CM30075 Advanced Computer Graphics: Raytracer Report

Candidate 21532

December 2022

Contents

1 Overview

1.1 Ray tracing

Ray tracing is a method used in computer graphics to simulate light moving through a three-dimensional scene. This report accompanies a piece of software that uses ray tracing to produce an image using the famous Utah teapot model.

1.2 Language and Libraries

Rust The software is written in Rust []. Rust was chosen for this project because of its speed and support for multi-threading. It is also fairly similar to the more commonly-used C++, making the translation of starter code fairly simple.

Glam Glam is a library that provides types and methods for vectors, one of the foundations of graphics software. There are also types that are safe to use in parallel, perfect for ray tracing, and a type for a 3D transformations.

png The software writes the pixel data produced into a .png image file. The png library allows this, and is easier than writing to a .ppm file and then converting it manually, despite the simplicity of the .ppm format. The library also handles the complexity of properly adjusting the colours if the image produced is too bright or dark.

Acap "As close as possible", abbreviated to acap, is a library that provides a number of different data structures which support search by proximity, including a KD tree. The library also includes methods for balancing and searching automatically.

1.3 Structure

The program is separated into multiple files, grouped by the types and traits that they contain. For types that implement a trait, their files are arranges into a folder names after the trait and the trait definition is in a file directly in the "src" folder. For example, the trait "Object" is defined in 'src/object.rs' and the type "Sphere" that implements Object is defined in 'src/object/sphere.rs'. Some files, like 'src/photonmap.rs', define multiple related types.

To run the program, you must have Rust installed: go to rustup.rs/ to do so. Then the command 'cargo run –package raytracer –release' may be used. The program will produce .png files as output in the root folder of the project.

The file 'src/main.rs' contains the main function, which initialises the frame buffer, scene and camera before rendering the scene and photon map. Finally, it calls the buffer to write its data to a file. This file also contains the function that creates the scene by specifying objects and their materials, as well as lights.

2 Basic Raytracer

2.1 Camera

In order to produce an image of a scene, a camera is used. The camera may be moved and rotated, its aperture adjusted, the number of samples per pixel altered, and an image size given. The camera position serves as the origin point for the rays that are fired through the image plane and into the scene. Each ray returns a colour which contributes to the final colour of the corresponding pixel it passes through.

Multisampling To create a more accurate image, multiple rays (samples) are fired through each pixel, and they all contribute equally to the final colour. The rays are fired with a small element of randomness to prevent them all taking the same path. Multisampling also allows the number of rays in the scene to be controlled. Although there are a greater number of rays per pixel, rays can be reflected or refracted (Section 3.1) stochastically (based on probability) rather than "branching", which generates multiple new rays from an intersection. Generating multiple new rays can result in a much greater quantity of them in the scene, slowing down the runtime considerably.

In addition, this technique allows for depth of field in the image (Section 3.3).

Multithreading

2.2 Objects

2.3 Lighting

3 Feature-Rich Rendering

3.1 Reflection and Refraction

3.2 CSG and Quadratic Surfaces

3.3 Depth of Field

4 Photon Mapping

4.1 Theory

4.2 Photon Map

4.3 Materials