PCBEX:
Point-Based Color Bleeding With Volumes
Thesis Defense

Christopher James Gibson

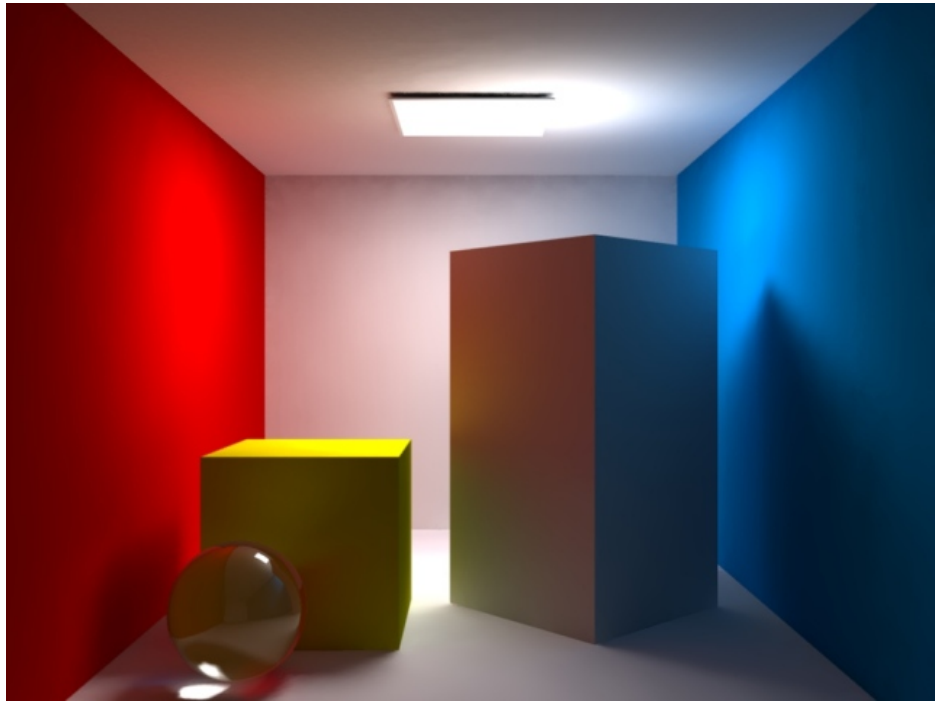
California Polytechnic University
CSC Department

June 9, 2011

Global Illumination

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



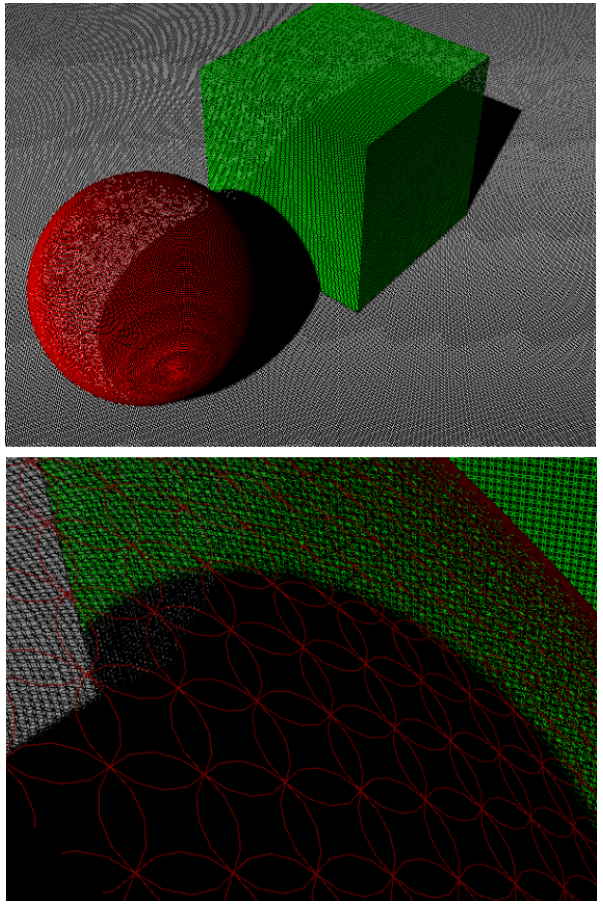
(image source: cgtutorials.com)

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)

Irradiance and Radiance

Irradiance Equation

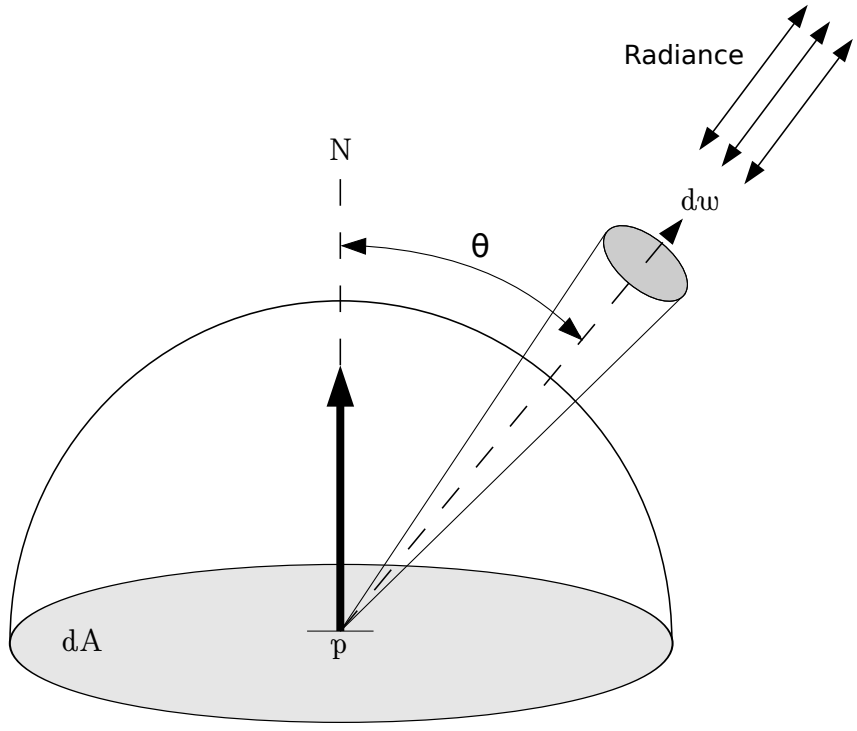
$$E = \frac{d\Phi}{dA}$$

Radiance Equation

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

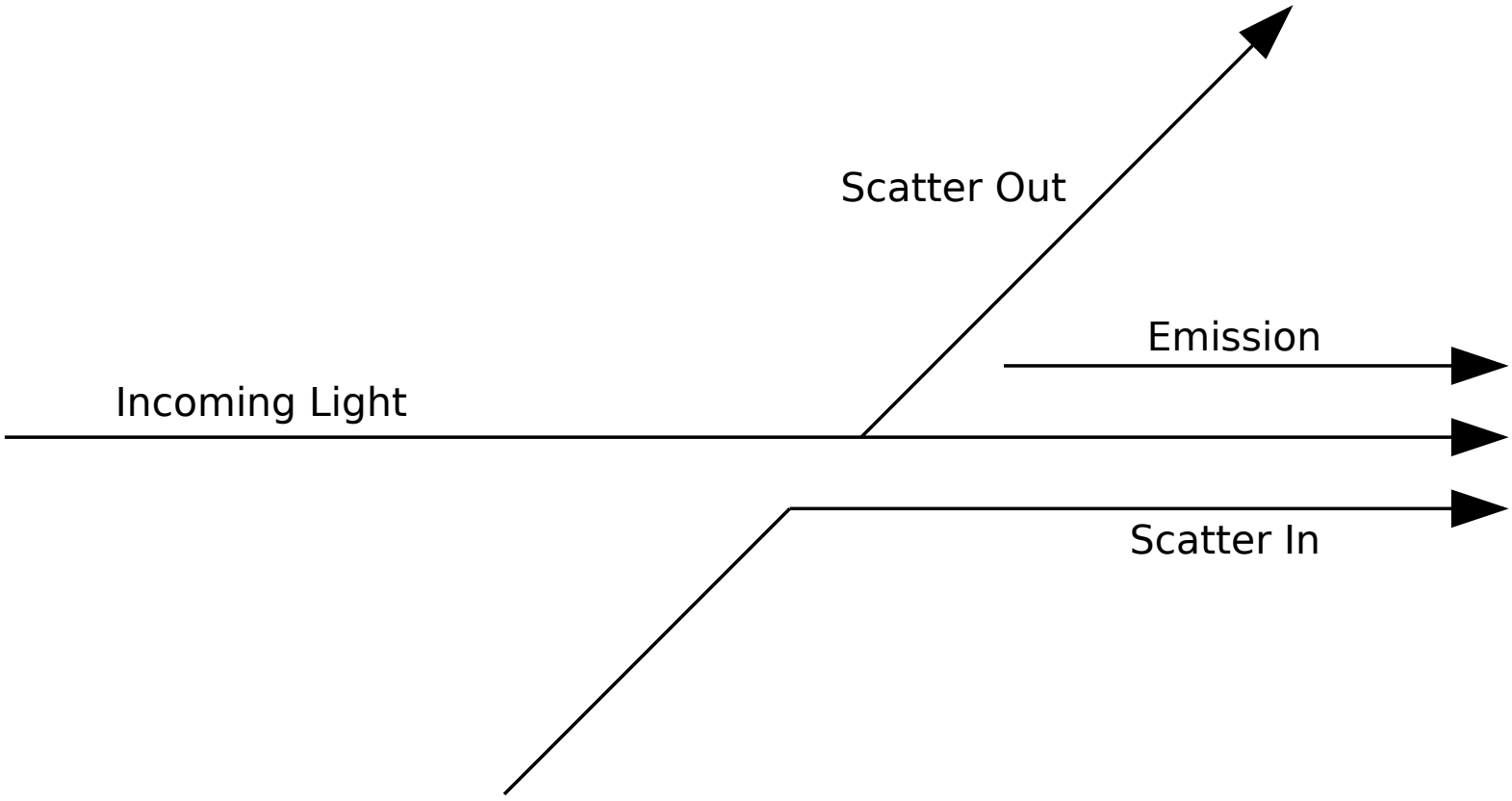
Radiance Invariance

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



Radiance diagram

Volume Lighting



Volume Lighting- Scatter, Absorption and Transmittance

Absorption Equation

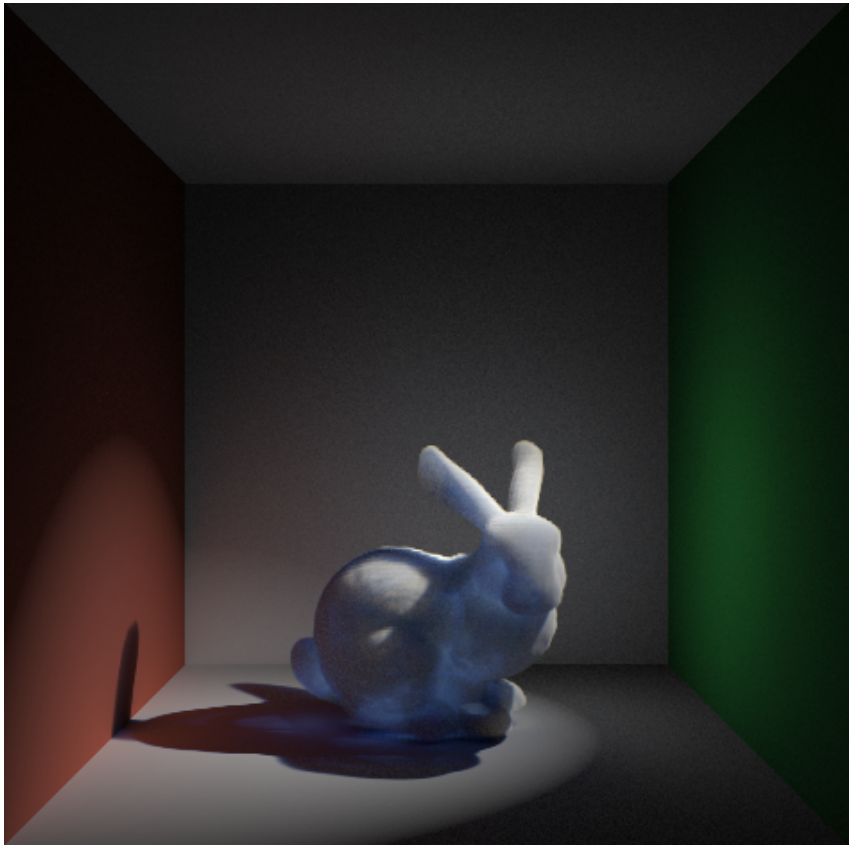
$$e^{-\int_0^d \sigma_a(p+tw,w)dt}$$

Scatter Out Equation

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

Transmittance Equation

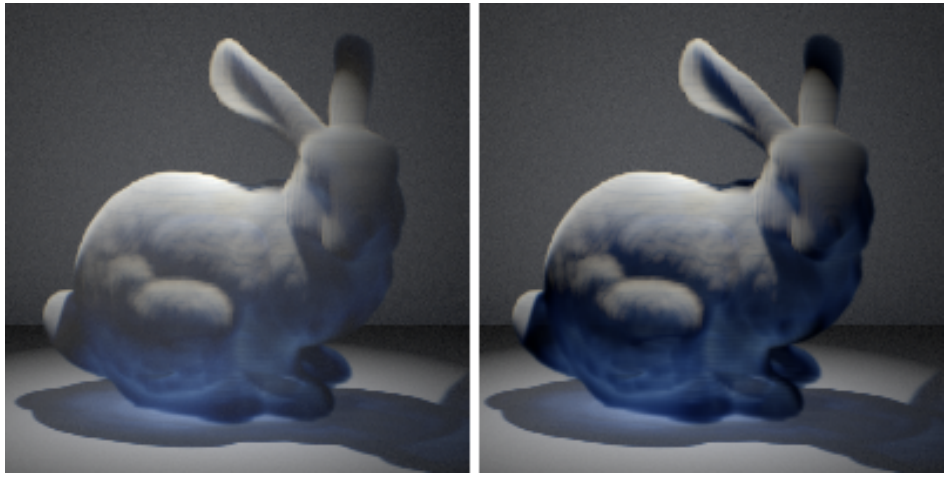
$$T_r(\mathsf{p} \rightarrow \mathsf{p}') = e^{-\int_0^d \sigma(p+tw,w)dt}.$$



Volume Lighting - Scatter-In

Scatter-In Equation

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



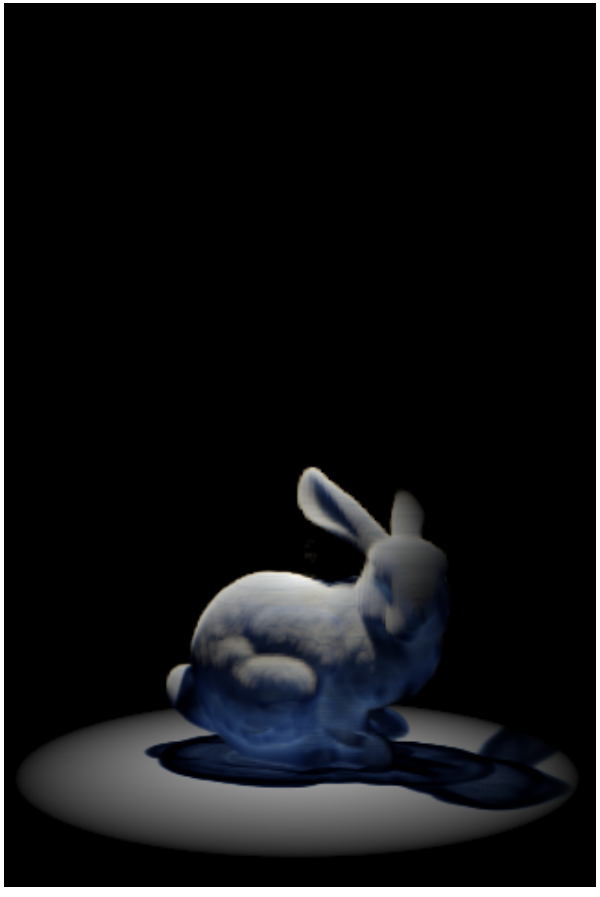
Scatter-in *(left)* and only direct lighting via transmittance *(right)*

Point-Based Color Bleeding (PCB)

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)

Motivation

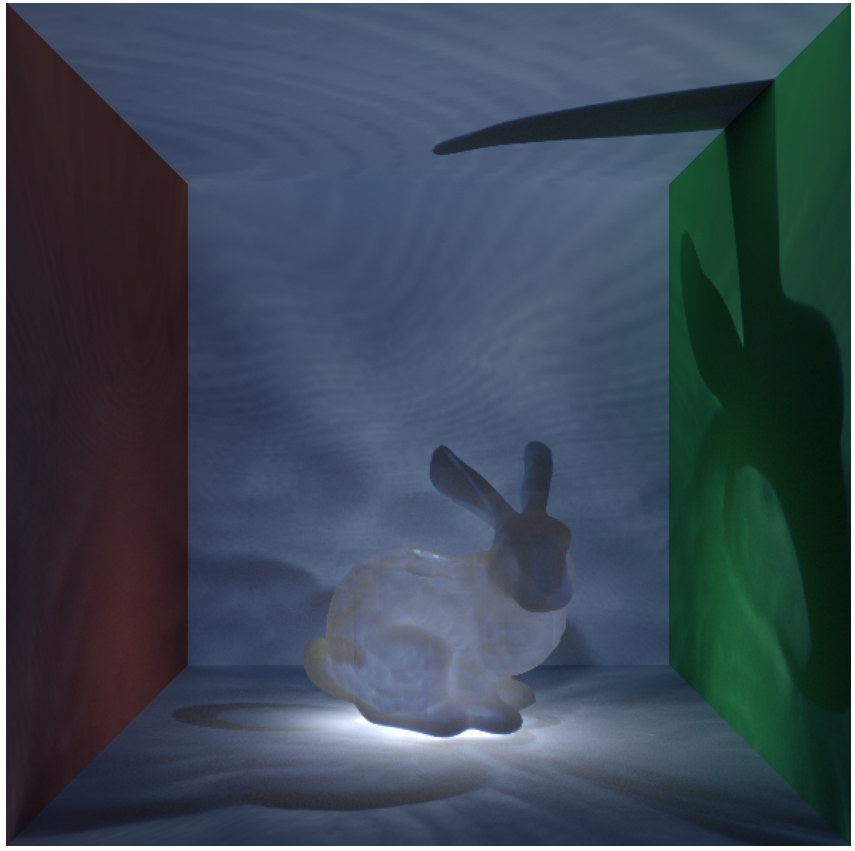


Complex volume lighting algorithms are computationally expensive and complex

Most GA algorithms do not include volume contributions

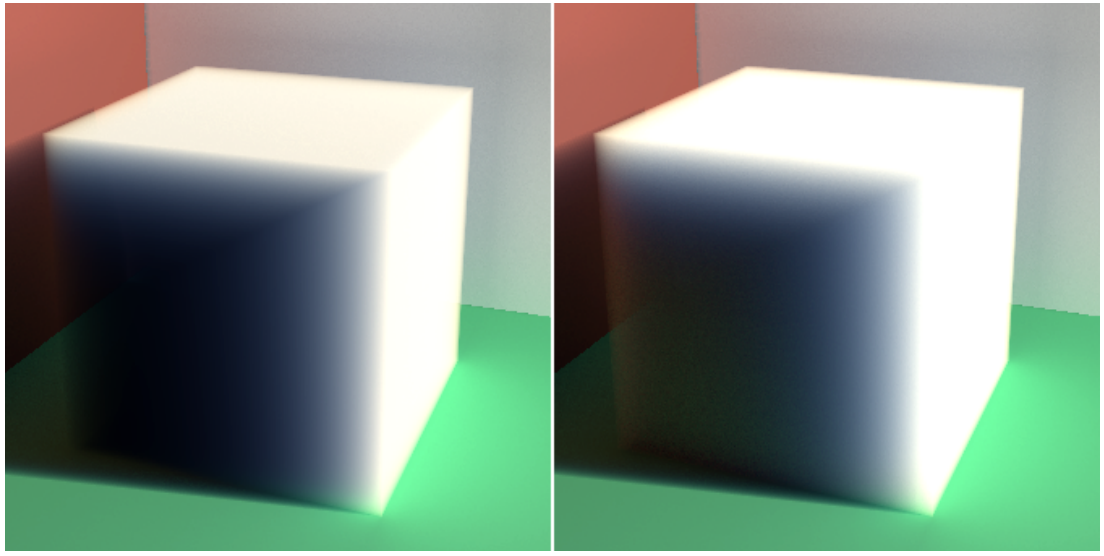
Goals

- 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



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

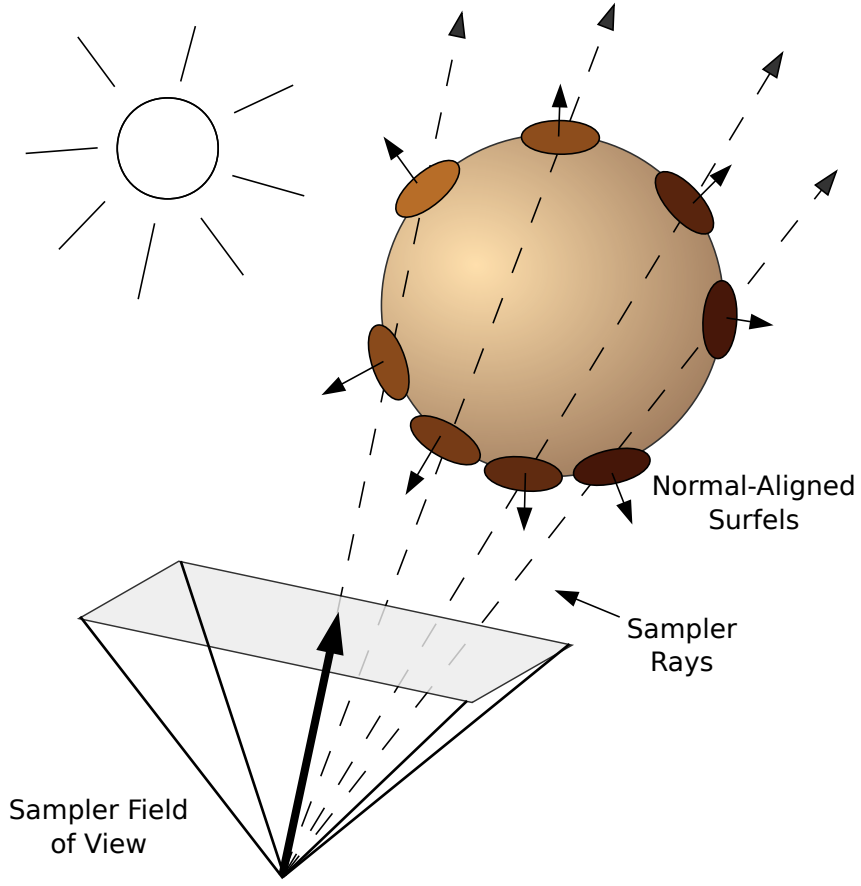


Sampling the Scene

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

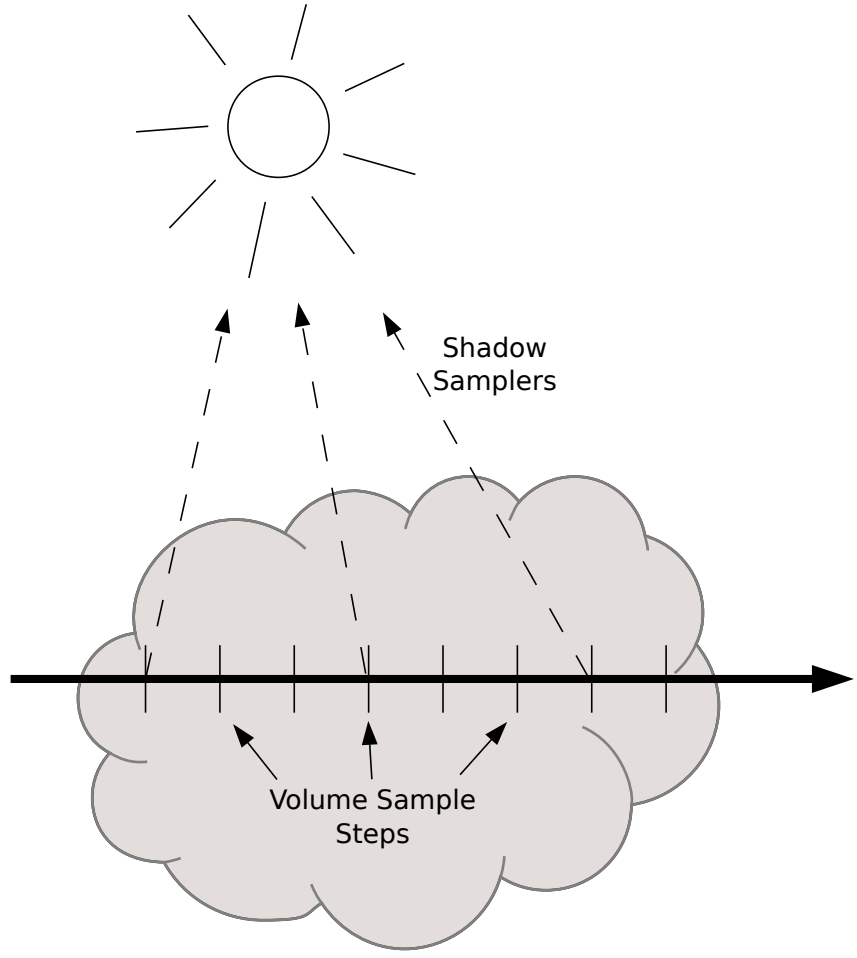


Sampling Volumes

Samples are taken at discrete steps across the volume domain

Scatter/Absorbtion/Lighting factors are averaged within each step

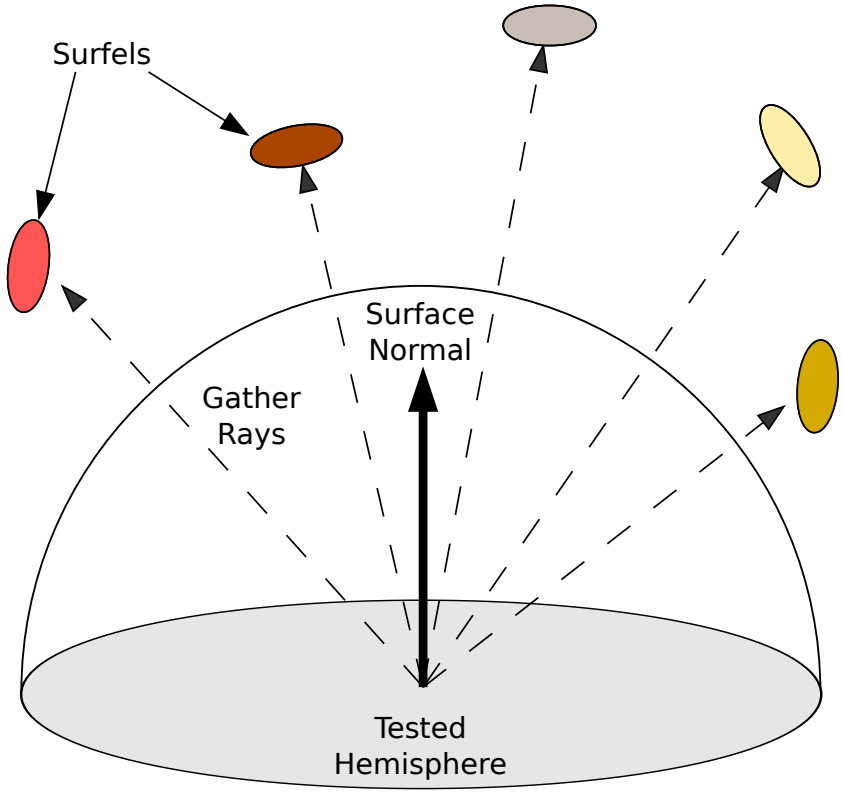
Lvoxels (spheres) instead of *surfels*



Gathering Light

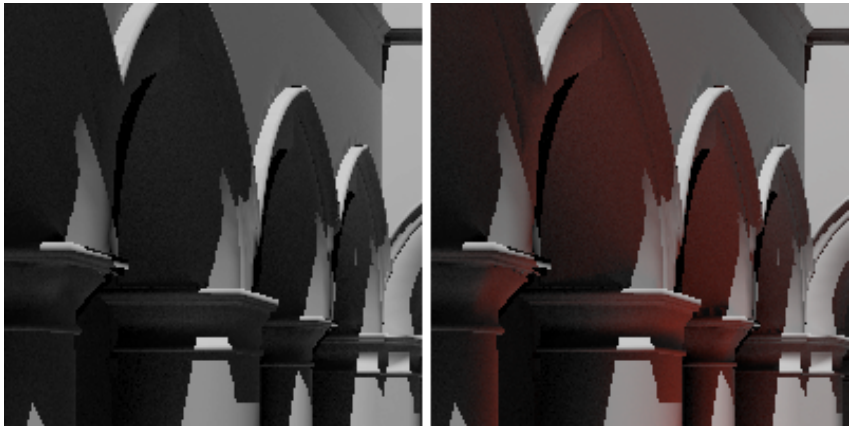
Samples cast into point cloud "gather" light via intersection

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

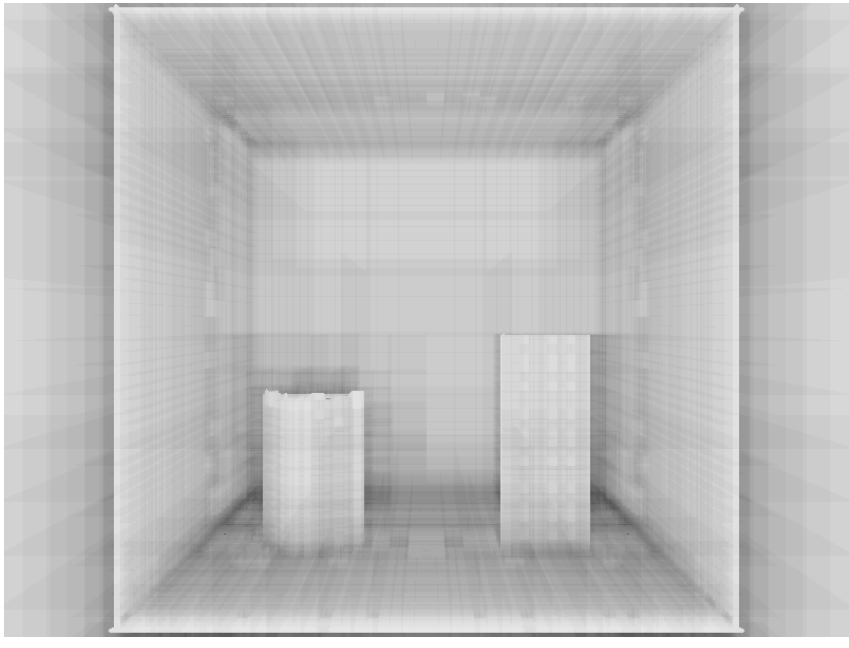


Integrating Volume Data - Scatter Out

Gather stage remains same, no changes necessary



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



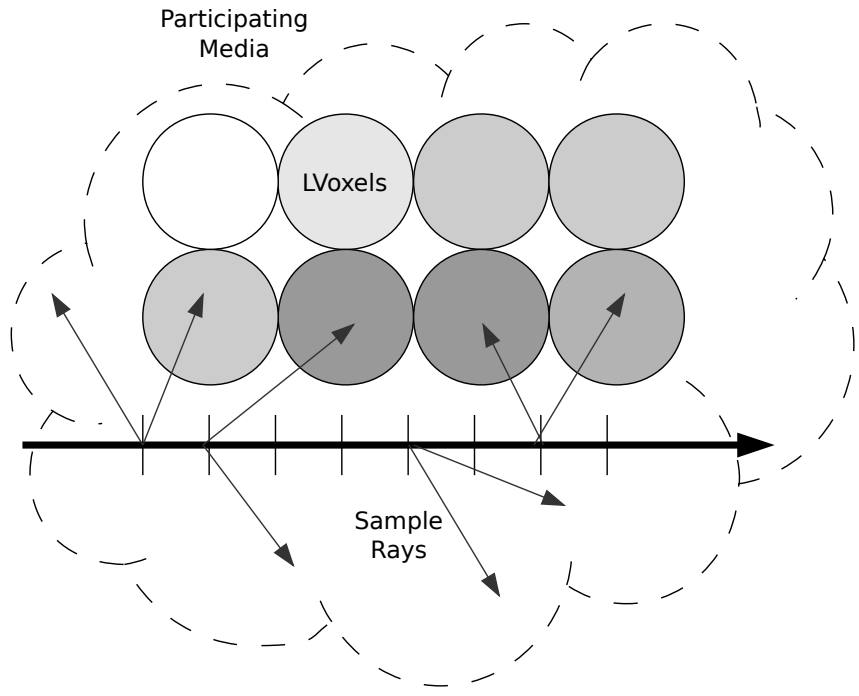
Must keep track of transmittance during traversal

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



Results - Environment

Test scene involved:

- 60,000 triangle Sponza Atrium Model
- Stanford CT Scan Data *(Bunny and Head)*

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

Results - Stanford Bunny

Scene	Render Time (s)	Image Delta	Memory Overhead
64^3 resolution volume			
Monte Carlo	3229 sec	NONE	NONE
Traditional PCB	348 sec	5.8%	466.3 MB (4.780%)
Extended PCB	433 sec	2.1%	466.7 MB (4.786%)
128^3 resolution volume			
Monte Carlo	3297 sec	NONE	NONE
Traditional PCB	348 sec	5.6%	466.3 MB (4.780%)
Extended PCB	402 sec	2.4%	467.5 MB (4.783%)
256^3 resolution volume			
Monte Carlo	3674 sec	NONE	NONE
Traditional PCB	348 sec	9.6%	466.3 MB (4.780%)
Extended PCB	417 sec	3.8%	466.4 MB (4.785%)

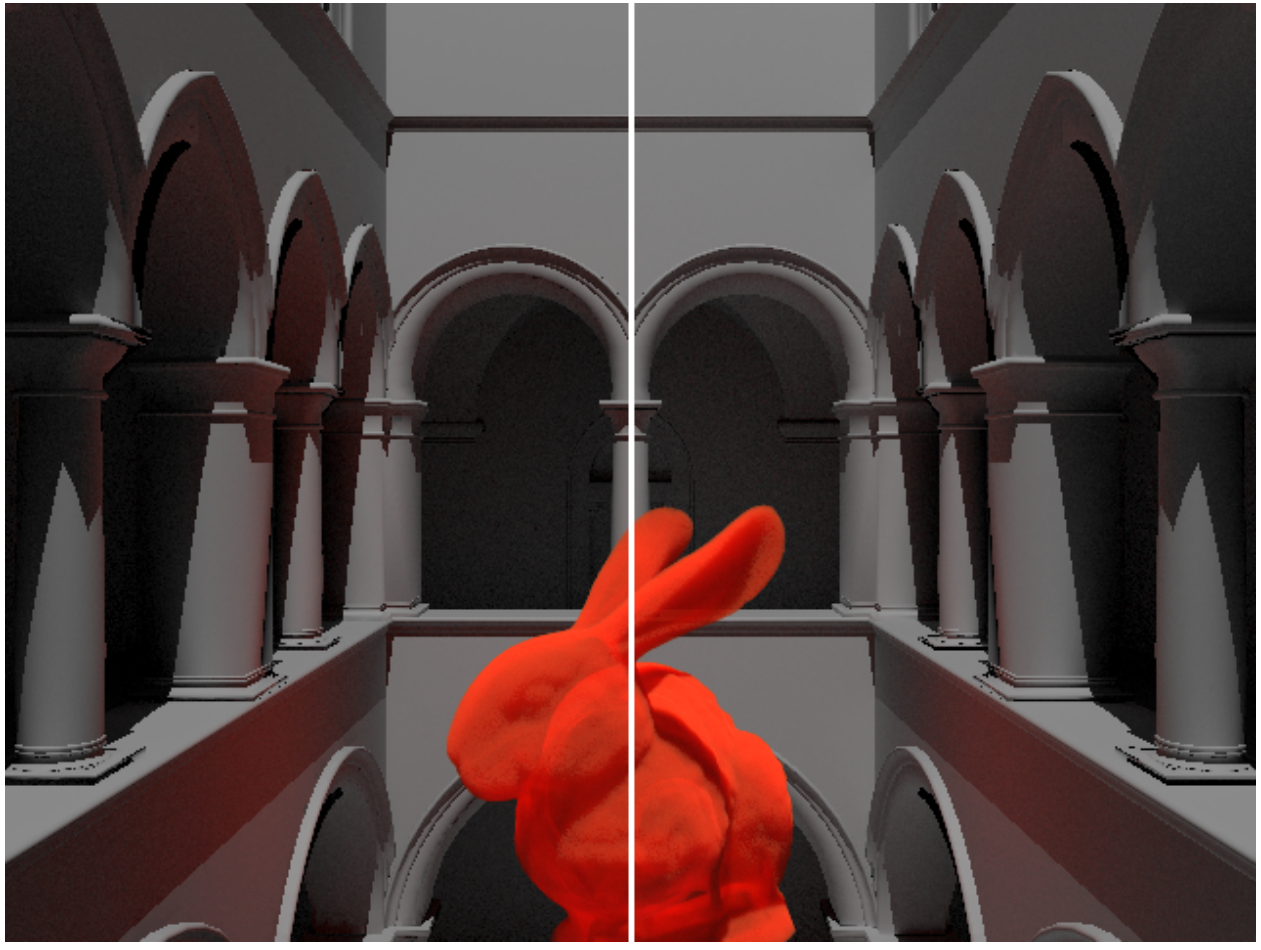
Table: Sponza Scene With Stanford Bunny Volume Runtime

Results - CT Head

Scene	Render Time (s)	Image Delta	Memory Overhead
64^3 resolution volume			
Monte Carlo	10150 sec	NONE	NONE
Traditional PCB	348 sec	14.2%	466.3 MB (4.780%)
Extended PCB	756 sec	3.7%	468.0 MB (4.800%)
128^3 resolution volume			
Monte Carlo	15811 sec	NONE	NONE
Traditional PCB	348 sec	14.4%	466.3 MB (4.780%)
Extended PCB	755 sec	4.2%	467.3 MB (4.790%)
256^3 resolution volume			
Monte Carlo	31373 sec	NONE	NONE
Traditional PCB	348 sec	14.2%	466.3 MB (4.780%)
Extended PCB	864 sec	4.3%	467.1 MB (4.790%)

Table: Sponza Scene With CT Head Volume Runtime

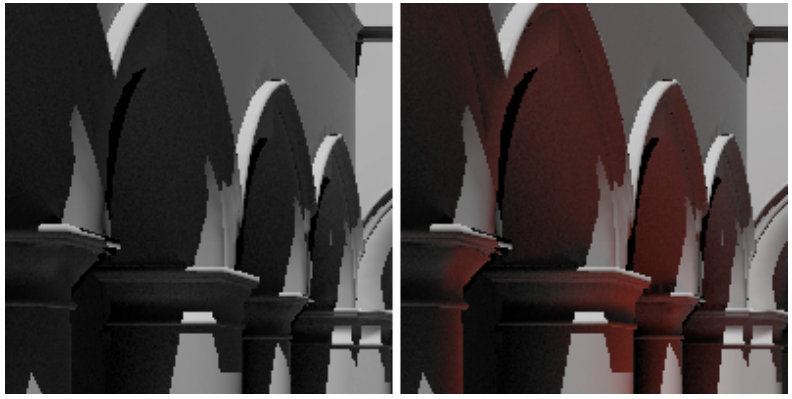
Sponza scene rendered with PCBEX



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

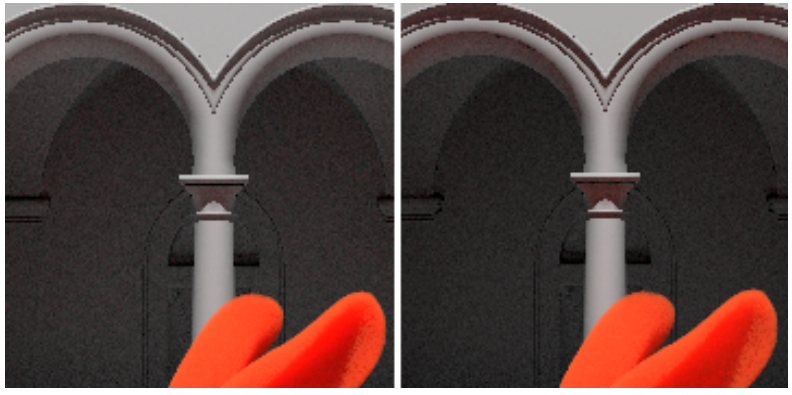
Image Comparison

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

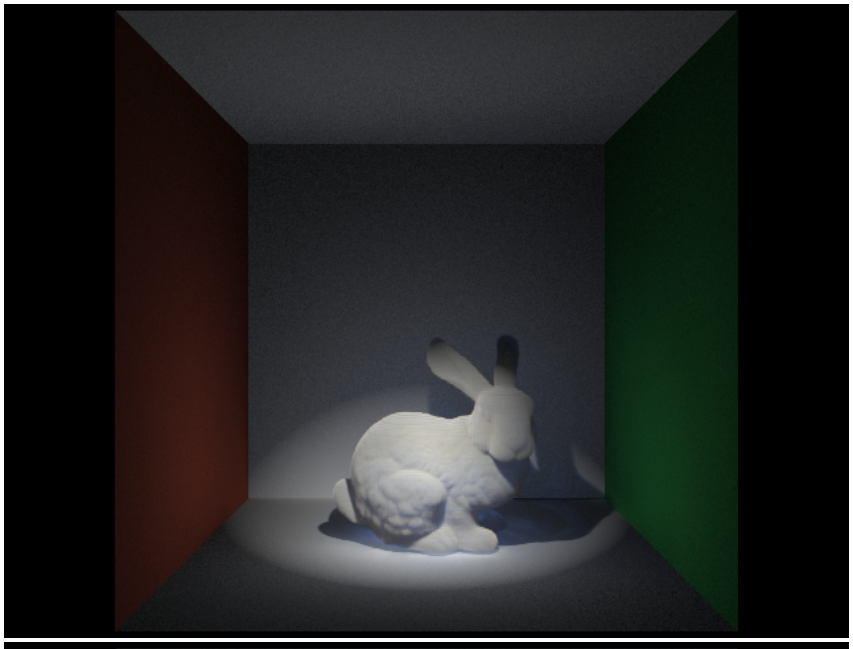
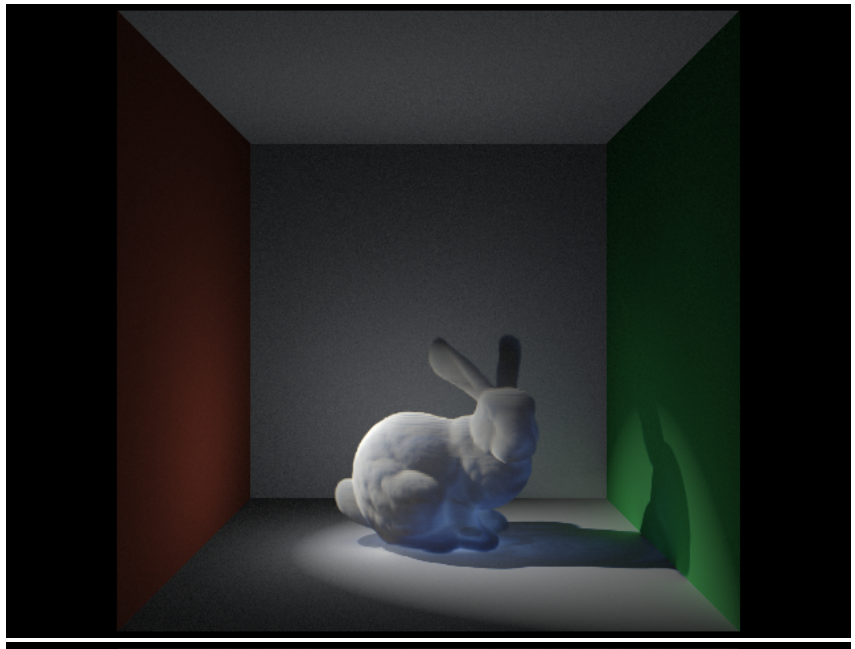
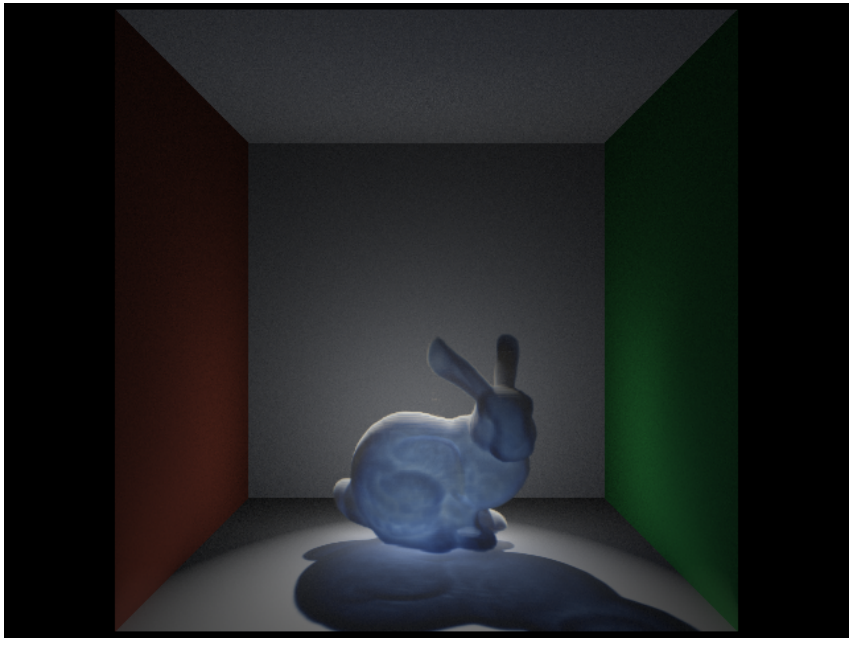
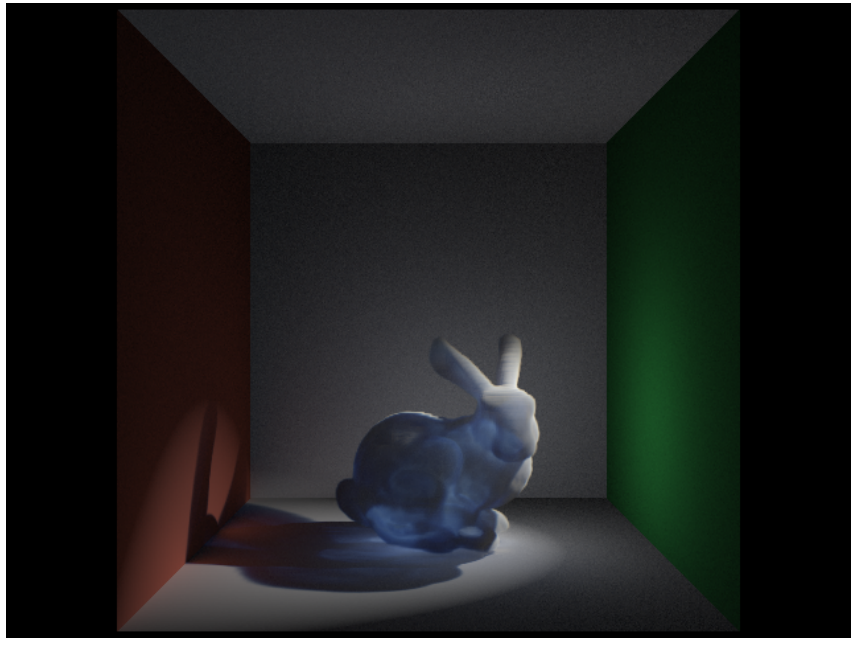


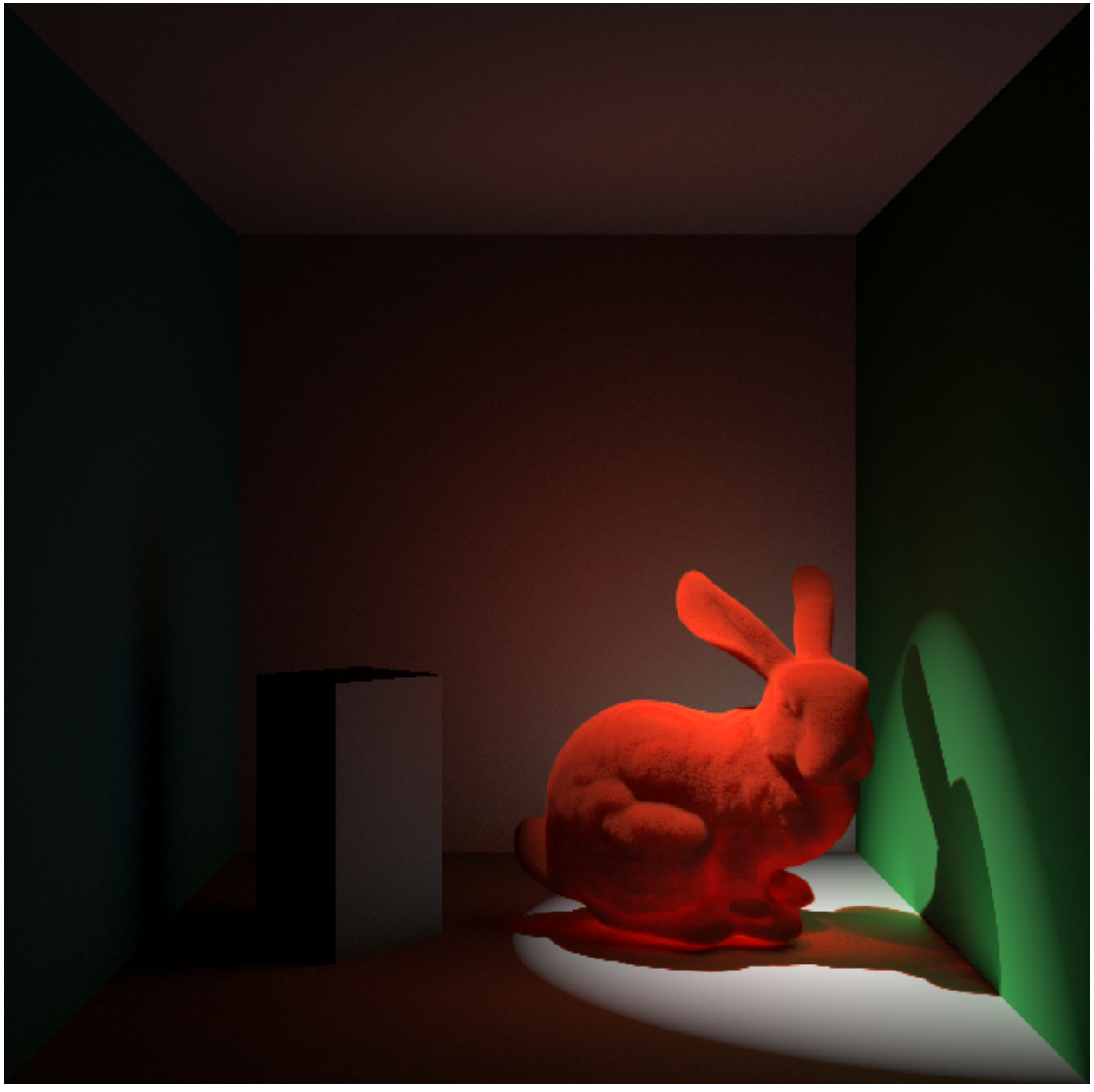
PBC vs PBCEX

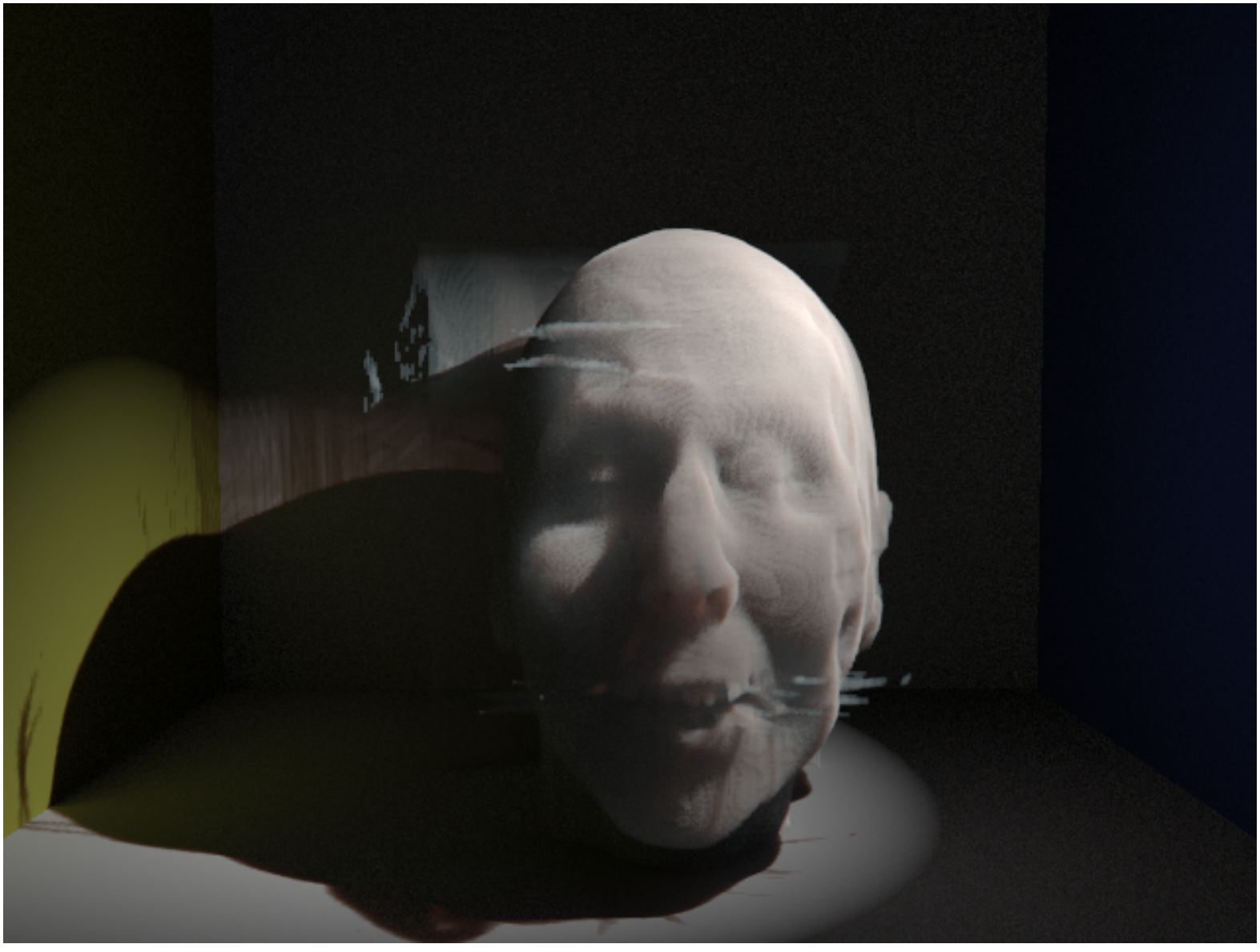
Full Monte Carlo renders differ only slightly from PCBEX renders.

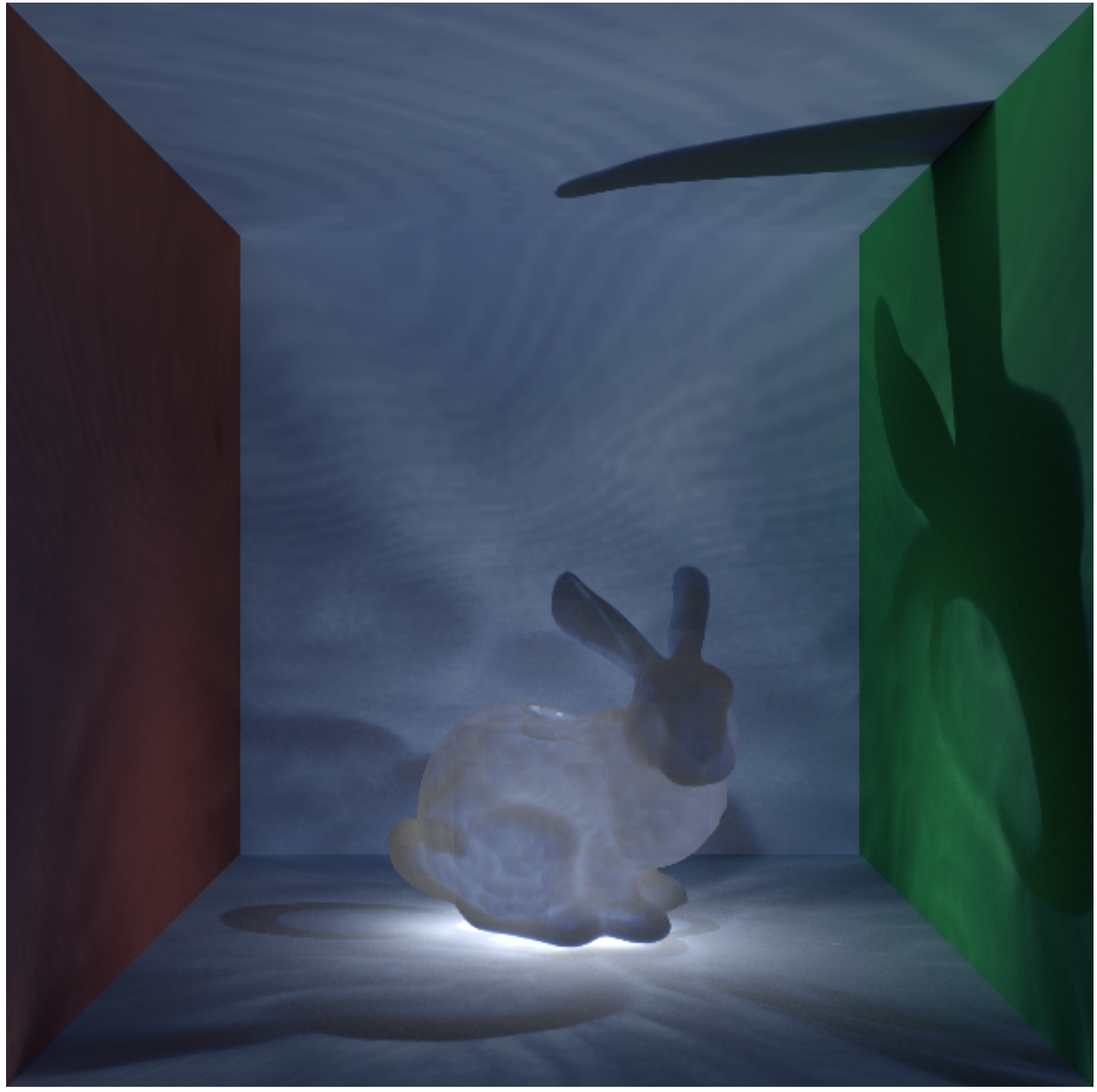


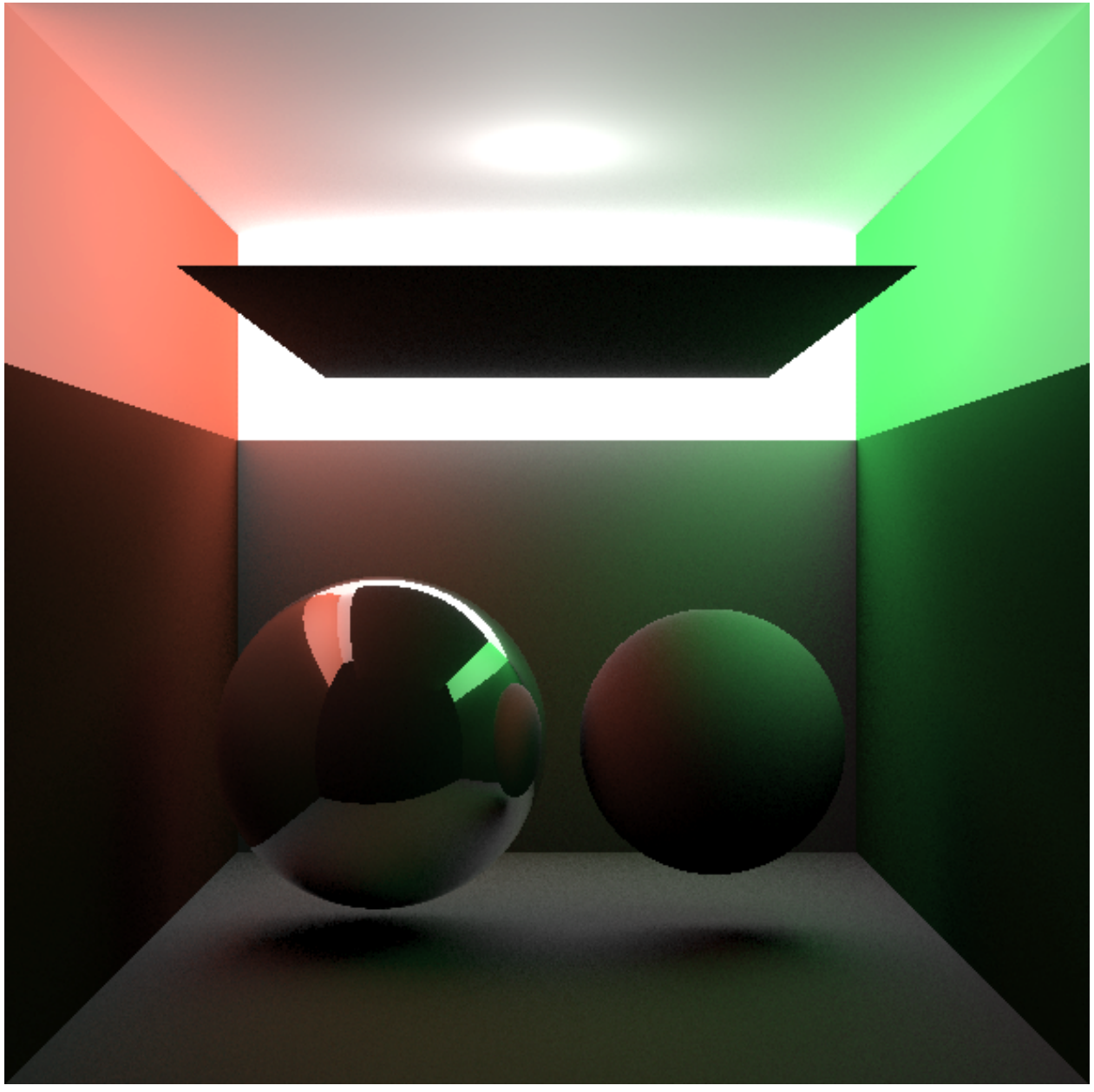
Monte Carlo vs PBCEX

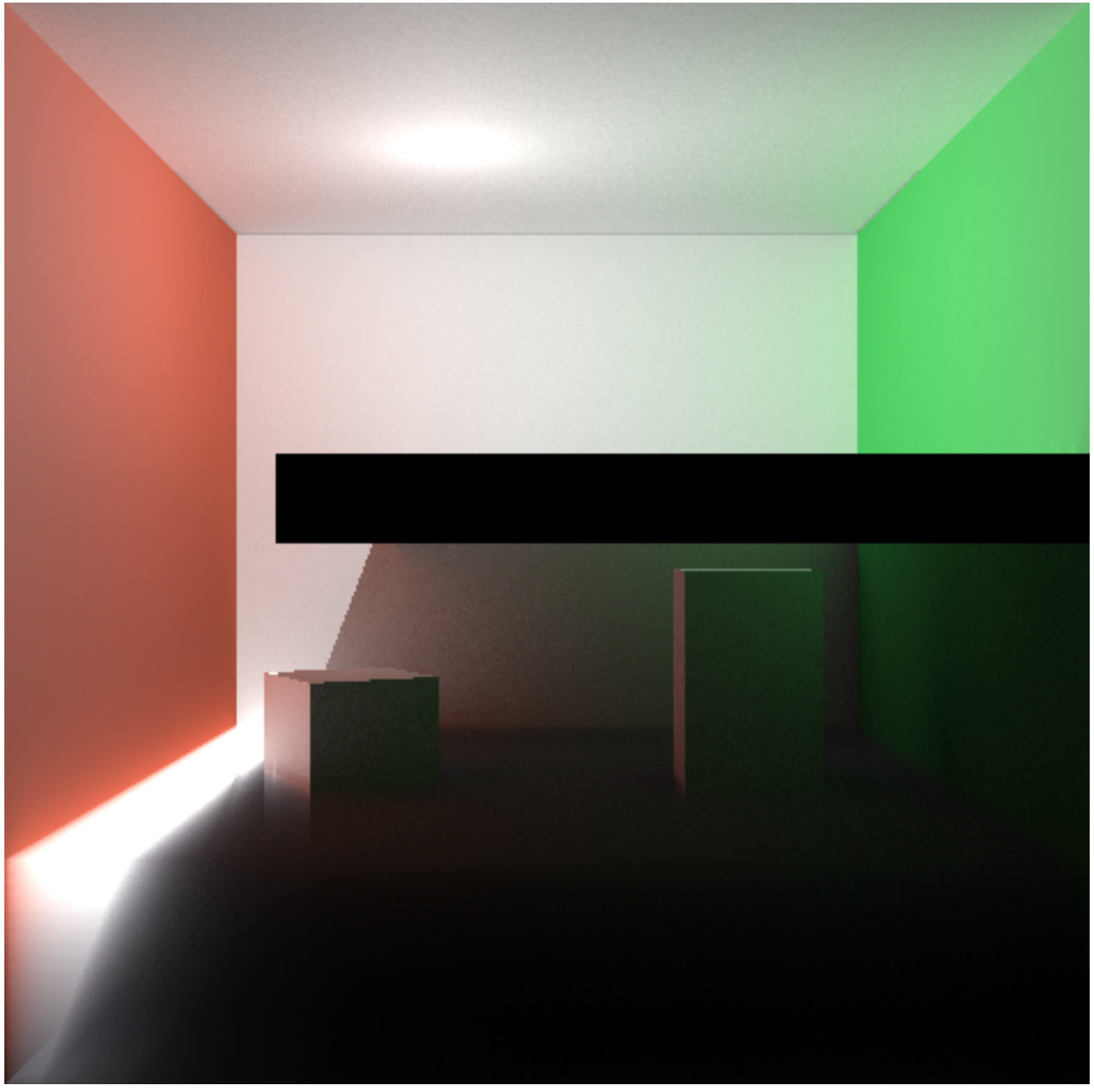















Future Work

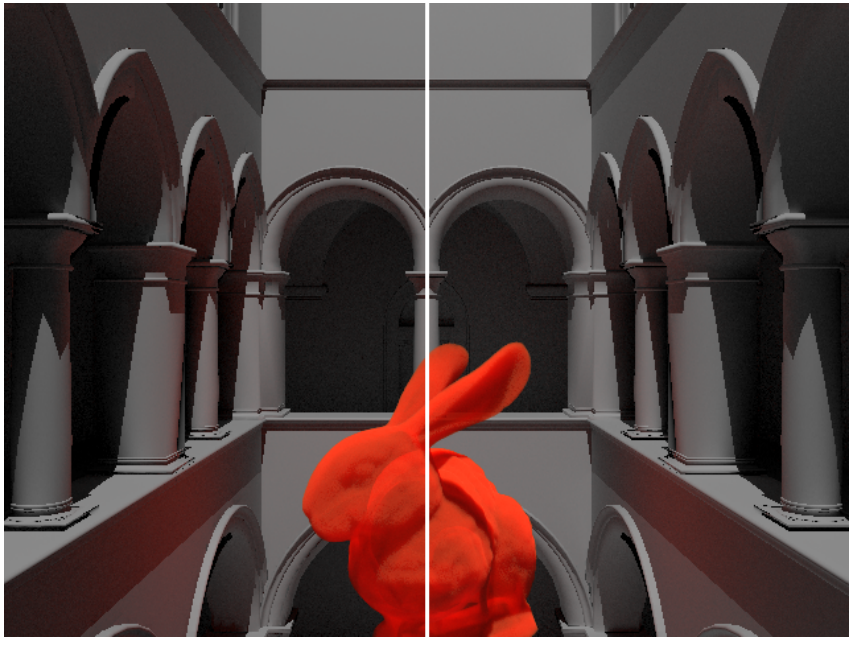
1. Multiple bounce
2. Phase functions for volumes
3. Optimal Sampling

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)*