Based on material by Carsten Dachsbacher, Jan Kautz, Michael Kenzel

Two classes of Rendering Algorithms

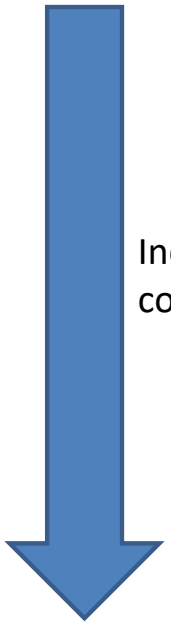
– Offline rendering
  - Complete scene information available at all times
– Online rendering
  - Generates image while streaming scene data
  - E.g., surface fragments only available during rasterization

You are here (usually)

What is Global Illumination?

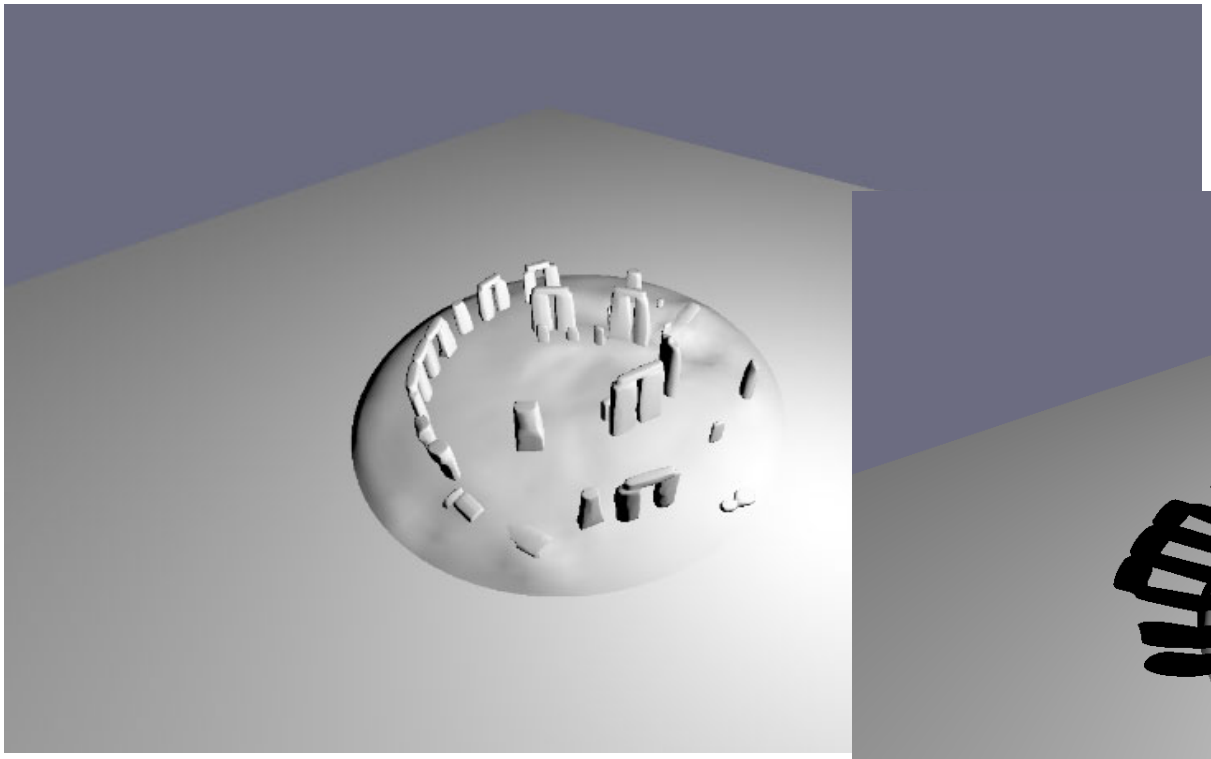
Objects influence each other‘s appearance

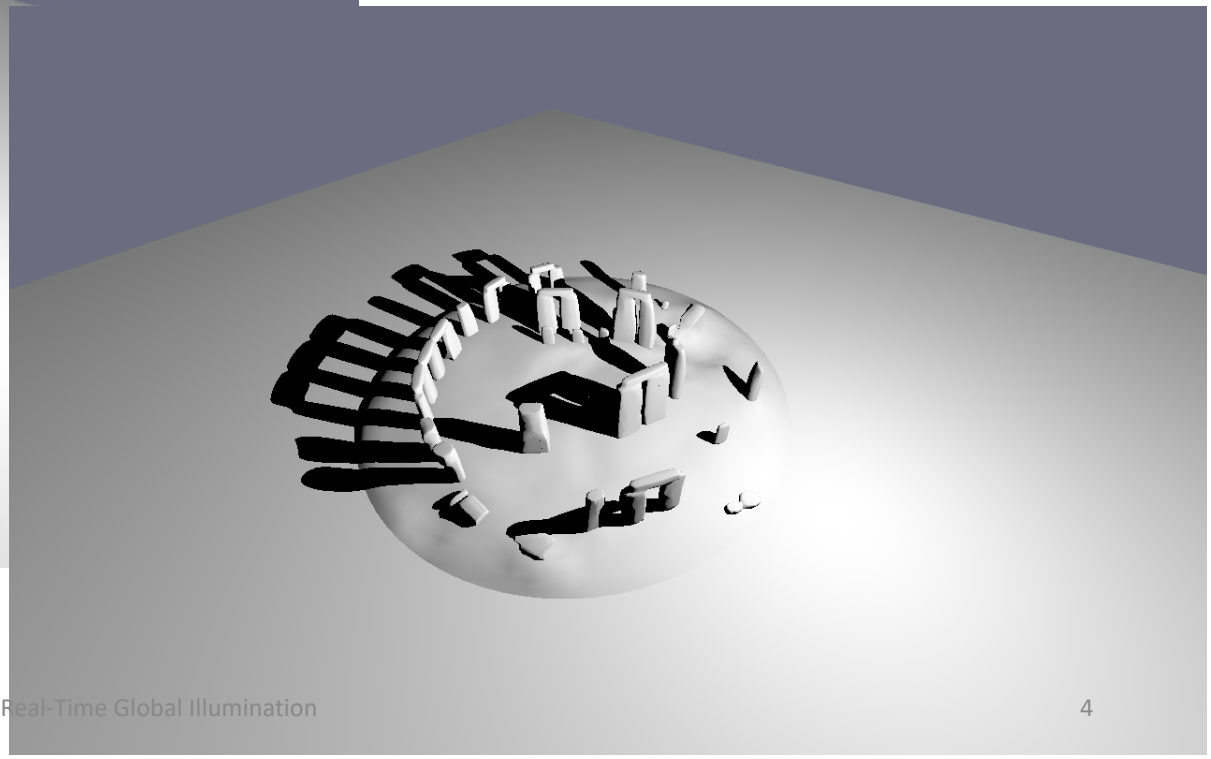
- Hard shadows
- Reflections
- Refractions
- Indirect Illumination
- Ambient Occlusion
- Caustics...



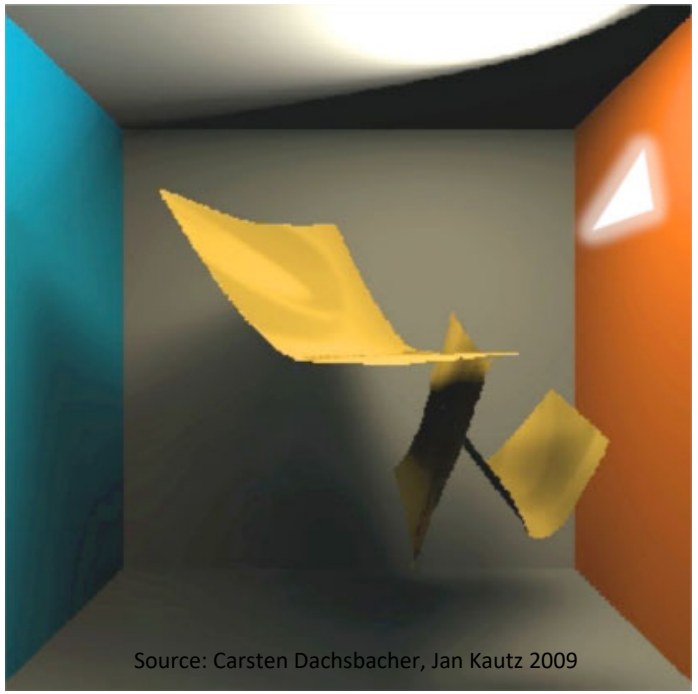
Increasing complexity

Adding Cast Shadows



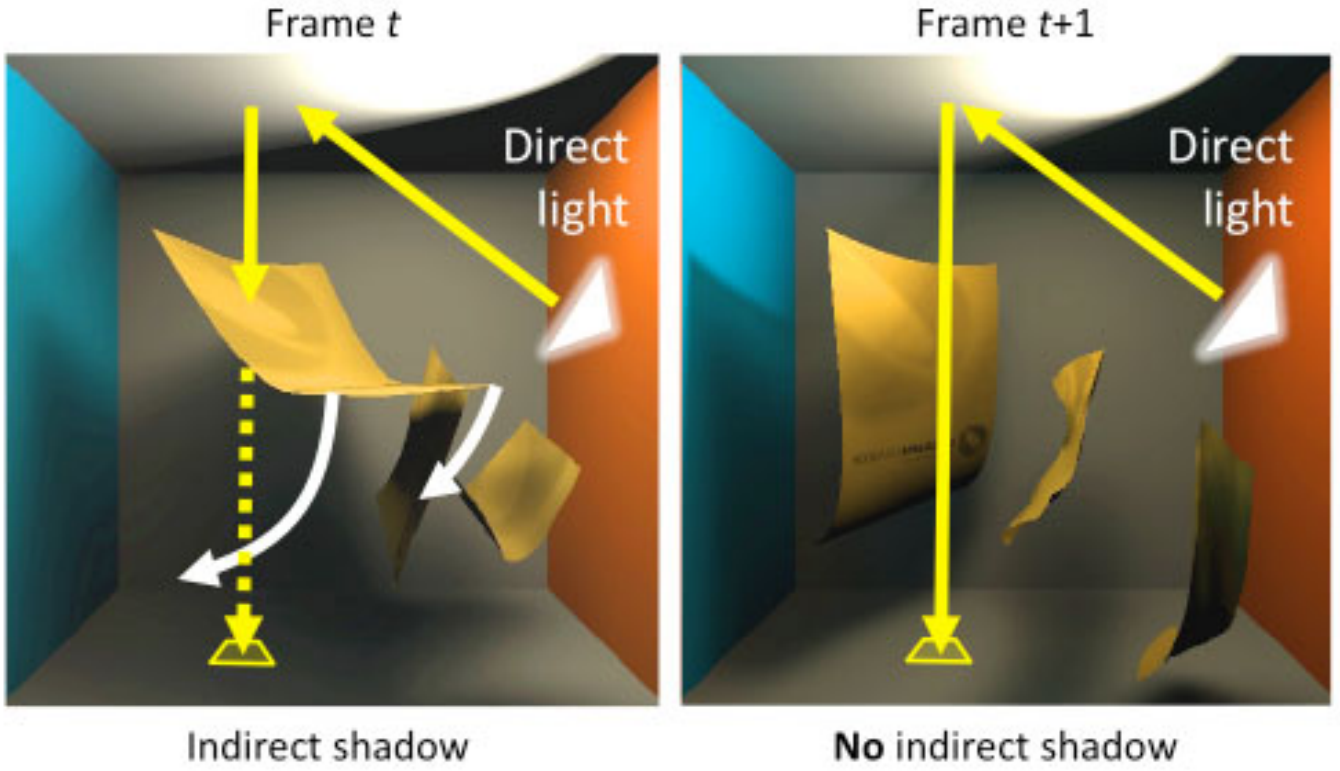


Adding Indirect Illumination

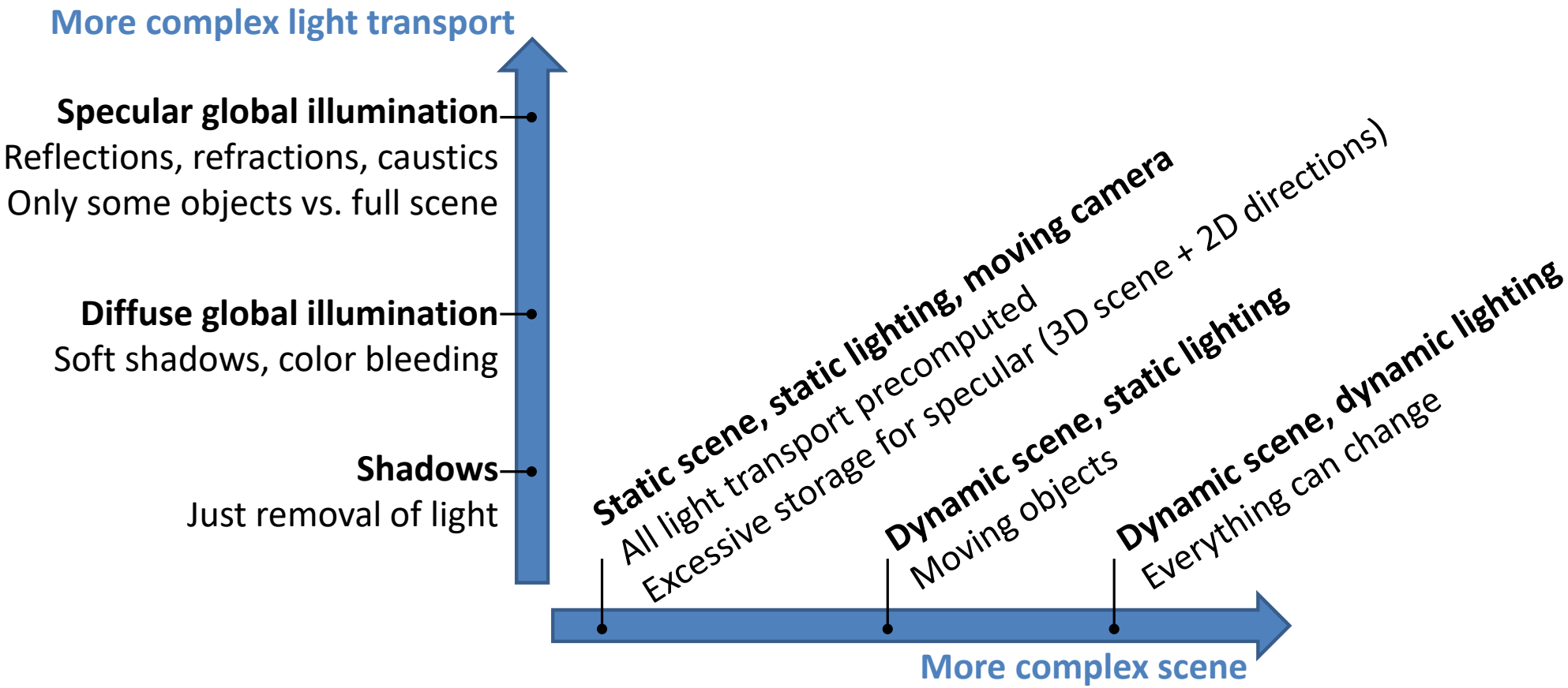


Source: Carsten Dachsbacher, Jan Kautz 2009

Shadows from Indirect Illumination



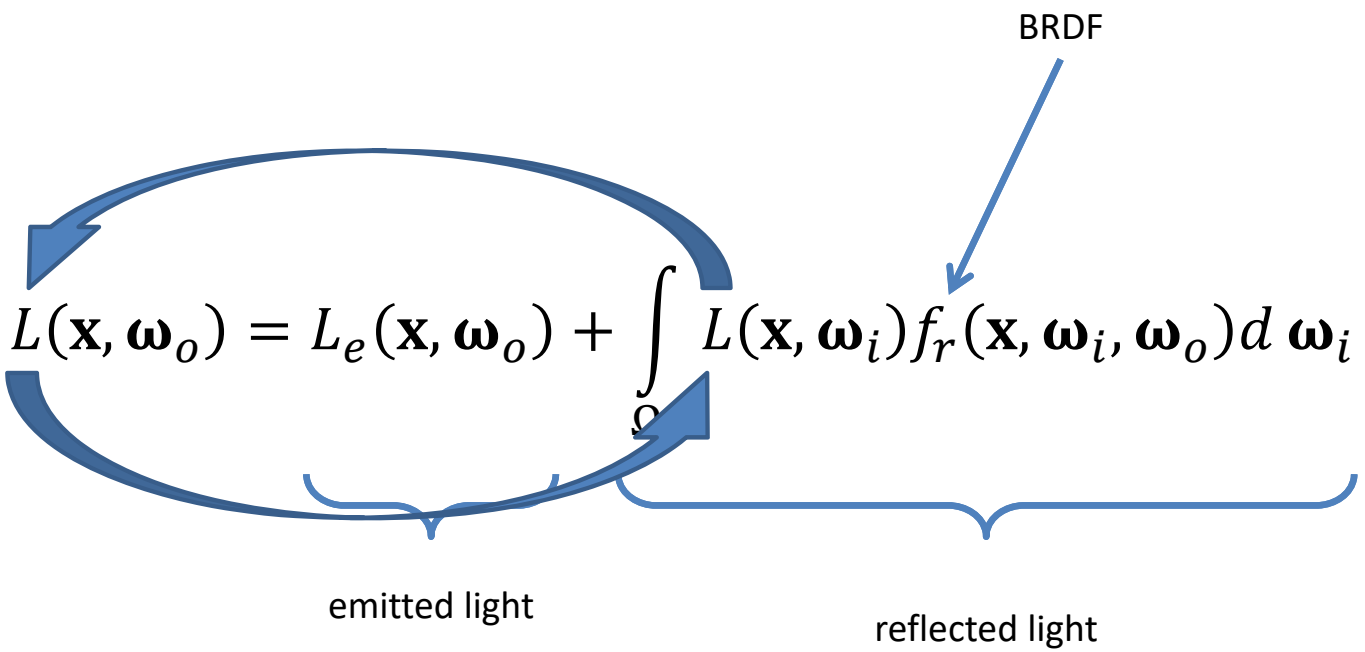
Two Dimensions of Complexity



Recap: Radiometry

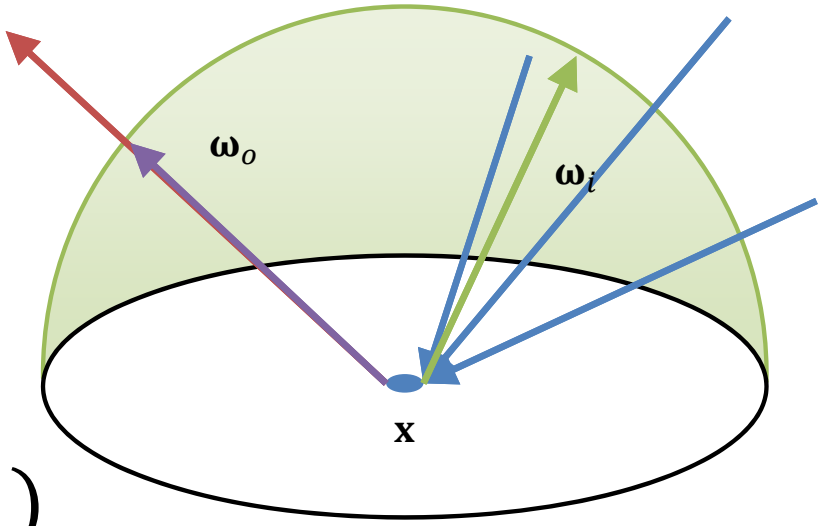
- What are we dealing with?
  - Light
  - Electromagnetic wave
- Light Field
  - Described by *plenoptic function* $L(\mathbf{x}, \boldsymbol{\omega})$
  - Amount of light passing through
    - Each point in space $\mathbf{x}$
    - In each direction $\boldsymbol{\omega}$
    - For every wavelength (usually just R, G, B)
    - For every moment (in practice, compute every frame separately)
    - ...

Rendering Equation



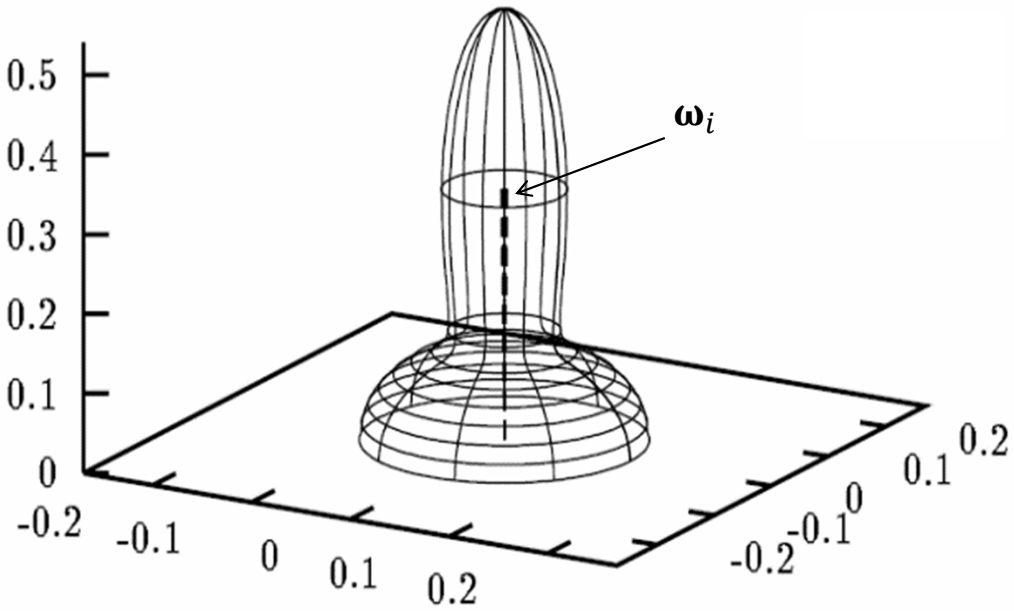
BRDF

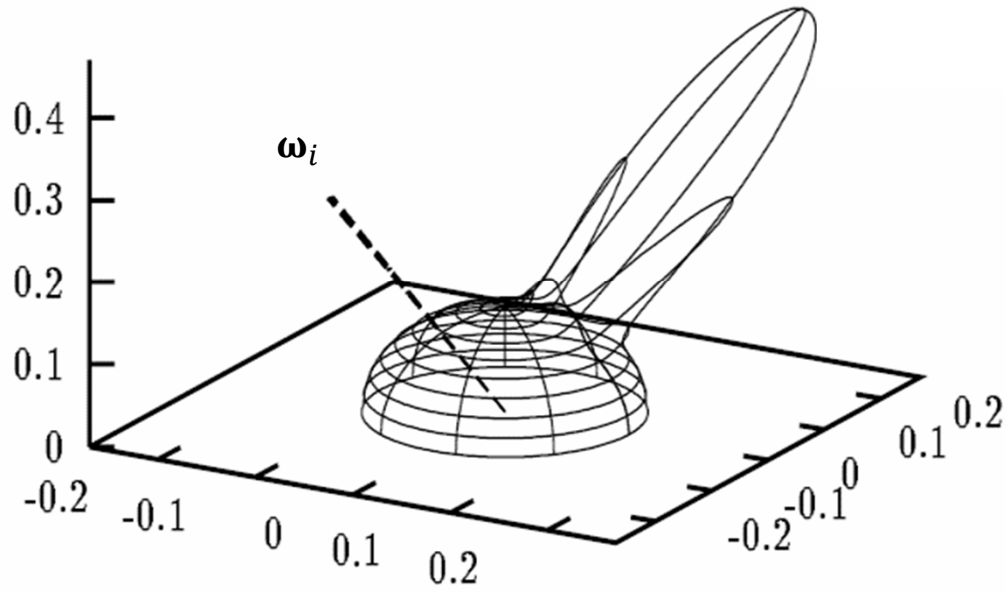
- Bidirectional Reflectance Distribution Function
- Describe how surface reflects light
  - At location $\mathbf{x}$
  - From incoming direction $\boldsymbol{\omega}_i$
  - To outgoing direction $\boldsymbol{\omega}_o$



$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$$

Example: Phong BRDF



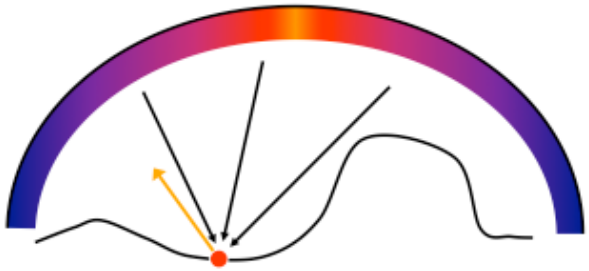


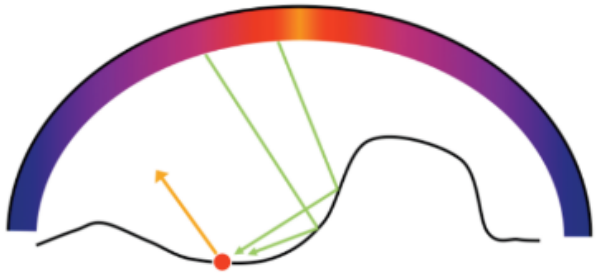
BRDF Properties

- Reciprocity
  - $f_r(\mathbf{x}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2) = f_r(\mathbf{x}, \boldsymbol{\omega}_2, \boldsymbol{\omega}_1) \;\; \forall \boldsymbol{\omega}_1, \boldsymbol{\omega}_2$
- Energy conservation
  - $\int_\Omega \; f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) d\, \boldsymbol{\omega}_i \leq 1 \;\; \forall \boldsymbol{\omega}_o$
- Positivity
  - $f_r(\mathbf{x}, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2) \geq 0$

Neumann Expansion of Indirect Illumination

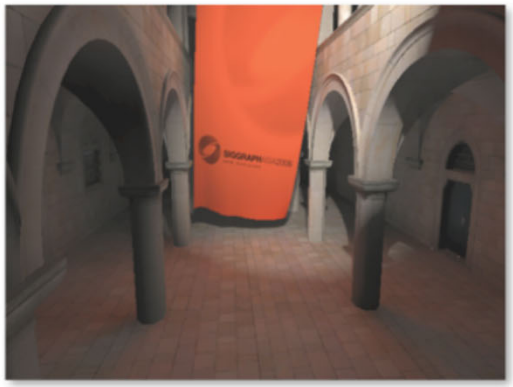
Outgoing radiance as infinite series: $L(\mathbf{x}, \boldsymbol{\omega}_o) = L_0(\mathbf{x}, \boldsymbol{\omega}_o) + L_1(\mathbf{x}, \boldsymbol{\omega}_o) + \cdots$





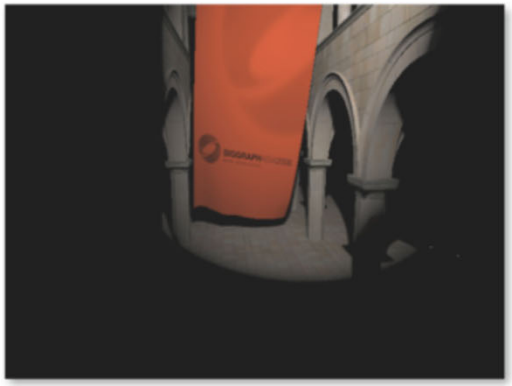
L_0 Direct lighting arriving at point $\mathbf{x}$

L_1 All paths from source that take 1 bounce



Direct + Indirect

$L_0(\mathbf{x}, \boldsymbol{\omega}_o) + L_1(\mathbf{x}, \boldsymbol{\omega}_o)$



Direct only

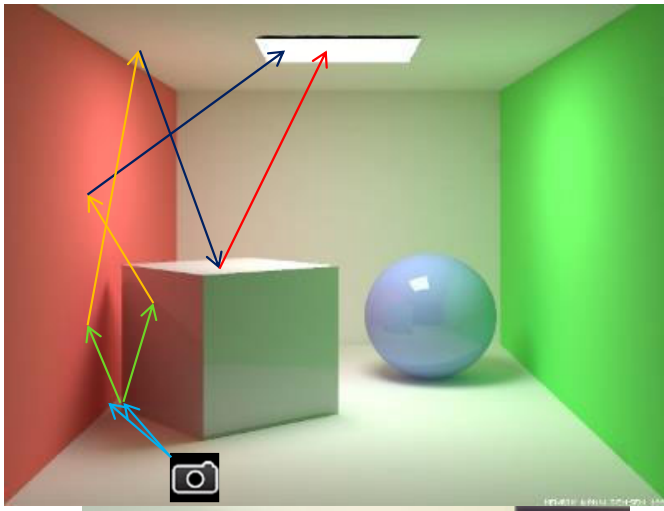
$L_0(\mathbf{x}, \boldsymbol{\omega}_o)$

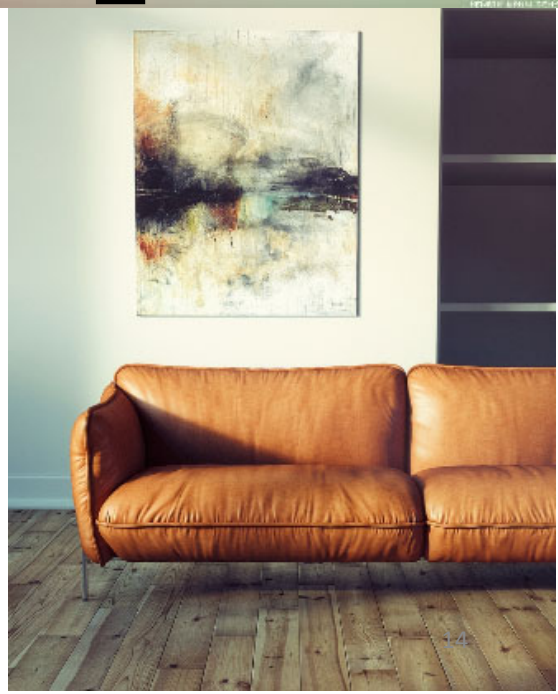
Indirect only

$L_1(\mathbf{x}, \boldsymbol{\omega}_o)$

Path Tracing

- Reference solution
- Solve rendering equation numerically
  - Follow all possible ray paths (specular+diffuse)
  - Monte Carlo approach → noise
- No fundamental limits concerning realism
- Not really efficient
  - Hard to find a path that connects camera and light source with strong (specular) transport





Two-Pass Global Illumination

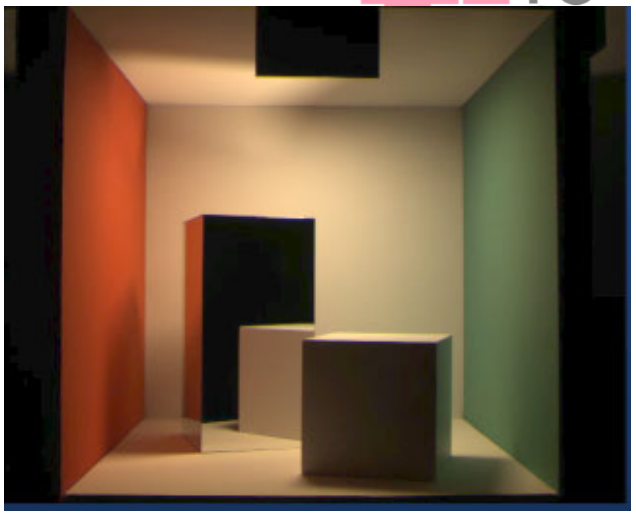
- 1st pass computes light transport in the scene
- 2nd pass collects lighting information to generate final image
- Every pass chooses independently
  - Type of light transfer (diffuse *or* specular)
  - Rendering method (ray tracing *or* rasterization)
  - Update rate (once *or* sometimes *or* every frame)

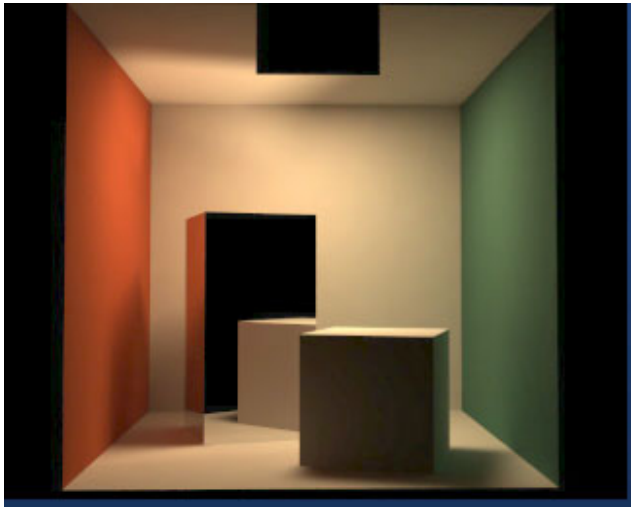
Examples of Two-Pass Methods

Method	Transport	1st pass result	2nd pass
Radiosity	Diffuse	Light maps	Render using texture maps
Photon mapping	Diffuse + some specular	Photon map	Raytracing
…			

Radiosity

- Pass 1: Create lightmaps
  - Finite elements approximation
  - Discretize scene into patches
  - Assume everything is perfectly diffuse
  - Light transport → linear equation system
  - Solve equation system
- Pass 2: Render using texture mapping



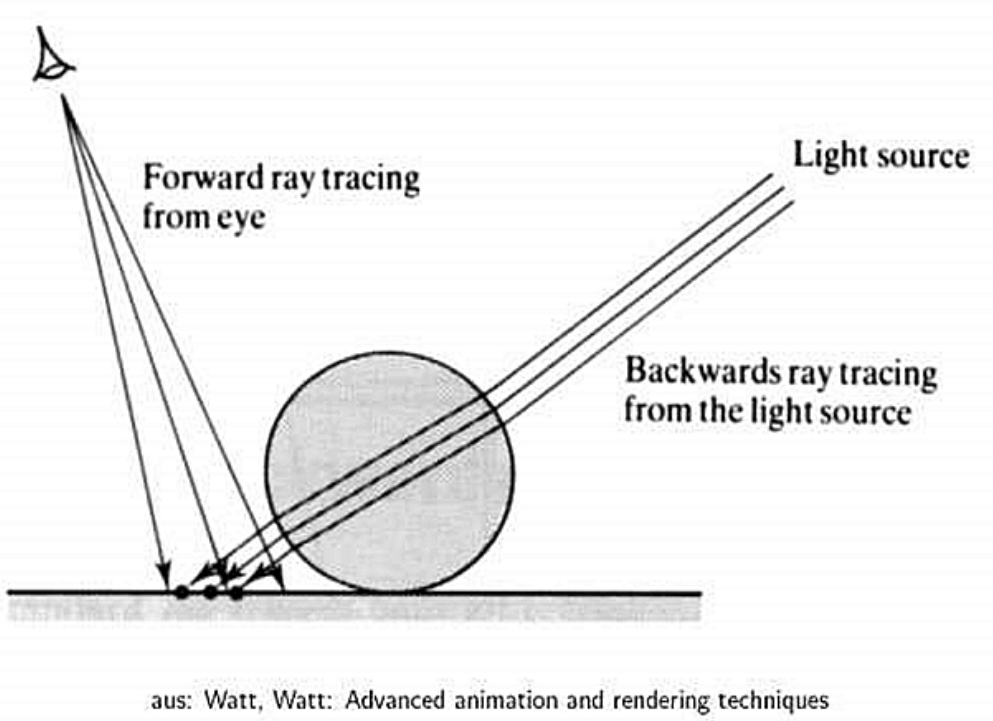


Source: Cornell University

Photon Mapping

Two passes

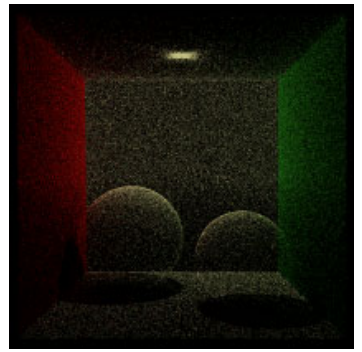
1. Photon map creation
   - Shoot photons from light source
   - Store particles hitting diffuse surfaces in data structure
2. Rendering using photon map

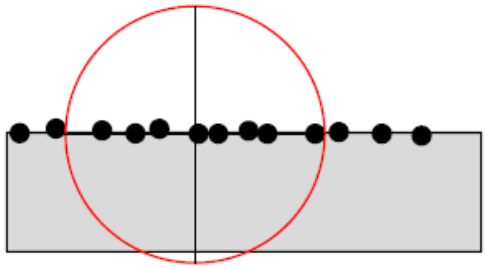


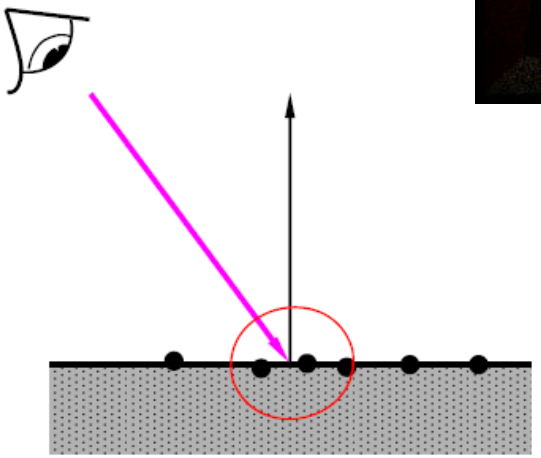
aus: Watt, Watt: Advanced animation and rendering techniques

Photon Mapping - Creation

- At point *x*: collect photons contained in sphere of radius *r*
- Number of photons = intensity at point *x*
- Can simulate complex light phenomena (e.g., caustics)
- Requires fast spatial search (kd-tree)

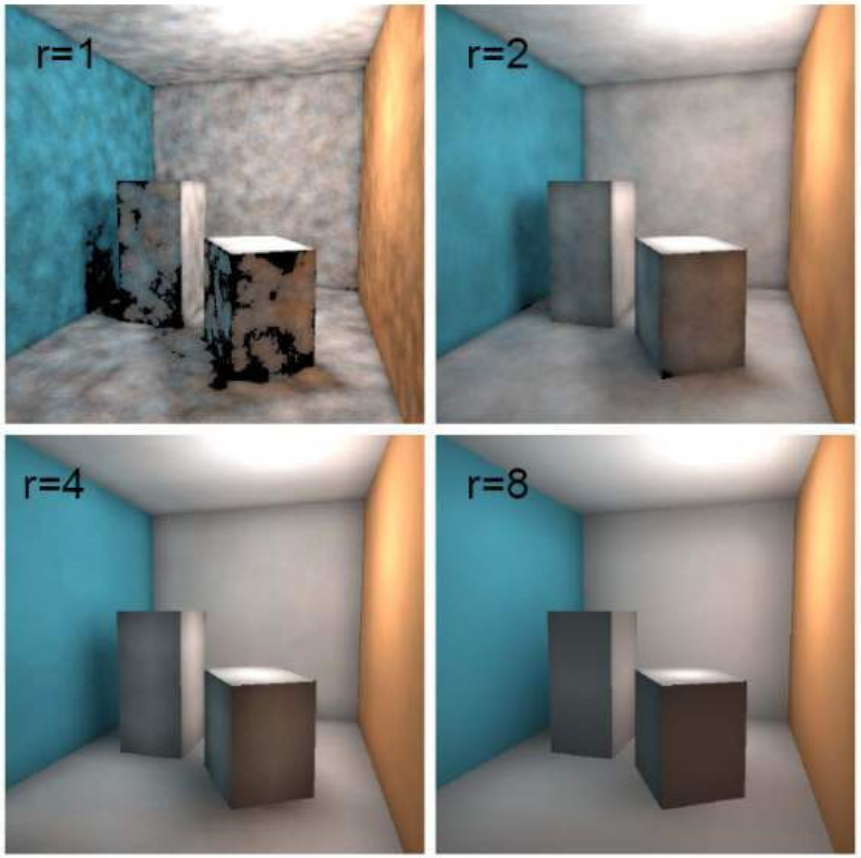






Photon Mapping - Rendering

Small radius
→noise



Large radius
→loss of detail

Render Cache

- Data structure connecting the passes
- What data to store
  - Radiance cache (outgoing illumination for every point), *or*
  - Irradiance cache (incoming illumination for every point)
- How to index
  - Points only on surfaces (needs a surface index, like kd-tree)
  - Points everywhere in space (needs a voxel grid or octree)

Render Cache Organization

- Organized only by position
  - Photon maps: sparse kd-tree
  - Classic Radiosity: radiance on surface points or patches
- Organized only by orientation
  - Environment map: cubic radiance map
- Organized in projective space
  - 3D positions in a 2D depth map
  - Shadow mapping, instant radiosity
- Organized by both position and orientation
  - Position as main index
  - Orientation as sub-indexed, often compression using Spherical Harmonics (SH)
  - Irradiance volumes *or* light propagation (=irradiance) volumes

How to Compute Illumination in Real Time

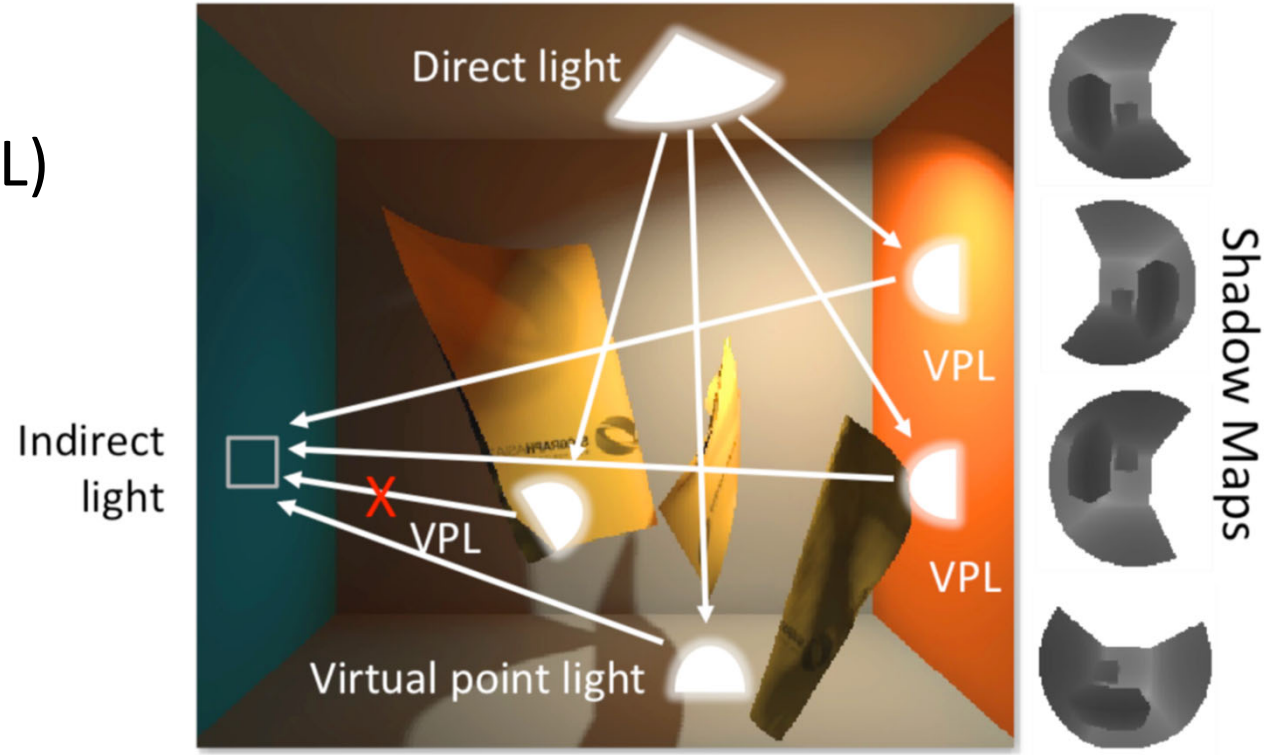
- General global illumination
  - Extremely high complexity
  - Global transport does not naturally fit to online rendering
- Where can we approximate?
  - Semi-global: global effects only for some objects
  - Low-order: allow only 1-2 bounces of light
  - Ignore specular light transport
  - Static scene
  - Static light sources
- Rasterization is more efficient than raytracing

Real-Time Two Pass Methods

Method	Transport	1st pass result	2nd pass
Radiosity	Diffuse	Light maps	Render using texture maps
Photon mapping	Diffuse + some specular	Photon map	Raytracing
Instant Radiosity	Diffuse	Shadow map (rasterization)	Deferred rendering (rasterization)

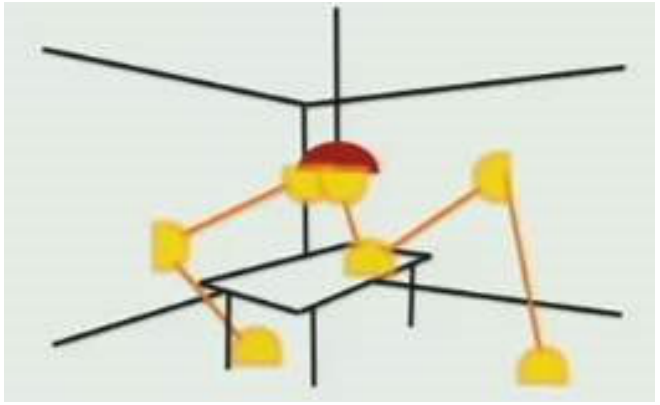
Instant Radiosity

- Idea: model the light bounces as light sources
- Virtual point lights (VPL)
- Convert indirect into direct light transport



Virtual Point Light Sources

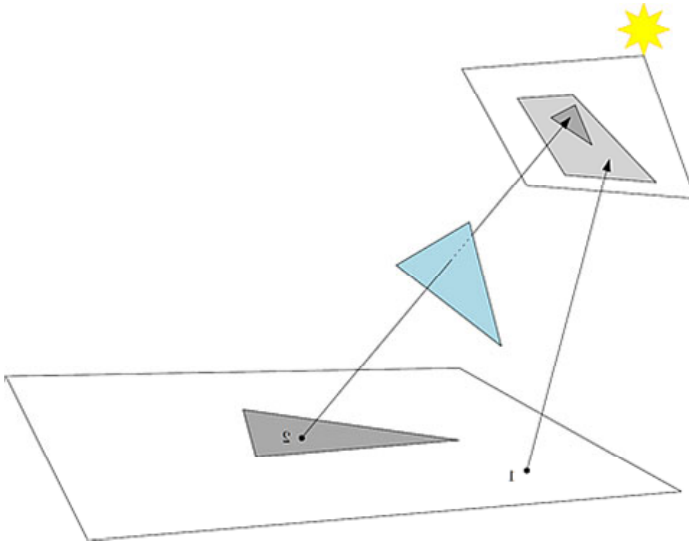
- Pass 1: VPL generation
  - Shoot photons from light source
  - Follow them through scene
  - At each hit point: create VPL
  - Russian roulette to end path
  - One shadow map for each VPL

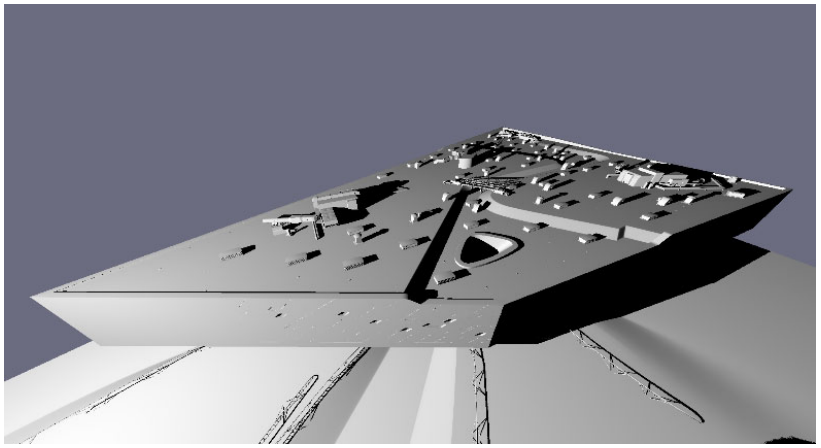


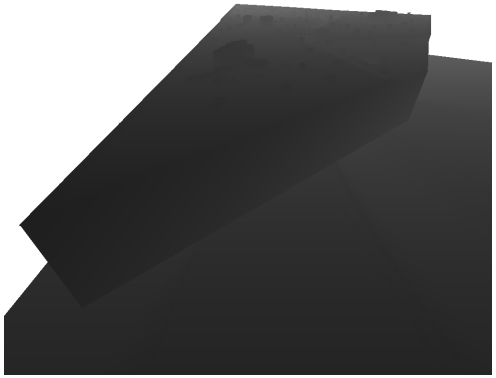
- Pass 2: Deferred rendering
  - Render considering all the VPL and their shadow maps

Recap: Shadow Mappi

- Shadows are a visibility problem
  - Point in shadow ←→ invisible for light source
  - Shadow maps use depth buffering

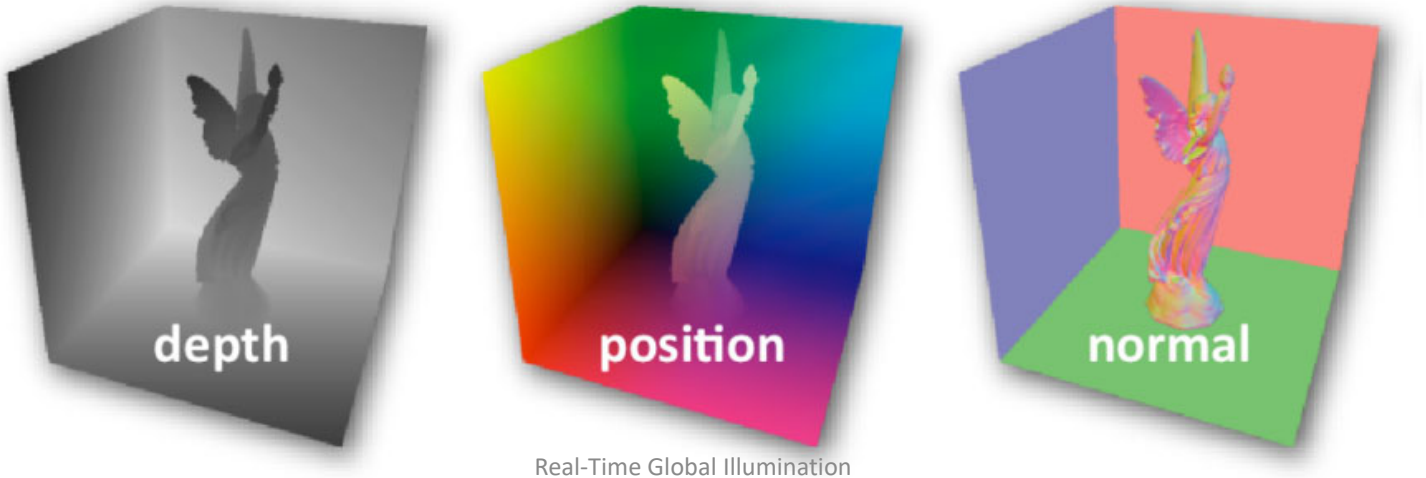






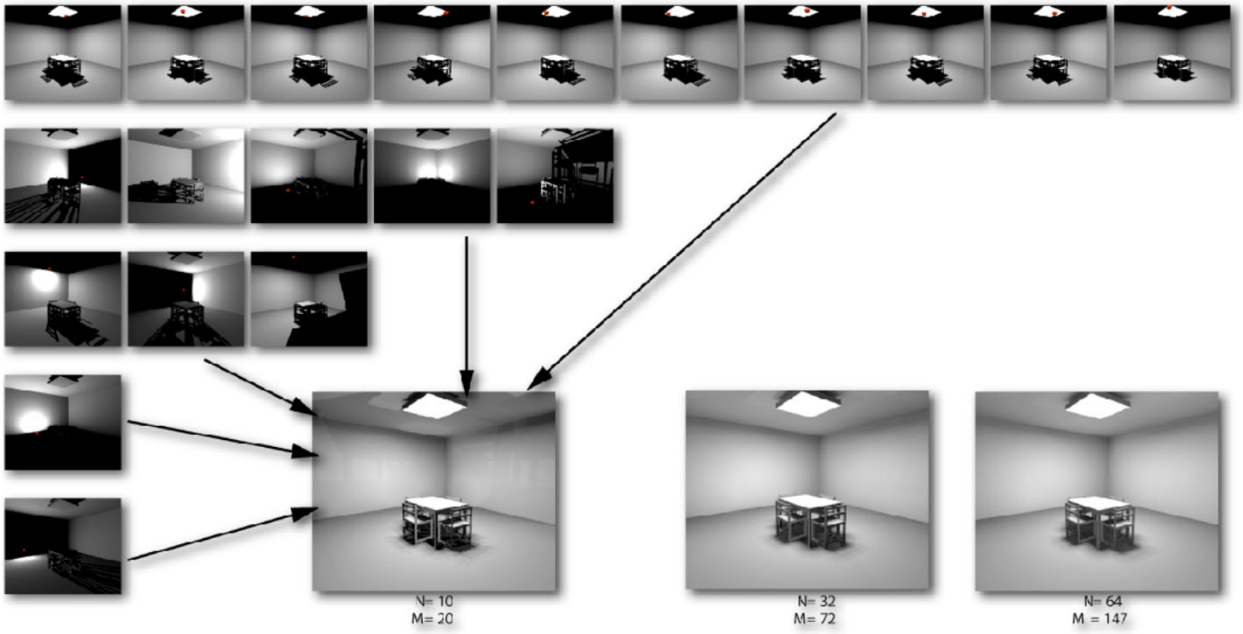
Recap: Deferred Shading

- Traditional shading requires many passes for many light sources
- Deferred shading rasterizes only once and stores in G-buffer
- Compute shading in G-buffer in separate task
- Decouples geometric complexity from lighting complexity



VPL Distribution

- VPLs are distributed according to
  - ρ^{i} = av. Reflectance, *i* = bounce

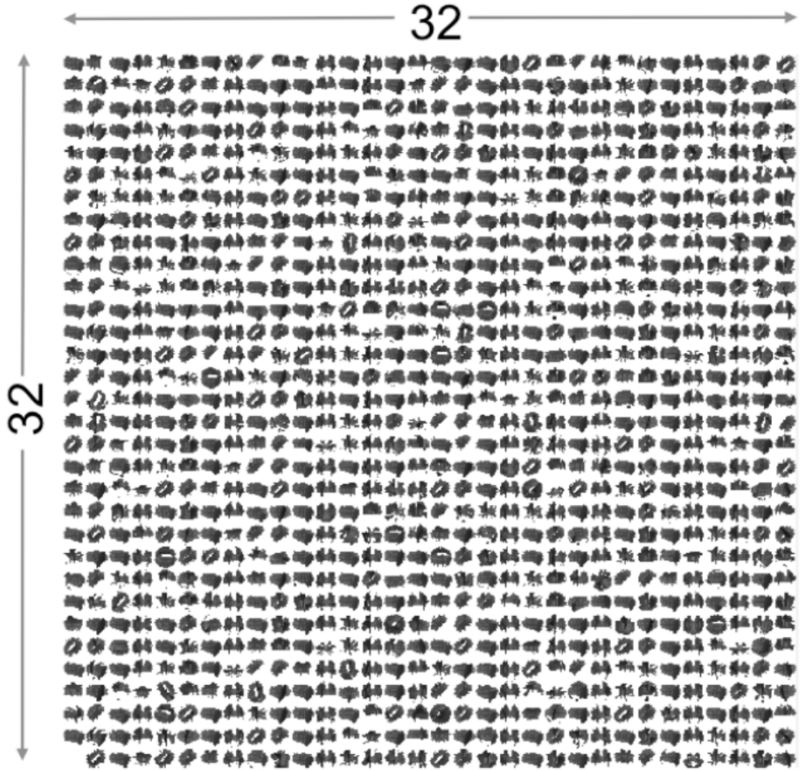



SIGGRAPH

Instant Radiosity

- Scene: 100K triangles
- Illumination: 1024 VPLs

→Drawing 100M triangles is too much...



Incremental Instant Radiosity

- Optimization assumes a semi-static scene
- Exploit frame-to-frame coherency
- Reuse VPLs from previous frames
  - Only a couple of new shadow maps per frame

Imperfect Shadow Maps

- VPL approaches do not require accurate geometry

High-Quality Depth

Low-Quality Depth
(20% corrupted)

Creating Imperfect Shadow Maps

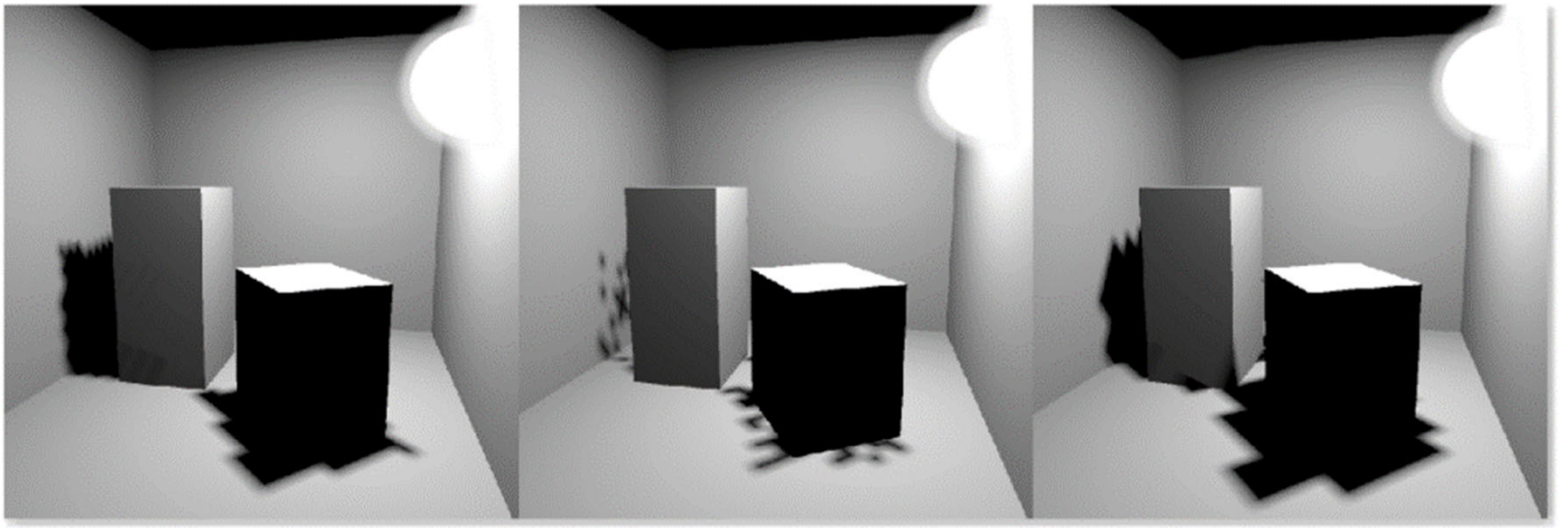
- Parabolic (omnidirectional) shadow map
- Coarse approximation of scene as point cloud
- Limited number of points

Classic
Triangles

Imperfect
Smaller points
Fewer points

Imperfect
Pull-Push
Hole Filling

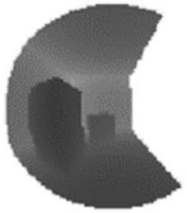
Dense Geometry with Pull-Push Interpolation



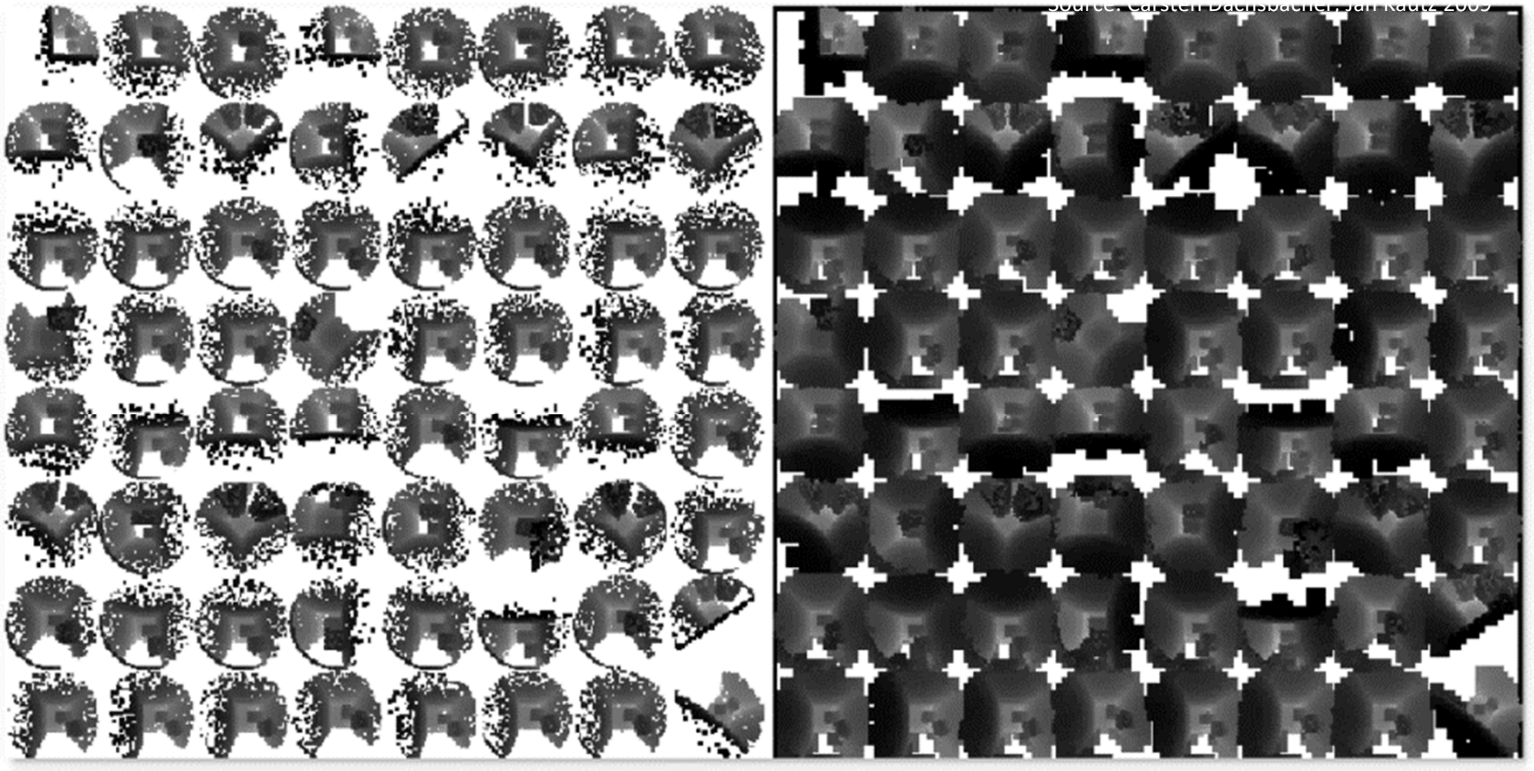
Classic

Without pull-push

With pull-push



Comparison Pull-Push

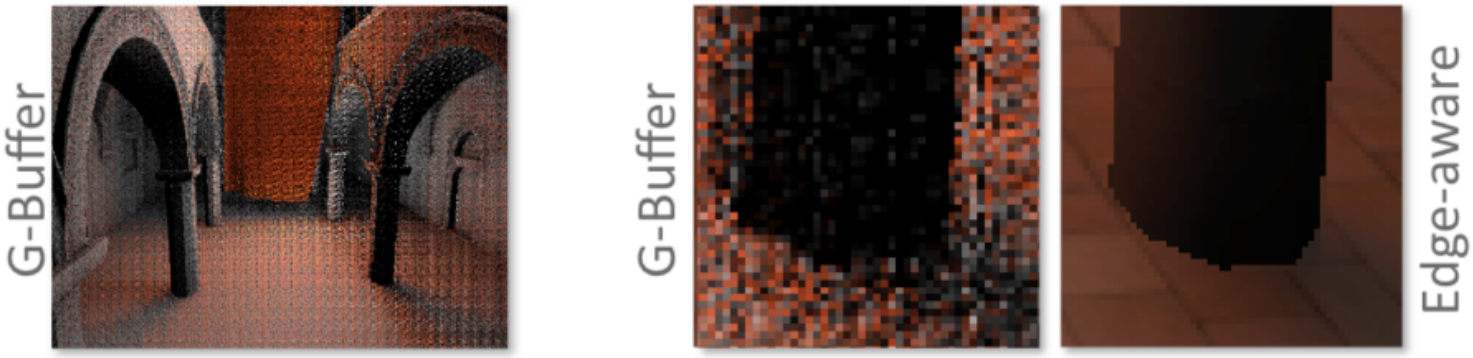


Without pull-push

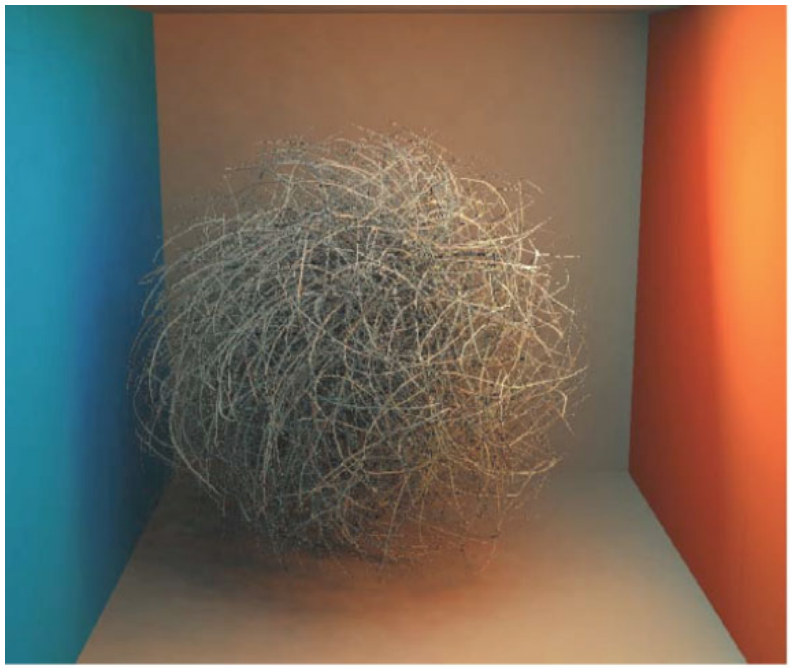
With pull-push

Interleaved Sampling

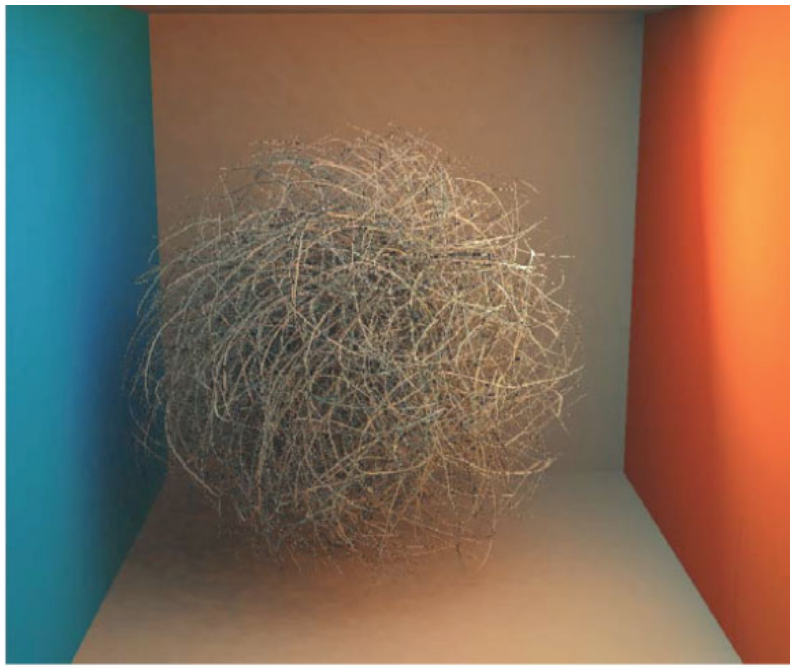
- Sample light sources randomly
  - Only few light sources per pixel
- Average results
  - Using edge aware (bilateral) filter



Imperfect Shadow Maps Results 1

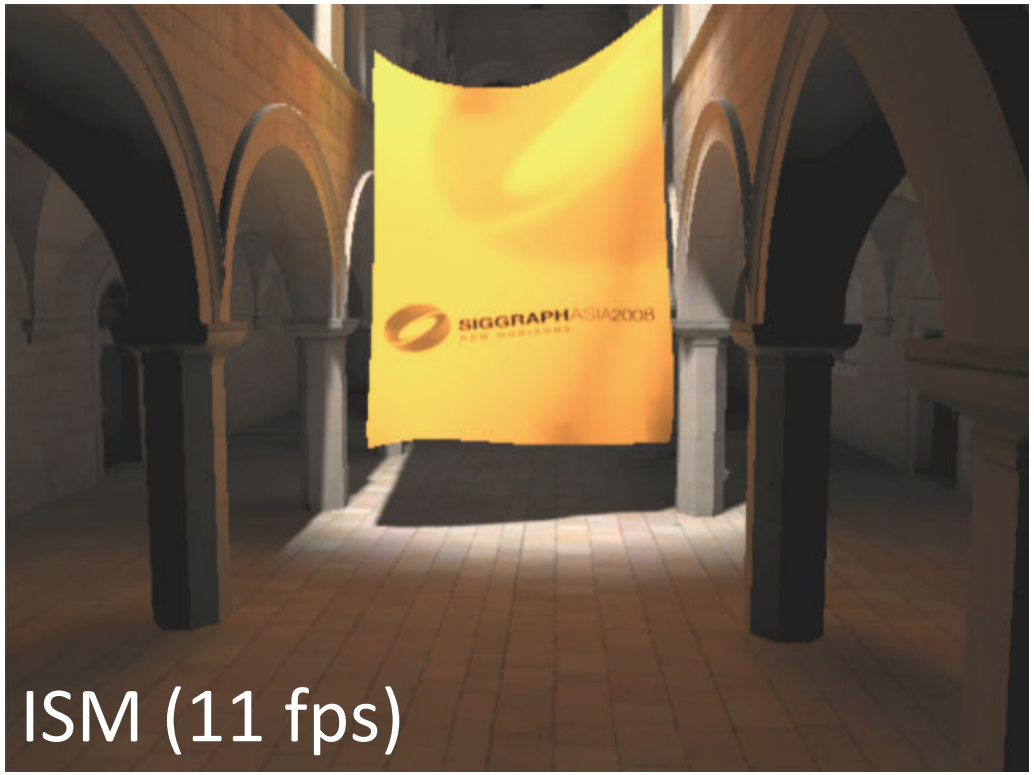


Reference *(hours)*



Imperfect Shadow Maps *(15Hz)*

Imperfect Shadow Maps Results 2



More Results and Effects

Cornell box horse

Christo‘s Sponza

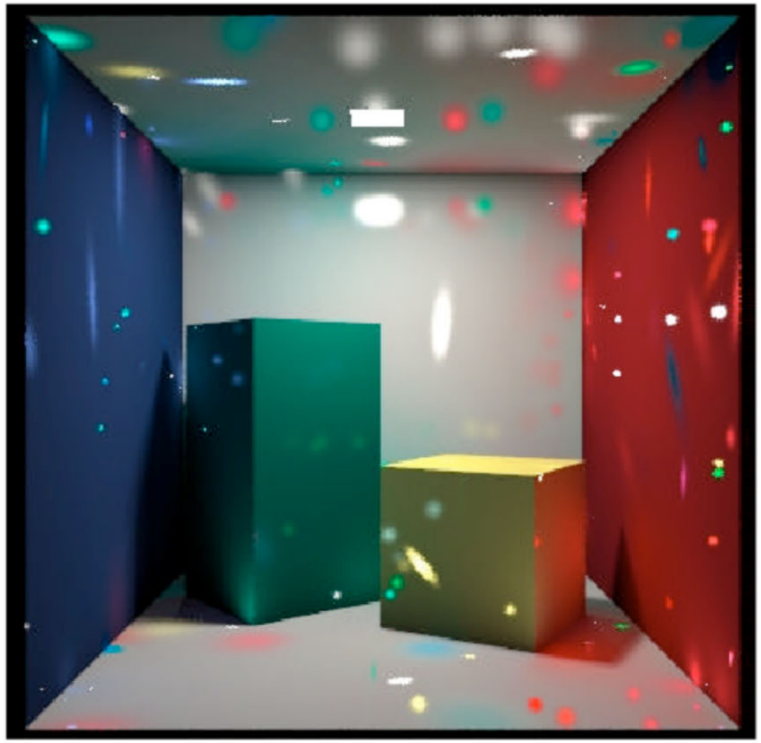
Multiple bounces

Caustics

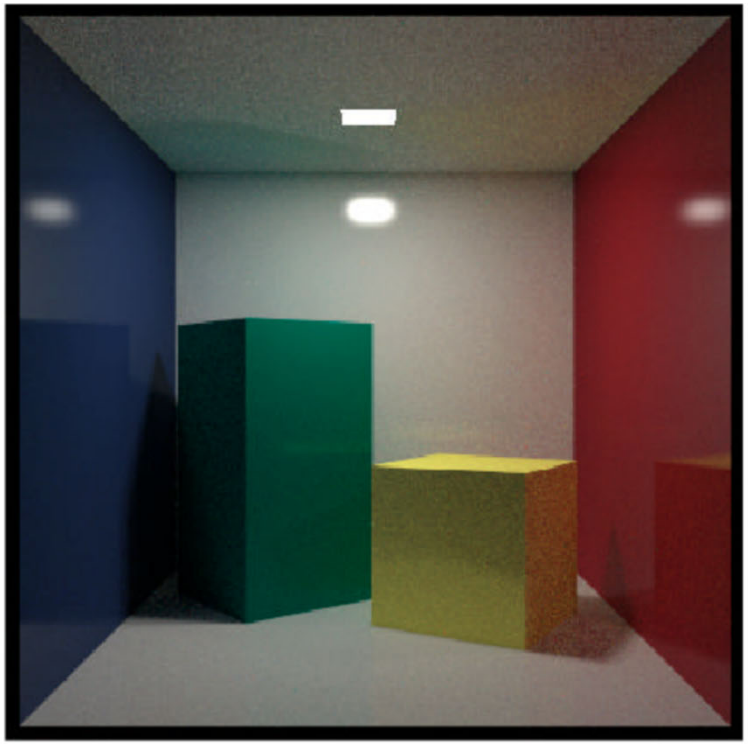
Source: Carsten Dachsbacher, Jan Kautz 2009

Problems of Instant Radiosity

- Scalability
- Performance
  - Can reach interactive framerates
  - But still not fast enough for games



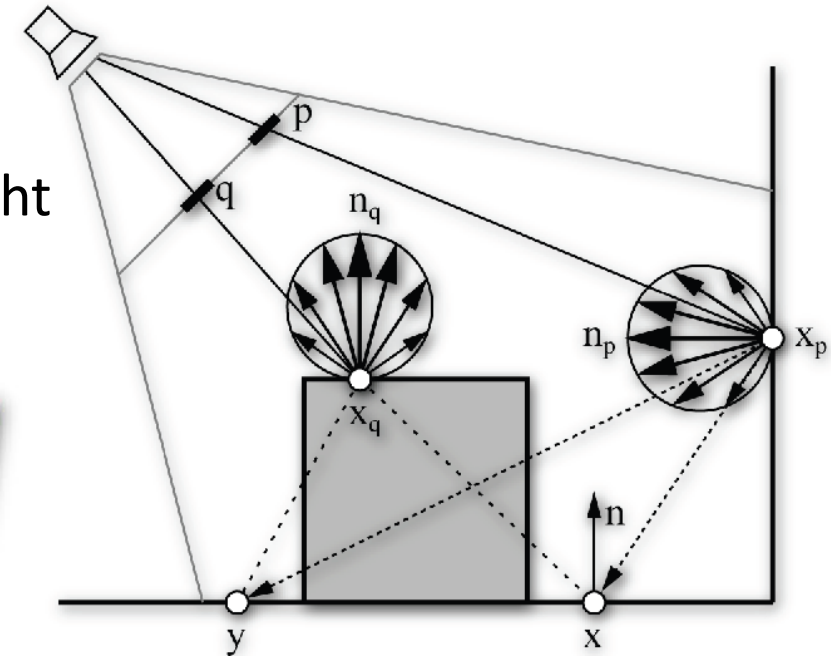
Instant Radiosity



Path Tracing

Reflective Shadow Maps

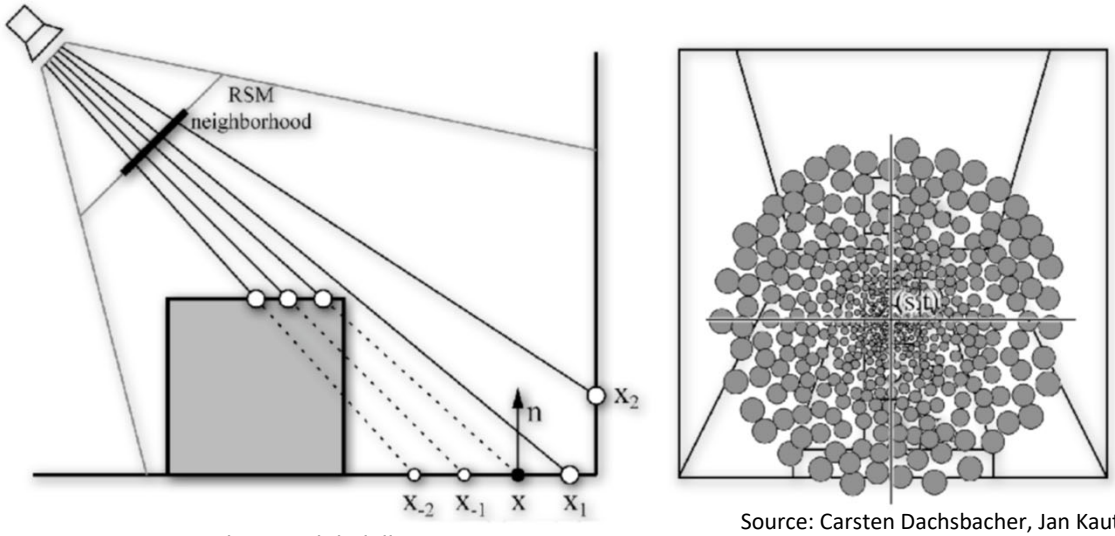
- Restrict to one bounce
- Single bounce illumination from surfaces seen by light source
- Each pixel is small light source
- During rendering of shadow map
  - Store additional information on received light
  - Called a *reflective shadow map* (RSM)



Source: Carsten Dachsbacher, Jan Kautz 2009

Rendering with Reflective Shadow Maps

- Gather illumination from RSM
- Too many pixel lights
- Restrict to samples close to current location

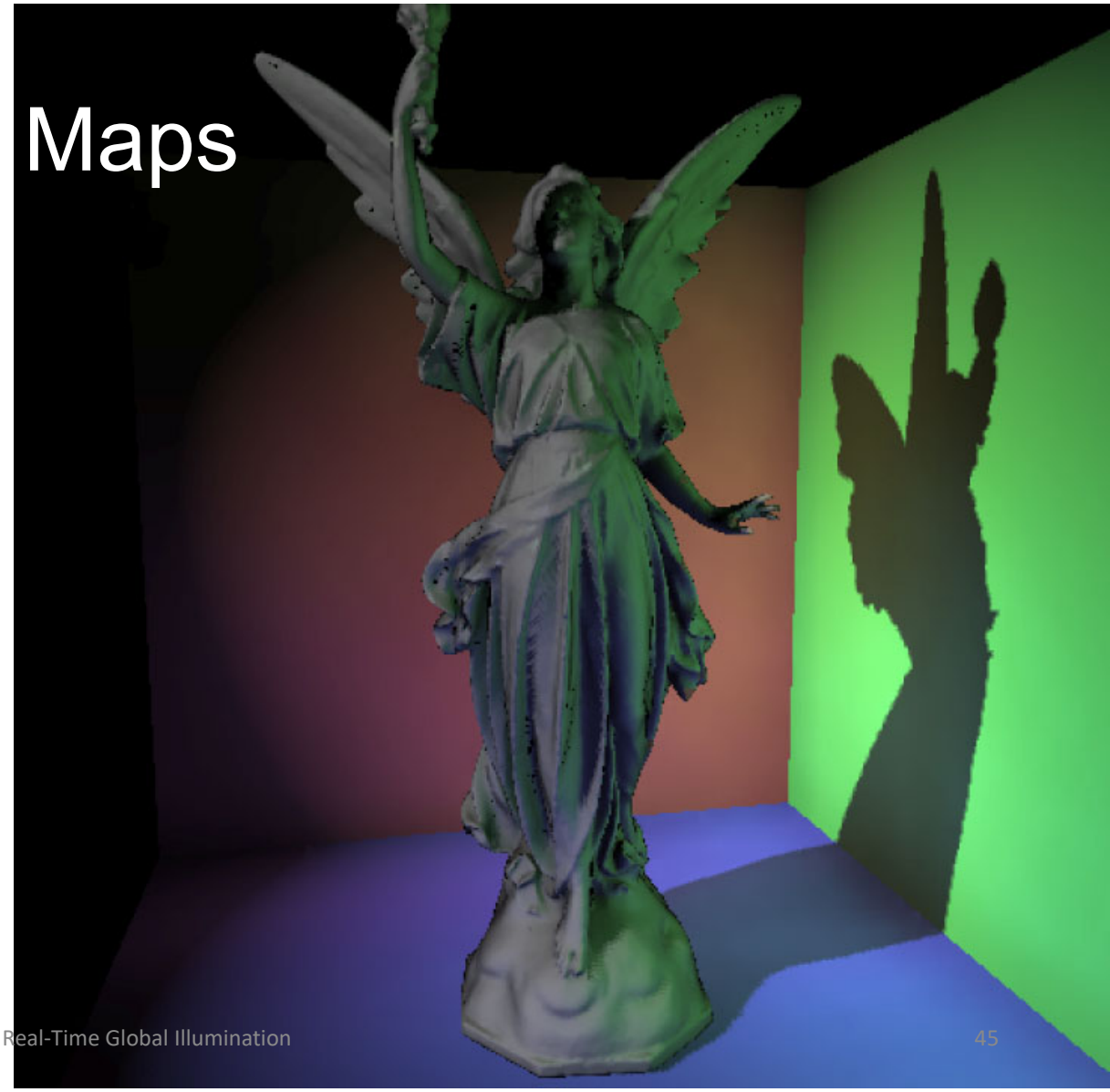


Source: Carsten Dachsbacher, Jan Kautz 2009

Reflective Shadow Maps Result

Reflective Shadow Maps

- Crude approximation
  - Diffuse reflectors
  - No indirect shadows
  - …



Precomputed Radiance Transfer

- Realistic, interactive illumination of complex scenes
  - Complex materials
  - Dynamically changing lighting environment
  - General light sources
  - Shadows, interreflections, translucency
- Do not want to use real-time ray tracing

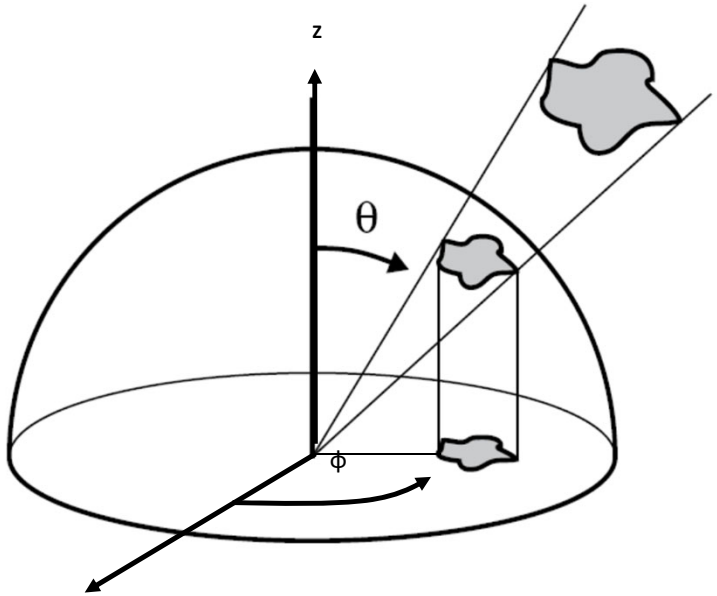
Method

Light sources and **light transport** are independent

1. Step: Precompute light transport
   - E. g., with ray tracing
   - Offline, slow and costly
   - Store compressed representation
     - Light sources as an environment map
     - Light transport function for each surface point
2. Step: evaluate scene illumination
   - During rendering, in real-time

Spherical Harmonics

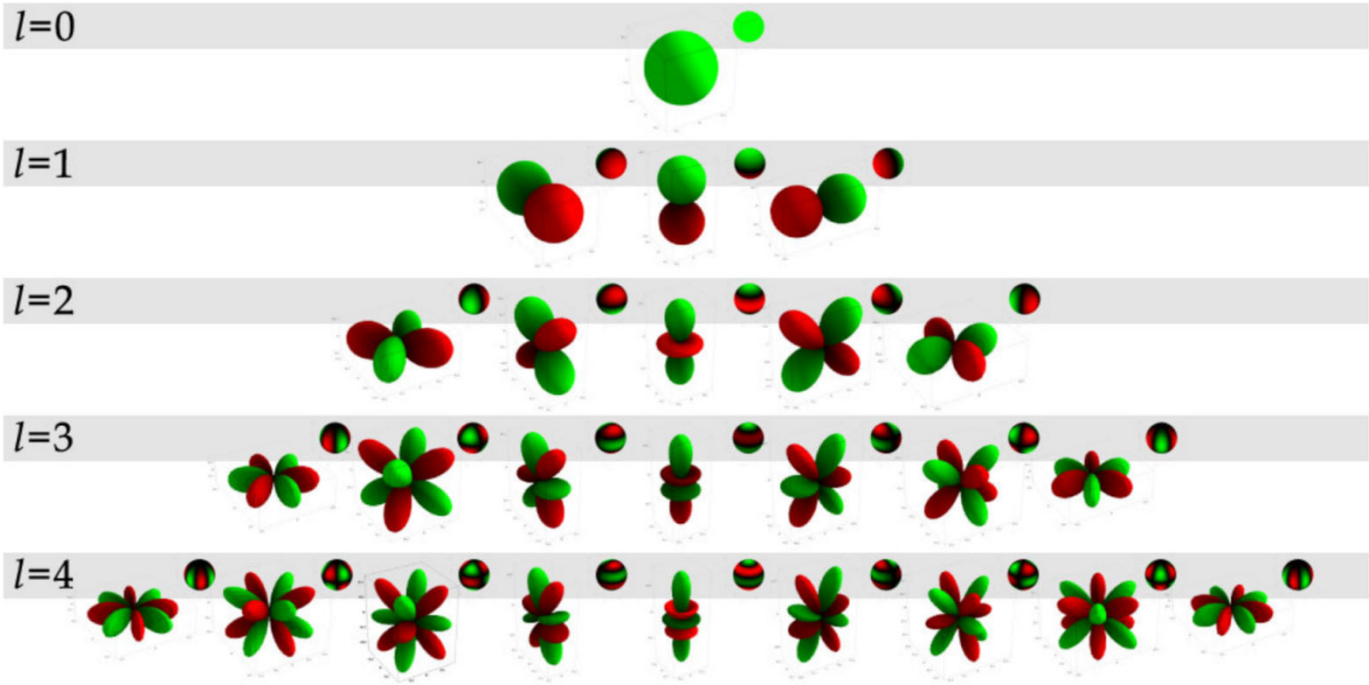
Base functions for representing functions with a spherical domain



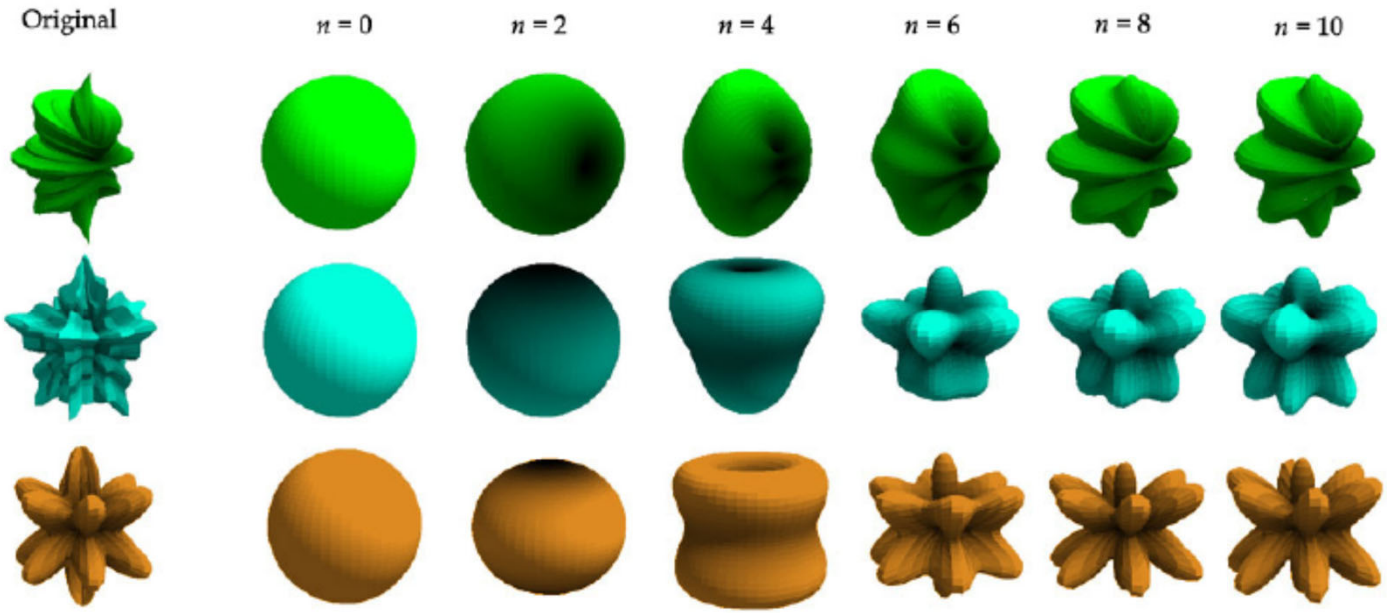
$$(\sin\theta\cos\varphi \quad \sin\theta\sin\varphi \quad \cos\theta) \mapsto (x \quad y \quad z)$$

Spherical Harmonics Bands

SH ... Basis functions for representing an arbitrary spherical function as a linear weighted sum

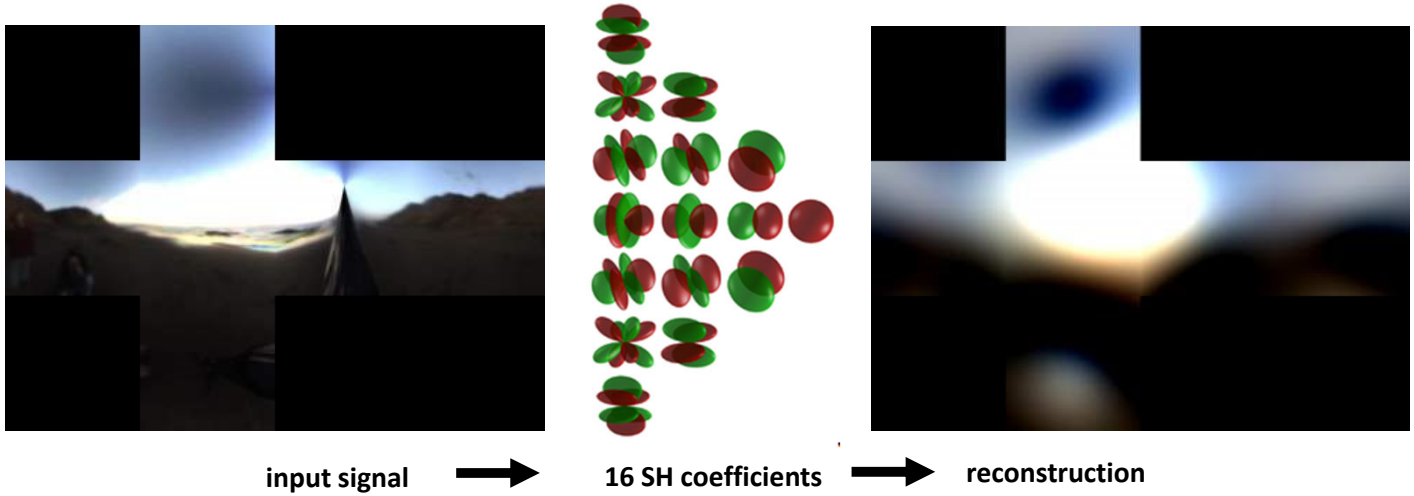


Examples for SH-Reconstruction



SH Representation for Environment Maps

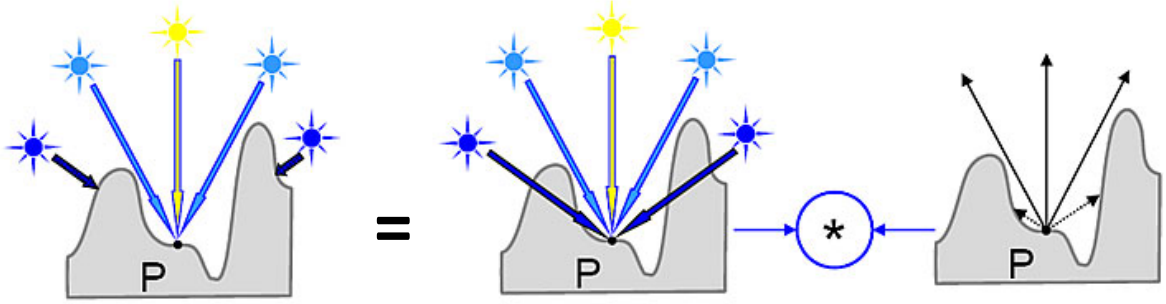
- HDR environment map approximated by SH
- 4 bands = 16 coefficients of SH
- Computation of coefficients by projection



SH Representation for Light Transfer

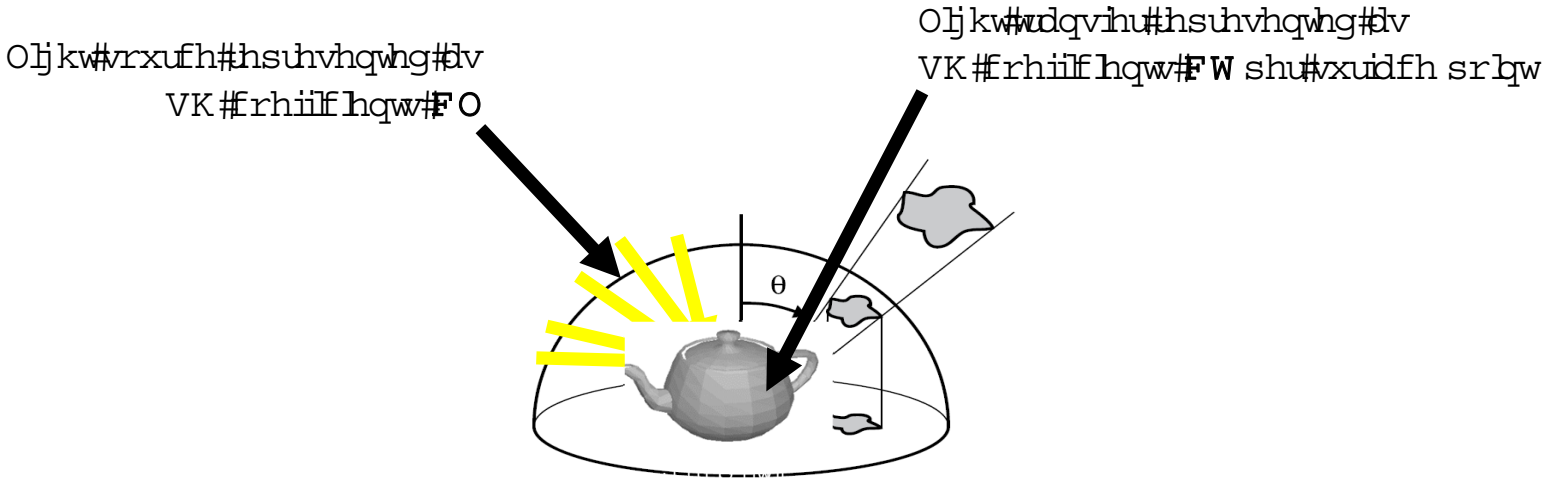
- Directional visibility, computed as for AO
- But stored per-point as SH coefficients (vertex texture)

$$I_{dir}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{in}(\omega_i) V(\omega_i) \cos\theta_i \Delta\omega$$



Shading | Illumination L_{in} | Visibility V

PRT Rendering



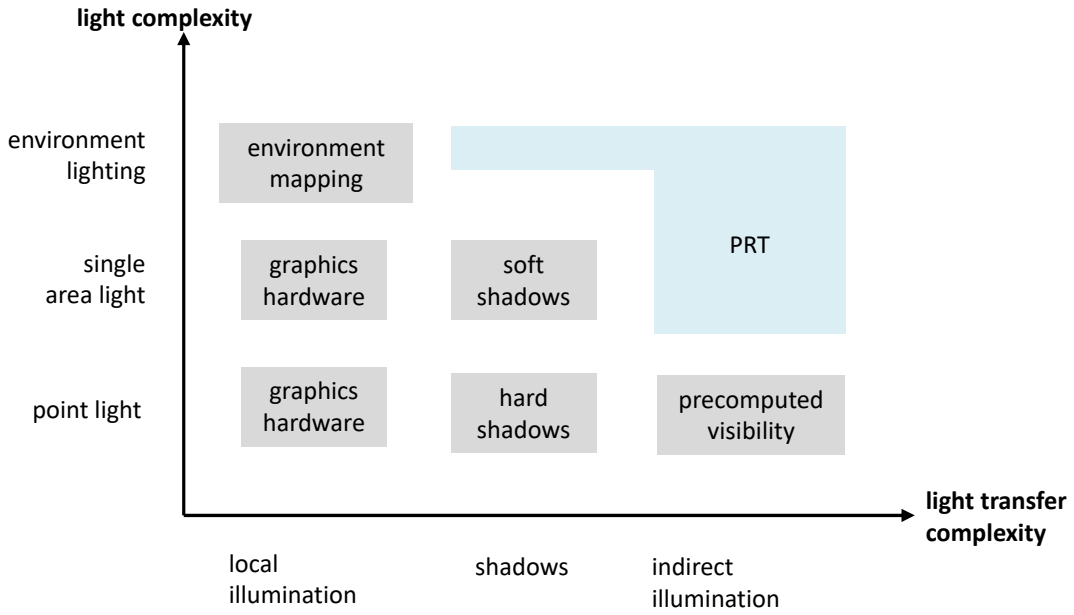
light coefficients **CL** →

light transfer coefficients **CT** →

rendering equation
I_{dir} = **CL** dot **CT** →

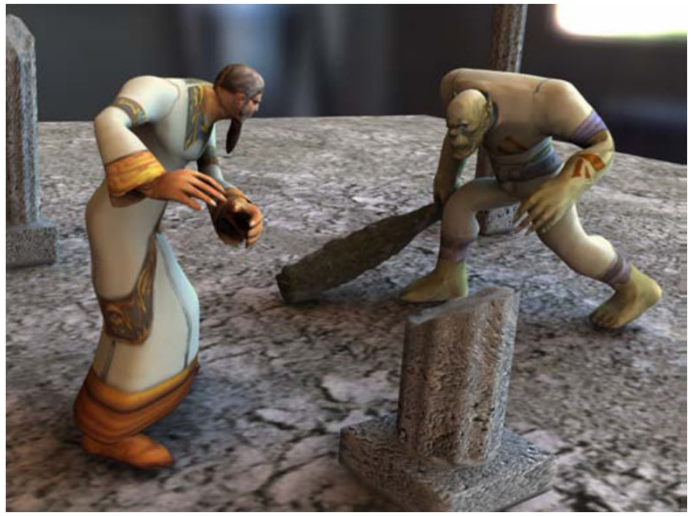
image

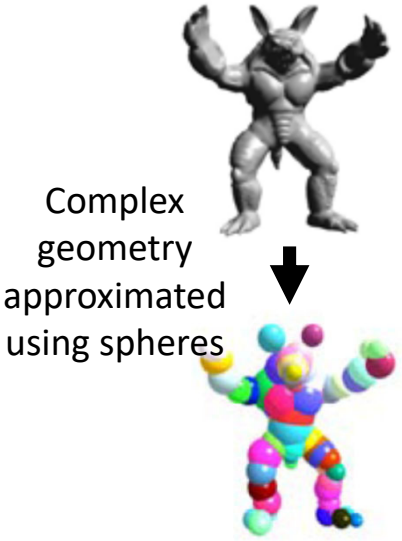
PRT – Applications



SH Lighting for Dynamic Geometry

- Dynamic geometry





PRT – Summary

- Advantages
  - Fast rendering (1 dot product)
  - Dynamic lighting environments
  - Supports complex light transfer (ambient occlusion and interreflections)
- Disadvantages
  - Only efficient for lighting environments of low-frequency
  - Typically 9 to 16 SH coefficients (3 to 4 bands)
  - Static scenes
  - Only diffuse surfaces (if using SH-based compression)

Questions?