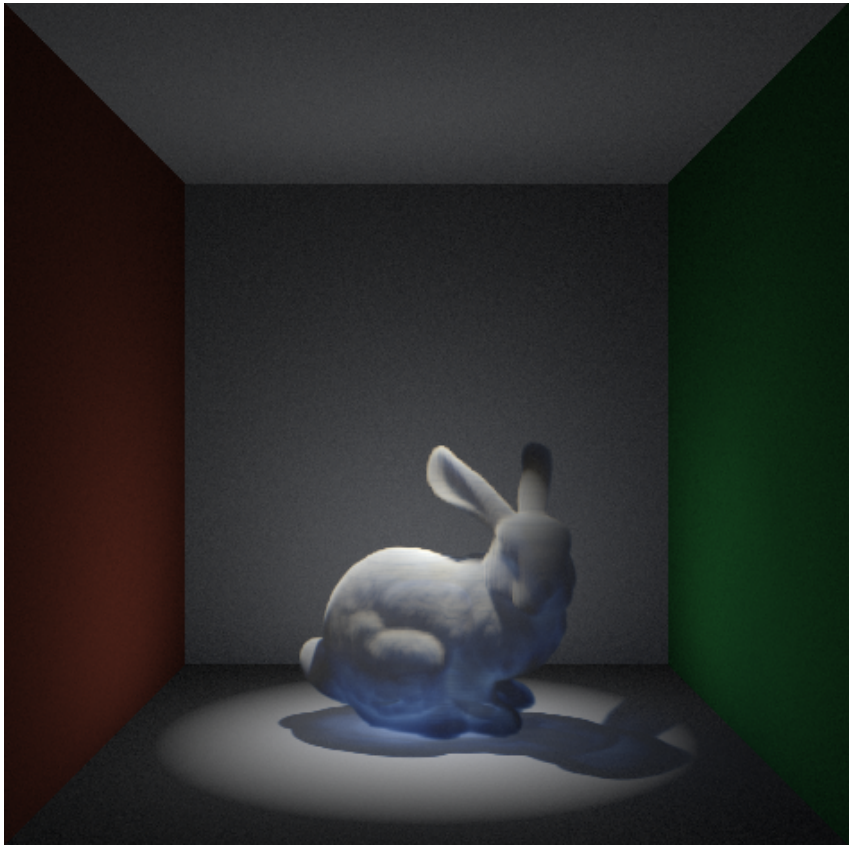
PCBEX: Point-Based Color Bleeding With Volumes Thesis Defense

Christopher James Gibson

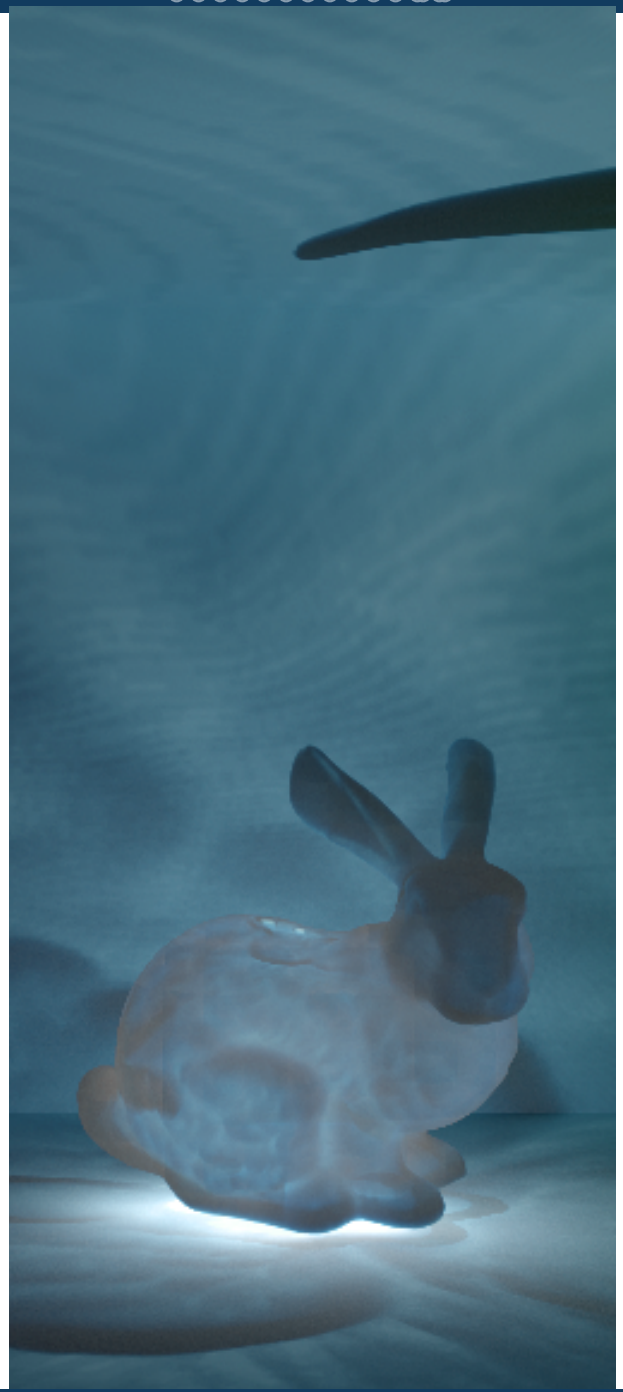
California Polytechnic University
CSC Department

June 9, 2011

Schedule



Outline



Graphics Intro

Rendering is the process of producing 2D images from a 3D scene description

At its core, all of **computer graphics** is the visualization of how light interacts in these scenes

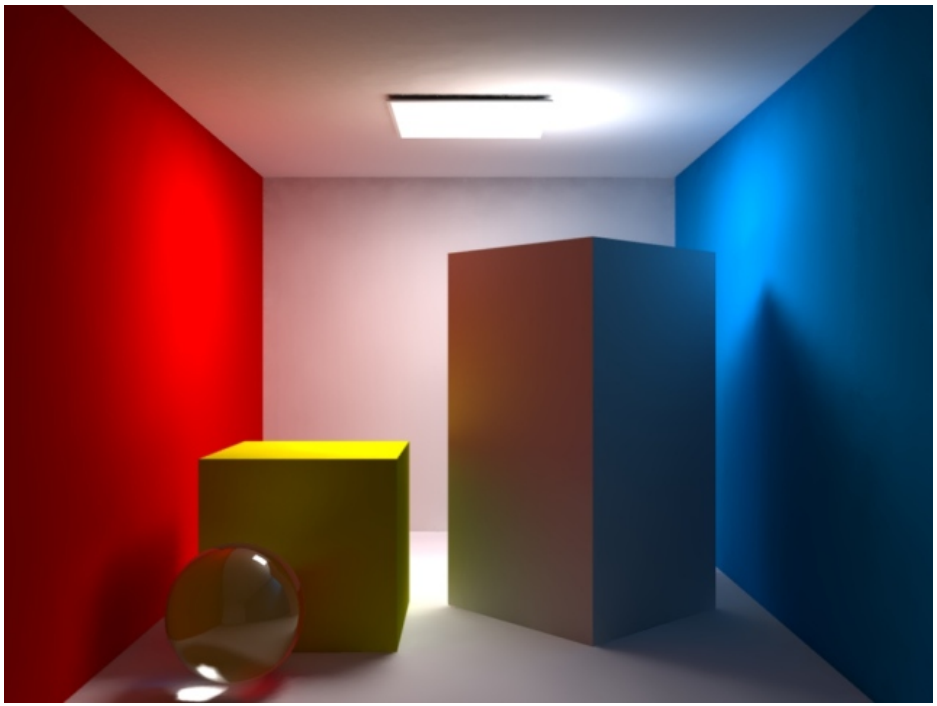
Modelling physically correct light interactions can get extremely computationally expensive

(image source: Matt Pharr)

Global Illumination Introduction

Light's interaction with various materials and mediums is complex and beautiful

Light does not simply hit surfaces, but bounces and passes through them as well

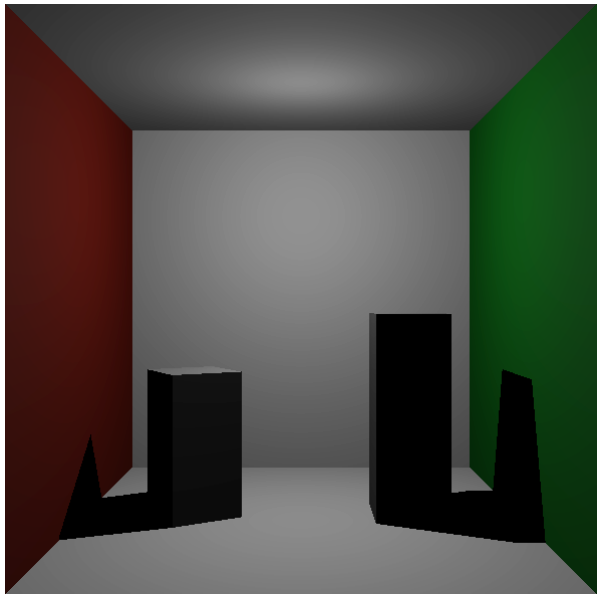


(image source: cgtutorials.com)

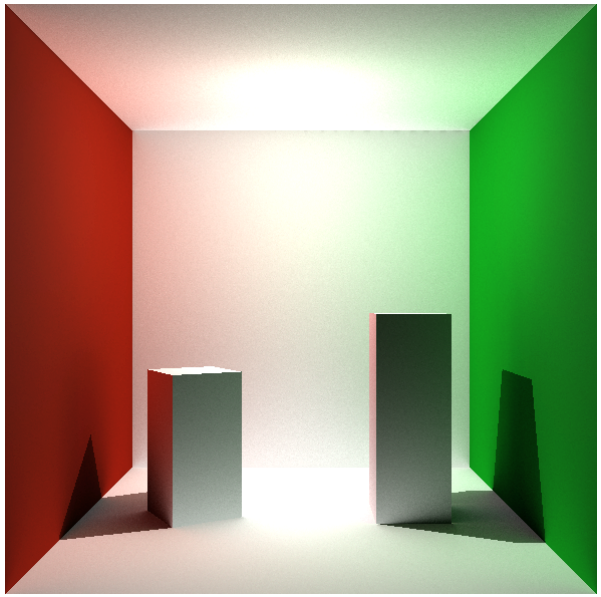
Global Illumination

Global Illumination represents the set of algorithms that estimate complex lighting systems

The results lead to richer, more realistic images



Direct Lighting



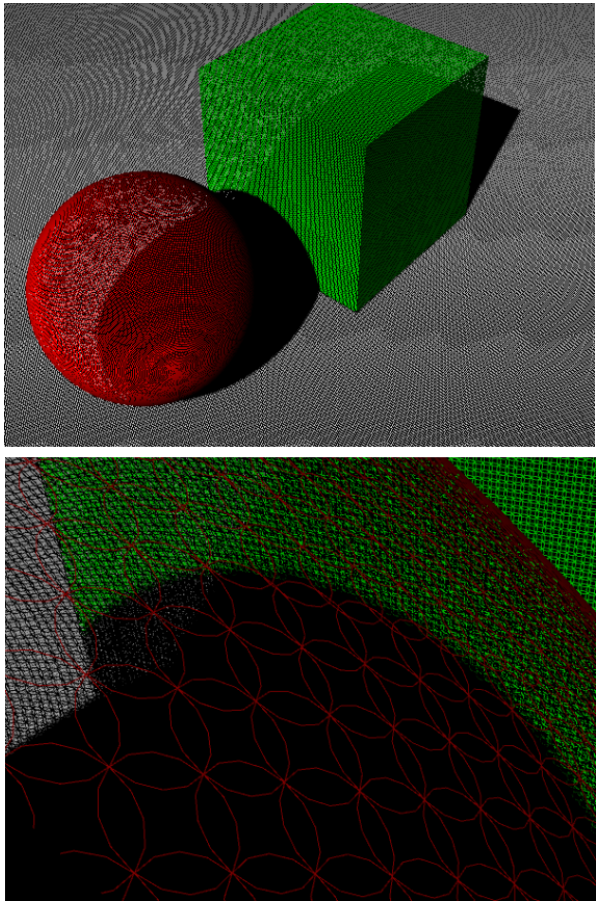
Direct Lighting & Global Illumination

Point-Based Color Bleeding (PCB)

Cheap, approximate global illumination effects using color bleeding

Utilizes direct light point cloud representation of scene's direct lighting

Already used heavily in production due to its performance advantages



Surfels in PCB

(image source: Per Christensen)

Volume Rendering

Volumes *(or participating media)* represent:

- Fog
- Smoke
- Water
- Clouds
- Other Mediums...

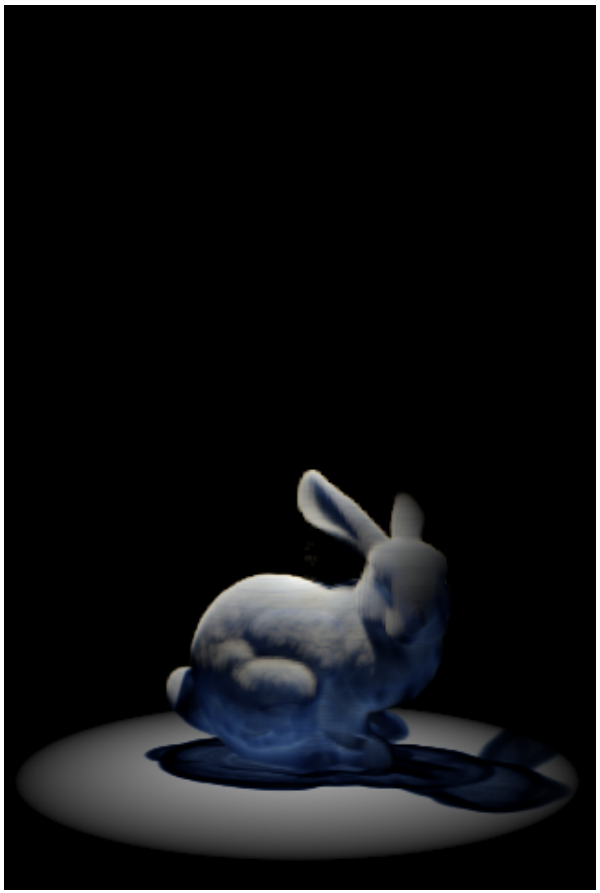
Better for visualizing three-dimensional datasets

Important to simulate and light properly for film, entertainment & medical uses

Very computationally expensive

(image source: DreamWorks Animation)

Motivation

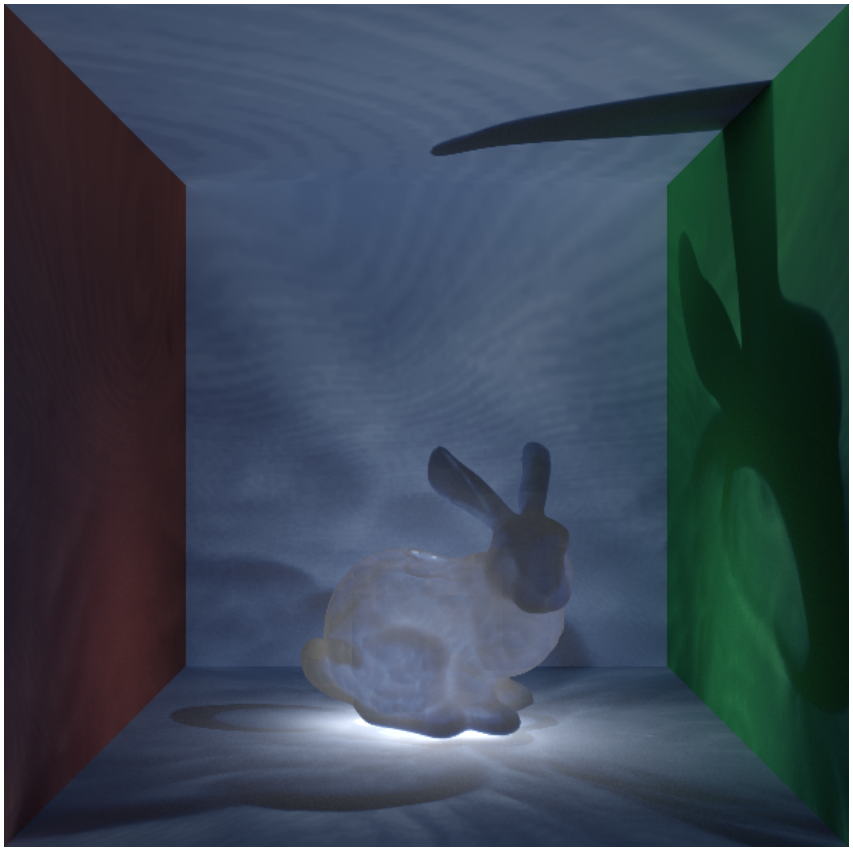


Complex volume lighting algorithms are computationally expensive and complex

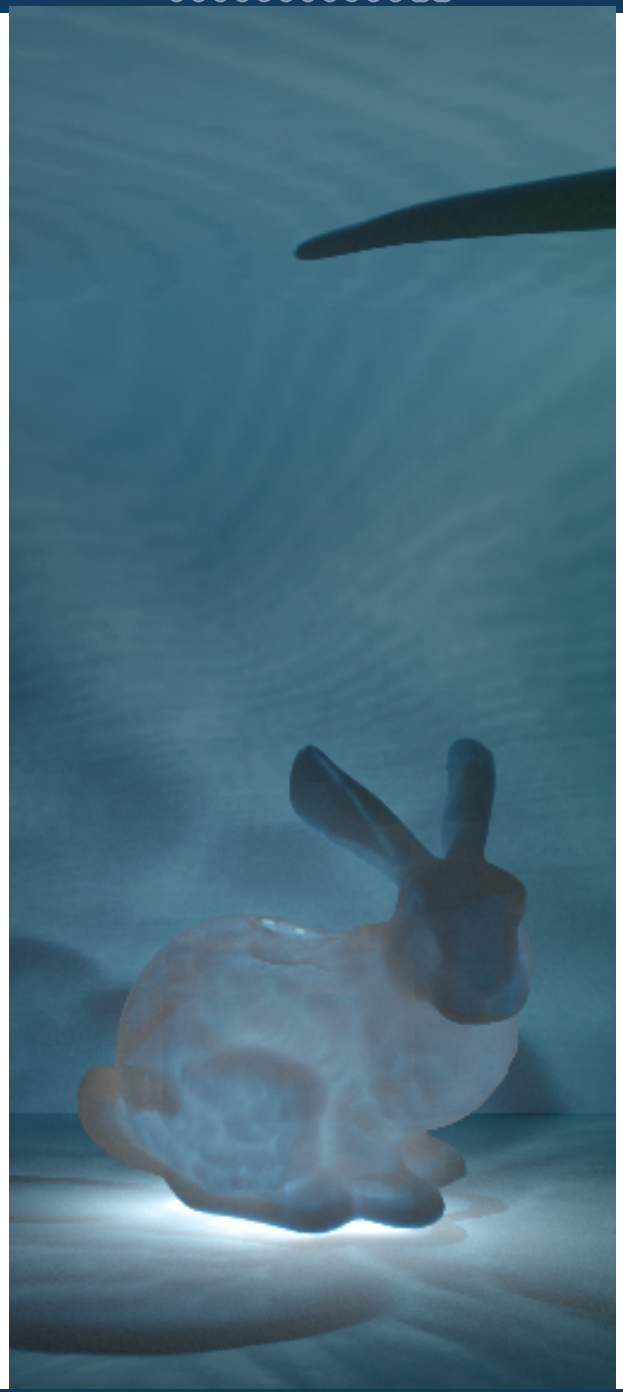
Most GA algorithms do not include volume contributions

Goals

- Modify existing PCB in order to include volumetric lighting
- Accurately model scatter, absorption and lighting properties
- Modify volume integration algorithm to add scatter-in effects
- Return comparable results with shorter overall runtimes



Outline



Related Works

Global illumination is a heavy field of research in computer graphics.

Involves many prominent technologies and algorithms:

- Photon Mapping
- Radiometry/Radiosity
- Precomuted Light Transport

Complex scene with & without global illumination

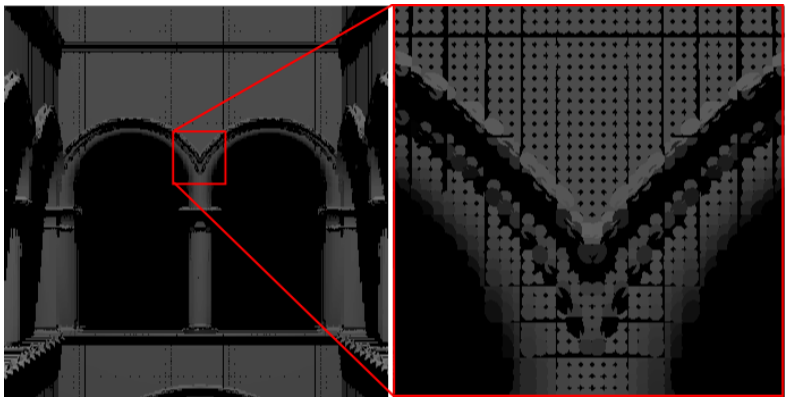
(image source: DreamWorks Animation)

Related Works: Point-Based Color Bleeding

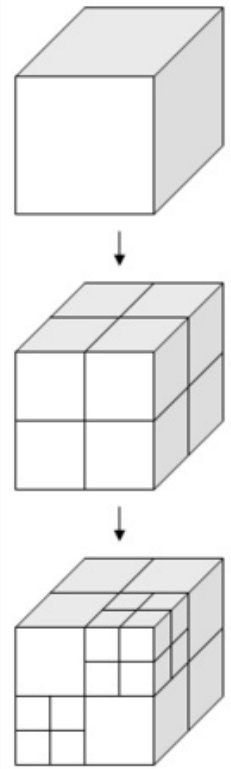
Point-Based Approximate Color Bleeding by Per Christensen.

Subset of scene geometry is thoroughly sampled, creating point cloud stored in an octree.

Point cloud is sampled to determine incoming radiance.



Point cloud representation

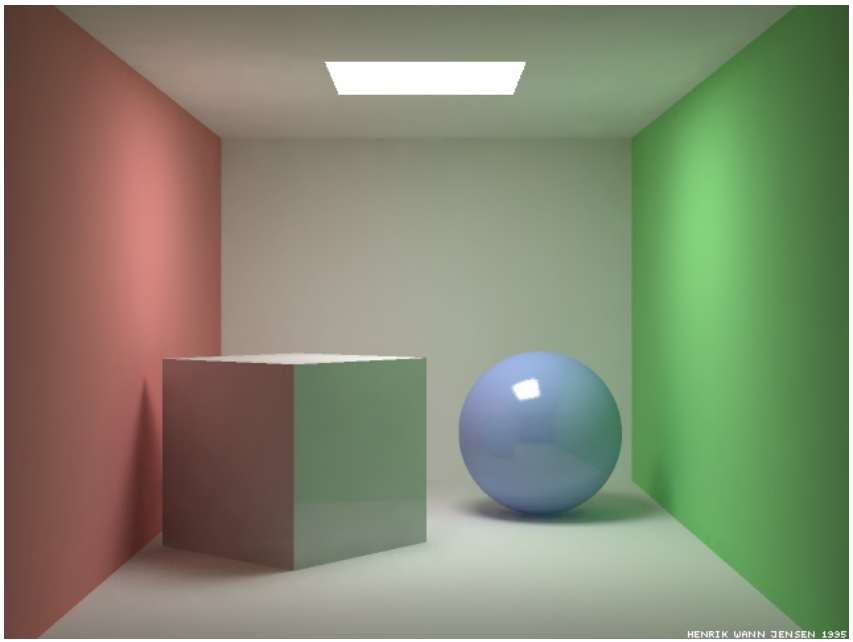


Octree subdivision

Related Works: Photon Mapping

Photons cast from lights interact with the scene. At each hit, a photon can:

- Bounce
- Pass Through
- Be Absorbed

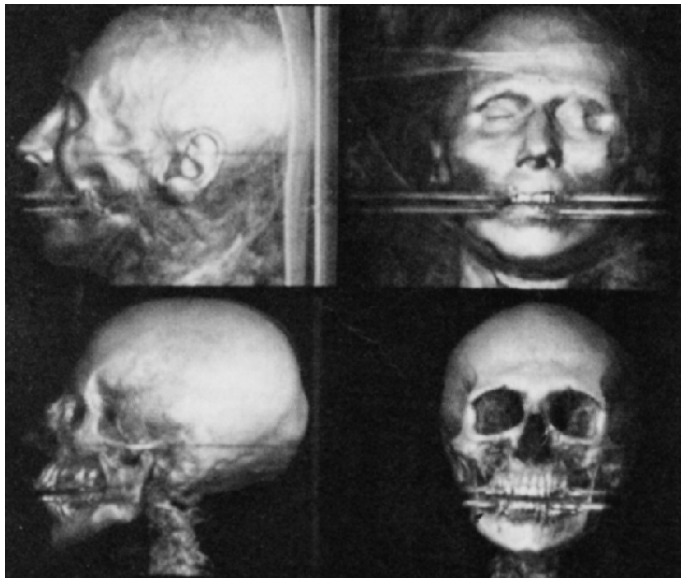


(image source: Jensen)

Related Works: Volume Rendering

Seminal research involved number of approaches...

- Polygonalization of straight voxels *(boxy)*
- Polygonal representation based on isosurfaces
- Opacity/Color arrays *(interpolation across voxels)*



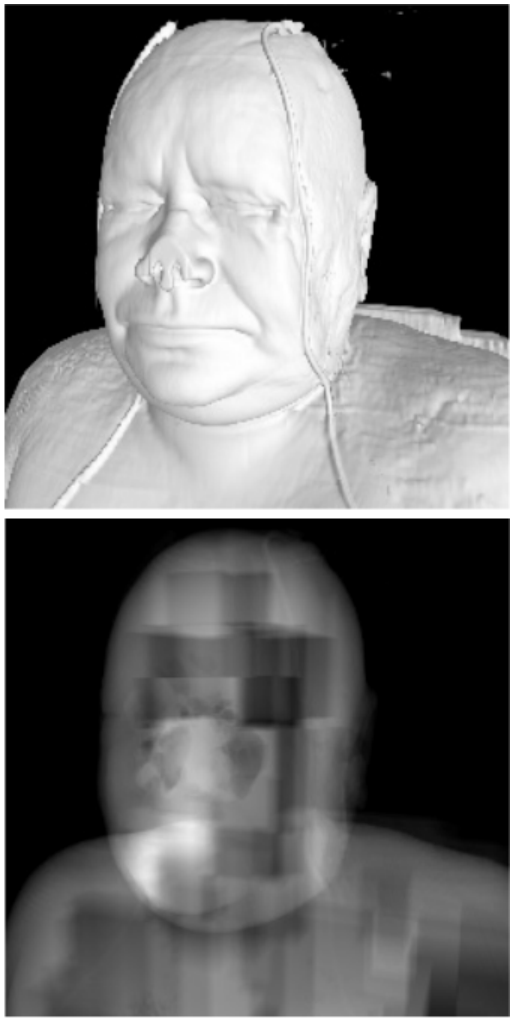
Renders of CT scan data

(image source: Marc Levoy)

Related Works: Volume Rendering

Multi-resolution volumes help save on memory and allow for on-demand loading

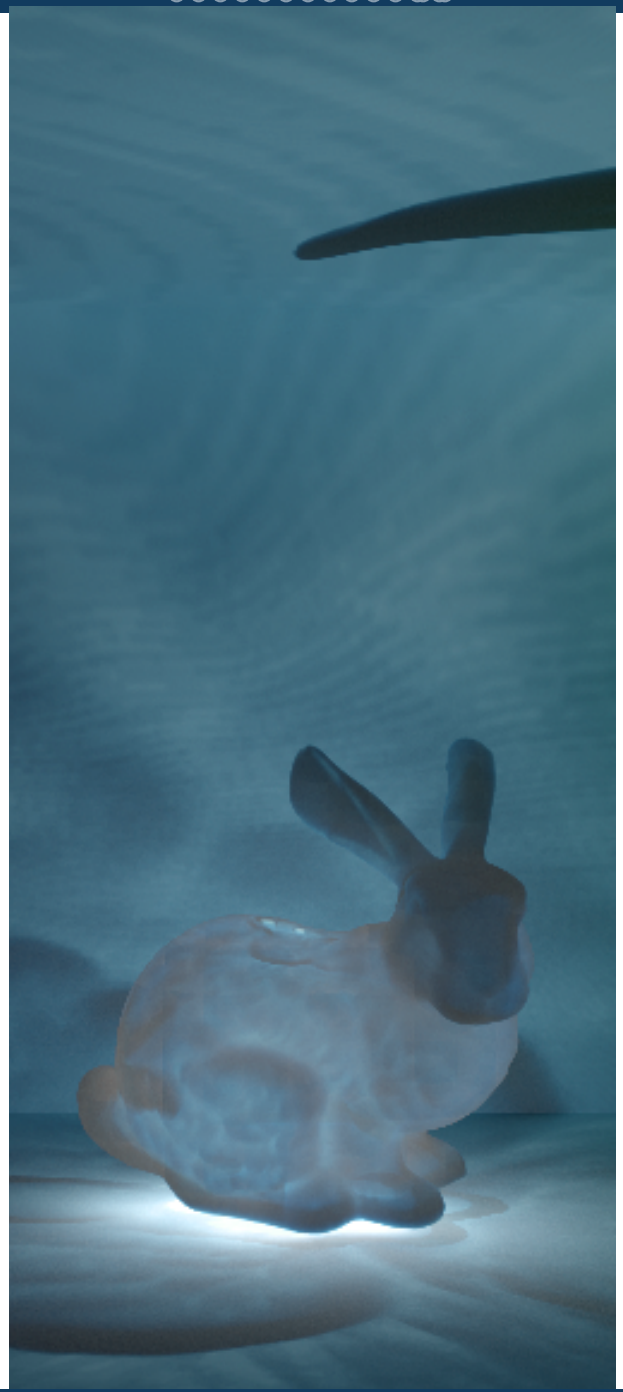
Occlusion techniques involving volume acceleration structures help avoid extraneous computations



Renders of CT scan data

(image source: S. Guthe)

Outline



Flux and Radiance

Flux is the measure of total light emitted *(Watts)*

Radiance Equation

$$L = \frac{d^2\,\Phi}{dw\;dA^{\perp}} = \frac{d^2\,\Phi}{dw\;dA\cos\theta}$$

Radiance Invariance

$$L(x \to y) = L(y \to x)$$

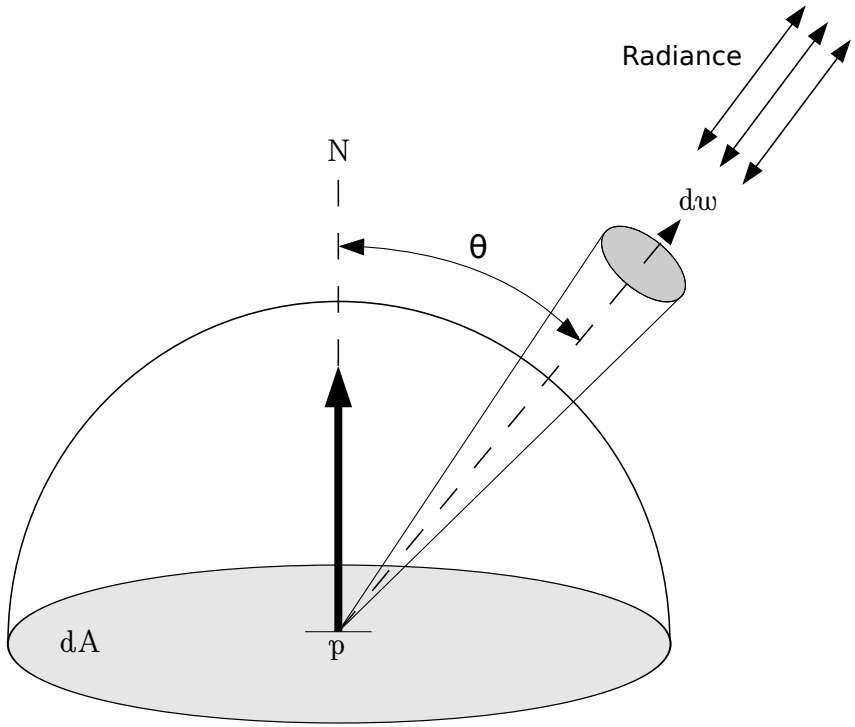
Irradiance

Irradiance Equation

$$E = \frac{d\Phi}{dA}$$

Taking radiance into account:

$$E = \int L(\mathrm{p} \leftarrow w)\cos\theta dw$$



Radiance diagram

BRDF

Bidirectional **R**eflectance **D**istribution **F**unction

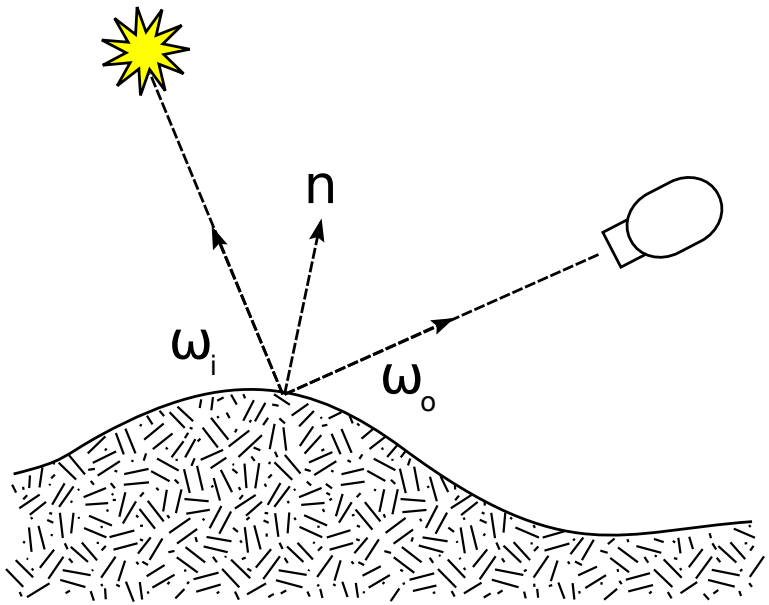
Gives us a formalism for describing the reflection from a surface

Incoming Radiance from w_i

$$dE(\mathsf{p}, w_i) = \int L_i(\mathsf{p} \leftarrow w_i) \cos\theta_i dw_i.$$

Incoming vs Outgoing Radiance

$$f_r(\mathsf{p}, w_o, w_i) = \frac{dL_o(\mathsf{p}, w_o)}{dE(\mathsf{p}, w_i)}$$



BRDF Visualization

(image source: Wikipedia)

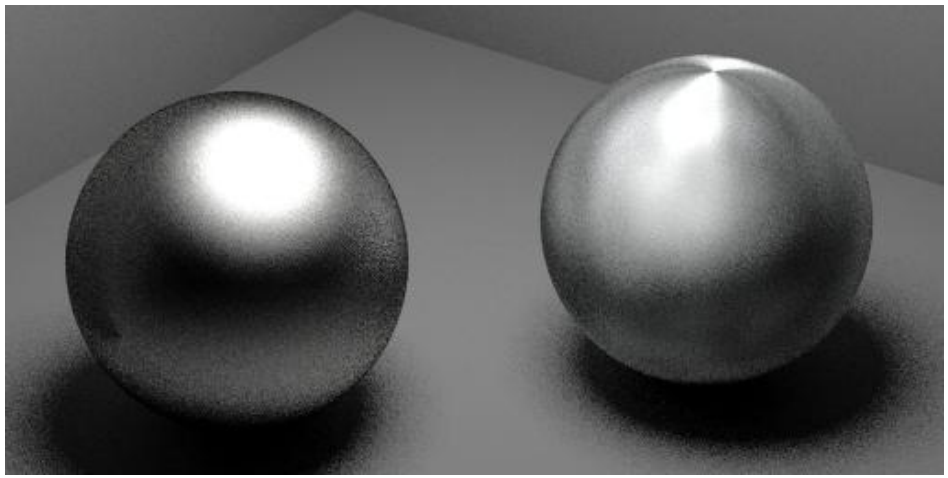
BRDF

Note: f_r is a function based on the reflection properties of the surface

Therefore, we can solve for L_o

Incoming Radiance from w_i

$$L_o(\mathsf{p}, w_o) = \int \mathbb{S}^2 f_r(\mathsf{p}, w_o, w_i)\, L_i(\mathsf{p}, w_i) cos\theta_i dw.$$



BRDF Visualization

(image source: Cornell.edu)

BSSRDF

Bidirectional **S**cattering-**S**urface **R**eflectance **D**istribution **F**unction

Helps model the interactions of light within a surface *(also known as subsurface scattering.)*

Far more complex than traditional BRDF evaluations.

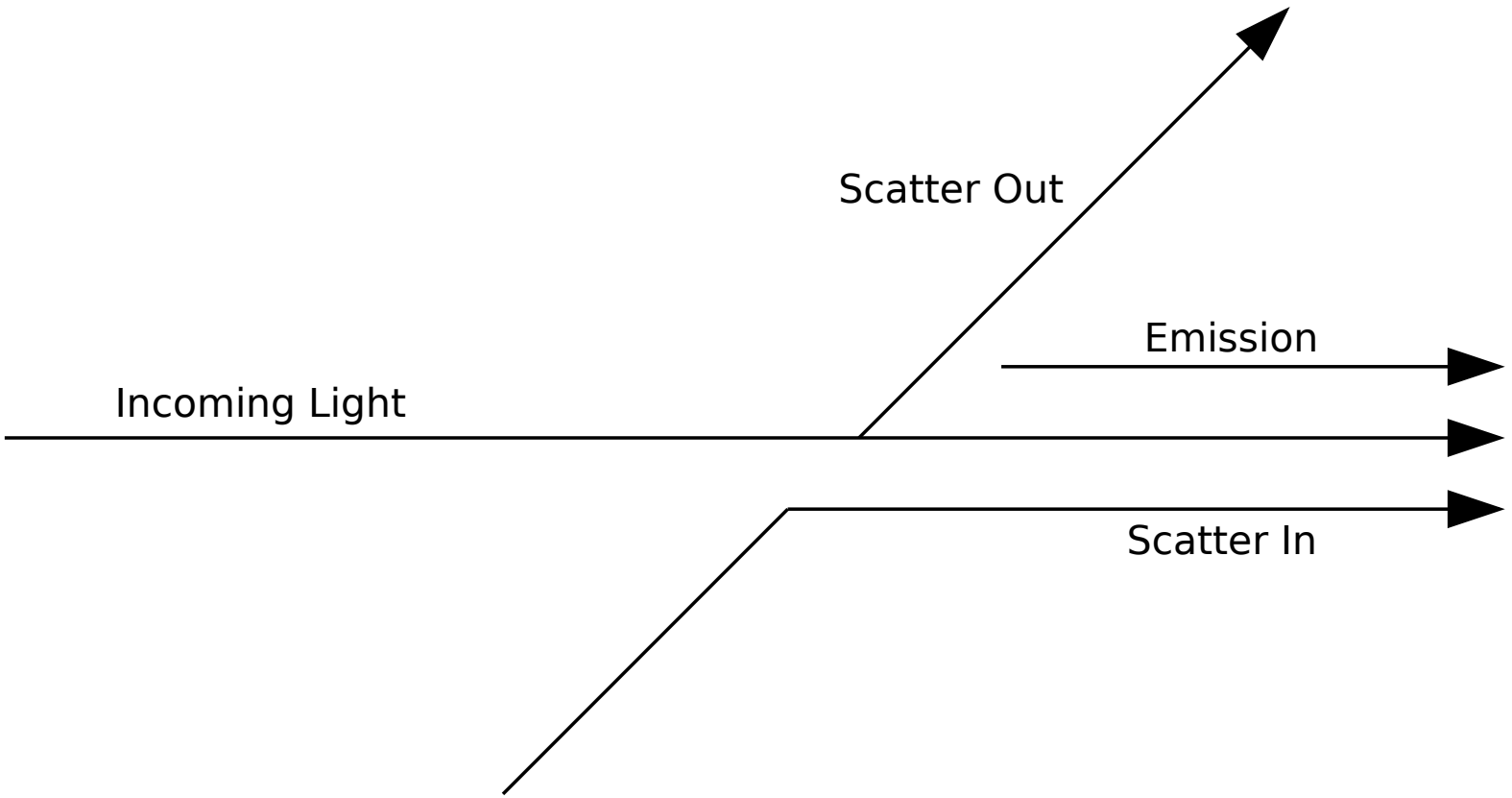
Incoming vs Outgoing Radiance

$$f_r(\mathrm{p}_o, w_o, \mathrm{p}_i, w_i) = \frac{dL_o(\mathrm{p}_o, w_o)}{dE(\mathrm{p}_i, w_i)}.$$

Light scattering inside of a statue

(image source: Nvidia)

Volume Lighting



Volume Lighting - Absorption

Absorption Equation

$$e^{-\int_0^d \sigma_a(p+tw,w)dt}$$

Some particles will absorb light passing through. Also known as extinction.

Most commonly seen in smoke or ash clouds

Extinction in a volume

(image source: Matt Pharr)

Volume Lighting - Scatter

Scatter Out Equation

$$dL_o(\mathrm{p}, w) = -\sigma_s(\mathrm{p}, w) L_i(\mathrm{p}, -w) dt$$

Scatter out is the reduction of light along a path as it is scattered by the participating media.

Clouds exemplify high scatter and low absorption.

Scatter in a volume

(image source: blenderartists.org)

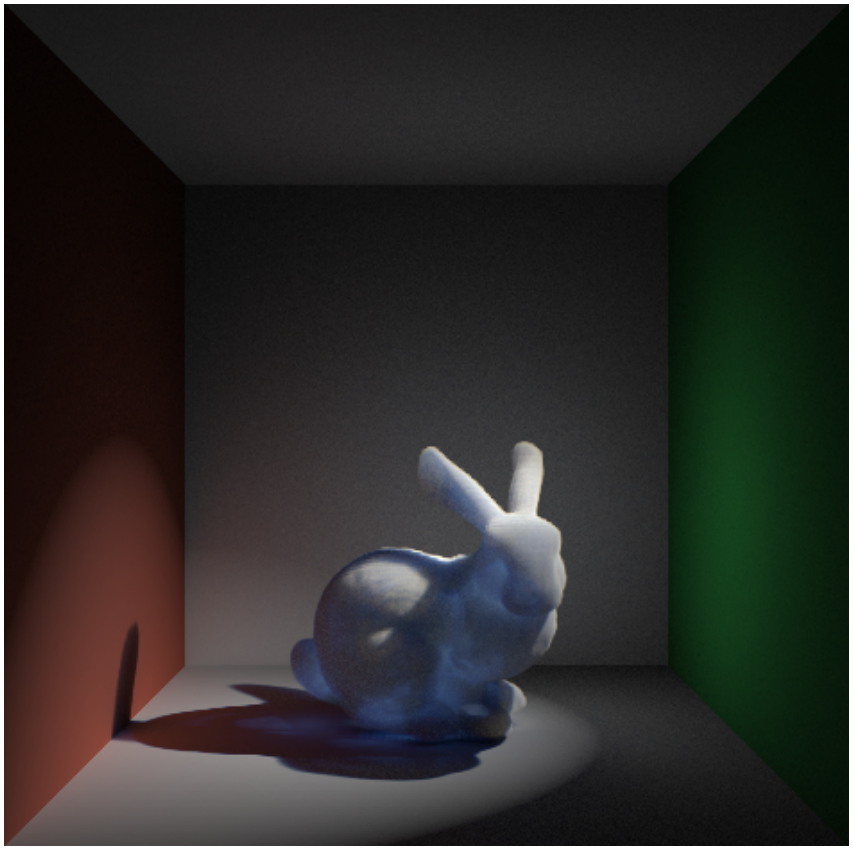
Volume Lighting- Transmittance

Transmittance Equation

$$T_r(\mathsf{p} \rightarrow \mathsf{p}') = e^{-\int_0^d \sigma(p+tw,w)dt}.$$

Evaluates how much light was left unabsorbed after passing through the volume.

Helps us estimate how much light bleeds through volumes from behind.

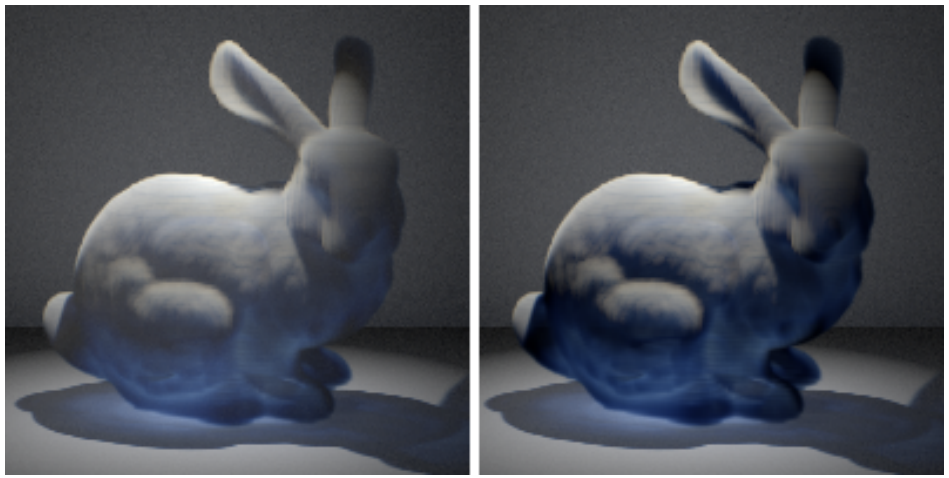


Light passing through the bunny volume

Volume Lighting - Scatter-In

Scatter-In Equation

$$S(\mathsf{p}, w) = L_{\mathsf{ve}}(\mathsf{p}, w) + \sigma_{\mathsf{s}}(\mathsf{p}, w) \int_{\mathbb{S}^2} phase(\mathsf{p}, -w' \rightarrow w) L_i(\mathsf{p}, w') \mathsf{d}w'.$$



Scatter-in *(left)* and only direct lighting via transmittance *(right)*

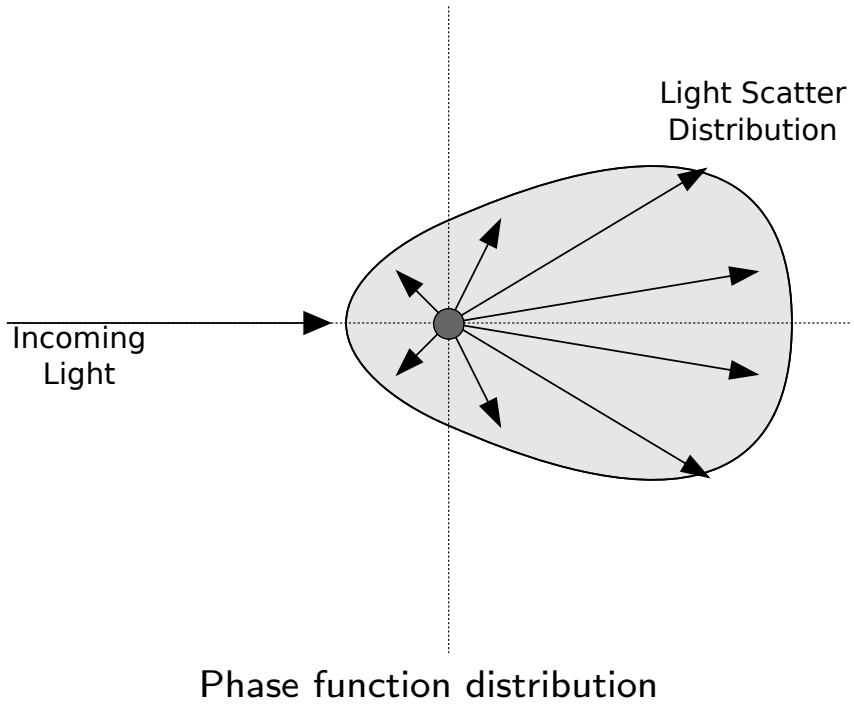
Volume Lighting- Phase Functions

Phase Function

Described as $phase(w \rightarrow w')$

Describes the probability that light will scatter in any given direction.

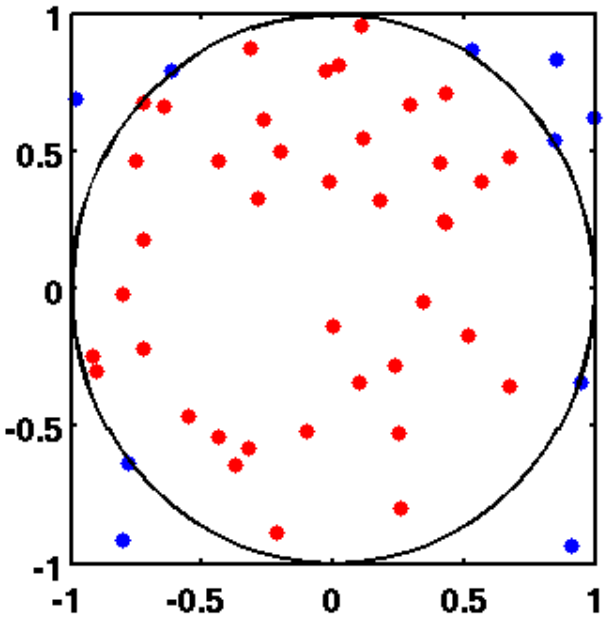
Used to estimate total scatter-in contribution.



Monte Carlo Integration

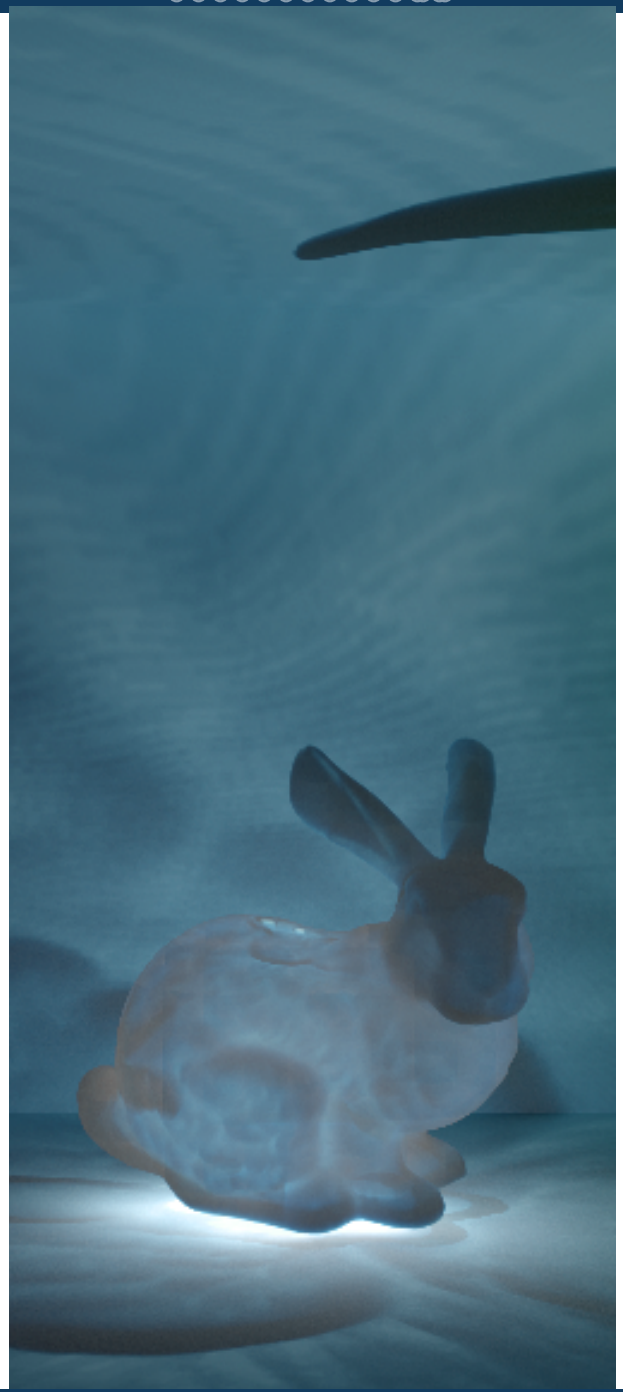
Monte Carlo methods allow estimation of complex systems through use of probability functions and random numbers.

Most useful to us is Monte Carlo Integration.



Randomly sampling a circle

Outline



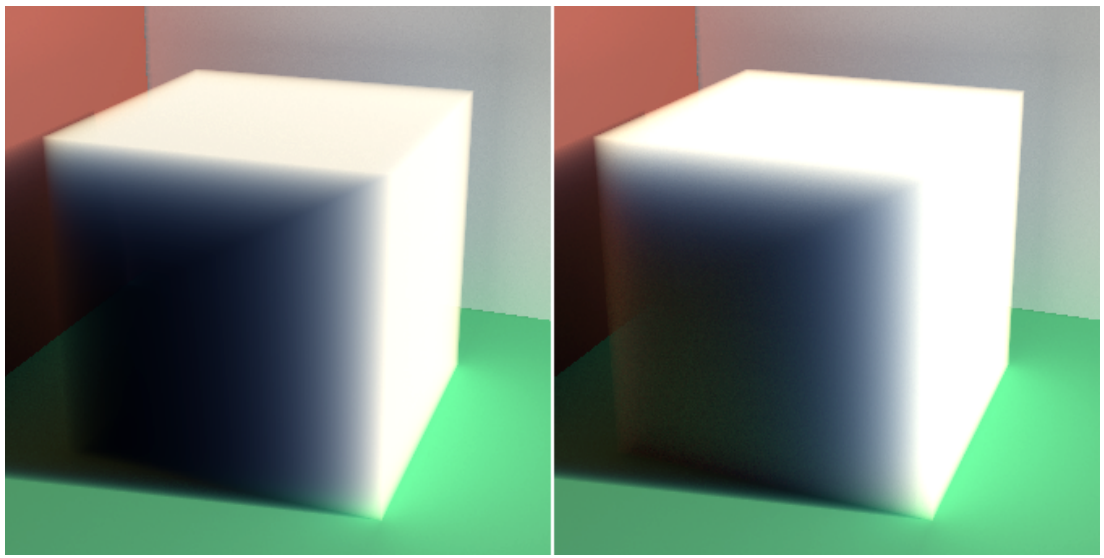
Point-Based Color Bleeding

1. Sample the scene and generate a point cloud
2. Perform normal ray tracing
3. Replace ambient estimates with a gather stage using surrounding point cloud

(image source: Disney)

Extension Overview

1. Sample the scene and generate a point cloud
2. Sample the participating media to evaluate scatter, absorbtion and direct lighting
3. Perform normal ray tracing
4. Replace ambient estimates with a gather stage using surrounding point cloud using a modified traversal algorithm
5. Model and scatter-in properties during volume integration

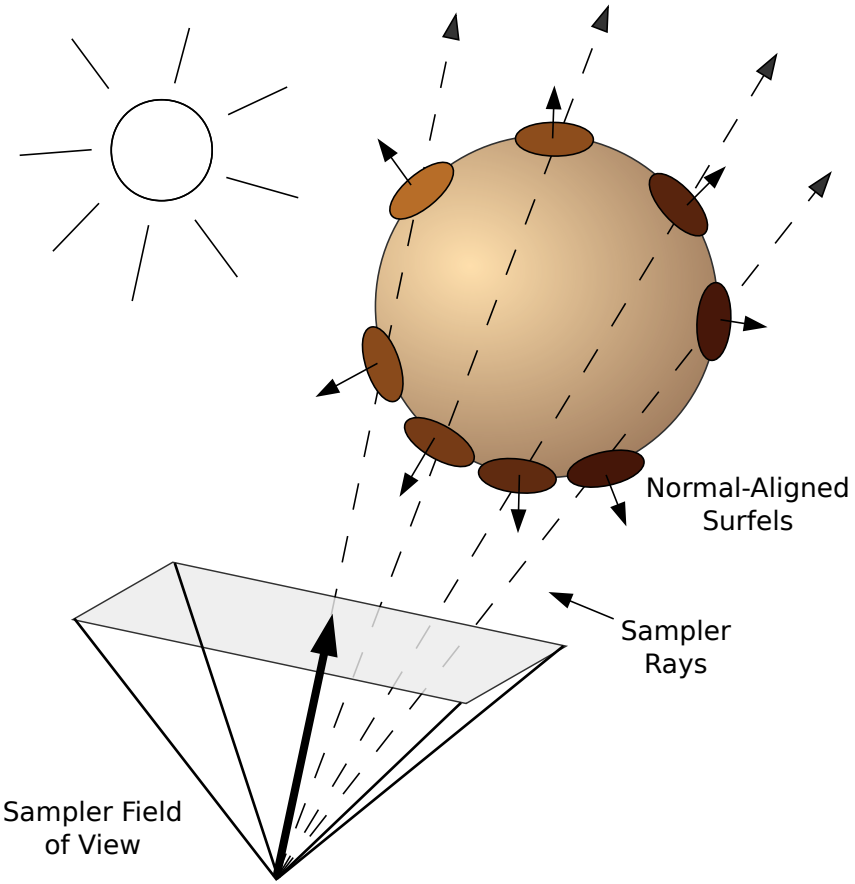


Sampling the Scene

Rays are cast and intersections with objects are recorded

Sampling "camera" is behind normal camera with wider field of view

Surfels are oriented at the intersections aligned along the surface normal

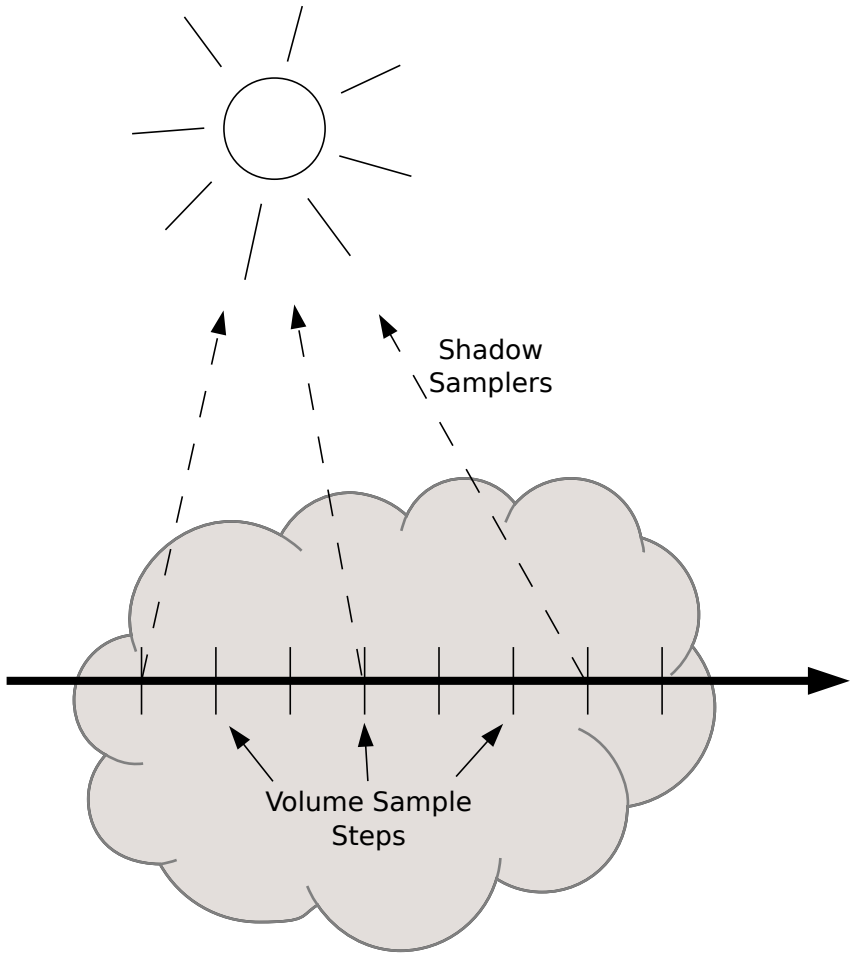


Sampling Volumes

Samples are taken at discrete steps across the volume domain

Scatter/Absorbtion/Lighting factors are averaged within each step

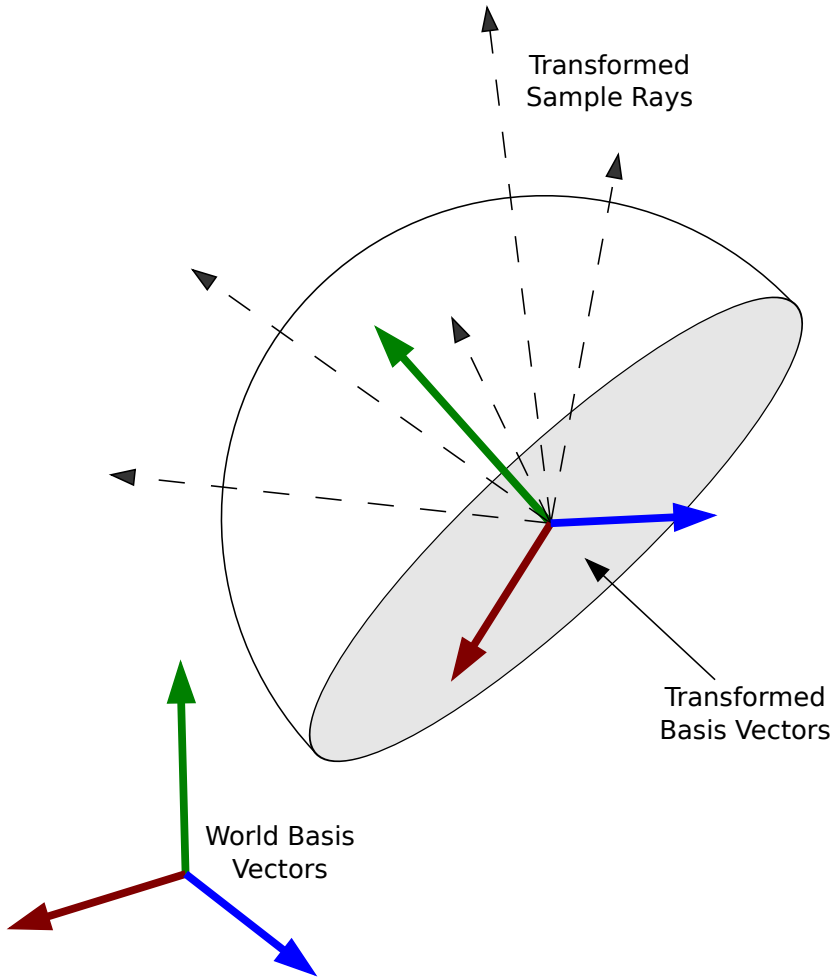
Lvoxels (spheres) instead of *surfels*



Gathering Light

At render time, rays are cast from normal camera

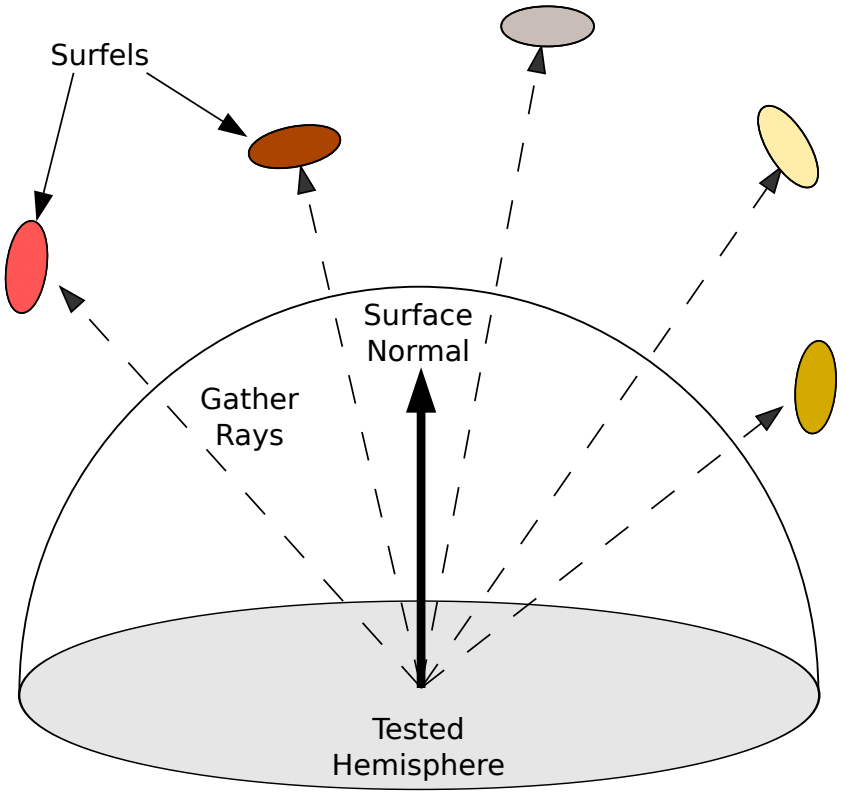
Samples are cast from intersecting surface oriented along its normal



Gathering Light

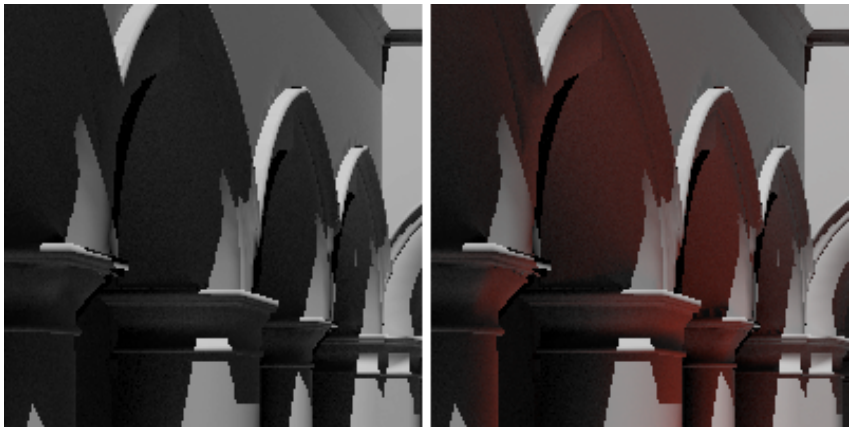
Samples cast into point cloud "gather" light via intersection

Samples are returned and used to evaluate the integral over the aligned hemisphere



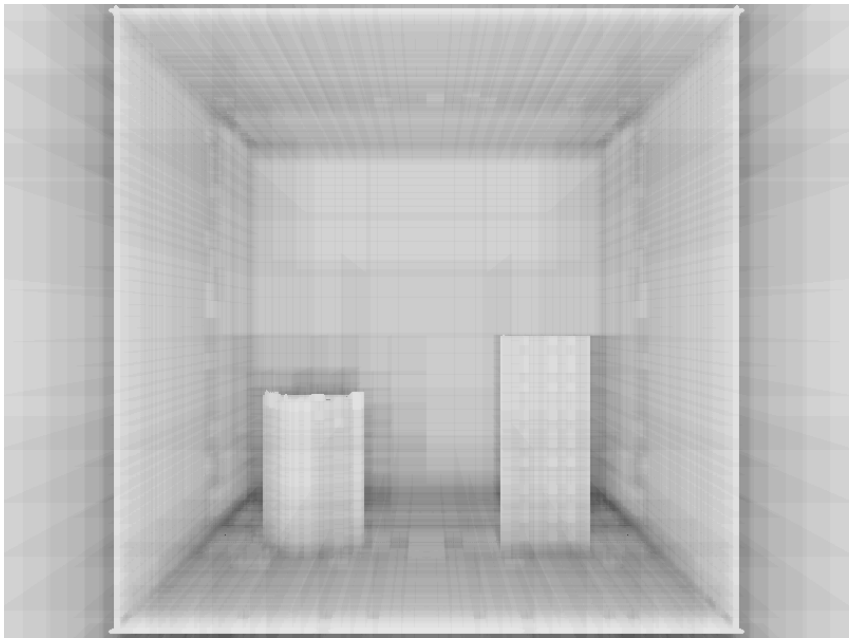
Integrating Volume Data - Scatter Out

Gather stage remains same, no changes necessary



Octree must be traversed from back-to-front in order to integrate properly

Must keep track of transmittance during traversal

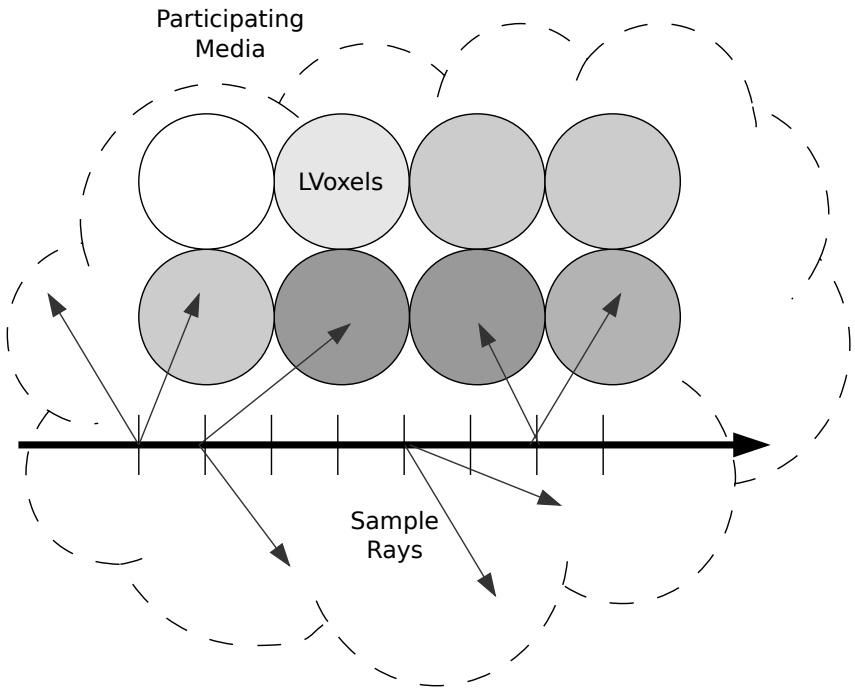


Integrating Volume Data - Scatter In

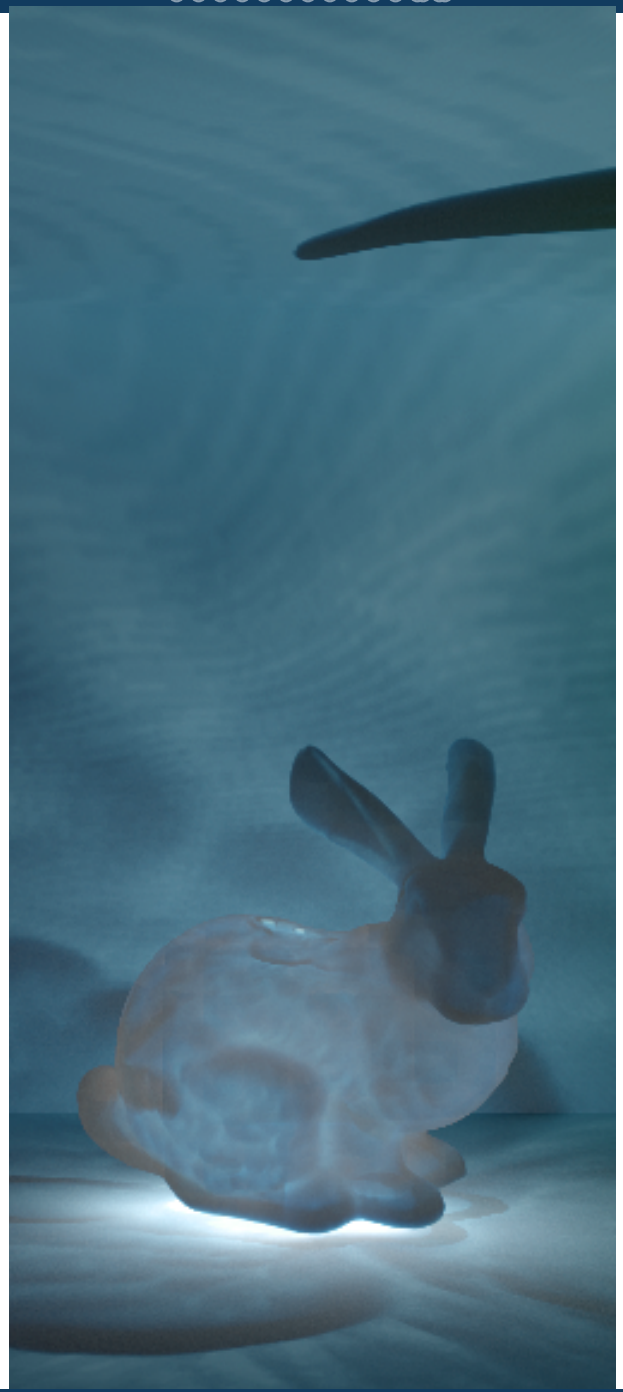
Modify integration code (normal volume rendering)

Add spherical sampling at each volume step

For increased performance, guarantee full distribution across multiple steps, not at each step



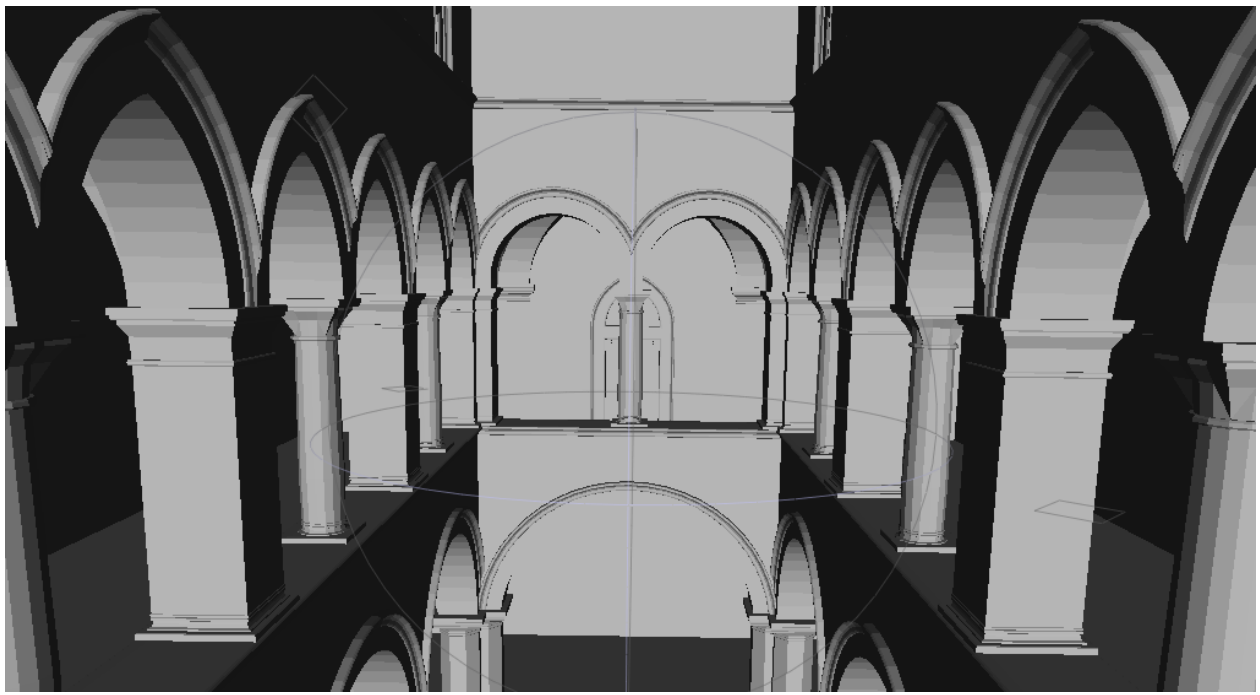
Outline



Results - Environment

Test scene involved:

- 60,000 triangle Sponza Atrium Model
- Stanford Bunny *(Volume Data)*



Results

Test run configuration:

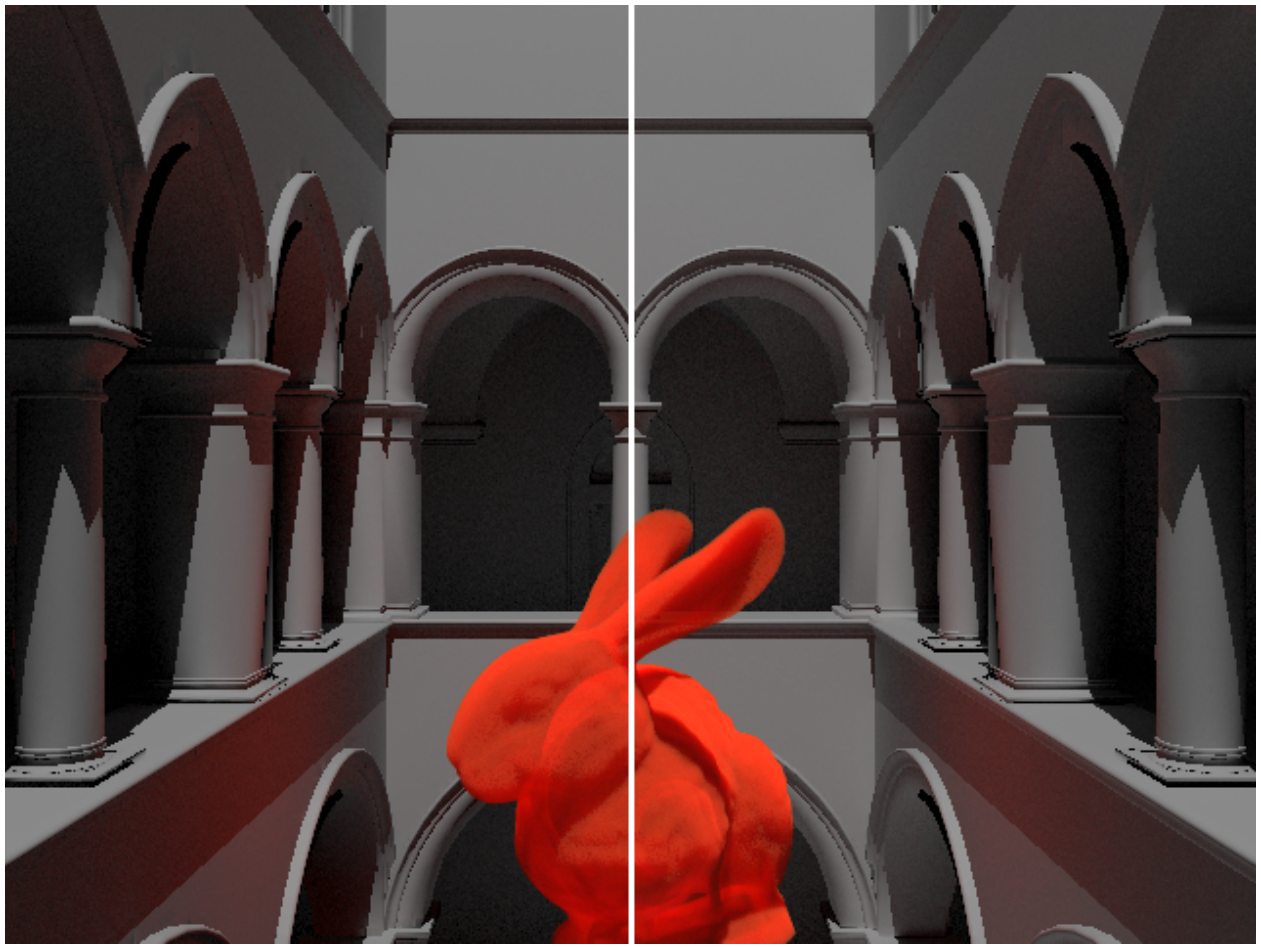
- Run on dual core Intel i5 3.4 GHz, 16 GB machine
- 128 samples per bounce (single bounce)
- 16 samples per in-scatter test
- OpenMP threads split image into vertical chunks for processing

Thread distribution across the image

Results

Scene	Render Time (s)	Image Delta	Memory Overhead
Monte Carlo w/o PCB	3351 sec	NONE	NONE
Traditional PCB	348 sec	11.0%	390 Mb (4.0%)
Extended PCB	397 sec	4.8%	395 Mb (4.1%)

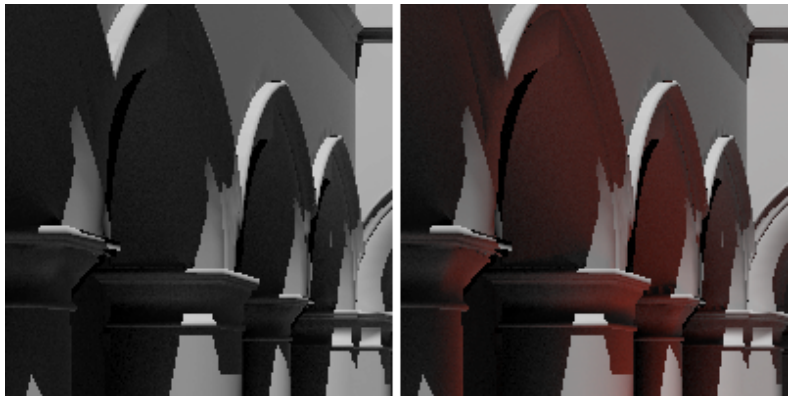
Sponza scene rendered with PCBEX



Comparison of PCBEX results *(left)* and Monte Carlo results *(right)*

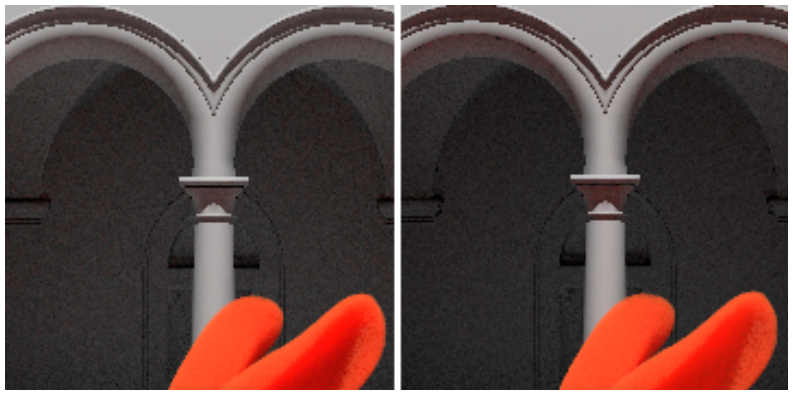
Image Comparison

Close-ups show drastic difference between PBC and PBCEX renders.

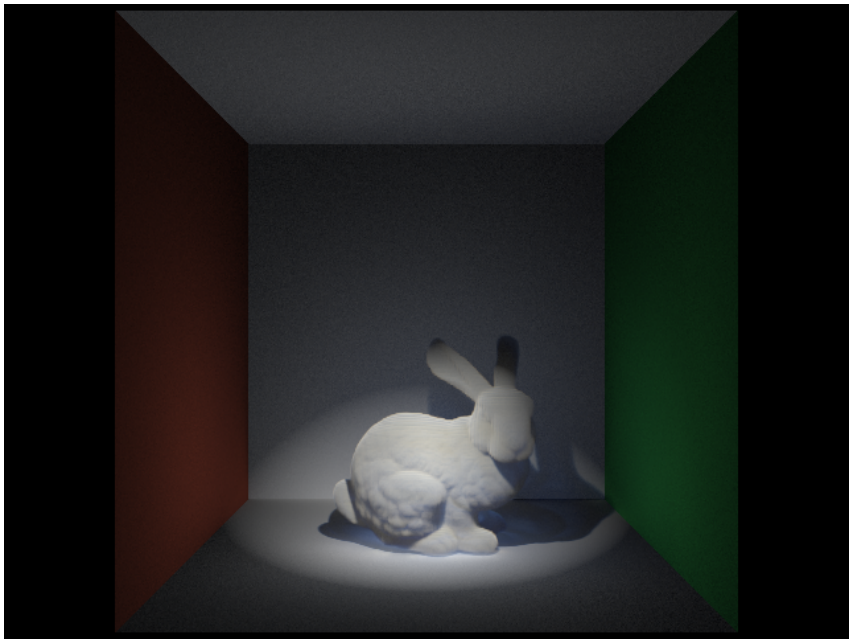


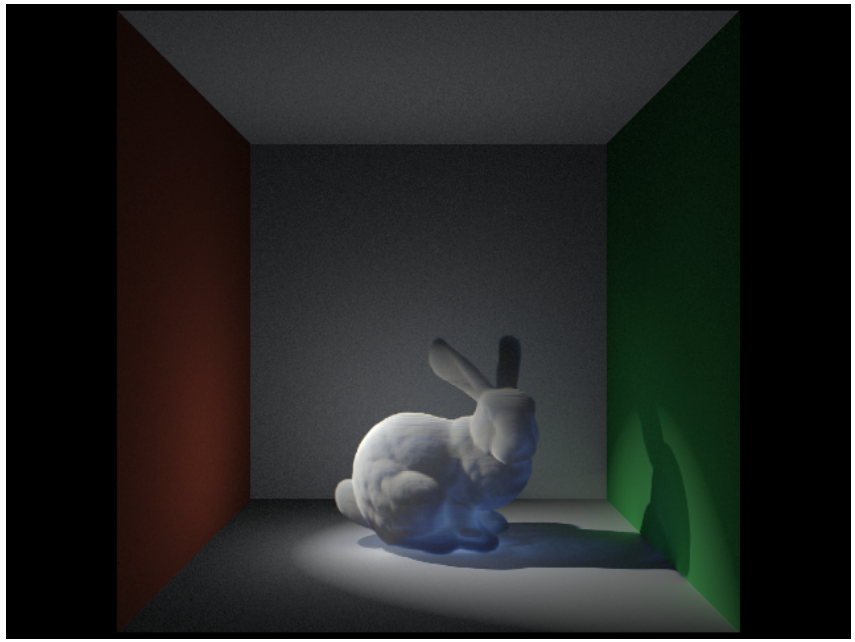
PBC vs PBCEX

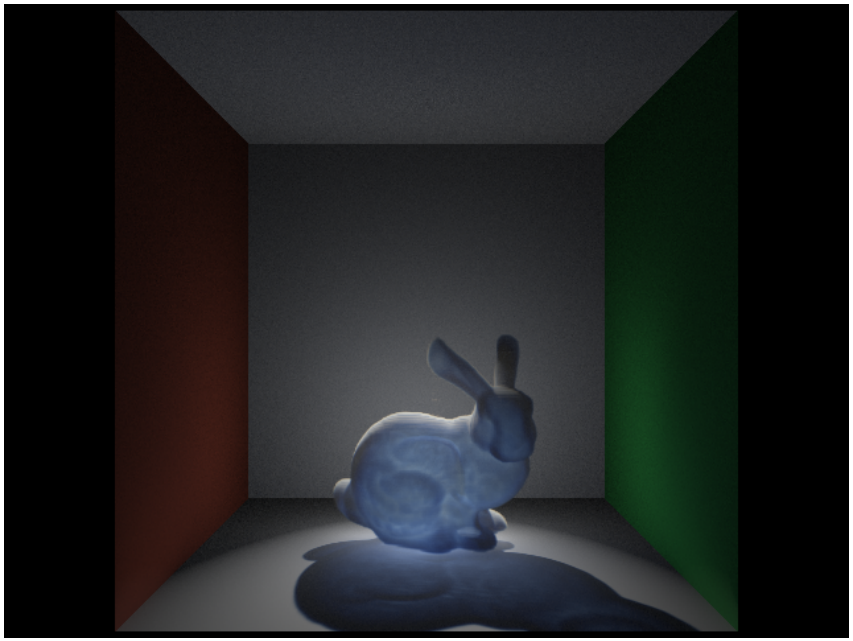
Full Monte Carlo renders differ only slightly from PCBEX renders.

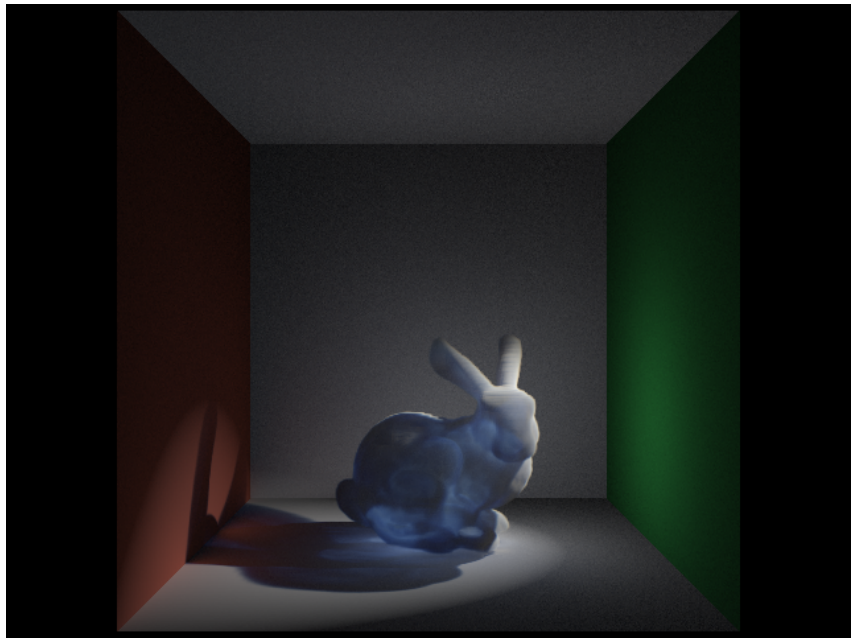


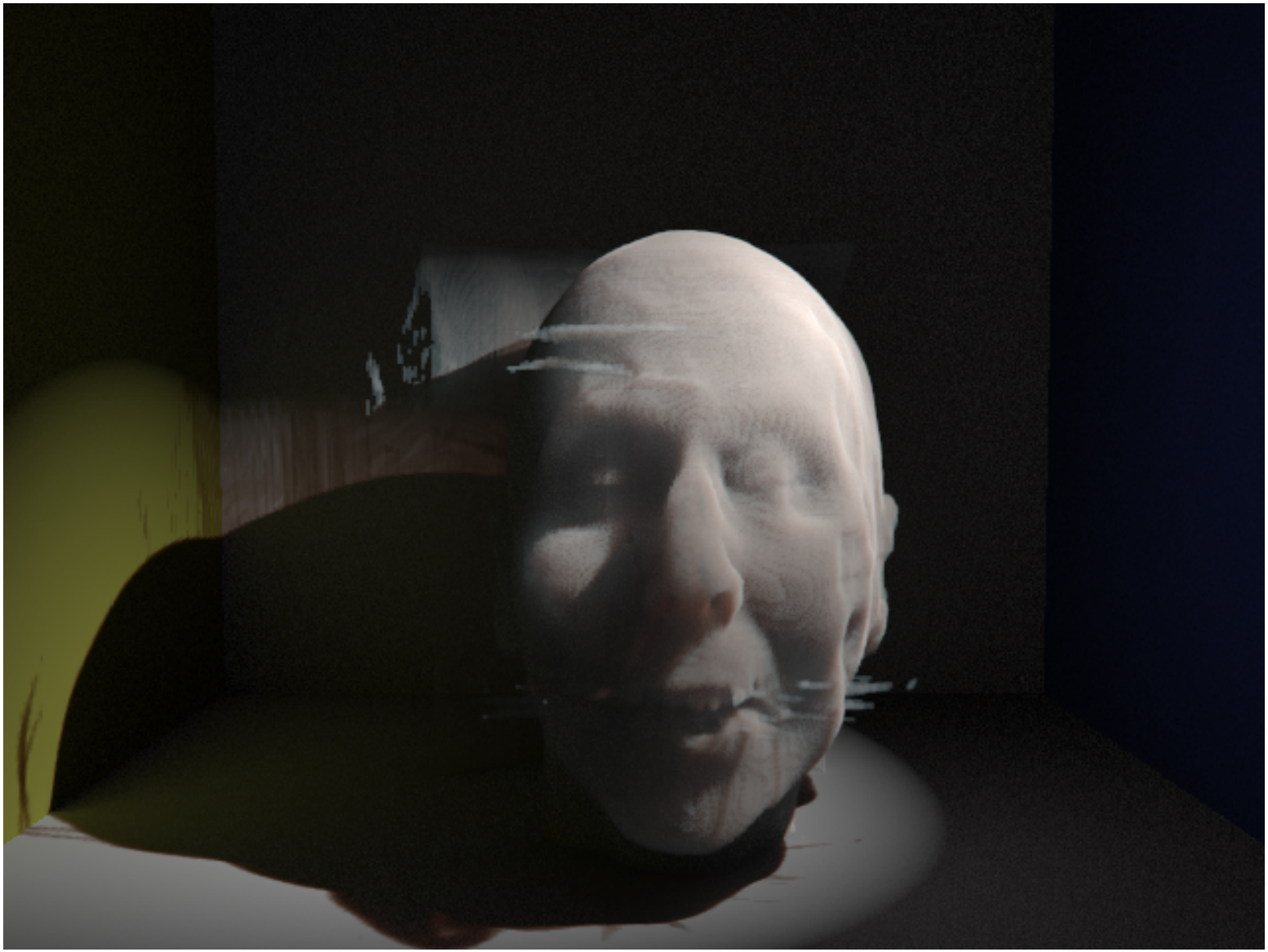
Monte Carlo vs PBCEX

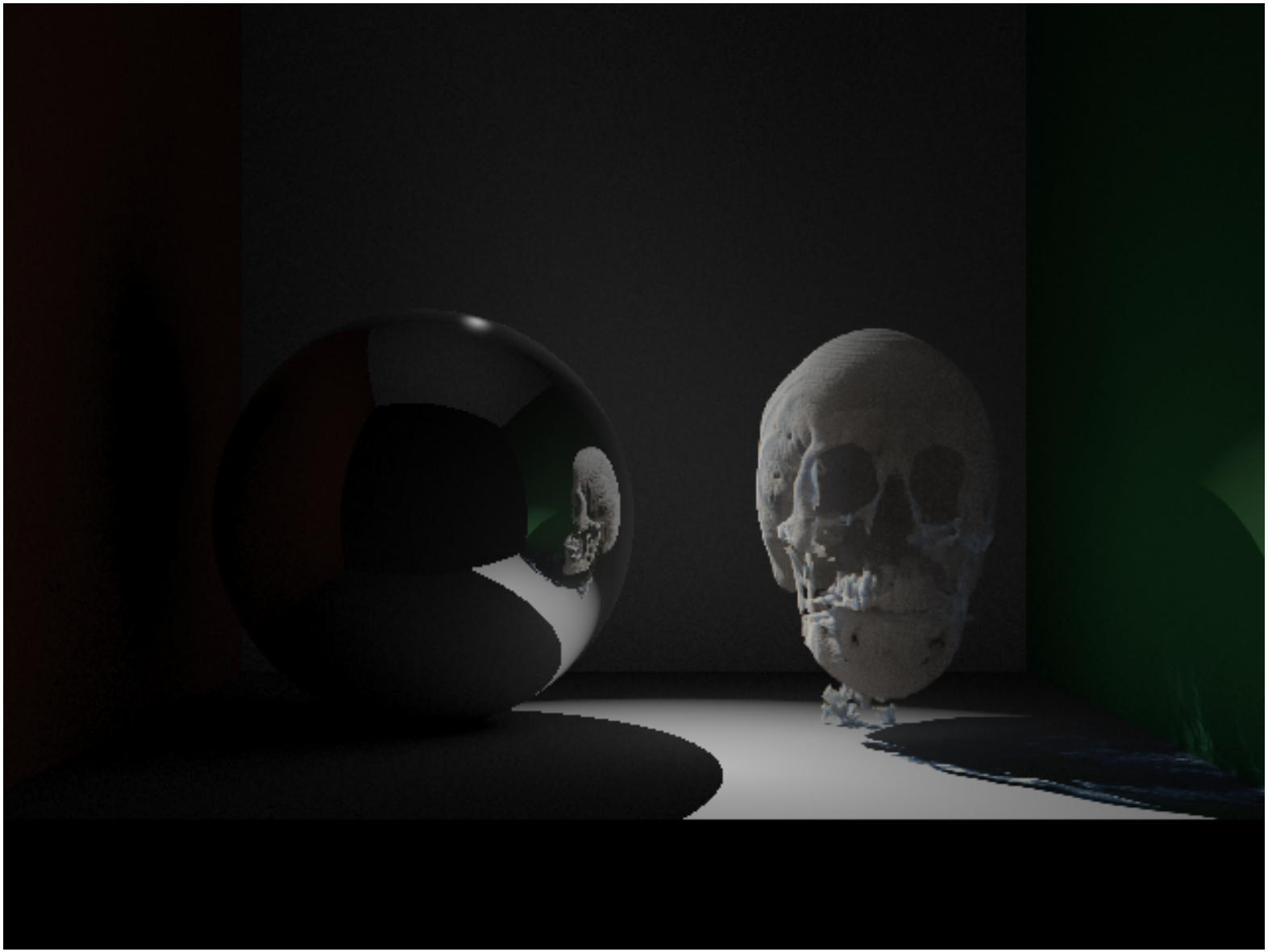


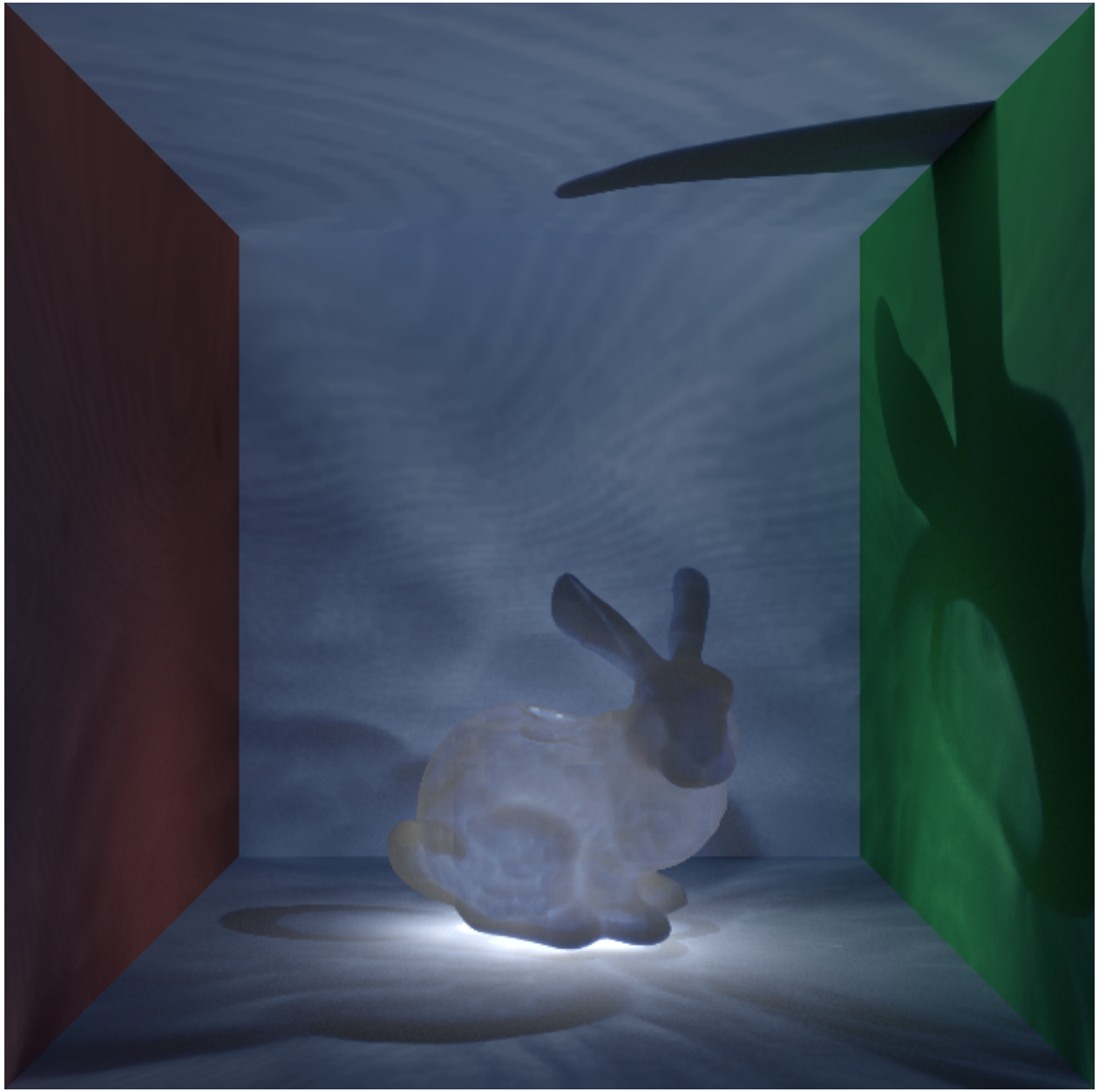


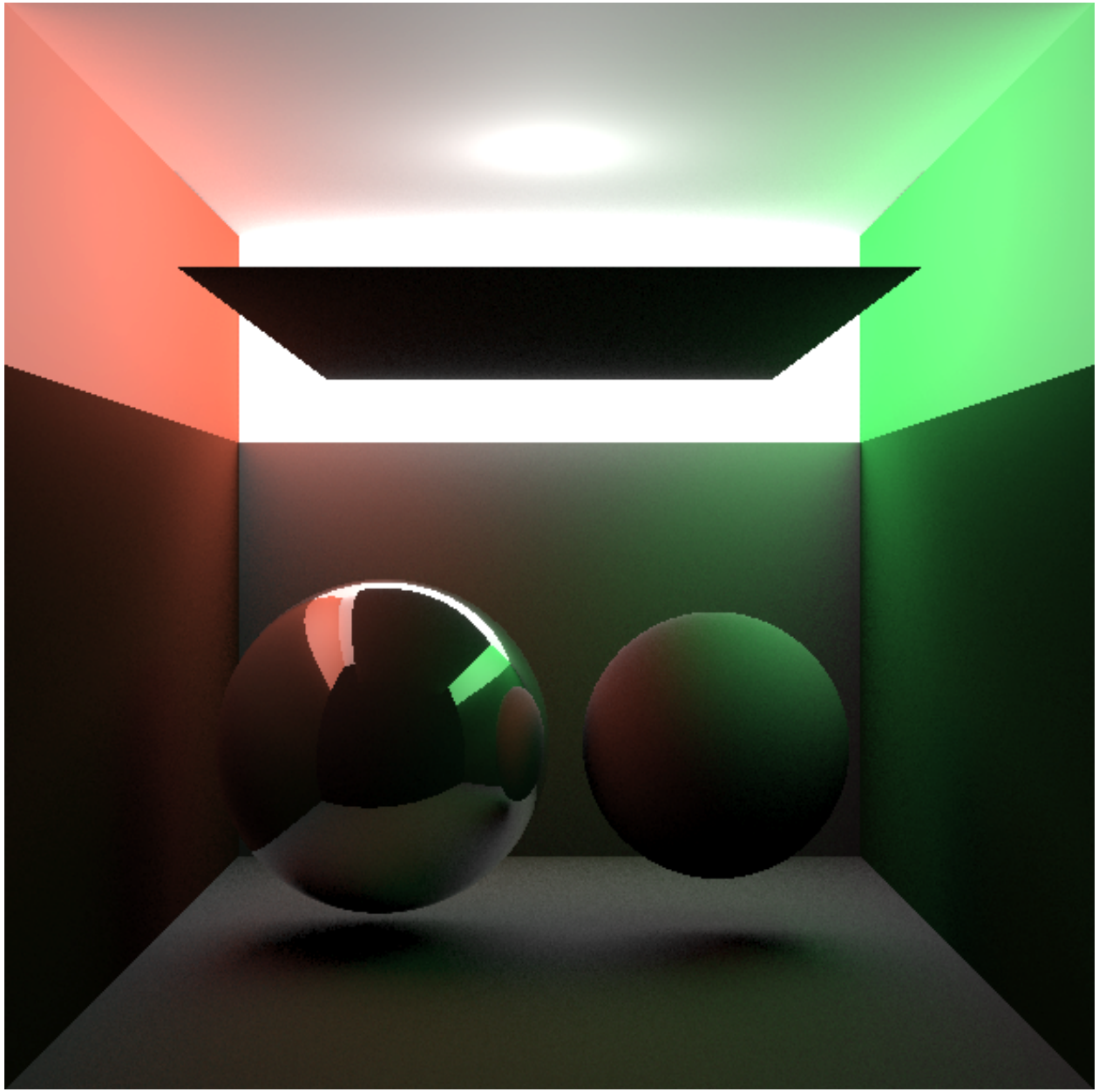


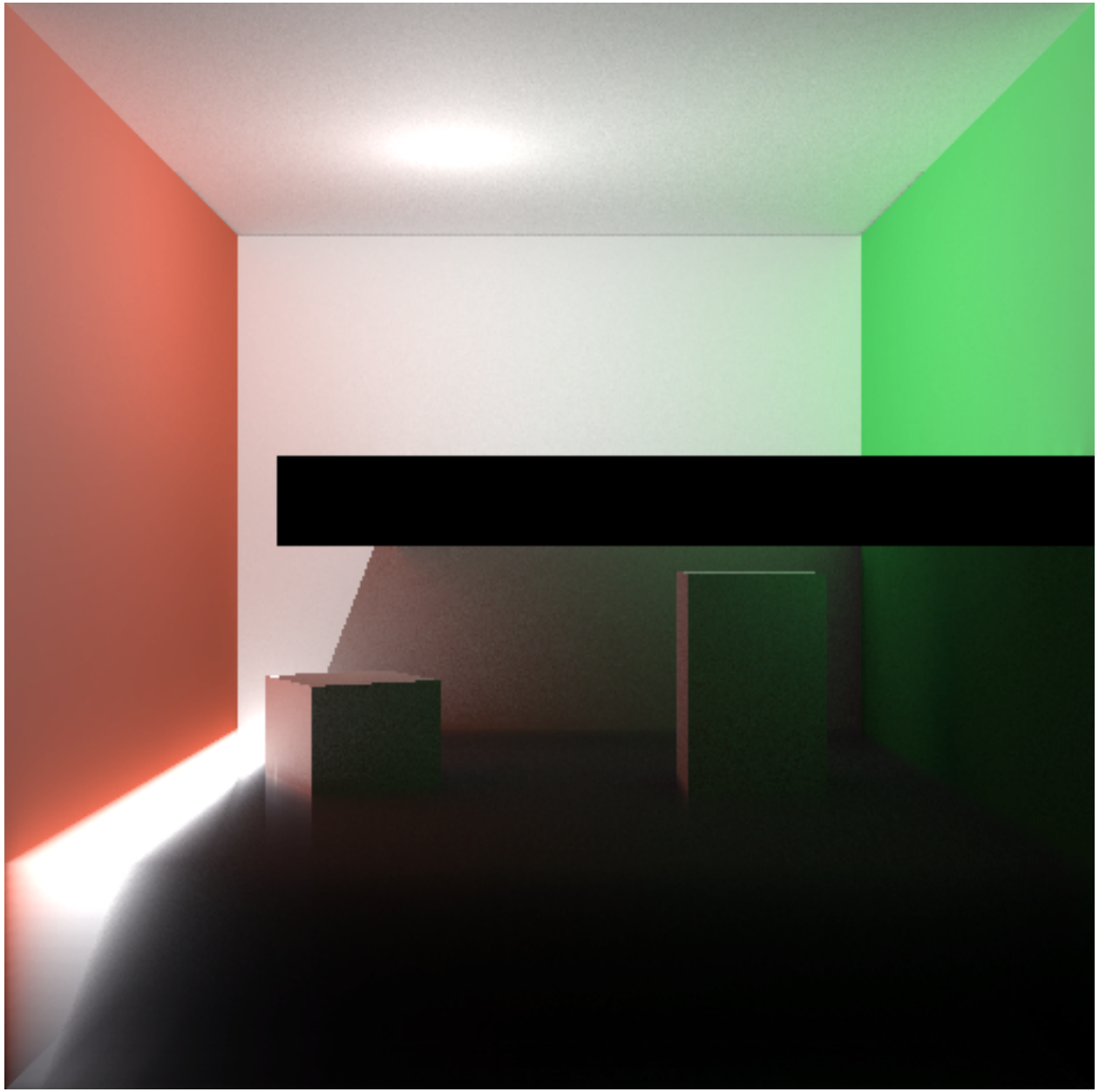




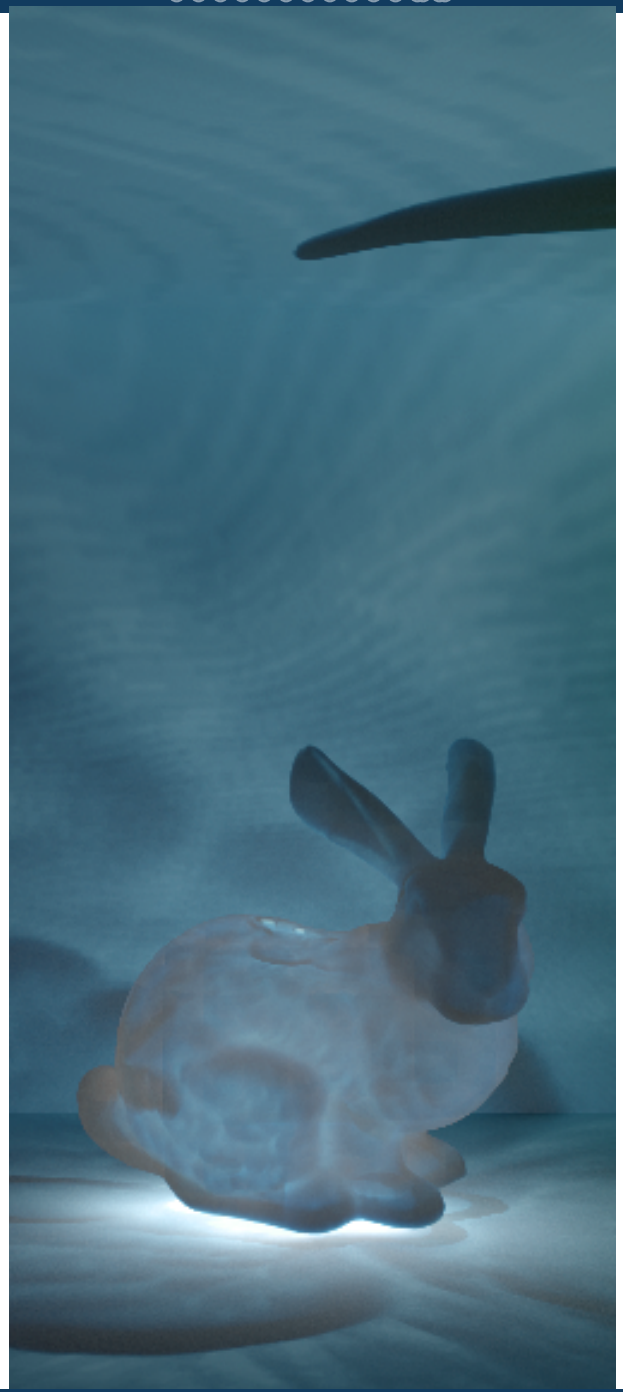








Outline



Future Work

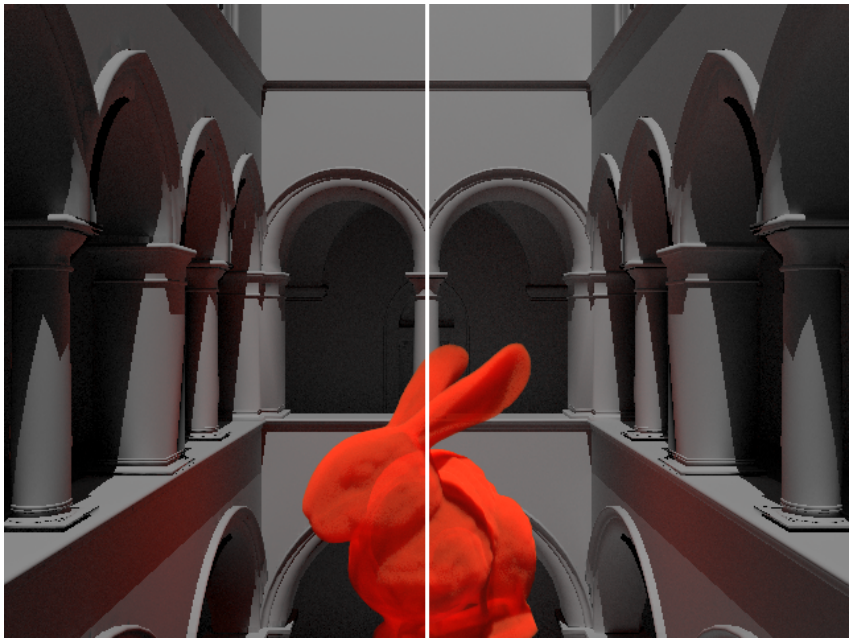
1. Multiple bounce
2. Phase functions for volumes
3. Parallelism
4. Optimal Sampling
5. GPU Acceleration

Conclusion

Modifications can be made to PCB to incorporate volume light contributions

Even simple implementations show great improvements in performance, nearly ten times faster than traditional Monte Carlo

Image quality is comparable to Monte Carlo results



PBCEX *(left)* vs Monte Carlo *(right)*