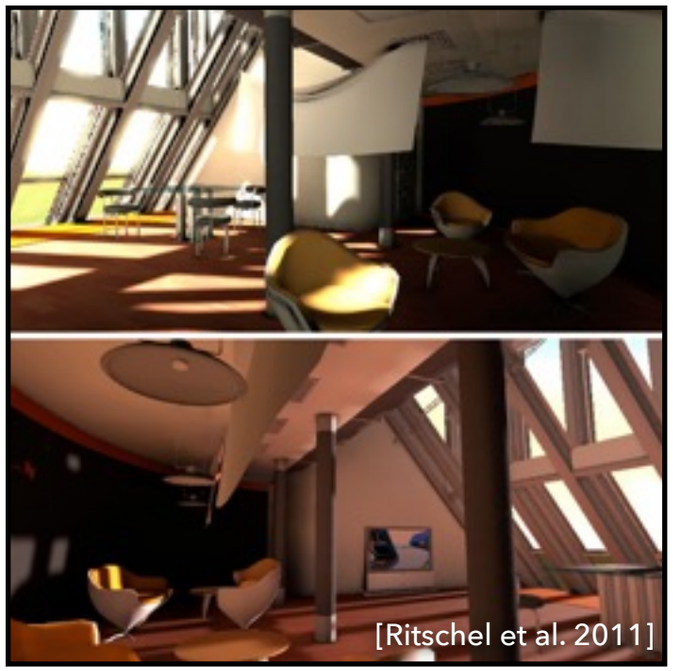
CS 87/187, Spring 2016

Rendering Algorithms

Many-Light Rendering



Prof. Wojciech Jarosz

wojciech.k.jarosz@dartmouth.edu

(most slides courtesy of Jan Novák)

Dartmouth

VCL

Administrativia

- Assignment 3 due tomorrow
- Assignment 4 will be posted tonight or tomorrow
  - due in 2 weeks, May 11
- Midterm (tentative) in class on Thursday May 12

Characteristics of Estimators

- *Unbiased* estimator
  - expected value equals the true value being estimated

$$E[F] = \int f(x)\,dx$$

  - variance (noise) is the only error
  - averaging infinitely many estimates (each with finite number of samples) also yields the correct answer

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \langle F^k \rangle = \int f(x) dx$$

Characteristics of Estimators

- *Bias* of an estimator
  - difference between the expected value of the estimator and the true value being estimated

$$\beta = E[F] - \int f(x)\,dx$$

  - expected average difference, expected error
  - averaging infinitely many estimates yields the correct answer plus the bias

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \langle F^k \rangle = \int f(x)dx + \beta$$

Characteristics of Estimators

- *Consistent* estimator
  - bias disappears in the limit

$$\lim_{N \to \infty} E[F] = \int f(x)\, dx$$

- ***Consistent*** estimators and ***unbiased*** estimators are asymptotically equivalent
  - both need an infinite number of samples to reduce the error to zero

Characteristics of Estimators

- *Mean Squared Error* (MSE) of an estimator
  - combines variance and squared bias

$$\mathrm{MSE}[F] = \mathrm{Var}[F] + \mathrm{Bias}[F]^2$$

- *Root Mean Squared Error* (RMSE)
  - has the same units as the quantity being estimated
  - for unbiased estimators equal to std. deviation

$$\mathrm{RMSE}[F] = \sqrt{\mathrm{MSE}[F]}$$

Rendering Techniques

- Examples of unbiased methods
  - Path tracing
  - Light tracing
  - Bidirectional path tracing
  - Many-light rendering (in theory)
- Examples of biased/consistent methods
  - Many-light rendering (practical implementations)
  - Photon mapping
  - Progressive photon mapping
  - Irradiance caching

Last time

Equations

- Rendering equation:

$$L_o(\mathbf{x}, \vec{\omega}_o) = L_e(\mathbf{x}, \vec{\omega}_o) + \int_{H^2} f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) L_i(\mathbf{x}, \vec{\omega}_i) \cos\theta_i \, d\vec{\omega}_i$$

- Measurement equation:

$$I_j = \int_{A_{\text{film}}} \int_{H^2} W_e(\mathbf{x}, \vec{\omega}) L_i(\mathbf{x}, \vec{\omega}) \cos\theta \, d\vec{\omega} d\mathbf{x}$$

Duality of Radiance & Importance

emitted *importance*

incident *radiance*

$$I_j = \int_{A_{\text{film}}} \int_{H^2} W_e(\mathbf{x}, \vec{\omega}) L_i(\mathbf{x}, \vec{\omega}) \cos\theta \, d\vec{\omega} d\mathbf{x}$$

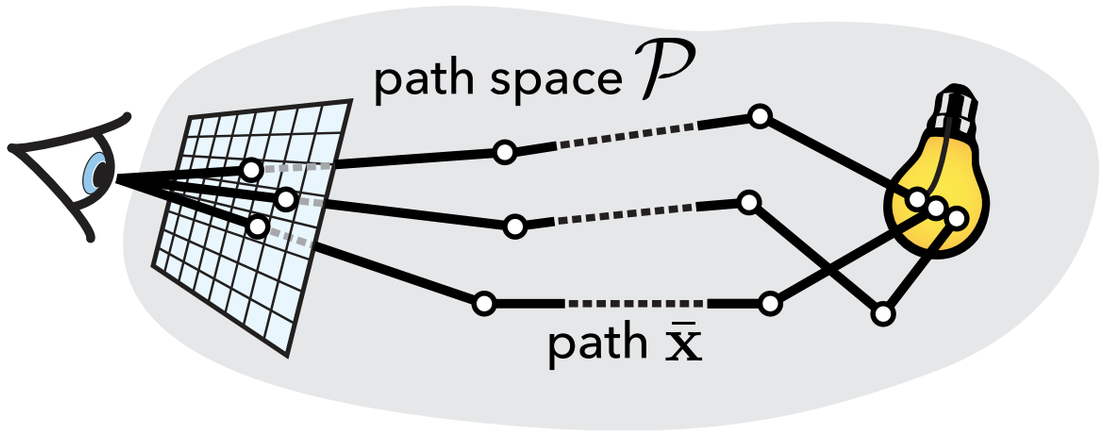
$$= \int_{A_{\text{light}}} \int_{H^2} W_i(\mathbf{z}, \vec{\omega}) L_e(\mathbf{z}, \vec{\omega}) \cos\theta \, d\vec{\omega} d\mathbf{z}$$

emitted *radiance*

incident *importance*

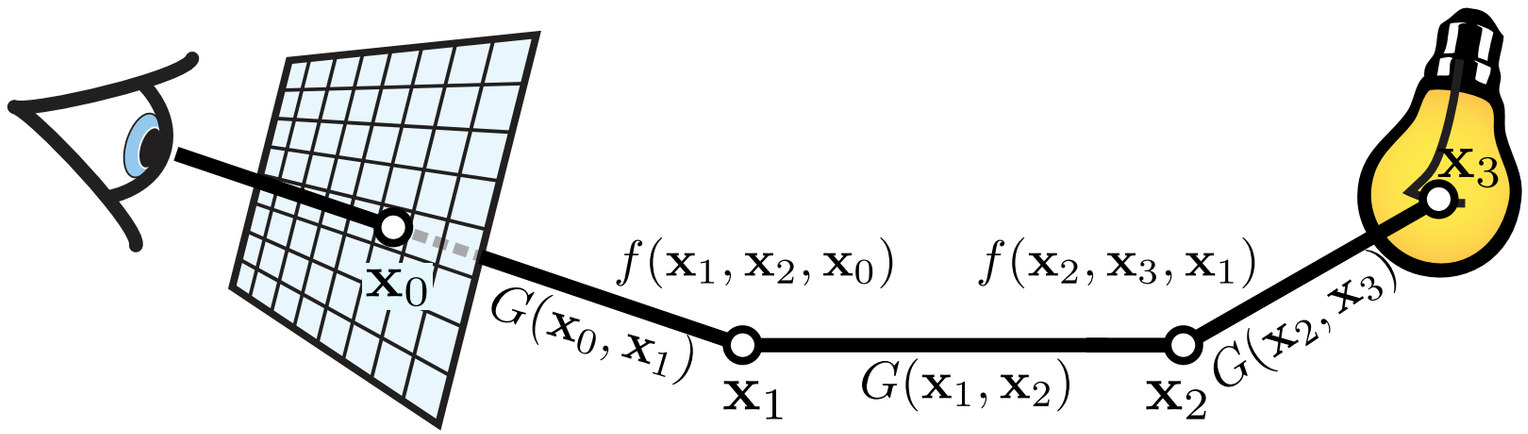
Path Integral Form of Measurement Eq.

$$I_j = \int_{\mathcal{P}} W_e(\mathbf{x}_0, \mathbf{x}_1) L_e(\mathbf{x}_k, \mathbf{x}_{k-1}) T(\bar{\mathbf{x}})\, d\bar{\mathbf{x}}$$

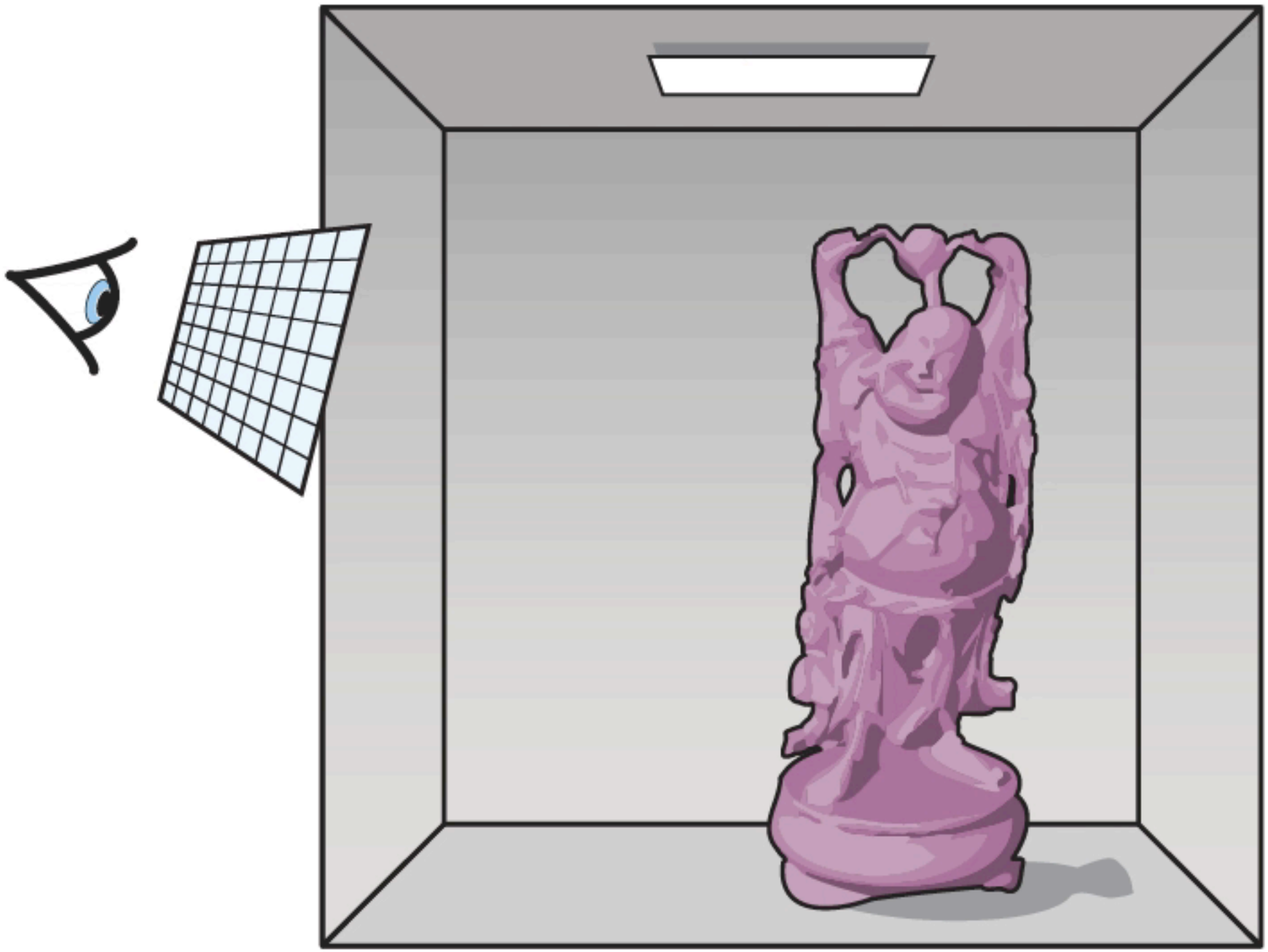


path throughput

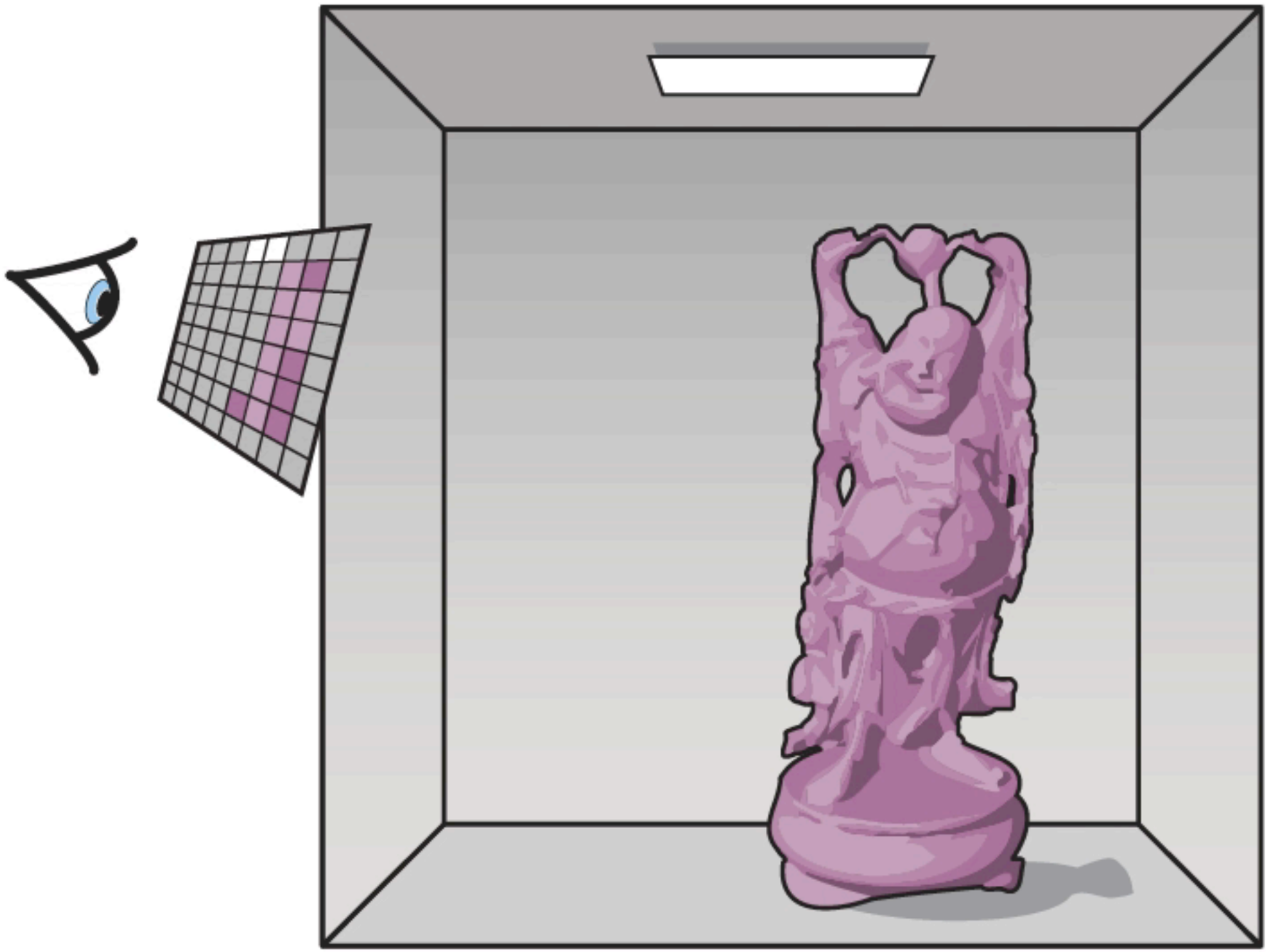
$$T(\bar{\mathbf{x}}) = G(\mathbf{x}_0, \mathbf{x}_1) \prod_{j=1}^{k-1} f(\mathbf{x}_j, \mathbf{x}_{j+1}, \mathbf{x}_{j-1}) G(\mathbf{x}_j, \mathbf{x}_{j+1})$$



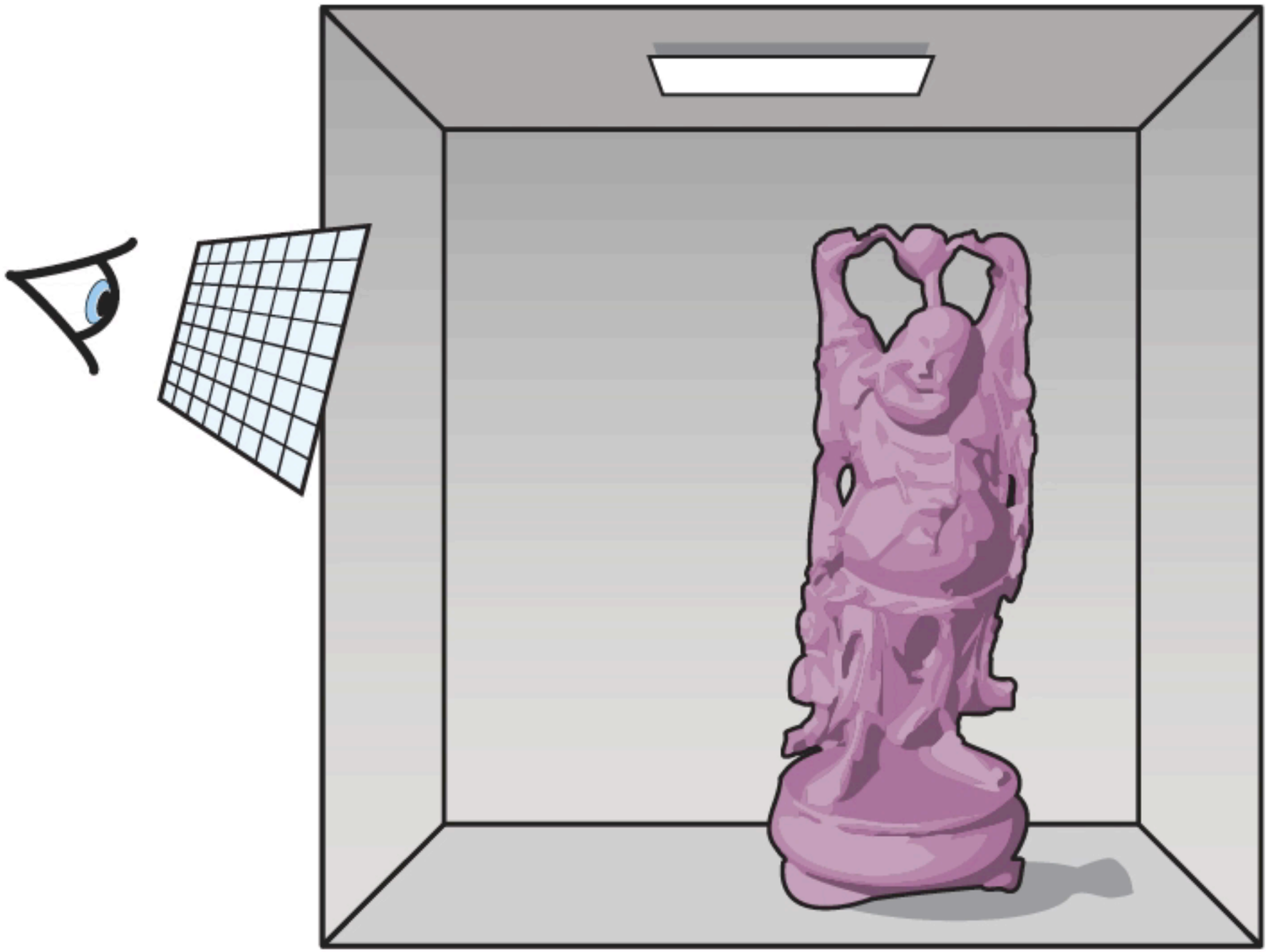
Light Tracing



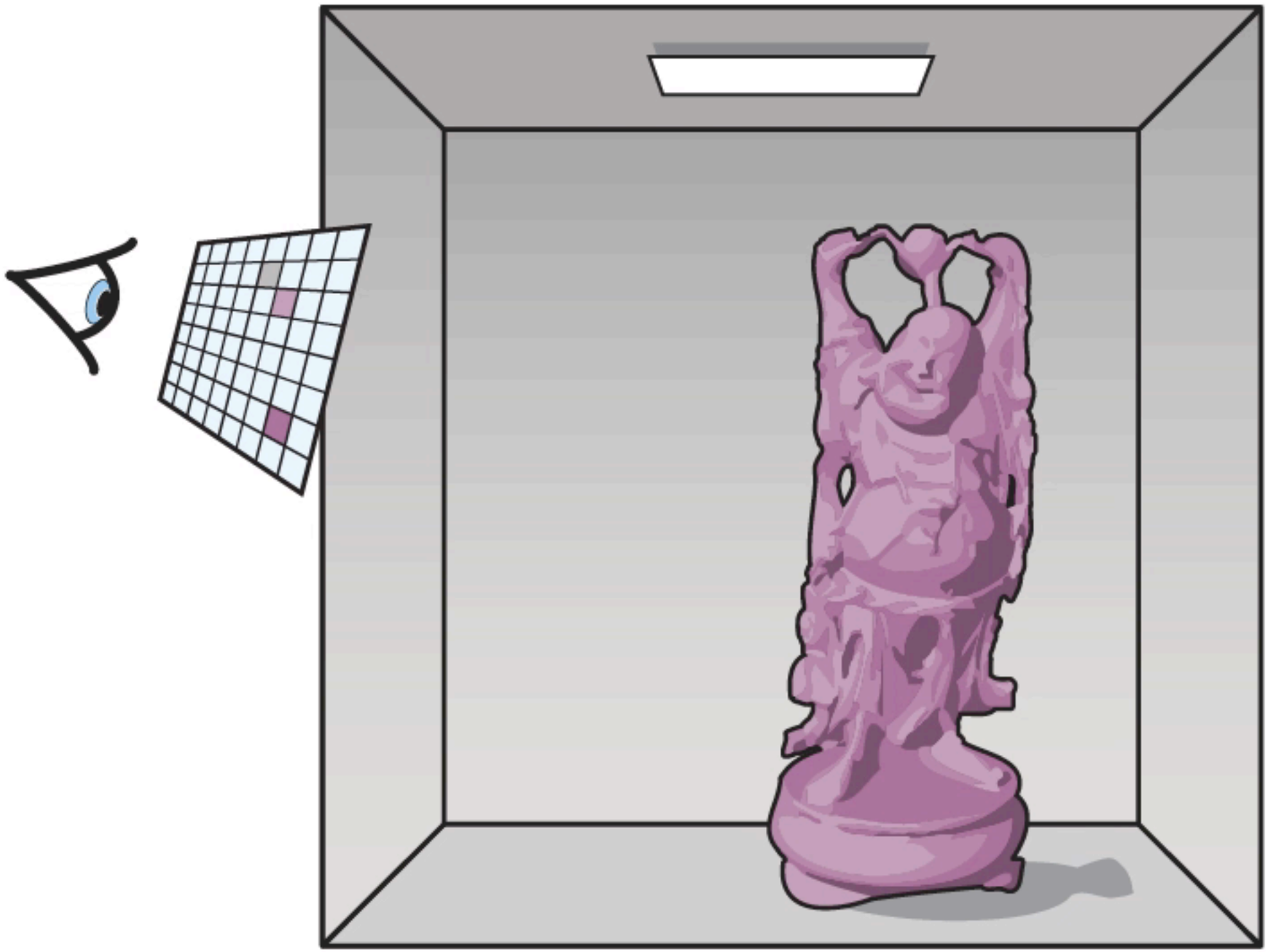
Light Tracing



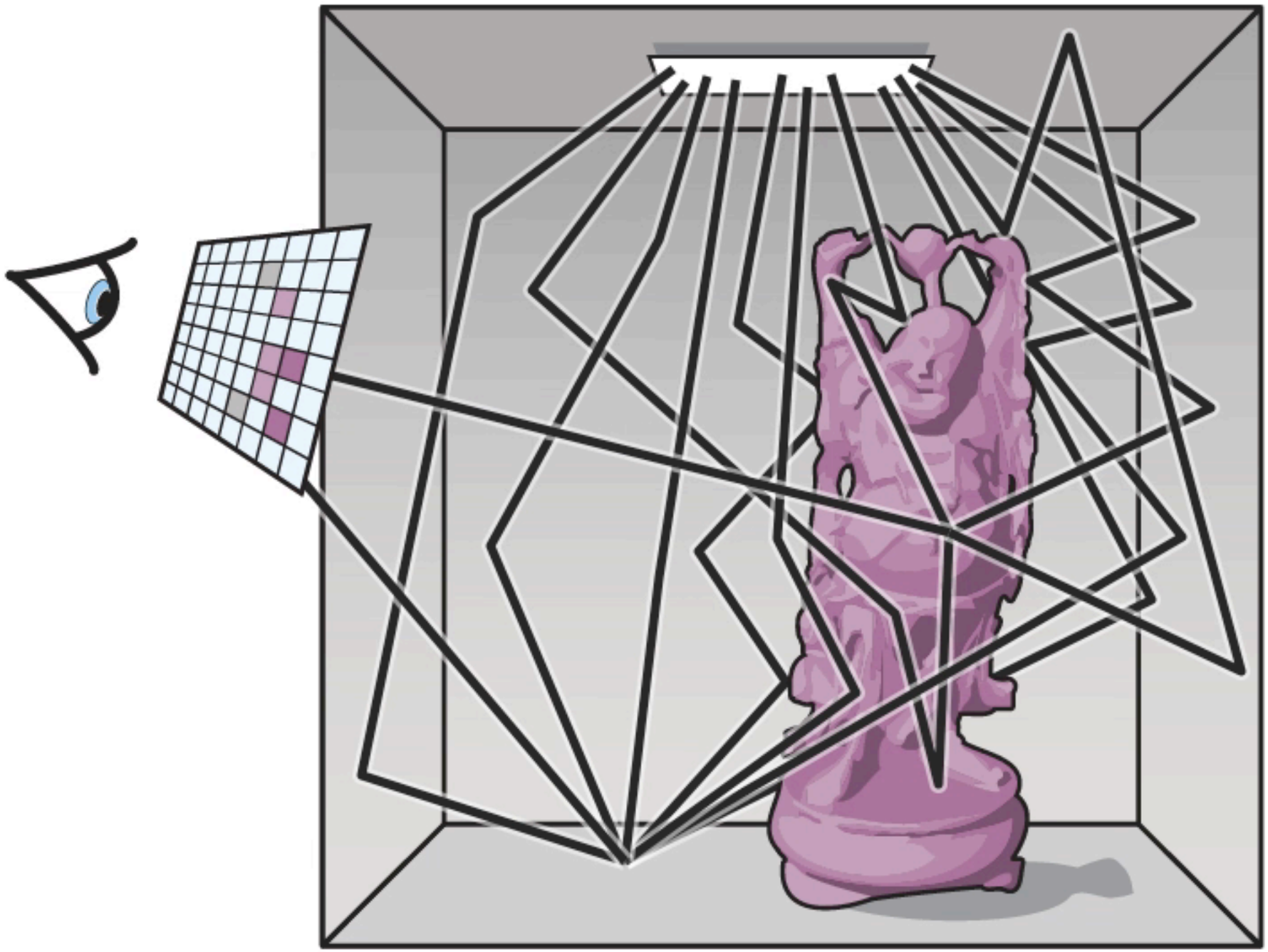
Path Tracing



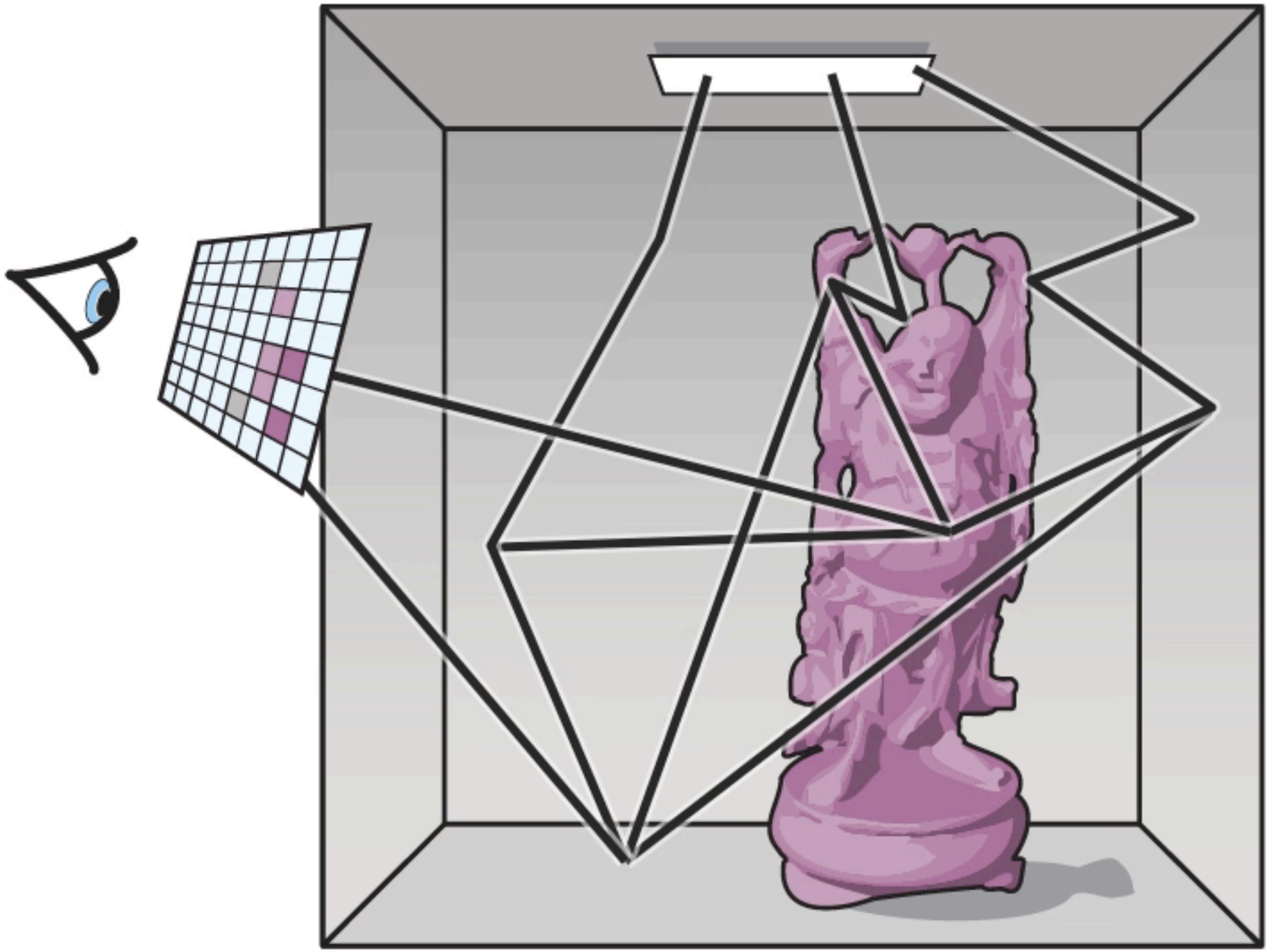
Bidirectional Path Tracing



Many-Light Rendering



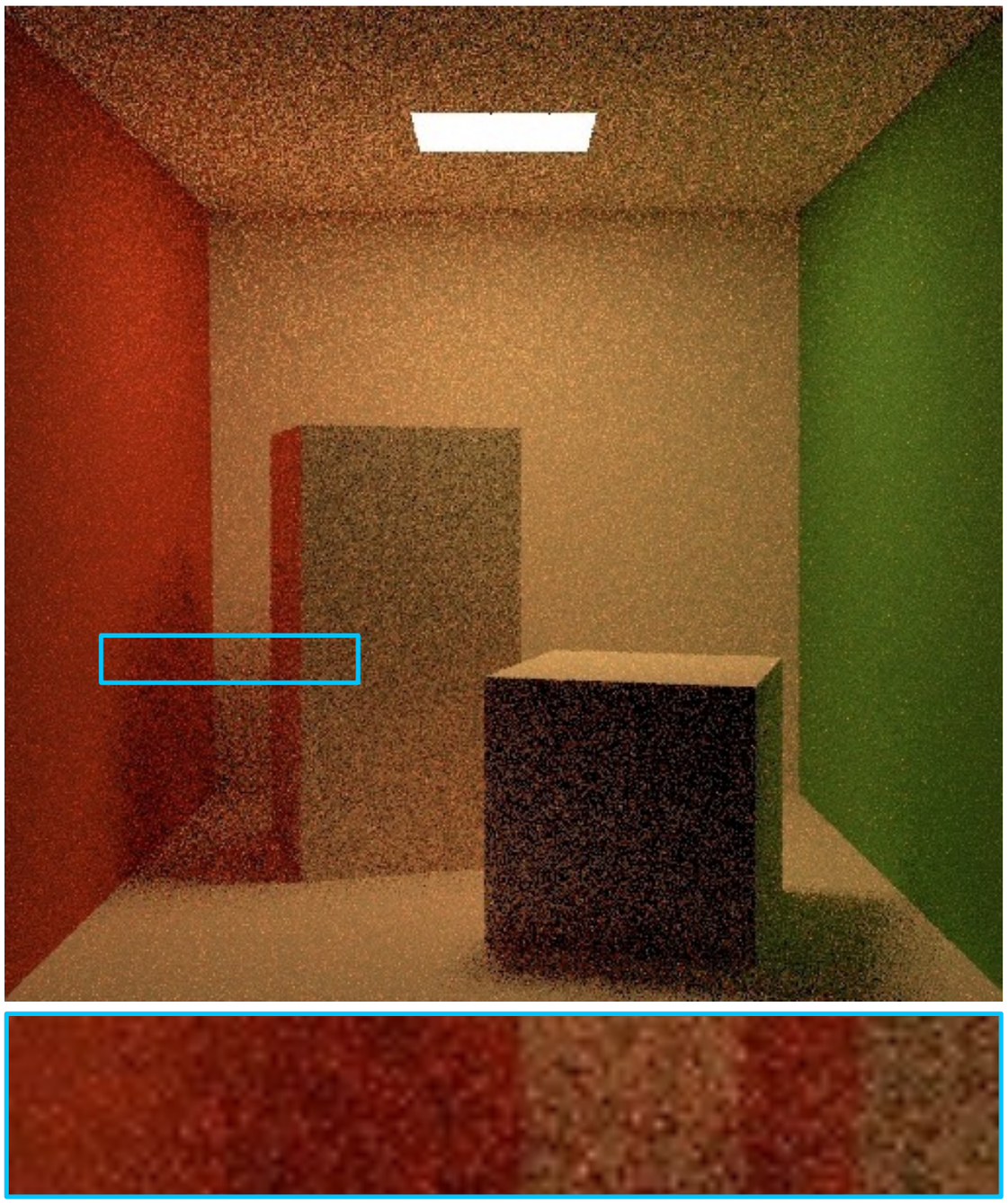
Many-Light Rendering



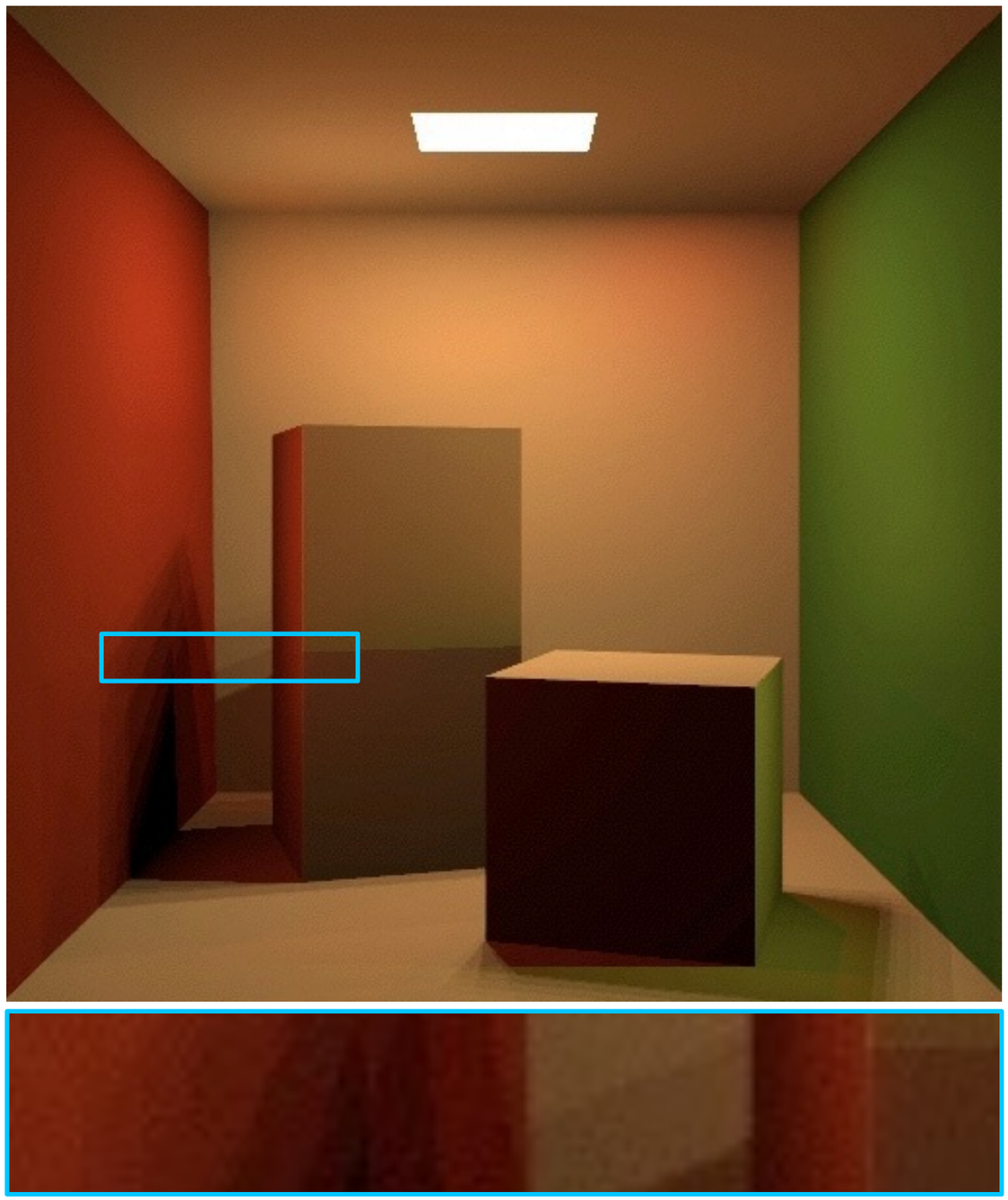
Many-Light (VPL) Rendering

- Main idea: convert the *global* illumination problem into a *direct* illumination problem

Path Tracing vs. Many-Lights



4 paths/pixel
(*random* error shows up as noise)



16 VPLs
(*correlated* error shows up as structured artifacts)

Today's Menu

- Many-Light Rendering
- Generation of Virtual Point Lights (VPLs)
- Lighting with VPLs
- Scalability

Many-Light Rendering

- Original idea presented as *Instant Radiosity* [Keller 1997]
- Now more than 30 extensions

Speed **Scalability** **Quality**

Real-time graphics

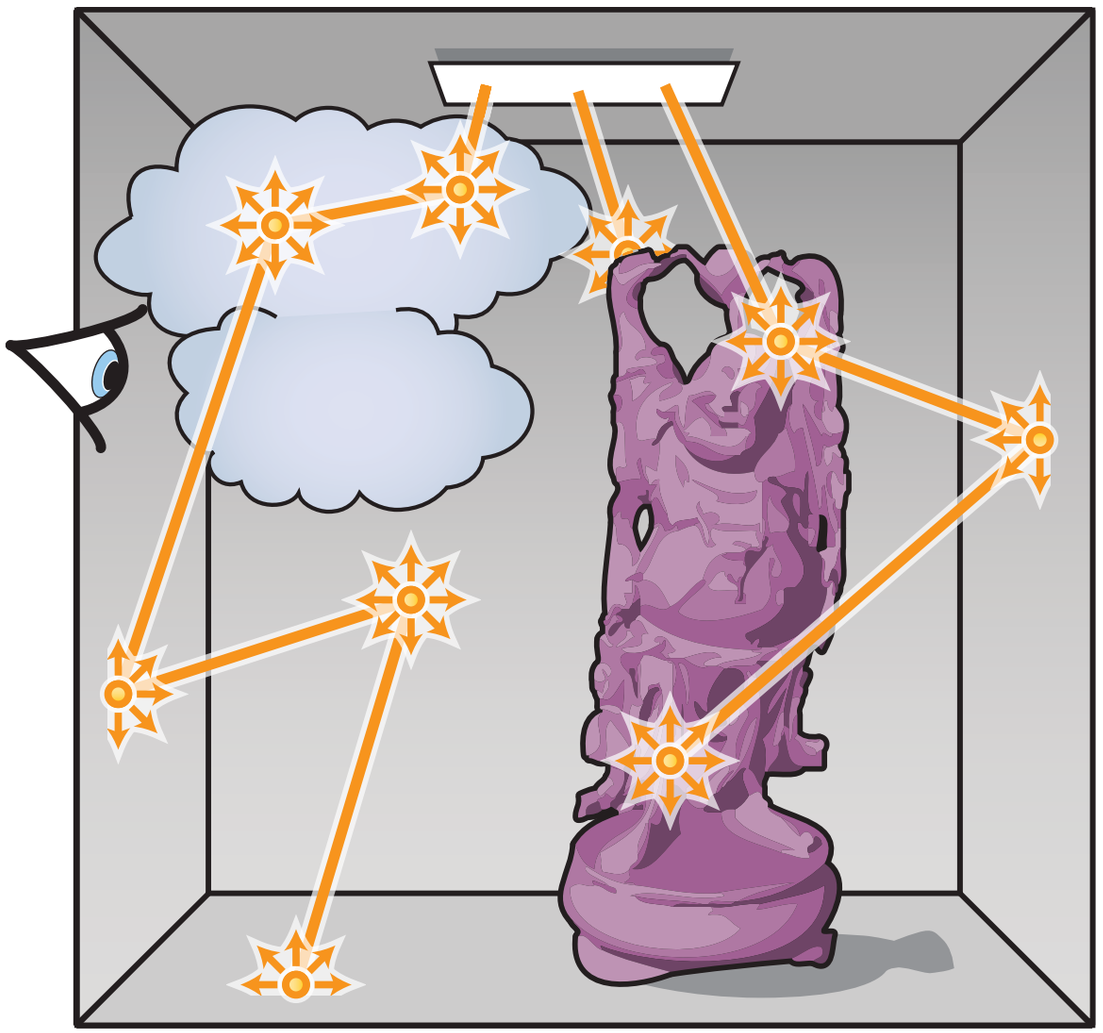
Interactive visualization

Movie production

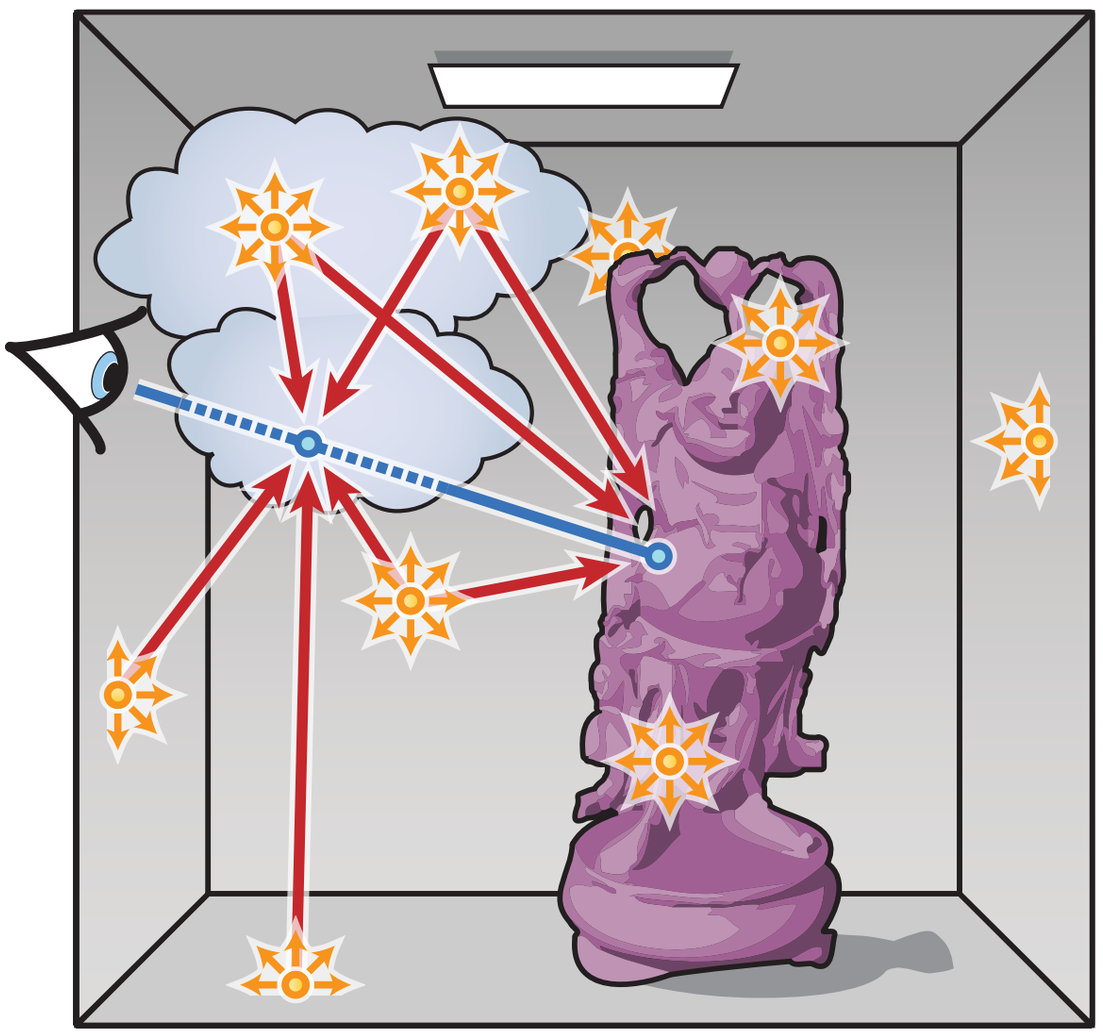
Product design

Many-Light Rendering

Generation of VPLs

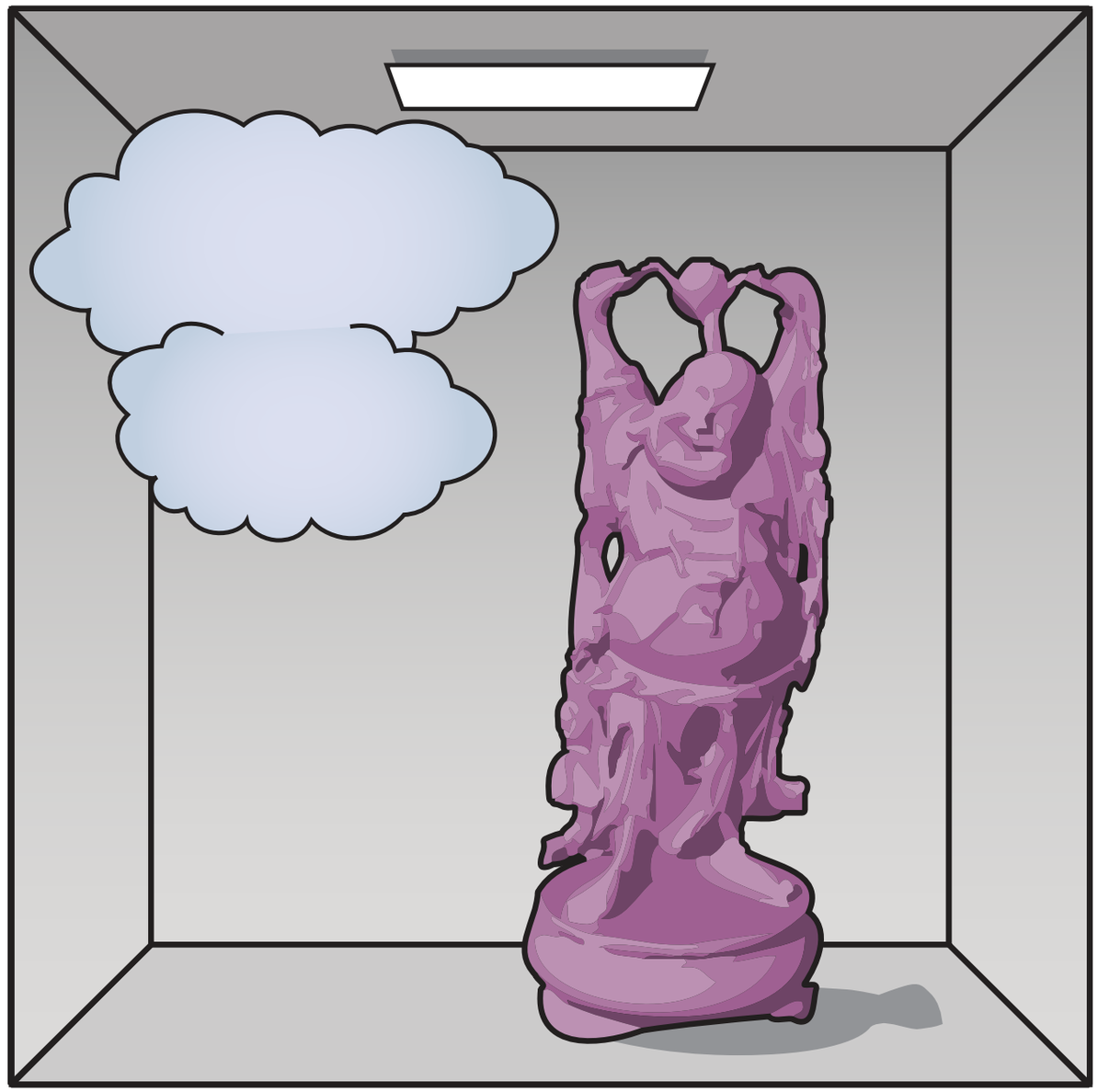


Lighting with VPLs

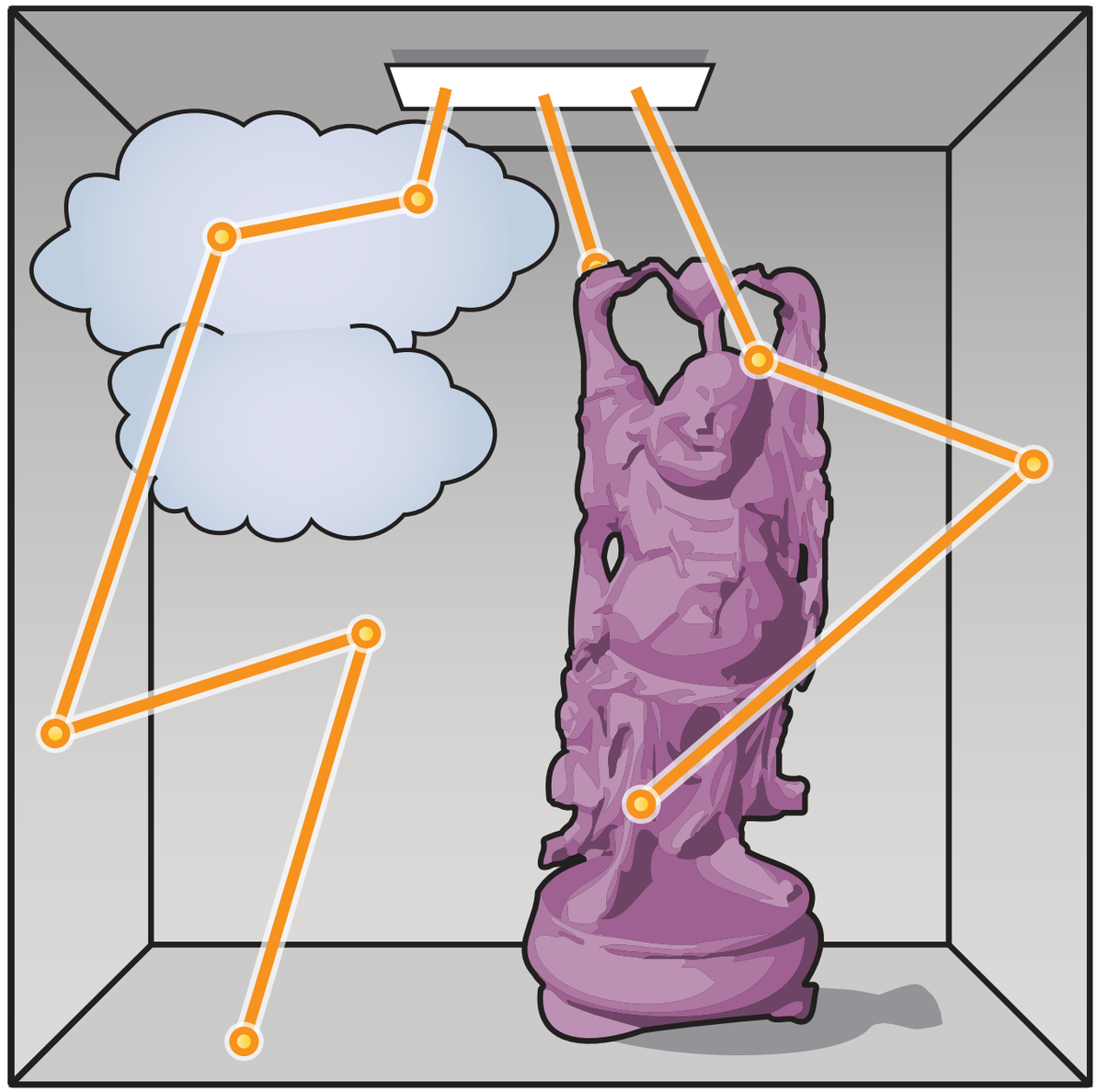


VPL = Virtual Point Light

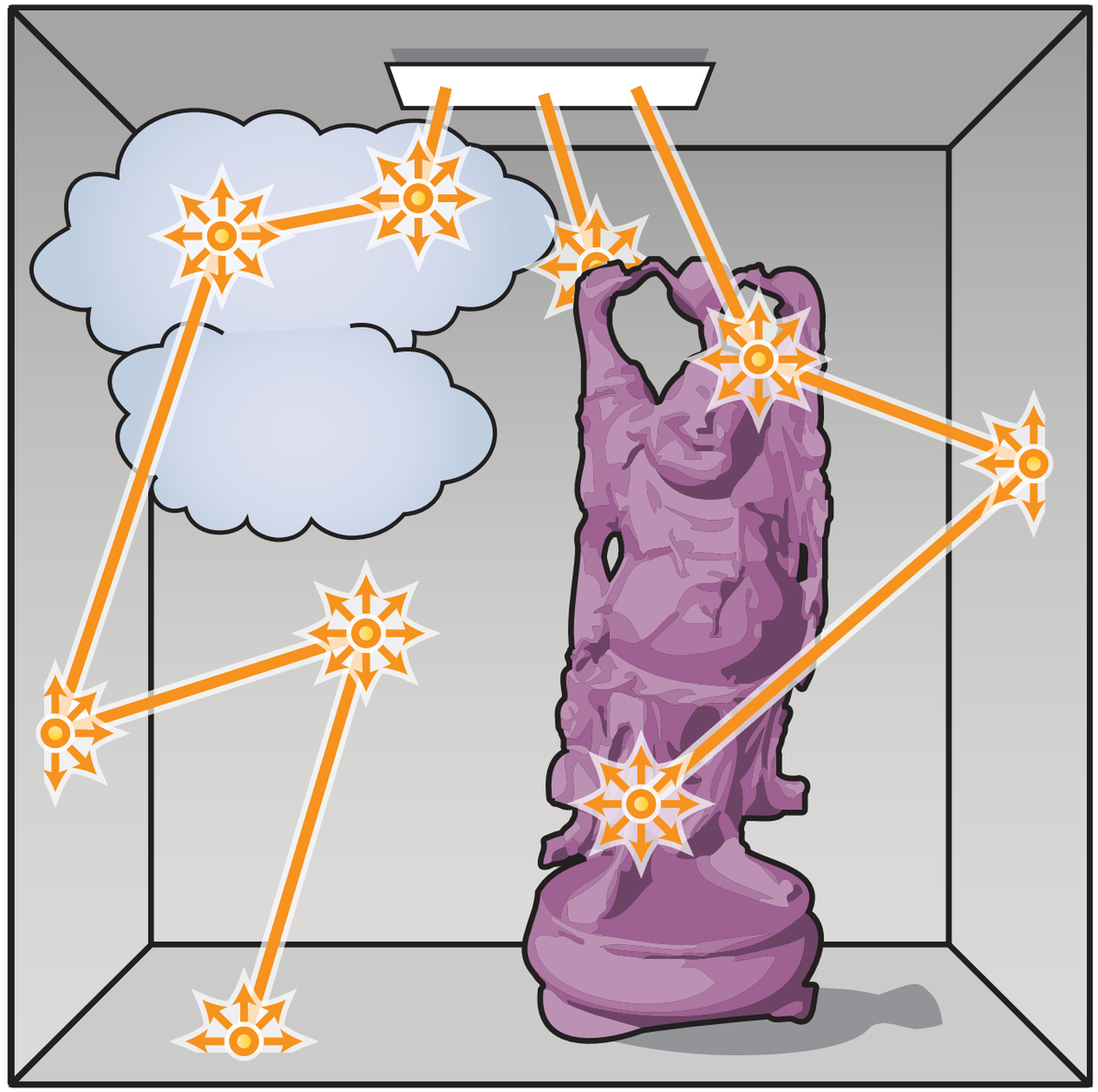
Basic VPL Generation



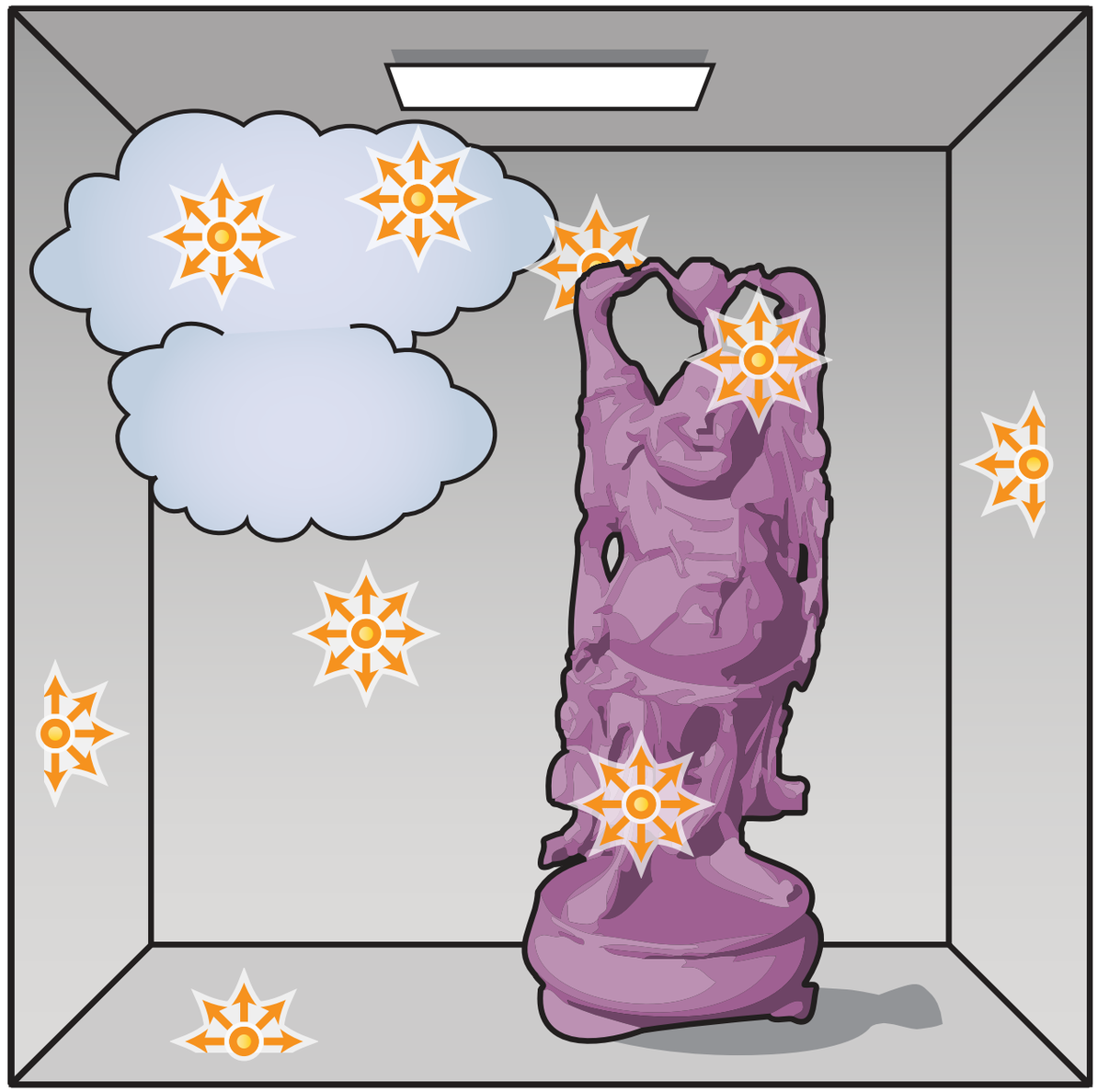
Basic VPL Generation



Basic VPL Generation



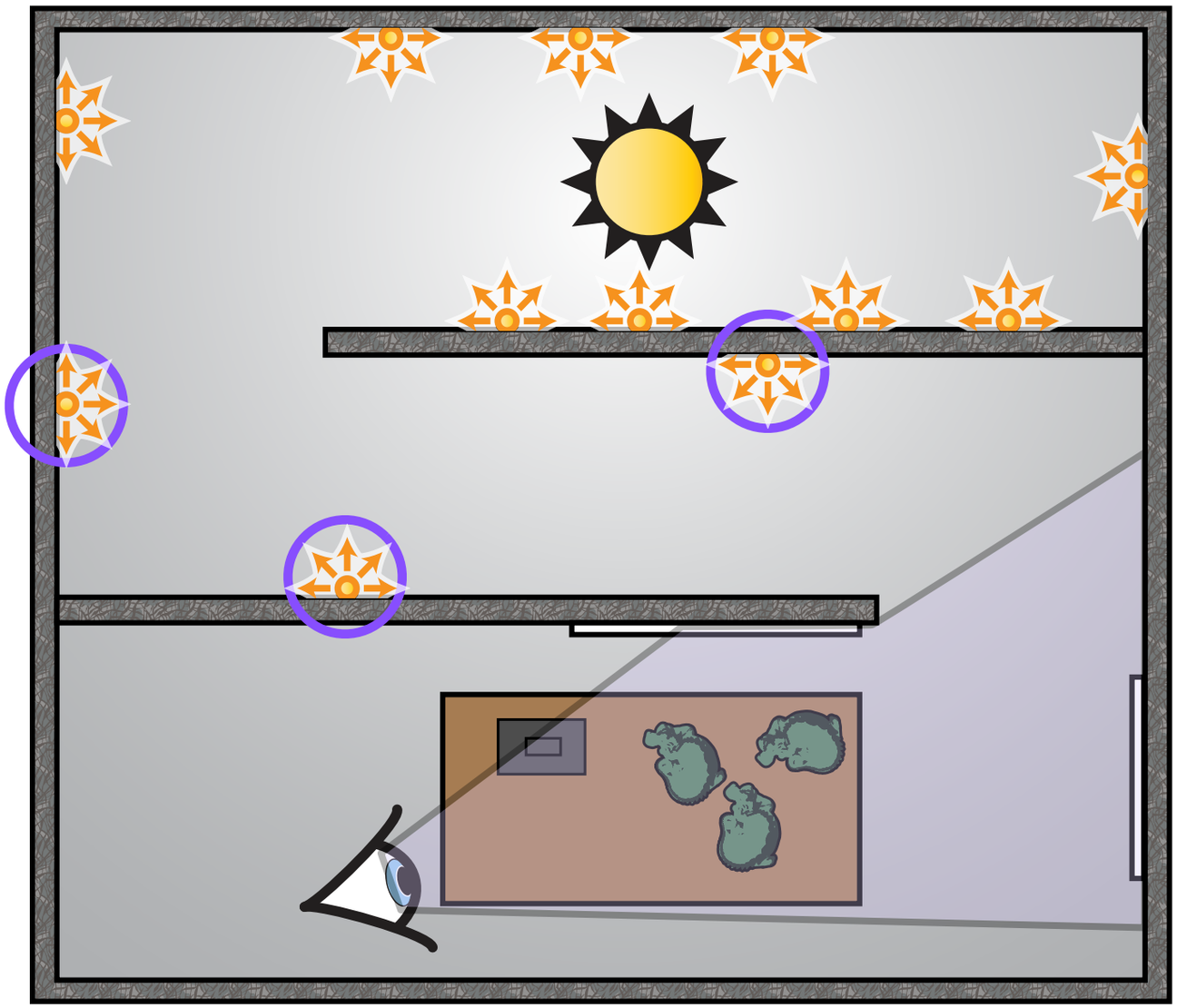
Basic VPL Generation



Basic Generation of VPLs

- In complex scenes, many VPLs do not contribute

Scene



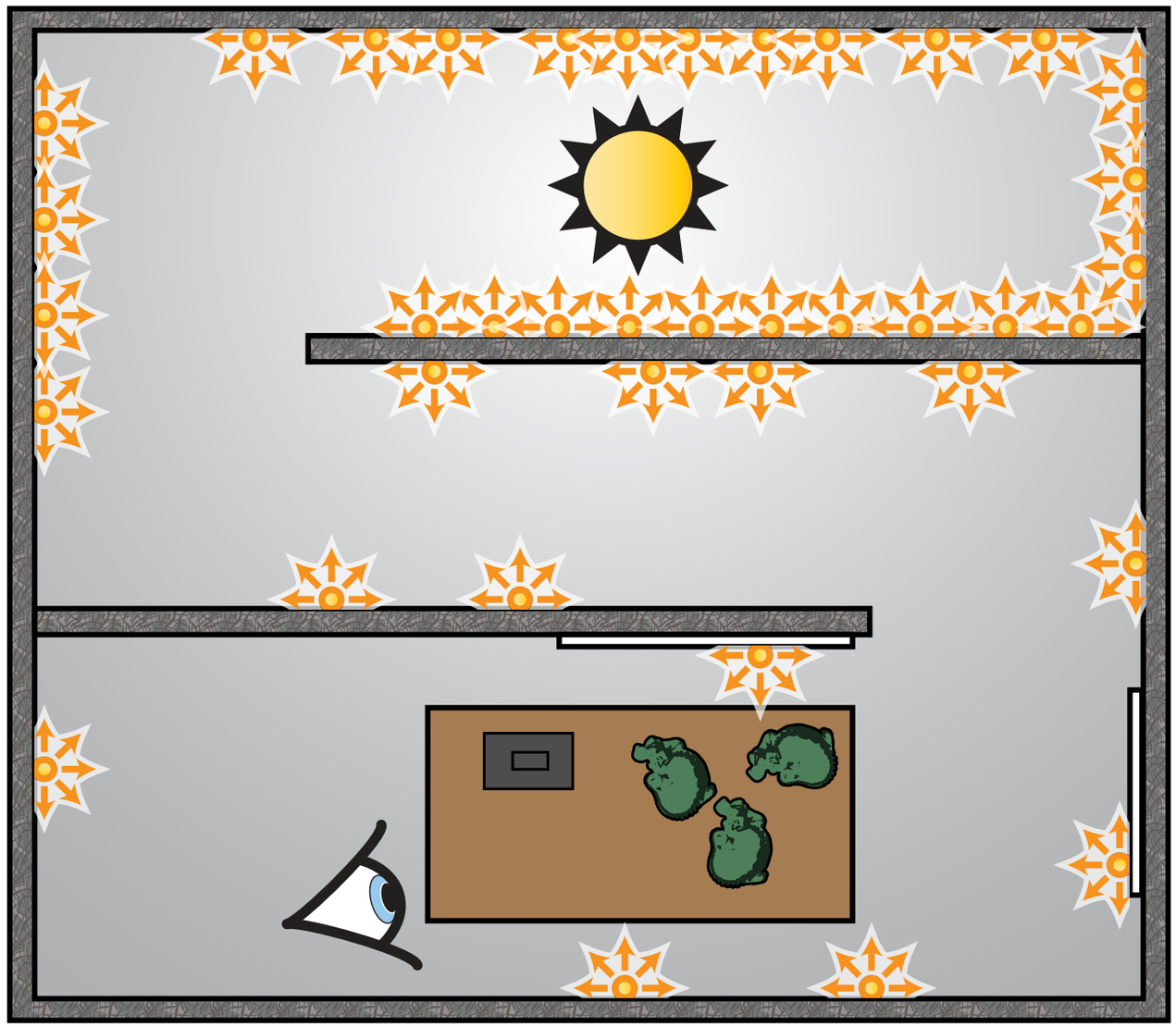
VPL rendering

Reference

Images courtesy of Segovia et al.

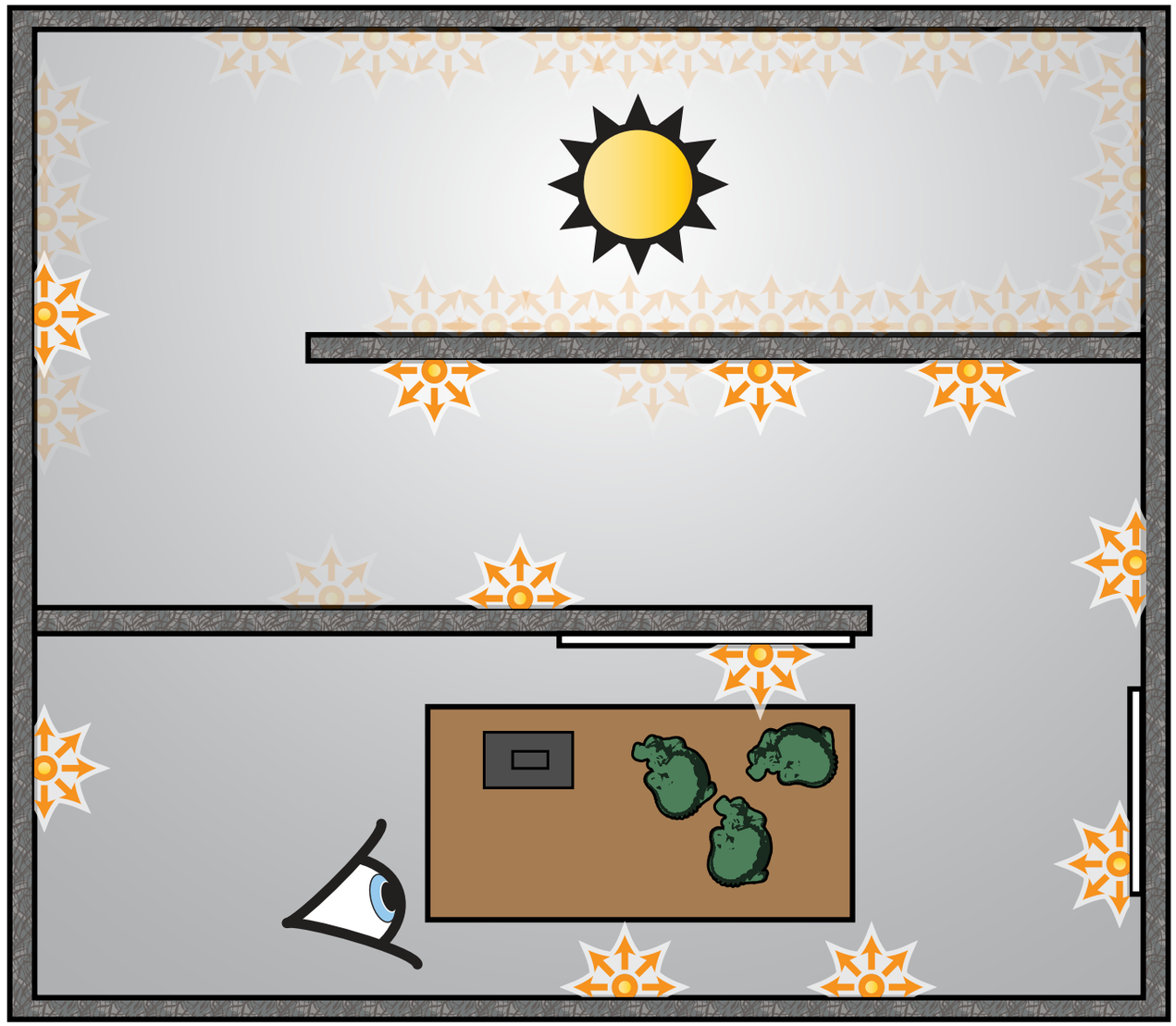
Improved Generation of VPLs

- Rejection of unimportant VPLs [Georgiev and Slusallek 2010]
  - generate many, reject those with low expected contribution



Improved Generation of VPLs

- Rejection of unimportant VPLs [Georgiev and Slusallek 2010]
  - generate many, reject those with low expected contribution



Improved Generation of VPLs

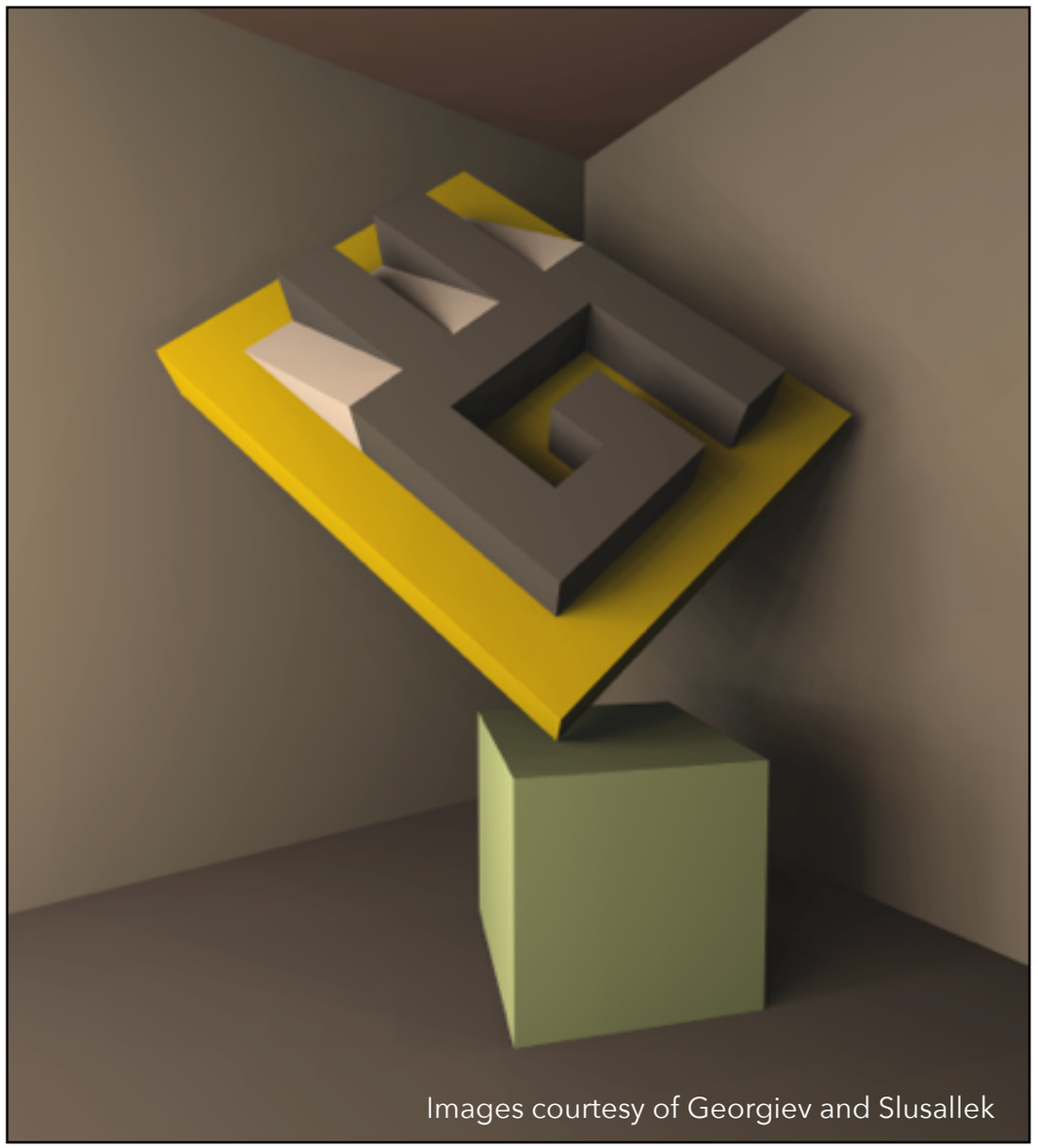
- Rejection of unimportant VPLs [Georgiev and Slusallek 2010]
  - generate many, reject those with low expected contribution
- Step 1: estimate average contribution of a VPL
  - a few pilot VPLs illuminate a few surface points seen by the camera
- Step 2: generate VPLs
  - for each (candidate) VPL:
    - estimate the contribution (to a few shading points)
    - use Russian roulette to either accept or reject

Improved Generation of VPLs

- Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Without rejection

With rejection (7% acceptance)



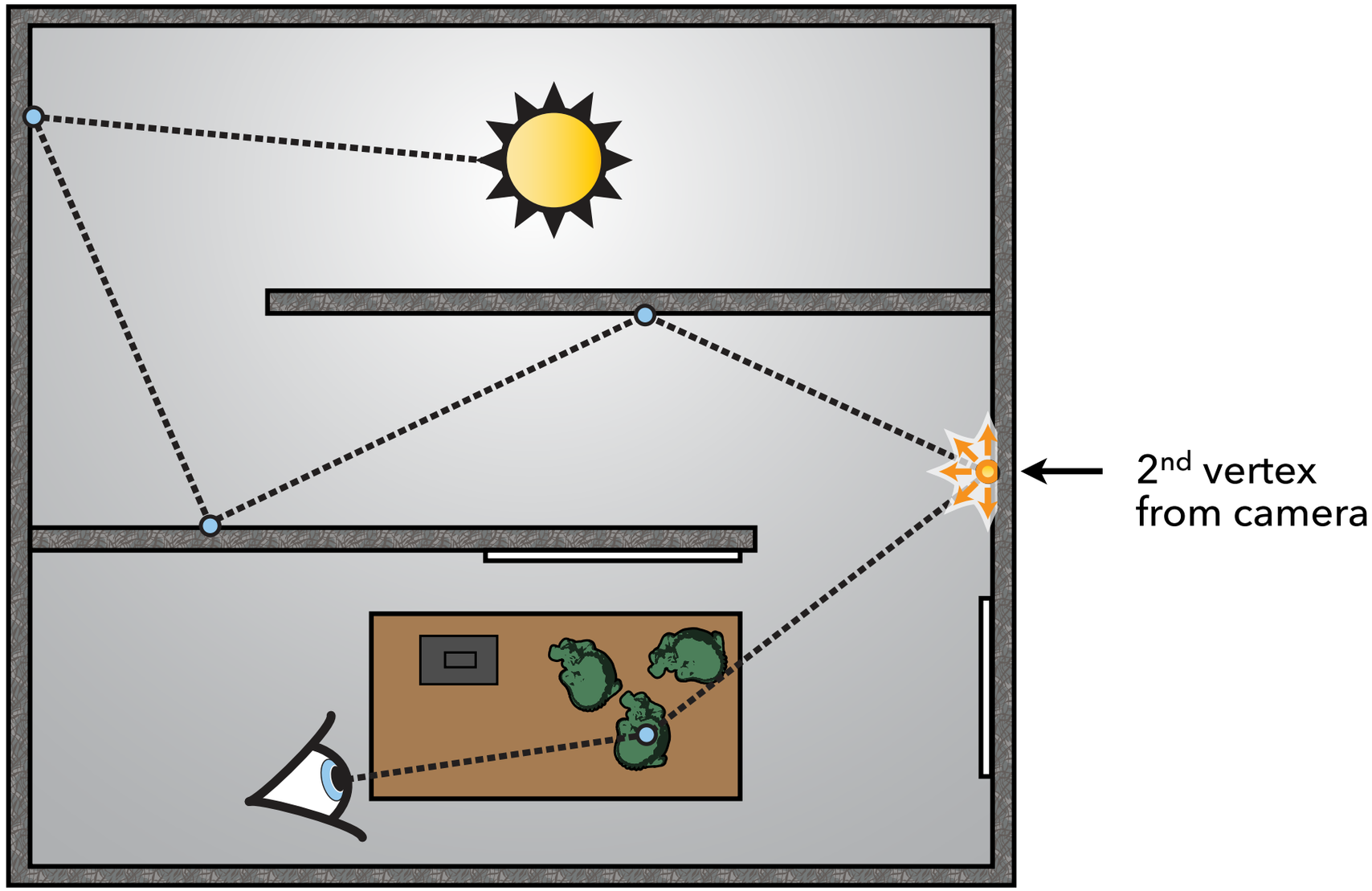
Images courtesy of Georgiev and Slusallek

Improved Generation of VPLs

- Rejection of unimportant VPLs [Georgiev and Slusallek 2010]
  - easy to implement
  - all VPLs contribute roughly equally
    (thanks to the Russian roulette)
  - can become costly!

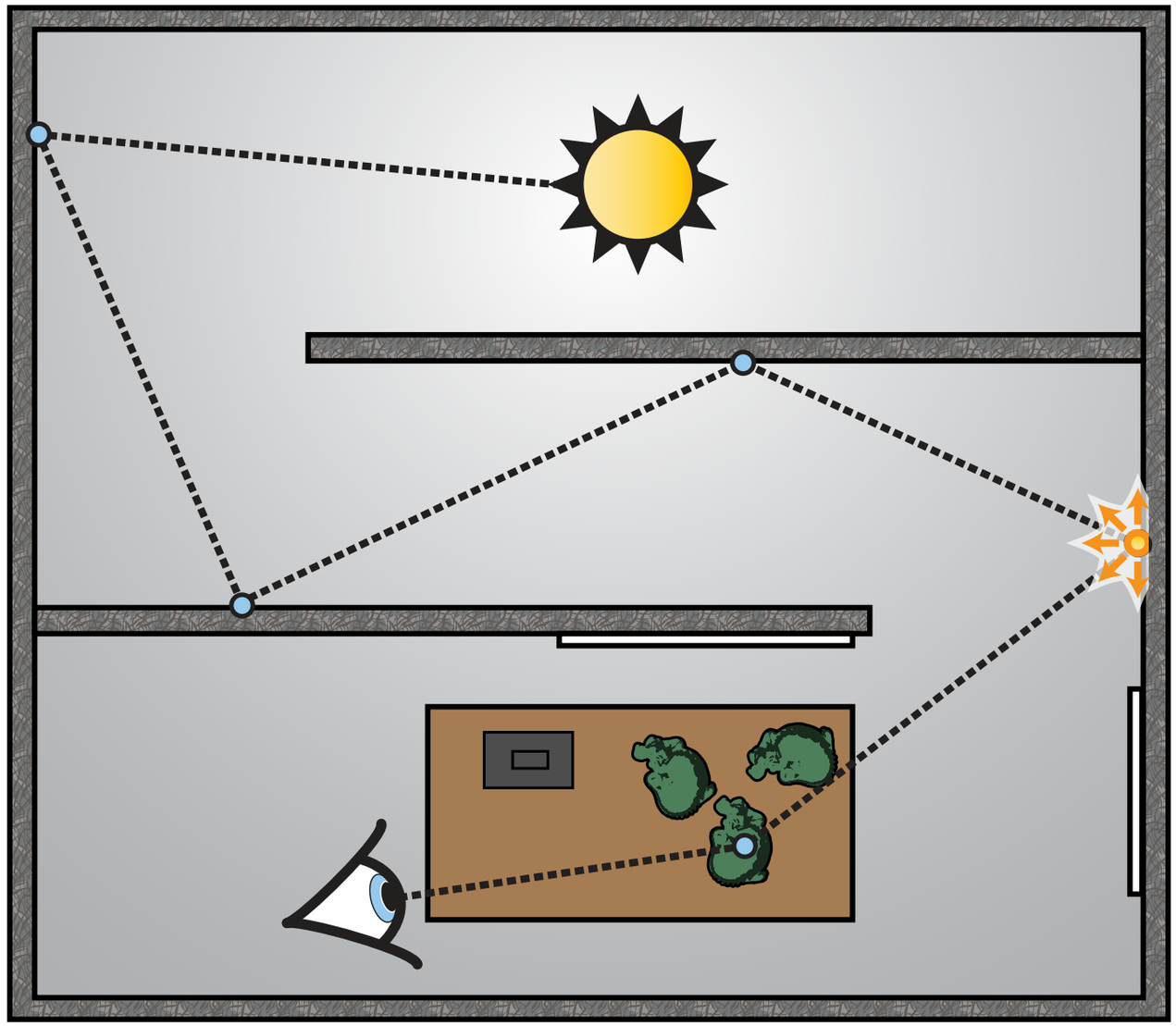
Improved Generation of VPLs

- Bidirectional generation of VPLs [Segovia et al. 2006]
  - construct a path (e.g. using bidirectional path tracing)
  - create a VPL at the 2nd bounce from the camera



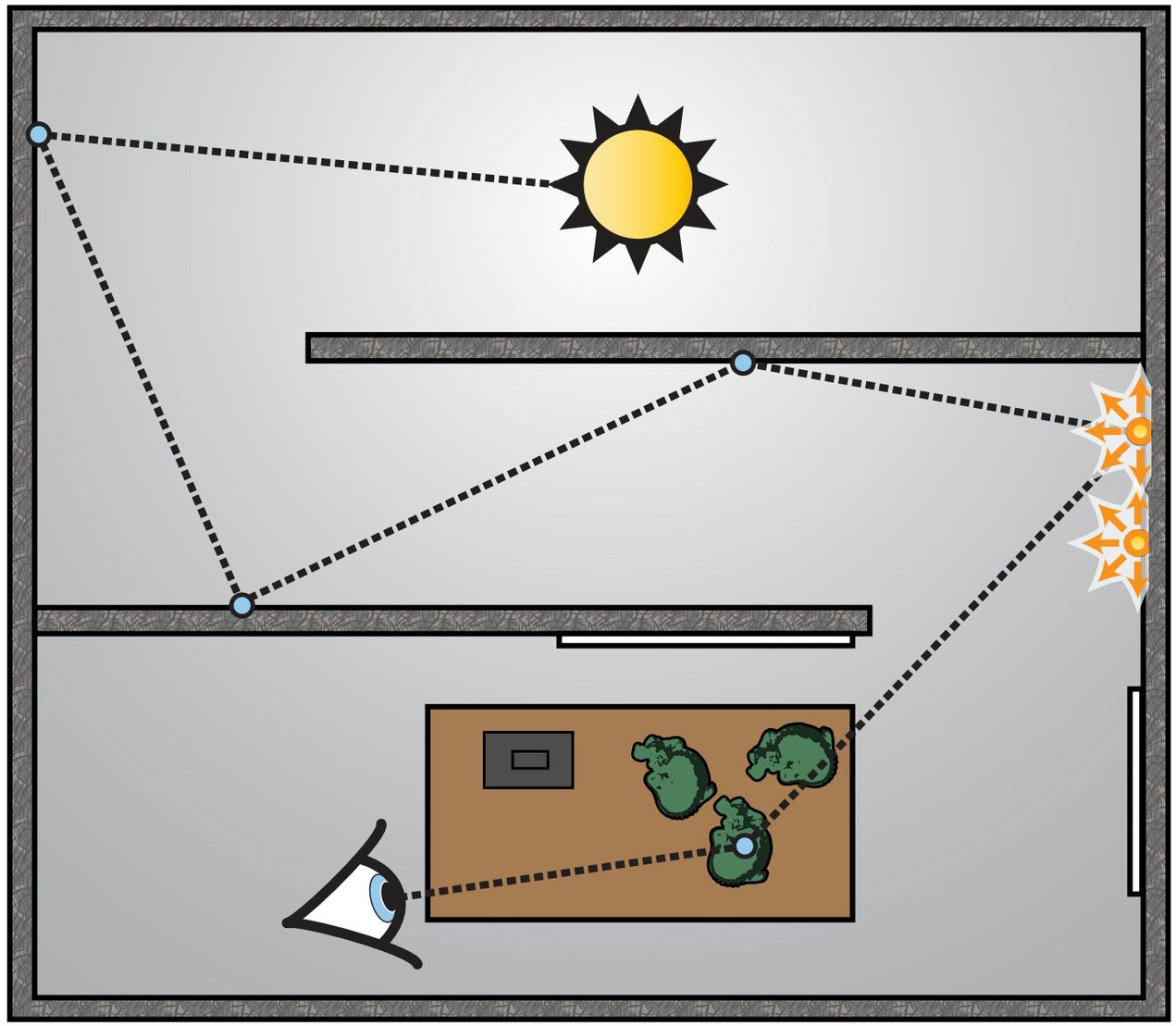
Improved Generation of VPLs

- Metropolis sampling of VPLs [Segovia et al. 2007]
  - generate additional VPLs by mutating paths



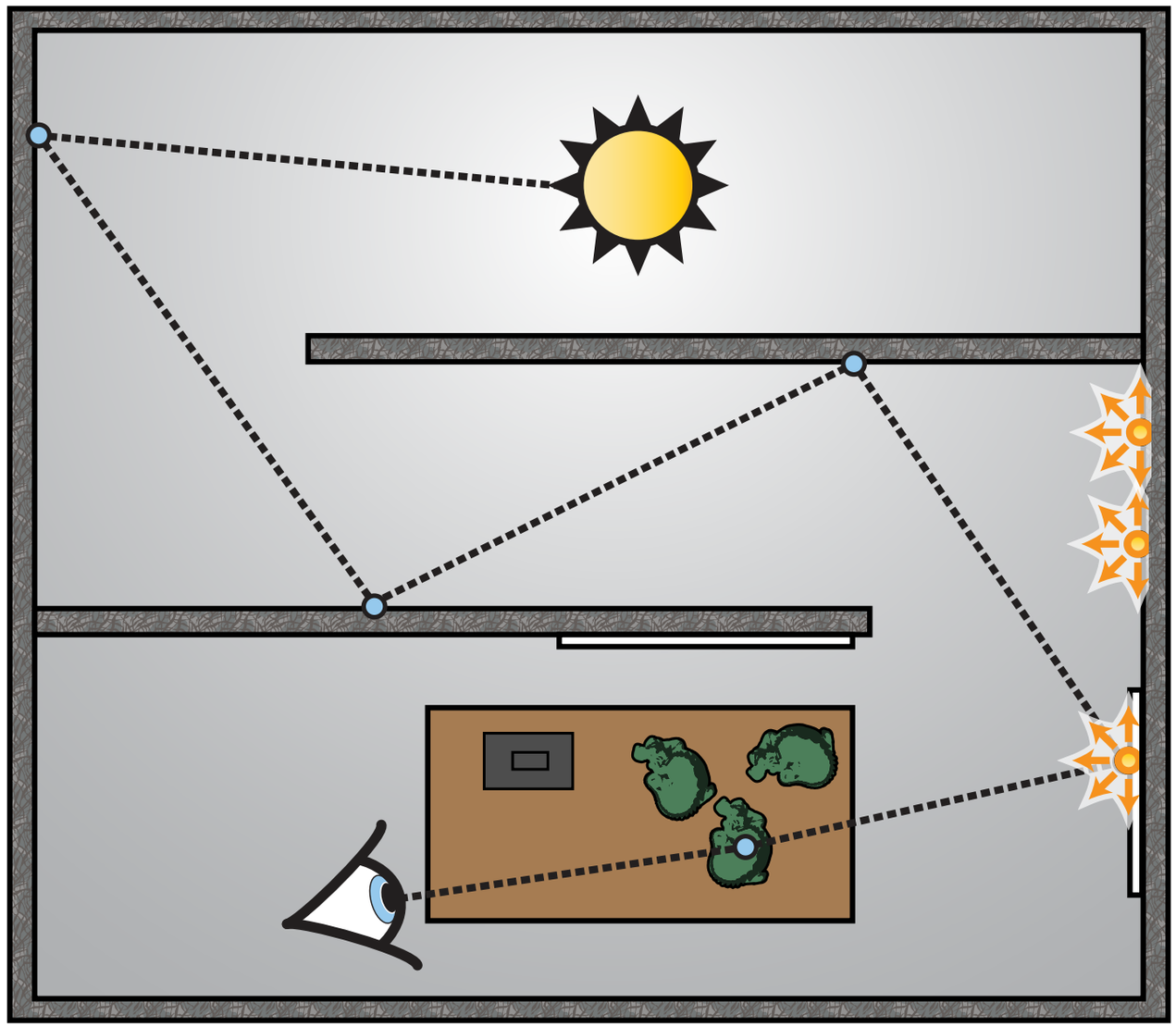
Improved Generation of VPLs

- Metropolis sampling of VPLs [Segovia et al. 2007]
  - generate additional VPLs by mutating paths



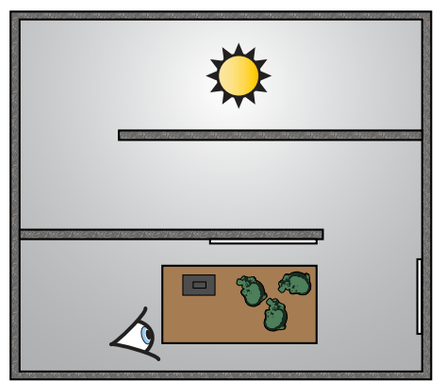
Improved Generation of VPLs

- Metropolis sampling of VPLs [Segovia et al. 2007]
  - generate additional VPLs by mutating paths



Improved Generation of VPLs

- Metropolis sampling of VPLs [Segovia et al. 2007]
  - generate additional VPLs by mutating paths

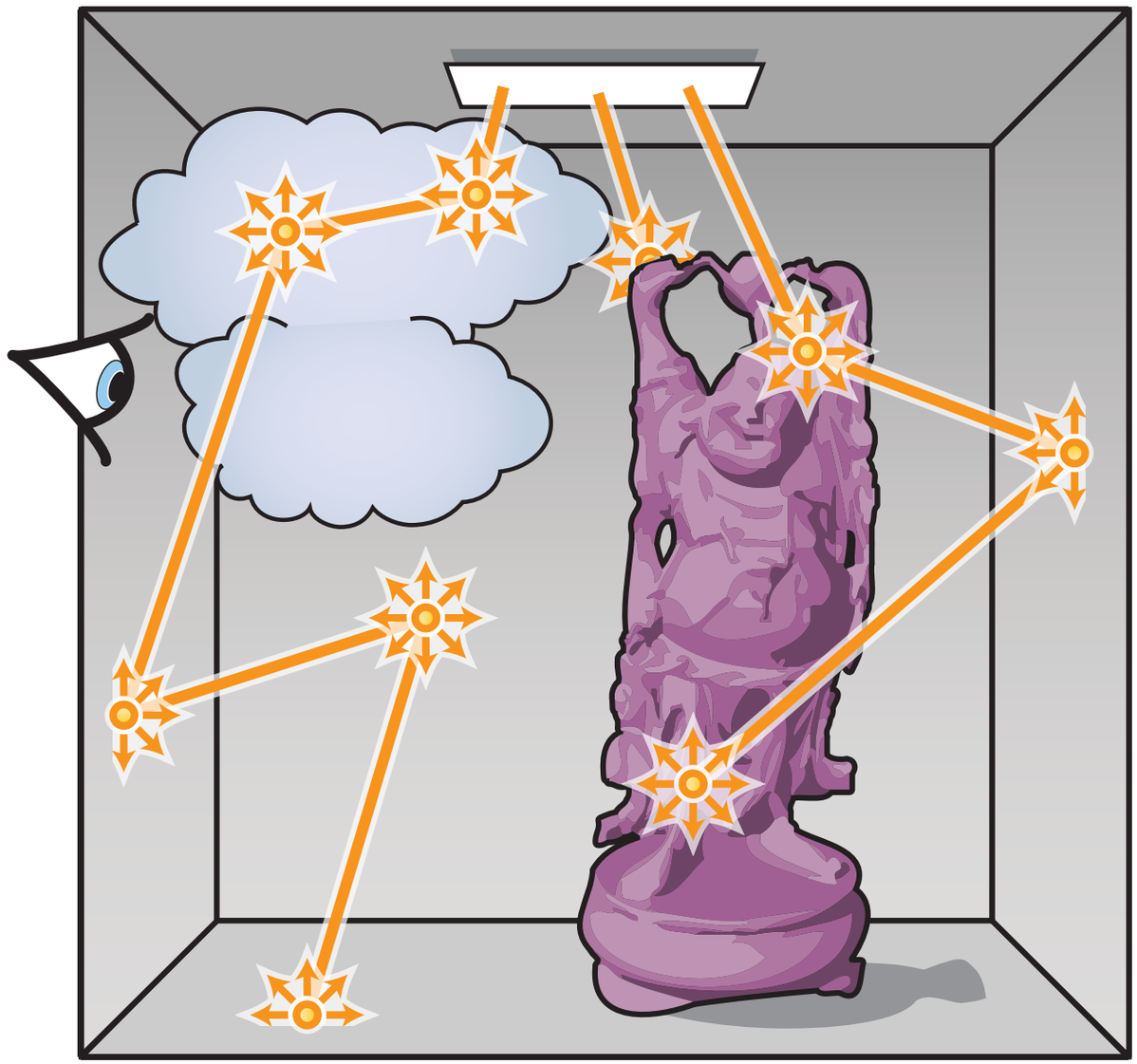


Instant Radiosity (IR) | Bidirectional IR | Metropolis IR

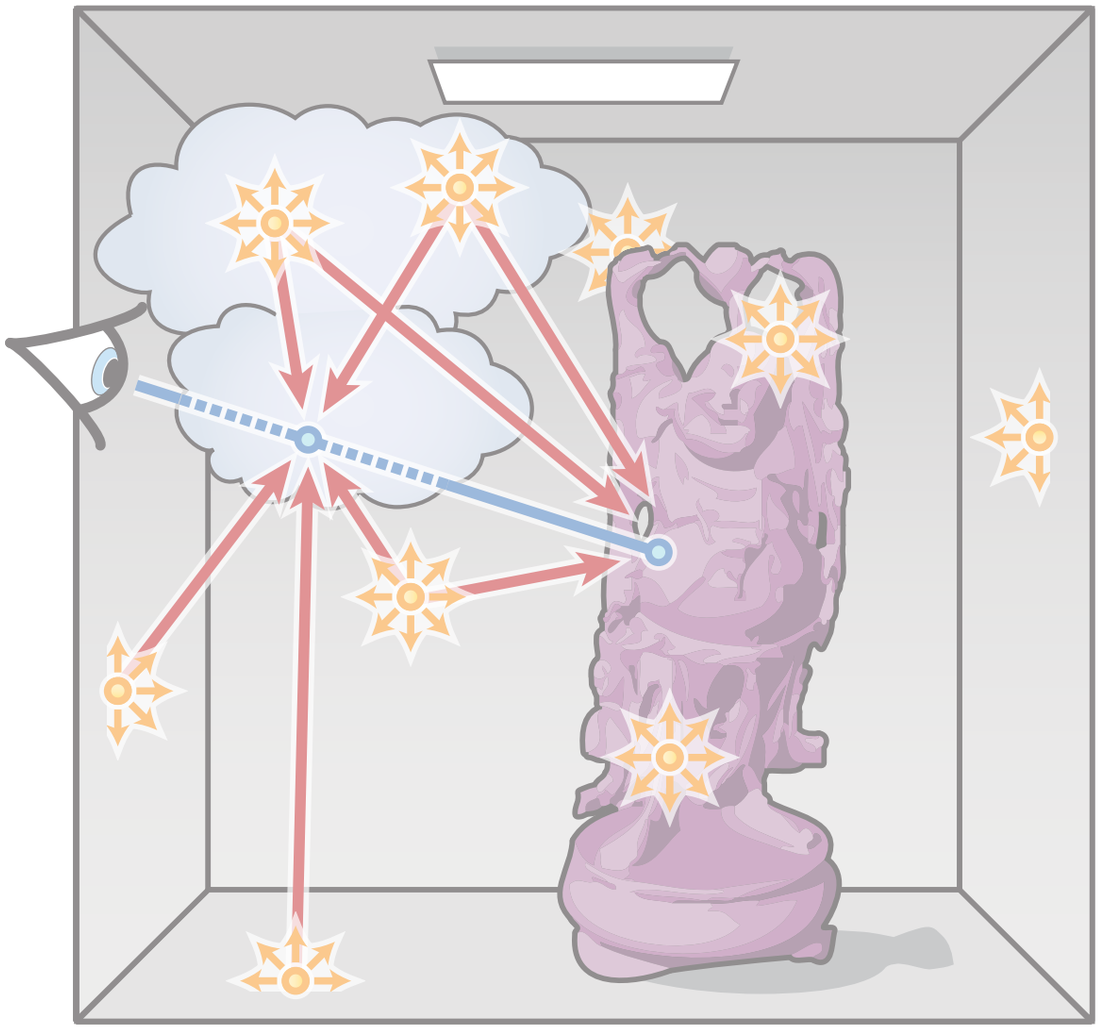
All approaches would ultimately yield the same image, but Metropolis gives good results already with a few VPLs.

Many-Light Rendering

Generation of VPLs

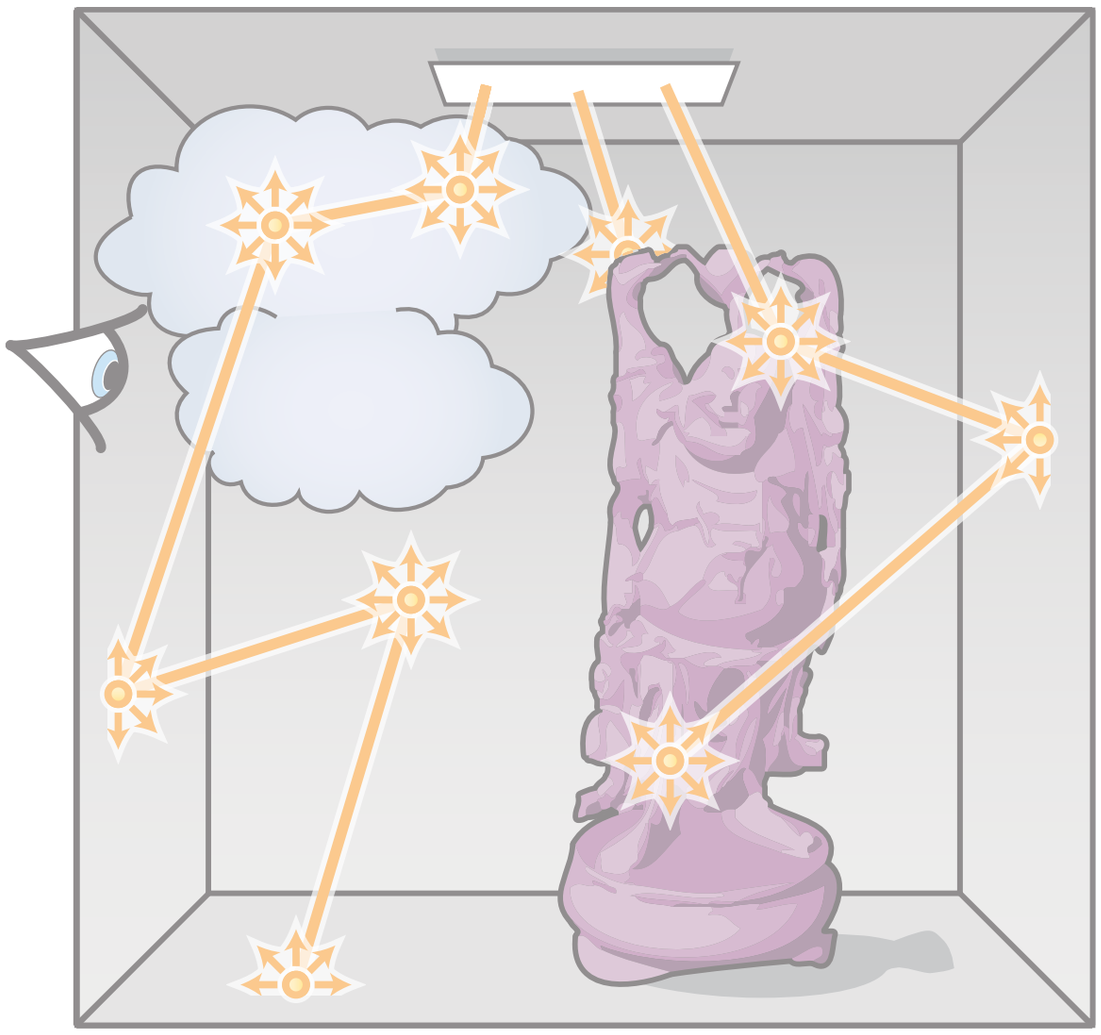


Lighting with VPLs

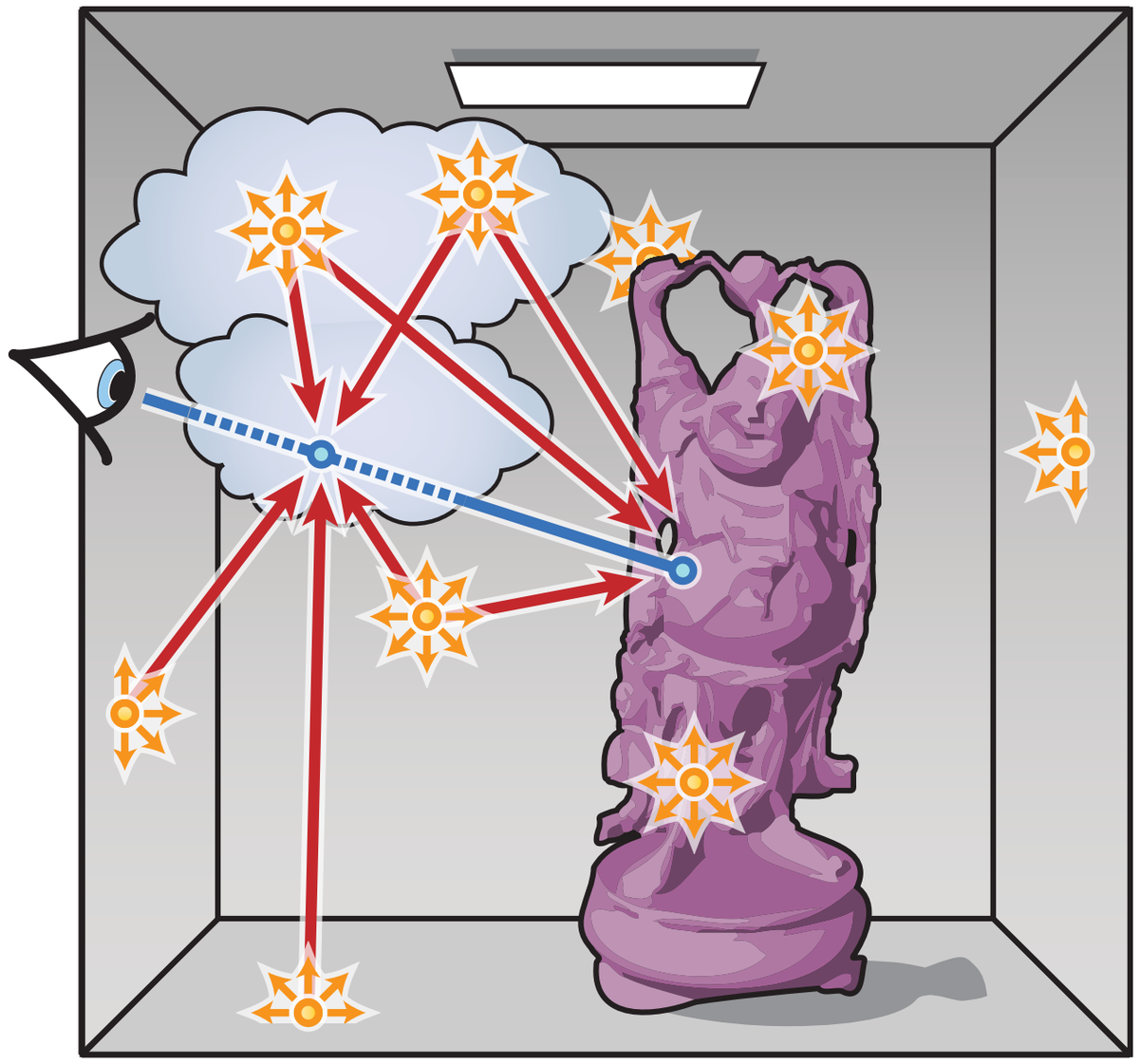


Many-Light Rendering

Generation of VPLs



Lighting with VPLs



Lighting with VPLs

- Rendering equation:

$$L(\mathbf{x}_1, \mathbf{x}_0) = L_e(\mathbf{x}_1, \mathbf{x}_0) + L_r(\mathbf{x}_1, \mathbf{x}_0)$$

$$L_r(\mathbf{x}_1, \mathbf{x}_0) = \int_A f_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_0) G(\mathbf{x}_1\, \mathbf{x}_2) L(\mathbf{x}_2, \mathbf{x}_1)\, dA(\mathbf{x}_2)$$

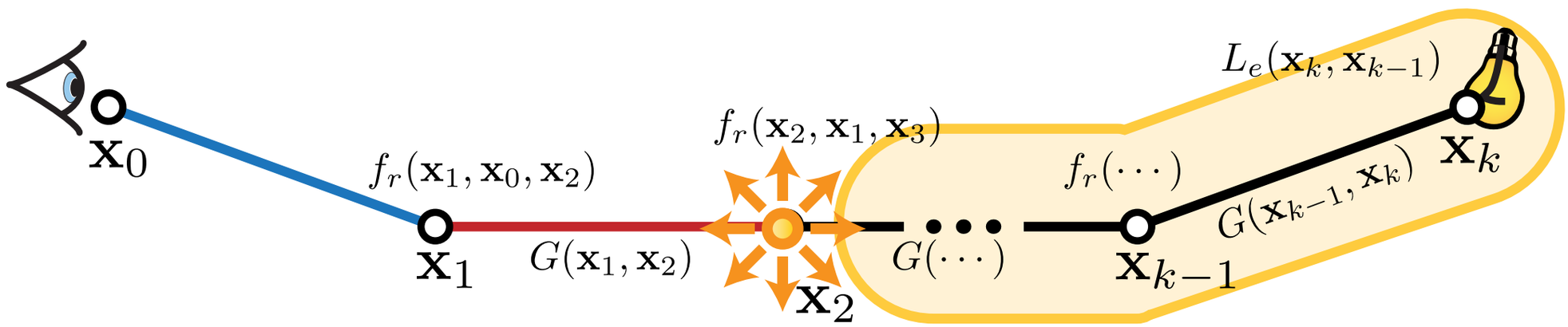
recursion is hidden in the generation of VPLs

$$\approx \sum_{i=1}^{N} f_r(\mathbf{x}_1, \mathbf{x}_{2,i}, \mathbf{x}_0) G(\mathbf{x}_1, \mathbf{x}_{2,i}) f_r(\mathbf{x}_{2,i}, \mathbf{x}_{3,i}, \mathbf{x}_1) I_i$$

Approximation using N VPLs

BRDF at i-th VPL

"Intensity" of i-th VPL



Lighting with VPLs

- Rendering equation:

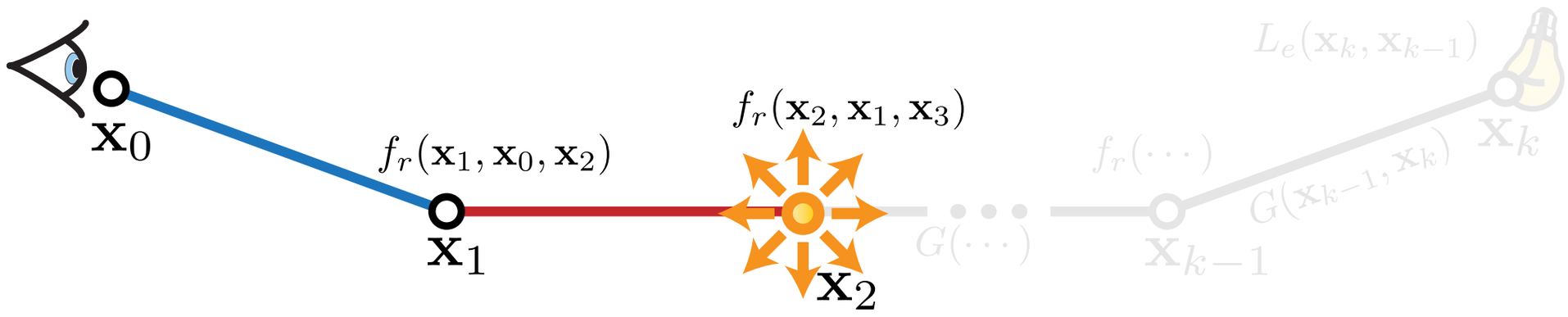
$$L(\mathbf{x}_1, \mathbf{x}_0) = L_e(\mathbf{x}_1, \mathbf{x}_0) + L_r(\mathbf{x}_1, \mathbf{x}_0)$$

$$L_r(\mathbf{x}_1, \mathbf{x}_0) = \int_A f_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_0) G(\mathbf{x}_1\, \mathbf{x}_2) L(\mathbf{x}_2, \mathbf{x}_1)\, dA(\mathbf{x}_2)$$

$$\approx \sum_{i=1}^{N} f_r(\mathbf{x}_1, \mathbf{x}_{2,i}, \mathbf{x}_0) G(\mathbf{x}_1, \mathbf{x}_{2,i}) f_r(\mathbf{x}_{2,i}, \mathbf{x}_{3,i}, \mathbf{x}_1) I_i$$

Throughput of the light path

Intensity of a VPL: $$I = \frac{L_e(\mathbf{x}_k, \mathbf{x}_{k-1}) T(\mathbf{x}_2, \cdots, \mathbf{x}_k)}{p(\mathbf{x}_2, \cdots, \mathbf{x}_k)}$$



Lighting with VPLs

- Rendering equation:

$$L(\mathbf{x}_1, \mathbf{x}_0) = L_e(\mathbf{x}_1, \mathbf{x}_0) + L_r(\mathbf{x}_1, \mathbf{x}_0)$$

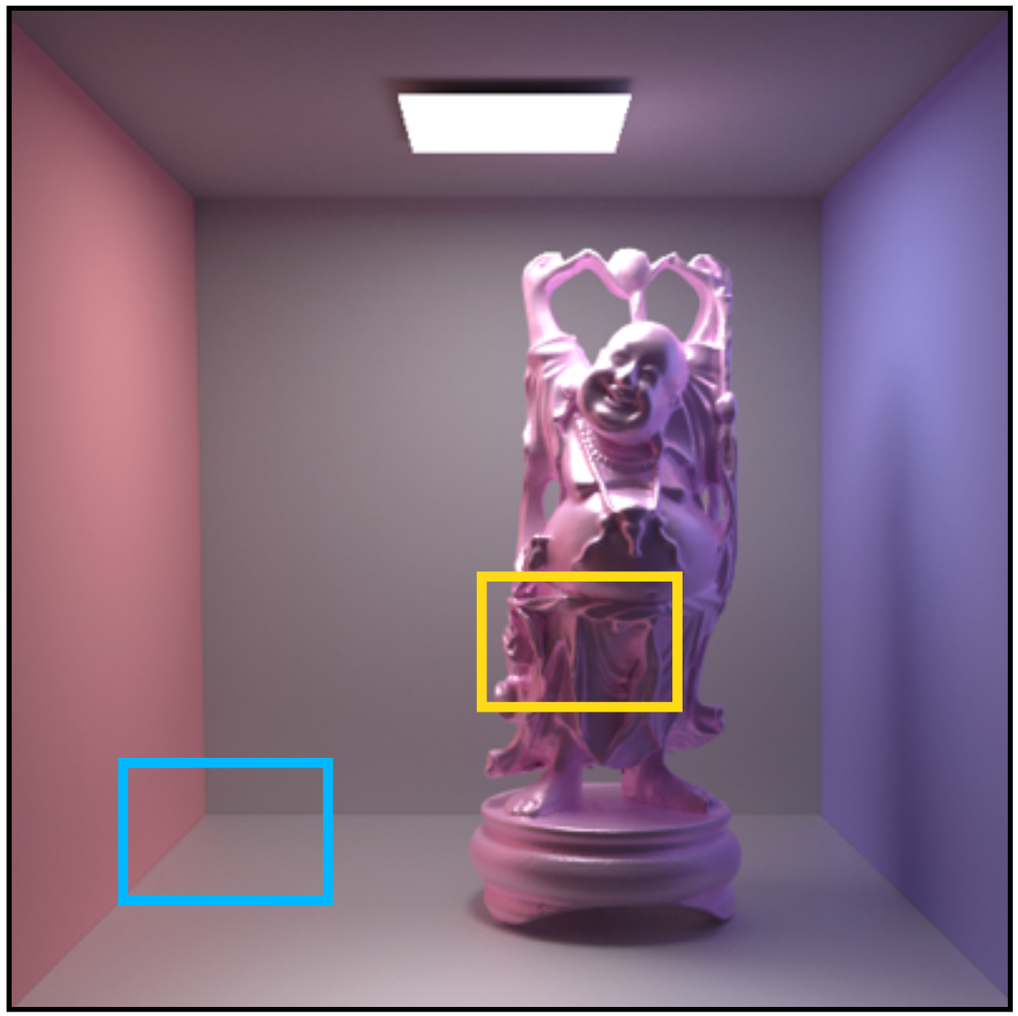
$$L_r(\mathbf{x}_1, \mathbf{x}_0) = \int_A f_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_0) G(\mathbf{x}_1\, \mathbf{x}_2) L(\mathbf{x}_2, \mathbf{x}_1)\, dA(\mathbf{x}_2)$$

$$\approx \sum_{i=1}^{N} f_r(\mathbf{x}_1, \mathbf{x}_{2,i}, \mathbf{x}_0) G(\mathbf{x}_1, \mathbf{x}_{2,i}) f_r(\mathbf{x}_{2,i}, \mathbf{x}_{3,i}, \mathbf{x}_1)\, I_i$$

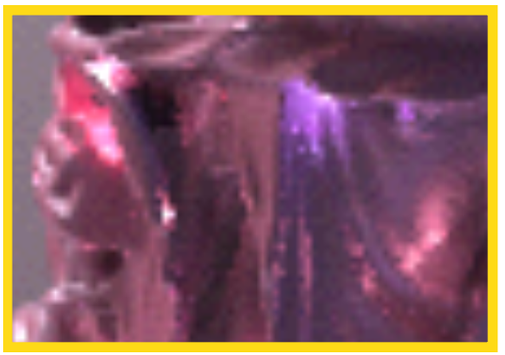
- For each VPL we store:
  - position
  - intensity
  - direction towards the previous vertex on the light path
  - BRDF & local shading frame (normal, tangent)

Lighting with VPLs

Reference



Approximation with VPLs

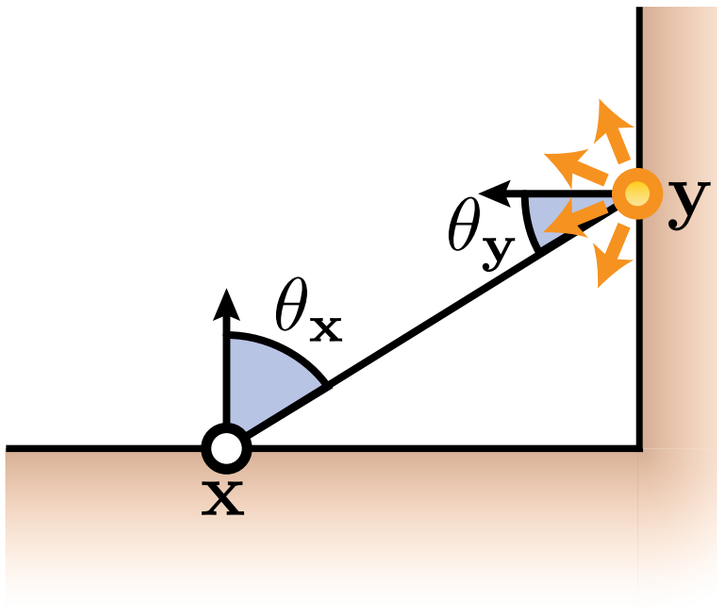


Lighting with VPLs

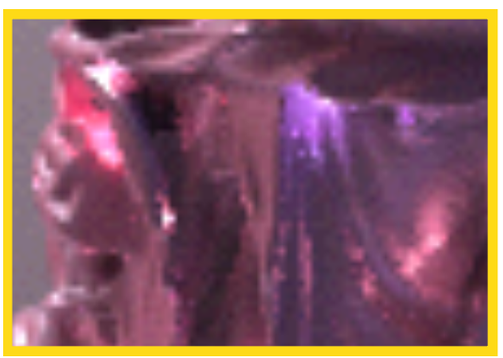
Splotches!!!

Reason: singularity in G-term

$$G(\mathbf{x},\mathbf{y}) = V(\mathbf{x},\mathbf{y})\frac{\cos\theta_{\mathbf{x}}\cos\theta_{\mathbf{y}}}{\|\mathbf{x}-\mathbf{y}\|^2}$$



Approximation with VPLs

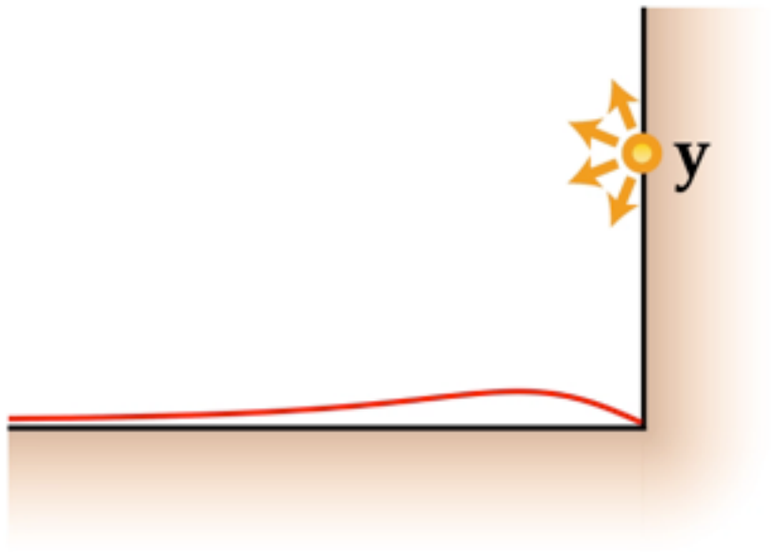


Lighting with VPLs

Splotches!!!

Reason: singularity in G-term

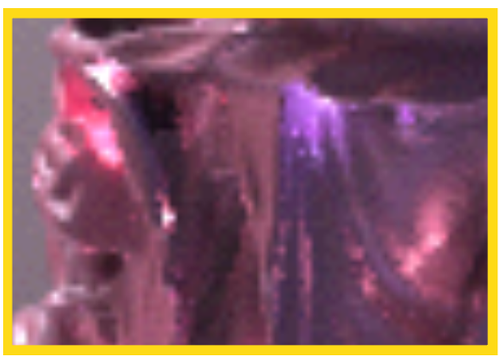
$$G(\mathbf{x},\mathbf{y}) = V(\mathbf{x},\mathbf{y})\frac{\cos\theta_{\mathbf{x}}\cos\theta_{\mathbf{y}}}{\|\mathbf{x}-\mathbf{y}\|^2}$$



Approximation with VPLs

All points are lit by the same set of VPLs

Correlation shows up as splotches



Bounding the G-term

- Remove the singularity by bounding (clamping) the geometry term to some maximum

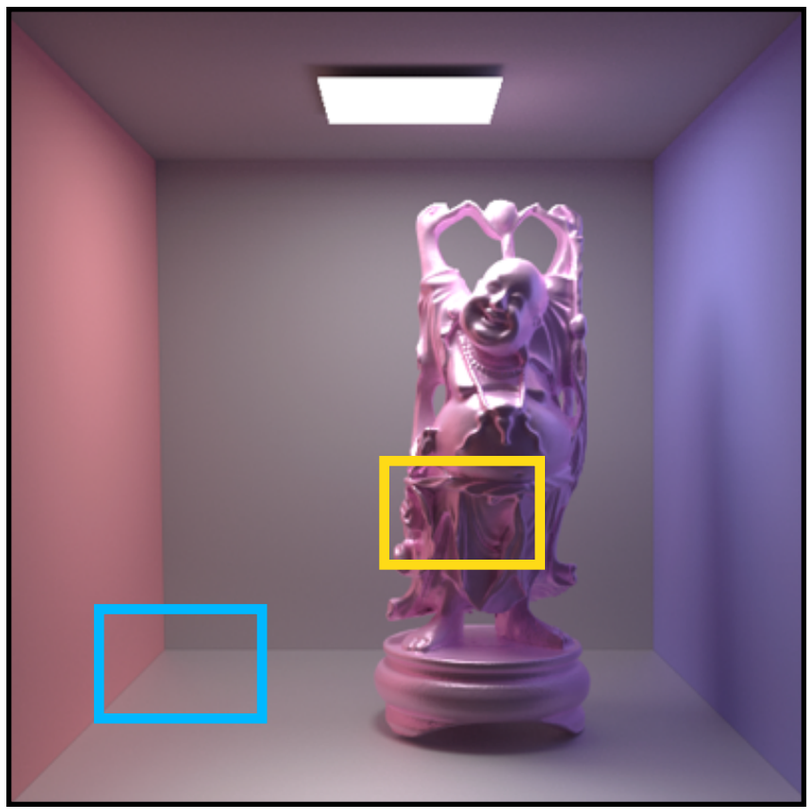
$$G_b(\mathbf{x}, \mathbf{y}) = \min(G(\mathbf{x}, \mathbf{y}), b)$$

User-defined maximum value

- Extremely simple and cheap 
- Removes short-distance light transport, cavities appear darker 

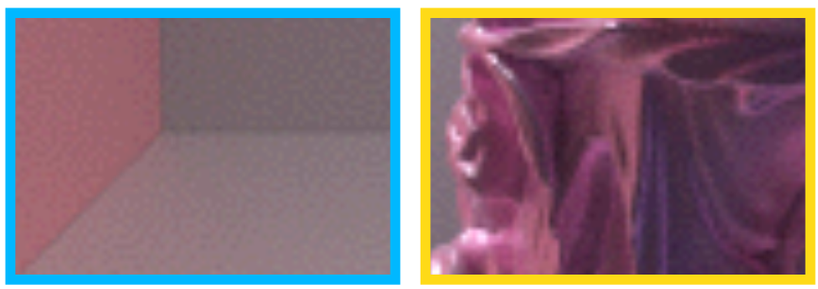
Bounding the G-term

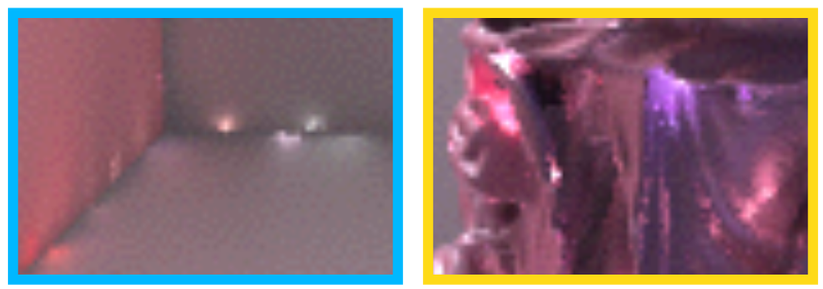
Reference

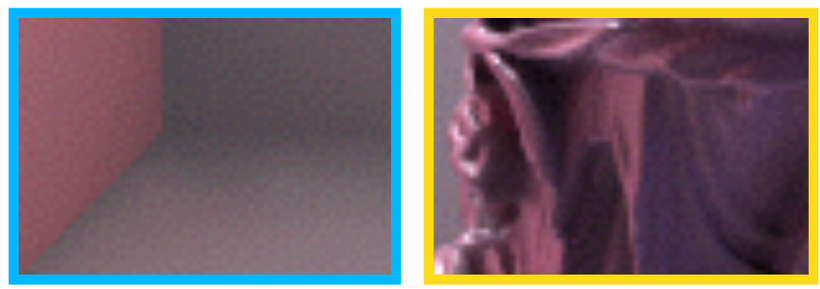


VPLs

VPLs with bounded G





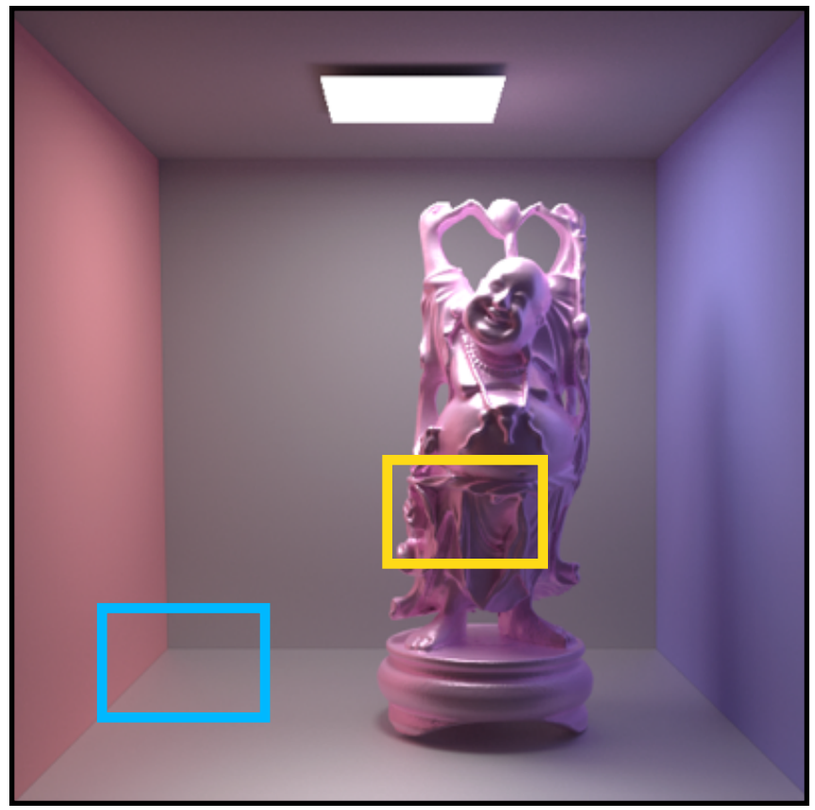


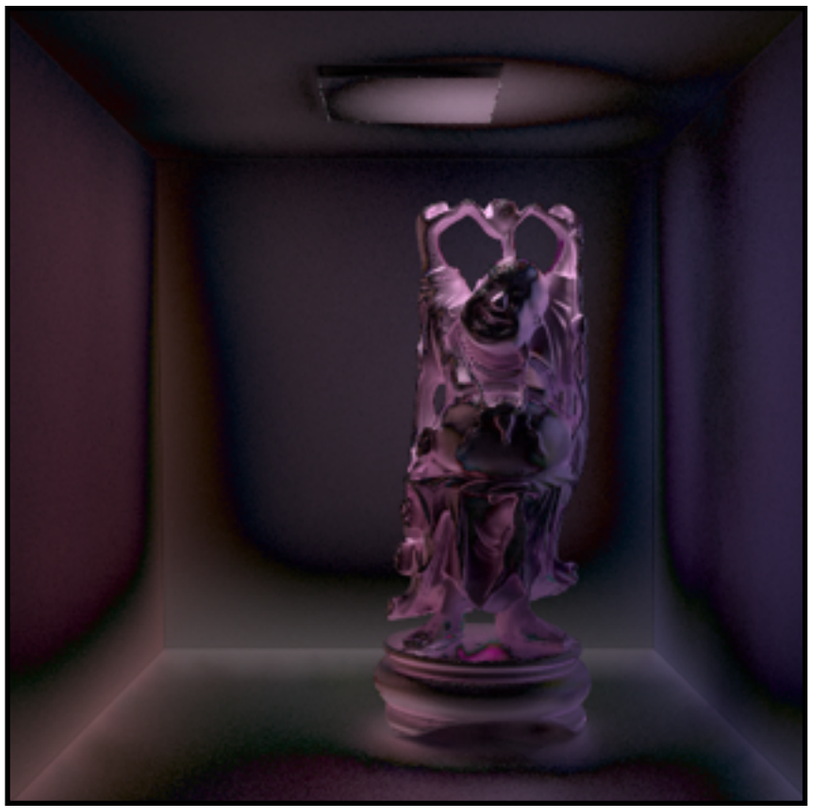
using $G(\mathbf{x}, \mathbf{y})$

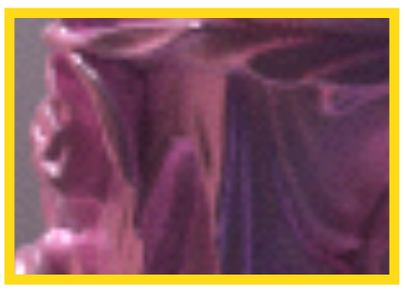
using $\min(G(\mathbf{x}, \mathbf{y}), b)$

Bounding the G-term

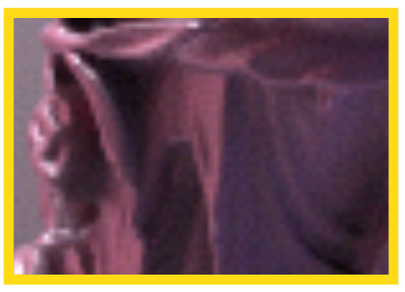
Reference ⟵ Difference ⟶ VPLs with bounded G







$$G(\mathbf{x}, \mathbf{y}) - G_b(\mathbf{x}, \mathbf{y})$$



We need to compensate for the energy loss (bias)!

Bias Compensation

- Use bounded VPLs to obtain a noise free, *biased* solution
- Recover the lost energy using different technique
  - [Kollig and Keller 2004] (use path tracing for bias compensation: slooow),
    [Raab et al. 2008] (path-traced bias compensation for volumes),
    [Davidovič et al. 2010] (create local lights, approximate),
    [Novák et al. 2011] (bias compensation in screen-space, approximate)
    [Engelhardt et al. 2012] (approximate bias compensation for volumes)
- Always some trade-off… speed, quality, elegance

Glossy Surfaces

- Glossy surfaces further emphasize splotches
  - Naive solution: bound the product $f_r(\mathbf{x}_1 \cdots) G(\mathbf{x}_1, \mathbf{x}_2) f_r(\mathbf{x}_2 \cdots)$

VPLs

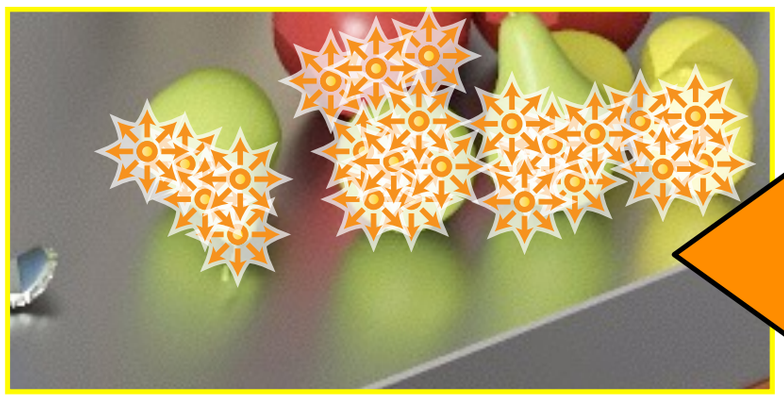
VPLs with bounded product

Reference

Local VPLs

- Solution 1: create additional (local) VPLs [Davidovič et al. 2010]
  - Global (bounded) VPLs distributed traditionally
  - Local VPLs distributed from camera, add the missing energy only

VPLs

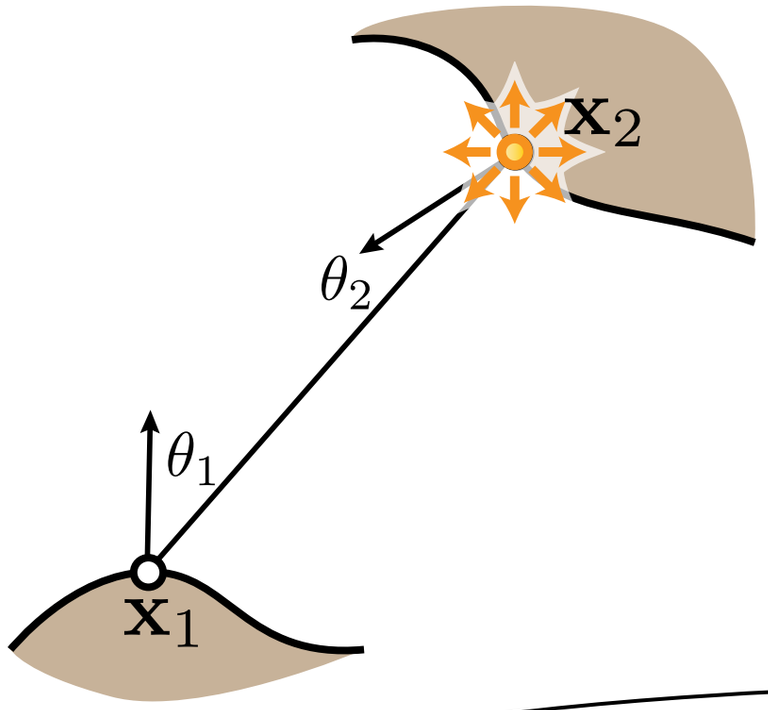


Reference

Finite-size Lights

- Solution 2: spread the energy spatially instead of concentrating it at single points
- Virtual *Spherical* Lights [Hašan et al. 2009]
  - energy distributed over surfaces inside a sphere
- Virtual *Ray/Beam* Lights [Novák et al. 2012ab]
  - energy is distributed along infinitesimal rays or beams with finite thickness [only for volumes]

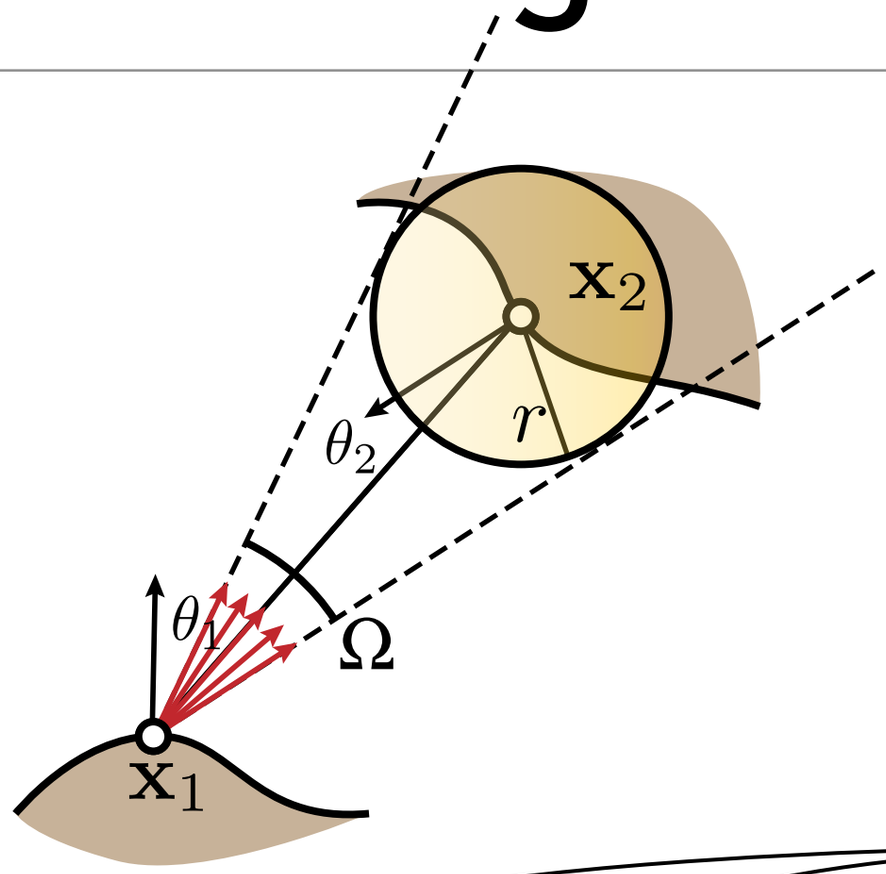
Virtual Spherical Lights



splotches!

point-to-point: $$I\, f_r(\mathbf{x}_1, \cdots) f_r(\mathbf{x}_2, \cdots) \frac{\cos\theta_1 \cos\theta_2}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

Virtual Spherical Lights



splotches!

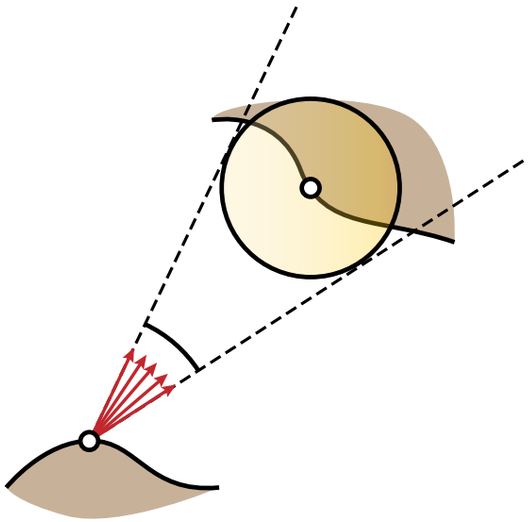
point-to-point: $$I\, f_r(\mathbf{x}_1, \cdots) f_r(\mathbf{x}_2, \cdots) \frac{\cos\theta_1 \cos\theta_2}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

approx. disk-to-point: $$\frac{I}{\pi r^2} \int_\Omega f_r(\mathbf{x}_1, \cdots) f_r(\mathbf{x}_2, \cdots) \cos\theta_1 \cos\theta_2 d\omega$$

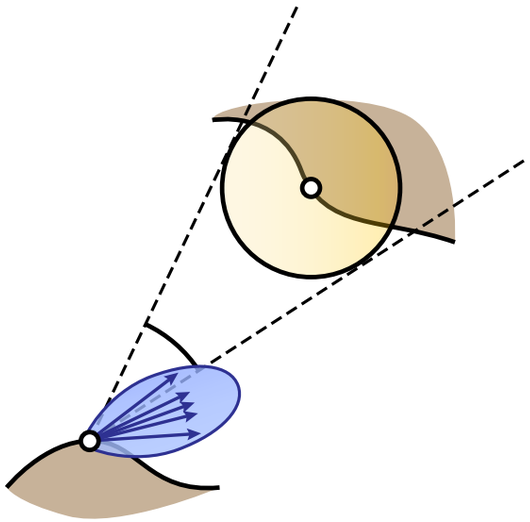
Surfaces inside the sphere are approximated by a disk facing $\mathbf{x}_1$

Virtual Spherical Lights

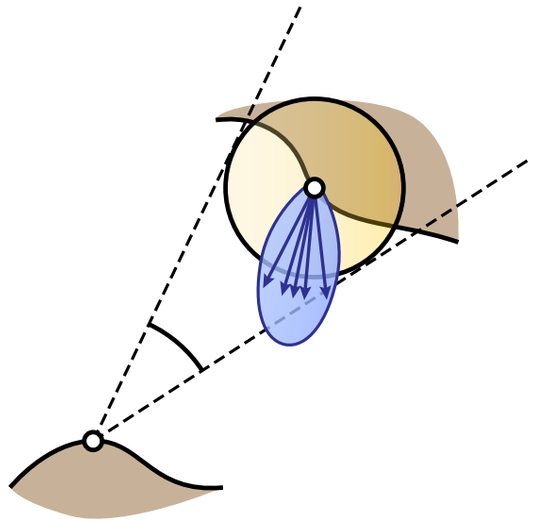
- Importance sampling the integral:



cone sampling

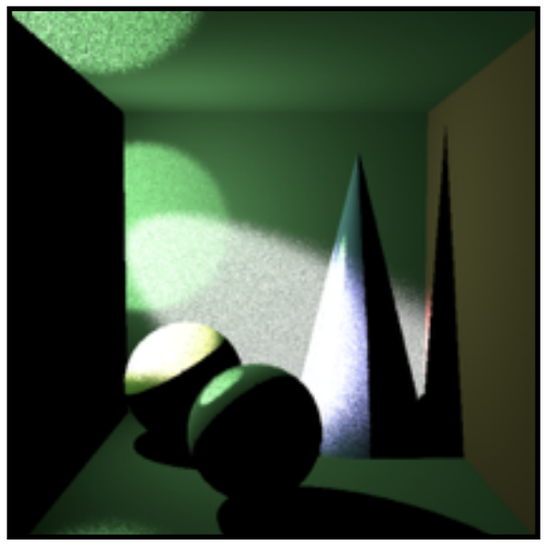


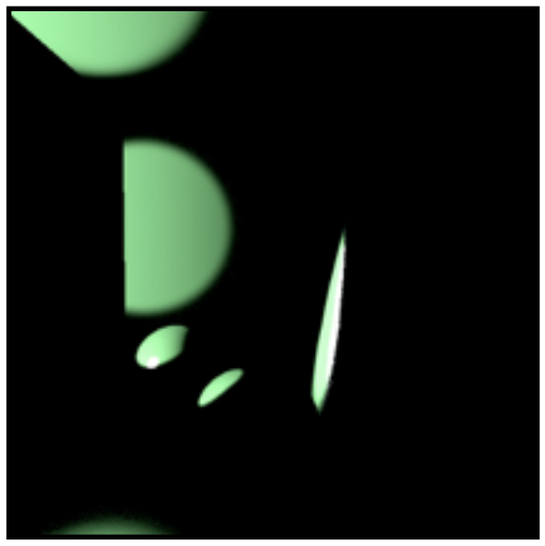
BRDF1 sampling

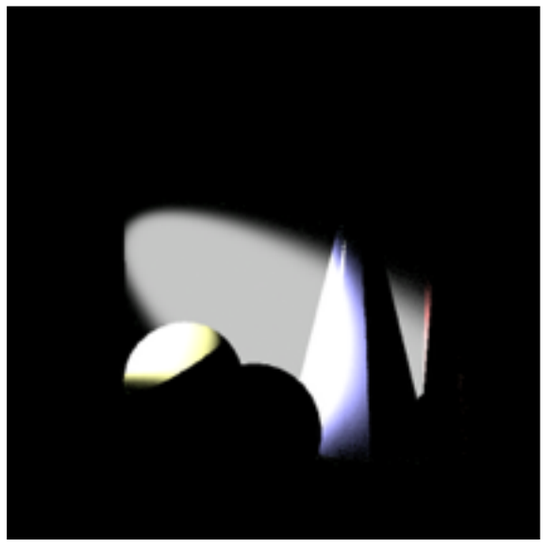


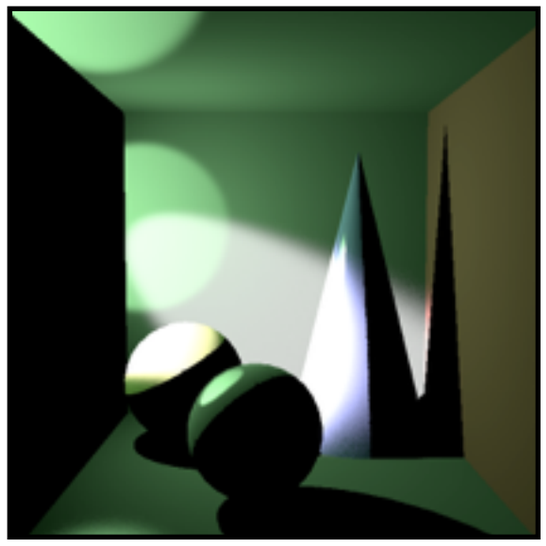
BRDF2 sampling

Multiple importance sampling







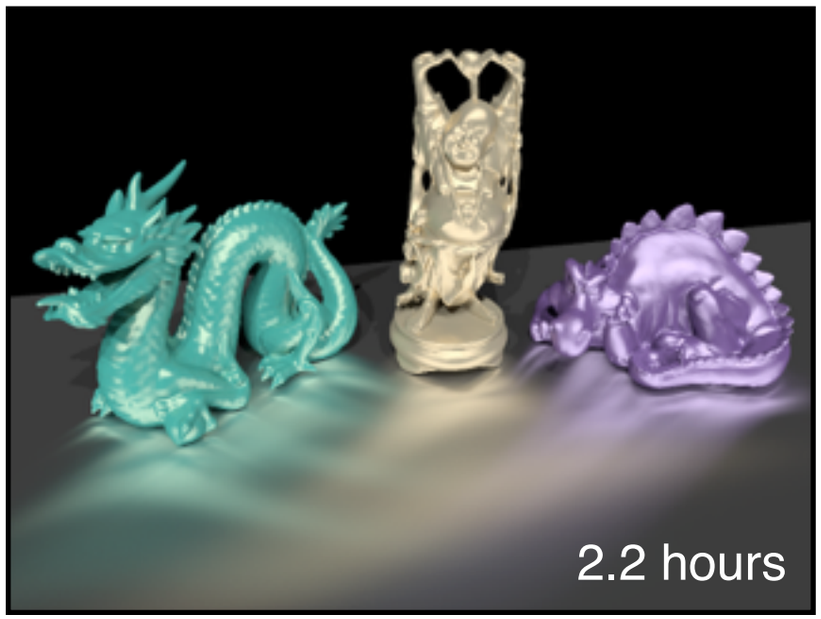


Virtual Spherical Lights

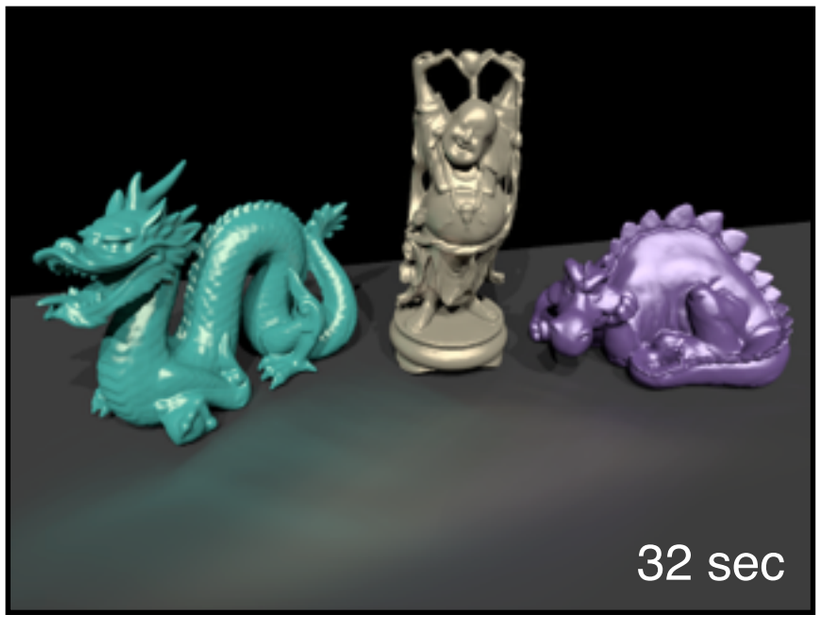
- Idea: spread the energy spatially instead of concentrating it at single points
- Advantages:
  - energy is blurred, not clamped
- Disadvantages:
  - blurring of energy introduces (another kind of) bias
  - requires an integral over the solid angle

Virtual Spherical Lights

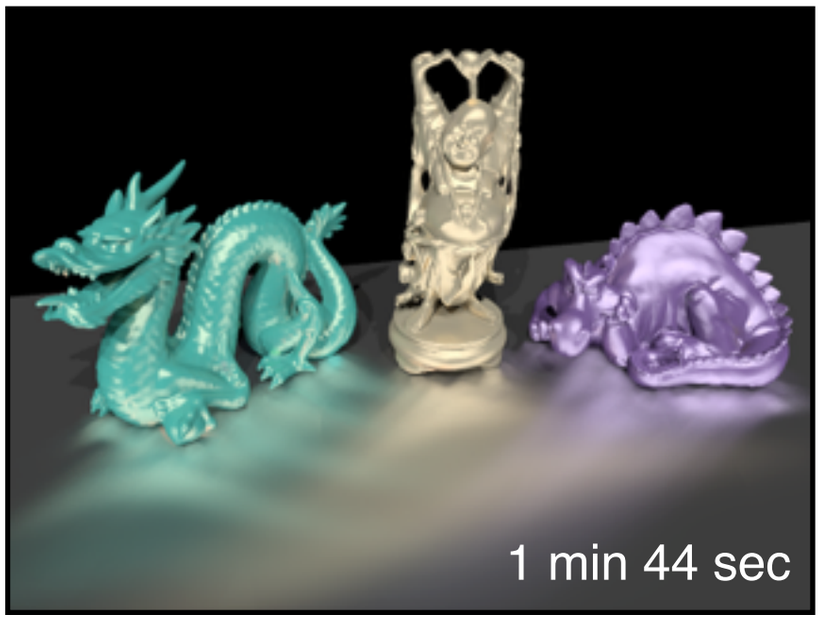
Reference



Bounded

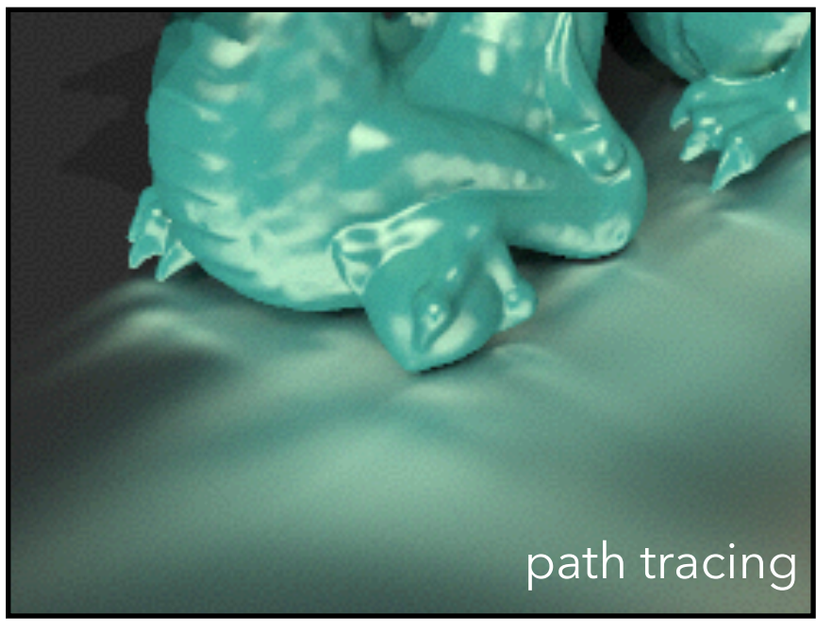


VSLs

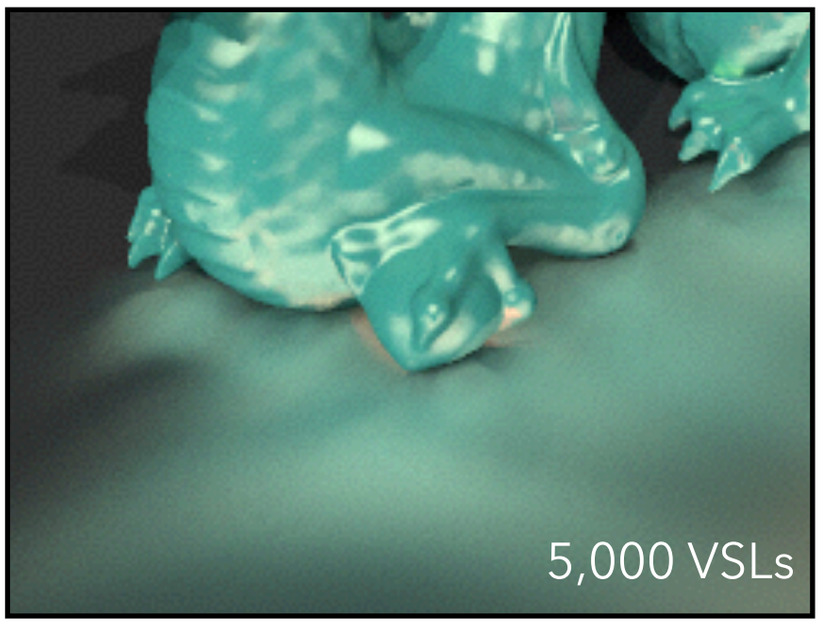


Virtual Spherical Lights

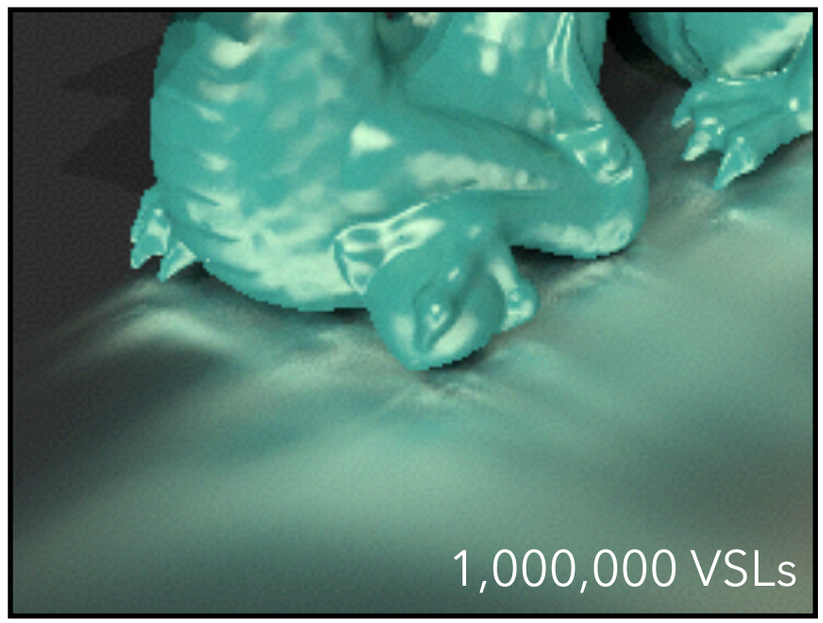
Reference



VSLs



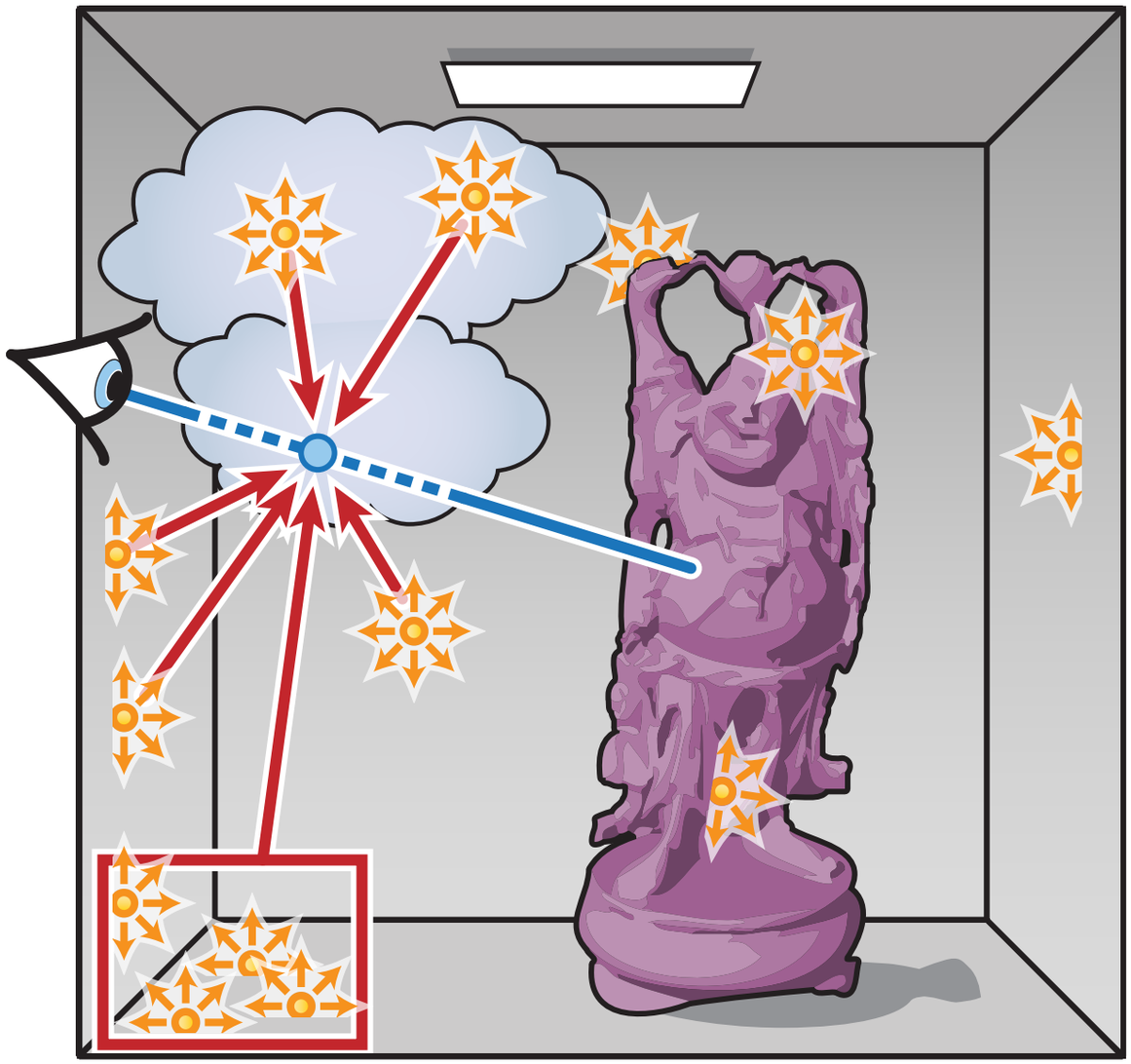
VSLs converged

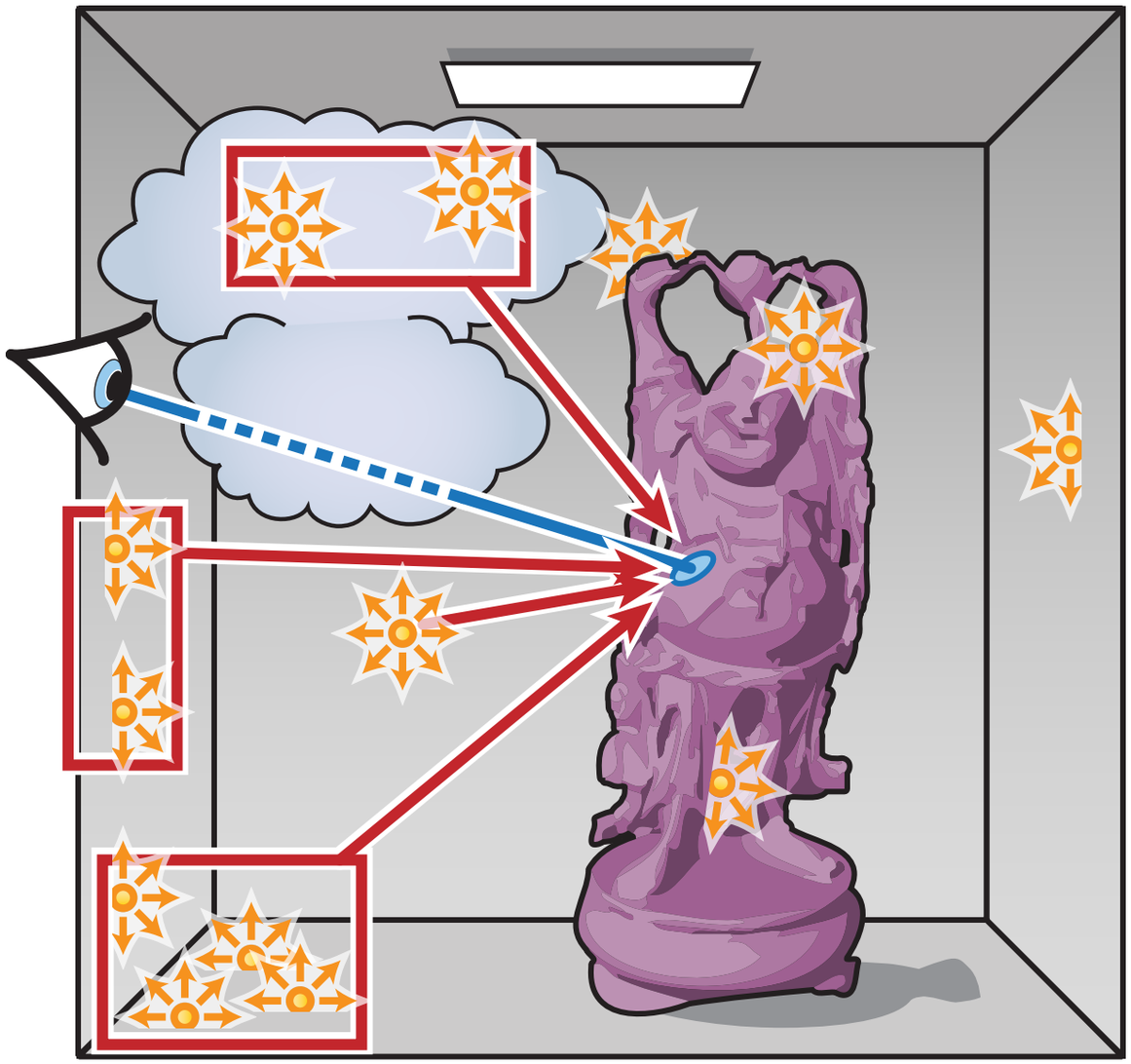


Linear cost = too expensive!

Scalability

- Goal: sub-linear evaluation cost



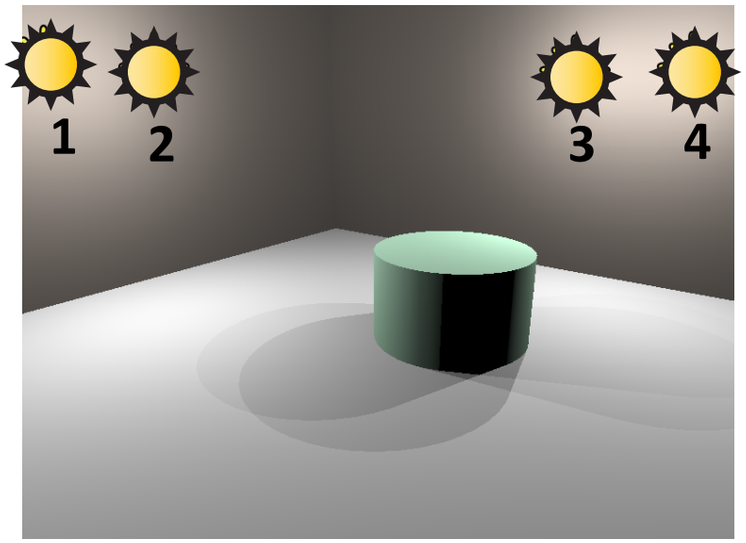


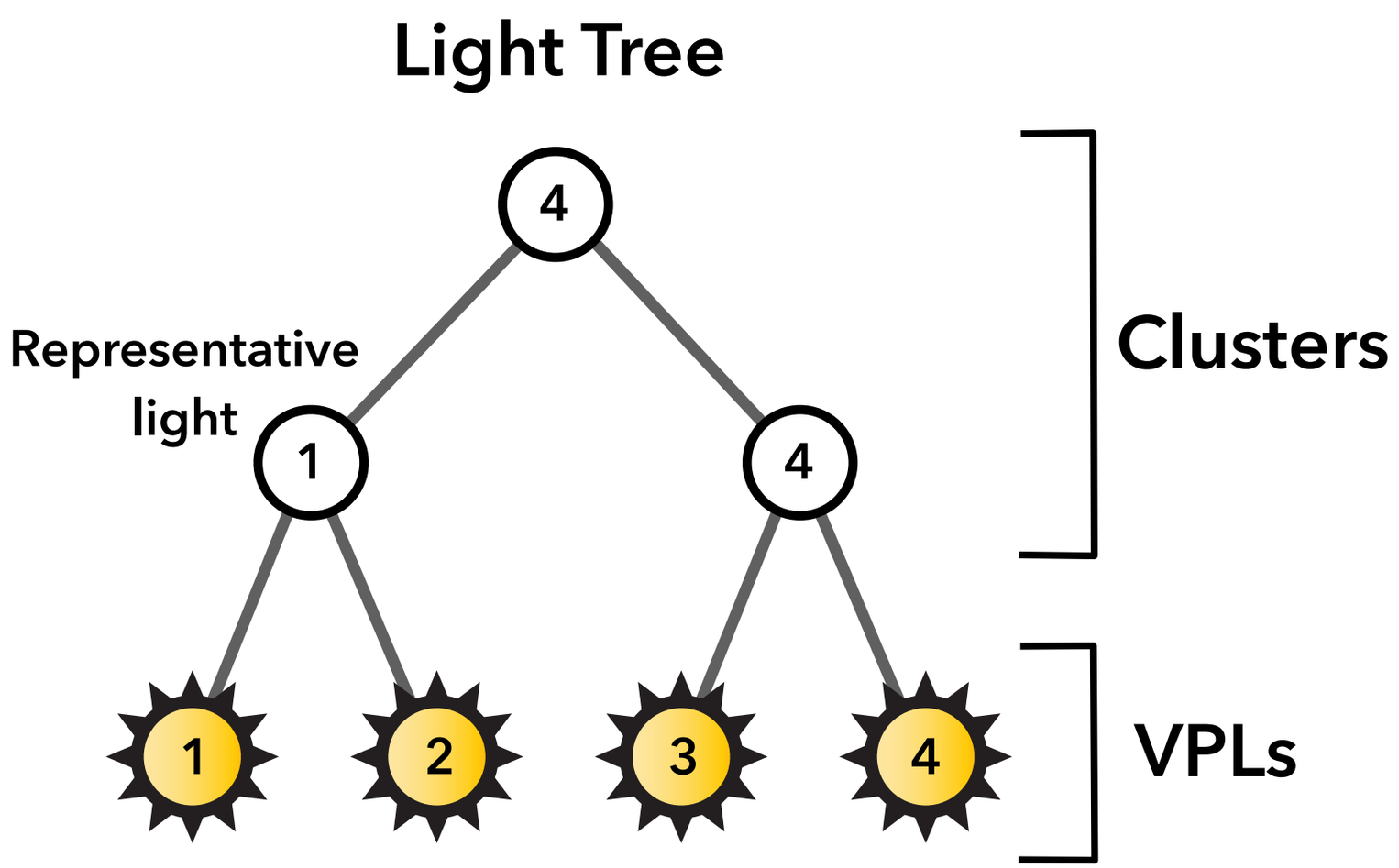
Scalability

- Goal: sub-linear evaluation cost
- Clustering of virtual lights:
  - Lightcuts [Walter et al. 2006]
  - Matrix Row-Column Sampling [Hašan et al. 2007]
  - Lightslice [Ou and Pellacini 2011]
- Importance sampling VPLs [Georgiev et al. 2012]

Lightcuts

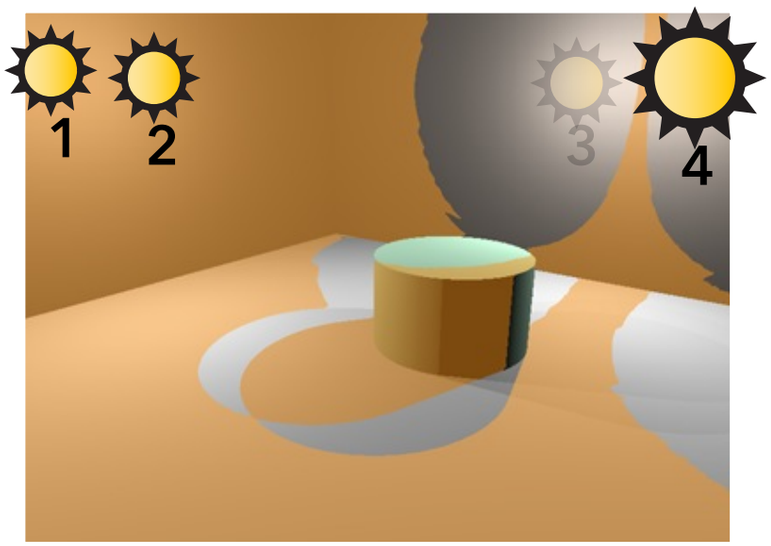
- Preprocess: build a *tree hierarchy*
- Rendering: adaptive clustering of VPLs

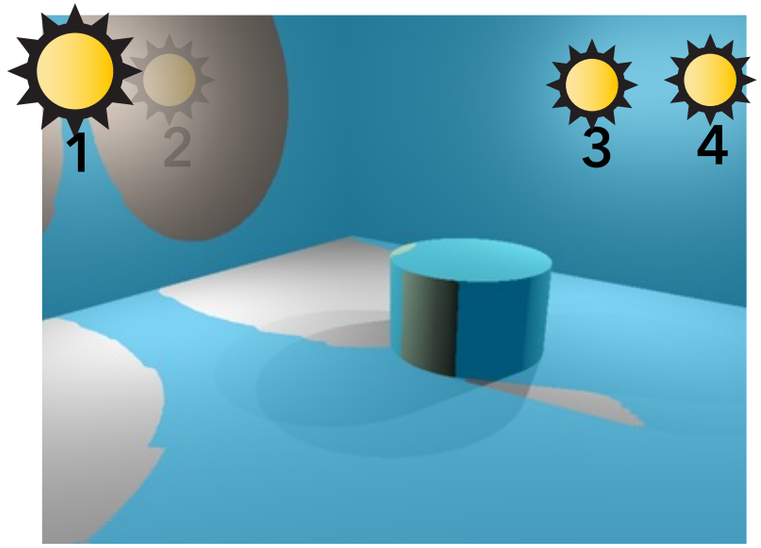


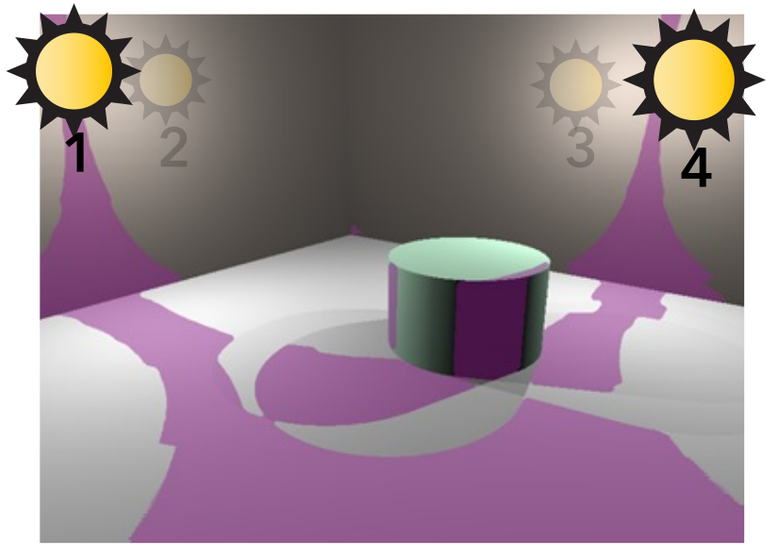


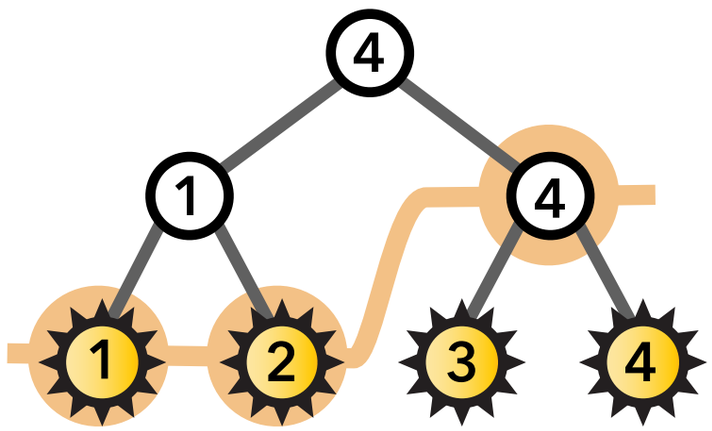
Lightcuts

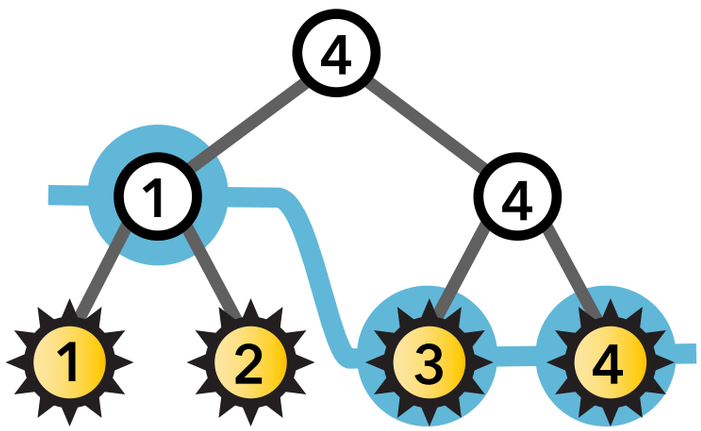
Three example cuts

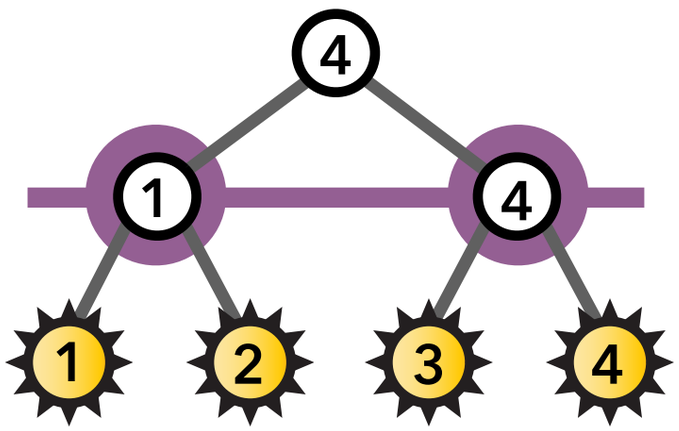






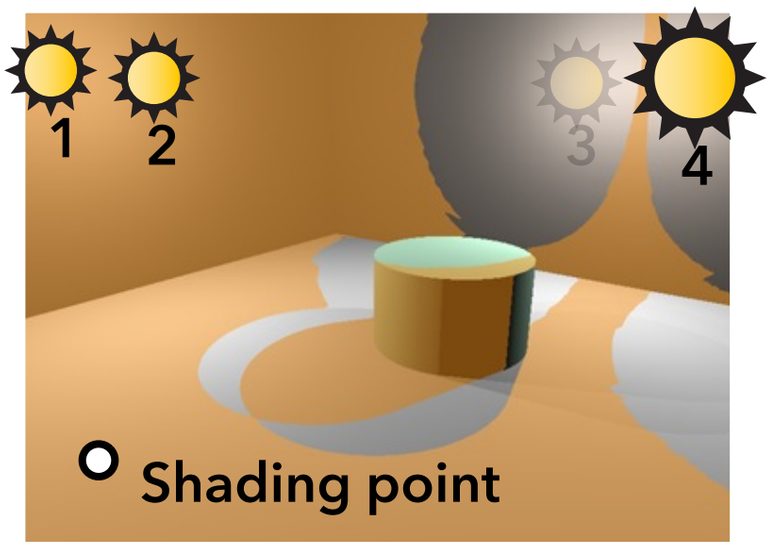


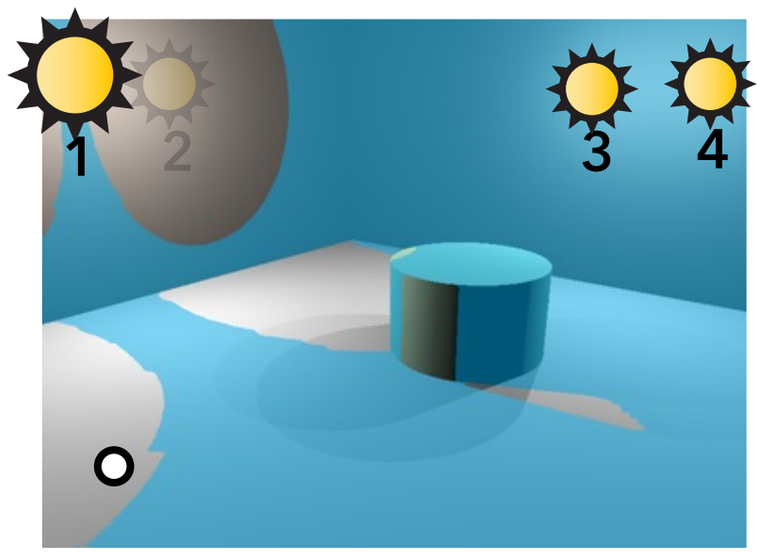


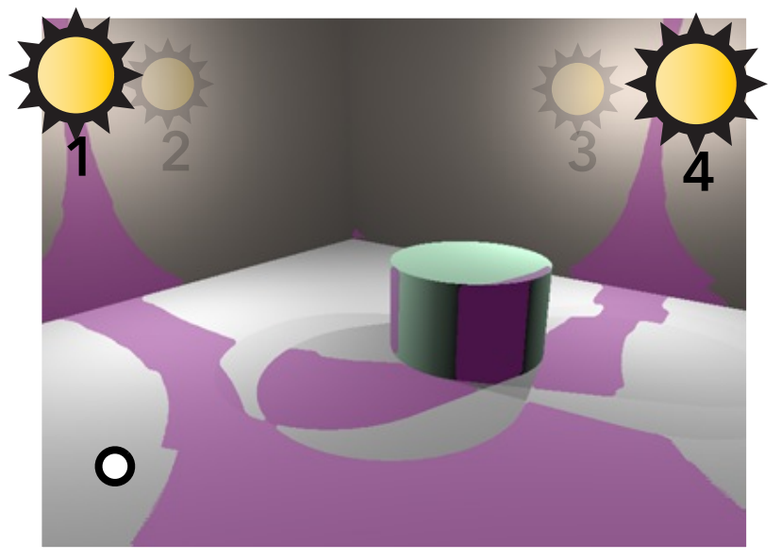


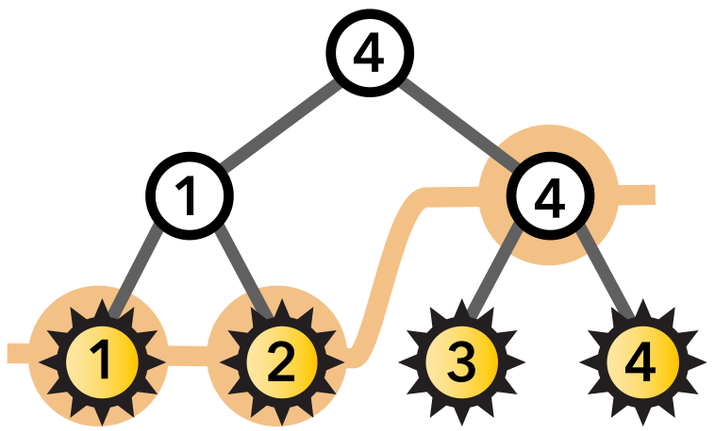
Lightcuts

Three example cuts

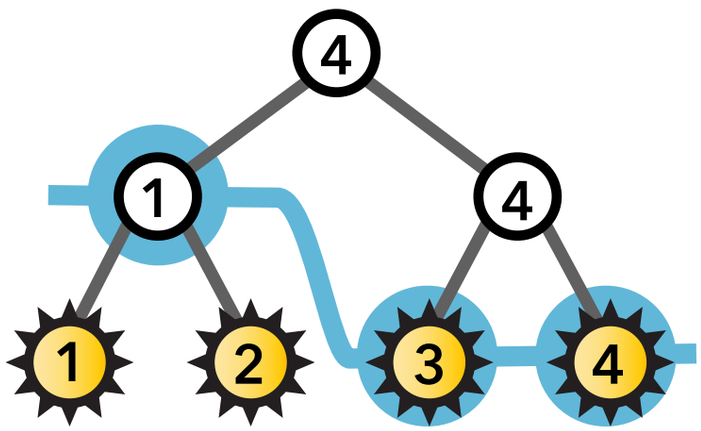




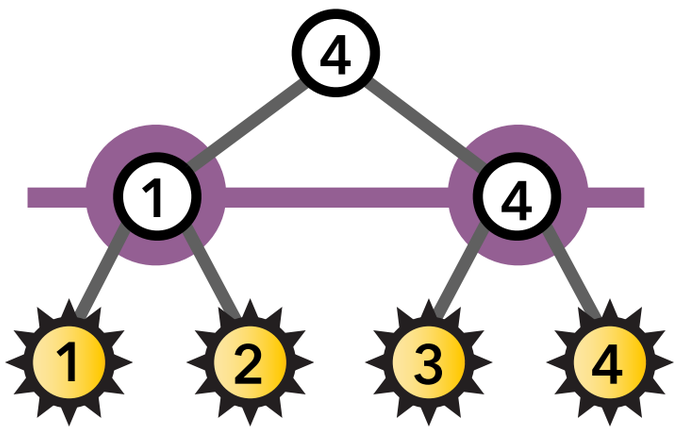




GOOD



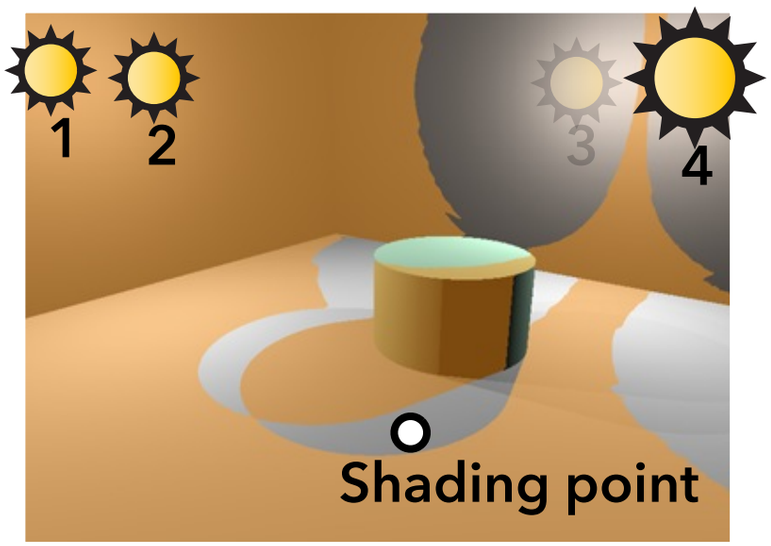
BAD

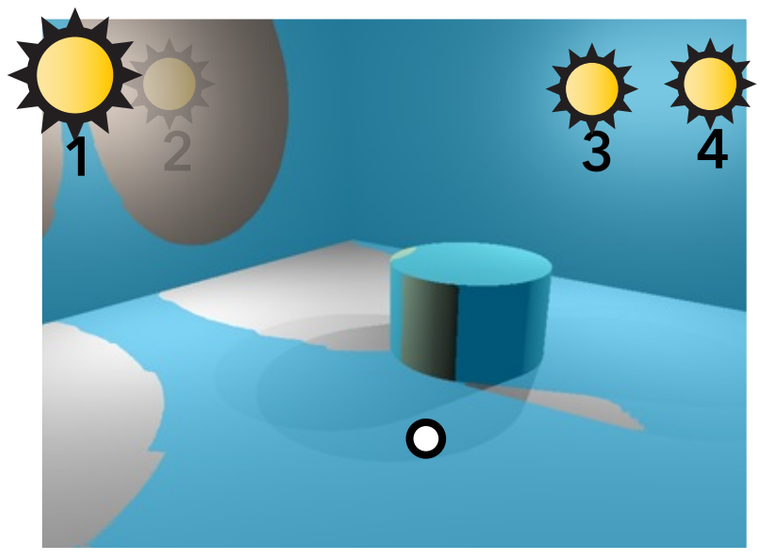


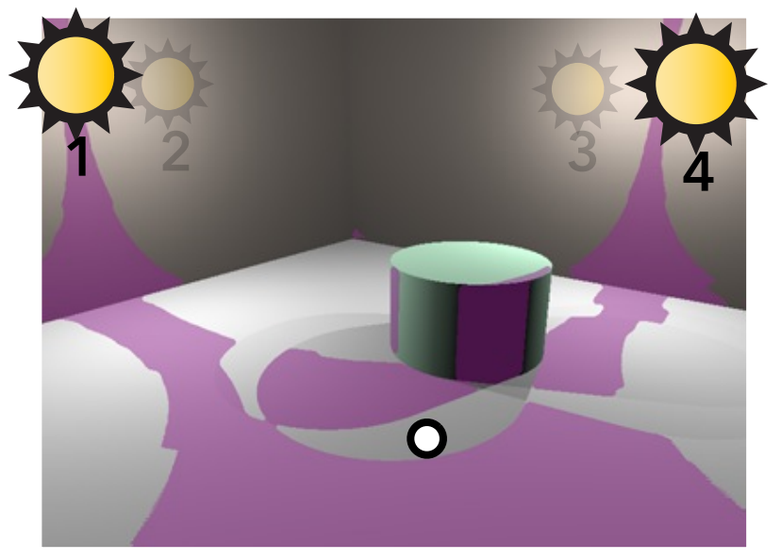
BAD

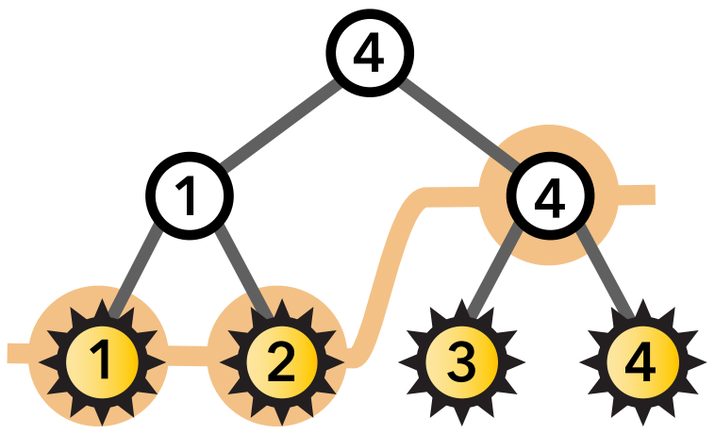
Lightcuts

Three example cuts

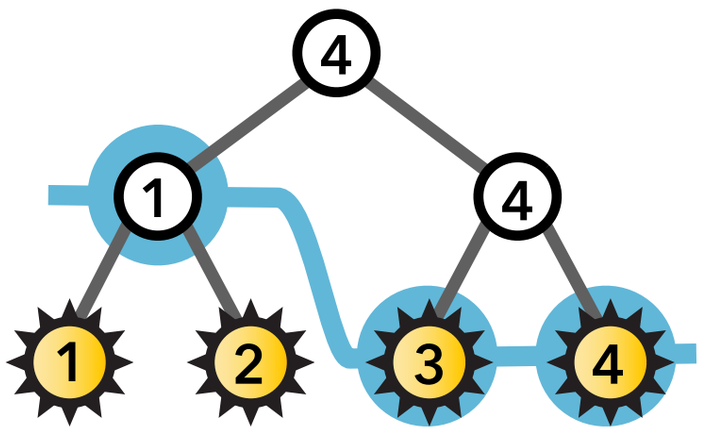




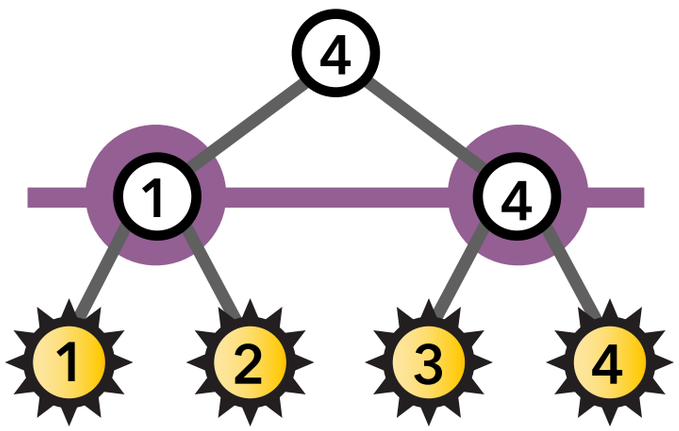




BAD



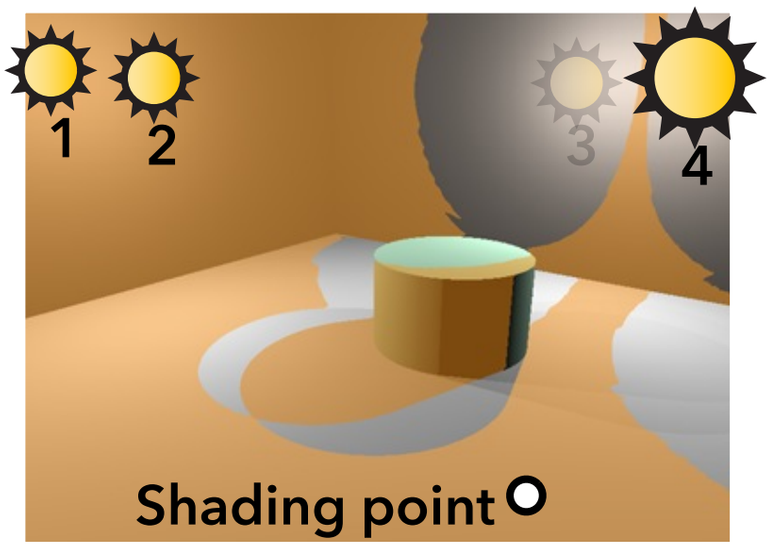
GOOD

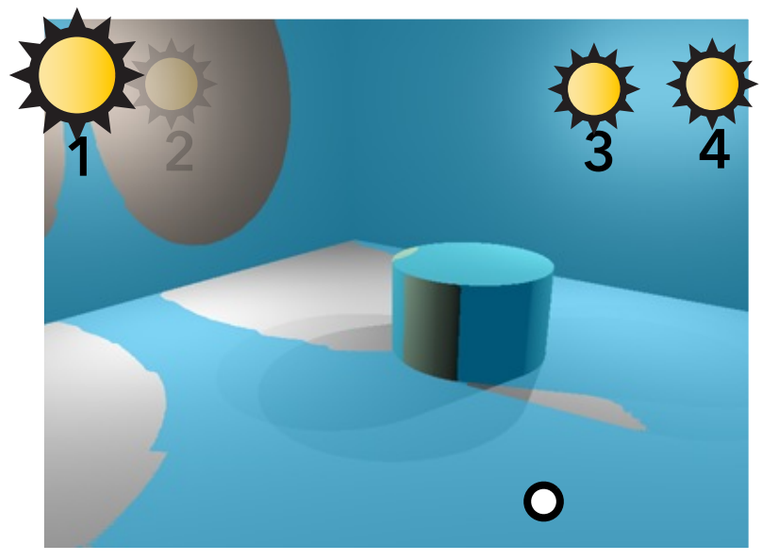


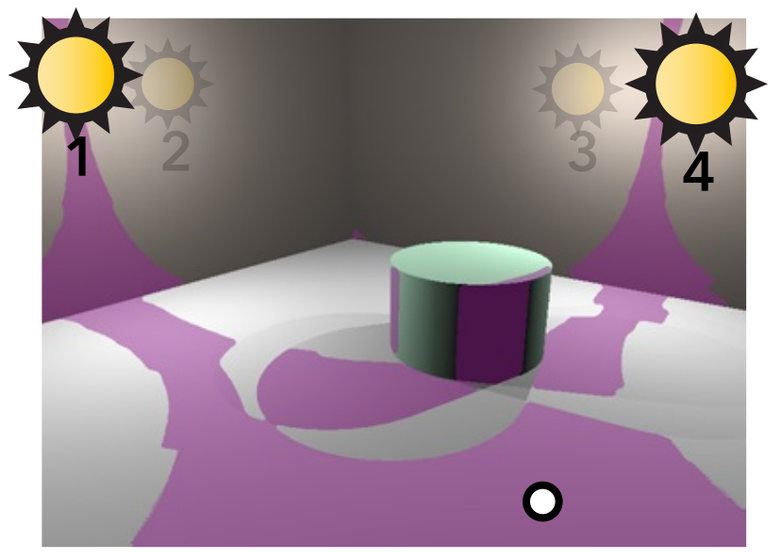
BAD

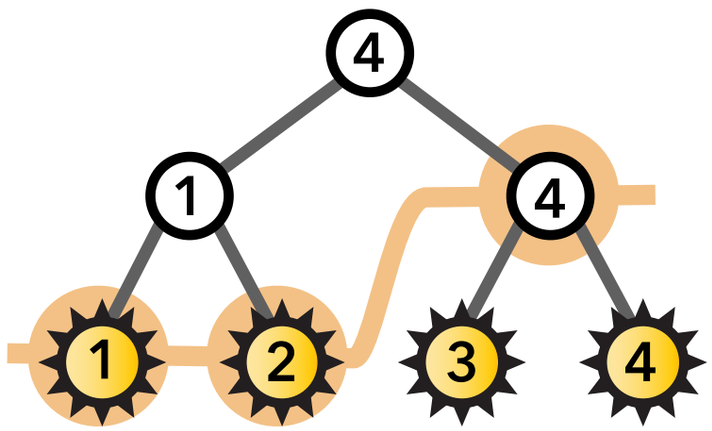
Lightcuts

Three example cuts

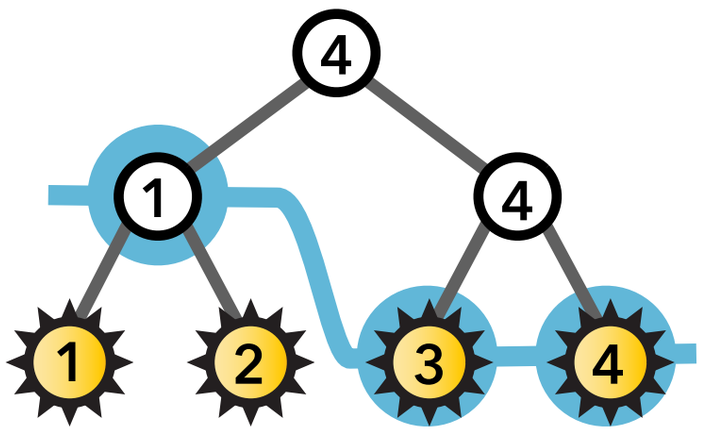




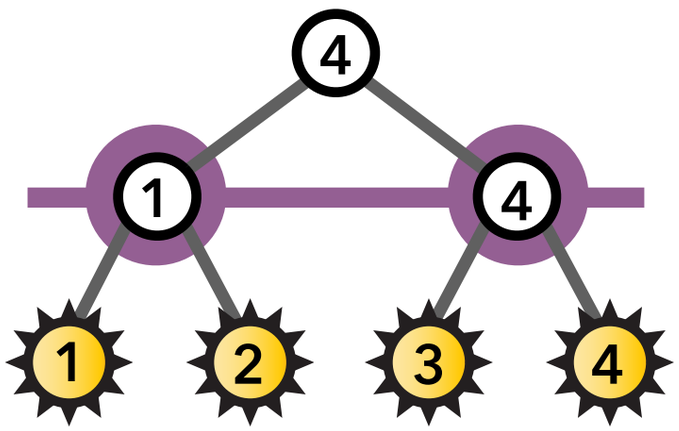




GOOD



GOOD

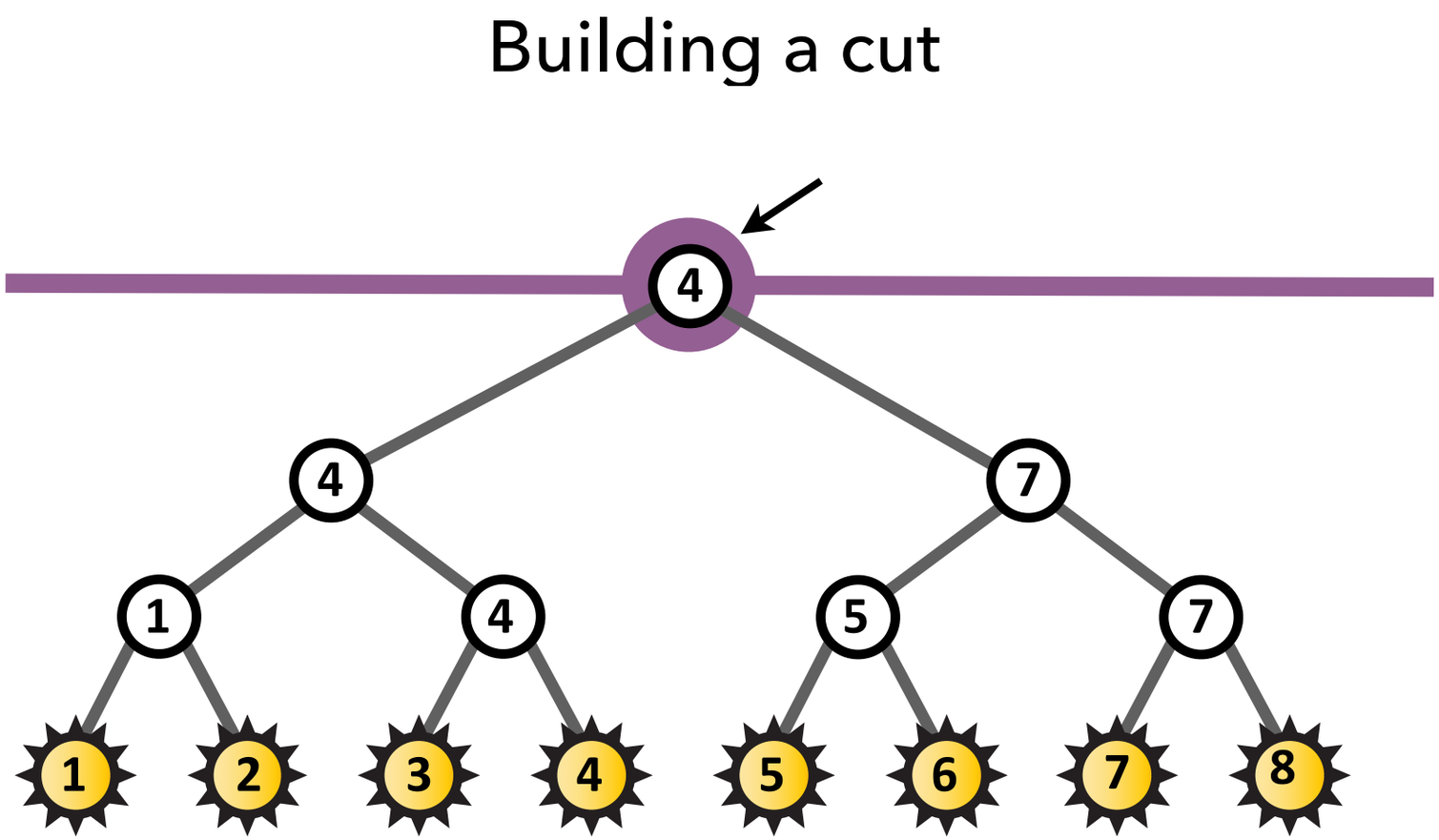


GOOD

Lightcuts

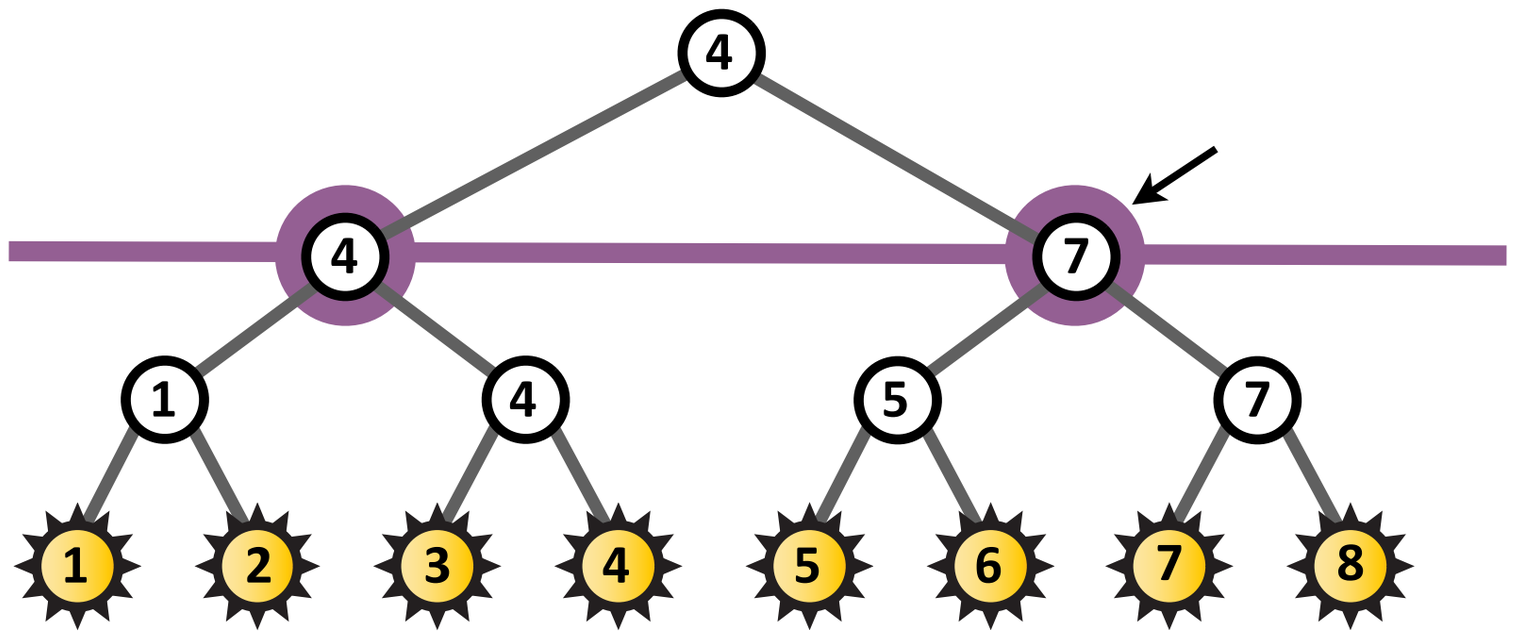
- Preprocess
  - convert illumination to points
  - build a light tree
- Rendering:
  - choose a cut for each shading point
  - avoid visible artifacts (bound maximum predicted error)
  - refine the cut if necessary

Lightcuts



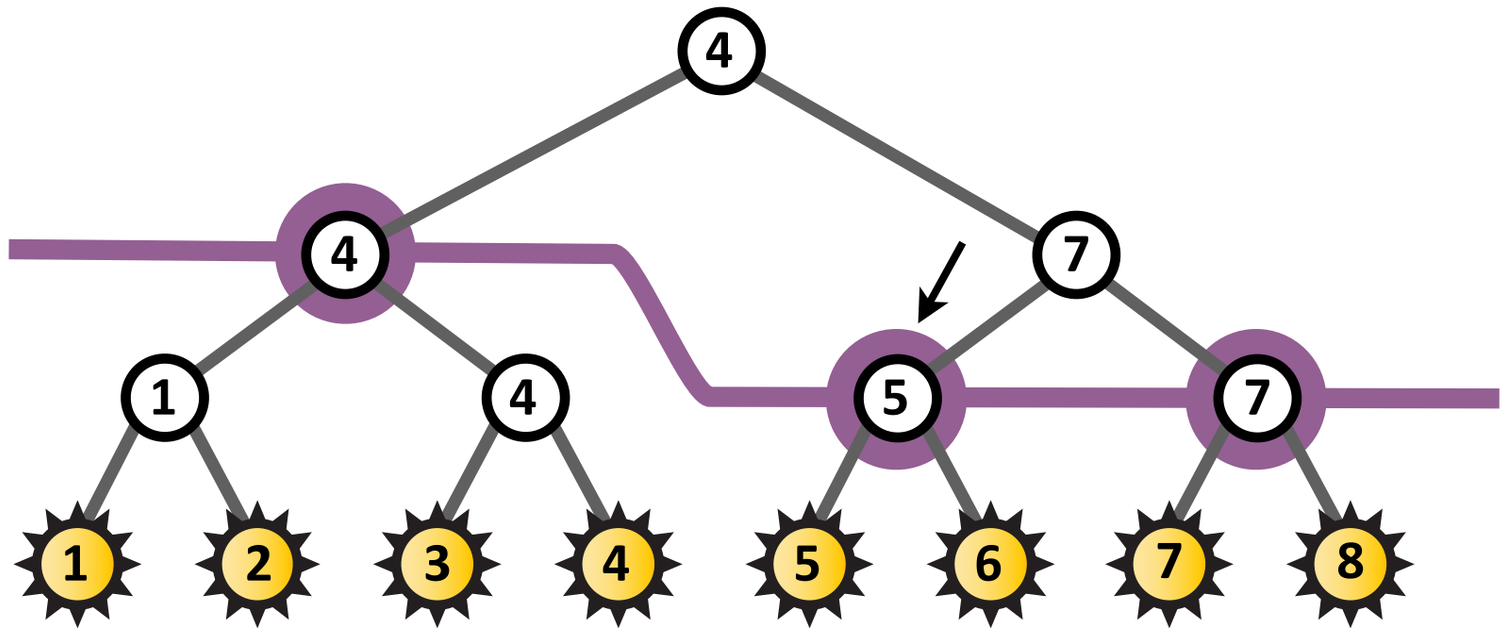
Lightcuts

Building a cut



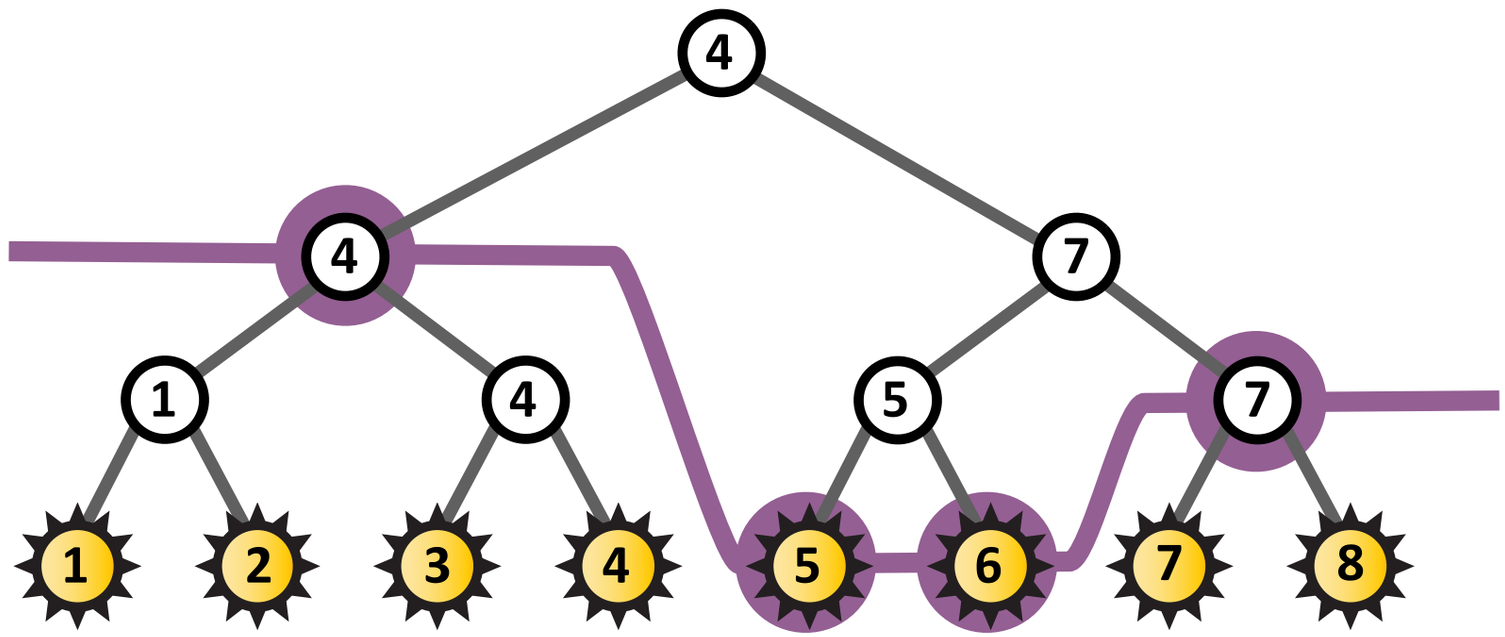
Lightcuts

Building a cut

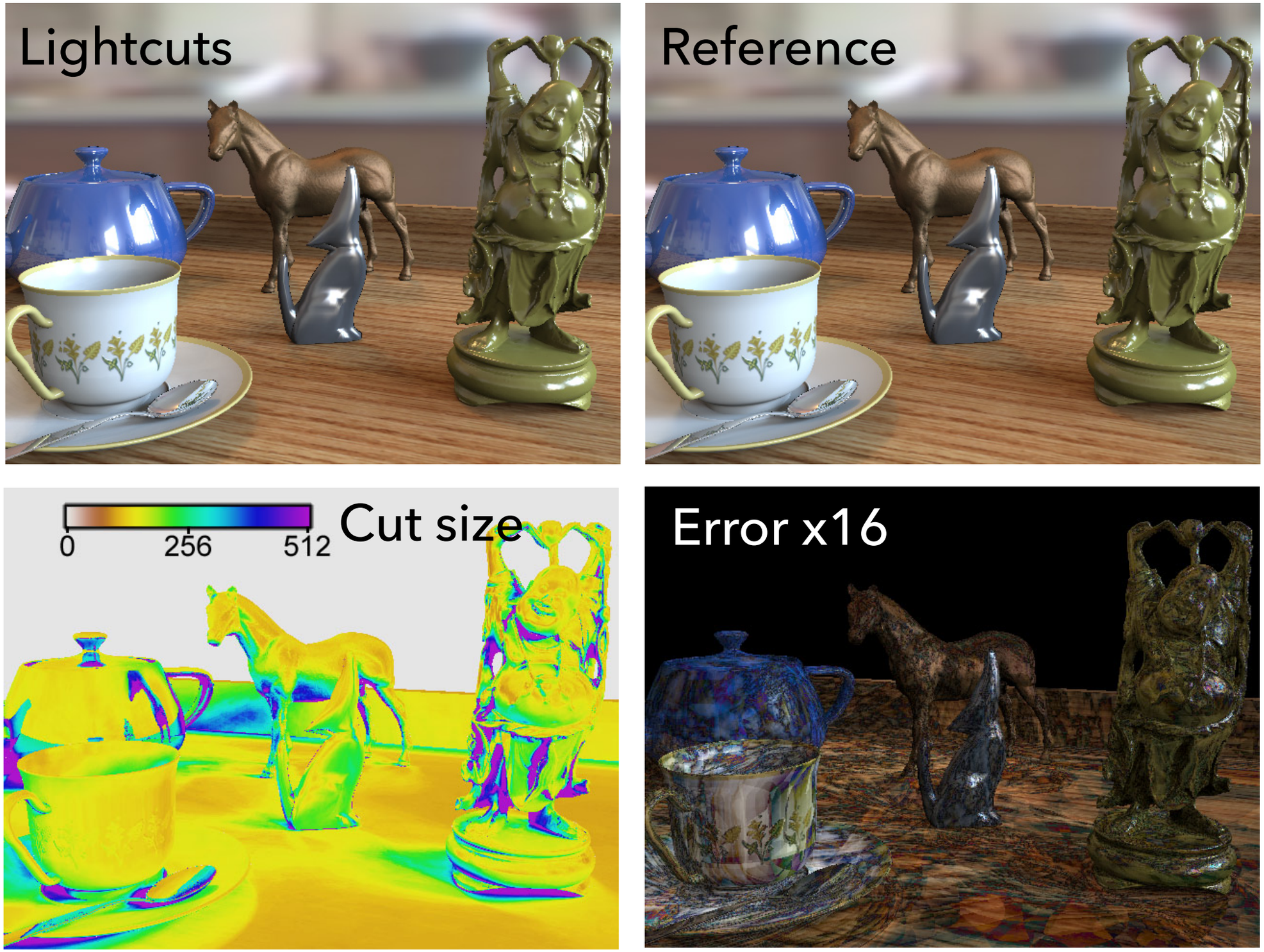


Lightcuts

Building a cut

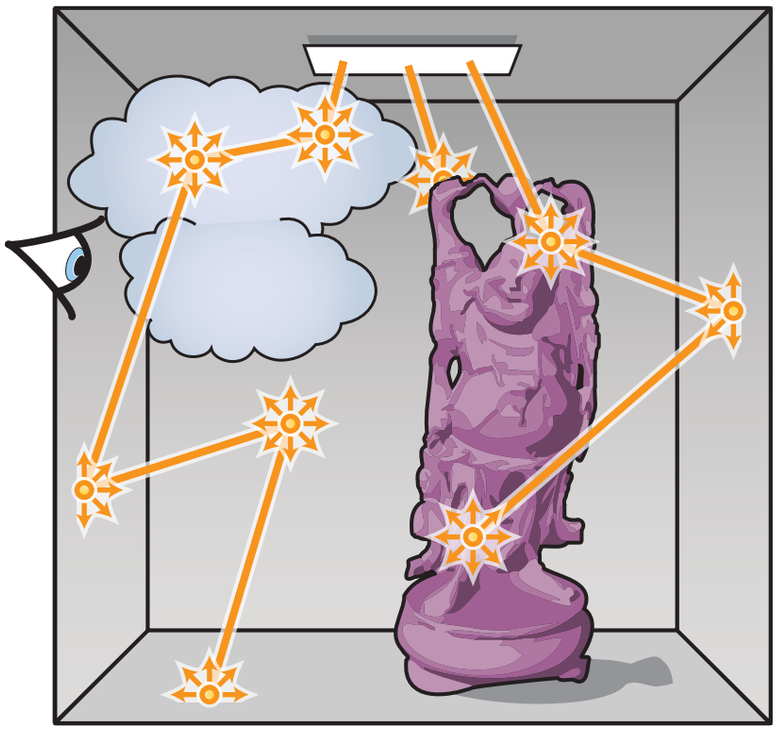


Lightcuts

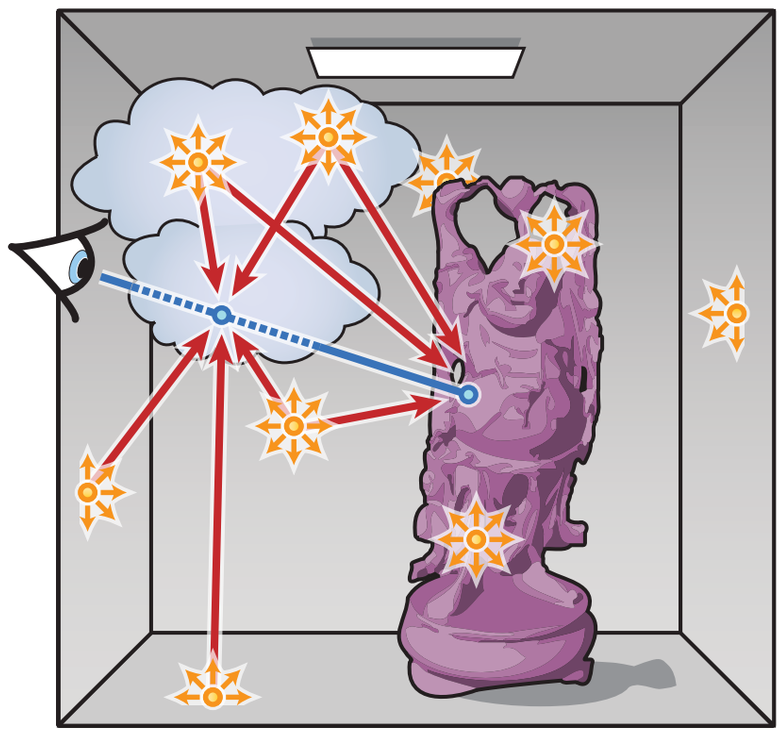


Many-Light Rendering

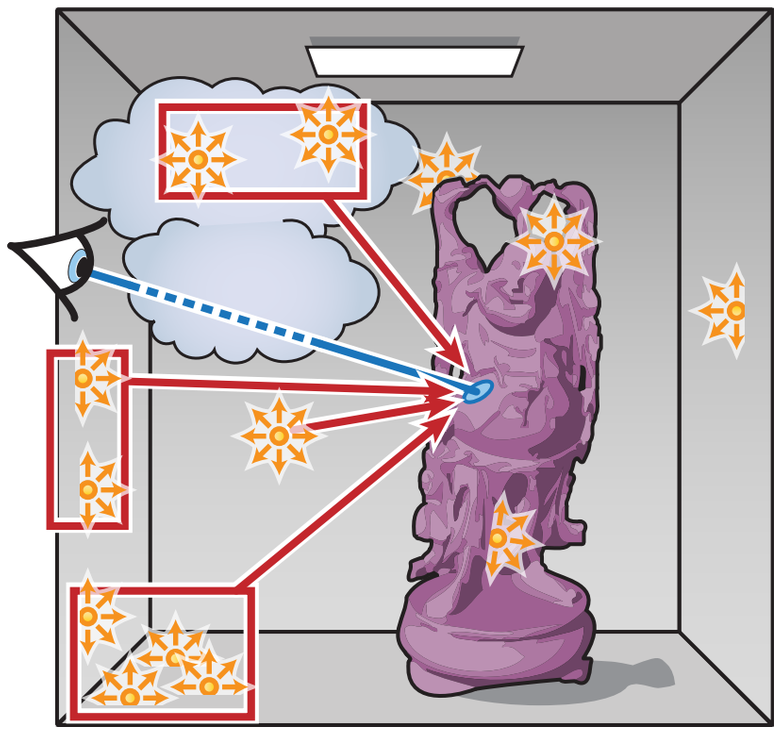
Generation of VPLs



Lighting with



Clustering of



Many-Light Rendering: Summary

- Advantages
  - basic method is easy to implement
  - great scalability
  - low amount of noise
  - GPU friendly, can be accelerated using shadow maps
  - popular in real-time, interactive, and fast offline rendering
- Disadvantages
  - horrible temporal stability shows up as flickering
  - hard to capture high-frequency transport (e.g. glossy reflections)

Reference	Abbreviated Paper Title	Category / Goal	Speed	Materials	Clustering	Remark
[Kel97]	Instant Radiosity	diffuse GI	real-time	D	no	basic method
[DS05]	Reflective Shadow Maps	single-bounce GI	real-time	D	no	no VPL visibility, importance sampling of VPLs
[DS06]	Splatting Indirect Illumination	single-bounce GI	real-time	D,(G)	no	no VPL visibility, importance sampling of VPLs
[LLK07]	Incremental Instant Radiosity	single-bounce GI	real-time	D	no	reuse VPLs over frames, for static scenes only
[NW09,10]	Multiresolution Splatting	single-bounce GI	real-time	D,(G)	yes	hierarchical shading, no VPL visibility
[DGRS09]	Clustered Visibility	reduce banding	real-time	D,(G)	k-means	reduce banding with virtual area lights
[PKD12]	Reflective Shadow Map Clustering	single-bounce, reduce banding	real-time	D	k-means	virtual area lights with no visibility computation
[KK04]	Illumination Presence of Weak Singularities	bias compensation	offline	D,G	no	bias compensation using final gathering
[NED11]	Screen-Space Bias Compensation	bias compensation	real-time	D,G	no	image-space technique for approximate bias compensation
[RSK08]	Unbiased GI with Participating Media	bias compensation	offline	PM	no	basic bias compensation technique for media
[ENSD12]	Approximate Bias Compensation	bias compensation	offline/interactive	PM	no	approximate bias compensation for participating media
[WKB12]	Bidirectional lightcuts	bias compensation	offline	D,G,SSS,PM	yes	short-range indirect illumination, uses Multidimensional Lightcuts
[HKWB09]	Virtual Spherical Lights	avoid singularities	offline	D,G	yes	inflate VPLs to spherical lights, uses MRCS for clustering
[DKH∗10]	Combining Global and Local Virtual Lights	reduce singularities	offline	D,G	yes	local lights for short-range indirect illumination
[NNDJ12b]	Virtual Ray Lights	reduce singularities	offline	D,G,PM	no	use light path segments as virtual ray lights
[NNDJ12a]	Progressive Virtual Beam Lights	avoid singularities	offline	D,G,PM	no	inflate virtual ray lights
[HPB07]	Matrix Row-Column Sampling	scalability	precompute	D,G	yes	compute a global clustering of VPLs
[HVAPB08]	Tensor Clustering for Many-Light Animations	scalability	precompute	D,G	yes	compute a global clustering of VPLs, support for animations
[OP11]	Lightslice: Matrix Slice Sampling	scalability	precompute	D,G	yes	localized/clustered cuts
[WFA∗05]	Lightcuts	scalability	offline	D,G	yes	per-shading point local cut of VPL hierarchy
[WABG06]	Multidimensional Lightcuts	scalability	offline	D,G,PM	yes	cut of VPLs and sensor points for depth of field, motion blur
[DGS12]	Progressive Lightcuts for GPU	scalability	offline/interactive	D,G	yes	progressive, GPU-friendly variant of [WFA*05]
[SIMP06a]	Bidirectional Instant Radiosity	VPL generation	real-time	D,G	no	bidirectional VPL generation
[SIP07]	Metropolis Instant Radiosity	VPL generation	real-time	D,G	no	bidirectional VPL generation with Metropolis sampling
[GS10]	Iterative Importance Sampling of VPLs	VPL generation	real-time	D,G	no	rejection sampling of VPL paths
[GKPS12]	Importance Caching for Complex Illumination	VPL selection	offline	D,G	no	importance caching of VPL contributions and improved selection
[RGK∗08]	Imperfect Shadow Maps	visibility	real-time	D,G	no	fast approximate shadow maps for VPLs
[REH∗11]	Making ISMs View-Adaptive	visibility	real-time	D,G	no	extension of [RGK*08], view-adaptive ISMs and VPL placement
[REG∗09]	Micro-Rendering	visibility	interactive	D,G	dna	compute shadow maps for VPLs or final-gather from VPLs
[HREB11]	ManyLODs	visibility	real-time	D,G	dna	fast many-view rasterization
[PGSD13]	Adaptive Quantization Visibility Caching	visibility	offline/interactive	D,G	dna	general visibility cache

Autodesk 360