CSE306 Assignment 1 Report

Note: All the images in this report can be found in the folder Assignment1 under the name underlined.

Lab 1:

For this lab, I started first the implementation of rendering basic spheres. I defined my **classes** (Vector, Sphere, Ray, etc.) in a file called *auxiliary.cpp* and a scene class in *scene.cpp*. To handle **ray-sphere** and **ray-scene** intersections, I added methods in the corresponding classes. Then I implemented **refraction** and **reflection** in my class <Scene> and displayed the three different spheres as in the example. We obtain the following:

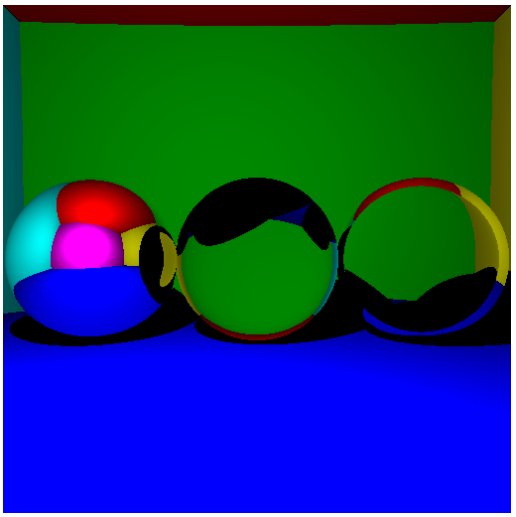


image1: Sphere with reflection, refraction and a hollow sphere with refraction
(intensity: 2e10, size 512x512, 32 rays, time taken 26565.7ms)

Then I modified my implementation so the **Fresnel reflection** is taken into account. Here is the result when the boolean fresnel is set to true in *main.cpp* and with the other parameters remaining the same:

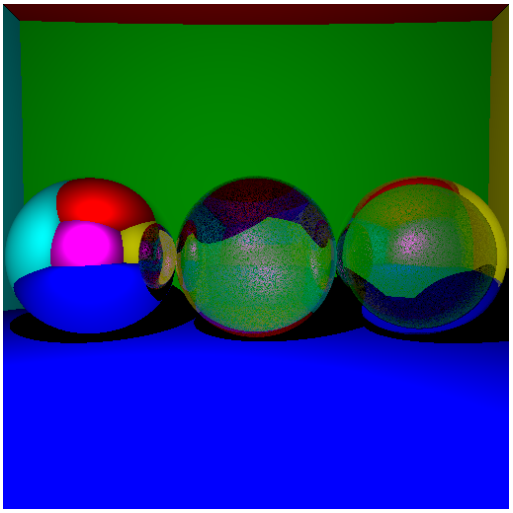


image2: Sphere with reflection, refraction and a hollow sphere with refraction with the Fresnel reflection taken into account (time taken 27267.2ms)

We observe that the result is better than in Image 1. However, the blending is not smooth as the bounds of the spheres are not smooth yet.

Lab 2:

I first understood the rendering equation and how to perform a **Monte Carlo numerical integration**. To do so, I did the exercise p.29 of the lecture notes in *boxmuller.cpp*. The result obtained for 1000 samples with 1 for loop and not 3 nested loops is 24.3276, which is good.

After that I added **indirect lighting** to my path tracer: I added the method random_cos to my <Vector> class using the formulas p.30 and improved the get_color function to perform an easier Monte Carlo integration. Keeping the same parameters, we obtained the following:

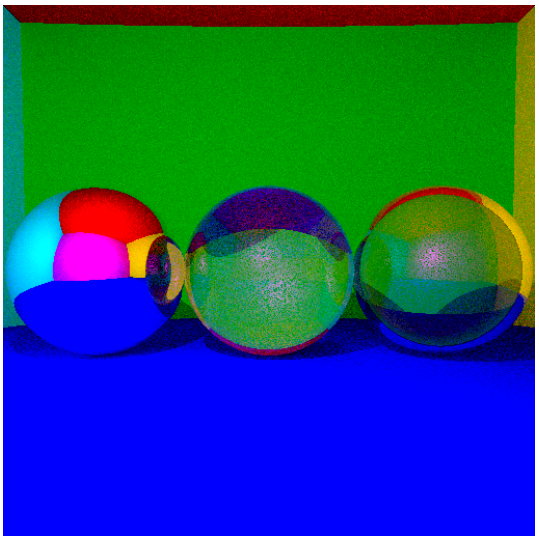


image3: Sphere with reflection, refraction and a hollow sphere with refraction with the Fresnel reflection and indirect lighting (time taken 129564ms)

Then I implemented antialiasing to make the shapes smoother and changed my center_ray function in my <Ray> class.

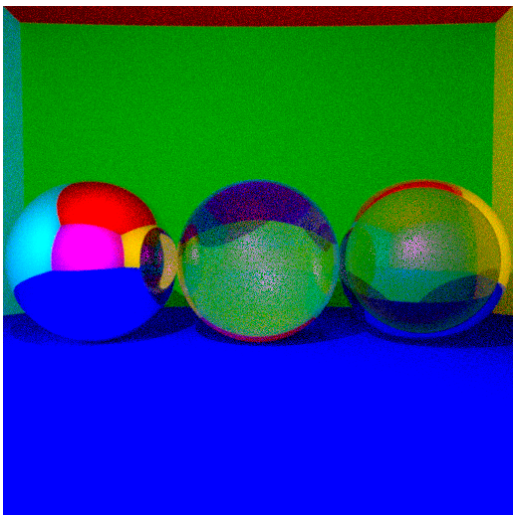
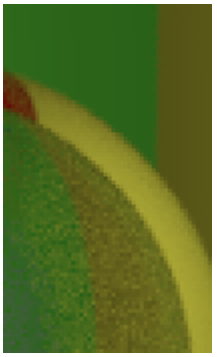


image4: Sphere with reflection, refraction and a hollow sphere with refraction with the Fresnel reflection, indirect lighting and antialiasing (time taken 129322ms)

Indeed, if you zoom in on spheres with and without antialiasing, we observe it improves the quality of the image.

zoom1: Without indirect lighting and antialiasing



zoom2: Without indirect lighting but with antialiasing

Lab 3:

I implemented mesh renderings. I created a new <TriangleMesh> class in *auxiliary.cpp* with a method intersect to handle ray intersection with the Moller-Trumbore algorithm. I added the <Geometry> class which <Sphere> and <TriangleMesh> inherit from. Then, I handled the bounding box feature creating a <Bbox> class and adding the function compute_bbox in the class <Scene>. Using the downloaded low poly cat mesh, we obtained the following:

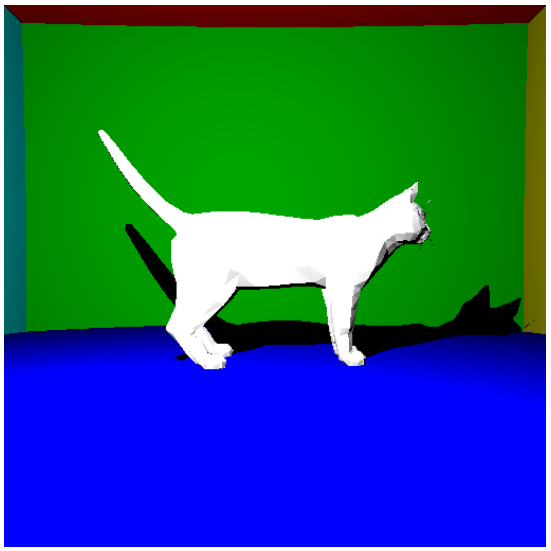


image5: Cat scaled by a factor 0.6, translated by the vector (0,−10,0), albedo is Vector (1,1,1), intensity 3e10, 32 rays, 5 light bounces, field of view 60 degrees, bounding box, without antialiasing, Fresnel refraction and indirect lighting (time taken 3.00365e+06ms ie 50 min)

Lab 4:

Then I added the BVH support to my ray-mesh intersection

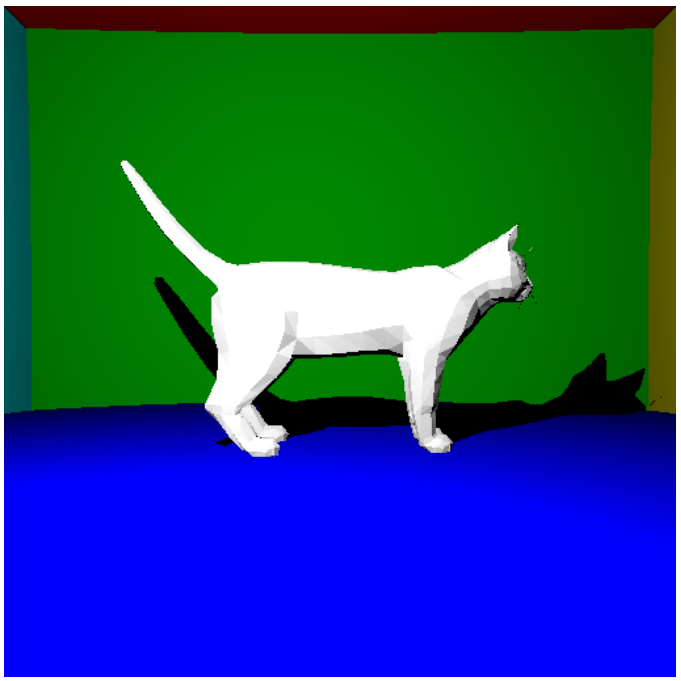


image6: same setting than in image5 but with BVH instead of bounding box (time taken 386908ms ie around 6 minutes)

Finally, now that BHV is implemented, we can set the parameters as expected in the assignment and obtain:

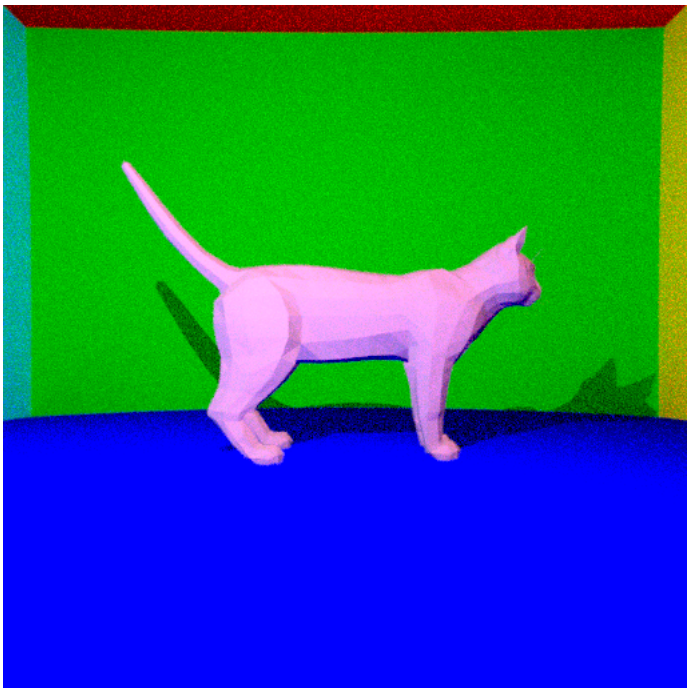


image7: Cat scaled by a factor 0.6, translated by the vector (0,−10,0), albedo is Vector (0.3,0.2,0.25), intensity 3e10, 32 rays, 5 light bounces, field of view 60 degrees, BHV, with antialiasing, Fresnel refraction and indirect lighting (time taken 1.48107e+06ms ie around 25 min)