Département d'Informatique et de Recherche Opérationnelle
Université de Montréal

Pierre Poulin
Bureau : 2389
poulin@iro.umontreal.ca

IFT3355 : Infographie
Hiver 2011
Travail Pratique #2 : 20%

Disponible :	Jeudi 17 février	
Remise :	Jeudi 17 mars	au début de la démo
équipes :	Deux étudiant(e)s	
Questions :	dift3355@iro.umontreal.ca	

1 Description

Le lancer de rayons est une technique populaire pour le rendu d'images de haute qualité car il permet de simuler plus naturellement que d'autres techniques de rendu certains phénomènes comme la réflexion et la réfraction. Dans ce travail, vous devez implémenter le lancer de rayons récursif. En particulier, vous devez :

1. calculer les rayons partant de l'oeil pour une caméra orthographique et perspective.
2. calculer l'intersection entre un rayon et divers objets dans leur espace canonique : cylindre, cube et maillage de triangles.
3. calculer l'illumination locale selon le modèle suivant :

$$L_{p\lambda} = L_{a\lambda}k_{a\lambda}S_\lambda + \sum_{i=1}^{M}\left[\frac{L_{l\lambda i}}{r_i^2}\Big(k_{d\lambda}S_\lambda(N\cdot L_i) + k_{s\lambda}(m_\lambda S_\lambda + (1-m_\lambda))(N\cdot H_i)^n\Big)\right] + k_{s\lambda}L_{r\lambda} + k_{t\lambda}L_{t\lambda}$$

pour $\lambda \in \{R, G, B\}$ et où
- $L_{a\lambda}$ est la lumière ambiante de la scène
- S_λ est la couleur de la surface
- $k_{a\lambda}$, $k_{d\lambda}$, $k_{s\lambda}$ et $k_{t\lambda}$ sont respectivement les coefficients de réflexion ambiante, diffuse, spéculaire et de transmission
- M est le nombre de lumière
- $L_{l\lambda i}$ est l'intensité de la i^{e} lumière
- r_i est la distance entre la lumière et le point illuminé
- N est la normale à la surface
- L_i est la direction de la lumière
- m_λ est le coefficient de réflexion métallique
- H_i est le vecteur bisecteur entre la direction de l'oeil et L_i (selon le modèle de réflexion spéculaire de Blinn)
- n est le coefficient de rugosité
- $L_{r\lambda}$ est la lumière réfléchie par miroitement

- $L_{t\lambda}$ est la lumière transmise par réfraction (ou réflexion totale interne, s'il y a lieu) ; chaque matériau a son propre indice de réfraction
- la profondeur maximum de récursion pour la réflexion et la réfraction est donnée en paramètre au programme.

4. déterminer l'ombrage et l'incorporer dans le modèle d'illumination locale.
5. simuler l'effet de profondeur de champ de la caméra. Pour cette partie, vous aurez à générer plusieurs rayons par pixel, ce qui permettra de les réorienter aléatoirement de telle sorte que plus un objet est loin du plan focal, moins il aura de chance d'être frappé au même endroit par tous les rayons. L'ensemble des rayons lancés par pixel représente donc un échantillonnage des couleurs de la scène qui sera moyenné pour donner la couleur finale au pixel. Donc plus un objet est loin du plan focal, plus son image sera floue.

   Pour ce faire, vous disposez de deux propriétés de la caméra, soit sa distance focale et la grosseur de la lentille. La distance focale définit à quelle distance les objets apparaîtront de façon net et la grosseur de la lentille définit la variance du flou par rapport à la distance focale.

   Comme expliqué plus haut, pour chaque pixel, vous aurez à générer plusieurs rayons. Cette quantité est donnée en paramètre au programme. Pour générer chacun de ces rayons, vous aurez à créer un rayon passant par le centre du pixel et noter le point intersectant le plan focal. En modifiant aléatoirement l'origine du rayon, nous obtenons un nouveau rayon dont l'origine est perturbée et dont la direction pointe vers le point d'intersection sur le plan focal. Cette perturbation se fera dans un cercle de rayon défini par la grosseur de la lentille et perpendiculaire à la direction `Front` de la caméra. Cette façon de faire fera en sorte qu'un objet situé sur le plan focal sera toujours intersecté au même endroit peu importe la perturbation, ce qui resultera en une image nette.

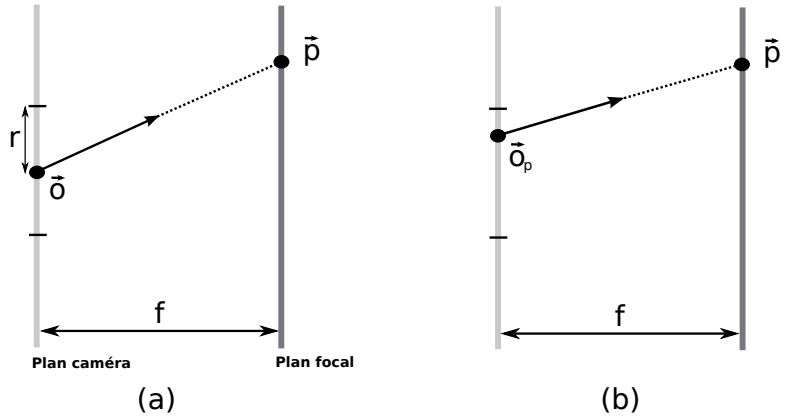

FIGURE 1 – Perturbation d'un rayon pour la profondeur de champ. f est la distance focale, $\vec{o}$ est la position de la caméra, r est associé à la grosseur de la lentille et $\vec{p}$ est le point d'intersection sur le plan focal. (a) Rayon original et (b) rayon perturbé avec $\vec{o}_p$ la nouvelle origine du rayon et dont la direction pointe vers $\vec{p}$.

2 Support au développement

Nous vous fournissons un code de base que vous *devez* compléter. Si vous modifiez l'interface, vous devez fournir les mêmes fonctionnalités de base afin de faciliter la correction. *A priori*, vous n'avez qu'à

modifier les endroits avec des commentaires pour `IFT3355` dans certains fichiers. Suivez ces commentaires pour savoir quoi mettre où. Vous pouvez bien entendu ajouter de nouvelles fonctions, méthodes ou classes. L'archive du code est disponible au chemin ∼`dift3355/pub/tp/tp2/tp2.tar.gz`. Pour extraire l'archive, faites ceci :

```
% tar zxf ~dift3355/pub/tp/tp2/tp2.tar.gz
```

Un exécutable (∼`dift3355/pub/tp/tp2/prototype`) vous permet d'avoir un aperçu du comportement que devrait avoir votre programme.

Pour compiler, utilisez la commande `make debug`. Si vous voulez compiler la version optimisée, exécutez plutôt `make release`. Pour exécuter le programme, la commande est :

```
% ./bin/debug/tp2     # si compilé en mode debug
```

ou

```
% ./bin/release/tp2   # si compilé en mode release
```

Dans la base fournie, les noeuds de la scène (classe `Object`) sont regroupés ensemble dans la classe `Scene`. Chaque noeud contient un système de coordonnées locales et un objet. Chaque objet contient une géométrie et un matériau. La géométrie est essentiellement de deux types : maillage (`Mesh`) ou implicite (`Box`, `Cylinder`, `Plane`, `Sphere`). Un maillage est simplement un ensemble de triangles, alors qu'une primitive implicite est définie par une équation quelconque. Toute géométrie est représentée dans son système de coordonnées locales, sans transformation. Vous devrez en tenir compte lors du calcul d'intersection. Notez qu'une même géométrie peut être partagée par des objets différents. Les primitives implicites sont définies ainsi :

$$\begin{array}{rl}
\text{sphère :} & x^2 + y^2 + z^2 = r^2 \\
\text{plan :} & y = 0,\ x \in [-\frac{s_x}{2}, \frac{s_x}{2}] \text{ et } z \in [-\frac{s_z}{2}, \frac{s_z}{2}] \\
\text{cube :} & x \in [-\frac{s_x}{2}, \frac{s_x}{2}],\ y \in [-\frac{s_y}{2}, \frac{s_y}{2}] \text{ et } z \in [-\frac{s_z}{2}, \frac{s_z}{2}] \\
\text{cylindre :} & x^2 + y^2 = r^2 \text{ et } z \in [-\frac{h}{2}, \frac{h}{2}]
\end{array}$$

Pour l'illumination, les informations nécessaires se retrouvent dans les classes `Light` (pour les lumières), `Material` (pour les matériaux) et `Scene`. Cette dernière, en plus de tous les noeuds, contient aussi toutes les lumières, l'illumination ambiante (le paramètre L_a du modèle) et la couleur du fond. Chaque lumière possède une intensité et une position dans l'espace et est de type ponctuel. Les matériaux, quant à eux, englobent le reste des coefficients du modèle d'illumination.

Finalement, le point d'entrée du lancer de rayons est la méthode `RayTracer::RayTrace`. Dans la plupart des fonctions de lancer de rayons, il y a un paramètre de type `Intersection` qui contient les informations sur l'intersection courante. Dans vos calculs, n'oubliez pas de remplir cette structure avec les bonnes informations.

3 Fonctionnalité

Usage : `./bin/debug/tp2 [scene] [options]`
ou
`./bin/release/tp2 [scene] [options]`

Lorsque démarré, il ouvre une fenêtre d'affichage où il est possible de visualiser la scène interactivement en mode OpenGL. La caméra se promène autour de son point d'intérêt. Voici la description des touches :

`Esc`	:	Quitte
`o`	:	Active la caméra orthographique
`p`	:	Active la caméra perspective
`r`	:	Démarre le lancer de rayons

Le résultat du lancer de rayons sera sauvegardé dans le fichier `data/result.png` et sera automatiquement affiché à l'écran.

Les fichiers de description de scène sont en format texte. Consultez les différents fichiers situés dans le dossier `data/scenes` pour avoir des exemples de création de tels fichiers. Les géométries avec maillage se trouvent dans des fichiers séparés. Le format accepté est *obj*, que vous pouvez éditer avec le logiciel de modélisation gratuit *Blender*.

4 Délivrables et rapport du travail

Nous aurons besoin d'une version électronique de votre programme pour l'évaluer. Pour la remise, faites les commandes suivantes :

```
% ssh remise
% cd <repertoire racine du projet>
% make dist                          # Crée l'archive à remettre
% tar zft tp2.tar.gz                 # Vérifie le contenu de l'archive
% remise ift3355 tp2 tp2.tar.gz
% remise -v ift3355 tp2 tp2.tar.gz  # Vérifie la remise
% exit
```

Vous devez aussi écrire un rapport d'environ 3-4 pages qui contiendra :

1. énoncé du problème à résoudre.
2. Description des techniques utilisées. Donnez les formules mathématiques utilisées, et comment vous les avez employées. Nous voulons ici le fondement de vos techniques, pas les détails de la disposition du code.
3. Problèmes reliés aux techniques implémentées, suggestions pour améliorer la solution au problème, extensions possibles.
4. Références.

5 Critères d'évaluation

Section	Critères	Pondération
Code	style, clarté, *efficacité*, résultats	80%
Rapport	esthétique, énoncé, descriptions	10%
Analyse	problèmes, améliorations, extensions	10%

6 Bibliographie

P. Shirley et S. Marschner. Fundamentals of Computer Graphics. 3ème édition, AK Peters, 2009.