Raytracer (3) - Algorithme de raytracing, et sortie graphique

1 Introduction

La première partie du projet a consisté à implanter les classes utiles pour représenter les objets géométriques dans le cadre du *raytracing*, et à calculer le premier objet touché par un rayon. La deuxième partie a doté votre programme d'un parser qui permet à l'utilisateur d'écrire sous forme textuelle les scènes à représenter. L'objectif de cette troisième et dernière partie est de programmer l'algorithme de *raytracing* et de permettre de visualiser les images obtenues.

2 Raytracing

2.1 Notations

On rappelle les notations utilisées dans la partie 2 pour les textures et la lumière.

Textures. Chacun des objets a un attribut pour décrire ses propriétés optiques. On propose le modèle de texture suivant (qu'il est possible d'enrichir) :

- une couleur de surface, qui est le triplet des valeurs pour les trois couleurs primaires rouge, vert et bleu, chaque valeur étant comprise entre 0 et 1 ;
- k_d un coefficient de réflexion diffuse (entre 0 et 1) ;
- k_s un coefficient de réflexion spéculaire (entre 0 et 1) ;
- n un coefficient de Phong (un réel strictement positif).

Sources lumineuses. Chaque source lumineuse est définie par son intensité (entre 0 et 1) et une direction. On considère dans ce cas des sources blanches situées à l'infini. Il est facile de faire des sources de couleur en considérant une intensité par canal RGB.

Lumière ambiante. On considère une lumière ambiante avec une intensité entre 0 et 1. On peut là aussi la définir par canal RGB.

2.2 Algorithme de raytracing

La caméra est placée sur l'axe z. Étant donnée une distance $d > 0$ de la caméra à l'origine (la caméra est placée en $(0,0,d)$), un angle horizontal α du champ de vision, une largeur l de l'écran et une hauteur h de l'écran, on imagine l'écran parallèle au plan $x-y$, situé à la distance $l/(2 * tan(\alpha/2))$ de la caméra en direction de l'axe z.

Pour construire l'image on calcule pour chaque point de l'écran la couleur à afficher. On trouve cette couleur en envoyant un rayon *cam* à partir de la caméra qui traverse le point de l'écran. De manière générale on définit la couleur C_r "vue" par un rayon r comme

- Si le rayon ne touche aucun objet alors la couleur est noire.

– Si l'objet touché le premier est o alors la couleur C_r à afficher est calculée comme

$$C_r = k_d I_a C + k_d \sum_{j=1}^{ls} (\vec{N} \cdot \vec{L_j}) I_j C + k_s \sum_{j=1}^{ls} (\vec{N} \cdot \vec{H_j})^n I_j C + k_s C C_{refl(r,o)}$$

où les sources lumineuses *visibles du point d'impact* sont énumérées $1, \ldots, ls$.

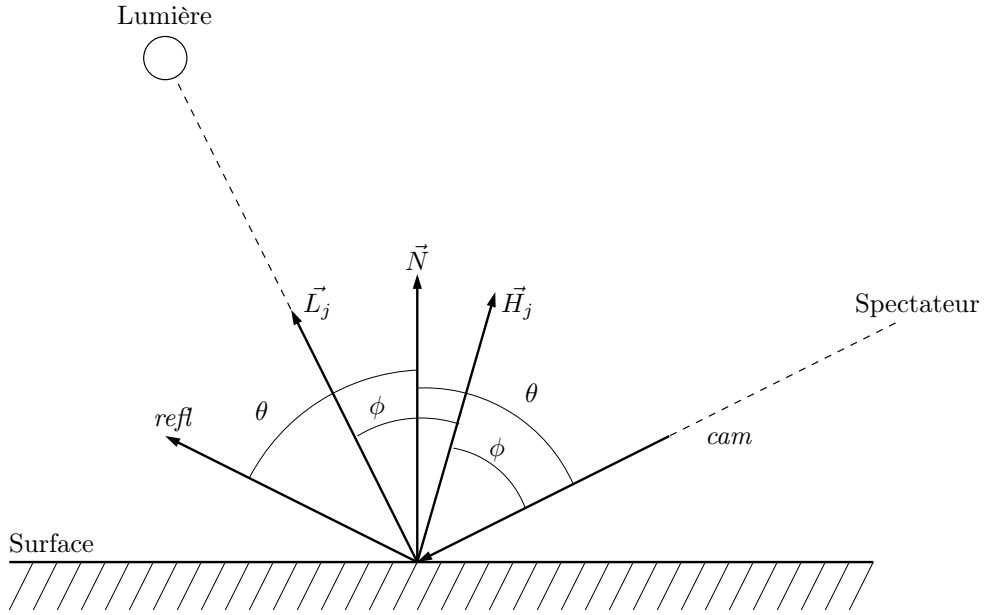

FIG. 1 – *Vecteurs liés au calcul de l'illumination.*

C	=	couleur de surface de l'objet o
k_d	=	coefficient de réflexion diffuse de l'objet o
k_s	=	coefficient de réflexion spéculaire de l'objet o
n	=	exposant Phong de l'objet o
I_a	=	intensité de la lumière ambiante
I_j	=	intensité de la j-ème source lumineuse
$\vec{N}$	=	normale à la surface de o au point d'impact
$\vec{L_j}$	=	direction de la j-ème source lumineuse
$\vec{H_j}$	=	direction bissectant $-cam$ et $\vec{L_j}$

Toutes les directions sont des vecteurs unitaires. La direction d'une source lumineuse est définie comme le vecteur qui pointe vers la source lumineuse. Notez que le vecteur $\vec{H_j}$ pointe vers l'extérieur de l'objet o (voir Figure 1).
Le rayon *refl(r,o)* est obtenu par réflexion du rayon r sur la surface de l'objet o. Le calcul de C_r est donc donné par une règle récursive. *Puisqu'il est possible que la récurrence ne se termine pas, on impose une borne artificielle au nombre d'appels récursifs.*

3 Travail à réaliser

3.1 Le rapport

Le rapport doit contenir les parties suivantes :

1. un descriptif du raytraceur, mais à un niveau d'abstraction assez haut pour que cela soit facilement compréhensible. Précisez les fonctionnalités, vos choix de programmation, éventuellement les points qui vous ont posé problème.
2. une partie rédigée pour le second rapport. *Vous pouvez rajouter à votre parser des fonctionnalités non présentes dans le rapport2, dans ce cas rajoutez les jeux de test correspondant;*
3. des jeux de test pour le raytracer (décrivez les images censées être obtenues et réellement obtenues),
4. votre code commenté.

Précisez bien quelles fonctionnalités vous traitez : mettez en valeur les options supplémentaires, et n'oubliez pas de parler également des limites du programme notamment si vous avez constaté des anomalies lors des tracés.

Le projet est à rendre au plus tard le **27 mai 2007**.
Les soutenances sont prévues le **31 mai 2007**.

3.2 Travail complémentaire

Si vous le souhaitez vous pouvez inclure des fonctionnalités supplémentaire à votre *raytracer* :

1. raytracing
   - transparence
   - sources lumineuses ponctuelles (et non à l'infini)
   - objets plus généraux
   - *anti-aliasing*
   - ...
2. parser (voir partie 2)
3. divers
   - optimiser la vitesse de tracé
   - options à l'éxécution (résolution de l'image, nom du fichier de sortie, ...)
   - gérer différents formats d'image