Université Paris-Saclay

État de l'art du projet PPN

Filière : M1 Calcul Haute Performance et Simulation

Années : 2025-2026

Sujet : Path tracing : méthodes de Monte Carlo pour la synthèse d'images

---

Rédigé par :
Nolwen DOLLÉANS
Prince Marcel MASSAMBA
Théo BAUER

Encadré par :
Mr. Aurélien DELVAL

Table des matières

Introduction **2**

1 Qu'est-ce que le Path Tracing ? **2**
1.1 Le Ray Casting 2
1.2 Implémentation du principe de rendu 3
1.3 Implémentation de la méthode Monte-Carlo 3

2 Comparaison avec des technique existantes **3**

3 Limites du Path Tracing **4**

Introduction

La modélisation en 3 dimensions est devenue incontournable depuis des décennies. Elle est en effet présente dans différents domaines et a des applications variées comme l'animation, les jeux vidéos ou encore la simulation réaliste. Cette manière de représenter notre monde dans un cadre réaliste provient notamment de la simulation de trajectoires des photons.

Pour ce projet, nous allons nous concentrer sur la technique du *Path Tracing*, combinant la technique du *Ray Tracing*, ainsi qu'une approche aléatoire pour résoudre des problèmes déterministes par la méthode de *Monte-Carlo*.

1 Qu'est-ce que le Path Tracing ?

Le Path Tracing est une technique de rendu 3D se basant sur la méthode de Monte-Carlo pour approximer l'équation de transport de la lumière. Le principe est d'échantilloner aléatoirement un grand nombre de chemins lumineux entre la caméra et les différentes sources de lumière présentes dans la scène.

1.1 Le Ray Casting

Le Path Tracing se base sur le **Ray Casting**. Le ray casting est une technique de rendu visuel qui a pour principe d'afficher des images avec une illusion de profondeur / 3D. Cette technique a été popularisée par le studio ID Software, qui l'a entre autre utilisé pour Wolfenstein 3D, DOOM et Quake. Elle fut souvent utilisée pour du temps réel, car elle était très peu coûteuse en termes de calcul pour les machines de l'époque. L'environnement est représenté de la manière suivante : une caméra, qui sert de point de vue à la manière d'un œil et une grille représentant notre environnement. À partir de la caméra, plusieurs rayons sont projetés aléatoirement. À chaque contact entre un rayon et une surface, nous effectuons une interpolation entre la position de la caméra et la position du pixel touché par l'un de nos rayons. Dès qu'un rayon atteint une surface, on calcule la distance du rayon et de la couleur du pixel atteint en fonction des propriétés de celle-ci. Plus la distance du rayon est grande, plus la surface est éloignée. Pour créer un effet de profondeur, nous calculons l'angle entre le rayon projeté et une surface. Plus l'angle est grand, plus l'effet de profondeur sera grand.

Cette technique apporte quelques inconvénients, d'où le fait qu'elle soit utilisée en complément d'autres techniques de rendu. Elle ne permet pas d'effectuer du rendu sur des objets 3D complexes. Pour du temps réel, nous sommes limité sur une rotation par rapport à l'axe de Y. Nous ne pouvons pas faire du rendu volumétrique, i.e rendu d'un effet de flamme ou liquide.

1.2 Implémentation du principe de rendu

Pour la rendre plus réaliste, le path tracing utilise l'équation de rendu (théorisé par James Kajiya en 1986) suivant :

$$L_r(\vec{x}, \omega_r, \lambda, t) = L_e(\vec{x}, \omega_r, \lambda, t) + \int_\Omega f_r(\vec{x}, \omega_i, \omega_r, \lambda, t) \cdot L_i(\vec{x}, \omega_r, \lambda, t) \cdot \cos(\theta_i)\, d\omega_i$$

avec :

- L_o la radiance sortante, L_e la radiance émise par la surface, f_r fonction de la réflectance bidirectionnelle (BRDF) et L_i est la radiance incidente.
- $\vec{x}$ le point d'impact du photon, avec Ω l'hémisphère centré autour de $\vec{x}$ contenant toutes les valeurs possible de ω_i.
- λ la longueur d'onde et t le temps.

Cette équation nous permet d'ajouter le principe de rebond des photons sur une surface, rendant cette approche beaucoup plus réaliste qu'avec simplement le ray casting qui ne prend en compte aucun rebond. Cependant, aucune solution générale n'est possible. C'est pourquoi nous allons utiliser la méthode Monte-Carlo pour obtenir une solution numérique qui converge vers la solution analytique pour un grand nombre de rayons.

1.3 Implémentation de la méthode Monte-Carlo

L'intégrale $\int_\Omega f_r(\vec{x}, \omega_i, \omega_r, \lambda, t) \cdot L_i(\vec{x}, \omega_r, \lambda, t) \cdot \cos(\theta_i)\, d\omega_i$ n'étant pas resolvable anlytiquement, nous allons l'approximer avec la formule suivante :

$$\frac{1}{N} \cdot \sum_{i=1}^{N} \frac{f_r(\vec{x}, \omega_i, \omega_r, \lambda, t) \cdot L_i(\vec{x}, \omega_r, \lambda, t) \cdot \cos(\theta_i)}{p(\omega_i)}$$

avec :

- $p(\omega_i)$ la probabilité que le rayon incident vienne d'une direction ω_i.

Nous pouvons donc approximer l'équation (1) par :

$$L_r(\vec{x}, \omega_r, \lambda, t) \approx L_e(\vec{x}, \omega_r, \lambda, t) + \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{f_r(\vec{x}, \omega_i, \omega_r, \lambda, t) \cdot L_i(\vec{x}, \omega_r, \lambda, t) \cdot \cos(\theta_i)}{p(\omega_i)}$$

2 Comparaison avec des technique existantes

- **Rasterization** : technique de rendu 3D utilisée pour convertir des objets 3D en une image 2D affichable à l'écran. Elle fonctionne triangle par triangle plutôt que pixel par pixel. Très rapide, optimisée pour des scènes avec des millions de triangles et avec des exigences en temps-réel. Mais les ombres/éclairages indirects, réflexions complexes et phénomènes optiques réalistes restent approximatifs.

- **Ray tracing** : lance des rayons partants de la caméra pour simuler les ombres, réflexions et occultations ne traitant alors pas toute la scène, mais uniquement les trajectoires des photons qui seront interceptés. Cela donne un rendu plus fidèle que le rasterizaton mais souvent limité à quelques rebonds ou approximations car plus coûteuse en temps et en énergie.

De nos jours, la technique de rendu la plus populaire est la rasterization. Bien que plus coûteuse en calcul, la rasterization demeure dominante dans les moteurs temps réel.

3 Limites du Path Tracing

Le Path-Tracing étant un modèle, il nous faut de poser des fonctionnements qui ne font pas diverger la simulation de la réalité. Dans le Path-Tracing, les chemins lumineux parcourent un ligne droite entre chacuns des rebonds, sans que d'autres éléments puissent les perturber ; en particulier d'autres chemins lumineux ou un effet de brume ou fumée. De plus, les rayonnements ne font pas appel aux proriétées des ondes éléctromagnétiques qui sont connus pour les photons. D'autres effets spécifiques ne sont pas simulables, telles que la fluorescence ou la polarisation. Cependant ces limitations ne représentent pas une vraie limite du modèle, leur impact est mineur sur le rendu.
Pour les limites liées à l'utilisation du Path-Tracing, il y a premièrement le problème de son implémentation pour un rendu en temps réel, le matériel nécessaire pour son implémentation est coûteux et donc le rendu devient dépendant de la capacité à traiter les images.
Deuxièmement, une implémentation avec une grande quantité d'objets dans la scène, de rayons lumineux et de rebonds considérés demande également de fortes ressources matérielles. Le Path-Tracing comprend de plus un test pour d'éventuelles intersections de chemins lumineux, ce qui demande des performances, particulièrement pour des modèles 3D complexes.

Conclusion

Au vus des ressources nécessaires pour une implémentation du path tracing, on a opté pour du pré-calculer. C'est-à-dire, générer une image en petite définition. Pour le format de l'image, nous avons opté pour du PPM, car simple à générer. Une image PPM est généralement représentée de la manière suivante :

P3 // Nombre magique pour des couleurs en format RGB

height // hauteur

width // largeur

255 // La valeur max d'une composante RGB, généralement 255

255 255 255 // 1 pixel
. . . .
255 255 255 // height x width ième pixel

L'image finale sera générée en fonction d'un buffer d'entier 24-bit qui aura une taille height x width, où chaque pixel correspondra à un entier 8-bit.
La manière dont nous implémentons le buffer peut avoir un impact sur la localité des données.
Nous avons implémenté 3 façons de ranger notre buffer :

- Un buffer d'entier non signé de 32-bit (le huitième octet pas nécessaire, car nous utilisons un format RGB + il n'existe pas de type primitive codé sur 3 octets, car la taille d'un type primitive sont tous des puissances de 2).
- Un triple buffer (Struct of Array).
- Un tableau d'éléments de taille 24-bit (Array of struct).

Cela nous permettra par la suite d'effectuer des mesures et de comparer la rapidité des implémentations.

Ainsi, le Path Tracing constitue une méthode réaliste mais coûteuse, dont notre implémentation vise à explorer les compromis entre qualité et performance.