3D Graphics: Improving the rendering pipeline using MSAA

BENAOUDIA Ilian

CHEN Eric

HELLO Adam

ROUILLARD Hugo

1. Introduction

Dans les pipelines de rendu temps réel, un problème récurrent de qualité visuelle est celui de l'aliasing. Ce phénomène se manifeste par des bords en "escaliers" sur les contours inclinés ou courbes des objets 3D. Il résulte d'un échantillonnage spatial insuffisant, où chaque pixel évalue la scène à un seul point (en son centre), ce qui ne permet pas de capturer correctement les détails fins.

Pour atténuer ce problème, plusieurs techniques d'anti-aliasing ont été développées. Le MSAA (Multisample Anti-Aliasing), propose une solution intermédiaire entre le suréchantillonnage complet (SSAA), trop coûteux, et les méthodes de post-traitement : il effectue plusieurs tests par pixel lors de la rastérisation, ce qui permet de lisser les bords géométriques sans multiplier les calculs de couleur.

Dans ce projet, nous avons intégré le MSAA dans un pipeline de rendu simple. Nous comparons leur impact sur la qualité visuelle et les performances, dans le but de réduire efficacement l'aliasing tout en préservant une bonne efficacité.

2. Contexte d'application

Le problème de l'aliasing est important dans de nombreux domaines de synthèse d'image, en particulier pour les applications interactives et temps réel notamment :

- **Jeux vidéo 3D et simulations interactives :** la qualité d'image a un impact direct sur l'expérience utilisateur. Le crénelage peut rendre les jeux moins immersifs, surtout à faible résolution.

- **Modélisation, visualisation scientifique :** L'aliasing nuit à la lisibilité des arêtes, surtout lors des zooms ou rotations. Le MSAA est souvent utilisé en temps réel pour améliorer le confort visuel sans altérer la géométrie, essentiel pour évaluer précisément les modèles.

- **Applications de réalité virtuelle et réalité augmentée :** l'image est proche des yeux et très fluide. Le MSAA, souvent en 4×, est préféré aux méthodes post-traitement qui floutent l'image, car il offre une meilleure netteté sans trop nuire aux performances.

- **Rendus de haute qualité :** Dans le domaine du rendu photoréaliste, on cherche à éliminer l'aliasing pour obtenir des images les plus précises possible. On utilise souvent le SSAA ou des filtres avancés mais le MSAA peut servir d'optimisation.

3. Méthode

Le MSAA repose sur le fait de subdiviser chaque pixel en plusieurs sous-échantillons pour mieux détecter les bords. L'implémentation se déroule en plusieurs étapes :

3.1. Génération des échantillons

Au lieu de réaliser un seul test de couverture au centre du pixel, on définit plusieurs positions d'échantillonnage à l'intérieur de chaque pixel.

- **MSAA 4x :** Quatre échantillons, positionnés aux coordonnées relatives (0.25, 0.25), (0.75, 0.25), (0.25, 0.75) et (0.75, 0.75).
- **MSAA 8x :** Huit échantillons, à des positions choisies pour éviter les alignements trop réguliers, permettent de couvrir le pixel de manière plus fine en répartissant les échantillons de façon moins symétrique.

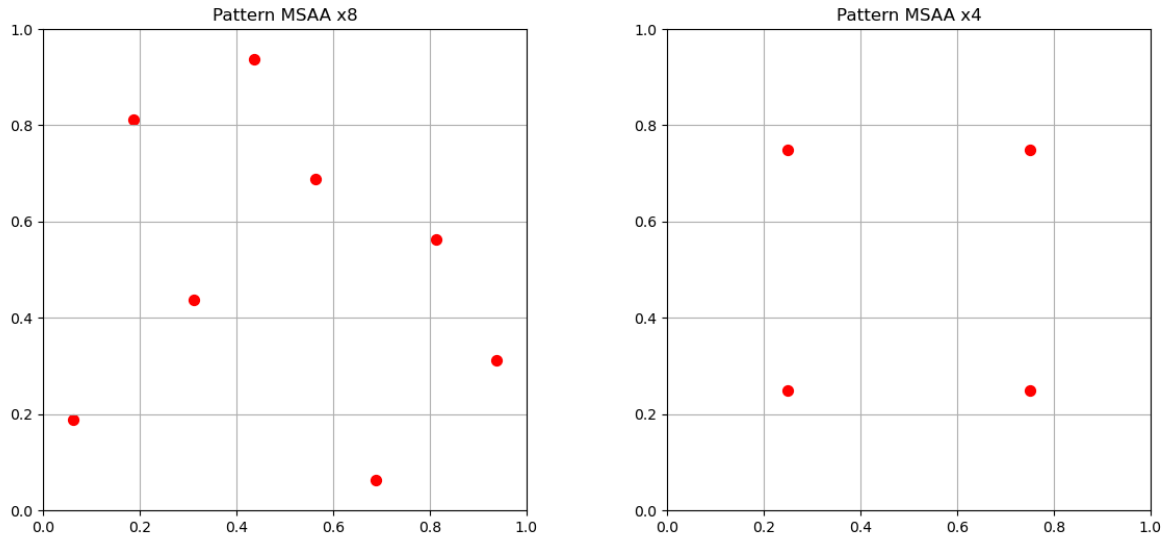

Figure 1: Pattern MSAA

3.2. Rasterisation et tests de couverture

Pendant la rasterisation, chaque triangle de la scène est traité de manière à tester la couverture de chacun des sous-pixels définis par les positions d'échantillonnage. Pour chaque sous-échantillon, la fonction de test (basée sur le calcul de l'aire via la méthode d'edge function) vérifie si ce point se trouve à l'intérieur du triangle. Si c'est le cas, le sous-échantillon est considéré comme couvert et un fragment est généré.

3.3. Stockage dans des buffers multisample

Afin de conserver les résultats, la pipeline dispose de tampons spécifiques :

- Un buffer de couleur multi-échantillon, de dimension (hauteur × largeur × N), où N est le nombre d'échantillons (4 ou 8), stocke la couleur calculée pour chaque sous-échantillon.

- Un buffer de profondeur multi-échantillon contient pour chaque sous-pixel la valeur de profondeur minimale obtenue lors des tests, garantissant ainsi une gestion fine de l'ordre des primitives.

3.4. Résolution du buffer multisample

Une fois la rasterisation terminée, l'étape finale consiste à résoudre le buffer multi-échantillon pour produire l'image finale :

- Pour chaque pixel, les N valeurs (couleur et profondeur) sont agrégées par une opération de moyenne, donnant une couleur unique par pixel.

- Cette étape permet de lisser les contours en combinant les contributions partielles des sous-échantillons, aboutissant à une transition progressive entre primitives et réduisant ainsi les effets d'aliasing.

3.5. Différences entre MSAA 4× et MSAA 8×

La différence principale entre les deux versions est dans le nombre d'échantillons :

- Le MSAA 4× nécessite 4 tests de couverture par pixel, ce qui permet de lisser les bords dans la plupart des cas avec un impact modéré sur le temps de calcul et la consommation mémoire.

- Le MSAA 8× multiplie ces tests par deux, offrant une qualité légèrement supérieure (des contours encore plus lisses) au détriment d'une charge de calcul et d'une utilisation mémoire plus élevées.

Figure 2 : Comparaison MSAA

Ces étapes illustrent comment l'algorithme du MSAA est intégré dans la pipeline graphique, depuis le choix des positions d'échantillonnage jusqu'à la production de l'image finale après résolution des buffers. Cette implémentation vise à atteindre un compromis optimal entre qualité d'image et performances, tout en étant adaptée à des environnements temps réel.

4. Évaluation expérimentale

(Dans cette section, nous prévoyons de présenter les résultats de tests pratiques de l'ajout du MSAA dans notre pipeline. Deux volets seront détaillés : les résultats visuels obtenus, et les performances mesurées.)

4.1 Résultats visuels

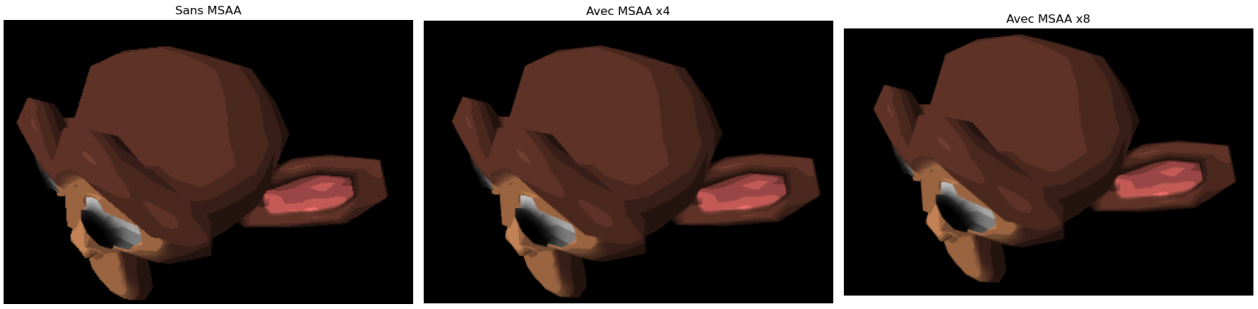

Figure 3 : Résultats sur le fichier suzanne.ply

Avec la tête de Suzanne dans trois configurations : sans MSAA, avec MSAA 4×, et MSAA 8×. Les images obtenues mettent clairement en évidence l'apport du multisampling. Sans anticrénelage, les arêtes du modèle apparaissent crénelées et les détails fins présentent un effet d'escalier très net. Avec MSAA 4×, on observe déjà une amélioration notable : les bords sont adoucis et l'objet paraît plus lisse. Avec MSAA 8×, le résultat est encore affiné, les contours sont quasiment lisses à l'œil nu, se rapprochant de la qualité qu'on obtiendrait avec un supersampling 8× complet.

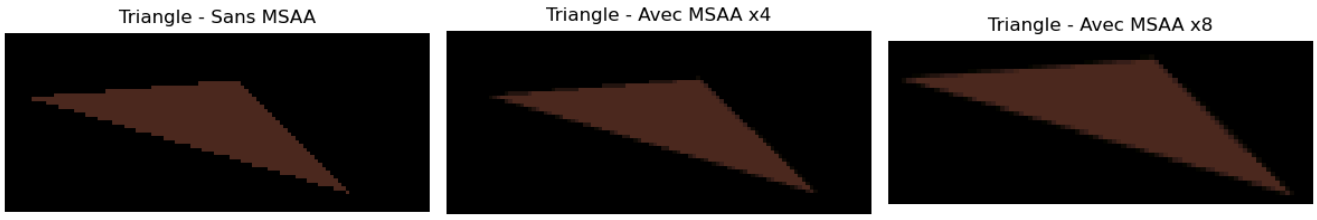

Figure 4 : Résultats plus explicite sur un seul triangle

Avec un triangle dans trois configurations : sans MSAA, avec MSAA 4×, et MSAA 8×. La différence est particulièrement visible, le rendu MSAA 8× propose une transition de pixels en dégradé de teinte, là où le rendu sans AA passe brutalement du modèle au fond, créant une ligne en escalier. En termes de qualité perçue, le MSAA 4× suffit souvent à éliminer le gros de l'aliasing, le 8× apportant un léger gain supplémentaire..

4.2 Performances

```
Temps de rendu sans MSAA : 3.1923 secondes
Temps de rendu avec MSAA x4 : 13.7143 secondes
Temps de rendu avec MSAA x8 : 27.9396 secondes
```

Figure 5 : Temps d'exécution des différent rendu

Nous avons chronométré le temps de rendu de notre pipeline pour la scène de test, en variant le mode d'anticrénelage. Sans MSAA, le rendu de l'image de taille 1280 x 720 prend 3.1923 secondes. Avec le MSAA 4× activé, le temps passe à 13.7143 s, soit une augmentation de 10.522 s. Avec MSAA 8×, le temps monte à 27.9396 s, soit 27.7473 s de plus par rapport au cas sans AA.

Ces résultats confirment que l'activation du MSAA induit un surcoût linéaire en fonction du nombre d'échantillons. Le surcoût observé entre 4× et 8× est approximativement proportionnel au double d'échantillons, ce qui est cohérent avec le fait que 8× effectue deux fois plus de tests de couverture et stocke deux fois plus de données que le 4×. Néanmoins, l'écart entre le x4 et le x8 n'est pas négligeable, montrant qu'au-delà de 4 échantillons les bénéfices visuels additionnels du MSAA peuvent coûter cher en performance.

5. Limites et alternatives

Bien que le MSAA soit efficace pour réduire l'aliasing géométrique, il présente plusieurs limites et inconvénients qu'il convient de discuter :

- **Coût en calcul :** Le MSAA augmente le nombre d'opérations de rastérisation (N tests par pixel au lieu d'un). Dans notre implémentation, nous avons observé un ralentissement à peu près linéaire avec N. Toutefois, le surcoût shader est limité aux pixels de contours multiples. Dans le pire cas (beaucoup de petits triangles entrelacés), chaque échantillon d'un pixel pourrait appartenir à un triangle différent et engendrer un appel shader séparé ; le MSAA approcherait alors le coût du SSAA complet. Dans ce cas mis à part, le MSAA est généralement plus performant que du supersampling équivalent, mais son coût n'est pas négligeable, surtout à 8×.

- **Coût en mémoire :** Le buffer multi-échantillonné consomme N fois plus de mémoire pour stocker les couleurs (et idem pour le depth buffer). Sur des écrans haute résolution, un MSAA 8× peut vite consommer une part importante de la mémoire vidéo. Des optimisations existent (par ex. ne pas stocker les échantillons inutilisés), mais en général activer le MSAA signifie accepter un usage mémoire accru.

Compte tenu de ces limites, d'autres techniques d'anticrénelage ont été développées et sont parfois préférées ou combinées avec le MSAA. Par exemple le Supersampling (SSAA), une méthode conceptuellement simple consistant à rendre l'image à une résolution N fois plus grande puis la réduire. Cela élimine pratiquement tout aliasing (géométrie + texture) au prix d'un coût énorme (N^2 pixels à shader). Le MSAA a justement été conçu pour approcher la qualité du SSAA avec un coût réduit. Le SSAA n'est utilisé qu'off-line ou pour référence qualité à cause de son coût.

En conclusion de cette analyse, le MSAA se révèle être un compromis puissant entre qualité et performance pour lisser les bords géométriques, tant que ses limites sont acceptables (mémoire, opacité, etc.) Aucune méthode unique n'est parfaite : le choix se fait en fonction du contexte d'application, d'où l'intérêt de connaître les forces et faiblesses du MSAA et des alternatives pour décider de la meilleure solution selon les besoins (qualité maximale vs. coût minimal, nature statique ou dynamique de la scène, présence de fine géométrie ou de textures détaillées, etc.).

Bibliographie

[1] T. Akenine-Möller, E. Haines, N. Hoffman, Real-Time Rendering, 4th ed., CRC Press, 2018.

[2] M. Pharr, W. Jakob, G. Humphreys, Physically Based Rendering: From Theory to Implementation, 3rd ed., Morgan Kaufmann, 2016.

[3] Khronos Group, "Multisample Anti-Aliasing (MSAA)" Vulkan Guide and Samples,

[4] NVIDIA, "Coverage Sampling Antialiasing (CSAA)" Whitepaper, 2006.

[5] AMD, "Enhanced Quality Anti-Aliasing (EQAA)" Developer Guide, 2011. https://gpuopen.com/eqaa/

[6] Saphirre Nation, "Anti-aliasing techniques comparison" CryEngine SDK Documentation.

[7] B. Golus, "Anti-aliased Alpha Test: The Esoteric Alpha To Coverage," Medium, 2020.

[8] Alexander Reshetov, "Morphological Antialiasing". High Performance Graphics 2009.

[9] MJP, "A Quick Overview of MSAA" My Name Is MJP 2016.