EL EUCH Farah

CSE306 : First Assignment Report

Files and description :

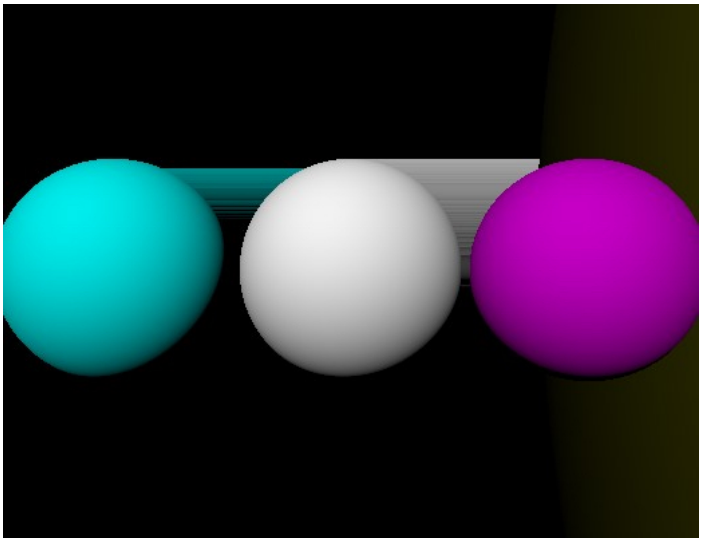
- vector.cpp : contains the class Vector with all useful vector-related functions
- mesh_loader.cpp : contains class TriangleMesh, BoundingBox and BVH
- sphere.cpp : contains the class Sphere with the function Ray/Sphere intersection
- objects.cpp : contains the class Camera, Ray, Intersection and Geometry (Sphere and TriangleMesh are Geometry subclasses)
- random.cpp : contains random_engine and random_device which create random structs of uniform distribution, and the implementation of BoxMuller.
- scene.cpp : contains the class Light, the class Scene which creates a scene with its elements and contains lighting and colour related classes (Lambertian and getColor) but also the implementation of the function Intersection creating intersections of the elements of the scene with a ray.
- main.cpp contains a main function that uses all the previous files to create the expected image.

- for testing : g++ -O3 -std=c++11 main.cpp -o a.out && ./a.out

Steps :

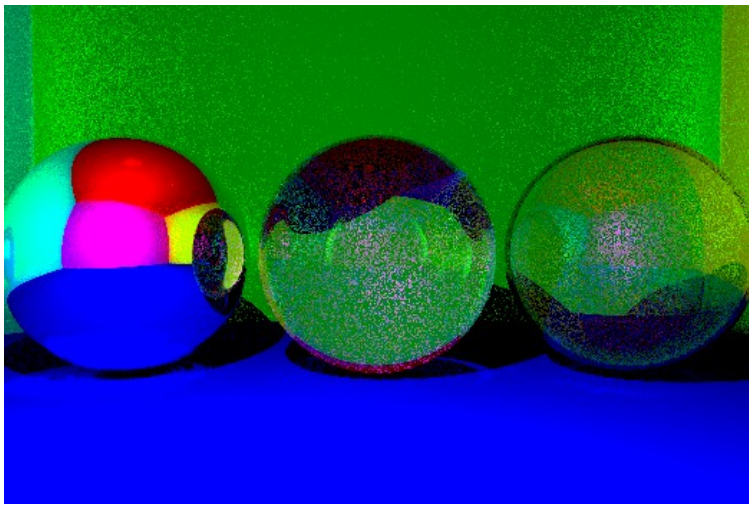
I did not always write down the time it took to render images but except for the last one, it was never less than 10s.

- This is the image I obtained after implementing the Vector, Ray, Sphere, Intersection and Scene classes, the Ray-Sphere and Ray-Scene intersections and Shading, shadows computation, and gamma computation.



it took about 5s to render this image

- The next step was adding reflections and refraction and the Fresnel Law, as well as the indirect lighting, the rendering equation, the Monte Carlo integration to what I already implemented. I also directly added parallelization. This is the image I obtained after all these implementations, by following the instruction of the course.



it took about 6s to render this image (but the resolution was quite low)

- The final step was about Meshes. It induced implementing the Ray-Plane, Ray-Triangle and Ray-Mesh intersections. After implementing this, I tested my codes by generating an image with very low resolution by just adding the cat to the previous scene.

After that I implemented the BoundingBox and BVH classes, using the codes in the lecture notes and adjusting them to what I already had. Then to test, I removed the useless spheres from my main.cpp, scaled the cat model by 0.6, translating it by a vector (0,-10,0). To do so, I created a void function transform(Vector offset, double scale) in the mesh_loader.cpp. This is the image I got with all these parameters. The rendering time was 1888s.