Pathtracing & Filtre SVGF

Rendu en temps réel avec débruitage spatio-temporel

Samuel Fournier (20218212)

Département d'informatique et de recherche opérationnelle
Université de Montréal

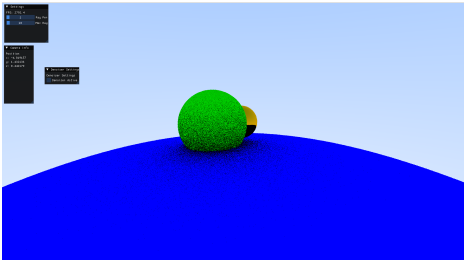
IFT 6150 – Hiver 2025

Plan de la présentation

Introduction & Objectifs

Le Problème :

- Le Pathtracing produit des images réalistes mais est extrêmement coûteux.
- Temps réel (30-60 FPS) $\rightarrow$ peu de rayons par pixel (SPP).
- Peu de SPP $\rightarrow$ Image très bruitée.



Exemple : 1 Rayon par Pixel (Bruité)

La Solution :

- **SVGF** (Spatiotemporal Variance-Guided Filtering).
- Algorithme de NVIDIA (Schied et al.).
- Combine accumulation temporelle et filtrage spatial intelligent.

Fonctionnement du Pathtracing

Pipeline simplifié :

1. **Ray Gen :** Création des rayons depuis la caméra.
2. **Intersection :** Test rayon vs scène (Optimisation AABB).
3. **Shading :** Éclairage indirect, matériaux, émission.
4. **Rebond :** Génération d'un nouveau rayon (Diffuse/Spéculaire).

Défi de performance

Complexité : $\mathcal{O}(P \cdot R \cdot B \cdot I)$.
Pour du temps réel, nous sommes limités à très peu de rayons par pixels.

Le concept de Reprojection

L'idée : Réutiliser l'information des images précédentes pour accumuler de la lumière.

Le G-Buffer (Geometry Buffer) : Nous stockons plus que la couleur :

- Profondeur (Depth)
- Normales
- Mesh ID
- **Motion Vectors** (Vecteurs de mouvement)

Motion Vectors : Calculer où se trouvait le pixel actuel dans l'image précédente ($UV_{prev} = UV_{curr} - Motion$).

Validité de l'Historique

On ne peut pas aveuglément réutiliser l'historique (objets qui bougent, ombres, désocclusion).
Trois tests de rejet :

1. **Profondeur :** Si l'écart est trop grand.
2. **Normale :** Si l'angle a trop changé.
3. **Mesh ID :** Si l'objet n'est plus le même.

```
// Exemple de logique de rejet
if(abs(currentDepth - historyDepth) > DEPTH_THRESHOLD) isValid = false;
if(dot(currentNormal, historyNormal) < 0.9) isValid = false;
if(currentMeshID != historyMeshID) isValid = false;

```

Résultat : Accumulation via moyenne mobile exponentielle (EMA) si valide.

Résultats de l'accumulation temporelle

$$oldAccumulationCount = historyColor.a$$

$$clampedOldCount = min(1024, oldAccumulationCount)$$

$$alpha = \frac{1}{clampedOldCount}$$

$$finalColor = mix(historyColor.rgb, noisyColor.rgb, alpha)$$

Plus la variable d'accumulation est grande, plus on favorise l'historique débruitée. On restreint le terme d'accumulation pour que l'accumulation ne favorise pas excessivement l'hitorique et ignore l'image qui vient d'être générée.

Filtre Spatial À-trous

Même avec l'accumulation temporelle, il reste du bruit (surtout zones en mouvement).
La méthode :

- Filtre "Wavelet" À-trous (avec des trous).
- 5 itérations.
- À chaque itération, on double l'espacement (2^k) entre les échantillons.
- Permet de couvrir un voisinage de 33×33 avec un noyau 5×5.

Variance-Guided : Le flou ne doit pas détruire les bords (edges). On utilise des poids (w).

Pondération du Filtre (Edge-Stopping)

Pour chaque pixel voisin, on calcule un poids $w = w_n \cdot w_z \cdot w_l$.

1. Normales (w_n)

Préserve les arêtes géométriques.

$$w_n = \max(0, n_c \cdot n_s)^{\phi_n}$$

2. Profondeur (w_z)

Évite de mélanger avant-plan et arrière-plan.

$$w_z = \exp\{-\frac{|d_c - d_s|}{d_g} \cdot \phi_d\}$$

3. Luminance / Variance (w_l)

Le cœur du SVGF. On normalise par la variance locale.

$$w_l = \exp\{-\frac{|l_c - l_s|}{\phi_l \cdot \sigma}\}$$

Plus la variance est élevée, plus le filtre autorise le flou pour lisser le bruit.

Couleurs et variance finales

La couleur finale est :

$$finalColor = \frac{\sum c_s \cdot w}{\sum w}$$

Et la nouvelle variance débruitée est :

$$finalVariance = \frac{\sum \sigma^2}{\sum w^2}$$

On applique un débruitage sur le guide du débruiteur.

Détails d'implémentation

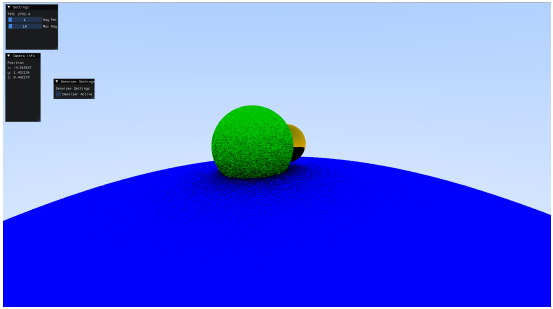
- **Langage :** C++ & GLSL (Compute Shaders).
- **API :** OpenGL 4.6.
- **Matériel :** AMD Ryzen 5 7600 + **Radeon RX 6800**.
- **Librairies :** GLFW, GLM, STB, ImGui.

Comparaison de Performance :

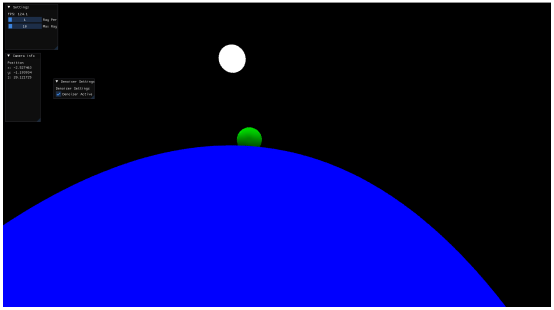
Méthode	SPP (Rayons/Pixel)	FPS Moyens
Pathtracing Brut	1000	$\sim$ 6.7 FPS
Pathtracing Brut	98	$\sim$ 1400 FPS (Bruité)
SVGF (Final)	**1**	$\sim$ 124 **FPS**

Table – Comparaison sur scène simple (4 sphères)

Comparaison Visuelle

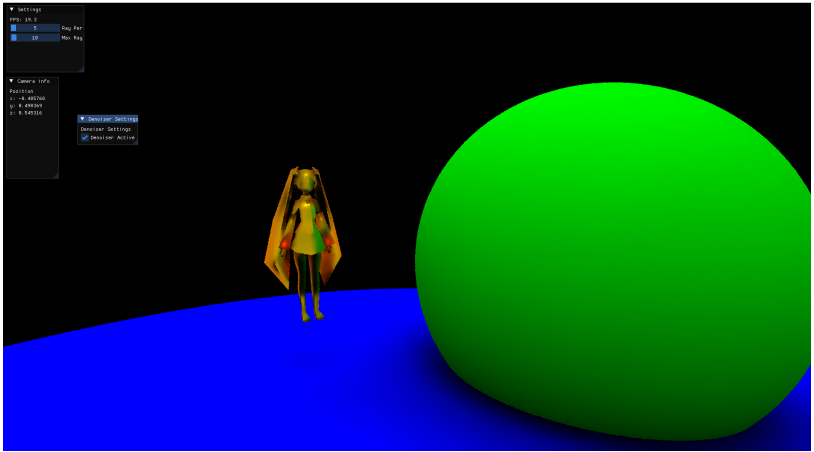


Entrée : 1 SPP (Bruité)



Sortie : SVGF Débruité

Résultats sur maillage (Hatsune Miku)



Scène avec 595 triangles, éclairage indirect uniquement.
Rendu à $\sim$ 20 FPS avec 1 rayon par pixel.

Conclusion

Résumé :

- Le SVGF rend le Pathtracing plus viable pour le temps réel.
- Qualité visuelle supérieure au rasterization classique.
- Passage de 6 FPS (1000 spp) à 120+ FPS (1 spp + SVGF).

Travaux futurs :

- Implémentation d'un **BVH** (Bounding Volume Hierarchy) pour supporter des scènes complexes.
- Ajout de l'éclairage direct (Next Event Estimation).
- Amélioration de la gestion des réflexions spéculaires (qui deviennent floues avec le SVGF actuel).

Merci de votre attention

Avez-vous des questions ?