SIM202 : Synthèse d'images

On s'intéresse à la synthèse d'images numériques 2D à partir de scènes 3D. Cette branche de l'informatique a connu un développement important avec l'accroissement de la puissance des ordinateurs pour des applications variées comme le développement de jeux vidéos, l'animation ou encore la modélisation industrielle.

Les algorithmes que nous allons voir sont généralement implémentés sur des cartes graphiques (GPU) au travers de la librairie graphique OpenGL par exemple. Cependant ce projet se veut une introduction à la synthèse d'images et on se limitera à une version séquentielle (sur CPU) des algorithmes considérés, une implémentation parallèle sur carte graphique dépassant largement le cadre de ce cours.

1 Lancer de rayons

Il existe de nombreuses techniques et algorithmes permettant de déterminer quelles sont les surfaces à afficher. L'algorithme de visualisation que nous utiliserons est l'algorithme de lancer de rayons. Il permet de traiter un grand nombre de cas et produit un rendu assez réaliste.

Figure 1: Exemple de rendu réaliste possible avec un algorithme de lancer de rayons (source: *wikipedia*).

1.1 Modélisation de la scène

Dans le cadre de ce projet on ne considère pas la partie modélisation de scènes 3D. On suppose que la géométrie de la scène 3D est décrite par un

maillage surfacique de triangles. Ce type de maillage est relativement facile à générer et offre une grande souplesse permettant de décrire des formes géométriques complexes.

La scène sera observée à travers une fenêtre d'observation que l'on décompose en une grille régulière, les pixels, constituant l'image. Cette fenêtre est située entre la scène et l'observateur. On peut distinguer deux représentations :

- Une représentation en perspective en situant l'observateur à distance finie de la fenêtre, on parlera de projection conique de la scène;
- Une représentation utilisant la projection orthogonale de la scène sur la fenêtre, pour laquelle on peut considèrer l'observateur comme étant à l'infini.

On pourra se contenter de cette deuxième représentation, plus facile à mettre en œuvre, dans un premier temps.

1.2 Principe de l'algorithme

L'idée générale de l'algorithme de lancer de rayons est de s'inspirer de la physique de l'optique géométrique pour calculer l'éclairage de la scène. On utilise le principe de réversibilité (ou loi du retour inverse) pour remonter le chemin des rayons lumineux du point d'observation aux sources lumineuses en passant par les objets éclairés.

Dans un premier temps on supposera tous les objets présents constitués de matériaux opaques. Les sources lumineuses seront considérées ponctuelles et non visibles (c'est à dire à l'extérieur du champ de l'image). Le principe général de l'algorithme est le suivant : pour chaque pixel P_{ij} de l'image (boucle sur les pixels)

1. On lance un rayon (demi-droite), appelé rayon primaire, du point d'observation O qui passe par le pixel P_{ij} considéré et traverse la scène;
2. On boucle sur tous les triangles du maillage pour déterminer l'ensemble des triangles ayant une intersection avec le rayon primaire;
3. Le triangle dont le point d'intersection, noté A, avec le rayon primaire est le plus proche du point d'observation est le triangle visible (pour l'observateur);
4. Pour chaque source lumineuse S_k éclairant la scène :
   (a) On lance un deuxième rayon, appelé rayon secondaire, du point A vers le point S_k;
   (b) On boucle sur tous les triangles du maillage pour déterminer si le rayon secondaire intersecte un autre triangle du maillage :

- Si on détecte une intersection, on arrête la boucle, le point A ne reçoit aucune contribution de la source S_k;

(c) On calcule la contribution à l'intensité lumineuse de la source S_k au point A, comme indiqué dans la section suivante.

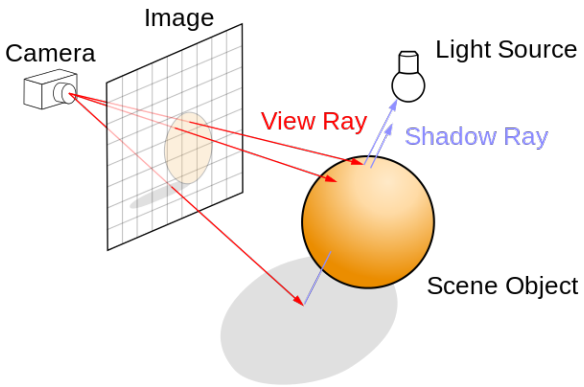

Figure 2: Principe du lancer de rayons (source: *wikipedia*).

2 Intensité lumineuse

2.1 Calcul

Pour chaque pixel on doit calculer trois intensités lumineuses correspondant aux trois canaux de lumière : rouge, vert et bleu. Chacune de ces intensités est calculée au point A selon une loi empirique comme suit. Étant donné N sources lumineuses d'intensité I_s supposées ponctuelles, le calcul de l'intensité lumineuse I est la somme de l'intensité lumineuse ambiante I_a et de l'intensité lumineuse produite par chaque source. Plus précisément,

$$I = I_a k_a + \sum_{s=1}^{N} I_s \left(k_d \cos\theta + k_r \cos^n \alpha \right), \tag{1}$$

où les coefficients k_a, k_d et k_r sont respectivement les coefficients de diffusion ambiante, de diffusion et de réflexion spéculaire. Ils dépendent du matériau. L'angle θ est l'angle formé par la normale à la surface au point considéré et la direction de la source lumineuse. L'angle α est l'angle entre la direction d'observation et la direction du rayon lumineux réfléchi par le point de la surface considérée. Le paramètre n caractérise la reflexion du matériau. Il existe différentes améliorations de ce modèle, par exemple en considérant une atténuation avec la distance parcourue par les rayons.

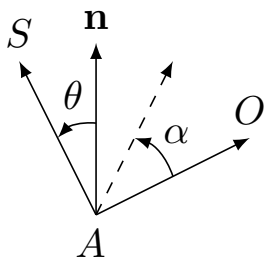

Figure 3: Représentation des angles et directions nécessaires au calcul d'intensité (source: *wikipedia*).

2.2 Méthodes d'ombrage

2.2.1 Ombrage plat

Cette méthode est la plus simple et consiste simplement à remplir chaque triangle visible uniformément de la même couleur en prenant la normale du triangle pour le calcul de l'intensité. Cela permet de limiter les opérations mathématiques nécessaires au rendu. Cependant le résultat est assez pauvre et les arrêtes du maillage sont très visibles.

2.2.2 Ombrage de Gouraud

L'idée est de calculer l'éclairage et la couleur aux trois sommets de chaque triangle. Pour évaluer la normale aux sommets, nécessaire au calcul de l'intensité lumineuse, on pourra considérer la moyenne des normales de chaque triangle autour du point considéré. La couleur à l'intérieur du triangle est ensuite calculée par une simple interpolation linéaire (coordonnées barycentriques).

Cette technique donne un résultat beaucoup plus réaliste, notamment pour les objets courbes rendus par des polygones en donnant l'illusion d'une courbure sur le triangle. Cependant cette méthode n'est pas adaptée pour un éclairage avancé comportant des reflets spéculaires, sauf à considérer des maillages fins donc gourmands en calcul.

2.2.3 Ombrage de Phong

Cette méthode est une variante de la méthode de Gouraud. Au lieu d'interpoler le résultat du calcul d'intensité, l'idée est d'interpoler la normale qui est utilisée dans le calcul. Le résultat est très réaliste.

3 Vers des visualisations plus réalistes

3.1 Optimisation du code

Comme on s'en rendra vite compte au fur et à mesure de l'avancement du projet, la méthode du lancer de rayons est très gourmande en calcul. Les points sensibles sont les boucles sur l'ensemble des triangles du maillage pour déterminer la collision ou non avec un rayon. Le rendu de scènes réalistes nécessitant des maillages constitués de nombreux triangles, le temps de calcul devient rapidement prohibitif. On peut optimiser l'agorithme décrit précédemment en introduisant un KDTree.

Le principe est basé sur un rangement des triangles dans des boites suivant un processus dichotomique. La taille des boites diminue en fonction de la profondeur de l'arbre. La construction de l'arbre est faite de la manière suivante:

- On commence par une boite englobant la scène entière avec l'ensemble des triangles;
- On découpe la boite sur sa plus grande longueur, créant ainsi deux boites filles;
- On distribue ensuite chaque triangle dans la boite correspondante;
- On recommence avec les deux boites filles.

En pratique on s'arrête quand un nombre raisonnable de triangles se trouve dans chaque boite.

Dans l'algorithme de lancer de rayons, le calcul de l'intersection d'un rayon avec les triangles se fait en descendant l'arbre en calculant à chaque étape l'intersection ou non avec la boite englobante.

A noter que la construction de l'arbre peut être relativement coûteuse. La construction sera neanmoins rentabilisée lors du rendu de la même scène en changeant le point d'observation ou la position des sources, qui ne nécessitent pas la reconstruction de l'arbre.

3.2 Optique géométrique

Dans l'algorithme de lancer de rayons décrit précédemment on était resté à une modélisation trés simplifiée de l'optique géométrique. Avec l'optimisation du code utilisant le KDTree, il est maintenant possible de calculer des rendus plus réalistes nécessitant le traitement de la transparence, la réfraction et la réflexion. Afin de calculer ces effets il est nécessaire de considérer pour chaque point A des objets visibles, non seulement les rayons secondaires vers les différentes sources lumineuses, mais également les rayons secondaires réfléchis et les rayons réfractés. Chaque rayon secondaire étant susceptible

de rencontrer d'autres objets, la procédure devient récursive. Une structure de donnée pouvant être utilisée est une structure d'arbre avec à chaque nœud les informations de couleur, d'éclairement et de transparence, les arcs modélisant les rayons. On limitera alors la profondeur de l'arbre à un niveau paramétrable par l'utilisateur afin de limiter les calculs qui deviennent vite prohibitifs. A noter que ce procédé ne permet pas de recréer des effets lumineux dépendant des propriétés ondulatoires de la lumière (interférences, irisations...).

3.3 Pour aller plus loin

- Anti-aliasing par sur-échantillonage : pour éviter les arrêtes dures dans l'image on peut utiliser un sur-échantillonage. L'idée est de lancer plusieurs rayons (4 ou 16 sur une subdivision régulière) pour un même pixel et d'effectuer ensuite la moyenne des intensités calculées.

- On pourra aussi généraliser le code pour pouvoir utiliser d'autres types d'éléments de maillage, comme les quadrangles par exemple.

4 Eléments de conception

Exemples de classes et méthodes à définir, dans un premier temps:

- Classe *Maillage*
  - comportant un vecteur de points et un vecteur de triangles;
  - pouvant être instancié à partir d'un fichier GMSH.

- Classe *Pixel*
  - définissant les trois valeurs RGB (rouge, vert, bleu) en entiers 8 bits non signés;
  - comportant des méthodes de calcul de l'intensité basées sur les trois types d'ombrages.

  On pourra par exemple définir l'opérateur de flux pour cette classe pour faciliter l'écriture sur un fichier.

- Classe *Image*
  - comportant un vecteur (ou matrice) contenant les pixels;
  - permettant l'export dans un fichier dans un format d'image.

  On pourra par exemple définir les opérateurs d'accès à un pixel de l'image, l'opérateur de flux pour l'écriture sur un fichier...

Dans un second temps, il sera nécessaire de développer les classes pour le KDTree. On pourra définir notamment :

- Classe *KDNode*
  - comportant les informations géométriques liées à la boite, un vecteur de triangles et des pointeurs vers les nœuds filles;
  - une méthode de construction des nœuds filles.

5 Organisation du travail

On pourra, à l'issue d'une concertation générale permettant de bien définir les classes et fonctionnalités du code, répartir le travail de la façon suivante :

- une partie sur la gestion du maillage par lecture de fichiers GMSH, et l'écriture des fichiers images de sortie: format BITMAP ou GIF/PNG pour des images animées (observateur ou source lumineuse tournant autour de la scène);
- une partie sur les méthodes d'ombrage et de calcul des intensités lumineuses;
- une partie sur la mise en oeuvre simple de l'algorithme de lancer de rayons.

Lorsque le code sera validé on s'attachera à l'optimiser et le généraliser: KDTree, reflexion et refraction...