CSE306 Ray Tracer Project Report

Jasmine Watissee (BX24)

May 2024

1 Introduction

This project is aimed at implementing a simple ray tracer in C++. This report contains a longer description of my work which is present on my GitHub (https://github.com/jasminewatissee/CSE306-Computer-Graphics). I will describe in more details the steps and features of my project.

2 Code Organisation

All of my code is present in main.cpp. It is organised in classes: one for the scene, one for the spheres, one for the triangle mesh, one for the rays, ... It can be run by simple running ./raytracer. The content of the scene can be modified in the function main(). I also have a soft_shadow.cpp which contains my attempt at implementing soft shadows which unfortunately does not work.

3 Spheres

The first step was to implement the ray tracer and being able to create and render spheres on my image. I started by creating different classes for each of my element: a Scene class, a Sphere class and a Ray class. I used spheres to create a basic scene composed of walls, a floor and a ceiling. And then added spheres in the middle of the scene.

We can also play with the surfaces and how the light reacts on those surfaces. The first step was to create diffuse surfaces, but then we can also create fully reflective surfaces, this looks like mirror surfaces. And we can also create refractive surfaces like glass.

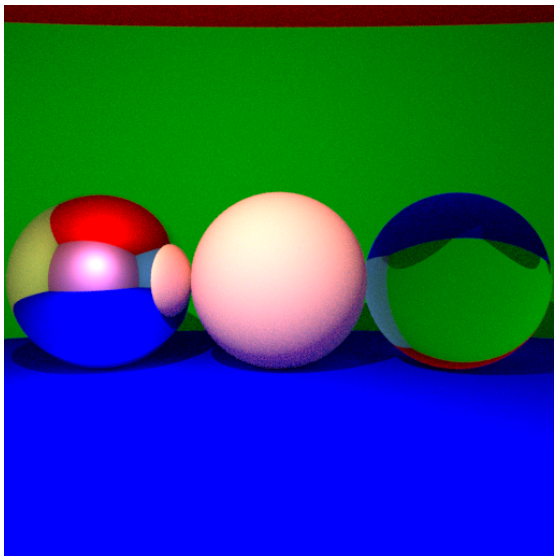
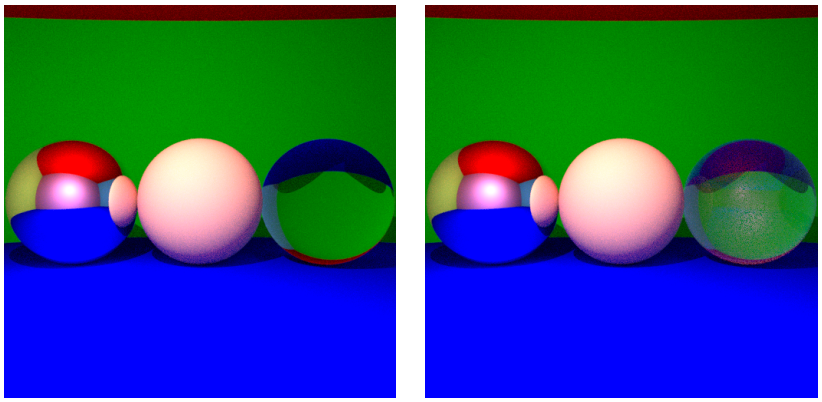


Figure 1: From left to right: reflective sphere, diffuse surface sphere and refractive sphere

4 Additional work on spheres

4.1 Fresnel effect

On my basic spheres, I added some more details. I added gamma correction, indirect lighting, antialiasing and Fresnel reflection for refractive spheres. I don't have images for each of them individually, in all of my Figures there is already indirect lighting, antialiasing and gamma correction.



(a) Without Fresnel (b) With Fresnel

Figure 2: Image with Fresnel effect for the transparent spheres

The Fresnel law permits partial reflection and partial refraction through Shlick's approximation. I implemented it and we can see the result in Figure 2.

4.2 Hollow sphere

A hollow sphere is implemented by a transparent sphere inside another one and inverting its normal. Meaning that the exterior with the air is actually inside, hence it is hollow. My result is given in Figure 3, the hollow sphere is on the right and we have again here the Fresnel effect.

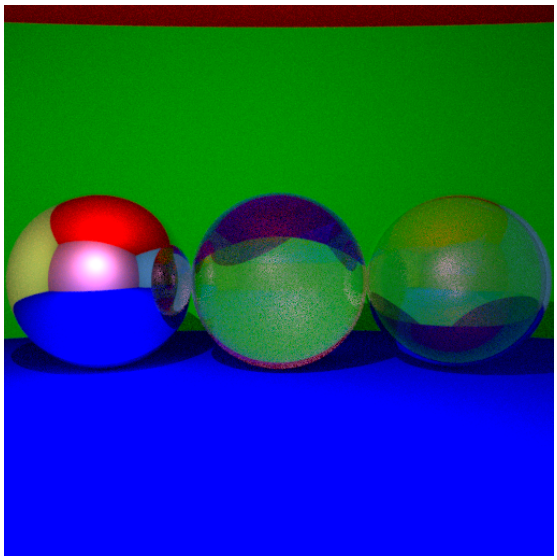


Figure 3: Image with a hollow sphere on the right

5 An attempt at soft shadows

Next, I attempted to create a sphere light source instead of a point one to do soft shadows. Unfortunately, my code does not work, we can see the result in Figure 4. I think I am pretty close to making it work so I decided to leave my code in my repository in soft_shadow.cpp.

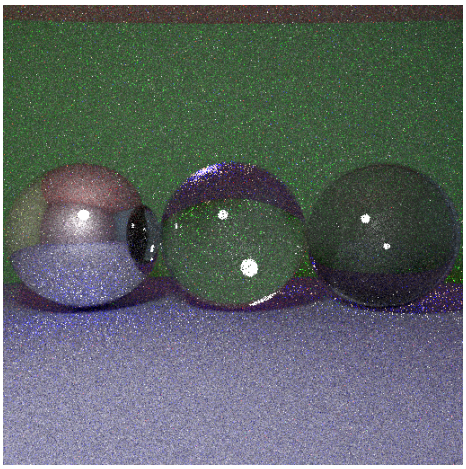


Figure 4: My non-working implementation of soft shadows

6 Triangle mesh objects

From now on, we introduced other forms than spheres. With .obj files, we can describe almost any object with triangle meshes. In my implementation, I render a cat as well as a stool. The principle once the object is created is the same as with a sphere: we see if the object intersects with the ray (here by checking if it intersects with a triangle of the mesh) and then deduce the color.

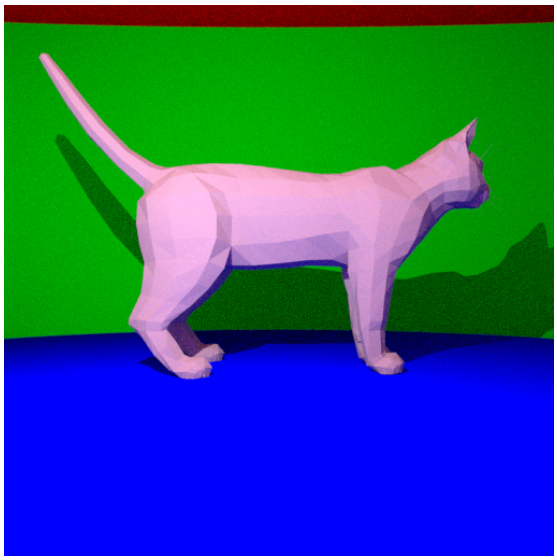


Figure 5: Image with a triangle mesh cat object

In Figure 5, we have 512 x 512 pixels, 64 rays, 5 bounces, around 750 lines of code and the code runs in 28 seconds.

7 Bounding Box and Bounding Volume Hierarchies

Just like this, the computation of the triangle mesh intersection is quite long. That is why I implemented first a bounding box: if the ray is out of the box then there is no need to check if it intersects with the cat.

In the last lab, I implemented the BVH. The Bounding Volume Hierarchies (BVH) is even better, we create a binary tree with our triangles to only check the intersection with a susbset of the triangles. With this technique, the intersection computation is much faster, it takes 28 seconds instead of multiple minutes.

8 Conclusion

In the end, I decided to combine all of these techniques to create a final scene composed of my cat, some spheres and a stool that I added. This scene shows most of the methods I mastered during this project.

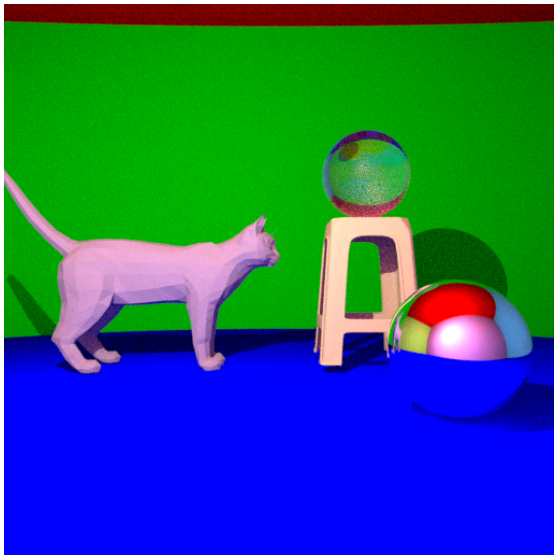


Figure 6: Final RayTracer image