Université Paris Diderot
Licence

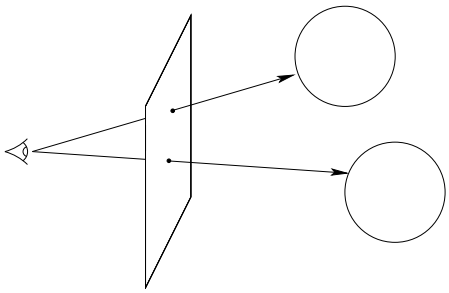
Programmation Fonctionnelle
Année 2010-2011

Projet n°1

Objets géométriques

1 Introduction

Un *raytraceur* est un programme de génération d'images réalistes, basé sur une idée assez simple. On suppose stockées en mémoire les représentations d'un ensemble d'objets disposés dans l'espace (sphères, plans, sources lumineuses, etc.). On choisit une position d'observation, ainsi que celle d'un rectangle représentant l'image à calculer, que l'on place entre l'observateur et les objets de la scène. La couleur de chaque pixel est alors déterminée par la lumière reçue par l'observateur lorsque son regard traverse ce pixel - en atteignant ou non un objet, et éventuellement plusieurs par reflexions successives. Cette lumière (couleur, intensité) est calculée grâce aux équations décrivant les propriétés optiques de la lumière et des objets.



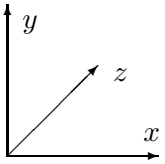
À partir d'une description des objets présents dans la scène à représenter, des différentes sources de lumières, de la position de l'observateur et de la position du rectangle représentant l'image, le programme se charge de calculer l'image correspondante.

Vous pouvez voir quelques exemples d'images de haute qualité obtenues avec un excellent raytraceur libre (`povray`) sur le site `http://hof.povray.org`. Le traceur réalisé dans ce projet sera plus modeste, mais permettra toutefois d'obtenir de bonnes images.

La première partie du travail consiste à définir les fonctions de base pour calculer les intersections entre un rayon et un objet.

2 Objets

Pour spécifier les positions des objets dans l'espace, on fixe le système de coordonnées suivant : l'axe des x va vers la droite, l'axe des y vers le haut et celui des z s'éloigne de l'observateur.



Dans la suite, pour décrire les rotations, on utilise les *vecteurs-rotations* : la rotation (r_x, r_y, r_z) est en fait une rotation autour de l'axe x d'angle r_x, suivie d'une rotation autour de l'axe y d'angle

r_y et enfin d'une rotation autour de l'axe z d'angle r_z. Pour chacune de ces rotations, on applique la règle de la *main gauche*, c'est-à-dire que, si le pouce de la main gauche indique la direction de l'axe, alors les autres doigts indiquent le sens de rotation. Autrement dit, il s'agit d'une rotation dans le sens anti-horaire (ou *trigonométrique*) par rapport à la direction de l'axe de rotation.

On définit trois sortes d'objets géométriques :

1. des **sphères**,
2. des **boîtes** à 6 côtés, dont les côtés opposés sont parallèles,
3. des **plans** d'une étendue infinie.

Chacun de ces objets est muni d'un attribut décrivant sa texture ; cet attribut sera pour l'instant en OCaml représenté par un paramètre de type permettant de représenter ces objets, et ne sera utilisé que dans la seconde partie du projet. La texture d'un objet contiendra des informations comme la couleur de sa surface, ou des coefficients décrivant la manière dont cette surface reflète la lumière.

Sur les différentes sortes d'objets, on peut appliquer les transformations suivantes :

1. **translation** par un vecteur ;
2. **rotation** autour du centre de l'objet[1], décrite par un vecteur-rotation ;
3. **dilatation** par un coefficient, qui multiplie les dimensions de l'objet par ce coefficient.

Les objectifs principaux de cette partie du projet sont :
- définir un type Ocaml pour représenter les objets géométriques,
- définir des fonctions pour les trois transformations décrites ci-dessus, et,
- définir la fonction qui calcule, pour un point de départ et une direction donnés, le premier objet atteint par un rayon lumineux issu de ce point et allant dans cette direction. Lorsque cet objet existe, cette fonction doit renvoyer plusieurs informations : la distance entre l'origine du rayon et l'objet touché, la direction de la surface au point d'impact du rayon, ainsi que la texture de l'objet. On ignore les intersections par l'intérieur des objets : par exemple, les plans sont supposés avoir un côté extérieur et un côté intérieur, ils laissent donc passer la lumière dans un sens, comme un miroir semi-réfléchissant.

3 Représentation des objets

La représentation suivante est choisie pour faciliter les calculs d'intersections décrits à la section 4.4.

Distance relative à une surface Les surfaces ayant un intérieur et un extérieur, on ne se contente pas de connaître la distance d'un point à une surface : on veut également savoir de quel côté de la surface le point se situe.

Pour cela, on utilise des distances *relatives* : une distance positive indique que le point est à l'intérieur de la surface, alors qu'une distance négative indique qu'il est à l'extérieur.

Conventions de représentation

1. Une sphère est décrite par son centre C et son rayon $r > 0$.
2. Un plan est décrit par un vecteur unitaire normal (un vecteur de longueur 1, orthogonal à la surface du plan et pointant vers l'extérieur) et par la distance relative d entre le plan et l'origine (c'est-à-dire la distance entre O et la projection orthogonale de O sur le plan, avec $d > 0$ ssi l'origine est du côté intérieur).
3. Pour une boîte, on choisit trois faces qui ont un sommet commun, pour chacune de ces faces ($i = 1$, 2 ou 3), on calcule :

[1] Pour un plan infini, on considère que le "centre" est la projection orthogonale de O sur ce plan.

- Le vecteur unitaire normal vers l'extérieur de la boîte $\vec{N}_i$,
- Le centre de la face C_{f_i},
- La distance relative, par rapport à l'origine, du plan qui contient la face d_{f_i},
- Le centre de la face opposée C_{oi},
- La distance relative, par rapport à l'origine, du plan qui contient la face opposée d_{oi},
- La moitié de la distance l_i entre la face et la face opposée.

4 Géométrie

4.1 Généralités

La *norme* d'un vecteur $\vec{V} = (V_x, V_y, V_z)$, c'est-à-dire sa longueur, est

$$\|\vec{V}\| = \sqrt{V_x \times V_x + V_y \times V_y + V_z \times V_z}$$

Le *produit scalaire* de deux vecteurs est défini par :

$$\vec{V} \cdot \vec{V'} = (V_x, V_y, V_z) \cdot (V'_x, V'_y, V'_z) = V_x \times V'_x + V_y \times V'_y + V_z \times V'_z$$

On a donc aussi $\vec{V} \cdot \vec{V} = \|\vec{V}\|^2$, soit $\|\vec{V}\| = \sqrt{\vec{V} \cdot \vec{V}}$. De plus, $\vec{V} \cdot \vec{V'} = \|\vec{V}\| \times \|\vec{V'}\| \times \cos(\alpha)$ où α est l'angle formé par les deux vecteurs.

L'origine du repère est notée $O = (0, 0, 0)$. Dans la suite, on confondra le point A et le vecteur $\vec{OA}$ qui ont les mêmes coordonnées. On note $|AB|$ la *longueur* du segment AB, définie par $\|\vec{AB}\|$.

Lorsqu'un rayon a pour origine un point S et pour direction un vecteur $\vec{D}$ (par convention, ce vecteur sera toujours unitaire), le trajet suivi par le rayon est donné par $R(t) = S + t\vec{D}$ pour $t \geq 0$.

4.2 Distance par rapport à l'origine d'un plan

Si $\vec{P}$ est un point quelconque sur un plan, et $\vec{N}$ le vecteur unitaire normale vers l'extérieur du plan, alors la distance relative par rapport à l'origine du plan est simplement $\vec{P} \cdot \vec{N}$.

4.3 Rotations

Rotation d'un vecteur La rotation d'un plan se limite à la rotation de son vecteur normal. La rotation d'angle α d'un vecteur $\vec{V} = (V_x, V_y, V_z)$ autour d'un axe donne un vecteur $\vec{W} = (W_x, W_y, W_z)$ obtenu en effectuant le produit de matrices

$$M \times \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix}$$

où M est l'une des matrices correspondante à l'axe : M_x, M_y ou M_z :

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad M_y = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \quad M_z = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Par exemple, la rotation décrite par le vecteur-rotation $(\alpha, 0, 0)$, appliquée à

Rotation autour d'un point Pour la rotation des boîtes, il faut effectuer une rotation des vecteurs normaux aux plans des faces (*c.f.* section 4.3). Il faut aussi effectuer une rotation des centres des faces autour du centre de la boîte

Intersection avec une boîte Le principe est de considérer successivement les trois couples face/face opposée. Dès qu'on trouve une intersection, on s'arrête. Ici, on prend par exemple la face 1.

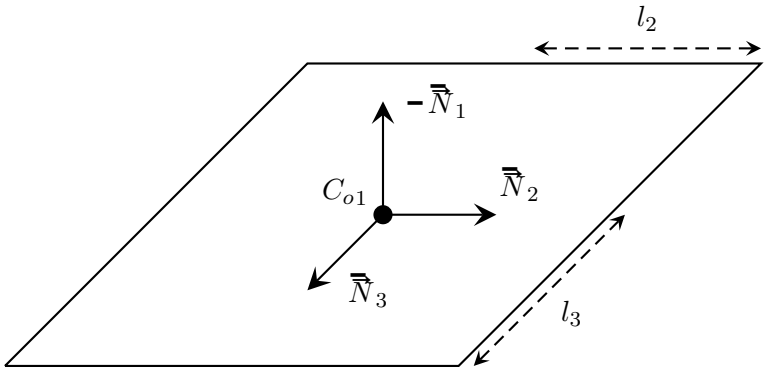
- Si $\overline{\overline{D}} \cdot \overline{\overline{N}}_1 > 0$, l'intersection a lieu avec le plan de la face opposée.
- Si $\overline{\overline{D}} \cdot \overline{\overline{N}}_1 = 0$, le rayon est parallèle aux faces : il n'y a pas d'intersection.
- Si $\overline{\overline{D}} \cdot \overline{\overline{N}}_1 < 0$, l'intersection a lieu avec le plan de la face considérée (ici, la face 1).

Supposons, par exemple que $\overline{\overline{D}} \cdot \overline{\overline{N}}_1 > 0$. On calcule alors l'intersection I (si elle existe) du rayon avec le plan contenant la face opposée (décrit par le vecteur $-\overline{\overline{N}}_1$ et la distance d_{o1}).

Si cette intersection existe, il faut encore vérifier qu'elle se produit bien sur la face de la boîte et pas à côté. Pour cela, il faut que la distance entre I et la droite passant par C_{o1} de direction $\overline{\overline{N}}_3$ soit inférieure à l_2 et que la distance entre I et la droite passant par C_{o1} de direction $\overline{\overline{N}}_2$ soit inférieure à l_3.

Ainsi, I est une intersection si $\left|\overrightarrow{C_{o1}I} \cdot \overline{\overline{N}}_2\right| \leq l_2$ et $\left|\overrightarrow{C_{o1}I} \cdot \overline{\overline{N}}_3\right| \leq l_3$

Quand il n'y a pas d'intersection, on recommence les calculs pour le couple basé sur la face 2 puis, le cas échéant, sur la face 3.



5 Travail à réaliser

Pour cette première partie vous devez définir le type nécessaire pour la représentations des objets suivant les explications données en section 3, et donner les fonctions permettant de transformer un objet (translation, rotation, ou dilatation), et permettant de calculer les intersections d'un rayon avec une liste d'objets. Votre code doit réaliser les types et fonctions suivants. Vous pouvez évidemment utiliser des types et des fonctions auxiliaires.

```
type vecteur
(* type représentant les vecteurs dans l'espace tri-dimensionnel *)

type rotation
(* type représentant les rotations dans l'espace tri-dimensionel *)

type 'texture eval_objet
(* type représentant les trois formes d'objets géometriques dans une
 représentation adaptée aux opérations géometriques. Le type est
 paramétré par un type de texture qui décrit les propriétés de
 surface d'un objet. *)

val translate: vecteur -> 'texture eval_objet -> 'texture eval_objet
(* translation d'un objet *)
```

```
val rotate: rotation -> 'texture eval_objet -> 'texture eval_objet
(* rotation d'un objet *)

val dilate: float -> 'texture eval_objet -> 'texture eval_objet
(* dilatation d'un objet *)

type rayon
(* type représentant les rayons *)

val intersecte: rayon -> 'texture trace_objet list -> bool
(* determine si un rayon intersecte un objet de la liste *)

exception Pas_d_intersection
val intersection: rayon -> 'texture trace_objet list -> float * vecteur * 'texture
(*
 renvoie, pour le premier objet touché par le rayon, la distance entre
 l'origine du rayon et l'objet, la direction de sa surface au point
 d'intersection et sa texture. Lève l'exception Pas_d_intersection si
 aucun objet n'est touché.
*)
```

Le projet est à faire en binôme.

Vous écrirez votre code dans un fichier OCaml à interpréter par l'interpréteur OCaml (la compilation n'est pas demandée pour ce premier rapport).

Il est indispensable que votre code soit *bien* documenté. La documentation peut se faire entièrement sous forme de commentaires OCaml mais vous pouvez aussi fournir un fichier texte supplémentaire.

Vous devrez également fournir un jeu de tests pour la fonction `intersection`. Pour chacun des tests, expliquez brièvement le test, donnez le résultat attendu et le résultat obtenu avec votre programme.

Votre programme et votre jeu de tests pour cette première partie du projet devront être rendus par email à votre chargé de TP. Les adresses sont

Groupe 1 L3 Informatique	`kerneis@pps.jussieu.fr`
Groupe 2 L3 Informatique	`renaud@pps.jussieu.fr`
Groupe 3 L3 Informatique	`Tahina.Ramananandro@normalesup.org`
Groupe L2 Math-Info	`Mehdi.Dogguy@pps.jussieu.fr`
Groupe M1 Linguistique	`Vincent.Padovani@pps.jussieu.fr`

La date limite pour rendre votre projet est le

Vendredi, 19 novembre, 18h00

À partir de cette date, un corrigé sera mis à disposition sur la page web du cours (`http://www.pps.jussieu.fr/~treinen/teaching/pf5/`), par conséquent, aucun projet ne pourra être accepté après cette date limite.