Projet de Java

Licence 3 – Informatique

Rémi Forax, Pierre Peterlongo, Benoit Olivieri
(forax@univ-mlv.fr, pierre.peterlongo@univ-mlv.fr, olivieri@univ-mlv.fr)

Description du projet

Le programme RayOfLight est un petit programme de Raytracing (lancé de rayon) permettant facilement de faire de la synthèse d'image simple.
Il existe déjà beaucoup de *raytracers* qui marchent très bien et qui font beaucoup de choses, le but n'est pas d'implanter toutes les fonctionnalités d'un *raytracer* classique mais un sous-ensemble cohérent et qui fonctionne.

Principe d'un Raytracer

Une scène correspond à un ensemble d'objets que l'on veut afficher à l'écran.
Pour chaque point de l'écran, on lance un rayon perpendiculaire (presque voir camera plus bas) à l'écran dans le but de toucher un objet de la scène.
Parmi les points d'intersections d'un rayon avec les différents objets de la scène,
on prendra le point le plus proche de l'écran pour simuler la profondeur.

La couleur du point de l'écran sera en première approximation la couleur de l'objet atteind par le rayon. En fait, cela ne dépend pas uniquement de la couleur de l'objet mais dépend aussi des lumières placées dans la scène, de la reflexion du rayon sur l'objet, etc.
On appele modèle d'illumination la formule qui permet de calculer la couleur d'un rayon à un point d'un objet.

Qu'est -ce qu'un rayon ?

Un rayon est caractérisé par un point (noté A par la suite) et un vecteur (t). Le point correspond à l'endroit de l'écran où d'un objet dont part le rayon. Le vecteur correspond à la direction du rayon.

Référentiel local et universel

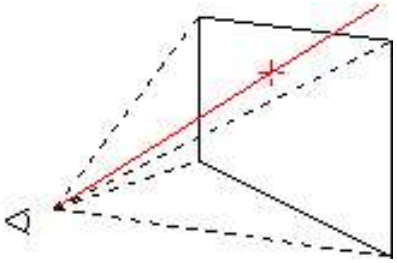
Lorsque l'on lance un rayon celui-ci a des coordonnées dans le référentiel de la scène globale. Il n'est pas très facile de calculer l'intersection d'un rayon avec un objet dans n'importe que référentiel. Il est plus facile de se placer dans le référentiel de l'objet car les équations sont plus simples.

Pour transformer un rayon d'un référentiel vers un autre, on utilise une matrice 4x4 appelée matrice en coordonnées homogènes qui permet d'exprimer les rotations, translations et agrandissements. (cf références sur les bases mathématiques)
Attention, exprimer une transformation sur un objet revient à exprimer la transformation inverse sur le rayon. De plus, pour simplifier les calculs, il est plus aisé d'utiliser, une fois tranformés pour être exprimés dans le référentiel de l'objet, des rayons unitaires[1].

1 Les rayons dont le vecteur $\vec{u}$ a sa norme (longueur) qui est égale à 1

La camera

La camera modélise l'écran ainsi que l'oeil de l'observateur. Les rayons sont lancés à partir de l'oeil et détermine la couleur du point de l'écran qui intersecte le rayon.



Une caméra est construite en indiquant la taille de l'écran en pixel (width x height) ainsi que la distance entre l'écran et l'oeil de l'observateur.

Les objets présent dans la scène

Il est demandé d'implanter uniquement deux types objets les sphères et les plans.

Soit une sphère placée en (0,0,0) et de rayon 1, M le point de contact entre le rayon et la sphère, $\vec{u}$ le vecteur du rayon et A le point sur l'écran.
Pour trouver la distance d'un rayon à la sphère :

- le rayon doit frapper la sphère $A\vec{M}=t\vec{u}$, avec $t>0$
  donc $\{x_M=t\,u_x+x_A, y_M=t\,u_y+y_A, z_M=t\,u_z+z_A\}$
- le point de contact est sur la sphère $x^2+y^2+z^2=1$

On cherche t qui exprime la distance entre un rayon et l'objet, en injectant la première équation dans la seconde équation, on obtient
$t^2({u_x}^2+{u_y}^2+{u_z}^2)+t(2(u_x x_A+u_y y_A+u_z z_A))+{x_A}^2+{y_A}^2+{z_A}^2-1=0$
C'est une équation du second degrés en t que l'on résoud facilement avec un discriminant.

Si le discriminant est négatif, le rayon n'intersecte pas la sphère
Si le discriminant est positif, il faut prendre la racine la plus petite, c-à-d le t le plus petit donc le point de contact avec la sphère le plus proche du rayon.

Pour un plan passant par l'origine (0,0,0) perpendiculaire à l'axe des z.
Pour trouver la distance d'un rayon au plan :

le rayon doit frapper la sphère $A\vec{M}=t\vec{u}$, avec $t>0$
donc $\{x_M=t\,u_x+x_A, y_M=t\,u_y+y_A, z_M=t\,u_z+z_A\}$
comme le rayon a un point de contact avec le plan $z_M=0$.

$$t=-\frac{z_A}{u_z}$$

si u_z est nul, il n'y a pas de point de contact car le rayon est parallèle au plan.
si t est négatif, le plan est derrière le rayon.

Les modèles d'illumination

Il est demandé d'implanter 4 modèles d'illumination :

- le modèle ambiant
- le modèle de Lambert
- le modèle de Phong
- le modèle de Whitted (modèle pas complet, juste la reflexion)

Pour que le problème reste simple, les modèles de Lambert, Phong et Whitted ne tiendront pas compte du fait qu'un objet peut être placé entre la lumière et l'objet frappé par un rayon.

Le modèle ambiant

Chaque objet de la scène possède une couleur et lorsque le rayon vient frapper cet objet, la couleur du rayon I (en fait, chaque composante RGB) est celle de l'objet frappé multipliée par un facteur de reflexion k_a .

$$I = k_a I_a$$

I_a correspond à la couleur de l'objet et k_a au facteur de luminosité ambiant de l'objet.

Les lumières de la scène ne sont pas prises en compte dans ce modèle.

Le modèle de Lambert

Ce modèle prend en compte la reflexion diffuse des lumières de la scène sur les objets.
La couleur en un point se calcule en additionnant à la lumière ambiante de l'objet
la contribution de chaque lumière de la scène.

$$I = k_a I_a + \sum_{i=1}^{n} I_s k_d \cos(\theta)$$

Cette contribution varie en fonction de l'intensité de la lumière I_l sur chaque composante RGB (une lumière n'est pas forcément blanche) et du cosinus de l'angle θ , angle entre la normale $\vec{N}$ à la surface de l'objet au point d'intersection avec le rayon et le vecteur $\vec{L}_I$ issue de la position de la lumière.

$\vec{L}_I = \vec{MS}$ et $\vec{MS}_x = S_x - M_x, \vec{MS}_y = S_y - M_y, \vec{MS}_z = S_z - M_z$

S_x , S_y et S_z correspondent à la position de la lumière plongée dans le référentiel de l'objet et M_x , M_y et M_z à la position de l'impact du rayon sur l'objet toujours dans le référentiel de l'objet.

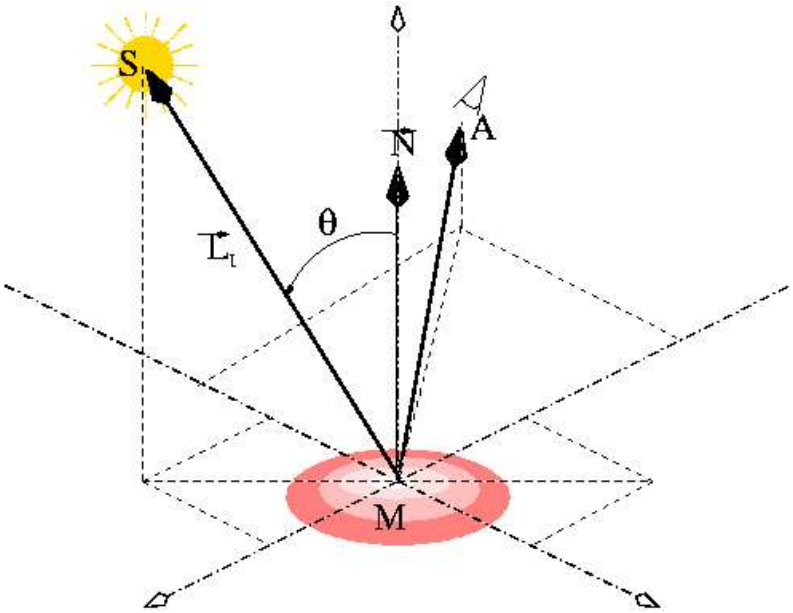
$$\cos(\theta) = \vec{L}_I \cdot \vec{N} = \vec{MS} \cdot N = \vec{MS}_x * \vec{N}_x + \vec{MS}_y * \vec{N}_y + \vec{MS}_z * \vec{N}_z$$ [2],

Le cosinus n'a de sens que pour des valeurs positives. Pour les valeurs négatives, l'apport de lumière est nul (l'objet est entre la source et le point d'impact).

k_d correspond au facteur associé à la reflexion diffuse d'un objet.

---

2 On rappelle que le produit scalaire de $\vec{u}$ et $\vec{v}$ est égal à $u_x v_x + u_y v_y + u_y v_y$



A correspond à la position du rayon dans le référentiel de l'objet.
M correspond au point d'impact du rayon sur l'objet.
S correspond à la position d'une lumière dans le référentiel de l'objet.

Le modèle de Phong

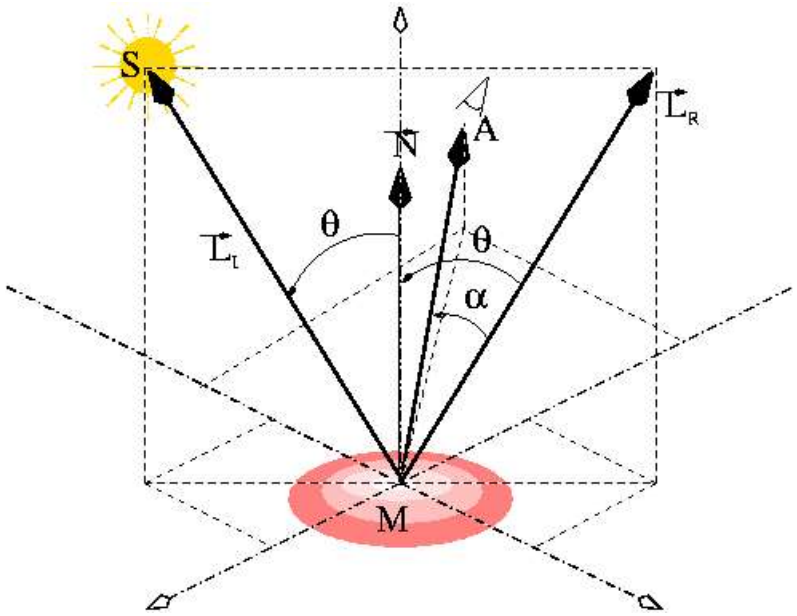
Ce modèle prend en compte en plus la partie spéculaire[3] de la lumière.

$$I = k_a I_a + \sum_{i=1}^{n} I_l (k_d \cos(\theta) + k_s ((\cos(\alpha))^m))$$

Pour cela il faut calculer le cosinus de l'angle alpha : $\cos(\alpha) = \vec{L_R} \cdot \vec{MA}$

Le vecteur de la lumière réfléchi $\vec{L_R}$ est calculé par le fait que $\vec{L_R}$ et $\vec{L_I}$, le vecteur de la lumière incidente, sont symétriques par rapport à $\vec{N}$ donc $\vec{L_I} + \vec{L_R} = 2\ \cos(\theta) * \vec{N}$.
On obtient $\vec{L_R}$ par la formule suivante :
$\vec{L_R} = 2\ \cos(\theta)\vec{N} - \vec{L_I}$ donc $\cos(\alpha)$ est égal à :
$\cos(\alpha) = (2\cos(\theta)\vec{N}_x - \vec{L_{Ix}})\vec{MA}_x + (2\cos(\theta)\vec{N}_y - \vec{L_{Iy}})\vec{MA}_y + (2\cos(\theta)\vec{N}_z - \vec{L_{Iz}})\vec{MA}_z$

3 C'est la tâche de couleur qui se déplace sur l'objet lorsque l'on change de point de vue. Cela correspond plus ou moins à la reflexion de la lumière sur la boule.



Le modèle partiel de Whitted
Ce modèle prend en compte la réflexion d'un objet sur un autre

$$I = k_a I_a + \sum_{i=1}^{n} I_s (k_d \cos(\theta) + k_s ((\cos(\alpha))^m)) + k_s I_r$$

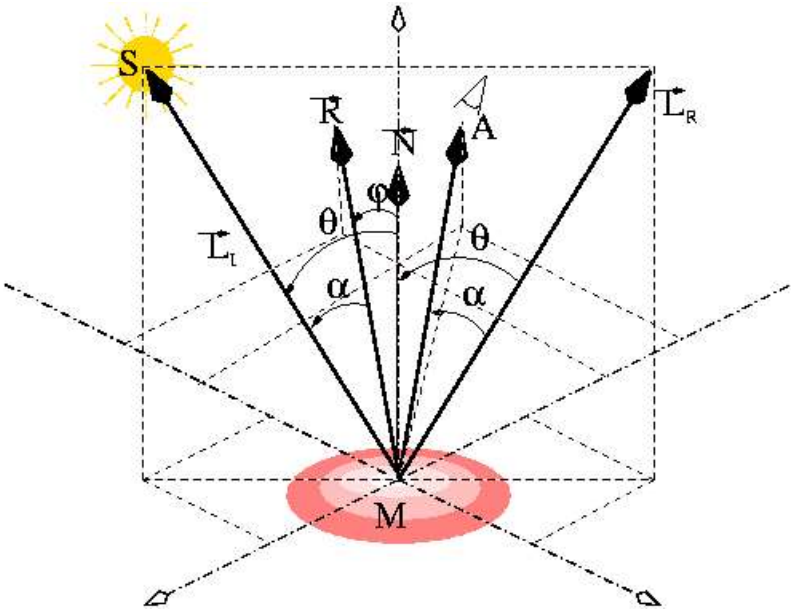
Le rayon réfléchi $\vec{R}$ est calculé par le fait que $\vec{R}$ et $\vec{MA}$ sont symétriques par rapport à $\vec{N}$ donc $\vec{MA} + \vec{R} = 2\ \vec{N} * \cos(\phi)$ donc $\vec{R} = 2\ \vec{N} * \cos(\phi) - \vec{MA}$
de plus

$$\cos(\phi) = \vec{MA} \cdot \vec{N} = \vec{MA}_x * N_x + \vec{MA}_z * N_z + \vec{MA}_z * N_z$$

L'intensité réfléchie I_r correspond à l'intensité du rayon envoyé suivant $\vec{R}$ à partir de M .

Comme le modèle de Whitted est récursif, il faut décider de relancer un rayon jusqu'à une certaine profondeur de récursion. On spécifie donc une profondeur (**depth**) qui correspond au nombre de rayons réfléchis qui seront calculés pour un rayon.

Le modèle de Whitted présenté ici n'est pas complet car il manque la transmission du rayon à l'intérieur de l'objet qui permet de gérer la transparence.
Pour le modèle de Whitted simplifié, l'apparence d'un objet dépend donc des coéfficients k_a , k_d , k_s et m .



Erreur fréquente : lorsque l'on relance un rayon réfléchi, il faut faire attention à ce qu'il ne rentre pas en collision avec l'objet qu'il vient de toucher.

Ligne de commande

java -jar rayoflight.jar [-time] scene.xml [-out image.png]

java -jar rayoflight.jar permet de lancer le programme.
-**time** indique que le temps de calcul[4] de la scène sera affiché en nano-secondes.
scene.xml correspond à un nom de fichier contenant une description
en XML d'une scène à raytracer.
-**out** permet d'indiquer le nom et le format de l'image voulu (ex: toto.gif, tutu.jpg)
sinon, par défaut, l'image générée est une PNG qui à le même nom que la scène en changeant l'extention.

Le format XML d'une scène

Une scène est décrite sous forme d'un fichier XML contenant les objets :

light
possède une position (x,y,z) et une couleur (**color**)

sphere, **plane**
objets qui peuvent avoir une couleur, un **material** et des **transformation**s

camera
possède une largeur et une hauteur en pixel et une distance entre l'oeil de l'observateur et la camera. Une camera peut subir des transformations

model
indique le modèle d'illumination parmi **ambient**, **lambert**, **phong** et **whitted**. **whitted** possède en plus un champs **depth** indiquant le nombre d'appels

4 Ce temps ne prend pas en compte le chargement de la scène ni la sauvegarde de l'image mais uniquement le temps de rendu

récursifs pour la réflexion.

color
spécifie une couleur au format RGB

material
spécifie pour un objet son comportement face à la lumière

transformation
indique les transformations sur un objet ou une camera. Les transformations possible sont :

translate x,y,z	effectue une translation.
rotate-x angle	effectue une rotation en x suivant un **angle** en radian.
rotate-y angle	effectue une rotation en y suivant un **angle** en radian.
rotate-z angle	effectue une rotation en z suivant un **angle** en radian.
scale factor	effectue un agrandissement suivant le facteur **factor**.

Exemple :

```
<?xml version="1.0"?>
<scene>
 <sphere>
  <color red="0.2" green="0.8" blue="0.2"/>
  <material ka="0.3" kd="0.3" ks="0.3" m="20"/>
  <transformation>
   <translate x="2" y="0" z="-2"/>
  </transformation>
 </sphere>

 <sphere>
  <color red="1.0" green="0.2" blue="0.0"/>
  <material ka="0.4" kd="0.5" ks="0.3" m="1"/>
  <transformation>
   <translate x="-2" y="0" z="-2"/>
  </transformation>
 </sphere>

 <light x="1" y="-4" z="-2">
  <color red="1.0" green="1.0" blue="1.0"/>
 </light>

 <camera width="600" height="600" eyeDistance="4">
   <transformation>
    <translate x="0" y="0" z="-20"/>
  </transformation>
 </camera>

 <model type="whitted" depth="2"/>
</scene>
```

Calendrier

Ce projet est à faire par binômes (cela veut dire deux personnes pas trois ni une).
Le rendu du projet est découpé en deux parties :

1. Pré-rapport indiquant l'architecture objet et tests d'utilisation.
   Date de rendu : le 31 mars, avant minuit.
2. Rendu du projet en lui-même.
   Date de rendu : le 11 mai, avant minuit.

Chaque rendu s'effectuera par mail sous forme d'une pièce jointe au format ZIP contenant l'ensemble des programmes et documents. Le mail devra être envoyé avant minuit aux trois adresses données au début de ce document.

Le fichier s'appelera nom1_nom2.zip avec nom1 et nom2 les nom des binomes dans l'ordre alphabétique.

Détail du premier rendu

Le premier rendu est composé de deux documents distincts :

un rapport d'architecture détaillant l'architecture objet proposée pour le programme.

un rapport de tests contenant des exemples de code commentés des tests à effectuer (les tests demandés sont détaillés plus bas).

Ces deux documents doivent être au format PDF.

L'architecture générale du projet (au moins 10 pages) :

Répondre aux questions :

Comment sont codées les transformations sur les objets ?
Comment l'algorithme de raytracing est codé en terme d'appels de méthodes entre les différents objets ?
Comment faire pour représenter les différents objets de la scène (boule et plan) ?
Qu'implique le format du fichier XML sur les objets ?
Comment faire pour utiliser un modèle d'illumination en fonction de ce qui est demandé dans le fichier XML ?

Doit suivre ensuite :

Le découpage en différentes classes (il y a environ 20 classes ici) en faisant un ou plusieurs schéma.
Le rôle de chaque classe.
L'ensemble des méthodes publiques de chaque classe.

Les tests : voici l'ensemble des essais à effectuer :

- création d'un script ANT de compilation.
- création d'un Jar exécutable.
- Parsing XML, création d'un parseur XML, les APIs DOM et SAX
  un exemple de chaque ainsi qu'une discussion pour motivé le choix d'une API plutôt que l'autre
- Image création d'une image en indiquant la couleur de chaque pixel
- ImageIO, trouver tous les formats de sauvegarde disponibles
- ImageIO, sauver une image au format PNG, JPG etc.

Chaque test doit être accompagné d'un texte expliquant les concepts manipulés, les instructions pour exécuter le test et le code pertinent du test (le plus clair possible, le plus efficace en restant lisible).

Eviter les copier/coller à partir du Web, la plupart du temps les exemples ne sont pas les plus pertinents quand il ne sont pas faux.

Remarque générale, essayez de faire des documents cohérents et lisibles, SVP.

Détail du second rendu

Le second rendu sera détaillé plus tard.

Références

- Jar exécutable
  http://java.sun.com/j2se/1.5.0/docs/guide/jar/jar.html

- Parsing XML
  cf paquetage javax.parser, org.w3c.dom et org.xml.sax

- Cours intéressants sur les bases mathématiques
  http://www710.univ-lyon1.fr/~ameyer/teaching/m1-image/c02_si_math.pdf
  http://www.siggraph.org/education/materials/HyperGraph/modeling/mod_tran/3d.htm

- Ant
  http://ant.apache.org/
  http://ant.apache.org/manual/

- ImageIO
  cf paquetage javax.imageio et plus particulièrement la classe ImageIO

- Génération d'une image en Java

```
BufferedImage image=
  new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
int[] data = ((DataBufferInt)image.getRaster().getDataBuffer()).getData();
//le tableau data est un tableau contenant width*height pixel encodé
//sur un int avec suivant le format 0xFF000000 | red<<16 | green<<8 | blue
```