M1-Images

Lancer de rayons

J.C. Iehl

March 24, 2016

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 ?

générer un rayon

transformations :

- model, view, projection, image...
- ou se trouve la camera dans le repère du monde ?
- ou se trouve le pixel x, y dans le repère 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 connait 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 $p \in plan(a, \vec{n})$.

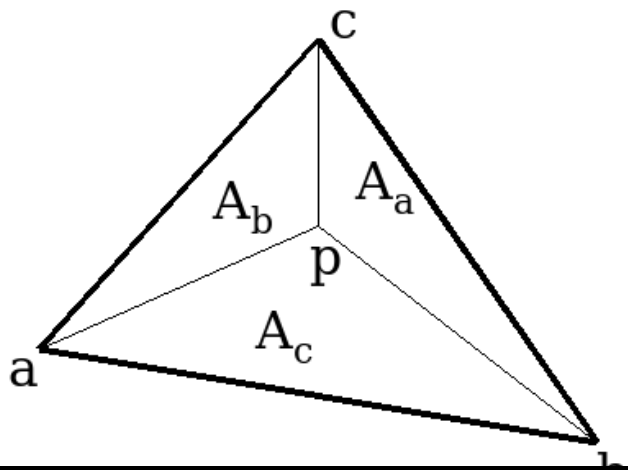
résultat :

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

intersections : rayon / triangle

rappels :

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



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 $p \in sphere(c, R)$
- $(p - c) \cdot (p - c) - R^2 = 0$

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 :

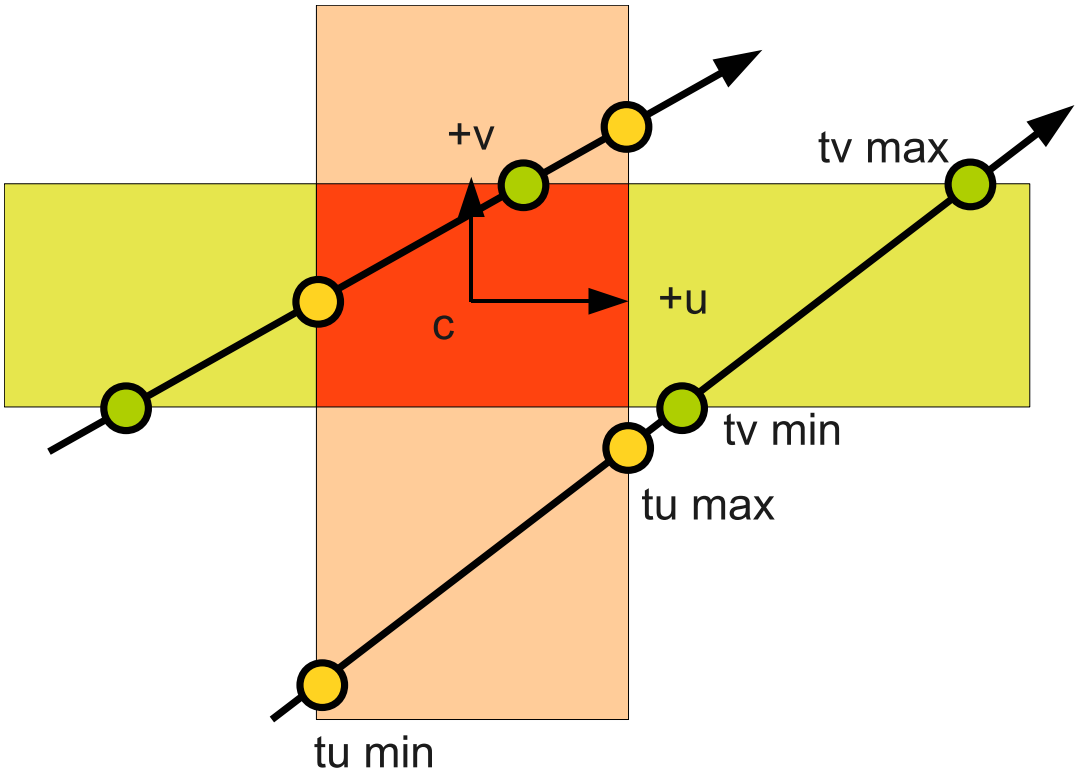
- $p(t) = o + t \cdot \vec{d}$,
- $p \in boite(c, \vec{u}, \vec{v}, \vec{w})$
- la boite est l'intersection de 3 paires de plans parallèles :
- $plan(c - \vec{u}, -\vec{u})$, $plan(c + \vec{u}, \vec{u})$,
- $plan(c - \vec{v}, -\vec{v})$, $plan(c + \vec{v}, \vec{v})$,
- $plan(c - \vec{w}, -\vec{w})$,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

résultat :

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

intersections : rayon / boite orientée

- l'intersection existe si l'intersection des intervalles n'est pas vide : $max(t^u_{min}, t^v_{min}, t^w_{min}) < min(t^u_{max}, t^v_{max}, t^w_{max})$



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...

intersection : plusieurs objets

attention :

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

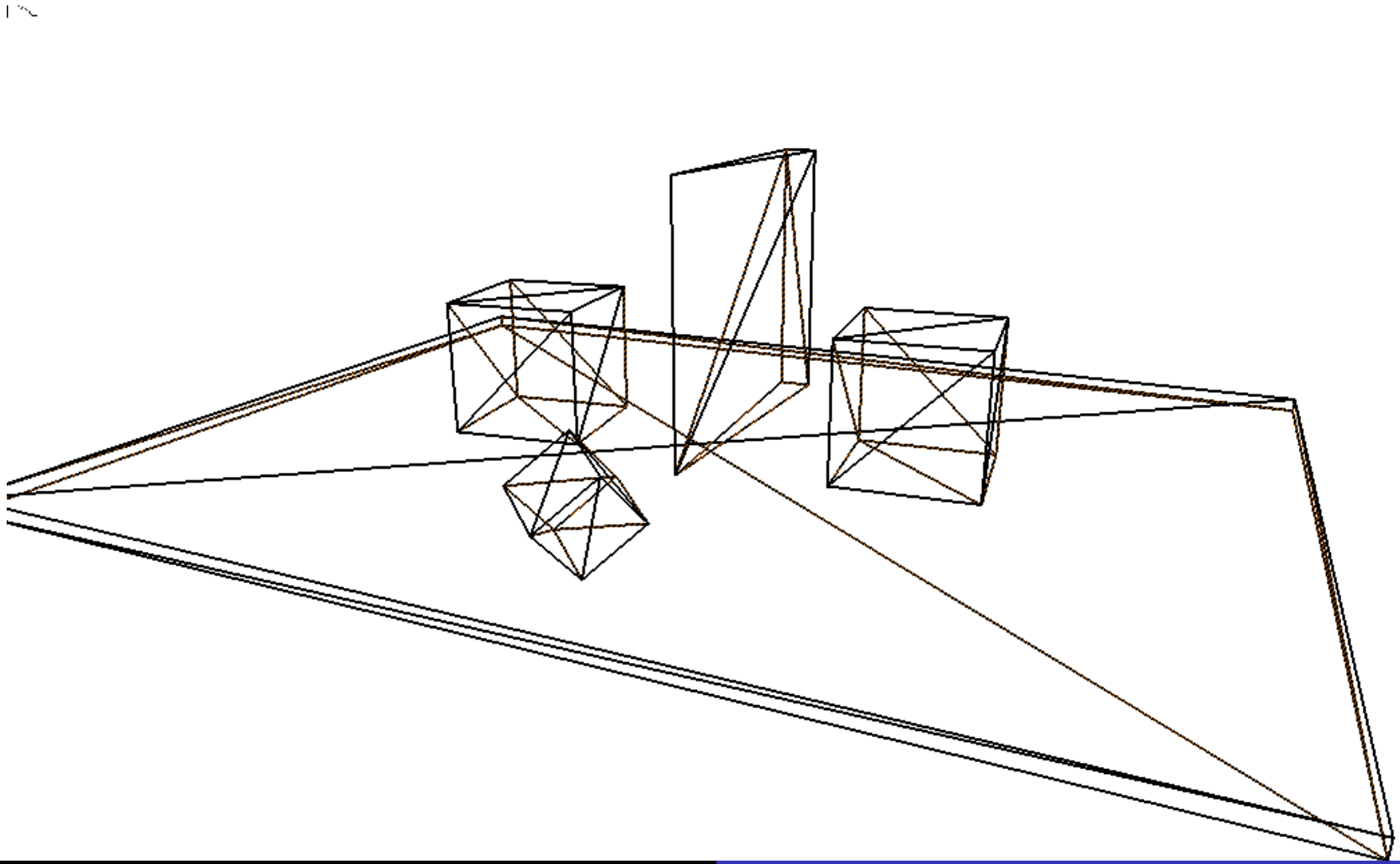
bilan

pour chaque pixel :

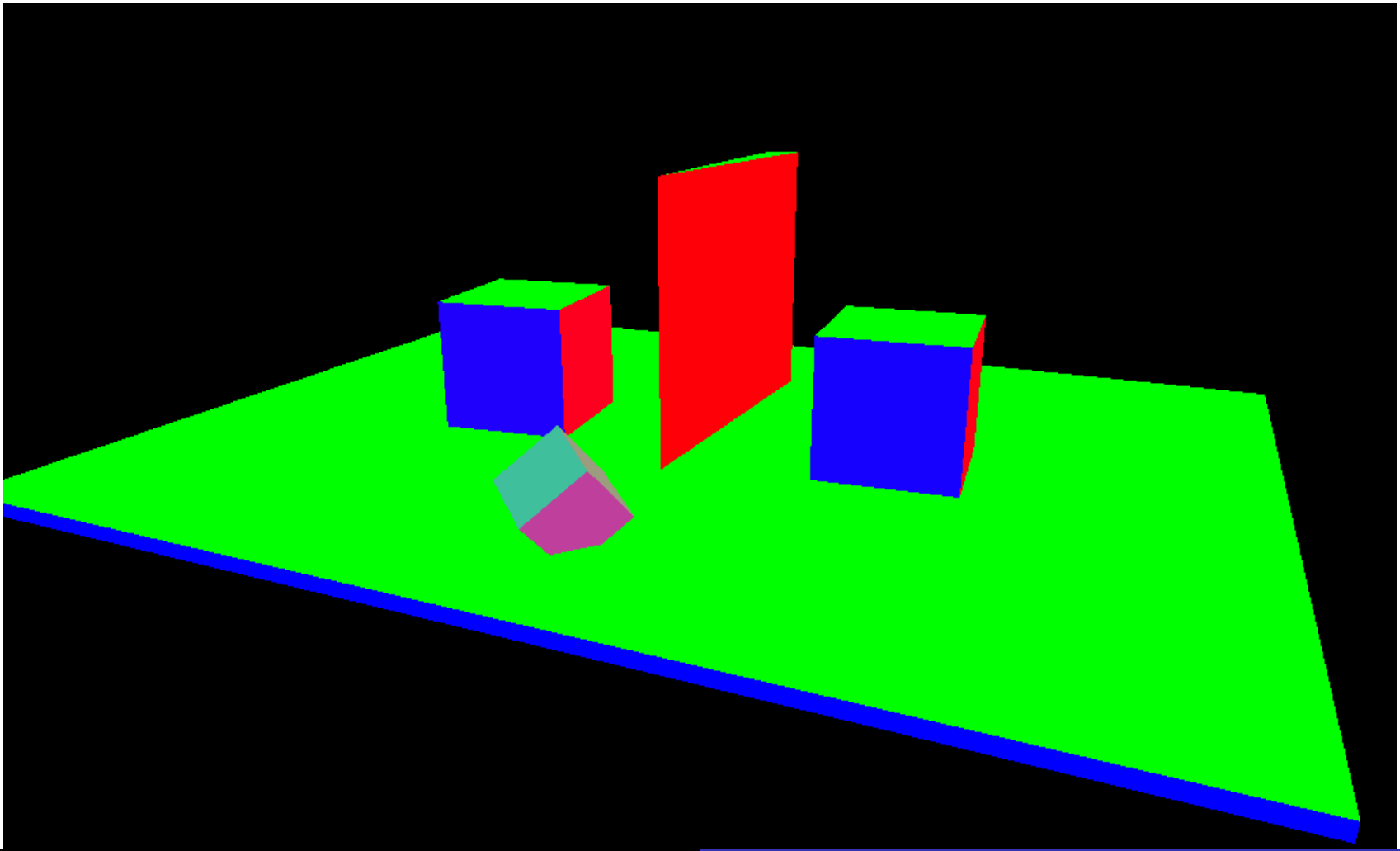
- générer un rayon,
- pour chaque objet :
- calculer l'intersection,
- garder l'objet le plus proche de la camera...

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

exemple : géométrie scène test



exemple : $abs(normal)$



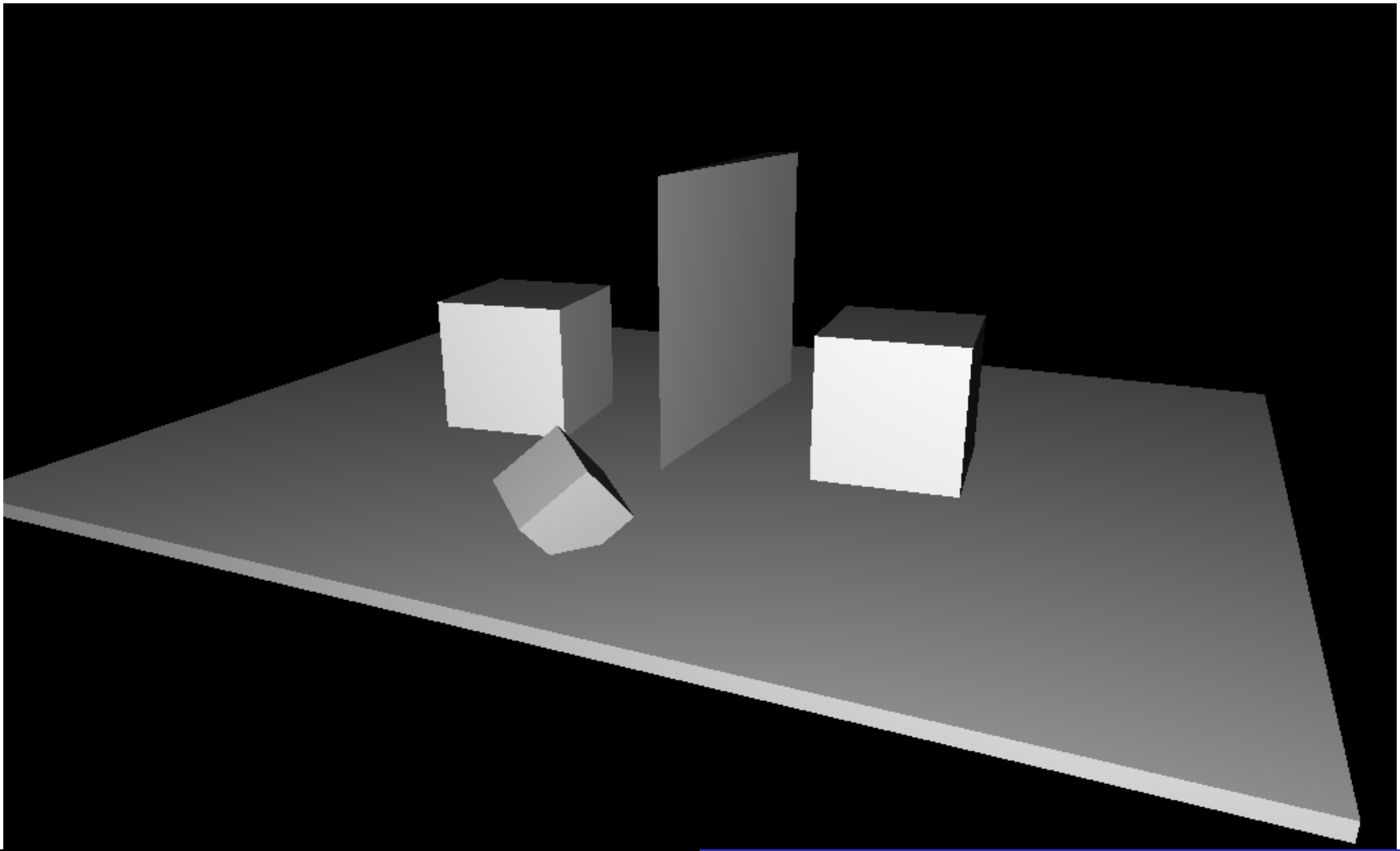
calculer une couleur...

qui dépend de l'orientation de l'objet :

- pourquoi ?
- réalité physique...

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

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 ?

orientation

??

- le point d'intersection, p, appartient à une surface...
- qui définit une normale $\overrightarrow{N(p)}$,
- quel est l'angle entre la normale au point d'intersection et la direction vers la camera, C ?

$$\cos\theta = \frac{\overrightarrow{N(p)} \cdot \overrightarrow{pC}}{||\overrightarrow{N(p)}|| \cdot ||\overrightarrow{pC}||}$$

orientation

en pratique :

- si les vecteurs sont de longueur 1 :
- float cos_theta= dot($normalize(\overrightarrow{N(p)}), normalize(\overrightarrow{pC})$)

exercice : et pour la position d'une lumière ?
exercice : et pour les ombres ?