Ray Tracer

COMPTE RENDU

réalisé par Pierre-Jean BESNARD & Louis BILLAUT

COMPTE RENDU **1**
Mise en contexte et conception **3**
Introduction 3
Architecture des classes 3
Produit final **4**
Fonctionnalités disponibles 4
Difficultés rencontrées 5
Utilisation **5**
Guide d'installation 5
Mise en marche 6
Exemples de rendu 6

Mise en contexte et conception

Introduction

Ce projet a été développé durant notre première année de Master logiciel dans le cadre du cours de synthèse d'image. Ce cours nous a été donné par M. Biri qui a également entièrement pensé ce sujet de projet. Nous devions donc mettre en place un programme capable d'effectuer des "lancers de rayons" afin de créer l'image numérique d'une scène constituée d'objets définis dans un fichier donné au programme. Plusieurs effets de lumière liés directement à la nature et à la couleur des différents objets constituants la scène devaient être calculés avec pour objectif de rendre une image la plus réaliste possible.
Environ 200 heures de travail cumulées ont étés nécessaires à la conception de ce raytracer.
Ce document explique brièvement les choix architecturaux faits pour concevoir le programme, fait état des fonctionnalités implémentées dans le produit final et explique en détail comment utiliser celui-ci correctement.

Architecture des classes

Un nombre non négligeable de classe a été développé afin de mettre en oeuvre ce raytracer. Voici une liste exhaustive des différents types d'objet et interfaces mis en place.

- Vector :
  Classe qui définit un vecteur, disposant de trois coordonnées. De nombreuses fonctions ont étés mises en place afin de manipuler cet objet essentiel dans ce projet.

- Ray :
  Classe définissant un rayon pouvant être renvoyé vers ou par une surface.

- Shape :
  Interface réunissant les différentes formes pouvant composer une scène. Shape réunit les caractéristiques communes au différentes formes.
  Implémentée par les classes : Sphere, Triangle, Rectangle et Cylinder, classes qui représentent les formes dont elles portent le nom.

- FlatPaintingRayTracer :
  Classe représentant un ray tracer simple, appliquant un lancé de rayon en "flat painting".

- SimpleRayTracer:

Classe permettant d'appliquer un lancé de rayon ne prenant en compte qu'une source de lumière multidirectionnelle. L'illumination des éléments de la scène dépend de leur forme.

- ComplexRayTracer :
  Classe mettant en oeuvre un lancé de rayon prenant en compte une source de lumière de type sphérique. L'illumination des objets de la scène est composée de la lumière diffuse comme le SimpleRayTracer, de la lumière ambiante et la lumière lié à l'aspect spéculaire de ceux-ci.

- Scene:
  Classe représentant une scène, dotée de plusieurs Shape et de lumières.

- Reader :
  Classe créant un objet de type Scene à partir d'un fichier JSON normalisé.

Produit final

Fonctionnalités disponibles

Cette section fait état des fonctionnalités implémentées dans le produit final.

Niveau 1 :
Le niveau 1 du sujet a été entièrement réalisé. Quatre formes différentes peuvent constituer une scène à savoir : un cylindre, une sphère, un rectangle et un triangle. La scène peut être chargée via un fichier de donnée JSON et rendue avec une technique de flat painting dans une image au format ppm.

Niveau 2 :
Le second niveau énoncé par le sujet a lui ici été réalisé dans son intégralité. L'illumination de chaque objet composant la scène dans ce mode dépend de sa forme, de sa position et de l'intensité de la lumière et de sa composante de Phong. Le rendu de la scène est affiché via une fenêtre graphique de la bibliothèque OpenGL.

Niveau 3 :
Le troisième et dernier niveau décrit par le sujet a été entièrement réalisé. Certaines caractéristiques particulières peuvent être appliquées sur les objets créés comme le fait d'être de type « miroir » ou « verre », si tel est le cas, les rayons sont renvoyés de manière récursive (limité à 5 récursions). Les rendus en ppm apparaissent légèrement bruités, une option de pixel sampling est également disponible afin de renvoyer plusieurs rayons par pixel et obtenir une image plus net et réaliste.

Fonctionnalités optionnelles :
Plusieurs fonctionnalités optionnelles ont étés ajoutées. Les lumières sont maintenant des sphères (sauf en flat painting). Le format des fichiers de scène à été simplifié et à été fixé sur le format JSON. Le niveau 3 à été amélioré et prend maintenant en compte les effets de lumière indirectes rebondissants sur les formes pour produire la scène qui, avec un pixel sampling important (minimum 80), semble bien plus réaliste.

Difficultés rencontrées

Le projet à été élaboré en un nombre d'heure non négligeable. Cela s'explique notamment par notre manque de formation quant au langage de programmation C++ mais également par notre novicité en synthèse d'image. Un travail conséquent de formation a donc été réalisé en amont, aussi bien en C++ qu'en synthèse d'image, afin de mener à bien ce projet pour lequel notre intérêt grandissait à mesure que nos connaissances s'enrichissaient.
Développer ce raytracer ne s'est pas fait sans difficulté, les obstacles ont étés nombreux notamment lorsque nous avons mis en oeuvre le niveau 3 de ce projet. Le rendu n'est pas parfait, le niveau 3 est facilement bruité et le niveau 2 assez lent. Nous sortons finalement grandis mais surtout enrichis par cette expérience, en C++ comme en synthèse d'image.

Utilisation

Guide d'installation

Prérequis :

Veillez à avoir installé sur votre machine les outils suivants avant de compiler ce projet :

- libjson *(sudo apt-get install libjsoncpp-dev)*
- openGL *(sudo apt-get install freeglut3-dev)*
- C++ 11

Les différentes parties du projet se trouvent dans les répertoires suivants :

- RayTracer/documentation : *La documentation déjà générée du projet*
- RayTracer/sources : *Les fichiers source du projet*
- RayTracer/sources/include : *Les sources utilisées pour le parseur de commande*
- RayTracer/examples : *Des exemples de scènes générées*
- RayTracer/scenes : *Des fichiers Json décrivant des scènes à utiliser*

Installation :

- Placez vous dans le répertoire source et lancer la commande :
    - cmake sources

- Vous pouvez ensuite compiler le projet en utilisant la commande :

- make

Mise en marche

Commandes et options :

L'application suit strictement les consignes énoncées dans le sujet. Pour plus de précisions, après avoir compilé le programme, tapez la commande :

- ./raytracer -help

Format des scènes :

Le nom des valeurs définissants les différents objet à été travaillé pour être le plus clair possible néanmoins la scène à l'adresse *RayTracer/scenes/example1.json* est commentée de manière à rendre compréhensible le format des scènes au maximum, il est donc conseillé d'observer le fichier avant toute utilisation.

Exemples de rendu

Plusieurs rendus ont étés testés, voici nos résultats. Le format d'image est trop lourd pour être inséré ici, vous pouvez retrouvez nos meilleures clichés dans le répertoire :

- *RayTracer/scenes*

Veuillez noter que :

- Dans le nom example1_level2_1pm l'abréviation ps signifie "pixel sampling" et indique le nombre de rayons lancés pour effectuer ce rendu.
- Toutes les scènes utilisées sont disponibles dans le fichiers RayTracer/scenes.