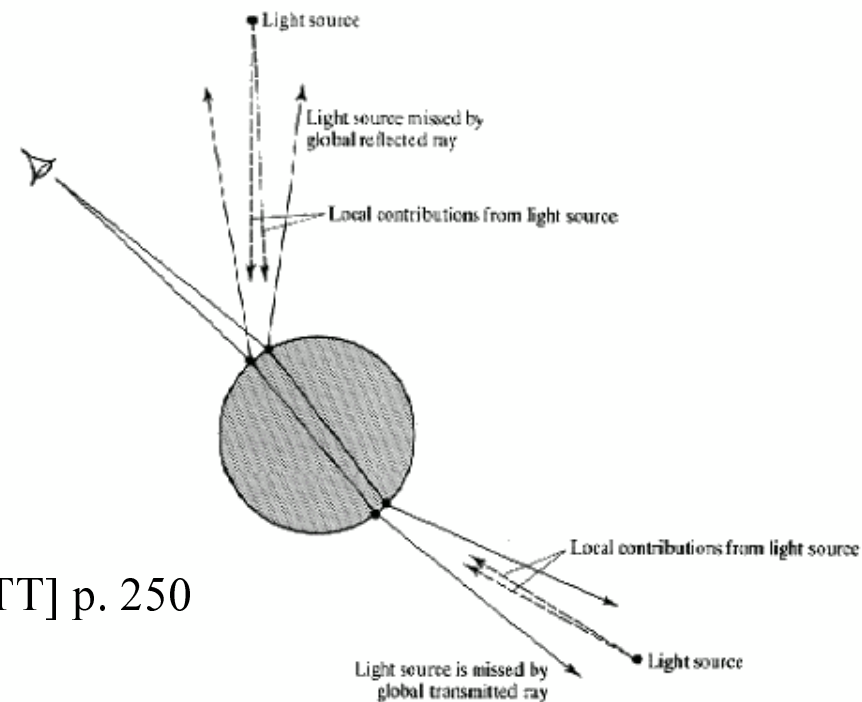
- Arbre de rayons :



- Dans le modèle de Whitted, les intensités R et T sont inversement proportionnelles aux longueurs des rayons.
- On s'arrête à un niveau fixé (2, 3,...)

## 9.2. Algorithme récursif

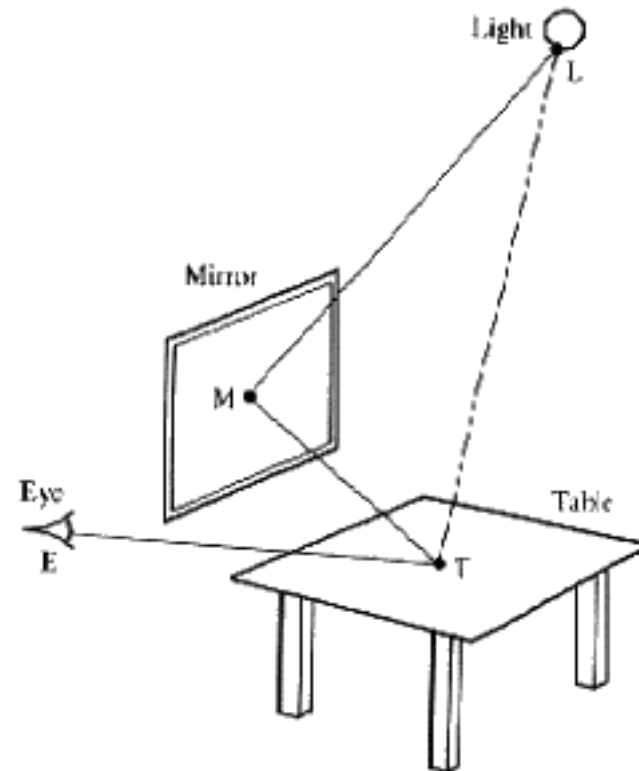
- Limites de l'approche :
  - La lumière reçue directement au point d'intersection n'est prise en compte que par un modèle empirique (Phong, par ex.)



[WATT] p. 250

## 9.2. Algorithme récursif

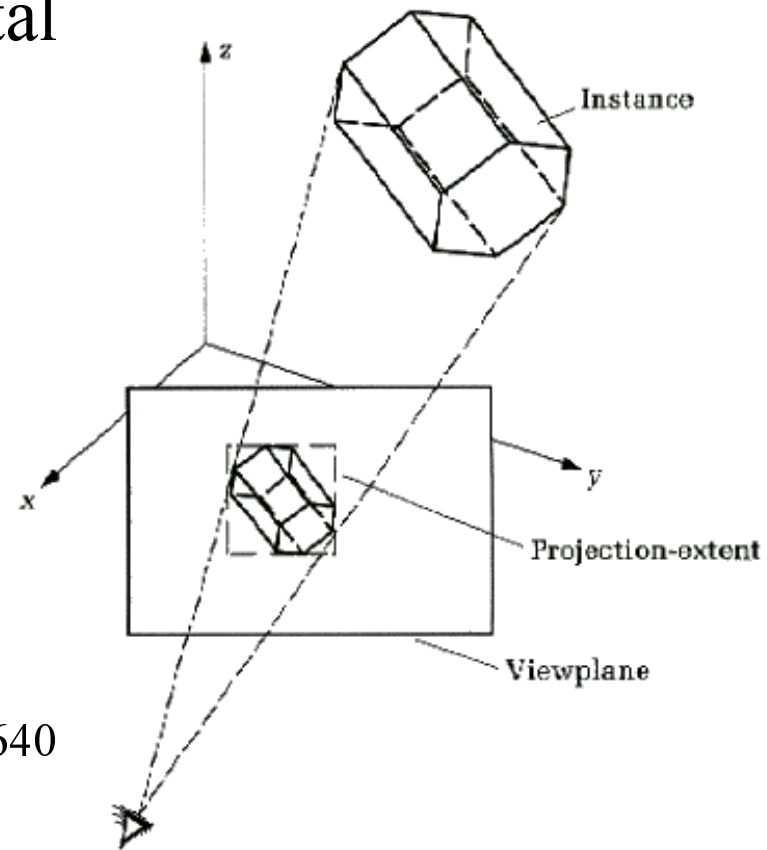
- Limites de l'approche :
  - Des rayons importants sont "ratés" : cas d'un miroir



[WATT] p. 251

## 9.3. Optimisations

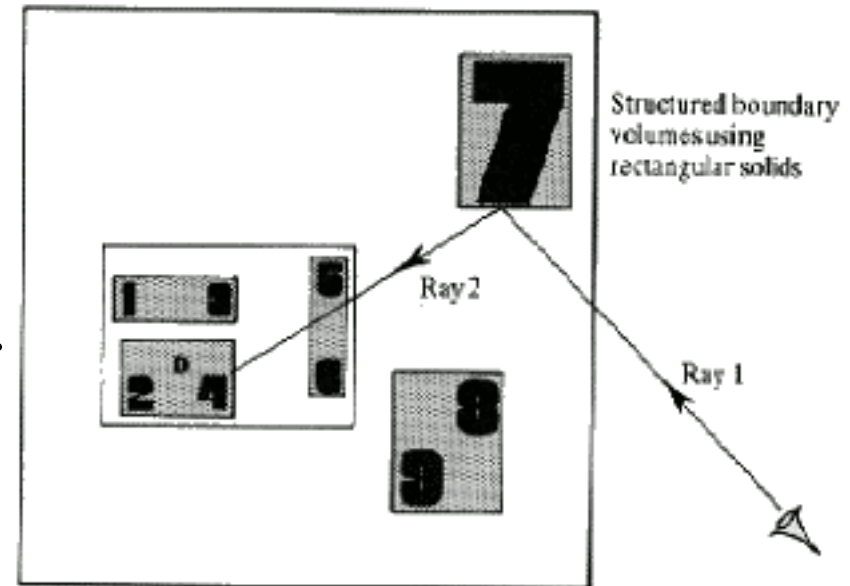
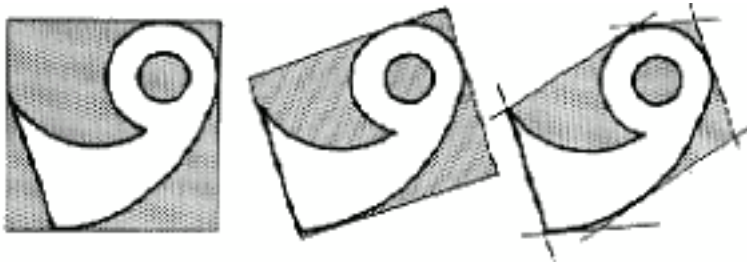
- Problème : calculs d'intersection rayon/objets  
>75% temps de calcul total
- Utilisation des *extents*  
pour le rayon initial



[HILL] p. 640

## 9.3. Optimisations

- Utilisation des volumes englobants
- Plusieurs techniques de volumes englobants : boîtes, sphères, *Slabs*, ...

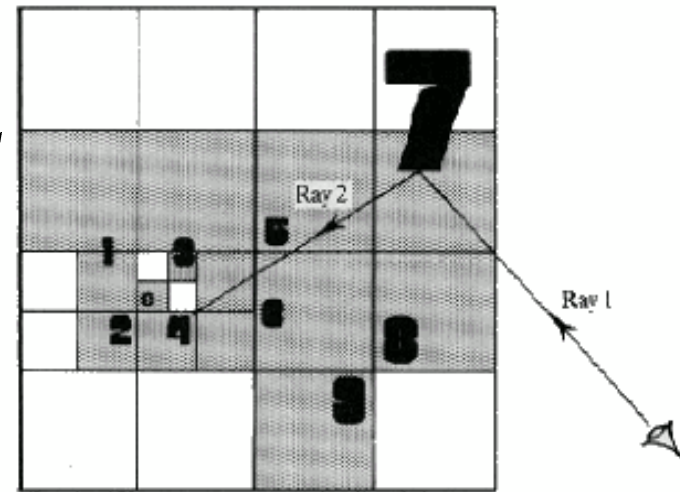


[WATT] p. 236 et 237

## 9.3. Optimisations

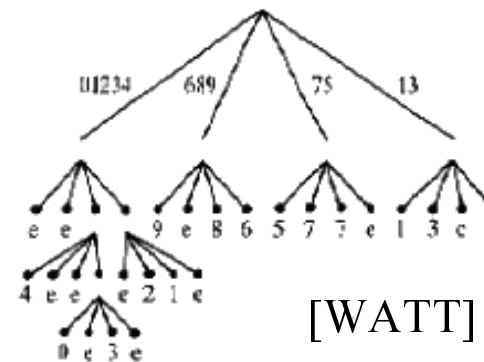
- Partitionnement de l'espace avec la technique des *octrees*

Quand un rayon entre dans une portion de l'arbre, seuls les objets qui en font partie sont concernés par les calculs d'intersection



(a)

- Nombreuses autres techniques (SEADS, BSP) : voir [WATT] chap. 9 pour une comparaison



[WATT] p. 244