Tessellated Voxelization for Global Illumination using Voxel Cone Tracing

Sam Freed

Advisor: Dr. Christian Eckhardt
Committee Members: Dr. Maria Pantoja, Dr. Aaron Keen

June 2018

California Polytechnic State University, San Luis Obispo

Outline

1. Introduction
2. Background
3. Implementation
4. Results and Conclusions

Introduction

Computer Graphics

How can we simulate light in a virtual world?

Physically accurate indirect lighting is too complex for real-time—approximate!

image from VCT and SVO for Real-time Global Illumination by Cyril Crassin

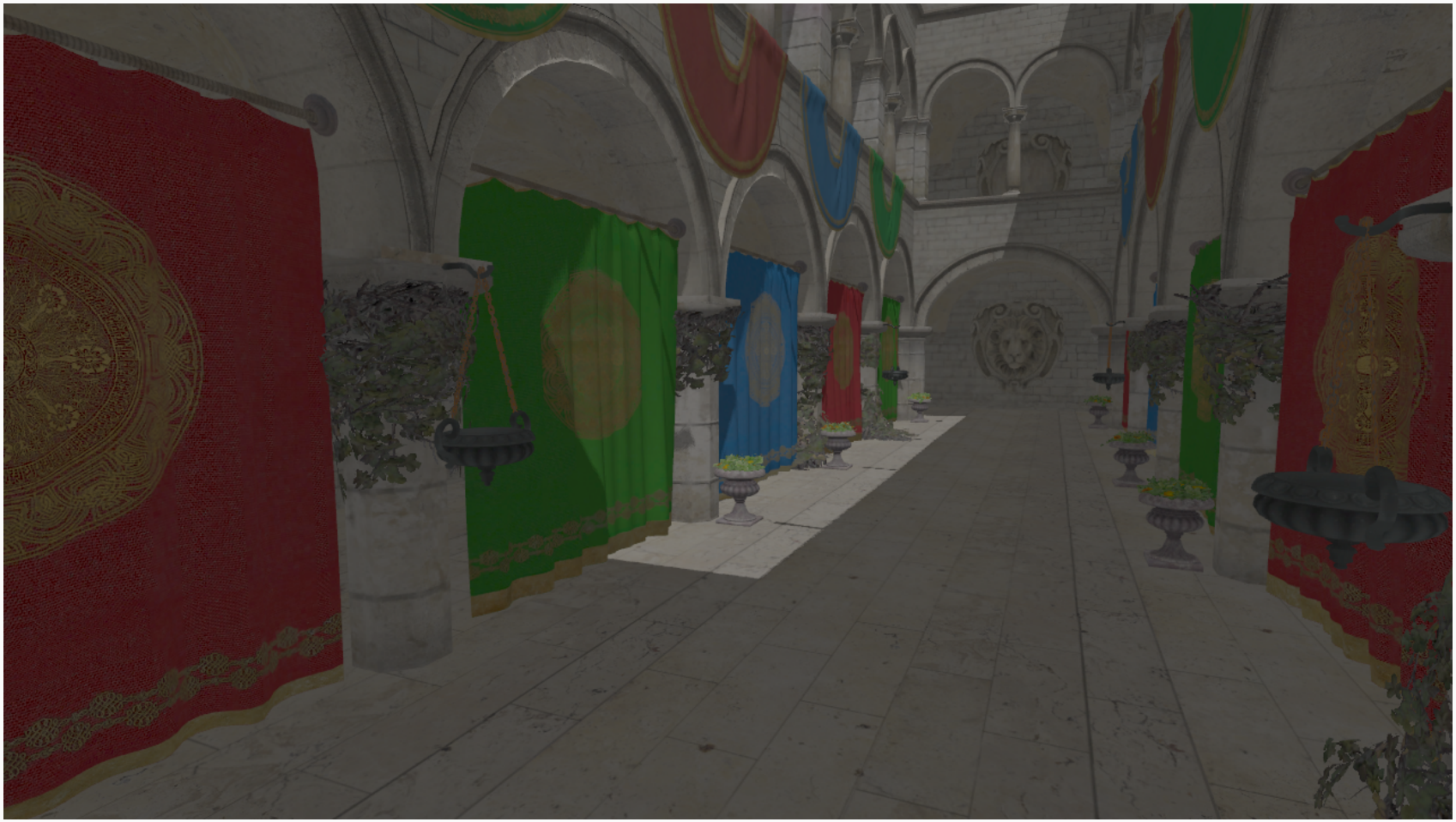
Real-Time Global Illumination

Some existing approaches:

Constant	Flat ambient
Partial	Ambient occlusion, screen-space approaches
Static	Baked lighting, light probes
Dynamic	Reflective Shadowmaps, Light Propagation Volumes, Voxel Cone Tracing

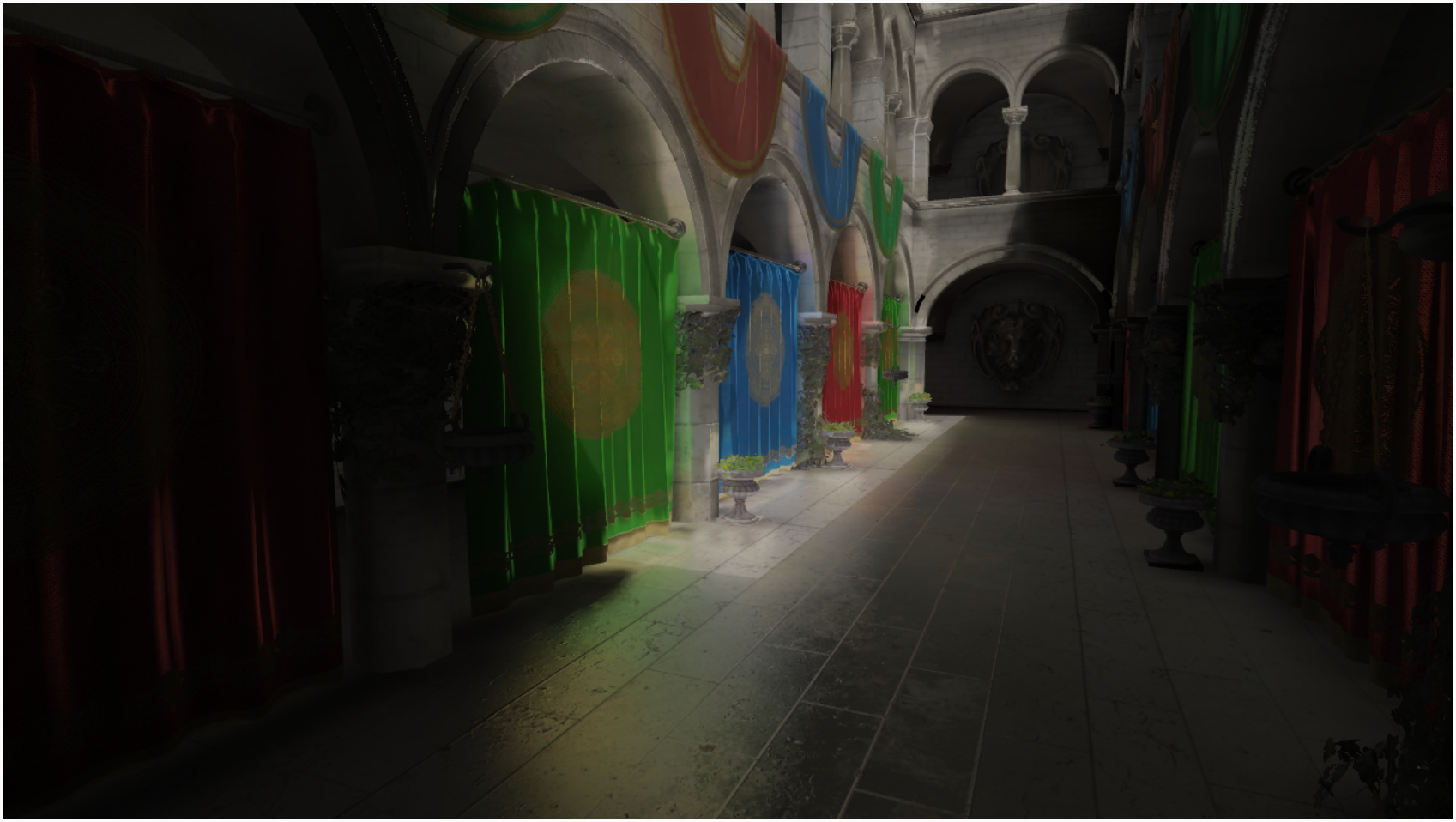
”

Real-Time Global Illumination



Flat ambient approximation

Real-Time Global Illumination



Full global illumination

Motivations

1. Real-time global illumination is difficult
    - Time and memory constraints
    - Multiple steps
2. Limited reference material
    - Proprietary code
    - Demo code
    - Engine code

Contributions

Implement

- Open-source and cross-platform
- Not tied to any particular engine

Extend

- Comparison of two methods of scene voxelization
- Investigation into warped voxels

Computer Graphics Primer

Goal

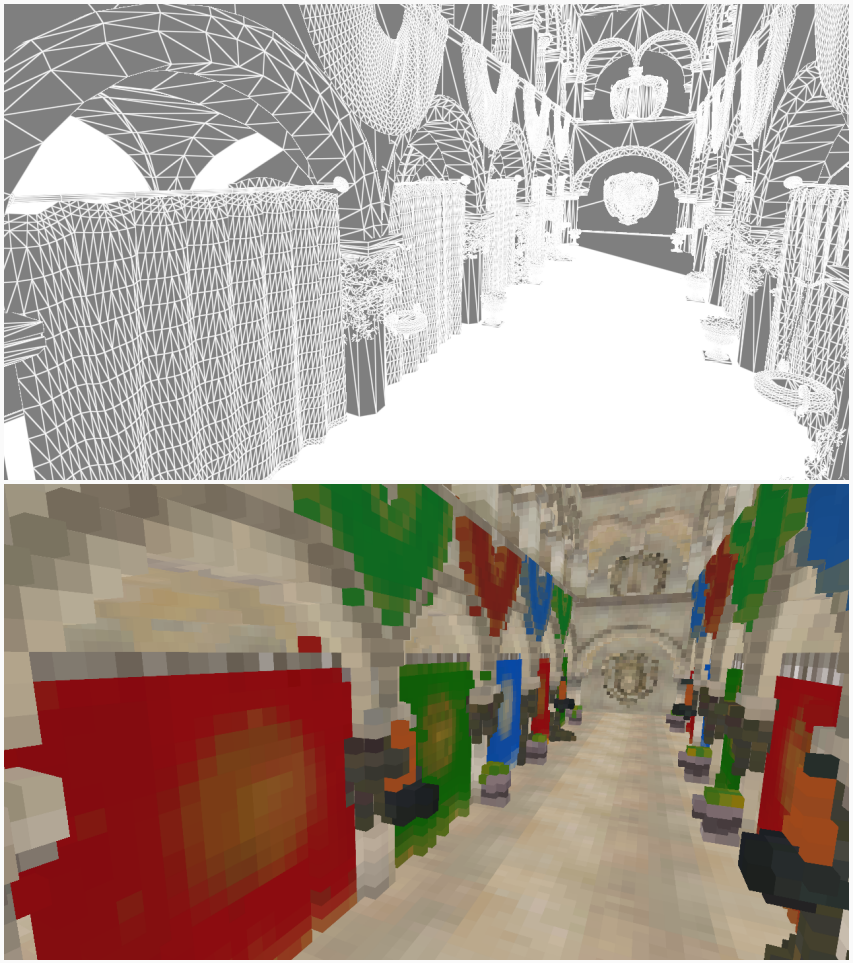
Given a virtual description of a scene, render an image.

Big Issues

1. How do we represent a scene? What information is required?
2. How is a 3D scene represented as a 2D image?
3. How do we render—how is the final pixel color computed?

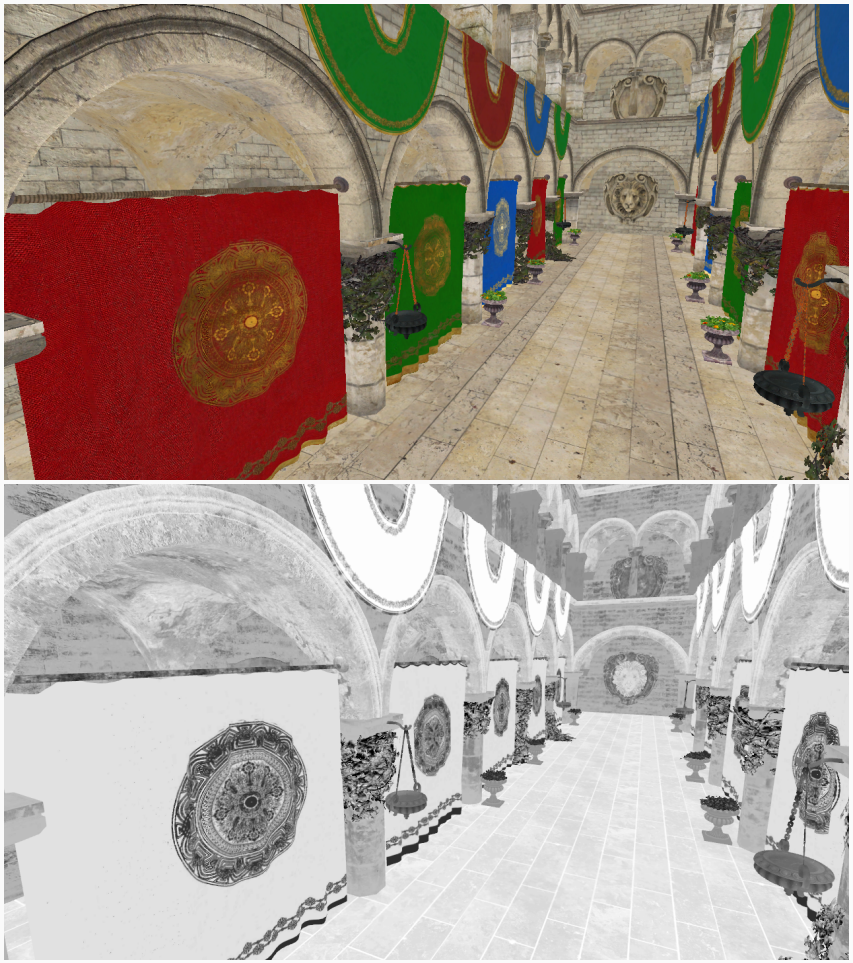
How do we represent a scene?

Geometry triangles, voxels



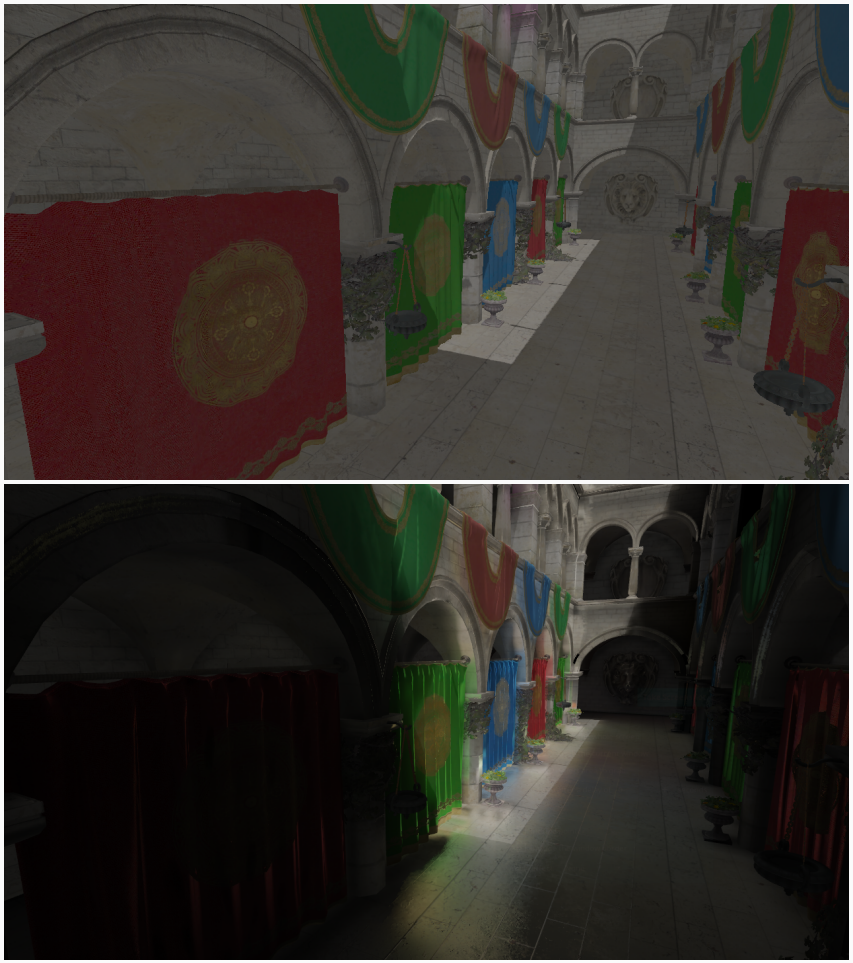
How do we represent a scene?

Materials colors and other properties



How do we represent a scene?

Lights positions, colors, etc.



How is a 3D scene represented as a 2D image?

Math!

All coordinates are transformed multiple times before ending up at their appropriate place on the screen.

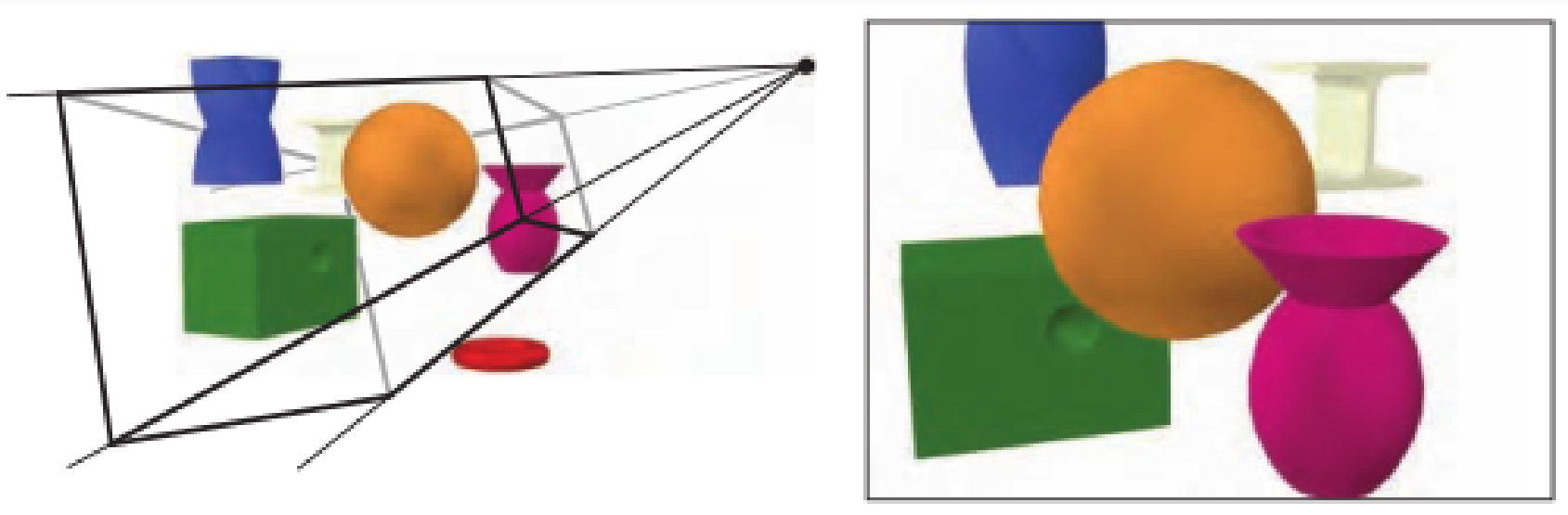
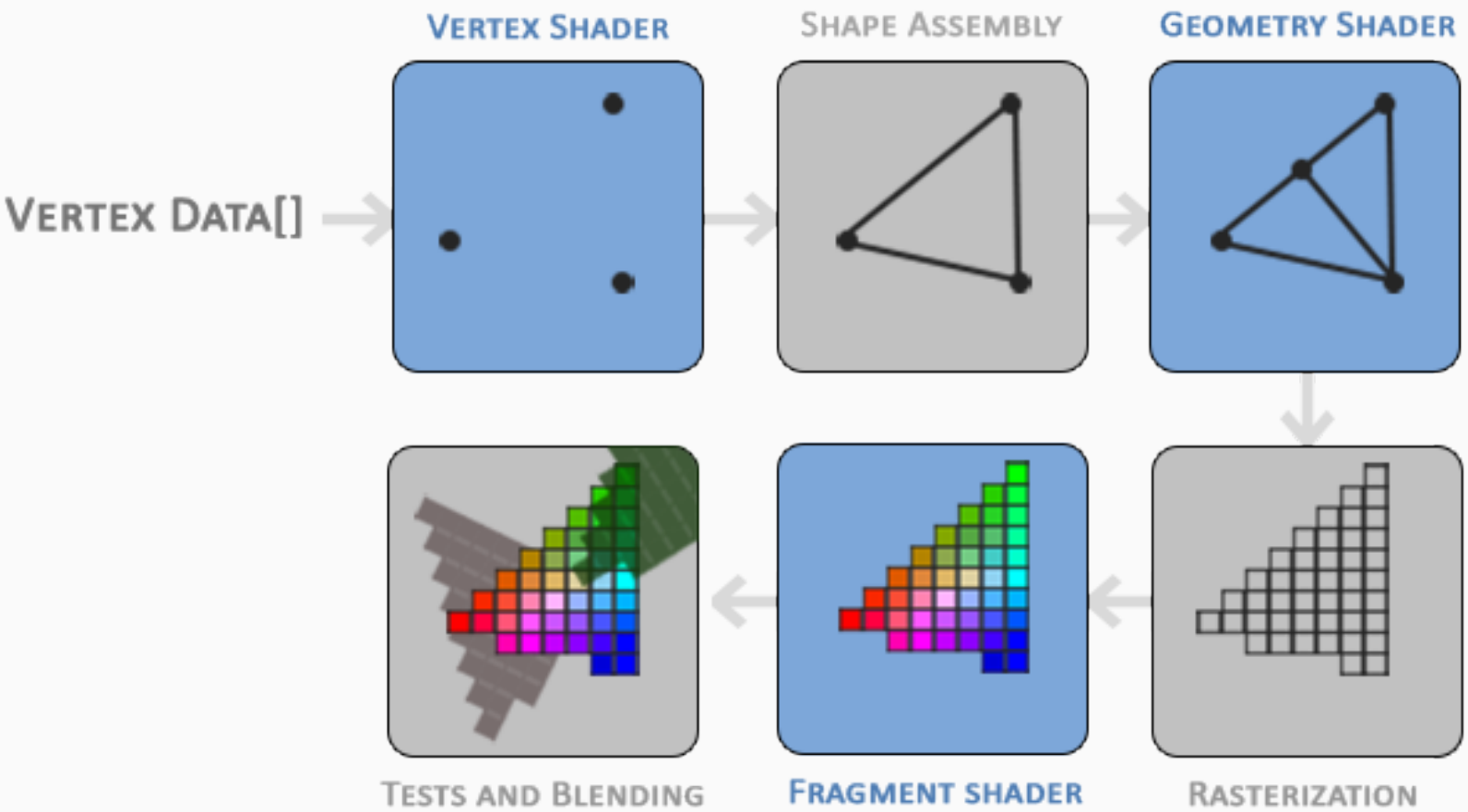
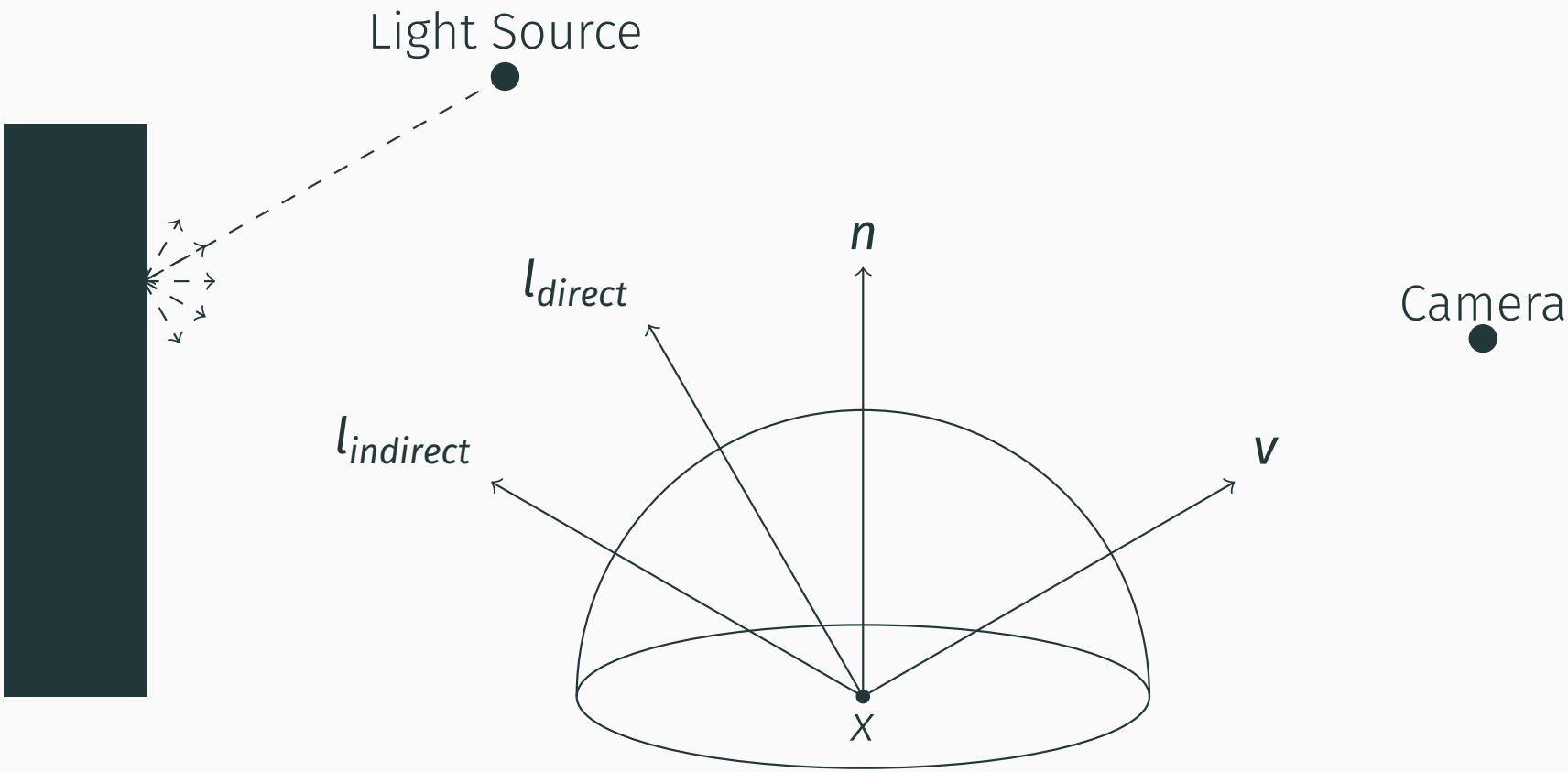


image from Real-Time Rendering

The Graphics Pipeline



Light Theory



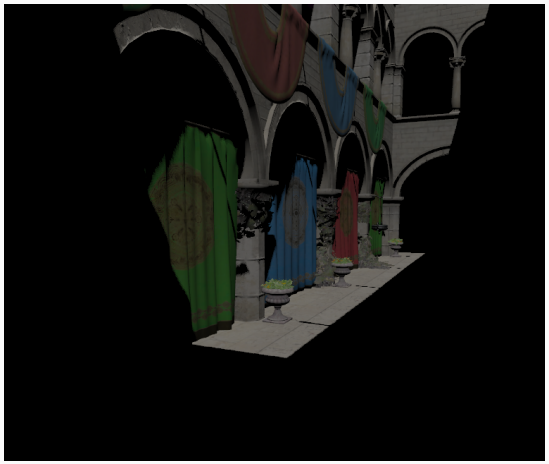
Real-Time Global Illumination

Most dynamic global illumination algorithms follow a few main steps:

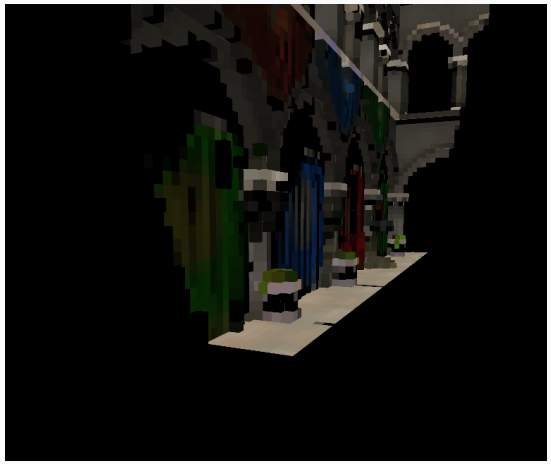
1. Construct representation of the scene
2. Calculate indirect lighting information
3. Collect indirect lighting when rendering

Overview of Renderer

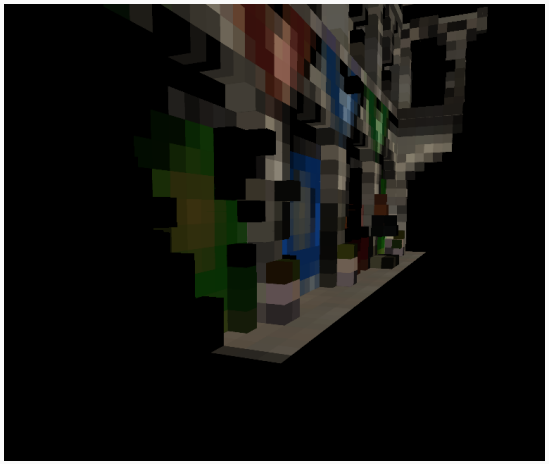
(1) Voxelize



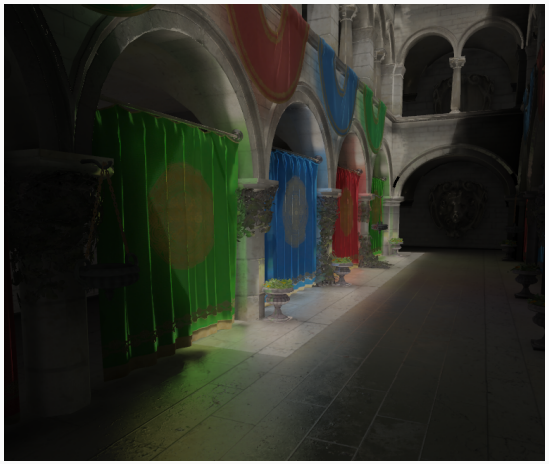
(2) Direct lighting



(3) Inject direct lighting into voxels



(4) Filter voxels



(5) Sample using voxel cone tracing

Voxelization with Tessellator

- Triangles are subdivided into multiple vertices during tessellation
- Tessellation level determines how much to subdivide

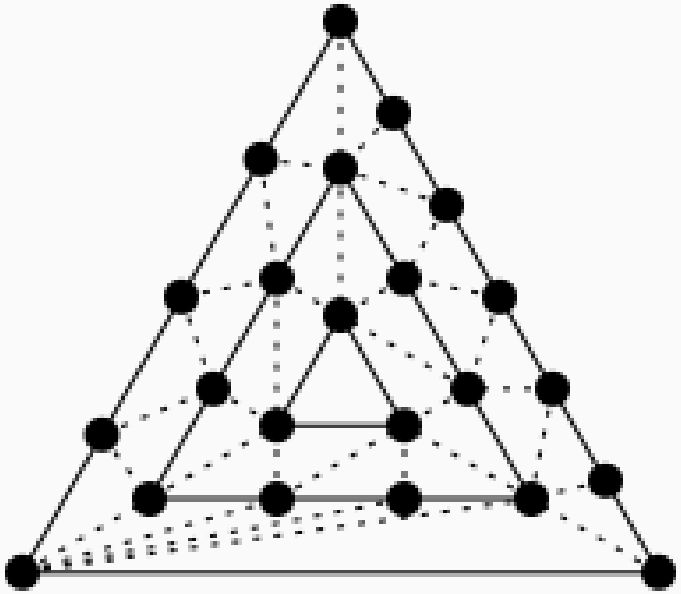
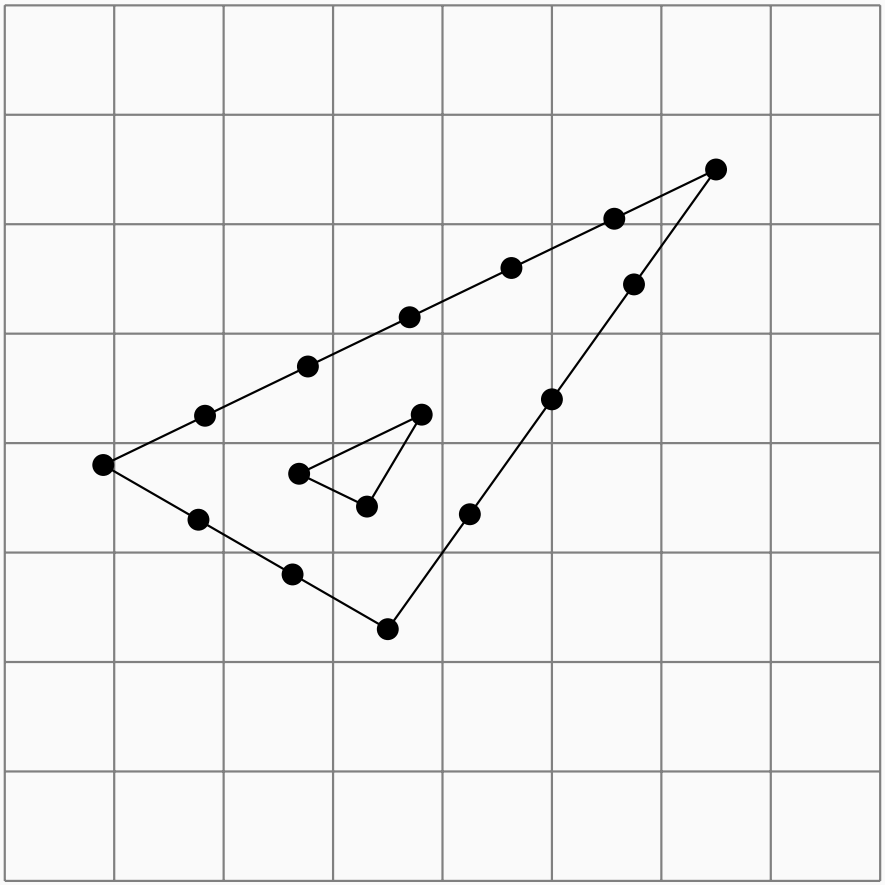


image from OpenGL Wiki

Voxelization with Tessellator

- Each vertex corresponds to a voxel
- Outer levels determined from respective edge lengths
- Inner level determined from maximum triangle altitude length



Voxelization with Rasterizer

Each fragment corresponds to a voxel

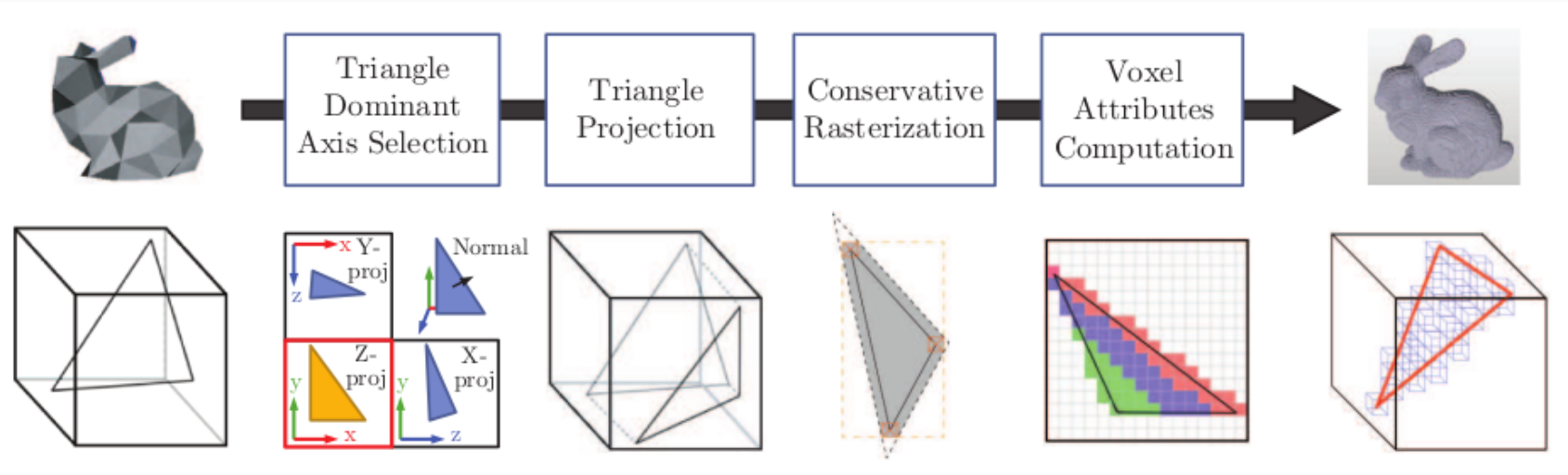
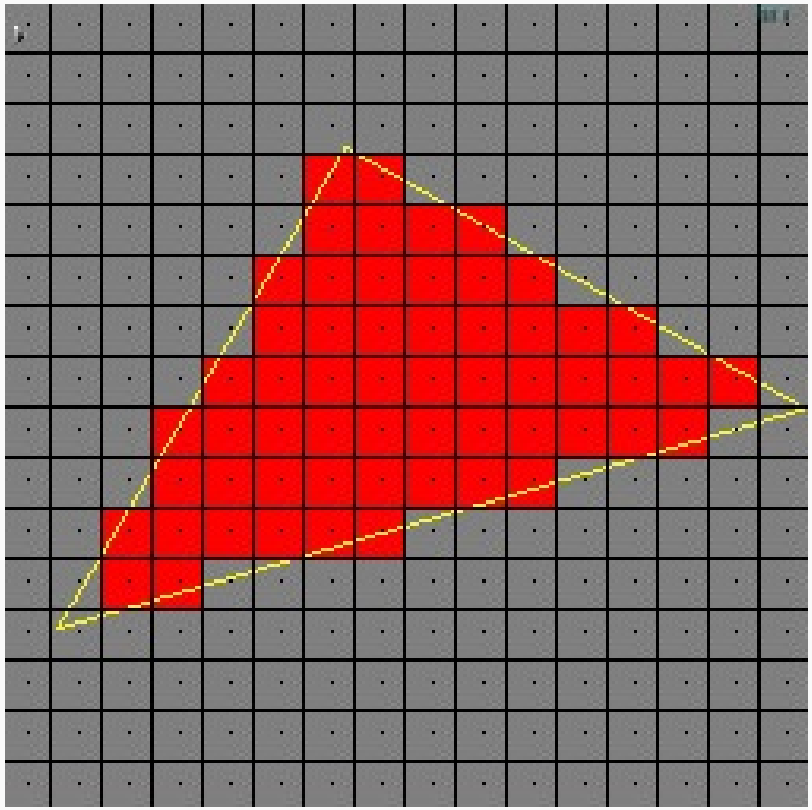
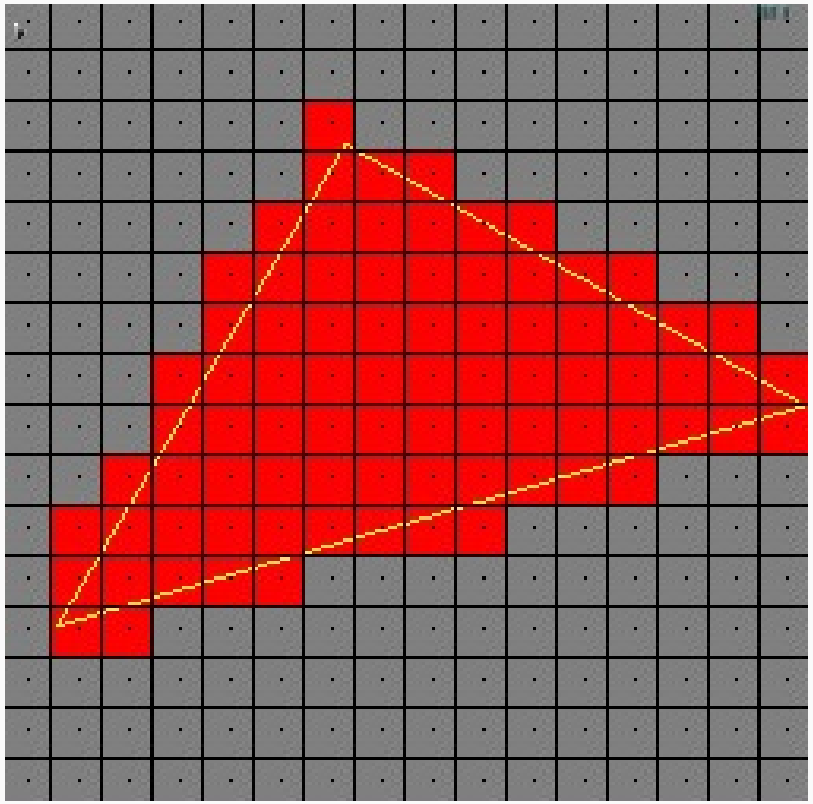


image from OpenGL Insights: Octree-Based Sparse Voxelization Using the GPU Hardware Rasterizer by Cyril Crassin and Simon Green

Conservative Rasterization



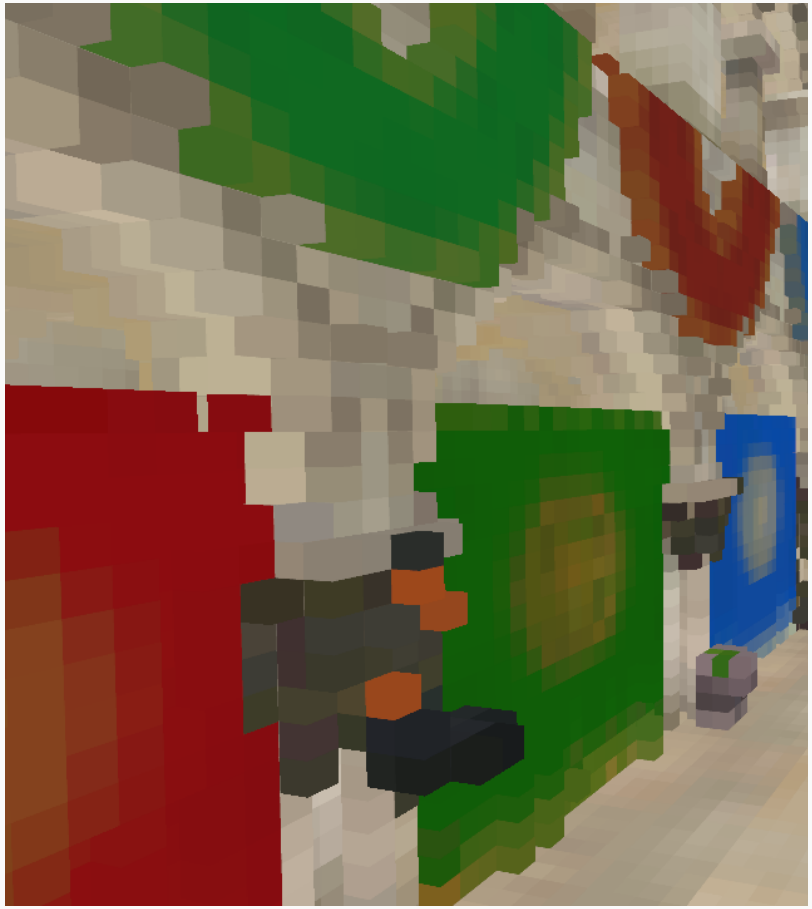
Off



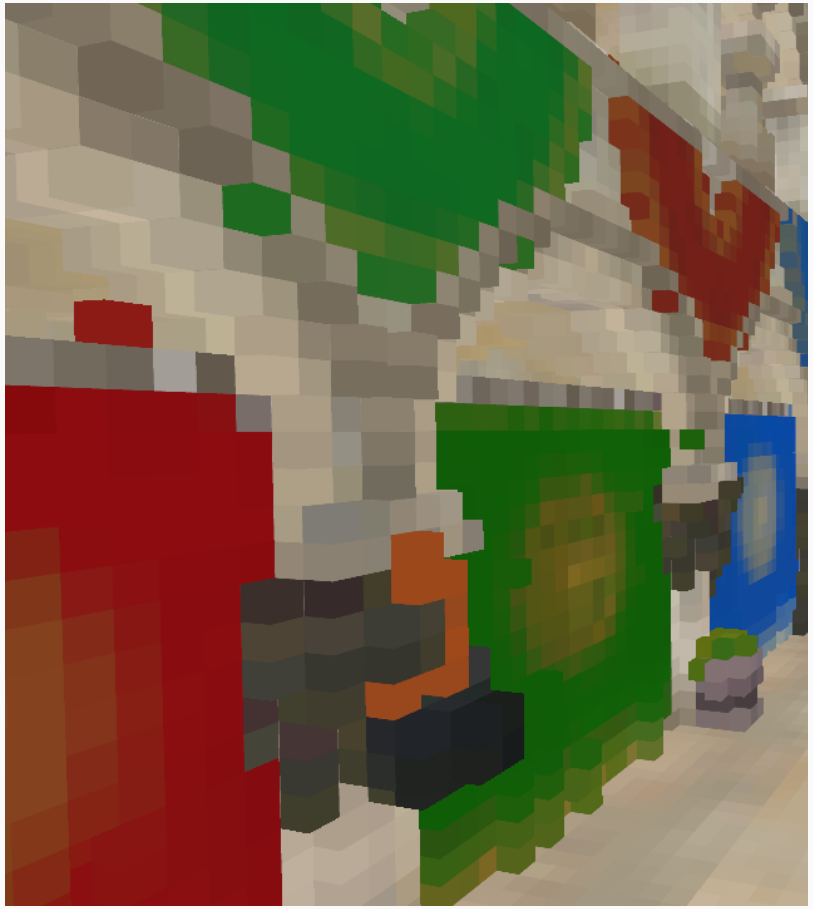
On

images by Jon Story from Don’t be conservative with Conservative Rasterization

Conservative Rasterization



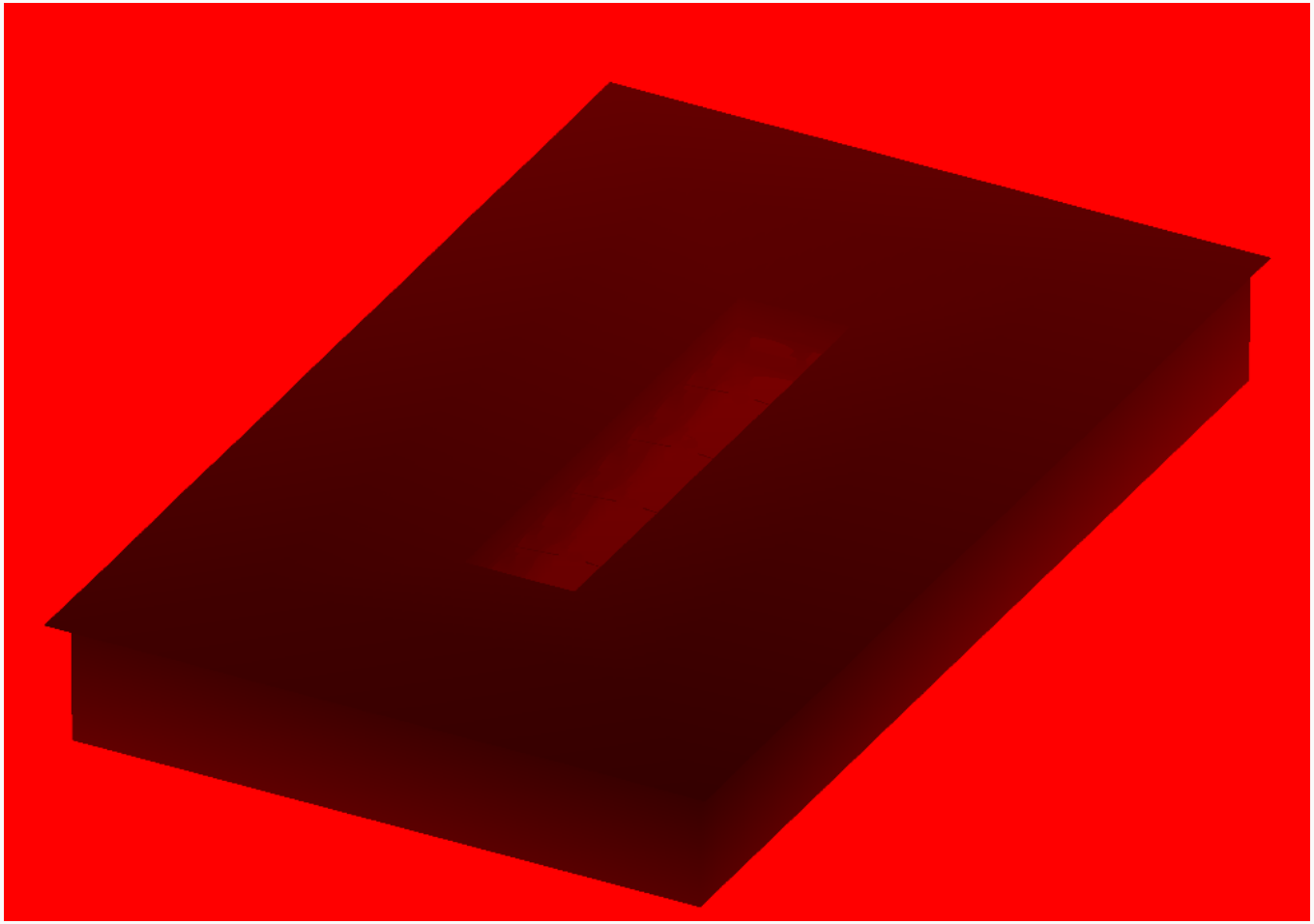
Off



On

Radiance Injection

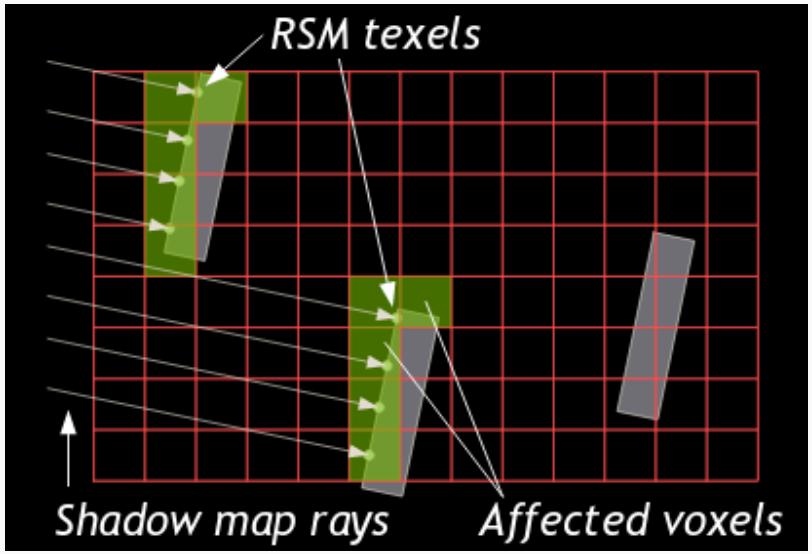
Use **shadowmap** to determine where direct light hits voxels

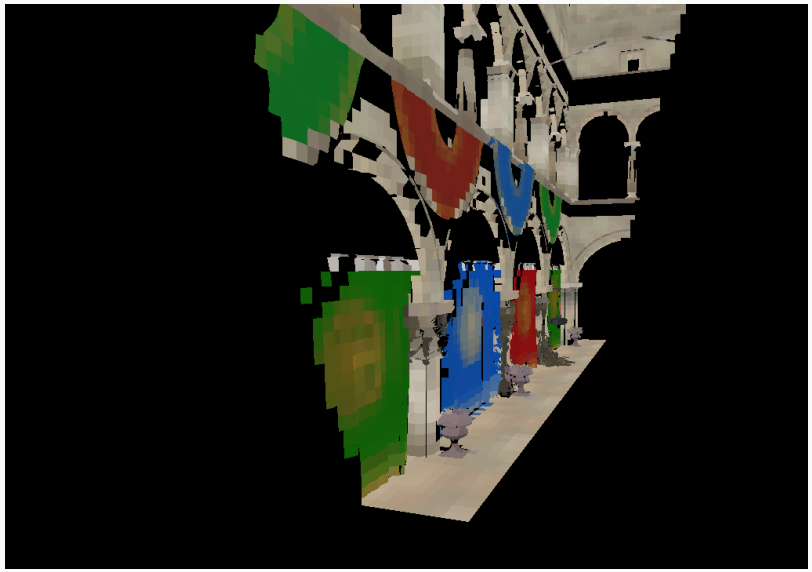


left image from Practical Real-time Voxel-based Global Illumination for Current GPUs by Alexey Panteleev

Radiance Injection

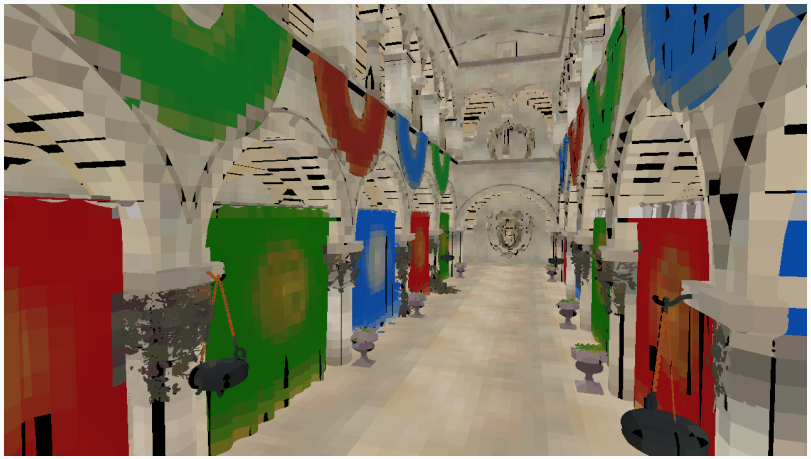
Use **shadowmap** to determine where direct light hits voxels





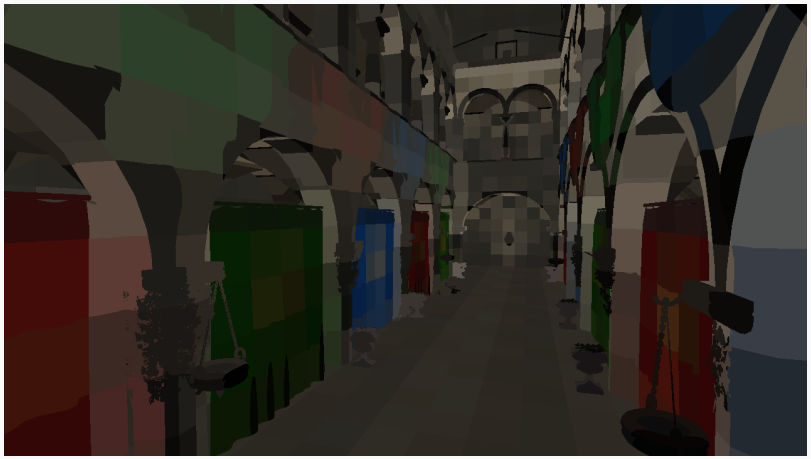
left image from Practical Real-time Voxel-based Global Illumination for Current GPUs by Alexey Panteleev

Radiance Filtering



Level 0

Level 1



Level 2

Level 3

Mipmaps

Indirect Lighting—Voxel Cone Tracing

Voxel Cone Tracing

1. Sample light and occlusion from the radiance texture along a particular direction
2. Adjust level of detail as we get farther from sample point

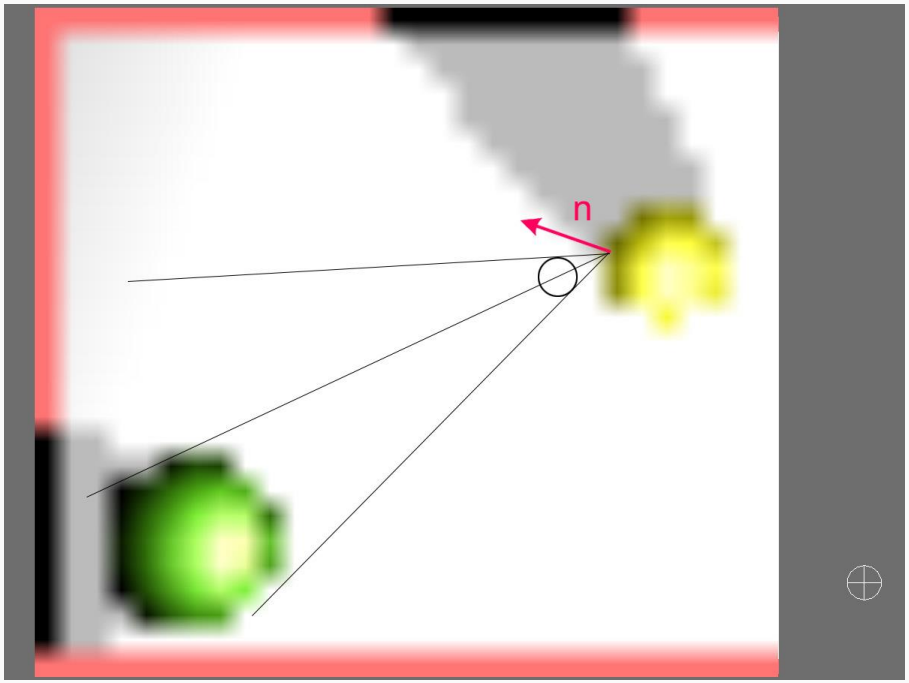


image from Christian Eckhardt

Indirect Lighting—Voxel Cone Tracing

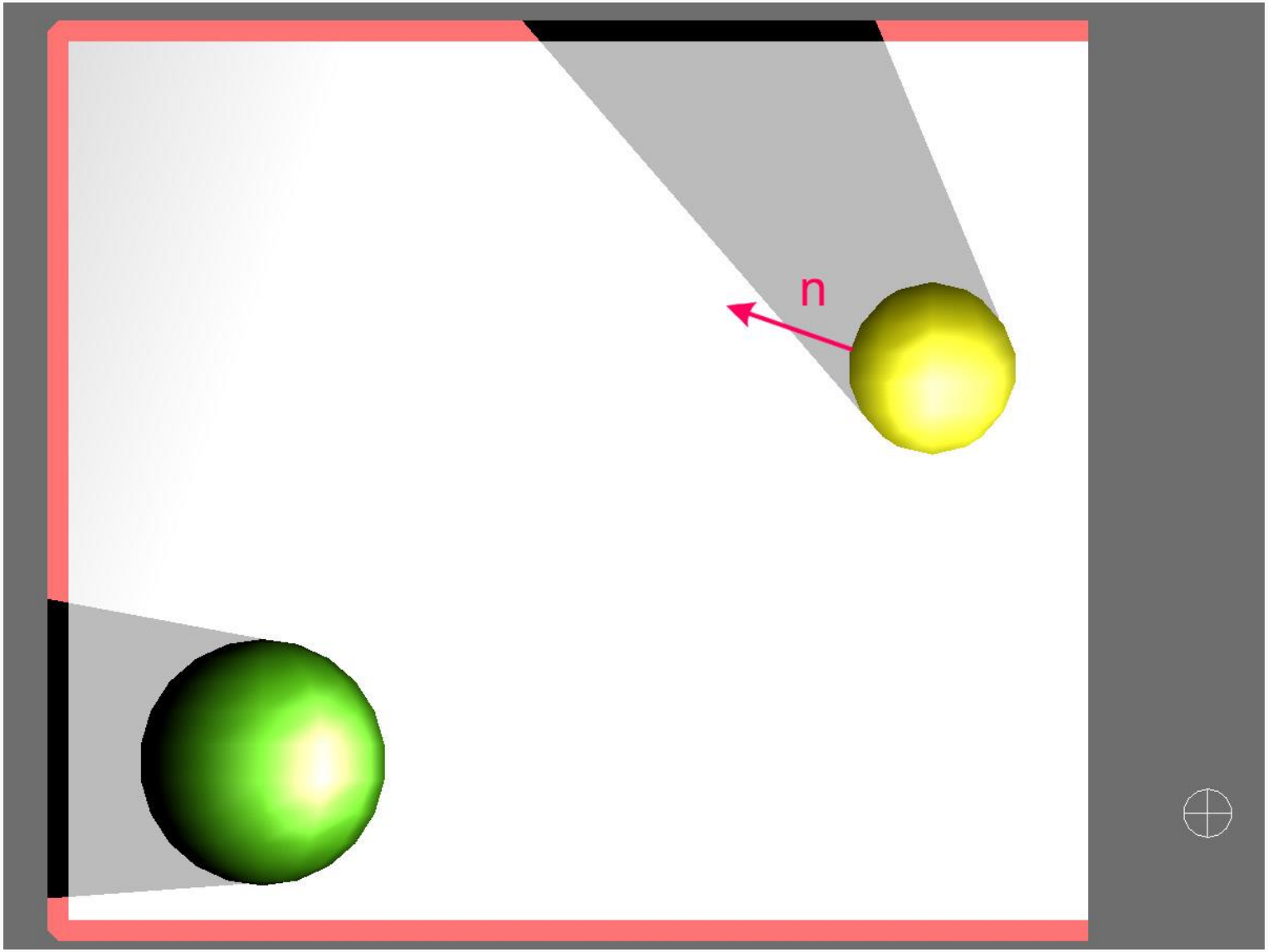


image from Christian Eckhardt

Indirect Lighting—Voxel Cone Tracing

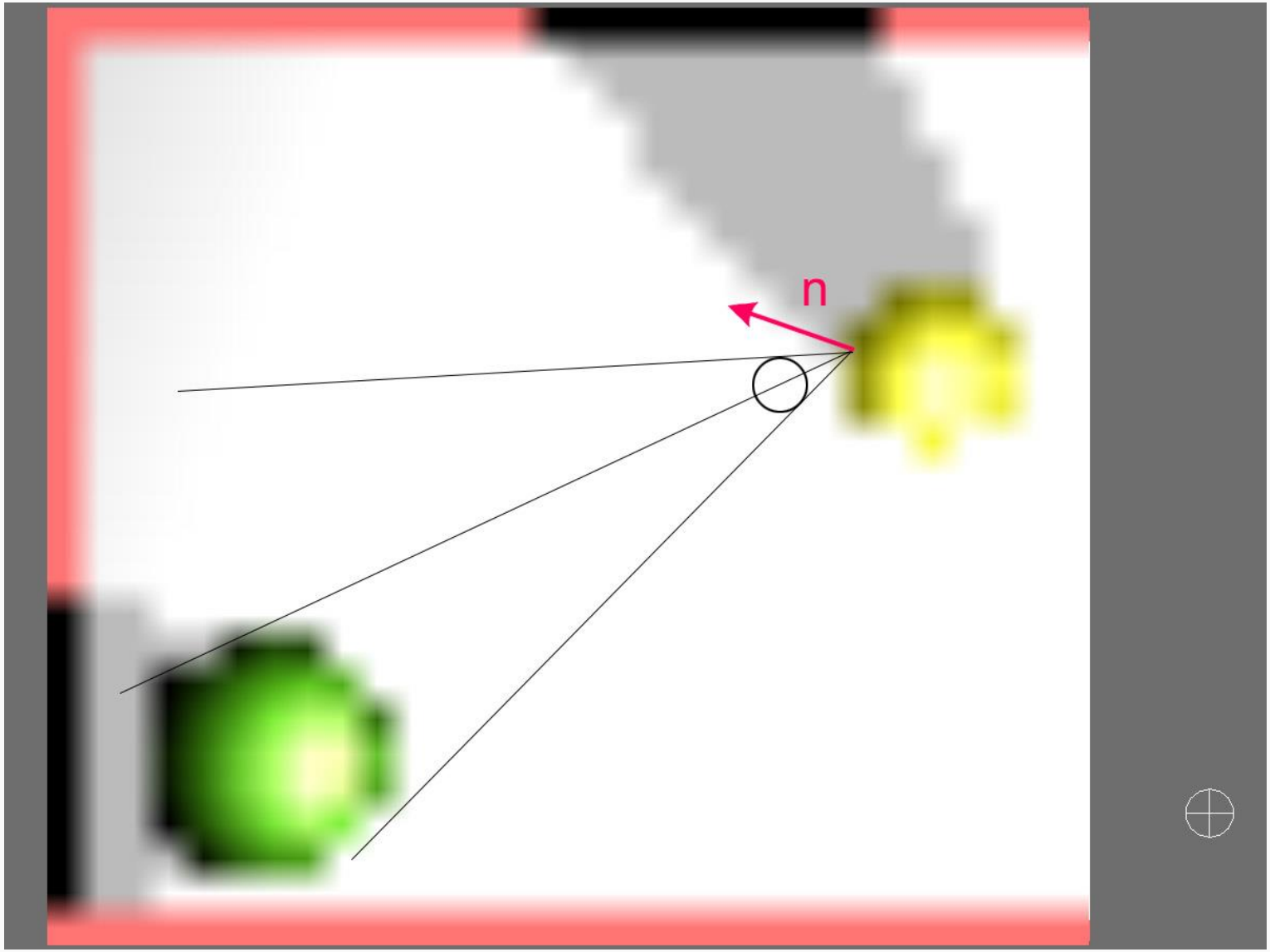


image from Christian Eckhardt

Indirect Lighting—Voxel Cone Tracing

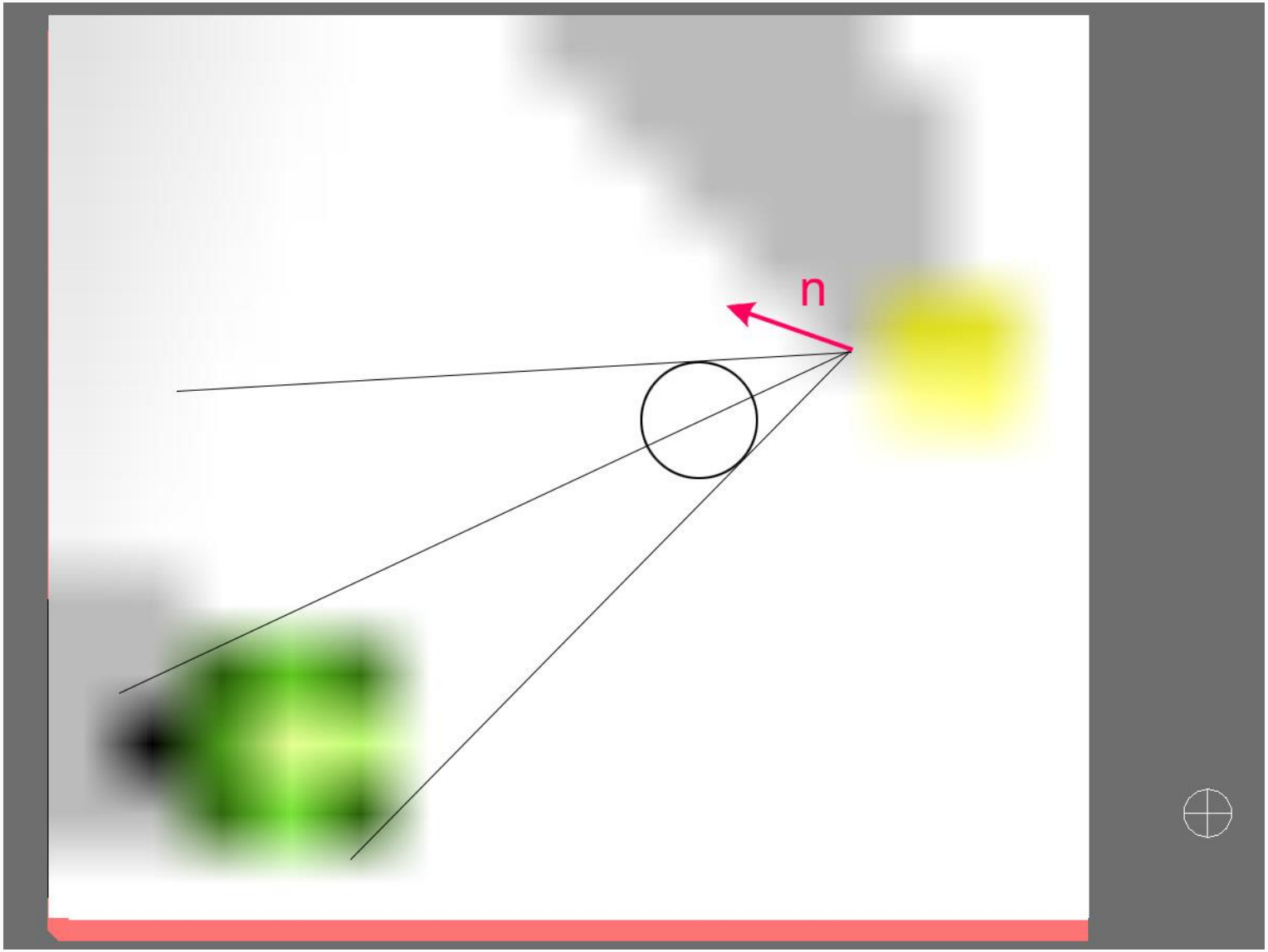


image from Christian Eckhardt

Indirect Lighting—Voxel Cone Tracing

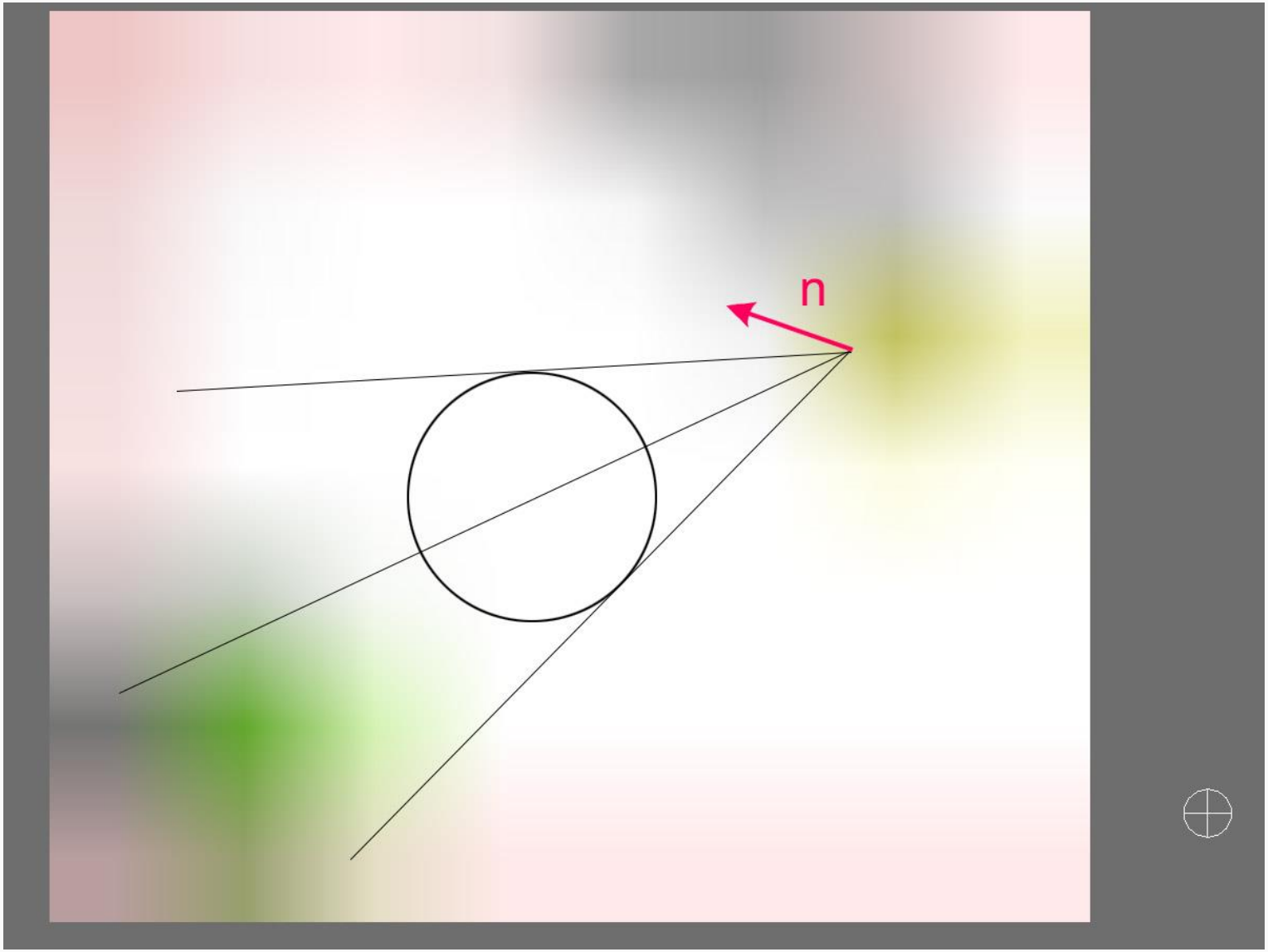


image from Christian Eckhardt

Indirect Lighting—Voxel Cone Tracing

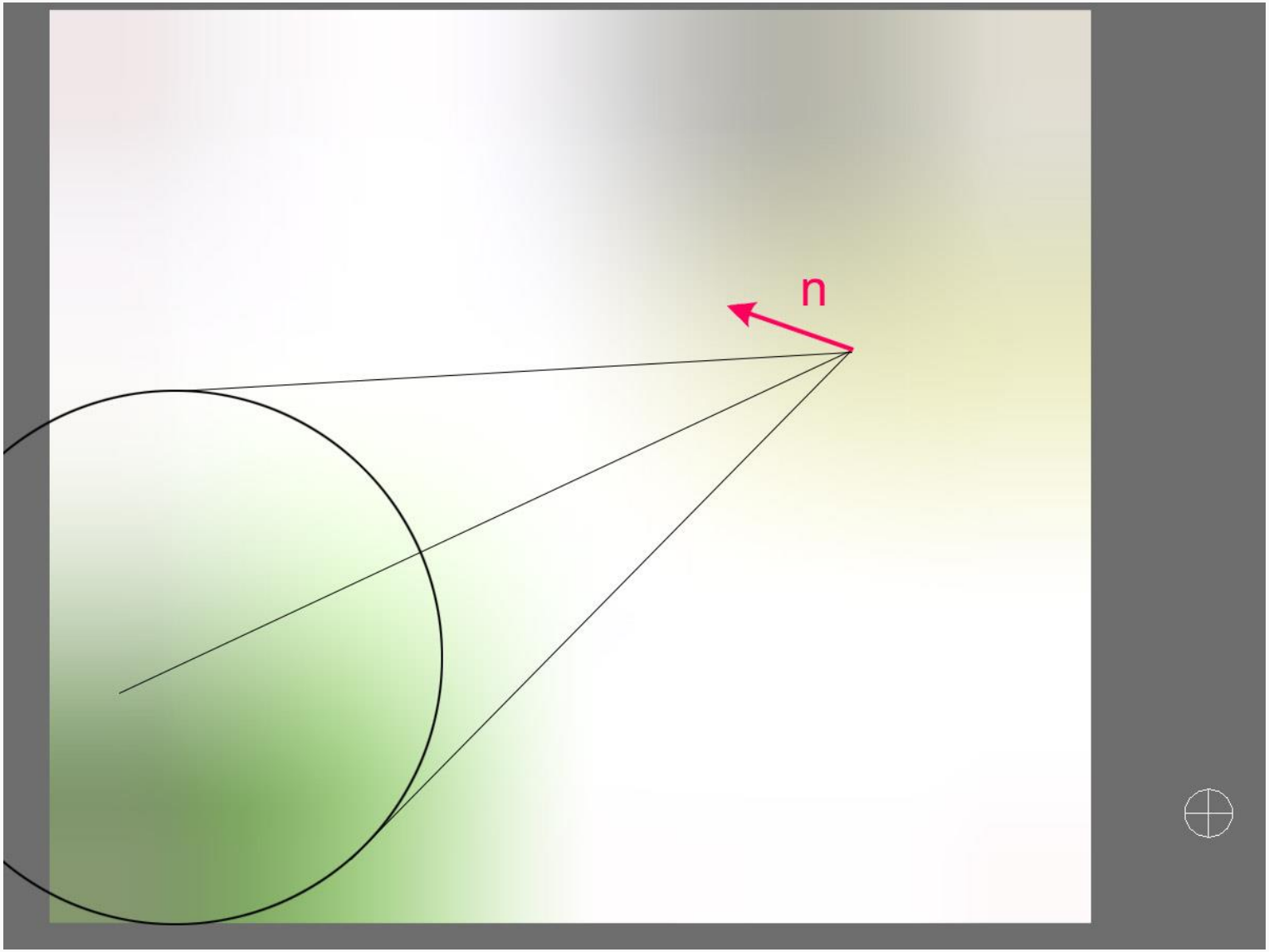


image from Christian Eckhardt

Indirect Lighting—Voxel Cone Tracing

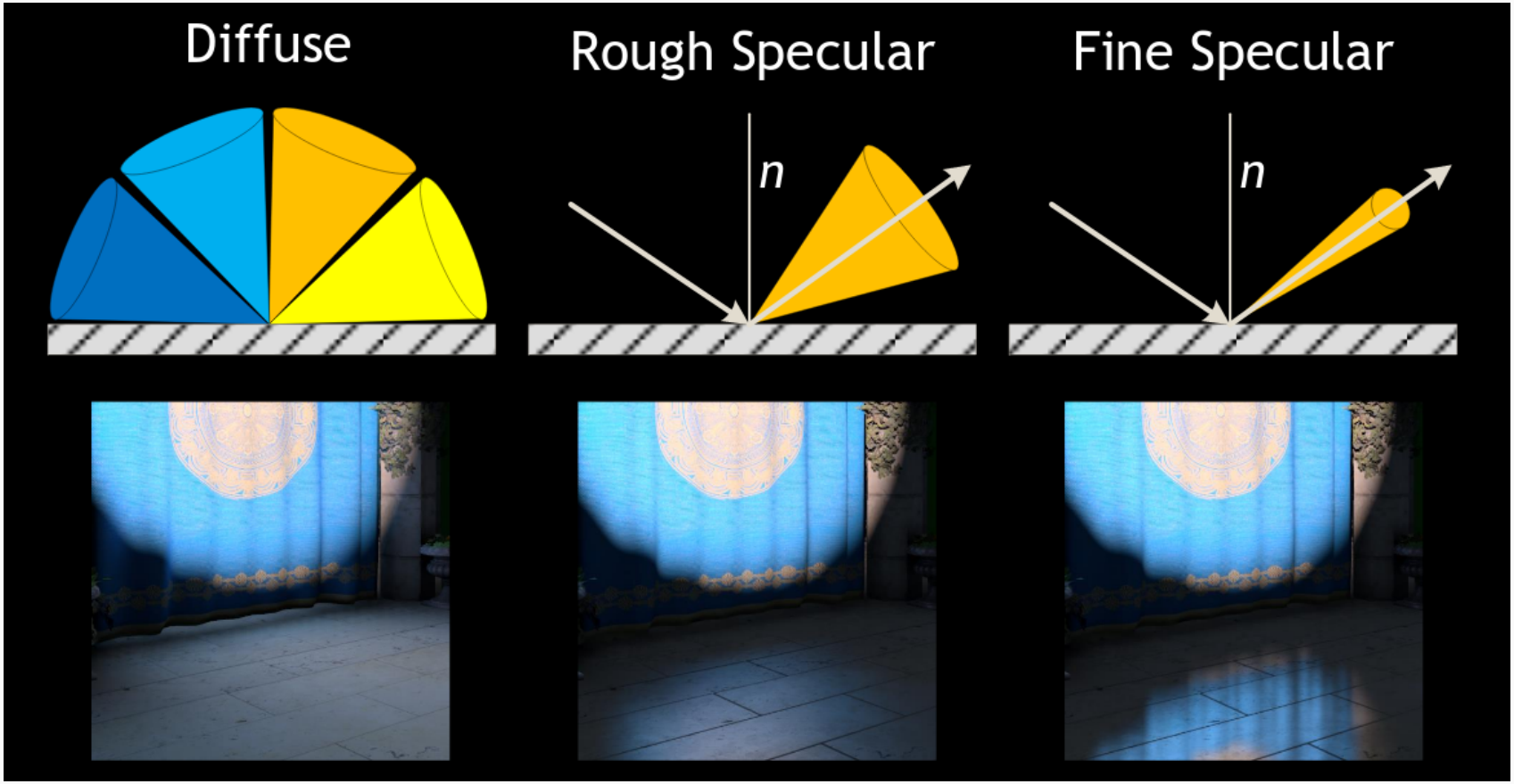
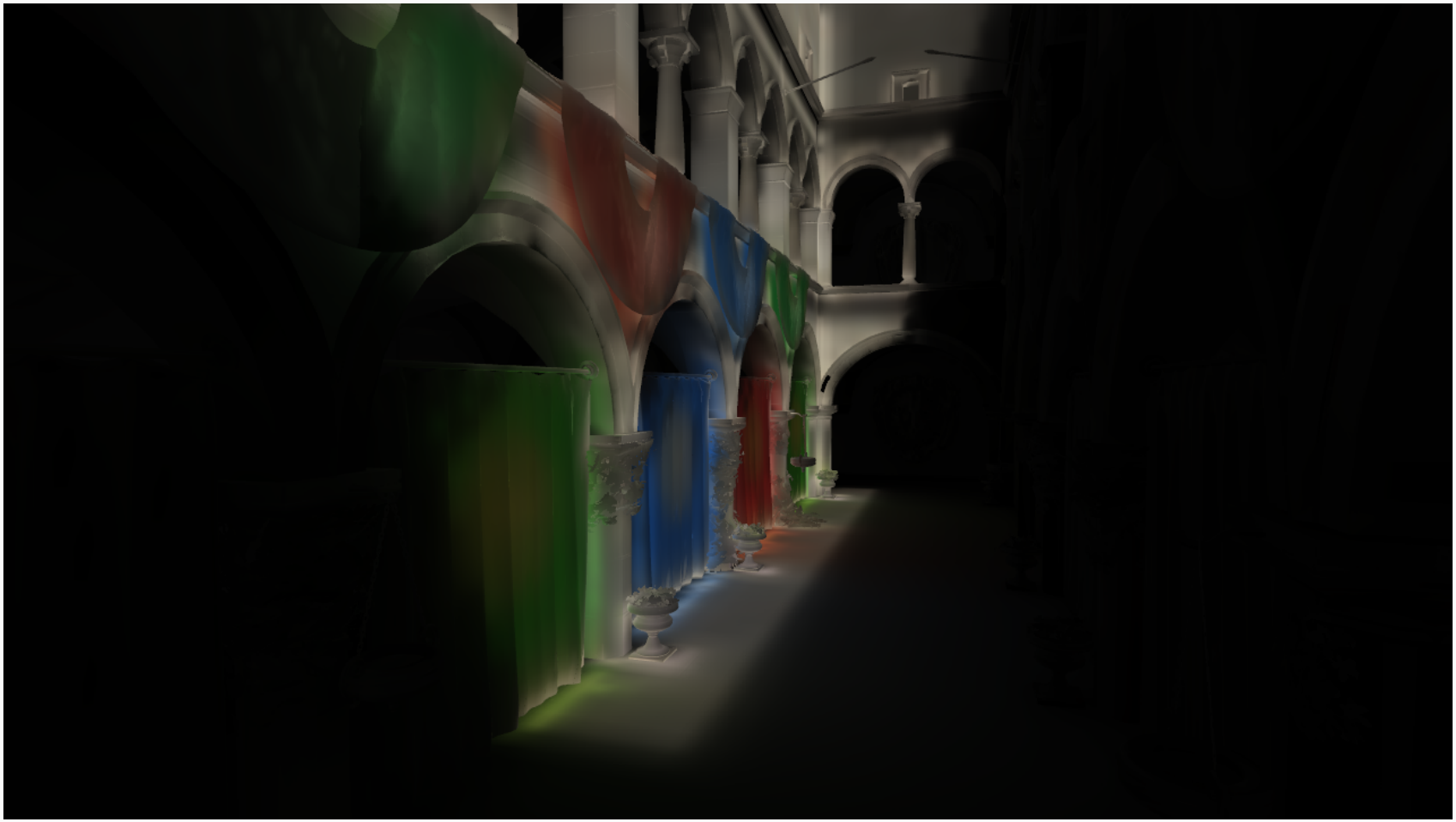


image from Practical Real-time Voxel-based Global Illumination for Current GPUs by Alexey Panteleev

Indirect Lighting



Diffuse Indirect (no occlusion)

Indirect Lighting

Specular Indirect (no occlusion)

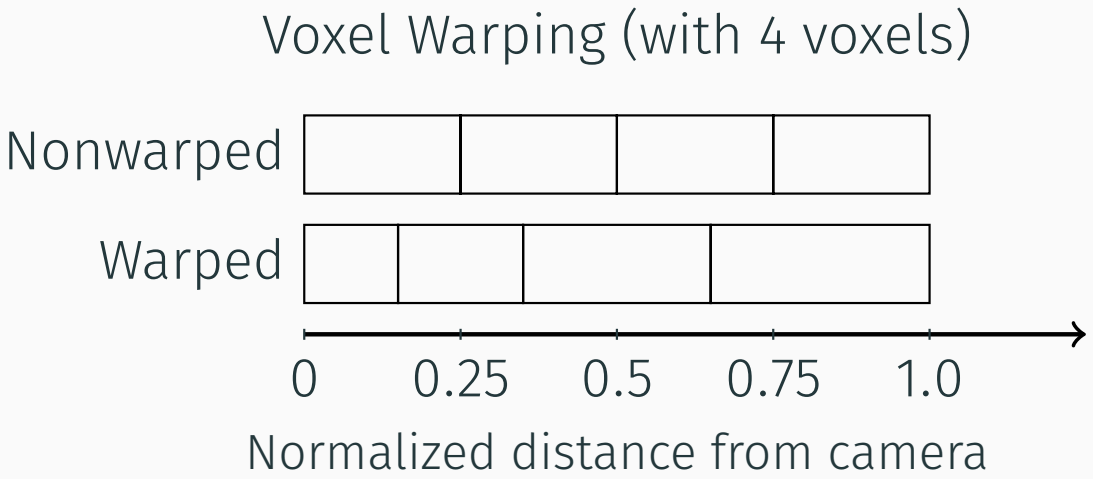
Indirect Lighting

Occlusion

Voxel Warping

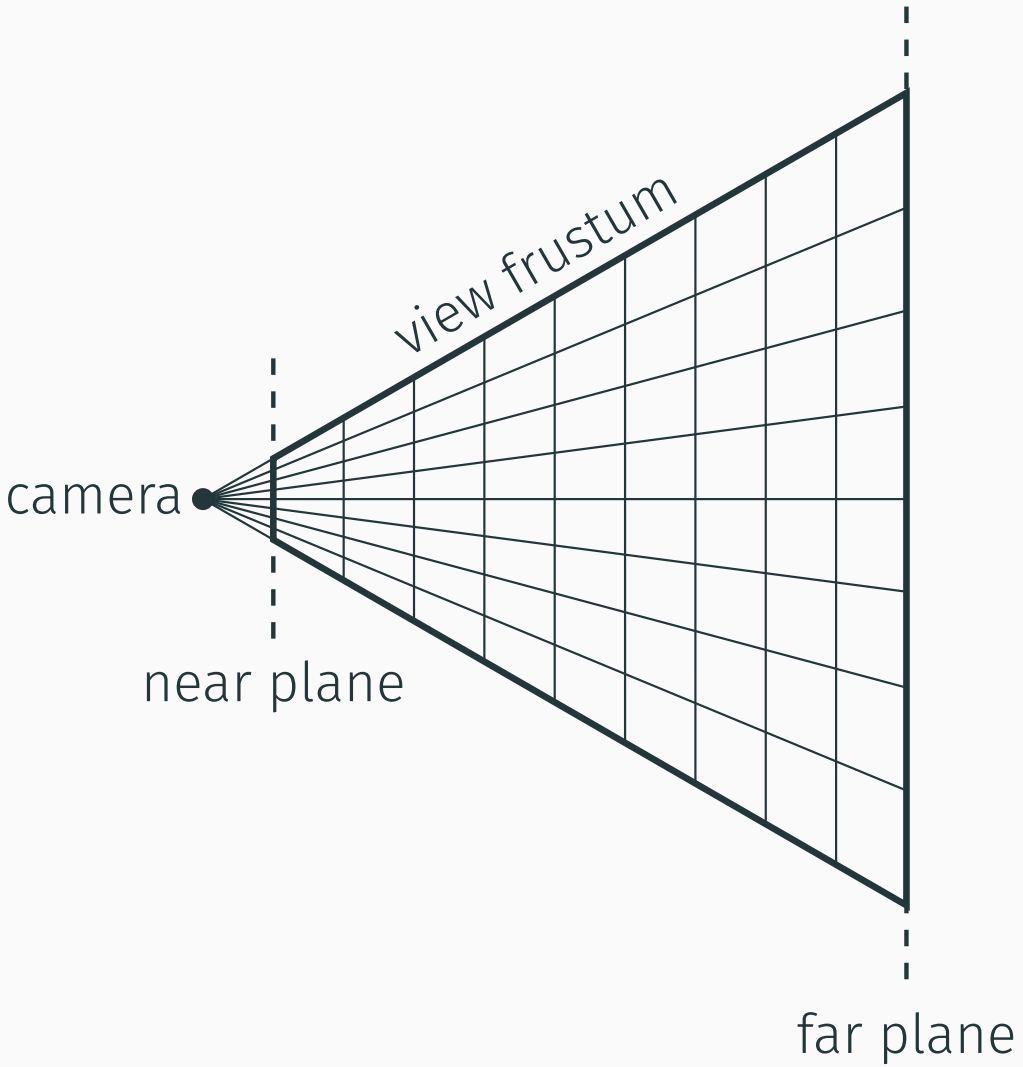
- Voxels are usually restricted to discrete sizes
- What if the size is not restricted?
    1. Vary with distance from camera
    2. Vary based on perspective

Vary with distance from camera



Vary based on perspective

- Use perspective projection to determine voxel size
- Makes voxel size based on relative size in screen space



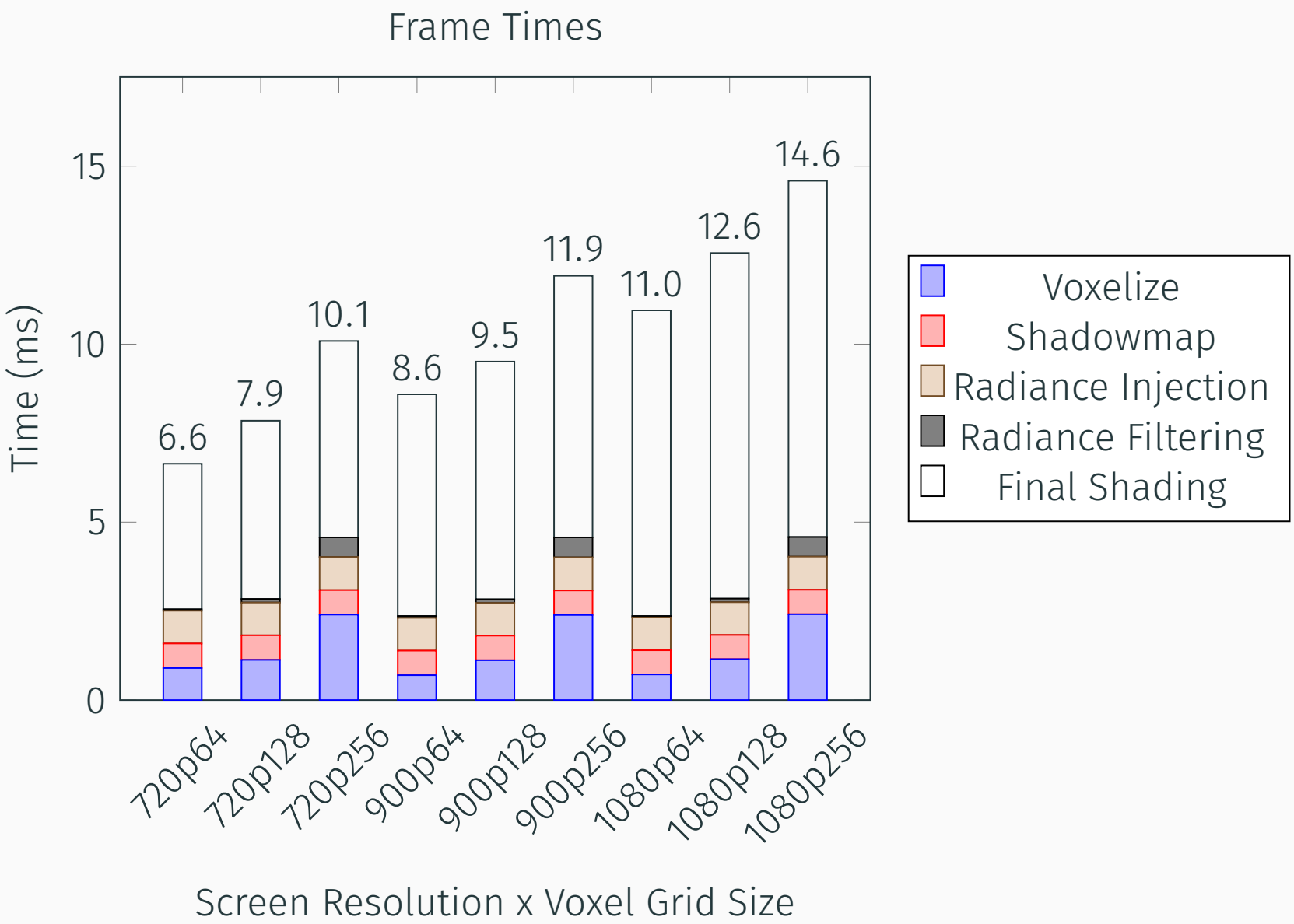
Results

64^3

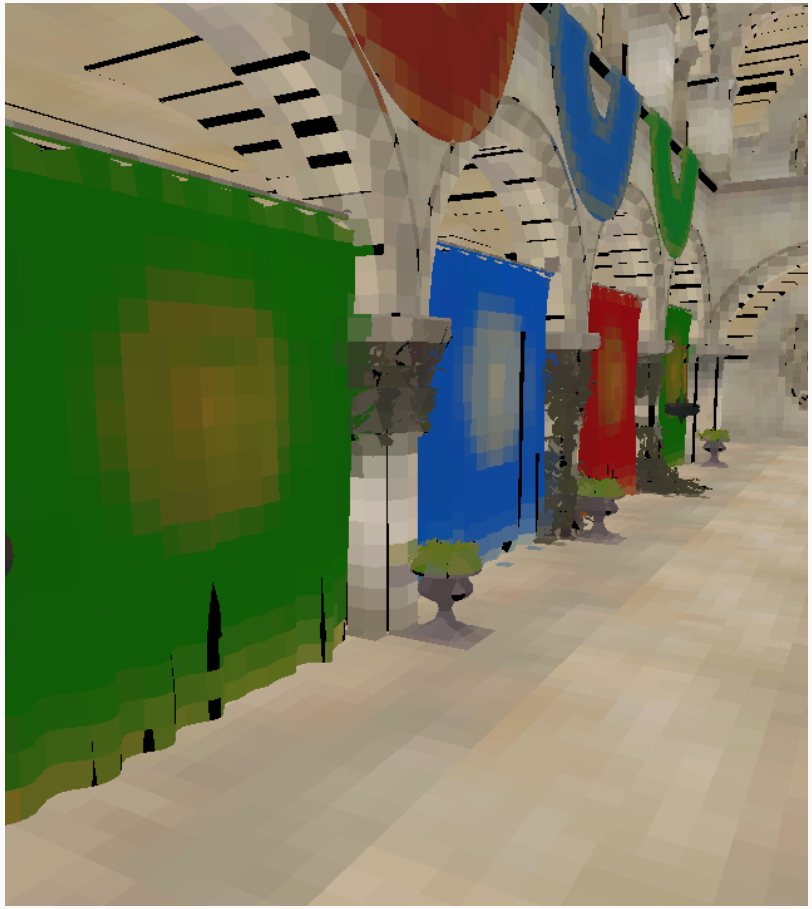
128^3

256^3

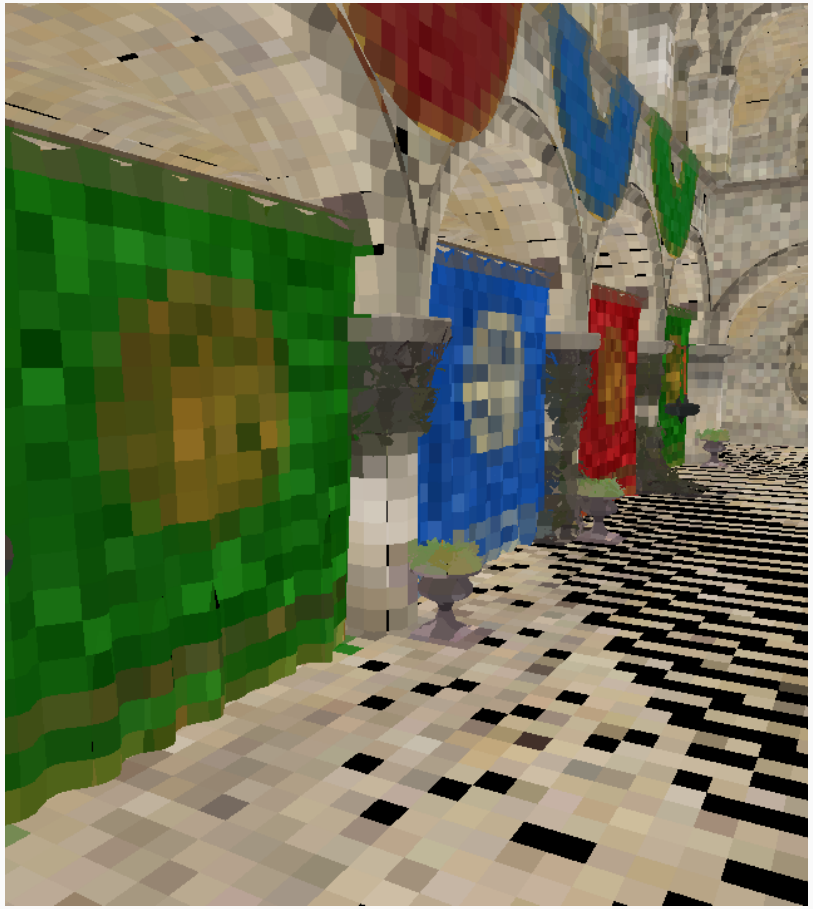
Performance



Rasterized vs. Tessellated Voxels



Rasterized voxels



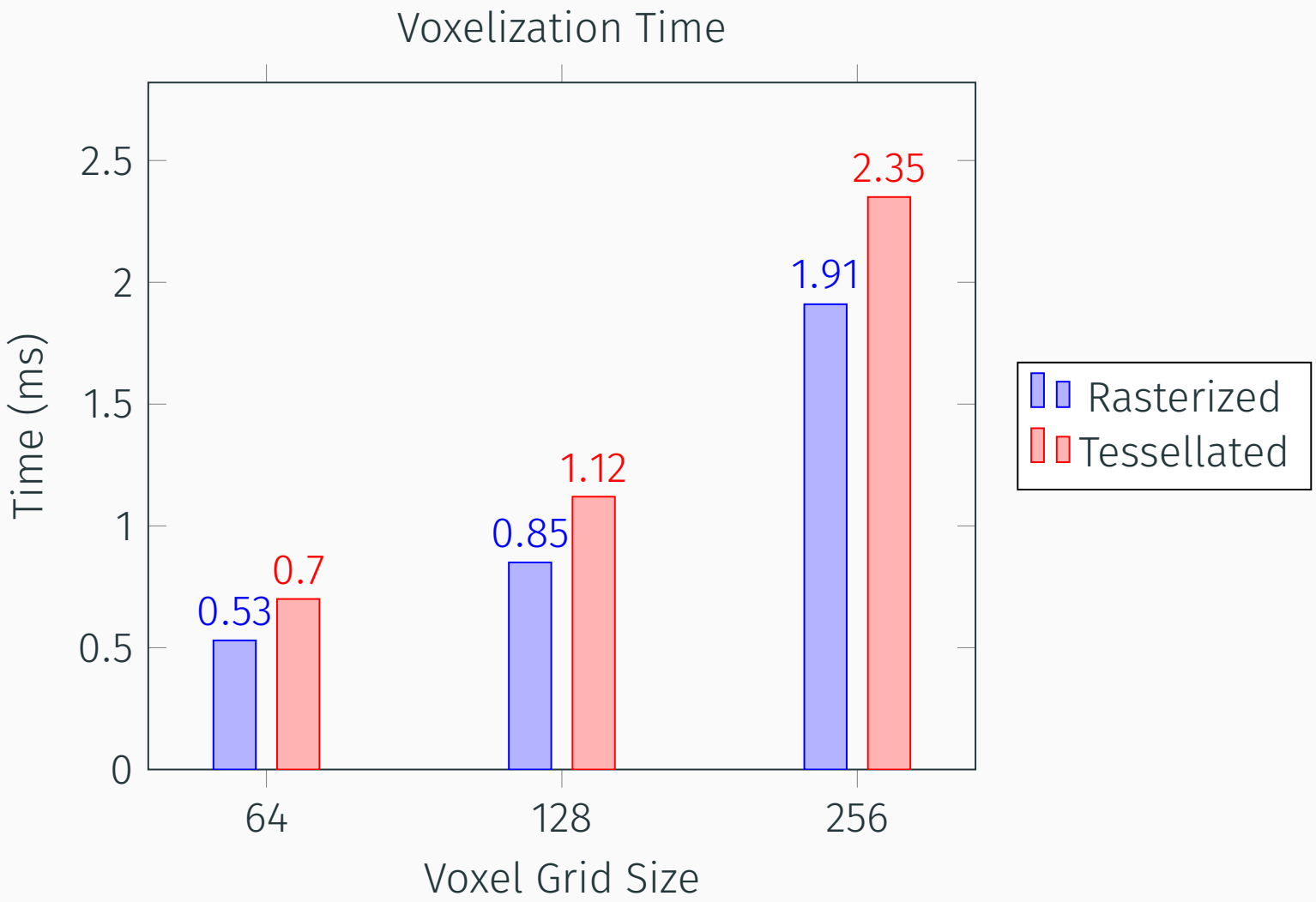
Tessellated voxels

Rasterized vs. Tessellated Voxels

Rasterized voxels

Tessellated voxels

Rasterized vs. Tessellated Voxels



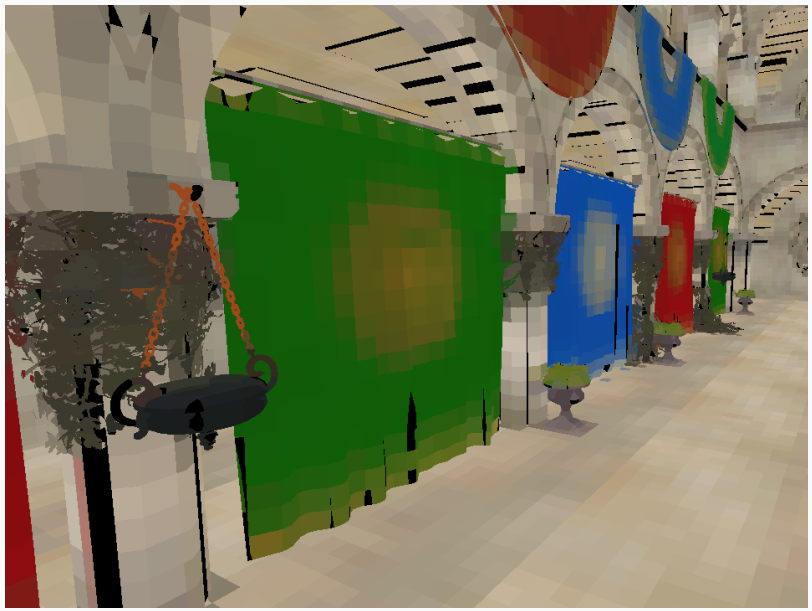
Rasterized vs. Tessellated Voxels

Summary

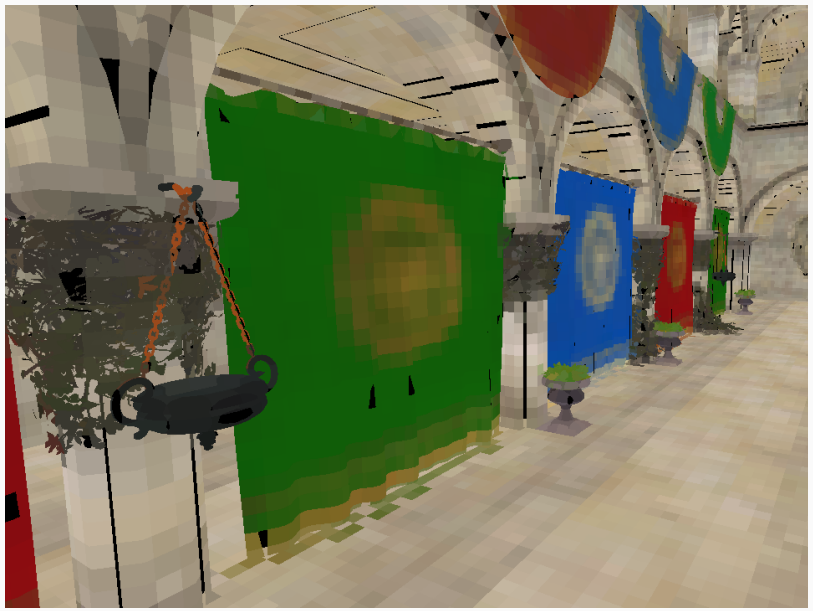
Rasterized: limited by rasterizer, slightly faster

Tessellated: limited by hardware, easier to implement

Voxel Warping

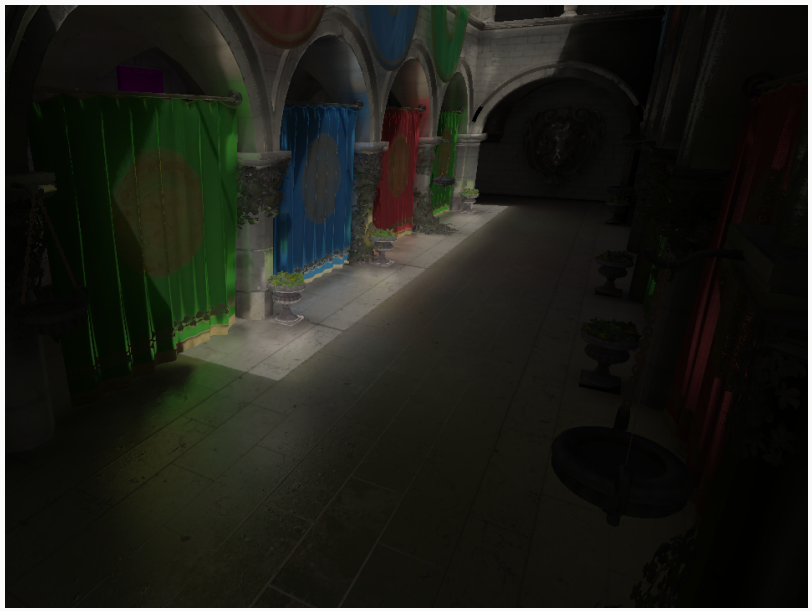


Without warping

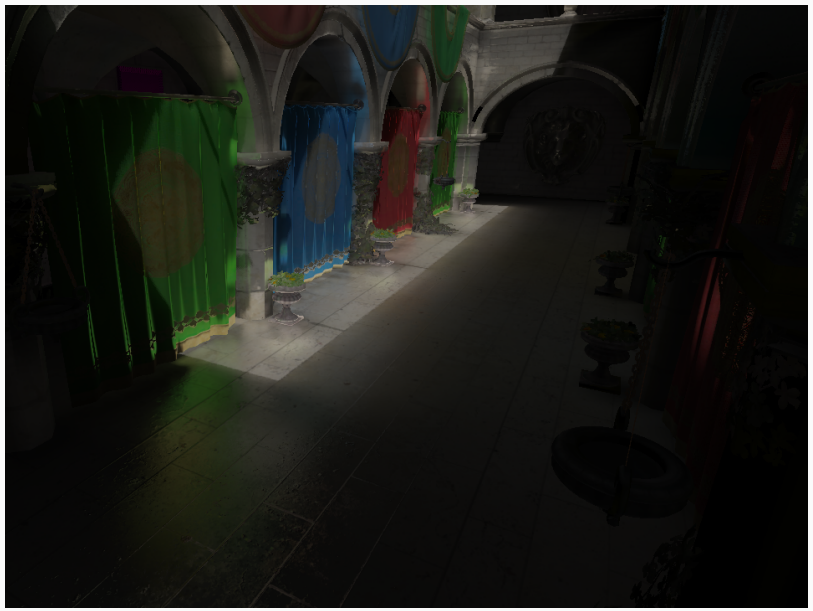


With warping

Voxel Warping

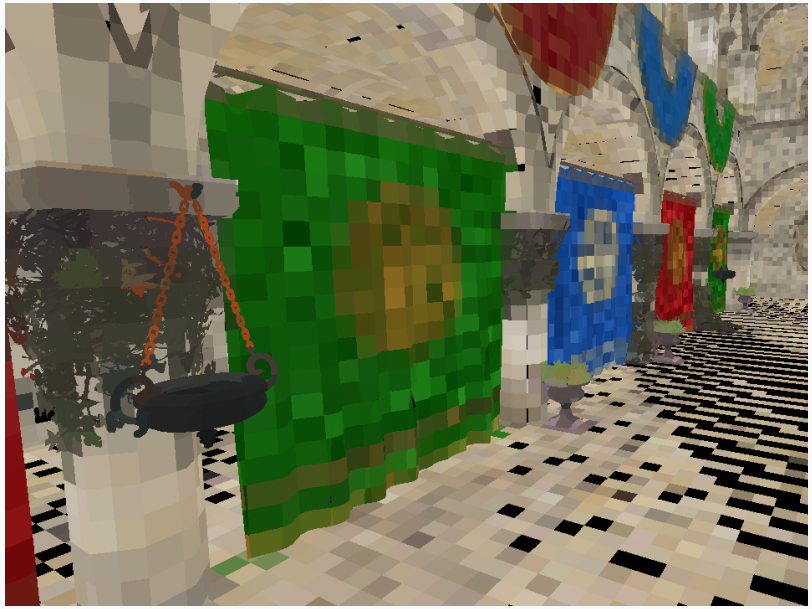


Without voxel warping

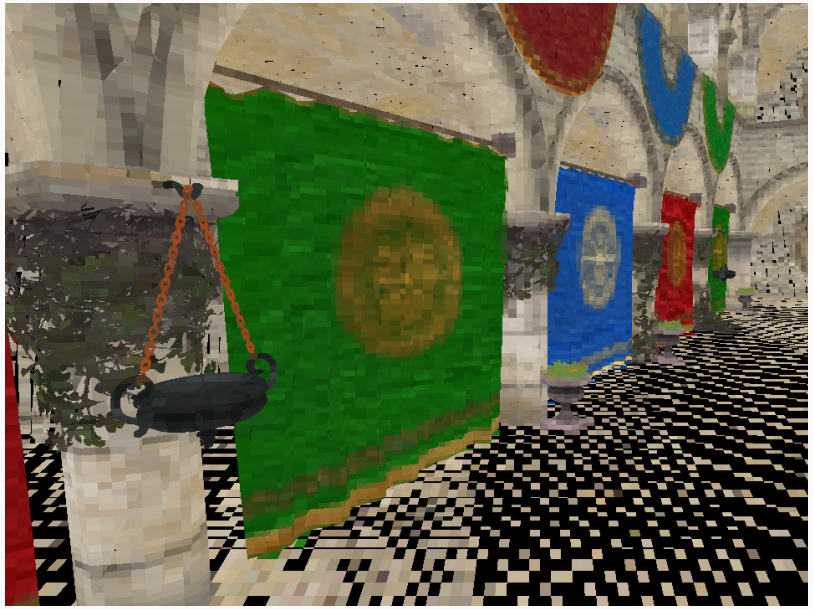


With voxel warping

Perspective Voxel Warping

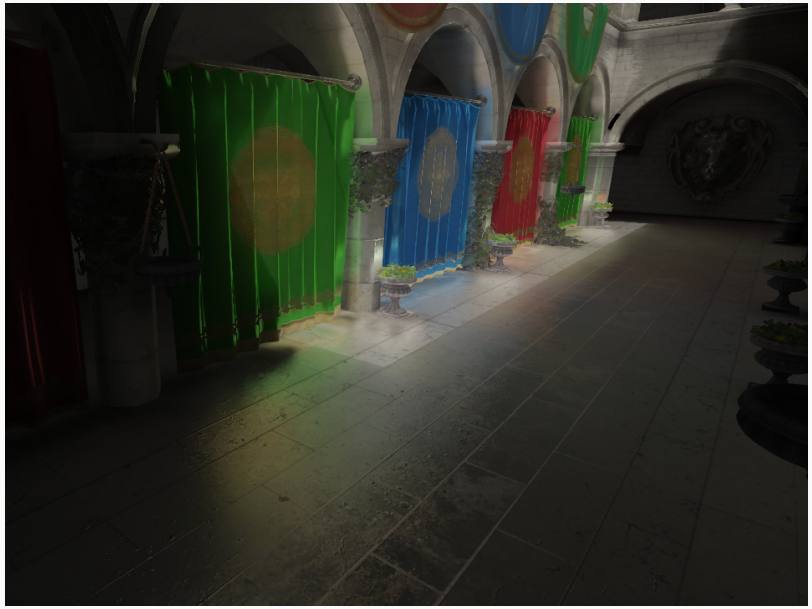


Without warping

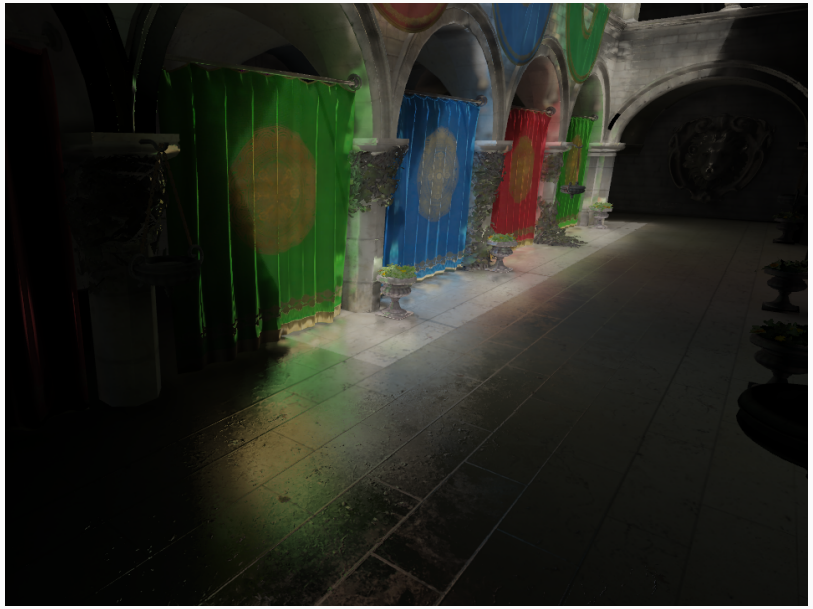


With warping

Perspective Voxel Warping



Without voxel warping



With voxel warping

Voxel Warping

- Adaptively change voxel resolution
- Bad artifacts when objects move (looking into ways to fix this)

Related Work

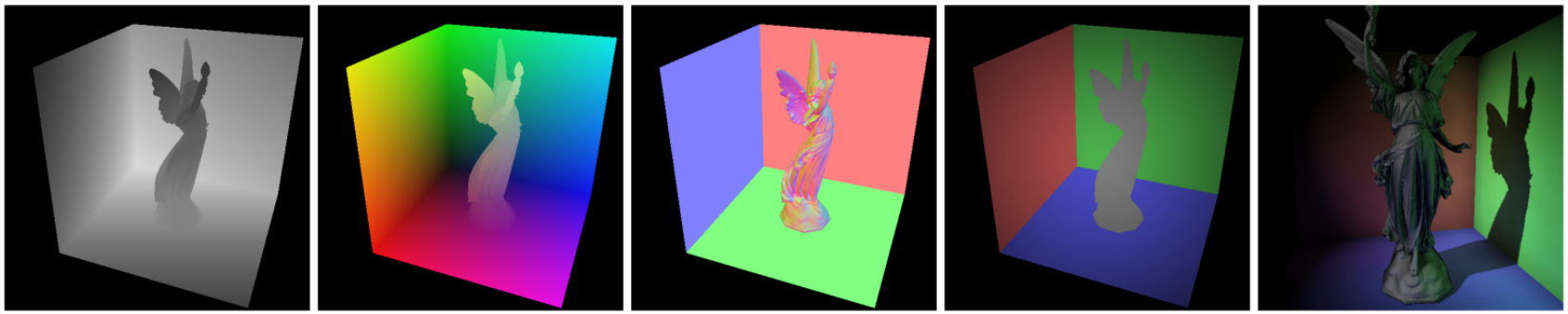
How does it compare with other methods?

Important parts of global illumination algorithms:

1. Scene representation?
2. Light computation?
3. Light sampling?

Related Work—Reflective Shadowmaps

1. Scene representation? Reflective shadowmap (RSM)
2. Light computation? None, use color and normal from RSM
3. Light sampling? Sample nearby points in RSM

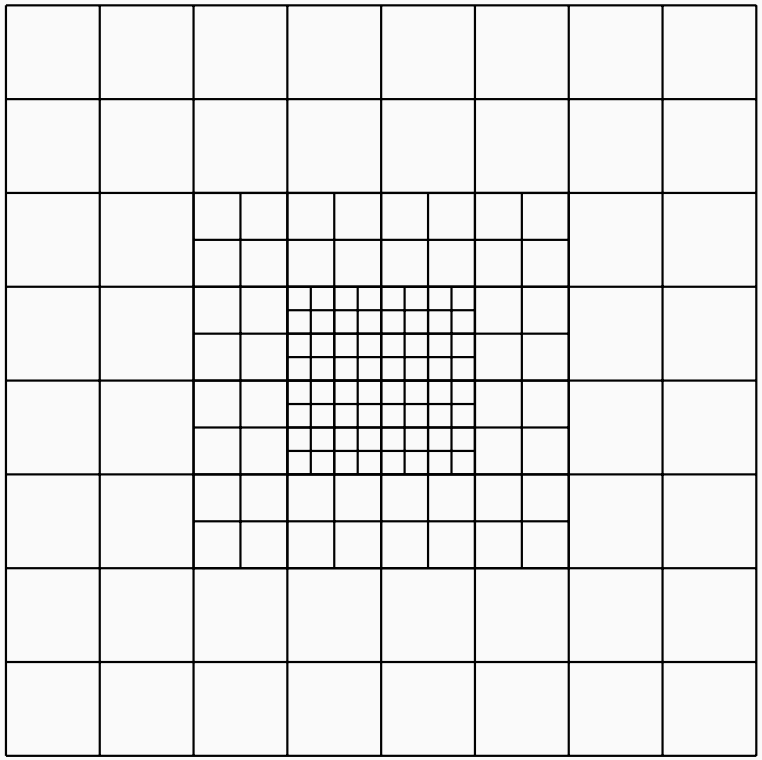


Reflective shadow map (depth, world position, normal, color) and result

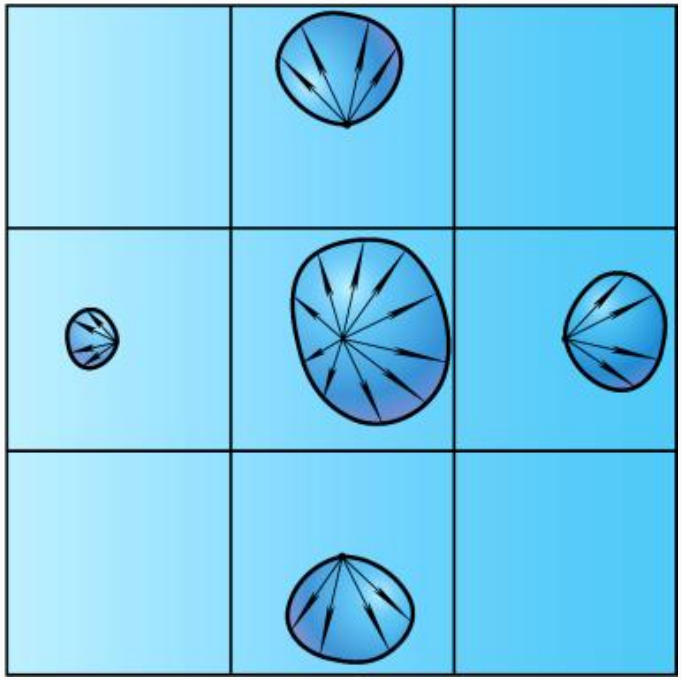
image from Reflective shadow maps by C. Dachsbacher and M. Stamminger

Related Work—Light Propagation Volumes

1. Scene representation? Cascaded voxel grid (incomplete)
2. Light computation? Iterative propagation
3. Light sampling? Texture lookup



Cascaded voxel grid



Light propagation

image from Light Propagation Volumes in CryEngine 3 by A. Kaplanyan

Related Work—Voxel Cone Tracing

1. Scene representation? Sparse voxel octree
2. Light computation? Mipmaps
3. Light sampling? Voxel cone tracing

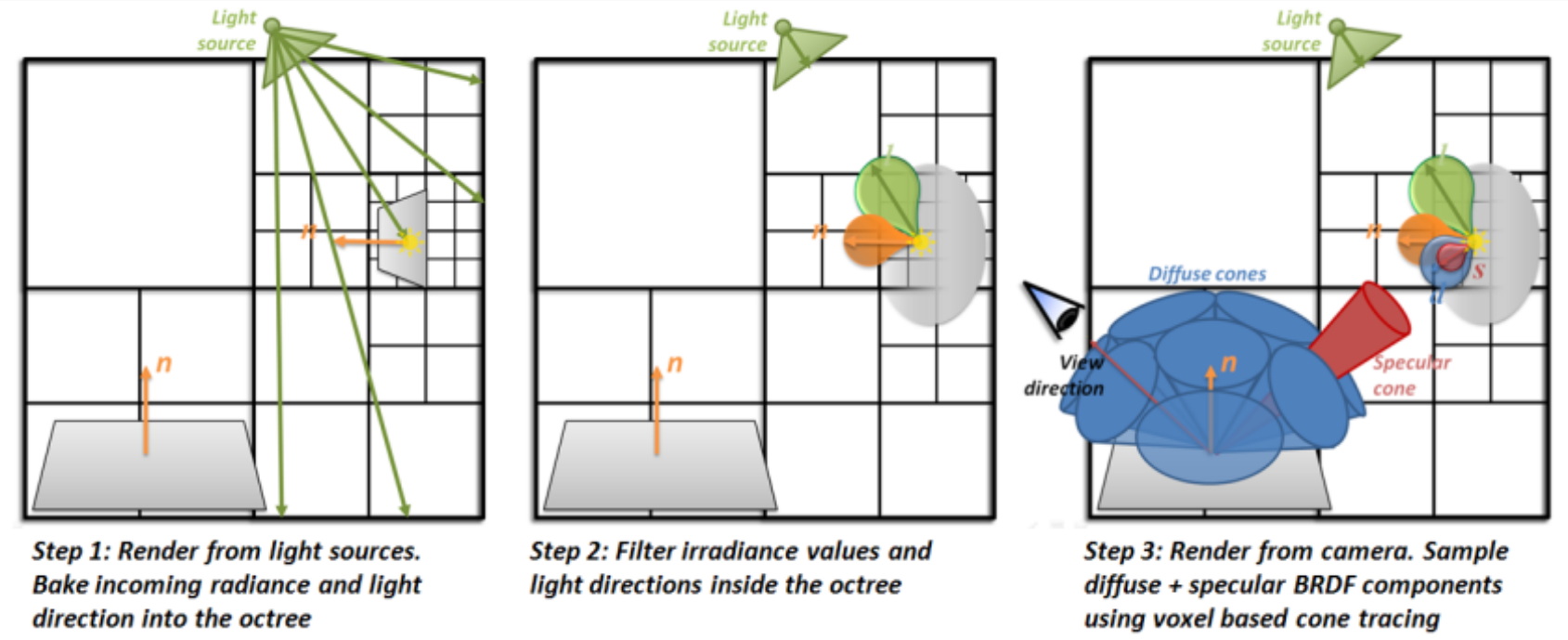


image from Interactive Indirect Illumination Using Voxel Cone Tracing by C. Crassin et al.

Conclusion

- Implementation[1] of real-time global illumination using voxel cone tracing
- Implementation and comparison of two voxelization methods
- Investigation into warped voxels

[1] Find the source here: `github.com/sfreed141/vct`

Future Work

- Try to alleviate flickering with warped voxels
- Filter using LPV method
- Cascaded sparse 3D textures
- Spherical harmonics, anisotropic filtering, adaptive cone tracing quality, other miscellaneous optimizations

Thank you!

Questions?