Ray Tracing In Functional Programming Language

Dr. Babul Prasad Tewari
Department of
Computer Science and Engineering
GKCIET
Malda, India
babul@gkciet.ac.in

Aditya Raj
Department of
Computer Science and Engineering
GKCIET
Malda, India
induaditya214@gmail.com

3^{rd} Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—to do**

***Index Terms*—ray tracing, three-dimensional graphics, functional programming.**

I. Introduction

There are two approaches to rendering a three-dimensional scene: object-order rendering and image-order rendering.

Object-order rendering involves iterating over each object and figuring out which set of pixels it maps to and coloring those pixels. In image-order rendering, for each pixel, the nearest visible object is found, and that pixel is colored using the given shading model. The former is classified into rasterization algorithms (or projective algorithms), and the latter is termed the ray tracing algorithm (image-space algorithm). Same effects, like reflections and shadows, are easy to implement in ray tracing compared to rasterization.

In ray tracing, image synthesis happens by shooting rays that go through the projection plane, and the nearest intersection with an object is found. If the object is reflective, rays bounce off it and hit other objects, which also influences color according to the shading model. In the current implementation, we have utilized the simplest shading model called Blinn-Phong shading and used the simplest three-dimensional geometrical object, a sphere, to illustrate ray tracing in the functional programming paradigm. To this end, we decided to use OCaml for its strong support for functional programming and its mechanisms to write imperative code as well, which will enable us to demonstrate alternative ways in the same programming language.

II. Methodlogy

A. Blinn-Phong Shading Model

B. Equations

to do

$$a + b = \gamma \tag{1}$$

"Fig. 1"

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In

Algorithm 1 Ray Tracing Algorithm

```
Initialize scene with objects and lights
Set canvas resolution (width, height)
For each pixel (x, y) in the canvas:
    Generate ray from camera through pixel (x, y)
    Initialize color to background color
    Initialize closest intersection distance to infinity
    For each object in the scene:
        If ray intersects object:
            Calculate intersection point
            Calculate normal at intersection point
            Calculate distance to intersection
            If distance is less than closest intersection distance:
                Update closest intersection distance
                Calculate color at intersection point
                Update color based on lighting
    Set pixel color to calculated color
Render the image
```

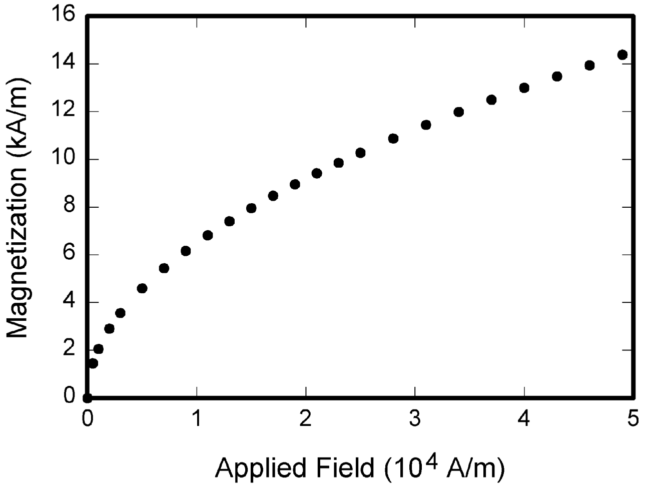

Fig. 1. Example of a figure caption.

the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

References

[1] A. Glassner, “An Introduction to Ray Tracing,” Morgan Kaufmann, 1989.

[2] E. Lengyel, “Mathematics for 3D game programming and computer graphics,” Cengage Course Technology PTR, 2019.

[3] R. Hartley, A. Zisserman, “Multiple view geometry in computer vision,” Cambridge University Press, 2004.

[4] K. Suffern, “Ray tracing from the ground up,” CRC Press, 2007.

[5] P. Shirley, R. K. Moreley, “Realistic ray tracing,” A K Peters Ltd, 2003.

[6] E. Lengyel, “Foundations of game engine development volume 2: Rendering,” Terathon Software LLC, 2019.

[7] S. Marschner, P. Shirley, “Fundamentals of computer graphics,” CRC Press, 2016.

[8] S. J. Gortler, “Foundations of 3D computer graphics,” MIT Press, 2012.

[9] M. Ashikmin, P. Shirley, “An anisotropic Phong light reflection model,” Technical Report UUCS-00-014, Computer Science Department, University of Utah, 2000.

[10] J. F. Blinn, “Models of light reflection for computer synthesized pictures,” Proceedings of SIGGRAPH 77, Computer Graphics, Vol. 11, No. 2, pp. 192–198, 1977.

[11] M. F. Cohen, J. R. Wallace, “Radiosity and realistic image synthesis,” Academic Press, San Diego, CA, 1993.

[12] R. L. Cook, K. E. Torrance, “A reflection model for computer graphics,” ACM Transactions on Graphics, Vol. 1, No. 1, pp. 7–24, 1982.

[13] R. L. McFall,and C. A. Cusack , Ray tracing as an object-oriented example for CS 1, The Journal of Computing Sciences in Colleges, 2010, pp.77–84.

[14] M. Pharr and G. Humphreys, “Physically Based Rendering: From Theory to Implementation,” Morgan Kaufman, 2004.

[15] A. Raj, “Very basic ray tracer,” 2025, gitHub repository. [Online]. Available: https://github.com/induaditya3/computer_graphics_ocaml/tree/main/ray_tracer

[16] T. Whitted, “An improved illumination model for shaded display,” Communi- cations of the ACM 23, June 1980.

[17] J. F. Hughes, A. van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner, K. Akeley, “Computer graphics: Principles and practice,” Addison-Wesley, 2013.