CSE306 Raytracer Report

Brook Nigatu

May 2024

1 Introduction

This report summarizes the features implemented in my ray tracer. For each feature a short description of the implementation is given along with an image illustrating the feature. **Render times are indicated below each image**.

All the code is in the file "Assignment1/ray_tracer.cpp".

Rendered objects are represented by the abstract class "Geometry", which has two children: "Sphere" and "Triangle Mesh".

A "Scene" object contains the objects to be rendered and the light source.

OpenMP is used to parallelize the computations done for each pixel.

2 Diffuse and Mirror Surfaces, Direct Lighting and Shadows for Point Light Sources

Figure 1 shows a diffuse sphere. Figure 2 is a rendering of a mirror sphere enclosed in a room with spherical walls. It also illustrates shadows. Only direct lighting is used for both images.

A ray is launched through the center of each pixel. If there is an intersection and the nearest point of intersection is visible from the light source the color is computed by the formula given in the lecture notes. Otherwise, the pixel is colored black.

If the intersected object is a mirror, the color is determined recursively (up to the maximum recursion depth) using a reflected ray.

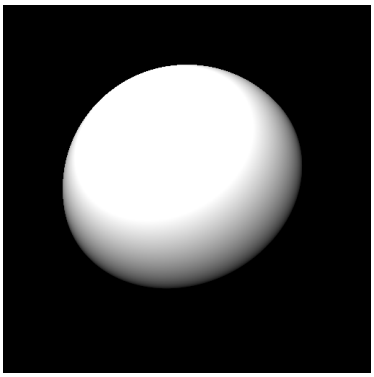

Figure 1: Diffuse Object: Render time < 1s

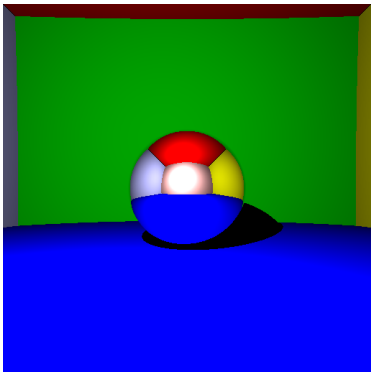

Figure 2: Mirror Object: Render time < 1s

3 Indirect Lighting for Point Light Sources (without Russian Roulette)

Indirect lighting is used to render the image in Figure 3. The Monte-Carlo method with finite light bounces described in the lecture notes is used to implement it. 64 rays were launched per pixel for the image.

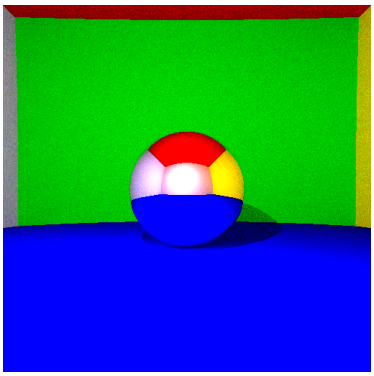

Figure 3: Indirect Lighting: Render time 2s

4 Antialiasing

Figure 4 shows a rendering of the same scene in Figure 3 with antialiasing. Each ray is shifted from the center of the pixel along the x-y plane using a random 2D standard gaussian vector generated using the Box-Muller method.

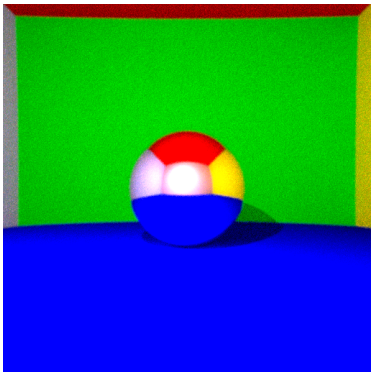

Figure 4: Antialiasing: Render time 2s

5 Mesh Support (with BVH)

Figure 5 shows a rendering of the provided cat mesh.

A BVH is used to optimize the intersection procedure. An iterative depth first search is used to traverse the BVH. While traversing, the closest triangle found so far is used to exclude farther boxes. Moreover, if both children of a node are to be visited, the child with the closer box is added to the stack last so that it is visited first.

The Moller-Trumbore algorithm and Phong interpolation are used to find intersections with triangles and interpolate norms at the intersection points respectively.

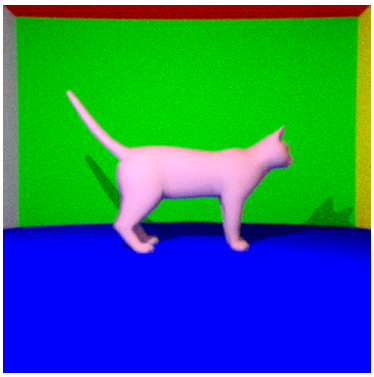

Figure 5: Mesh Support with BVH: Render time 48s