Université Gustave Eiffel

Cours de Synthèse d'Images 1
Master 1 Informatique

Projet de Synthèse d'images

Maîtriser le lancer de rayons simple dans une scène hiérarchisée

Présentation du sujet :

La technique du lancer de rayon a été vue en cours, les slides sont disponibles sur ma page web. Par ailleurs, de nombreuses documentations sont disponibles sur Internet. Voir par exemple la page wikipedia sur le lancer de rayon (attention la partie 'Futur et prospectives' est largement dépassée). Il existe pas mal de code disponible mais probablement pas adapté à votre projet.

L'objectif du projet sera de réaliser, en binôme, un lancer de rayon en C/C++.

Travail demandé :

Niveau	Travail demandé	« Module » informatique à réaliser	Note max
1	Lancer de rayon simple	- chargement de la scène par fichier - caméra statique - restitution d'une image finale - lancer de rayon en *flat painting*	8
2	Lancer de rayon interactif	- insertion des lumières et calcul du modèle de Blinn-Phong - restitution de l'image dans une fenêtre de visualisation - déplacement de la caméra - lancer de rayon « progressif »	12
3	Lancer de rayon interactif récursif	- calcul de l'ombrage - calcul des reflets & de la transparence - pixel sampling - extensions des formats d'entrée et de sortie	16
-	Lancer de rayon interactif récursif optimisé	- création d'une structure de donnée hiérarchique - lancer de rayon exploitant la structure de données - création d'une scène de plus de 100 objets canoniques	+ X note maximale 20

Les extensions peuvent s'appliquer sur les « niveaux » 1, 2 et 3.

De plus, 2 points sont attribués au rapport final & à la qualité du rendu : qualité du code, facilité d'installation et de compilation, et respect des spécifications demandées.

Contact :

Pour toutes demandes ou précisions, merci d'envoyer un mail à vos deux enseignants sur les mails suivants : biri@u-pem.fr et incerti@u-pem.fr. Merci de bien envoyer un mail **aux deux en même temps** afin que nous soyons tout deux informés de votre demande.

Détails des modules informatiques à réaliser

Chaque niveau doit aboutir à une application « standalone » en C/C++, qui fonctionne sous linux, et qui permet de réaliser les fonctionnalités demandées et détaillées ci-dessous.

Niveau 1 : le lancer de rayon simple

Format et chargement de la scène

En lancer de rayon (comme en rasterisation d'ailleurs), les scènes virtuelles à rendre sont constituées d'objet canonique. Ce sont les éléments « unitaires » constituant la scène sur lesquels les tests d'intersection avec les rayons sont fait. Dans notre cas, la scène sera constituée des objets canoniques suivant : sphère, rectangle, facette (triangle ou quadrilatère) et cylindre. Pour le niveau 1, vous rendrez une scène constituée d'au moins un objet de chacun de ces types.

La construction de la scène doit se faire via le chargement d'un fichier au format que vous désirez parmi les possibilités suivantes : simple fichier texte (txt), json, XML. Le format « interne » de ces fichiers est laissé à votre appréciation mais chaque objet devra comporter les informations suivantes :
- Ses caractéristiques géométriques intrinsèques : coordonnées des 3 points pour un triangle, centre et rayon pour une sphère…
- Des informations sur sa couleur ou ses propriétés radiométriques nécessaires aux modèles d'éclairement (coefficient de réflexion diffuse par exemple)
Éventuellement, pour pouvez associer à chaque objet une matrice de transformation mais cela est laissé à votre appréciation.

Le fichier contiendra également la position et l'orientation de la caméra. Enfin, vous pourrez définir des propriétés supplémentaires comme la position & l'intensité de sources de lumière ou d'autres points de vue, exploitable pour les prochains niveaux de l'application.

Votre application devra être capable de lire ce fichier et de réagir en conséquence.

L'application du lancer de rayon

Pour le niveau 1, vous devrez donc réaliser un exécutable qui prend en entrée un fichier au format défini par vos soins (voir ci-dessus) décrivant la scène et qui doit restituer une image en sortie.

L'exécutable, appelé lray, donc pouvoir répondre à la commande suivante :

```
lray -n 1 -i <mon_fichier.format> -o image.ppm
```

L'option -n défini le « niveau » du lancer de rayon exécuté. L'option -i défini le fichier d'entrée. L'option -o défini l'image de sortie.

Ainsi le fichier mon_fichier.format contient la scène et la position caméra et le programme génère une image au format PPM contenant une vue de la scène depuis la caméra. Ce lancer de rayon enverra un rayon par pixel et renverra une image en *flat painting* (i.e. une couleur par objet) de la scène.

Le format de fichier ppm est très simple, vous pouvez en trouver les spécifications ici :
https://fr.wikipedia.org/wiki/Portable_pixmap

Niveau 2 : le lancer de rayon interactif

Cette partie, légèrement plus technique, permet de visualiser directement le rendu que l'on obtient dans une fenêtre de visualisation. Vous pouvez, pour ce faire, vous appuyer sur la bibliothèque fournie par M. Incerti ou directement via OpenGL. Vous devrez, en *offline*, construire l'image en tirant un rayon par pixel, puis l'afficher dans la fenêtre GL. Vous pouvez réitérer le processus en moyennant l'image, cela vous permet ainsi d'affiner l'image finale interactivement. C'est particulièrement utilisé pour des algorithmes plus puissant comme le *path tracing*, mais ici cela permettra essentiellement de voir la scène directement et à positionner la caméra.

Visualisation de l'image dans une fenêtre et interaction utilisateur

Ici, au lieu de générer une image finale, l'idée est d'afficher cette image dans une fenêtre de visualisation. Il faut donc générer une fenêtre et, simplement, remplir cette fenêtre avec l'image (sous forme de tableau d'`unsigned char`) générée par le lancer de rayon.

Si vous utilisez OpenGL, remplir le framebuffer – la fenêtre – se fait simplement avec les instructions glDrawPixels (et éventuellement glDrawBuffers avec la cible GL_FRONT).

De plus, grâce à l'action de touches sur le clavier, vous permettrez à l'utilisateur de bouger la caméra. En avançant, en reculant et en orientant la caméra à gauche & à droite (au minimum). Le choix des touches devra être explicité, soit dans le rapport, soit explicitement via le programme comme une option d'aide en ligne de commande, ou l'action d'une touche dans l'application. A chaque déplacement, l'image doit être entièrement recalculée et ré-affichée. Cela permet donc

Lancer de rayon « progressif »

Tant que la caméra ne bouge pas, le programme continue à « rendre », en *offline*, l'image et à « moyenner » l'image finale, soit celle qui est affichée dans la fenêtre. En gros, tant que la caméra ne bouge pas, vous demandez le rendu de l'image avec un rayon par pixel. L'image générée doit alors se « moyenner » avec l'image déjà calculée, pixel par pixel. Pour « moyenner » chaque pixel, vous pouvez utiliser la formule suivante (pour chaque canal) :

$$P_{new} = \frac{1}{n} P_{arrive} + \frac{n-1}{n} P_{ancien}$$

où P_{new} sera la nouvelle valeur du pixel, n le nombre d'itération (de rendu effectué), P_{arrive} la valeur du pixel que l'on vient de calculer et P_{ancien} l'ancienne valeur du pixel

Dans votre application, une touche devra permettre d'arrêter et de reprendre à volonté ce rendu « progressif ».

Calcul du modèle d'illumination de Blinn-Phong

Vous modifierez le format de fichier de description de la scène afin d'insérer des lumières dites ponctuelle (constituée d'un point et éclairant dans toutes les directions). Ces lumières sont donc caractérisées par une position et une intensité lumineuse.

Les lumières entraîneront un éclairage de la scène. Pour le modèle d'éclairage nous utiliserons le modèle de Blinn-Phong rappelé ci-dessous pour chaque lumière ponctuelle. Si X est le point illuminé, S la position de la source de lumière, l'illumination vue depuis le point de vue V s'écrit :

$$L(X \to V) = \frac{L}{4\pi XS^2}[\rho_d \underline{\vec{N} . \vec{I}} + \rho_s (\underline{\vec{N} . \vec{H}})^s]$$

avec :

- ρ_d est la couleur diffuse (flat painting) ou coefficient de réflexion diffuse de l'objet (3 composantes r,v,b)
- ρ_s est le coefficient de réflexion spéculaire de l'objet (3 composantes r,v,b)
- s est la brillance (shininess) de l'objet (usuellement comprise entre 2 & 256)
- $\vec{N}$ est la normale à la surface en X
- $\vec{I}$ est le vecteur d'illumination et vaut $\frac{\vec{XS}}{\|\vec{XS}\|}$

- $\vec{H}$ est le half vector et est la *normalisation* du vecteur $\frac{\vec{I}+\vec{V}}{2}$ avec $\vec{V}=\frac{\vec{XV}}{\|\vec{XV}\|}$
- $\underline{\vec{u}.\vec{v}}=\vec{u}.\vec{v}$ si $\vec{u}.\vec{v}\geqslant 0$ et 0 sinon

L'application du lancer de rayon optimisé interactif

Pour le niveau 2, vous devrez donc réaliser, comme pour le niveau 1, un exécutable qui prend en entrée un fichier au format défini par vos soins (voir ci-dessus) décrivant la scène et qui doit soit restituer une image si l'option -n est suivi du nombre 1, soit passer en mode interactif si le chiffre après l'option -n est égal à 2 (ou plus). Dans ce cas, il ne doit pas y avoir d'option -o (sauf éventuellement à permettre un rendu, à l'appui d'une touche, dans ce fichier).

Ainsi l'exécutable, appelé lray, donc pouvoir répondre, au niveau 2, à la commande suivante :

```
lray -n 2 -i <mon_fichier.format>
```

Niveau 3 : le lancer de rayon interactif récursif

Le lancer de rayon « niveau 1 ou 2 » reste trop limité en terme de représentation d'effet. Votre algorithme permettra maintenant d'une part de pouvoir tirer plusieurs rayons par pixel, et d'autre part de permettre le calcul des réflexions et réfractions spéculaires pures.

Le *pixel sampling*

Votre application devra permettre de calculer des images en lançant plusieurs rayons par pixel. Ce paramètre, appelé *pixel sampling*, devra être spécifié dans la ligne de commande (voir ci-dessous). Au lieu de tirer un seul rayon au milieu du pixel, il faut maintenant tirer N rayons (N étant le *pixel sampling*). Chaque rayon doit alors passer par un point tiré aléatoirement dans le pixel. Chaque rayon renvoie une couleur et l'ensemble de ces couleurs est moyenné pour obtenir la couleur finale du pixel.

Calcul de l'ombrage

Pour chaque point où l'on calcul le modèle de Blinn-Phong, il faut vérifier si il est illuminé ou non par chaque source de lumière ponctuelle. Pour ce faire, il est nécessaire de créer une fonction qui vérifie, pour deux points de la scène 3D, et donc un segment, si celui-ci intercepte un autre objet de la scène. Si il est intercepté, alors le point n'est bien sûr pas éclairé. En substance, le modèle d'éclairage de Blinn-Phong devient en fait :

$$L(X \rightarrow V) = \frac{L}{4\pi XS^2} \nabla(X,S)[\rho_d \underline{\vec{N}.\vec{I}} + \rho_s (\underline{\vec{N}.\vec{H}})^s]$$

où $\nabla(X,S)$ vaut 0 ou 1 suivant que le point est illuminé ou non.

Contrairement à la fonction de lancer de rayon que vous aurez probablement codé précédemment qui renvoie une valeur de luminance, cette nouvelle fonction de test d'ombre ne renvoie qu'un booleen. De plus, en raison des erreurs numériques inhérentes aux calculs en flottant, il est nécessaire de prévoir un léger *epsilon* au début et à la fin du segment testé afin que la surface de départ et d'arrivée ne soient pas considérées comme des intercepteurs de ce segment.

Récursion pour les réflexions et réfractions spéculaires pures

Votre application devra être capable de calculer les réflexions et réfractions spéculaires pures, comme vu en cours et en TD. Chaque objet devra spécifier, dans le fichier de spécification de la scène, s'il dispose d'une composante de réflexion spéculaire pure, d'une composante de réfraction spéculaire pure, ou d'une composante de Blinn-Phong. Des paramètres de réflexions spéculaires pures devront être défini dans le même fichier. Pour la réfraction, il sera également nécessaire de définir les indices de réfractions des objets (et il y a toute une subtilité pour savoir si le rayon est à l'intérieur ou à l'extérieur d'un objet).

Lors d'une réfraction ou d'une réflexion spéculaire pure, il convient de relancer un rayon pour connaître la luminance incidente au point considéré. Il faut donc faire une récursion. Un nombre limite de récursion peut être prédéfini dans votre programme (pour éviter d'éventuelles boucles infinies).

L'application du lancer de rayon progressif amélioré récursif

Pour le niveau 3, vous devrez donc réaliser, comme pour le niveau 1 et 2, un exécutable qui prend en entrée un fichier au format défini par vos soins (voir ci-dessus) décrivant la scène et qui doit restituer une image en sortie avec toujours l'option `-o`. Vous devrez rajouter en plus une option `-ps` (pour pixel sampling) permettant de lancer plusieurs rayons par pixels. Ainsi l'exécutable, appelé lray, donc pouvoir répondre à la commande suivante :

```
lray -n 3 -ps 16 -i <mon_fichier.format> -o image.ppm
```

A vous de voir si vous souhaitez directement passer en mode interactif ou en mode offline pour ce niveau 3.

Extensions

La liste présentée ici n'est pas exhaustive et vous pouvez à loisir proposer d'autres extensions. Quoiqu'il en soit, **citez toutes les extensions que vous avez proposé dans votre rapport**. En particulier ces extensions peuvent modifier le format de fichier d'entrée et ceci doit être aussi explicité dans votre rapport. Pour chaque extension, on indique pour indication les points que cela représente

- Extensions des formats d'entrée et de sortie : plutôt qu'une sortie en image au format ppm, on peut sortir des images au format jpg ou png (1 pt)
- Des algorithmes de création procéduraux de scènes complexes (2 pt)
- Texture sur les objets : les textures peuvent être lue pour récupérer par exemple la couleur ou les coefficients de réflexion diffuse (2 pt)
- D'autres types de lumière comme la lumière de type spot qui éclaire depuis un point vers un cône donné (axe + angle) ou la lumière directionnelle (1 pt)
- D'autres modèles d'éclairage comme Ward ou Cook-Torance. (1 pt)
- Des structures de données accélératrices pour les tests rayons – objets canoniques. (4 pts)

Rendu du projet

Éléments à rendre

Vous devrez rendre pour une date et des modalités qui vous seront communiquées ultérieurement les éléments suivants :

- Votre code source sous la forme d'une archive compressée au format .tgz ou .tar.gz. Le format rar n'est pas autorisé. Le code se déploiera dans un répertoire contenant le nom de famille des différents membres du binôme (ou éventuellement monôme / trinôme).
  OU
- Un lien vers votre répertoire GIT en ligne (par exemple sur github ou tout autre site de ce type).

De plus :

- Vous rendez un rapport **succinct** (2-5 pages max) décrivant le moyen de compiler et d'exécuter le code (et notamment toutes la partie IHM) ainsi que vos résultats (timing, screenshot).

Votre projet devra respecter les considérations techniques ci-dessous. Le non respect des considérations techniques et des éléments vu ci-dessus impactera la note sur 2 dédiée au doublon rapport/projet et, en cas « d'abus », peut même entraîner des pénalités.
Notez qu'un tel travail entraînera une bonne note correcte (max 18 donc). Mais pour aller plus loin (dans le « fun » & la note), vous pourrez proposer les extensions vu dans la section ***Extensions*** ci-dessus.

Considérations techniques

- Votre code compilera sous linux avec les bibliothèques autorisées (voir ci-dessous).
- Les bibliothèque utilisables, en plus des bibliothèques standard du C & du C++, seront : GLEW, glut, GL & GLU, SDL (1 ou 2), SFML ASSIMP, ainsi que GLM et, les bibliothèques/outils/codes vu en TD. Vous pouvez également utiliser des bibliothèques permettant le chargement de format de type XML ou JSON, ainsi que d'interpréter la ligne de commande, mais vous devrez les compiler directement via votre application ou clairement les indiquer dans le rapport afin de nous permettre de les installer. Enfin, il est possible d'utiliser d'autres bibliothèques mais seulement avec mon accord : il suffit alors de m'envoyer un mail (biri@u-pem.fr).