Assignment 3

NAME: YUXINFAN
STUDENT NUMBER:2023533044
EMAIL: FANYX2023@SHANGHAITECH.EDU.CN

1 INTRODUCTION

This report describes the implementation of a basic ray tracer, including ray-triangle intersection, BVH acceleration, direct illumination, and refractive materials.

2 IMPLEMENTATION DETAILS

2.1 Ray-Triangle Intersection

I implemented the MöllerTrumbore intersection algorithm, which is a fast method for calculating the intersection of a ray and a triangle in three dimensions without needing to precompute the plane equation of the plane containing the triangle.

Given a ray $R(t) = O + tD$ and a triangle defined by vertices V_0, V_1, V_2, the intersection point P can be expressed using barycentric coordinates (u, v):

$$P = (1 - u - v)V_0 + uV_1 + vV_2 \tag{1}$$

Equating the ray and triangle equations gives a linear system:

$$O - V_0 = -tD + u(V_1 - V_0) + v(V_2 - V_0) \tag{2}$$

Let $E_1 = V_1 - V_0$, $E_2 = V_2 - V_0$, and $T = O - V_0$. The system can be solved using Cramer's rule:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times E_2) \cdot E_1} \begin{bmatrix} (T \times E_1) \cdot E_2 \\ (D \times E_2) \cdot T \\ (T \times E_1) \cdot D \end{bmatrix} \tag{3}$$

The intersection is valid if $u \geq 0, v \geq 0, u + v \leq 1$ and t is within the valid range $[t_{min}, t_{max}]$.

2.2 Ray-AABB Intersection

I used the Slab Method for Ray-AABB intersection. An Axis-Aligned Bounding Box (AABB) is the intersection of three pairs of parallel planes (slabs). For a ray to intersect the box, it must intersect all three slabs within the same time interval.

For each axis $i \in \{x, y, z\}$, we compute the intersection times with the slab planes:

$$t_{0,i} = \frac{p_{min,i} - o_i}{d_i}, \quad t_{1,i} = \frac{p_{max,i} - o_i}{d_i} \tag{4}$$

We ensure $t_{0,i} \leq t_{1,i}$ by swapping if necessary. The intersection interval $[t_{enter}, t_{exit}]$ is updated incrementally:

$$t_{enter} = \max(t_{enter}, t_{0,i}), \quad t_{exit} = \min(t_{exit}, t_{1,i}) \tag{5}$$

If $t_{enter} \leq t_{exit}$ and the interval overlaps with the ray's valid time range, an intersection occurs.

2.3 BVH Construction

I implemented a Bounding Volume Hierarchy (BVH) using a recursive top-down approach.

- **Stop Criteria**: The recursion terminates when the tree depth reaches `CUTOFF_DEPTH` (22) or the number of primitives in a node is small (≤ 4).
- **Splitting Heuristic**: I employed the Median Split method.
  (1) Determine the axis with the largest extent of the node's bounding box.
  (2) Sort the primitives based on their centroids along this axis. I used `std::nth_element` to find the median element efficiently in $O(N)$ time without fully sorting the array.
  (3) Split the primitives into left and right child nodes at the median index.

2.4 Direct Illumination Integrator

The `IntersectionTestIntegrator` computes the radiance at a pixel by integrating direct lighting.

- **Anti-Aliasing**: For each pixel, I generate spp rays with random sub-pixel offsets to perform Monte Carlo integration over the pixel area.
- **Refraction Handling**: If a ray hits a refractive surface, the integrator recursively calls `Li` with a new ray spawned in the refracted direction (computed via Snell's Law).
- **Direct Lighting**: For diffuse surfaces, I compute the contribution from the point light source:

  $$L_o = \rho \cdot \frac{\Phi}{4\pi r^2} \cdot \max(0, N \cdot L) \cdot V(p, light) \tag{6}$$

  where Φ is the light flux, r is the distance to the light, and V is the visibility term (1 if visible, 0 if occluded).
- **Shadow Ray**: A shadow ray is cast from the intersection point to the light source. If it intersects any object before reaching the light (checked via `scene->intersect`), the point is in shadow.

2.5 Refraction (BSDF)

I implemented perfect specular refraction based on Snell's Law: $\eta_i \sin\theta_i = \eta_t \sin\theta_t$.

- **Relative IOR**: I determine the relative index of refraction $\eta = \eta_i/\eta_t$ by checking the dot product of the incident ray and the normal $(D \cdot N)$. If entering, $\eta = 1/\eta_{mat}$; if exiting, $\eta = \eta_{mat}$.
- **Total Internal Reflection (TIR)**: I calculate $\sin^2\theta_t = \eta^2(1 - \cos^2\theta_i)$. If this value is greater than 1, TIR occurs, and no refraction happens (in this implementation, we simply return).
- **Direction**: The refracted direction is computed as:

  $$T = \eta D + (\eta\cos\theta_i - \sqrt{1 - \sin^2\theta_t})N \tag{7}$$

Computer Graphics I Course Assignment

student number:2023533044
email: fanyx2023@shanghaitech.edu.cn

3 RESULTS

The renderer successfully produces images with direct illumination, shadows, and refraction effects.

Fig. 1. Rendered scene with refraction effects