

# LA SYNTHÈSE D'IMAGES

- RENDU PHOTORÉALISTE -

Jonathan Fabrizio

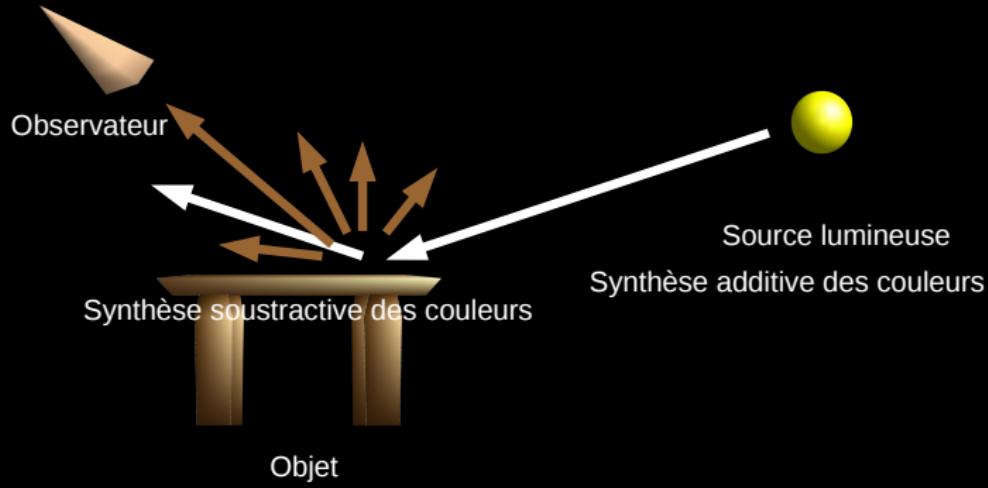
<http://jo.fabrizio.free.fr>

Version : Tue Feb 18 10:00:00 2025

- Objectif :
  - Génération d'images réalistes
- Avec :
  - une contrainte de temps faible...
- Stratégies :
  - *Object-based rendering algorithms*  
Illumination globale calculée indépendamment du point de vue
  - *Image-based rendering algorithms*  
Illumination calculée partiellement, en fonction du point de vue
  - *Deterministic rendering algorithms*
  - *Monte Carlo rendering algorithms*

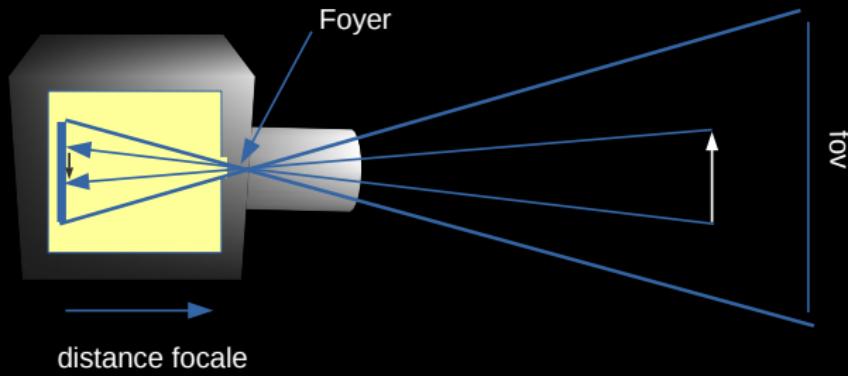
- *Raytracing*
- *Path Tracing et Bidirectional Path Tracing*
- *Point-Based Global Illumination - PBGI*
- ...
- *Radiosity*
- *Photon map*
- ...

# Formation de l'image

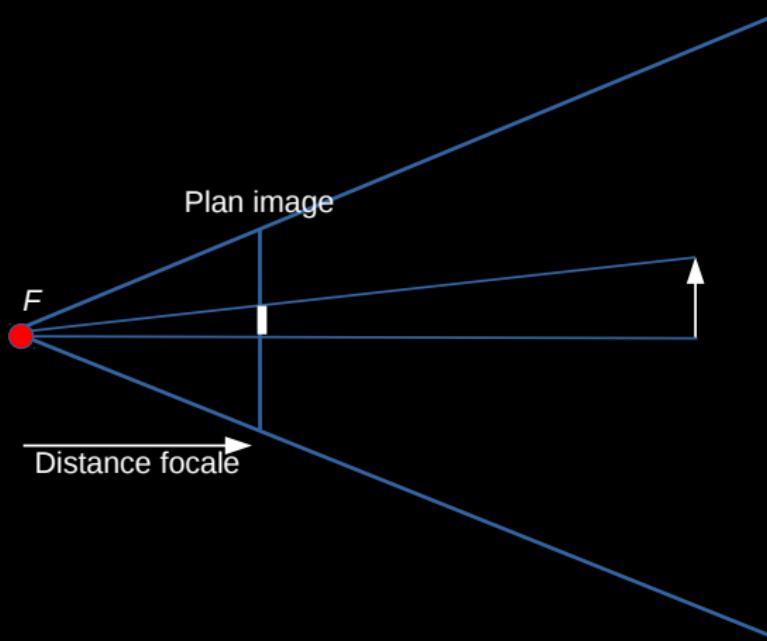


## Formation de l'image

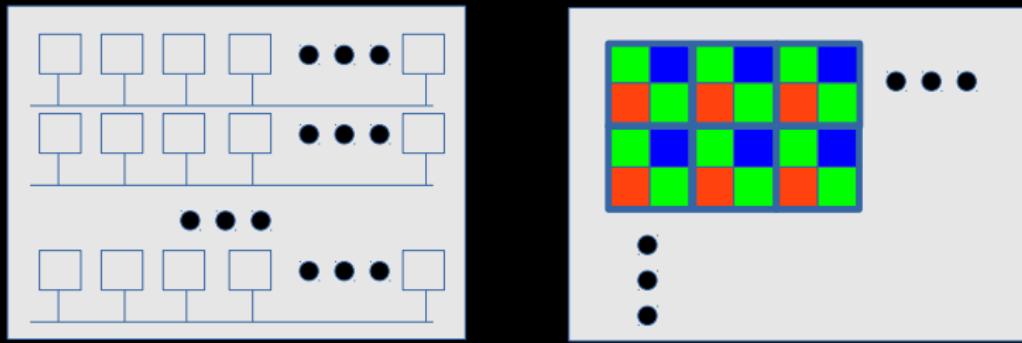
## Capture de l'image



Le modèle sténopé :



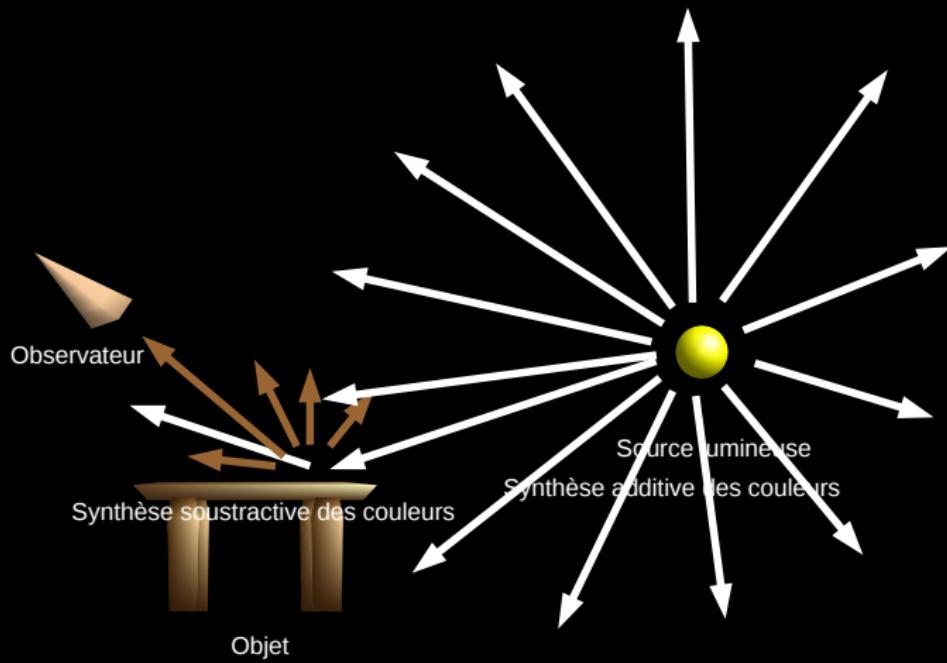
## Capture de l'image (capteur CCD, CMOS)



Dans notre cas nous pouvons faire passer des rayons avec différentes longueurs d'onde par le même point

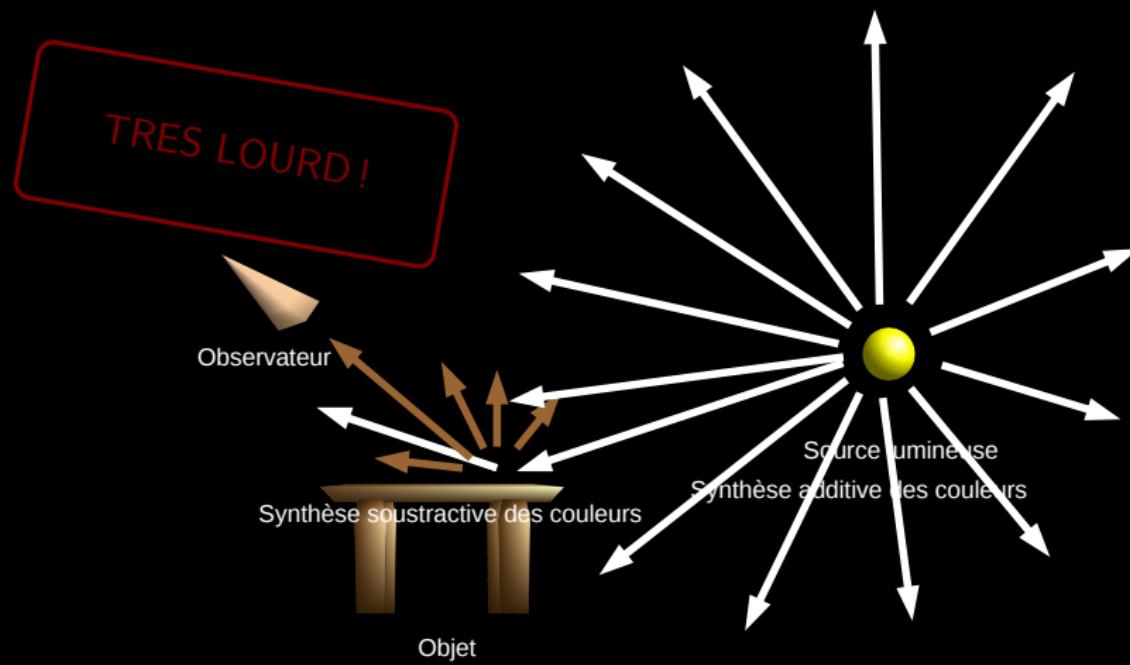
- On peut facilement modéliser la caméra
- Il faut réussir à modéliser l'éclairage
  - Idée : « suivre » les rayons lumineux pour trouver le chemin parcouru depuis la source lumineuse, jusqu'à l'œil.

- Principe : Lancer une “infinité” de rayons depuis la source pour espérer trouver ceux qui frappent l’œil de l’observateur



## Rendu photoréaliste

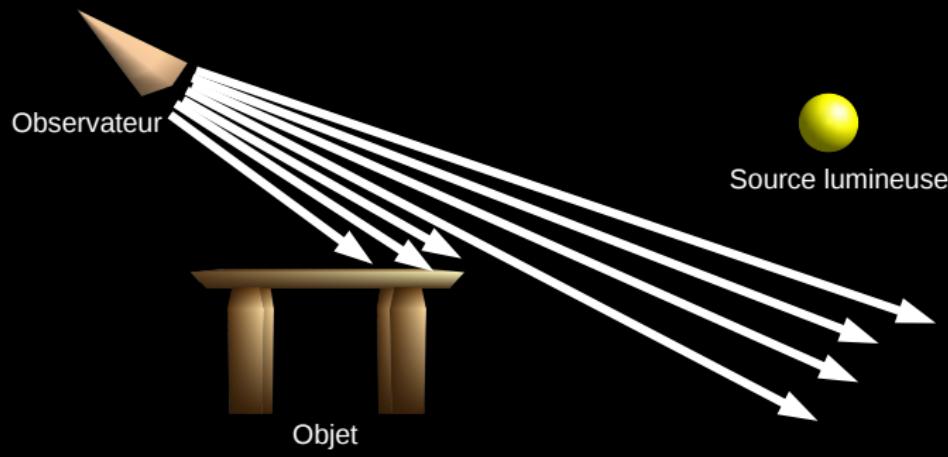
- Principe : Lancer une “infinité” de rayons depuis la source pour espérer trouver ceux qui frappent l’œil de l’observateur



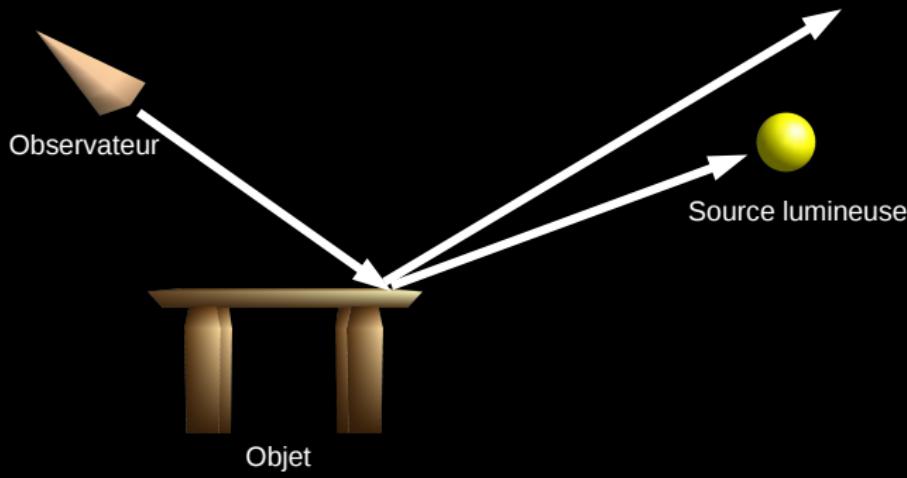


- Historique
  - 68, Appel (*raycasting ??*),
  - 80, Whitted (ajoute les effets optiques : reflexion, transparence...).
- Principe
  - Idée de base : Difficile de suivre tous les rayons partant de la source en revanche il est possible d'estimer le chemin inverse.

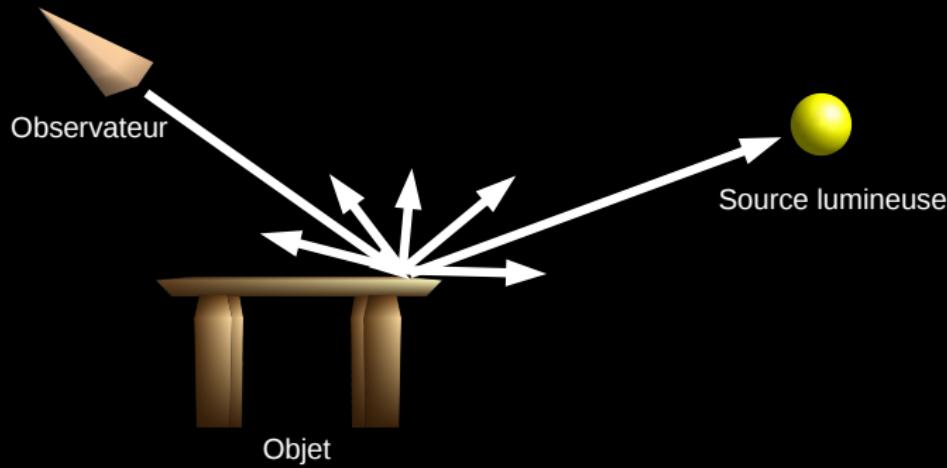
- Principe
  - Faire le chemin inverse pour trouver les objets «vus»



- Principe
  - Pour chaque objet vu, on peut estimer une approximation de l'éclairage local

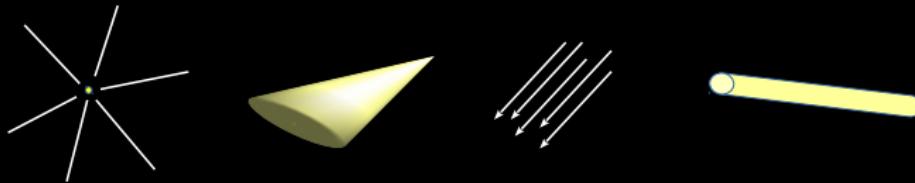


- Principe
  - Approximation de deux types de contributions :
    - la partie diffuse
    - la partie spéculaire



- Calcul de l'illumination locale :
  - Composante diffuse
  - Composante spéculaire
  - Apport des sources primaires
  - Apport des sources secondaires

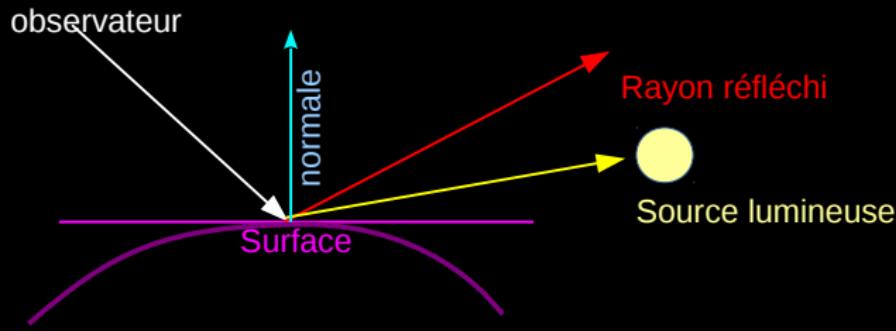
- Calcul de l'illumination locale :
  - Composante diffuse
  - Composante spéculaire
  - Apport des sources primaires
  - Apport des sources secondaires
- Source primaires :
  - Lumières ponctuelles
  - Spots
  - Lumières directionnelles
  - Objets lumineux



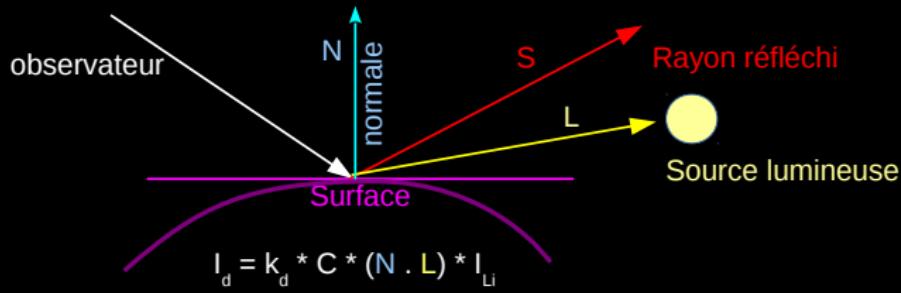
- Calcul de l'illumination locale :
  - Composante diffuse
  - Composante spéculaire
  - Apport des sources primaires
  - Apport des sources secondaires
- Source secondaires :
  - Les autres objets éclairés

- Modèle local

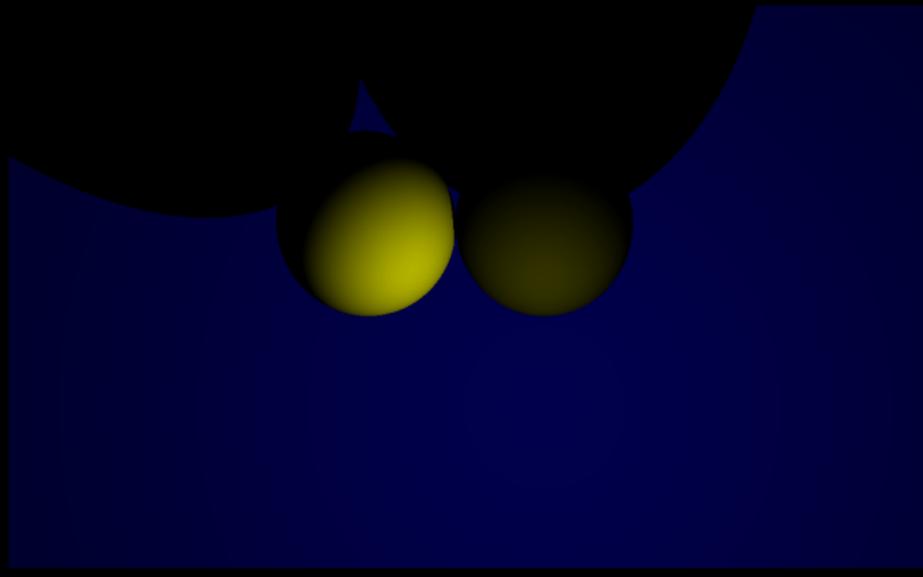
---



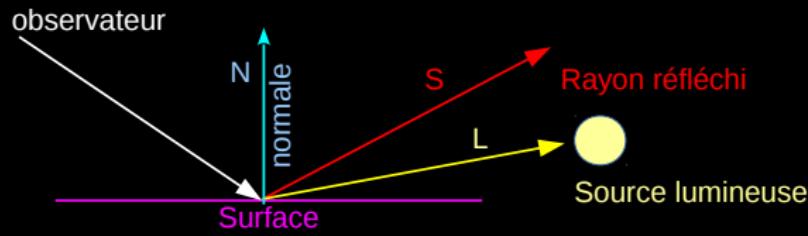
- La composante diffuse
  - La propriété de diffusion de la surface est  $k_d$
  - La couleur de la surface est  $C$



- La composante diffuse

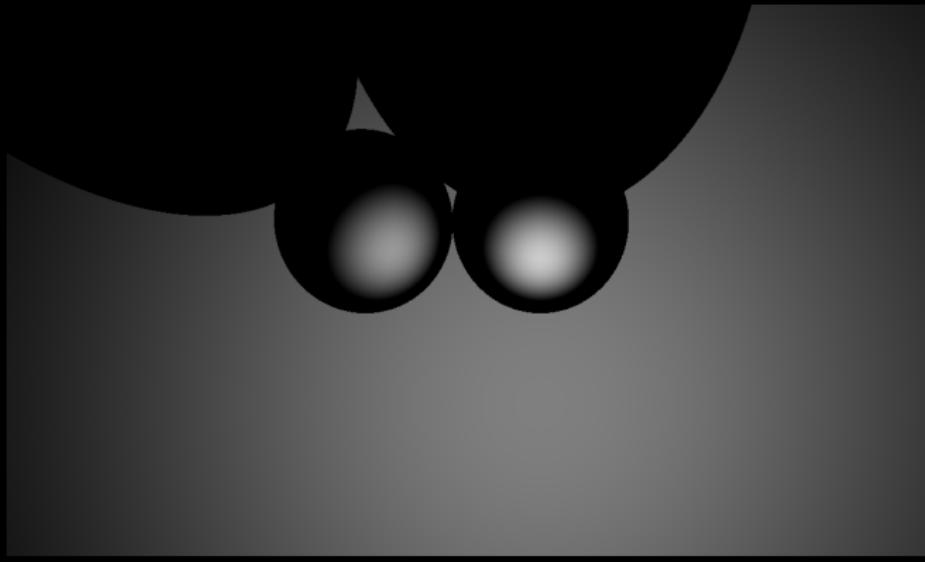


- La composante spéculaire
  - La propriété de réflexion de la surface est  $k_s$

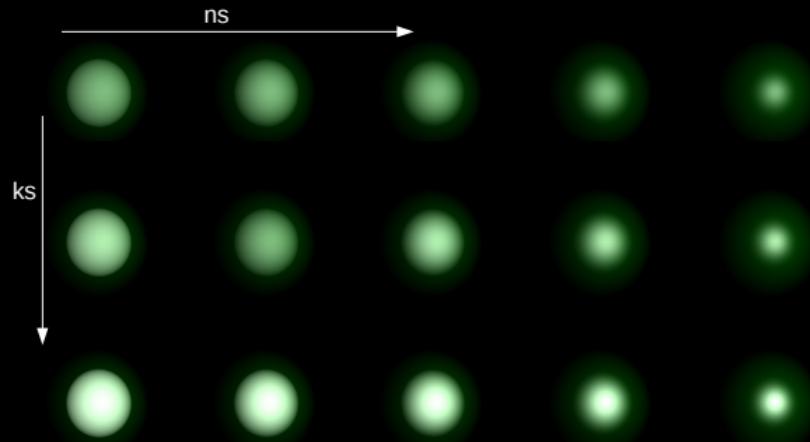


$$I_s = k_s * I_{Li} * (S \cdot L)^{ns}$$

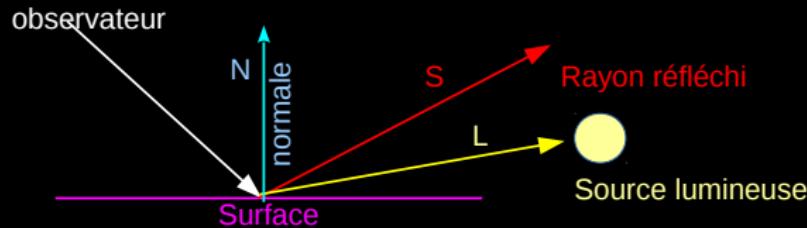
- La composante spéculaire



- La composante spéculaire



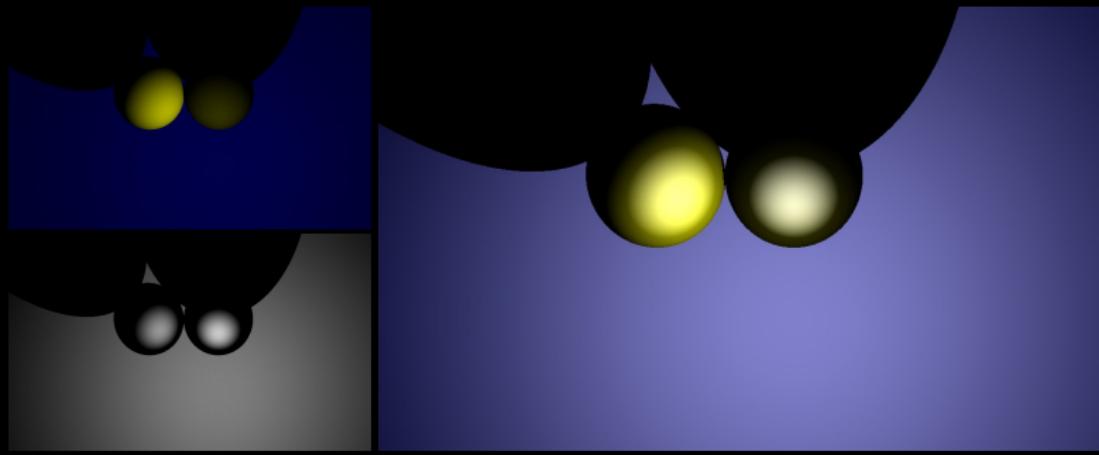
- Calcul de l'illumination locale :
  - Composante diffuse
  - Composante spéculaire
  - Apport des sources primaires
  - Ne tient pas compte des sources secondaires



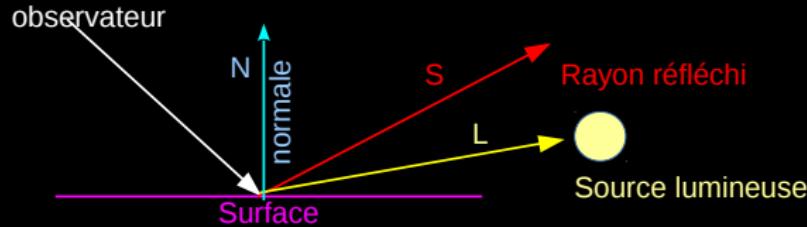
$$I = k_d \cdot (N \cdot L_i) * I_{Li} + k_s \cdot I_{Li} * (S \cdot L_i)^{ns}$$

- Les «  $k_d$  » incluent la couleur
- Il faut sommer toutes les sources lumineuses  $i\dots$

Résultat :



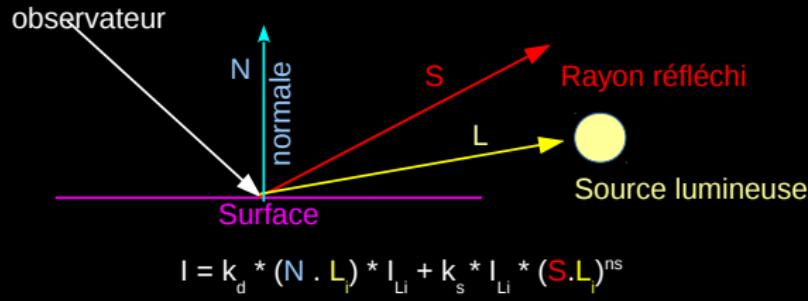
- Calcul de l'illumination locale :
    - Composante diffuse
    - Composante spéculaire
    - Apport des sources primaires
    - Ne tient pas compte des sources secondaires
- 



$$I = k_d \cdot (N \cdot L_i) \cdot I_{Li} + k_s \cdot I_{Li} \cdot (S \cdot L_i)^{ns}$$

- ajout possible d'un coefficient d'atténuation  $f(d)$  ( $d \rightarrow$  distance)
- $f(d) = 1/d$
- $f(d) = 1/d^2$
- $f(d) = 1/(d + k)$
- ...

- Calcul de l'illumination locale :
  - Composante diffuse
  - Composante spéculaire
  - Apport des sources primaires
  - Ne tient pas compte des sources secondaires



Quel modèle de couleur prendre ?

## Étape 1 : Prise en compte des sources primaires Algorithme :

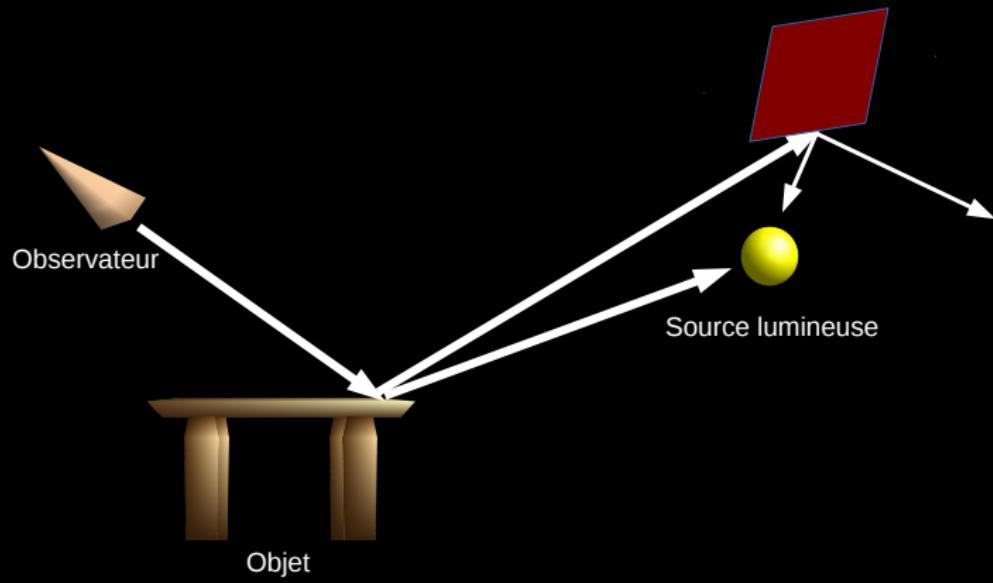
- Pour l'ensemble des points de l'image For all point in the picture :
  - Calculer le vecteur directeur du rayon lumineux  $v$  partant de l'observateur
  - Chercher les intersections de ce rayon lumineux avec l'intégralité des objets de la scène et garder le plus proche
  - Calculer le niveau d'éclairement au point d'intersection en sommant l'apport diffus et spéculaire pour chaque source lumineuse

Étape 1 : Prise en compte des sources primaires

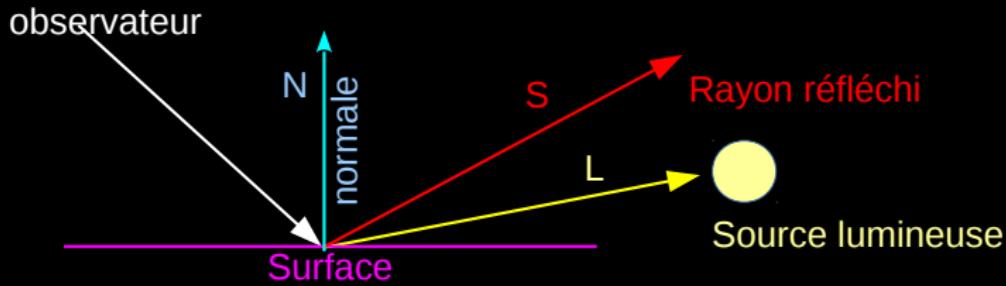
Problèmes :

- Ne tient pas compte des sources lumineuses secondaires
- Ne gère pas les ombres

Prise en compte des sources secondaires :



Prise en compte des sources secondaires :



$$I = k_d * (N \cdot L) * f(d) * I_L + k_s * I_{Li} * (S \cdot L)^{ns} + k_s * I_S$$

## Étape 2 : Prise en compte des sources primaires et certaines sources secondaires

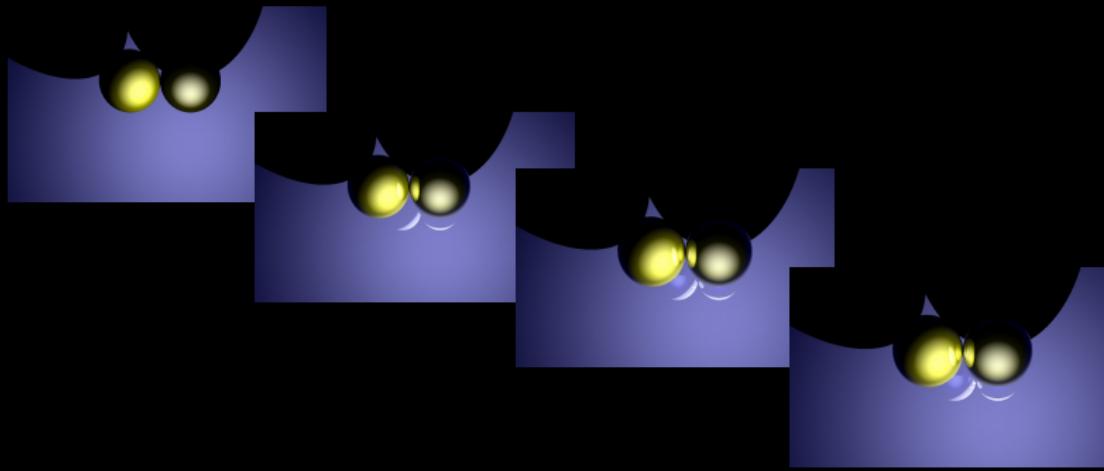
Algorithme :

- Pour l'ensemble des points de l'image :
  - Calculer le vecteur directeur du rayon lumineux  $v$  partant de l'observateur
  - Chercher les intersections de ce rayon lumineux avec l'intégralité des objets de la scène et garder le plus proche
  - Relancer un rayon dans la direction de  $S$  puis calculer le niveau d'éclairement récursivement
  - Calculer le niveau d'éclairement au point d'intersection en sommant l'apport diffus et spéculaire pour chaque source lumineuse ainsi que l'éclairement dans la direction de  $S$

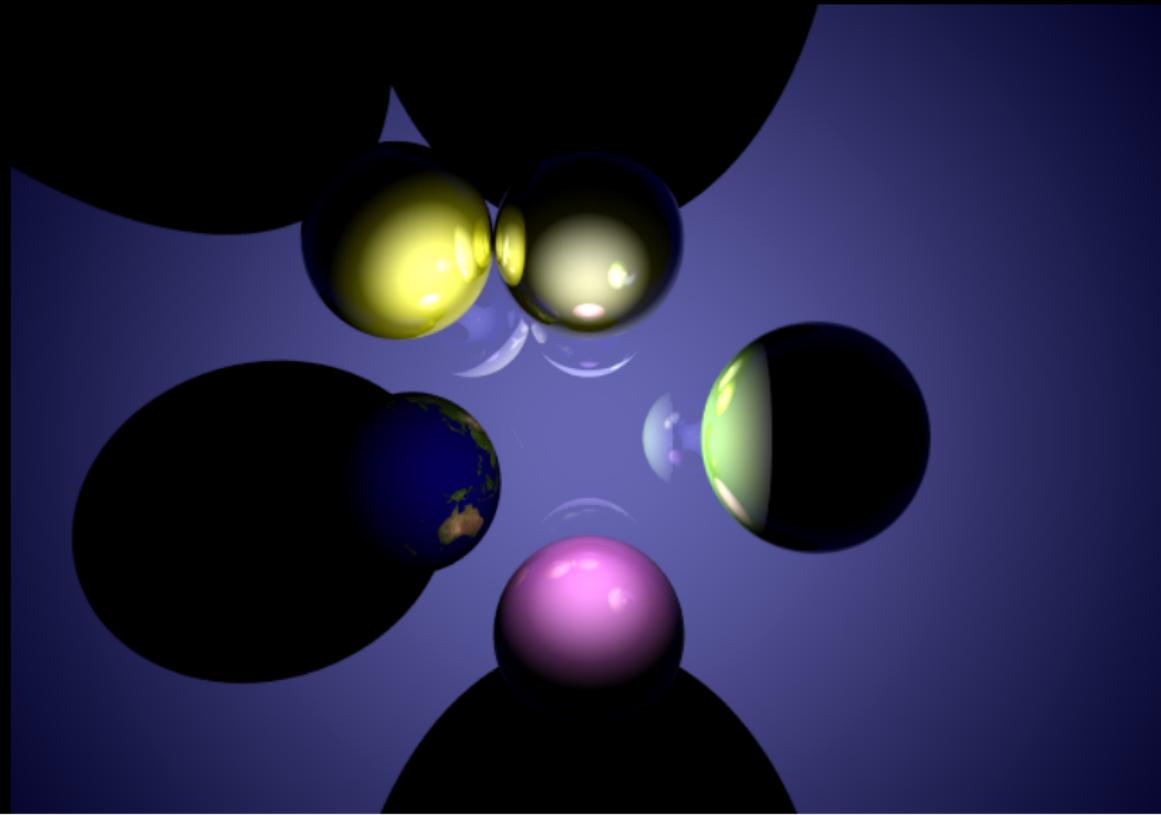
### Étape 3 : Prise en compte de l'ombre

- Pour l'ensemble des rayons que l'on « lance » vers les sources primaires, il faut chercher si un objet de la scène ne s'est pas inséré entre le point considéré et la source. Pour cela, il faut à nouveau calculer l'intersection du rayon avec l'ensemble des objets de la scène et prendre le plus proche.

Résultats :



Résultats :

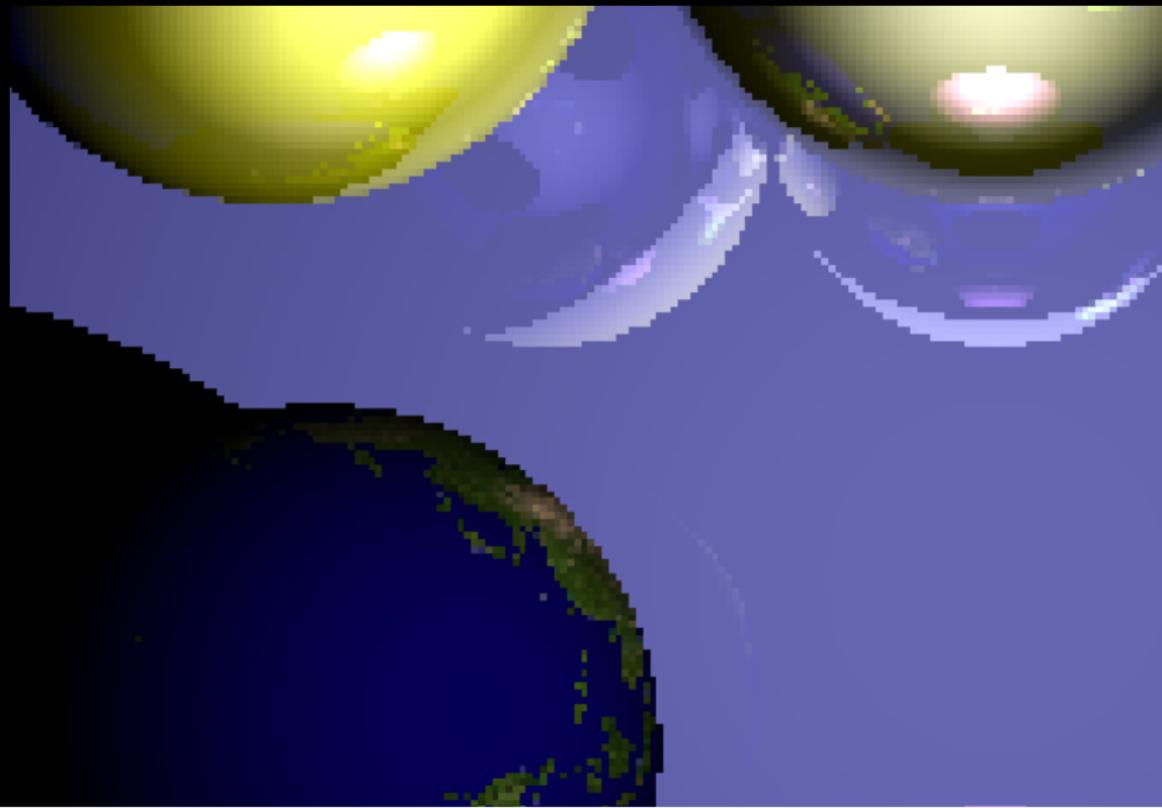


- L'algorithme du *raytracing* est un processus simple, récursif
- Il faut être capable, pour chaque objet, de calculer la normale en chaque point
- Il faut réfléchir à la condition d'arrêt

- Avantages
- Inconvénients

- Avantages
  - Algorithme simple et rapide à mettre en œuvre
  - Génère des images honorables
  - ...
- Inconvénients
  - Temps de calcul un peu élevé
  - Pas gestion de la profondeur de champ et autres effets
  - Mauvaise gestion des ombres (frontières trop brutales)
  - Sources secondaire pas suffisamment prises en compte (éclairage indirect incorrect)
  - Objets transparents
  - « *Aliasing* »
  - ...

## Problème de l'*aliasing*



Problème de l'«aliasing»

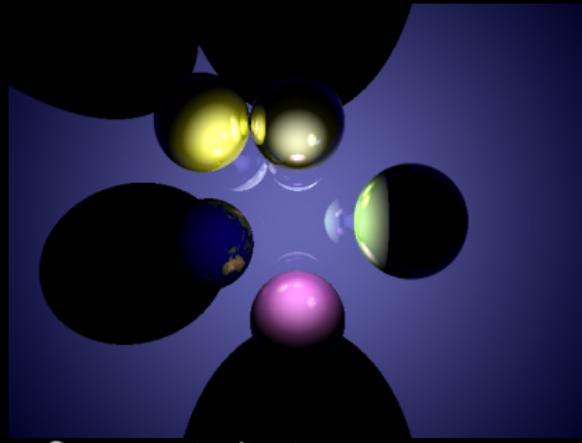
Solutions :

## Problème de l'«aliasing»

Solutions :

- Sur-échantillonnage
  - Lancer plusieurs rayons pour chaque pixel
    - De manière organisée
    - Au hasard
  - Lancer plusieurs rayons pour chaque pixel où le gradient est élevé
    - Bon résultats mais peut être très lent
- Post-filtrage
  - Résultat moyen mais très rapide

## Résultats

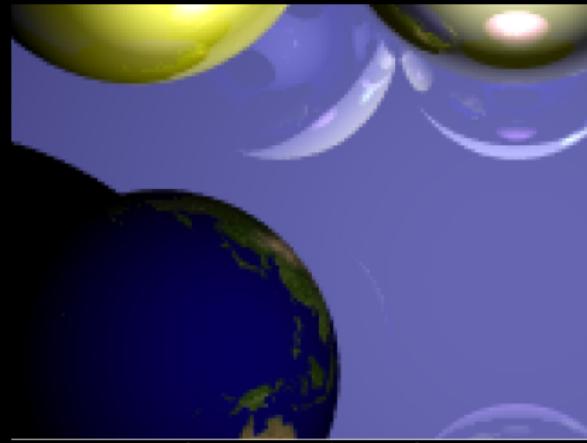
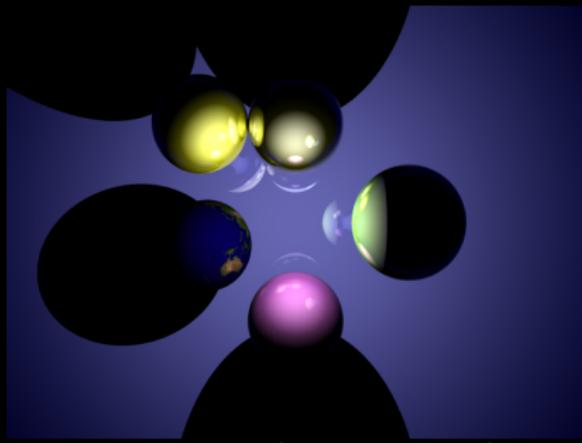


Sans anti-aliasing

Temps : de l'ordre de la seconde

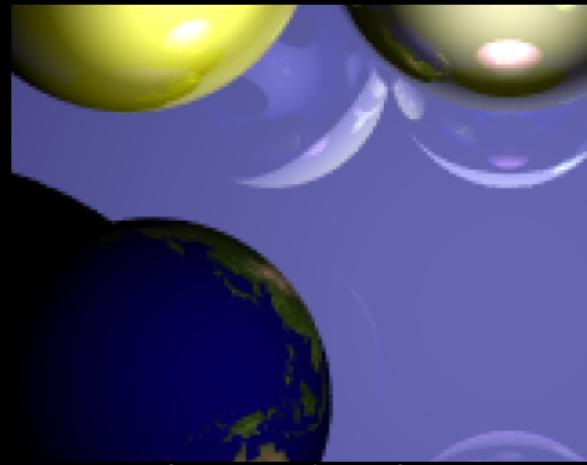
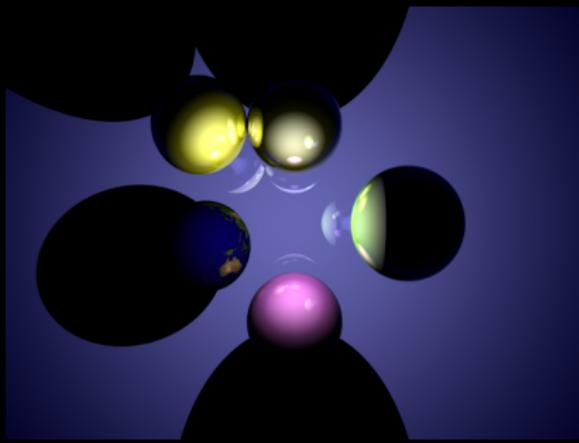


## Résultats



Anti-aliasing (Upsampling) sur toute l'image (50 rays/pixel)  
Temps : de l'ordre de 7-8 secondes

## Résultats



Anti-aliasing sur les zones de gradient élevé (50 rays/pixel)  
Temps : de l'ordre de la seconde

Problème du temps de calcul

Solutions :

## Problème du temps de calcul

Solutions :

- Volumes englobants
- Projection sur un plan/partition de l'espace
- Pre-trier les objets ?
- Calcul parallèle
- Utilisation d'OpenGL
- ...

Problème des objets transparents  
Solution :

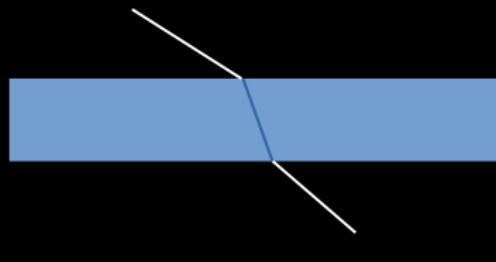
## Problème des objets transparents

Solution :

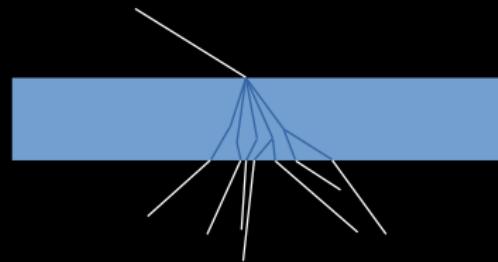
- Comme nous avons relancé le rayon réfléchi, il faut « suivre » le rayon réfracté
  - Loi de la réfraction

## Problème des objets transparents

Solution :  
Milieux transparents



## Objets translucides



- Loi de la réfraction (Snell Decartes) :  
 $n_1 \sin i_1 = n_2 \sin i_2$
- Distribution probabiliste

## Problème des objets transparents

- Tenir compte du rayon réfracté pour l'illumination locale :

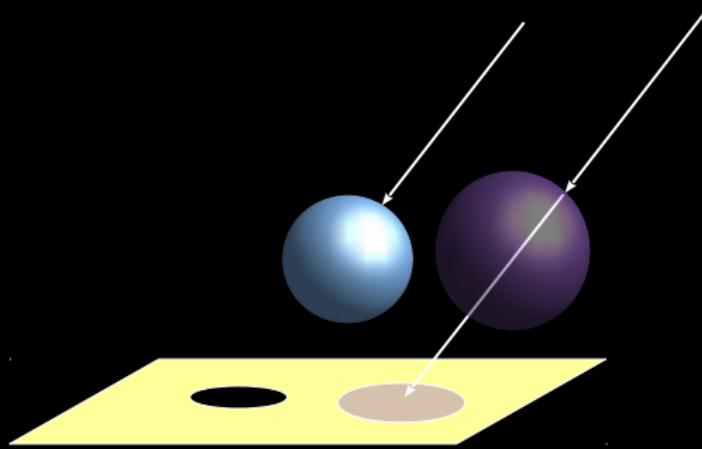
$$I = I_d + I_S + I_t + k_t T$$

## Problème des objets transparents

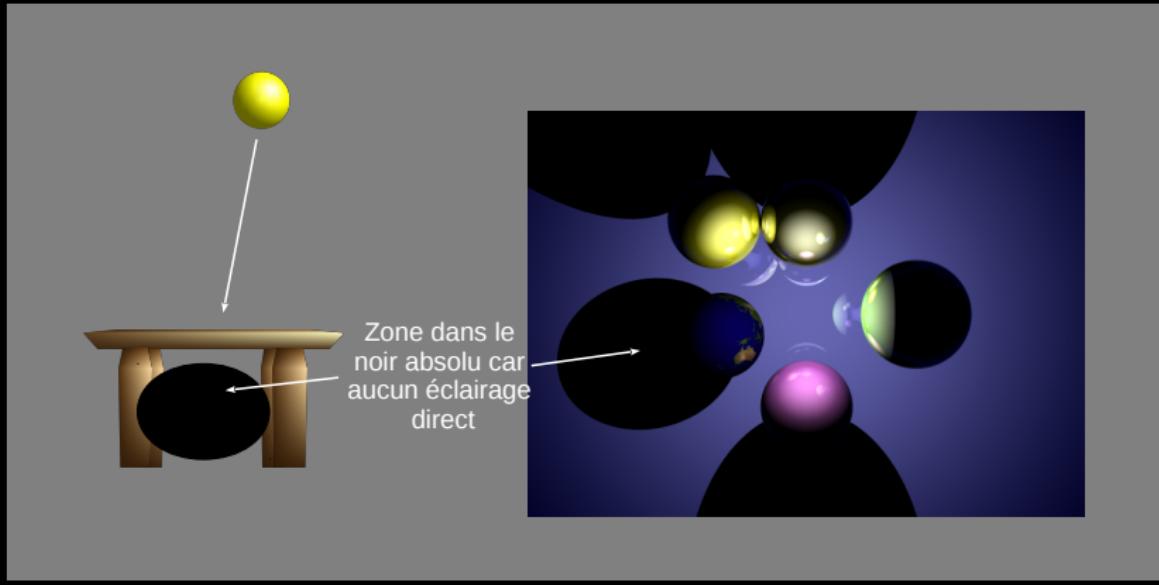
- Calcul de l'ombre des objets transparents...
  - Solution approchée :

## Problème des objets transparents

- Calcul de l'ombre des objets transparents...
  - Solution approchée :
    - Ne pas dévier le rayon mais filtrer les longueurs d'ondes



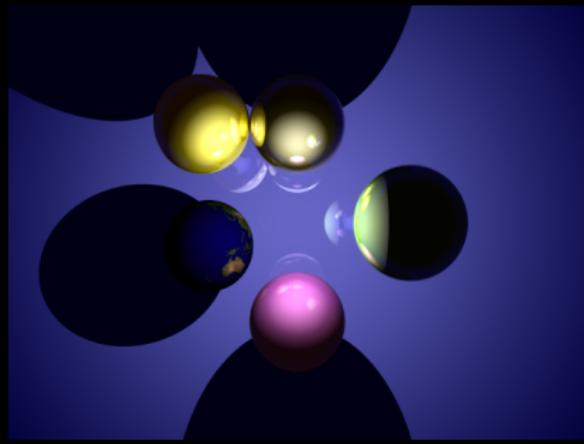
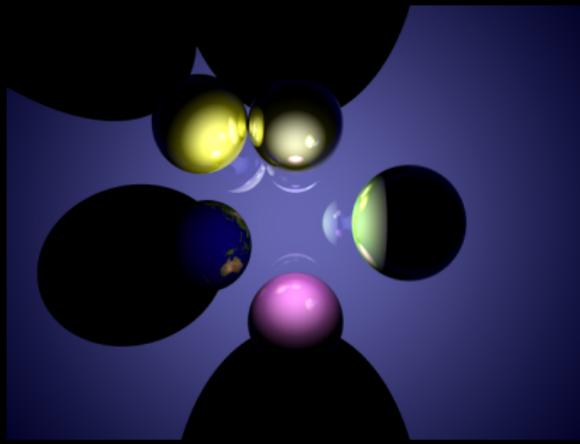
## Problème de l'éclairage indirect



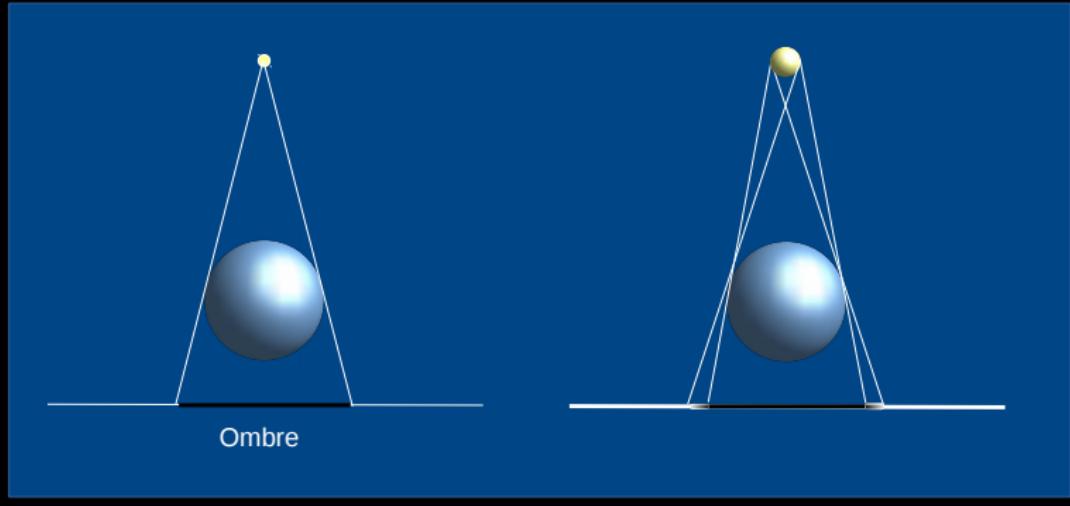
- Problème de l'éclairage indirect
- Solution :

- Problème de l'éclairage indirect
  - Solution :
    - Ajouter une lumière ambiante
- $$I = k_a * I_a + I_d + I_s + I_r + I_t$$
- Solution vraiment approximative

## Résultats



## Problème de l'ombre



Problème de l'ombre  
Solution :

## Problème de l'ombre

Solution :

- Ne plus considérer une lumière comme ponctuelle
  - Problème du temps de calcul

## Bilan :

- Avantages :
  - Algorithme très simple
  - Donne des images honorables
- Des problèmes majeurs persistent
  - Les sources secondaires ne sont pas suffisamment bien gérées
  - Les objets transparents non plus

## Amélioration :

- Raytracing distribué (84)
  - Sur-échantillonnage pour simuler
    - les ombres douces
    - la profondeur de champ
    - ...
  - Ne règle pas le problème de l'apport de la diffusion des sources secondaires
  - Quantité de calcul énorme

## Conclusion

- Algorithme simple
- Nécessite beaucoup d'améliorations pour avoir des images photoréalistes.
- Déjà implémenté dans les GPU :
  - Linear Swept Spheres
  - BVH
  - ...
- Améliorations liés à l'utilisations de réseaux de neurones :
  - débruitage
  - Sur la carte vidéo : Tensor cores (Neural Texture Compression, Neural Radiance Cache), DLSS (AI optical flow, Super resolution...)



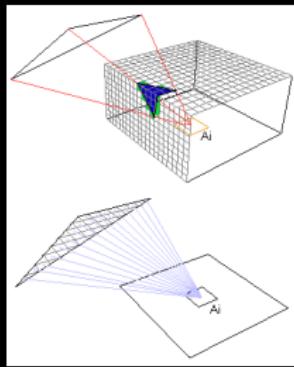
### La Radiosité

- On essaye d'estimer la « radiosité » de chaque élément de la scène, c'est à dire la quantité d'énergie que chaque élément émet...

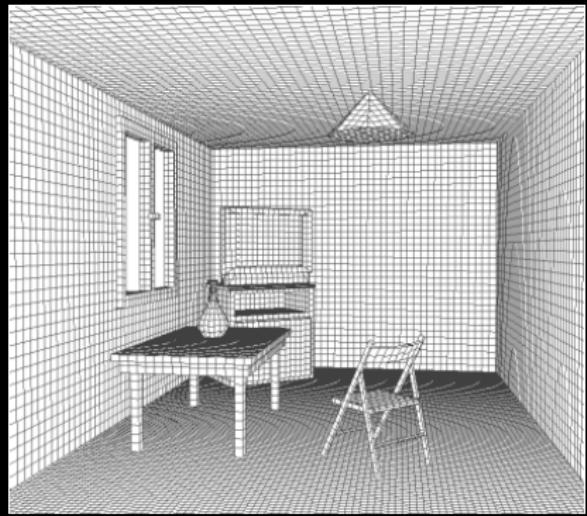
- $B_i$  la radiosité de la surface  $i$
- $E_i$  la quantité de lumière émise par la surface  $i$
- $P_i$  la fraction de lumière incidente qui est réfléchie par la surface  $i$
- $F_{ji}$  la fraction de lumière quittant la surface  $i$  et atteignant la surface  $j$

$$B_i = E_i + P_i \sum_j (F_{ji} B_j)$$

Calcul des  $F_{ij}$  par hemi-cubes



source : wikipedia



Ne permet pas directement de calculer une vue de la scène mais simplement l'illumination globale

- Avantages :
  - Prend mieux en compte les sources secondaires
  - Calculée une fois pour toutes
- Inconvénients
  - Tient compte que de la diffusion
  - Assez lourd
  - Obligation d'avoir un maillage (il faut discréteriser les surfaces)
  - Objets transparents ?

# *Photon Map*

- Photon Map
  - Pré calcul de l'illumination de la scène.
  - Lancement de rayons lumineux depuis les sources et calcul des accumulations des photons.
- Avantages
  - Permet de modéliser plus proprement les sources secondaires, les ombres portées (...) et surtout les objets transparents (caustiques).
- Inconvénients
  - Calculs
  - Complexité

Résultats :...

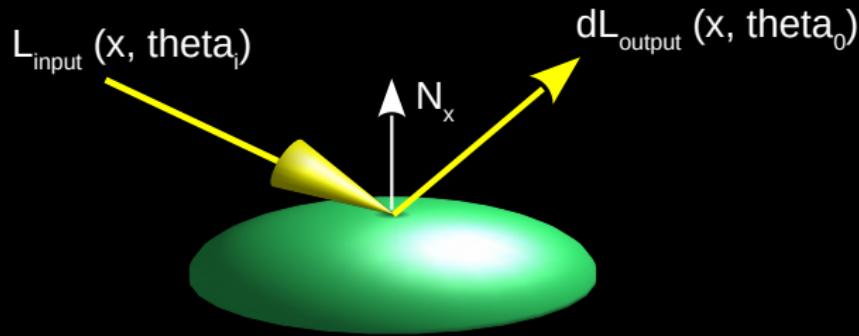
Améliorations :

- Projection Maps
- Visual importance map (3-pass Technique)
- Shadow photons
- ...

## *Path Tracing / Bidirectional Path Tracing*

- Modélisation des propriétés de réflexion des surfaces :  
(Bidirectional reflectance distribution function – BRDF) (idem pour la transmission)
- Solution pour résoudre l'illumination

*BRDF : Bidirectional reflectance distribution function (Réflectivité bidirectionnelle)*



$$f_r(x, \theta_i, \theta_o) = \frac{\partial L_{output}(x, \theta_o)}{L_{input}(x, \theta_i) |\theta_i \cdot N_x| \partial w_i}$$

BRDF :

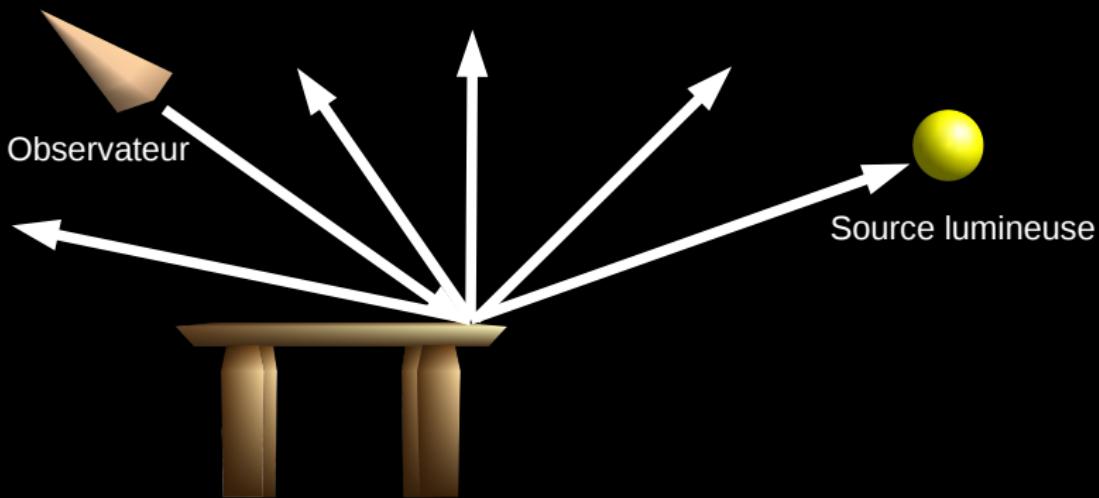
- Conservative  
 $\int f_r(x, \theta_i, \theta_o) L_{input}(x, \theta_i) |\theta_i \cdot N_x| \partial w_o \leq 1$
- Réciprocité de Helmholtz  
 $f_r(x, \theta_i, \theta_o) = f_r(x, \theta_o^{-1}, \theta_i^{-1})$
- Positivité

BRDF :

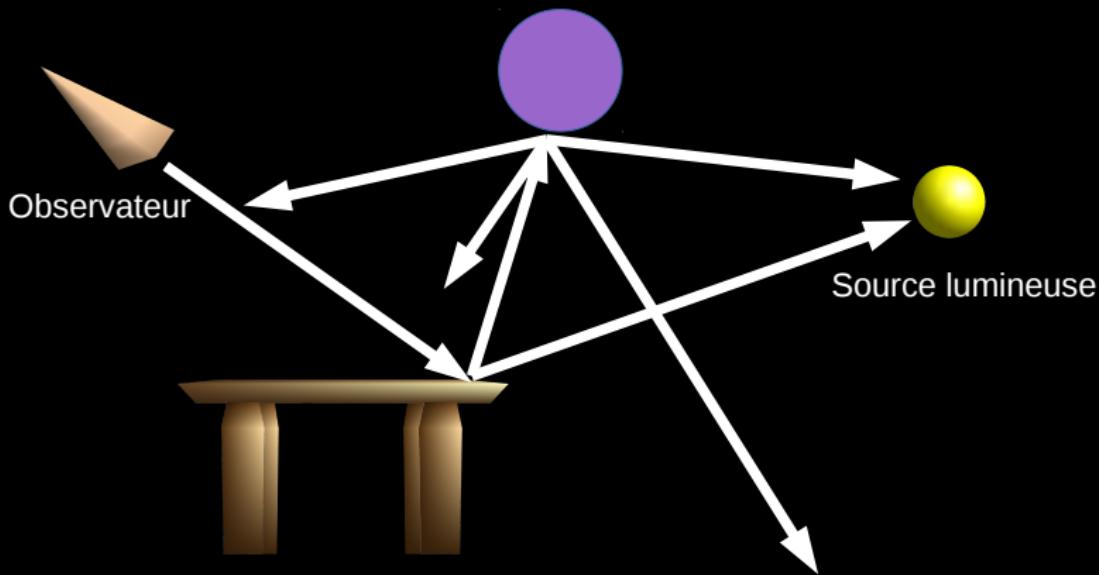
- Mesurée
  - Goniophotometer
  - ...
- Modèles
  - Blinn-Phong
  - Cook-Torrance
  - GGX
  - ...

$$f_r(x, \theta_i, \theta_o) = k_d f_d(x, \theta_i, \theta_o) + k_s f_s(x, \theta_i, \theta_o)$$

### Principe du rendu



### Principe du rendu

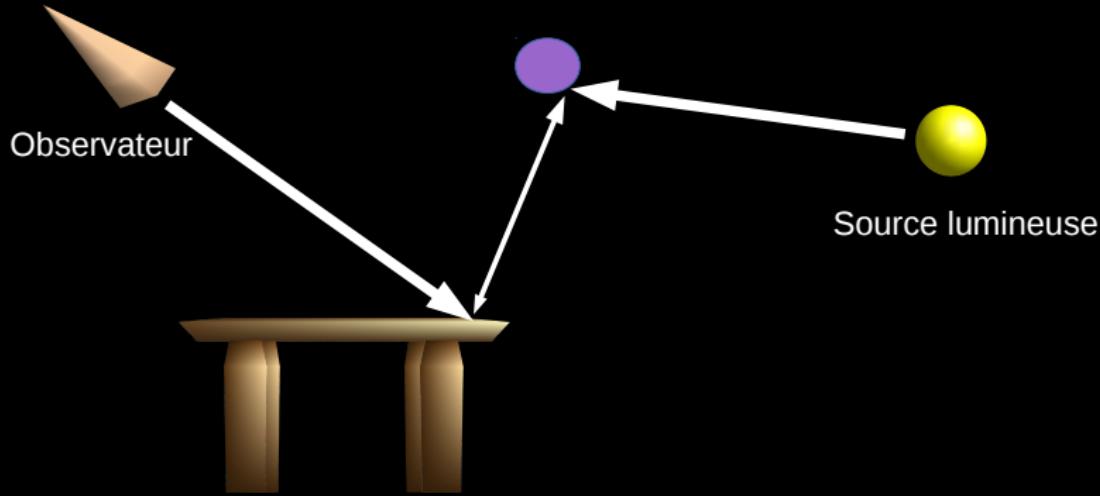


- Avantages :
- Inconvénients :

- Avantages :
  - Rendu réaliste
  - Convient bien aux scènes d'extérieurs
  - Prend bien en compte l'apport des autres objets
  - Rend les caustiques
  - Possibilité de modéliser les effets (profondeur de champ...)
- Inconvénients :
  - lent
  - bruité (Il faut beaucoup d'itérations pour converger)

## *Bidirectional Path Tracing*

- Amélioration du calcul du rendu
    - Lancement des rayons depuis l'observateur et depuis les sources
- 



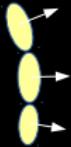
Avantages :

- Facilite la recherche de chemin vers la source lumineuse
- Permet de modéliser des petites sources lumineuses

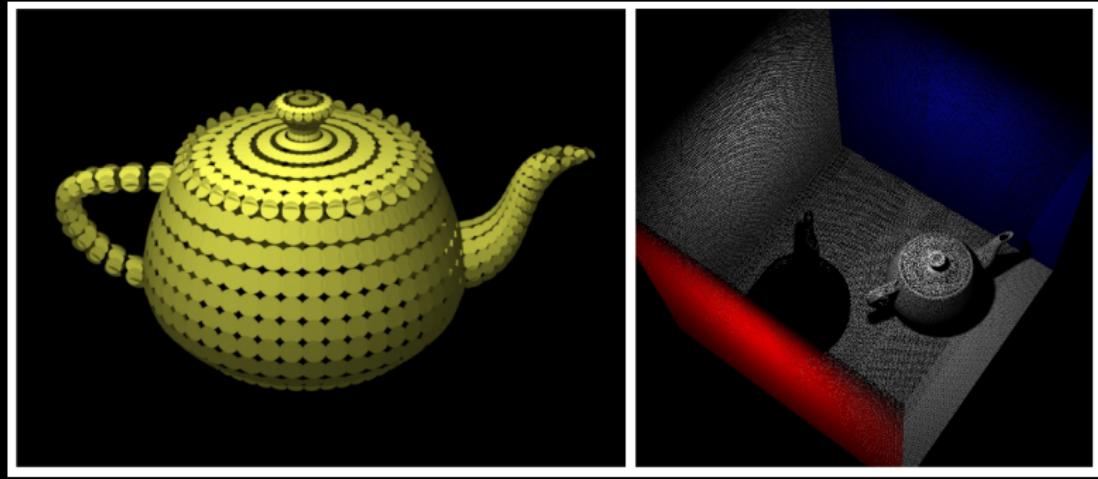


- Méthode pour estimer l'illumination globale
- Beaucoup utilisée pour le cinéma
- Avantages
  - Rapide
  - Image non bruitée (*pas d'artefacts temporel*)
- Inconvénients
  - Pas aussi précis que le *raytracing*
  - Difficile de gérer les effets miroir

- Approximation de la scène par nuage de points
  - Un point = un disque de couleur
  - Calcul de l'illumination directe de la scène



- Approximation de la scène par nuage de points
  - Un point = un disque de couleur
  - Calcul de l'illumination directe de la scène

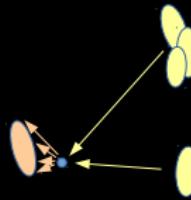


source : Point-Based Global Illumination for Movie Production, H. Christensen

## Regroupement des points

### Calcul de l'illumination globale

- Calcul de la contribution des points sur un point :
  - Pour les points éloignés
    - Utilisation du cluster
  - Pour les points proches
    - Raytracing
  - Pour les autres points
    - Utilisation directe du disque



## Bilan et remarques

## Illustration

```
#version 3.5;

#declare Radiosity=off;
#declare Photons=off;
#include "rad_def.inc"

global_settings{

    max_trace_level 15
    assumed_gamma 1
    #if (Radiosity=on)
    radiosity {
        Rad_Settings(Radiosity_Final,on,on)
    }
    #end
    #if (Photons=on)
    photons {
        spacing 0.002
    }
    #end
}
```

## Illustration

```
camera {  
    location <1,1.5,-3>  
    look_at <0,0,0>  
}  
  
light_source {  
    <0,4,-3>  
    rgb <1,1,1>  
}
```

## Illustration

```

plane {
    y
    0
    texture{
        finish{
            reflection 0.4
            diffuse 1
        }
        pigment {
            checker
            rgb <1,1,1>
            rgb <0.5,0.5,0.5>
        }
    }
    #if (Photons=on)
    photons {
        target
        reflection on
        refraction off
    }
    #end
}

box {
    <-0.5,0,-0.5>
    <0.5,1,0.5>
    pigment {
        rgb <1,0,0>
    }
    #if (Photons=on)
    photons {
        target
        reflection on
        refraction off
    }
    #end
}

sphere {
    <1,0.3,-1>
    0.2
    texture{
        finish {
            diffuse 1
        }
        pigment {
            rgb <0,0,1>
        }
    }
    #if (Photons=on)
    photons {
        target
        reflection on
        refraction off
    }
    #end
}

sphere {
    <-0.6,0.5,-1.4>
    0.3
    pigment {
        rgbt <1,1,1,1>
    }
    interior{ior 1.5}
    #if (Photons=on)
    photons {
        target
        reflection off
        refraction on
    }
    #end
}

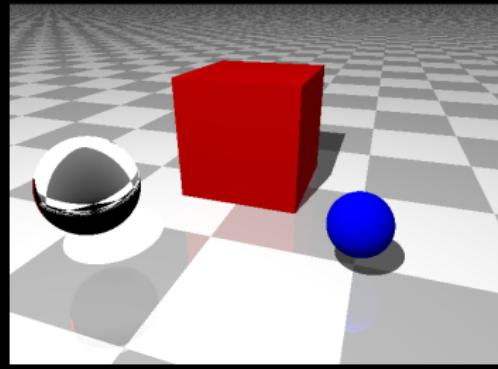
```

## Illustration

- Rendu simple

```
povray +Itest.pov +Otest.tga +W640 +H480
```

real 0m2.129s user 0m1.628s



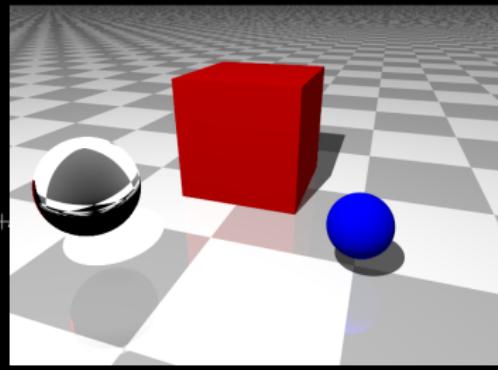
Calculé avec : Persistence Of Vision RAYtracer

## Illustration

- Rendu simple avec  
*anti-aliasing*

```
povray +ltest.pov +Otest_aa.tga +W640 +H480 +
```

real 1m8.129s user 1m6.420s



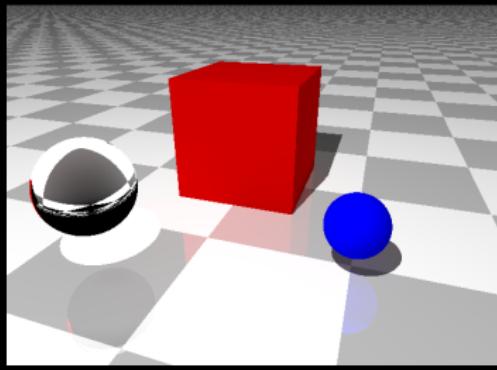
Calculé avec : Persistence Of Vision RAYtracer

## Illustration

- Rendu avec la radiosité

```
povray +Itest.pov +Otest_r.tga +W640 +H480
```

real 0m19.309s user 0m18.161s



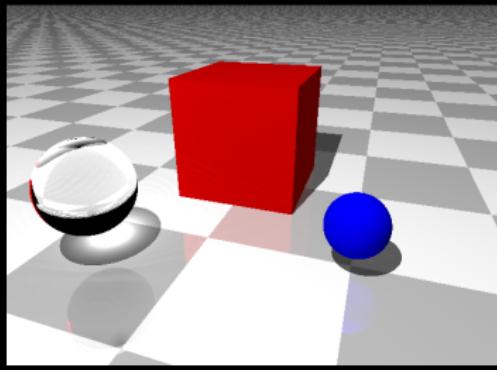
Calculé avec : Persistence Of Vision RAYtracer

## Illustration

- Rendu avec les photons

```
povray +Itest.pov +Otest_p.tga +W640 +H480
```

```
real 1m16.313s user 1m14.545s
```



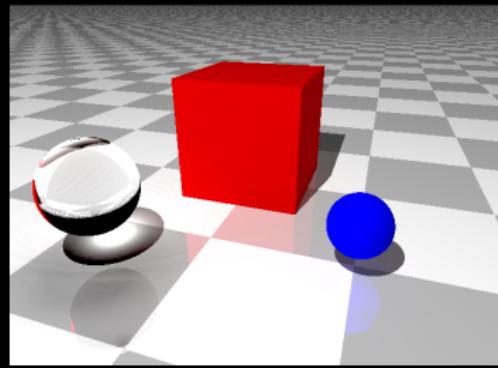
Calculé avec : Persistence Of Vision RAYtracer

## Illustration

- Rendu avec la radiosité et les photons

```
povray +ltest.pov +Otest_rp.tga +W640 +H480
```

real 2m39.034s user 2m36.274s



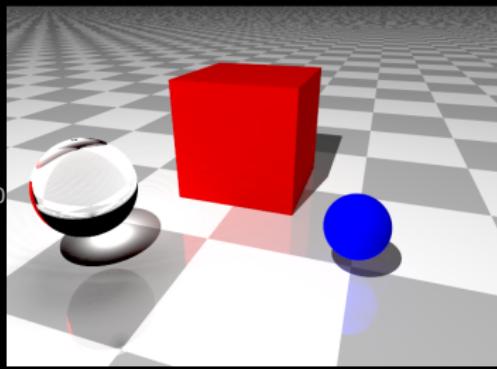
Calculé avec : Persistence Of Vision RAYtracer

## Illustration

- Rendu avec *anti-aliasing*

```
povray +Itest.pov +Otest_aarp.tga +W640 +H480
```

real 26m34.801s user 26m38.660s



Calculé avec : Persistence Of Vision RAYtracer

# Rendu photoréaliste

Bilan :

- *Raytracing*
  - Calcul l'illumination en fonction d'un point de vue
  - Calcul l'illumination approximatif : gère mal les objets transparents, les lumières secondaires, les ombres portées...
  - Améliorations avec le *raytracing* distribué
  - On peut combiner cet algorithme avec des techniques de calcul d'illumination globale pour palier à ces problèmes
- *Radiosity*
  - Calcul l'illumination globale
  - Gère que la diffusion mais améliore l'apport des lumières secondaires
- *PhotonMap*
  - Calcul l'illumination globale
  - Plus difficile à mettre en œuvre (implémentation, artéfacts...)
  - Gère bien les objets transparents (caustiques) et éventuellement les ombres portées et les sources secondaires
- *PathTracing*
  - Gère bien les objets transparents, les lumières secondaires, les ombres portées
  - Calcul très long
  - Risque d'apparition de bruit
- *P.B.G.I.*

- Remarques sur l'implémentation

- Remarques sur l'implémentation
  - Doit être bien réfléchie
  - Parallélisation possible
  - Utilisation du *GPU* possible
  - Débruitage possible
  - ...

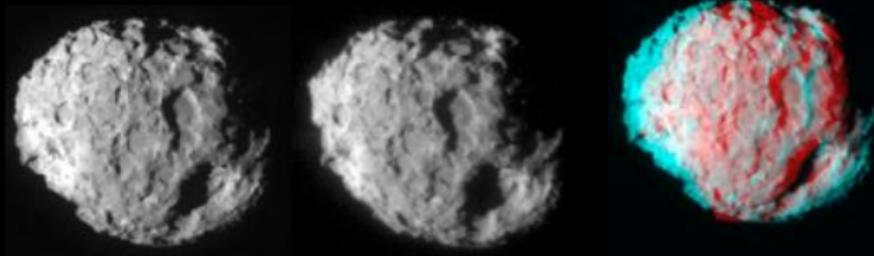
### Modélisation :

- Pour chaque « forme » il faut être capable de :
  - calculer la normale en chaque point,
  - calculer l'intersection avec un droite,
  - éventuellement calculer les coordonnées de la texture

- Calcul des intersections : dans le repère monde ou le repère objet ?

Pour aller plus loin :

- Textures
- Autres effets (Brouillard, Bleu atmosphérique, ...)
- ...
- génération d'anaglyphes (cyan et rouge (espacement  $1/30 * f$ ))



source : NASA

Fin ?

Pas encore...

## *Raycasting*

Principe :

- On ne lance que les rayons depuis l'observateur et on ne calcule pas les rebonds...

(Raytracing est une extension du raycasting ?)

Wolfenstein :

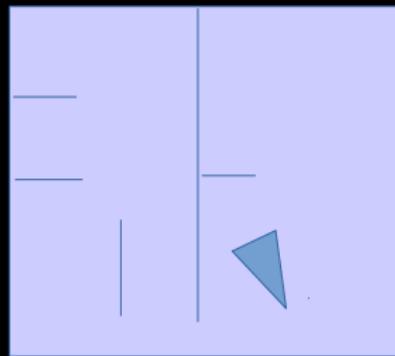
- on lance les rayons dans le plan !



source : Wolfenstein

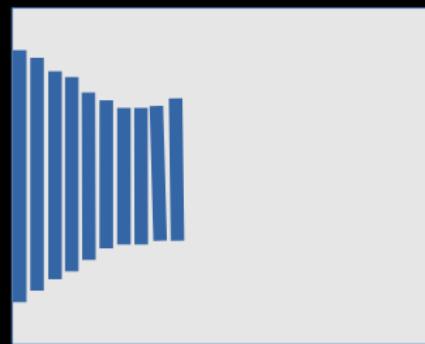
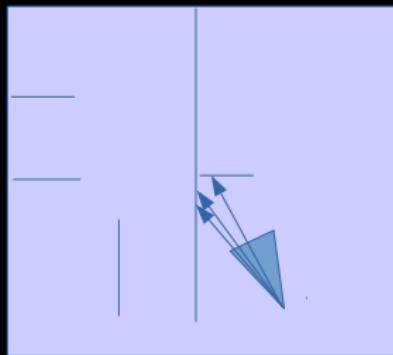
Wolfenstein :

- on lance les rayons dans le plan !



Wolfenstein :

- on lance les rayons dans le plan !

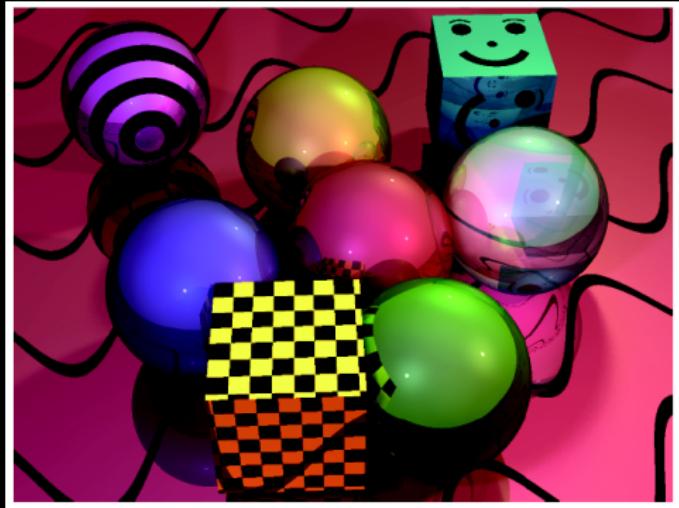


La longueur du rayon permet de conclure sur la hauteur du mur

- 1 rayon donne 1 colonne de l'image + gestion des objets

- Algorithme rapide
- On est loin du rendu photoréaliste...

Cette fois, c'est la fin !



source : Michel Huynh & Samuel Kvaalen

