# Projet Raytracing - rendu final

Université de Montpellier

*Léa Serrano M1 IMAGINE*

Lien de mon git pour ce projet : https://github.com/LeaSerrano/M1-IMAGINE-Prog3D-LancerRayon.git

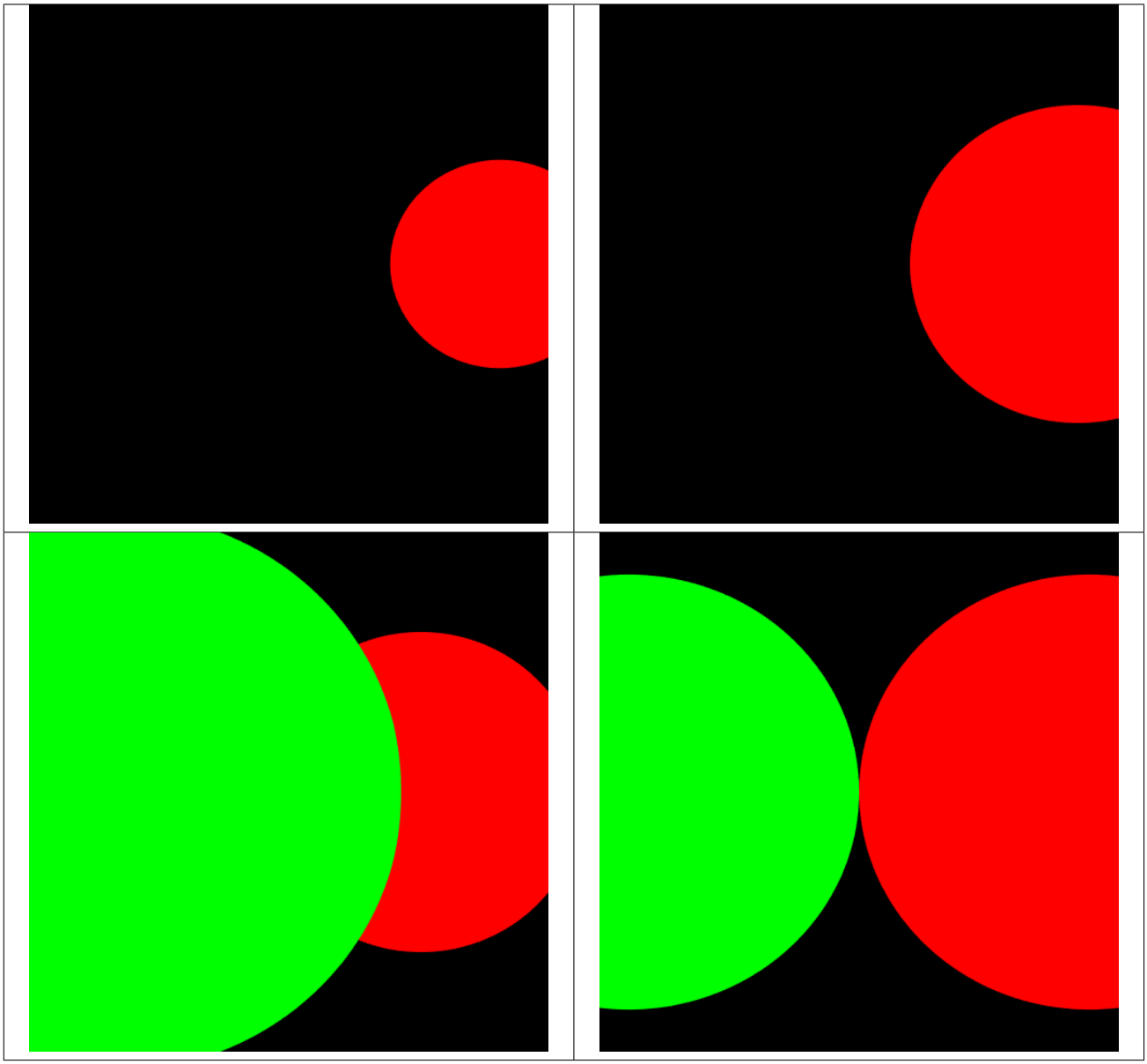
## Table des matières

# 1 Phase 1

**Intersection rayon-sphère :**

Pour commencer, nous devions réaliser l'intersection du rayon avec la sphère. Ce qui nous indique que cela fonctionne, c'est le fait qu'on voie nos sphères apparaitre sur le rendu.

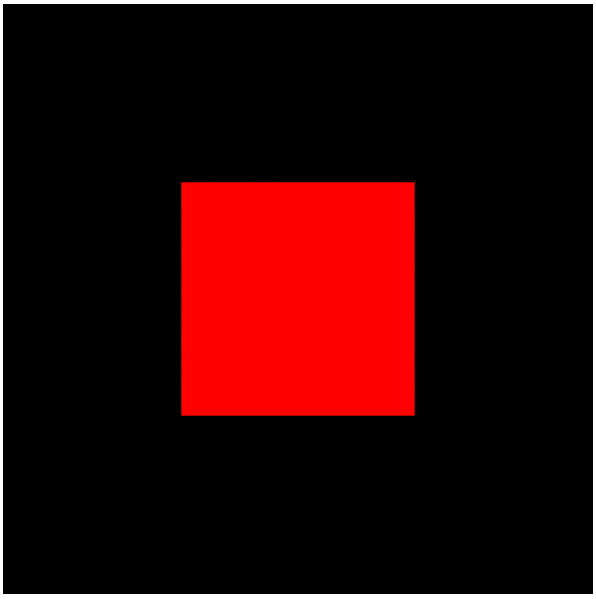
Voici le réusltat obtenu :



On voit ici que cela fonctionne car on a bien nos sphères qui sont présentes sur le rendu.

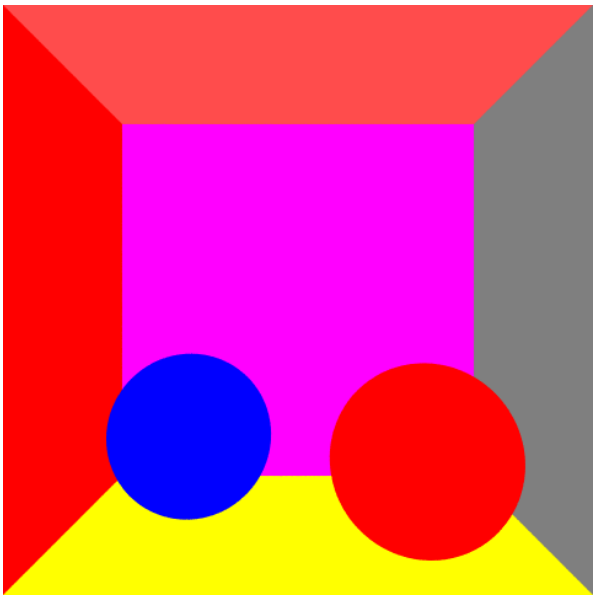
**Intersection rayon-carré :**

Ensuite, nous avons réalisé la même chose mais avec les carrés :



**Intersection rayon-carré et rayon-sphère (la boîte de Cornell) :**

Lorsque l'on fait notre rendu sur la boîte de Cornell du projet, on va faire à la fois une intersection avec des sphères et à la fois avec des carrés, voici le résultat obtenu :

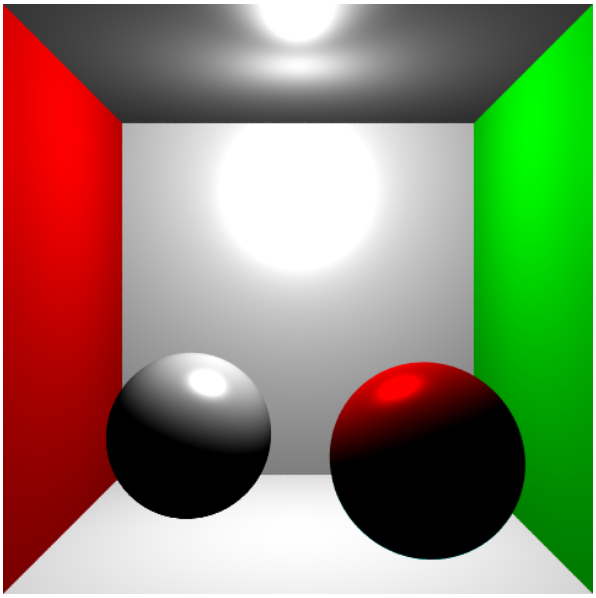


## 2 Phase 2

**Illumination de Phong :**

Pour cette seconde phase, nous avons tout d'abord calculé l'illumination de Phong. Cela se fait très simplement en ajoutant les composantes diffuses, spéculaires et ambientes de nos objets en fonction de la lumière de la scène.

Voici le rendu que j'ai obtenu :

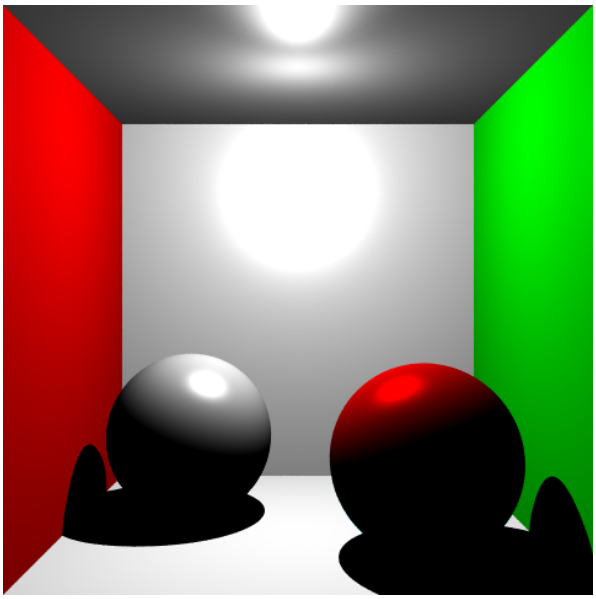


On voit que cela a créé des ombres et de la brillance sur les objets de la scène.

**Ombres :**

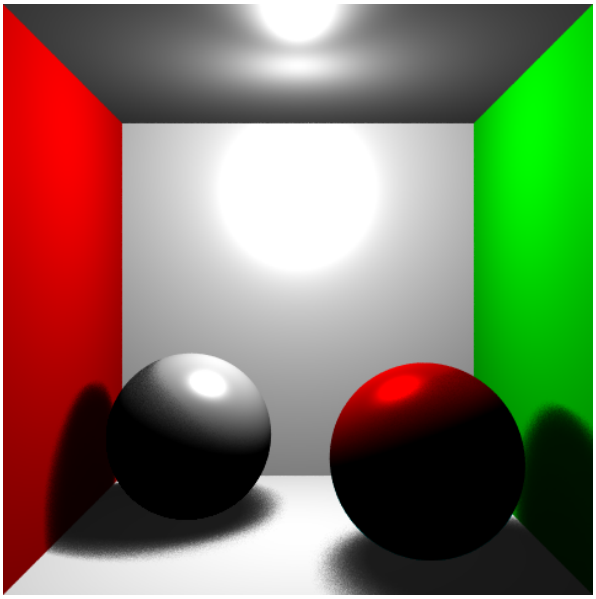
Pour ajouter encore plus de réalisme à notre scène, il nous manque l'ajout d'ombres dans notre scène. Pour calculer nos ombres, on doit tester si le point d'intersection de notre rayon a un objet entre lui et la lumière, si c'est le cas alors le point fait partie de l'ombre.

Voici le résultat que j'ai obtenu :



**Ombres douces :**

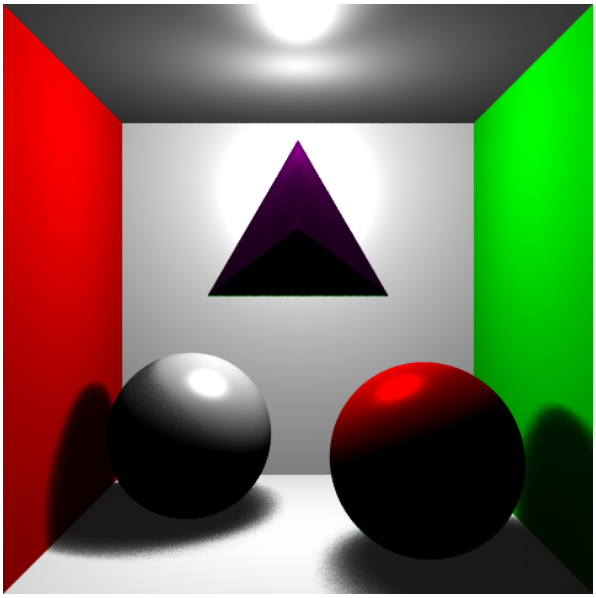
Ensuite, pour améliorer le rendu visuel de notre scène, on a calculé des ombres douces. Cela fonctionne de la même manière qu'avant mais nous allons calculer une pénombre en envoyant depuis notre point d'intersection, plusieurs rayons aléatoirement.



# 3 Phase 3

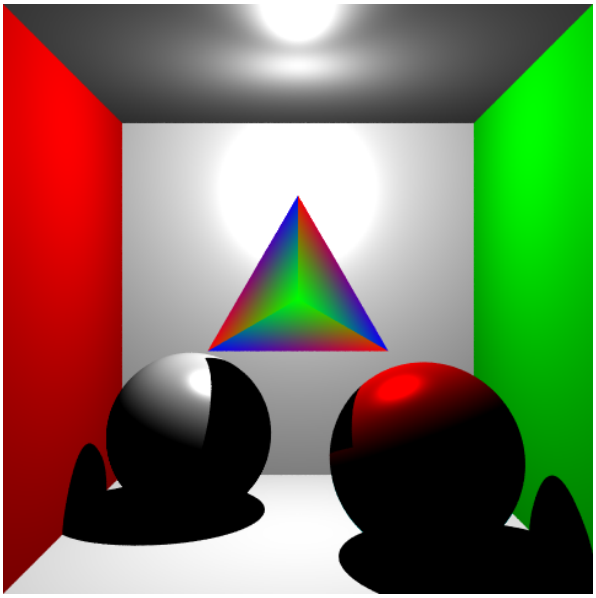
**Chargement d'un maillage :**

Il fallait, en premier lieu, calculer les intersection entre notre rayon et le maillage importé.
Voici le résultat que j'ai obtenu :



Ensuite, il fallait interpoler les valeurs des sommets, je l'ai fait en utilisant les normales du maillage

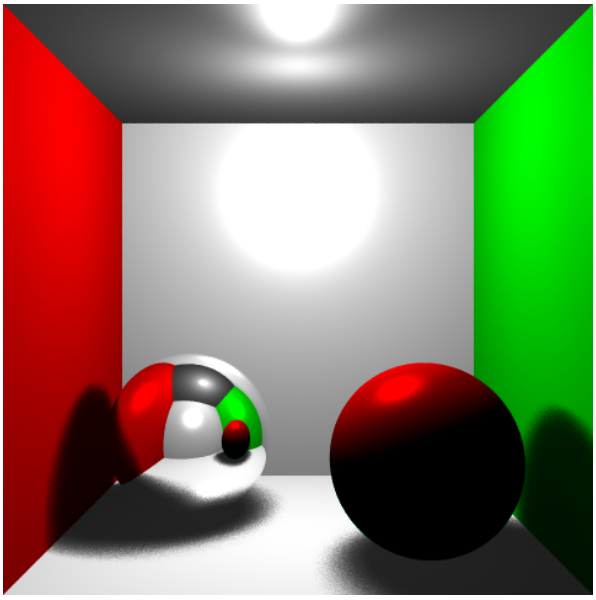
chargé :

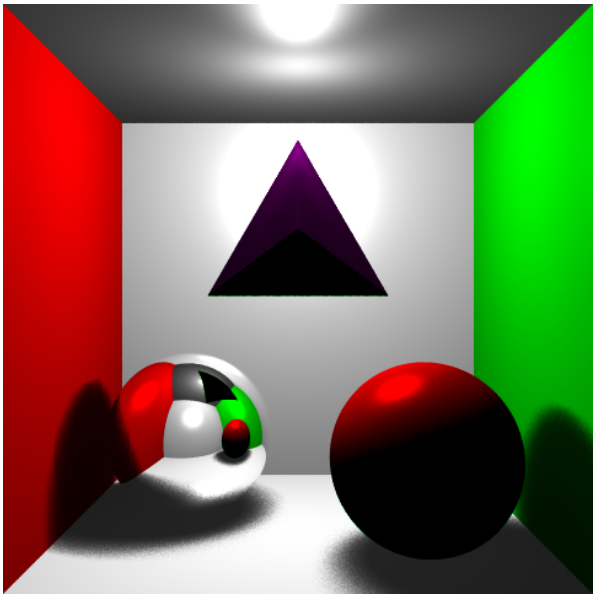


**Sphère miroir :**

Ensuite, nous devions réaliser une sphère réfléchissante (c'est à dire une sphère miroir). Cela se fait en calculant la réflection de notre point d'intersection.

Voici le résultat que j'ai obtenu :

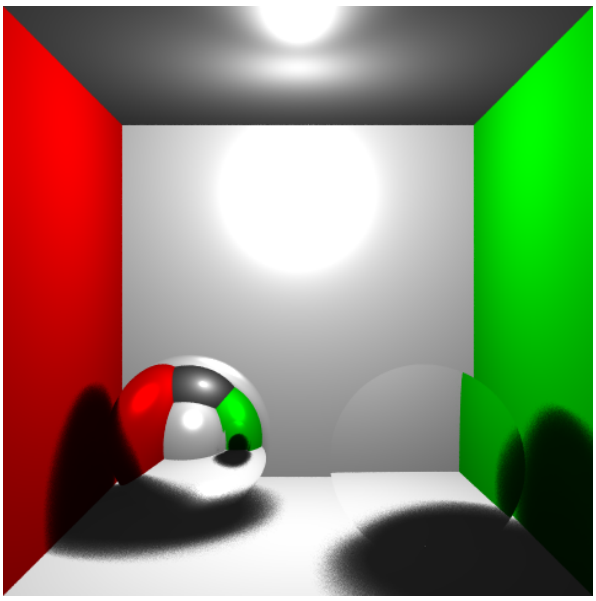


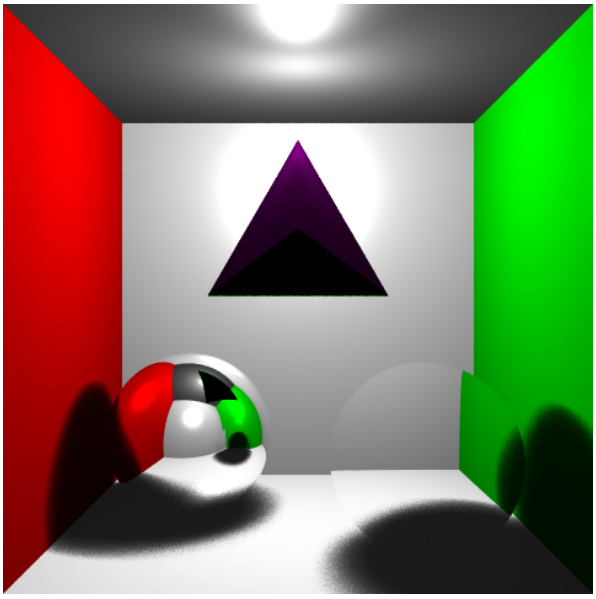


**Sphère transparente :**

Nous avons aussi réalisé une sphère transparente, cela se fait en calculant la réfraction de notre point d'intersection.

Voici le résultat obtenu :





**Kd-Tree :**

J'ai aussi réalisé un Kd-Tree. Il fallait tout d'abord construire notre Kd-Tree puis lui donner une liste de points qui vont correspondre aux éléments que l'on veut mettre dans la boite englobante.

Ensuite on récupère une valeur pour l'intersection du Kd-Tree (grâce à l'intersection de la boite englobante) et on va la comparer aux t que l'on obtient lorsque l'on calcule les intersections de la scène.

Dans mon code, je n'ai ajouté que mon mesh dans le Kd-Tree. Mon code n'est pas optimal puisque pour afficher l'image ci-dessous, sans le Kd-Tree cela met 33 secondes et avec cela met 32 secondes. Aussi sur le rendu avec Kd-Tree, on a un petit glitch sur le mesh.

Malgré le fait que mon Kd-Tree n'est pas très optimisé, la structure est tout de même là et est fonctionnelle.