

# Rasterization vs Ray Tracing: Shadow Quality and Performance Analysis

Frankie Nieves Aviles

COMP4046-096

December 12, 2025

# Presentation Overview

- 1 Project Motivation
- 2 Background
- 3 History of Rendering Techniques
- 4 History of Shadow Techniques
- 5 Why C++?
- 6 Understanding the Metrics
- 7 Math Behind Shadows
- 8 Case Studies
  - Case 1: 3 Spheres
  - Case 2: 5 Spheres
  - Case 3: 5 Spheres, 2 Lights
  - Case 4: 1 Sphere, 2 Lights
- 9 Observations
- 10 Project Hardships
- 11 Conclusion

# Project Motivation

Ray tracing is known for:

- Physically accurate shadows
- More realistic illumination
- Best visual quality

Rasterization is known for:

- Extreme speed
- Real-time performance
- Simplicity and efficiency

**Goal:** Compare them numerically and visually—especially their shadow behavior—and see whether a rasterizer can approximate ray-tracing shadows while staying fast.

# Rasterization vs. Ray Tracing: Background

## Rasterization

- Has dominated real-time graphics for 40+ years.
- Hardware accelerated since early GPUs.
- Extremely fast but shadows are approximations.

## Ray Tracing

- Developed in the 1960s, popularized in the 1980s.
- Simulates real rays of light: accurate shadows, reflections, refractions.
- Computationally expensive until recent RTX hardware.

## Why compare them?

- Ray tracing gives better shadows.
- Rasterizers are still used everywhere due to speed.
- Shadow differences can now be quantified numerically.

# History of Rendering Techniques

## Rasterization

- Rasterization remains the most cost-effective way to deliver interactive 3D graphics for gaming, film, and CAD, forming the backbone of modern graphics.
- Rasterizers replaced expensive vector displays on workstations, becoming standard in PCs and game consoles.
- Developed gradually through GPU evolution; no single creator, but widely credited to graphics hardware pioneers.

## Ray Tracing

- Concept introduced by **Arthur Appel (1968)** for image generation via ray casting.
- Popularized by **Turner Whitted (1980)** with recursive ray tracing including reflections, refractions, and shadows.
- Produces photorealistic images but historically slow until modern RTX GPUs.

# History of Shadows in Computer Graphics

## Ray-Traced Shadows (Shadow Rays)

- Introduced by **Turner Whitted (1980)**.
- Shoots rays from surface points to light sources to determine occlusion.
- Produces accurate hard and soft shadows.
- Enabled realistic reflections and refractions, revolutionizing offline rendering.
- Computationally expensive—limited early adoption to non-real-time rendering.

## Shadow Maps (Rasterization Shadows)

- Invented by **Lance Williams (1978)**.
- Renders scene from light's perspective to create a depth map.
- Efficient for real-time graphics, widely used in games.
- Handles many lights at once, but can have jagged or weird-looking shadows.
- Foundation for modern real-time shadows like cascaded shadow maps.



## Why C++?

- High performance for pixel-by-pixel rendering and heavy math.
- Full control over memory, buffers, and 3D scene data.
- Object-oriented design: organize cameras, lights, models, triangles, and instances clearly.
- Easy to create custom functions for transformations, shading, and shadows.
- Greater flexibility to move, rotate, and scale objects dynamically.
- Works well with math and image libraries.

*Short explanation: "C++ was chosen because it's fast, precise, and gives full control over 3D objects, transformations, and custom rendering functions."*

# Understanding the Metrics

## Shadow Pixels (brightness < threshold)

- A shadow pixel is any pixel where brightness is less than 0.3 (scale 0-1).
- Identifies which areas of the scene are in shadow versus lit.
- Counting shadow pixels allows us to measure the **shadow coverage** numerically.
- Lets us compare how much shadow each rendering method produces.

## Shadow Area Ratio

- Percent of total pixels classified as shadows.
- Determines how “dark” the scene is under each method.

## Pixels Per Second (Performance)

- Rendering speed measurement.
- Rasterizers excel here; ray tracing is slower per pixel.

# Shadow Metrics Formulas

## Shadow Pixel Determination:

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

## Shadow Area Ratio:

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

These formulas allow us to quantify shadow coverage numerically, giving an objective metric for comparison between rasterization and ray tracing.

# Basic Light/Shade Calculation

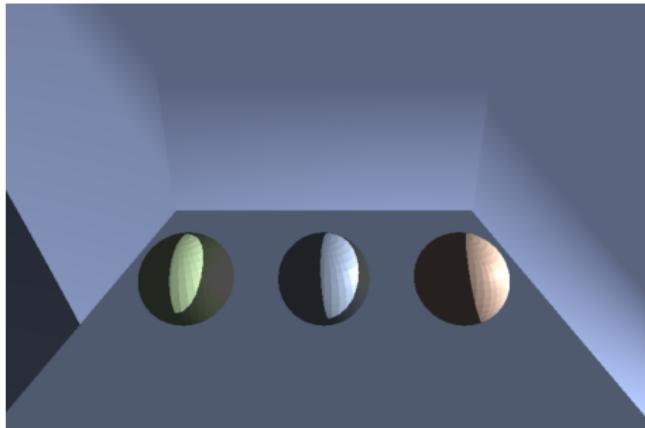
## Simplified Illumination Equation (Ray Tracing):

$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}}$$

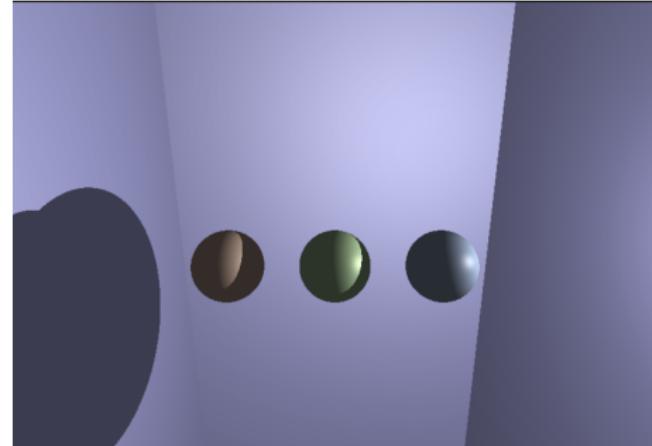
- $I_{\text{ambient}}$ : Base light in the scene.
- $I_{\text{diffuse}}$ : Light scattered from surfaces (depends on angle to light source).
- $I_{\text{specular}}$ : Highlights (mirror-like reflection from surfaces).

Rasterizers approximate this calculation per pixel quickly, while ray tracing simulates rays to capture full interactions of light and shadow.

## Case 1: 3 Spheres



Rasterizer



Ray Tracer

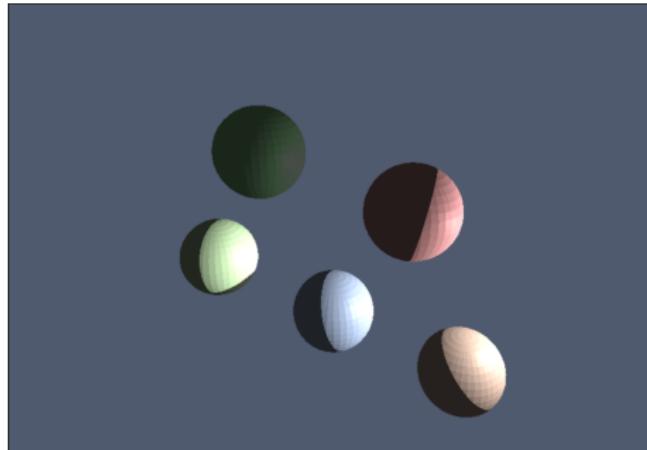
## Case 1 Metrics

Method	Shadow Pixels	Shadow Area (%)	Pixels per Second
Rasterizer	35423	7.37979	1.68533e+006
Ray Tracer	67640	14.0917	603861

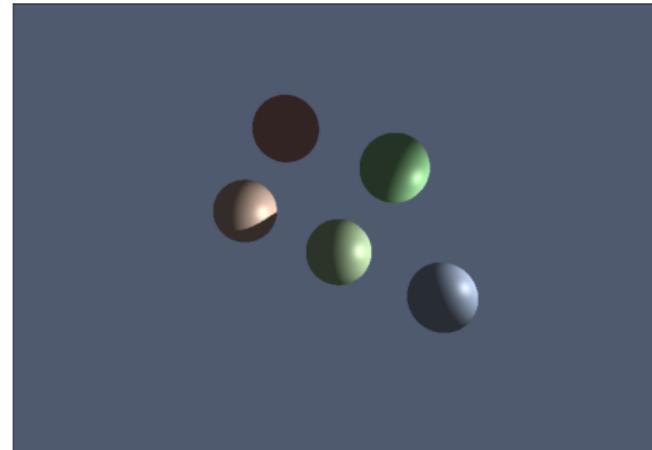
Explanation: Rasterization is very fast but underestimates shadow coverage because it approximates light occlusion. Ray tracing captures more accurate shadows at the cost of performance.

Note: In later cases, shadows on walls or floors are not included—the main focus is on shadows between objects themselves, not on surfaces. This is because ray tracing clearly excels at casting shadows on surfaces, and including them would overshadow any results rasterization would have.

## Case 2: 5 Spheres



Rasterizer



Ray Tracer

## Case 2 Metrics

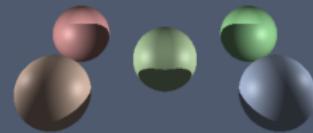
Method	Shadow Pixels	Shadow Area (%)	Pixels per Second
Rasterizer	27433	5.71521	1.51539e+007
Ray Tracer	18341	3.82104	1.75553e+006

Explanation: Rasterization remains faster and shows slightly higher shadow coverage due to approximate shadow projection, whereas ray tracing produces more precise shadow placement with fewer pixels affected by visual glitches.

## Case 3: 5 Spheres, 2 Lights



Rasterizer



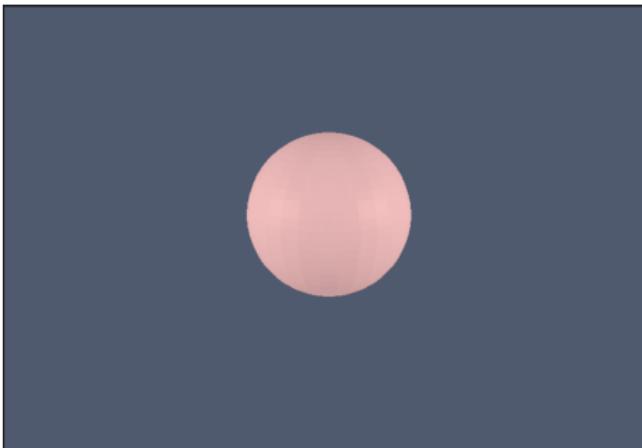
Ray Tracer

## Case 3 Metrics

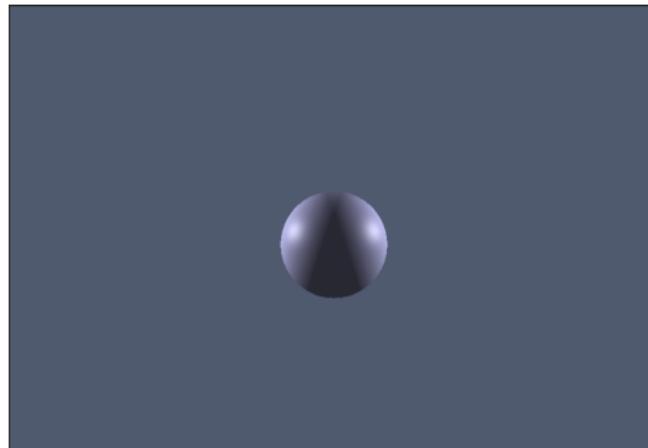
Method	Shadow Pixels	Shadow Area (%)	Pixels per Second
Rasterizer	5036	1.04917	8.79314e+006
Ray Tracer	24014	5.00292	291.391

Explanation: Multiple lights increase shadow complexity. Rasterization approximates shadows quickly but misses subtle overlaps, whereas ray tracing calculates full light interactions, giving more realistic shadows.

## Case 4: 1 Sphere, 2 Lights



Rasterizer



Ray Tracer

## Case 4 Metrics

Method	Shadow Pixels	Shadow Area (%)	Pixels per Second
Rasterizer	0	0	1.27775e+007
Ray Tracer	13268	2.76417	142.851

Explanation: Rasterizer shows no shadows here due to light placement and approximation limits, while ray tracing captures subtle shadows created by multiple light sources.

## Overall Observations

- Ray tracing consistently produces more accurate shadow detail.
- Rasterizer shadows sometimes miss subtle occlusion.
- Rasterization remains significantly faster.
- Culling optimization noticeably improved rasterizer speed.

# Project Hardships

- Implementing the shadow function was challenging:
  - Correctly identifying shadow pixels required careful thresholding.
  - Ensuring the shadow projection matched the light and object placement.
  - Adding a second light increased complexity, as multiple shadow directions had to be considered.
- Aligning the scenes for rasterizer and ray tracer for fair comparison:
  - Moving spheres and adjusting lights until both methods matched visually.
  - Small differences in placement could dramatically affect shadow coverage.
- Balancing performance and accuracy:
  - Optimizing rasterizer speed without losing shadow detail.
  - Ray tracer was slower but required verification that shadows were correct.

# Conclusion

## Takeaways:

- Ray tracing gives superior shadow quality — expected.
- Rasterization can come surprisingly close with smart optimizations.
- Numerical comparison allows objective evaluation of shadow quality.
- For real-time applications: rasterization still wins.
- For realism: ray tracing remains unmatched.

## The tests show that:

Rasterizer can get very close to ray tracer shadows while staying much faster, with some smart implementation it can either level the playing field or surpass in various areas.

# Thank You