Tim Valencony
14/05/2023

CSE306 Assignment 1 Report

Lab1:

In this lab, we render the basic spheres. For this, various classes were defined, such as: Vector, Sphere, Ray, and Scene. We then proceeded to handle intersections between ray-scene and ray-sphere, this is done in the respective *intersect* functions of the respective *Scene* and *Sphere* classes.
The result is the following:

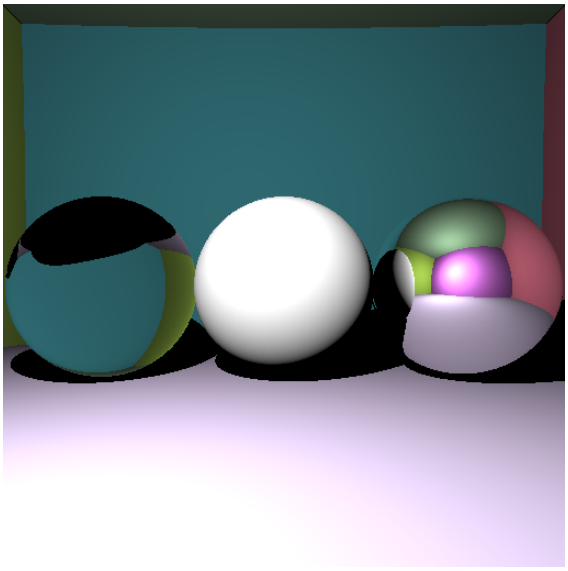


Image 1: Sphere with refraction, full sphere, and sphere with reflection
(I = 2E10, H & W = 512, 64 rays, computation time = 47,974 ms)

Next, we implemented the Fresnel Reflection. Here are the results after the implementation:

Tim Valencony
14/05/2023

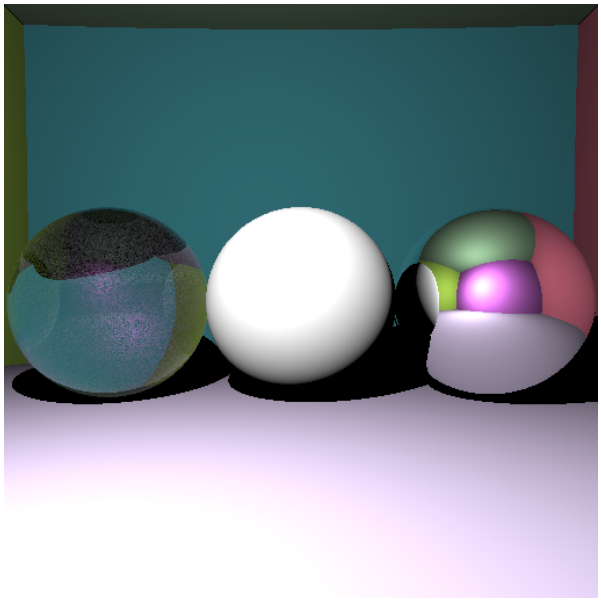


Image 2: Sphere with refraction, full sphere, and sphere with reflection (Fresnel Light)
(I = 2E10, H & W = 512, 64 rays, computation time = 47,188 ms)

Lab2:

We then introduced indirect lighting for point light sources without russian roulette. For this we defined new functions like *random_cos* in the *Vector* class. We also updated the *get_color* function from the *Scene* class in order to include a better Monte Carlo Integration. We obtain the following results:

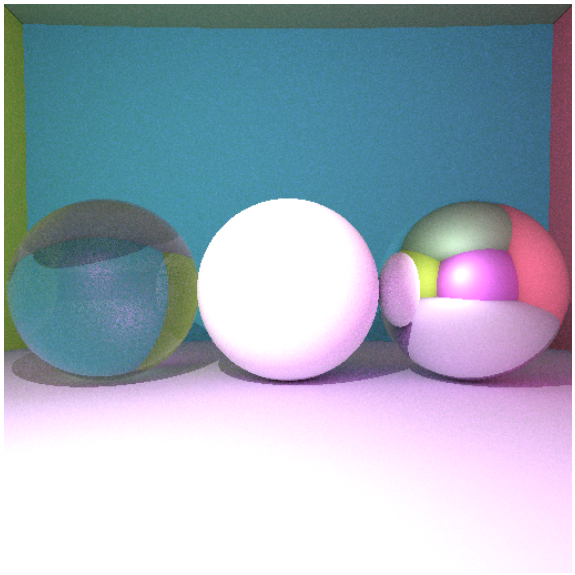


Image 3: Sphere with refraction, full sphere, and sphere with reflection (Fresnel Light & Indirect Lighting)
(I = 2E10, H & W = 512, 64 rays, computation time = 236,496 ms)

Tim Valencony
14/05/2023

Next, we implemented antialiasing in order to smooth the surfaces of the shapes. This is done by adding random components to the first two components of the ray direction.
We obtain the following result:

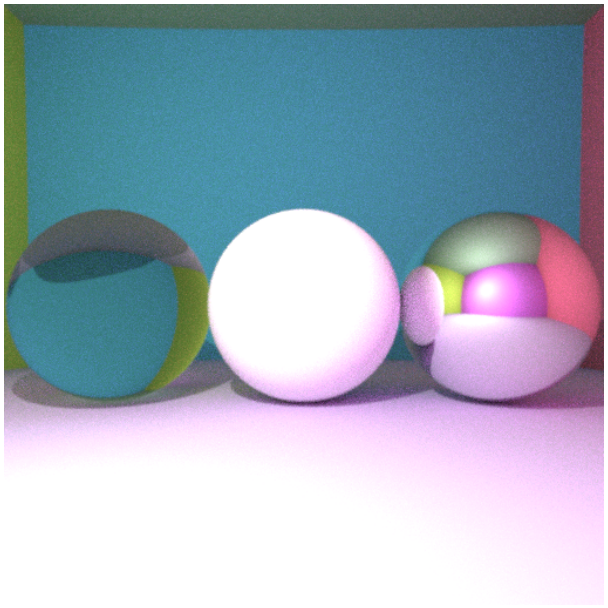


Image 4: Sphere with refraction, full sphere, and sphere with reflection (Fresnel Light & Indirect Lighting & Antialiasing)
(I = 2E10, H & W = 512, 64 rays, computation time = 240,140 ms)

Lab3:
Using the code provided in the lecture notes, we implemented a *TriangleMesh* and *TriangleIndices* class. We also handled ray intersection using a method from the lecture notes (Möller-Trumbore intersection algorithm).
Furthermore, we also created a class *Object* which handles both *Spheres* and *TriangleMesh*.
Then, we proceeded to create a *BoundingBox* class. This necessitated additional functions in the *TriangleMesh* class, such as the *compute_bounding_box* function. We obtain the following result for the cat:

Tim Valencony
14/05/2023

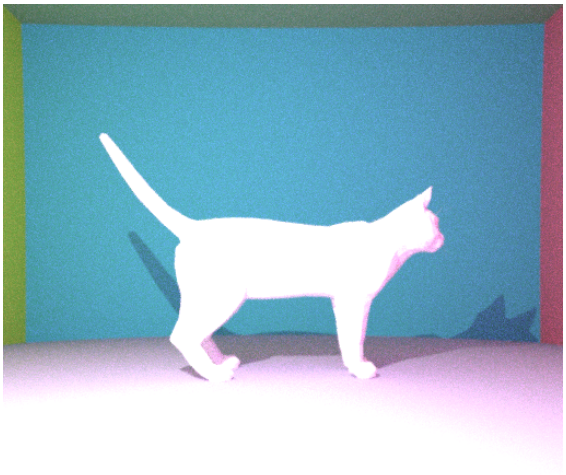
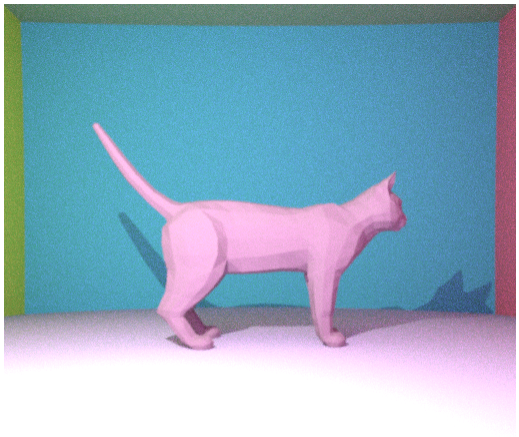


Image 5: Cat translated by (0, -10, 0), albedo (1, 1, 1), field of view of 60 degrees and scaled by 0.6 (Antialiasing & Fresnel Light & Indirect Lighting & Bounding Box)
(I = 3E10, H & W = 512, 64 rays, computation time = 1,452,178 ms)

Lab4:
Lab 4 was about adding BVH to the ray-mesh intersection in order to reduce the computation time. I obtained the same image but this time with computation time = 473448 ms.
Now that everything is set-up, we can obtain the final result, which is as follows:



Final Result: Cat translated by (0, -10, 0), albedo (0.3, 0.2, 0.25), field of view of 60 degrees and scaled by 0.6 (Antialiasing & Fresnel Light & Indirect Lighting & BVH)
(I = 3E10, H & W = 512, 64 rays, computation time = 485,177 ms)

Tim Valencony
14/05/2023

Sources: The code was largely inspired by what was coded in class by the professor (Mr. Nicolas Bonneel). Furthermore, when having trouble in figuring out how to move forward, this github repository helped me in unlocking the situation.