Raytracer (2) - Parser

1 Introduction

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

Pour terminer le programme, il reste deux tâches à effectuer :

– doter votre programme d'un parser qui permet à l'utilisateur d'écrire sous forme textuelle les scènes à représenter,

– et programmer l'algorithme de raytracing proprement dit, qui à partir d'objets géométriques produit l'image correspondante.

Cette partie vise à réaliser le premier point. Elle est plus libre que la précédente. Seules les attentes minimales sont exposées, sous forme de "buts à atteindre": votre projet doit fournir ces fonctionnalités, mais vous choisissez l'architecture de votre programme.

La section 2 présente la représentation informatique des objets utilisés dans les scènes. La section 3 donne la spécification du langage de scénarios à parser. Enfin la section 4 précise le planning et la notation du projet.

2 Représentation informatique des objets manipulés

2.1 Représentation informatique des couleurs

Un codage classique des couleurs consiste à décomposer une couleur en ses trois composantes primaires Rouge Vert Bleu[1] (RGB en anglais). Chaque composante a une valeur réelle (un `float` pour nous) comprise entre 0 et 1. Le blanc vaut alors $(1,1,1)$ et le noir $(0,0,0)$. Toutes les opérations sur les couleurs se font terme à terme, et les valeurs plus grandes que 1 sont ramenées à 1, plus petites que 0 ramenées à 0.

Formellement, on définit *la somme c de deux couleurs* $c_1,c_2 \in [0,1]^3$ par $\forall i \in \{R,V,B\}$, $c(i) = max(min(c_1(i)+c_2(i),1),0)$. De façon similaire, on définit la multiplication sur les couleurs : *le produit c de deux couleurs c_1 et c_2* est défini par $\forall i \in \{R,V,B\}$, $c(i) = max(min(c_1(i).c_2(i),1),0)$.

Avec ce codage on a une valeur "idéale" de la couleur. Mais l'affichage est limité en nombre de couleurs disponibles. Une constante Max est fixée qui donne la précision. Dans la suite, nous prendrons la valeur 255, qui constitue le choix habituel. Chaque composante est alors codée par un entier compris entre 0 et Max. Aussi au moment du rendu on obtiendra le codage approché en multipliant chaque composante par Max.

1. Dans des systèmes plus élaborés on ajoute un canal pour la transparence.

2.2 Propriétés optiques des objets géométriques

Dans la description d'une scène 3D, plusieurs facteurs (autres que les facteurs géométriques) vont influer sur les couleurs : la texture de chaque objet, les sources lumineuses et l'éclairage ambiant. Chacun de ces paramètres sera utilisé par l'algorithme de raytracing.

Lumière ambiante. On considère une lumière ambiante avec une intensité $I \in [0,1]$. On peut là aussi la définir en format RGB.

Positionnement de la caméra. Le point d'observation de la scène, appelé "caméra", est décrit par deux valeurs : la distance à l'origine $d \in \mathbb{R}^+$ et un angle de vision $\alpha \in [0,2\pi]$.

Sources lumineuses. Chaque source lumineuse est définie par son intensité $I \in [0,1]$ et une direction donnée par un vecteur directeur. On considère dans ce cas des sources blanches situées à l'infini. Il est possible de définir des sources de couleur en considérant une intensité donnée en format RGB.

Textures. Chacun des objets géométriques a un attribut `texture` utilisé pour décrire ses propriétés optiques. On propose le modèle de texture suivant (qu'il est possible d'enrichir) :

- un coefficient de réflexion diffuse $k_d \in [0,1]$,
- un coefficient de réflexion spéculaire $k_s \in [0,1]$,
- un coefficient de Phong $n \in \mathbb{R}^{+*}$.
- une couleur de surface $C \in [0,1]^3$ dans le modèle RGB,

3 Description des scènes 3D et parser

3.1 Scénarios

On appelle *scénario* la description textuelle de la scène 3D à représenter. Un scénario défini donc les objets suivants :

- la lumière ambiante,
- la position de la caméra (donnée par une distance à l'origine et un angle de vision),
- les sources lumineuses,
- les objets géométriques (plans, sphères et cubes).

3.2 Le langage de description

Un fichier de description consiste en trois parties :

1. les propriétés globales du scénario ;
2. une suite de définitions de procédures ;
3. une suite d'instructions pour placer des objets et sources lumineuses.

Un exemple "complet" de fichier de description est donné dans l'annexe A. Nous donnons dans la suite la syntaxe de chacune de ces parties.

Propriétés globales. Les propriétés globales du scénario sont la lumière ambiante et la position de la caméra. La définition de la lumière ambiante a la forme suivante :

```
ambient I
```

La position de la caméra est donnée comme suit :

```
camera
distance d
angle alpha
end
```

Définition de procédures. Une *définition de procédure* est constituée du mot clef `proc` suivi du nom de la procédure, puis, entre parenthèses, des paramètres formels avec leurs types (au moins `objet`, `flottant`, `entier`) séparés par des virgules. Le corps de la procédure est donné par une liste d'instructions. La fin de la procédure est signalée à l'aide du mot clef `end`.

Instructions. Les différentes *instructions* sont

- l'affectation `let` *identificateur* = *expression*. L'effet d'une affectation est seulement visible dans la portée locale. Par conséquent une procédure ne peut pas modifier des variables globales,
- l'appel d'une procédure[2], donné par le nom de la procédure suivi des arguments entre parenthèses et séparés par des virgules,
- la définition d'un objet géométrique sous la forme `put` *expression*,
- une instruction conditionnelle `if` ... `then` ... `else` ... `end`.

Expressions. Les *expressions* dénotent des valeurs flottantes, des objets et des sources lumineuses. Elles sont bâties sur des identificateurs, des constantes et des opérateurs.

Sources lumineuses. Une *source lumineuse* est décrite par :
`put light`
`rotation` r_x,r_y,r_z
`intensity` i
`end`
où r_x, r_y et r_z sont des expressions donnant la rotation (la rotation $0,0,0$ correspond à une lumière qui vient du haut), et i une expression qui dénote l'intensité.

Texture. Une *texture* est donnée par la syntaxe suivante :
`kd` k_d
`ks` k_s
`phong` n
`color` c_r,c_g,c_b
où k_d, k_s, n, c_r, c_g et c_b sont des expressions donnant les coefficients de réflexion diffuse, de réflexion spéculaire, l'exposant de Phong, et les trois composantes de la couleur. Les composantes des couleurs prennent des valeurs entières comprises entre 0 et 255 (attention, l'échelle change entre la représentation interne et le langage de description). Dans la suite, nous utiliserons le symbole t pour noter une description de texture satisfaisant cette syntaxe.

Objets géométriques. Les syntaxes permettant de décrire *plans*, *boîtes* et *sphères* sont respectivement :

2. Les procédures récursives sont autorisées.

```
plane
rotation rx,ry,rz
shift s
t
end
```

```
box
center cx,cy,cz
rotation rx,ry,rz
length lx,ly,lz
t
end
```

```
sphere
center cx,cy,cz
radius r
t
end
```

où t est une texture.
Les opérations que l'on peut appliquer à un objet o ou à une source lumineuse o sont la translation, la rotation et l'homothétie. Elles sont données par les syntaxes suivantes :

```
translate o
by cx,cy,cz
end
```

```
rotate o
by rx,ry,rz
end
```

```
scale o
by f
end
```

Opérateurs et constantes. On utilise les opérateurs arithmétiques et booléens habituels ($+, -, \times, /, \wedge, \vee, \neg$) ainsi que les fonctions trigonométriques (*sin*, *cos*) et les constantes standards (π).

Commentaires. Tout texte entre le caractère # et la fin de ligne est un commentaire.

4 Travail à réaliser

4.1 Consignes pour le parser

Syntaxe. Votre parser doit se conformer aux spécifications ci-dessus concernant la syntaxe. Vous pouvez rajouter tout ce que vous voulez, mais la compatibilité doit être assurée. Par exemple si vous autorisez les sources de couleur RGB, vous devez quand même permettre les sources de lumière blanche définies par une seule valeur.

Gestion des erreurs. Vous devez tester les erreurs syntaxiques. Le minimum est de gérer des erreurs telles que : couleurs pas dans les bornes, erreur de type, variable inconnue.

Travail requis. Pour le parser, le *strict minimum* est de reconnaître la syntaxe ci-dessus, sans variables, ni conditionnelles, ni expressions booléennes, ni procédures. Les autres parties de la syntaxe qui ont été décrites ci-dessus sont des pistes pour enrichir votre parser. Plus le parser est riche (et fonctionne), meilleure sera la note.

4.2 Le rapport 2

Pour le rapport 2, vous coderez un type représentant la syntaxe abstraite des descriptions des scénarios et l'analyse lexicale et syntaxique (fichiers jflex et cup). *Vous créerez un petit programme principal qui lit un fichier de description et qui affiche s'il est syntaxiquement correct ou pas*. Vous préparerez également un jeu de tests qui servira à la fois pour le programme à rendre pour la partie 2, et aussi pour le raytracer complet.

Le rapport doit contenir les parties suivantes :

1. présentation et motivation des choix concernant les fonctionnalités de votre parser,
2. la grammaire d'entrée et une liste des cas d'erreur gérés.

3. en annexes :
    - vos jeux de test pour le parser,
    - le code de votre parser.

Le rapport est à rendre au plus tard **le 5 avril 2007**. Ce rapport est intermédiaire : il sert à évaluer votre avancement. De plus, il vous oblige à prendre du recul, et à ne pas prendre de retard. *Cependant, il ne fixe pas la suite du projet : vous pourrez encore enrichir votre parser pour la version finale du projet.*

A Exemple de scénario

```
ambient 0.4

camera
  distance 5000
  angle 0.8
end

# placer n copies de l'objet o, decales par shiftx,shifty,shiftz
proc repeat (objet object,flottant shiftx, flottant shifty, flottant shiftz, entier n)
  if n < 1
  then
  else
    put object
    repeat(translate object by shiftx,shifty,shiftz end,
           shiftx, shifty, shiftz, n-1)
  end
end

put light
  rotation 0,0,0
  intensity 0.3
end

let boule = sphere
  center -500.0,-450.0,4500.0
  radius 100.0
  kd 1.0
  ks 1.0
  phong 2
  color 255,20,20
end

put sphere
  center 0,-500,8000
  radius 2000
  kd 0.8
  ks 1
  phong 4
  color 240,240,240
end

put plane
  rotation 0.08,0.0,0.0
  shift -520
  kd 0.8
  ks 0.2
  phong 1.5
  color 200,200,200
end

repeat (boule,300,0,0,5)
```