Ray Tracing

Luc Anchling et Timothée Teyssier

October 22, 2023

1 Introduction

La technique de calcul d'optique par ordinateur dite de *lancer de rayon* (ou *raytracing* en anglais) est une méthode utilisée lors du rendu en synthèse d'image pour simuler des principes physiques optiques (réfraction, réflexion...) pour rendre plus réaliste la scène générée.

Elle consiste, à l'inverse de la réalité où la lumière va de l'objet à l'oeil (ou caméra dans notre cas d'étude), de simuler le parcours inverse de la lumière en faisant partir les rayons lumineux de la caméra vers les objets puis vers les lumières. Dans la pratique, pour chaque pixel de l'image, on lancera un (ou plusieurs) rayon(s) depuis le référenciel caméra dans la scène 3D pour calculer les points d'intersections avec des primitives appartenant à cette dernière et estimer en continuant le chemin de ce rayon en direction des sources lumineuses la luminosité en ce point.

Dans notre cas d'étude, la scène où l'on va implémenter le raytracing est composée d'objets relativement simples (3 sphères et 1 plan - voir figure 1). La description du principe de calcul des intersections entre nos rayons et ces deux types de primitives (sphère et plan) sera décrit dans la suite.

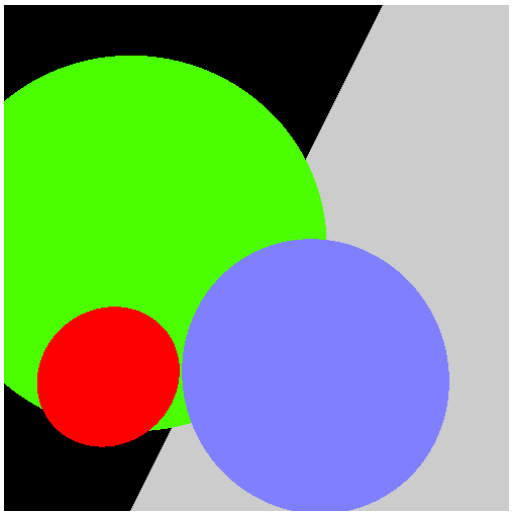

Figure 1: Scène d'origine composée de 3 sphères et d'un plan

2 Calcul d'intersections des primitives

Les intersections seront déterminées entre notre rayon que nous lanceront et les différentes primitives qui composent notre scène : un plan et des sphères.

2.1 Avec un plan

Pour un plan $\mathcal{P}$ défini $\forall x_p \in \mathcal{P}$ tq $< x - x_p, n_p >= 0$ avec n_p la normale du plan. L'intersection avec une droite passant par x_0 est donnée par le système suivant :

$$\begin{cases} x(t) = x_0 + tu \\ < x(t) - x_p, n_p >= 0 \end{cases} \tag{1}$$

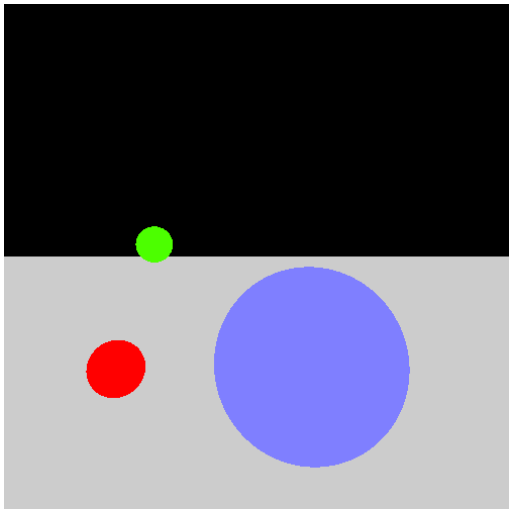

Figure 2: Intersection entre le rayon et le plan

2.2 Avec une sphère

L'intersection entre une sphère de centre x_0 et de rayon r avec une droite passant par x_s et de vecteur directeur u est donnée par la solution du système.

$$\begin{cases} x(t) = x_0 + tu \\ ||x(t) - x_s|| = r \end{cases} \tag{2}$$

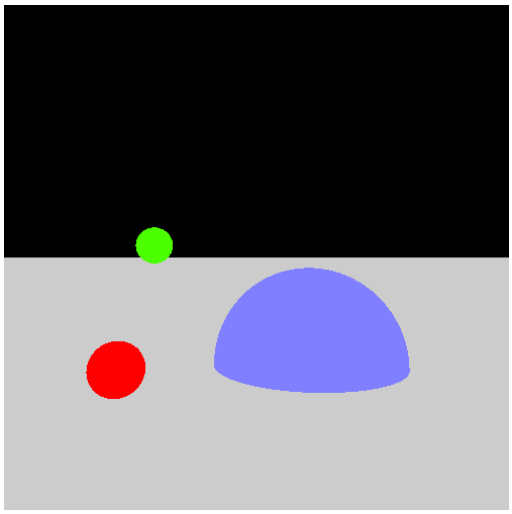

Figure 3: Intersection la plus proche entre le rayon et le plan et le rayon et les sphères

3 Ombrage

Pour ajouter l'ombrage, nous allons nous intéressé à la fonction *isinshadow()*.Nous devons déterminer si le point courant se situe dans l'ombre d'un objet par rapport à la lumière. La méthode ici est de déterminer si le rayon de direction p_light - p (avec p notre point courant) intersecte un objet entre ces deux points si l'intersection est bien présente, alors cela signifie que le point p se situe dans l'ombre. On observe ainsi comme résultat la figure 4 où nous pouvons voir très clairement les points qui se situent dans l'ombre ou non, cependant ici notre image n'est pas encore réaliste puisqu'il manque l'illumination afin de permettre que le contraste entre la partie dans l'ombre et la partie illuminée soit faible.

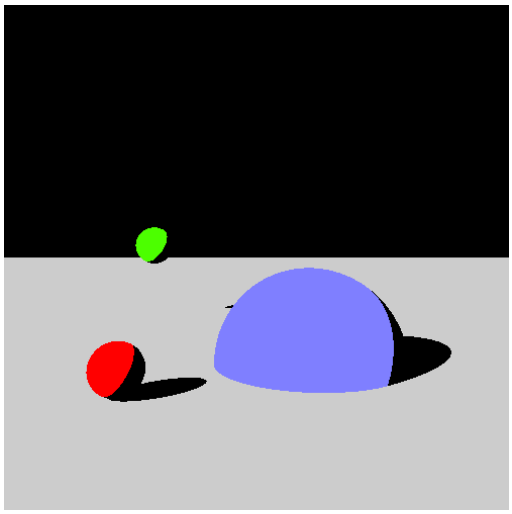

Figure 4: Affichage de l'ombrage pour les sphères et le plan

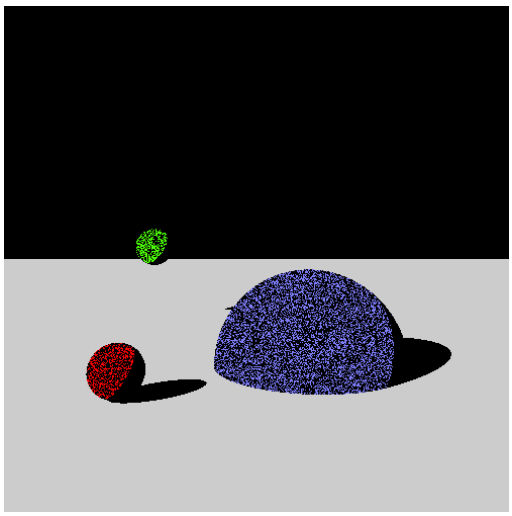

Figure 5: Affichage de l'ombrage pour les sphères et le plan sans offset et normalisation

Attention à bien utiliser la fonction *offset()* sur notre rayon tout en pensant à bien normaliser le rayon, sinon nous pouvons obtenir le résultat sur la figure 5

4 Illumination

Dans cette partie, nous commençons tout d'abord par calculer l'illumination de Phong au point courant en fonction de la position de lumière et de la caméra, tout en prenant compte de la couleur d'origine de l'objet plus sa matière. Nous pouvons finalement observer sur la figure 6 les tâches de spécularité sur nos trois sphères, tout comme la couleur ambiante et la couleur diffuse.

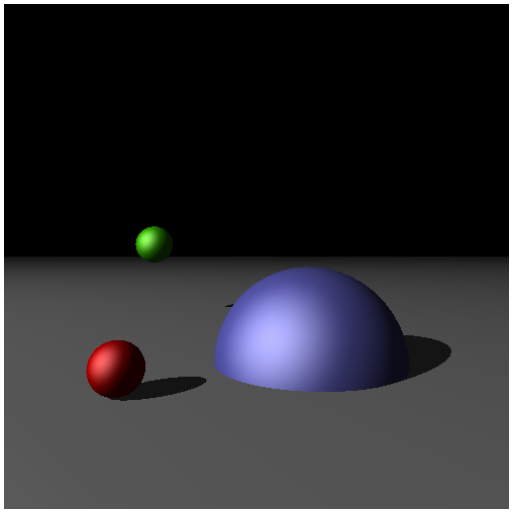

Figure 6: Ajout de l'illumination de Phong de la scène sur notre image 2D

Maintenant que l'ombrage et l'illumination sont bien présentes sur notre image, nous pouvons rajouter de nouveaux éléments afin de rendre notre image le plus réaliste possible. La prochaine étape consiste à intégrer la réflexion.

5 Réflexion

On va considérer maintenant que chaque rayon poursuit sa course après la première primitive rencontrée pour ainsi implémenter une notion de réflexion. Dans les faits, pour chaque rayon incident, on va lancer un second rayon symétrique par rapport à la normale de contact, ce procédé est généralisable à N lancées de rayons consécutifs pour modéliser le phénomène de réflexion en chaîne. Une notion d'atténuation rentrera en compte lors de la pondération effectuée pour le calcul de la couleur en chaque pixel.

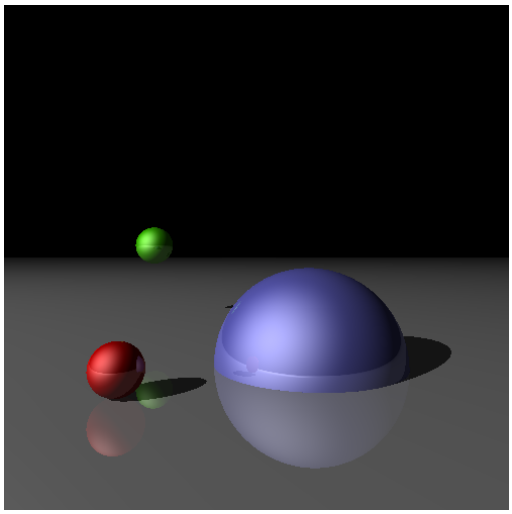

Figure 7: Ajout de la réflexion sur notre image avec $N = 5$ niveaux de réflexions

La réflexion est bien présente sur notre image résultante (voir figure 7), maintenant puisqu'un seul rayon est lancé par pixel, on observe des artefacts visuels de type crénelage sur les contours des objets (et de leurs ombres), dans la suite nous verrons une technique permettant de résoudre ce problème : l'anti-aliasing.

6 Anti aliasing

Pour palier au problème de crénelage, on va réaliser un échantillonage de la scène sous différentes perspectives en lançant plusieurs rayons depuis chaques pixels dans des direction variables autour de la direction principale. On va ensuite calculer la couleur en chaque pixel en pondérant la couleur de chaque échantillons ainsi créés.

Cela va permettre d'améliorer la qualité visuelle en la rendant plus lisse en réduisant le nombres d'artefacts visuels de type crénelage. Cependant, cela exige davantage de puissance et de temps de calcul.

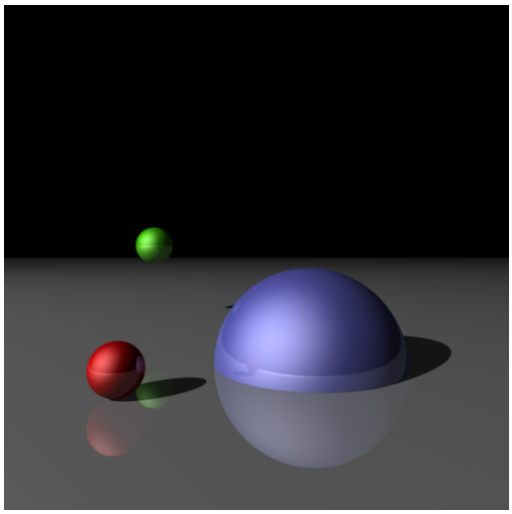

Figure 8: Ajout de l'Anti aliasing

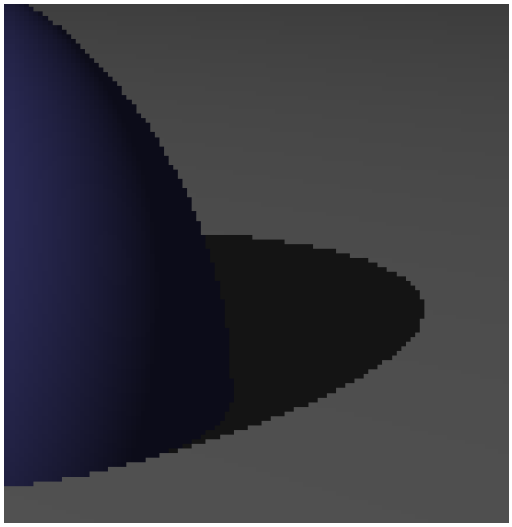

Figure 9: Sans Anti-aliasing

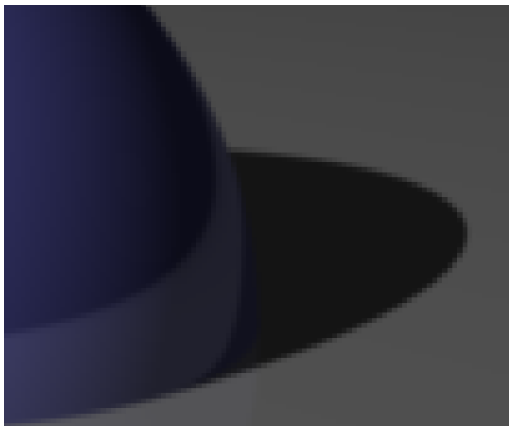

Figure 10: Avec Anti-aliasing

On observe bien une atténuation du crénelage présent sur les contours des objets et de leur ombres. Il a été lissé avec un effet de flou au niveau de ces dernières.

7 Travaux supplémentaires

7.1 Ombrage doux

Pour l'instant, nous utilisons des lumières ponctuelles qui entraînent alors des transitions franches sur nos objets au niveau des ombres comme observé sur la figure 7. Pour remédier à ce problème et obtenir des résultats plus réalistes nous pouvons ajouter plusieurs lumières se situant très proches de la lumière ponctuelle principale, cela va ainsi créer différents niveaux de transition d'ombres équivalent à un léger flou que nous pouvons observer sur la figure suivante. A chaque fois que nous ajoutons des lumières à notre lumière ponctuelle d'origine, nous devons bien penser à pondérer la puissance de toutes les lumières ponctuelles se trouvant dans la scène.

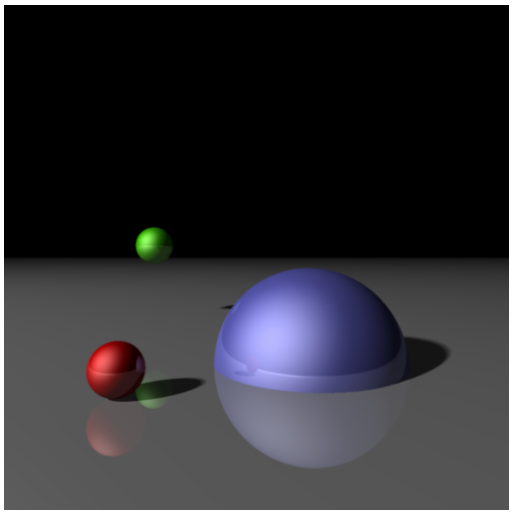

Figure 11: Ombrage doux avec trois lumières ponctuelles

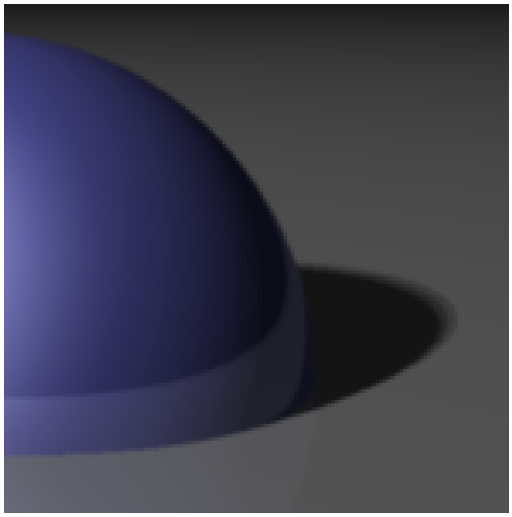

Figure 12: Zoom ombrage doux avec trois lumières ponctuelles

En ajoutant trois autres sources de lumières ponctuelles, on obtient ainsi ombres très distinctes comme si notre objet était entouré de grandes lumières dans la vie réelle.

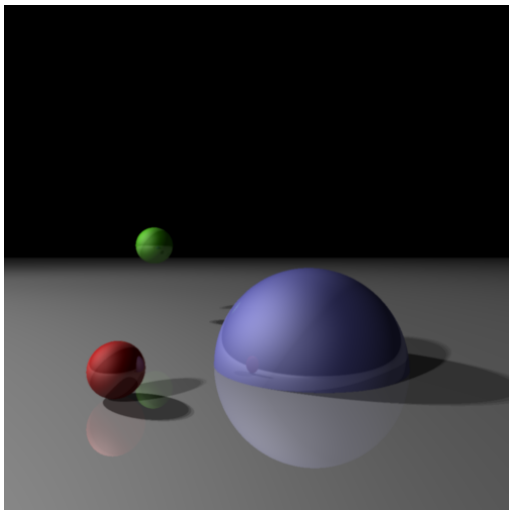

Figure 13: Zoom ombrage doux avec six lumières ponctuelles

7.2 Ajout de triangles à notre scène

Nous pouvons également ajouter un nouveau type de primitives dans notre scène : les triangles, qui sera intéressant par la suite pour afficher des maillages grâce à la méthode du ray-tracing. Pour cela nous devons tout d'abord calculer l'intersection entre un triangle et un rayon afin de détecter si nos rayons intersectent le triangle. On observe le résultat suivant.

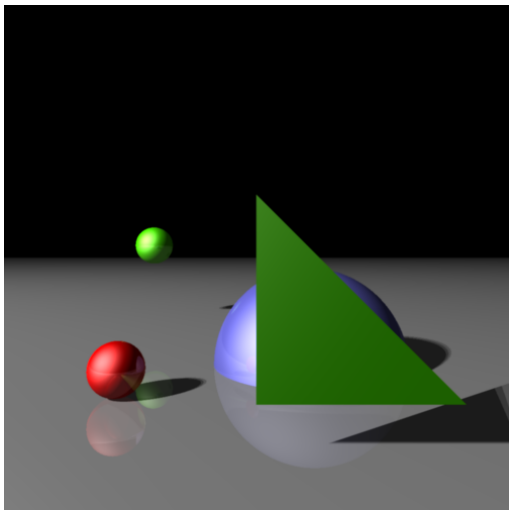

Figure 14: Ajout d'un triangle à notre scène

Nous pouvons également afficher des maillages contenant plusieurs triangles, pour cela, nous devons tout d'abord charger le modèle, puis dans une boucle ajouter tous les triangles composant le modèle.

8 Conclusion

En conclusion, le raytracing est une technique d'informatique graphique essentielle pour simuler des phénomènes optiques, offrant un réalisme visuel exceptionnel. Tout au long de ce TP, nous avons exploré plusieurs facettes du raytracing et mis en œuvre des méthodes pour améliorer ses capacités.

Nous avons débuté par l'intersection rayon-primitive, permettant de déterminer où les rayons lumineux interagissent avec les objets de la scène. En ajoutant l'ombrage, nous avons pu distinguer les zones éclairées de celles dans l'ombre, renforçant la crédibilité visuelle. Puis l'illumination de Phong a ensuite été intégrée pour calculer les couleurs des surfaces en fonction de la lumière, de la caméra et des propriétés des matériaux, améliorant significativement le réalisme de l'image.

La réflexion a apporté un niveau supplémentaire d'authenticité, permettant aux rayons lumineux de rebondir sur les surfaces, mais cela a introduit des crénelages sur les contours. Pour remédier à ce problème, l'anti-aliasing a été implémenté en échantillonnant la scène à partir de différentes perspectives, offrant une apparence plus lisse. Pour encore améliorer le rendu des ombres, nous avons enfin ajouter l'ombrage doux en utilisant plusieurs sources lumineuses ponctuelles pour créer des transitions d'ombres plus réalistes.

Cependant, il est important de noter que le raytracing exige des ressources de calcul substantielles, nécessitant des compromis entre qualité et efficacité. Malgré ces défis, le raytracing demeure une méthode puissante pour générer des images 3D réalistes, et son évolution continue promet de repousser les limites du réalisme visuel dans l'informatique graphique.