

GÉOMÉTRIE ALGORITHMIQUE

RAY TRACING

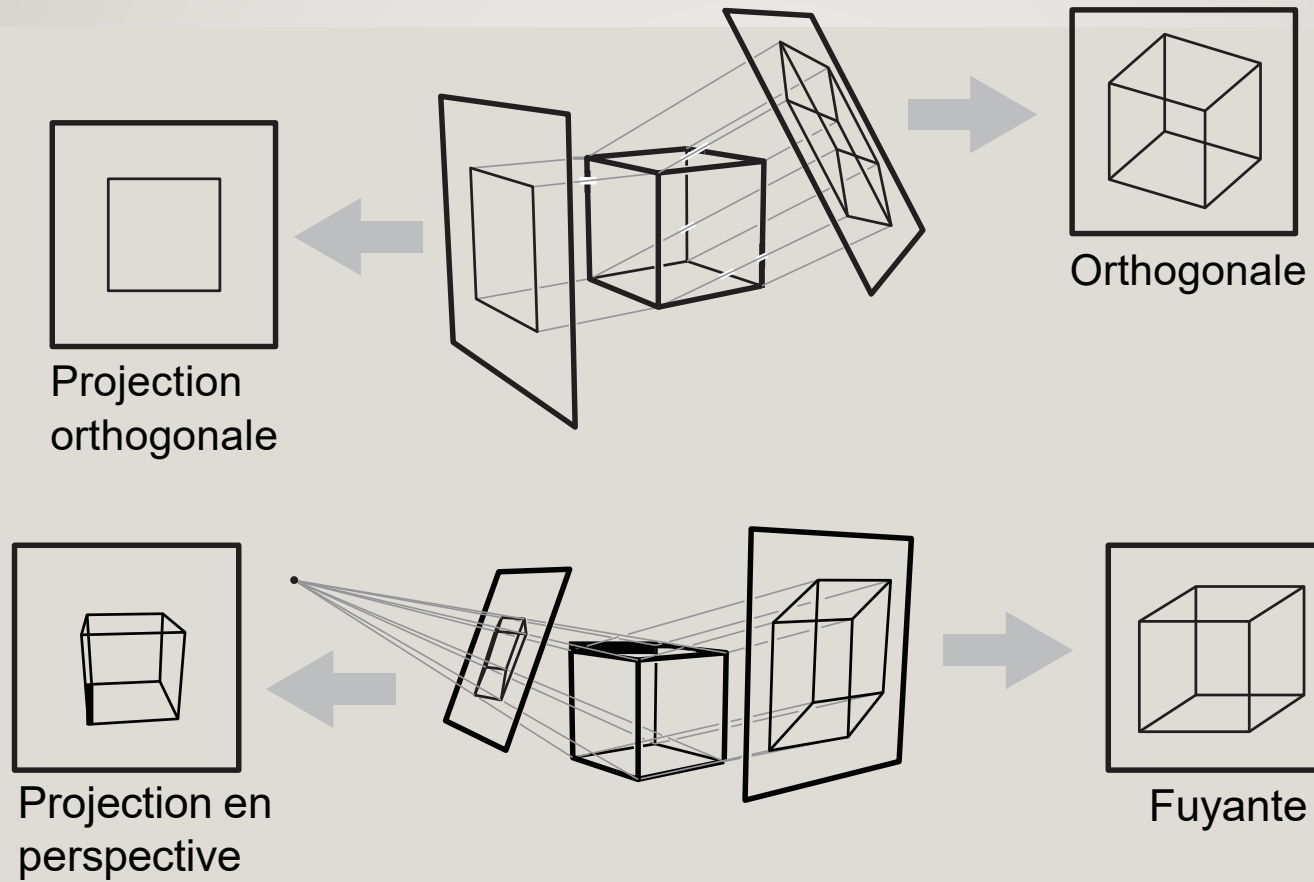
INTRODUCTION

- L'affichage d'objets 3D par Rasterization est une approximation qui a le mérite de permettre des calculs très rapides. Mais celui-ci a des limites notamment :
 - Difficulté à calculer des réflexions d'objets mobiles
 - Impossibilité de simuler de manière réaliste les objets transparents
 - Calcul des ombres complexe avec une précision limitée
- L'approche par lancé de rayons (Ray Tracing) permet de palier à tous ces inconvénients mais nécessite un temps de calcul bien supérieur. On la réserve donc au calcul d'images fixes ou de séquences vidéo précalculées.

LANCÉ DE RAYONS

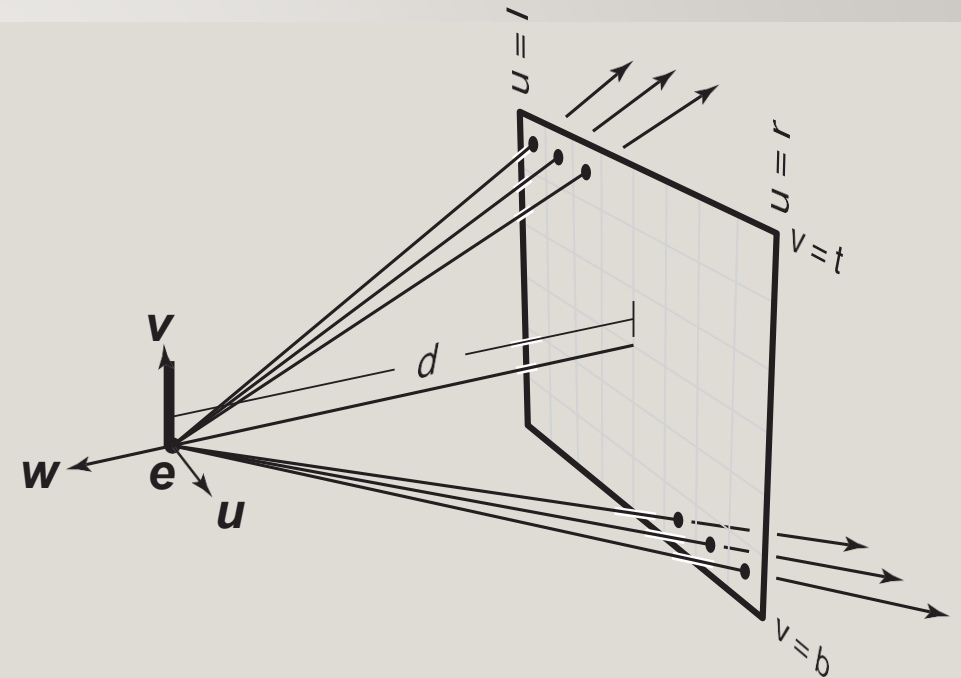
Parcourir une scène avec des rayons

LES PROJECTIONS



RAYONS ENVOYÉS DEPUIS LA CAMERA

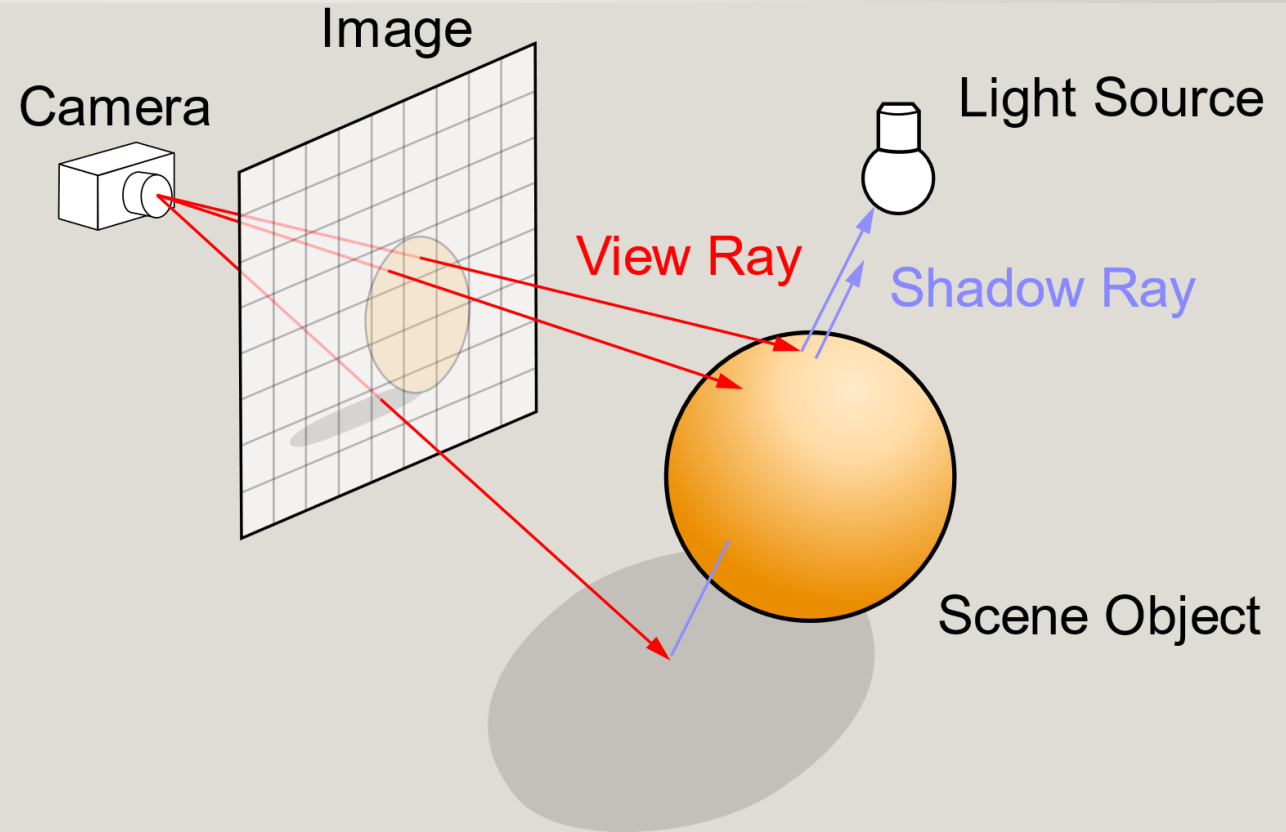
- Les rayons sont envoyés depuis la caméra en direction de la scène en parcourant le plan correspondant à l'écran
- L'objectif est de calculer la couleur de chaque pixel de l'écran en suivant le trajet du rayon effectuant le trajet inverse de la lumière. Si le rayon atteint une source de lumière, on calcule sa couleur en fonction des surfaces rencontrées, sinon on prend la couleur du fond de la scène.



Project perspective
une seule origine, plusieurs
directions

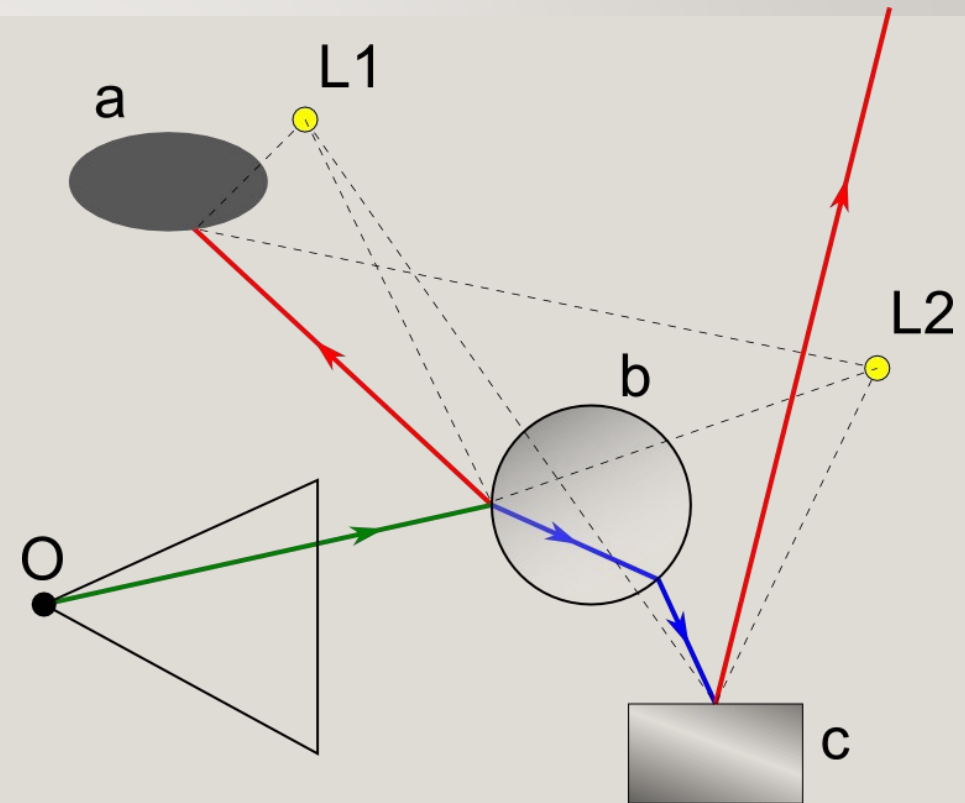
PARCOURS DU RAYON

- Le rayon part de la caméra, rencontre les différents objets de la scène.
- Au point de contact, un rayon de type ombre est envoyé vers la lumière pour savoir si le point est éclairé.
- Si le point est éclairé la couleur est calculée en utilisant l'éclairage Phong.



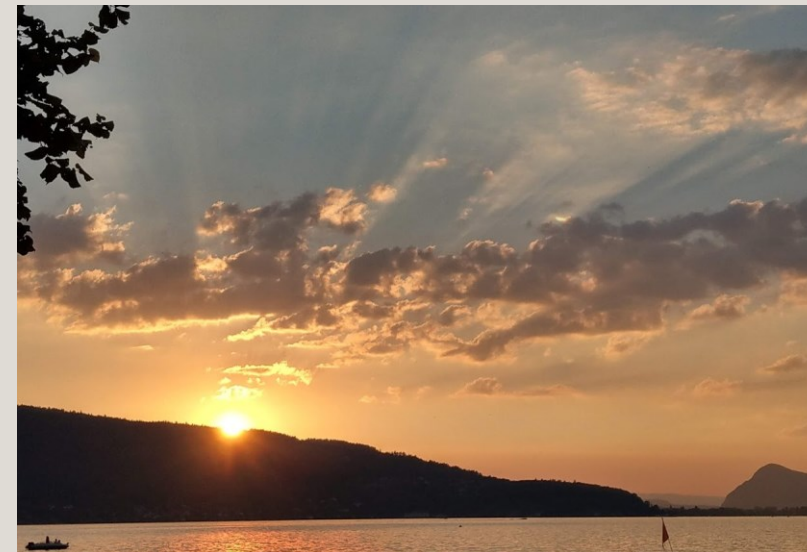
RECURSIVITÉ

- Sur une surface transparente 'b' avec un indice de réfraction, les rayons peuvent être réfléchis (rouge) ou réfractés (bleu).
- Sur une surface spéculaire 'c', les rayons sont réfléchis (rouge).
- Sur une surface diffuse 'a', les rayons sont absorbés.
- A toutes les intersections, les rayons d'ombres sont lancés vers chacune des sources lumineuses (pointillés).



ALGORITHME RAY CASTING

```
pour chaque ligne y de l'image faire
  pour chaque colonne x de l'image faire
    i ← MAX_DIST
    générer un rayon de vue à travers le pixel(x,y)
    pour chaque objet de la scene faire
      calculer l'intersection rayon / objet
      si (intersection et plus courte distance à l'œil) alors
        sauver i
      fsi
    fpour
    si (i != MAX_DIST) alors
      calculer la normale à la surface au point i
      pour chaque source lumineuse / faire
        générer un rayon de i vers l
        pour chaque objet de la scene (sauf celui de i) faire
          calculer l'intersection rayon / objet
          si (pas d'obstacle) alors
            calculer la couleur et l'intensité de i en fonction de l
          fsi
        fpour
      fpour
    sinon // aucune intersection entre le rayon de vue et la scène
      couleur et insensite du fond
    fsi
  sauver le pixel(x,y, couleur)
fpour
```



ALGORITHME RAY TRACING

```
Algorithme Raytracing(orig, dir, prof , coul )
données
    orig , dir  : vecteur
    prof : entier
    coul : couleur
variables
    pt_inter, dir_reflexion, dir_transmission : vecteur
    coul_loc, coul_ref, coul_trans : couleur
    trans : couleur

si (prof > PROF_MAX) alors
    coul ← noir
sinon
    trouver le point d'intersection le plus proche (s'il existe)
    si (pas d'intersection) alors
        coul ← COUL_FOND
    sinon
        coul_loc ← contribution de la coul. locale au pt d'intersection
        dir_reflexion ← calcul de la direction du rayon réfléchi
        Raytracing(pt_inter, dir_reflexion, prof+1, coul_ref)
        dir_transmission ← calcul de la direction du rayon transmis
        Raytracing(pt_inter, dir_transmission, prof+1, coul_trans)
        coul ← Combiner(coul_loc, ka , coul_ref, kr , coul_trans, kt )
    fsi
fsi
```



ALGORITHME OMBRE PORTÉS

```
Algorithme Ombre porté (I,S)
creer un rayon R depuis I vers la lumiere S
creer la liste L des intersections de R
si (L n'est pas vide) alors
    si (L[0] > DIST(I,S)) alors
        retourner couleur noire
    sinon
        retourner l'illumination normale
fsi
```

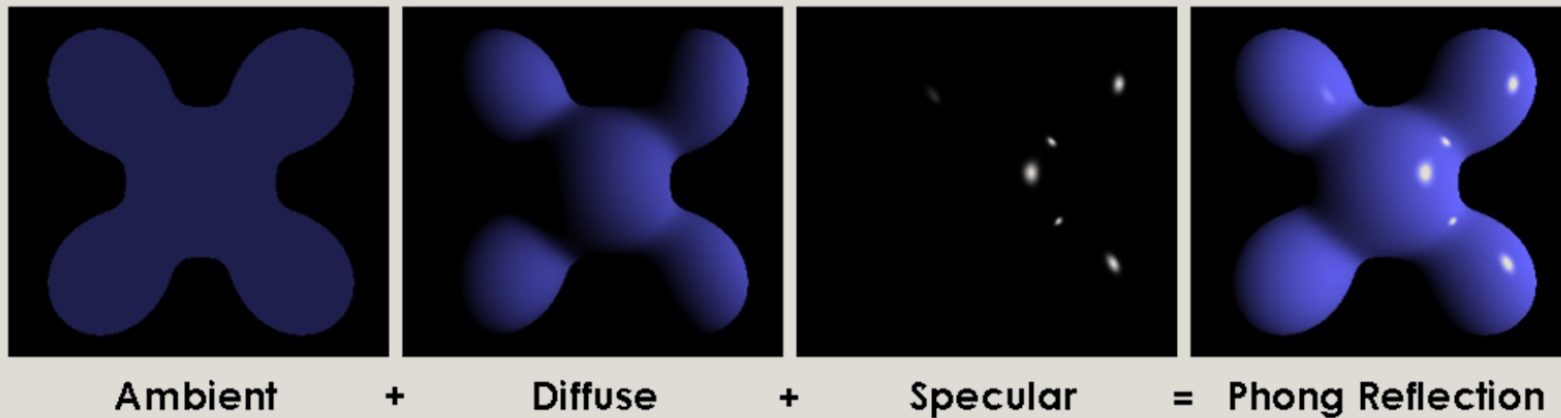


CALCUL DE LA LUMINANCE

$$I = k_a I + k_d I (L \cdot N) + k_s I (R \cdot V)^n$$

I_a contribution ambiante, I_i intensité de la lumière incidente, k_d et k_s contributions diffuse et spéculaire, toutes deux dépendantes du matériau de la surface, et n coefficient de surbrillance.

Cette formule est utilisée si le point d'incidence est en vue directe de la lumière (vecteur ombre ne rencontre pas d'obstacle sinon seule la lumière ambiante est utilisée). Si plusieurs source de lumières sont présentent dans la scène on additionne la contribution de toutes celles qui éclairent le point. A chaque reflexion ou réfraction un facteur de transmissivité ou/et de réflexion est utilisé pour pondérer la lumière. La couleur finale est la somme des luminances calculées dans le parcours du rayon.



INTERSECTION DROITE / SPHERE

Soit un segment de droite défini par :

$$I = A + tD$$

A un point de la droite et D son vecteur directeur unitaire.

D'autre part, l'équation d'une sphère de centre (l, m, n) et de rayon r est :

$$(x - l)^2 + (y - m)^2 + (z - n)^2 = r^2$$

Remarque : normale à une surface sphérique, soit (x_i, y_i, z_i) les coordonnées du point d'intersection, la normale à la surface en ce point est donnée par :

$$N = \left(\frac{x_i - l}{r}, \frac{y_i - m}{r}, \frac{z_i - n}{r} \right)$$

INTERSECTION DROITE / SPHERE

En substituant x, y et z par leur forme paramétrée $A + tD$ dans l'équation implicite de la sphere on obtient une équation quadratique en t de la forme :

$$at^2 + bt + c = 0$$

$$\Delta = b^2 - 4ac$$

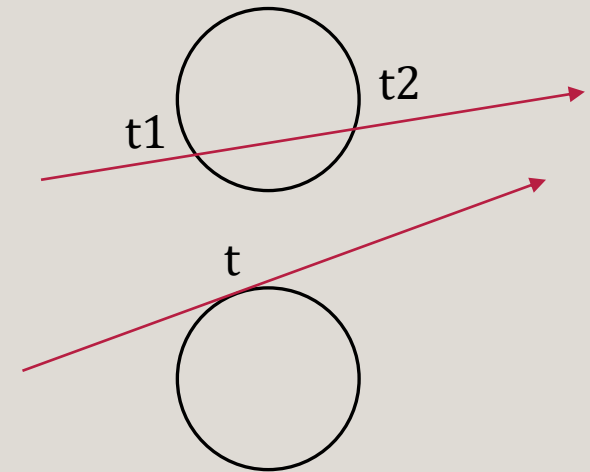
- Si $\Delta > 0$, il y a deux solutions réelles à l'équation,

$$t1 = \frac{-b - \sqrt{\Delta}}{2a} \text{ et } t2 = \frac{-b + \sqrt{\Delta}}{2a}$$

- Si $\Delta = 0$, le segment est tangent à la sphere

$$t = \frac{-b}{2a}$$

- Si $\Delta < 0$, le segment de droite n'intersecte pas la sphère



INTERSECTION DROITE / TRIANGLE

- Equation d'un plan : $ax + by + cz = d$
- a, b et c constituent les coordonnées du vecteur normal au plan, $n = [a \ b \ c]$. L'équation du plan devient : $n \cdot v = d$
- Si on considère l'équation du rayon : $A + tD$
- En substituant v par $A + tD$ on obtient comme valeur de t :

$$t = \frac{d - n \cdot A}{n \cdot D}$$

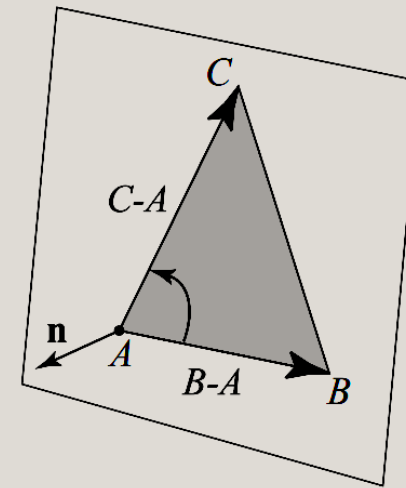
INTERSECTION DROITE / TRIANGLE

- Nous avons désormais le point d'intersection du rayon sur le plan, il faut maintenant déterminer si ce point est dans le triangle.

- Pour calculer la normal n nous avons l'équation :

$$n = \frac{(B - A) \times (C - A)}{\|(B - A) \times (C - A)\|}$$

- n doit être normalisé pour fonctionner dans les calculs suivantes



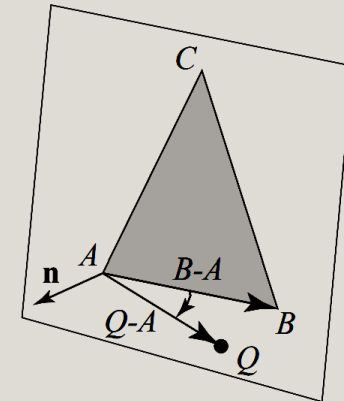
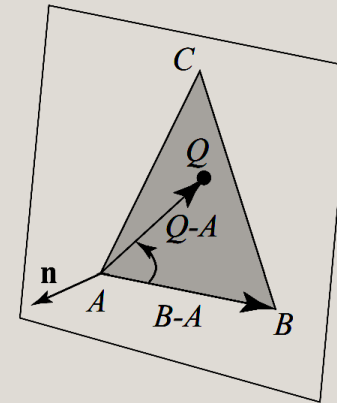
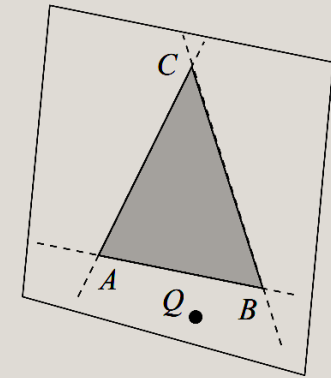
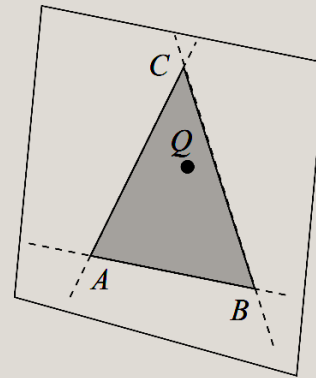
INTERSECTION DROITE / TRIANGLE

- Pour savoir si notre point est dans le triangle il suffit de savoir de quel côté de chaque segment il se trouve on utilise le produit vectoriel de AB et AQ par exemple et on compare son signe à celui de la normale.
- Le point est dans le triangle quand les 3 équations suivantes sont vérifiées

$$[(B - A) \times (Q - A)] \cdot n \geq 0$$

$$[(C - B) \times (Q - B)] \cdot n \geq 0$$

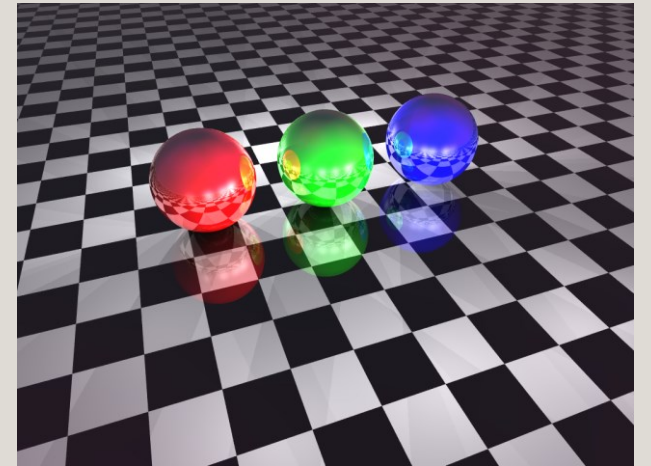
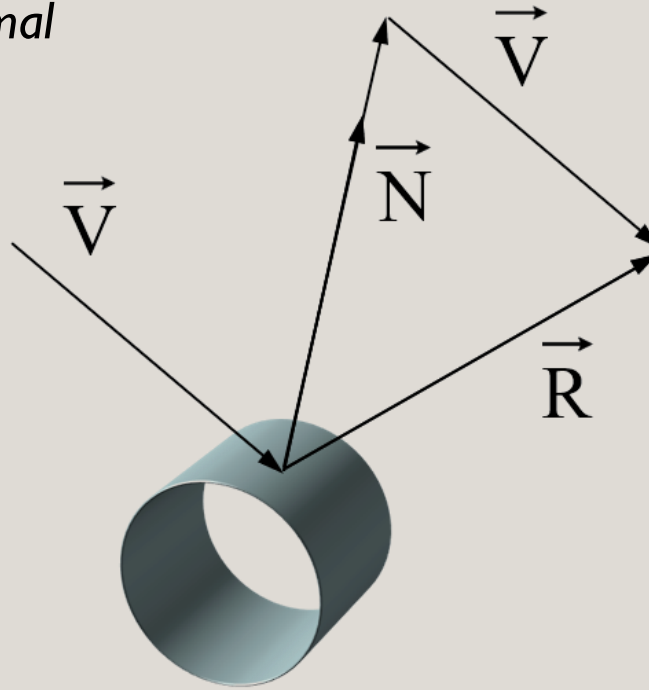
$$[(A - C) \times (Q - C)] \cdot n \geq 0$$



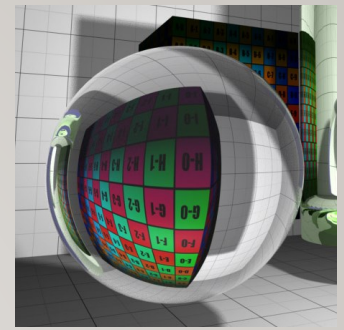
RÉFLEXION

- On calcule le rayon réfléchi à partir du vecteur de l'observateur et du vecteur normal

$$\vec{R} = -2(\vec{N} \cdot \vec{V}) \cdot \vec{N} + \vec{V}$$



RÉFRACTION

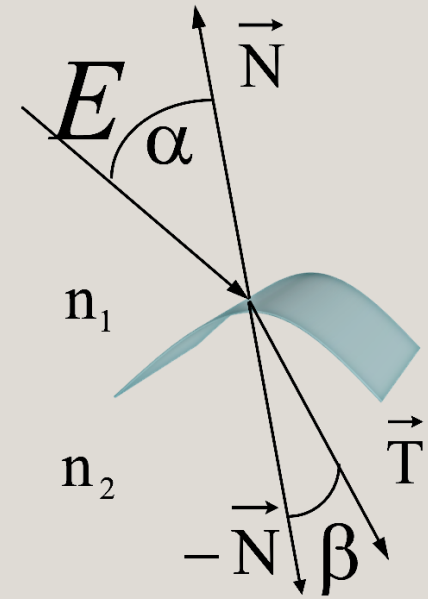


$$n_1 \sin(\alpha) = n_2 \sin(\beta)$$

$$\frac{\sin(\alpha)}{\sin(\beta)} = \frac{n_2}{n_1} = n_{21}$$

- En utilisant la loi de Snell-Descartes on obtient

$$\vec{T} = -n_{12}\vec{E} + \vec{N} \left(n_{12} \cdot \cos(\alpha) - \sqrt{1 + n_{12}^2 \cdot (\cos^2(\alpha) - 1)} \right)$$

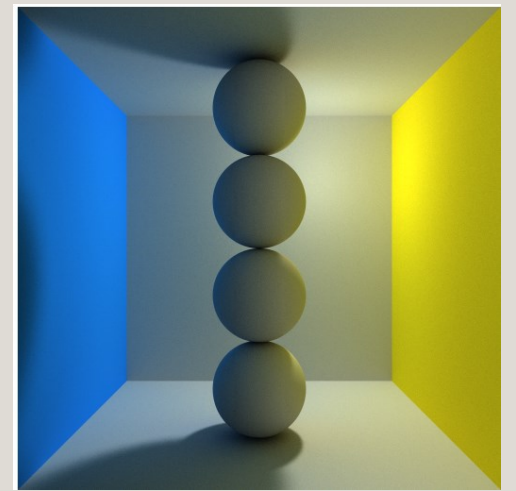
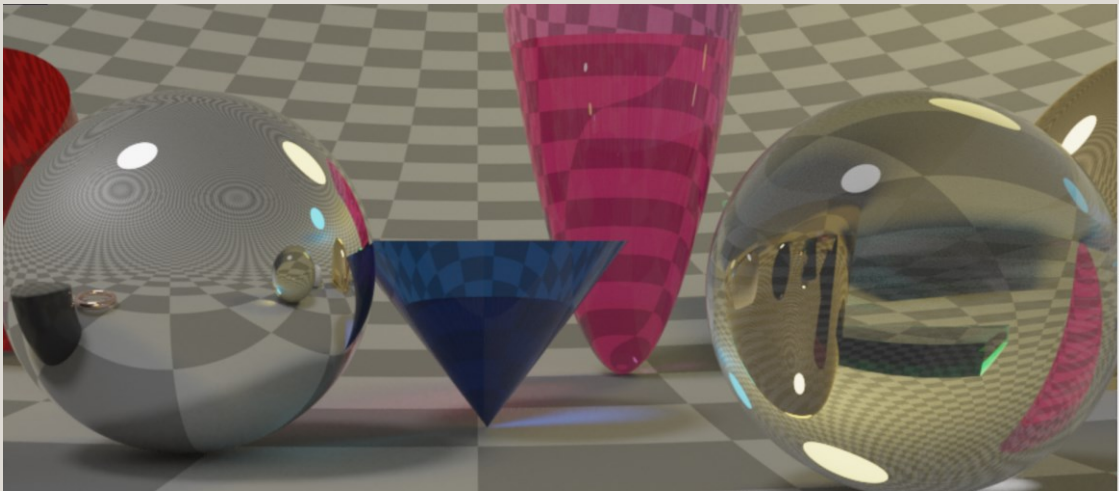


RESULTAT FINAL



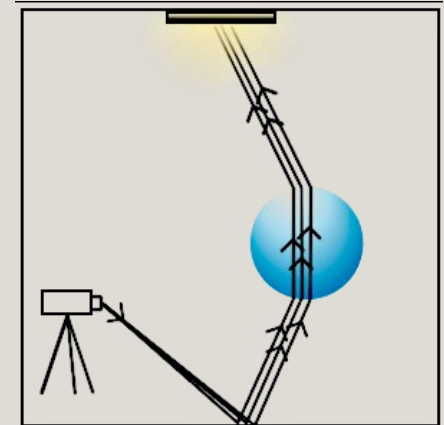
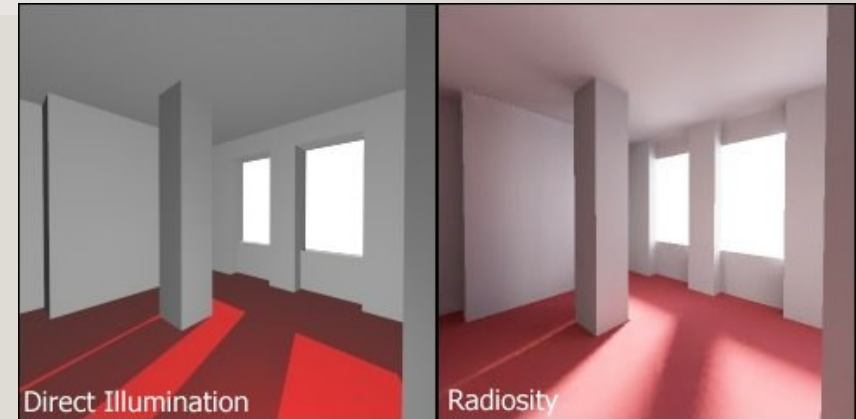
DEMOS

- https://erichlof.github.io/THREE.js-PathTracing-Renderer/Geometry_Showcase.html
- https://erichlof.github.io/THREE.js-PathTracing-Renderer/Cornell_Box.html
- <http://madebyevan.com/webgl-path-tracing/>



RADIOSITÉ

- Pour améliorer le rendu d'une scène il est possible d'envoyer plusieurs rayons par pixel au hasard (Metropolis Light Transport) et ainsi éviter d'utiliser des astuces pour faire un rendu réaliste.
- A gauche il faut 3 sources de lumières une ambiante, un spot pour les ombres et une lumière omnidirectionnelle sans ombre pour améliorer le rendu
- A droite en radiosit  une seule source de lumi re est utilis e. Celle-ci doit avoir des caract ristiques g om trique pour  tre atteinte par le rayon final contrairement   l'approche Phong qui utilise une lumi re ponctuelle.



APPROCHE HYBRIDE

- Il est possible de mélanger la Rasterization avec la Ray Tracing, c'est ce que font les cartes Nvidia RTX.
- La Rasterization est faite normalement mais les textures de reflets et les ombres sont calculées en temps réel par lancé de rayons sur les objets de la scène.
- De nombreuses combinaisons peuvent être utilisées pour permettre un rendu temps réel plus réaliste.