Assignment 3:
Basic Ray Tracing

NAME: 钱嘉烨 (QIAN JIAYE)
STUDENT NUMBER: 2022533118
EMAIL: QIANJY2022@SHANGHAITECH.EDU.CN

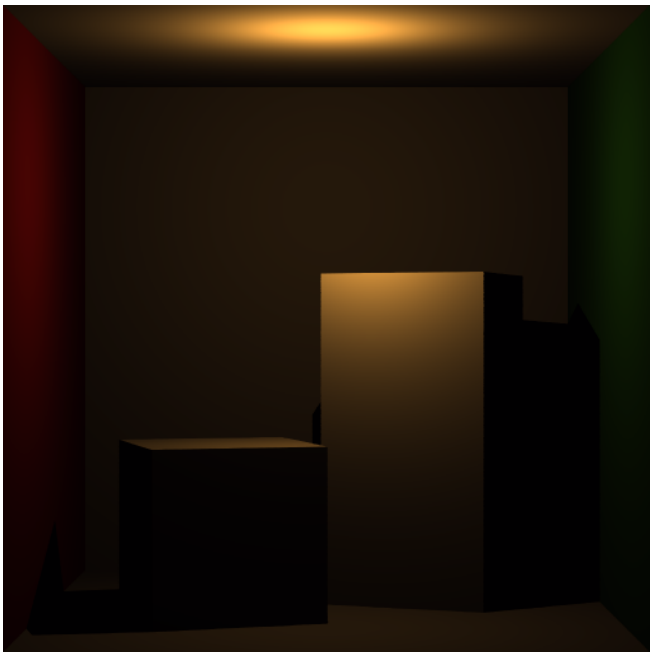

Fig. 1. Result for direct illumination.

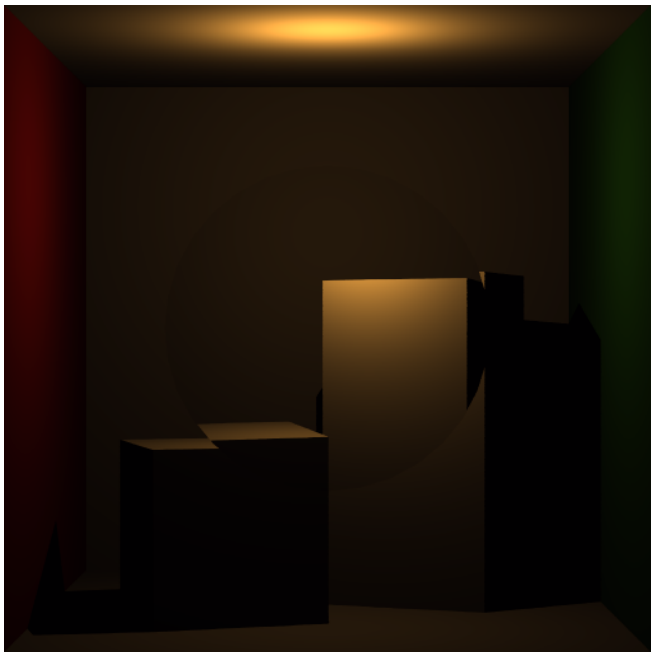

Fig. 2. Result for perfect refractive materials.

1 INTRODUCTION

In this assignment, I implemented basic ray tracing with a direct lighting function with diffuse BRDF. Correct ray refraction for perfect refraction material is also been implemented. To trace the ray, ray-triangle intersection (Sec. 2.1) and ray-AABB intersection (Sec. 2.2) has been implemented first. Then I implemented the BVH search tree construction to make the ray tracing more efficient (Sec. 2.3). Finally, I implemented the `IntersectionTestIntegrator` (Sec. 2.4) and `PerfectRefraction` (Sec. 2.5) for basic ray tracing validation.

2 IMPLEMENTATION DETAILS

2.1 Ray-Triangle Intersection

In the implementation, I used the linear-algebra-based method to compute the intersection point between a ray and a triangle by barycentric interpolation of the triangle. The formula is as follows: given the triangle vertices $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2$, the ray origin $\boldsymbol{o}$ and direction $\boldsymbol{d}$, we solve for parameters u, v, and t such that

$$(1-u-v)\,\boldsymbol{v}_0 + u\,\boldsymbol{v}_1 + v\,\boldsymbol{v}_2 = \boldsymbol{o} + t\,\boldsymbol{d}.$$

Then the final parameters u, v, and t can be formulated as

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\boldsymbol{s}_1 \cdot \boldsymbol{e}_1} \begin{bmatrix} \boldsymbol{s}_2 \cdot \boldsymbol{e}_2 \\ \boldsymbol{s}_1 \cdot \boldsymbol{s} \\ \boldsymbol{s}_2 \cdot \boldsymbol{d} \end{bmatrix}$$

where $\boldsymbol{s} = \boldsymbol{o} - \boldsymbol{v}_0$, $\boldsymbol{e}_1 = \boldsymbol{v}_1 - \boldsymbol{v}_0$, $\boldsymbol{e}_2 = \boldsymbol{v}_2 - \boldsymbol{v}_0$, $\boldsymbol{s}_1 = \boldsymbol{d} \times \boldsymbol{e}_2$, and $\boldsymbol{s}_2 = \boldsymbol{s} \times \boldsymbol{e}_1$. Finally, the program checks whether u, v, and t are valid to determine if the ray intersects the triangle. In particular, the implementation also tests whether $\|\boldsymbol{e}_1 \times \boldsymbol{e}_2\|_2$ is zero to detect degenerate triangles; if so, the ray is considered to never intersect that triangle.

2.2 Ray-AABB Intersection

To determine whether a ray intersects an Axis-Aligned Bounding Box (AABB), the implementation first computes the parametric intersection distances between the ray and the six bounding planes of the AABB. Specifically, for each axis (x, y, and z), the entry and exit points of the ray with respect to the minimum and maximum bounds of the box are calculated. These values are then compared

to identify the overall interval in which the ray may remain inside the box. See https://www.scratchapixel.com/lessons/3d-basic-rendering/minimal-ray-tracer-rendering-simple-shapes/ray-box-intersection.html for more details.

2.3 BVH Construction

The construction of a Bounding Volume Hierarchy (BVH) is a recursive process. First, a bounding box is computed that encloses all objects in the current region. If the recursion depth exceeds a predefined threshold or the number of objects falls below a certain limit, a leaf node is created, signaling the termination of recursion. Otherwise, the bounding box's longest axis is chosen as the splitting direction, and the objects are divided at the median of their centroids along this axis. The two resulting subsets are then used recursively to construct the left and right subtrees. Finally, the bounding boxes of the subtrees are merged and stored as the bounding box of the current internal node.

2.4 Intersection Test Integrator

To perform rendering, we sample several points on each pixel and construct rays emitted from the camera center. After projection computation (see Sec. 2.5), each ray eventually intersects with an object. Since we are rendering perfectly diffuse objects and both the intersection point and the new ray from the intersection to the light source are known, we can directly compute the cosine of the angle between the surface normal at the intersection and the new ray. This gives us the reflectance, allowing us to finally determine the color contribution of the sample.

2.5 Ray Tracing for Perfect Refraction

To simulate perfectly refractive objects, we perform specific computations during ray tracing. Whenever a ray intersects a perfectly refractive object, we update the original ray to a new ray that originates from the intersection point, with its direction determined according to the refraction. If the incident ray cannot undergo refraction, we instead account for the possibility of total internal reflection.

3 RESULTS

The result for direct illumination is shown in Fig. 1, which shows correct diffuse shading. The result for perfect refractive materials is shown in Fig. 2, which shows correct refraction effects.