Assignment 1: Basic Ray Tracing

NAME: CHEN LINXI
STUDENT NUMBER: 2022533132
EMAIL: CHENLX2022@SHANGHAITECH.EDU.CN

1 INTRODUCTION

In this assignment, we implement a basic ray tracer for the Cornell box scene following the specification of CS171 Assignment 3: Basic Ray Tracing. Our work can be divided into mandatory components and several optional extensions:

Mandatory components.

- Implement ray–triangle intersection and ray–AABB intersection for basic geometry queries.
- Build and traverse a BVH acceleration structure on top of these intersection routines.
- Complete the `IntersectionTestIntegrator` and the `PerfectRefraction` BSDF to handle both solid and perfectly refractive materials.
- Implement a direct lighting integrator with a diffuse BRDF and shadow testing using a hard point light source.
- Add anti-aliasing via multi-ray sampling per pixel within a sub-pixel aperture.

Optional extensions.

- Texture mapping: support both procedural checkerboard textures and image-based textures through the existing texture framework.
- Soft shadows: approximate rectangular area lights by multi-sampling around a finite-sized light region to produce penumbrae on the Cornell box floor and objects.

2 IMPLEMENTATION DETAILS

2.1 Ray–Triangle Intersection

In theory, the intersection between a ray and a triangle can be formulated as:

- The triangle $\triangle(v_0, v_1, v_2)$ lies on a supporting plane with normal $\mathbf{n} = (v_1 - v_0) \times (v_2 - v_0)$ and plane equation $\mathbf{n} \cdot \mathbf{x} = d$, where $d = \mathbf{n} \cdot v_0$.
- A ray is parameterized as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$; solving $\mathbf{n} \cdot (\mathbf{o} + t\mathbf{d}) = d$ gives the plane hit distance t.
- Once the hit point $\mathbf{q} = \mathbf{r}(t)$ is known, we can use an inside–outside edge test to decide whether $\mathbf{q}$ lies inside the triangle.

2.2 Ray–AABB Intersection

- An axis-aligned bounding box is viewed as the intersection of three 1D intervals along x, y, and z; along each axis, the ray enters and exits a slab at two parametric distances t^i_{near} and t^i_{far}.
- The global entry and exit distances are then $t_{\text{enter}} = \max_i t^i_{\text{near}}$ and $t_{\text{exit}} = \min_i t^i_{\text{far}}$; an intersection exists iff $t_{\text{enter}} \le t_{\text{exit}}$ and $t_{\text{exit}} \ge 0$.

2.3 BVH Construction and Traversal

Bounding Volume Hierarchies (BVHs) are tree structures that accelerate ray tracing by grouping nearby primitives into nested bounding volumes:

- Each internal node stores a AABB bounding box that tightly encloses all primitives in its subtree, while each leaf node references one or a small set of primitives.
- During ray casting, we first intersect the ray with node bounding boxes; if a node's box is missed, the entire subtree can be safely skipped.

Conceptually, BVH construction is often formulated as a top–down recursive partitioning problem:

- Starting from all primitives, we compute their union bounding box and select a split rule (e.g., longest axis or surface area heuristic) to divide them into two groups.
- This process is applied recursively to each group to create the left and right children, until some stop criterion is met (such as a minimum number of primitives per leaf or a maximum tree depth).
- The result is a binary tree where upper levels capture coarse spatial structure, and lower levels refine the partitioning around individual primitives.

2.4 IntersectionTestIntegrator and Perfect Refraction

From a theoretical point of view, our `IntersectionTestIntegrator` can be seen as a very simple path tracer that only accounts for direct illumination and a restricted set of BSDFs:

- For each camera ray, we repeatedly trace it through the scene using the BVH until either it misses all geometry or first encounters a diffuse surface; at that diffuse hit point we approximate the outgoing radiance by a single-bounce direct lighting term.
- Surfaces are classified by their BSDF: ideal diffuse surfaces terminate the ray while perfectly refractive surfaces are treated as specular events that only change the ray direction.

- For perfectly refractive materials, we assume ideal specular transmission governed by Snell's law: given the outgoing direction and the interface normal, we compute the transmitted direction using the relative index of refraction; when total internal reflection occurs, we fall back to perfect mirror reflection instead.

2.5 Direct Lighting with Diffuse BRDF and Shadow Testing

At a diffuse hit point, we approximate the outgoing radiance by a very standard direct-illumination model:

- To determine visibility, we cast a "shadow ray" from the surface point towards the light; if this ray hits any geometry before reaching the light position, the point is considered to be in shadow and its direct contribution from this light is set to zero.
- In our implementation we optionally include a simple inverse-square distance attenuation factor for the point light (proportional to $1/r^2$), making nearby regions brighter and distant regions dimmer while still keeping the model intentionally non-physically-accurate as suggested in the assignment.

2.6 Anti-Aliasing via Multi-Ray Sampling

Finally, to reduce aliasing along object silhouettes and sharp edges, we apply a simple supersampling scheme in image space:

- Instead of casting a single ray through the center of each pixel, we generate multiple rays per pixel with slightly jittered positions inside the pixel footprint (a sub-pixel aperture).
- Each ray is traced independently through the scene using the same intersection and shading logic; the resulting radiance samples are then averaged to obtain the final pixel color, which effectively approximates a box filter over the pixel area.

2.7 Soft Shadows with Area Light (Optional)

For the optional soft-shadow component, we extend the hard point-light model to mimic a small rectangular area light:

- Instead of treating the light as a point, we conceptually replace it by a finite patch in space; at each shading point we sample multiple positions on this patch and cast one shadow ray towards each sample.
- If a shadow ray to a particular light sample is occluded, that sample contributes nothing; if it is unoccluded, it contributes a standard diffuse term similar to the hard-shadow case. The final visibility factor is approximated by the fraction of unoccluded samples.

3 RESULTS

After finishing the mandantory tasks,we can render the scene shown as below:

Fig. 1. Scene with hard shadow and no texture

After adding texture, we can render the scene shown as below:

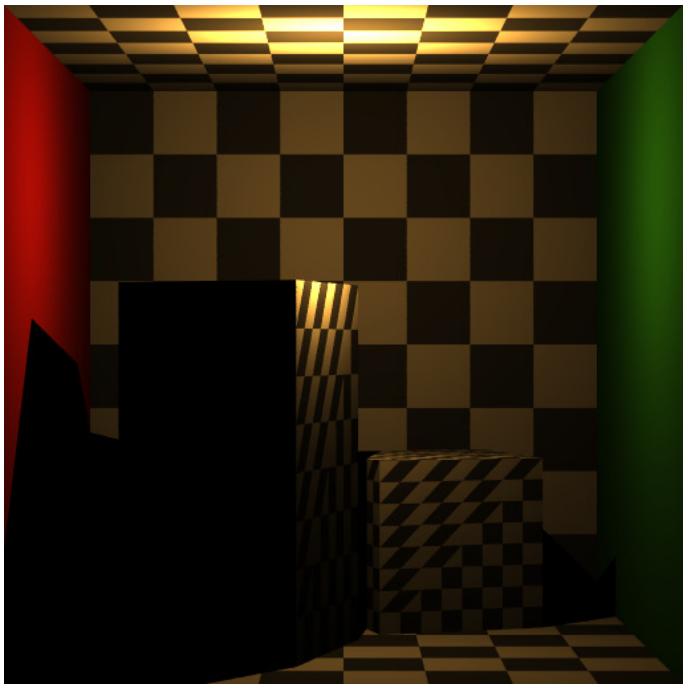

Fig. 2. Scene with Hard Shadow and checkerboard texture

Objects in the scene are mannualy placed, so it is different from the first result.

After implementing soft shadow, we can render the scene shown as below:

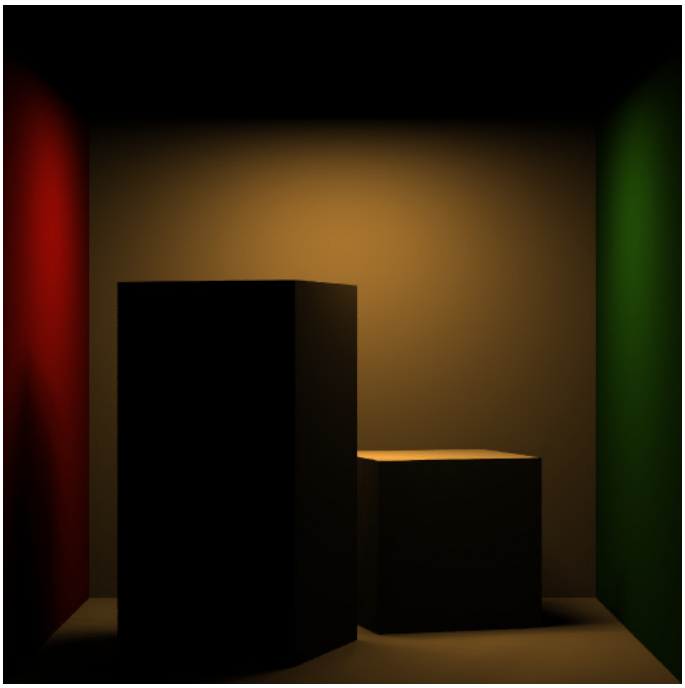

Fig. 3. Scene with soft shadow and no texture