Informatique Graphique

Rapport de TP

Informatique Graphique

Élève :
Cecchinato Rémi

Enseignant :
Bonneel Nicolas

Introduction

Ce rapport présente la réalisation de mon raytracer et des résultats que j'ai obtenus.

Le raytracer a été entièrement programmé en language C. Le code peut-être retrouvé sur github à l'adresse $https : //github.com/RemiCecchinato/raytracing$. Pour compiler le projet, il suffit de compiler le fichier $main.c$ avec les mêmes options que la ligne de commande présente dans le fichier $build.sh$.

Raytracer

Le premier objectif était de dessiner une sphère. C'est ce qui est fait sur la figure 1 avec une sphère rouge sur un fond blanc. La sphère ressemble plus à un disque tant qu'il n'y as pas d'effet de lumière. C'est pourquoi j'ai rajouté une source de lumière sur la figure 2.

FIGURE 1 – Sphère rouge sur fond blanc.

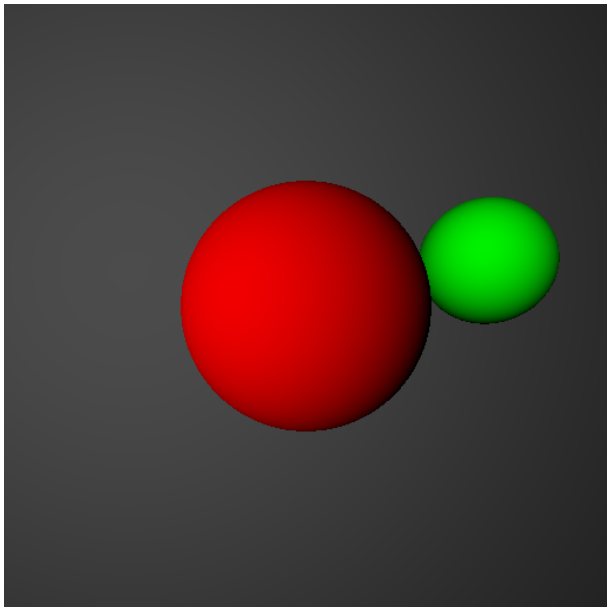

FIGURE 2 – Deux sphères éclairées par une lumière blanche.

Afin d'ajouter au réalisme, j'ai rajouté de la correction gamma et des ombres au raytracer. C'est ce qui est visible sur la figure 3. L'intérêt d'utiliser du raytracing plutôt qu'une autre méthode de rendu est sa facilité à prendre en compte les réflexions. J'ai donc ensuite ajouté des réflexions visibles sur la figure 4.

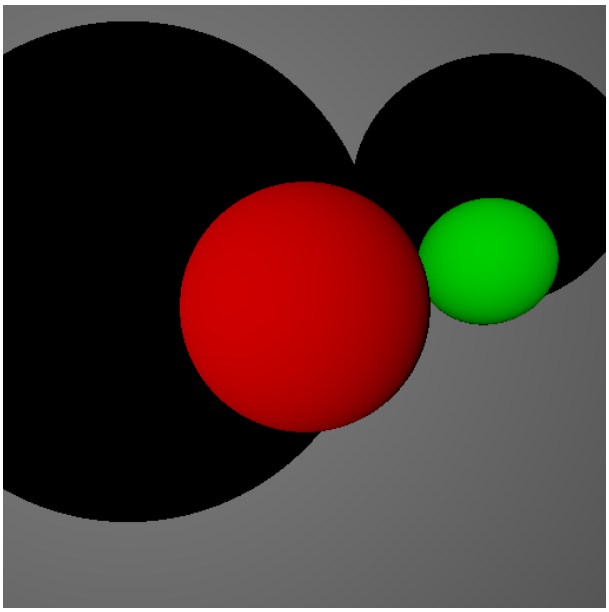

FIGURE 3 – Ajout de la correction gamma et des ombres.

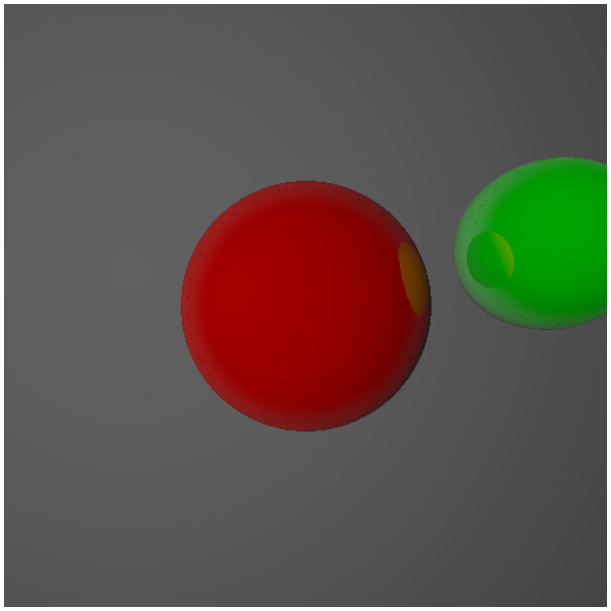

FIGURE 4 – Ajout des réflexions.

Les deux types de surfaces que j'ai ensuite rajouté sont les mirroirs et les objets transparents sur la figure 5. Jusqu'ici toutes les réflexions ne se faisaient que selon une seule direction. Pour ajouter au réalisme, il est nécessaire de modéliser des réflexions où les rayons peuvent rebondir dans toutes les directions. C'est ce qui est présenté sur la figure 6. Alors que tous les rendus précédents ne nécessitaient que quelques dizaines de mili-secondes pour être calculés, ce dernier demande environ 1 minute 30 après parallélisation. Cela est dû au fait que de nombreux rayons par pixels sont nécessaires pour avoir une belle image à cause de l'aléa introduit lors des rebonds. Cela combiné à une direction aléatoire lors du lancement initial du rayon permet cependant de réaliser un anticrénelage. L'intensité de la lumière a été modifiée entre le rendu de ces deux images.

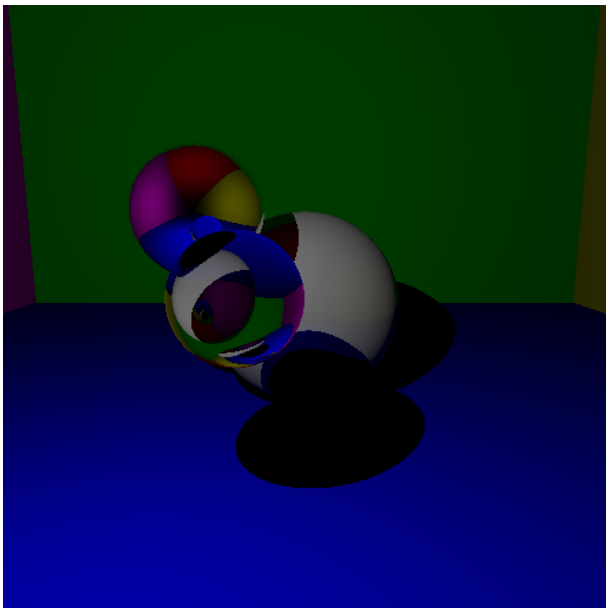

FIGURE 5 – Ajout de la transparence et des surfaces mirroitantes.

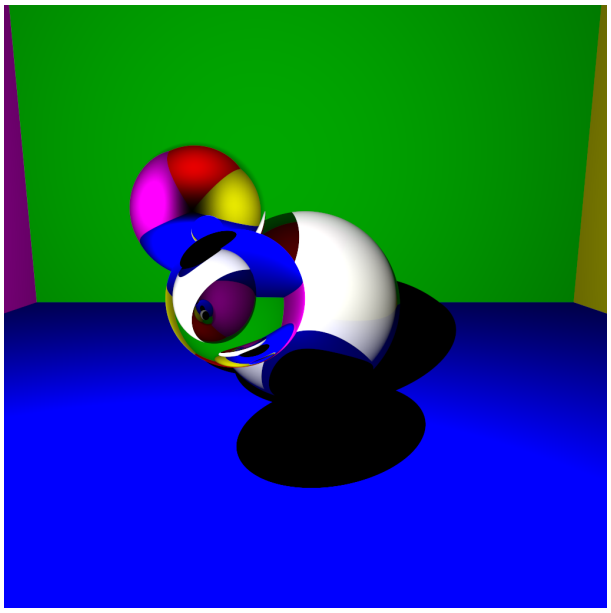

FIGURE 6 – Ajout des réflexions dans des directions aléatoires

Enfin, le dernier objectif était de dessiner un modèle 3D grâce au raytracing. J'ai donc réalisé le rendu d'un chien figure **??** sans les textures mais avec uniquement le maillage de base. Ce premier rendu nécessitait environ une demi-heure de calculs. Pour palier à ce problème j'ai divisé le maillage récursivement et spatialement afin de ne tester que les quelques triangles que chaque rayon est susceptible de toucher. Cela à permis de réduire les temps de calcul à environ une minute. Une fois que ces mécanismes de base ont fonctionné, j'ai rajouté la texture du chien sur la figure 8.

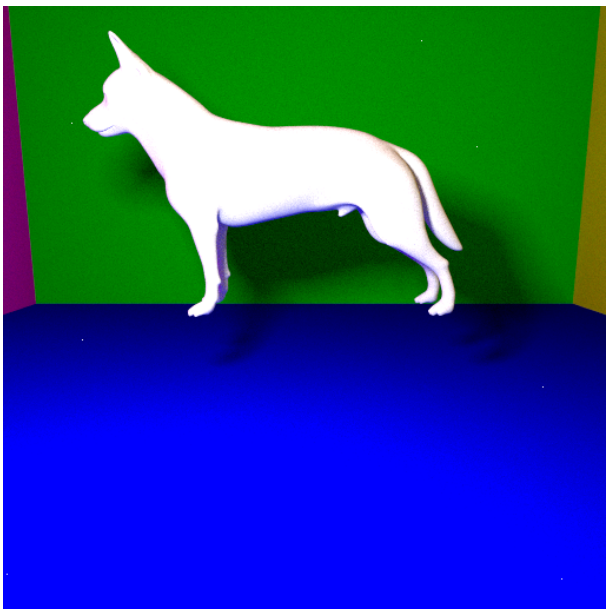

FIGURE 7 – Premier rendu du chien.

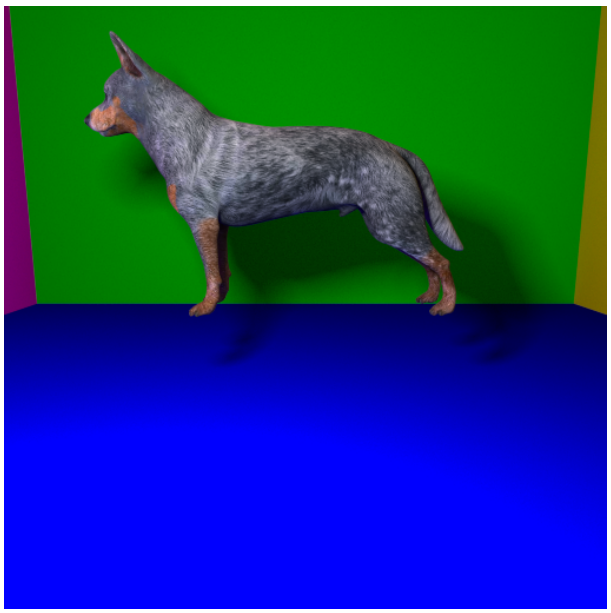

FIGURE 8 – Ajout de la texture.

Il est cependant possible de remplacer la texture du chien par d'autres matériaux, comme sur cette dernière image où j'ai rajouté un chien miroir et un chien transparent.

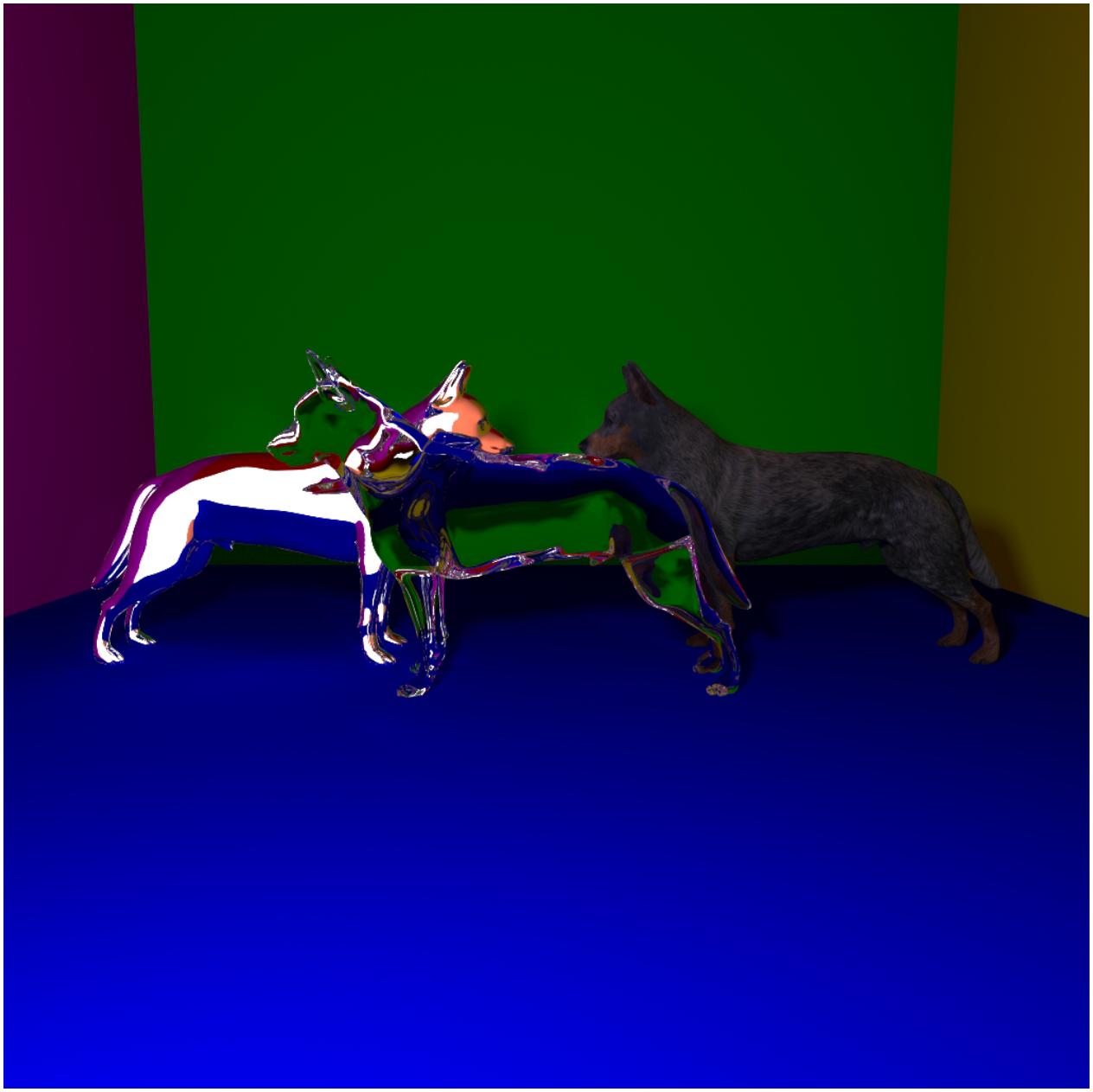

FIGURE 9 – Qui n'a jamais rêvé d'un chien miroir ou d'un chien transparent après tout ?

Conclusion

J'ai implémenté tout ce qui a été demandé dans le cours à l'exception de la profondeur de champs.