ÉCOLE NORMALE SUPÉRIEURE DE LYON

Université Claude Bernard Lyon 1

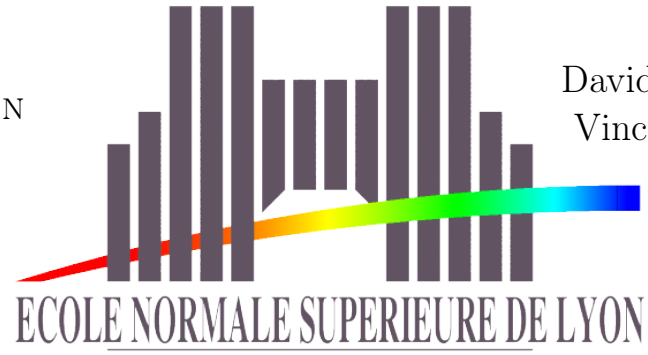
ENS de Lyon
Université Claude Bernard Lyon 1

Département d'Informatique
Cours CGDI
Projet

Moteur de Rendu avec du Raytracing

Élève :
Pacôme Luton

Enseignant
David Coeurjolly
Vincent Nivoliers



4 avril 2022

1 Introduction

Le but de ce projet est de découvrir le raytracing. Parmis les différentes manières de modéliser une scene 3D, j'ai choisi d'utiliser des fonctions implicites et donc du sphere-tracing.

La structure principale du code a été très fortement inspirer de Raytracing in One Weekend Shirley, 2020.

La manière de médéliser certains objets par des fonctions implicites a été prise sur ShaderToy.

2 Objects

2.1 Forme

Les formes des objets sont décrit par des fonctions implicites, prise sur ShaderToy. Comme par exemple cette étoile de la mort :

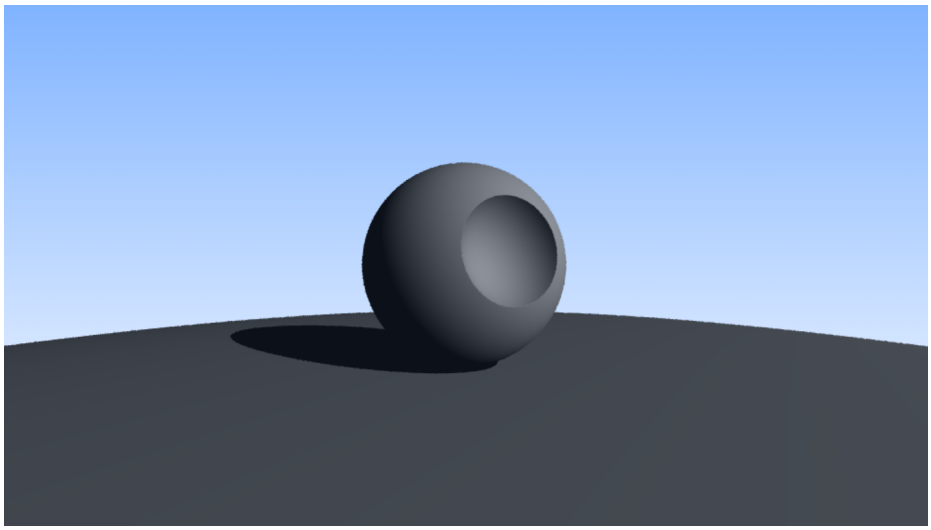


Figure 1 – Etoile de la mort

Pour rendre cette image j'ai utilisé une lumière à l'infini, ce qui simplifie les calculs.

2.2 Matériaux

J'ai implémenté 2 textures différentes, le miroir et le matériaux lambertien.

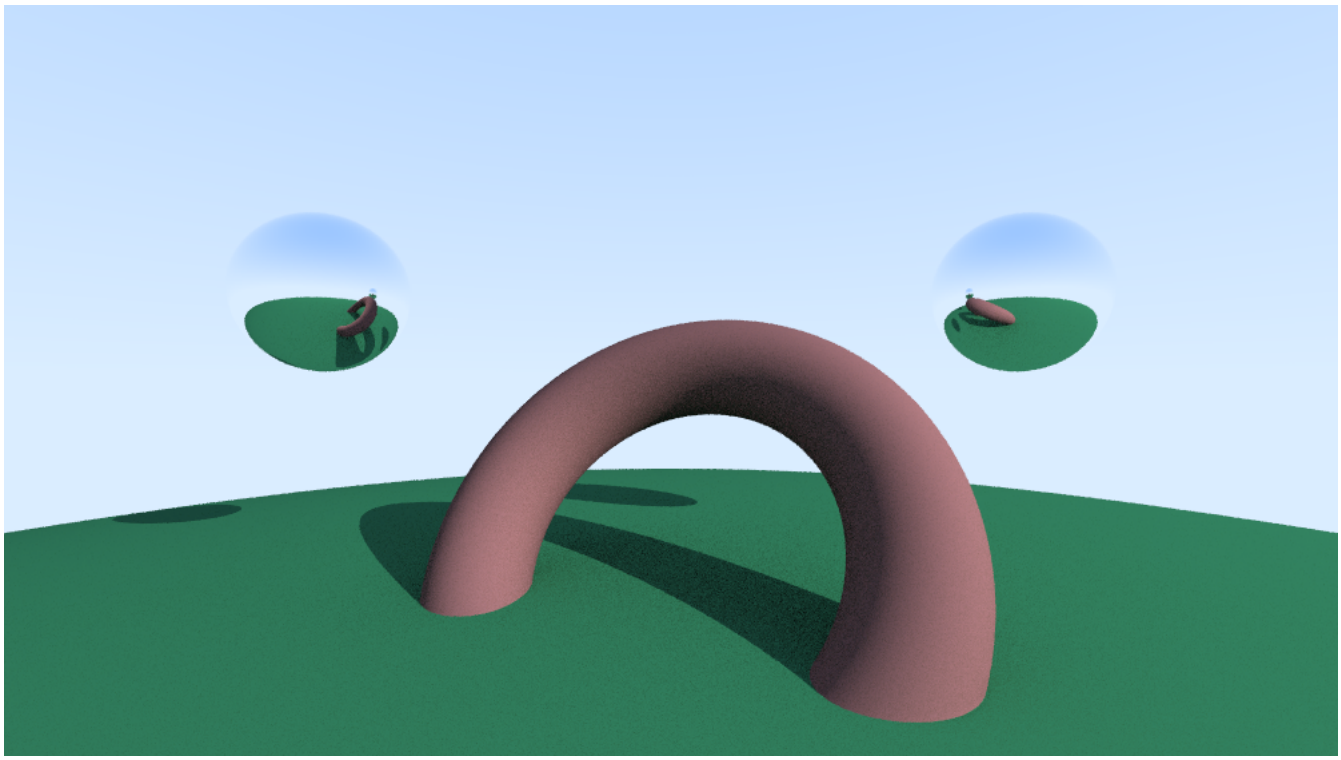


FIGURE 2 – Un donuts avec 2 miroir sphériques

3 Sampling

3.1 Uniform Sampling

J'ai commencé par faire un sampler uniform. Pour cela on utilise l'équation de la lumière :

$$I = \int_{\Omega} f(\omega_i) BRDF(\omega_o, \omega_i)(n \cdot \omega_i) d_{\omega_i}$$

ou :

- f : est la lumière qui vient de la direction ω_i.
- $BRDF$: est la fonction propre au matériaux qui décrit comment la lumière est absorbé et renvoyé. Ici, on a que des matériaux lambertien donc $BRDF(\omega_o, \omega_i) = \frac{C}{\pi}$, où C est la couleur de l'objet.

On calcul cette intégrale avec la méthode de Monte Carlo avec de l'importance sampling.

$$\hat{I} = \frac{1}{N} \sum_{j=1}^{N} f(\omega_i) BRDF(\omega_o, \omega_i)(n \cdot \omega_i) / pdf(\omega_i)$$

ou les ω_i sont décrit par par la fonction de probabilité pdf.

Dans notre cas, on tire aléatoirement que l'hémisphère en direction de la normal. Ainsi, $pdf(\omega_i) = \frac{1}{2\pi}$.

Ce qui donne un image avec un certain bruit :

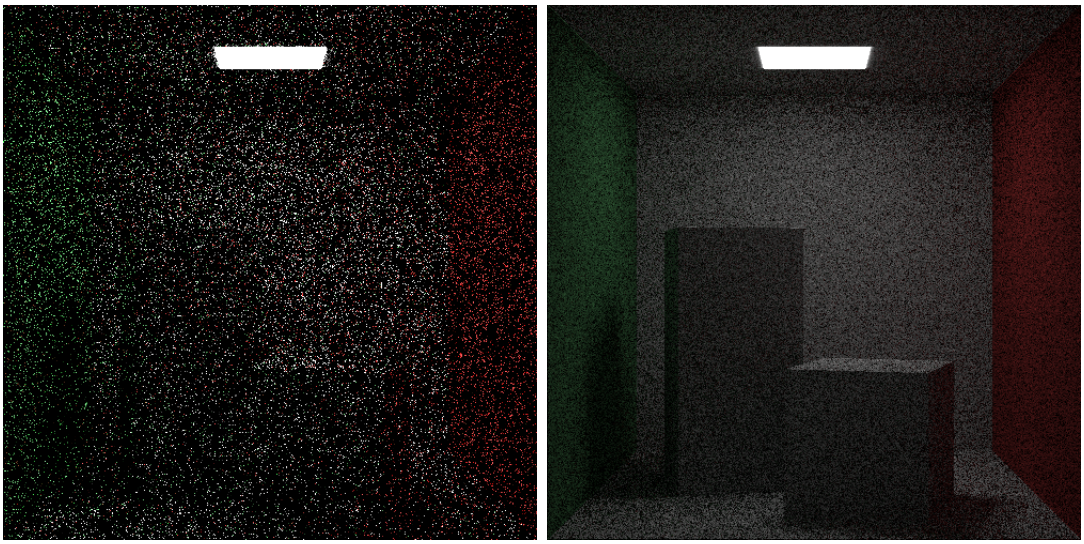


FIGURE 3 – 4 et 128 rayons par pixel

3.2 Direct Light Sampling

Puis j'ai fait un sampler qui se dirige vers la lumière direct. On a changé notre fonction de distribution de probabilité. Avec un peu de math on trouve :

$$pdf(\omega_i) = \frac{d^2}{cos(\alpha) \times area}$$

ou :

- d est la distance entre le point et la lumière.
- α est l'angle avec lequel on arrive sur le surface lumineuse.
- $area$ est l'aire de la surface lumineuse vers laquel on va.

Ce qui donne au final un image avec très peu de bruit, mais sans lumière à certains endroits :

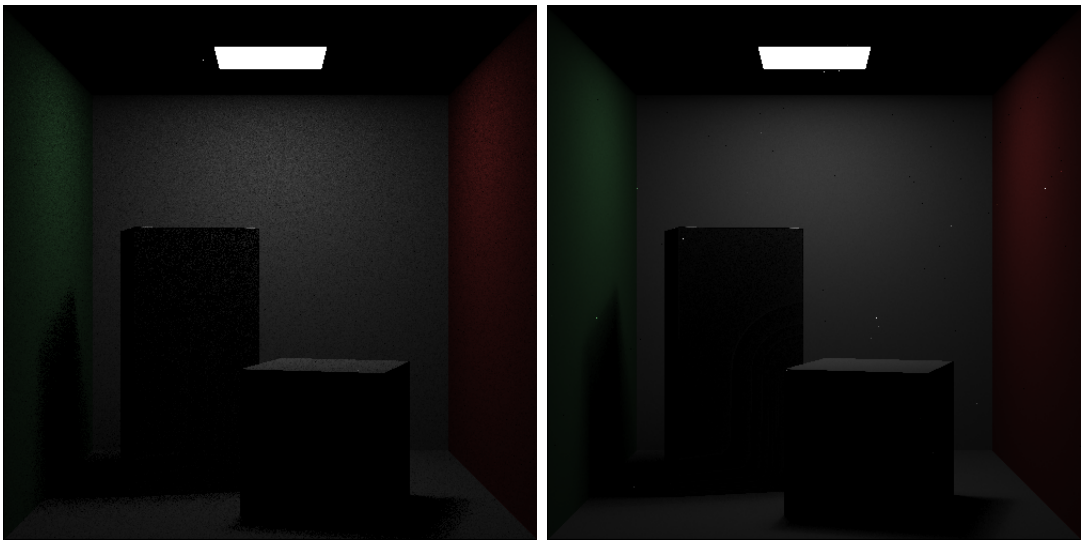


FIGURE 4 – 4 et 128 rayons par pixel

3.3 Multi Importance Sampling

Enfin, on peut facilement sommer des distributions de probabilité, Ainsi, j'ai implémenté un sampler qui fait un mixte des 2 méthodes précédentes. Ce qui permet de corrigé les défauts de chacune des 2 méthodes précédentes.

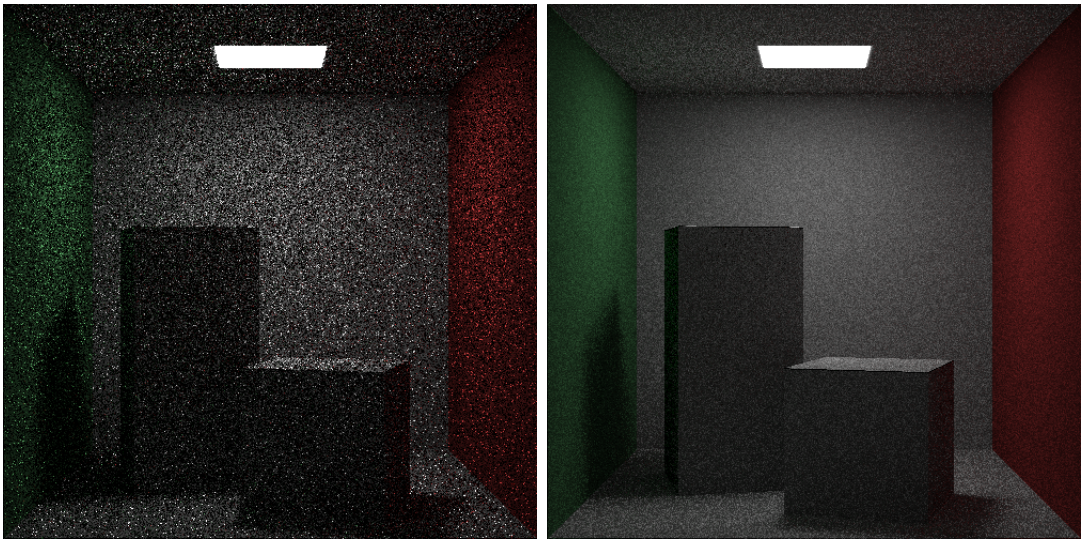


FIGURE 5 – 4 et 128 rayons par pixel

3.4 Rapide comparaison

On voit que la variance est plus faible dans le sampler du milieu :

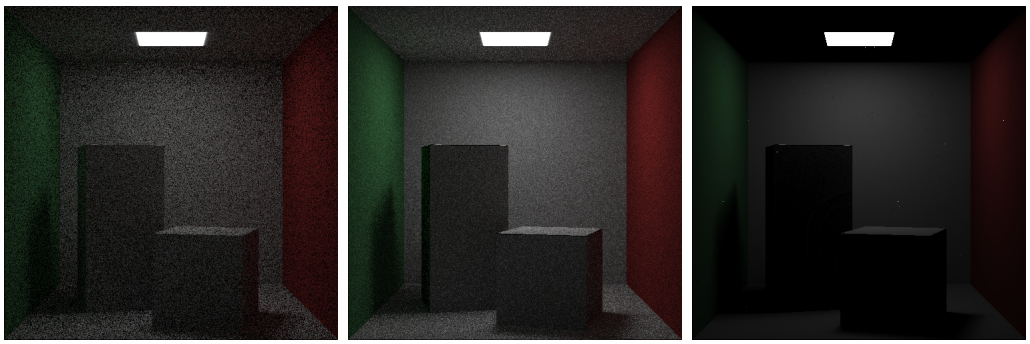


FIGURE 6 – uniform - mixte - light

4 Conclusion

Je me suis bien amusé, et je vais continué ce projet. La prochain étape et d'implémenter d'autres sampler. Comme par exemple ceux décrits ici : *Uni(corn|form) tool kit* s. d.

Bibliography

SHIRLEY, Peter (2020). *Ray Tracing in One Weekend*. https://raytracing.github.io/books/RayTracing
URL : https://raytracing.github.io/books/RayTracingInOneWeekend.html.

Uni(corn|form) tool kit (s. d.). urlhttps ://utk-team.github.io/utk/.