TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

An Interactive Ray Tracing Program

Submitted By:
Amrit Prasad Phuyal (074BEX403 /PUL074BEX004)
Ashlesh Pandey (074BEX407)
Asim Maharjan(074BEX408)
Dipendra Raj Panta (074BEX411)

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL

March 5th, 2020

Abstract

This report describes an interactive ray tracer, a software through which the user can place objects and change their properties, and then finally generate high quality images using ray tracing. The front-end UI was created using OpenGL, an application programming interface for rendering 2D and 3D graphics by utilizing the GPU and Dear ImGUI, an open source graphical user interface library that utilized OpenGL. We have used the modern core profile OpenGL version 4.3.

Acknowledgement

We would like to extend our deepest gratitude towards Dr. Basanta Joshi, lecturer of Computer Graphics for his encouragement and guidance in the completion of this case study. We acknowledge our deepest appreciation to the Department of Electronics and Computer Engineering, Pulchowk Campus, for the provision of the project demonstration for the subject.

Lastly, we extend our gratitude towards our family and all our classmates for their cooperation during this case study. We wouldn't be able to complete the project without their support.

Amrit Prasad Phuyal

Ashlesh Pandey

Asim Maharjan

Dipendra Raj Panta

Contents

1 Introduction

Ray tracing is a technique used to generate high quality realistic images using the physical properties of light. Although the ray tracer mechanism is a well-established method of rendering high quality images, the computation time limitation makes it an expensive choice.

With this program, people can generate complex images with ease as they will have full control over the lighting effects and object placement. This enables people with even minimal knowledge of graphics to render beautiful scenes. With the added benefit of using 3D models, the more experienced user can do a lot more.

2 Implementation Details

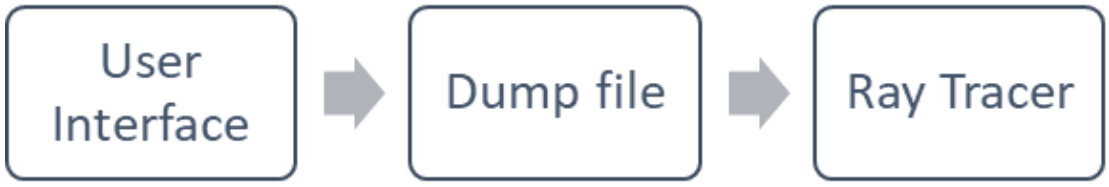


Figure 1: Block Diagram

Our software consists of two parts: the front-end UI created using OpenGL and Dear ImGUI. We create and modify the objects in the UI. We can also add extra lights and such. For lighting in the UI, we have employed Phong Shading. All the lights are approximated by a point light source. We use three different shaders for our purposes, one for rendering the grids, one for the lines, and one for rendering our objects in the world using the Phong lighting effect.

We have used C++ with OpenGL version 4.3 to create the UI. Once the user has created his world in the UI, they can create a file which can be input into the ray tracer to finally generate the image using ray tracing.

The ray tracing algorithm used can be summed up as follows:

1. For every pixel in the image, send multiple rays through each pixel
2. For each ray sent through that pixel, calculate the color of the ray
3. The color of the pixel is found by averaging the color of every ray through that pixel

For every pixel ‘p’ in the image:

Pixel_color = {0.0f, 0.0f, 0.0f};

For a fixed number of samples ‘n’:

Ray r = a random ray through the pixel

Pixel_color = pixel_color + get_ray_color(r);

pixel_color = pixel_color / n

The pseudo code for the get_ray_color is as follows:

get_ray_color(Ray r):

If r hits any object in the world:

Emitted_color = light emitted by the object

Ray r1 = ray in direction in which the material of the object scatters the light

Color1 = get_ray_color(r1)

Scatter_color = albedo value of material * Color1

Return Emitted_color + Scatter_color

Else

Return {0,0,0}

The albedo value of a material is simply the total fraction of light that is reflected by that object.

The direction of the ray r1 depends on the type of material it strikes. For e.g. for metallic materials the scattered direction is very likely to be close to the reflected direction but for diffuse (or matte) surfaces, the scattered direction is equally likely to be in every direction. Similarly, for glass, the incident ray gets refracted through the glass instead of being reflected back.

3 Objectives

The objectives of our program are as follows:

1. To learn about and implement modern rendering techniques using OpenGL
2. To implement theoretical knowledge of the subject in a proper program
3. To learn about the mechanisms of ray tracing and its uses and limitations

4 Features

The features of our program are as follows:

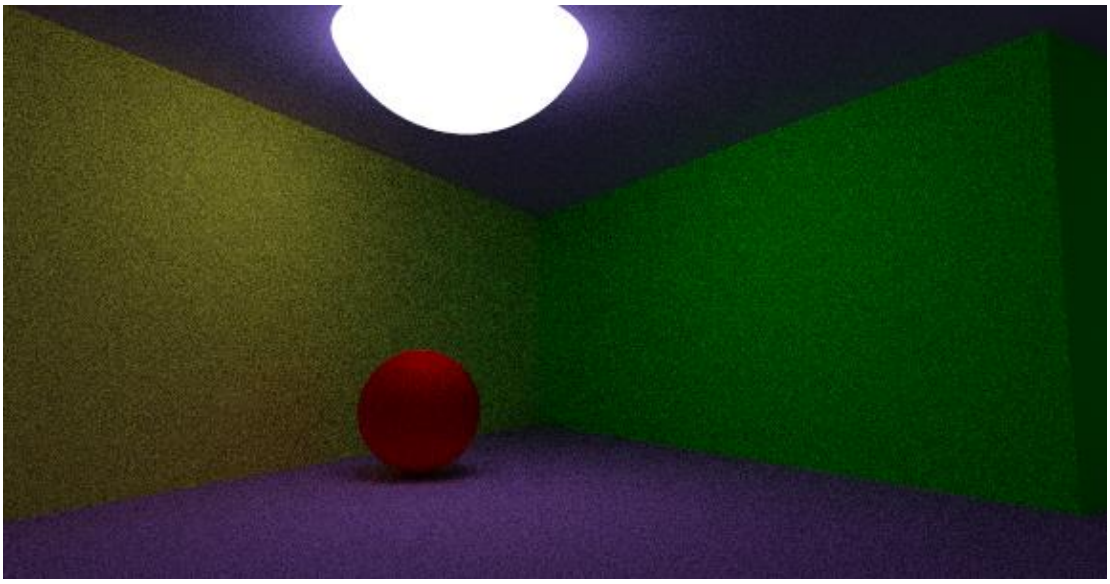
1. User can place and edit objects as they like
2. They can add as much light sources as they like
3. They can generate high quality images

5 Source Code

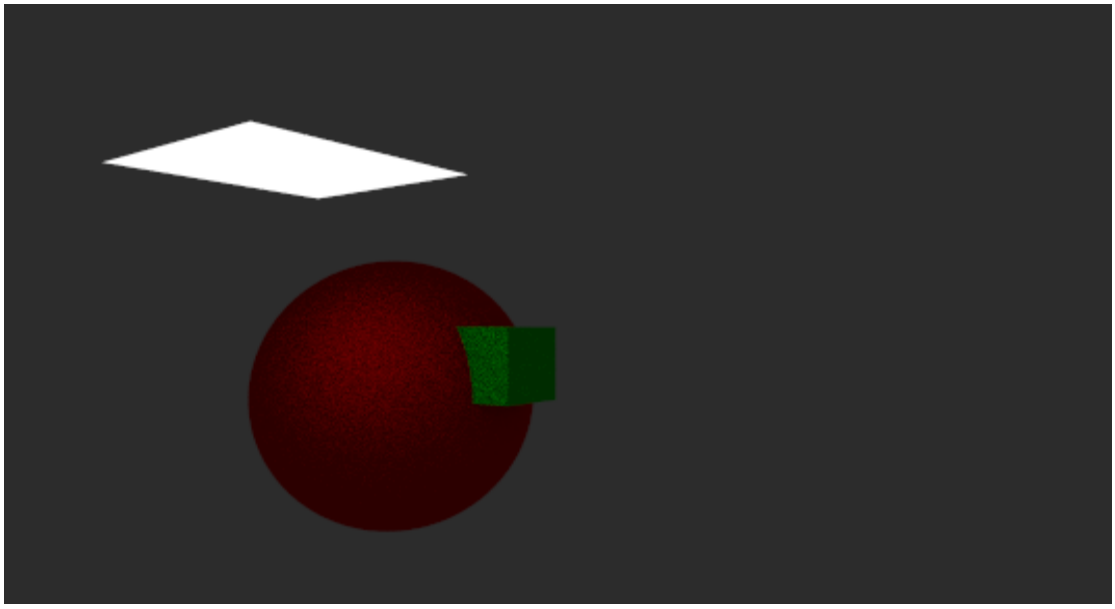
The source code to our project can be found at the github repository given by the link obtained after scanning the QR Code below:

6 Result

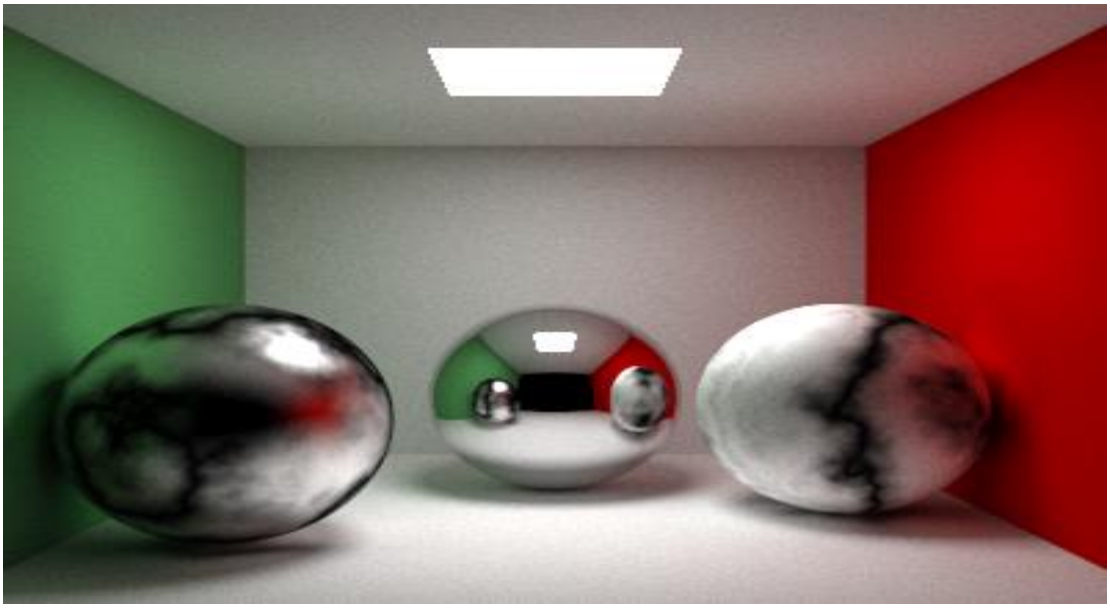
The following images are some of the output images collected during our project.



Output 1



Output 2



Output 3

More output images can be found at the same repository as that of the source code.

7 Conclusion

We were able to complete this project within the stipulated time frame. The concept of ray tracing is relatively new to us, and we are glad to have tried to implement that in our project.

8 References

1. M. Pharr, J. Wenzel, and G. Humphreys, *Physically based rendering: from theory to implementation*. Amsterdam: Morgan Kaufmann, 2017.
2. Ray Tracing in a Weekend series by Peter Shirley