Raytracer Performance Optimization Report

Dawood Kingafg6
University Name
Course Name

April 28, 2025

Abstract

This report documents the profiling, analysis, and optimization of the raytracer project based on the trayracer codebase. The goal was to improve performance through memory optimizations, multithreading, and static analysis.

Contents

1 Introduction

Briefly describe the goal of the assignment: - Render a single frame using a CPU-based raytracer. - Measure and optimize performance. - Implement multithreading. - Document all improvements and results.

2 Project Setup

Explain: - Initial project cloning. - CMake and Ninja build system. - Basic benchmark program structure. - Compilation environment (Linux, GCC version, etc.)

3 Benchmarking

Describe how the benchmark was tested: - Command-line parameters: resolution, rays per pixel, number of spheres. - Initial performance results (time taken, MRays/s).

4 Profiling and Analysis

4.1 Tools Used

List the tools you used: - `perf` - `valgrind` (massif, cachegrind) - (Any other profilers if used)

4.2 Findings

Show initial bottlenecks: - High CPU usage points - Memory bottlenecks - Expensive function calls

5 Optimizations

For each optimization:

- Explain what was optimized
- How it was optimized
- Before/After performance numbers

Example subsections:

5.1 Memory Access Optimization

5.2 Math Function Optimization

5.3 Multithreading

6 Multithreading Implementation

Describe: - How rendering was parallelized. - Number of threads. - Synchronization methods (mutexes, locks, etc.) - Final multithreaded performance results.

7 Results

- Table comparing initial vs optimized performance
- Final MRays/s achieved
- Graphs or charts (optional)

8 Conclusion

Summarize: - Overall performance gain - Lessons learned - Possible future improvements

9 References

List any online resources, articles, or books you used.