CPU-based Ray Tracing for Global Illumination

Teng-Sung Yu
Uppsala University
sodamax1740@gmail.com

Zhen-Tang Huang
Uppsala University
huang1473690@gmail.com

Tse-An Hsu
Uppsala University
jasonhsutsean@gmail.com

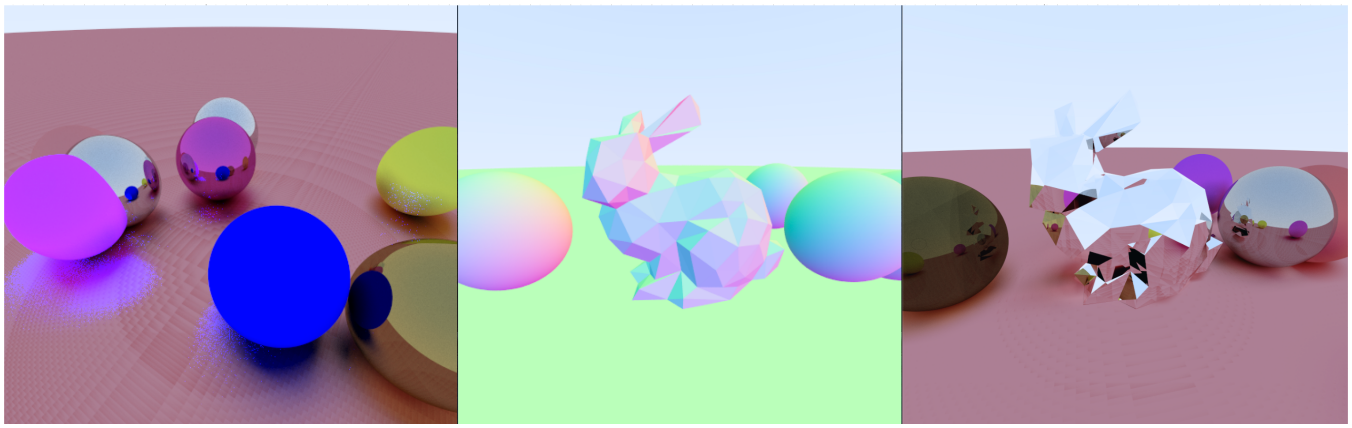

Figure 1. Results of ray tracer

1 Introduction

In computer graphics, ray tracing is a rendering method that is frequently used to produce photorealistic images. To accurately produce reflections, and shadows, it involves modelling the path of light as it interacts with the objects in a scene. A crucial aspect of ray tracing is global illumination, which is the accurate simulation of light interacting with all surfaces in a scene. In this project, we will be mainly based on the book "Ray Tracing in One Weekend".[1]

- Implement anti-aliasing.
- Implement rendering of diffuse surfaces.
- Add a few more spheres and metal surface types to our scene.
- Add an ImGUI window for controlling rendering parameters such as material properties, anti-aliasing, number of bounces, etc.
- Improve the speed of rendering low-polygon meshes.

2 Our Approach

2.1 Implement anti-aliasing

We average many samples for each pixel and add a random intensity per sample in order to achieve anti-aliasing.

2.2 Implement rendering of diffuse surfaces

Update the ray color by randomly scattering the incoming ray to a new random direction ray. After that apply attenuation to the new ray color. In this way we can implement the diffuse surface to the spheres.

2.3 Add a few more spheres and more surface types to your scene

We implemented metal surface to the spheres. The method we implement metal surface is that we calculate the reflect ray direction according the incoming ray. We also add a weight variable, called fuzz and multiply with a random direction to reflect ray in order to control the roughness of the metal surface.

2.4 Add an ImGUI window for controlling rendering parameters

We have ImGUI window for controlling rendering parameters such as material properties(value of fuzz), anti-aliasing, and number of bounces. It could also display the surface normals and toggle gamma correction on/off. We create some corresponding variables to control the parameters.

2.5 Speed up rendering low-polygon meshes

We first calculate the bounding volume (sphere) of the low-polygon meshes. We add a bounding volume test to check the ray. We will determine whether the ray will intersect the bounding sphere or not. If the ray does intersect the volume, we will continue to calculate the ray hitting the meshes. Otherwise, we could skip computing this ray and simply return the background color. We could almost reduce the rendering time by half with this approach compared to the original algorithm.

References

[1] Peter Shirley. 2020. *Ray Tracing in One Weekend.* https://raytracing.github.io/books/RayTracingInOneWeekend.html