Réalisation d'un *Raytraceur.*

1 Introduction

Le projet consiste à programmer un raytraceur (outil de rendu graphique) dans le langage Java. Un raytraceur calcule pixel par pixel l'image à afficher, en fonction d'une description de la scène. Le programme final devra prendre en entrée une description textuelle de la scène à représenter, et construira l'image correspondante.

1.1 buts du projet

Le projet de programmation du second semestre vise à mettre en pratique les connaissances acquises dans les domaines suivants : compilation et langages formels, conception objet. De plus il vise à vous sensibiliser à la réalisation de programmes de taille conséquente, pour lesquels plusieurs modules doivent interagir entre eux. Enfin vous avez dans ce projet une *autonomie* certaine. Le cahier des charges n'est pas figé, et si certaines parties sont obligatoires, vous pouvez rajouter ce qui vous semble approprié, du moment que vous êtes capables de justifier vos choix.

1.2 le raytracing

Un *raytraceur* est un outil qui permet de créer des images réalistes en utilisant une idée assez simple : pour chaque point de l'image, on calcule la lumière qui arrive à cet endroit en utilisant les équations décrivant les propriétés optiques de la lumière et des objets[1]. On décrit les objets présents dans la scène à représenter, les différentes sources de lumières et la position de l'observateur, le programme se charge de calculer l'image correspondante.

Cette approche est lourde mais donne de très belles images. Elle est donc utilisée pour réaliser des images fixes ou des films, mais en aucun cas des jeux. Si vous avez accès à internet vous pouvez voir quelques exemples d'images impressionnantes obtenues avec un bon raytraceur gratuit (POVRAY) sur le site `http://www.povray.org`. Dans ce projet, on va réaliser un traceur plus modeste, mais qui permet toutefois d'obtenir de bonnes images.

2 Le projet

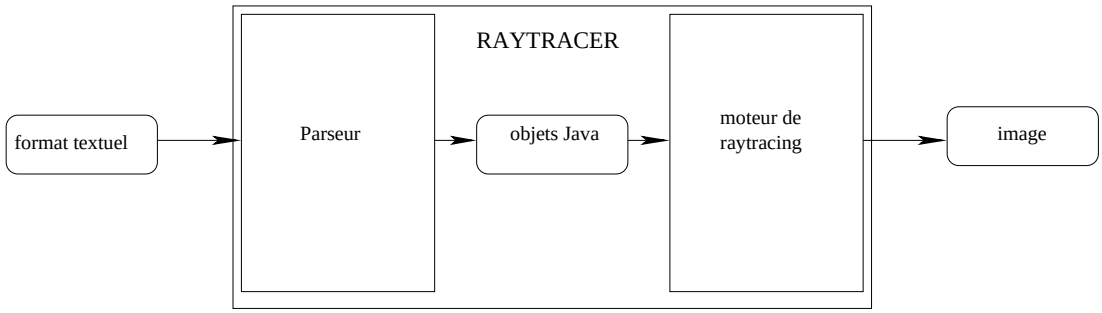
Notre programme doit calculer et afficher les images qui sont décrites par l'utilistateur. On peut donc distinguer au moins 3 modules du programme :

- le parseur, qui traduit la description textuelle de la scène donnée par l'utilisateur (objets et leurs positions, caméra, lumière ambiante, ...) en une représentation interne au programme ;
- le noyau, qui contient les objets géométriques qui représentent la scène, et qui calcule l'image correspondante ;

[1] Ces propriétés seront rappelées, ce n'est pas un sujet d'optique)

– le rendu, qui affiche l'image à l'utilisateur (cette partie est assez facile, car des bibliothèques la gèrent).

Concrètement, le programme final sera semblable à la figure suivante.



3 Plan de travail

Les séances de projet vont se dérouler comme suit. D'une part des séances durant lesquelles vous travaillerez le projet proprement dit. D'autre part des séances de TPs qui vous donneront les bases techniques nécessaires au projet.

3.1 le projet

Le projet se divise en 2 parties.

1. La première partie consiste à concevoir et implanter les objets dont vous aurez besoin pour le raytracing. Cette partie est assez guidée, puisque les objets et leurs interfaces seront imposés[2]. L'accent est mis surtout sur la découverte du langage Java et sur les concepts objets tels que l'héritage.
2. La deuxième partie consiste à terminer le raytraceur. Vous devrez donc implanter l'algorithme de raytracing complet, ce qui constituera le noyau de votre programme. Vous devrez aussi définir l'interface entre votre programme et le monde extérieur. Par exemple, vous aurez à définir votre langage d'entrée, à choisir les fonctionnalités, ... Dans cette partie vous aurez beaucoup plus de choix. Les spécifications sont données informellement, ce qui laisse la place à l'interprétation.

Finalement, il faudra rendre tout au long du projet de petits rapports pour vérifier l'avancée des travaux. Un rapport est à rendre à la fin de la prmière partie, un autre au milieu de la seconde partie et le rapport final est à rendre a la fin du projet avant la soutenance finale.

3.2 les TPs

Des séances de TP pour assimiler les techniques nécessaires au projet sont prévues. Pour le moment, deux TPs sont préparés : un tutoriel Java et un tutoriel de Lex/Yacc pour apprendre à faire des parseurs. Si vous ressentez le besoin d'autres séances, n'hésitez pas à nous le signaler.

Les TPs et le projet avanceront de concert. Les élèves qui iront plus vite que d'autres sur les séances de TPs pourront utiliser celles-ci comme séances de projet.

[2]Si vous souhaitez faire autrement, c'est possible mais il faut pouvoir le justifier.

3.3 planning

Le planning prévisionnel est le suivant :

première partie *TPs Java, réalisation des classes de base*	**6 février au 3 mars**
deuxième partie, début *TPs parseur, conception* *rapport intermédiaire*	**6 mars au 7 avril**
vacances	**8 avril au 23 avril**
deuxième partie, fin *terminer le programme, rapport final*	**24 avril au 26 mai**
soutenance	**5 au 7 juin**

4 Évaluation

4.1 les critères de notation

Sans rentrer dans les détails, voici les critères sur lesquels se basera l'évaluation.

- la conception, qui devra utiliser les possibilités offertes par l'approche objet, tout en restant aussi claire et logique que possible. Vous devez être conscient de vos choix, et capables de les justifier.
- la qualité de la programmation : si il n'est pas demandé aux élèves d'être des virtuoses du Java, le programme final *doit* fonctionner pour obtenir la moyenne. Ce qui implique qu'il doit avoir été testé convenablement auparavant, la découverte d'un bug à la soutenance étant du plus mauvais effet [3]
- la présentation (rapport final et soutenance) : vous devrez soutenir votre projet. Il s'agira donc de motiver vos choix, et de savoir mettre en valeur les atouts de votre programme. En clair, présentez votre travail sous le meilleur jour, mais soyez prêt à justifier chaque assertion.
- la remise à temps des rapports.

Enfin pour éviter les malentendus, voici quelques exemples de projets à éviter. Ce que le projet ne doit pas être :

- un programme qui ne fonctionne pas ; dans ce cas la moyenne ne peut être attribuée.
- un rapport baclé ; en plus de perdre les points relatifs au rapport, le correcteur risque d'être plus exigeant à la soutenance. Si l'étudiant n'est pas clair dans ses réponses, il pourrait aussi perdre les points de conception.
- un rapport qui ne correspond pas au programme ;

4.2 travail minimal et travail optionnel

Pour chaque partie du projet, les attentes minimales seront indiquées. Un projet qui tourne et qui respecte ces attentes minimales, accompagné d'un rapport concis, clair et honnête sera noté au moins 12.

[3] Il faut en particlier vérifier la robustesse du code sur des entrées inhabituelles.

Vous pouvez rajouter toutes les fonctionnalités que vous trouverez utiles. Si elles marchent et que vous justifiez de leur utilité, cela sera pris en compte dans la notation. Dans chaque partie seront données des pistes pour aller plus loin.

4.3 conseils

Voici quelques conseils pour le projet :
- Préparez par écrit une description de votre implémentation *avant* de coder.
- Procédez étape par étape. Ne rajoutez un morceau que si vous avez testé le code déjà écrit.
- Faites déjà le travail minimal, puis rajoutez ensuite les fonctionnalités que vous désirez. Vous assurez ainsi un programme qui fonctionne et une note correcte.
- Ne négligez pas la présentation, c'est essentiel dans de nombreux domaines et particulièrement en recherche.

Il est plus facile de rendre rapide un programme correct, que de rendre correct un programme rapide.
- Tanenbaum (?)

Contacts :
Fabrice Chevalier `chevalie@lsv.ens-cachan.fr`
Pierre-Alain Reynier `reynier@lsv.ens-cachan.fr`