



ÉCOLE CENTRALE LYON

MOS 8.5
INFORMATIQUE GRAPHIQUE
RAPPORT

Rapport Informatique Graphique

Élèves :
Amaury GIARD

Enseignant :
Nicolas BONNEEL

Table des matières

1 Présentation du projet	2
2 Développement	2
2.1 Sphère	2
2.2 Miroirs et transparence	3
2.3 Fresnel	3
2.4 Flou de caméra	4
2.5 Mesh et Bounding Box	4
2.6 BVH	5
2.7 Lissage de Phong	5
2.8 Textures	6

1 Présentation du projet

Nous cherchons dans ce projet à pouvoir faire un rendu graphique d'une scène de notre choix avec des méthodes de ray-tracing. Le projet est codé en C++. Nous étudierons d'abord à faire le rendement d'une scène contenant des sphères, pour réaliser un premier rendement visuel auquel on pourra ajouter des améliorations pour accroître le réalisme de la scène. Nous nous intéresserons ensuite aux Mesh pour générer des objets plus complexes.

2 Développement

2.1 Sphère

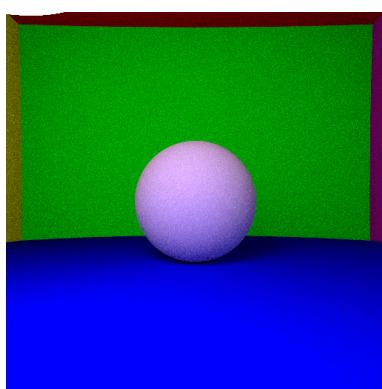
On commence par le rendu visuel d'une sphère. Ceci nous permet d'avoir une première figure simple à modéliser, que l'on utilisera pour l'arrière-plan de notre scène, et qui nous permettra de faire d'observer visuellement les différentes améliorations graphiques que nous implémenterons.

On implémente d'abord une classe Vector qui nous servira de base pour tous nos vecteurs tridimensionnels. Elle est utilisée pour construire la classe Ray qui modélise chacun des rayons de lumière de notre modélisation, définie à partir d'un point et d'une direction.

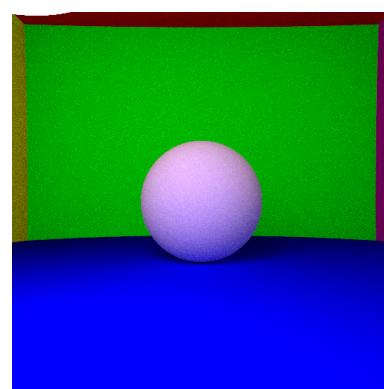
On implémente désormais la classe Sphere, qui prend pour argument un centre, un rayon et un albedo. Elle contient la méthode intersect qui détermine si un rayon passe par la sphère en solvant l'équation $\|X - C\|^2 = R^2$.

On implémente ensuite la classe Scène qui nous permettra de faire un rendement visuel contenant plusieurs objets dans une même image. Elle contient des méthodes permettant d'ajouter des objets à la scène, d'appeler les méthodes intersect de ces objets et de récupérer la couleur des pixels de l'image générée.

On instantie alors une scène à laquelle on ajoute plusieurs objets. Tout d'abord une sphère qui servira de source de lumière (on avait commencé avec des rayons tous issus d'un même point mais une source large permet de faire des ombres lisses, plus réalistes). On ajoute ensuite des très grosses sphères qui nous serviront d'arrière-plan. Enfin, on ajoute une sphère au milieu pour générer un rendu.



(a) 50 rays, 3 reflections, 14s



(b) 100 rays - 7 reflections - 55s

La couleur est déterminée en sommant les couleurs directe et indirecte des objets. Pour chaque rayon, la couleur directe vient de l'albedo de l'objet étudié. La lumière indirecte

est déterminée à l'aide du rayon réfléchi sur l'objet. On observe que le nombre de rebonds considérés augmente grandement le temps de calcul. Les objets sont considérés comme diffus.

Pour avoir un rendu le plus réaliste possible, on augmente le nombre de rayons par pixel et le nombre de rebonds que ces rayons peuvent effectuer. On moyenne ensuite les couleurs obtenues pour chacun des rayons afin de déterminer la couleur de l'image générée.

Pour une image plus réaliste, nous avons également implémenté l'anti-aliasing (contours plus lisses) et la correction gamma (contraste moins fort).

2.2 Miroirs et transparence

En ajoutant des booléens aux arguments de la classe Sphère, on peut créer 2 nouveaux types d'objets : les miroirs sphériques et les sphères transparentes.

Ces nouveaux arguments nous permettent d'orienter les rayons réfléchis par les objets de la scène, soit par réfection, soit par réfraction. Pour une sphère en plastique, on insère une seconde sphère dans la première légèrement plus petite.

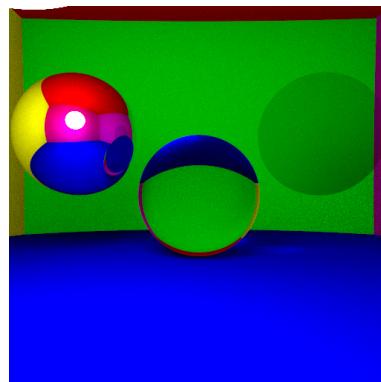


FIGURE 2 – Miroir(gauche), Verre(milieu), Plastique(droite)

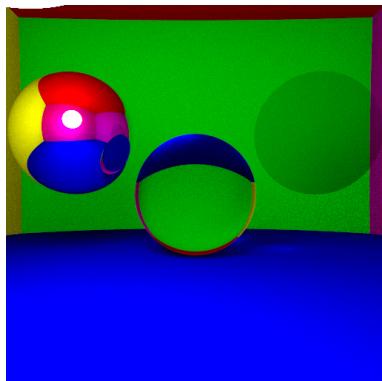
2.3 Fresnel

Pour un rendu plus réaliste, on implémente la loi de Fresnel : ??

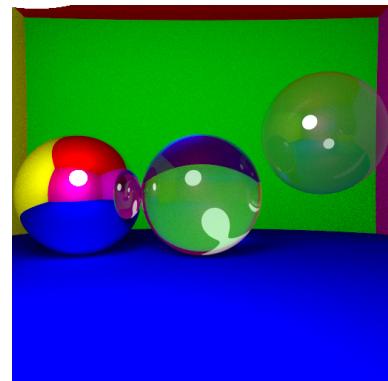
$$k_0 = \frac{(n_1 - n_2)^2}{(n_1 + n_2)^2}$$

$$R = k_0 + (1 - k_0)(1 - |\langle N, \omega_i \rangle|)^5$$

$$T = 1 - R$$



(a) Sans Fresnel



(b) Avec Fresnel

L'ajout de cet effet ne modifie pas significativement le temps de rendement.

2.4 Flou de caméra

Pour implémenter le flou associé à une caméra, on fait virtuellement passer les rayons émis par un dispositif optique simulant la caméra, avec une ouverture et une distance de mise au point.

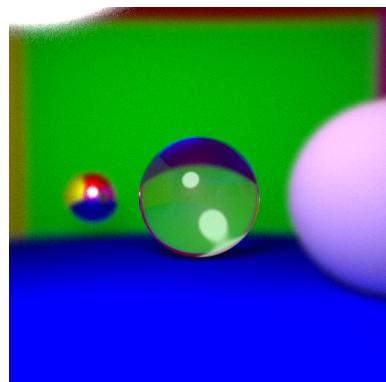


FIGURE 4 – Flou de caméra

2.5 Mesh et Bounding Box

On essaie désormais d'importer un triangle mesh dans notre scène. On vérifie alors l'intersection des rayons avec chacun des triangles du mesh, selon l'équation :

$$(e_1 \ e_2 \ -u) \begin{pmatrix} \beta \\ \gamma \\ t \end{pmatrix} = (O - A)$$

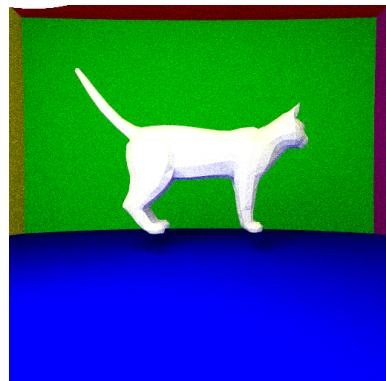
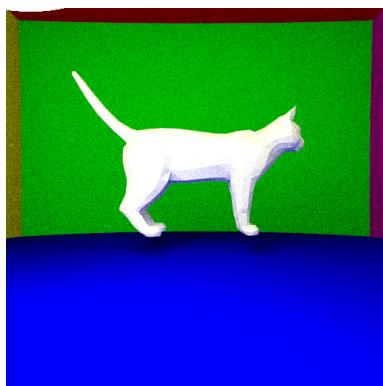


FIGURE 5 – Mesh d'un chat - 50 rayons - 7 rebonds

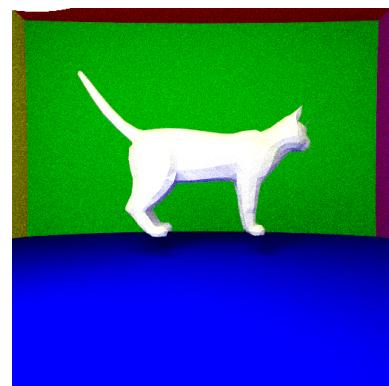
Pour accélérer le temps de calcul, on implémente une Bounding Box. On ne considère pas l'intersection des rayons avec les triangles du mesh si le rayon ne rentre pas dans une boîte contenant le mesh.

2.6 BVH

Cette méthode utilisant uniquement une BBox demeure toutefois très lente. On implémente donc un Bounding Volume Hierarchy, qui est un arbre binaire de BBox, construit de façon à ce que l'union de deux noeuds fils corresponde au noeud parent. On associe alors chacun des triangles du mesh à la plus petite BBox le contenant. Pour chaque rayon, on vérifie s'il passe dans une BBox, si oui on vérifie pour les fils de cette BBox et on itère.



(a) BBox - 18min



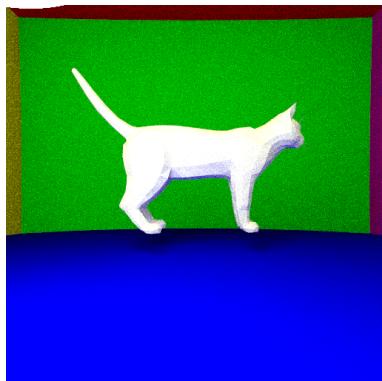
(b) BVH - 1min 30s

Le temps de calcul est ainsi bien diminué, et il n'y a aucun perte visuelle.

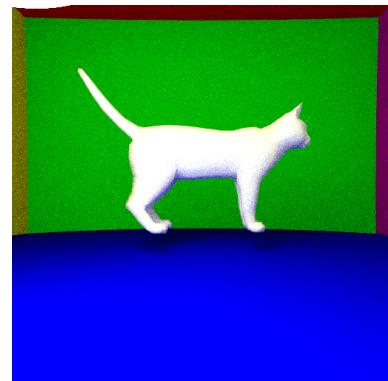
2.7 Lissage de Phong

Pour lisser les angles liés à la construction du mesh, on utilise un lissage de Phong :

$$\hat{N}(P) = \frac{\alpha(P)N_A + \beta(P)N_B + \gamma(P)N_C}{\|\alpha(P)N_A + \beta(P)N_B + \gamma(P)N_C\|}$$



(a) Sans lissage



(b) Avec lissage de Phong

2.8 Textures

En utilisant un fichier image, on peut ajouter des textures sur le mesh. On utilise pour ce faire l'indice de chacun des triangles, auquel on peut associer un point du fichier image.

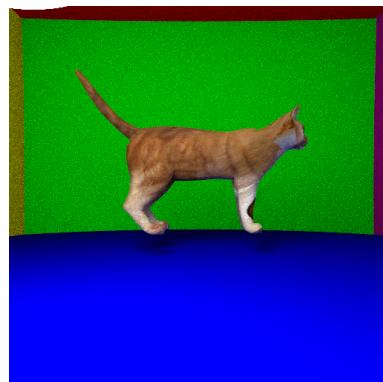


FIGURE 8 – Mesh avec les textures - 1min

On observe un artefact visuel sur la patte avant droite du chat, avec un peu de recherche, on se rend compte que les bonnes textures ne sont pas toujours associées au bon point du mesh.

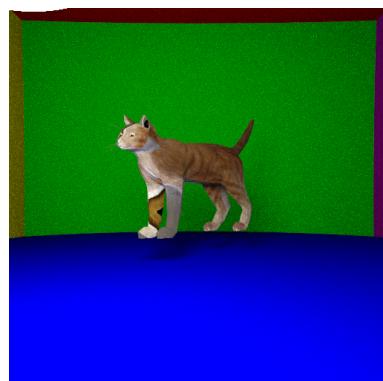


FIGURE 9 – Erreur de texture

En effectuant une rotation sur le mesh, on observe qu'il y a une erreur de texture. Certains points du mesh semblent ne pas être associés au bons points du fichier image.

Après vérification, il ne s'agit a priori pas d'une erreur provenant du fichier. Je n'ai pas réussi à corriger cette erreur dans mon code.