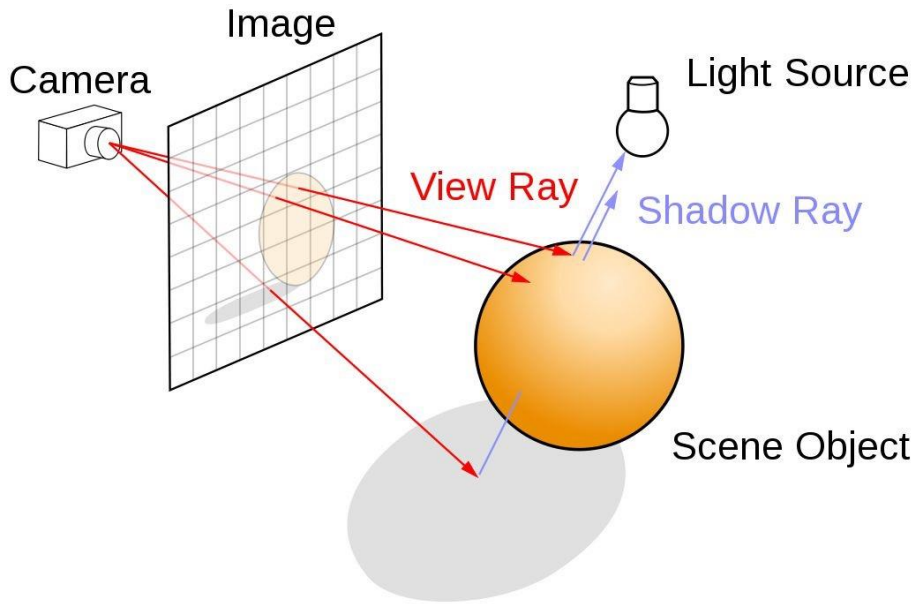
Ray Tracing

Joonas Praks

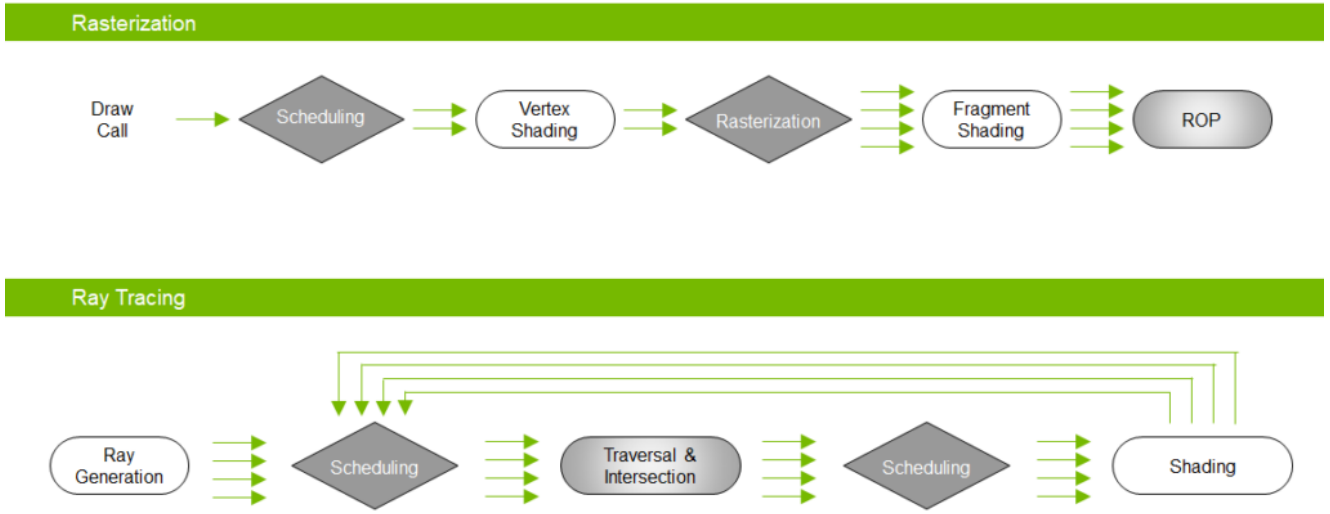
Introduction

- Rendering algorithm
- Let’s intersect the world with rays!



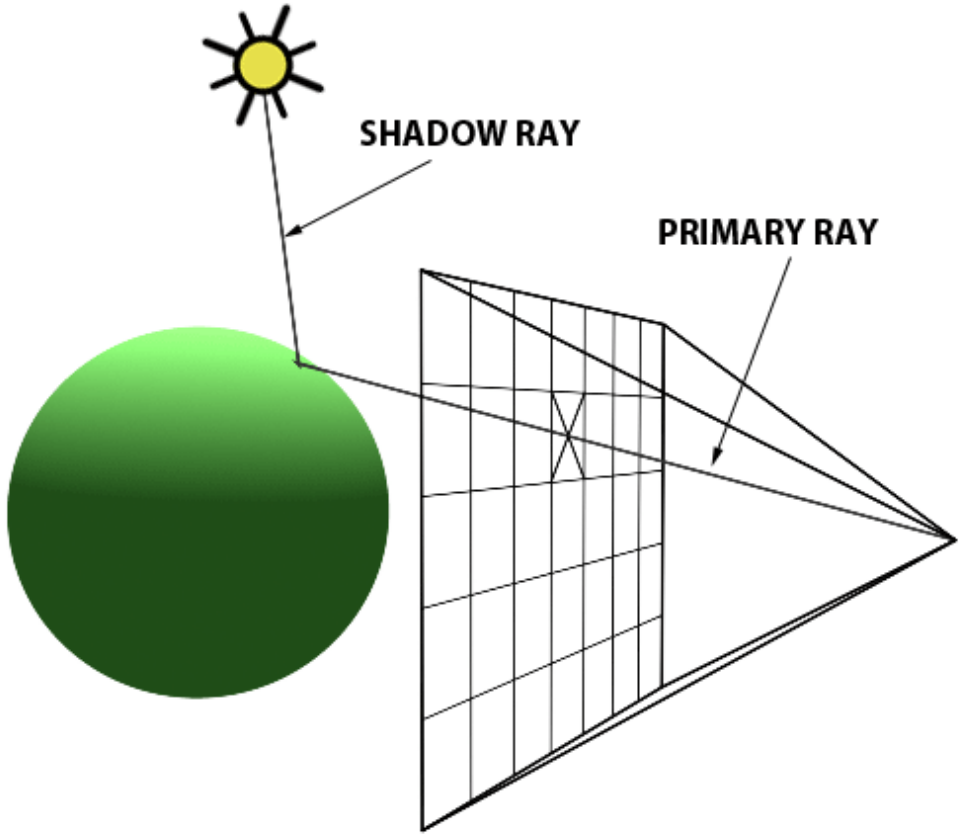
Introduction

- Reiterate the rasterization pipeline
- What are we missing?
  - Shadows
  - Reflections
  - Refractions



Implementation

- How do we calculate intersections?
- What happens if we don’t reach the light source?
- What happens if the material is transparent?



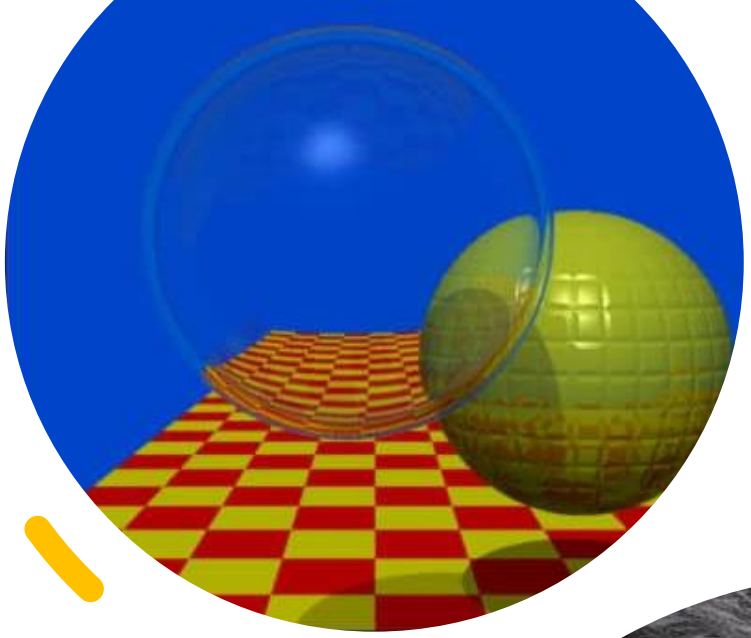
Why is this rendering so inefficient?

Comparing against rasterization

- Pipelines?
- Memory?
- Computational complexity?
- Code complexity?

Evolution

- Shadows - Appel
- Reflections - Whitted
- Refractions - Whitted
- Motion blur - Cook
- Depth of field - Cook
- Gloss - Cook
- Caustics?



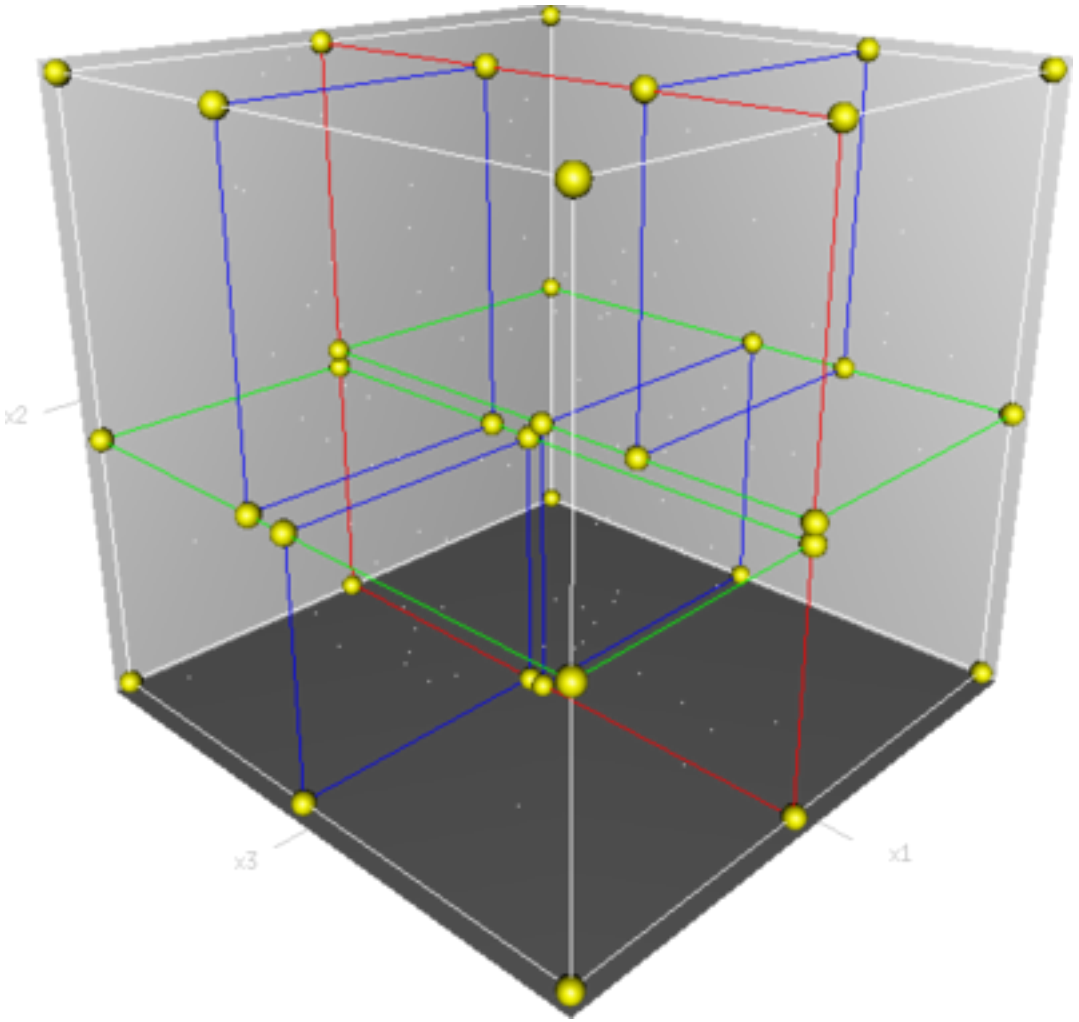
Can we mix the two?

High Definition Render Pipeline

- **Ray Traced Reflections**
- **Ray Traced Global Illumination**
- **Ray Traced Shadows**

Advancements

- Tracing from eye vs the light source
- Distributed Ray Tracing
- ray grouping (disney),
- Bounding Volume Hierarchy (BVH)
  - octrees->kdtrees,
- Denoising filtering



Sources

- https://youtu.be/uqJBw7hOLqw
- https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview
- https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/
- https://www.youtube.com/watch?v=gBPNO6ruevk
- https://www.youtube.com/watch?v=Ahp6LDQnK4Y

Thank you for listening!