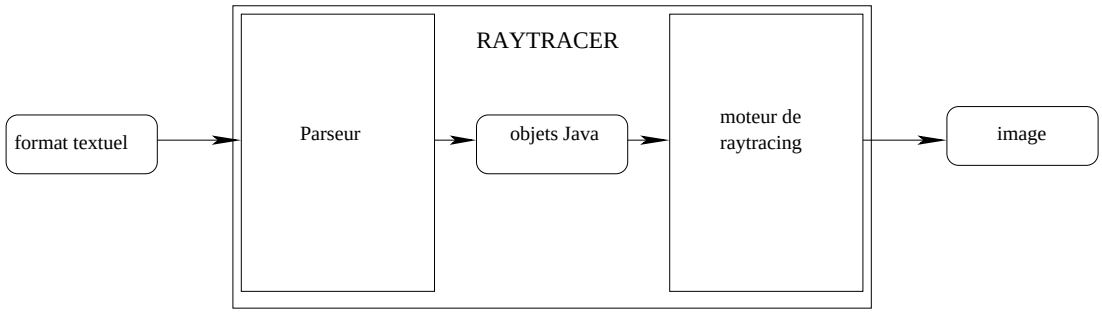
Raytraceur (1) - Objets géométriques, rayons et intersections

1 Introduction

Le projet consiste à programmer un raytraceur (outil de rendu graphique). Le programme final prend en entrée une description textuelle de la scène à représenter (*scénario*), et construit l'image correspondante. Un scénario est constitué des sources de lumières, de la luminosité ambiante, de la position de la caméra et d'un ensemble d'objets géométriques. Nous nous restreignons à des figures simples : plans, sphères et boîtes.



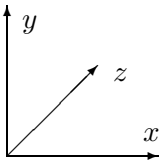
La première partie du travail consiste à implanter les classes utiles pour représenter les objets géométriques dans le cadre du *raytracing*. L'objectif principal est de donner une fonction qui calcule, pour un point de départ et une direction, le premier objet qui est touché.

La section 2 précise les objets géométriques considérés et les manipulations de base. La section 3 donne une représentation des objets plus adaptée au *raytracing*. La section 4 précise les classes que vous devez implanter. Enfin l'annexe A rappelle les notions de géométrie nécessaires.

2 Objets géométriques manipulés

2.1 notations

Pour repérer les 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, suivi d'une rotation autour de l'axe y d'angle r_y et enfin d'un 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 indique le sens de rotation. Autrement dit : si on regarde au long de l'axe la rotation va contre le sens de l'horloge.

Pour calculer la nouvelle description d'un objet, on a parfois besoin de faire des rotations de vecteur dans l'espace. La rotation d'angle α d'un vecteur autour d'un axe se calcule en faisant le produit à gauche avec la matrice 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, si $V = (v_x, v_y, v_z)$, la rotation d'angle α autour de l'axe x appliquée à V donne le vecteur : $(v_x, v_y \cos(\alpha) - v_z \sin(\alpha), v_y \sin(\alpha) + v_z \cos(\alpha))$.

2.2 les objets géométriques

Les trois formes d'objets géométriques que l'on va utiliser sont :

1. la **sphère** décrite par les coordonnées de son centre et par son rayon ;
2. la **boîte** décrite par les coordonnées de son centre et par ses trois dimensions largeur (initialement suivant l'axe x), hauteur (initialement suivant l'axe y) et profondeur (initialement suivant l'axe z) et par une *matrice de rotation* (qui décrit la rotation que l'on applique à la boîte par rapport à son centre) ;
3. le **plan** qui est décrit par une matrice de rotation r et une distance relative d. En fait, cet objet est un demi-espace solide : initialement, c'est l'ensemble des points tels que $y \leq 0$ auquel on fait subir la rotation r et translaté pour que la distance du demi-espace à l'origine soit d (avec la convention que d est positif si l'origine est contenue dans le demi-espace et négatif sinon) .

Chacun de ces objets a également un attribut pour décrire sa texture. Comme la texture ne sera utilisée réellement que dans la seconde partie, nous ne la décrivons pas içi.

2.3 manipulations sur les objets

Sur les différents types d'objets, on veut pouvoir appliquer les transformations suivantes :

1. la **translation** par un vecteur ;
2. la **rotation** par un vecteur-rotation ;
3. le **dilatation** par un coefficient, qui multiplie les dimensions de l'objet par ce coefficient.

L'objectif principal de cette partie du projet est de donner une fonction qui calcule, pour un point de départ et une direction, le premier objet qui est touché. On aura en fait besoin de savoir la distance entre l'origine du rayon et l'objet touché, la direction de la surface à l'intersection, et la texture de l'objet. *On ignore les intersections par l'intérieur des objets (cela peut se produire suite à des erreurs de précisions).* Les bases nécessaires au calcul d'intersection sont rappelées dans l'annexe A.

3 Optimisation pour le *raytracing*

La représentation précédente des objets géométriques, calquées sur leur représentation intuitive, permet d'exécuter facilement les opérations géométriques décrites au-dessus. Cependant, cette représentation n'est pas forcement la bonne pour faire du raytracing. Pour une image de 600 par 400 pixels, par exemple, on a besoin de calculer 240,000 fois l'intersection d'un rayon avec le même ensemble d'objets. On a donc tout intérêt à utiliser pour le raytracing une autre représentation qui évite de refaire plusieurs fois les mêmes calculs, en les stockant une fois pour toute dans la représentation. On propose les représentations suivantes :

- La sphère est décrite par son centre C et son rayon $r > 0$.
- Le plan est décrit par son vecteur unitaire normal (vers l'extérieur) et par la distance relative d à l'origine ($d > 0$ ssi l'origine est à l'intérieur).
- Pour la boîte, on choisit trois faces qui ont un sommet commun, pour chacune de ces faces ($i = 1$, 2 ou 3), on calcule :
  - Le vecteur unitaire normal vers l'extérieur de la boîte $\overrightarrow{N}_i$
  - Le centre de la face C_{f_i}
  - La distance relative, par rapport à l'origine, du plan qui contient la face (comme pour le plan) 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 Travail à réaliser

4.1 travail minimal

Pour cette première partie, vous devez définir les classes nécessaires pour les deux représentations des objets et donner les méthodes permettant de calculer les intersections de rayons avec une liste d'objets. Votre code doit implanter les classes et méthodes suivantes. Vous pouvez évidemment utiliser des classes et méthodes auxiliaires.

- la classe **vecteur** représente les vecteurs dans l'espace tri-dimensionnel.
- la classe **rotation** représente une rotation sous la forme d'une matrice. On peut construire une rotation à partir d'un vecteur-rotation.
- *l'interface* **texture** est une description abstraite d'une texture.
- *l'interface ou classe abstraite* **objet_geo** donne le comportement des objets géometriques que l'on manipule (en représentation intuitive). Les opérations disponibles sont
  - la translation à partir d'un vecteur,
  - la rotation à partir d'une matrice rotation,
  - la dilatation à partir d'un coefficient flottant,
  - la transformation en l'**objet_ray** correspondant.
- les classes **sphere_geo**, **plan_geo** et **boite_geo** *implantent* l'interface **objet_geo**.
- *l'interface ou classe abstraite* **objet_ray** donne le comportement des objets géometriques en représentation adaptée au raytracer.
- les classes **sphere_ray**, **plan_ray** et **boite_ray** *implantent* l'interface **objet_ray**.
- la classe **rayon** représente les rayons lumineux (une source et une direction). La fonction principale prend un tableau de **objet_ray** et un rayon et retourne soit *l'exception* **No_intersection**, soit pour le premier objet touché la distance entre l'origine du rayon et l'objet, la direction de sa surface au point d'intersection et sa texture.

4.2 travail optionnel

Cette première partie est très guidée, aussi votre latitude est assez faible. Vous pouvez modifier la hiérarchie de classes proposée, à condition de pouvoir garder toutes les fonctionnalités. Cependant il est conseillé de s'en tenir aux indications. Vous pouvez aussi envisager d'autres objets, mais méfiez vous du calcul d'intersection.

4.3 consignes

1. Cette partie est à **rendre au plus tard le vendredi 3 mars**.
2. Il est indispensable que votre code soit *documenté*. La documentation peut se faire entièrement sous forme de commentaires Java mais vous pouvez aussi fournir un fichier texte supplémentaire.
3. Vous devez également faire un jeu de tests pour la méthode d'intersection. Pour chacun des tests expliquez brièvement le test, donnez le résultat attendu et le résultat obtenu avec votre programme. Les tests devraient vous donner une confiance raisonnable dans votre programme, pour l'intersection et les manipulations géométriques. Ces opérations sont nécessaires à la suite du projet, il est donc *indispensable de s'assurer que leur implantation est correcte.*

A Géométrie

Le *produit scalaire* de deux vecteurs est défini par : $\overrightarrow{V} \cdot \overrightarrow{V'} = (V_x, V_y, V_z) \cdot (V'_x, V'_y, V'_z) = V_x V'_x + V_y V'_y + V_z V'_z$. De plus, $\overrightarrow{V} \cdot \overrightarrow{V'} = |\overrightarrow{V}||\overrightarrow{V'}| \cos(\alpha)$ où α est l'angle formé par les deux vecteurs. La *longueur* d'un vecteur $|\overrightarrow{V}|$ est donc $\sqrt{\overrightarrow{V} \cdot \overrightarrow{V}}$.

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

On suppose que le rayon a pour origine le point S et pour direction le vecteur unitaire $\overrightarrow{D}$. Le trajet suivi par le rayon est donné par $R(t) = S + t\overrightarrow{D}$ pour $t \geq 0$.

A.1 Intersection avec un plan

On suppose le plan décrit par la distance relative d et le vecteur unitaire normal $\overrightarrow{P}$. Un point Q est dans le plan (demi-espace) ssi $\overrightarrow{P} \cdot Q \leq d$,

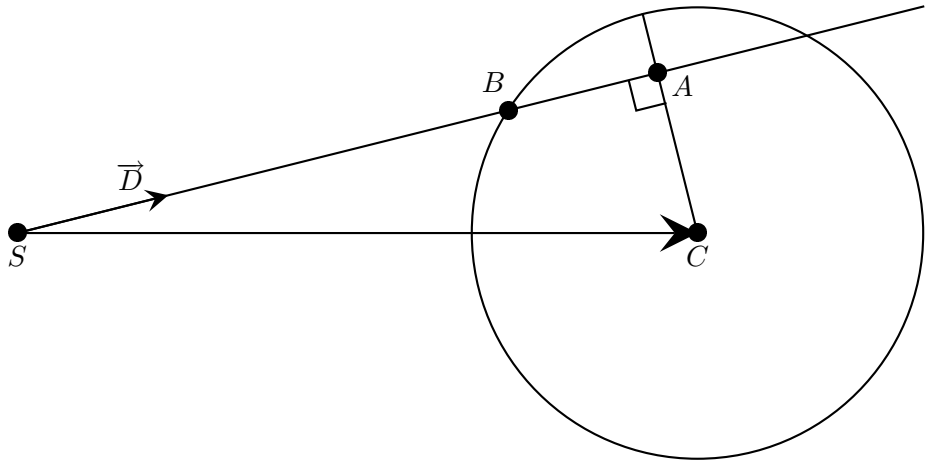
- Si $\overrightarrow{P} \cdot S \leq d$, l'origine du rayon est à l'intérieur du demi-espace, pas d'intersection.
- Si $\overrightarrow{P} \cdot \overrightarrow{D} = 0$, la direction est parallèle au plan, pas d'intersection.
- Sinon, on cherche t tel que $\overrightarrow{P} \cdot R(t) = d$: l'intersection se produit pour

$$t_i = \frac{(d - \overrightarrow{P} \cdot S)}{\overrightarrow{P} \cdot \overrightarrow{D}}$$

Si $t_i \leq 0$, pas d'intersection (le rayon s'éloigne du plan), sinon intersection au point $R(t_i)$

A.2 Intersection avec une sphère de centre C et de rayon r

- Si $\overrightarrow{SC} \cdot \overrightarrow{D} \leq 0$, le rayon s'éloigne de la sphère ; sinon on calcule le point A de la trajectoire le plus près du centre $|SA| = \overrightarrow{SC} \cdot \overrightarrow{D}$ (attention, il faut que $\overrightarrow{D}$ soit unitaire).
- On calcule $|AC|^2 = |SC|^2 - |SA|^2$.
- Si $|AC| \geq r$ alors il n'y a pas d'intersection. Sinon, on calcule $|AB| = \sqrt{|BC|^2 - |AC|^2} = \sqrt{r^2 - |AC|^2}$.
- Finalement, $|SB| = |SA| - |AB|$ et, comme $\overrightarrow{D}$ est unitaire, on a $t_i = |SB|$; il y donc intersection au point $B = R(t_i)$. Le vecteur normal unitaire à la surface au point d'intersection est alors $\overrightarrow{CB}/|CB|$.



A.3 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 $\overrightarrow{D} \cdot \overrightarrow{N}_1 > 0$, l'intersection ne peut être qu'avec la face opposée.
- Si $\overrightarrow{D} \cdot \overrightarrow{N}_1 = 0$, rayon parallèle aux faces, pas d'intersection
- Si $\overrightarrow{D} \cdot \overrightarrow{N}_1 < 0$, l'intersection ne peut être qu'avec la face 1.

Supposons, par exemple que $\overrightarrow{D} \cdot \overrightarrow{N}_1 > 0$. On calcule alors l'intersection I (si elle existe) du rayon avec le plan contenant la face opposée (décrit par le vecteur $-\overrightarrow{N}_1$ 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 $\overrightarrow{N}_3$ soit inférieure à l_2 et que la distance entre I et la droite passant par C_{o1} de direction $\overrightarrow{N}_2$ soit inférieure à l_3.

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