CPSC314
Assignment 4: Raytracing

Due 23:59:59, Friday, Dec 04, 2020

0 Plagiarism Policy

By handing this assignment in, you agree to the course plagiarism policy as stated here: https://www.students.cs.ubc.ca/~cs-314/Vjan2013/cheat.html. Please read it, and make sure you understand it.

1 First Part (50 pts)

In the first part of this assignment, you will implement a simple ray tracer following Peter Shirley's book: "Ray Tracing in One Weekend". The book will guide you through creating a fully functional raytracer with the following features:

- Basic ray casting
- Positionable camera
- Sphere intersections
- Antialiasing
- Lambertian Surfaces
- Reflection
- Refraction
- Depth of Field

We encourage you to write all the code. At the face to face grading, we will assume that you understand every part of it. However, if you are struggling with bugs, the code is available here: https://github.com/RayTracing/raytracing.github.io/tree/master/src/common So you can compare it with yours.

The book is available here: https://www.realtimerendering.com/raytracing/Ray%20Tracing%20in%20a%20Weekend.pdf
More information: https://in1weekend.blogspot.com/2016/01/ray-tracing-in-one-weekend.html

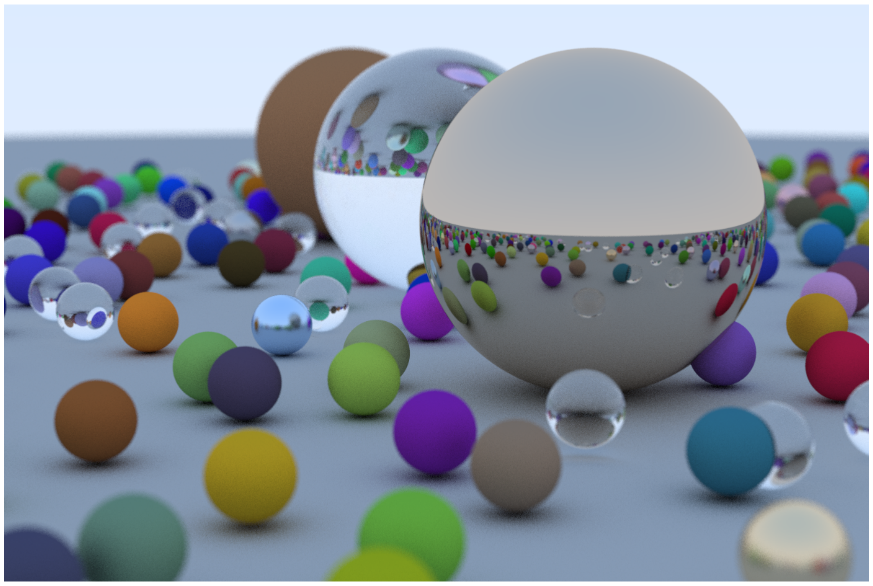

Figure 1: Rendering example for the first part. (Peter Shirley 2018)

2 Second Part (50 pts)

In the second part of the assignment, you will need to extend the raytracer to account for the following features:

- Triangles (15 pts.): You will need to implement ray-triangle intersections and create a ”cube” class so you can render at least a rotated box in your scene
- Shadows: As defined in our lecture (raytraced shadows) (15 pts)
- Point lights (5 pts)
- Directional lights (5 pts)
- Blinn Materials (10 pts.)

3 Bonus Part (8 pts)

For the bonus part, you could implement at least two of the following features:

- Soft shadows
- Surface patch (Based on Bézier splines)
- Adding cones
- Adding torus
- Bump mapping
- Accelerated raytracing with space partition algorithms
- GPU raytracing (here is a useful link: https://developer.nvidia.com/blog/accelerated-ray-tracing-cuda/)

If you don't feel confident enough with C++, you might implement your raytracer in another language. However, you need to understand that almost all raytracers are written in C/C++ due to performance requirements. If you use Python or JavaScript, you will spend way more time waiting for your renders (Way more). Even in C++, a full-featured render could take more than 30 minutes on a good computer. It could also be more difficult because you will not have any reference code to compare. But it is up to you. We will not penalize the use of another programming language.

4 Hand-in Instructions

You do not have to hand in any printed code. Create a README.txt file that includes your name, student number, and login ID, and any information you would like to pass on to the marker. Create a folder called "a3" under your "cs314" directory. Put all the source files and your README.txt file in there. Do not use further sub-directories. The assignment should be handed in with the exact command:

```
handin cs314 a4
```

Alternatively, you can use the web interface (https://my.cs.ubc.ca/docs/hand-in), which does exactly the same thing. Log in with your CWL credentials, write "cs-314" for the course, "A4" for the assignment name, and submit your zipped assignment folder.

It is always in your best interest to make sure your assignment was successfully handed in. To do this, you may either use the `check submissions` button in the online handin, or using the -c flag on the command line:

```
handin -c cs314 a4
```