

# M1-Images

## Lancer de rayons

J.C. Iehl

October 4, 2022

## résumé des épisodes précédents

pipeline graphique :

- ▶ décrire l'objet en fonction de la méthode d'affichage,
- ▶ pipeline openGL / fragmentation / rasterization.

## afficher des objets

plusieurs problèmes :

- ▶ problème 1 : déterminer où se trouve l'objet (par rapport à la camera),
- ▶ problème 2 : déterminer l'ensemble de pixels (correspondant à la forme de l'objet),
- ▶ problème 3 : donner une couleur à chaque pixel.

## afficher des objets

### 2 organisations :

- ▶ pour chaque objet : déterminer l'ensemble de pixels, (que se passe-t-il lorsque plusieurs objets se "dessinent" sur le même pixel ?)
- ▶ pour chaque pixel : trouver l'objet visible,

trouver l'objet visible pour chaque pixel : trouver l'objet le plus *proche* de la camera.

## afficher des objets

2 cours :

- ▶ la dernière fois : openGL et carte graphique, solution 1,
- ▶ aujourd'hui : lancer de rayons, solution 2.

## lancer de rayons

pour chaque pixel :

- ▶ générer un rayon,
- ▶ pour chaque objet :
- ▶ calculer l'intersection,
- ▶ garder l'objet le plus proche de la camera...

calculer la couleur du pixel en fonction de l'objet trouvé...

## générer un rayon

rayon :

- ▶ droite passant par le centre d'un pixel de l'image
- ▶ ...

trouver 2 points ? ou 1 point et une direction ?

## générer un rayon

2 points :

- ▶ le rayon passe par la camera (son centre de projection),
- ▶ et par un point au centre du pixel...

position de la camera dans le repère du monde ?

position d'un pixel dans le repère du monde ?



## g n rer un rayon

transformations :

- ▶ model, view, projection, image...
- ▶ ou se trouve la camera dans le rep re du monde ?
- ▶ ou se trouve le pixel  $x, y$  dans le rep re du monde ?

## générer un rayon

origine :

- ▶ la camera se trouve en  $\{0, 0, 0\}$  dans le repère... camera !
- ▶ transformation dans le repère du monde...

$$o(x, y) = [V]^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

## générer un rayon

extrémité :

- ▶ pixel  $x, y$ , dans le repère image...
- ▶ et le  $z$  ?
- ▶  $\{x, y, z \equiv 0\}$  sur le plan proche,
- ▶  $\{x, y, z \equiv 1\}$  sur le plan loin,
- ▶ transformation dans le repère du monde...

$$e(x, y, z) = [I \times P \times V]^{-1} \begin{bmatrix} x \\ y \\ z \equiv 0, 1 \\ 1 \end{bmatrix}$$

## générer un rayon

plusieurs solutions :

- ▶ on connaît 3 points sur le rayon...
- ▶  $o(x, y)$ ,  $e(x, y, 0)$ ,  $e(x, y, 1)$ ,
- ▶ il suffit d'en choisir 2 :
- ▶  $o(x, y)$  et  $e(x, y, 0)$ ,
- ▶ ou  $e(x, y, 0)$  et  $e(x, y, 1)$ .  
(permet de reproduire exactement la projection d'openGL)

# intersections

cas simples :

- ▶ plan,
- ▶ sphere,
- ▶ triangle,
- ▶ cube.

utiliser une représentation de l'objet et du rayon permettant de faire le calcul...

## intersections : rayon / plan

rappels :

- ▶  $p(t) = o + t \cdot \vec{d}$ ,
- ▶  $\vec{n} \cdot \overrightarrow{ap} = 0$  sur le plan de normale  $\vec{n}$  passant par  $a$  et  $p \in \text{plan}(a, \vec{n})$ .

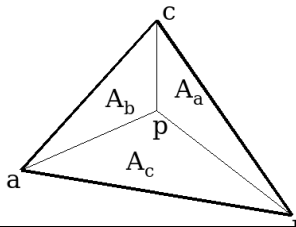
résultat :

- ▶  $\vec{n} \cdot \overrightarrow{ap(t)} = 0$
- ▶  $\vec{n} \cdot ((o + t \cdot \vec{d}) - a) = 0$
- ▶  $\vec{n} \cdot ((o - a) + t \cdot \vec{d}) = 0$
- ▶  $t = \frac{(a-o) \cdot \vec{n}}{\vec{d} \cdot \vec{n}}$

## intersections : rayon / triangle

rappels :

- ▶  $p(t) = o + t \cdot \vec{d}$ ,
- ▶  $p(\alpha, \beta) = \alpha a + \beta b + (1 - \alpha - \beta)c$  si  $p \in \text{triangle}(a, b, c)$
- ▶  $\alpha = A_b/A$ ,  $\beta = A_a/A$
- ▶  $\gamma = 1 - \alpha - \beta = A_c/A$



## intersections : rayon / triangle

résultat :

"Fast, Minimum Storage Ray-Triangle Intersection"

T. Moller, B. Trumbore, 1997



## intersections : rayon / sphere

rappels :

- ▶  $p(t) = o + t \cdot \vec{d}$ ,
- ▶  $(p_x - c_x)^2 + (p_y - c_y)^2 + (p_z - c_z)^2 - R^2 = 0$  si  
 $p \in \text{sphere}(c, R)$
- ▶  $(p - c) \cdot (p - c) - R^2 = 0$

résultat :

- ▶  $(o + t \cdot \vec{d} - c) \cdot (o + t \cdot \vec{d} - c) - R^2 = 0$
- ▶  $(\vec{d} \cdot \vec{d})t^2 + 2\vec{d} \cdot (o - c)t + (o - c) \cdot (o - c) - R^2 = 0$

## intersections : rayon / boite orientée

rappels :

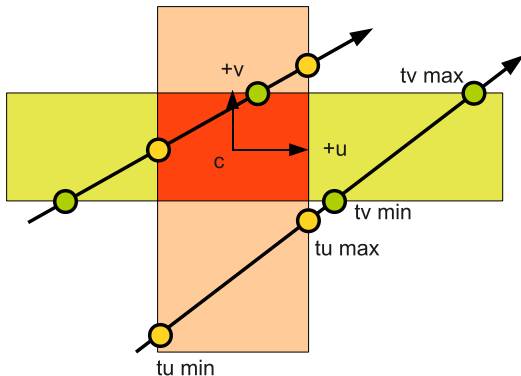
- ▶  $p(t) = o + t \cdot \vec{d}$ ,
- ▶  $p \in \text{boite}(c, \vec{u}, \vec{v}, \vec{w})$
- ▶ la boite est l'intersection de 3 paires de plans parallèles :
- ▶  $\text{plan}(c - \vec{u}, -\vec{u})$ ,  $\text{plan}(c + \vec{u}, \vec{u})$ ,
- ▶  $\text{plan}(c - \vec{v}, -\vec{v})$ ,  $\text{plan}(c + \vec{v}, \vec{v})$ ,
- ▶  $\text{plan}(c - \vec{w}, -\vec{w})$ ,  $\text{plan}(c + \vec{w}, \vec{w})$ ,

résultat :

- ▶ calculer  $t_{min}$  et  $t_{max}$  pour chaque paire de plans (cf. intersection rayon / plan),

## intersections : rayon / boite alignée sur les axes

- l'intersection existe si l'intersection des intervalles n'est pas vide :  $\max(t_{min}^u, t_{min}^v, t_{min}^w) < \min(t_{max}^u, t_{max}^v, t_{max}^w)$



## intersections : rayon / boite alignée sur les axes

détails et astuces de calculs :

"An Efficient and Robust Ray-Box Intersection Algorithm"

A. Williams, S. Barrus, R.K. Morey, P. Shirley, 2005

## intersection : plusieurs objets

et avec plusieurs objets ?

- ▶ le plus simple : tester tous les objets,
- ▶ et ne garder que l'intersection valide la plus proche...

mais : peut mieux faire...

schema.

## intersection : plusieurs objets

### hiérarchie de volumes engobants :

- ▶ grouper les objets (ou les primitives) proches,
- ▶ calculer une forme simple englobant le groupe,
- ▶ si un rayon ne touche pas l'englobant, il ne touche pas non plus les objets du groupe,
- ▶ recommencer jusqu'à obtenir un seul groupe...

## intersection : plusieurs objets

intersection rayon / arbre :

- ▶ si le rayon touche l'englobant de la racine de l'arbre,
- ▶ visiter les fils du noeud,
- ▶ recommencer...

quels noeuds / feuilles vont être visités ?

schema.

## intersection : plusieurs objets

comment trouver rapidement l'intersection la plus proche  
(de l'origine du rayon) ?

ou se trouve l'intersection la plus proche de l'origine du rayon  
(a priori) ?

peut-on limiter le nombre de feuilles visité ?



## intersection rayon / arbre

trouver l'intersection la plus proche :

- ▶ qu'est ce que change l'ordre de visite des fils ?
- ▶ idée : visiter en premier le fils le plus proche de l'origine du rayon.

schema.

## intersection rayon / arbre

algorithme :

- ▶ si le rayon  $o + [t_{min}..t_{max}] \cdot \vec{d}$  touche l'englobant :
- ▶ calculer les intersections du rayon et des 2 fils,  $t_{gauche}$  et  $t_{droite}$
- ▶ visiter, en premier, le fils "proche",
- ▶ puis, visiter le fils "loin".

fils "proche" :  $\min(t_{gauche}, t_{droite})$ ,

fils "loin" :  $\max(t_{gauche}, t_{droite})$ .

## intersection rayon / arbre

limiter le nombre de feuilles visitées :

- ▶ ou se trouvent les groupes d'objets intéressants ?
- ▶ entre l'origine et une intersection, le rayon est un segment, plus une demi-droite...

schema.

## intersection rayon / arbre

algorithme :

- ▶ init :  $t = \infty$
- ▶ si le rayon  $o + [t_{min}...t] \cdot \vec{d}$  touche l'englobant du noeud :  
calculer les intersections avec les fils,  
visiter le fils proche, puis le fils loin
- ▶ si le noeud est une feuille :  
calculer l'intersection avec les primitives de la feuille,  
 $t_{feuille} = \min(t_1, t_2, \dots)$   
 $t = \min(t, t_{feuille})$

## intersection : plusieurs objets

### attention :

- ▶ les coordonnées de l'origine et de l'extrémité du rayon sont dans le repère du monde,
- ▶ dans quel repère connaît-on les coordonnées des objets ?
- ▶ si nécessaire, changement de repère de l'objet ou du rayon...

arbre à 2 niveaux : niveau 1 place les objets, niveau 2 primitives de chaque objet, dans leur repère local...

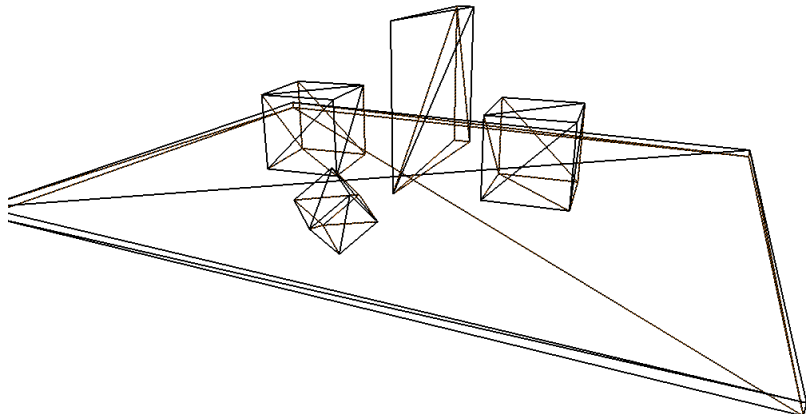
## bilan

pour chaque pixel :

- ▶ générer un rayon,
- ▶ pour chaque objet :
- ▶ calculer l'intersection,
- ▶ garder l'objet le plus proche de la camera...

calculer la couleur du pixel en fonction de l'objet trouvé... cf cours sur les matières

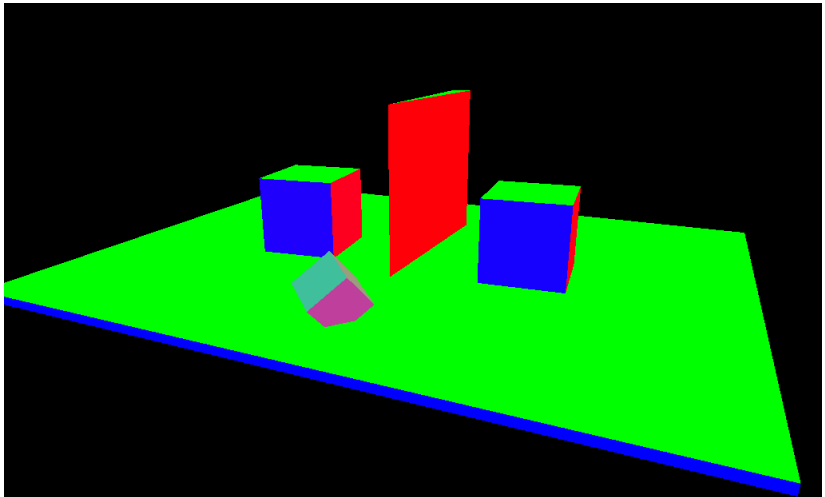
## exemple : géométrie scène test



Introduction  
Lancer de rayons  
Intersections  
et alors ?

notions  
j'aime pas les triangles...  
c'est trop lent...

exemple :  $abs(normal)$





## calculer une couleur...

qui dépend de l'orientation de l'objet :

- ▶ pourquoi ?
- ▶ réalité physique...

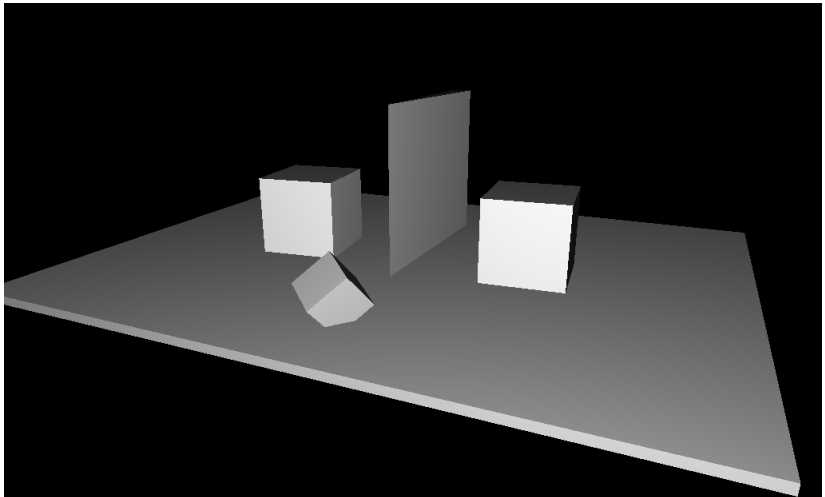
et accessoirement, ca permet de distinguer les objets...

Introduction  
Lancer de rayons  
Intersections  
et alors ?

notions

j'aime pas les triangles...  
c'est trop lent...

exemple : orientation



## orientation ?

laquelle ?

- ▶ dans le repère local, monde, camera ?
- ▶ ...

## orientation ?

laquelle ?

- ▶ par rapport à la position de la camera,
- ▶ éclaire l'objet avec une "torche" ...

comment éclairer un objet par rapport à une lumière ?

## quel est l'interet du lancer de rayons ?

quelles sont les différences :

- ▶ fragmentation de primitives (points, lignes, triangles) utilisée par une carte graphique,
- ▶ calcul des intersections avec un rayon ?

## quel est l'interet du lancer de rayons ?

### intersections " plus souples" :

- ▶ pas obligé de découper la surface des objets en triangles,
- ▶ intersection avec les " vraies" surfaces : plans, spheres, etc.
- ▶ intersection avec des formes différentes : nuages, fractales, champ de distance,

### simulations " plus simples" :

- ▶ reflets, transparence,
- ▶ ombres,
- ▶ éclairage global...

mais on peut faire ça : champ de distance



cf [shadertoy](#)

## quels sont les problèmes du lancer de rayons ?

### memoire :

- ▶ fragmentation : traite un seul triangle à la fois, pour un seul point de vue
- ▶ lancer de rayons : traite tous les triangles, pour des points de vues quelconques

### efficacité :

- ▶ fragmentation : matériel dédié, très (très) rapide,
- ▶ lancer de rayons : calculer toutes les intersections est très (très) long,  
structures accélératrices obligatoires... cf hiérarchie d'englobants, cours précédent)

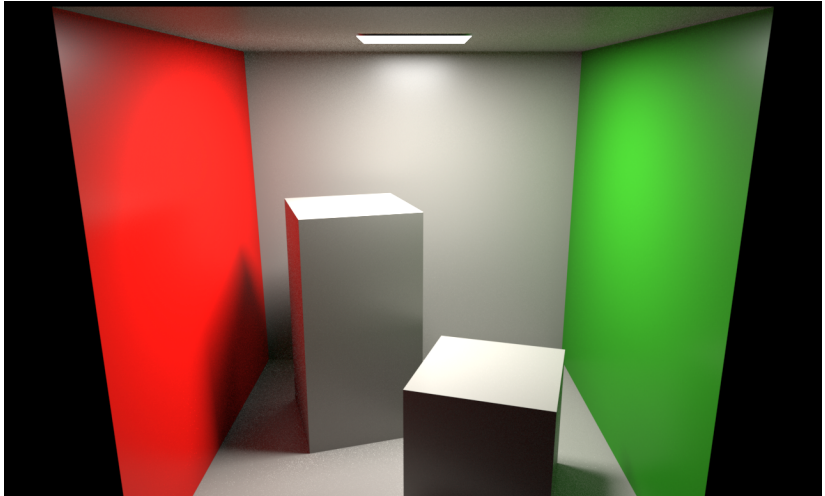


Introduction  
Lancer de rayons  
Intersections  
et alors ?

notions

j'aime pas les triangles...  
c'est trop lent...

mais on peut faire ça : éclairage global

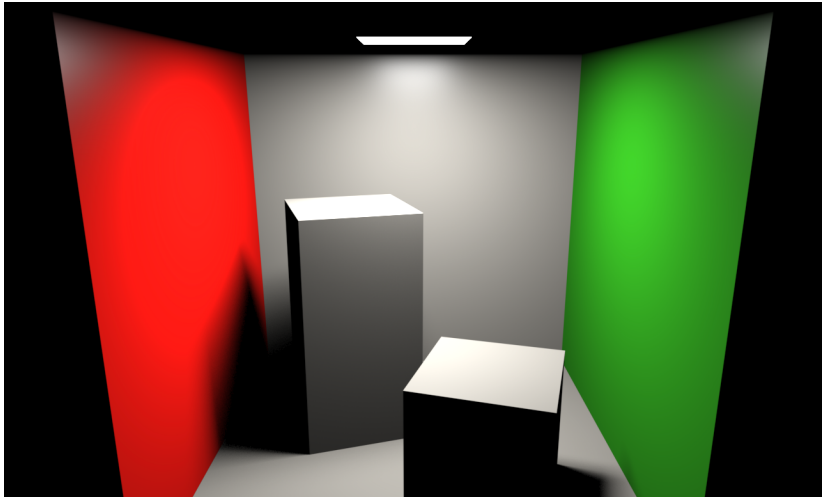


Introduction  
Lancer de rayons  
Intersections  
et alors ?

notions

j'aime pas les triangles...  
c'est trop lent...

mais on peut faire ça : éclairage local



Introduction  
Lancer de rayons  
Intersections  
et alors ?

notions

j'aime pas les triangles...  
c'est trop lent...

mais on peut faire ça : + parties diffuse et reflechissante



## et si c'est toujours trop lent ?

on peut utiliser une carte graphique...

- ▶ pour faire les calculs d'intersection,
- ▶ en parallèle, sur plusieurs milliers de rayons en même temps...

cf tuto\_raytrace\_fragment dans gKit2.

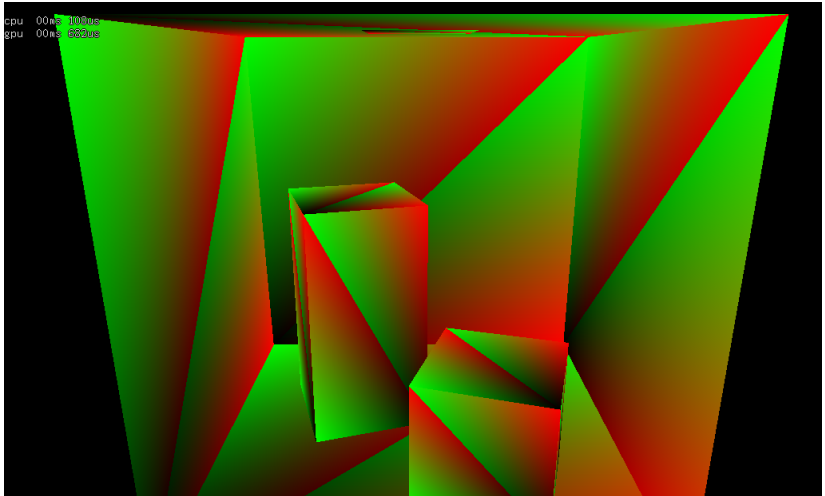
< 1ms pour 1M de rayons

Introduction  
Lancer de rayons  
Intersections  
et alors ?

notions  
j'aime pas les triangles...  
c'est trop lent...

tutos/M2/tuto\_raytrace\_fragment.cpp

```
cpu 00ms 100us  
gpu 00ms 680us
```



Introduction  
Lancer de rayons  
Intersections  
et alors ?

notions  
j'aime pas les triangles...  
c'est trop lent...

et si c'est toujours trop lent ?



cf [shadertoy](#)