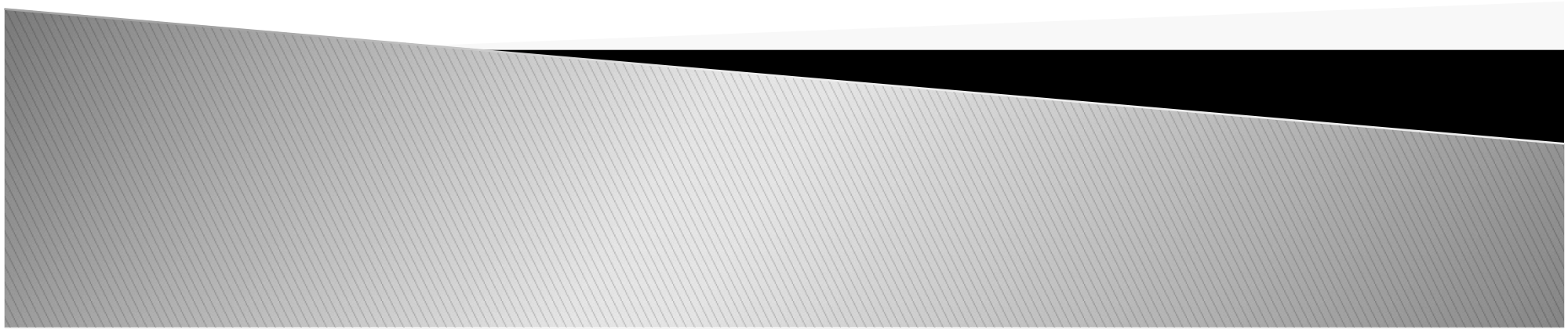
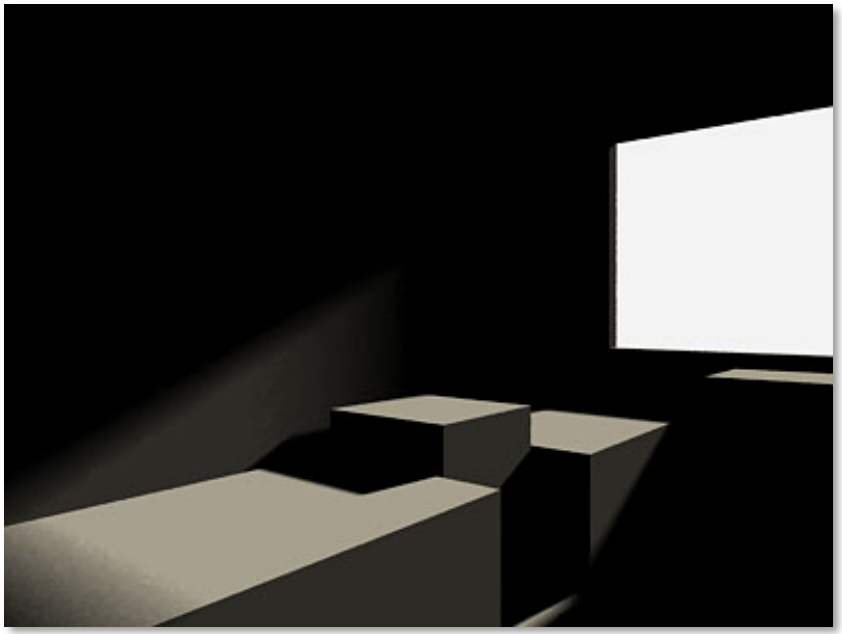
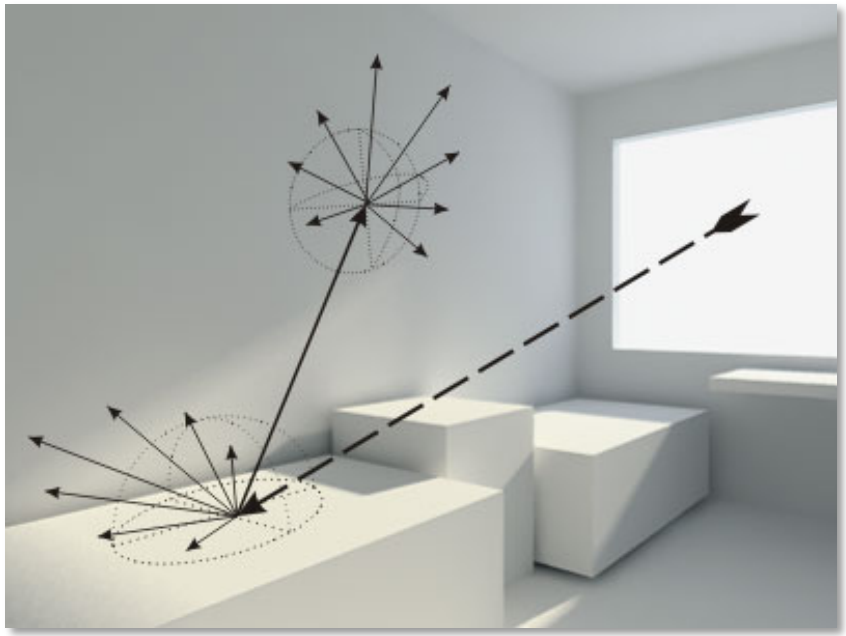
Global Illumination



Direct and indirect lighting

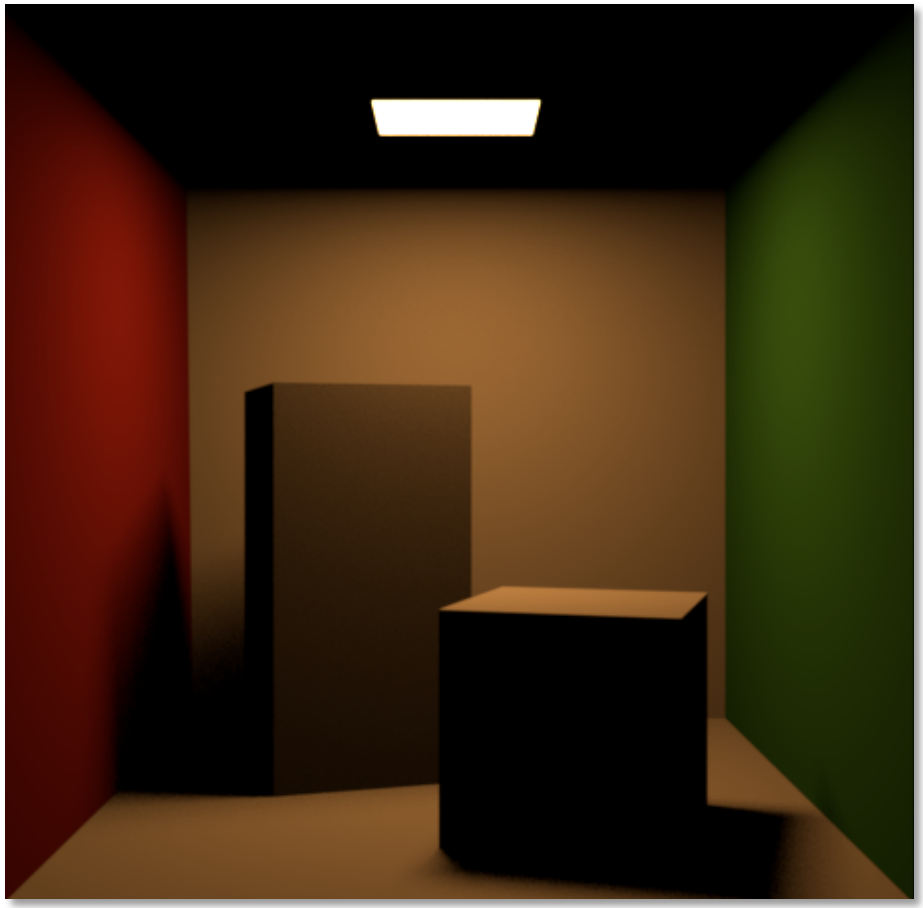


Direct :
local properties

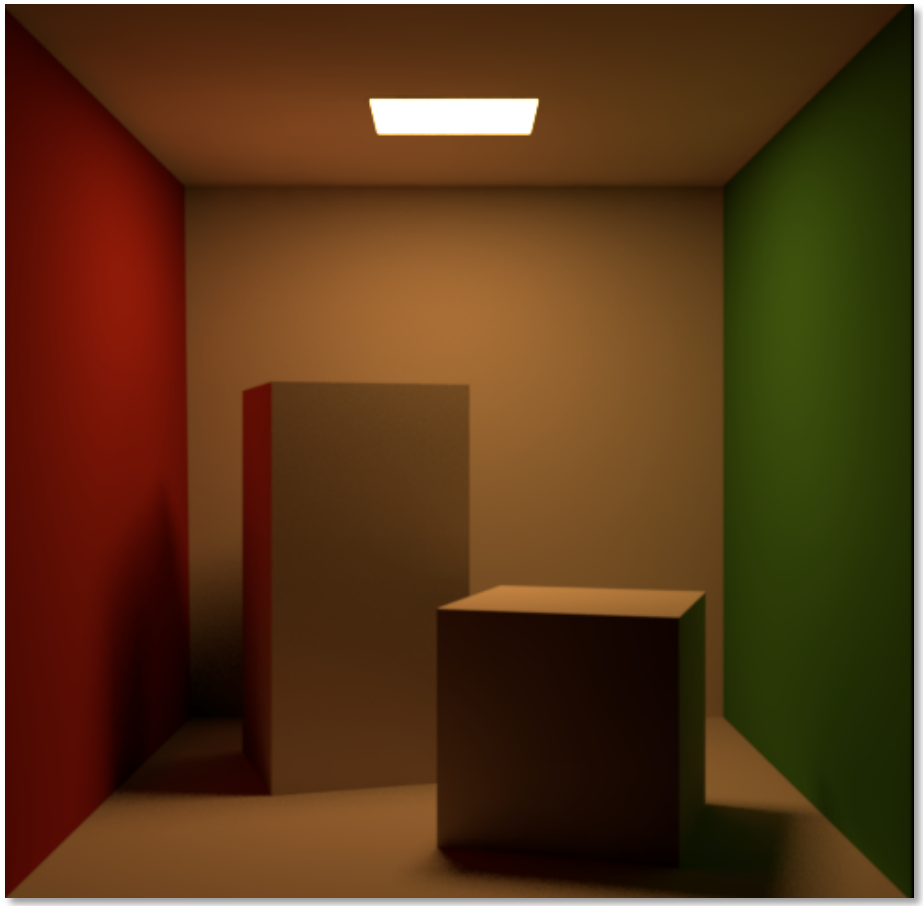


Indirect :
global problem

Direct and indirect lighting

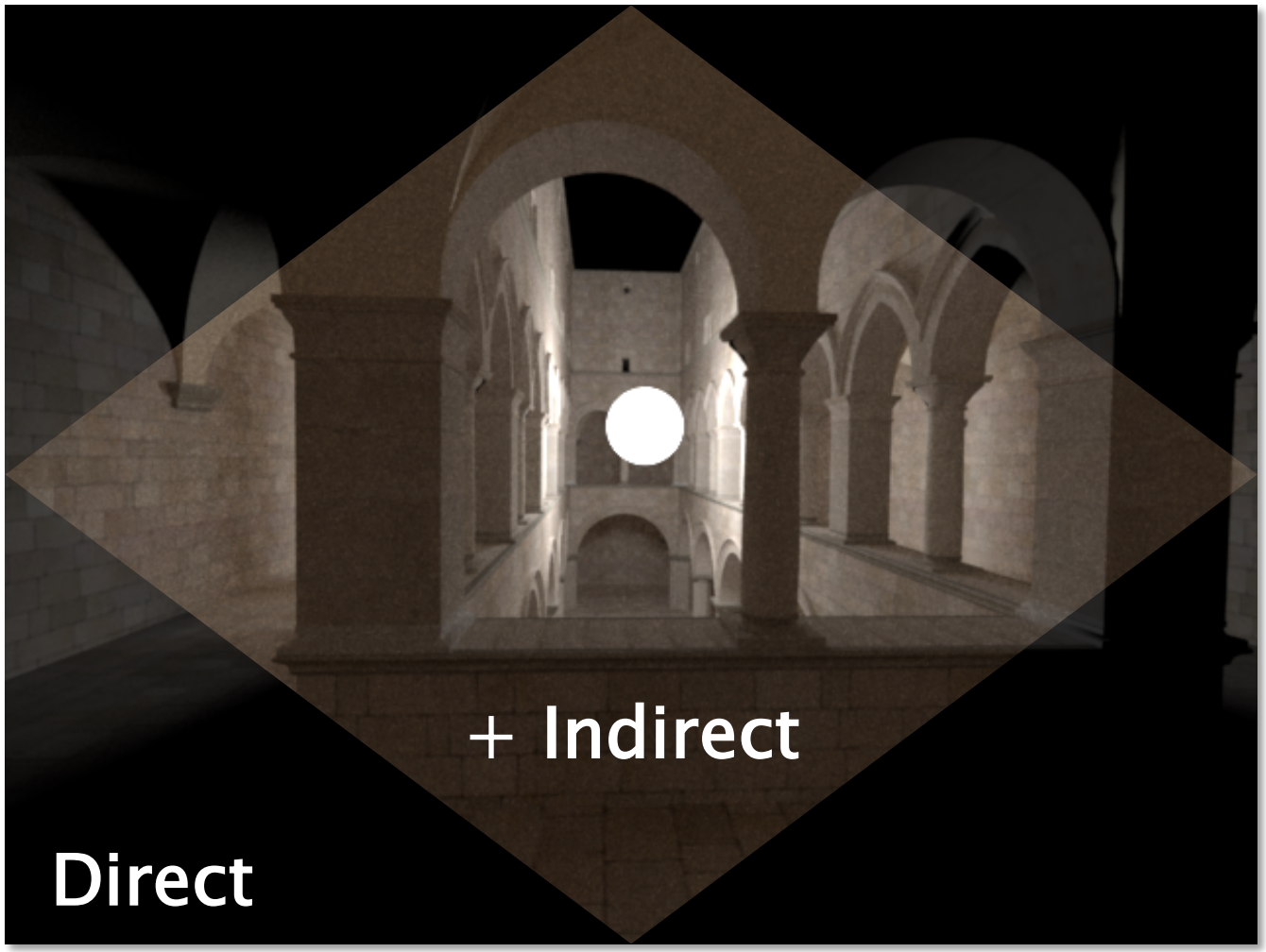


Direct



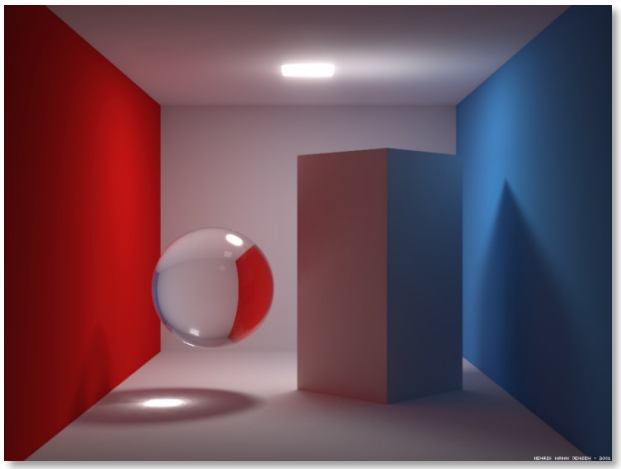
+ Indirect

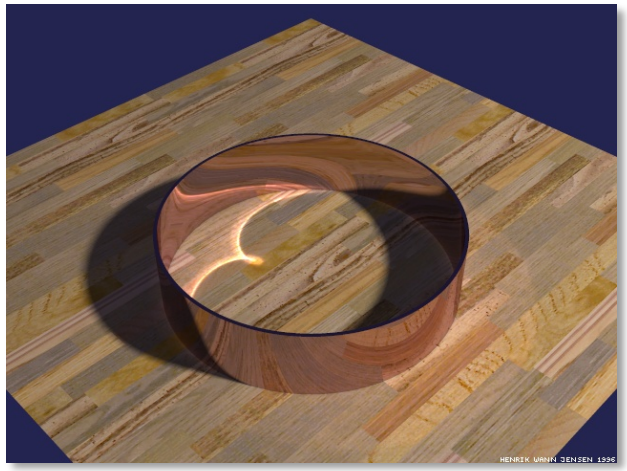
Direct and indirect lighting



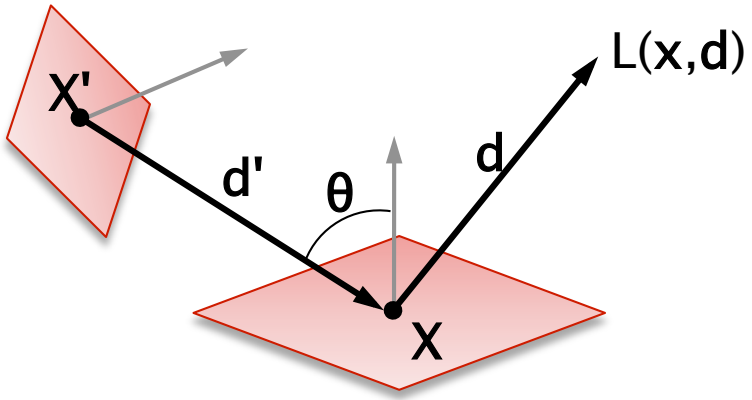
Global illumination

- Interactions between objects
- Light transport
- Reflections, refraction, diffusion
- Conservation of light energy





The rendering equation

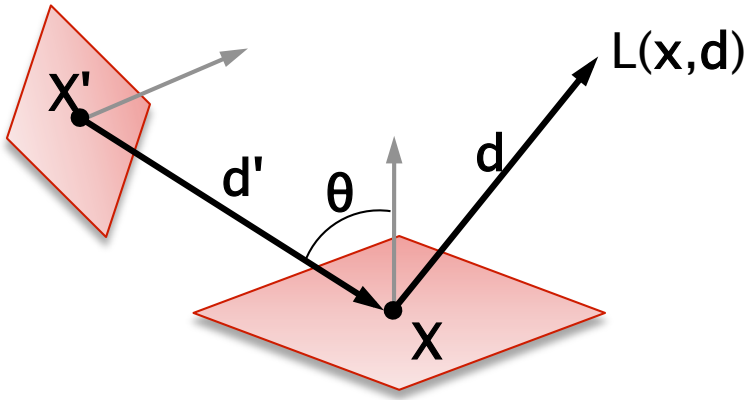


Radiance unit: Watt/m²/sr

$$\underbrace{L(x,d)}_{} = E(x,d) + \int \rho(x,d,d') \, v(x,x') \, L(x',d') \, G(x,x') \, dA$$

Radiance leaving point **x**
in the direction **d**

The rendering equation



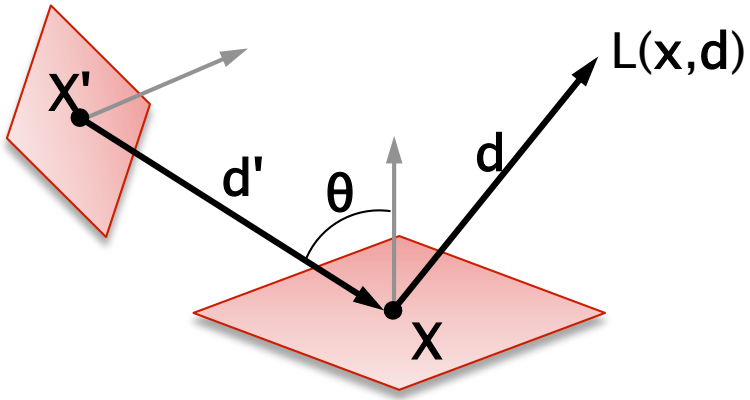
Radiance unit: Watt/m²/sr

$$L(x,d) = E(x,d) + \int \rho(x,d,d') \, v(x,x') \, L(x',d') \, G(x,x') \, dA$$

Radiance emitted from **x**:

non-zero only if **x** belongs to a light source

The rendering equation

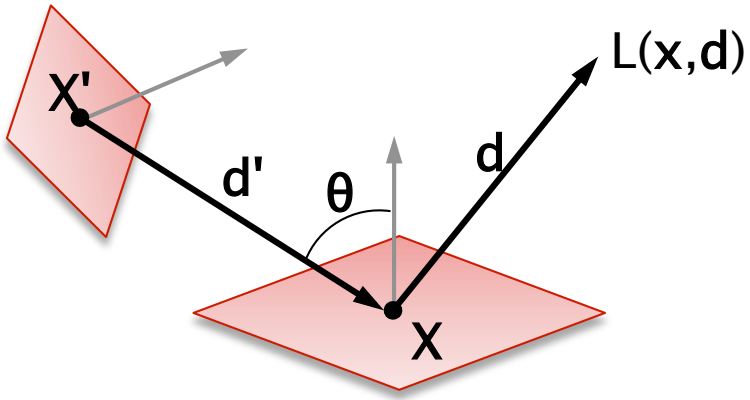


Radiance unit: Watt/m²/sr

$$L(\mathbf{x},\mathbf{d}) = E(\mathbf{x},\mathbf{d}) + \int \rho(\mathbf{x},\mathbf{d},\mathbf{d'})\, v(\mathbf{x},\mathbf{x'})\, L(\mathbf{x'},\mathbf{d'})\, G(\mathbf{x},\mathbf{x'})\, dA$$

Integrating the contribution from all the surfaces

The rendering equation

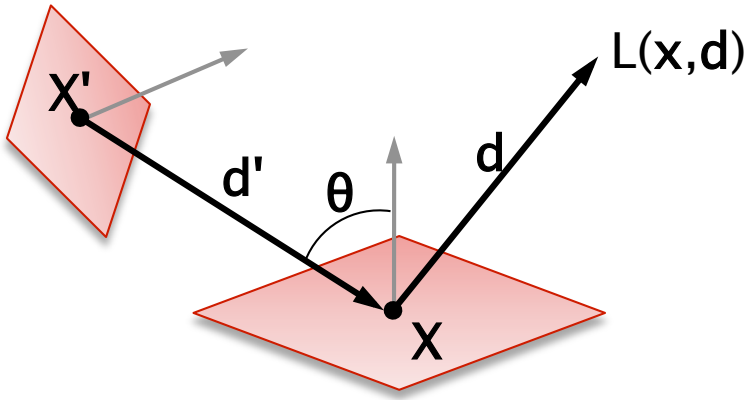


Radiance unit: Watt/m²/sr

$$L(\mathbf{x},\mathbf{d}) = E(\mathbf{x},\mathbf{d}) + \int \rho(\mathbf{x},\mathbf{d},\mathbf{d'})\, v(\mathbf{x},\mathbf{x'})\, \underbrace{L(\mathbf{x'},\mathbf{d'})}\, G(\mathbf{x},\mathbf{x'})\, dA$$

Incoming Radiance
from point **x'** in the direction **d'**

The rendering equation

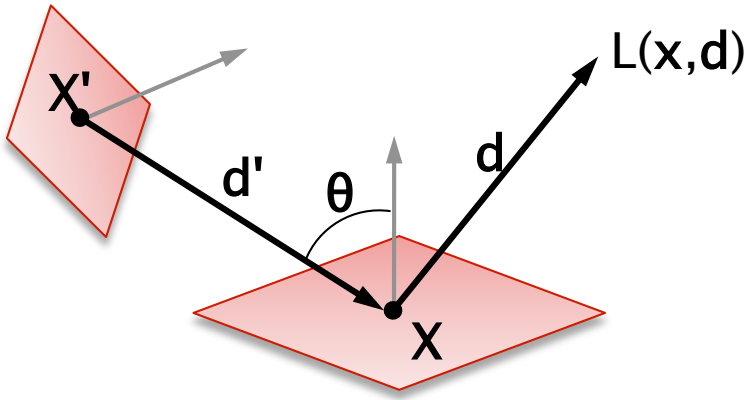


Radiance unit: Watt/m²/sr

$$L(\mathbf{x},\mathbf{d}) = E(\mathbf{x},\mathbf{d}) + \int \rho(\mathbf{x},\mathbf{d},\mathbf{d'})\, v(\mathbf{x},\mathbf{x'})\, L(\mathbf{x'},\mathbf{d'})\, G(\mathbf{x},\mathbf{x'})\, dA$$

Multiplication by the reflectance (BRDF)
of the surface at point **x**

The rendering equation



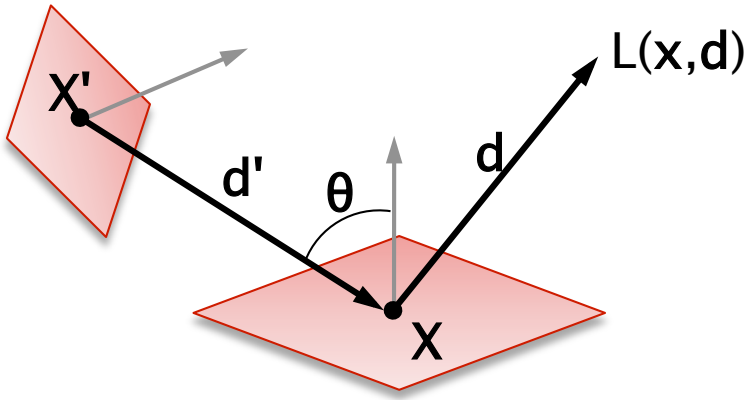
Radiance unit: Watt/m²/sr

$$L(\mathbf{x},\mathbf{d}) = E(\mathbf{x},\mathbf{d}) + \int \rho(\mathbf{x},\mathbf{d},\mathbf{d'})\, v(\mathbf{x},\mathbf{x'})\, L(\mathbf{x'},\mathbf{d'})\, G(\mathbf{x},\mathbf{x'})\, dA$$

Visibility between **x** and **x'**

1 when the two points are visible from each other, 0 otherwise

The rendering equation

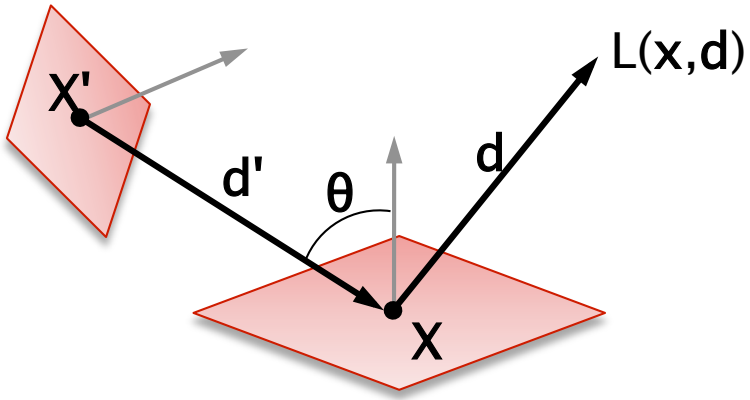


Radiance unit: Watt/m²/sr

$$L(\mathbf{x},\mathbf{d}) = E(\mathbf{x},\mathbf{d}) + \int \rho(\mathbf{x},\mathbf{d},\mathbf{d'})\, v(\mathbf{x},\mathbf{x'})\, L(\mathbf{x'},\mathbf{d'})\, G(\mathbf{x},\mathbf{x'})\, dA$$

Geometric factor depending on the surfaces x and x'

The rendering equation



Radiance unit: Watt/m²/sr

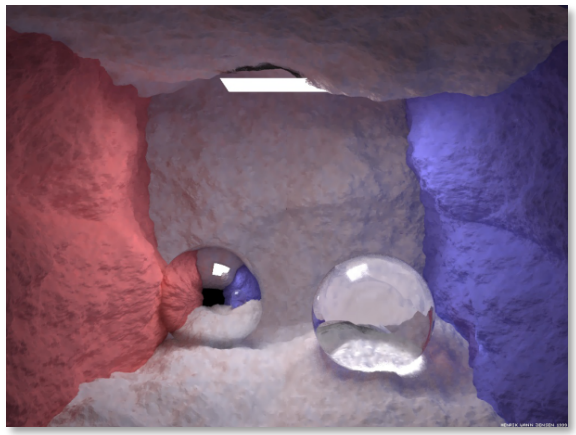
$$L(\mathbf{x},\mathbf{d}) = E(\mathbf{x},\mathbf{d}) + \int \rho(\mathbf{x},\mathbf{d},\mathbf{d'})\, v(\mathbf{x},\mathbf{x'})\, L(\mathbf{x'},\mathbf{d'})\, G(\mathbf{x},\mathbf{x'})\, dA$$

Full general analytical solution impossible

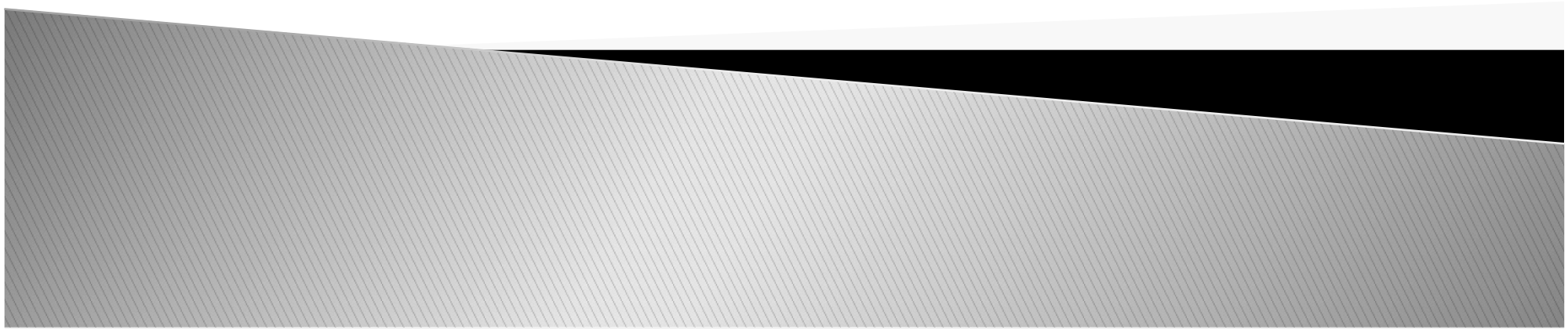
Two discretisations

- **Radiosity**
  - Discretize the geometry: exchanges between patches
  - All objects are diffuse

- **Ray-tracing** and its extensions (Monte-Carlo path tracing, Photon mapping…)
  - Sampling the integral
  - Optical laws

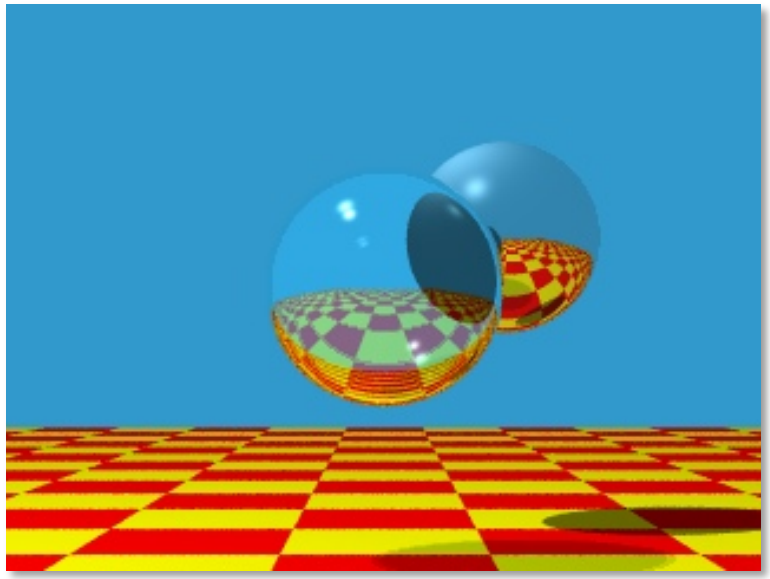


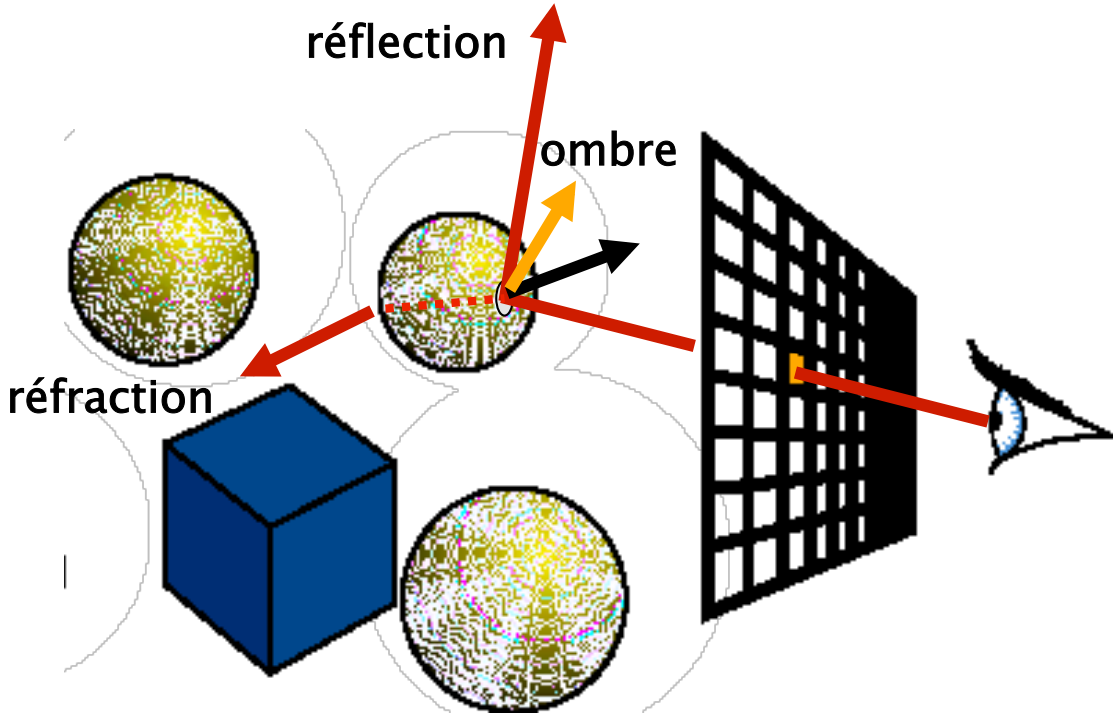
Ray tracing



« Whitted Ray Tracing » (1980)

- One ray per pixel
- Three new rays are created



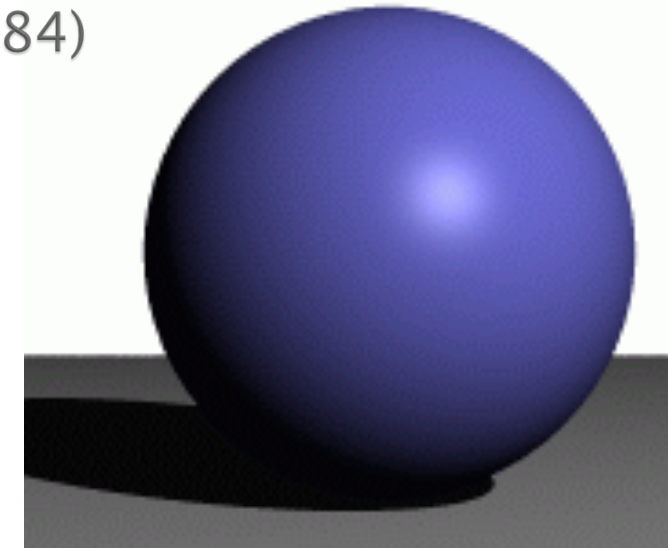


Even more rays

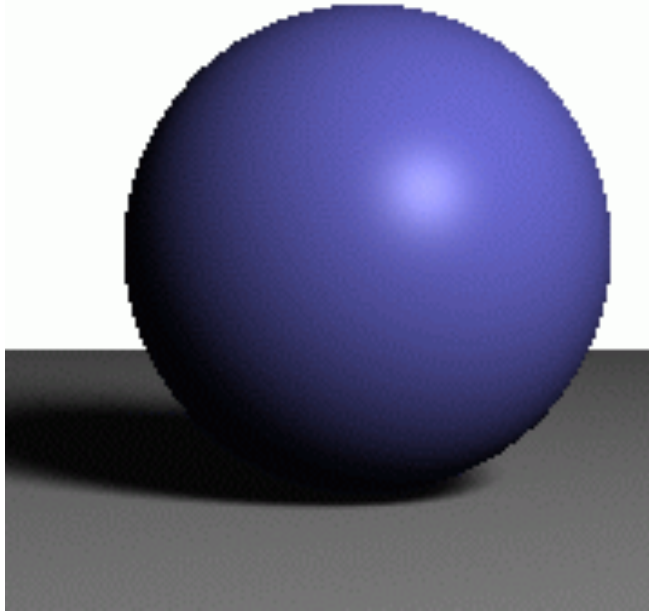
« Distributed Ray Tracing » Cook et al. (1984)

- Soft shadows
  - Several rays for each extended light source

Point light source



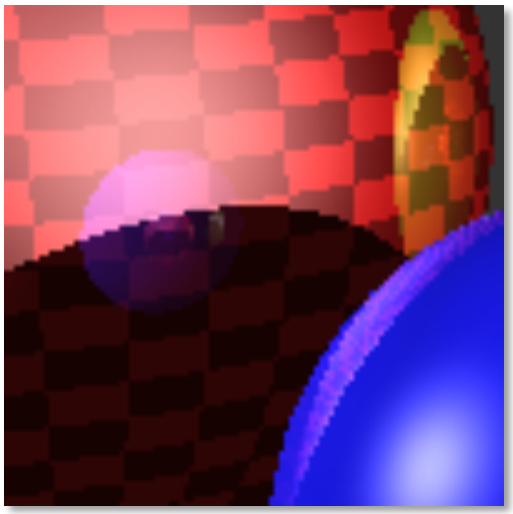
Extended light source



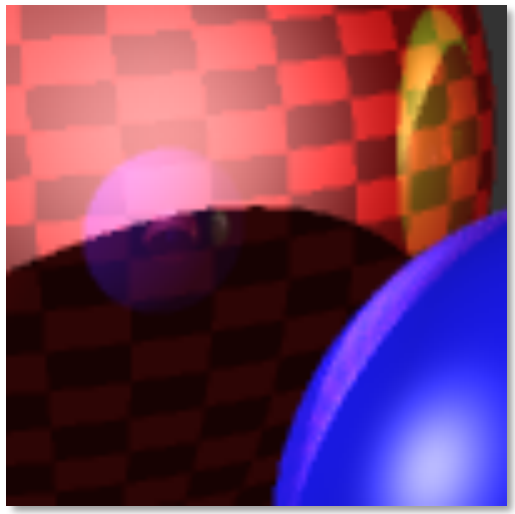
Even more rays

« Distributed Ray Tracing » Cook et al. (1984)

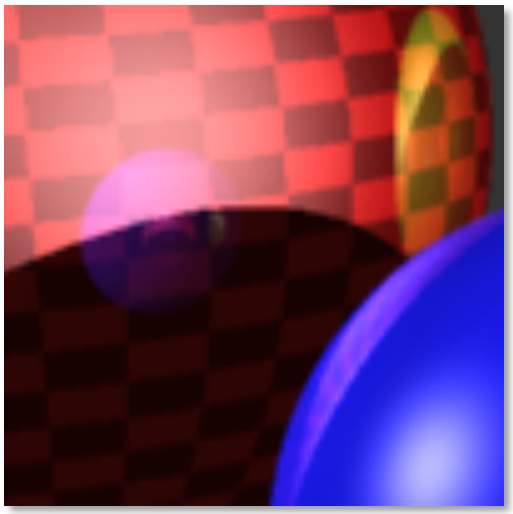
- Soft shadows
  - Several rays for each extended light source
- Anti-aliasing
  - Several rays for each pixel



1 rayon



2 rayons

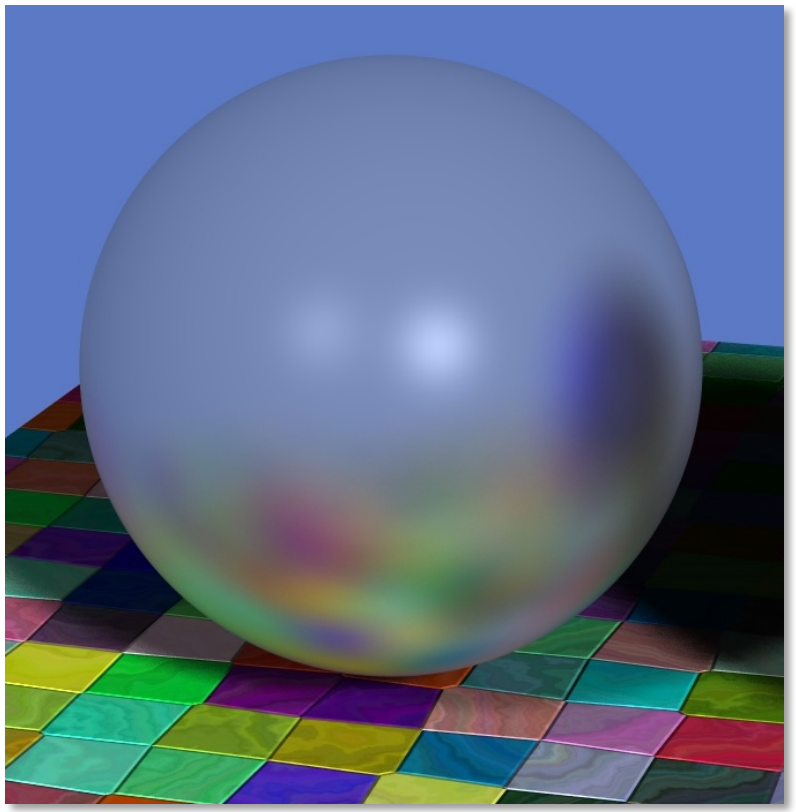


3 rayons

Even more rays

« Distributed Ray Tracing » Cook et al. (1984)

- Soft shadows
  - Several rays for each extended light source
- Anti-aliasing
  - Several rays for each pixel
- Glossy Reflection
  - Several rays are reflected



Even more rays

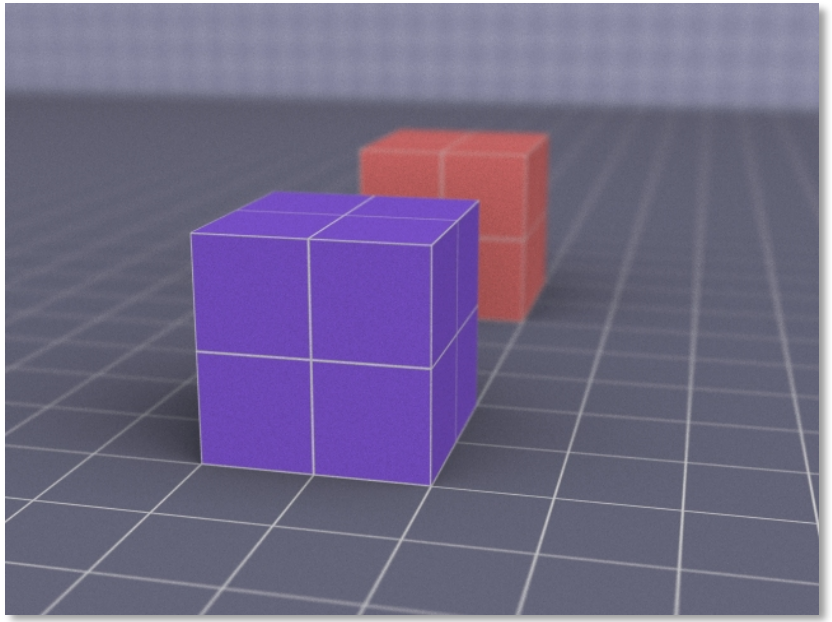
« Distributed ray tracing » Cook et al. (1984)

- Soft shadows
  - Several rays for each extended light source
- Anti-aliasing
  - Several rays for each pixel
- Glossy Reflection
  - Several rays are reflected
- Motion blur
  - Several rays during time

Even more rays

« Distributed Ray Tracing » Cook et al. (1984)

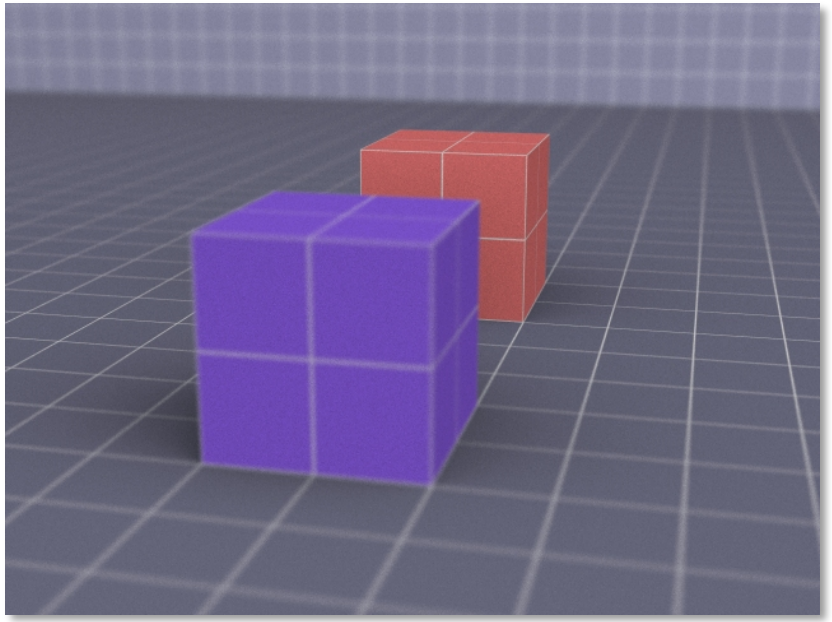
- Soft shadows
  - Several rays for each extended light source
- Anti-aliasing
  - Several rays for each pixel
- Glossy Reflection
  - Several rays are reflected
- Motion blur
  - Several rays during time
- Depth of field
  - Several rays per pixel, focusing with a lens



Even more rays

« Distributed Ray Tracing » Cook et al. (1984)

- Soft shadows
  - Several rays for each extended light source
- Anti-aliasing
  - Several rays for each pixel
- Glossy Reflection
  - Several rays are reflected
- Motion blur
  - Several rays during time
- Depth of field
  - Several rays per pixel, focusing with a lens



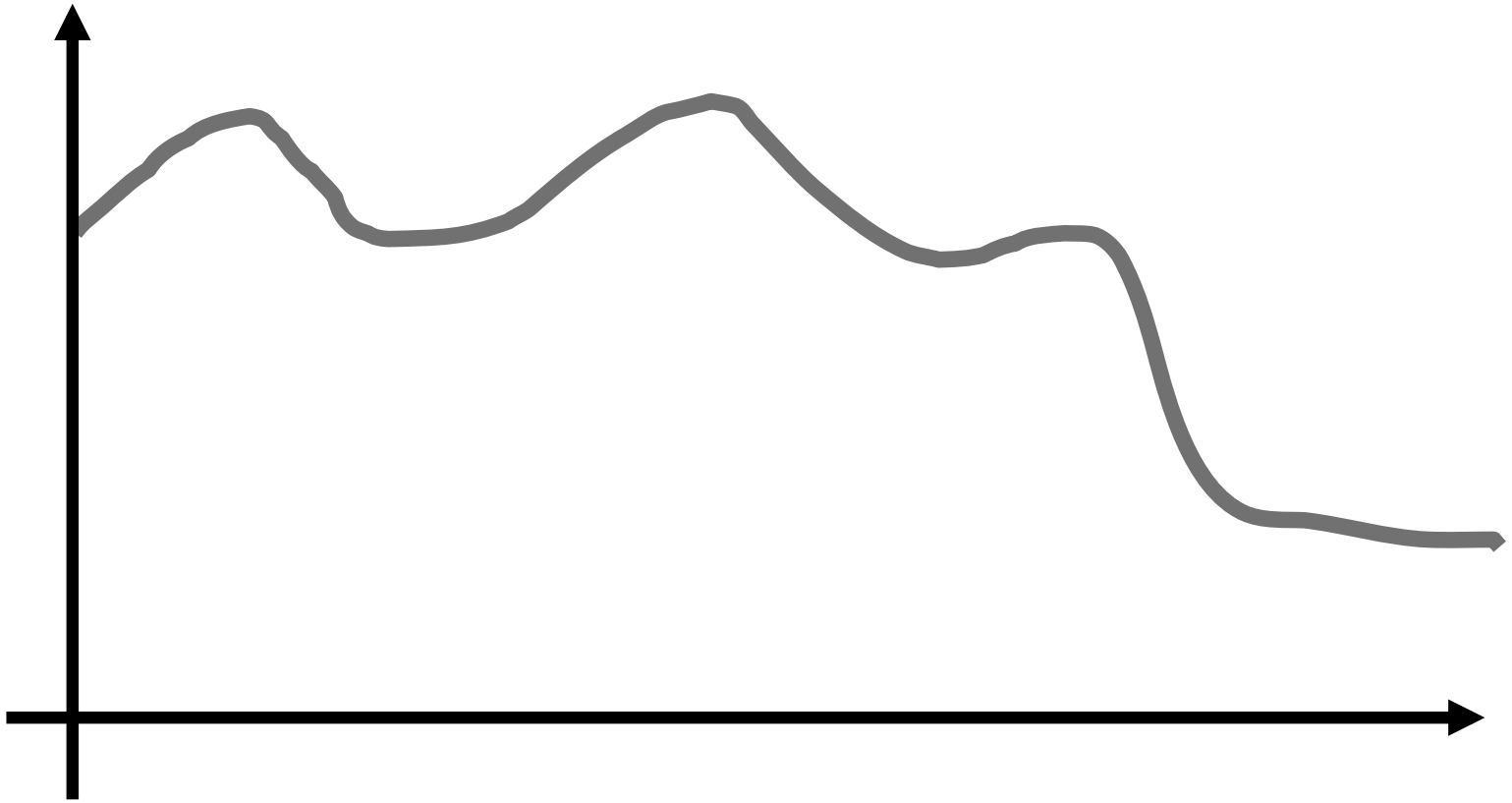
Ray tracing = integrating!

- Integrating what?
  - **light sources**: soft shadows
  - **pixels**: anti-aliasing
  - **BRDF**: glossy reflections
  - over **Time**: motion blur
  - over the **lens**: depth of field
  - over the **hemisphere**: indirect lighting
  - over **light paths**: global illumination
- Generic method for computing multi-dimensional integrals:

Monte Carlo Integration

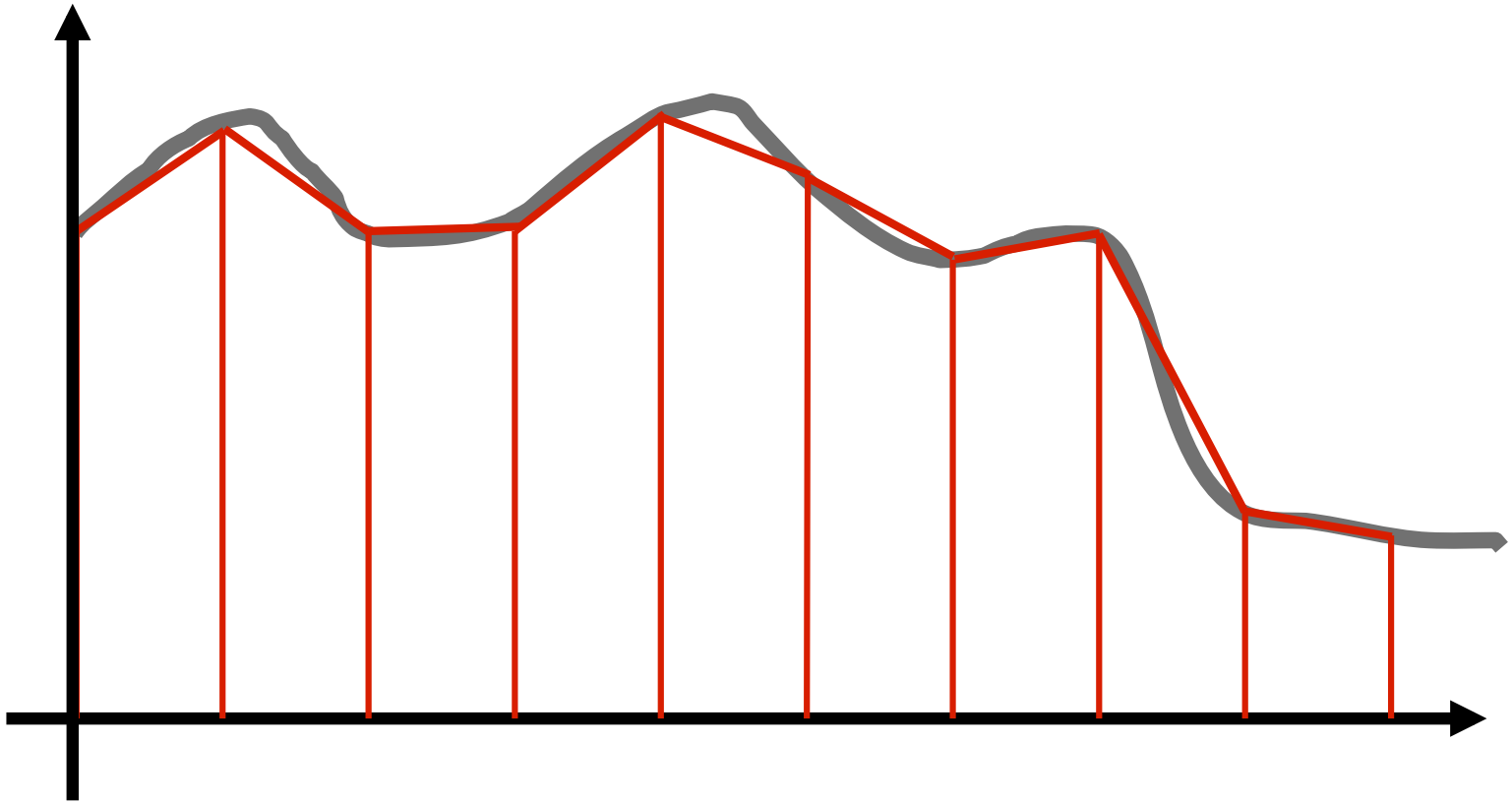
1D Integrals

- Integral of an arbitrary function
- Continuous problem ⇨ discretization



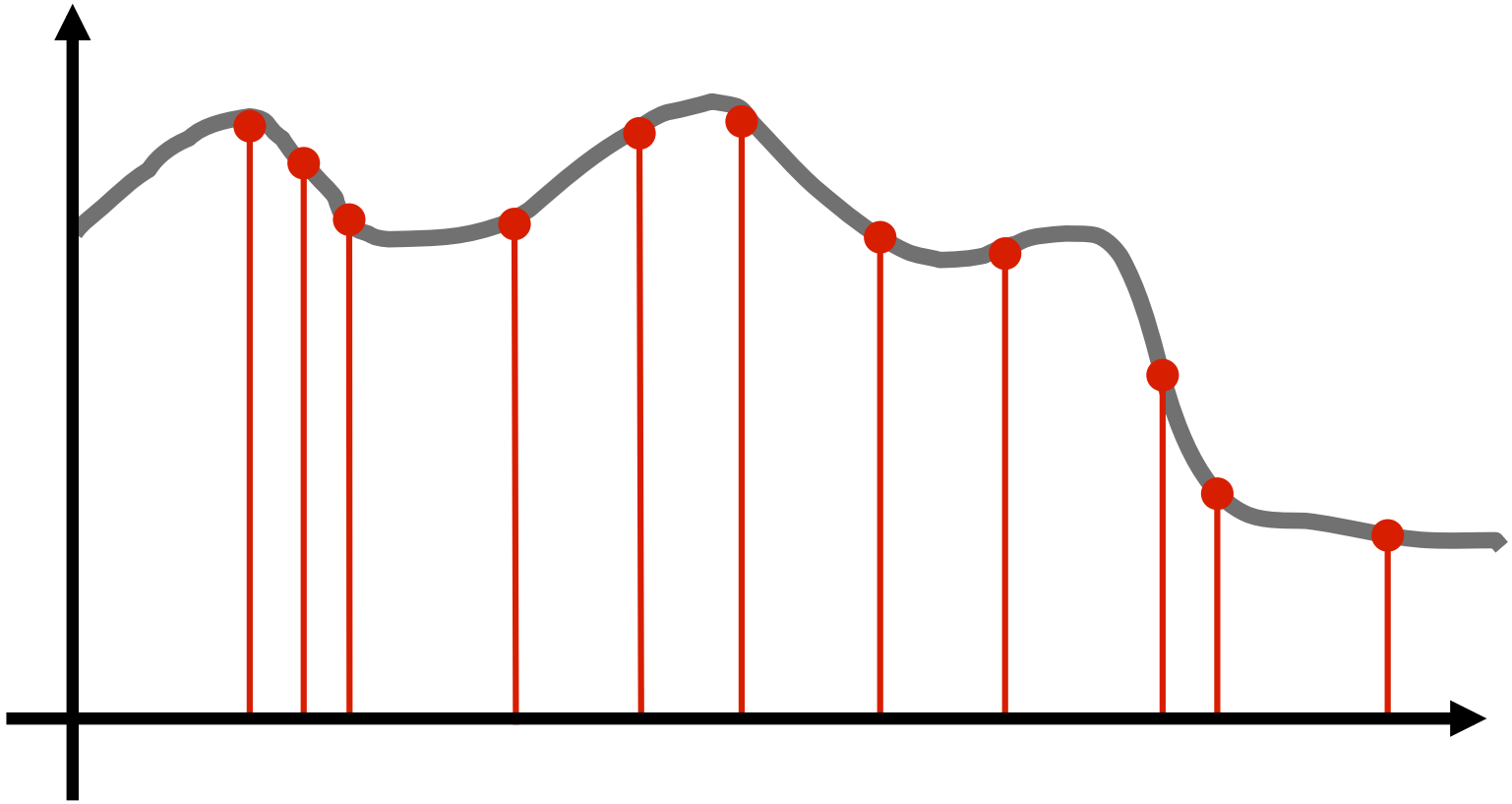
1D Integrals

- Trapezoidal approximation:
  - Also Simpson's rule, midpoint rule...



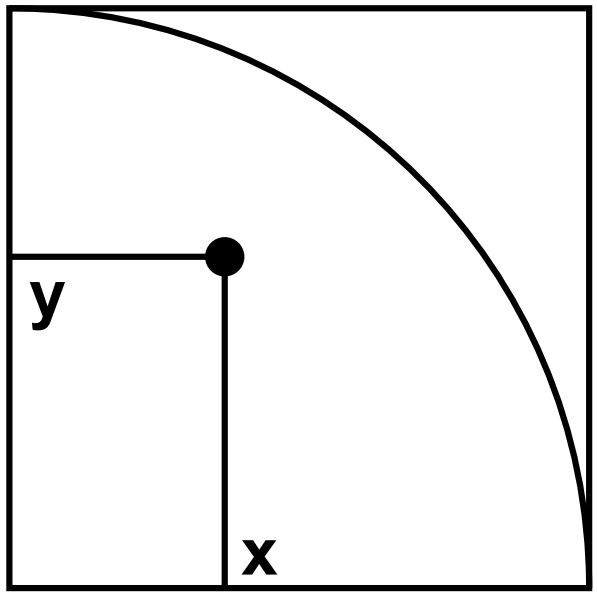
1D Integrals

- Monte Carlo: random sampling
  - Don't keep the distance between the n samples
  - But on average, expect it to be 1/n



Monte Carlo: computing π

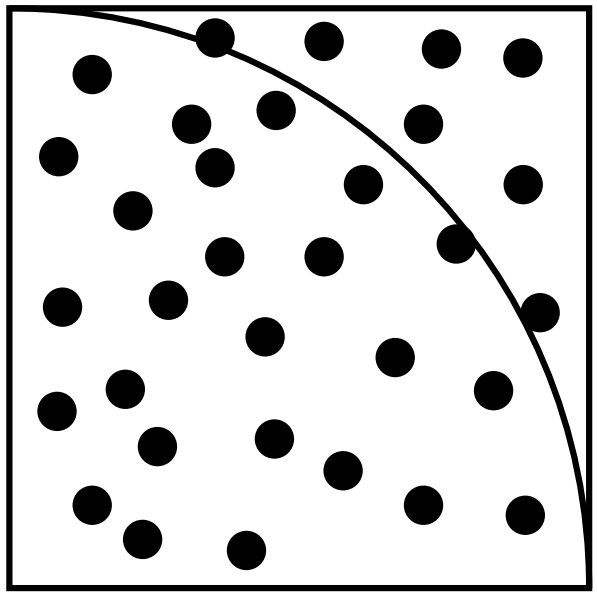
- Take a square
- Take a random point (x,y) in the square
- Test whether it is inside the ¼ disc ($x^2+y^2 < 1$)
- Probability is π /4



Integral of the function equal to 1 on the disc, 0 outside

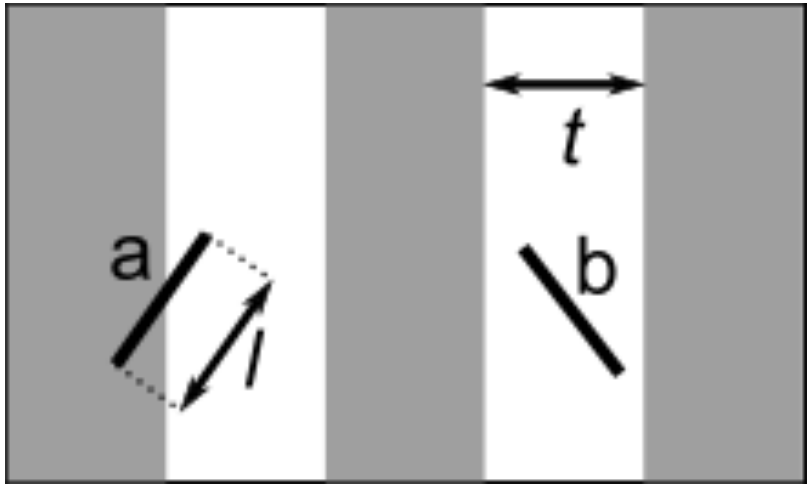
Monte Carlo: computing π

- Probability is π /4
- n = # points inside / # total points
- $\pi \approx$ n * 4
- Error depends on the number of samples



See also: Buffon's needle

- Floor made of parallel strips of wood
- Throw needles on the floor
  - The needle either crosses the line or it doesn't
  - Count number of time it crosses the line
  - Divide by total number of throws
- Result is connected to π
- $P = 2l/t\pi$



Why not use Trapezoidal rule?

- To compute π, Monte Carlo is not highly efficient
- But convergence rate independent from dimension
- ⇨ Monte Carlo integration very efficient for higher dimensions

Continuous random variables

- Random variable **x**
- Probability distribution: **p(x)**
  - Probability that this variable is between x and x+dx is p(x) dx

Expected value (mean)

$$E[x] = \int_{-\infty}^{\infty} xp(x)dx$$

$$E[f(x)] = \int_{-\infty}^{\infty} f(x)p(x)dx$$

- Expected value is linear:

  $E[f_1(x) + a\, f_2(x)] = E[f_1(x)] + a\, E[f_2(x)]$

Monte Carlo integration

- Take the function f(x) with x in [a b]
- We want to compute: $$I = \int_a^b f(x)dx$$
- Take a random variable **x**
- If **x** has a uniform distribution, **I=E[f(x)]**
  - By definition of expected value

Sum of random variables

- Take **N** random variables, independent, identical distribution (IID) x_i (N échantillons)
  - Same probability (here uniform)
- Define:

$$F_N = \frac{1}{N} \sum_{j=1}^{n} f(x_i)$$

Monte Carlo estimator

- By linearity of expected value:
  $E[F_N] = E[f(x)]$

Variance

$$\sigma^2 = E[(x - E[x])^2] = \int_{-\infty}^{\infty} (x - E[x])^2 p(x) dx$$

- Measures the distance to expected value
- Standard deviation σ: square root of variance
- Properties:
  - $\sigma^2[x+y] = \sigma^2[x] + \sigma^2[y] + 2\,\mathrm{Cov}[x,y]$
  - $\sigma^2[ax] = a^2\,\sigma^2[x]$

About the variance

$$\sigma^2[F_N] = \sigma^2 \left[\sum_{j=1}^{n} \frac{f(x_i)}{N} \right]$$

- Independent variables ⇨ Cov[x_i, x_j]=0 si i≠j

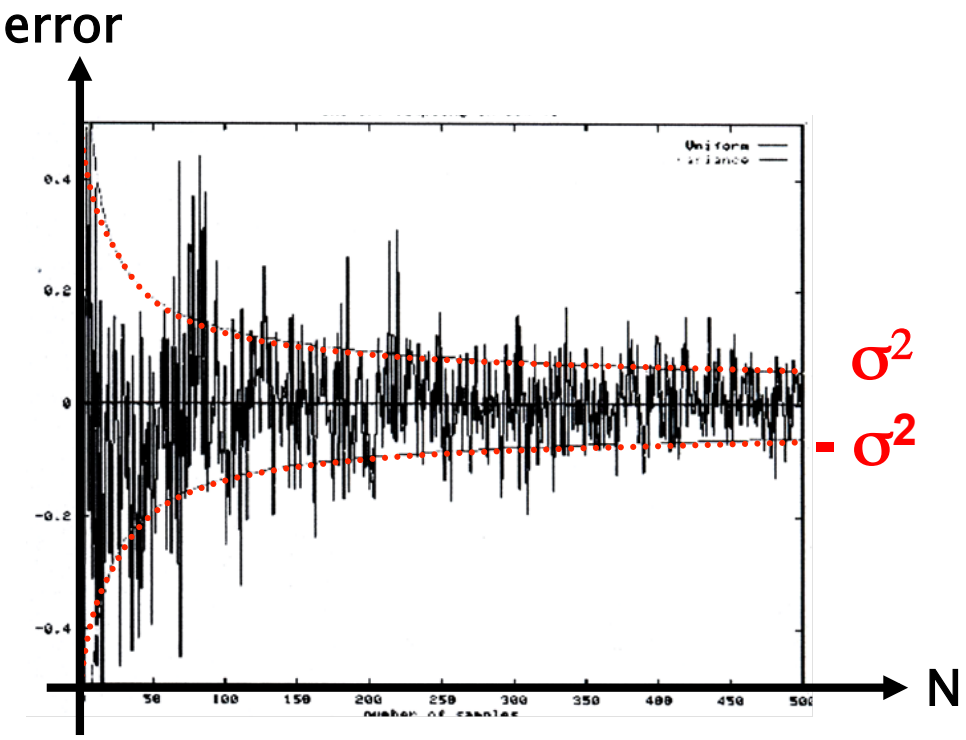
$$\sigma^2[F_N] = \frac{\sigma^2[f(x)]}{N}$$

- thus σ (error) decreases with $\sqrt{N}$
- ⇨ **slow convergence**

Example

$$I = \int_0^1 5x^4 dx$$

- In theory, I=1.0
- In practice, with a uniform distribution



Monte Carlo integration: pros

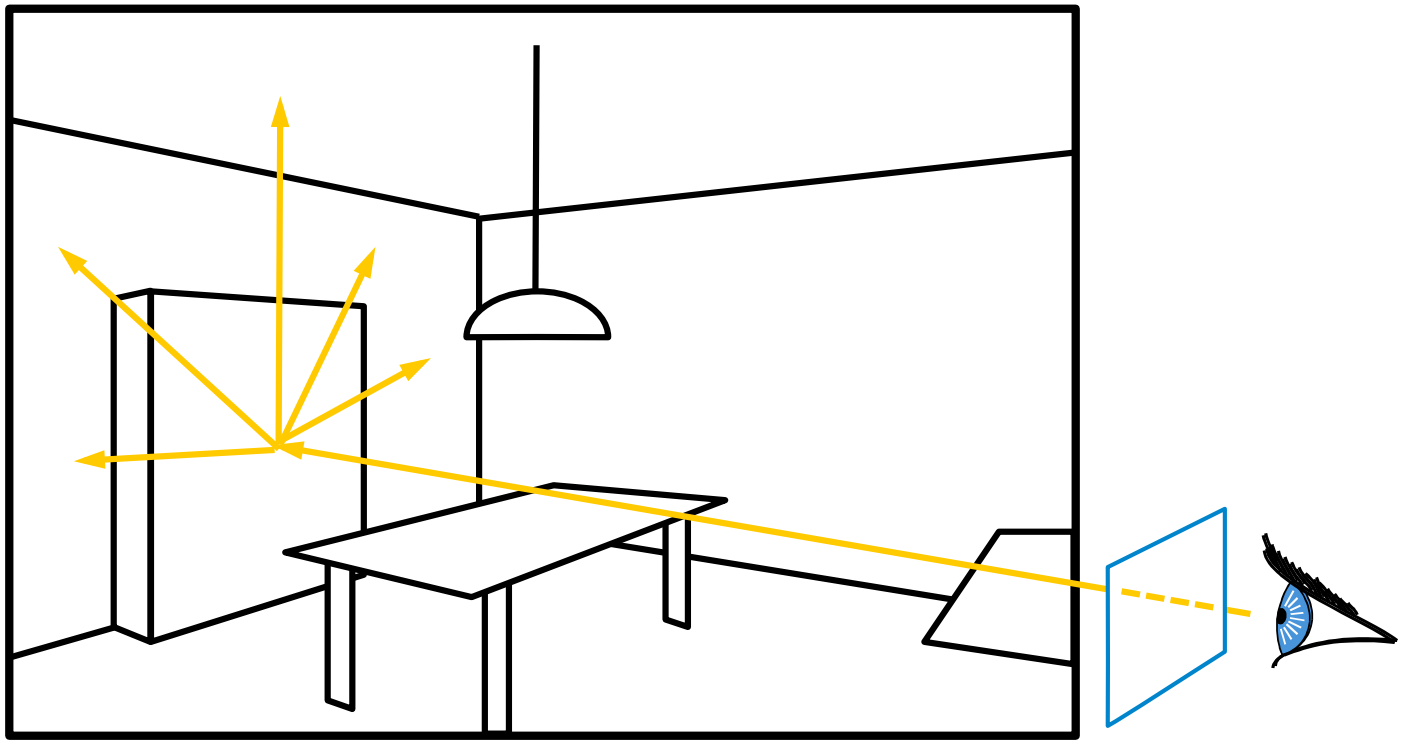
- Few restrictions on the function to integrate
  - No requirements on continuity, regularity …
  - Only needs sampling on a single point
- Same convergence rate on higher dimensions
- Very simple

Monte Carlo integration: pros

- Noise
- Slow convergence
- Efficient implementation harder

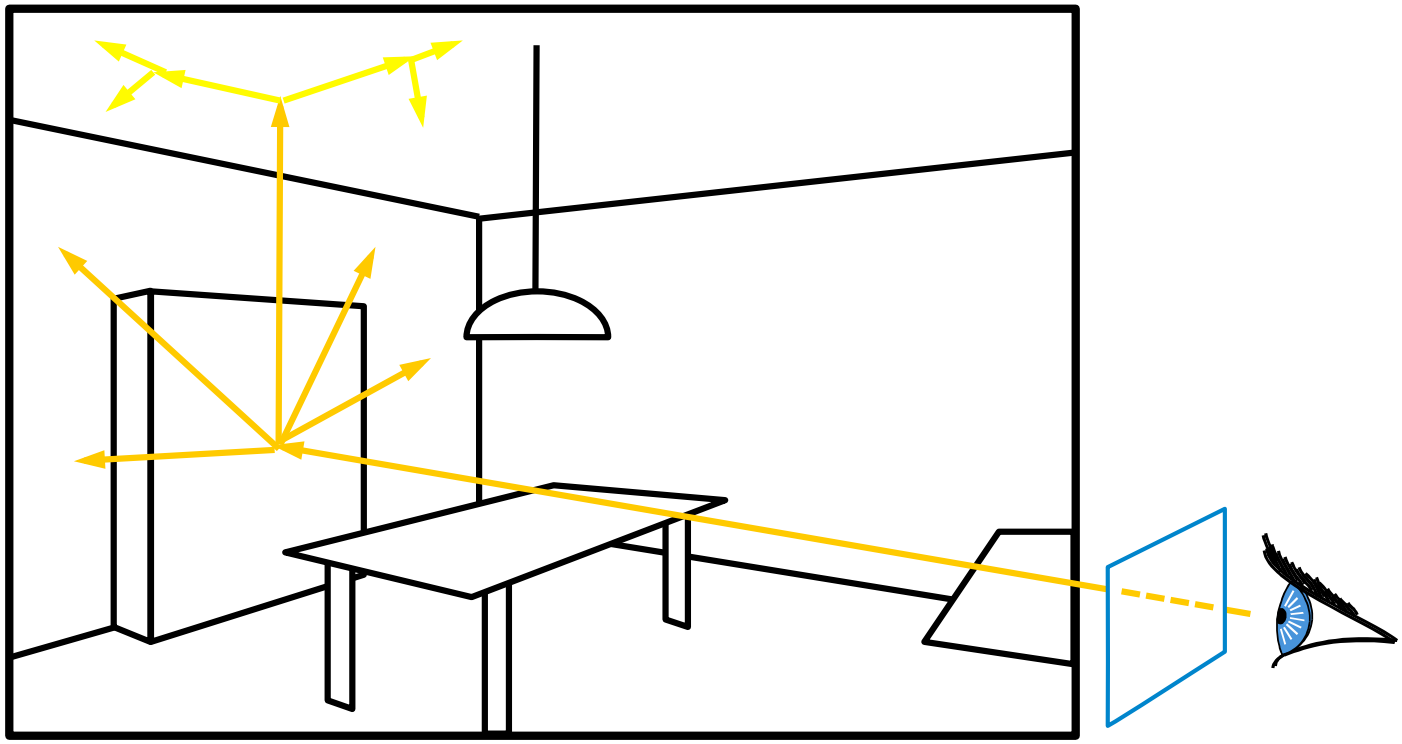
Monte Carlo methods

- On ray for every pixel
- For each visible point : random sampling of rays, accumulate radiance



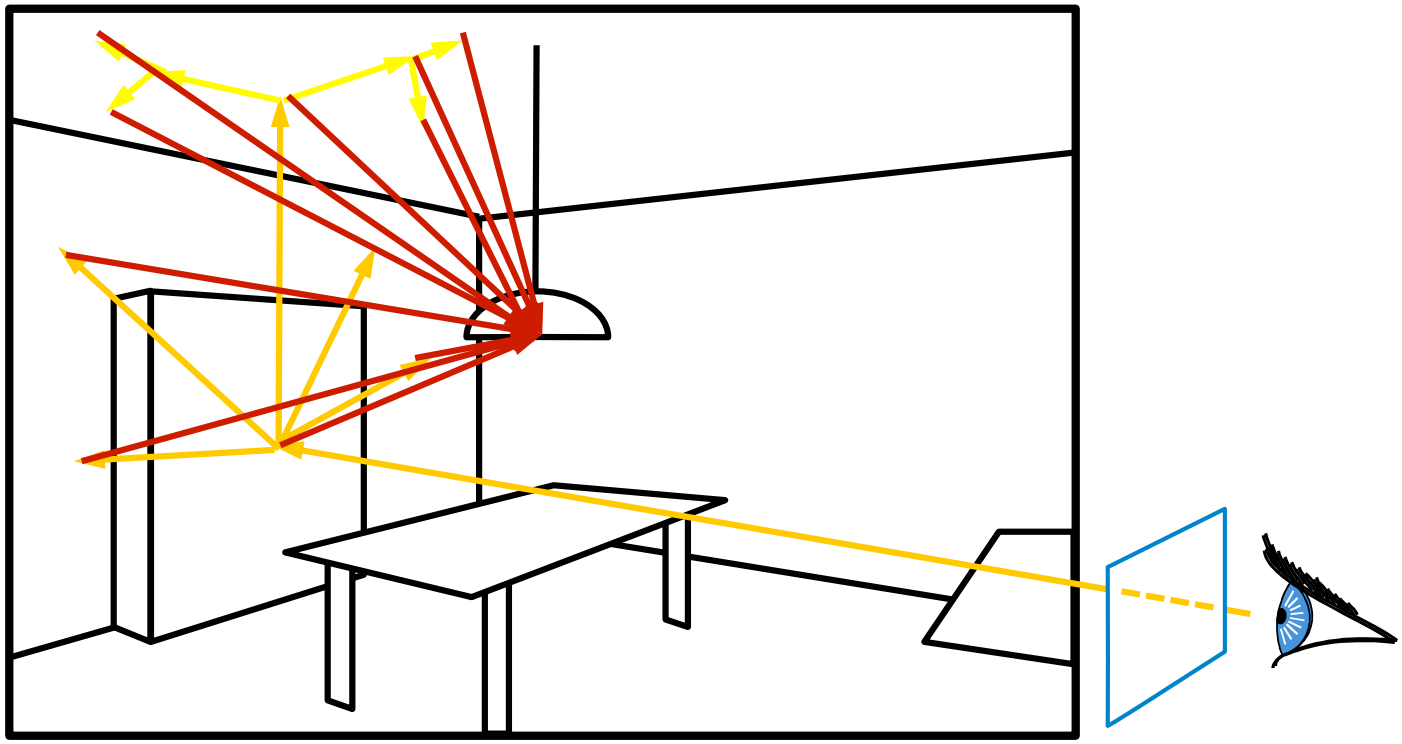
Monte Carlo methods

- On ray for every pixel
- For each visible point : random sampling of rays, accumulate radiance
- Keep going, recursively

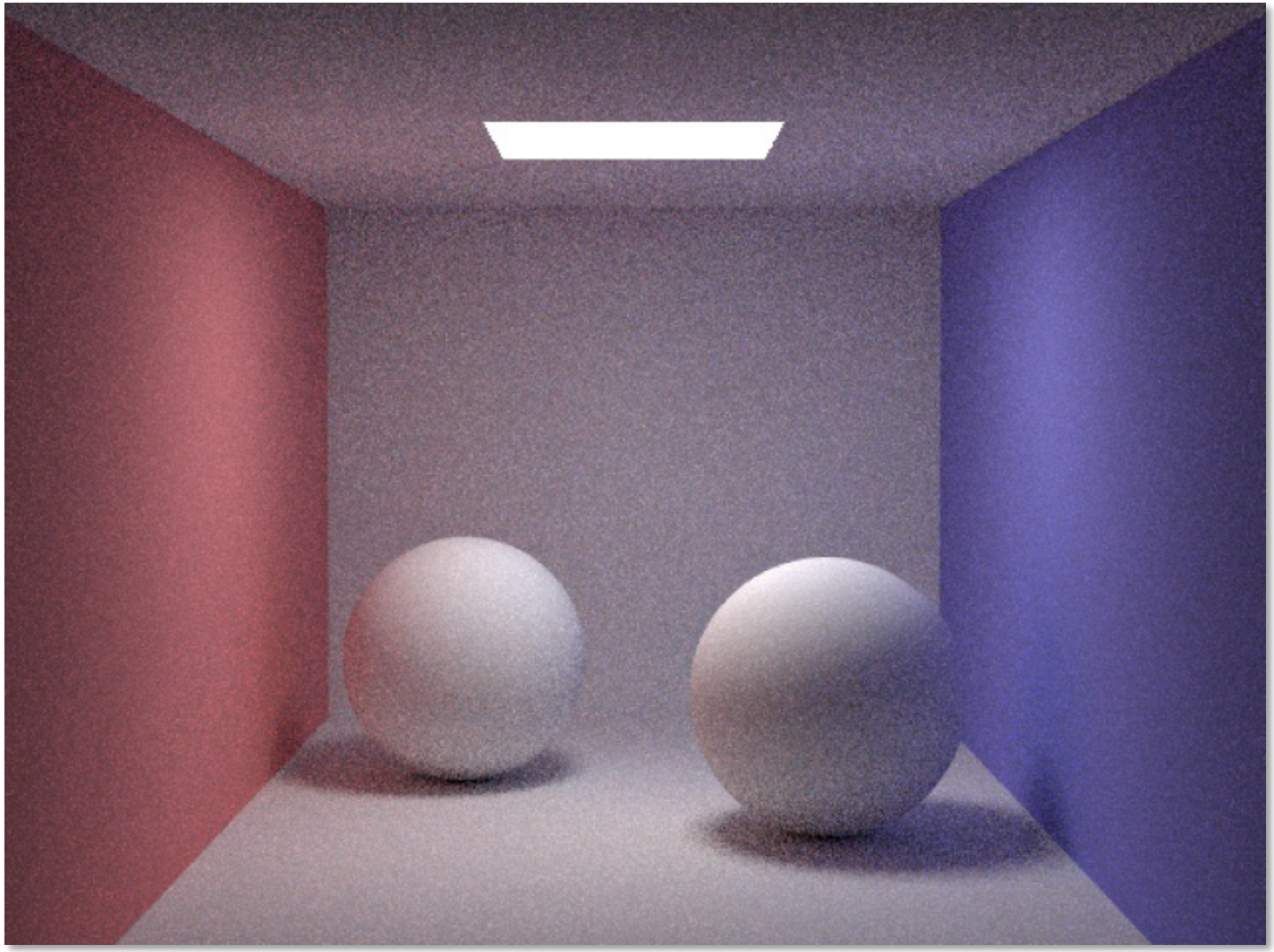


Monte Carlo methods

- On ray for every pixel
- For each visible point : random sampling of rays, accumulate radiance
- Keep going, recursively
- Sample the light source each time

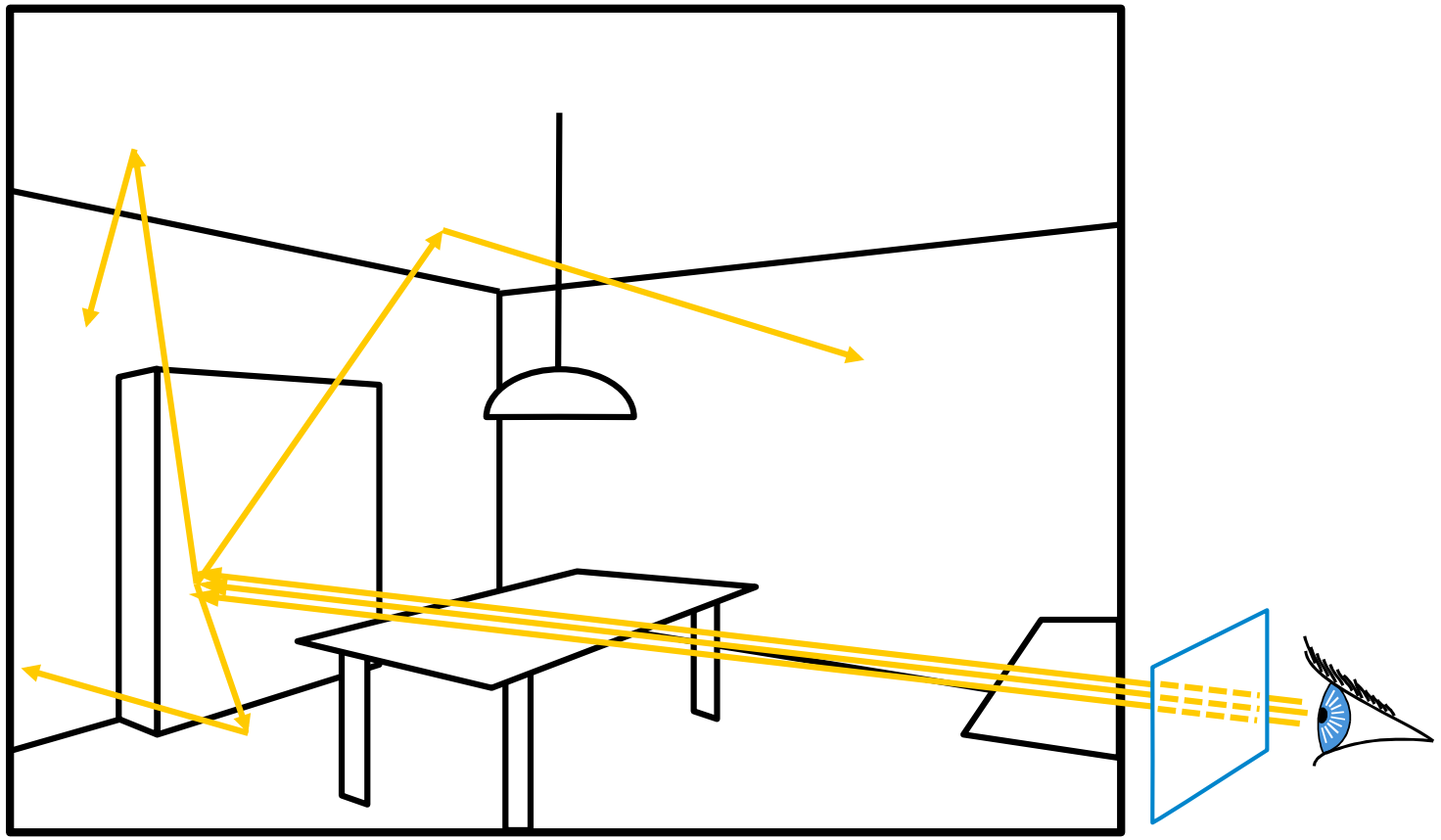


Results



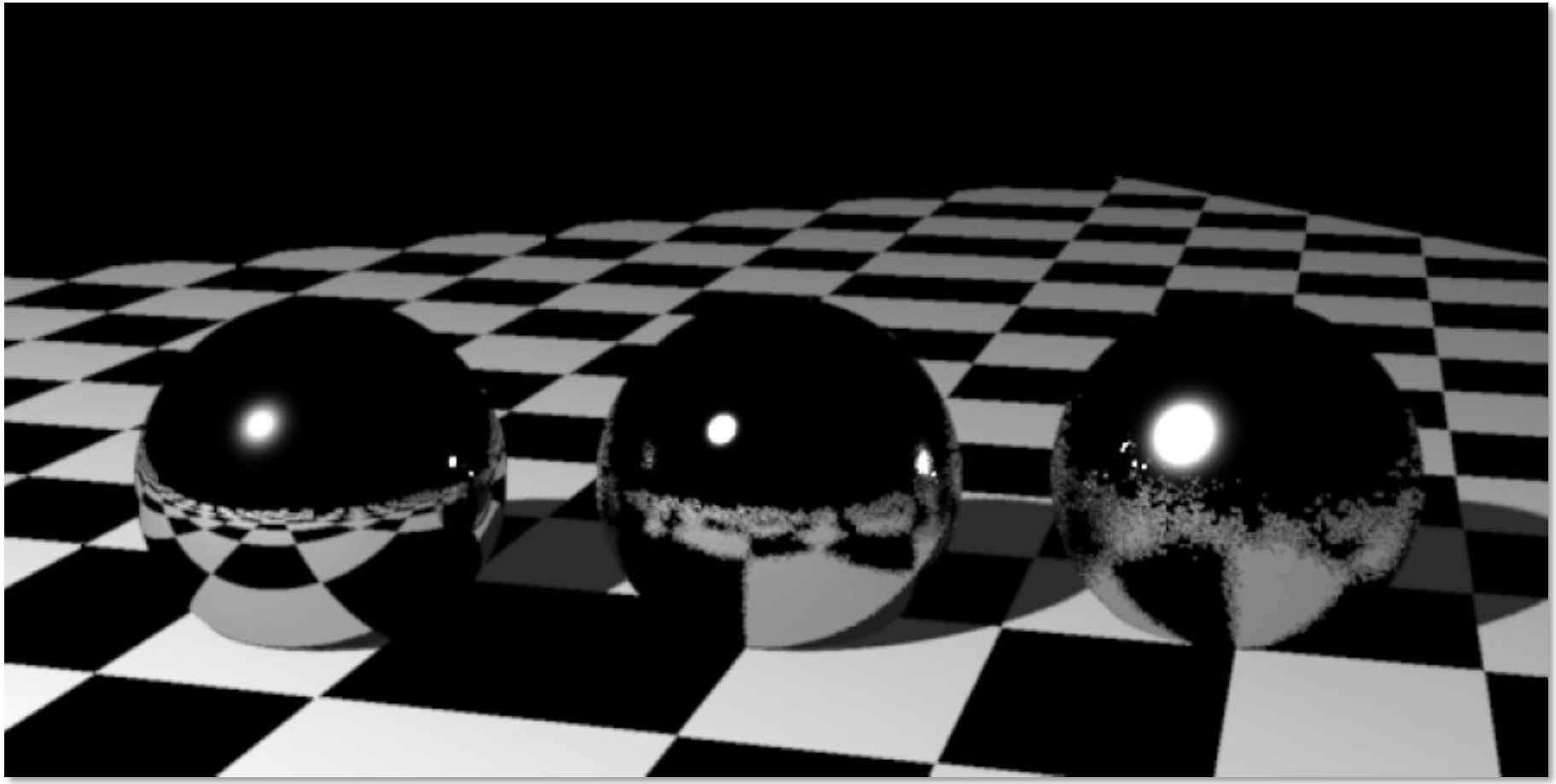
Monte Carlo Path Tracing

- Trace only one ray at each recursion
- But trace several (hundreds of) primary rays for each pixel



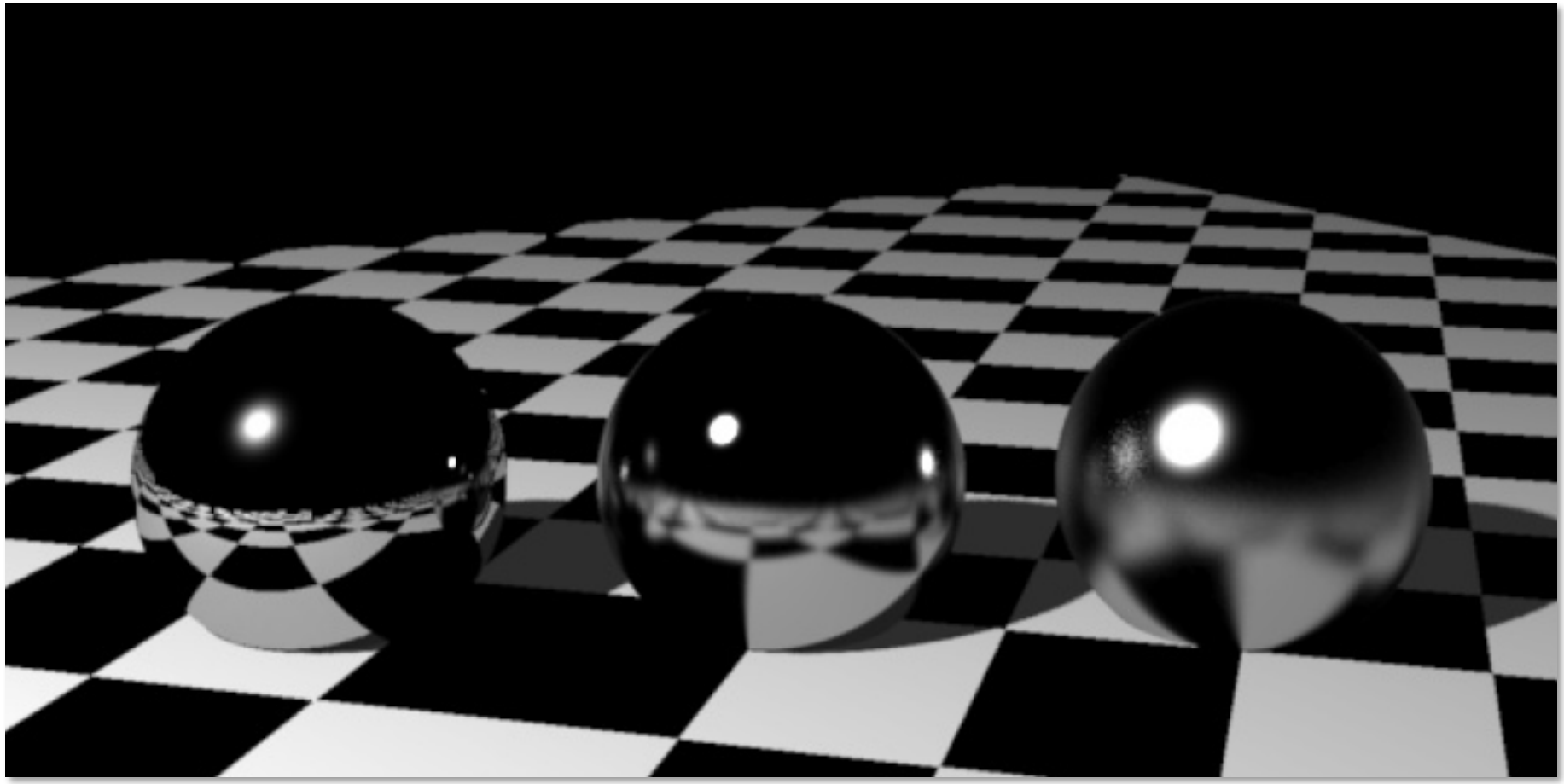
Results

- 1 sample per pixel



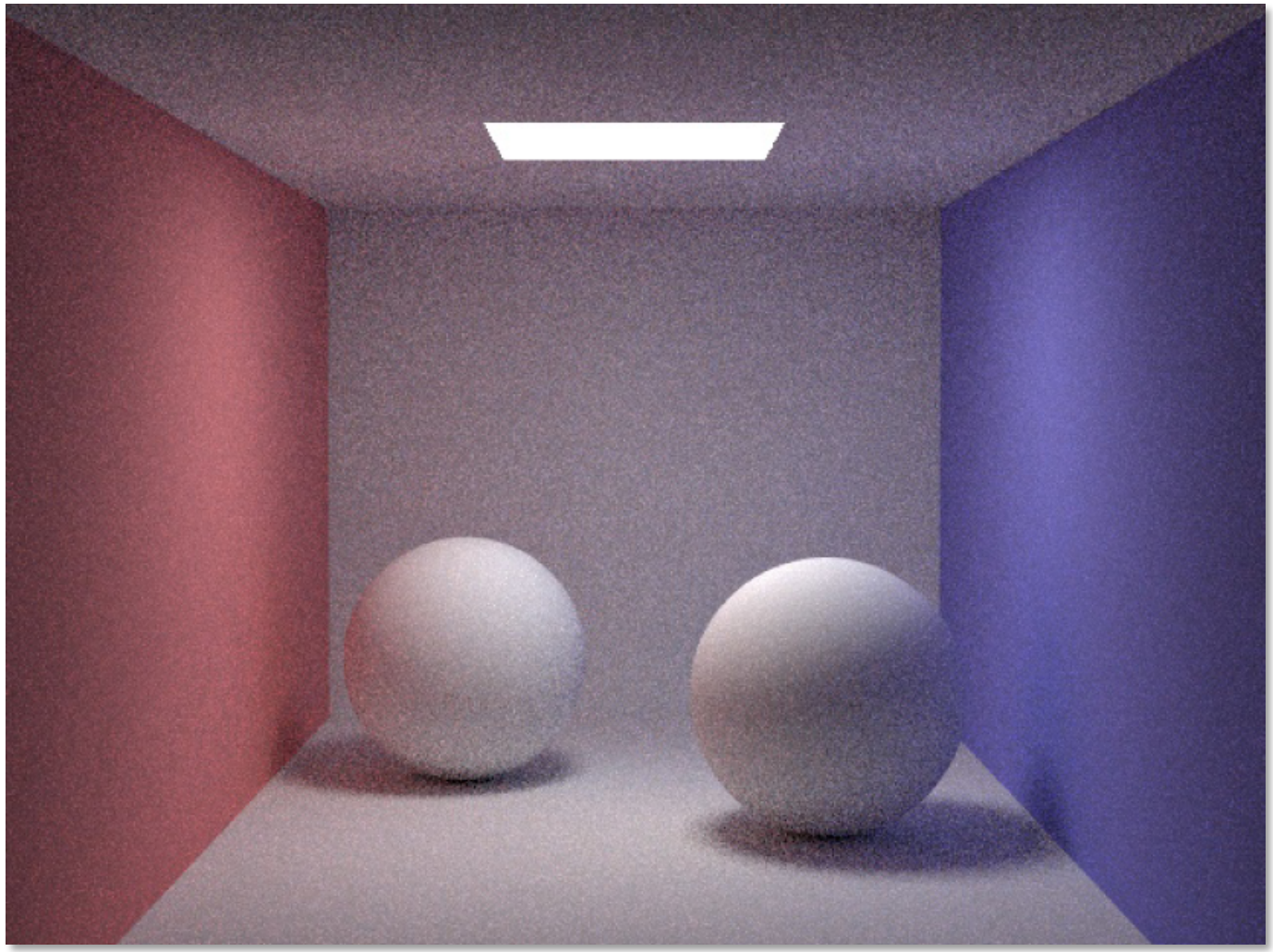
Results

- 256 samples per pixel



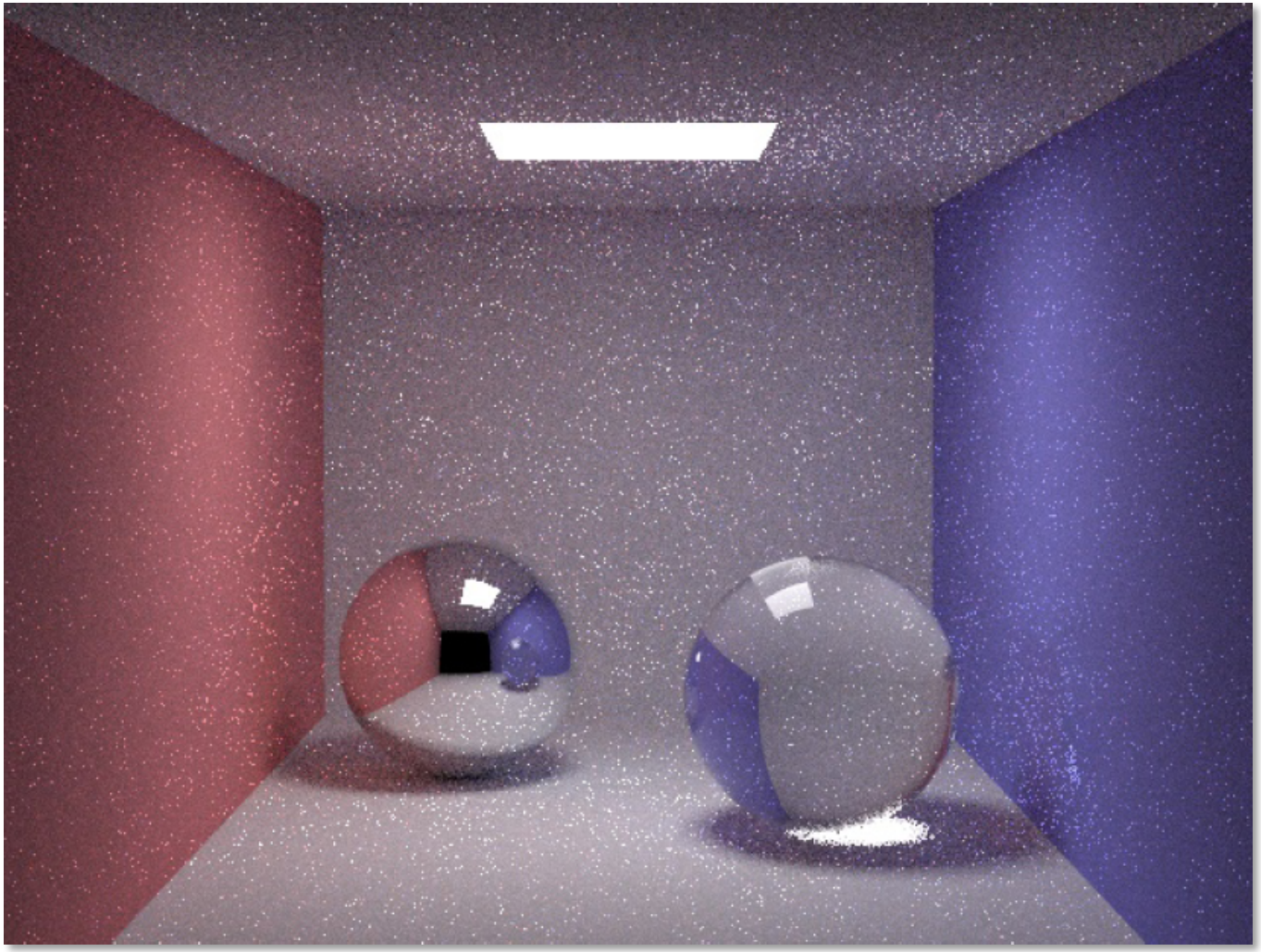
Results: diffuse materials

- 10 paths/pixel



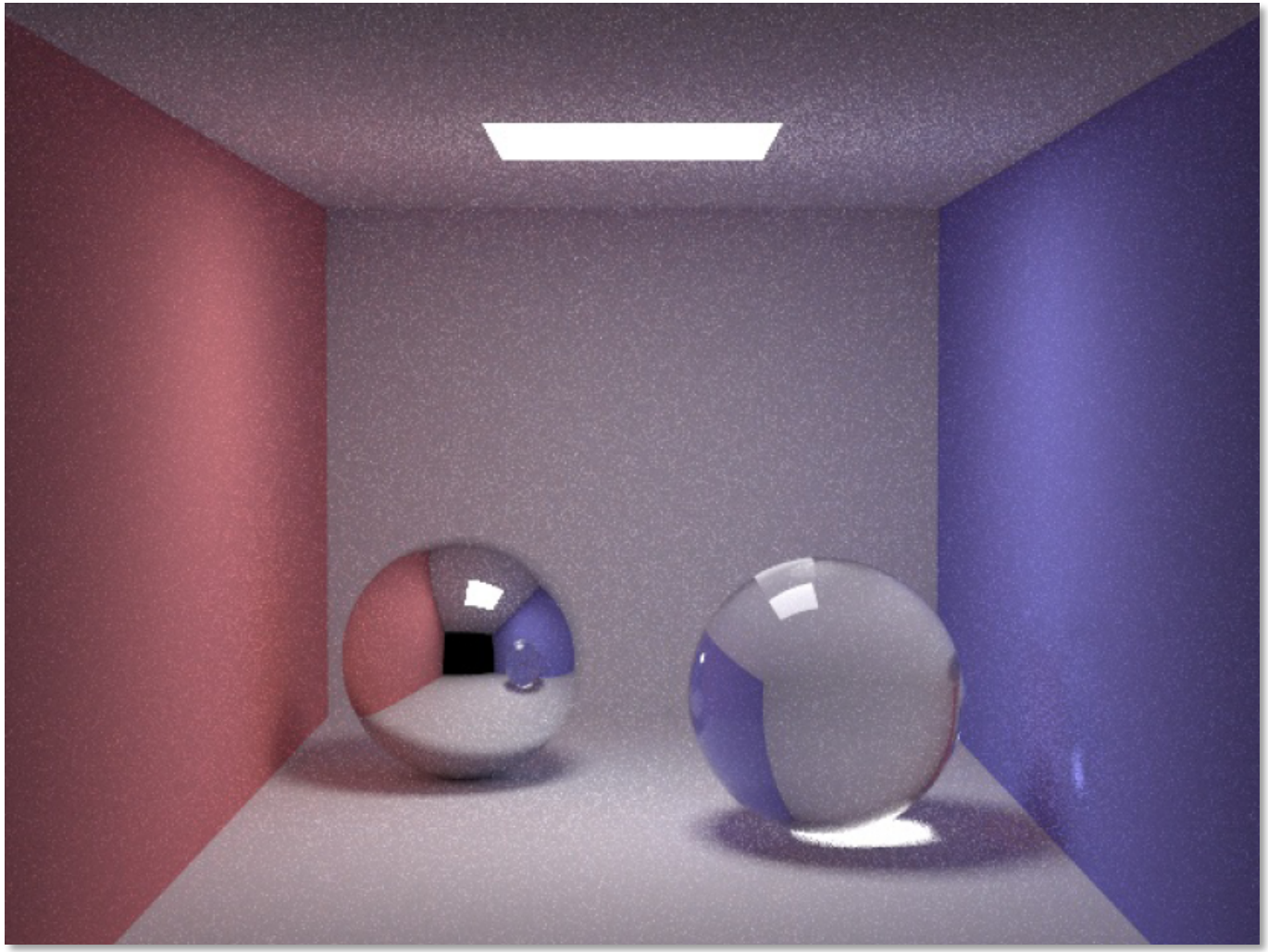
Results: shiny materials

- 10 paths/pixel



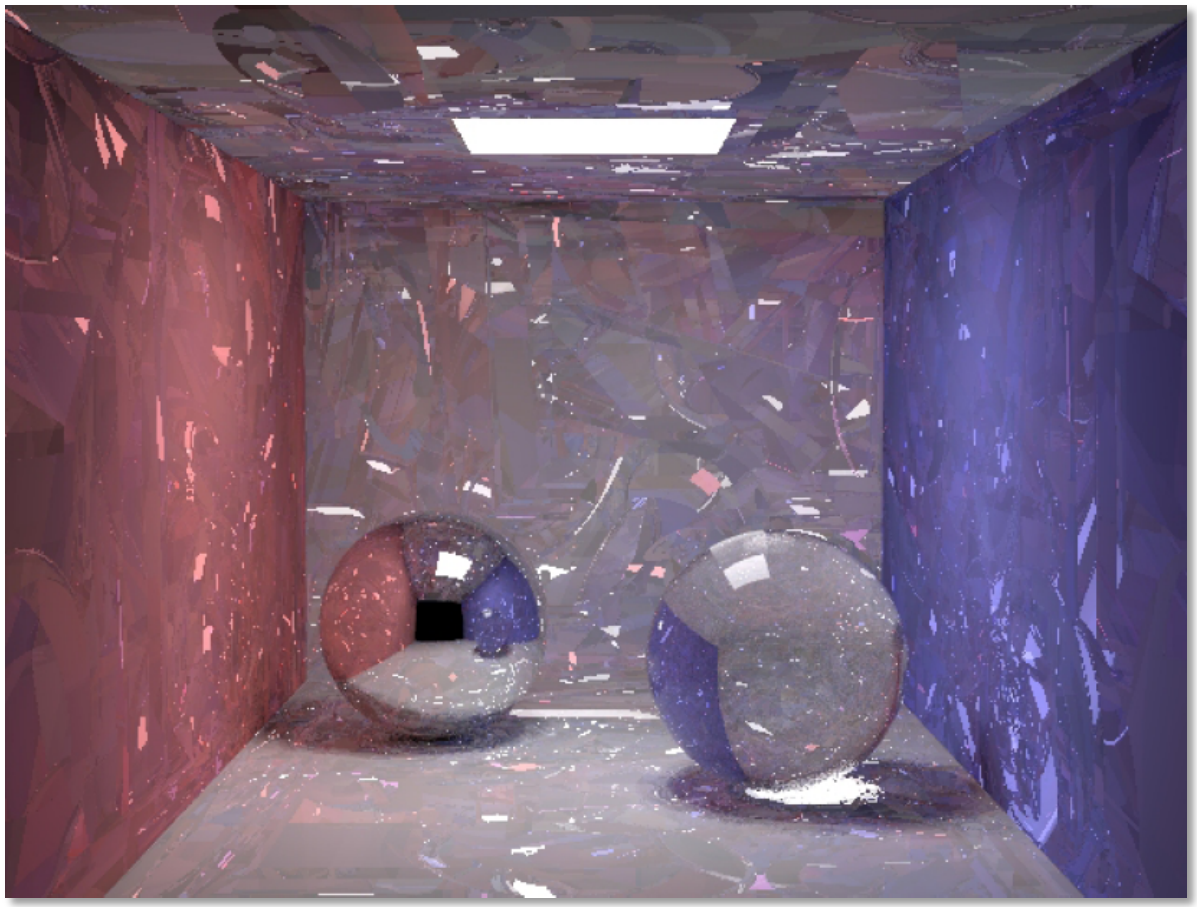
Results: shiny materials

- 100 paths/pixel



Why random sampling?

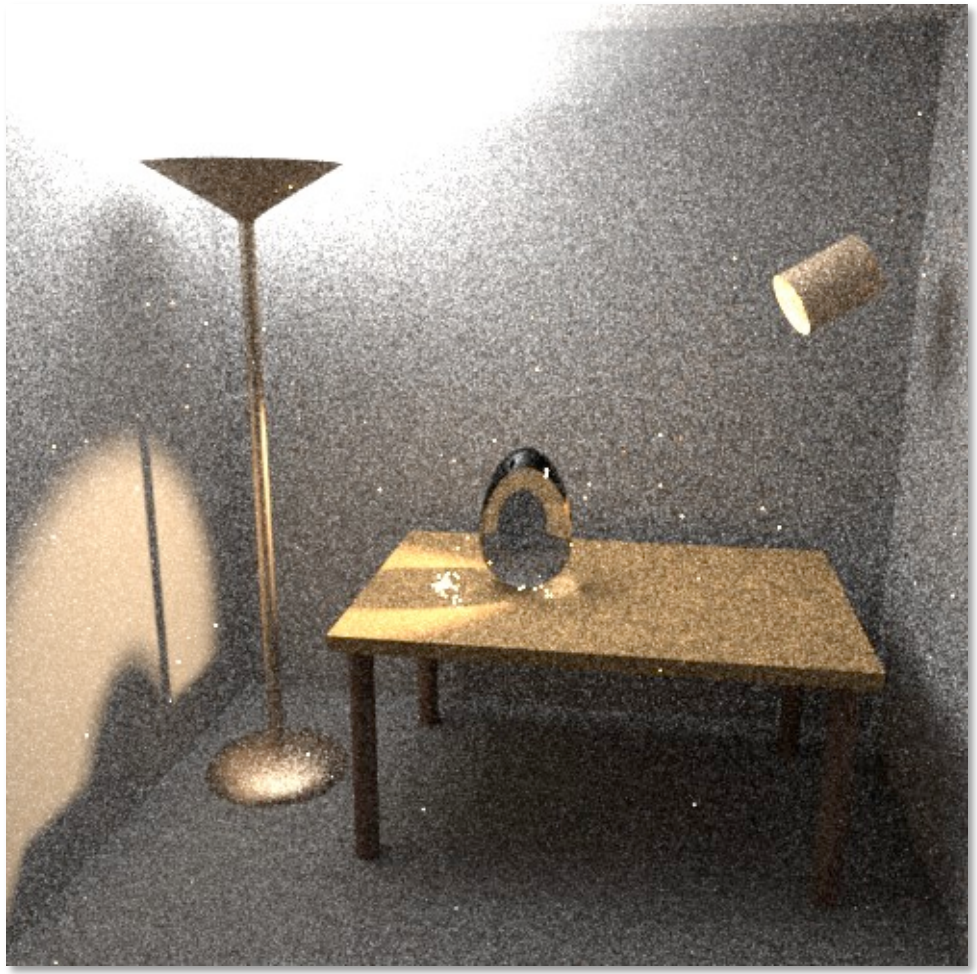
- Pseudo-random sampling sequence
- The structure appears:



Summary

- Send rays randomly
- Sample the rendering equation
- No requirements
  - Any kind of reflectance
  - Any kind of geometry
- Highly adaptive
- Can be noisy and / or very slow
  - Reducing variance: **importance sampling**
  - Speed-up: **irradiance caching**

Importance sampling



Random sampling

Optimal sampling
(Veach and Guibas 1995)

Non-uniform distribution

- N samples with **probability p(x)**
- Monte Carlo estimator becomes:

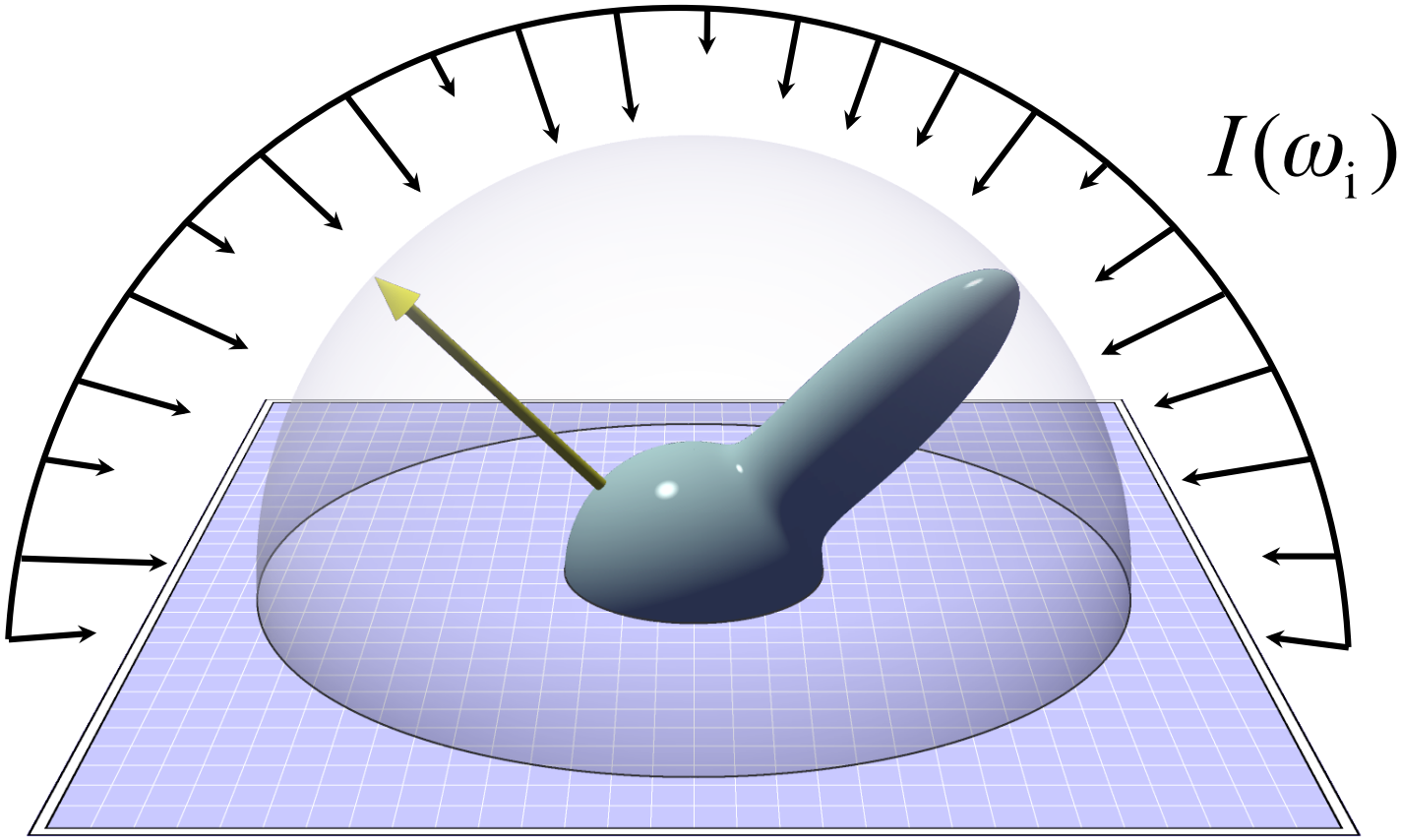
$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

- Probability **p** allows a better sampling of the domain

How can we choose p?

Example: glossy reflections

- Integral over hemisphere of directions
- BRDF x cosine x incoming light

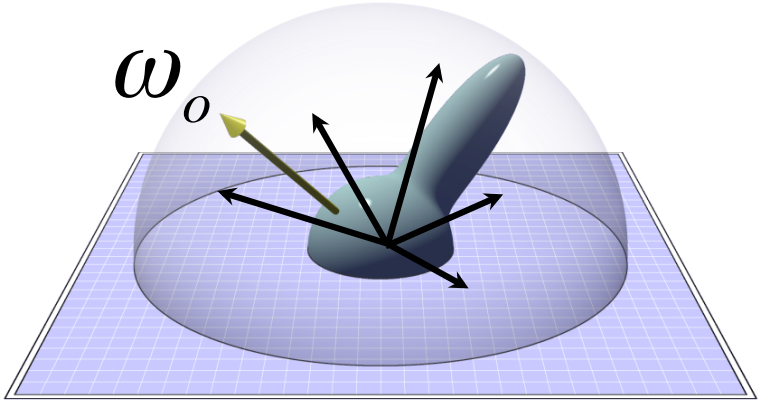


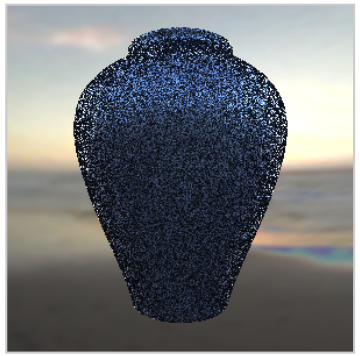
Slide courtesy of Jason Lawrence

Sampling a BRDF

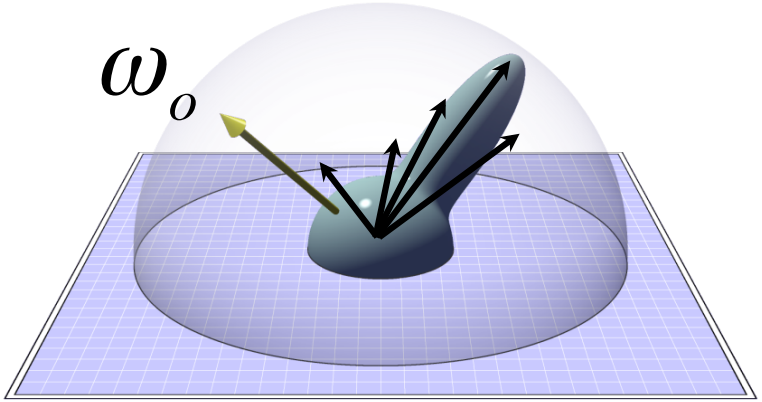
5 Samples/Pixel

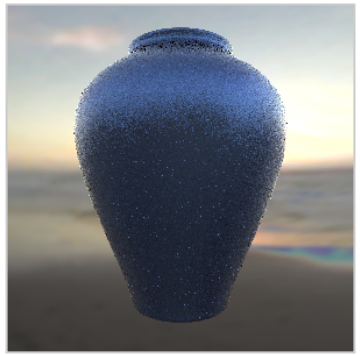
$U(\omega_i)$





$P(\omega_i)$



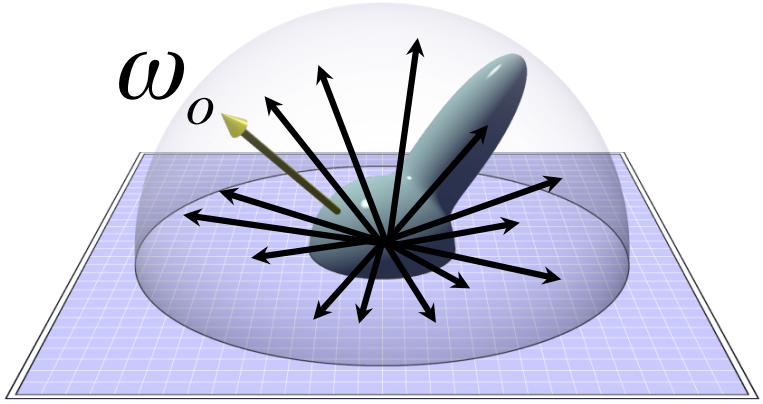


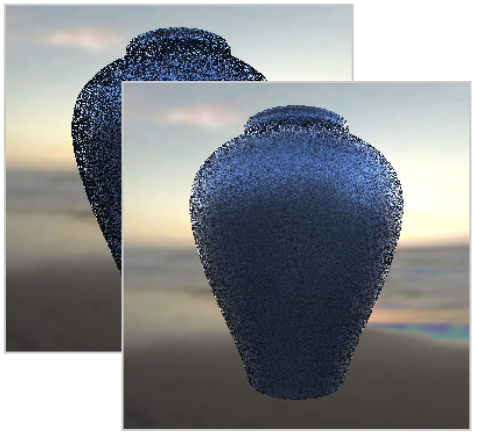
Slide courtesy of Jason Lawrence

Sampling a BRDF

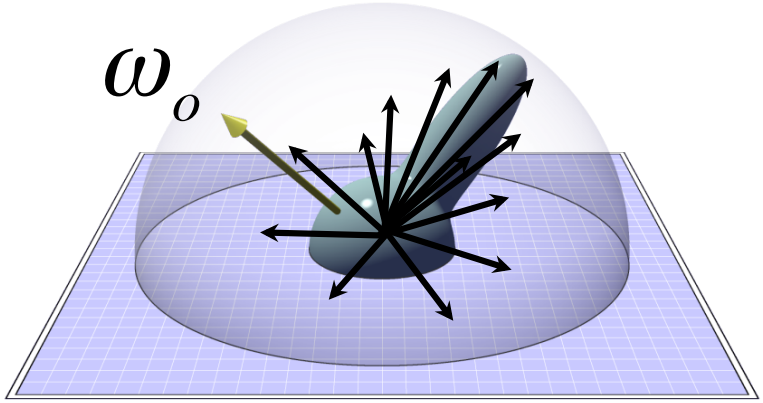
25 Samples/Pixel

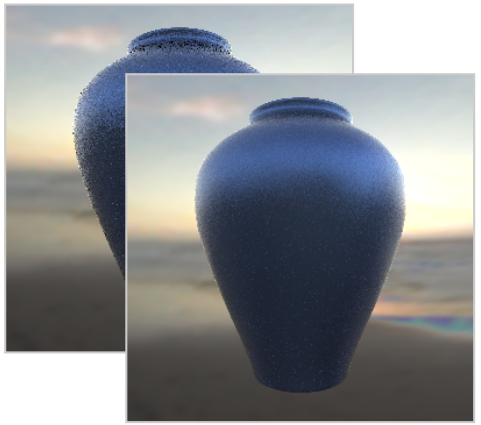
$U(\omega_i)$





$P(\omega_i)$

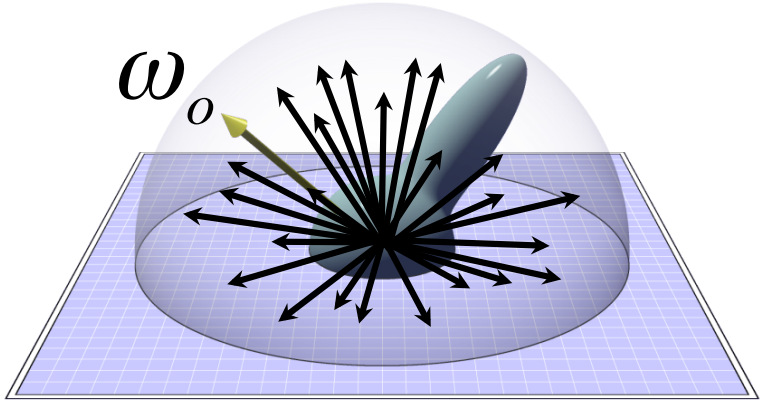




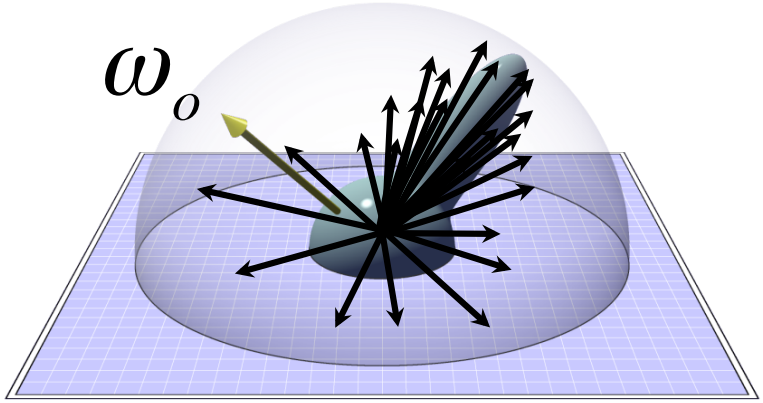
Slide courtesy of Jason Lawrence

Sampling a BRDF

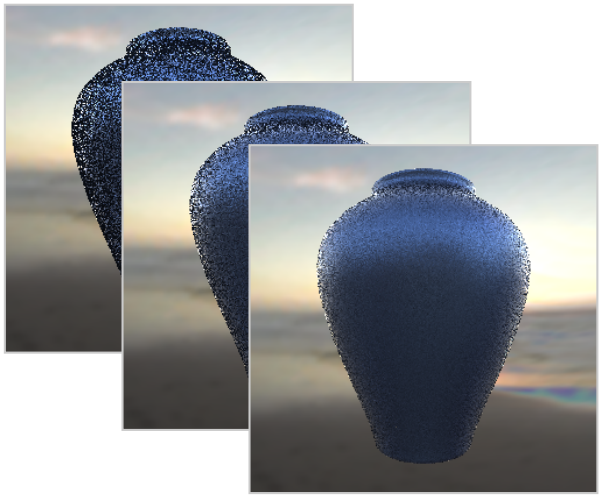
$U(\omega_i)$

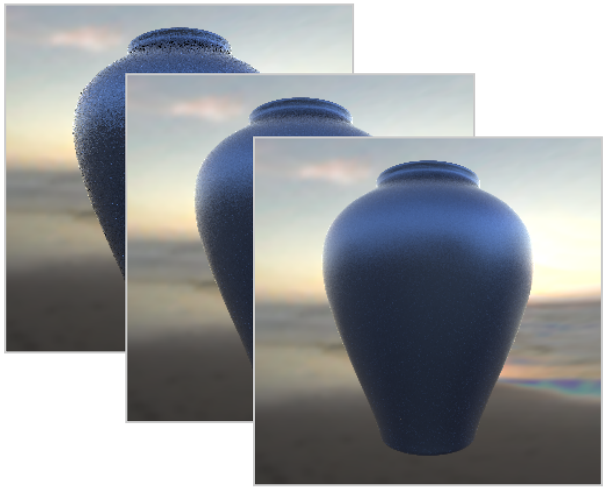


$P(\omega_i)$



75 Samples/Pixel





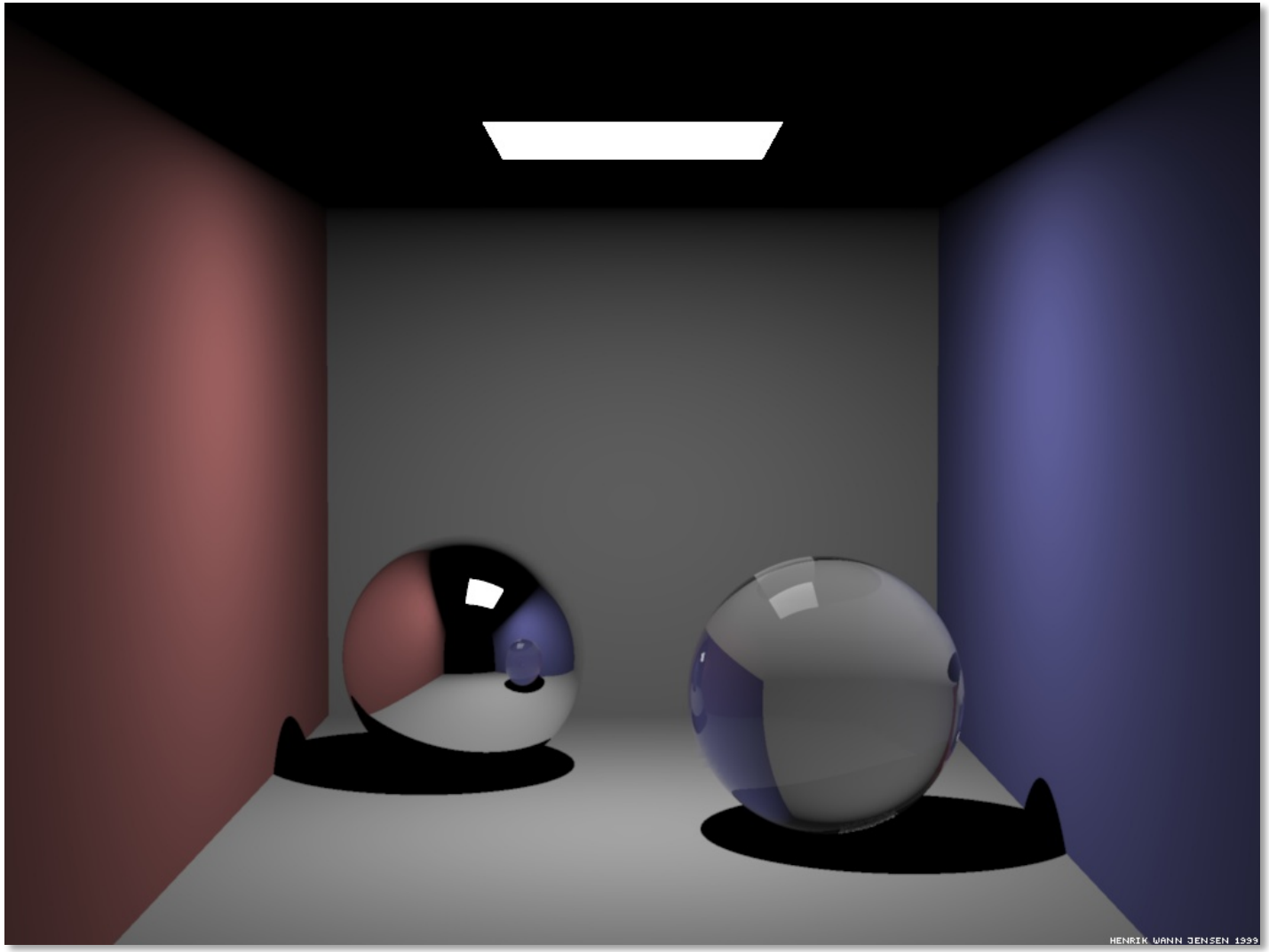
Slide courtesy of Jason Lawrence

Importance sampling

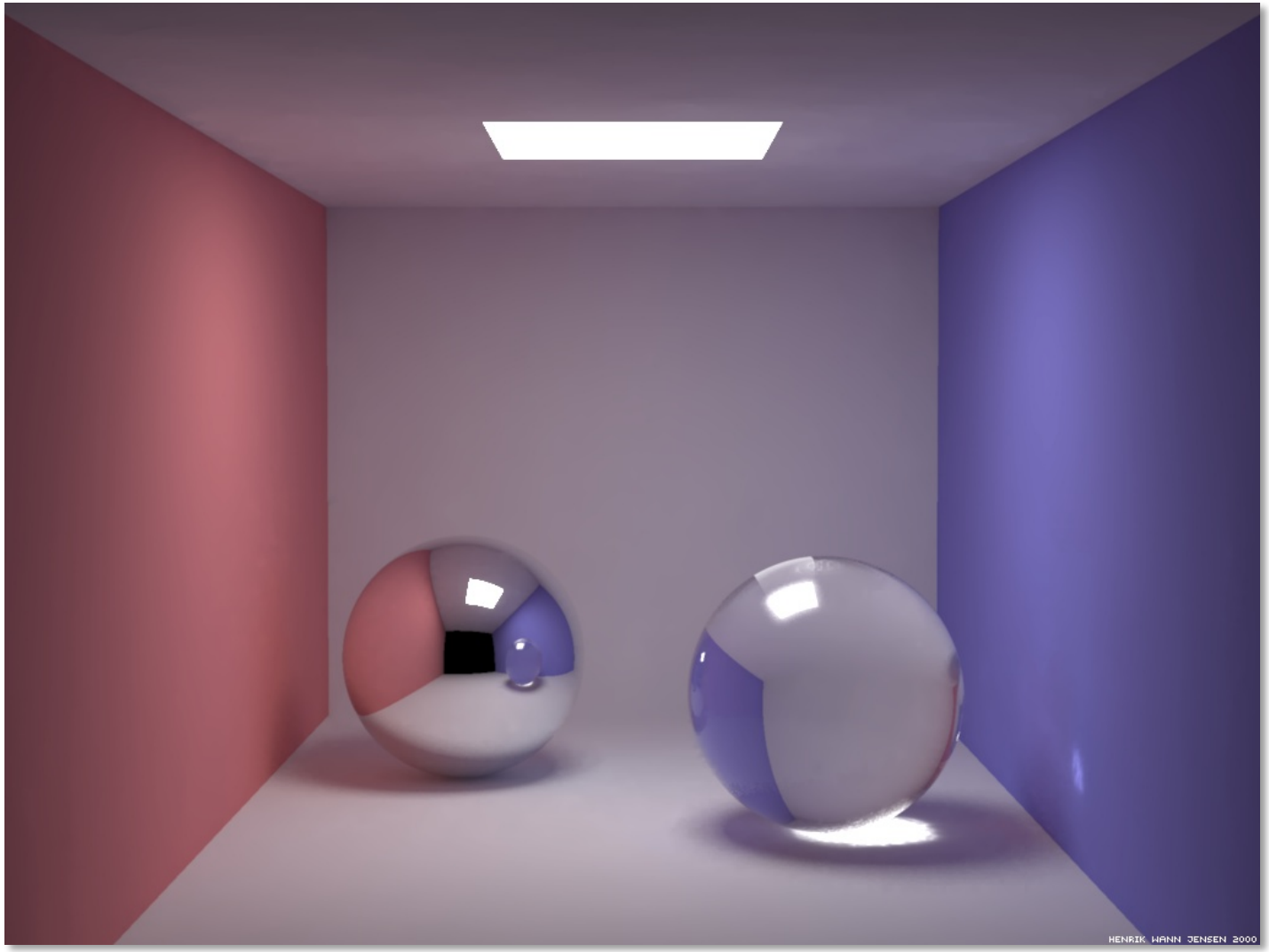
$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

- Choose **p** wisely to reduce variance:
  - **p** must *look like* **f**
  - Doesn't change convergence with $\sqrt{N}$ (reduces the constant)

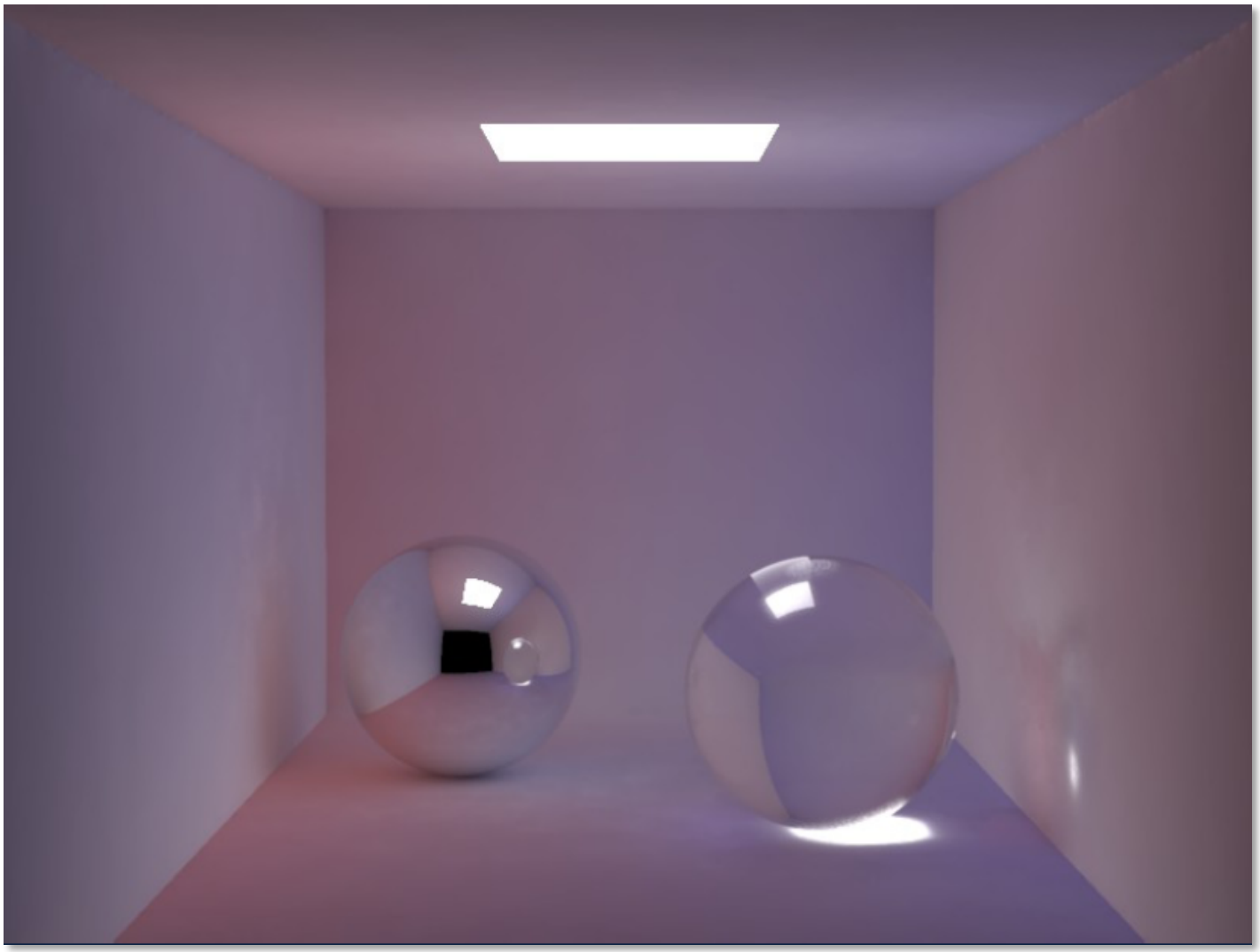
Direct lighting



Global illumination

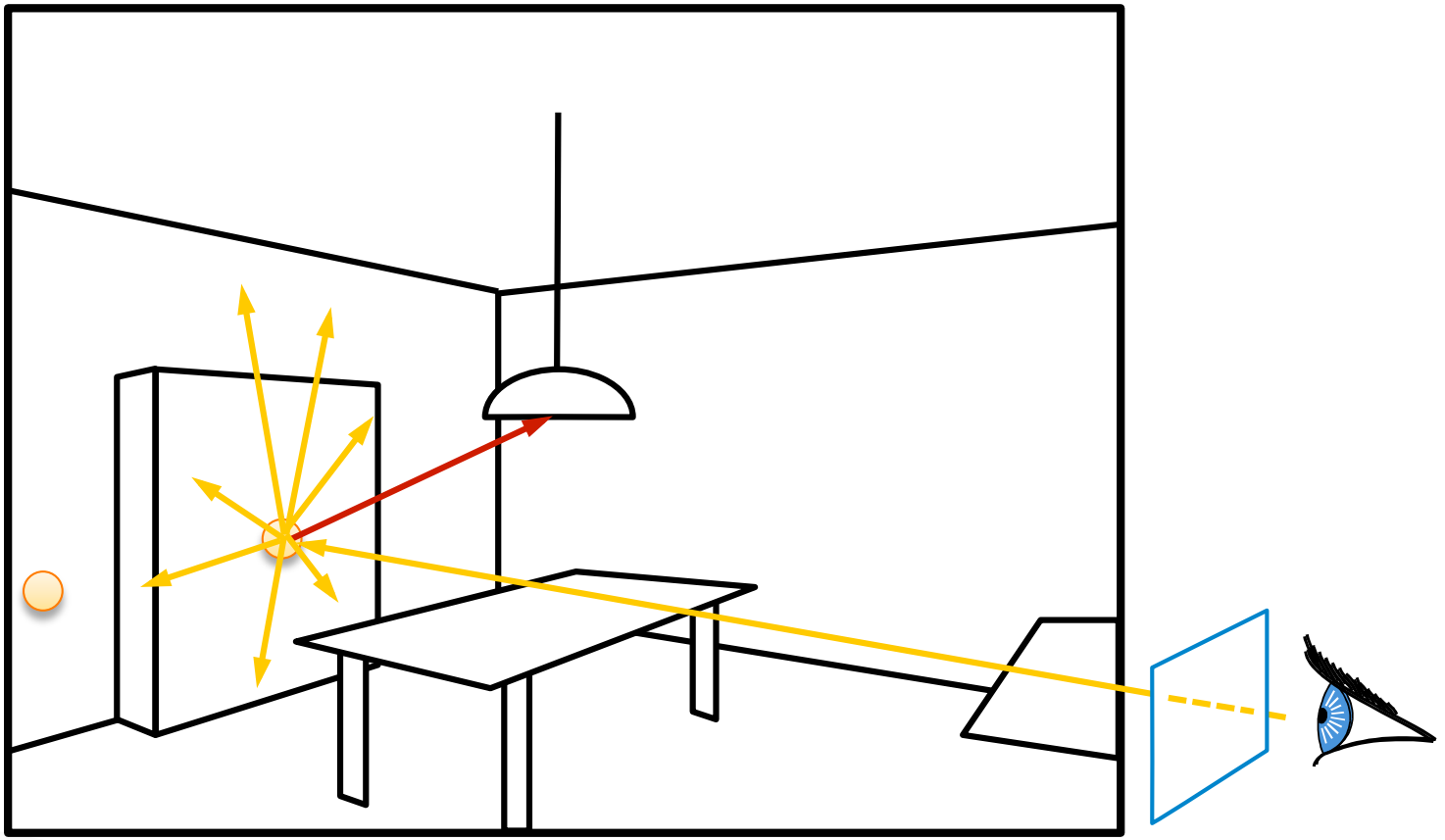


Indirect lighting



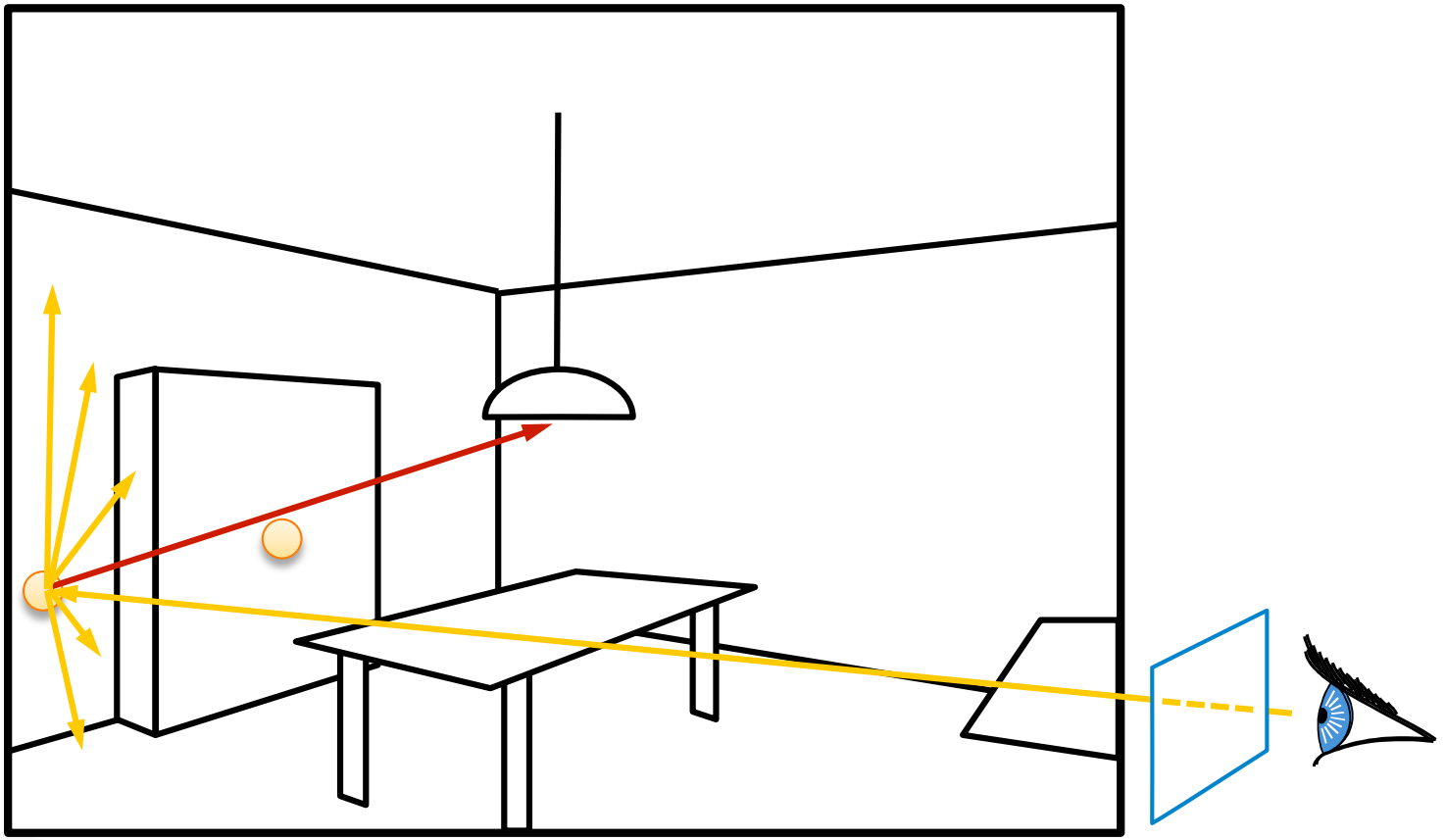
Irradiance cache

- Indirect lighting changes slowly in space



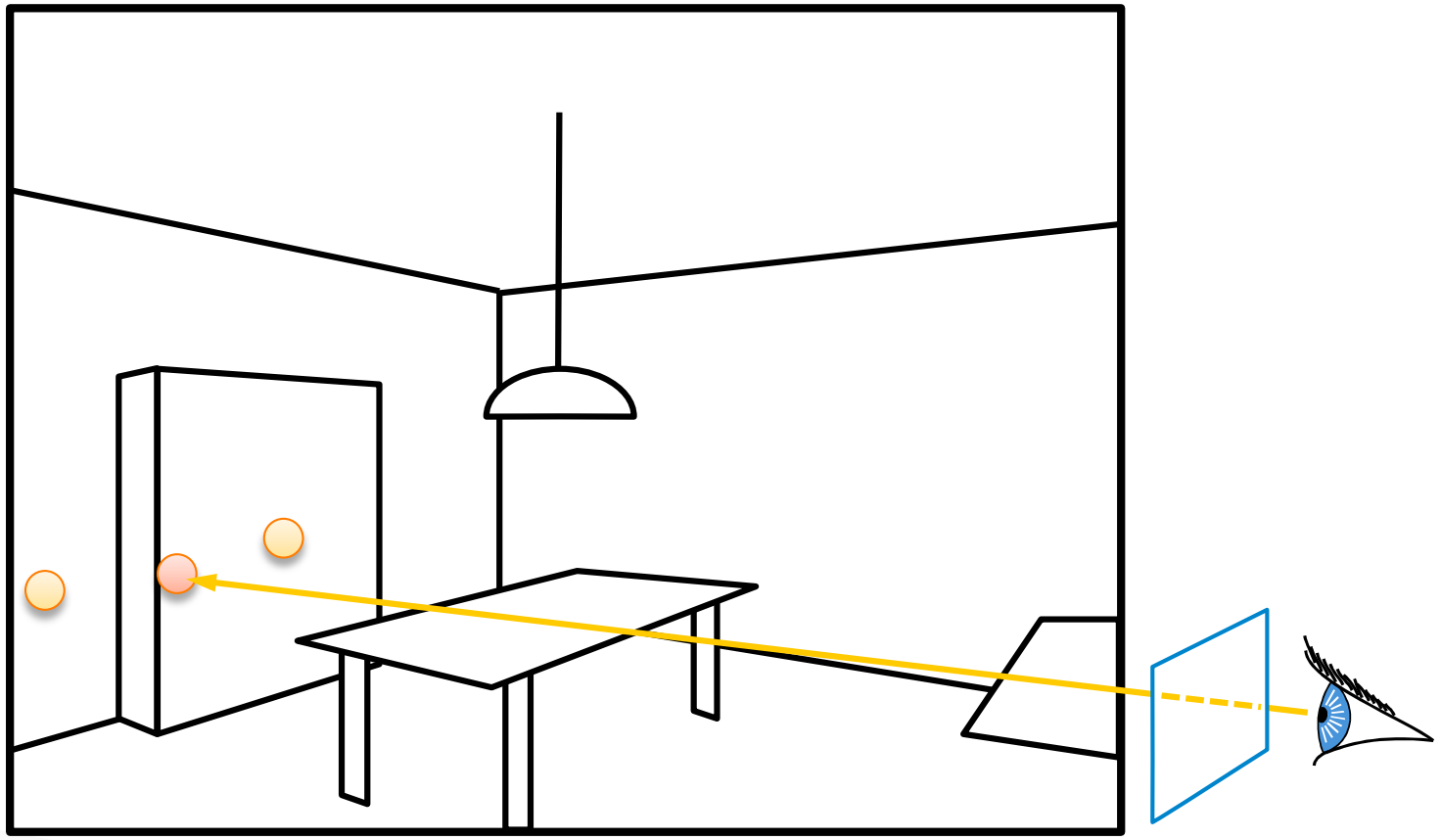
Irradiance cache

- Indirect lighting changes slowly in space



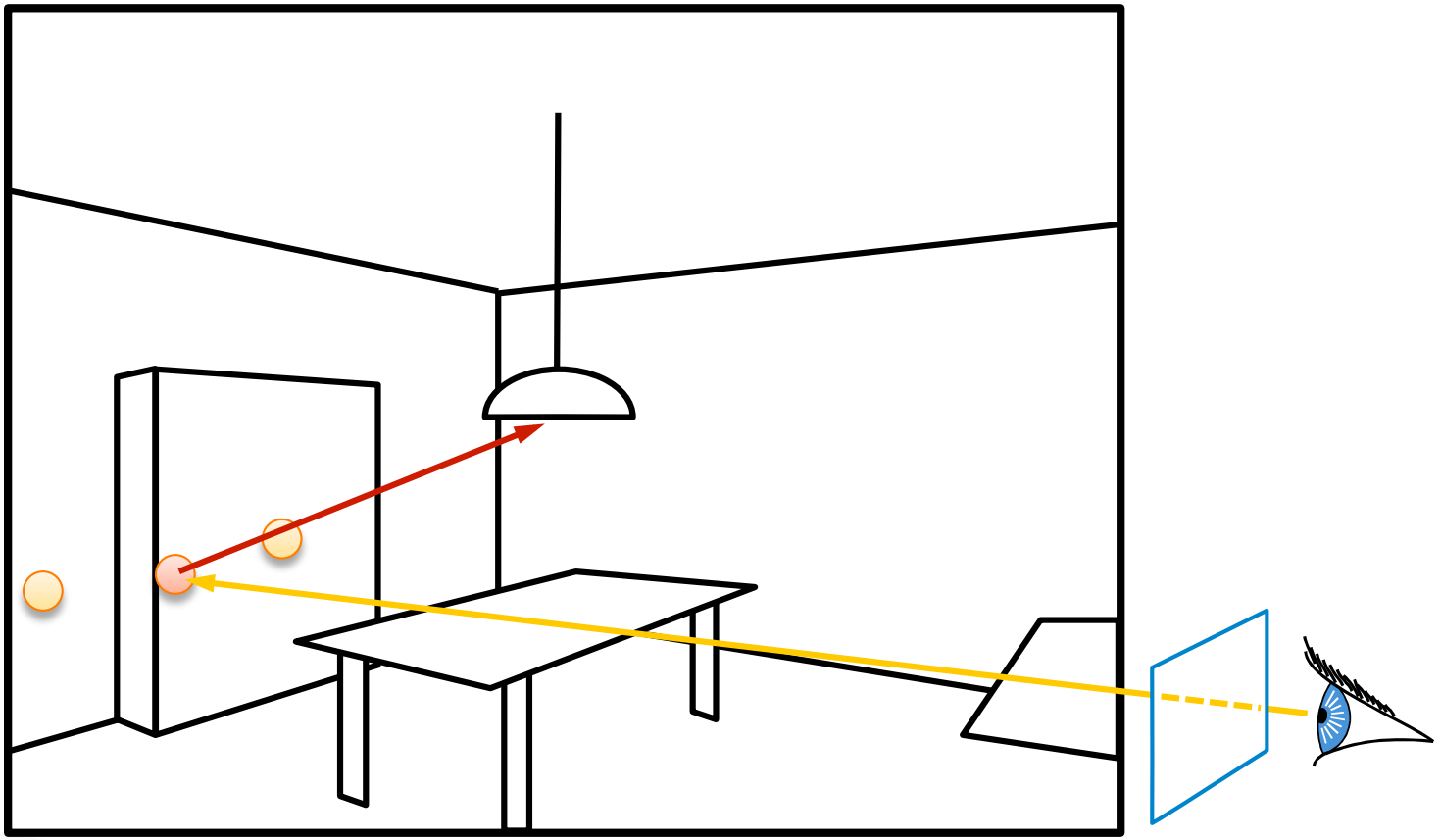
Irradiance cache

- Indirect lighting changes slowly in space
- Interpolate between neighboring values



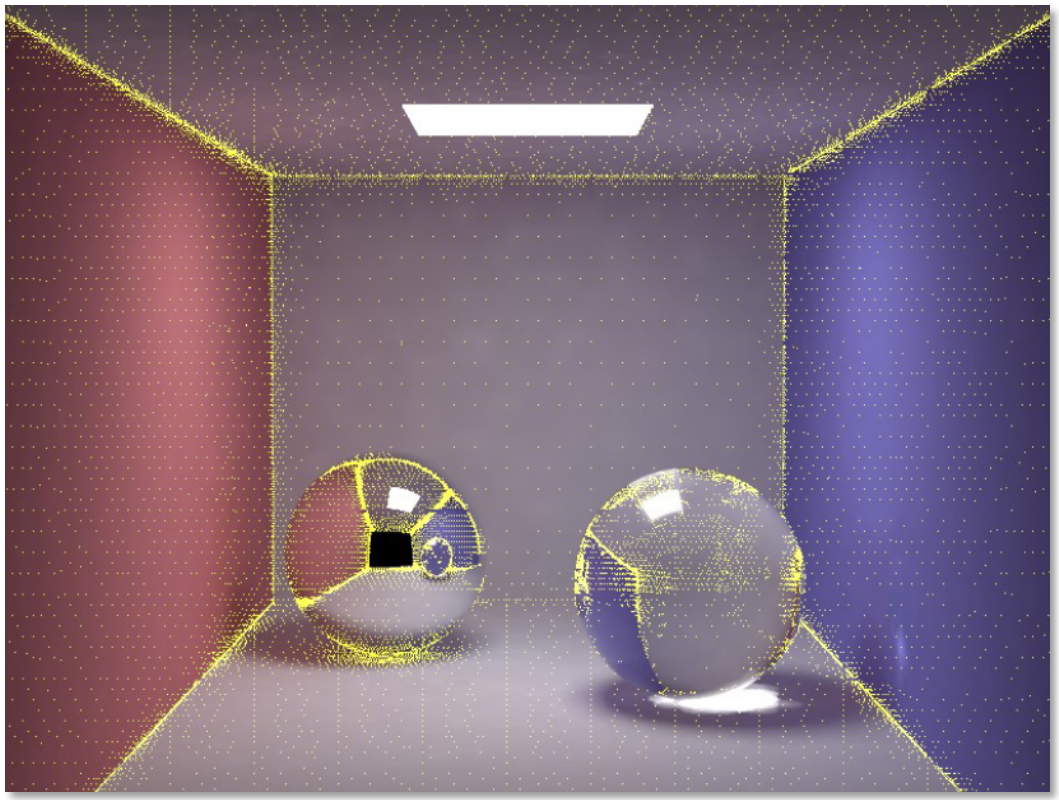
Irradiance cache

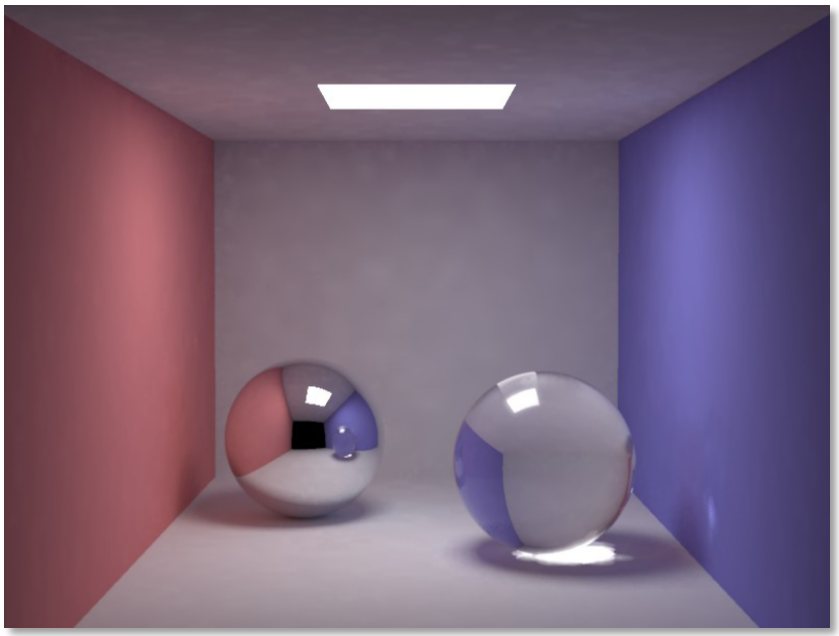
- Indirect lighting changes slowly in space
- Interpolate between cached values
- But full computation for direct lighting



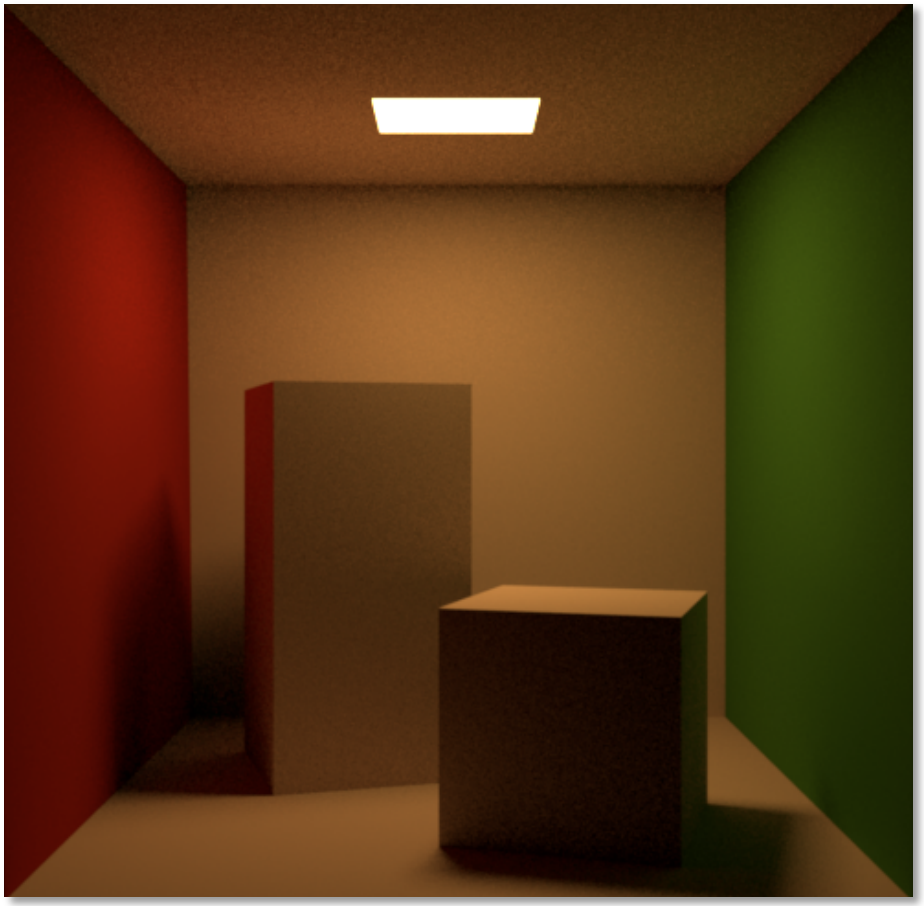
Irradiance cache

- Yellow dots: computation of indirect lighting





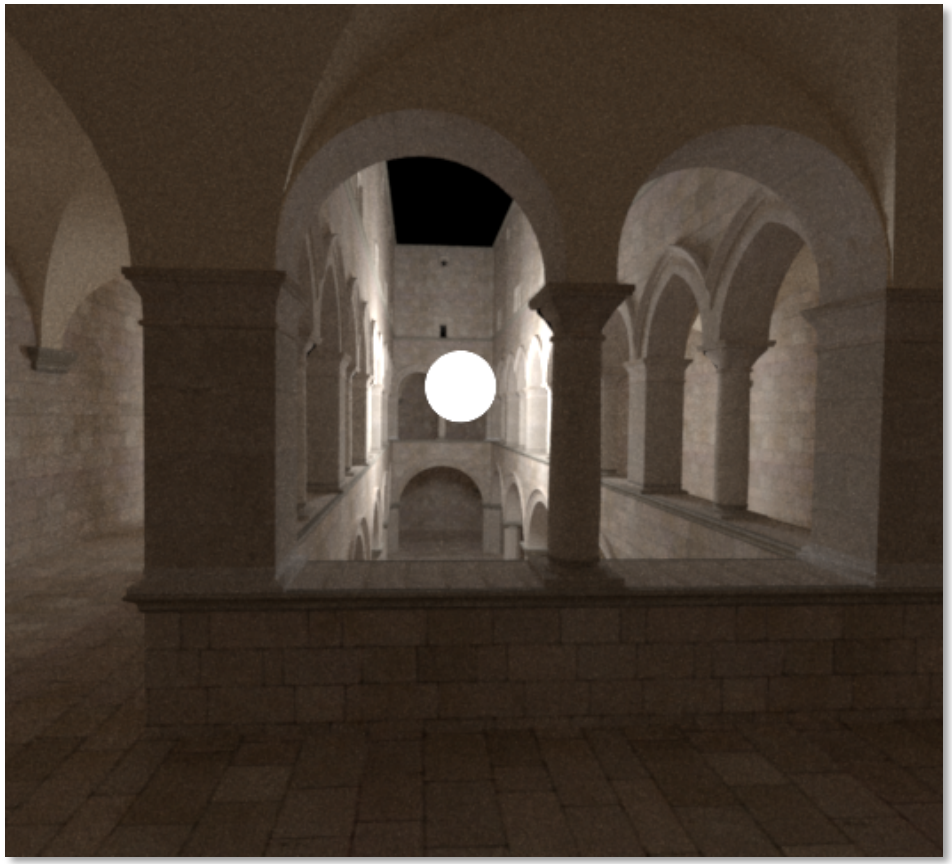
Irradiance cache



Path Tracing

Path tracing + Irradiance cache

Irradiance cache

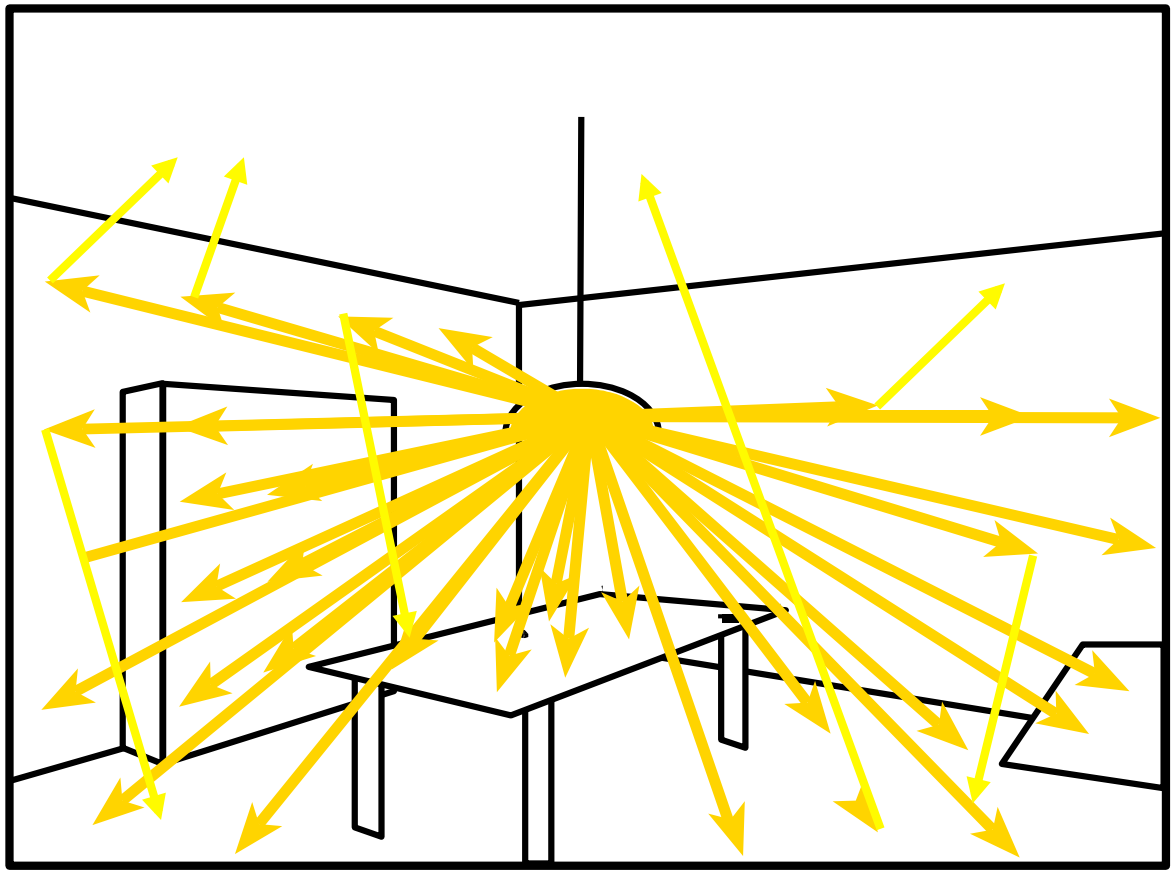


Path Tracing

Path tracing + Irradiance cache

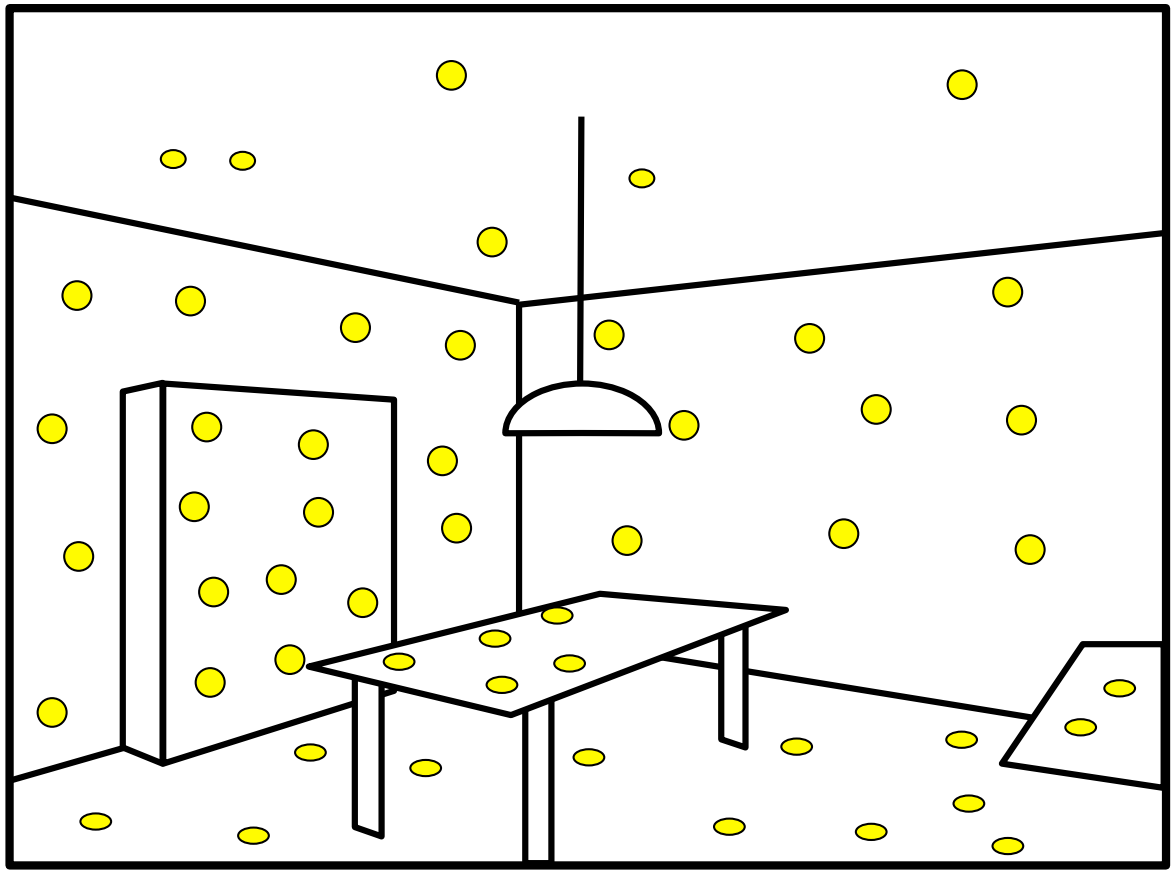
Photon mapping

- Pre-computation: throw rays from the light sources



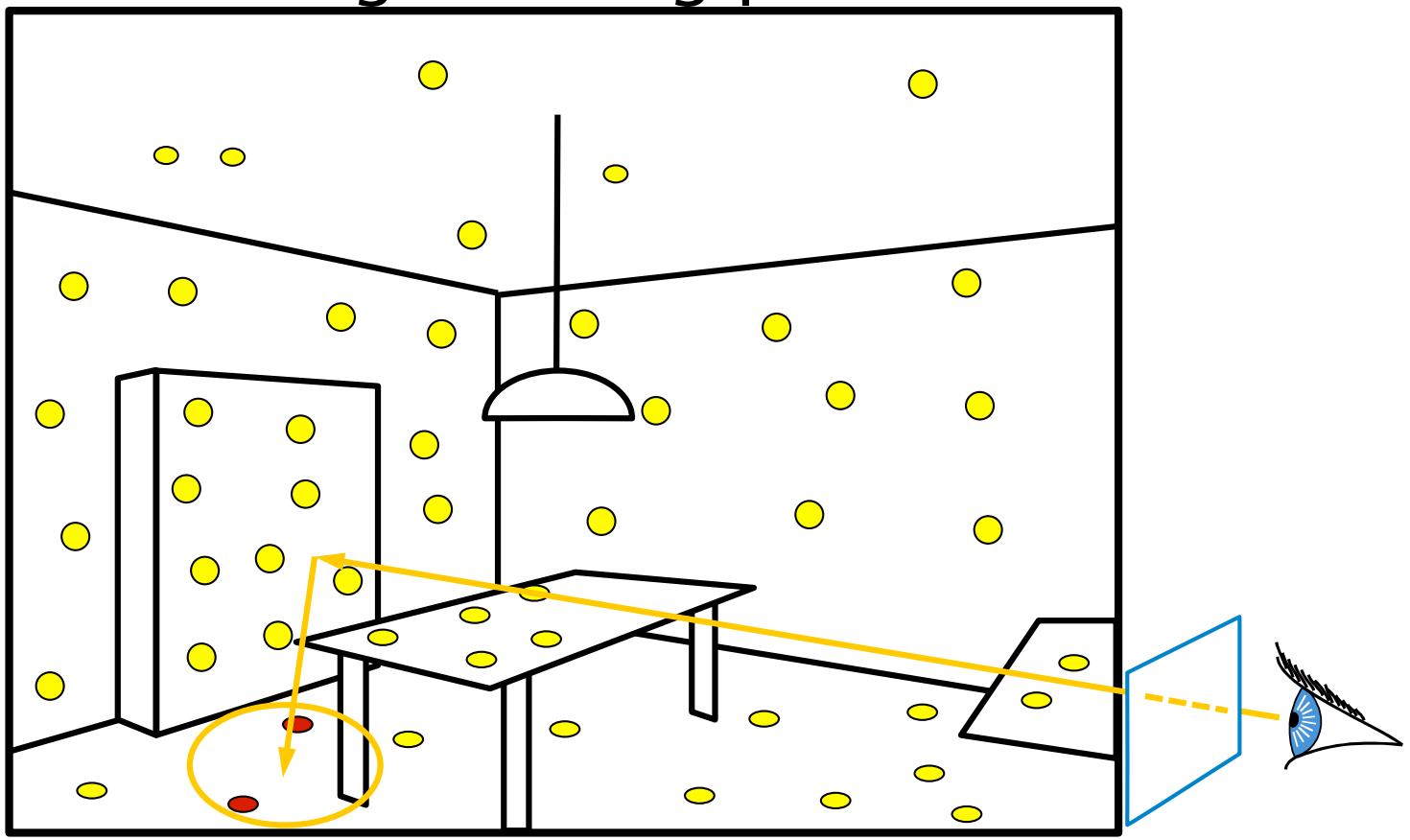
Photon mapping

- Store photons (position + intensity + direction) on the geometry or inside an data structure

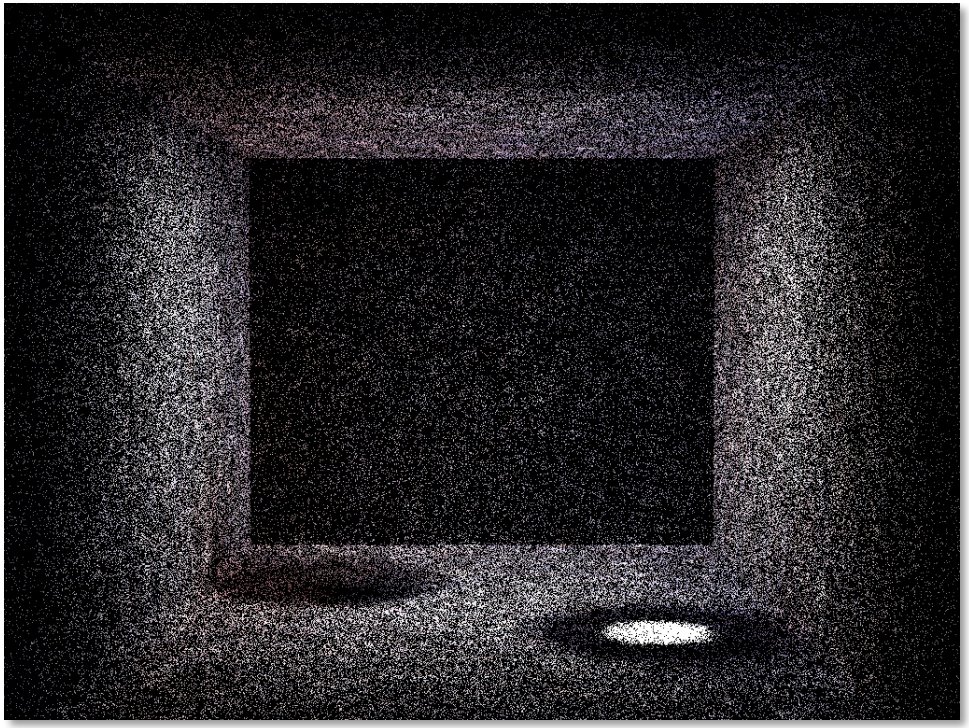


Photon mapping – rendering

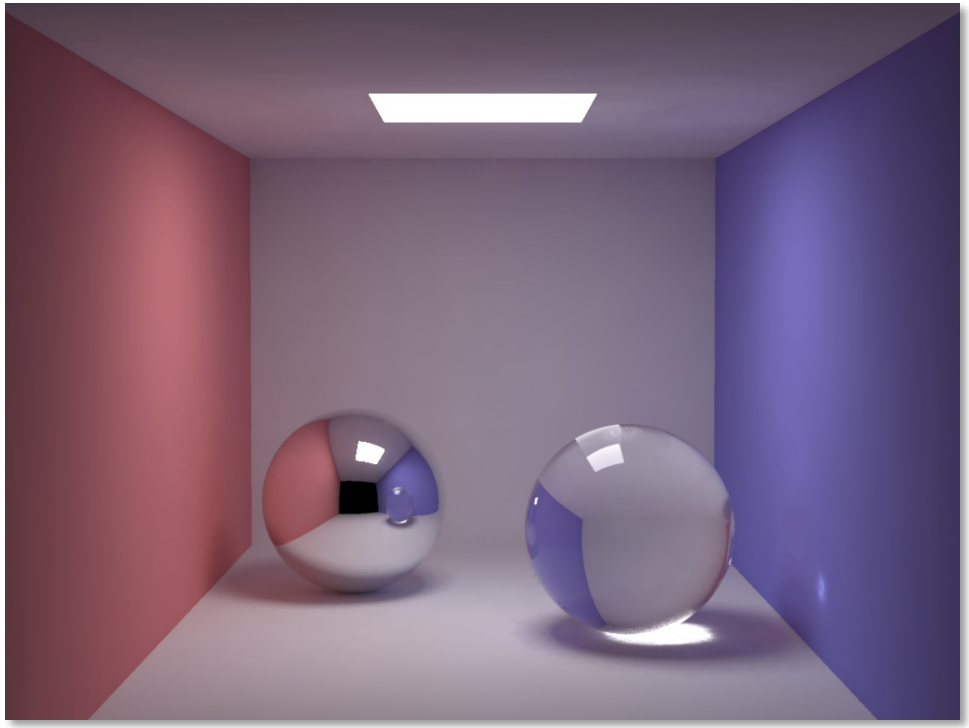
- Throw primary rays
- Radiance for secondary rays by gathering radiance from neighbouring photons



Results



Photon map



Rendu final

Results

- Jensen (1996)
  - Direct visualization of the photon map: 6min

Results

- Jensen (1996)
  - Final gather pass: + 51 mn

Results

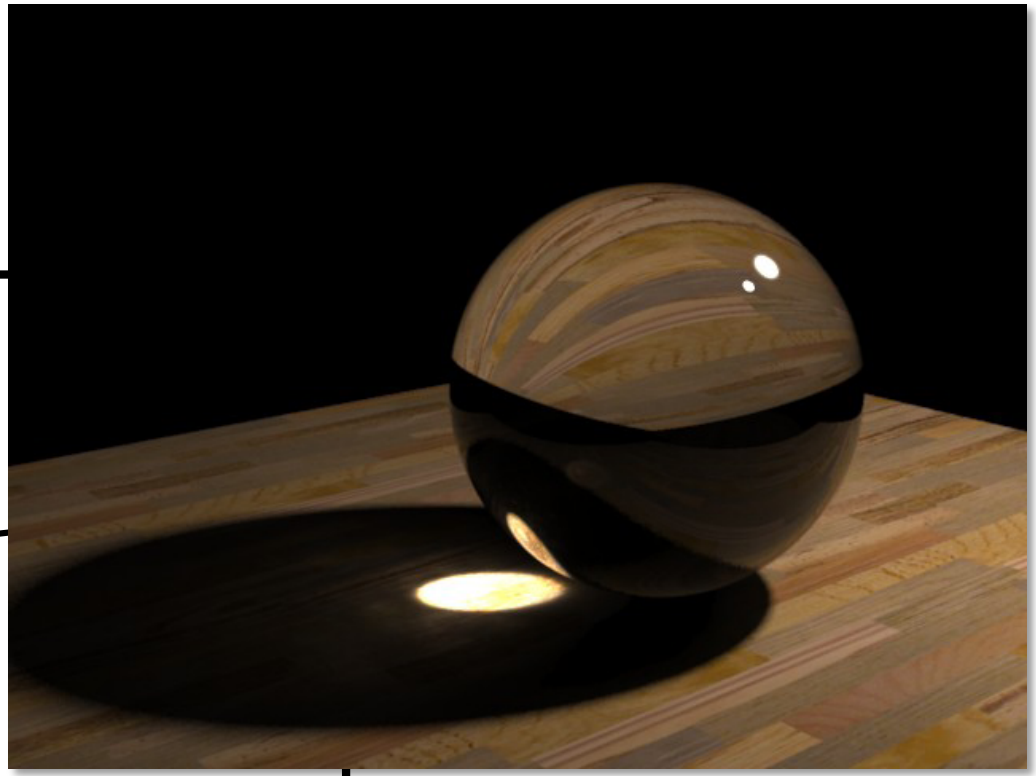
- Walter (1998)
  - Global Illumination: 28h
  - Interactive rendering

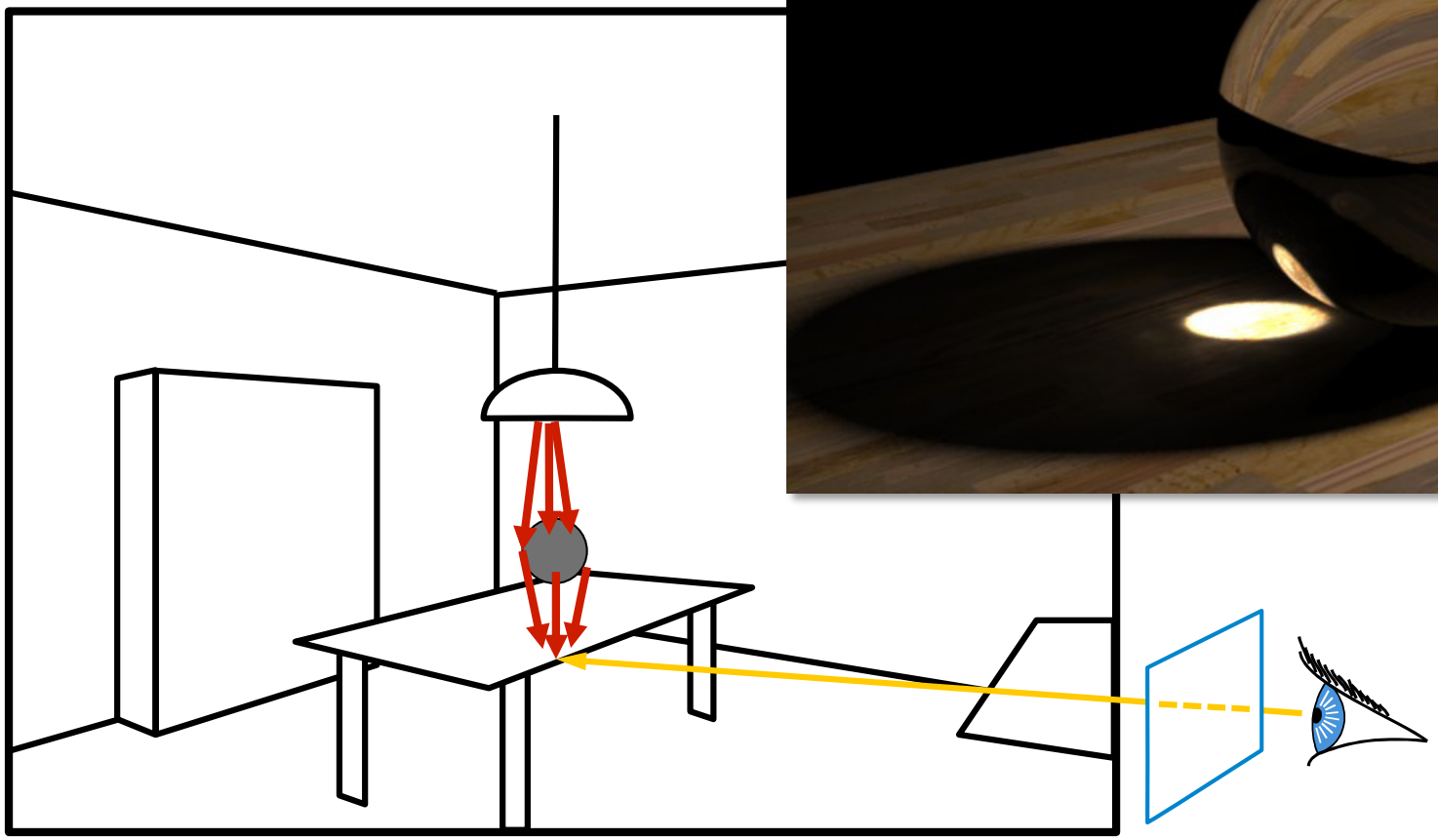
Results

Mitsuba (http://www.mitsuba-renderer.org/)

Caustics

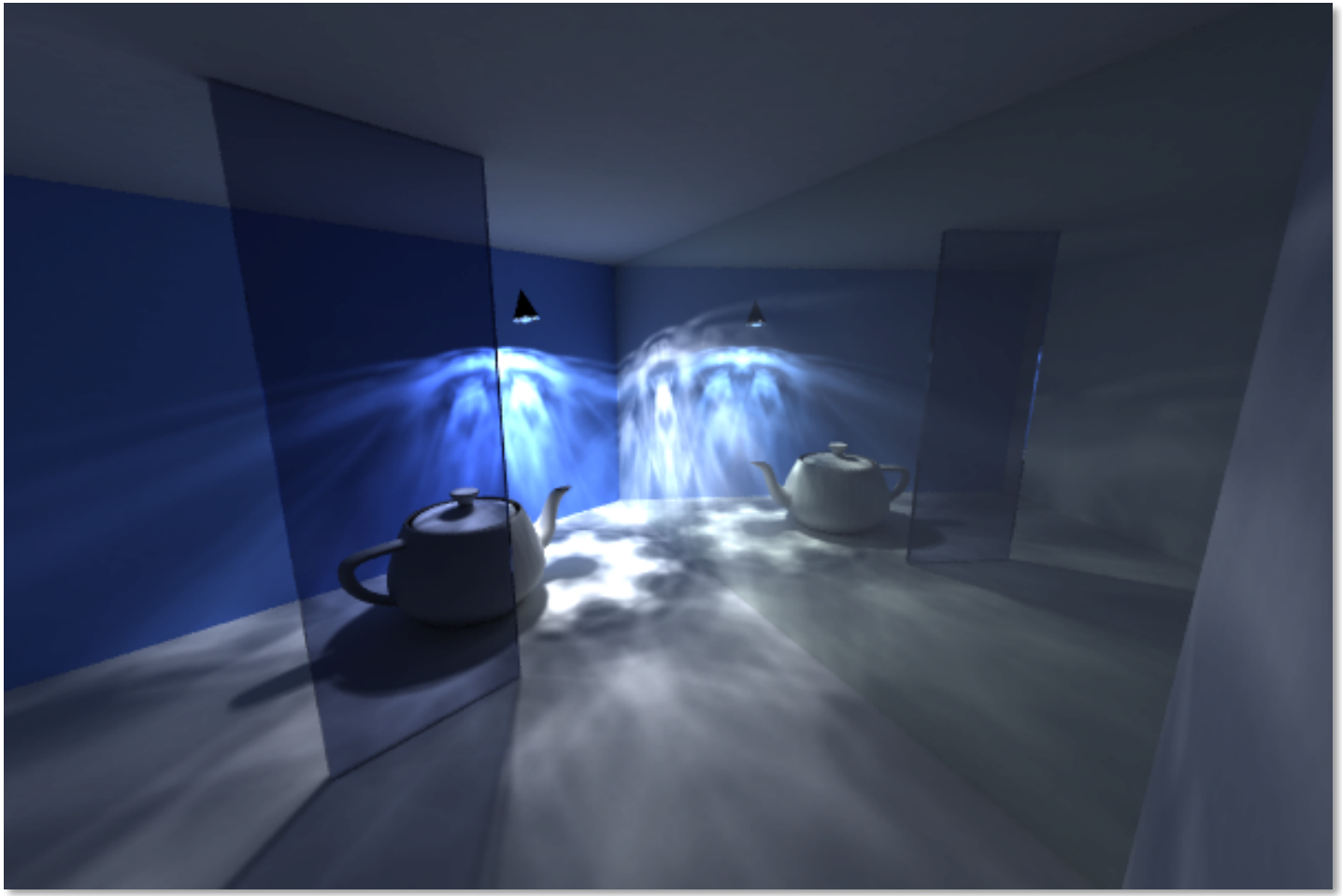
- Separate Photon map for refraction





Results

V-Ray 1.5 for 3ds Max



Results

5 millions photons from a single light source

Results

CyberMotion 3D-Designer

Results

Yafray : ray tracer open source with Photon Mapping, integrated with Blender.

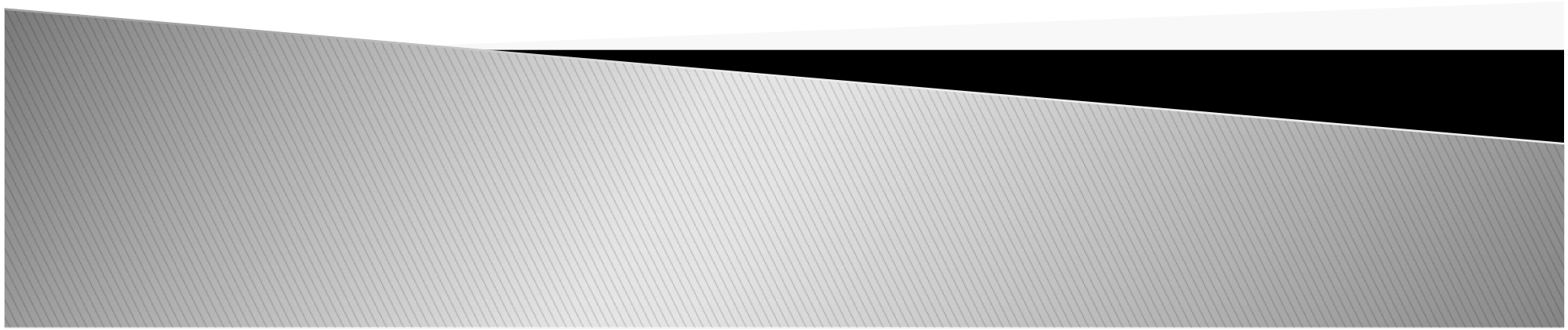
Results

http://en.wikipedia.org/wiki/POV-Ray

Photon Mapping: summary

- Pre-computation view-point independent
  - Storage on the surfaces
- Good representation for caustics
- Noisy: smoothing based on the samples
  - Reconstruction of the radiance function
- Can be coded in two passes with a ray-tracer
  - One pass for each direction

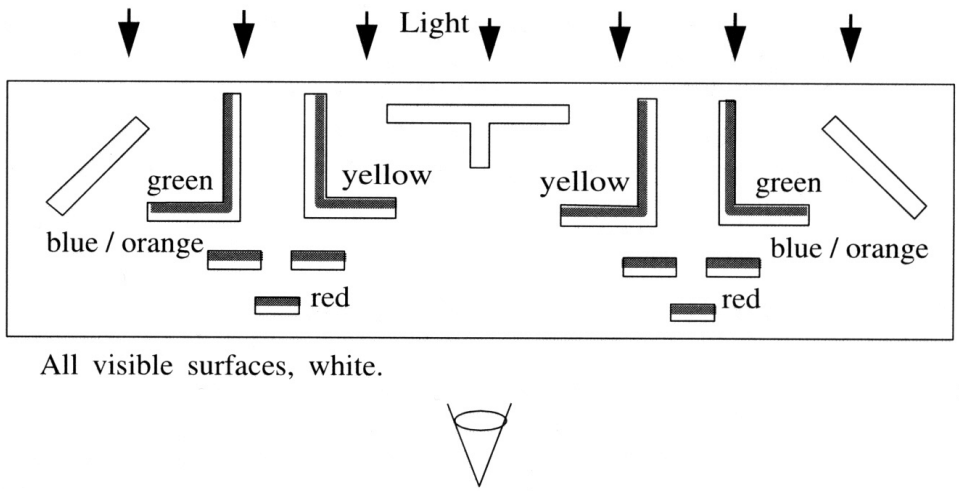
Radiosity



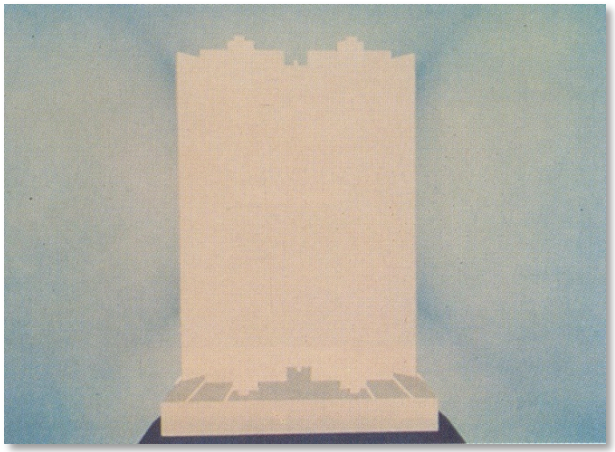
Radiosity

- Taking into account all inter-reflections

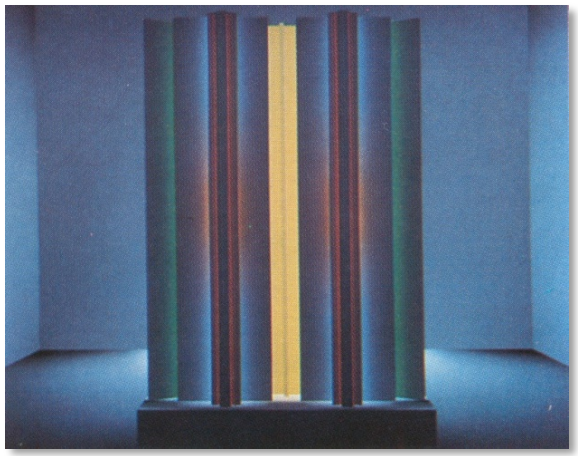
Sculpture by John Ferren



Photo



Ray-tracing



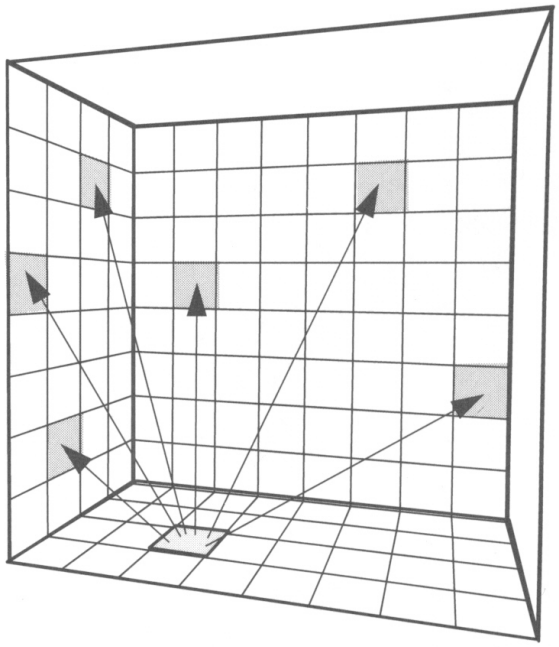
Radiosité

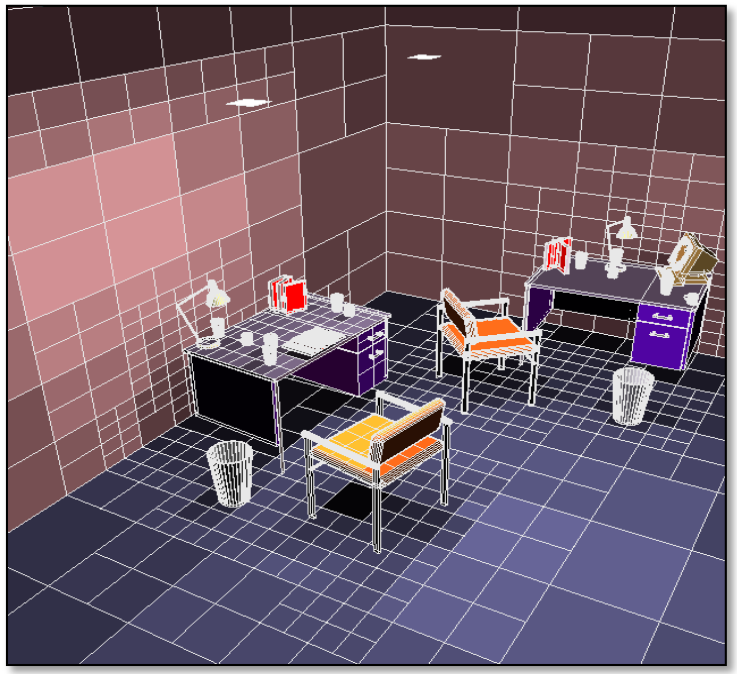
Radiosity methods [1984]

- Hypothesis: diffuse materials
- Radiance, BRDF… are **independent** from the **direction**
- ⇨ Simplifies the rendering equation
- Radiosity method:
  - Discretize this equation in object space
    - (viewpoint independent)
  - Solve the discretized equation
  - Render the scene using the illumination

L'équation de radiosité

- Environnement échantillonné sous la forme de patchs discrets, de taille finie, émettant et réfléchissant la lumière uniformément sur leurs surface (choix d'une base)





Simplification and discretization

$$L(\mathbf{x},\mathbf{d}) = E(\mathbf{x},\mathbf{d}) + \int \rho(\mathbf{x},\mathbf{d},\mathbf{d'})\ v(\mathbf{x},\mathbf{x'})\ G(\mathbf{x},\mathbf{x'})\ dA$$

- Simplified:

$$B(\mathbf{x}) = E(\mathbf{x}) + \rho_x \int B(\mathbf{x'})\ \underbrace{v(\mathbf{x},\mathbf{x'})\ G(\mathbf{x},\mathbf{x'})}_{\text{Form factor}}\ dA$$

- Discrete version:

$$B_i = E_i + \rho_i \sum_j F_{ji}\ B_j\ A_j / A_i$$

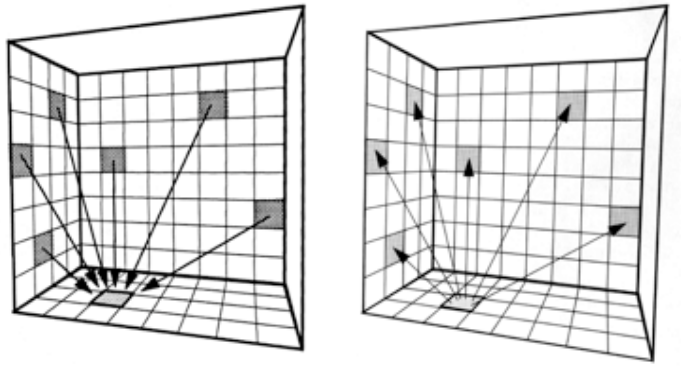
  - B_i, B_j: radiosity for patches i et j (in W/m2)
  - E_i: emission for patch i
  - F_{ji} **form factor,** characterizes proportion of energy leaving patch j arriving on i
  - A_i et A_j : areas for patches i and j

Matrix representation

- Grouping together all elements:

$$\begin{pmatrix} B_0 \\ \\ B_n \end{pmatrix} = \begin{pmatrix} E_0 \\ \\ E_n \end{pmatrix} + \begin{pmatrix} \rho_i F_{ji} \end{pmatrix} \begin{pmatrix} B_0 \\ \\ B_n \end{pmatrix} \quad \Leftrightarrow \quad B = E + MB$$

- Matrix equation, to solve iteratively
  - Relaxation methods (*gathering* / *shooting*)



Form factor

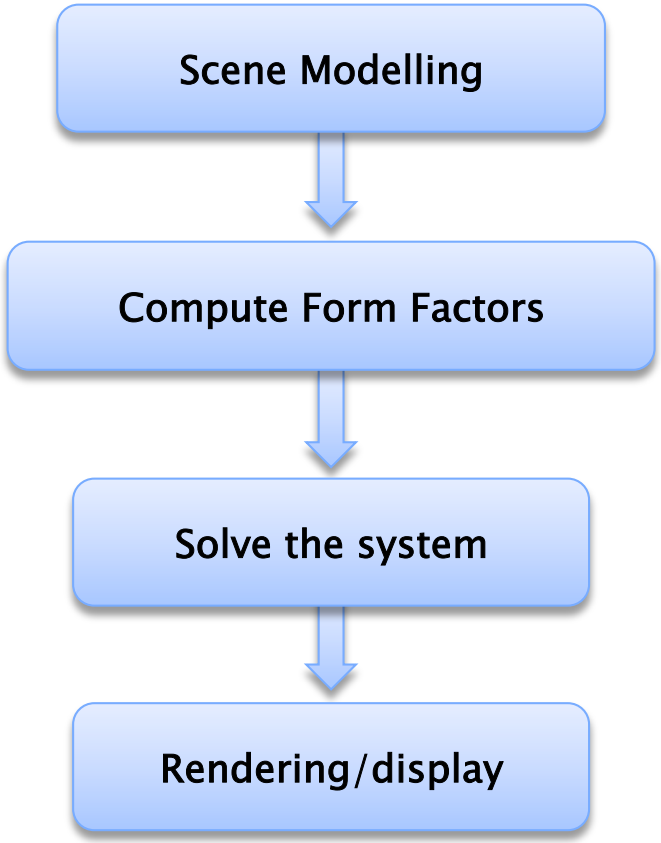
- Form factor F_{ij} from a patch A_i towards a patch A_j :

$$F_{ij} = \int_{A_i} \int_{A_j} v(x,x') \frac{\cos(\theta)\cos(\theta')}{\pi r^2} dx dx'$$

- **Problem**: computing this integral (4D). No analyticl solution
  - ⇨Approximated solutions: projection on a hemisphere or a hemicube.

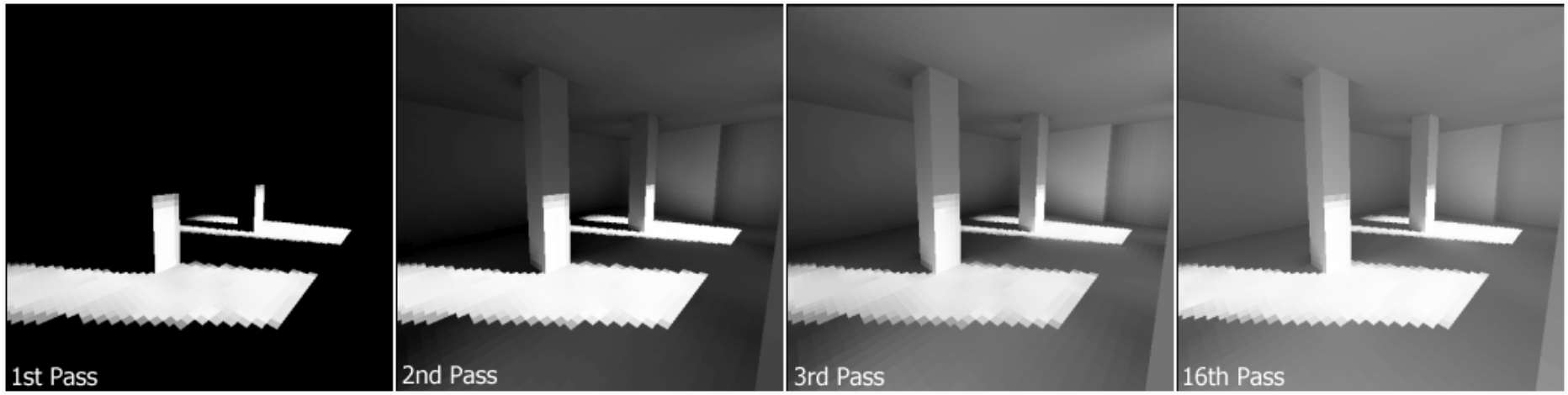
Solving radiosity

- Pipeline for computing global illumination :



Solving radiosity

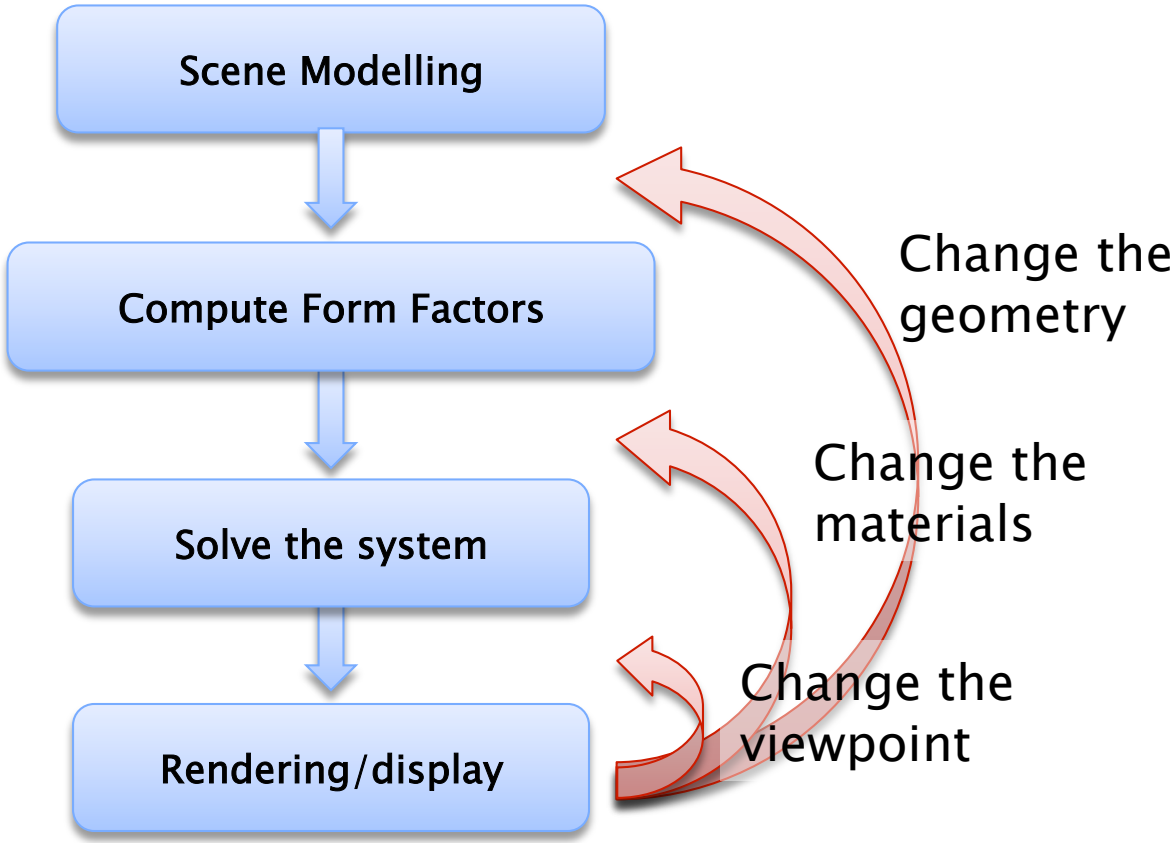
- Iterative solving:



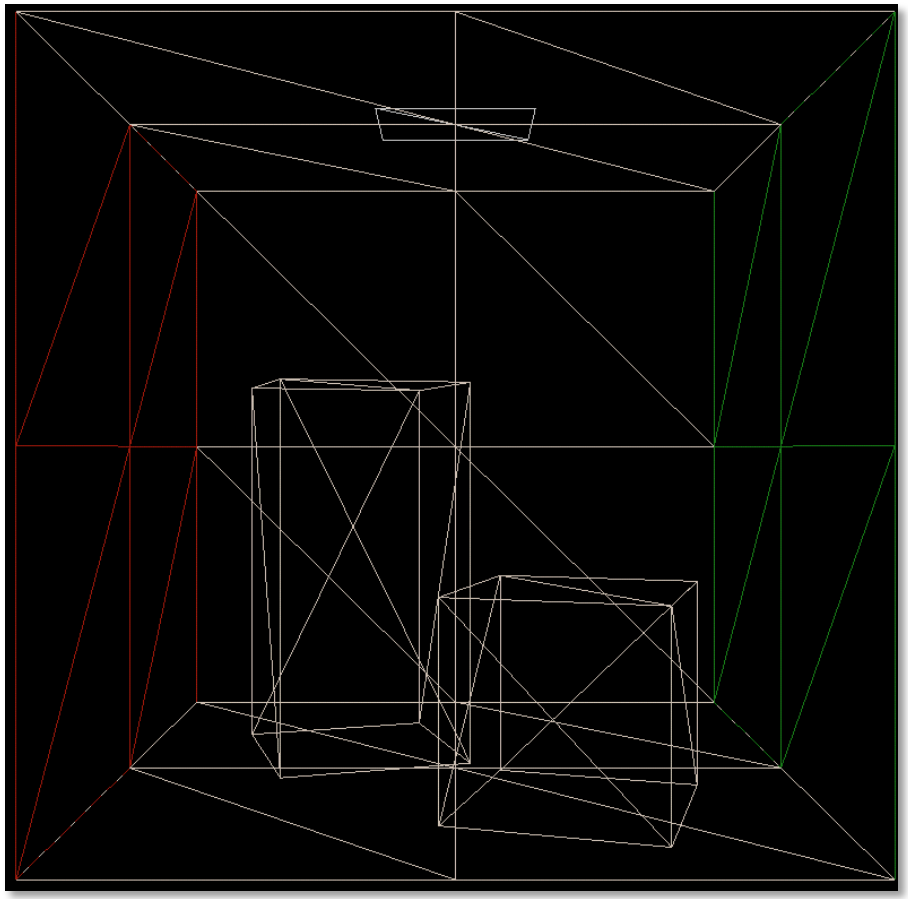
Question – 1 mn

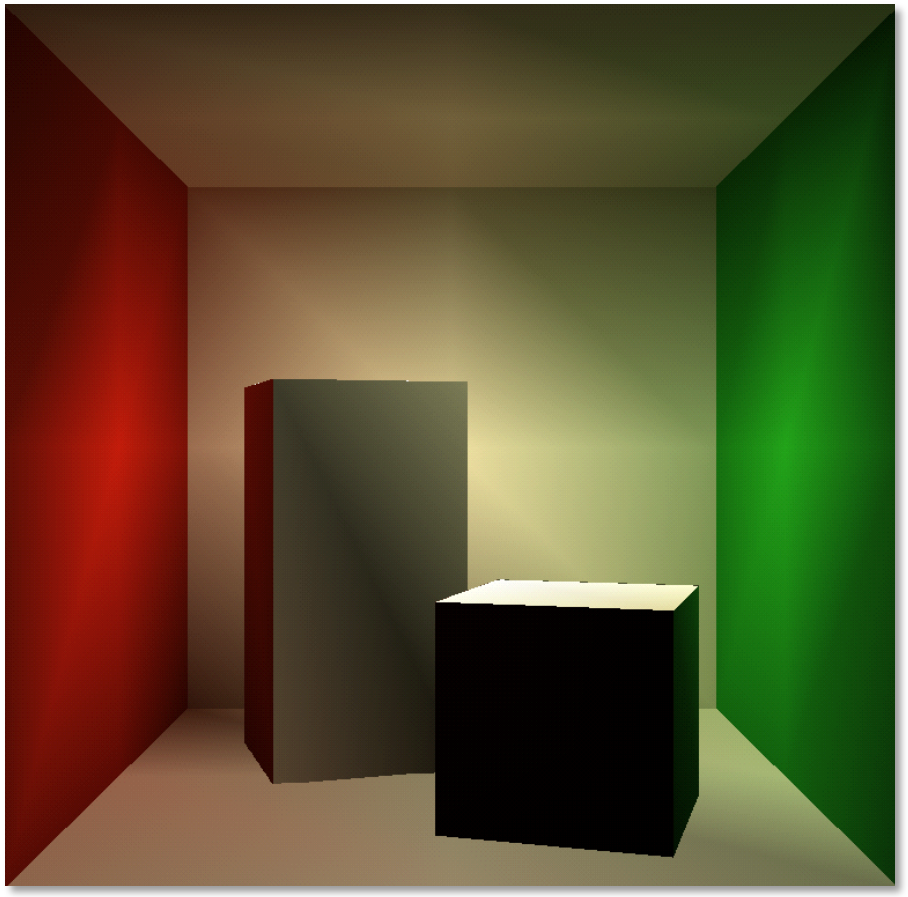
- What must be recomputed if something changes in the scene?
  - Geometry
  - Reflectance properties
  - Viewpoint

Solving radiosity

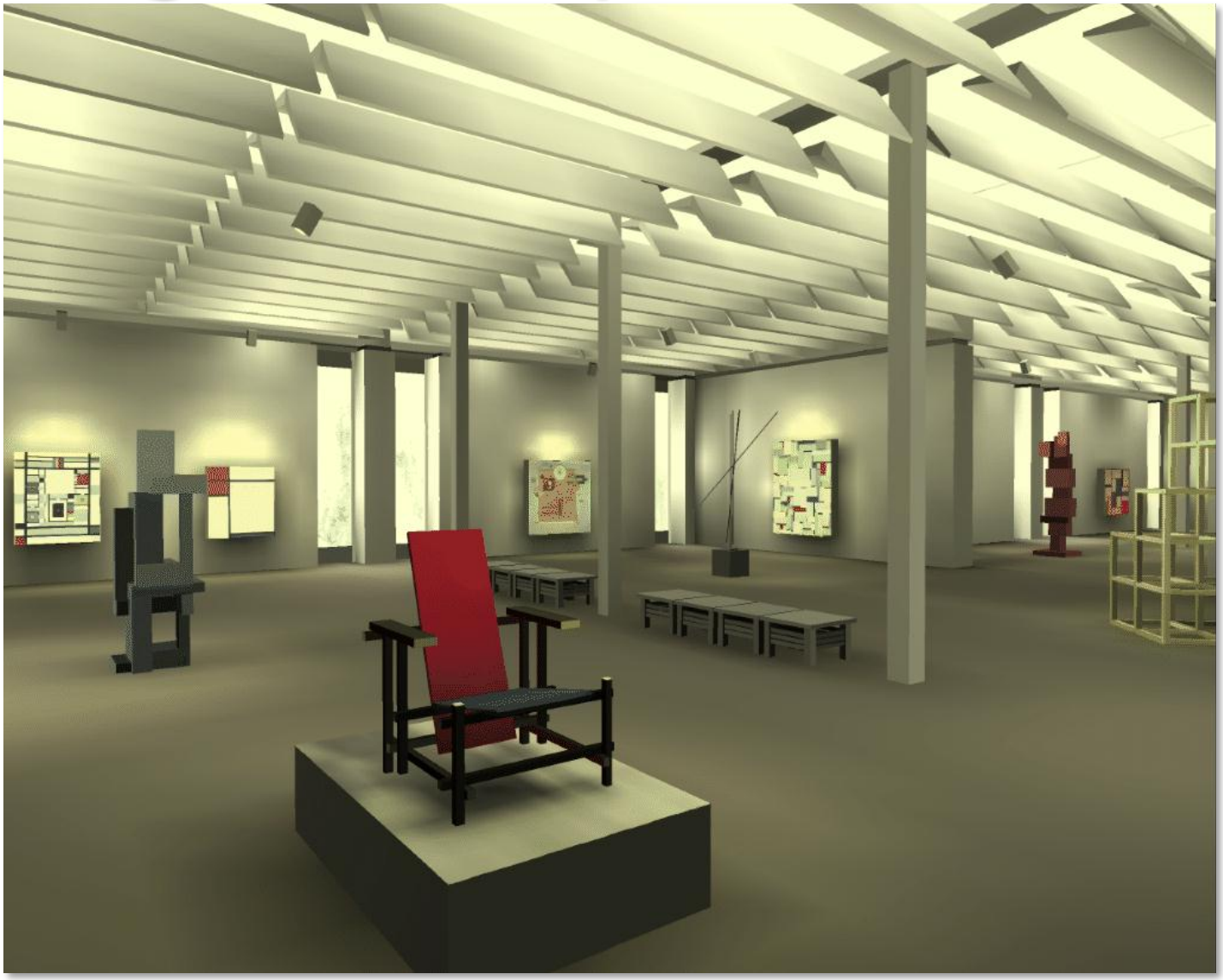


Solving radiosity





Solving radiosity



Museum simulation. Cornell University. 50,000 patches.

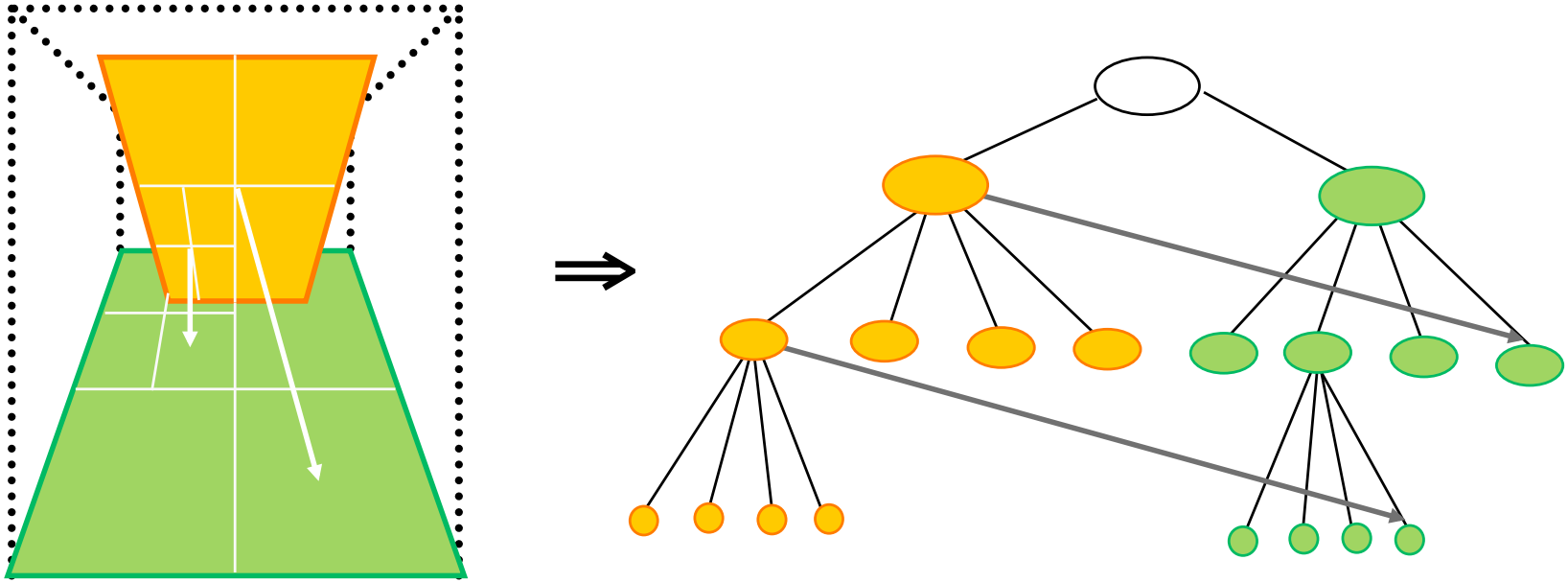
Radiosity: pros

- Computations are viewpoint independent
- Okay for complex scenes
- Solving light exchanges
  - Interactive rendering

Radiosity: cons

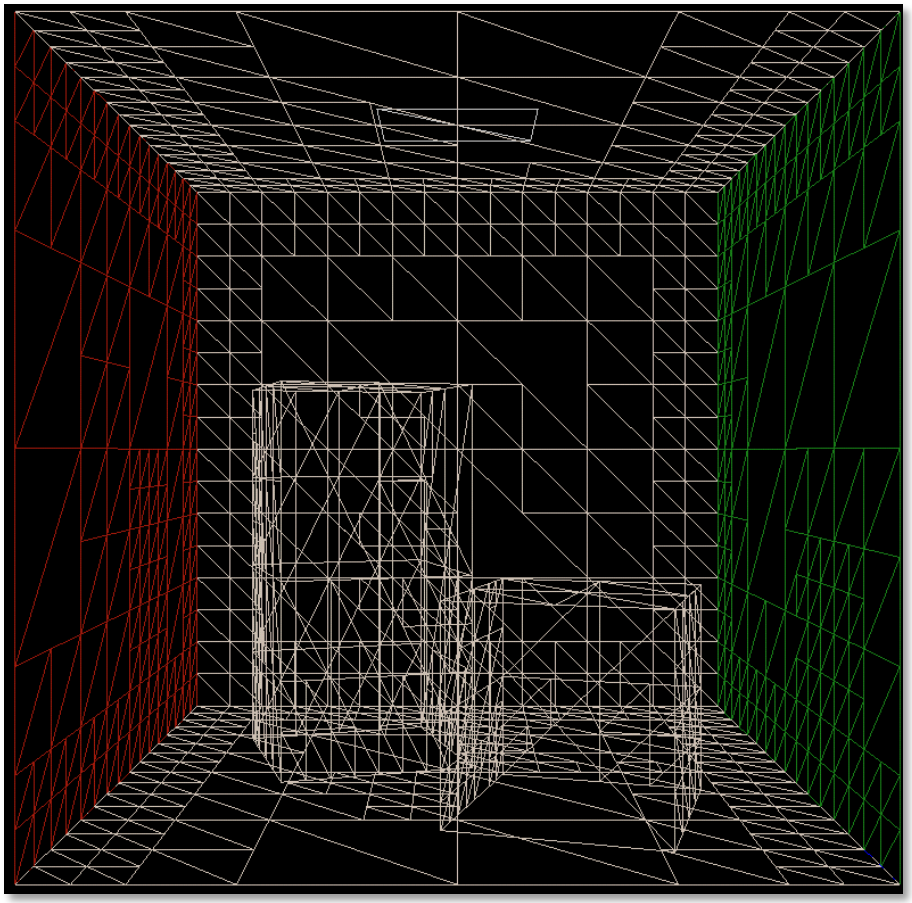
- Memory cost
- Only diffuse materials
  - « final gather » using Ray-Tracing
- Meshing
  - Discontinuity mesh
- Long pre-computations
  - Possible speed-up: hierarchical radiosity

Hierarchical radiosity [Hanrahan91]

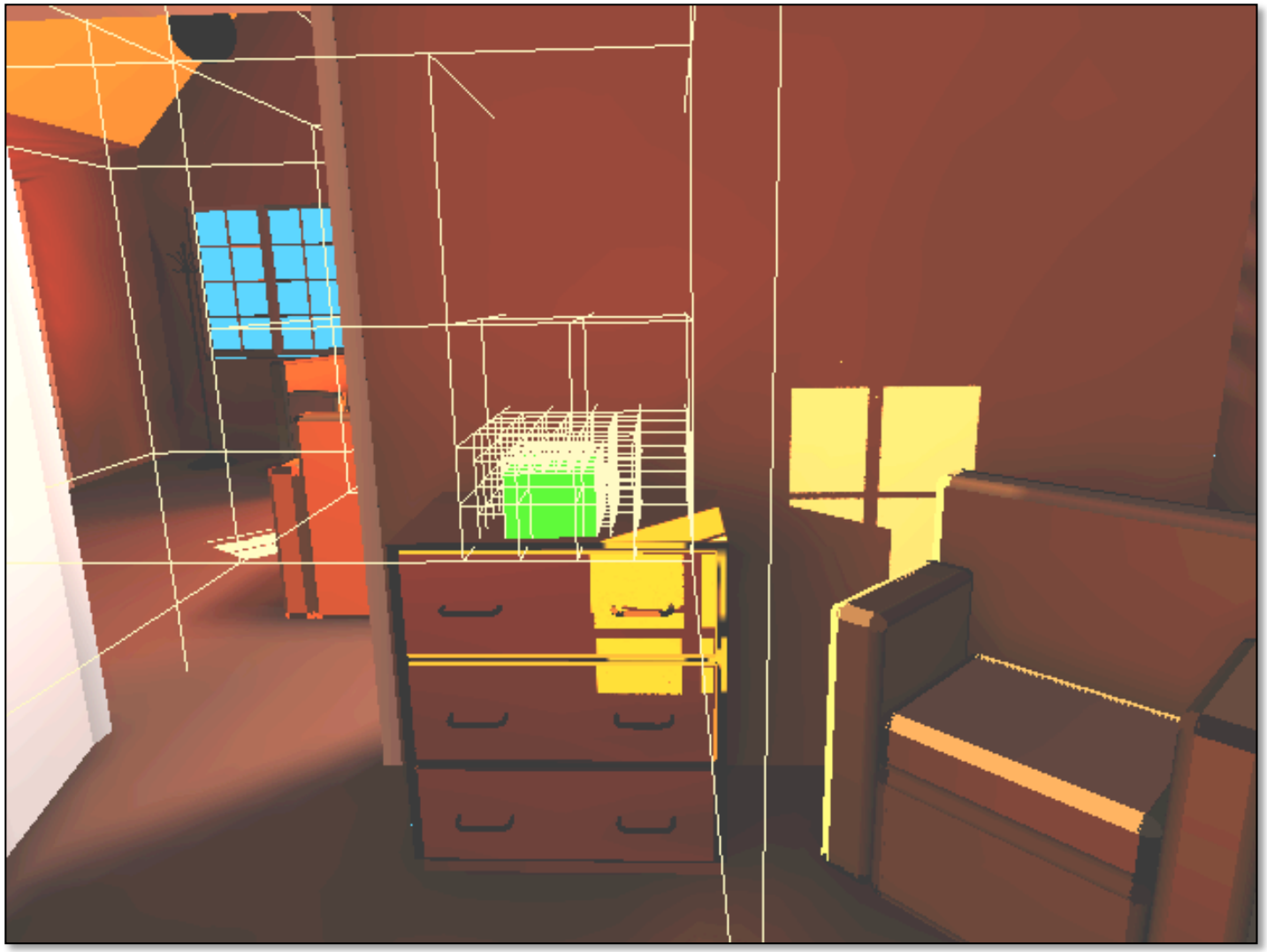


- Computation at different hierarchical levels
- Push-pull

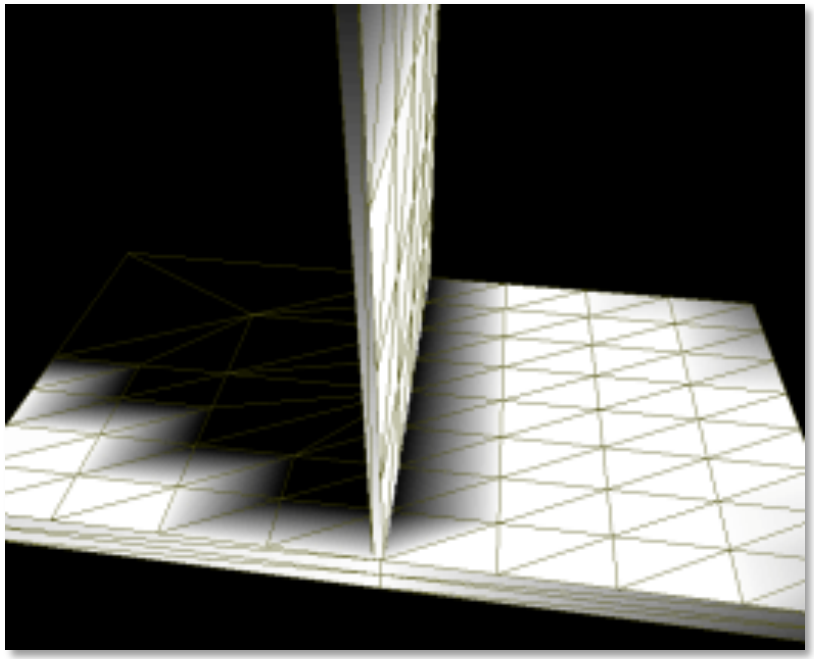
Hierarchical radiosity



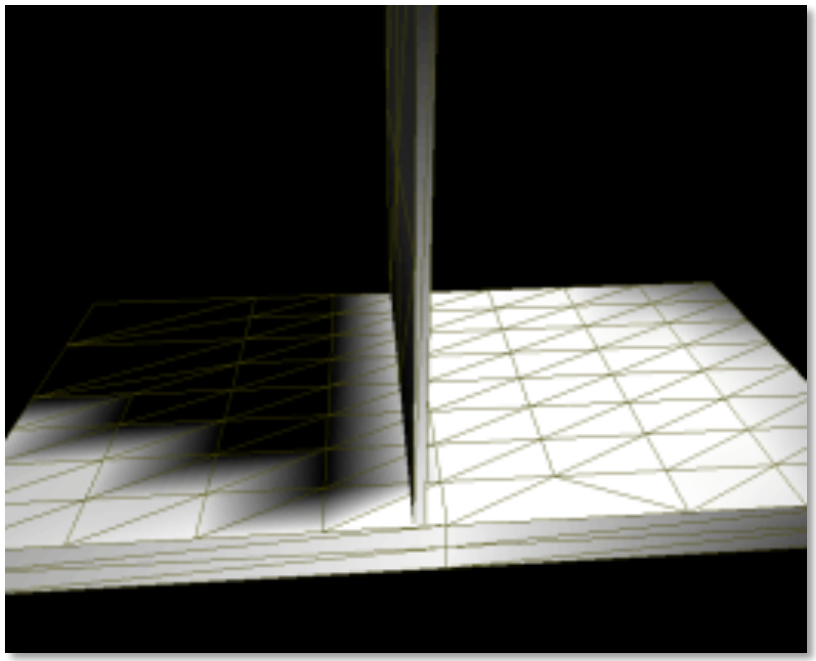
Hierarchical radiosity



Mesh quality

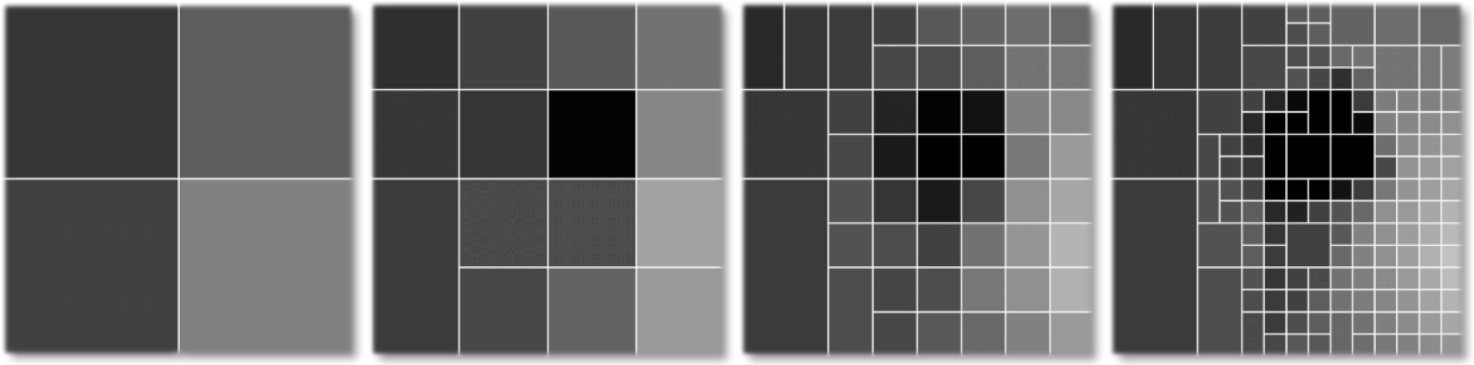


Shadow leak

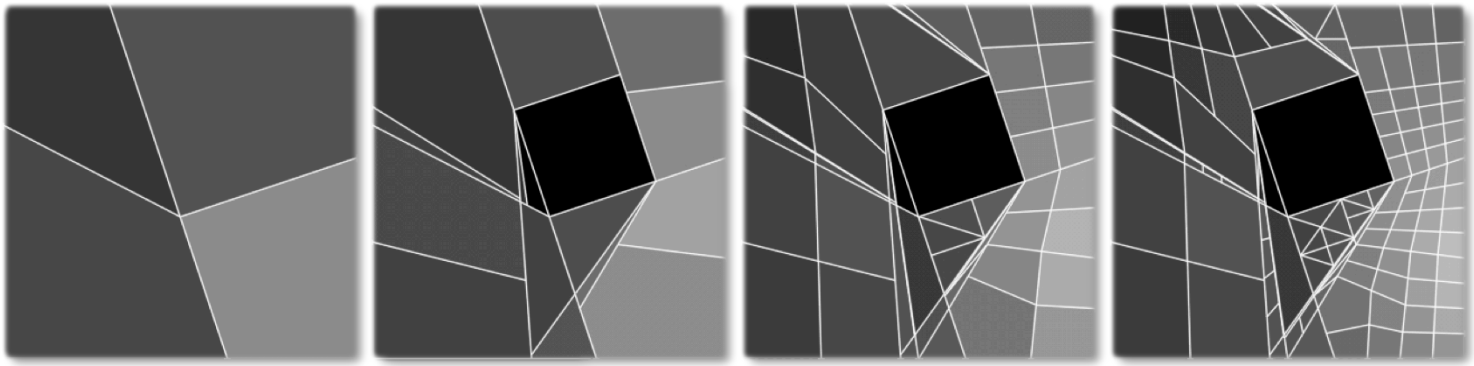


Light leak

Discontinuity meshing



Regular subdivision



Discontinuity meshing

Discontinuity meshing

Discontinuity meshing

Discontinuity meshing

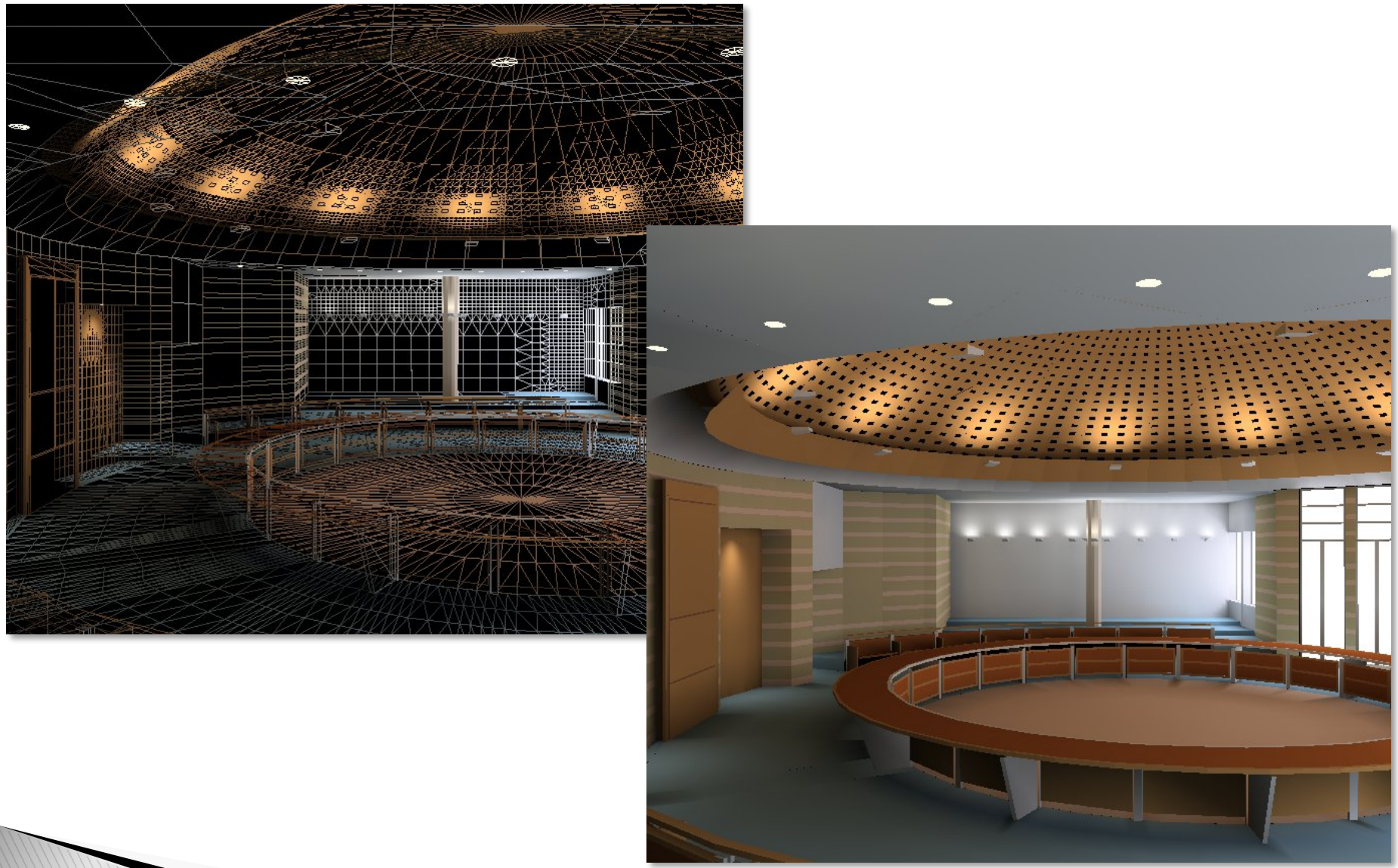
Résultats et comparaison

Rendu Scan-line.
(3DS MAX)

Rendu en radiosité et
lancer de rayons.

Results and comparison

Results and comparison



Results and comparison

Results and comparison

Radiosity today

- Used by architects (Lightscape)
- Used to precompute diffuse lighting for some video games (light maps)

- Not an active research topic anymore
  - Monte-Carlo is more generic
  - But pre-computed radiance transfer is similar (used e.g. in Max Payne 2)