Rasterization vs. Ray Tracing: Shadow Quality and Performance Analysis

Frankie Nieves Avilés
University of Puerto Rico
Puerto Rico, USA
frankie.nieves@upr.edu

Abstract

This project presents a quantitative and visual comparison between rasterization and ray tracing, with a specific focus on shadow accuracy and performance. While rasterization is known for real-time efficiency, ray tracing offers physically accurate illumination and shadows. Using custom C++ implementations of both methods, we evaluate shadow coverage, shadow pixel detection, rendering speed, and multi-light behavior across several controlled test scenes. Numerical metrics reveal where rasterization approximations fall short and where ray tracing excels. Results show that rasterization can approximate many shadow behaviors efficiently, but ray tracing remains unmatched in realism and subtle light interactions.

Keywords

Ray Tracing, Rasterization, Shadows, Rendering, Computer Graphics, Performance Analysis

ACM Reference Format:
Frankie Nieves Avilés. 2025. Rasterization vs. Ray Tracing: Shadow Quality and Performance Analysis. In *SIGGRAPH Student Project*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/0000000.0000000

1 Introduction

Ray tracing and rasterization are two foundational techniques in computer graphics. Rasterization has historically dominated real-time applications because of its exceptional speed and hardware optimization. Conversely, ray tracing provides physically accurate lighting and shadow behavior by simulating the trajectories of light rays.

The goal of this project is to measure how closely a rasterizer can approximate ray-traced shadows while maintaining high performance. By comparing both methods numerically—specifically through shadow pixel counts, shadow area ratios, and rendering speed—this should provide a structured and empirical evaluation useful for students, developers, and graphics researchers.

SIGGRAPH, University of Puerto Rico Mayagüez

ACM ISBN 000-0-0000-0000-0
https://doi.org/10.1145/0000000.0000000

2 Background

2.1 Rasterization

Rasterization converts 3D objects into 2D fragments efficiently. It has been the backbone of interactive 3D graphics for over 40 years, powering games, simulations, and films. While fast, its shadows often rely on approximations such as shadow maps or depth sampling.

2.2 Ray Tracing

Ray tracing traces light rays from the camera or surface points to light sources. Introduced by Arthur Appel (1968) and popularized by Whitted (1980), it produces highly realistic shadows, reflections, and refractions. Historically too slow for real-time applications, ray tracing has become practical only with modern GPU acceleration (e.g., NVIDIA RTX).

2.3 Why Compare Them?

Ray tracing generates superior shadows, but rasterization remains essential for real-time graphics. Understanding both their visual and numerical differences provides insight into where rasterization can approximate ray tracing effectively.

3 Related Work

Shadow computation methods have evolved from early ray-traced shadows introduced by Whitted to the widely used shadow maps by Williams (1978). Modern techniques include cascaded shadow maps, variance shadow maps, soft shadow filters, and hybrid ray-tracing/rasterization approaches seen in modern game engines.

This work builds on these foundations by offering a controlled, numerical comparison of shadow quality using custom rendering implementations.

4 Methodology

4.1 Why C++ Was Chosen

C++ provides:

- High performance required for pixel-by-pixel rendering.
- Full control over memory, buffers, and geometric data.
- Freedom to build custom shading, transformation, and shadow functions.
- Flexibility to dynamically move, rotate, and scale objects.

This control allowed the development of two parallel implementations: a rasterizer and a ray tracer that share camera, scene, and lighting parameters for fair comparison.

4.2 Shadow Metrics

Shadow Pixel Definition:

$$\text{ShadowPixel} = \begin{cases} 1 & \text{if brightness} < 0.3 \\ 0 & \text{otherwise} \end{cases}$$

Shadow Area Ratio:

$$\text{ShadowAreaRatio} = \frac{\text{Number of Shadow Pixels}}{\text{Total Pixels}} \times 100\%$$

Performance Metric: Pixels per second, measuring total render output rate.

4.3 Scenes and Cases

Four scenarios were tested:

(1) 3 spheres, 1 light
(2) 5 spheres, 1 light
(3) 5 spheres, 2 lights
(4) 1 sphere, 2 lights

Scenes were carefully aligned so that both rasterizer and ray tracer render identical geometry and lighting.

5 Results

5.1 Case 1: 3 Spheres

Rasterizer: 7.38% shadow area. Ray tracer: 14.09% shadow area. Ray tracing shows higher accuracy, capturing subtle occlusion.

5.2 Case 2: 5 Spheres

Rasterizer: 5.71%. Ray tracer: 3.82%. Rasterization slightly overestimates shadows due to approximation artifacts.

5.3 Case 3: 5 Spheres, 2 Lights

Rasterizer: 1.05%. Ray tracer: 5.00%. Multiple lights amplify ray tracing's advantage in accuracy.

5.4 Case 4: 1 Sphere, 2 Lights

Rasterizer: 0%. Ray tracer: 2.76%. Rasterizer fails to capture subtle shadows entirely.

6 Discussion

Results demonstrate clear trade-offs:

- Rasterization is consistently faster—by factors of 10–100×.
- Ray tracing produces more realistic and precise shadows.
- Approximation errors in rasterization become more pronounced with multiple light sources.

Shadow pixel metrics proved effective for quantifying differences in light occlusion.

6.1 Challenges

- Shadow function implementation required careful brightness thresholding.
- Keeping both renderers visually synchronized was difficult: small sphere or light movements drastically affected metrics.
- Multi-light scenarios added significant complexity to shadow evaluation.

7 Conclusion

This study shows that rasterization can approximate ray-traced shadows under certain conditions, but ray tracing remains significantly more accurate, especially in complex lighting. Rasterization still dominates real-time graphics due to its speed, but ray tracing offers unparalleled realism.

Future work may explore hybrid rendering systems that blend both methods or incorporate modern hardware-accelerated ray tracing.

Code Availability

The full source code, datasets, and reproducible results are available at: https://github.com/FrankieNieves/raytracing-vs-rasterization-shadows

References

[1] A. Appel. "Some Techniques for Shading Machine Renderings of Solids." *AFIPS*, 1968.
[2] T. Whitted. "An Improved Illumination Model for Shaded Display." *Communications of the ACM*, 1980.
[3] L. Williams. "Casting Curved Shadows on Curved Surfaces." *SIGGRAPH*, 1978.