Voxel Ray Tracing

Ethan Tucker
Advisor: Dr. Hsu

Presentation Outline

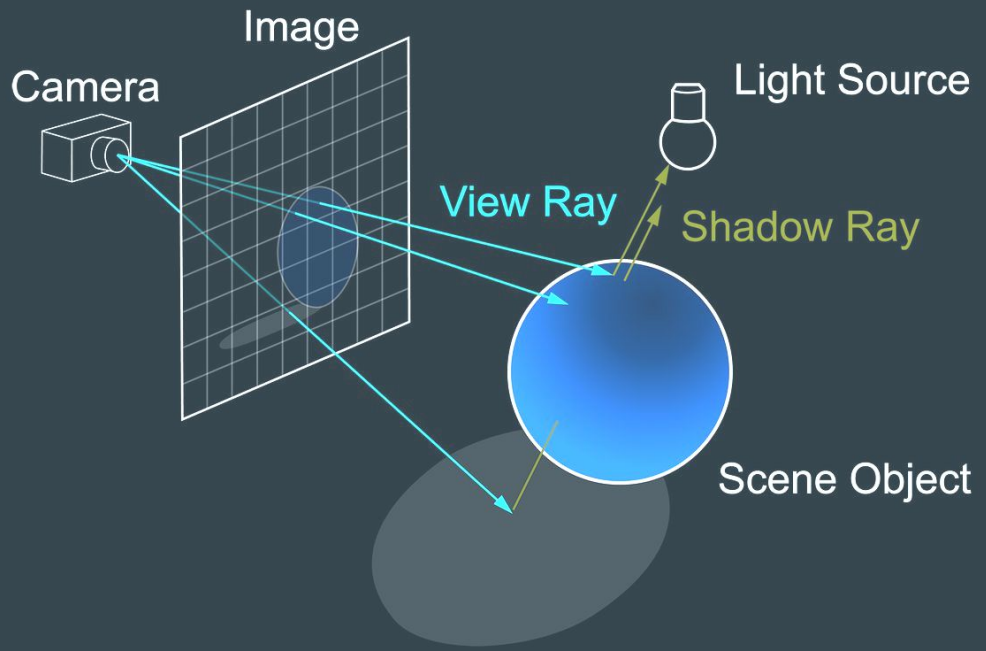
- Project Description
- Background Info
- Tools & Platforms
- Features
- Timeline

My Project

- Create a voxel ray-traced rendering engine
  - Real-time (>30 FPS) performance
  - Lighting effects (shadows, reflections, etc.)
- Provide a baseline for games or visualizations
  - Designed to be extended

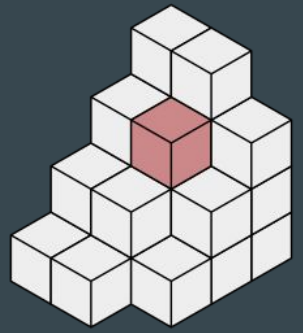
Background - Ray Tracing

- Rendering algorithm that simulates bouncing light
- Enables straightforward implementation of many effects
  - Shadows, AO
  - Reflections, Refraction
- Traditionally too slow for real time usage



Background - Voxels

- Volumetric Pixel
- Often used for destructible environments in games (e.g. Minecraft)
- Applications for data visualization (e.g. medical scans)
- Can speed up ray tracing



Prior Work - Teardown

- A voxel-based destruction game
- Uses ray tracing for all rendering
- Inspiration for my project

Tools and Platforms

- C++ Desktop Application
- CMake Buildsystem
- CLion IDE
- Vulkan Rendering API
- GLSL Shaders
- Glfw Window
- ImGui GUI

Feature Requirements

Minimum Viable Product

- Voxel scene storage in 3D texture
- Screen-space voxel ray tracing for direct lighting
- Ray-traced ambient occlusion and shadows
- Image denoising filter

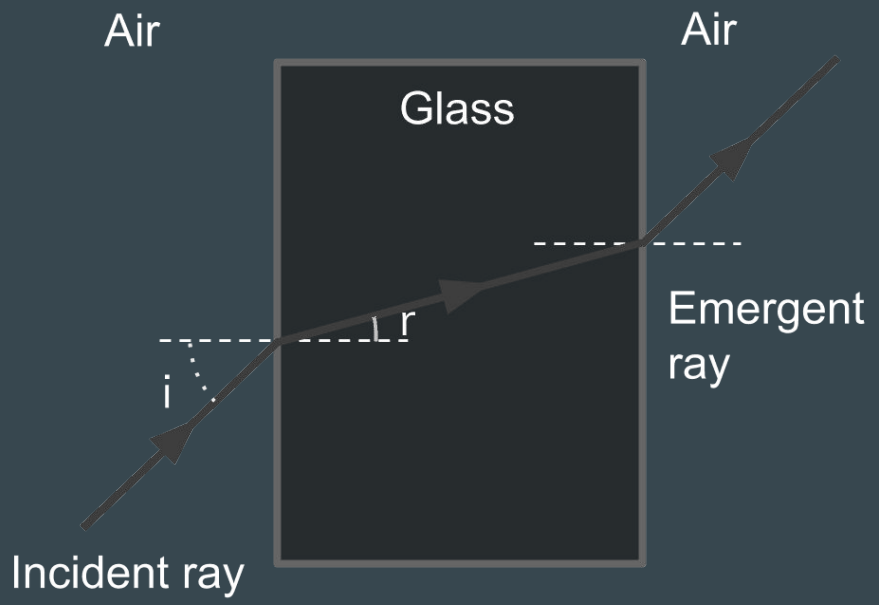
Version 1.0

- GUI interface to adjust rendering settings
- Loading existing scenes from Magicavoxel .vox format
- Skybox image-based lighting
- Ray-traced reflections

Future Work

Version 2.0

- Transparent voxels and refraction
- Emissive materials
- Multiple voxel volumes
- Voxel animations



Timeline

August - C++ project setup, Vulkan hello world ✔

September - Renderer design, ray tracing basics ✔

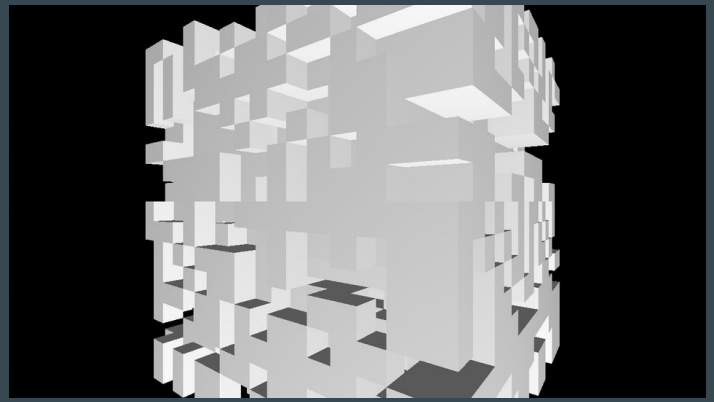
September 26th - Requirements Presentation

October - Remaining MVP features

November 7th - Design Presentation

November - Version 1.0 Features

Dead Week - Finals Presentation



Questions?