Reinforcement Light Rays Path-Tracer Progress Report

Callum Pearce

cp15571@my.bristol.ac.uk

***Abstract*—This document contains my weekly progress of my thesis as part of my 4th year masters unit COMSM0111 in 2018/19. It is mainly useful to myself for reflecting on the decisions I have made throughout the project and why I chose them. However, I also hope it gives a detailed overview of how the project evolved with time for any other reader.**

I. Introduction

Each section of this document describes a single week of work. For each section I have decided to include the following break down:

- **Goals**: A few set goals I aimed to achieve in that week and the motivation behind them.
- **Research/Implementation Details**: What was done, and how it was achieved.
- **Resources**: Describes what resources were notably helpful during the week for research/implementation details.
- **Reflection**: What I believed went well in the week, what to avoid in the future, and were the goal outlined achieved? Finally, if necessary, what has changed for the project as a whole?

Week 1

Building off dome preliminary research I decided I had to build a ray-tracer in order to start any work with my project. This would be a good way of refreshing the basics of computer graphics.

A. Goals

1) Build a basic ray-tracer from scratch using only SDL and GLM as external libraries

B. Research/Implementation Goals

This week was fairly simple in terms of implementation. I mainly based my work on the ray-tracer I built with my project partner in my 3rd year of university. I used a similar project structure and followed the lab-sheets from `COMS30015` by Carl Henrik Ek. By the end of the week I had built a ray-tracer which simulated the following in real-time:

- Constructing surfaces from triangle primitives and projecting them onto a 2D pixel plane for camera viewing via ray-tracing.
- Supports camera movement
- Direct illumination

C. Resources

As mentioned the main resources used were those provided for `COMS30015` by Carl Henrik Ek in 2017. They gave me a good refresher on all the core concepts of ray-tracing and the mathematics behind it. I have based my rendering on the Cornell Box scene which is a classical scene for testing computer graphical renderings.

D. Reflection

I met my goal this week and built a well designed code-base to go with it.

Week 2

With a basic ray-tracer up and running, it was then time to add global illumination (indirect lighting) into the rendered scene and other features to make the basic ray-tracer complete for my purposes.

E. Goals

1) Implement Monte-Carlo global illumination within the ray-tracing pipeline (to go alongside direct light. It is Monte-Carlo global illumination I am planning to base my reinforcement learning technique presented by NVIDIA [1] on.
2) Create an object loader for the scene to test rendering in different scenes. I need a scene which has very low light levels in certain parts of the scene in order to show how reinforcement learning reduces noise in these areas.

F. Research/Implementation Goals

My implementation worked as follows; for every pixel in the image, calculate the colour by finding the direct light at that point combined with the indirect light. Where indirect light was sampled by shooting a ray into the scene and scattering it, then recursively finding the illumination at these points (via Monte Carlo).

For the object loader, I read in all vertices and then built triangles to form the defined surfaces in the file by using fan-triangulation. These triangles would be built into the scene by a call to the script with the file name of the `.obj` file to load.

I also introduced `openmp` to the project to speed things up by parallelising the ray-tracers pixel painting loop.

G. Resources

ScratchPixel 2.0 [3] provided an excellent description of Monte Carlo global illumination for a ray-tracer. As for the object loader, opengl gave a great tutorial for simple processing of `.obj` files [2].

H. Reflection

Monte Carlo global illumination was available for a custom scene as I had set out to achieve in this week. I also introduced an object loader which allows me to load in a scene of my choice, which will become especially useful when comparing different methods later on in the coursework. However, due to the introduction of global illumination the ray-tracer is far slower and takes a significant amount of time (nearly a day) to render a high quality image.

WEEK 3

I. Goals

1)

J. Research/Implementation Goals

K. Resources

L. Reflection

WEEK 4

M. Goals

1)

N. Research/Implementation Goals

O. Resources

P. Reflection

REFERENCES

[1] Ken Dahm and Alexander Keller. Learning light transport the reinforced way. *arXiv preprint arXiv:1701.07403*, 2017.

[2] opengl. Model loading. http://www.opengl-tutorial.org/beginners-tutorials/tutorial-7-model-loading/.

[3] ScratchPixel. Scratchpixel2.0. http://www.scratchapixel.com/.