Ray Tracing Renderer

Mei Yixuan

January 8, 2022

1 Introduction

This document describes the design of a simple ray tracing renderer, along with some sample scenes rendered using it. I finished it as course project for Advanced Computer Graphics, given by Prof. Shimming Hu in Tsinghua University.

Complete code for this renderer can be found in https://github.com/AntonyMei/RayTracingRender, along with all external libraries and resources. This code is designed for this course project only and implies no warranty, use at your own risk. All code follow Apache License 2.0, except those external libraries and resources.

2 Results

This section shows some of the demo scenes rendered using this renderer. All scenes are rendered under 3840 * 2160 (or 3840 * 3840 if aspect ratio is 1) with more than 1000 samples per pixel and tracing depth 50.

2.1 Hollow Glass Ball

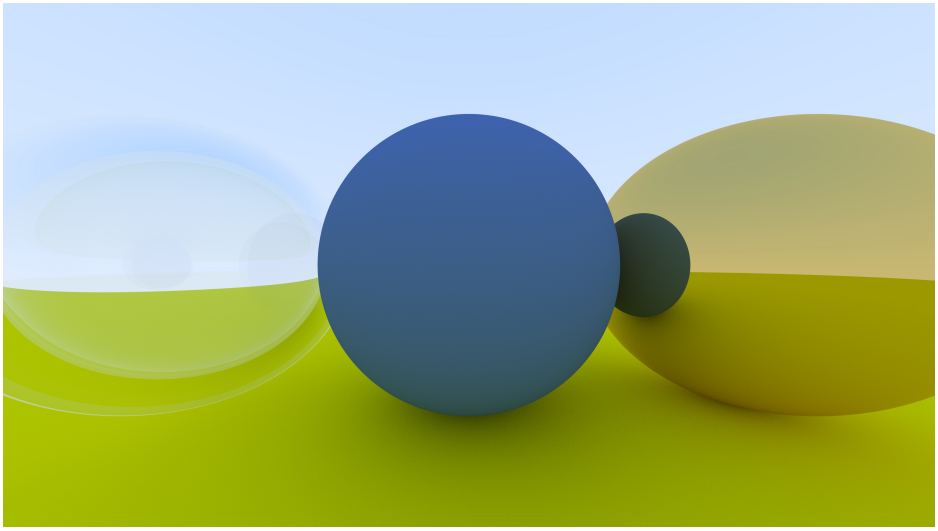

Figure 1: Hollow Glass Ball

This scene shows the usage of Lambertian (pure diffuse), Metallic (pure reflection) and Dielectric (refraction) material.

2.2 Hollow Glass Ball Small FOV

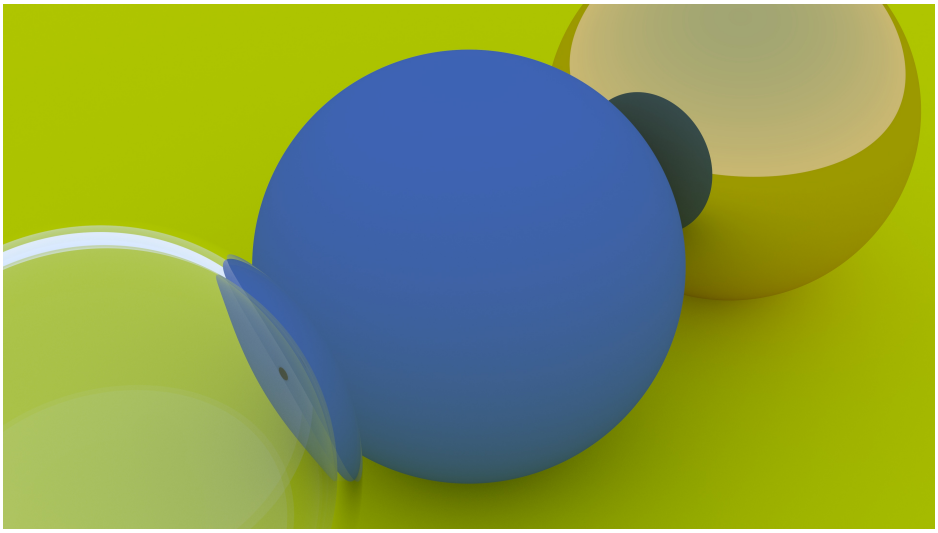

Figure 2: Hollow Glass Ball Small FOV

This scene is identical to the last one, but with smaller camera FOV. Note that edge of the dielectric ball has reflection rather than refraction, which is physically accurate.

2.3 Hollow Glass Ball Off Focus

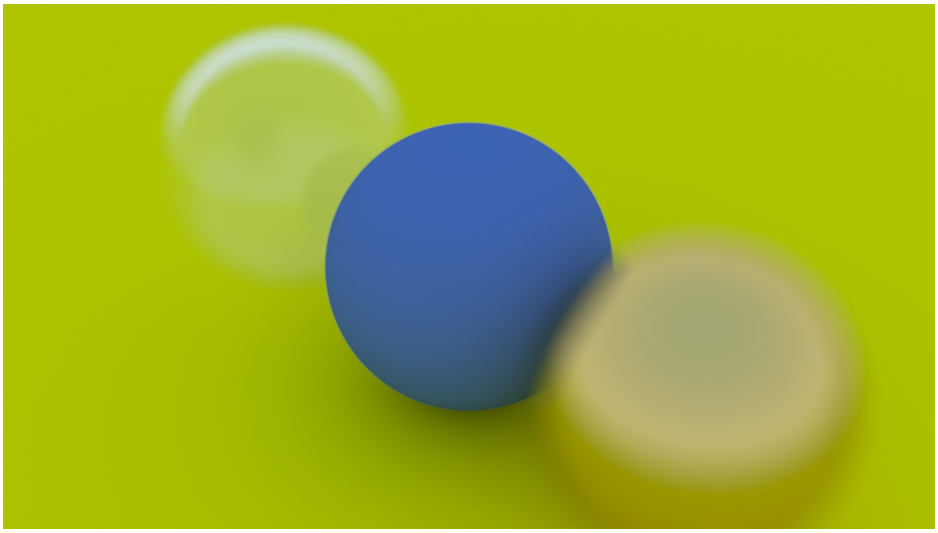

Figure 3: Hollow Glass Ball Off Focus

This scene shows off-focus blur effect.

2.4 Many Balls

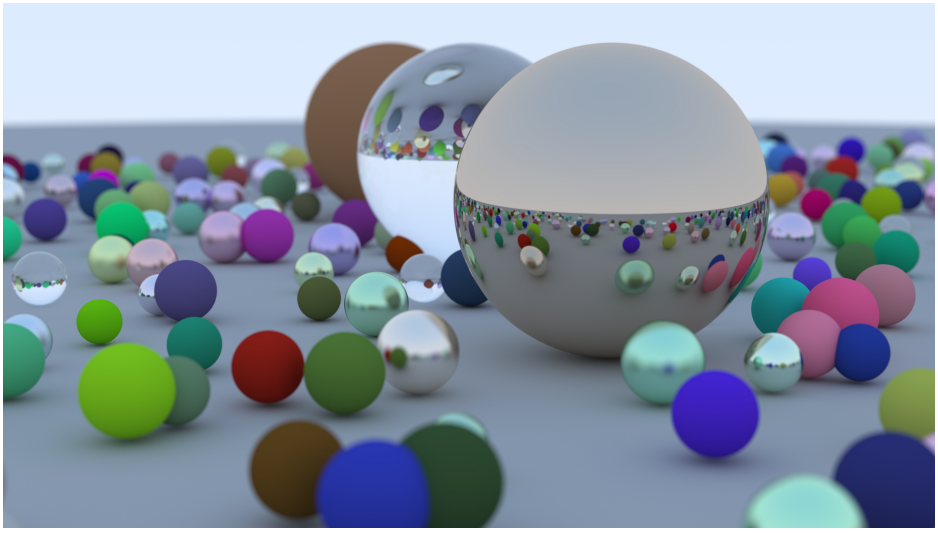

Figure 4: Many Balls

This scene contain many balls. Note that metallic material can use fuzziness to simulate imperfect reflection.

2.5 Motion Blur

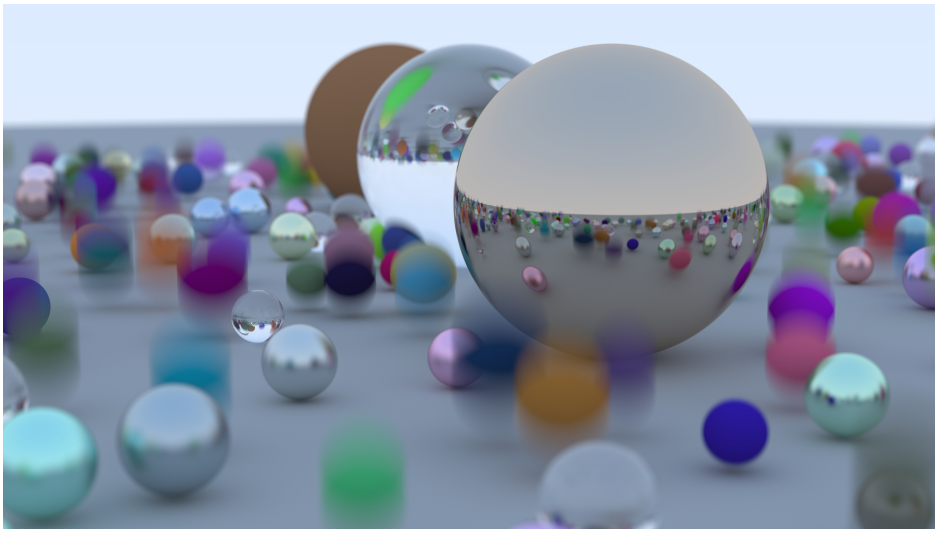

Figure 5: Motion Blur

This scene shows motion blur effect.

2.6 Motion Blur Checker

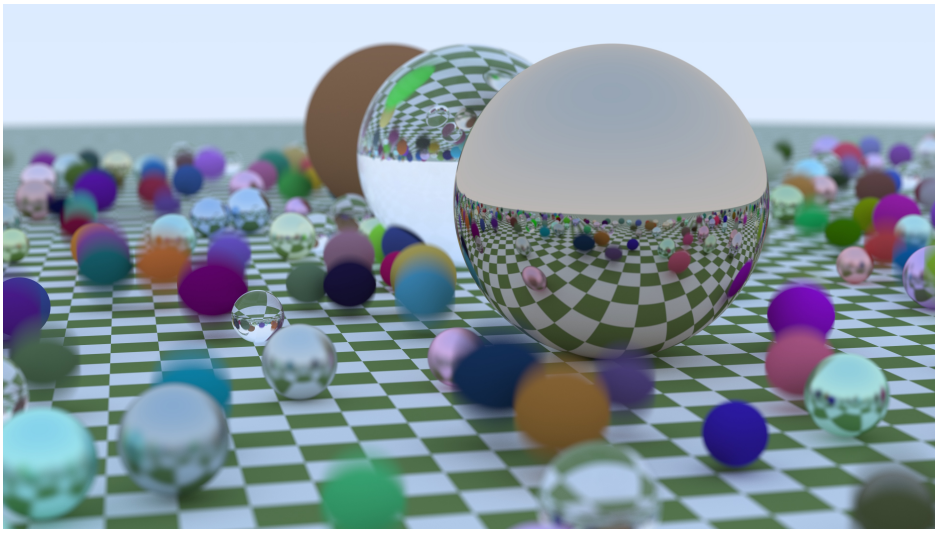

Figure 6: Motion Blur Checker

This scene adds a simple procedural texture (checker texture) to the last one.

2.7 Earth

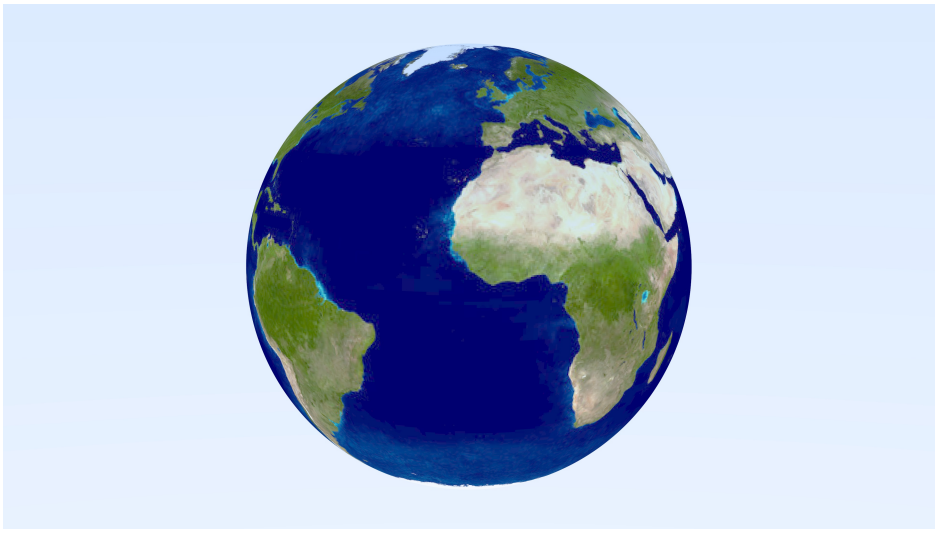

Figure 7: Earth

This scene is a ball with image texture.

2.8 Cornell Box Series

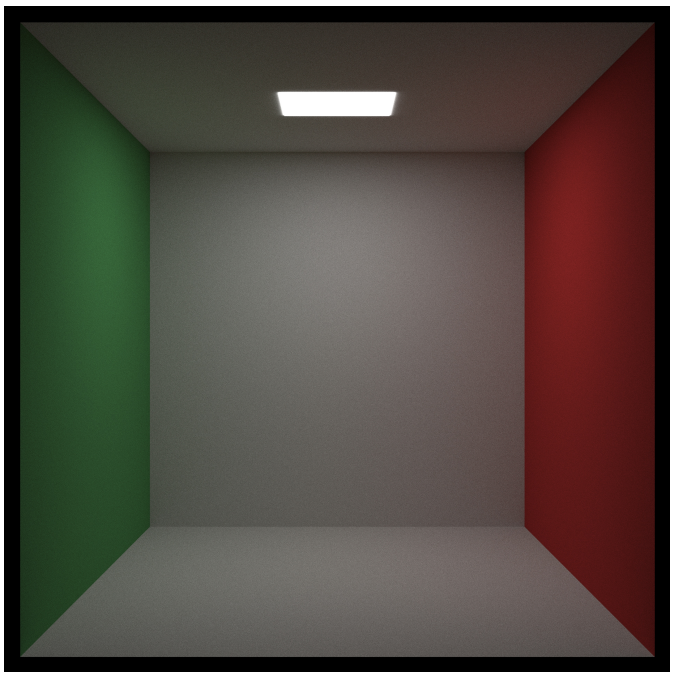

Figure 8: Cornell Box Empty

This is an empty Cornell Box constructed with axis-aligned rectangles.

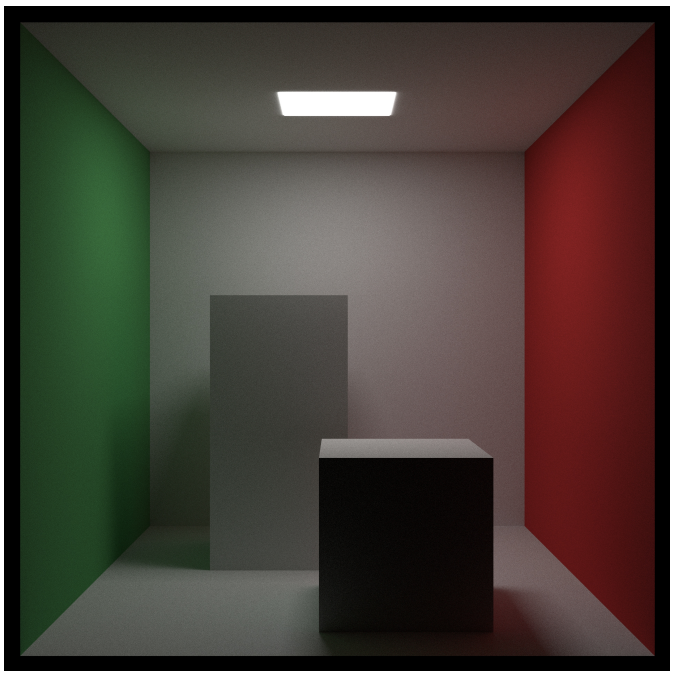

Figure 9: Cornell Box Tow Blocks

Added two boxes to the last scene.

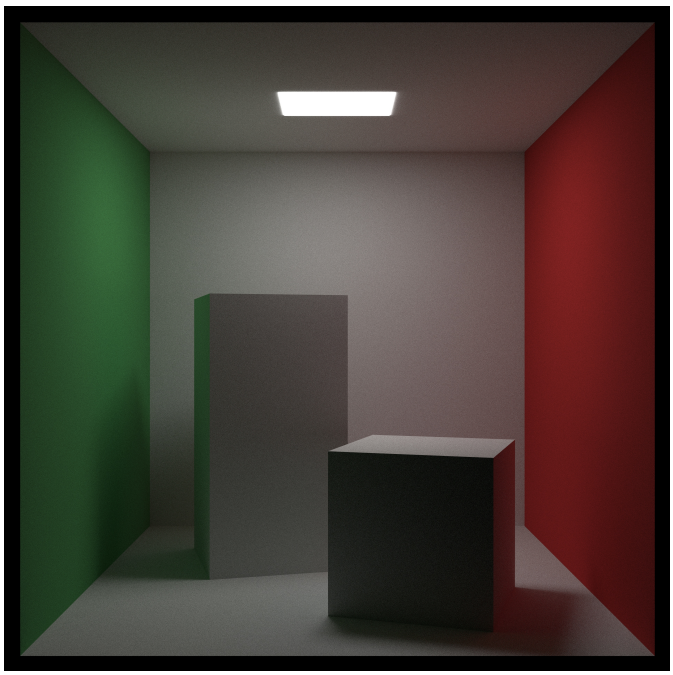

Figure 10: Cornell Box

Add rotation to the two boxes. Color bleeding is obvious on two surfaces facing walls.

2.9 Cornell Box Participating Media

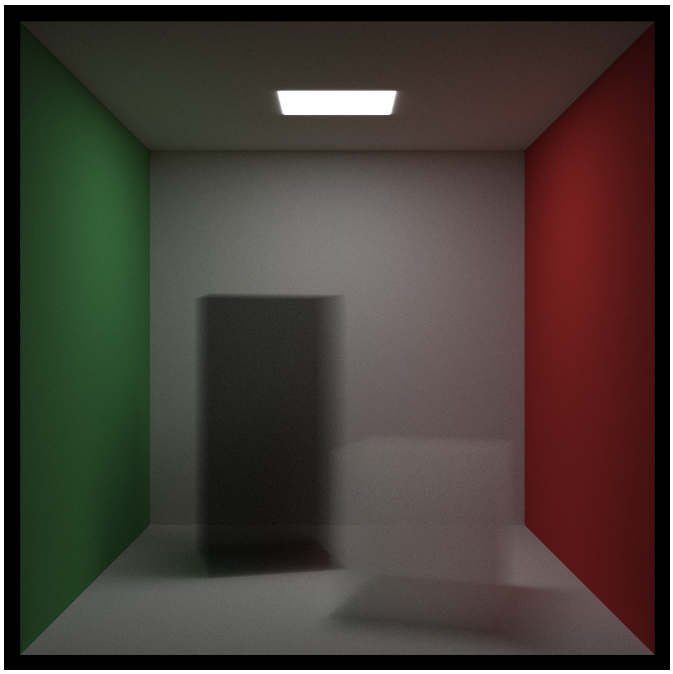

Figure 11: Cornell Box Participating Media

This scene replace the two original boxes with participating media boxes. These boxes are assigned with isotropic material to simulate the effect of smoke. This is a technique often used in volumetric rendering.

2.10 Book2 Final

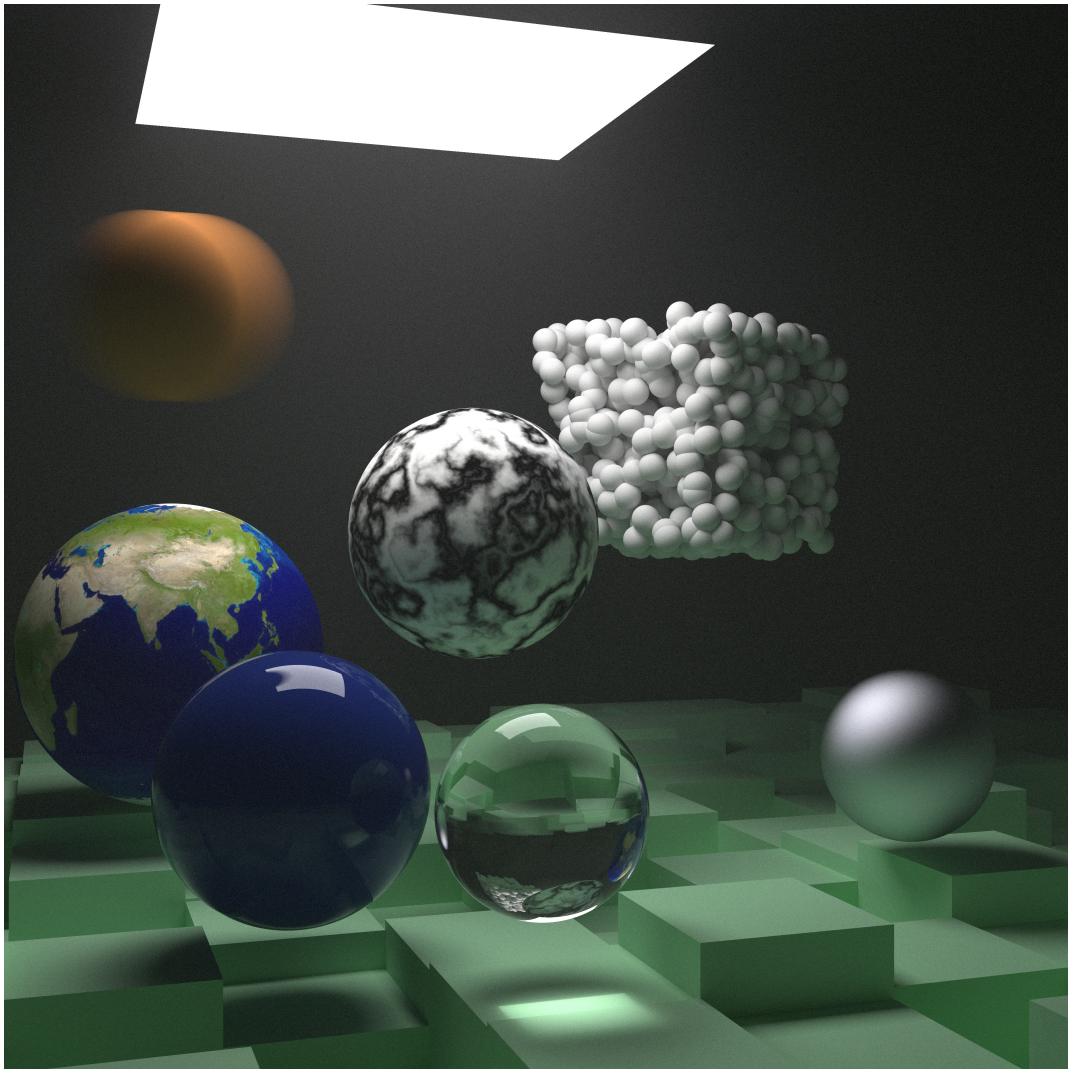

Figure 12: Book2 Final

This scene is a modified version of the one used by Peter Shirley in Ray Tracing Mini-Books 2. It combines may techniques. The whole scene is filled with thin smoke, rendered using participating media. The orange ball on the top left shows motion blur. The earth ball shows image textures. The marble ball in the middle is a complex example of procedural texture computed using Perlin noise. The box on the top right contains 10000 balls as components and use BVH to accelerate.

2.11 Test Obj

Figure 13: Test Obj

A simple scene used in obj test. This Cornell Box is made of triangles. It is stored in a .obj file (with material description in .mtl file) and imported using Tinyobjloader.

2.12 Sponza Sun

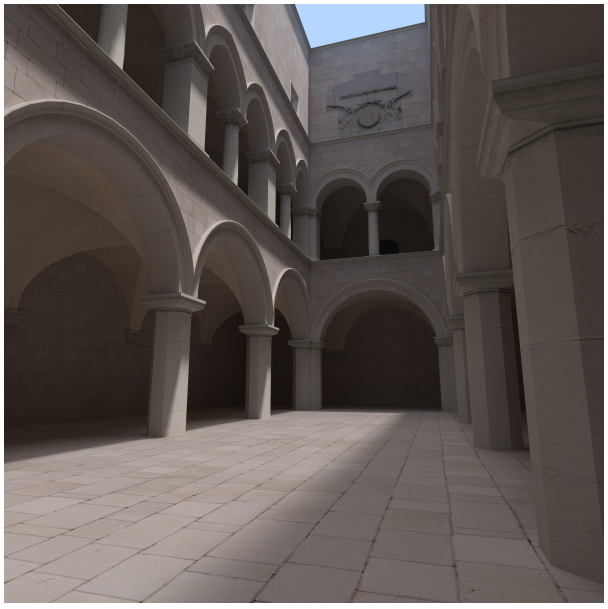

Figure 14: Sponza Sun

This scene is a more complicated one with image textures. Note that in this scene, a biased sampling technique is used such that the sun gets more samples.

2.13 Sponza Crytek Series

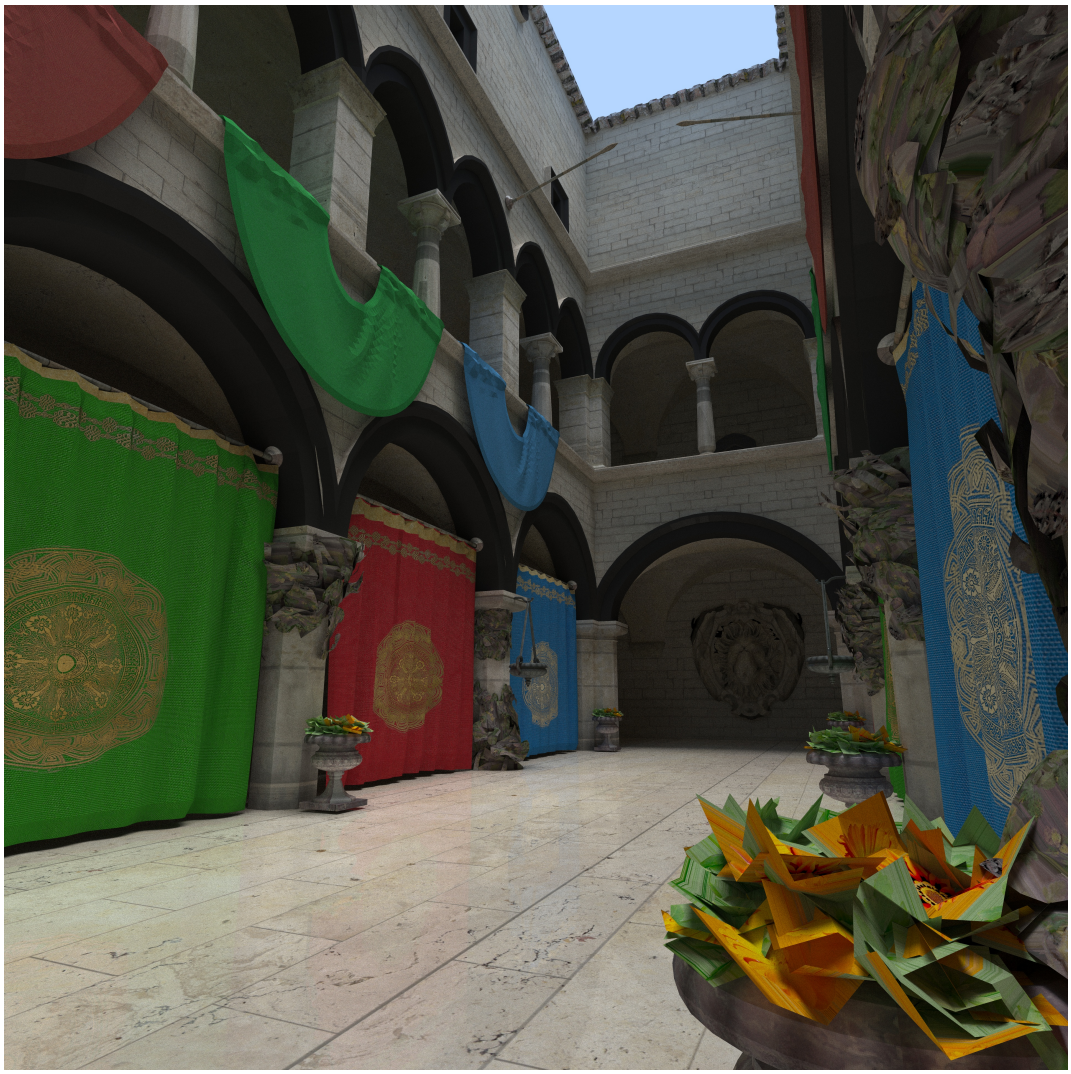

Figure 15: Sponza Crytek Cloudy

This scene is a remastered version of the last one from Crytek cooperation. It is rendered unbiasedly with Path Tracing. Note that bump map is enabled to add more details to the geometry (like the lion in the back). The following one is the same scene with a more complicated skybox. It can be used to simulate extremely sunny weather.

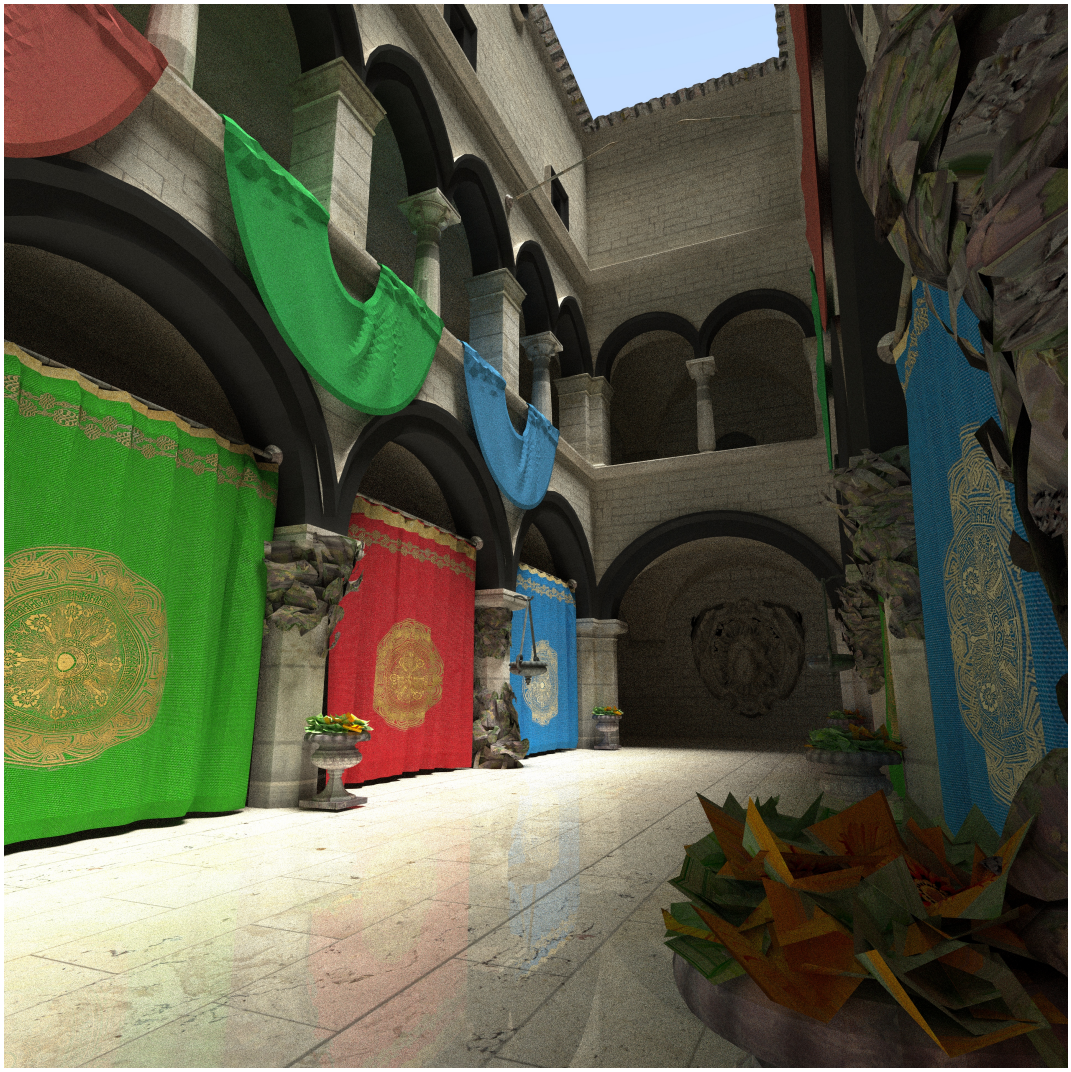

Figure 16: Sponza Crytek Sunny

2.14 Photon Mapping Series

A bunny rendered using Photon Mapping. When rendering the first image, photon tracing depth is set to one so that the caustic effect can be seen clearly. The correct effect is shown in the second image.

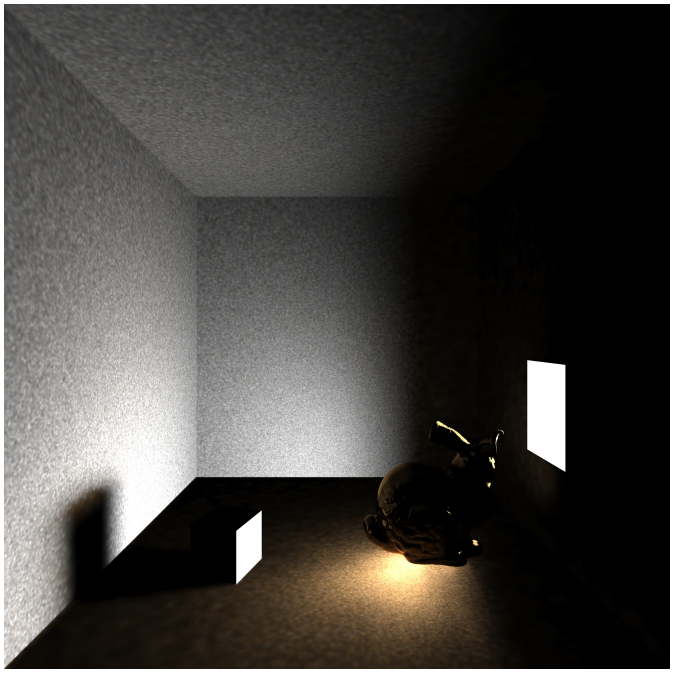

Figure 17: Photon Mapping Bunny Caustic

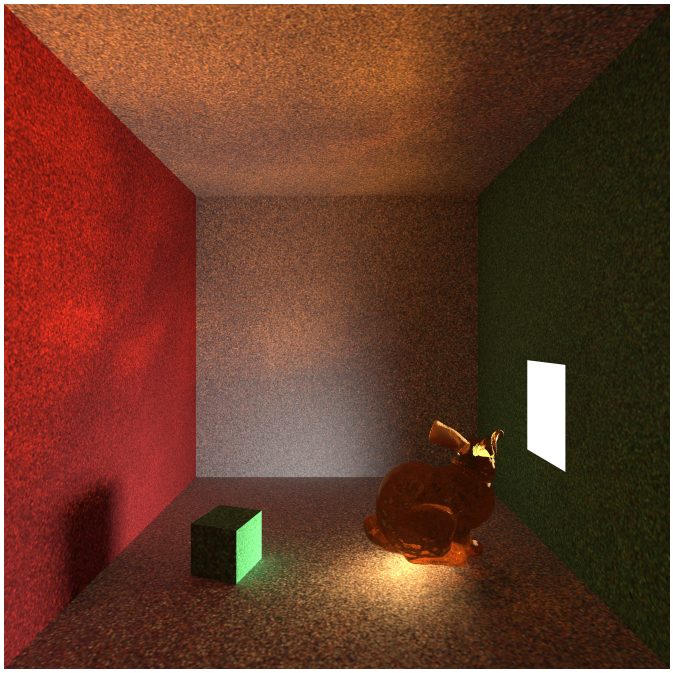

Figure 18: Photon Mapping Bunny

3 Usage

How set scene before compilation

step 1: set image settings (customization (e.g. changing aspect_ratio to 16.0 / 9.0) can be done by modifying render_scene in main.cpp)
step 2: choose integrator (Photon Mapping only supports one scene, Path Tracing has step 3 of setting scene)
step 3: set scene, camera and skybox (scene functions can be found in src/scenes)

How to run this renderer

Windows:
step 1: compile with CLion using MSVC compiler
step 2: run "Project.exe 8", here 8 means using 8 threads (after running we will get 8 .partial files)
step 3: run "packager.exe 8", here 8 means combining 8 partial files (after running we will get a .ppm image, which can be opened using OpenSeeIt)
step 4: run python convert.py, which converts .ppm into .jpg (note that opencv for python is required to run this)
Linux:

run "bash linux_run.sh", which compiles the renderer, uses 80 processes to run it and process result into a .ppm and a .jpg file. (Note that for default Sponza Crytek scene, about 150 GB memory is required. This requirement is proportional to the number of processes.)

4 Renderer Function Menu

The basic functionalities of this renderer are listed in this section. Detailed description with code and explanation can be found in the following chapters.

4.1 Core Renderer:

Integrator: Monte Carlo Path Tracing, Photon Mapping.
Accelerator: BVH (AABB with SAH), Kd-Tree.
Hardware Acceleration: OpenMP on Windows, Multi-processing on Linux.

4.2 Objects:

Hittable Objects: Triangle, Triangular Mesh, Box, Sphere.
Complex Models: .obj Model with .mtl Material Description.
Skybox: Constant Skybox, Directional Skybox, Realistic Skybox, Multi-layer Skybox.
Transforms: Rotation, Translation.

4.3 Materials & Textures

Materials: Lambertian, Metal, Dielectric, PBR material, Isotropic, Diffuse Light.
Textures: Color Texture, Checker Texture, Perlin Noise Texture, Marble Texture, Image Texture, Bump Texture.

4.4 Visual Effects

Camera Effects: Off-focus Blur, Motion Blur.
Volumetric Rendering: Participating Media.

Other Visual Effects: Gamma Correction, Caustic.

5 Renderer Infrastructures

5.1 Math Utilities

5.2 Basic Datatypes

5.3 External Libraries

6 References

Code References:

[**1**] Ray Tracing Mini-books, by Peter Shirley (https://raytracing.github.io)

[**2**] Physically Based Rendering: From Theory to Implementation, by Matt Pharr, Wenzel Jakob, and Greg Humphreys (https://github.com/mmp/pbrt-v3)

[**3**] Dezeming Family (https://dezeming.top/)

External Libraries:

[**1**] Tinyobjloader (https://github.com/tinyobjloader/tinyobjloader)

[**2**] stb_image (https://github.com/nothings/stb)

Model Resources:

[**1**] Morgan McGuire, Computer Graphics Archive, July 2017 (https://casual-effects.com/data)