Travaux Dirigés de Simulation Radiométrique Avancée

Romain Luu

22 février 2018

Nous codons ici un simulateur de lancers de rayons. Dans ce document, nous allons décrire les différentes fonctionnalités codées avec une illustration des résultats. Celui-ci fonctionne en simulant une scène avec les objets, qu'ils soient des volumes géométriques simples ou complexes définis par un maillage - celui-ci est alors rastérisé en triangles - avec une caméra et une méthode d'intégration. Il simule ensuite le lancer de rayons au-travers des pixels de la caméra et échantillonne de manière stochastique la valeur de couleur que doit prendre chaque pixel.

1 Lancers de rayon - fonctionnalités de la caméra

La caméra est modélisée par un point représentant un sténopé avec une matrice ($512 * 512 pixels^2$ par défaut) On tire alors un rayon dans chaque pixel, avec une direction aléatoire dans la limite spatiale du pixel afin de supprimer les phénomènes de Moiré artificiels induits par une régularité d'échantillonnage trop importante.
Une méthode d'antialiasing est aussi implémentée, afin de pouvoir tirer plusieurs rayons par pixel dans des directions aléatoires, afin de pouvoir réduire le bruit d'échantillonnage induit par l'utilisation de lancers de rayons, particulièrement visible sur des arêtes franches, ou des fréquences spatiales élevées.

2 Matériaux

Notre logiciel permet de simuler plusieurs types de BRDF, selon un modèle simple de diffusion ou bien selon des modèles plus complexes comme le modèle empirique de Phong qui tient compte d'une forme de lobe bien particulière pour la réflexion spéculaire ou bien celui de Ward, qui permet une paramétrisation des coefficients α_x et α_y pour contrôler une anisotropie.

FIGURE 1 – Scène (scenePerso.scn) avec trois teletubbies, avec pour BRDF (de gauche à droite) : Phong, Diffus, Ward. A gauche, la configuration de la scène avant rendu. A droite, le rendu de la scène.

Nous pouvons avec la BRDF de Phong changer l'exposant (= le rayon du lobe principal) ce qui, d'aspect, change le caractère diffus/spéculaire de la réflexion (= mat/brillant)

FIGURE 2 – Trois teletubbies avec une BRDF de Phong avec pour exposant (de gauche à droite) 1,10,150

Nous pouvons avec la BRDF de Ward changer la forme des lobes en fonction des deux directions (∥et ⊥ au R_O) ce qui, d'aspect, rend une caractère anisotrope de la réflexion.

FIGURE 3 – Trois teletubbies avec une BRDF de Ward avec pour α_x et α_y : [0.2,0.1], [0.1,0.1], [0.1,0.5]

FIGURE 4 – Les pdf des BRDF de Diffus (au milieu) et Ward(à droite) (je n'ai pas codé Phong) : on peut imaginer la forme des lobes

3 Intégrateurs

Les intégrateurs permettent de calculer de manière récursive les contributions environnantes qui permettent de colorer chaque pixel de la caméra. Il tiennent compte des trajectoires inter-objets qui participent à la diffusion, des contributions spéculaires, des contributions directes des lumières, des réflexions et de l'environnement extérieur, simulé par un papier peint. Nous avons codé successivement plusieurs intégrateurs, afin de pouvoir :

1. calculer pour chaque intersection de rayon la contribution de plusieurs rayons incidents (intégration stochastique) typiquement 100
2. faire de l'antialiasing
3. tenir compte des sources de lumière étendues ainsi que de leurs textures
4. faire du Pathtracing : lancer plusieurs rayons de même direction initiale mais qui ne se divisent pas à chaque intersection, ce qui recrée un seul chemin
5. roulette russe : au lieu de définir un nombre arbitraire petit de récursions, on introduit une probabilité d'absorption par le matériau, ce qui permet de limiter sans biais la convergence.

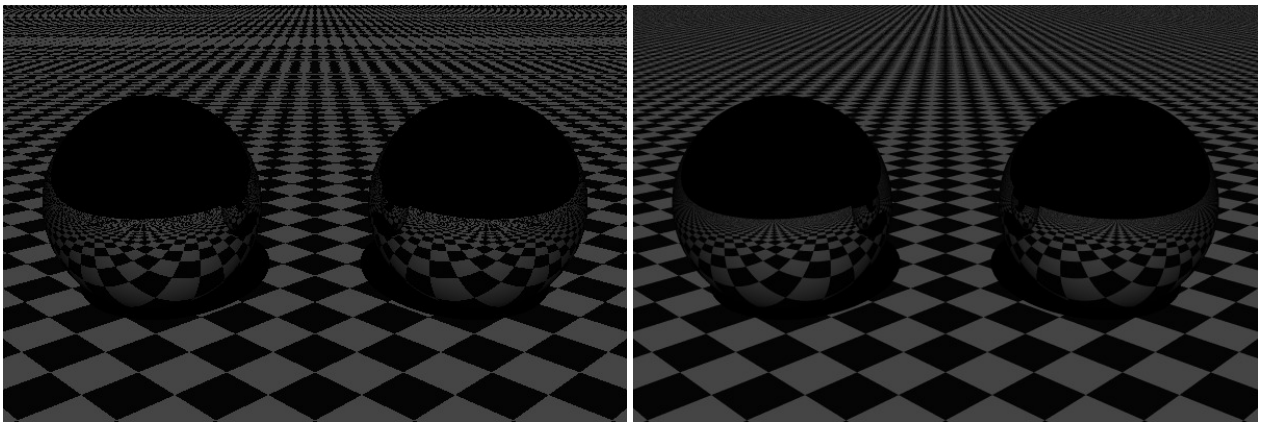

FIGURE 5 – Anti-aliasing : 1 sample/pixel(12s de rendering) vs 16 samples/pixel(192s de rendering)

Prise en compte de sources étendues et texturées : lors du calcul de l'éclairage direct, on tire un rayon aléatoire dans la direction de la source étendue et on reprend la couleur de la source au point d'intersection avec le rayon.

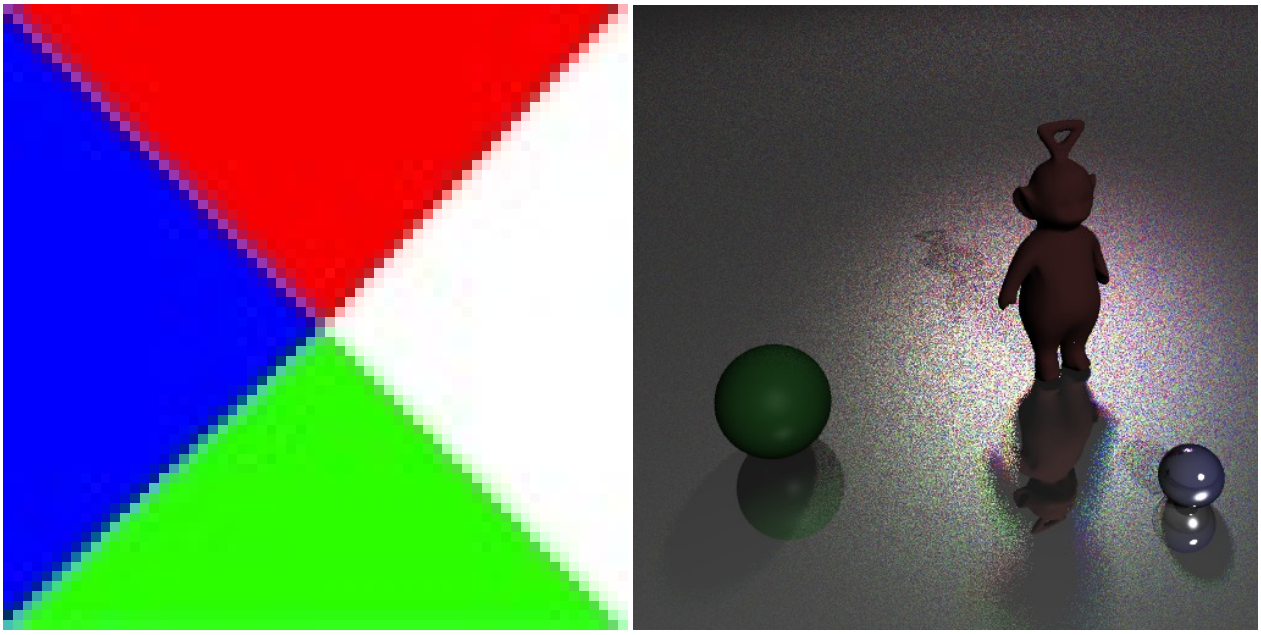

FIGURE 6 – On peut simuler des sources de lumière directionnelles mais aussi des sources surfaciques texturées : à gauche la texture de la source carrée texturée utilisée dans le rendu des scènes, et à droite les ombres portées des différentes couleurs de la source (Rq : je n'arrive pas à flip l'image de la source dans le bon sens)

4 Résultats

En chargeant la scène *box2.scn*, on peut observer pas mal des capacités du programme :

- intégrateur path_mats qui fait du ray-tracing à deux rebonds
- anti-aliasing de 20 rayons par pixel
- une sphère de BRDF de type Ward ($\alpha_x = 0.1$ et $\alpha_y = 0.9$)
- un sol de BRDF de type Ward ($\alpha_x = 0.01$ et $\alpha_y = 0.09$)
- des murs diffus rouge, gris et bleu
- une source de lumière étendue texturée

On constate que le mur blanc du fond possède du bruit très coloré, lorsque le bruit sur les autres murs est de la même couleur que le mur.

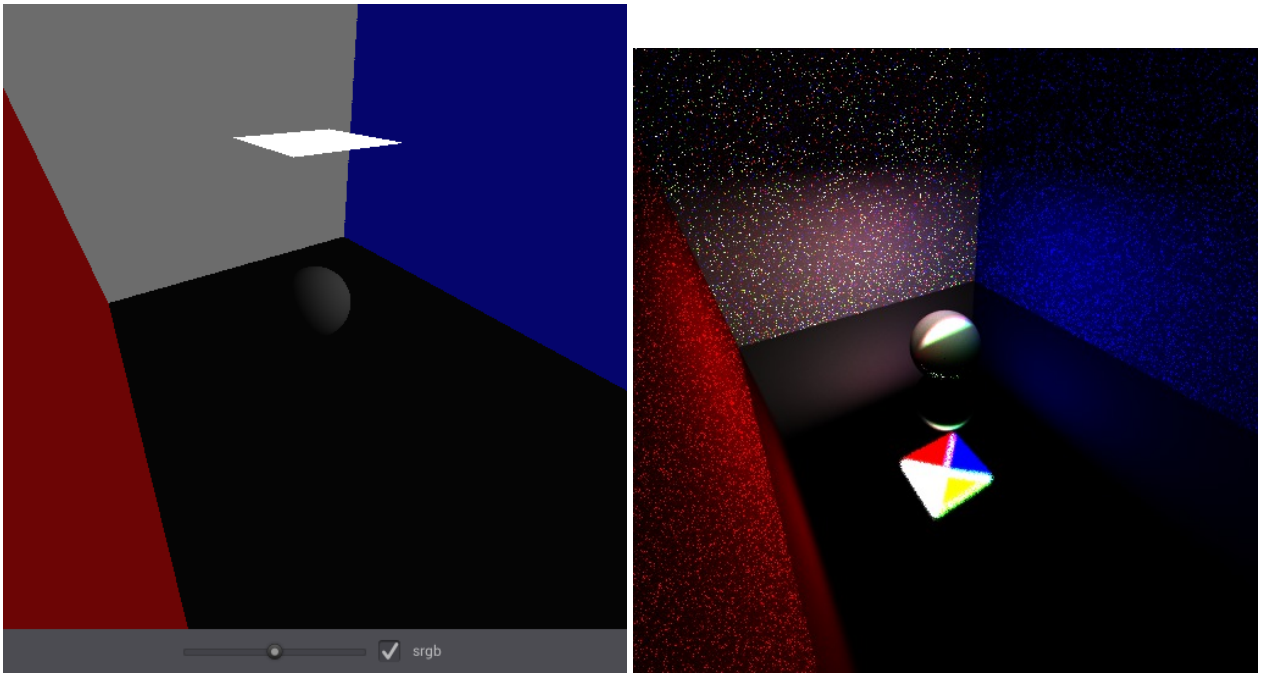

FIGURE 7 – La scène Box2, et son rendu

5 Limites de mon implémentation

1. En générant box2, j'ai encore parfois une erreur de l'intégrateur, calculant une valeur d'irradiance absurde. Ces erreurs sont marginales et lissées par l'antialiasing et les 100 lancers par rayon de pixel. Sûrement un effet de bord.
2. Il existe encore des biais dans mon code :
   — max_recursion toujours pris en compte (bien que l'on puisse le fixer à une grande valeur) car je n'ai pas la certitude de la convergence si jamais je l'enlève
   — la limite itérative de 100 dans Ward : :Sample_IS.
3. J'ai l'impression que mon programme contient encore une autre erreur et "fuit" : mon programme devient de plus en plus lent au fur et à mesure de la génération de rendu. Cela est encore acceptable pour générer box2 avec un nombre max de récursions mais en implémentant la roulette russe, mon programme ralentit trop, avec en plus l'apparition de carrés non rendus, qui sont toujours situés en limite de sol et de mur d'ailleurs...

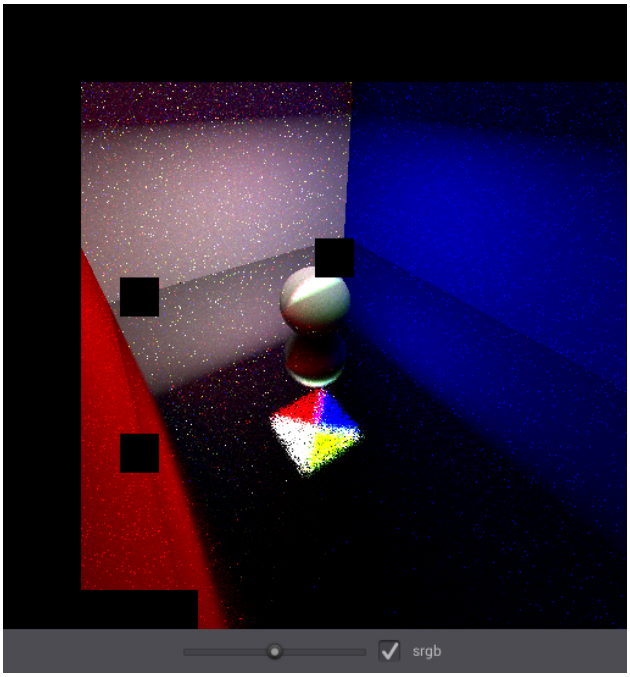

FIGURE 8 – quand max_recursion est réglé à 3, et l'antialiasing à 1 (au lieu de 20 au départ)

EDIT : j'ai réussi à m'en défaire en ajoutant une condition limite d'itérations (100) à Ward : :Sample_IS. Il semblerait qu'il y ait des pixels au bord du sol qui génèrent une très grande proportion de rayons dans le mauvais sens...

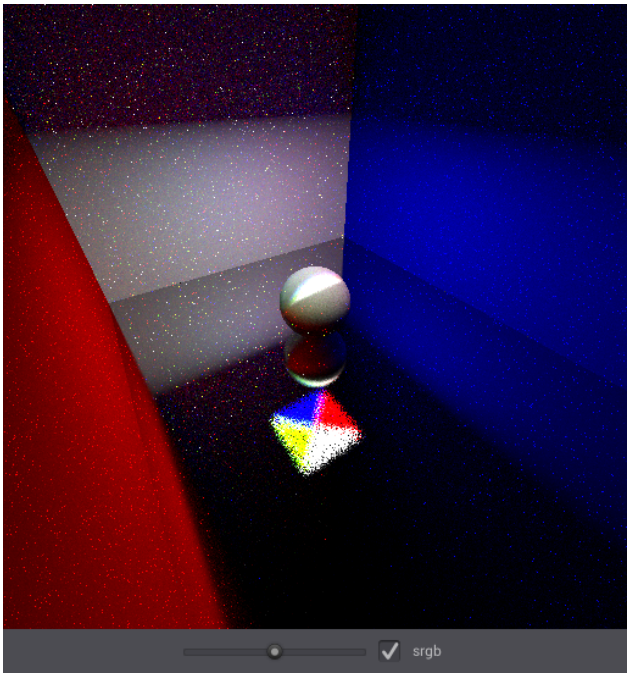

FIGURE 9 – quand max_recursion est réglé à 10, et l'antialiasing à 2 (au lieu de 20 au départ)