Assignement 1: The Ray Tracer

I. Introduction

The ray tracer project was realized in four different labs. All the code I have written is in the raytracer.cpp file.

II. Diffuse surfaces, reflection, and refraction

During the first week, I implemented reflection, refraction, gamma correction and direct lighting. In the figure below, you can see the result from this first session.

The albedos chosen are:

- (0.3, 0.2, 0.6) for the balls
- (0.7,0.2,0.25) for the floor, (0.7,0.1,0.25) for the ceiling, (0.7,0.3,0.25) for the left, (0.7,0.1,0.25) for the right, (0.7,0.3,0.25) for the back, (0.7,0.3,0.25) for the front

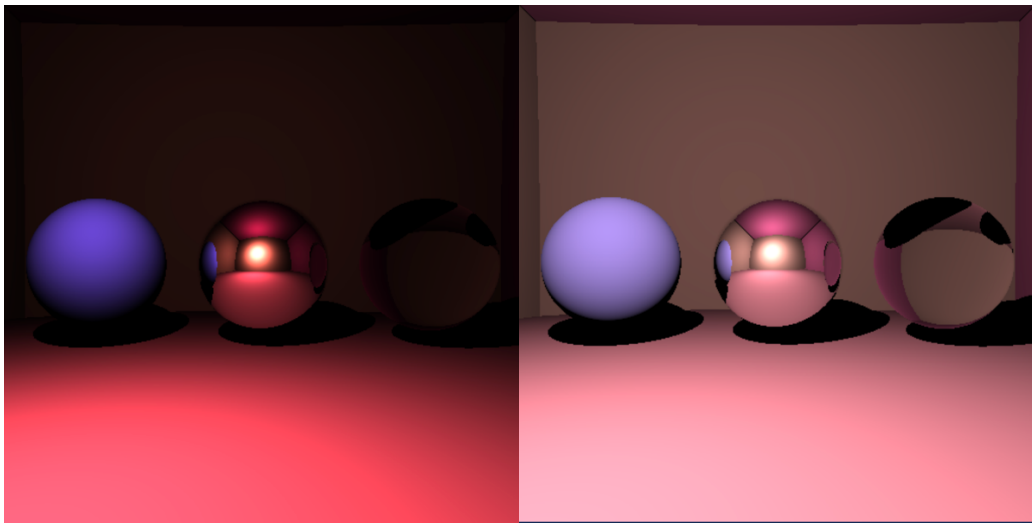

Figure 1 Figure 2

These two renderings took 255 ms. As you can see, we have a “normal” sphere, a mirror sphere, and a transparent sphere. On the left, there is no gamma correction and a smaller intensity of I = 2.10^7, and on the right, there is gamma correction, and the intensity is I= 2.10^10.

III. Indirect lighting and antialiasing

In the second lab, I added indirect lighting and antialiasing. I also added parallelization for the loop computing the color of the pixels as adding antialiasing slows down the process.

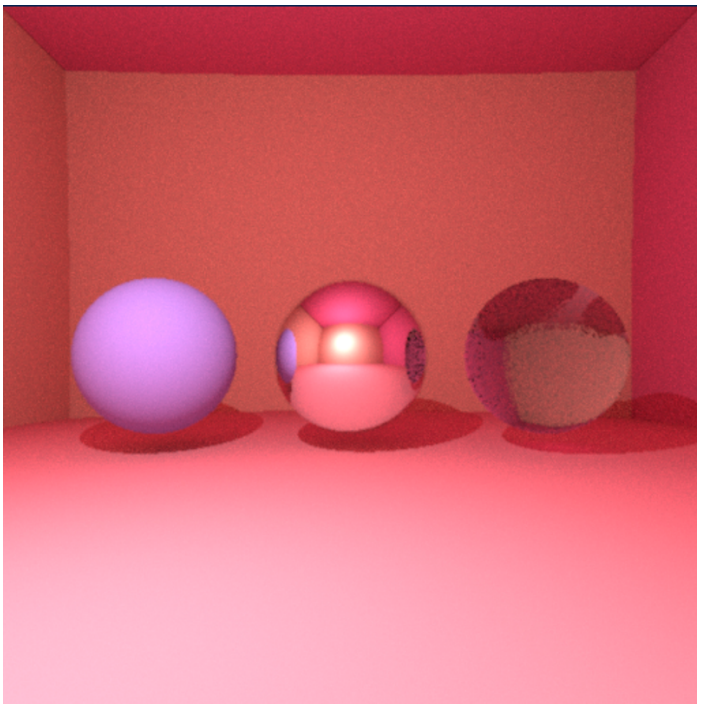

Figure 3

In Figure 3, you can see the result of implementing these two features. For 64 paths and the same albedo intensity as Figure 2, the rendering took 71298 ms with parallelization and 72195 ms without. Here is also a zoom for the antialiasing.

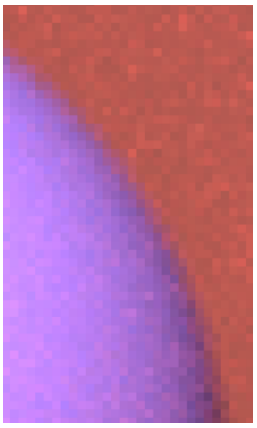

Figure 4

IV. Meshes

Now, it was time to code how to support meshes. Thus, I wrote the class TriangleMesh, and regrouped it in a more significant class called “Geometry” with sphere so that the Scene could handle either of them. I first implemented the mesh support with the Möller–Trumbore ray-triangle intersection algorithm. The time to render the figure was approximately 50 minutes with only the Möller–Trumbore ray-triangle intersection algorithm. Thus, I decided to focus on implementing BVH instead of implementing the intersection with one bounding box.

V. Meshes accelerated

The final version of the code, the one in GitHub, includes the BVH support instead of the simple bounding box one. The rendering time was drastically lower and took 790228 ms. The result is given below in Figure 7. The albedo, as well as the intensity, are the same. I also used 64 rays per pixel.

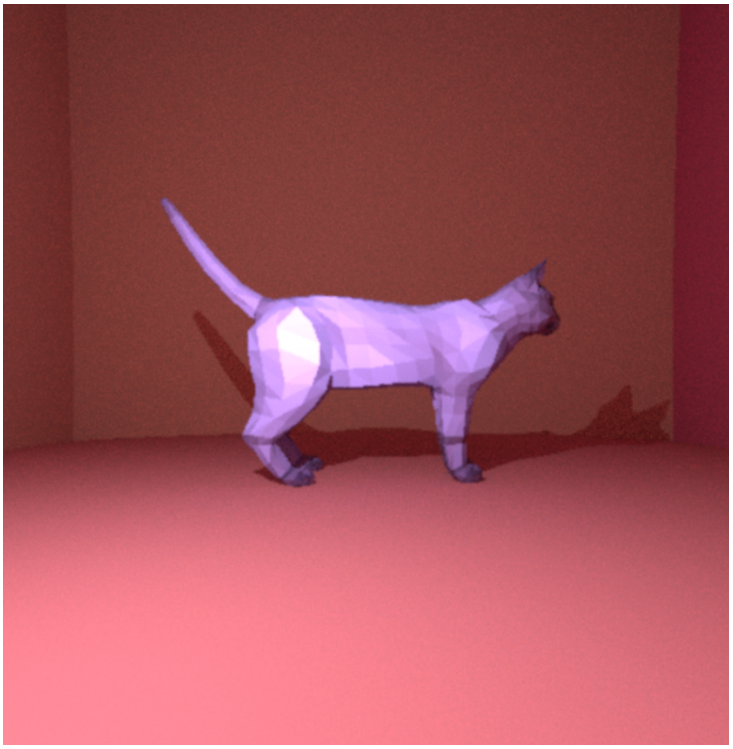

Figure 7