Synthèse d'image 3D

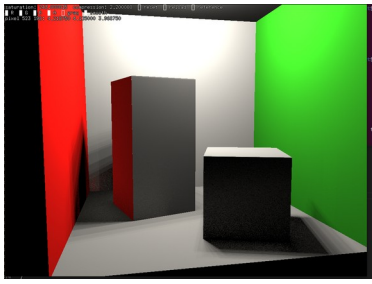
Rapport TP2

Barros Mikel-ange – Scouflaire Thomas

Partie 1 : éclairage direct

L'éclairage direct n'a pas posé de problèmes , il nous a suffi de recopier la formule donnée en cours pour obtenir des résultats satisfaisants (voir la section exemple). L'implémentation n'a pas été longue et nous avons pu nous consacrer à la partie 7 lumière indirecte.

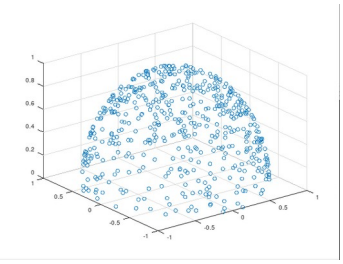
Afin d'avoir de meilleurs résultats, nous avons tonnemappé chaque pixel de l'image avec la formule p(i)=p(i)^(1/2,2)



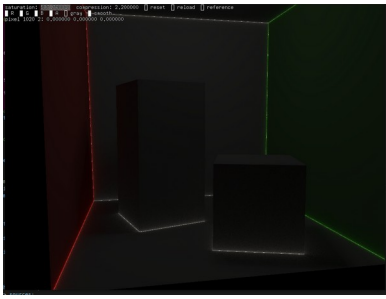
Partie 2 : éclairage indirect

L'éclairage indirect quant à lui nous a posé plus de problèmes, nous avons dû refaire 3 fois notre fonction d'éclairement indirect.
Une première fois car nous n'utilisions pas les formules de pdf données par vous ce qui nous donnait une répartition aberrante des points sur notre hémisphère.

Suite à cela nous avons implémenté l'équation 34 extraite de [GIC03] et nous avons vérifié qu'elle était correcte en utilisant octave (cela était avant que vous nous parliez de l'application direction de gkit).



Cependant, malgré cette version plus cohérente de l'hémisphère, nous avions encore des problèmes qui après beaucoup de recherches se sont avéré venir de notre fonction de couleur directe, en effet, si on fait partir un rayon proche d'un angle de la scène, il va rencontrer son voisin à une distance L très faible et la partie (cos theta * cos theta s)/L devient alors très grande ce qui donne cet effet d'accumulation sur les bords.



Afin de corriger ce problème nous avons redéfinis une fonction de couleur direct qui elle ne prend pas en compte les paramêtres cos theta, cos theta s et L. Les résultats de cette version étant satisfaisants, nous nous sommes arrêtés là. (voir la partie exemple)

Malheureusement l'ensemble du tp tourne sans bvh car nous n'avons pas eu le temps d'en implémenter un, cela pose des problèmes de temps de calcul comme vous pourrez le voir plus loin.

Partie 3 : exemple

Dans les exemples le nombre de rayons désigne le nombre de rayons par source, donc si il y a deux sources ce nombre doit être multiplié par 2.

La cornell box : 2 sources de lumières, 492 positions

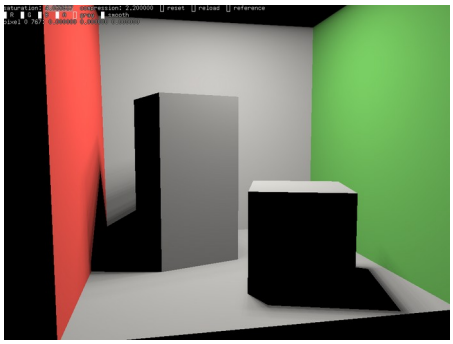


Figure 1: 32 rayons, 0 rebonds, 26 secondes d'execution

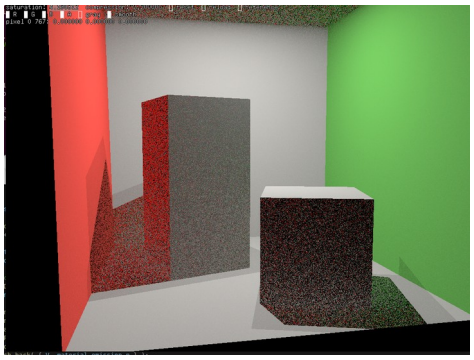


Figure 2: 1 rayon, 1 rebonds, 5 secondes d'execution

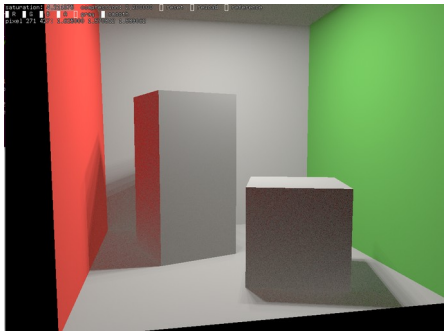


Figure 3: 8 rayons, 16 rebonds, 37 secondes d'execution

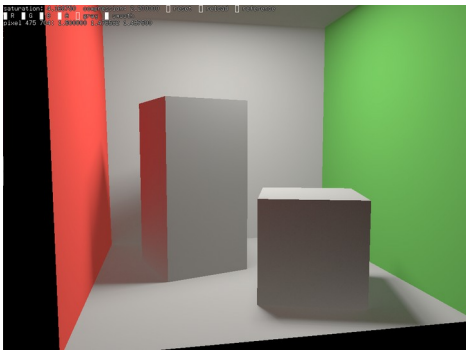


Figure 4: 256 rayons, 512 rebonds, 40 minutes d'execution

La cornell box modifiée : 2 sources, 96 positions

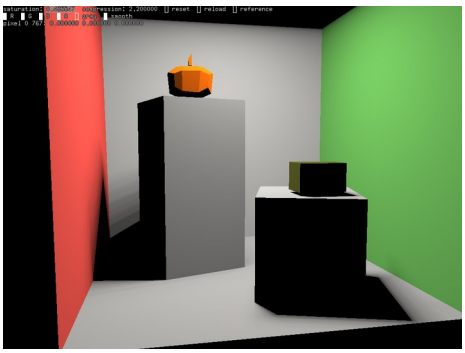


Figure 1: 32 rayons, 0 rebonds, 92 secondes d'execution

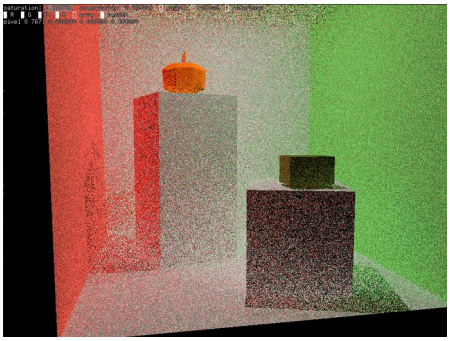


Figure 2: 1 rayon, 1 rebonds ,9.5 secondes d'execution

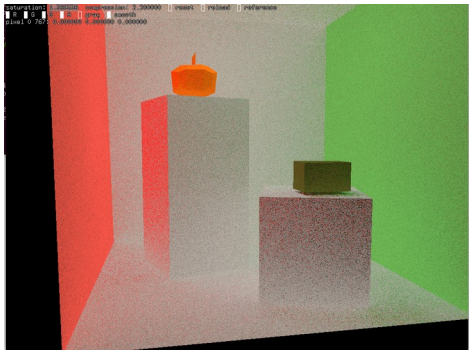


Figure 3: 8 rayons, 16 rebonds, 105 secondes d'execution

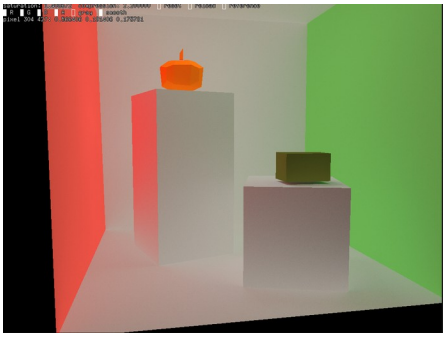


Figure 4: 256 rayons, 512 rebonds, 1h10 d'execution

Sapin de noël : 228 sources, 3000 positions (simplification de la scène de base trop lourde (5199 sources, 22000 positions)

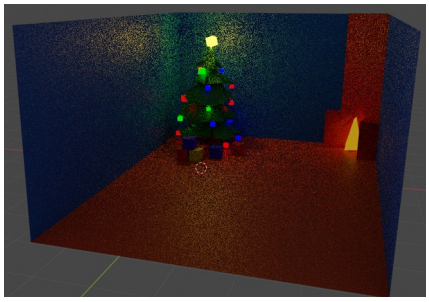


Figure 1: rendu blender, path tracing

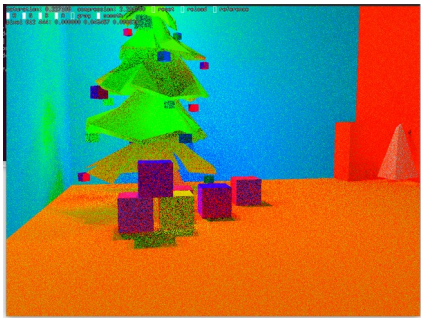


Figure 2: 1 rayons, 1 rebond 20 minutes d'execution

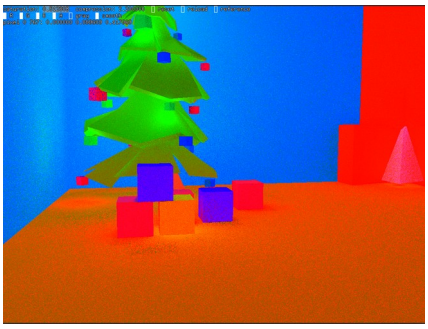


Figure 3: 8 rayons, 16 rebonds, 3h d'execution

L'absence de bvh se fait vraiment sentir, le fait de devoir parcourir l'ensemble des triangles pour chaque source et chaque rebond alourdi énormément la scène et sur une scène avec beaucoup de sources et de triangles il devient très difficile d'obtenir un résultat rapide avec beaucoup de rayons.

Référence :

[GIC03] GI Compendium
[JCI20] Cours de M2 sur le ray tracing

Lien vers le code : https://forge.univ-lyon1.fr/p1508726/rendu-si3d