Assignment 3: Global Illumination

NAME: CHANGGUI XU
STUDENT NUMBER: 2022533095
EMAIL: XUCHG2022@SHANGHAITECH.EDU.CN

1 INTRODUCTION

In this assignment, the following tasks are finished.

- Must1: Compile the source code and configure the language server environment.
- Must2: Implement ray-triangle intersection functionality.
- Must3: Implement ray-AABB intersection functionality.
- Must4: Implement the BVH (Bounding Volume Hierarchy) construction.
- Myst5: Implement the IntersectionTestIntegrator and PerfectRefraction material for basic ray tracing validation, handing refractive and solid surface interactions.
- Must6: Implement a direct lighting function with diffuse BRDF and shadow testing.
- Must7: Implement anti-aliasing via multi-ray sampling per pixel within a sub-pixel aperture.

2 IMPLEMENTATION DETAILS

2.1 Ray-triangle Intersection

In ray tracing, ray-triangle intersection is one of the fundamental calculations. In this project, we implemented **ray-triangle intersection** using the well-known **Möller-Trumbore algorithm**, which is an efficient method for ray-triangle intersection. The algorithm uses vector math to calculate the intersection point between a ray and a triangle.

Given the three vertices of the triangle $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$, the parametric equation of the ray is:

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

where $\mathbf{o}$ is the origin of the ray, $\mathbf{d}$ is the direction vector of the ray, and t is a parameter representing the distance from the origin to the intersection point. The two edge vectors of the triangle are:

$$\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0, \quad \mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$$

To calculate the intersection, the following steps are performed:

1. Compute the cross product between the ray direction and the triangle's edge:

$$\mathbf{h} = \mathbf{d} \times \mathbf{e}_2$$

This step helps us to determine whether the ray is parallel to the triangle.

2. Compute the determinant $\mathbf{a}$:

$$\mathbf{a} = \mathbf{e}_1 \cdot \mathbf{h}$$

If $\mathbf{a}$ is close to zero, the ray is parallel to the triangle and does not intersect it.

3. Solve for the intersection parameters u, v, and t:

$$u = \frac{\mathbf{f} \cdot \mathbf{h}}{\mathbf{a}}, \quad v = \frac{\mathbf{g} \cdot \mathbf{h}}{\mathbf{a}}, \quad t = \frac{\mathbf{e}_2 \cdot \mathbf{h}}{\mathbf{a}}$$

where $\mathbf{f}$ and $\mathbf{g}$ are vectors derived from the ray's origin to one of the triangle's vertices.

4. Check if the intersection point is inside the triangle: Ensure the intersection parameters satisfy:

$$u \geq 0, \quad v \geq 0, \quad u + v \leq 1$$

If these conditions are met, the intersection point lies inside the triangle.

2.2 Ray-AABB Intersection

An AABB is defined by two opposite corners, typically represented as the minimum and maximum points in 3D space. The edges of the AABB are aligned with the coordinate axes, which makes the intersection calculations relatively simple.

The key to performing ray-AABB intersection is solving for the times t_{in} and t_{out}, where the ray enters and exits the AABB. Given a ray with the parametric equation:

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

where $\mathbf{o}$ is the ray's origin and $\mathbf{d}$ is the ray's direction, the AABB is defined by its minimum and maximum bounds $\mathbf{min} = (x_{min}, y_{min}, z_{min})$ and $\mathbf{max} = (x_{max}, y_{max}, z_{max})$.

To determine whether the ray intersects the AABB, we calculate the intersection times for each dimension x, y, and z. We first compute the inverse of the ray's direction:

$$\mathbf{inv\_dir} = \frac{1}{\mathbf{d}}$$

Then, for each dimension, we calculate the entry and exit times for the ray:

$$t_{\text{min}} = \frac{\mathbf{min} - \mathbf{o}}{\mathbf{d}}, \quad t_{\text{max}} = \frac{\mathbf{max} - \mathbf{o}}{\mathbf{d}}$$

If $t_{\text{min}} > t_{\text{max}}$, we swap the values. The ray intersects the AABB if the t_{min} and t_{max} values overlap in all three dimensions.

2.3 BVH Construction

The idea behind BVH is to recursively partition the scene into smaller sets of objects by enclosing them in bounding volumes. Each bounding volume represents a group of objects, and these volumes are organized into a binary tree. The internal nodes of the BVH represent bounding volumes that enclose their child nodes, while the leaf nodes represent individual objects in the scene.

© 2025
Computer Graphics I Course Assignment

The construction process generally follows these steps:

(1) Compute the BB for each object in the scene. This is typically done by calculating the AABB for each object.
(2) Sort the objects based on their spatial location (often by the centroid or position).
(3) Recursively split the objects into two groups based on the median of their bounding boxes. Each group is recursively partitioned until each leaf node contains a small number of objects.
(4) Build the BVH tree, where each node contains a bounding volume that encloses all the objects in its subtree.

The main advantage of BVH is that it organizes the scene objects in a way that allows for fast intersection testing by eliminating large portions of the scene (that do not intersect the ray) early in the traversal.

2.4 Basic Ray Tracing Validation

In the project, we implemented the **IntersectionTestIntegrator** and the **PerfectRefraction** material for basic ray tracing validation. These components are essential for simulating interactions between rays and surfaces, particularly for handling refraction and solid surface interactions, such as reflections and diffuse interactions.

The IntersectionTestIntegrator is responsible for computing ray-object intersections and calculating the color contribution based on material properties such as diffuse reflection, perfect refraction, and direct lighting. It iterates over the rays cast into the scene and tests for intersections with objects. Once an intersection is found, the color contribution from the light sources is computed, taking into account the material's properties.

The integrator works by tracing rays through the scene and interacting with surfaces. If the material is **diffuse**, it calculates the direct lighting contribution. If the material is **perfectly refractive**, the ray is refracted using Snell's Law to calculate the refraction direction.

The PerfectRefraction material simulates perfect refraction according to **Snell's Law**. Snell's Law relates the angle of incidence θ_1 and the angle of refraction θ_2 through the refractive indices of the two media:

$$\eta_1 \sin(\theta_1) = \eta_2 \sin(\theta_2)$$

Where η_1 and η_2 are the refractive indices of the incident and transmitted mediums, respectively. Given the incoming ray direction **wi** and the normal **n**, the refraction direction **wt** can be computed as follows:

$$\mathbf{wt} = \eta_1 \mathbf{wi} + (\eta_1 \cos(\theta_1) - \cos(\theta_2))\mathbf{n}$$

where θ_1 is the angle between the incident ray and the surface normal, and θ_2 is the angle of refraction.

2.5 Direct Lighting

The **diffuse BRDF** models the ideal Lambertian reflectance, where the amount of light reflected by a surface is independent of the viewing angle. The reflected radiance is proportional to the cosine of the angle between the light direction and the surface normal. This is described by the Lambertian reflection model:

$$L_r = \frac{\rho}{\pi} L_i \cos(\theta)$$

where: L_r is the reflected radiance. ρ is the albedo (reflectivity) of the surface. L_i is the incident radiance from the light source. θ is the angle between the light direction **l** and the surface normal **n**.

In addition to the diffuse reflection, the light intensity is attenuated based on the square of the distance from the light source to the surface point. The attenuation factor is:

$$\text{attenuation} = \frac{1}{r^2}$$

Where r is the distance from the light source to the point of intersection on the surface.

In the real world, not all light rays from a light source reach the surface due to occlusions by other objects. **Shadow testing** is implemented by casting a ray from the surface point towards the light source. If this ray intersects any geometry in the scene before reaching the light, the point is considered shadowed, and no direct light contribution is computed for that point.

The shadow ray is tested by casting a ray from the surface point **p** to the light source position **l**. If the ray intersects any object before reaching the light, the point is in shadow.

2.6 Anti-Aliasing

Anti-aliasing aims to reduce the visual artifacts caused by undersampling, which occurs when the image resolution is too low relative to the scene's detail. In ray tracing, this manifests as jagged edges, also known as "staircase" effects, on objects and boundaries. The **multi-ray sampling** technique solves this problem by taking multiple samples within each pixel and averaging the results. This technique is commonly referred to as supersampling.

The theory behind multi-ray sampling is simple: for each pixel, we randomly or uniformly sample several rays from different subpixels within the pixel's area. By casting these rays, we gather a more accurate representation of the lighting and color at that pixel. The final pixel color is then computed by averaging the contributions from all the rays, which helps smooth out high-frequency details and reduce aliasing.

Mathematically, we can describe the anti-aliasing process as follows:

$$L_{\text{final}} = \frac{1}{N} \sum_{i=1}^{N} L_i$$

where: L_{final} is the final color of the pixel. N is the number of rays sampled per pixel (i.e., the sample count). L_i is the color contribution from the i-th ray.

3 RESULTS

Fig. 1 shows the situation of direct illumination, and Fig. 2 shows the situation of perfect refraction.

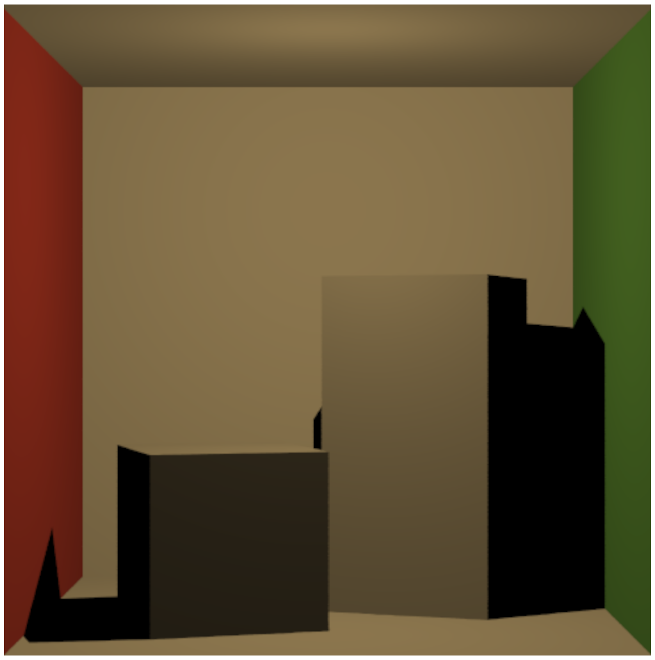

Fig. 1. Direct illumination

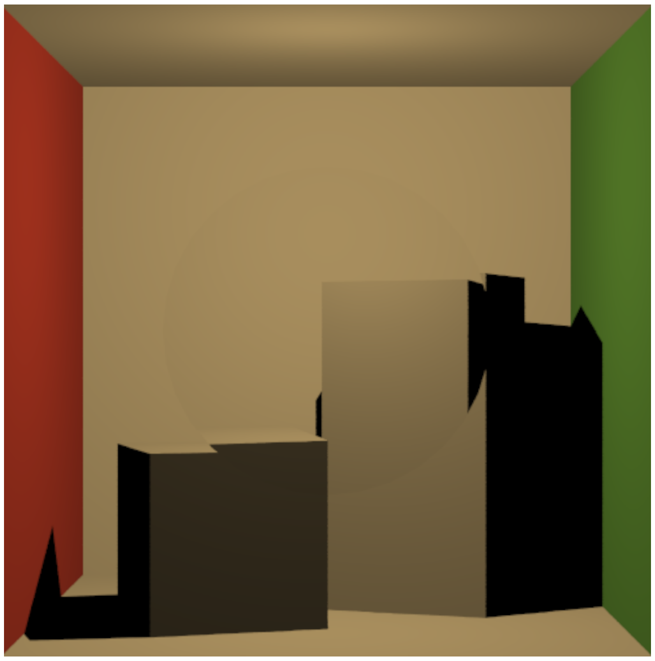

Fig. 2. Prefect refraction