Computer Graphics, CSE306, Assignment 1

Natali Gogishvili

May 21, 2024

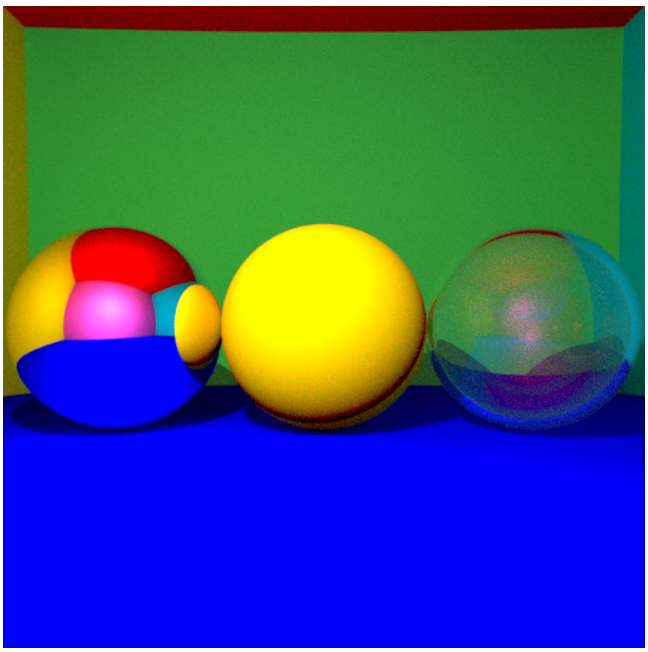


Figure 1: Mirror, diffuse surface and transparent surface using Fresnel. 100 rays per pixel, 5 bounces. Time: 13 min 58 sec.

1 Introduction

As part of this project I implemented all the required parts of the TDs, including displaying diffuse and mirror surfaces, direct lighting and shadows for point light sources, anti-aliasing, ray-mesh intersection using simple boxing and BVH. In addition, I implemented optional parts of the project such as: transparent surfaces (full and hollow), Fresnel law, normal maps, and texture. I also implemented rotation of mesh along y axis.

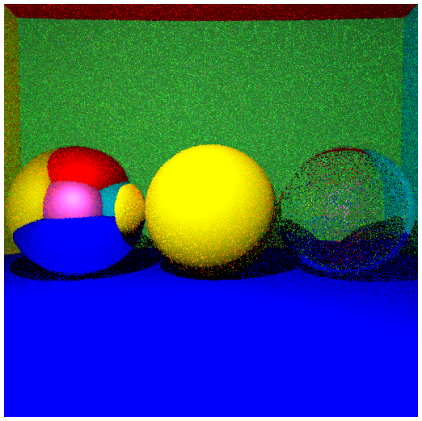
2 Notes

All figures are of size 512 x 512 pixels. I always used 5 bounces (maximal depth of intersect function). I obtained results on my local laptop, which has 8 CPUs (to take into consideration when I display timing, since lab machines have 20 CPUs). Also, throughout this report I will call number of rays per pixel K.

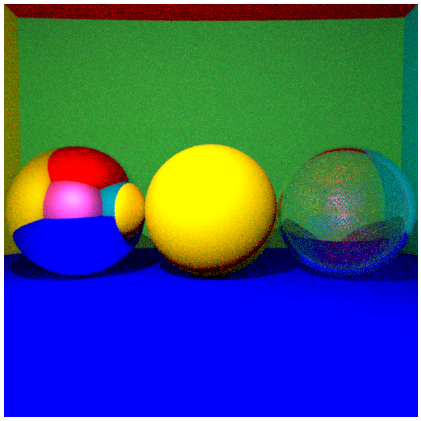
3 Intersections with Spheres

All objects inherit from class Geometry, which defines intersect function. For intersecting ray with a sphere I used formulas on slide 26, TD1. After the intersection is obtained we call getColor function to get a color which must be displayed. In case of diffuse surfaces color is calculated by formula on slide 29, TD1. In case of a mirror, new ray is shot from the intersection point. In the case of transparent surfaces with Fresnel, random number gets generated and based on it's value, ray gets either reflected or refracted. For this to work and not be too noisy, we do several iterations per pixel and average the results. We also use indirect shadows. For this, when intersection happens we shoot a ray in random direction and based on this calculate indirect light (even if the object is in the shadow). To smoothen the result (to not see it pixel by pixel), we use anti-aliasing, for which we add sampling inside of a pixel. For this we use a simple Gaussian kernel.

Below are the results with K = 1, K = 5, K = 10 iterations (with aliasing and indirect shadows).



(a) Mirror, diffuse, and hollow sphere, K = 1, 5 bounces. Time: 8 sec.

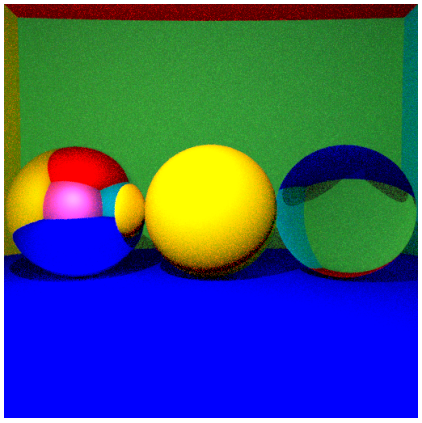


(b) Mirror, diffuse, and hollow sphere, K = 10, 5 bounces. Time: 1min16sec.

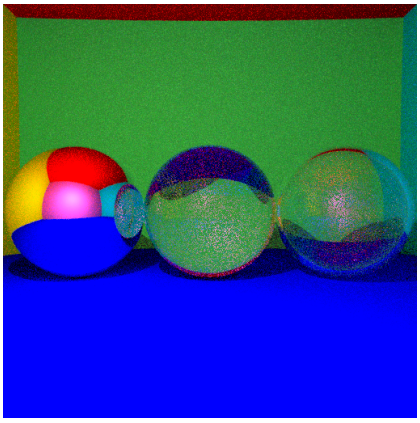
Figure 2: Comparison of different K values. With Fresnel.

We use a trick to display hollow sphere easily. For hollow sphere we use two spheres, one inside and one outside, with inverse refraction index. For the

outside sphere refindex = 1.5 and for inside sphere refindex = 1/1.5. This works well with the way we implement refraction.



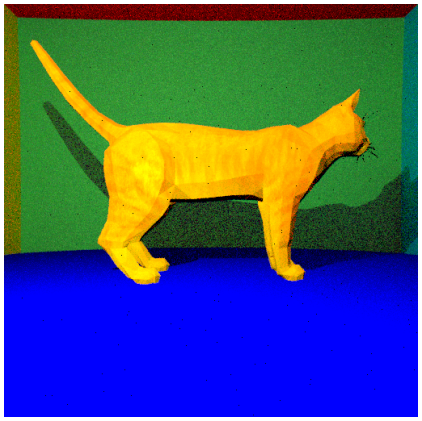
(a) Mirror, diffuse, and full sphere, without Fresnel, K = 10, 5 bounces. Time: 1min15sec.



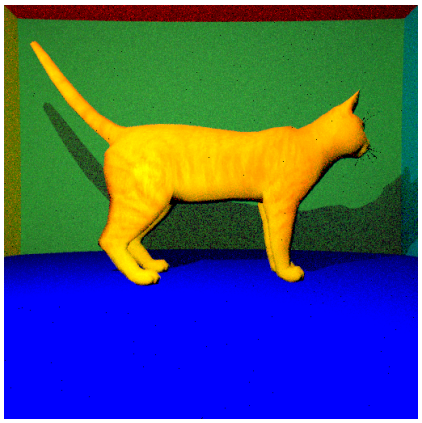
(b) Mirror, full transparent, and hollow sphere, with Fresnel, K = 10, 5 bounces. Time: 1min24sec.

4 Triangle Mesh

For this part, I first implemented Möller Trumbore Ray-Triangle intersection algorithm. It is inside intersectTriangle function. Intersection with mesh then happens if intersection happens with one of its triangles. I sped this process up first with simple box and then with BVH. BVH sped it up from several minutes to around one minute for K = 10. Scaling and translation are done on vertices right after loading the mesh.

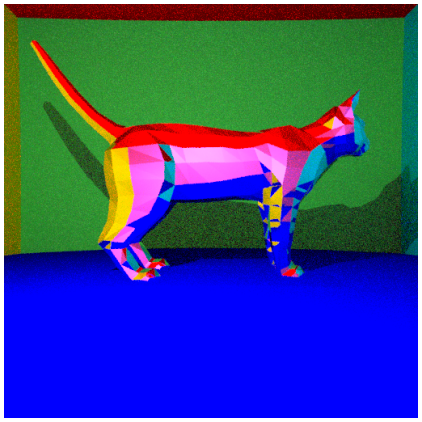


(a) Cat using BVH and texture. K = 10, 5 bounces. Time: 1min40sec.

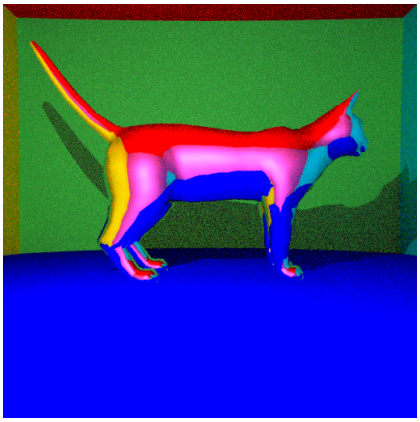


(b) Cat using BVH, provided normals, and texture. K = 10, 5 bounces. Time: 1min36sec.

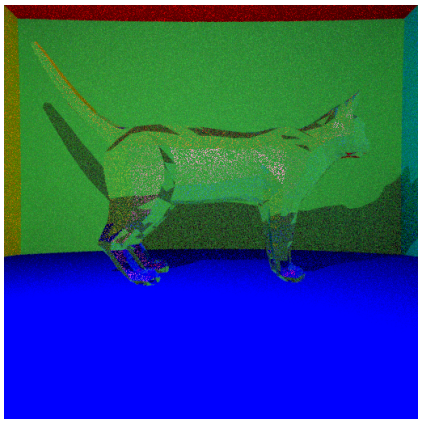
Figure 4: Comparison of a cat with and without provided normals. Without BVH (only bounding box) it took 19min32sec.



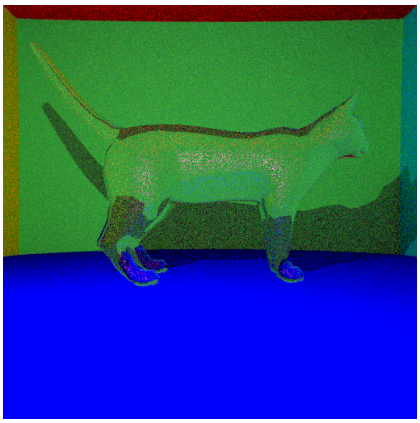
(a) Mirror cat, no normals. K = 10, 5 bounces. Time: 1min32sec.



(b) Mirror cat, with normals. K = 10, 5 bounces. Time: 1min34sec.

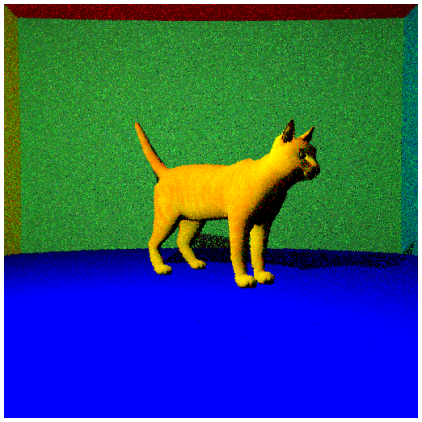


(a) Transparent cat, no normals. K = 10, 5 bounces. Time: 1min42sec.

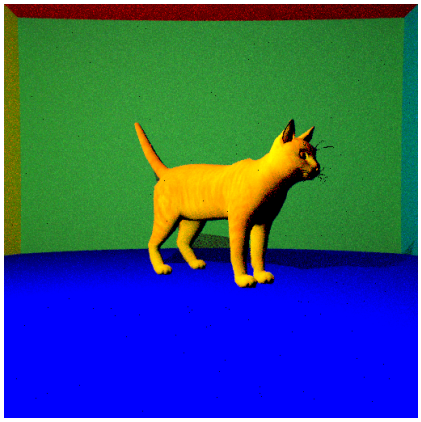


(b) Transparent cat, with normals. K = 10, 5 bounces. Time: 1min34sec.

Additionally, I also implemented rotation of a cat along y direction. I am first rotating it and then computing BVH, so optimization still works.



(a) Rotated cat. K = 1, 5 bounces. Time: 8.6sec.



(b) Rotated cat. K = 10, 5 bounces. Time: 1m25sec.

Figure 7: Comparison of rotated cat with different K values for 5 bounces