CSE 167 Final Project Write-Up

Ray Tracer Implementation

Hanbin Tan

March 2026

1 Overview

This report documents the implementation of a Monte Carlo path-tracing ray tracer. Starting from a blank rendering pipeline, I incrementally built up perspective ray generation, ray-primitive intersections, anti-aliasing, direct illumination with soft shadows, global illumination via multiple bounces, mirror reflections, and Russian Roulette path termination.

2 Task 1: Ray Creation

Implemented perspective ray generation in `RayTracer::ray_thru_pixel()`. For each pixel (i, j) with sub-pixel offset (x, y), the ray direction is:

$$\mathbf{d} = \text{normalize}(\alpha \cdot \mathbf{u} + \beta \cdot \mathbf{v} - \mathbf{w})$$

where $\mathbf{u}, \mathbf{v}, \mathbf{w}$ are the camera's right, up, and negative-forward basis vectors, and:

$$\alpha = \tan\left(\tfrac{\text{fovy}}{2}\right) \cdot \text{aspect} \cdot \left(\frac{2(i+x)}{W} - 1\right), \quad \beta = \tan\left(\tfrac{\text{fovy}}{2}\right) \cdot \left(1 - \frac{2(j+y)}{H}\right)$$

Verified by setting `debug_color = 0.5 * dir + 0.5` and comparing with Figure 16a.

Figure 1: Task 1: Debug visualization of ray directions (1 spp, 1 bounce, scene 1).

3 Task 2: Ray Intersections

3.1 Ray-Sphere (Task 2.1)

Solved the quadratic equation $\|\mathbf{r}_o + t\mathbf{r}_d - \mathbf{c}\|^2 = R^2$. Expanding yields $at^2 + bt + c = 0$ where $a = \mathbf{r}_d \cdot \mathbf{r}_d$, $b = 2(\mathbf{r}_o - \mathbf{c}) \cdot \mathbf{r}_d$, $c = \|\mathbf{r}_o - \mathbf{c}\|^2 - R^2$. The nearest positive root gives the hit distance, and the normal is $\mathbf{n} = \text{normalize}(\mathbf{p} - \mathbf{c})$.

3.2 Ray-Triangle (Task 2.2)

Implemented the Möller-Trumbore algorithm. Given triangle vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$, we compute edges $\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0$ and $\mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$, then solve for barycentric coordinates (u, v) and parameter t using Cramer's rule. The normal is interpolated from vertex normals: $\mathbf{n} = (1 - u - v)\mathbf{n}_0 + u\mathbf{n}_1 + v\mathbf{n}_2$ for smooth shading.

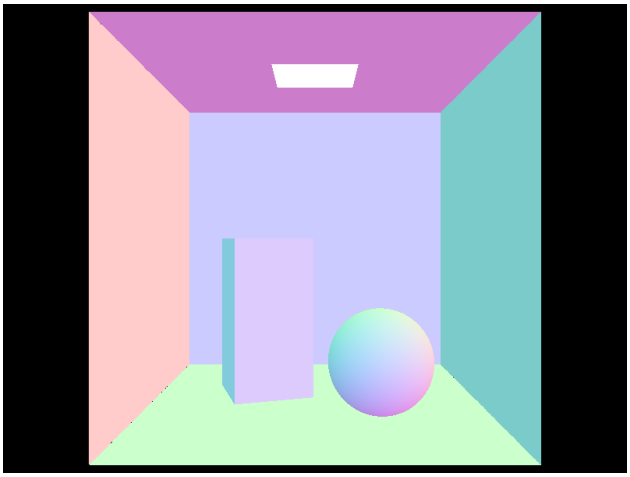


Figure 2: Task 2: Normal shading with sphere and triangle intersections (1 spp, 1 bounce, scene 1).

4 Task 3: Anti-Aliasing

Replaced the fixed pixel center $(0.5, 0.5)$ with uniform random samples $x, y \sim U(0, 1)$. With N samples per pixel, each ray passes through a different sub-pixel position, and the results are averaged. This stochastic sampling smooths jagged edges at object boundaries.

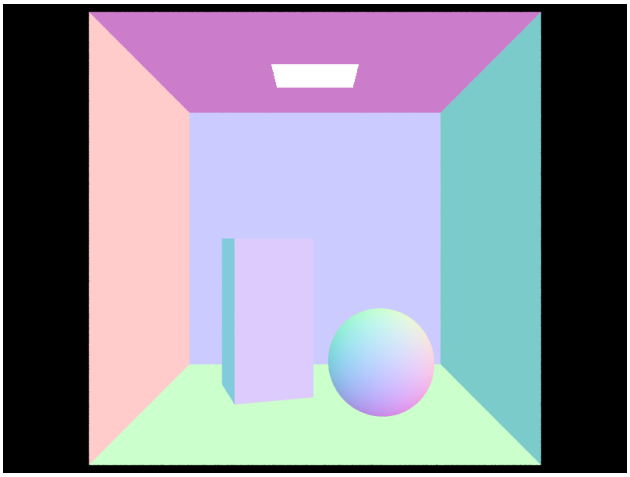


Figure 3: Task 3: Anti-aliased rendering (10 spp, 1 bounce, scene 1).

5 Task 4: Single Ray Bounce (Direct Lighting)

Implemented Lambertian direct lighting with Next Event Estimation (NEE). For each surface hit, we sample $K = 8$ points on each area light and evaluate:

$$D_{\text{diffuse}}(x) \approx \frac{\boldsymbol{\rho}}{\pi} \sum_{\ell} \frac{|A_\ell|}{K} \sum_{k=1}^{K} \left[\frac{\mathbf{E}_\ell}{|A_\ell|} \cdot V(x, y_k) \cdot \frac{c_x \cdot c_y}{\|y_k - x\|^2} \right]$$

where $c_x = \max(\mathbf{n}_x \cdot \boldsymbol{\omega}_k, 0)$ is the receiver cosine, $c_y = \max(\mathbf{n}_y \cdot (-\boldsymbol{\omega}_k), 0)$ is the emitter cosine, and $V(x, y)$ is the binary visibility function evaluated via shadow rays.

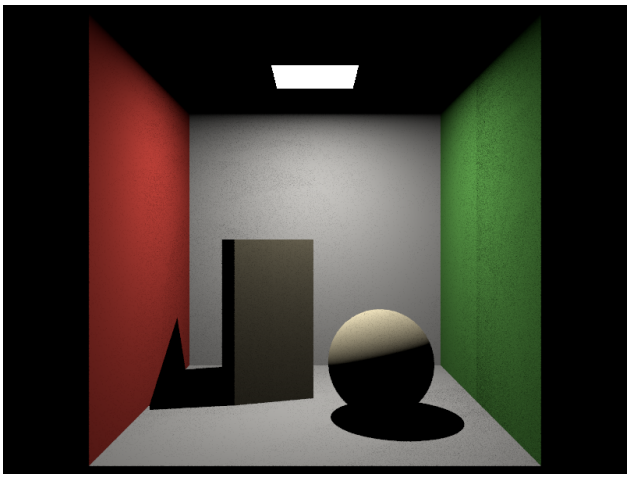


Figure 4: Task 4: Direct lighting with hard shadows (10 spp, 1 bounce, scene 1).

6 Task 5: Soft Shadows

Implemented uniform random sampling on the square area light:

$$\mathbf{p} = \mathbf{c} + u \cdot s \cdot \hat{\mathbf{t}} + v \cdot s \cdot \hat{\mathbf{b}}, \quad u, v \sim U\left(-\tfrac{1}{2}, \tfrac{1}{2}\right)$$

Different shadow ray samples see different portions of the area light, producing penumbra regions where the light is partially occluded.

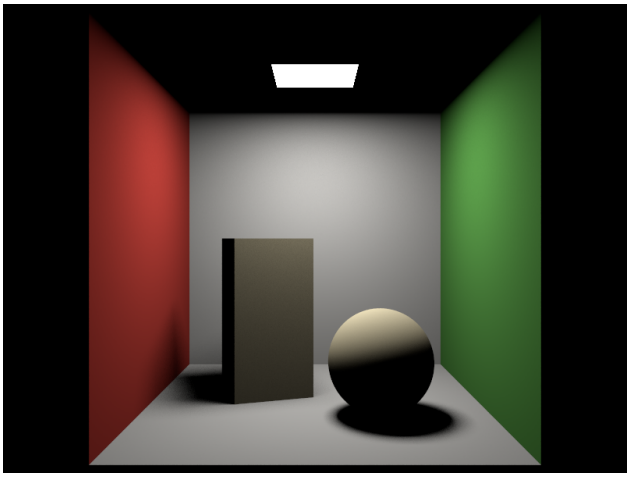


Figure 5: Task 5: Soft shadows from area light sampling (100 spp, 1 bounce, scene 1).

7 Task 6: Multiple Ray Bounces

7.1 Task 6.1: Diffuse Material

Implemented cosine-weighted hemisphere sampling for diffuse bounces. Given two uniform random numbers $s, t \in [0, 1]$:

$$u = 2\pi s, \quad v = \sqrt{1-t}, \quad \mathbf{d}_{\text{local}} = \begin{bmatrix} v\cos u \\ \sqrt{t} \\ v\sin u \end{bmatrix}$$

This local direction is rotated to align with the surface normal. The throughput is updated as $W_{\text{wip}} *=$ albedo, and `allow_emissive_hit` is set to `false` to avoid double-counting direct light.

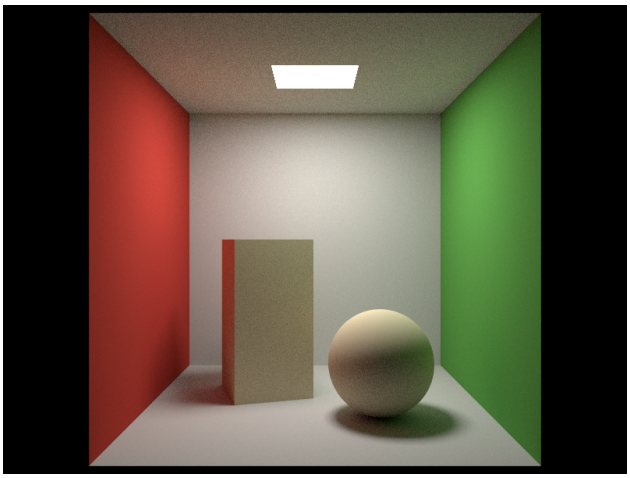


Figure 6: Task 6.1: Global illumination with color bleeding (100 spp, 3 bounces, scene 1).

7.2 Task 6.2: Mirror Material

Implemented perfect specular reflection:

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}$$

The throughput is scaled by the mirror's reflectance. Unlike diffuse bounces, mirror continuations keep `allow_emissive_hit = true` since the only way a mirror "sees" a light is through reflection.

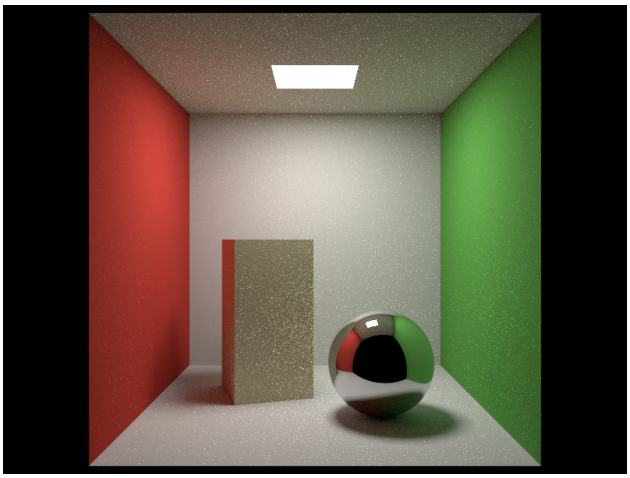


Figure 7: Task 6.2: Mirror sphere reflecting the Cornell box (100 spp, 3 bounces, scene 2).

8 Task 7: Russian Roulette Path Termination

8.1 Motivation

A fixed maximum bounce count introduces bias by truncating paths that could still carry significant energy. Russian Roulette provides an unbiased alternative: instead of hard-cutting at bounce N, we probabilistically terminate rays and compensate survivors.

8.2 Algorithm

At each bounce after the first, we draw a uniform random number. If it falls below the termination probability λ, the ray is killed. Otherwise, the ray survives and its throughput W_{wip} is divided by $(1 - \lambda)$:

```
if (ray.n_bounces > 1) {
    if (rand01() < lambda) {
        ray.terminate = true;
        return ray;
    }
    ray.W_wip /= (1.0f - lambda);
}
```

8.3 Why This Is Unbiased

A path of length n survives with probability $(1 - \lambda)^{n-1}$. The throughput compensation $\frac{1}{(1-\lambda)^{n-1}}$ exactly cancels this, so:

$$\mathbb{E}[\text{compensated contribution}] = (1 - \lambda)^{n-1} \cdot \frac{1}{(1 - \lambda)^{n-1}} \cdot L_n = L_n$$

8.4 Choice of λ

I chose $\lambda = 0.2$ (20% termination probability per bounce). This means:

- 80% of rays survive each bounce
- Expected path length before termination: $1/\lambda = 5$ additional bounces
- Paths carrying little energy are naturally pruned

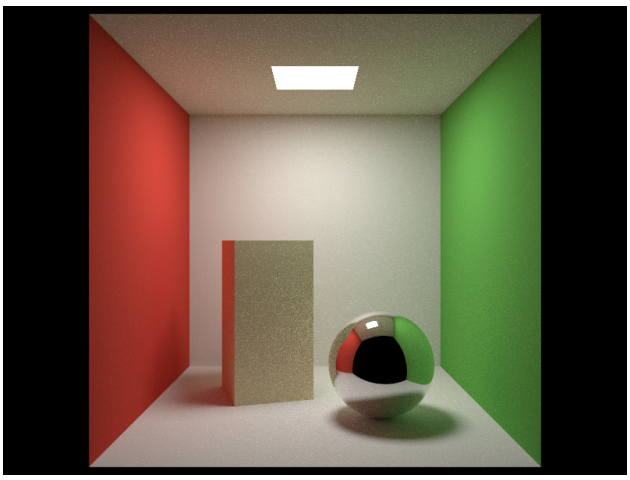
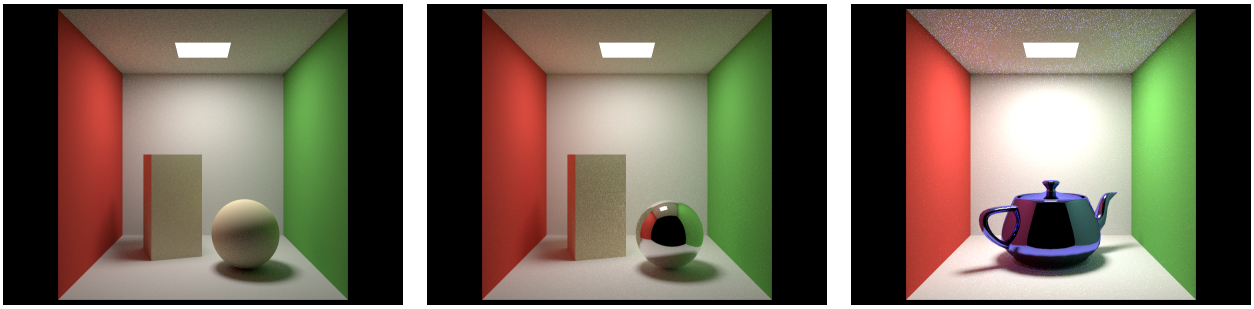


Figure 8: Task 7: Russian Roulette with mirror sphere (500 spp, max 100 bounces, scene 2). The high bounce limit is handled efficiently by probabilistic termination.

9 Results Gallery



(a) Cornell box, 100 spp, 3 bounces

(b) Mirror sphere, 500 spp, 100 bounces

(c) Mirror teapot, 50 spp, 3 bounces

Figure 9: Favorite renders showcasing global illumination, mirror reflections, and complex geometry.

9.1 Challenges

- Getting the shadow ray origin offset correct to avoid self-intersection (shadow acne).
- Understanding the Monte Carlo normalization factor $|A_\ell|/K$ for area light sampling.
- Obtaining the emitter cosine term c_y by intersecting the shadow ray with the light model to get the surface normal at the sample point.

9.2 Learning Outcomes

This project provided hands-on experience with the core components of a physically-based renderer: camera models, geometric intersection, Monte Carlo integration, material BRDFs, and variance reduction techniques (Russian Roulette). The incremental structure made it possible to verify each component independently before combining them.