A Little Raytracer

Loïs Paulin

Décembre 2017

Résumé

Dans ce rapport nous présentons le travail effectué au cours du projet de Méthode Mathématique pour la Synthèse d'Images. Ce projet consistait en l'implémentation d'un lanceur de rayon permettant de rendre des sphères avec des matériaux diffus, spéculaires et transparents ainsi que du calcul d'éclairage indirect en utilisant une méthode d'intégration de Monte-Carlo. Notre lanceur de rayon implémente aussi des fonctions de rendu de modèles 3D constitués de faces triangulaires et chargés à partir de fichier obj.

1 Structure du code

Le projet a été séparé en onze classes représentants les différents éléments constituants du lanceur de rayon.

1.1 La classe Image

La classe *Image* sert à stocker et enregistrer une image au format png selon la méthode donnée dans le StackOverflow donné dans le sujet.

1.2 Les classes d'objets

Les objets placés dans la scène sont représentés par la classe abstraite *Object* qui possède les fonctions virtuelles pures de calcul d'intersection et de calcul de couleur d'une intersection. Ces fonctions travaillent avec la classe Intersection qui stocke les informations liées à l'intersection (position, normale, objet intersecté, distance de l'origine du rayon, ...). Les classes héritant d'*Object* sont *Sphere* et *Mesh* qui implémentent leurs fonctions d'intersection. *Mesh* permet d'utiliser deux méthodes de calcul des normales. Une avec une normale par face et la seconde avec des normales par interpolées depuis les sommets pour avoir un aspect lissé. Pour l'instant le choix entre ces deux méthodes est codé en dur et à commenter/décommenter dans Mesh.cpp.

1.3 Les classes de matériaux

Les classes de matériaux sont représentés par la classe abstraite *Material* qui possède la fonction abstraite pure *getCol* qui calcule la couleur renvoyé par l'intersection d'un rayon avec ce matériel. Les matériaux implémentés qui héritent tous de *Material* sont *DiffuseMat*, *SpecularMat* et *TransparentMat*. Ils implémentent chacun leur propre version de *getCol*.

1.4 Les classe de scène

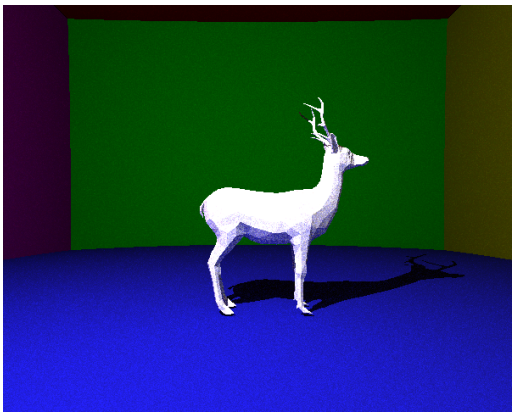
La classe *Camera* sert à décrire une caméra pouvant regarder dans toutes les directions et la classe *Light* sert à décrire une source de lumière ponctuelle. La classe *Scene* contient un tableau des lumières, un tableau des objets, la couleur du ciel et la caméra servant au rendu.

1.5 Les classes Vec3 et Ray

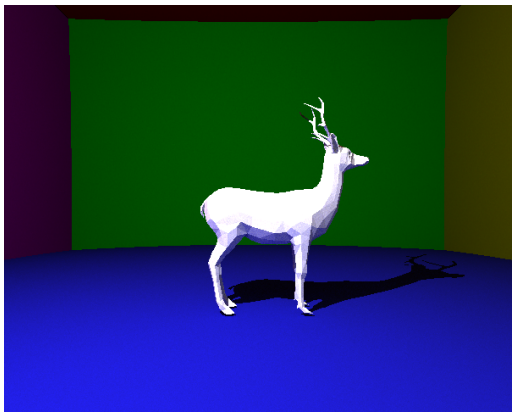
Les classes Vec3 et Ray implémentent tous les outils mathématiques nécessaires.

2 Résultats

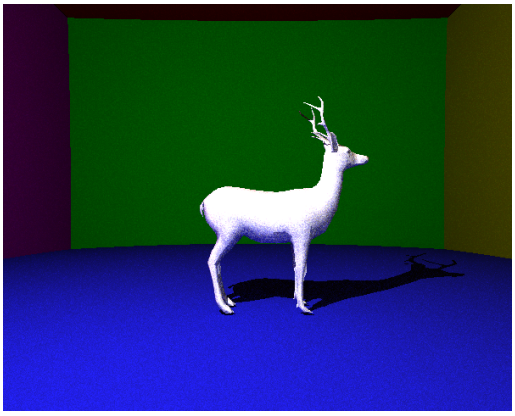
Voici une série de scène rendues permettant d'observer tous les objets et matériaux à différents nombre d'échantillons par pixels. Les rendus de maillages sont très longs, l'ajout d'une structure d'accélération des calculs d'intersection est nécessaire au fonctionnement du lancé de rayon sur des maillages complexes.



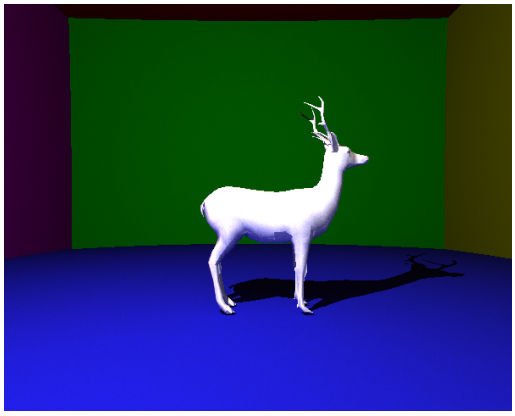
(a) Cerf diffus 10 spp (2m40)



(b) Cerf diffus 100 spp (26m)

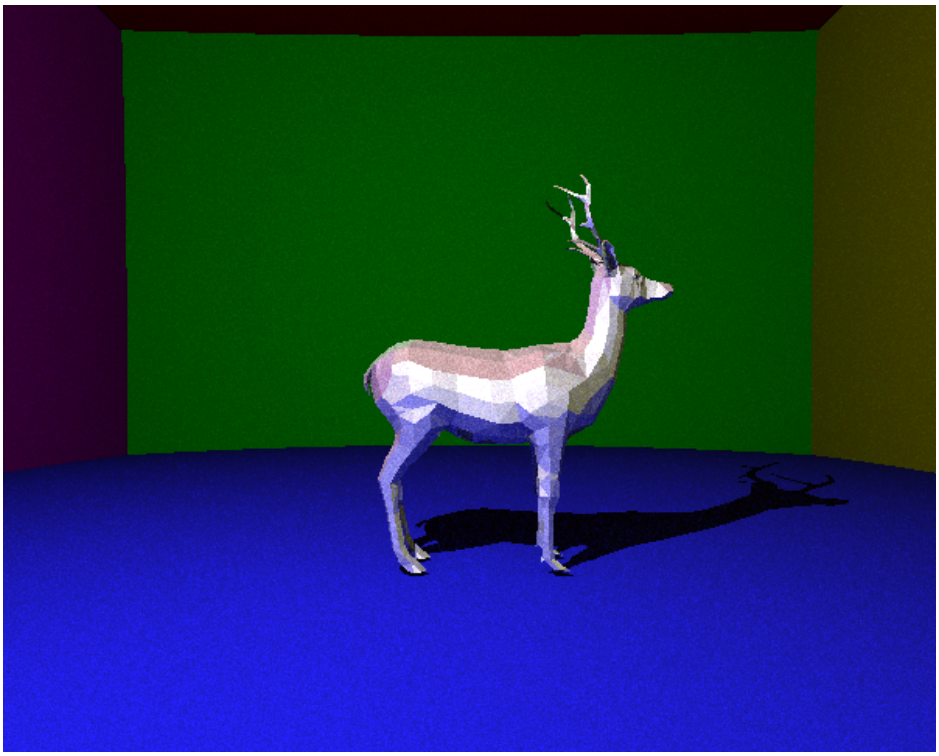


(c) Cerf diffus 10 spp (2m45)

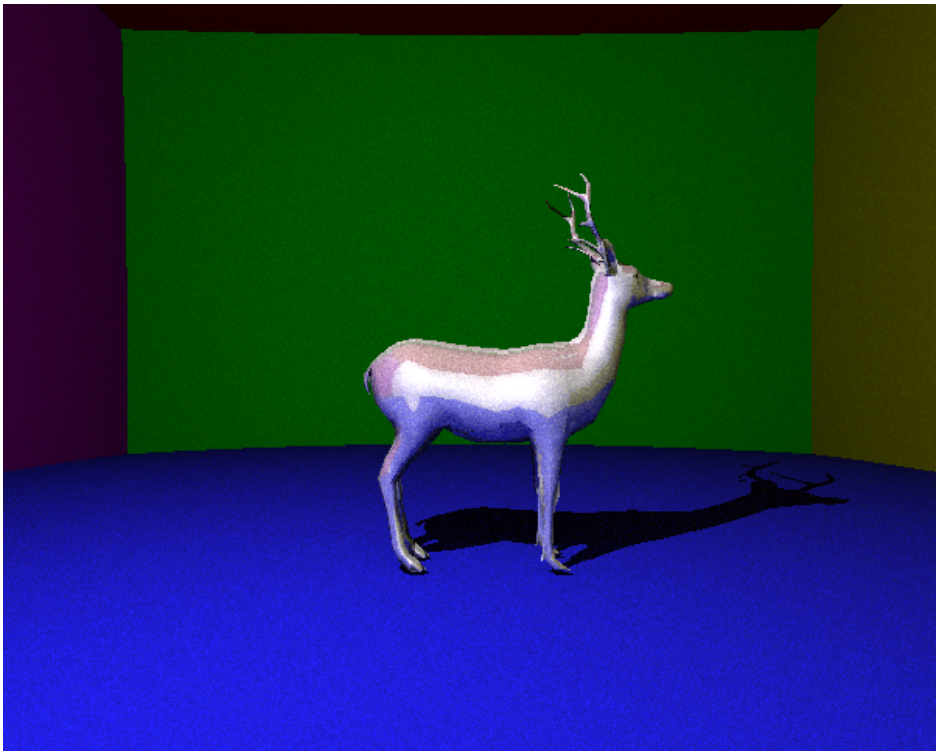


(d) Cerf diffus 100 spp (30m)

FIGURE 1 – Rendus de maillages (1500 sommets) diffus à 5 rebonds. Comparaison normales "douces"/"dures"

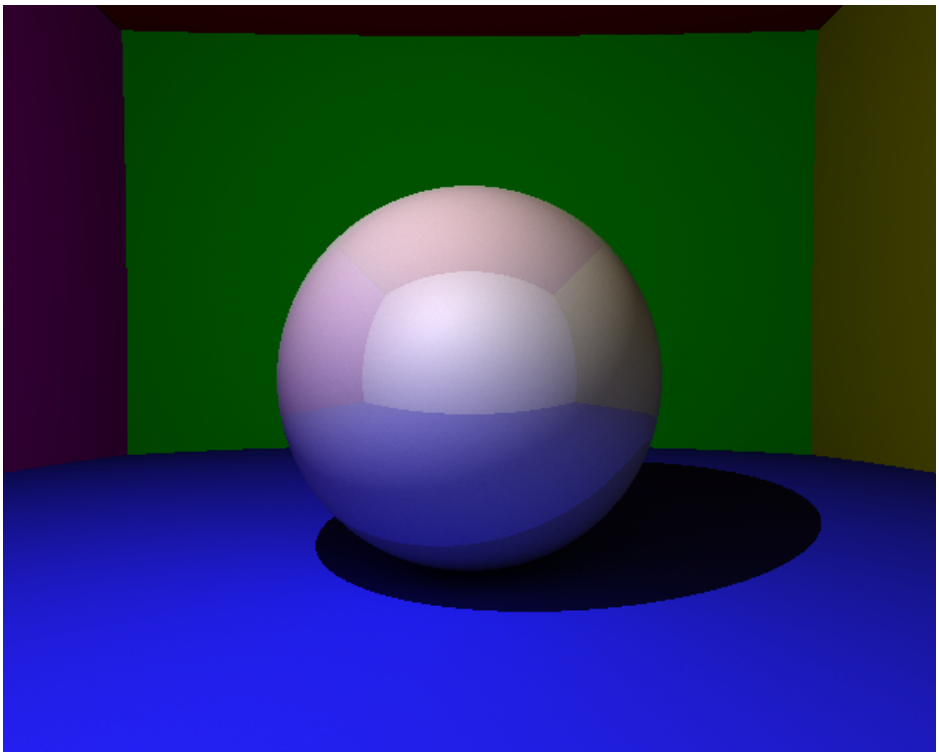


(a) Cerf spéculaire 10 spp (2m39)

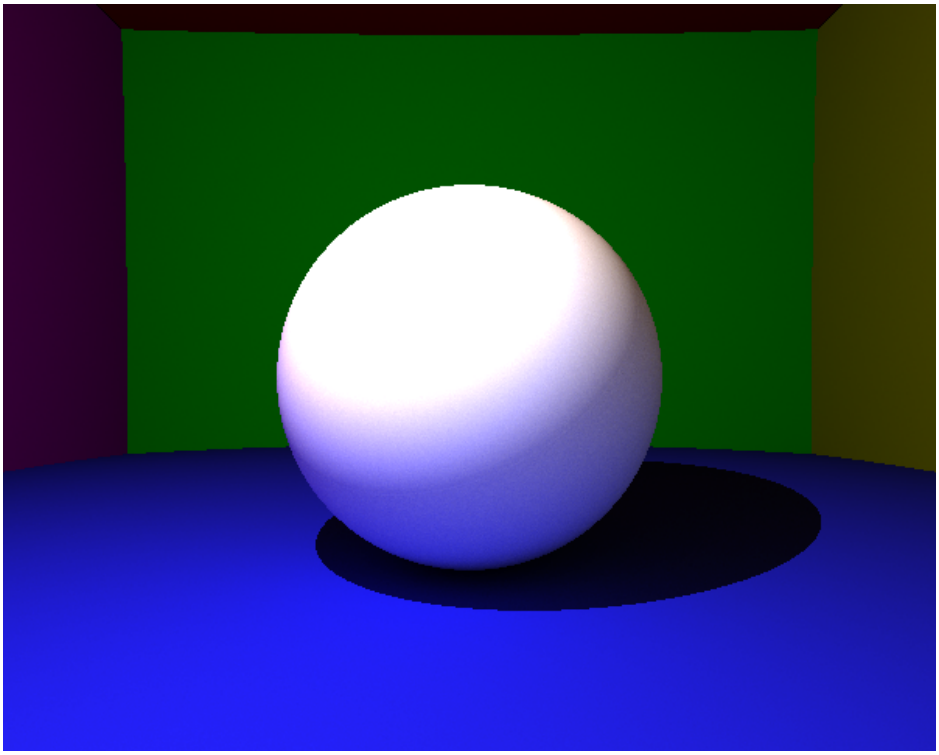


(b) Cerf spéculaire 10 spp (2m42)

FIGURE 2 – Rendus de maillages (1500 sommets) spéculaires à 5 rebonds. Comparaison normales "douces"/"dures"

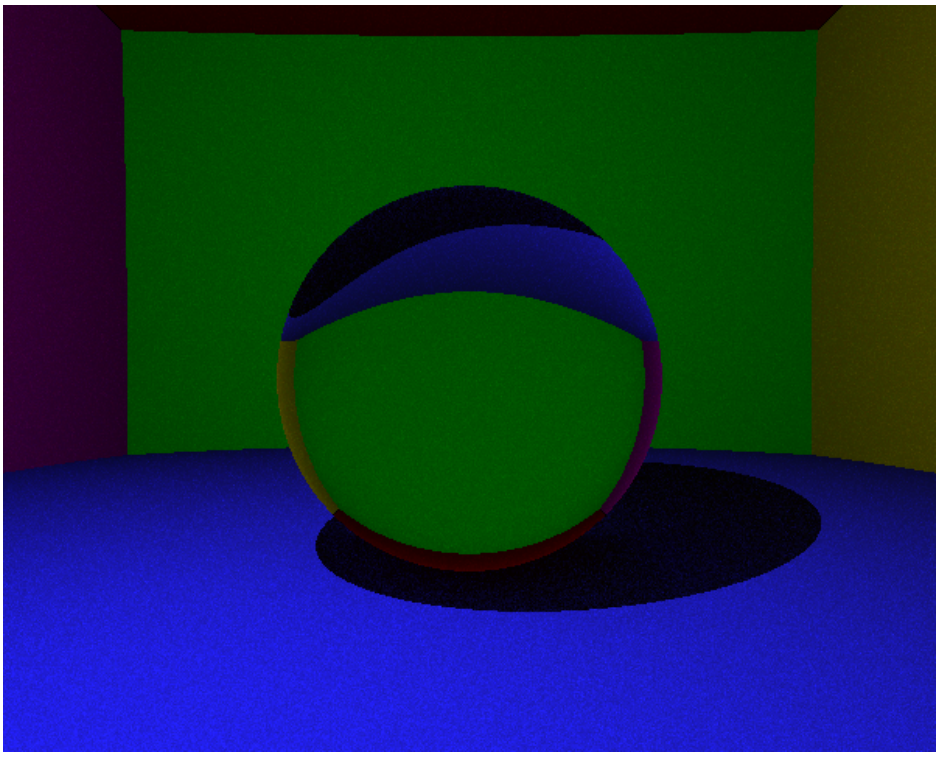


(a) Sphère spéculaire (21m17)

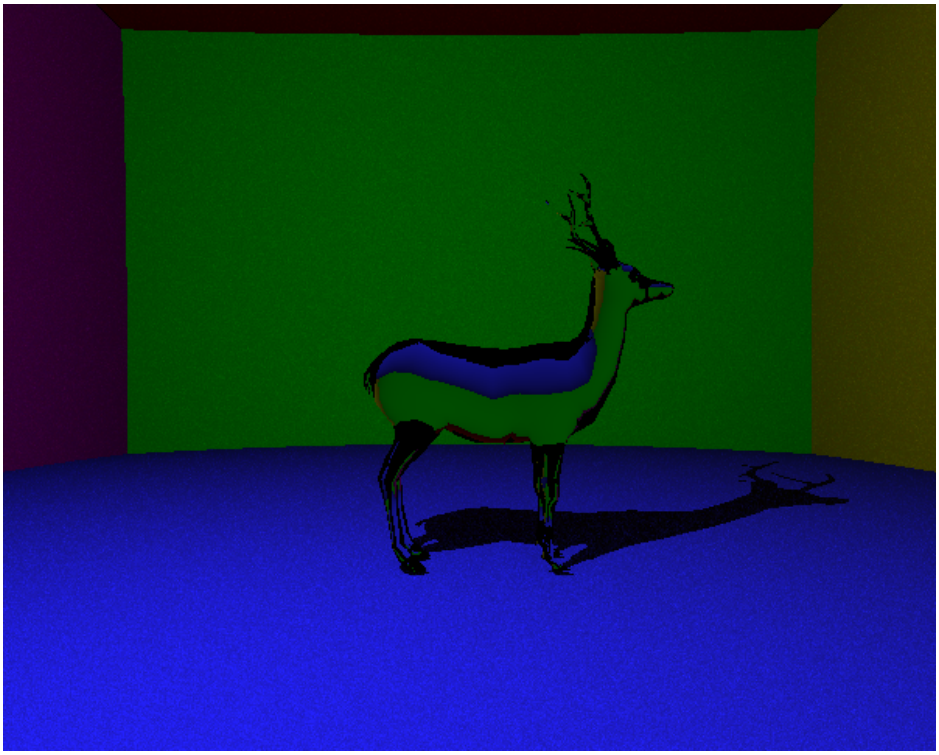


(b) Sphère diffuse (21m13)

FIGURE 3 – Rendus de Sphères 1000spp 5 rebonds



(a) Sphère transparente (3s9)



(b) Sphère diffuse (2m39)

FIGURE 4 – Rendus de matériaux transparents. Les caustiques ne sont pas gérées. Pour cela il faudrait un path tracer bi-directionnel

3 Ajouts depuis la présentation

3.1 Correction Gamma

Correction Gamma ajoutée lors de l'enregistrement des images. La couleur entre 0 et 1 est élevée à la puissance $\frac{1}{2.2}$. Cela rend plus visible l'éclairage indirect dans l'ombre de la sphère.

3.2 Anti-Aliazing

Implémentation de l'Anti-Aliazing en utilisant une répartition gaussienne de variance 0.5 des échantillons sur un pixel. Il pourrait être intéressant d'essayer de déformer la gaussienne pour la faire correspondre à la forme carrée du pixel. Un booléen permet de l'activer désactiver dans la classe Caméra.

3.3 Algorithme d'intersection de triangle

Implémentation d'un algorithme d'intersection de triangle à partir de la résolution de l'équation d'intersection plan/rayon et crédit à Jean-Claude Iehl pour l'implémentation de l'algorithme de Möller-Trumbore. L'implémentation naïve est environ 2.5 fois moins performante que Möller-Trumbore.

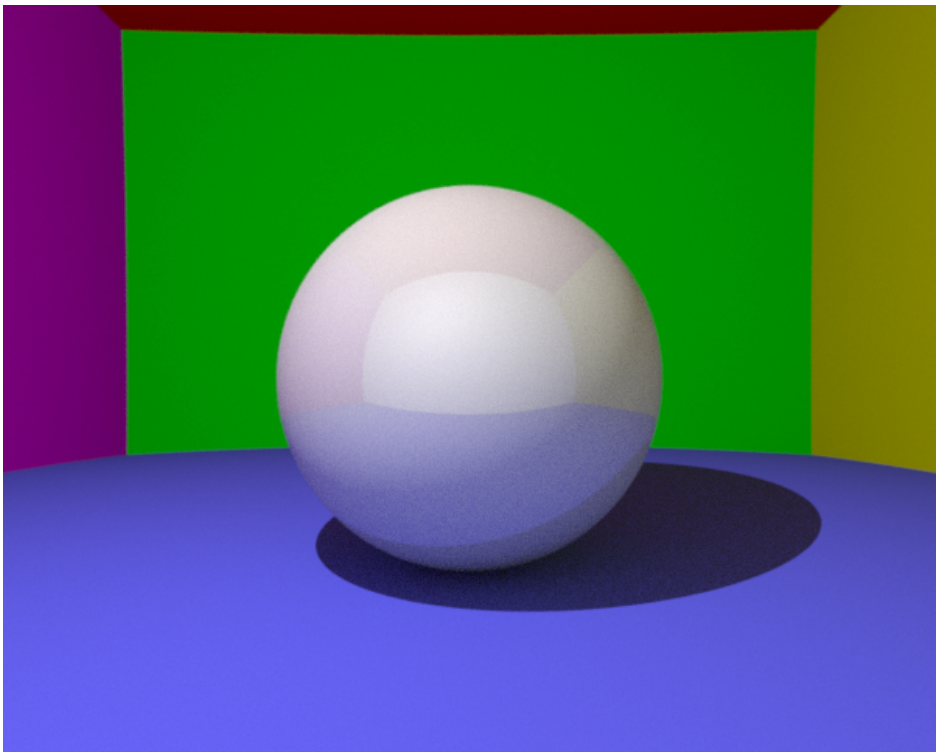


FIGURE 5 – Sphere spéculaire avec anti-aliazing 100 sample par pixel 2 rebonds