Project 2: PathTrace

Name: Minsuk Kim(m.kim)
Instructor: Dr. Gary Herron
Class: CS500
Semester: Spring 2022
Date: 2/16/2022

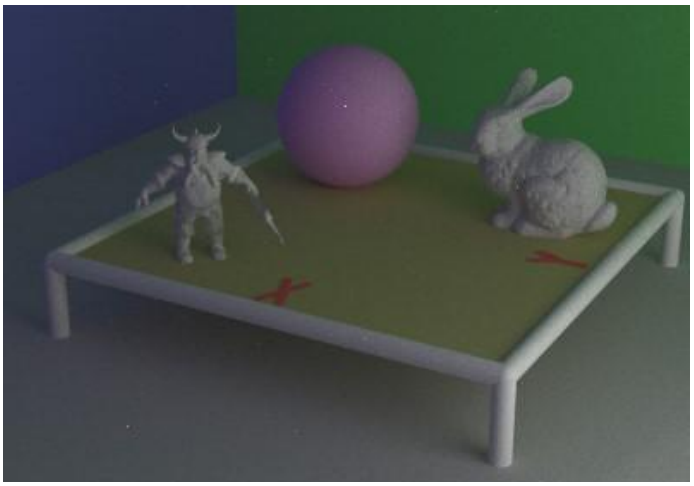


Table of Contents

Introduction

Overview

The main purpose of the project is to implement the path-tracing feature. The program will generate an extra ray toward a random direction after the ray hits a certain object until the ray finds a light object. The generated ray stored cumulative color information and when the ray hits a light object, it calculates the color of the pixel position where the first ray starts. The project uses a very simple BRDF for lighting calculation. I started the project with the previous project that was about ray casting. I also used an external library Assimp that read the object files and BVH for bounding volume hierarchy.

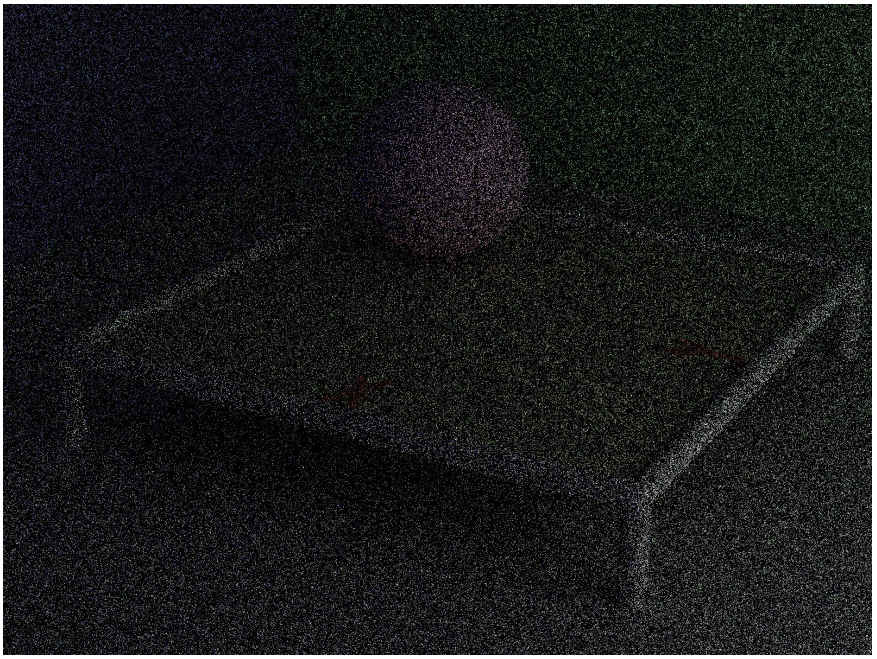
Implementation

First, the program loaded the scene file and created the shapes for each object file. The program casts ray for each pixel. When the ray collides with a certain object that is not a light object, the ray will generate another ray that towards a random direction. The random direction is distributed vector around a normal vector in the hit position. Then, when the ray hits the light object, the program reads cumulative data that stored previous hit objects' colors and calculates the color of the pixel. The program also uses the explicit light connection that shoots another ray from the hit position to a random light object. When the ray that shoots toward the light object hits the same light object, the program adds the light color to the current color. The program repeatedly generates rays to the same position for better accurate light calculation. The program also uses anti-aliasing by changing random offset to a pixel position in each pass.

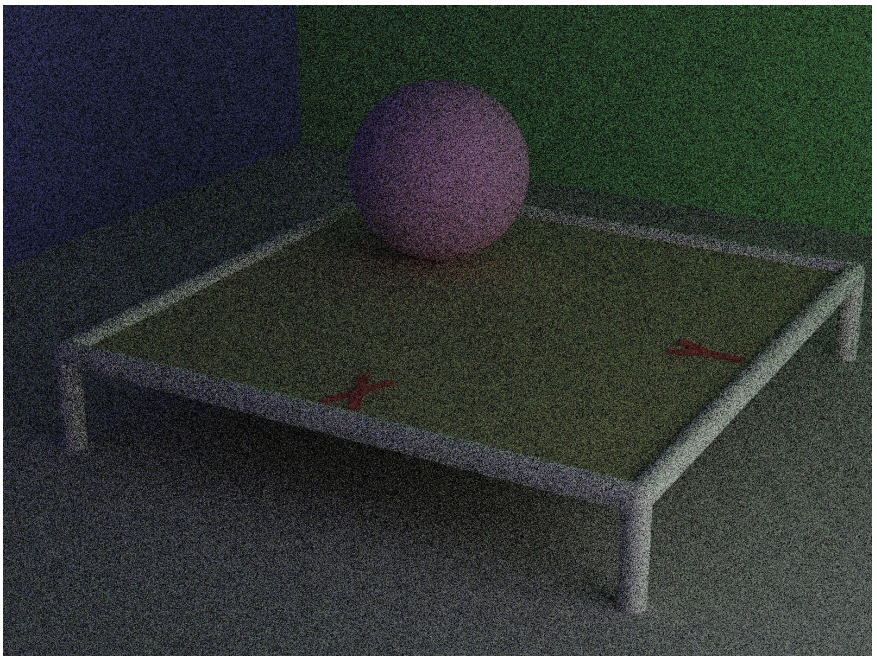
Result Images

images with implicit lighting connection

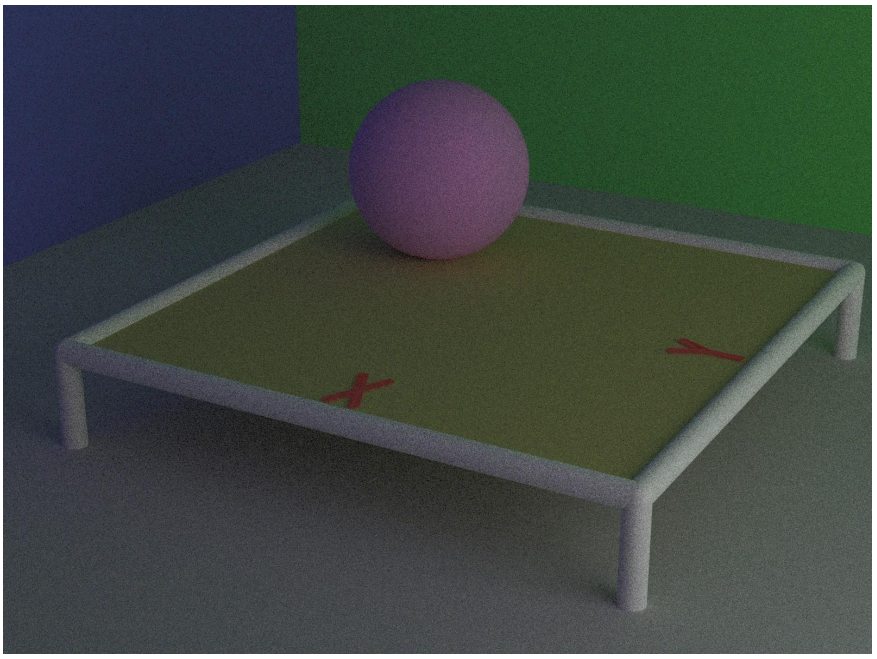
1 pass with 1600x1200 image



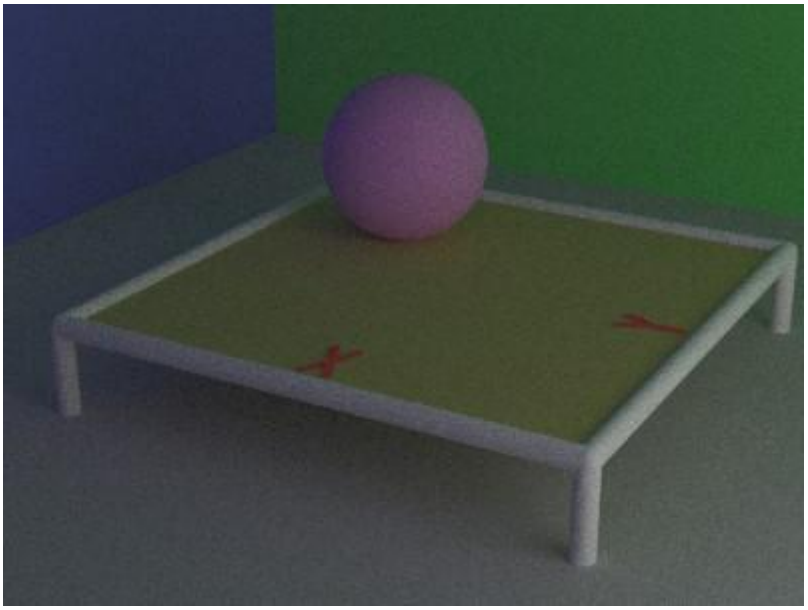
8 passes with 1600x1200



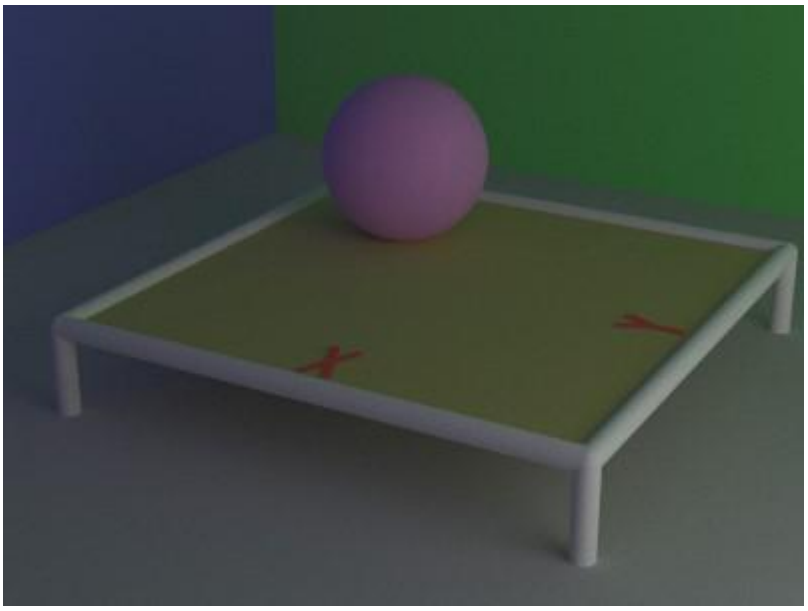
64 passes with 1600x1200



512 passes with 400x300

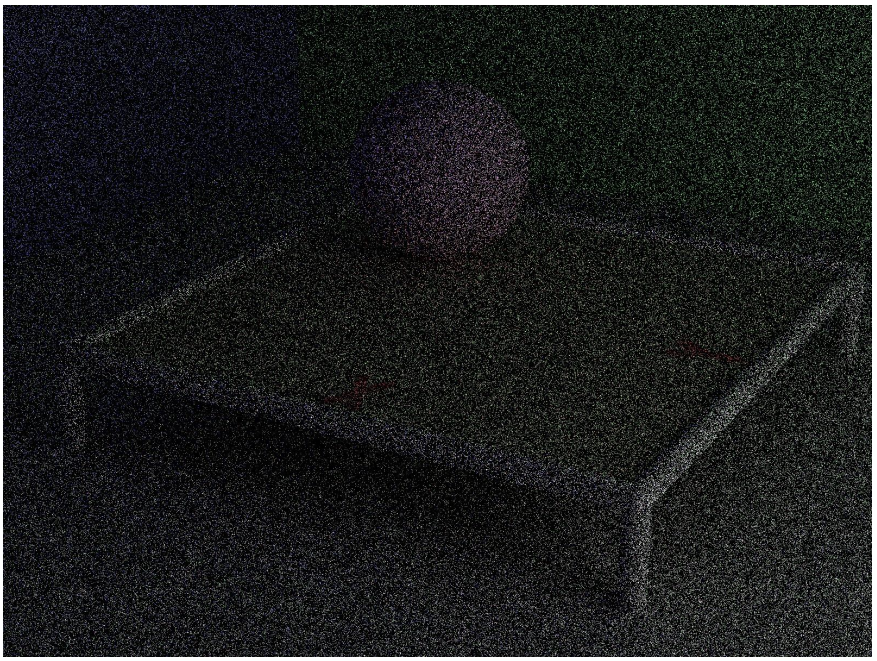


4096 passes with 400x300

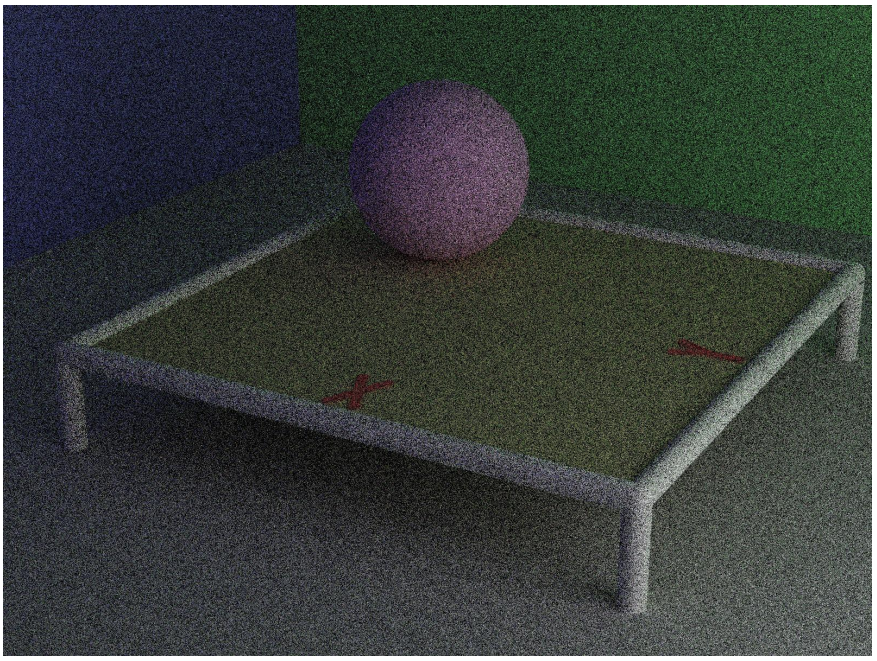


images with implicit + explicit lighting connection

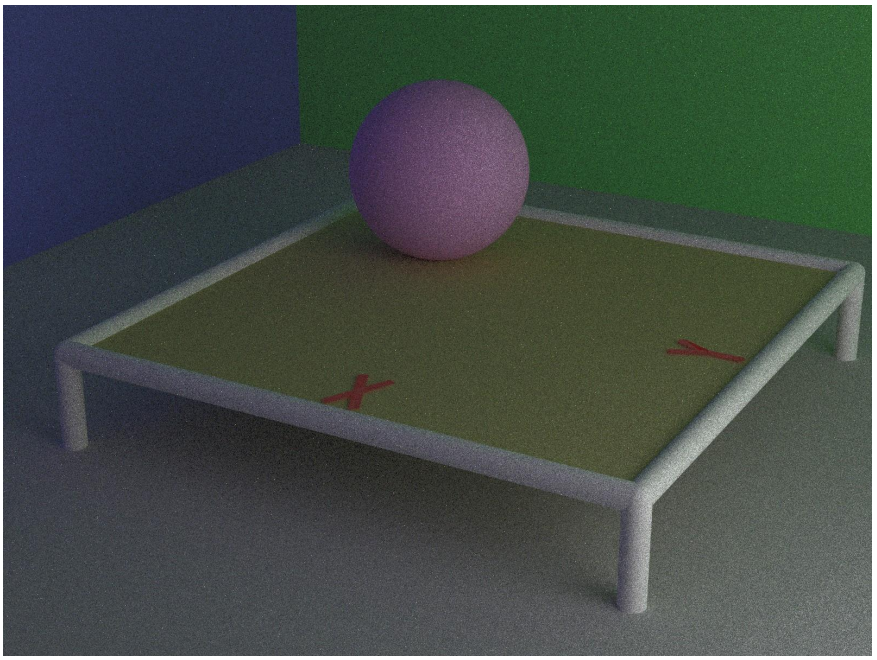
1 pass with 1600x1200 image



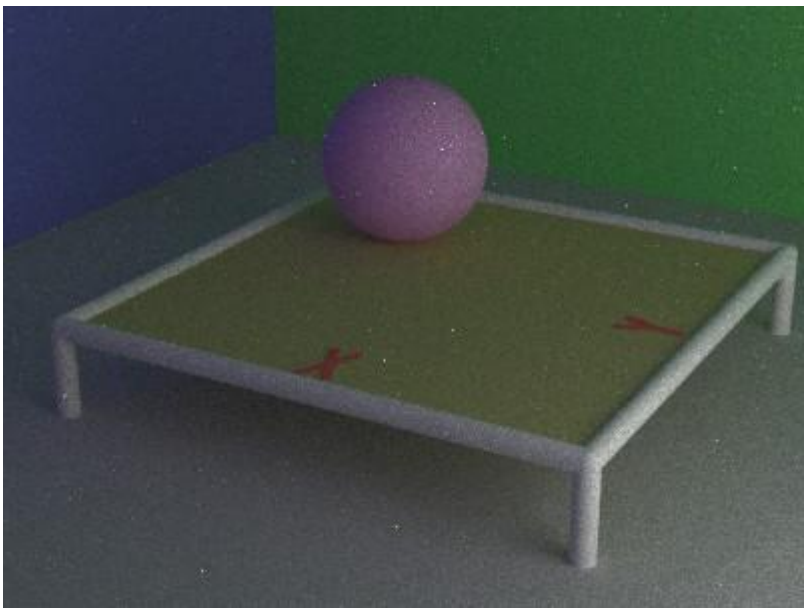
8 passes with 1600x1200 image



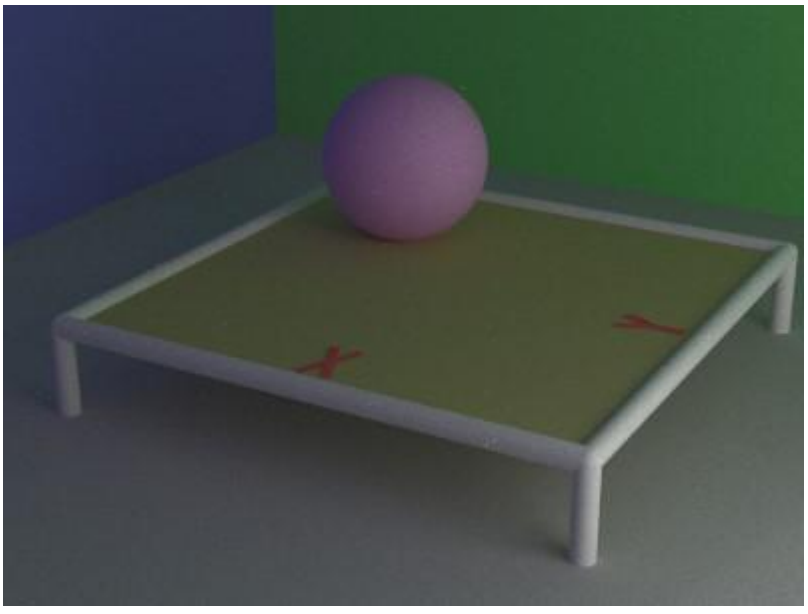
64 passes with 1600x1200 image



512 passes with 400x300 image



4096 passes with 400x300 image



Notes

Random Position

To cast a new ray, the program needs to find the direction of the ray. There are two random points for this project: points on a hemisphere(used in implicit light connection), points on a sphere(used in implicit light connection). For the random points on a hemisphere, the points are calculated by

$$\xi_1 = random\ number\ [0, 1]$$

$$\xi_2 = random\ number\ [0, 2\pi]$$

$$s = \sqrt{(1 - \xi_1)}$$

$$K = (s \cdot cos\xi_2, s \cdot sin\,\xi_2, \sqrt{\xi_1})$$

$$if\ |A.z - 1| < \varepsilon,\ d = K$$

$$if\ |A.z + 1| < \varepsilon,\ d = (K.x, -K.y, -K.z)$$

$$B = Z\ x\ A$$

$$C = A\ x\ B$$

$$d = K.x \cdot B + K.y \cdot C + K.z \cdot A$$

where d is a directional vector for the new ray and A is the normal vector of the original ray's hit point. ε is epsilon (in this project it is set to 0.001).

The random points on a sphere are calculated by

$$\xi_1, \xi_2 = random\ number\ [0, 1]$$

$$z = 2\xi_2 - 1$$

$$r = \sqrt{(1 - z^2)}$$

$$a = 2\pi \cdot \xi_2$$

$$P = C + R \cdot N$$

where C is the center of the sphere and R is the radius of the sphere.

It is implemented on line 346 in raytrace.cpp

Implicit light connection

The implicit light connection is generating a new ray when the ray hits an object. The generated ray starts with the hit point and directs toward the vector that is randomly generated vector around the normal of the hit point. Whenever a ray hits an object, it calculates the scattering value from the object information. The cumulative light is equal with cumulative light * Kd / RussianRoulette for this object(because we use the simplest BRDF for lighting calculation). The RussianRoulette determines how many rays should be generated. When the random number is larger than RussianRoulette, the ray will not generate new rays and go to the next pixel. When the ray hits an object that is light, the ray evaluates the final color of the pixel and returns it. The final color is diffuse color * cumulative light. When the explicit light is enabled, the final pixel color is diffuse color * cumulative light * 0.5 + explicit color.

It is implemented on line 270 in raytrace.cpp

Explicit light connection

The implicit light connection is generating a new ray when the ray hits an object. The generated ray starts with the hit point and directs toward the random points on random lights in the world. When the ray hits the same light objects, it adds explicit color with 0.5f * W * (f / p) * light color, where W cumulative light in implicit light connection part and f is the dot product of normal of hit point and direction toward light multiply by light color divided by PI. The f is the multiplication of the probability of light and converted to an angular measure of two rays(old and new). The probability of light is 1 / (area of light multiplied by the number of lights in the world). The converted to an angular measure of two rays is

$$A_p = old\ ray\ hit\ point,\ A_N = normal\ of\ old\ ray\ hit\ point$$

$$B_p = new\ ray\ hit\ point,\ B_N = normal\ of\ new\ ray\ hit\ point$$

$$D = A_p - B_p$$

$$angular\ measure\ = (A_N \cdot D) \cdot (B_N \cdot D)/(D \cdot D)^2$$

It is implemented on line 254 in raytrace.cpp