README.md

Ray Tracing : CPU version vs CPU + Bounding Box + BVH version vs GPU version

Author

- 網媒所 王哲瑋 Che Wei Wang R10944037

Code Structure

- `main.js` ：setting up rendering scene using webgl.
- `raytrace.frag` , `raytrace.vert` ：fragment shader and vertex shader where ray tracing is implemented.
- `m3.js` , `m4.js` , `v3.js` ：some 3D maths functions.
- `utilities.js` ：some convenient utility functions.
- `index.html` ：web page that hold the render canvas.

How To Execute/Demo

- visit https://wayne0419.github.io/GPURayTracing/ for a demo of the GPU version.

About The Project

In this project, three versions of Ray Tracer are being implemented.

1. CPU version
2. CPU + Bounding Box & Binary Volume Hierarchy Optimization version.
3. GPU version (WebGL fragment shader).

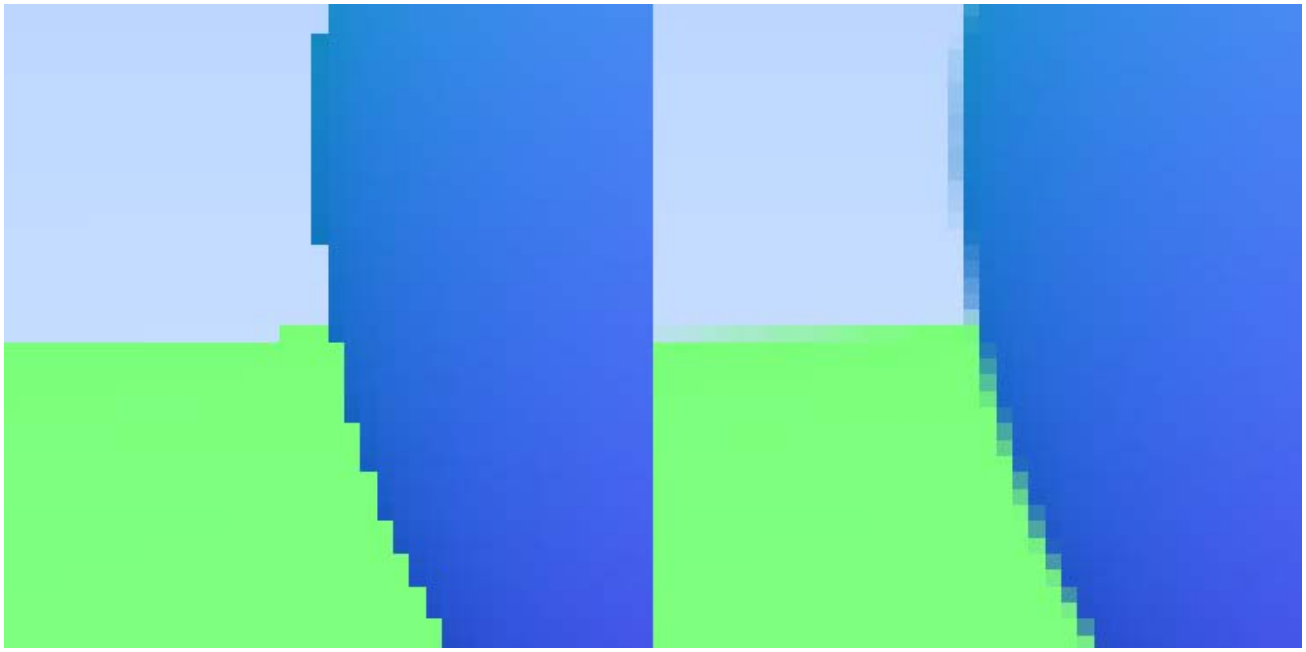
I will give a brief introduction of the implementation of the ray tracer and then do a render speed comparison between them.

Implemented Feature Details Of The Ray Tracer

Anti-Aliasing

The method that I use to do anti-aliasing is that when rendering a pixel, instead of sending a ray through the center of a pixel and using the color that ray gets to render the whole pixel, I send multiple rays through different positions of that pixel and then use the average color of those rays to render the pixel.

Below is a comparison of without vs. with anti-aliasing.



Three Materials

Diffuse

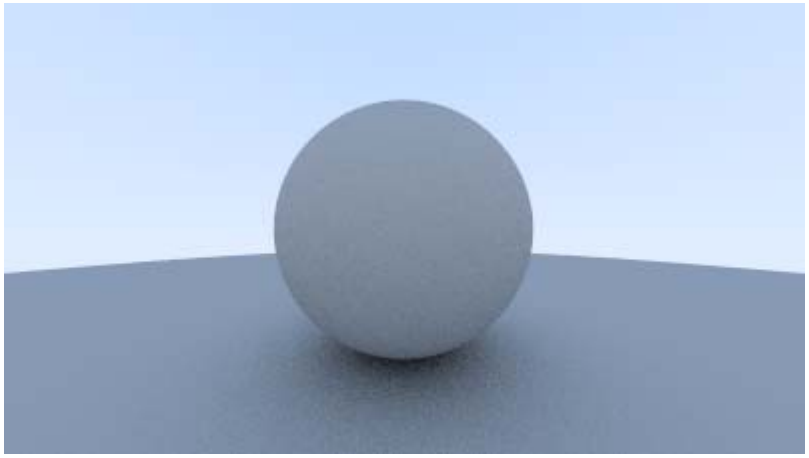
Three different diffuse method are being implemented.

The idea of a diffuse surface is that when light/ray that hits it, the reflected light/ray goes into random direction.

The difference between this three diffuse methods lies in the way they randomize the reflected light/ray.

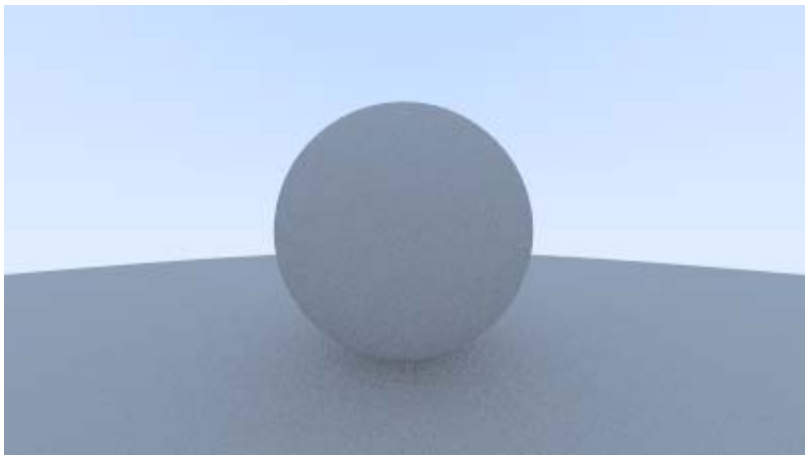
- Unit Sphere Diffuse

For unit_sphere diffuse, the reflected target lies at a random position inside the unit sphere tangent to the hit object at the hit point.



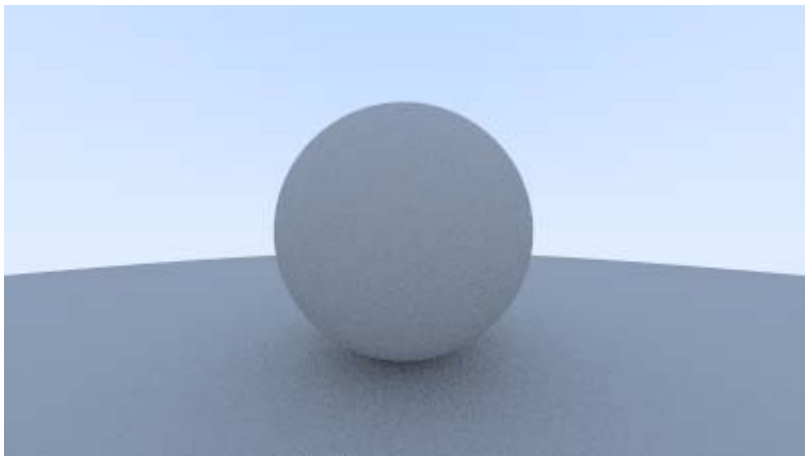
- Hemisphere Diffuse

For hemisphere diffuse, the reflected target lies at a random position inside a unit hemisphere centered at the hit point.



- Lambertian Diffuse

For lambertian diffuse, the reflected target lies at a random position on the surface of the unit sphere tangent to the hit object at the hit point.

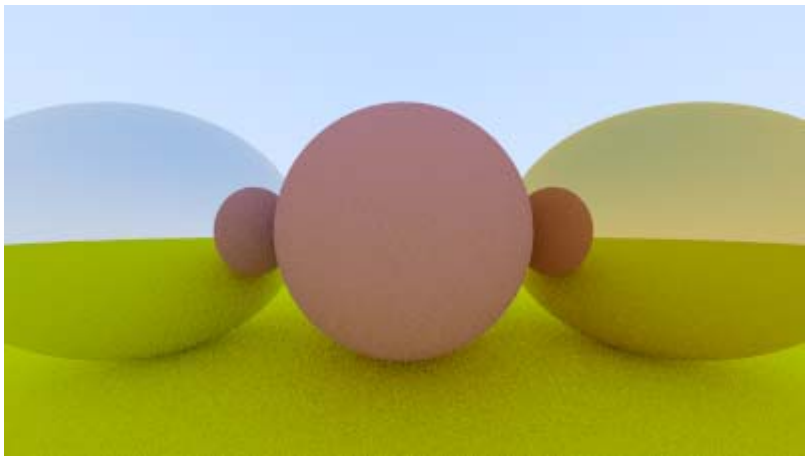


Metal

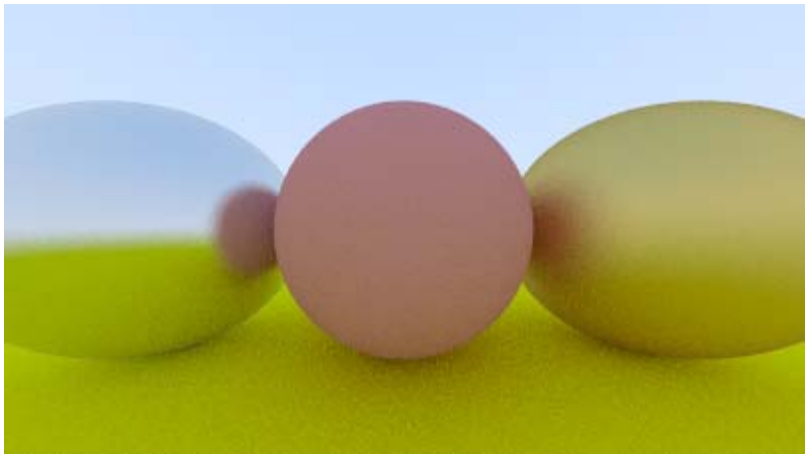
Metal material include shiny metal and fuzzy metal.

The difference between them is that for fuzzy metal, a small random shift is applied to the target of the reflected ray.

- Shiny Metal



- Fuzzy Metal

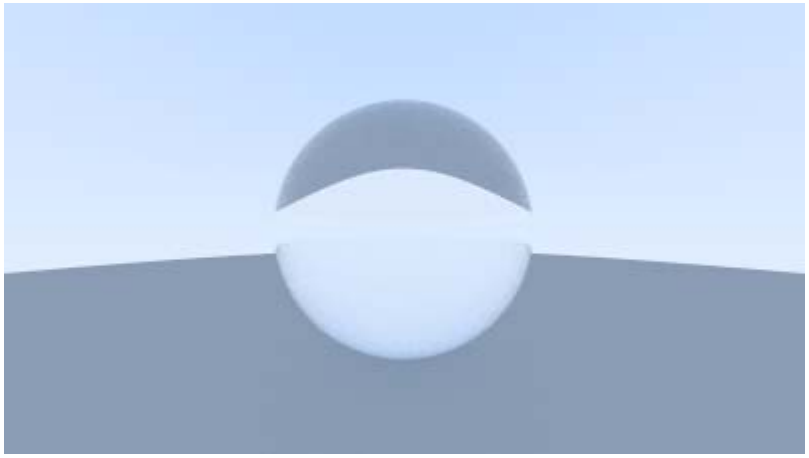


Glass

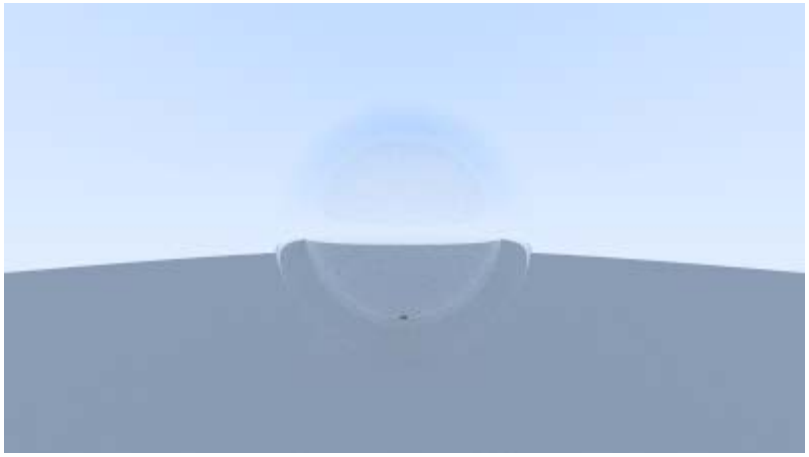
To implement a glass surface, we need to decide when should a ray reflect, when should it refract.

When the incident ray angle is large enough, only reflection happens.

When the incident ray angle decreases, the chance of refraction increases.



By putting a second glass sphere with negative radius inside the first glass sphere, I can get a hollow glass effect.



Putting Thre Materials together

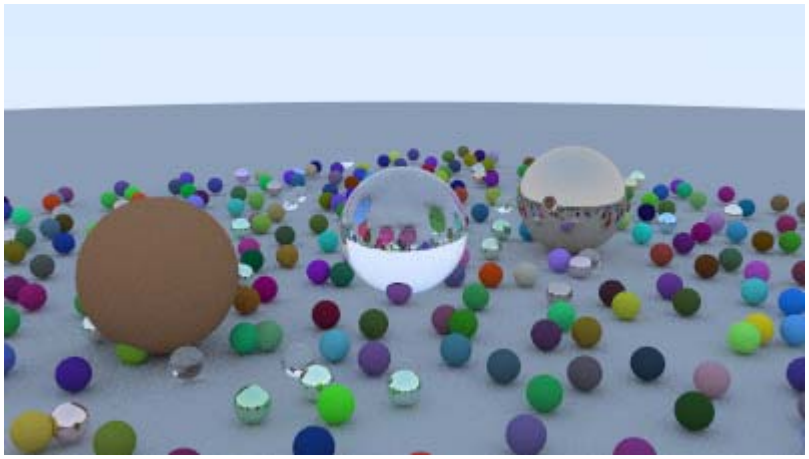
Render Speed Comparison

CPU version vs. CPU + Bounding Box & Binary Volume Hierarchy Optimization version

- Device: GTX 960M + i7-6700HQ
- 400x225p, 100 anti-aliasing samples, 10 max ray bounces

- ~ 500 Spheres

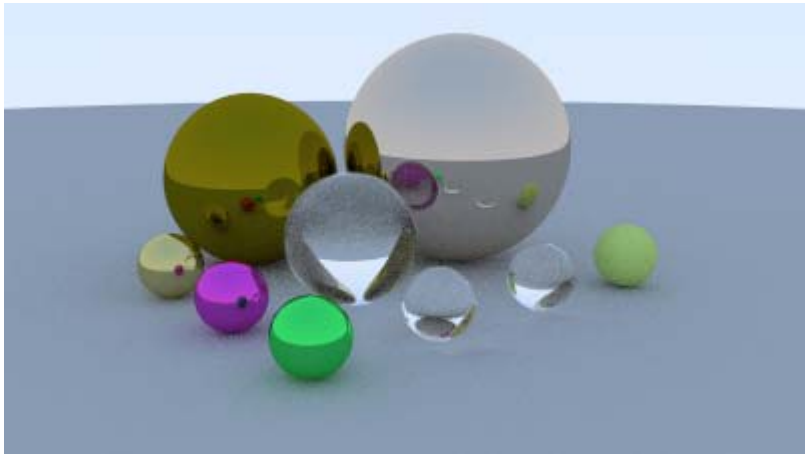
CPU version	CPU + Bounding Box & Binary Volume Hierarchy Optimization version
12963 s (3.6 hrs)	296 s (5 mins)



CPU version vs. GPU version

- Device: GTX 960M + i7-6700HQ
- 400x225p, 80 anti-aliasing samples, 10 max ray bounces
- ~ 10 Spheres

CPU version	CPU + BB & BVH version	GPU version
56 s	73 s	~ 1 s Try it yourself



Reference

Ray Tracing In One Weekend

WebGL Fundamental