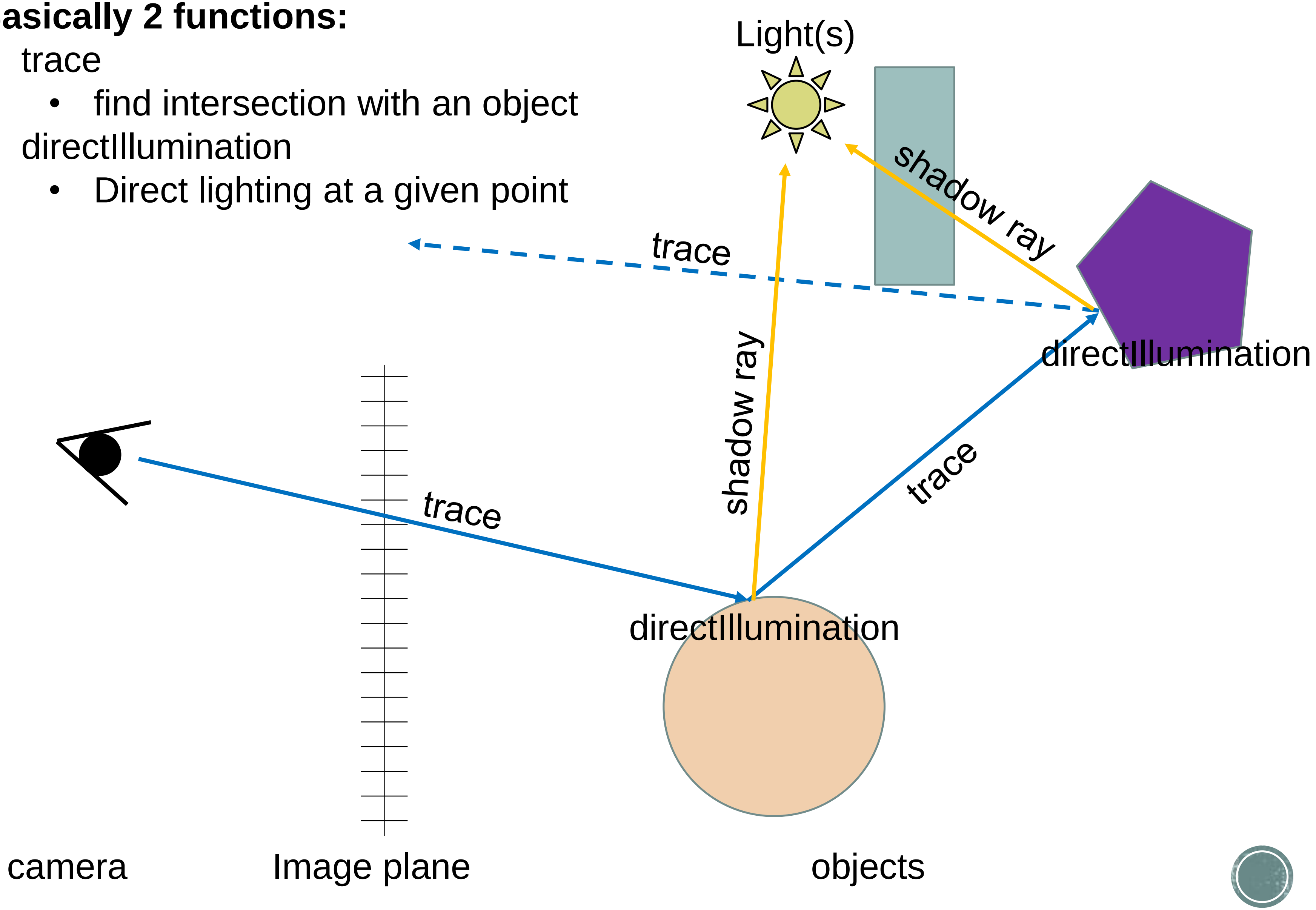
Advanced image synthesis

Romain Vergne – 2014/2015

Ray tracing

Basically 2 functions:

- trace
  - find intersection with an object
- directIllumination
  - Direct lighting at a given point



Ray tracing

- color trace(ray) {
  - hit = intersectScene(ray)
  - if(hit) {
    - color = directIllumination(hit)
    - if hit is reflective
      - color += c_refl * trace(reflected ray)
    - if hit is transmissive
      - color += c_trans * trace(refracted ray)
  - } else
    - color = background_color
  - return color
- }

Ray tracing

- color directIllumination(hit) {
  - color = (0,0,0)
  - for each light L {
    - T = cast shadow ray to L
    - if hit is not shadowed by L
      - color += Ambient+diffuse+specular terms(L,hit)
  - }
  - return color
- }

Physics → Maths

- **B**idirectionnal
- **R**eflectance
- **D**istribution
- **F**unction

$$f(\mathbf{l}, \mathbf{v})$$

$$L_o(\mathbf{v}) = \int_\Omega f(\mathbf{l}, \mathbf{v}) \otimes L_i(\mathbf{l})(\mathbf{n} \cdot \mathbf{l}) d\omega_i$$

Outgoing radiance

BRDF

Ingoing radiance

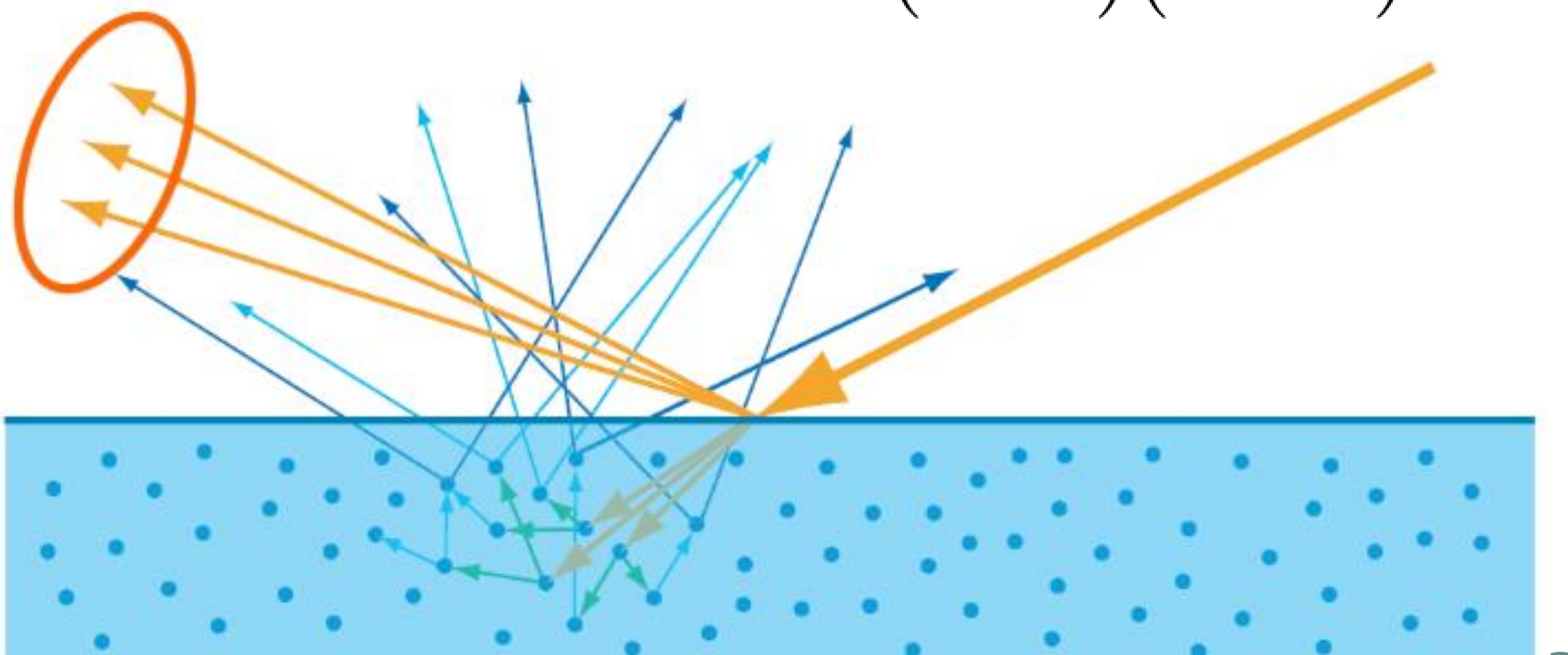
Surface orientation

Reflectance equation

Microfacet theory

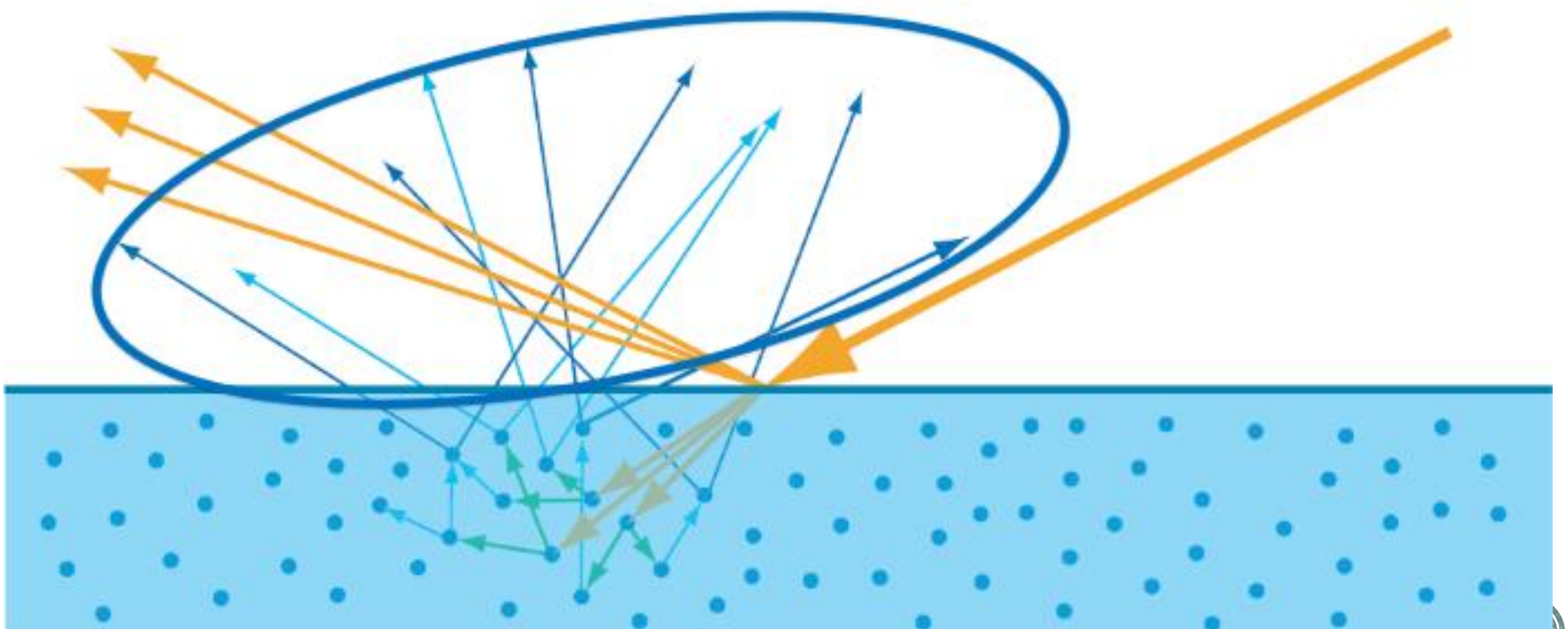
- Surface reflection (specular term)

$$f(\mathbf{l},\mathbf{v}) = \frac{F(\mathbf{l},\mathbf{h})G(\mathbf{l},\mathbf{v},\mathbf{h})D(\mathbf{h})}{4(\mathbf{n}\cdot\mathbf{l})(\mathbf{n}\cdot\mathbf{v})}$$



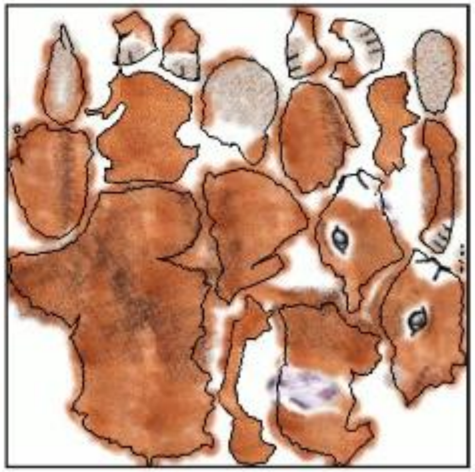
Microfacet theory

- Subsurface reflection (diffuse term)
  - Constant: $f_{\text{Lambert}}(\mathbf{l}, \mathbf{v}) = \dfrac{\mathbf{c}_{\text{diff}}}{\pi}$

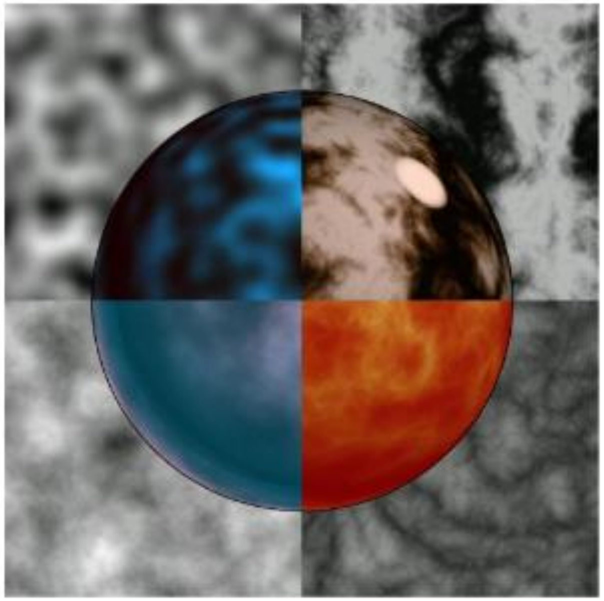


Textures

- From data
  - Colors, coefs, normals stored in 2D images

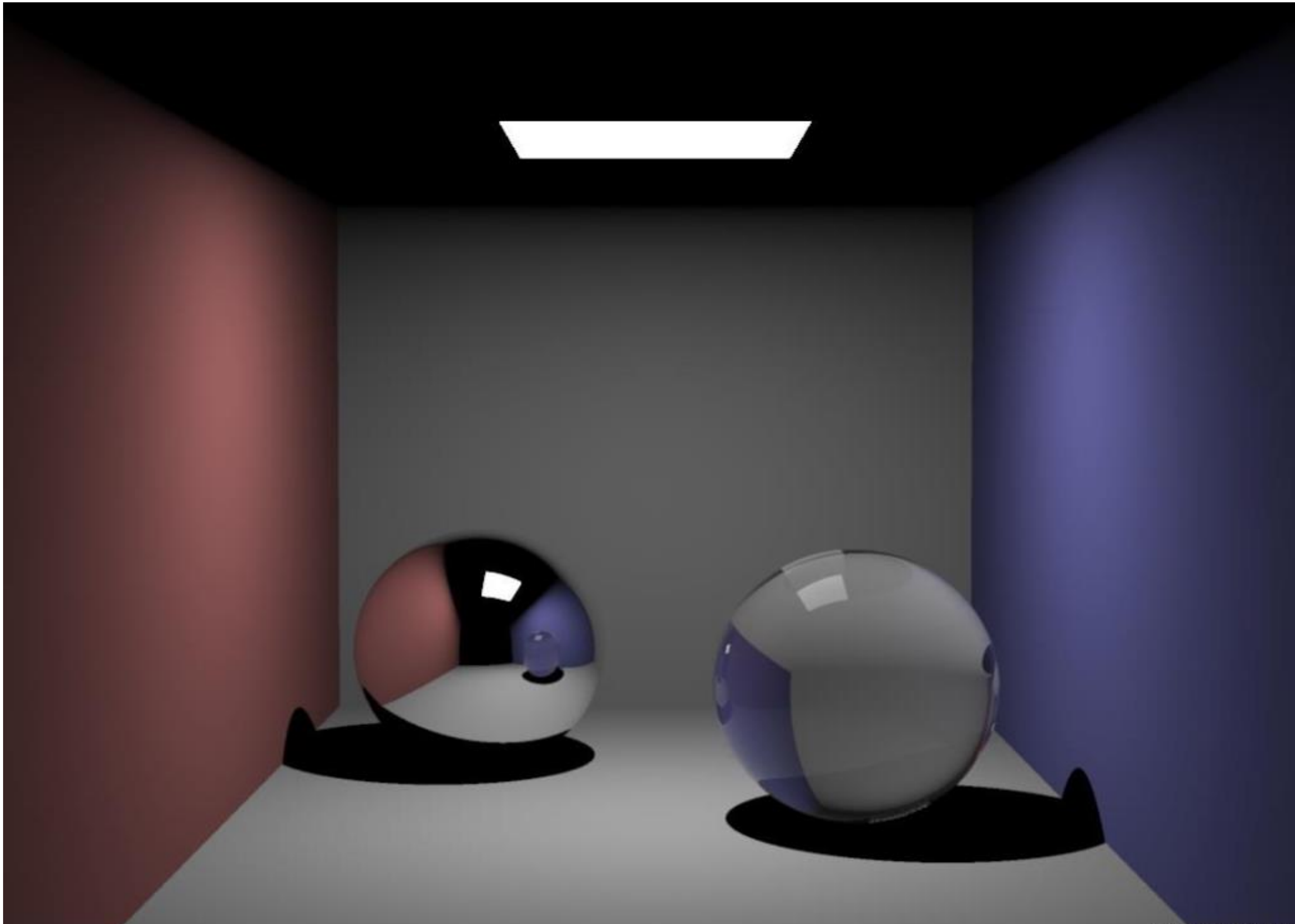


- Procedural shader
  - Little program that compute info at a given position

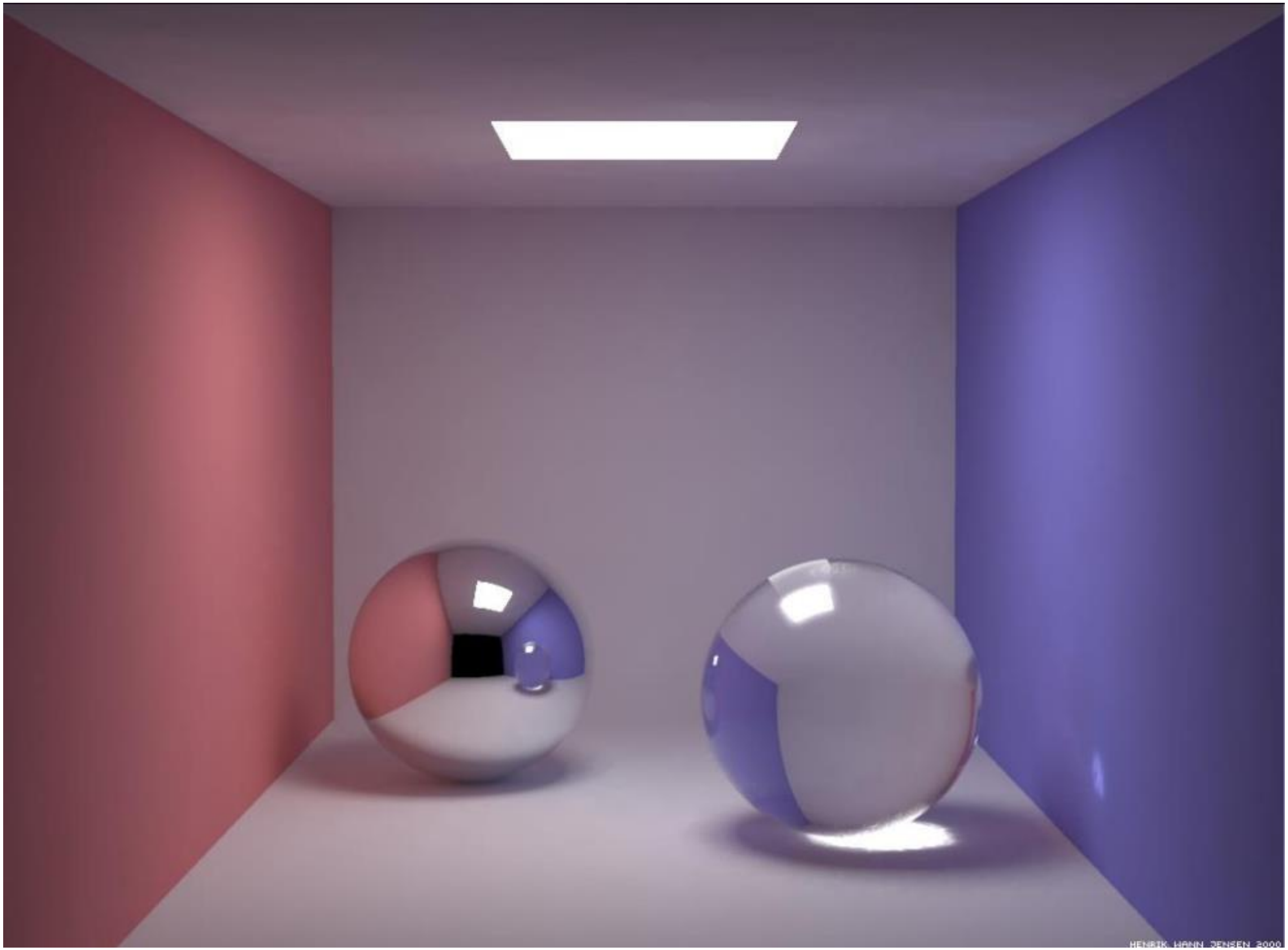


Textures

What is missing?

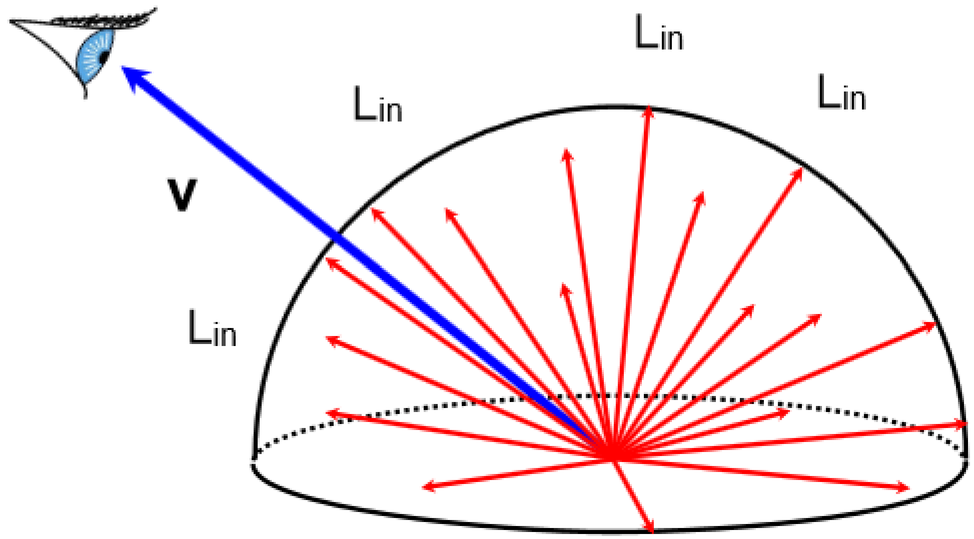


What is missing?



Reflectance equation

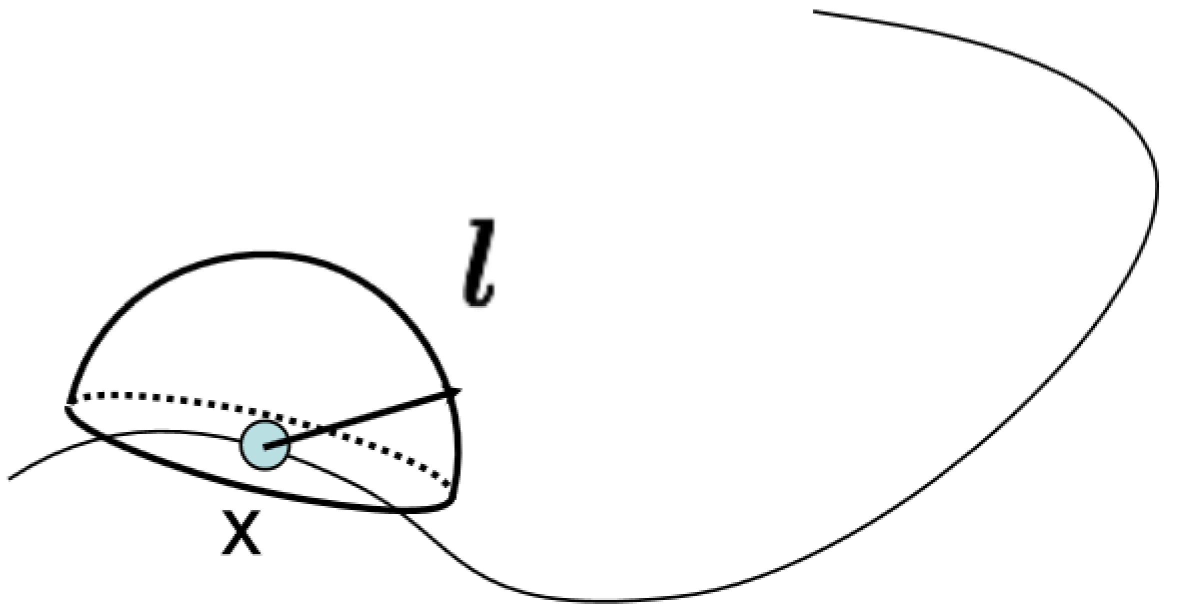
$$L_o(\mathbf{v}) = \int_{\Omega} f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l}) d\omega i$$



- Where does Lin come from?

Reflectance equation

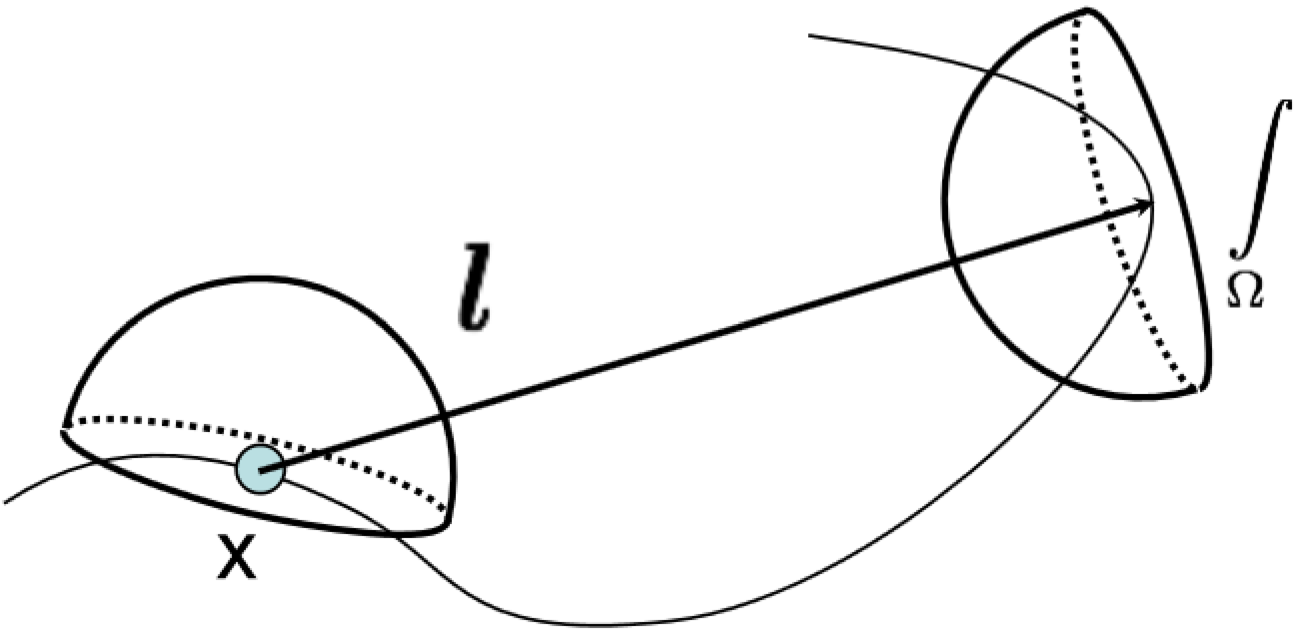
$$L_o(\mathbf{v}) = \int_\Omega f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l}) d\omega i$$



- Where does Lin come from?

Reflectance equation

$$L_o(\mathbf{v}) = \int_\Omega f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l}) d\omega i$$

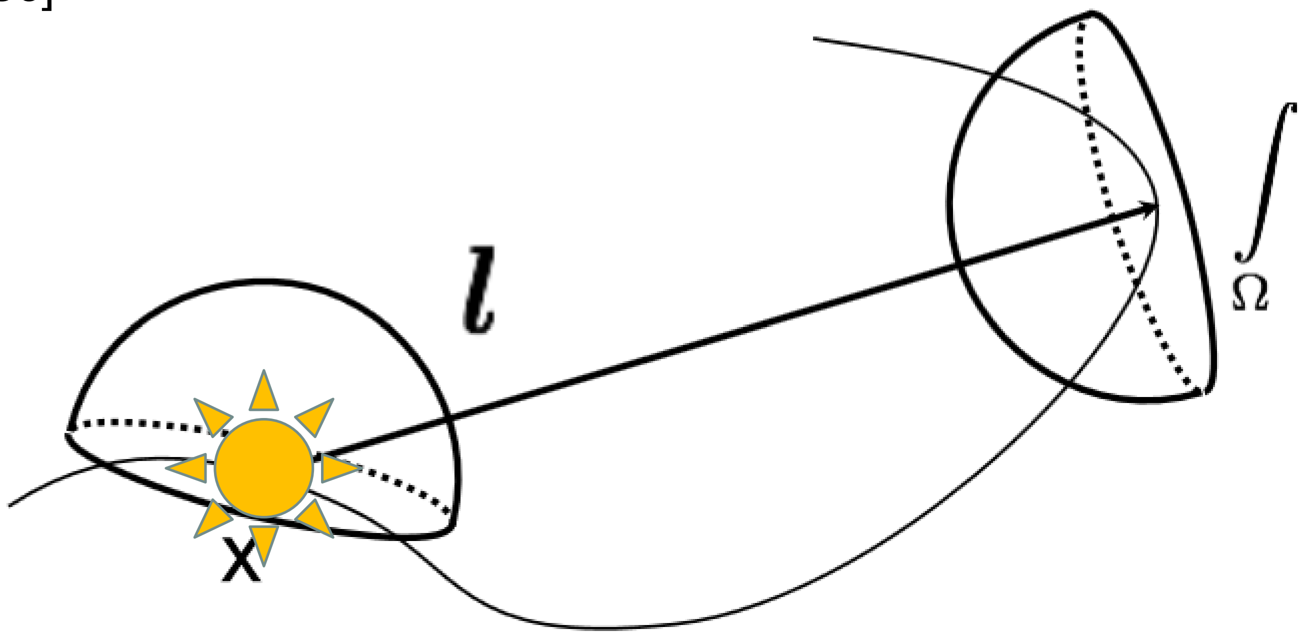


- Where does Lin come from?

Rendering equation

$$L_o(\mathbf{v}) = \int_\Omega f(\mathbf{l},\mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l}) d\omega i + E_{out}$$

[Kajiya 1986]



- Where does Lin come from?

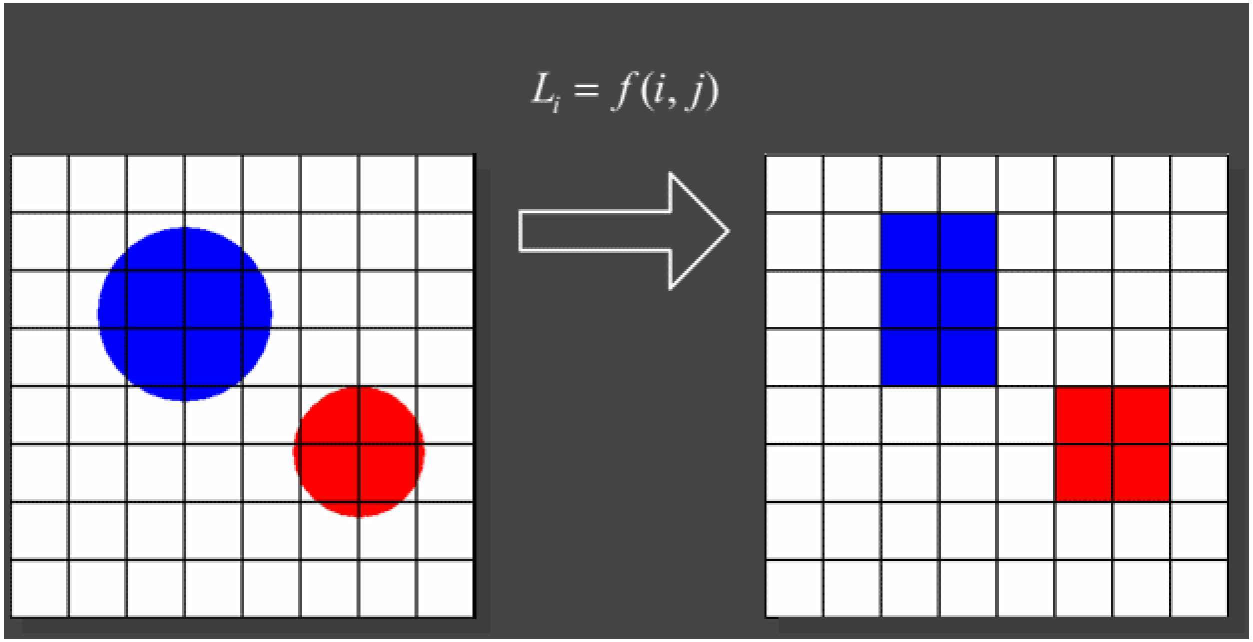
Rendering equation

$$L_o(\mathbf{v}) = \int_\Omega f(\mathbf{l},\mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l}) d\omega i + E_{out}$$

- Analytic solution usually impossible
- Lots of ways to solve it approximately:
  - e.g. ray-tracing – but approximation only...
  - How can we do better?

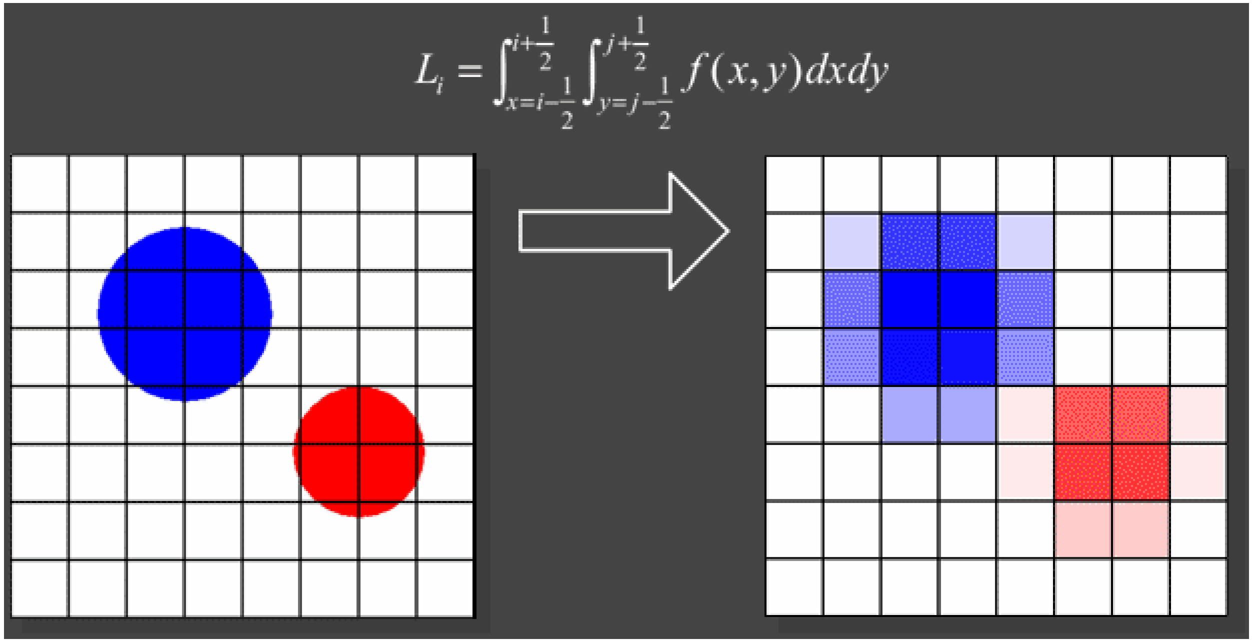
Soft contours (antialiasing)

- One ray per pixel
  - True or false…



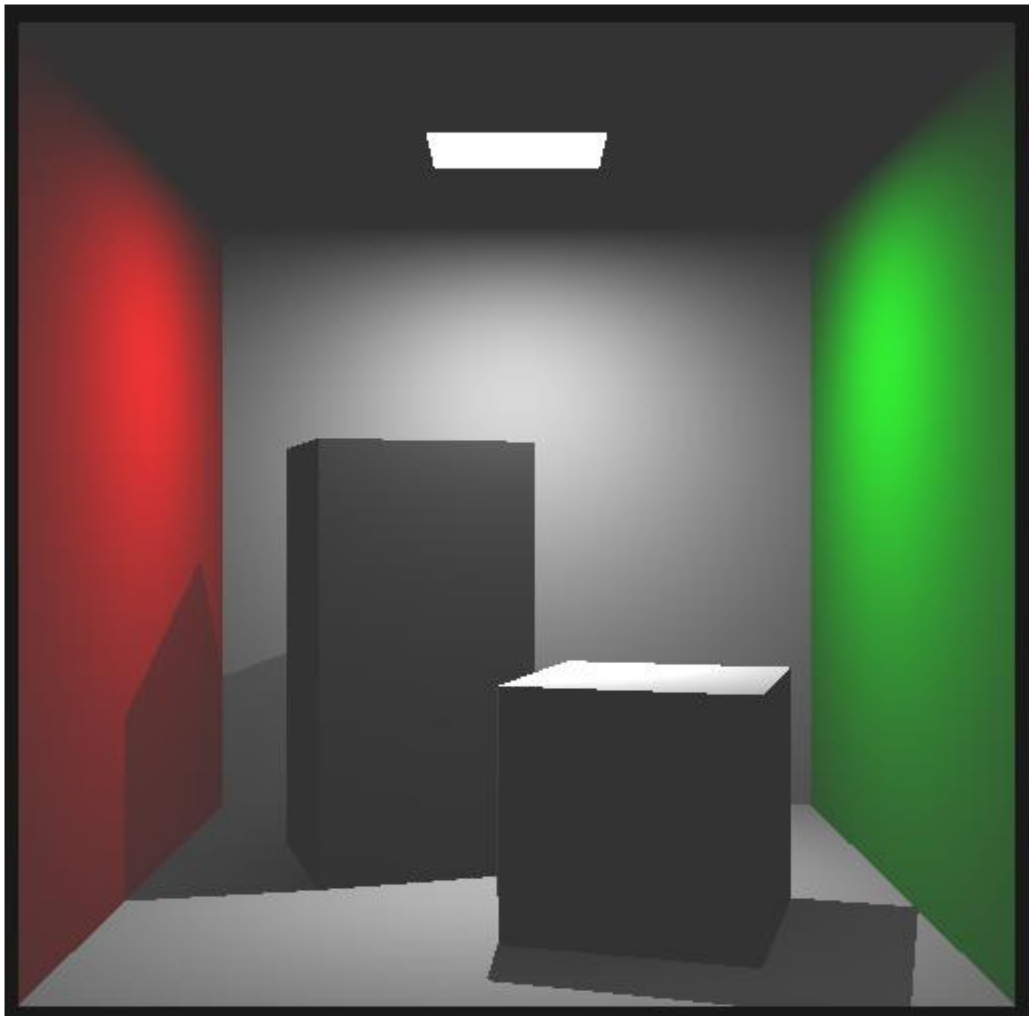
Soft contours (antialiasing)

- One ray per pixel
  - True or false…
- Multiple rays per pixel
  - Average result (sum + divide by number of rays)



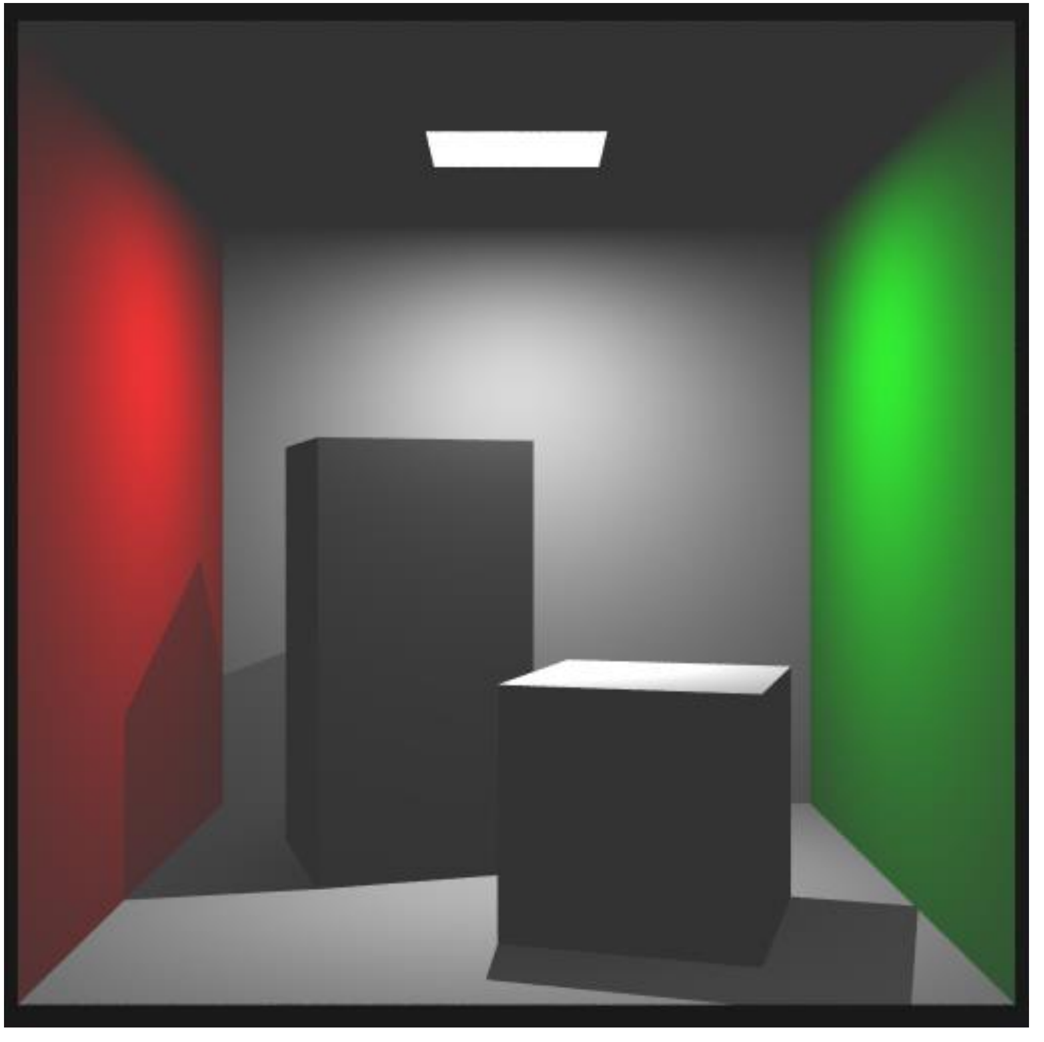
Soft contours (antialiasing)

No antialiasing



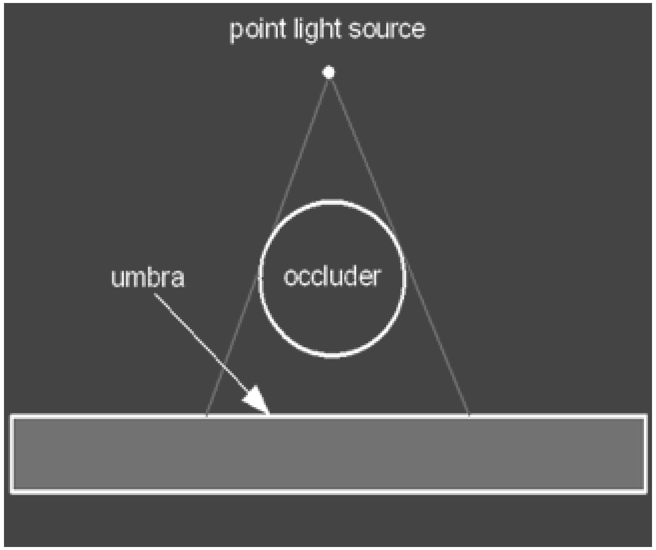
Soft contours (antialiasing)

16 samples / pixel



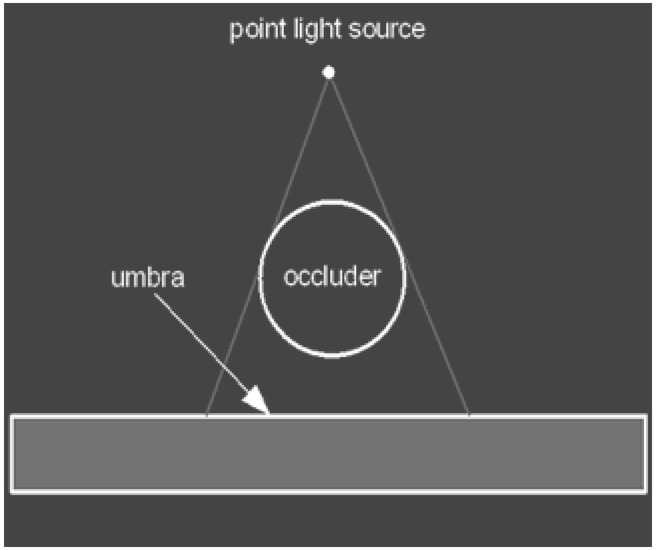
Soft shadows

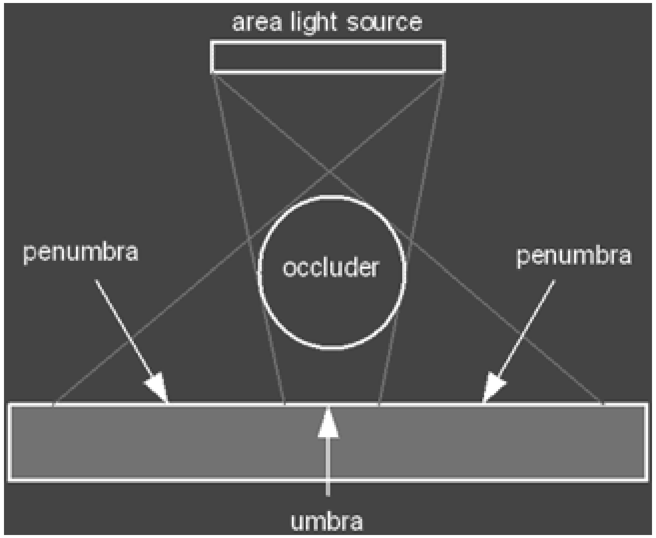
- One sample per light
  - Hard (or noisy) shadows



Soft shadows

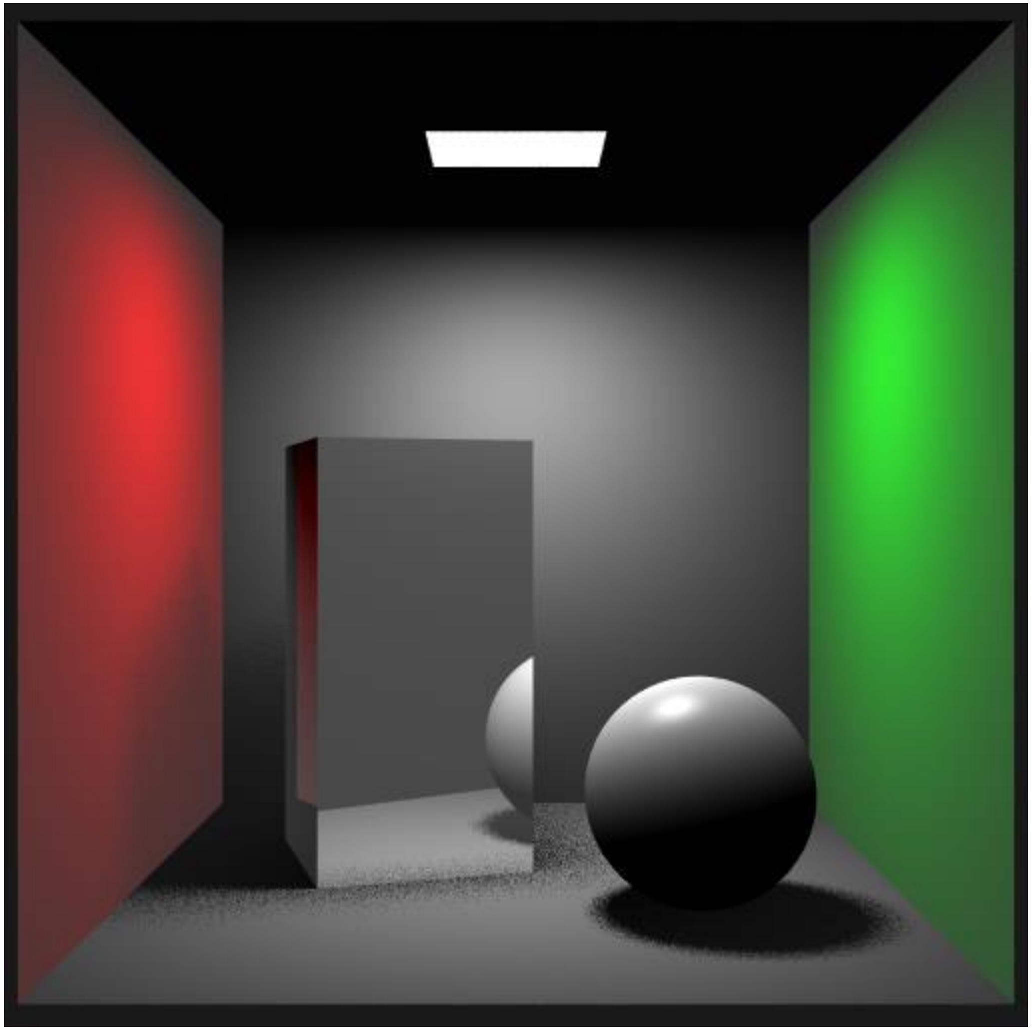
- One sample per light
  - Hard (or noisy) shadows
- Multiple samples per light
  - Soft shadows





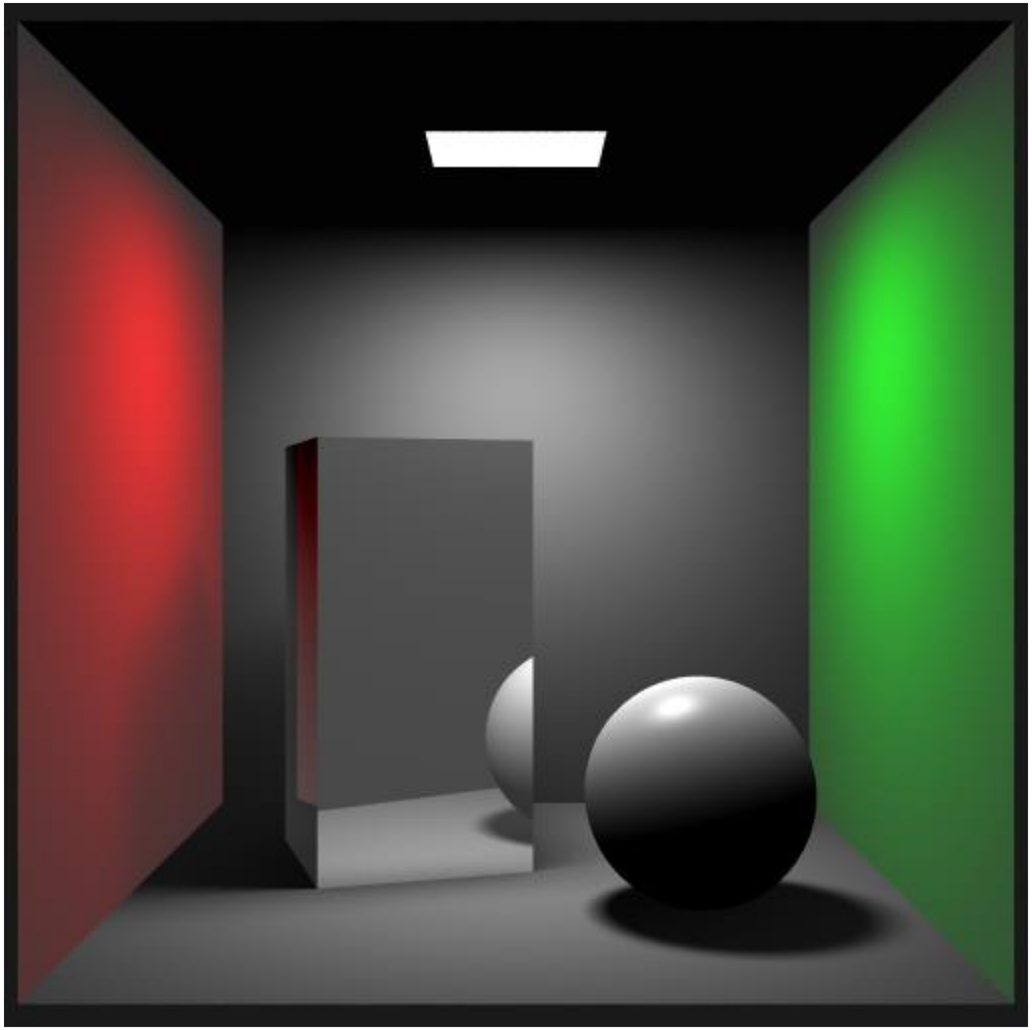
Soft shadows

1 (random) sample / light



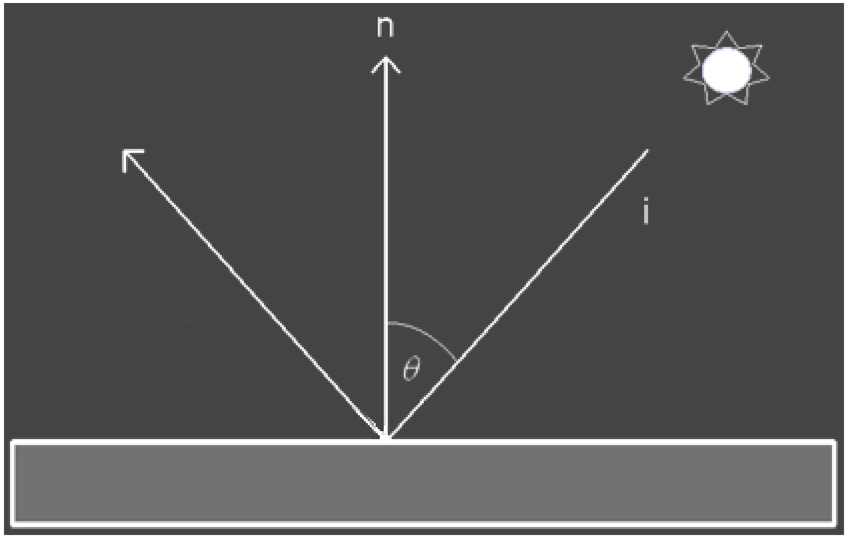
Soft shadows

1 (random) sample / light



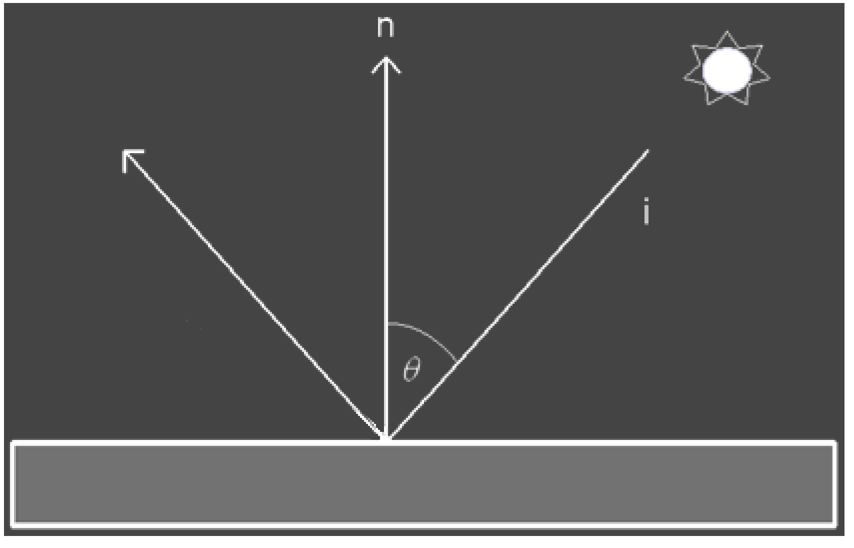
Soft reflections (glossiness)

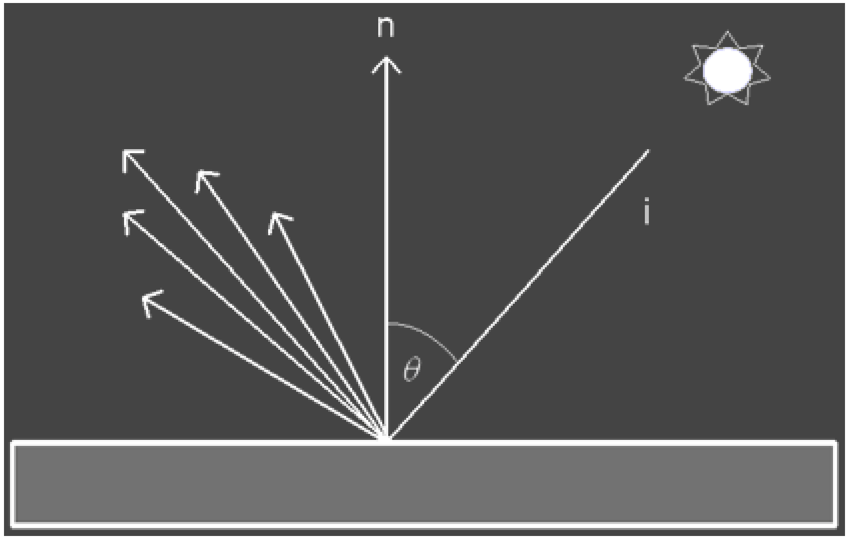
- One sample per hemisphere
  - Specular reflections



Soft reflections (glossiness)

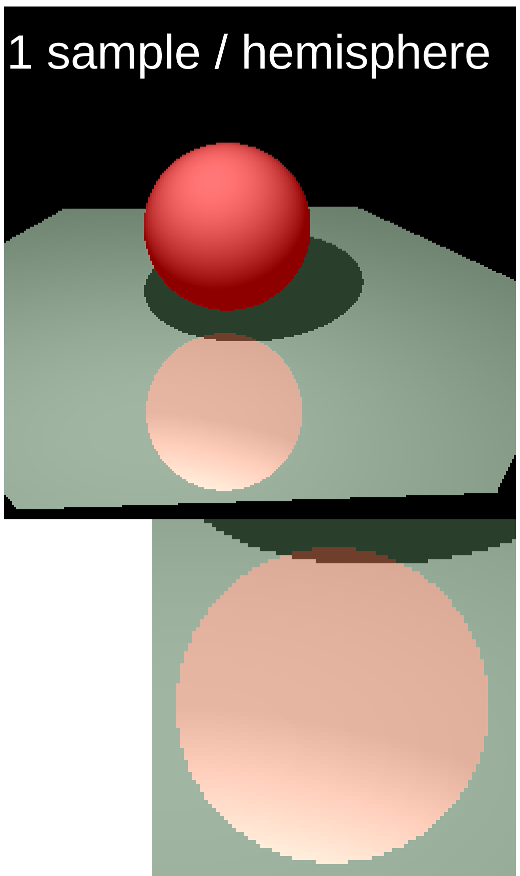
- One sample per hemisphere
  - Specular reflections
- Multiple samples per hemisphere
  - Glossy materials





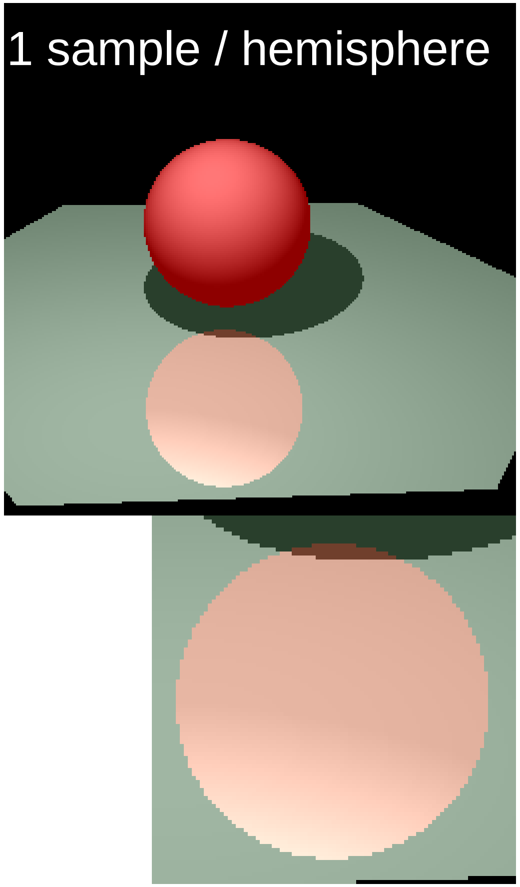
Soft reflections (glossiness)

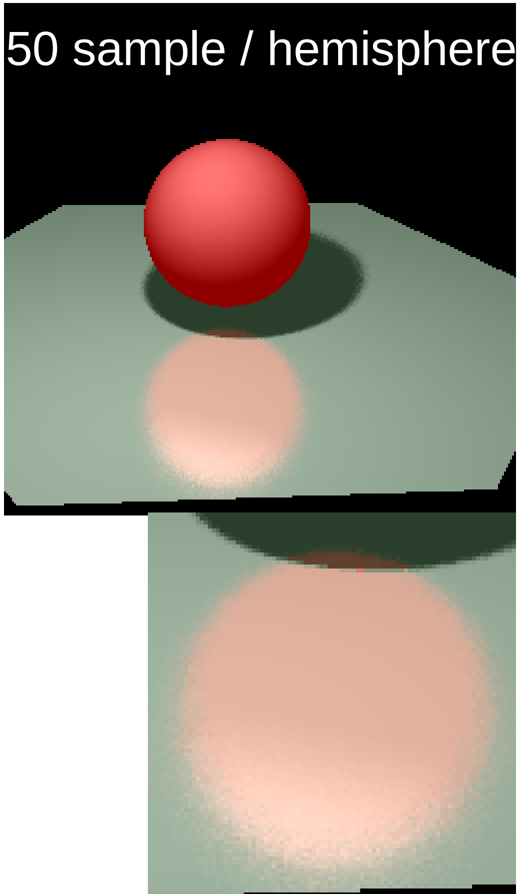
- One sample per hemisphere
  - Specular reflections
- Multiple samples per hemisphere
  - Glossy materials



Soft reflections (glossiness)

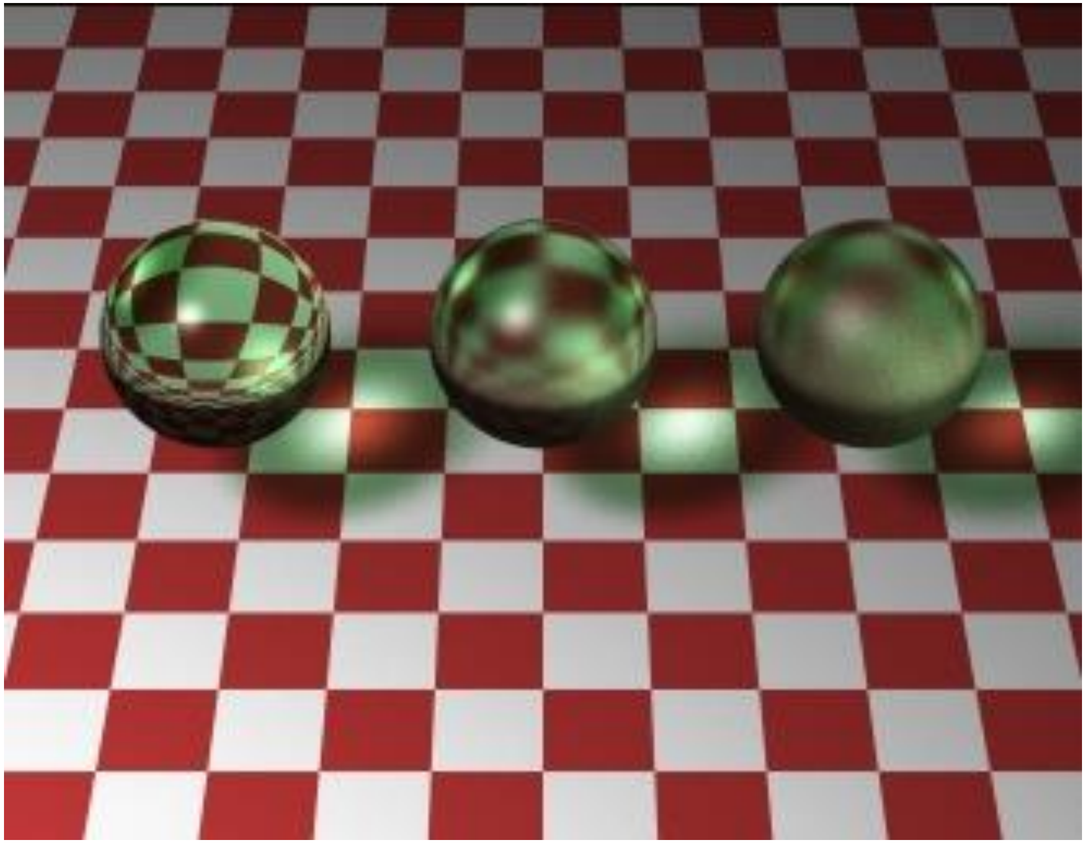
- One sample per hemisphere
  - Specular reflections
- Multiple samples per hemisphere
  - Glossy materials





Soft refractions (translucency)

- One sample per hemisphere
  - Specular refractions
- Multiple samples per hemisphere
  - Translucent materials

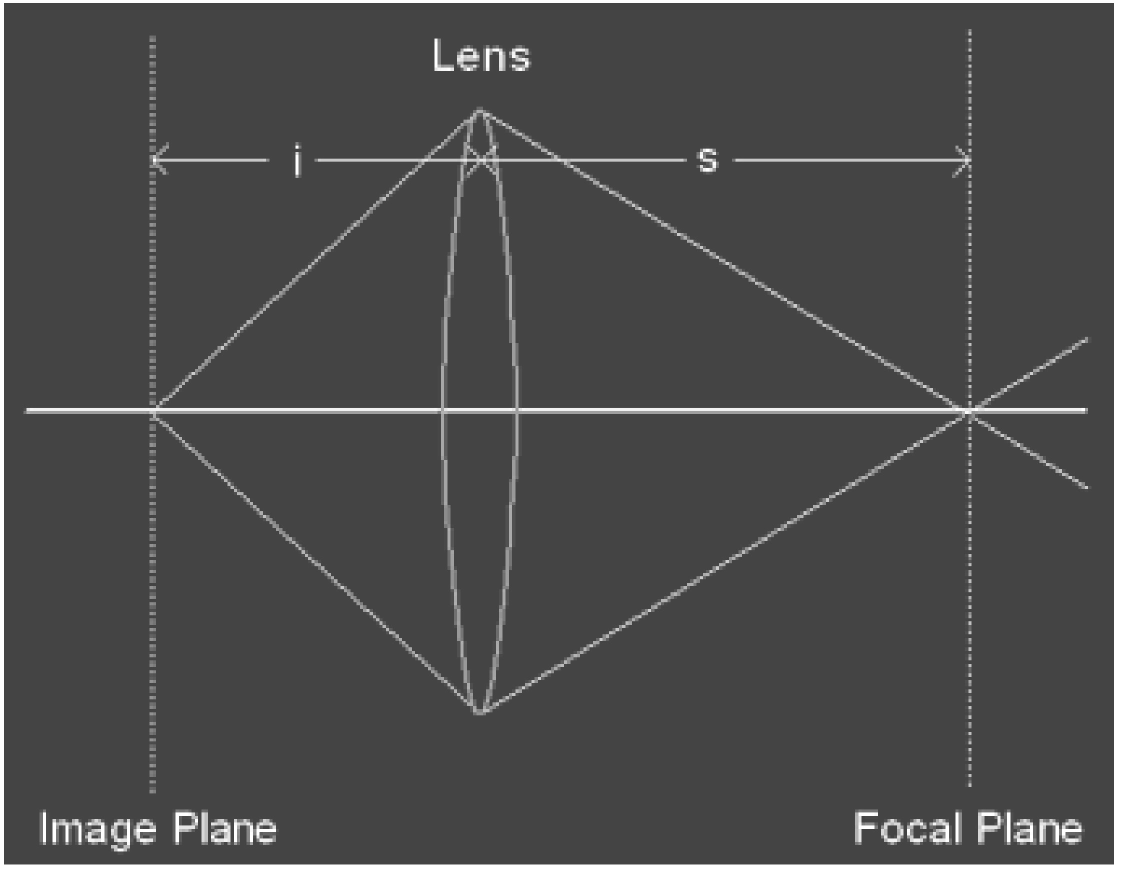


Soft wavelenghts (diffraction)

- Sampling wavelengths

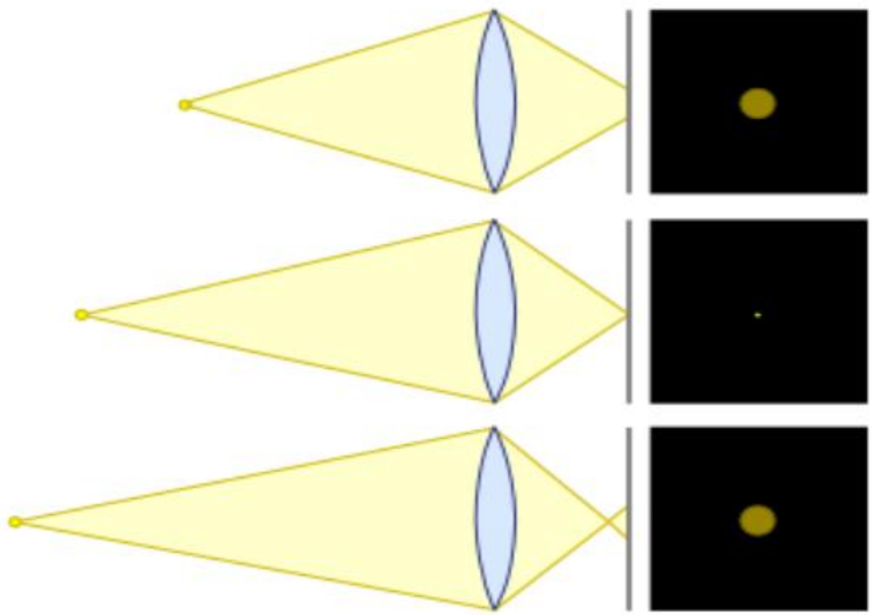
Soft depth (focal length)

- Sampling aperture



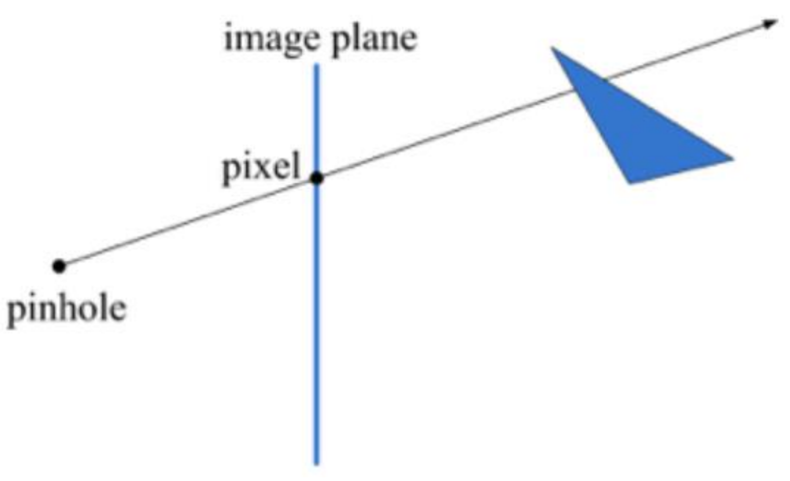
Soft depth (focal length)

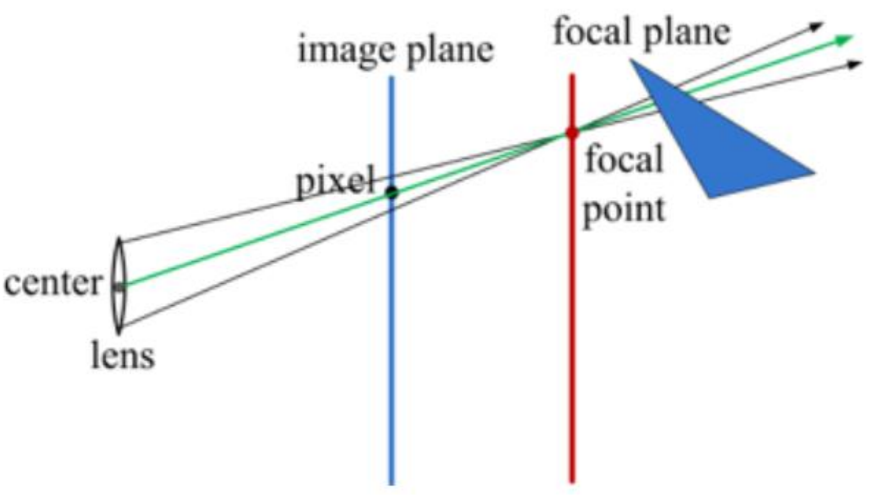
- Sampling aperture



Soft depth (focal length)

- Sampling aperture in practice
  - Sample a circular region
  - In the direction of the focal point
    - Intersect focal plane at the direction in the center of the disk





Soft depth (focal length)

- Sampling aperture in practice
  - Sample a circular region
  - In the direction of the focal point
    - Intersect focal plane at the direction in the center of the disk

Soft animation (motion blur)

- Sampling time
  - Objects in motion (between time 1 and 2)

Summary

Distributed ray tracing

- Sampling pixels → antialiasing
- Sampling light → soft shadows
- Sampling wavelengths → dispersion
- Sampling aperture → depth of field
- Sampling time → motion blur
- Sampling reflection function → blurred reflections
- Sampling refraction function → blurred refractions
- Sampling paths→ interreflections

$$L_o(\mathbf{v}) = \int f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l}) d\omega_i$$

Summary

Distributed ray tracing

- Sampling pixels → antialiasing
- Sampling light → soft shadows
- Sampling wavelengths → dispersion
- Sampling aperture → depth of field
- Sampling time → motion blur
- Sampling reflection function → blurred reflections
- Sampling refraction function → blurred refractions
- Sampling paths→ interreflections

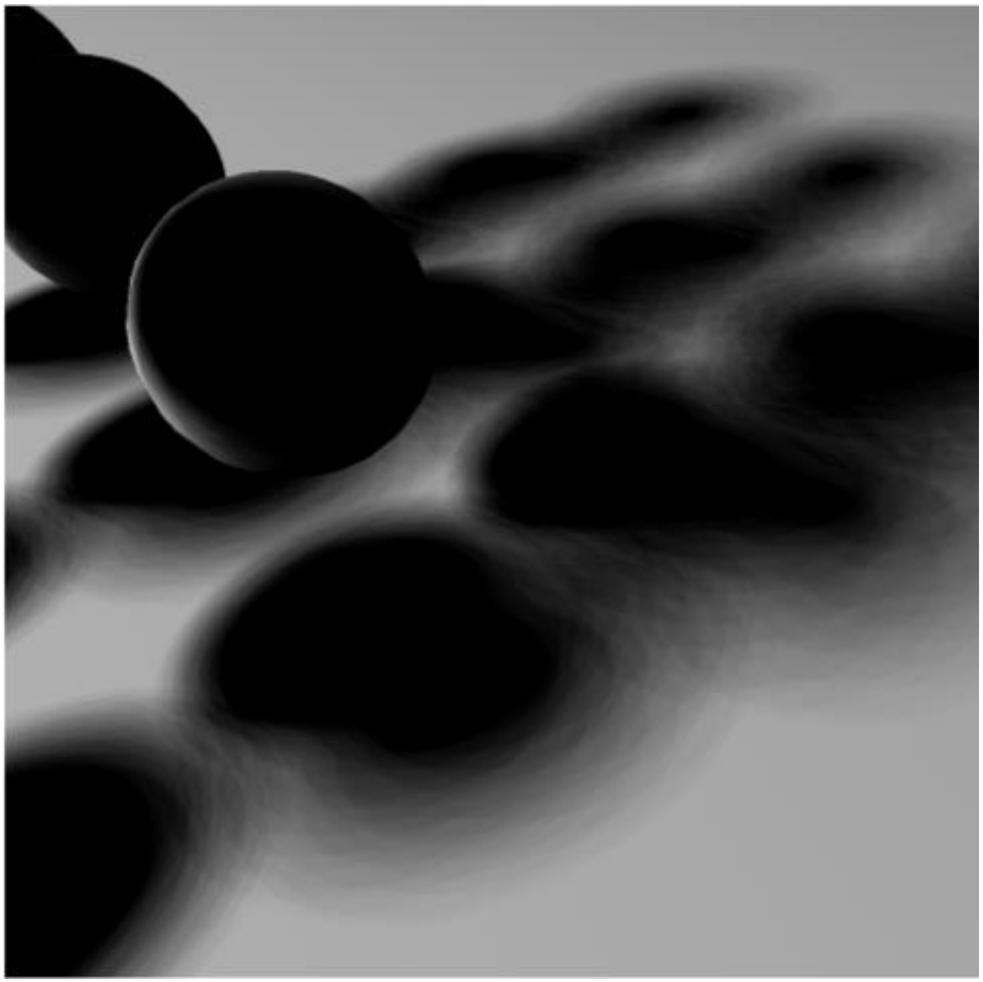
$$L_o(\mathbf{v}) = \int f(\mathbf{l},\mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l})d\omega_i$$

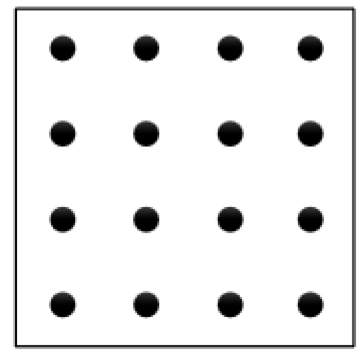
$$\Downarrow$$

$$L_o(\mathbf{v}) = \int\int\int\int\int\int\int f(\mathbf{l},\mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l})d\omega_i d_x d_y d_u d_v d_t d_\lambda$$

Sampling strategy

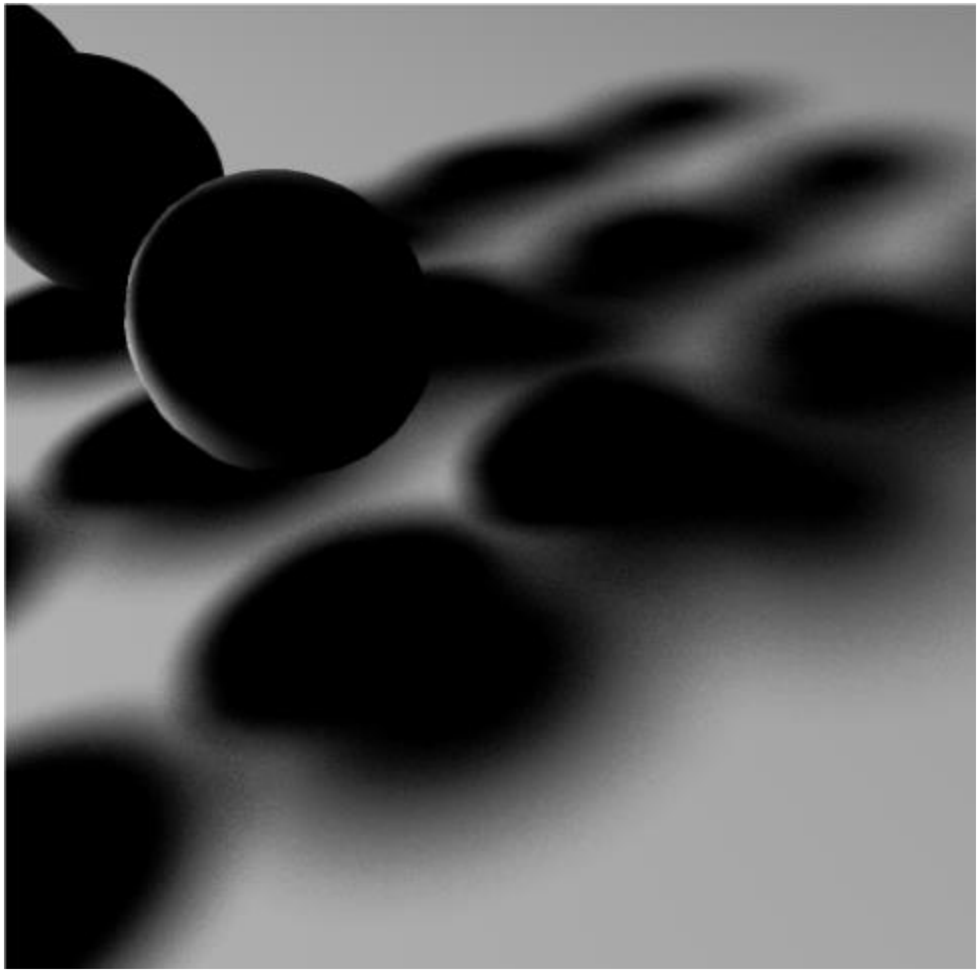
- Regular sampling

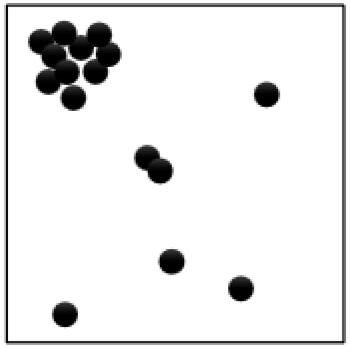




Sampling strategy

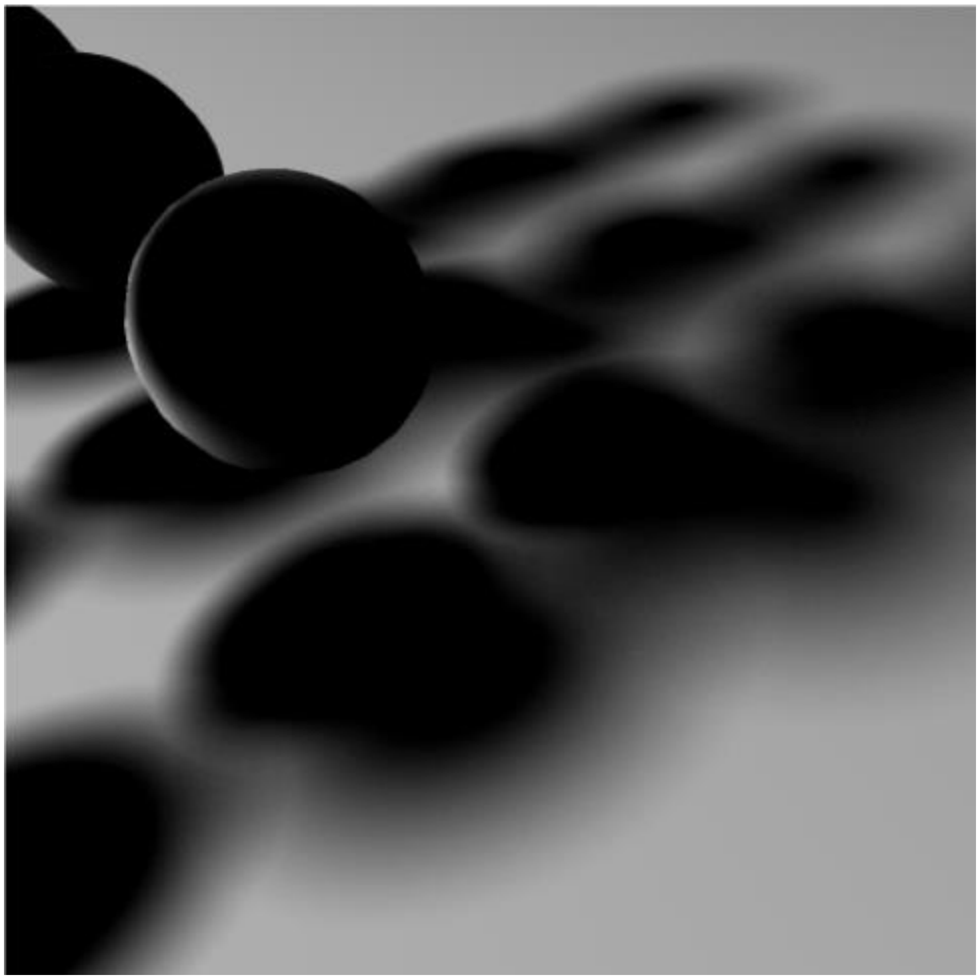
- Uniform sampling

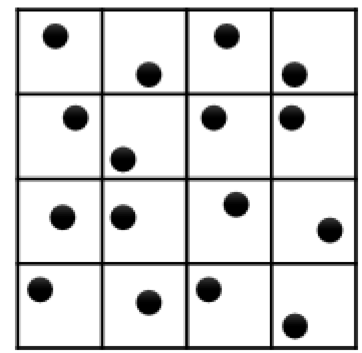




Sampling strategy

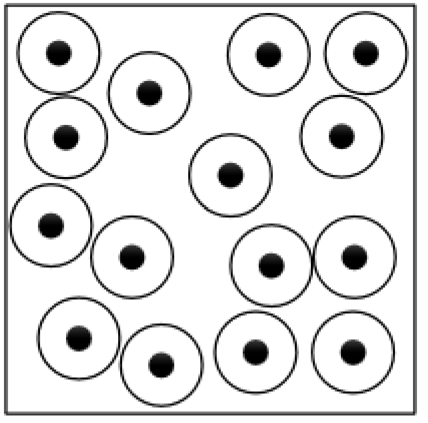
- Stratified sampling

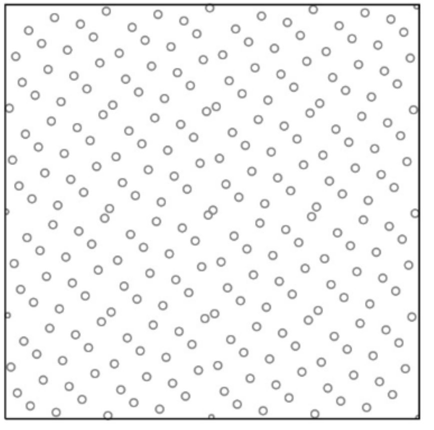




Sampling strategy

- Regular sampling
- Uniform sampling
- Stratified sampling
- And many others
  - Poisson disk sampling
  - Low discrepancy sampling
  - …





Ray tracing produces realistic images

Path notation

- Each path begins and ends with:
  - E – the eye
  - L – the light

Path notation

- Each path begins and ends with:
  - E – the eye
  - L – the light
- Each bounce involves interaction with a surface:
  - D – diffuse reflection
  - G – glossy reflection
  - S – specular reflection

Path notation

- Each path begins and ends with:
  - E – the eye
  - L – the light
- Each bounce involves interaction with a surface:
  - D – diffuse reflection
  - G – glossy reflection
  - S – specular reflection
- Ray-casting:
  - E(D,G)L
- Ray-tracing:
  - E[S*](D,G)L
- All paths:
  - E(D,G,S)*L

Path notation

- Each path begins and ends with:
  - E – the eye
  - L – the light
- Each bounce involves interaction with a surface:
  - D – diffuse reflection
  - G – glossy reflection
  - S – specular reflection
- Ray-casting:
  - E(D,G)L
- Ray-tracing:
  - E[S*](D,G)L
- All paths:
  - E(D,G,S)*L

Color bleeding?
Caustics?

Mathematical theory

- Rendering equation is complex
  - No analytical solution

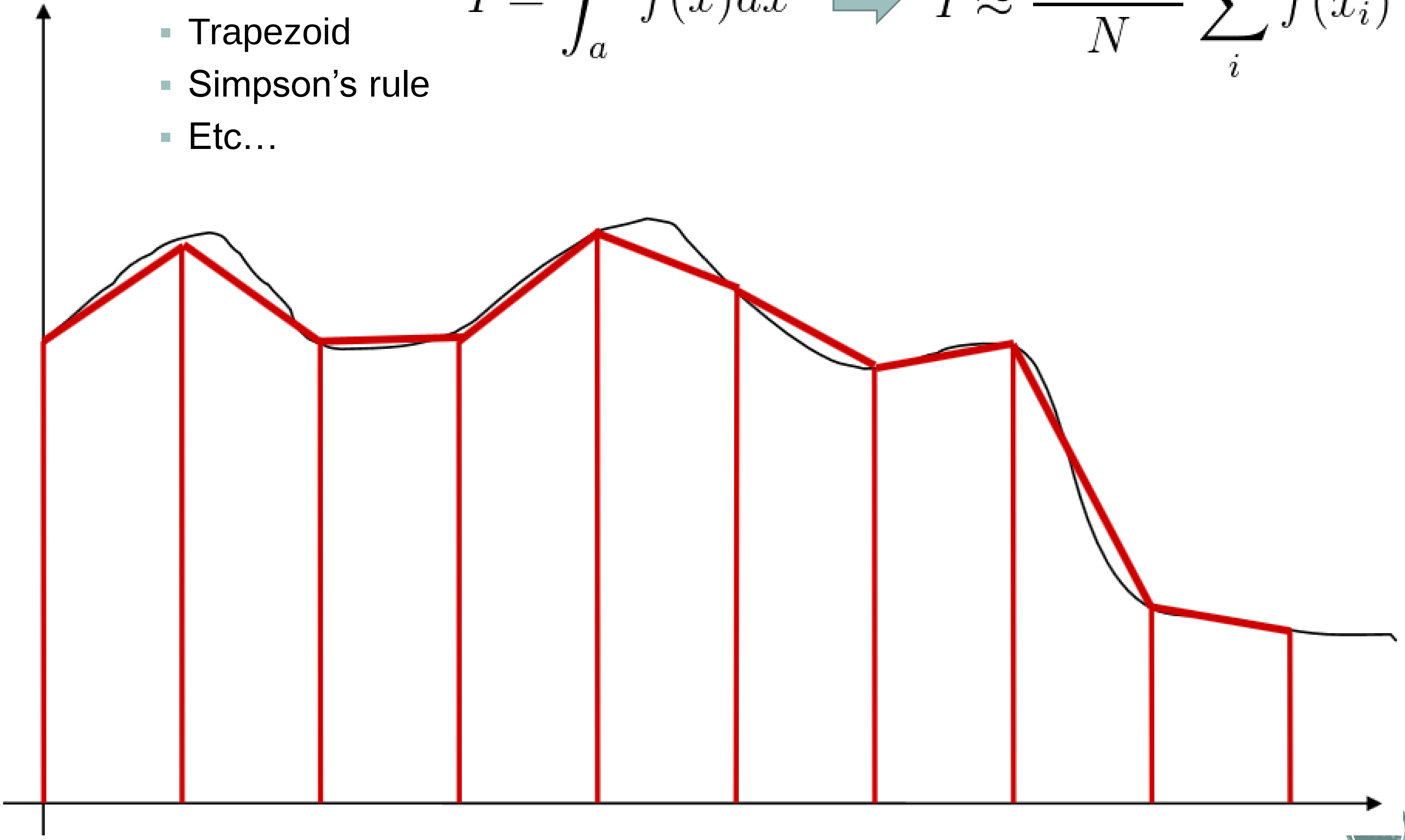
$$L_o(\mathbf{v}) = \int f(\mathbf{l},\mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l}) d\omega_i$$

  - → numerical scheme
    - Monte-carlo = powerfull tool to solve integrals that do not have analytical solutions (such as the rendering equation)

Mathematical theory

- Discretization
  - Trapezoid
  - Simpson's rule
  - Etc…

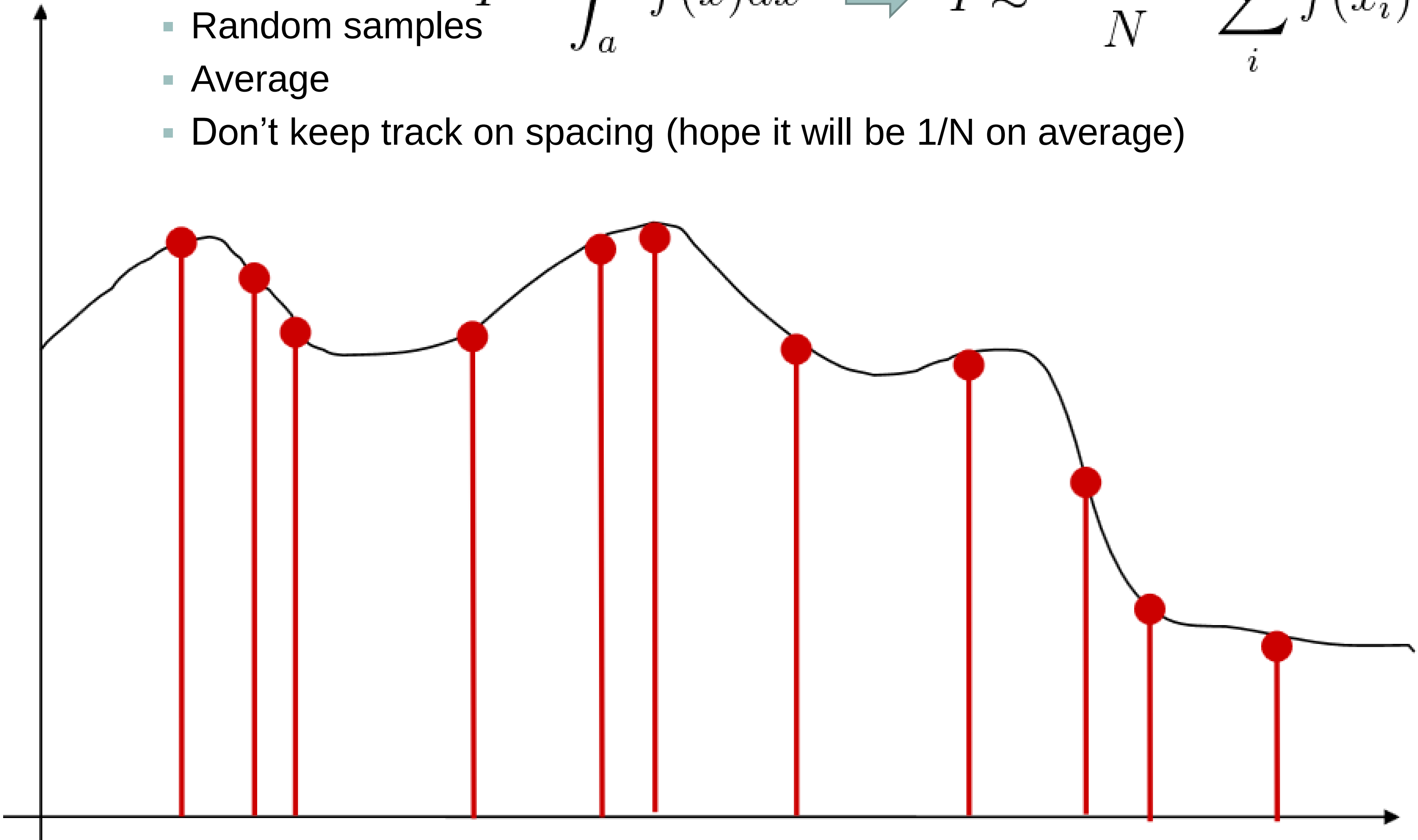
$$I = \int_a^b f(x)dx \Rightarrow I \approx \frac{(b-a)}{N} \sum_i^N f(x_i)$$



Mathematical theory

- Monte-carlo
  - Random samples
  - Average
  - Don't keep track on spacing (hope it will be 1/N on average)

$$I = \int_a^b f(x)dx \Rightarrow I \approx \frac{(b-a)}{N}\sum_i^N f(x_i)$$



Mathematical theory

- Monte-carlo
  - Random samples
  - Average
  - Don't keep track on spacing (hope it will be 1/N on average)

$$I = \int_a^b f(x)dx \Rightarrow I \approx \frac{(b-a)}{N} \sum_i^N f(x_i)$$

- MC estimator:

$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{1/(b-a)}$$

Mathematical theory

- Monte-carlo
  - Random samples
  - Average
  - Don’t keep track on spacing (hope it will be 1/N on average)

$$I = \int_a^b f(x)dx \Rightarrow I \approx \frac{(b-a)}{N}\sum_i^N f(x_i)$$

- MC estimator: $$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{1/(b-a)}$$

- Expected value: $$E[F_N] = \int_a^b f(x)dx$$

- Variance: $$\sigma^2[F_N] = E[F_N^2] - (E[F_N])^2$$

Mathematical theory

- MC estimator: $$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{1/(b-a)}$$
- Generalization: $$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{pdf(x_i)}$$

Mathematical theory

- MC estimator: $$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{1/(b-a)}$$

- Generalization: $$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{pdf(x_i)}$$

Depends on the sampling strategy

Mathematical theory

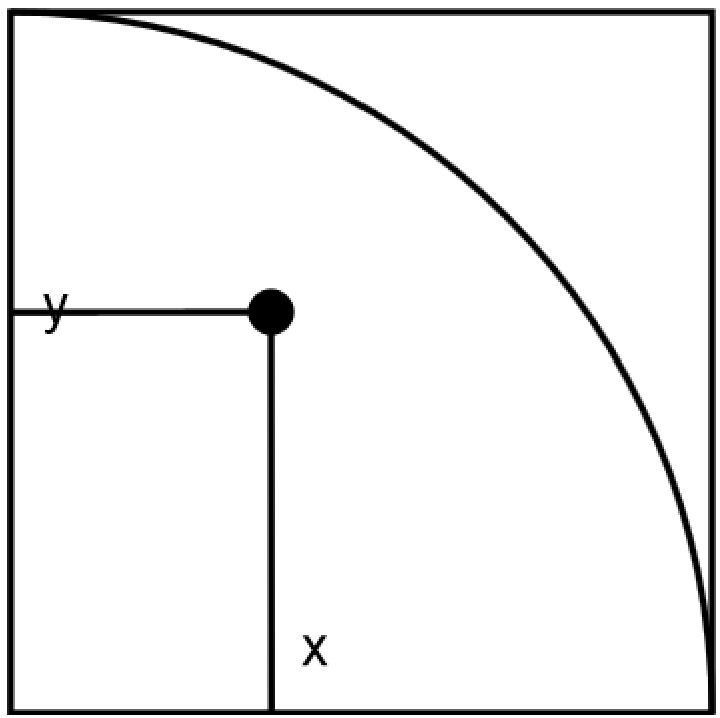
- MC estimator: $$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{1/(b-a)}$$
- Generalization: $$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{pdf(x_i)}$$

Depends on the sampling strategy

- Convergence rate: $O(\sqrt{N})$
  - Divide error by 2 needs 4x samples

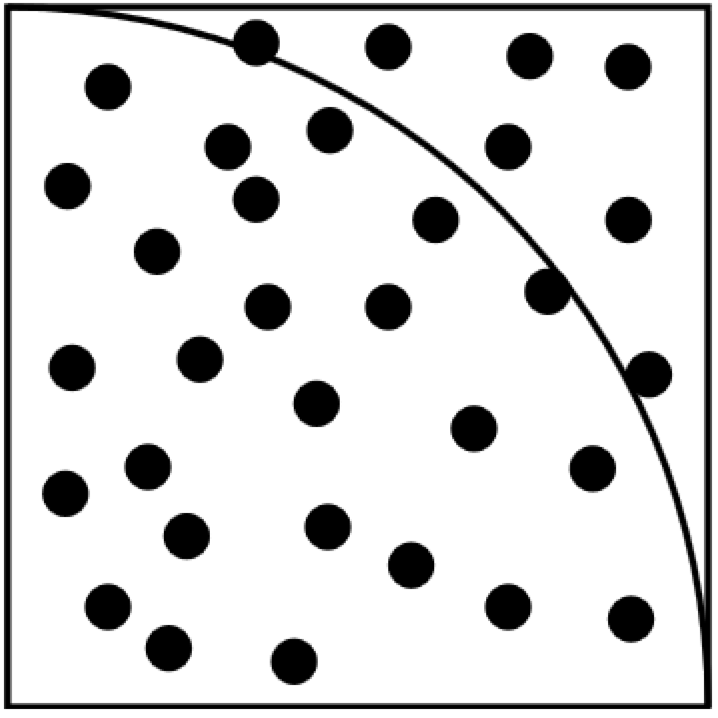
Example: compute PI

- Integral of the function that is 1 inside the circle and 0 outside
  - Probability = PI/4



Example: compute PI

- Integral of the function that is 1 inside the circle and 0 outside
  - Probability = PI/4
- Sample random positions
  - Count ratio n = #inside / #total
  - PI ~= 4*n



Pros / cons

- Slow for 1D problems, but:
  - Convergence is independent of dimensions
  - Few restrictions on the integrand
  - Conceptually simple
  - Efficient to solve at just a few points

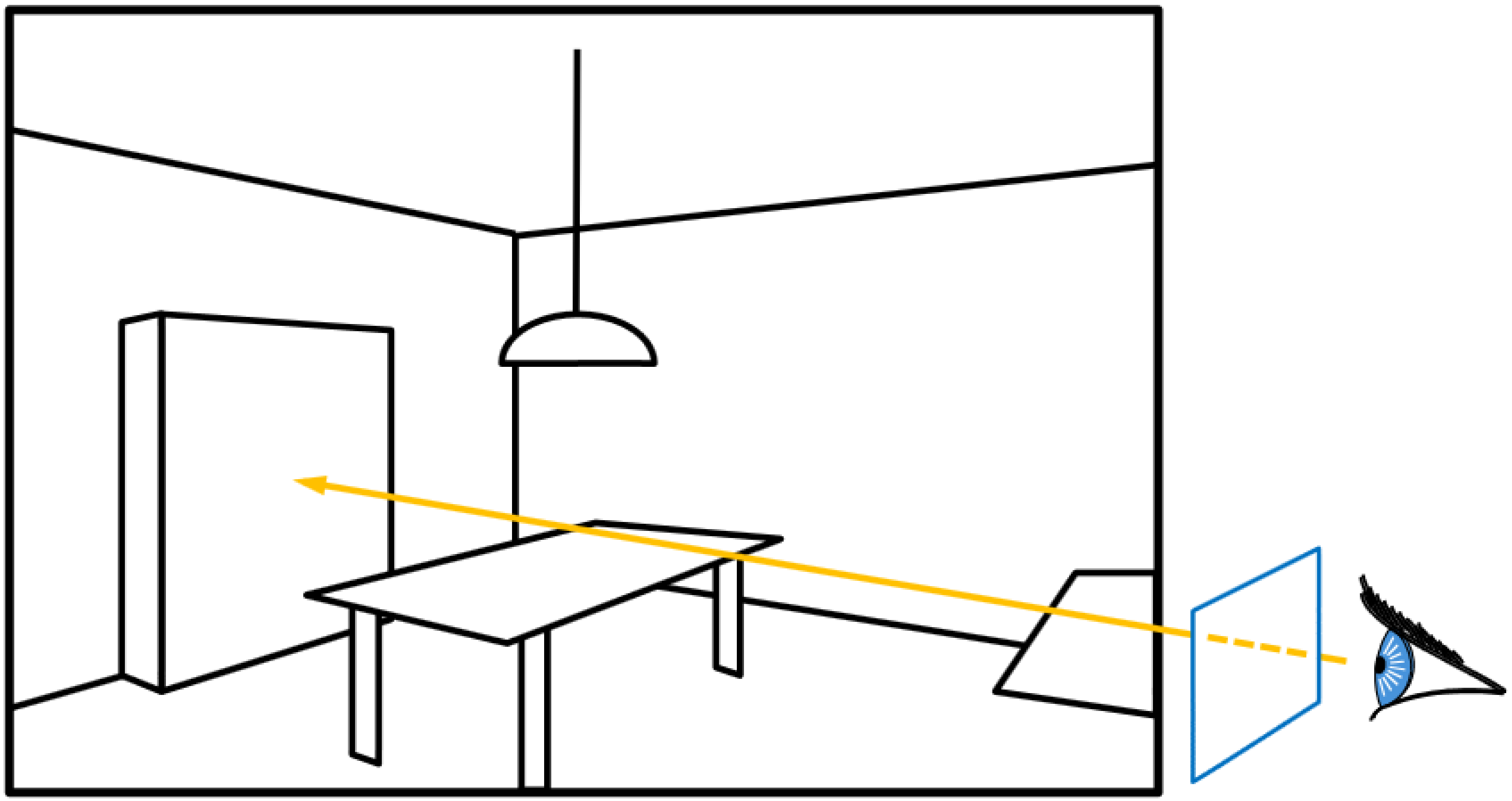
Pros / cons

- Slow for 1D problems, but:
  - Convergence is independent of dimensions
  - Few restrictions on the integrand
  - Conceptually simple
  - Efficient to solve at just a few points

- Cons:
  - Noisy
  - Slow convergence
  - Good implementation is hard

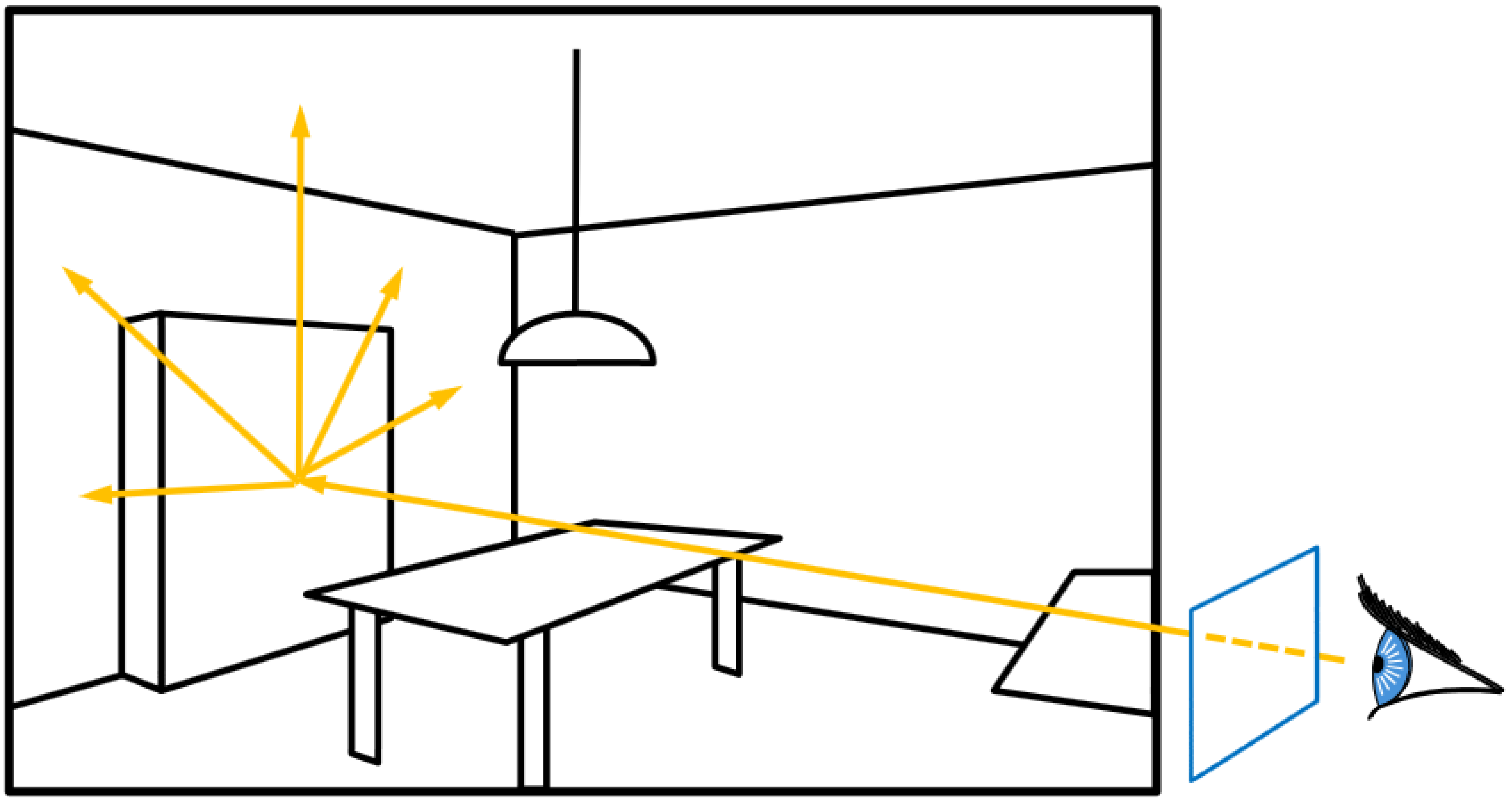
Monte-carlo ray-tracing

- One ray per pixel



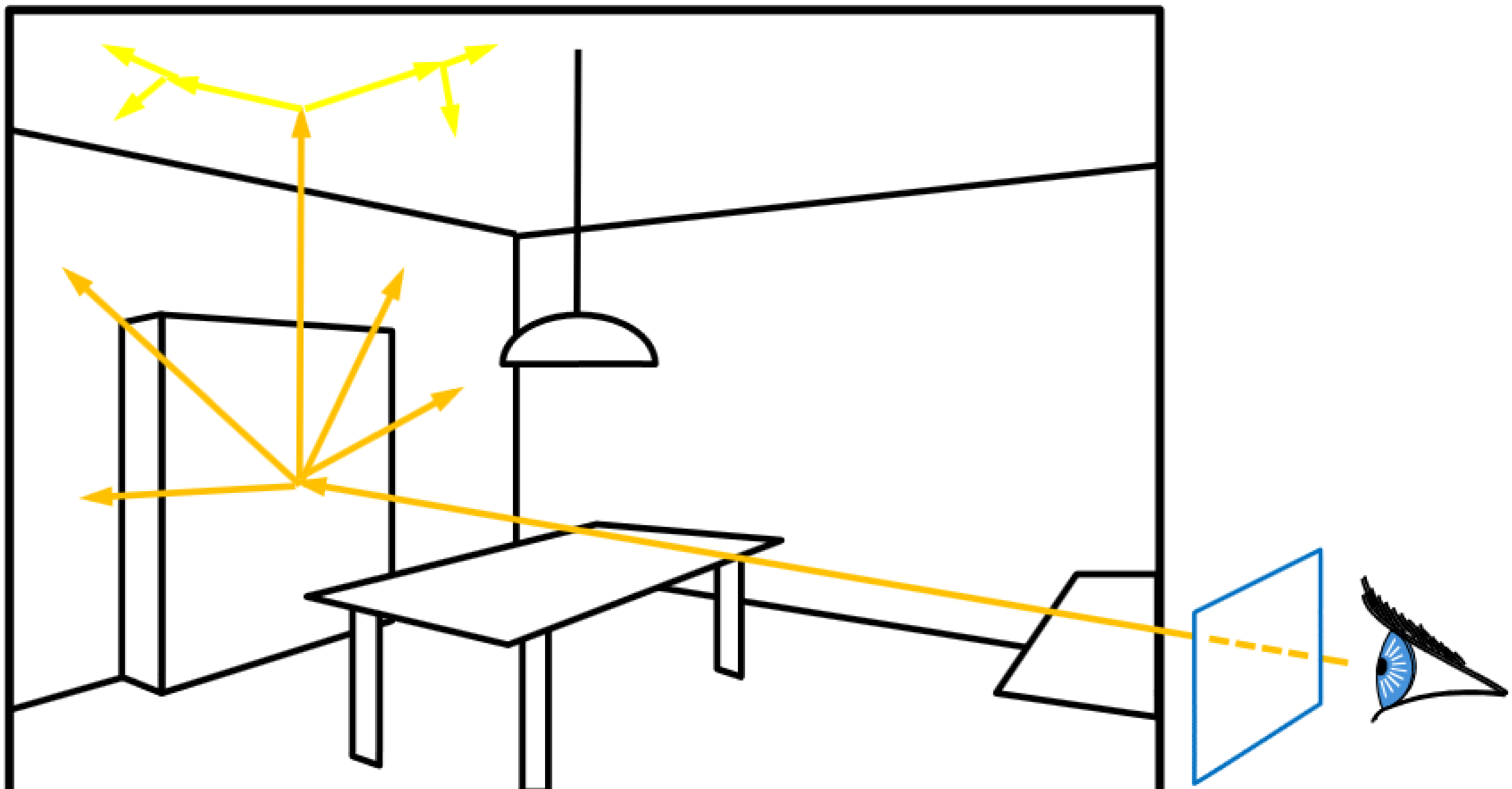
Monte-carlo ray-tracing

- One ray per pixel
- On each point, generate random rays, accumulate radiance



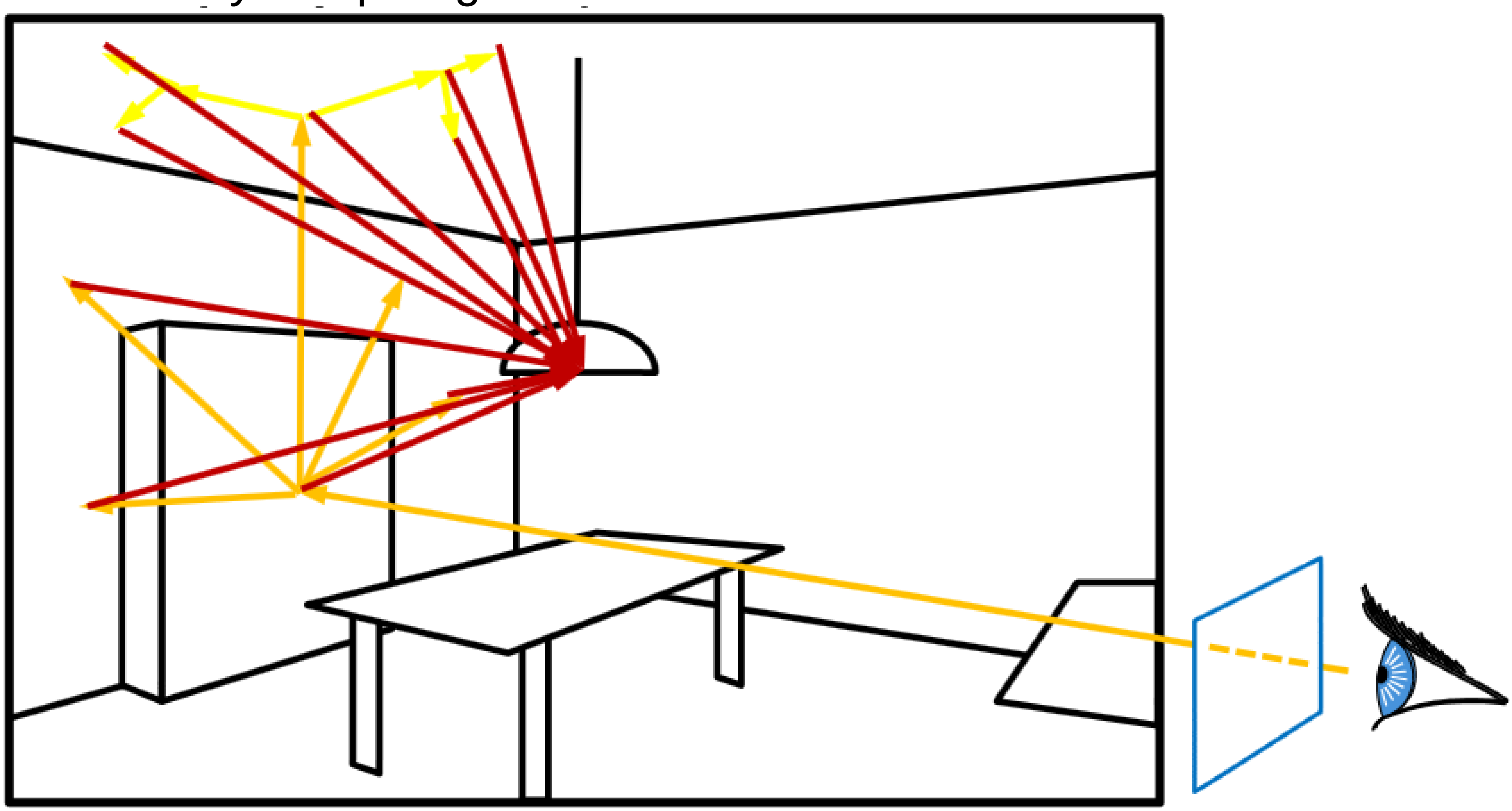
Monte-carlo ray-tracing

- One ray per pixel
- On each point, generate random rays, accumulate radiance
- Recurse

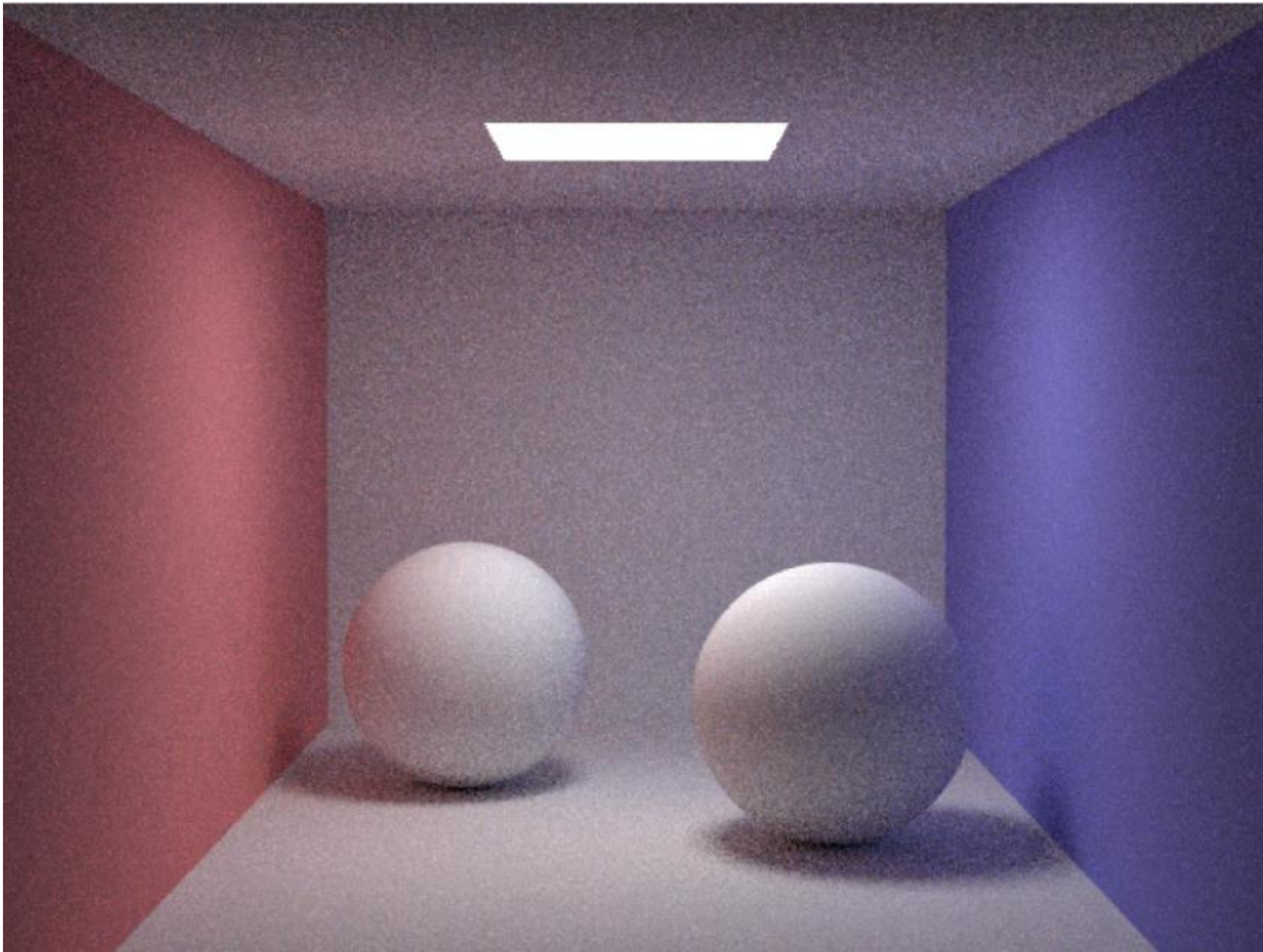


Monte-carlo ray-tracing

- One ray per pixel
- On each point, generate random rays, accumulate radiance
- Recurse
- Systematically sample light

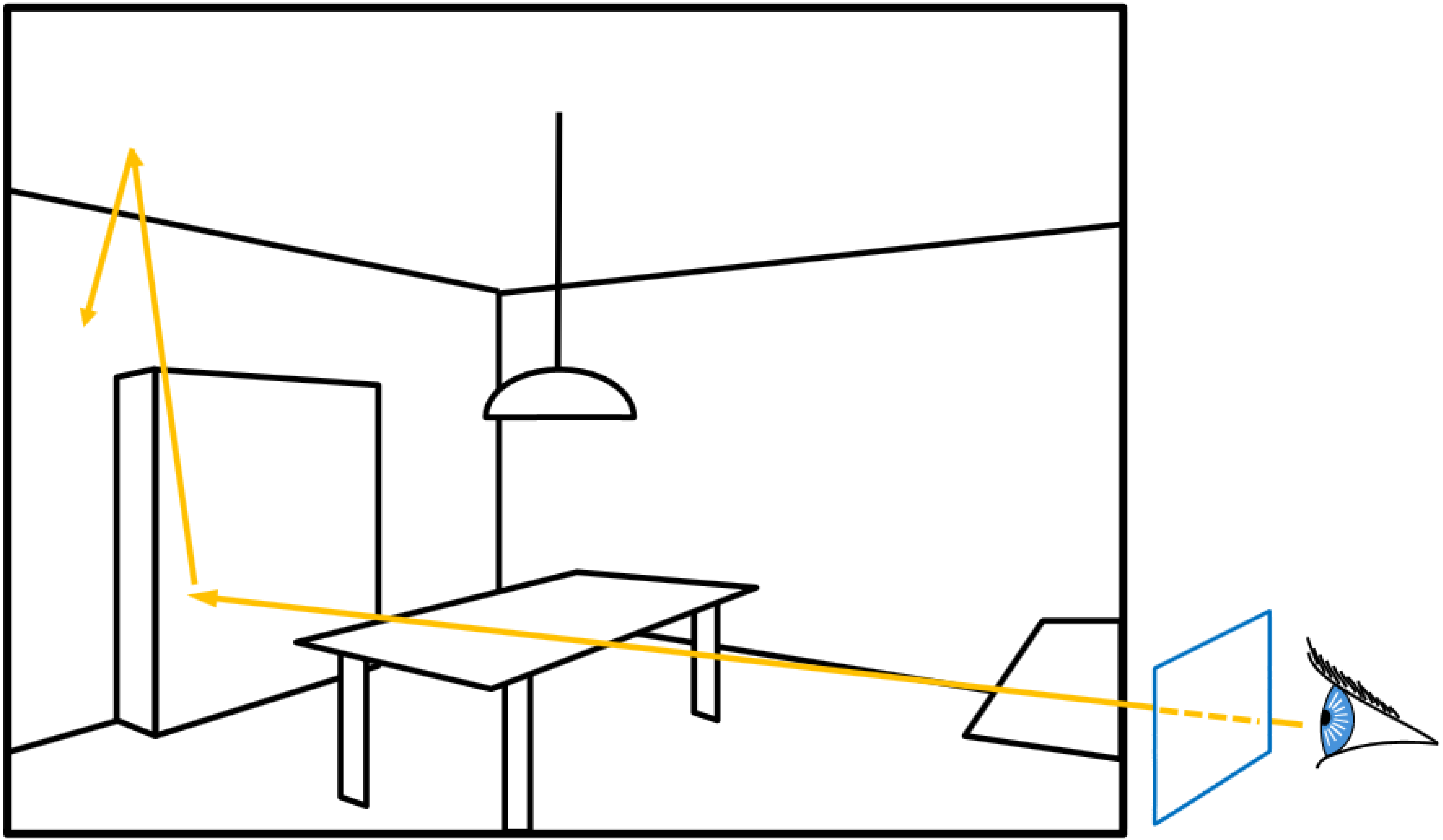


Monte-carlo ray-tracing



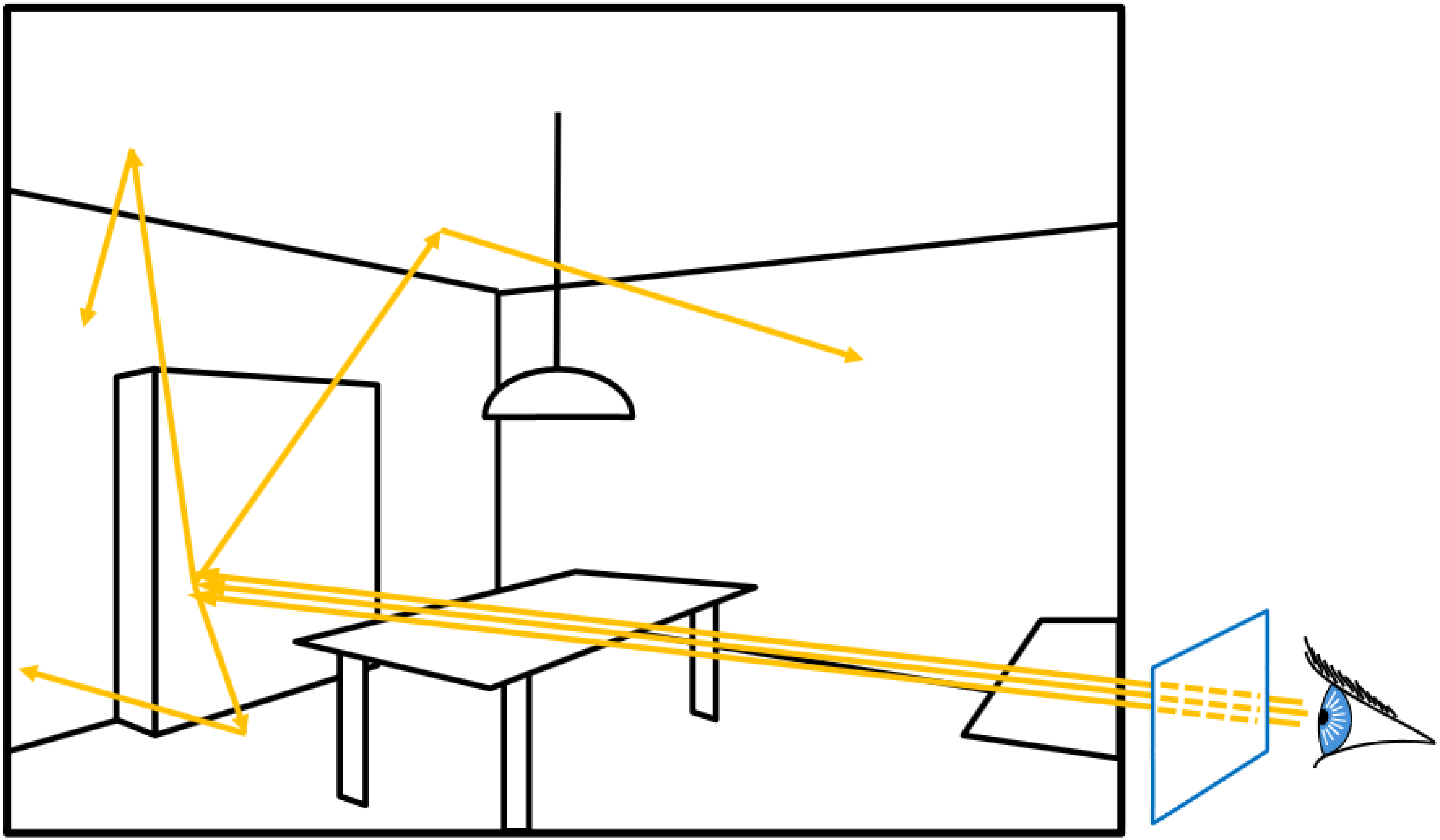
Monte-carlo path-tracing

- One single ray per bounce



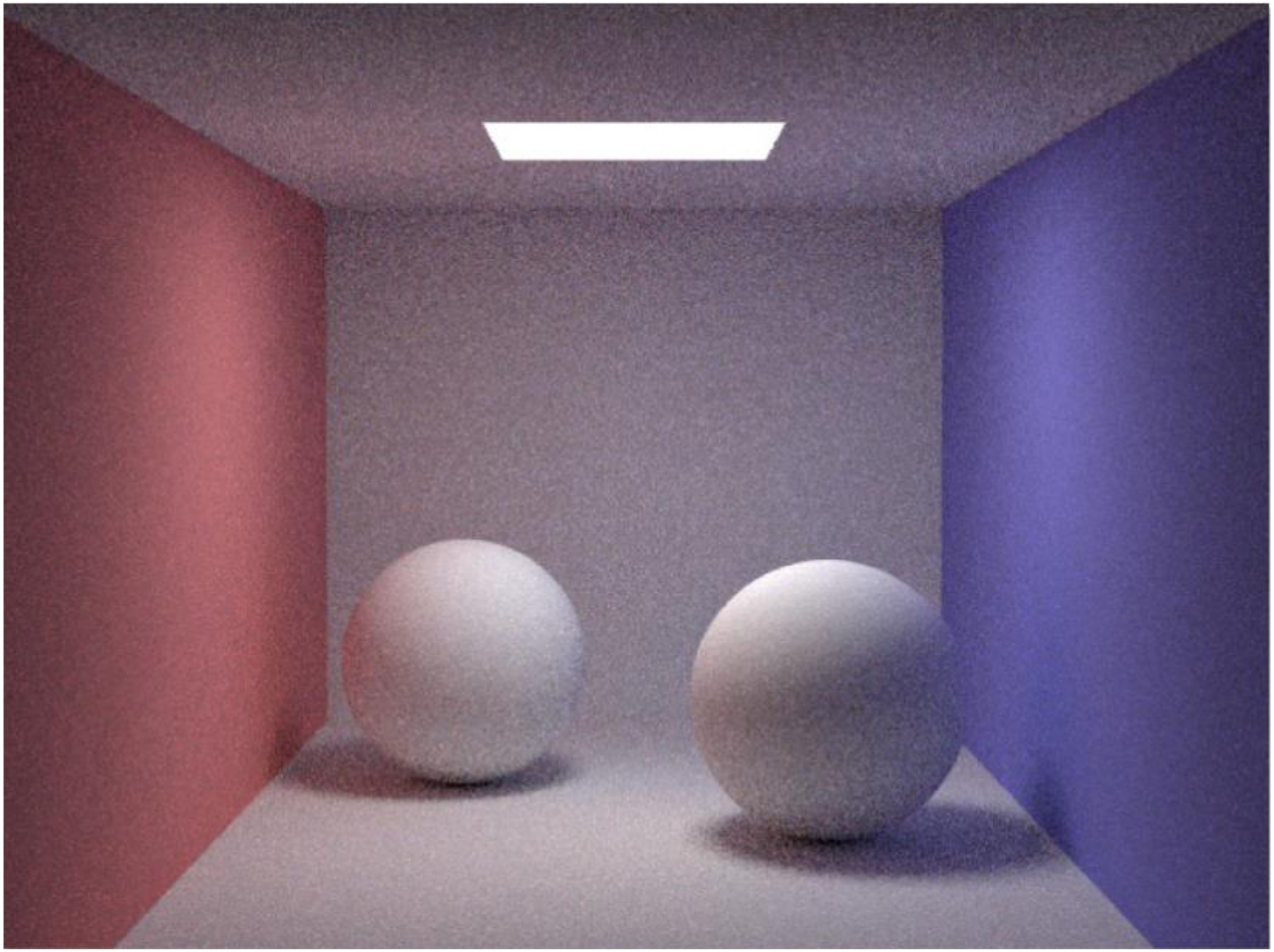
Monte-carlo path-tracing

- One single ray per bounce
- But hundreds of ray per pixel



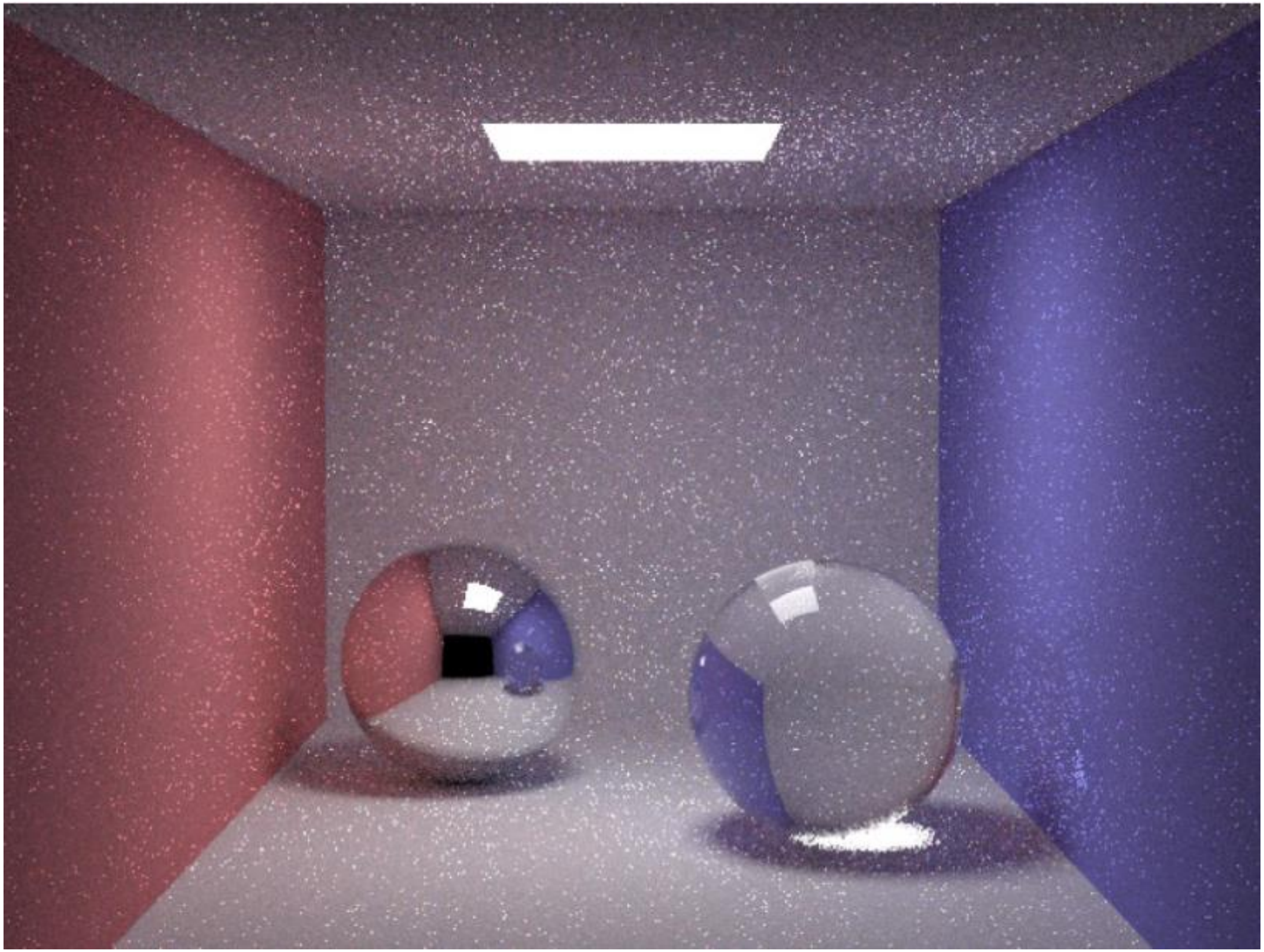
Monte-carlo path-tracing

- 10 paths / pixel



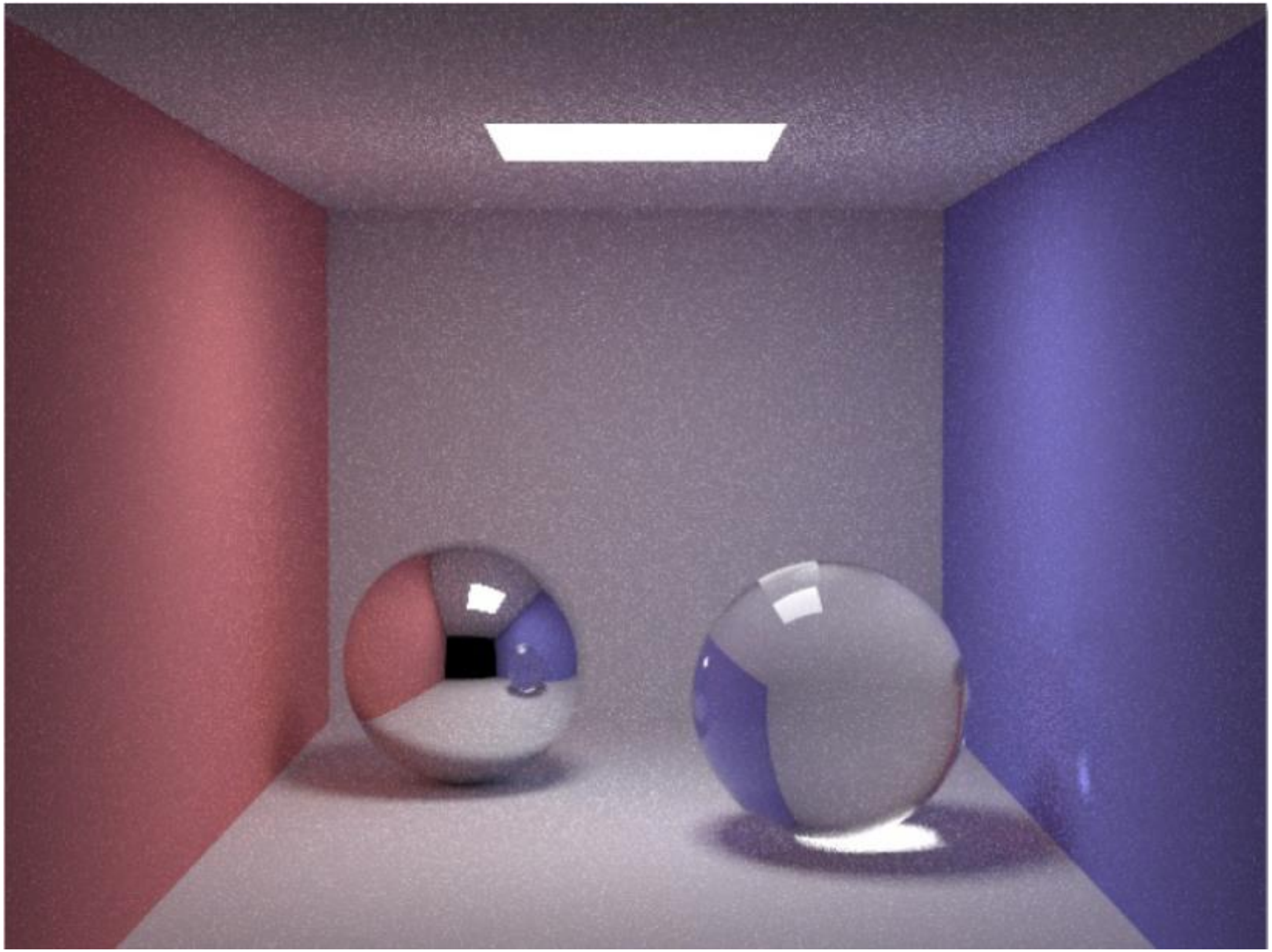
Monte-carlo path-tracing

- 10 paths / pixel



Monte-carlo path-tracing

- 100 paths / pixel



Importance sampling

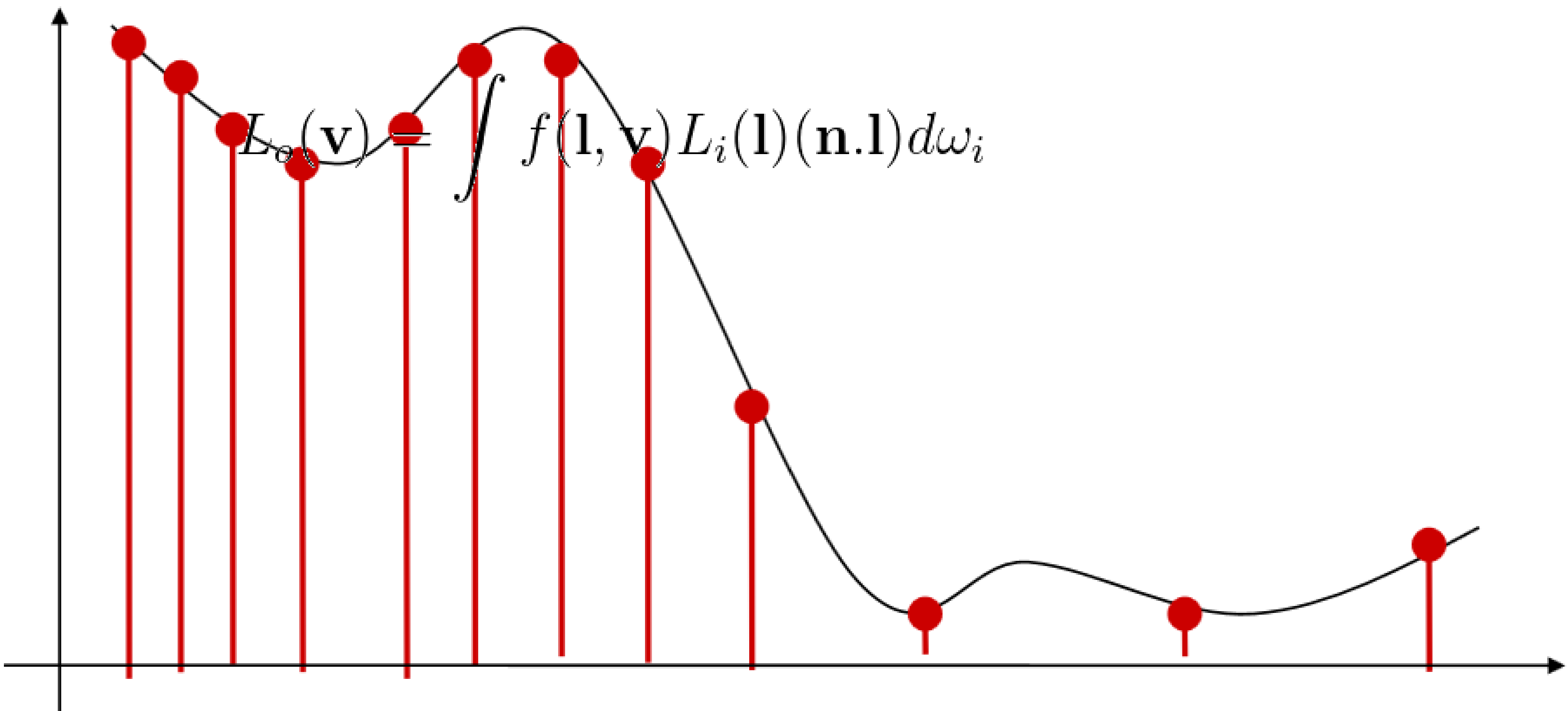
Naïve sampling strategy

Optimal sampling strategy

Importance sampling

- Sampling strategy
  - Non-uniform sampling
  - Sample more in places where there are likely to be larger contributions to the integral

$$L_o(\mathbf{v}) = \int f(\mathbf{l},\mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l})\, d\omega_i$$



Importance sampling

- Sampling strategy
  - Non-uniform sampling
  - Sample more in places where there are likely to be larger contributions to the integral

$$L_o(\mathbf{v}) \approx \frac{1}{N} \sum \frac{f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l}) (\mathbf{n}.\mathbf{l})}{pdf(\mathbf{l})}$$

Importance sampling

- Sampling strategy
  - Non-uniform sampling
  - Sample more in places where there are likely to be larger contributions to the integral

$$L_o(\mathbf{v}) \approx \frac{1}{N} \sum \frac{f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l})(\mathbf{n}.\mathbf{l})}{pdf(\mathbf{l})}$$

Send more rays in the direction of reflection
(depending on glossiness properties)

Importance sampling

- Sampling strategy
  - Non-uniform sampling
  - Sample more in places where there are likely to be larger contributions to the integral

$$L_o(\mathbf{v}) \approx \frac{1}{N} \sum \frac{f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l}) (\mathbf{n}.\mathbf{l})}{pdf(\mathbf{l})}$$

Avoid rays at grazing angles

Send more rays in the direction of reflection
(depending on glossiness properties)

Importance sampling

- Sampling strategy
  - Non-uniform sampling
  - Sample more in places where there are likely to be larger contributions to the integral

$$L_o(\mathbf{v}) \approx \frac{1}{N} \sum \frac{f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{l}) (\mathbf{n}.\mathbf{l})}{pdf(\mathbf{l})}$$

Avoid rays at grazing angles

Send more rays in the direction of reflection
(depending on glossiness properties)

Importance sampling

- Sampling strategy
  - Non-uniform sampling
  - Sample more in places where there are likely to be larger contributions to the integral

$$L_o(\mathbf{v}) \approx \frac{1}{N}\sum \frac{f(\mathbf{l},\mathbf{v})L_i(\mathbf{l})(\mathbf{n}.\mathbf{l})}{pdf(\mathbf{l})}$$

Avoid rays at grazing angles

Send more rays in the direction of reflection (depending on glossiness properties)

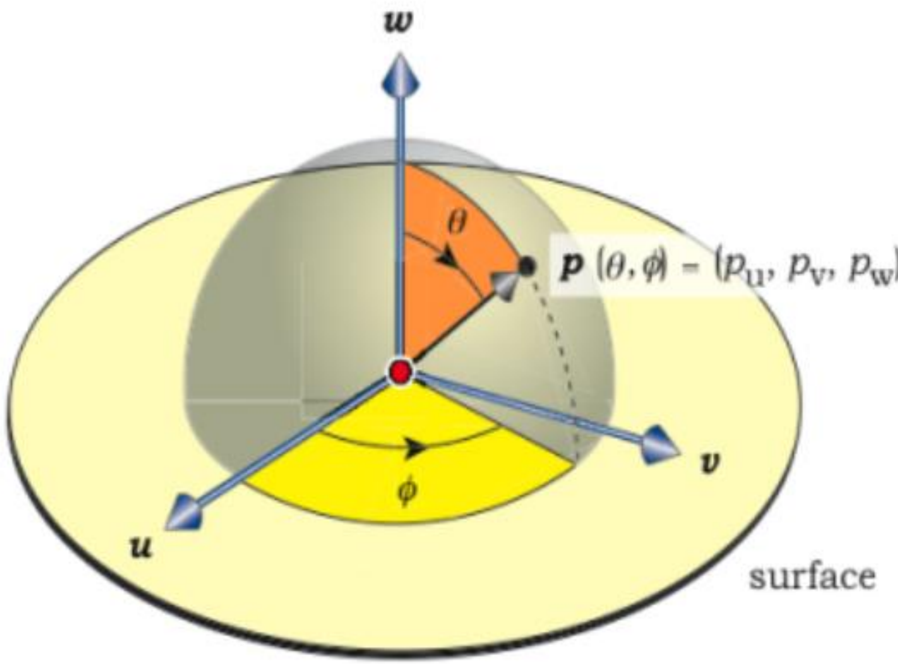
Do not forget pdf

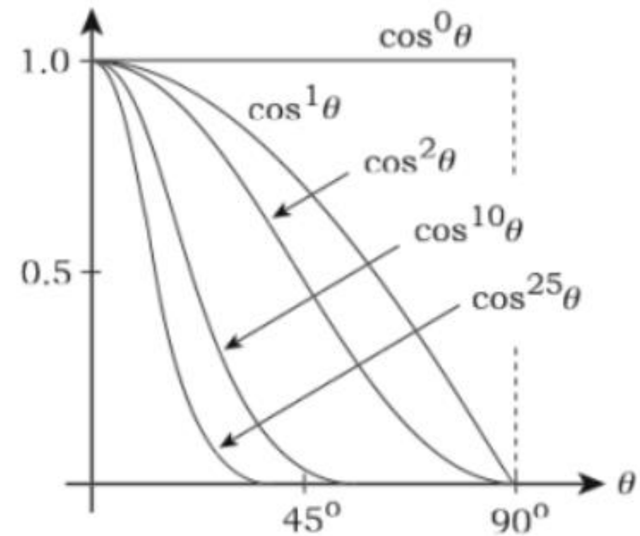
Uniform sampling the hemisphere = 1/vol(S) = 1/2PI

Importance sampling

- Sampling the hemisphere

$$(x \in [0,1), y \in [0,1)) \rightarrow \left(\phi = 2\pi x, \theta = \cos^{-1}\left[(1-y)^{1/(m+1)}\right]\right)$$



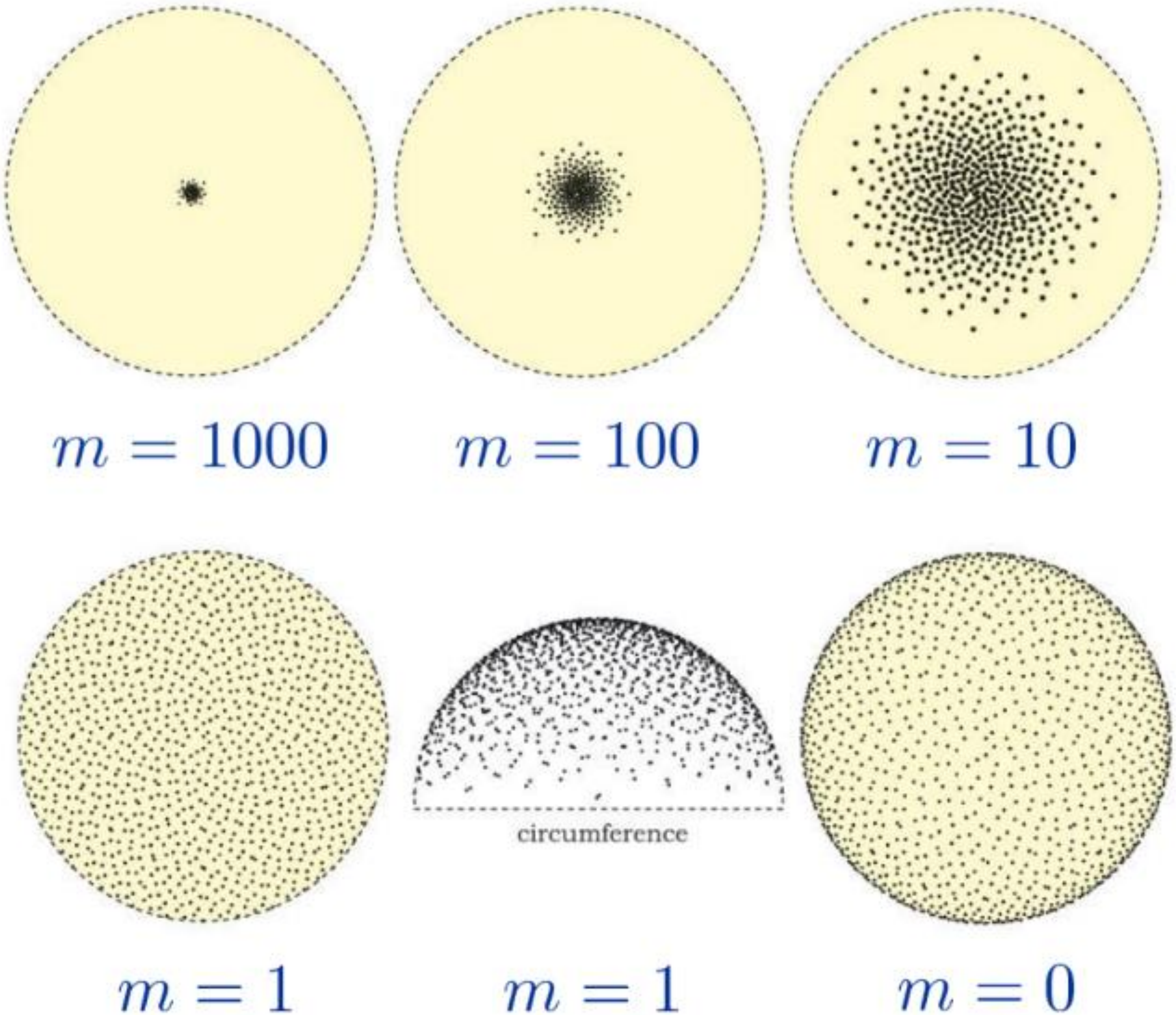


Loi du cosinus surélevé

$$\boldsymbol{p} = \sin\theta\cos\phi\,\boldsymbol{u} + \sin\theta\sin\phi\,\boldsymbol{v} + \cos\theta\,\boldsymbol{w}$$

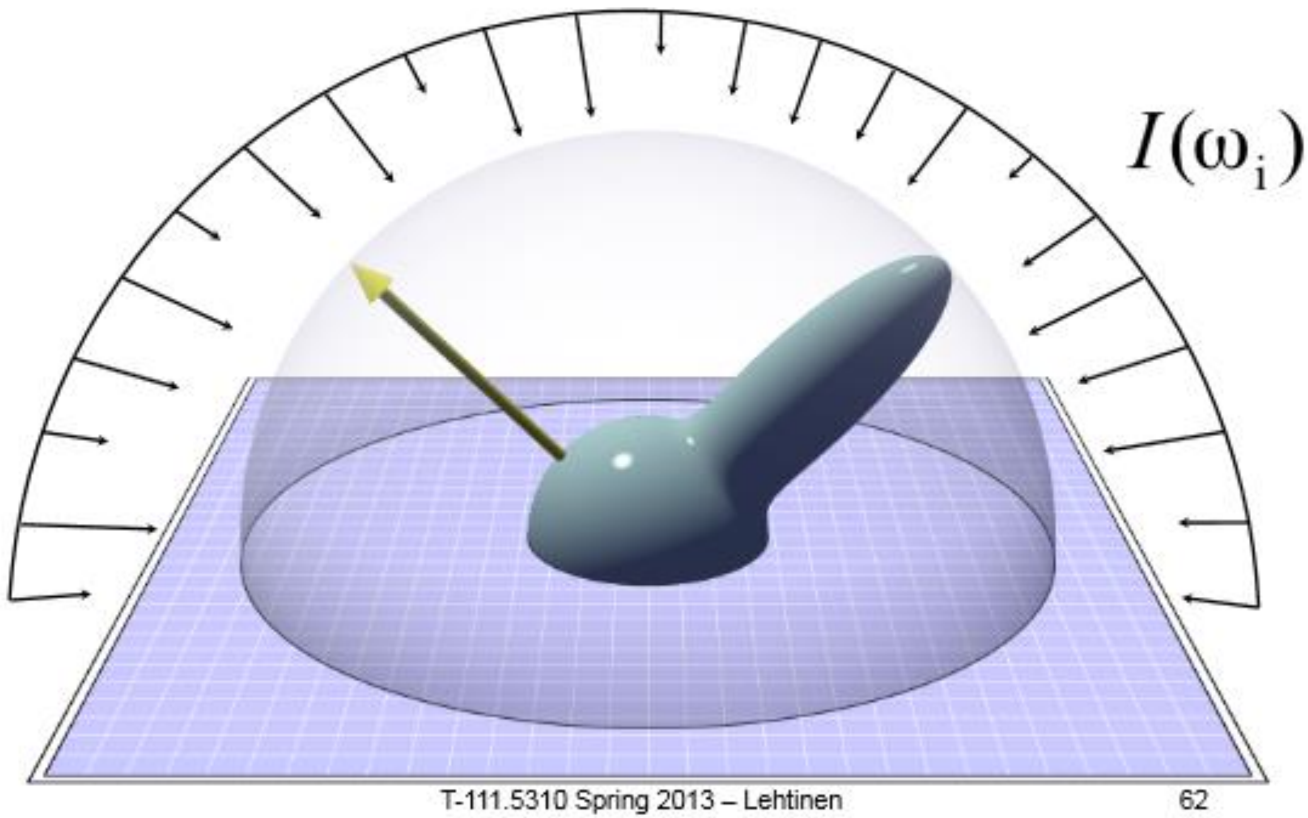
Importance sampling

- Sampling the hemisphere



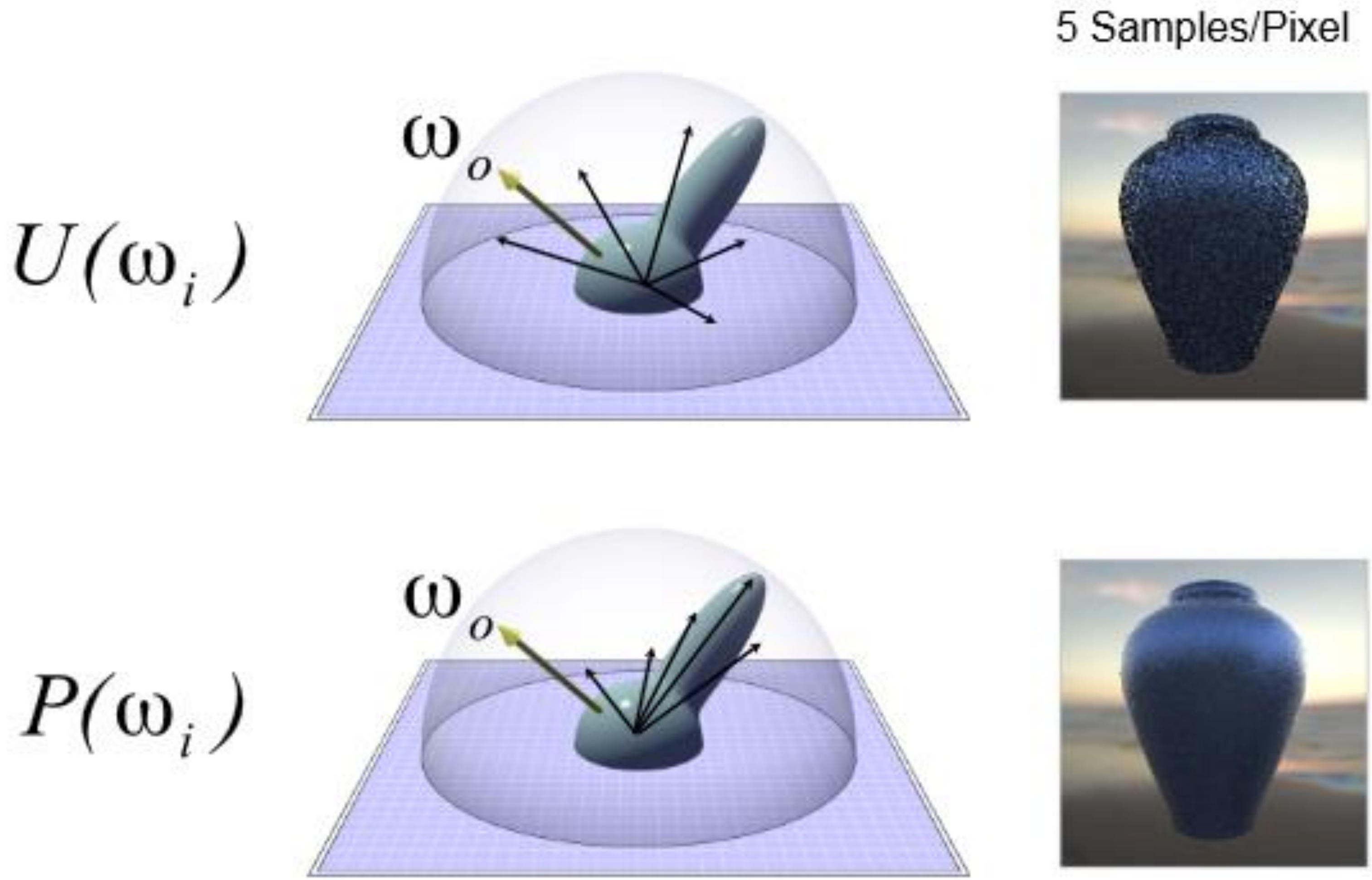
Importance sampling

- Sampling the BRDF



Importance sampling

- Sampling the BRDF

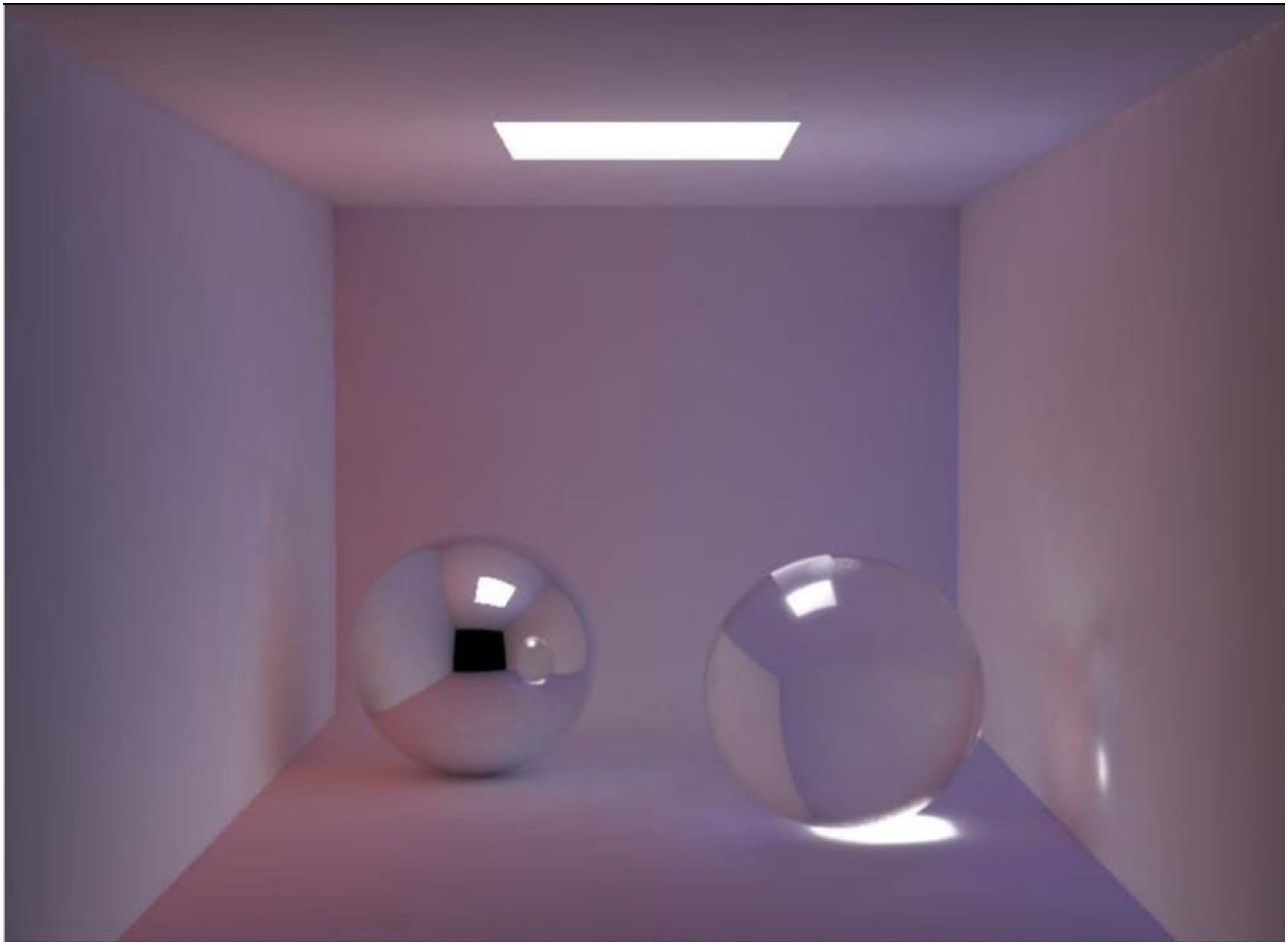


Path-tracing extensions

- Bidirectional path tracing [Lafortune, Veach]
  - Combine lights from light and camera
- Metropolis [Veach]
  - Extension to bi-directional
  - Reuse and mutation of important paths

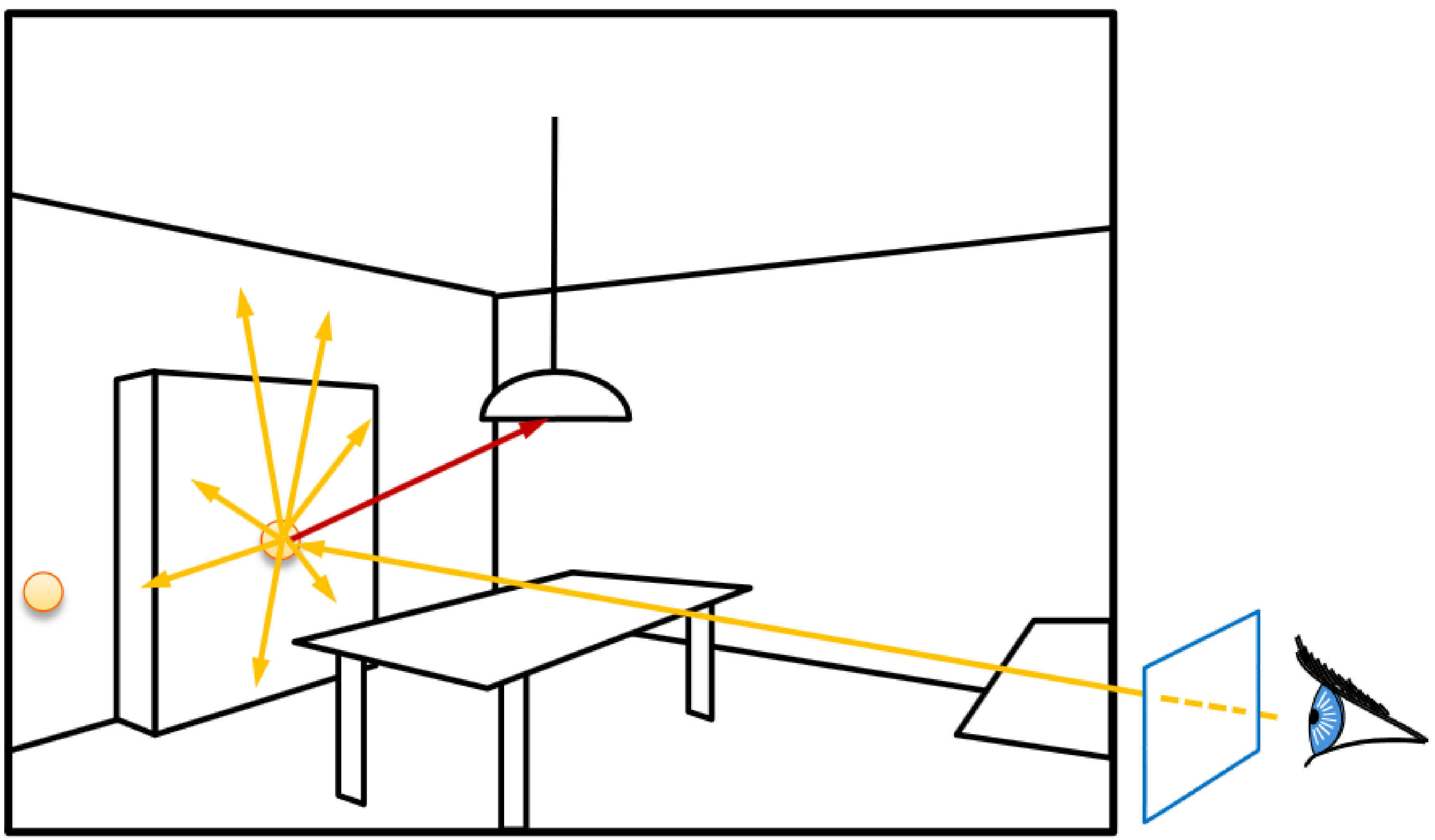
Caching approaches

- Indirect lighting is mostly smooth



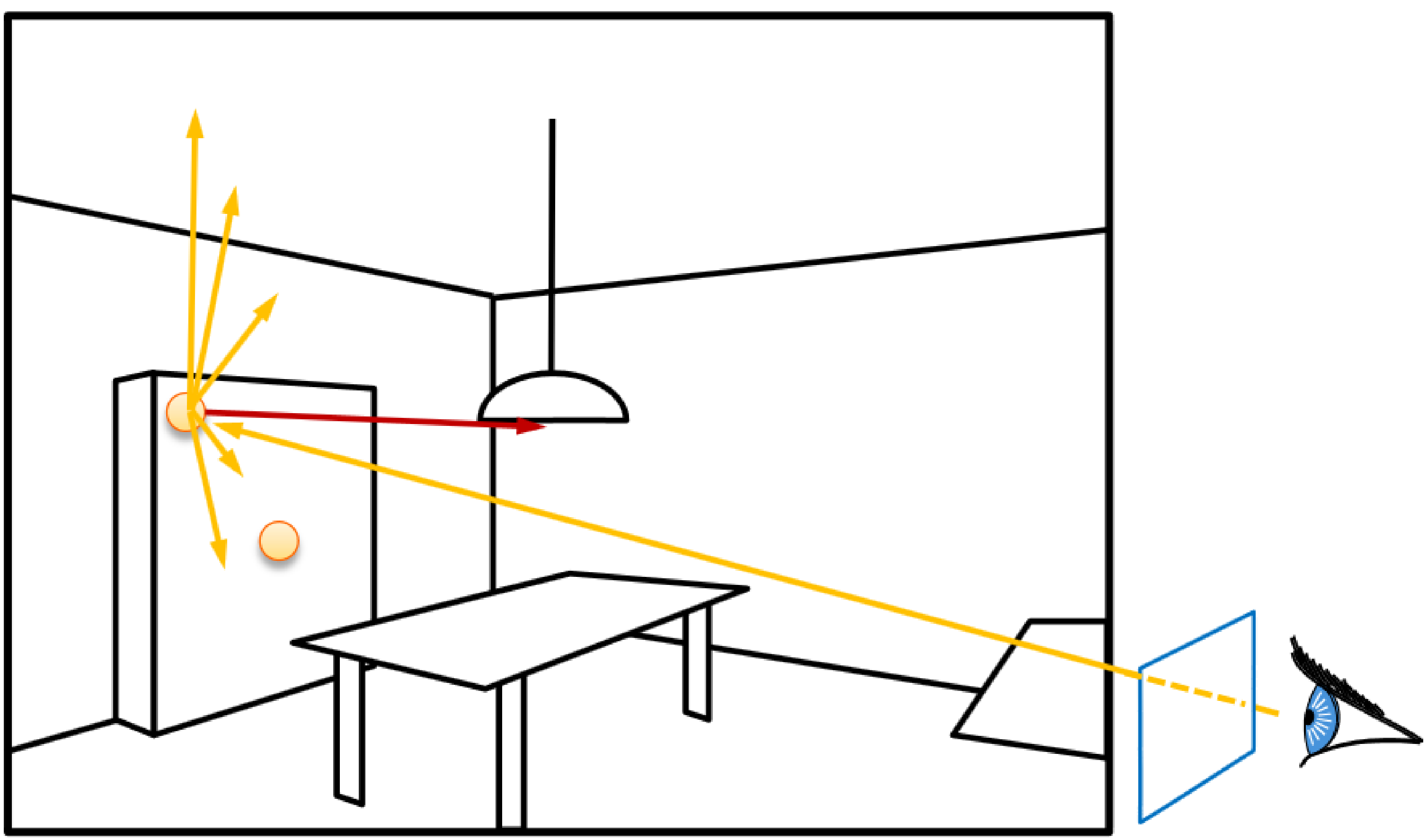
Irradiance caching

- Indirect lighting is mostly smooth



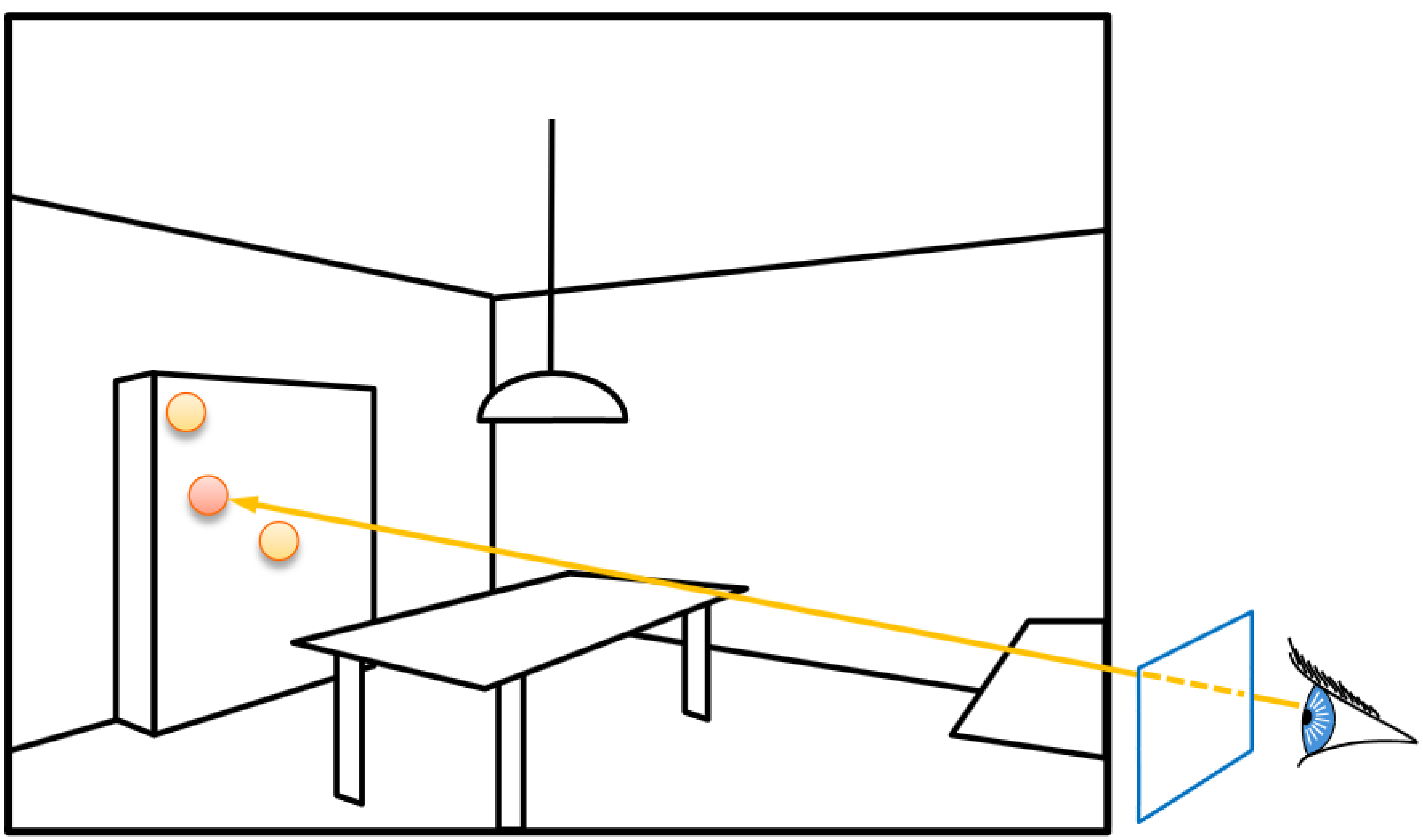
Irradiance caching

- Indirect lighting is mostly smooth



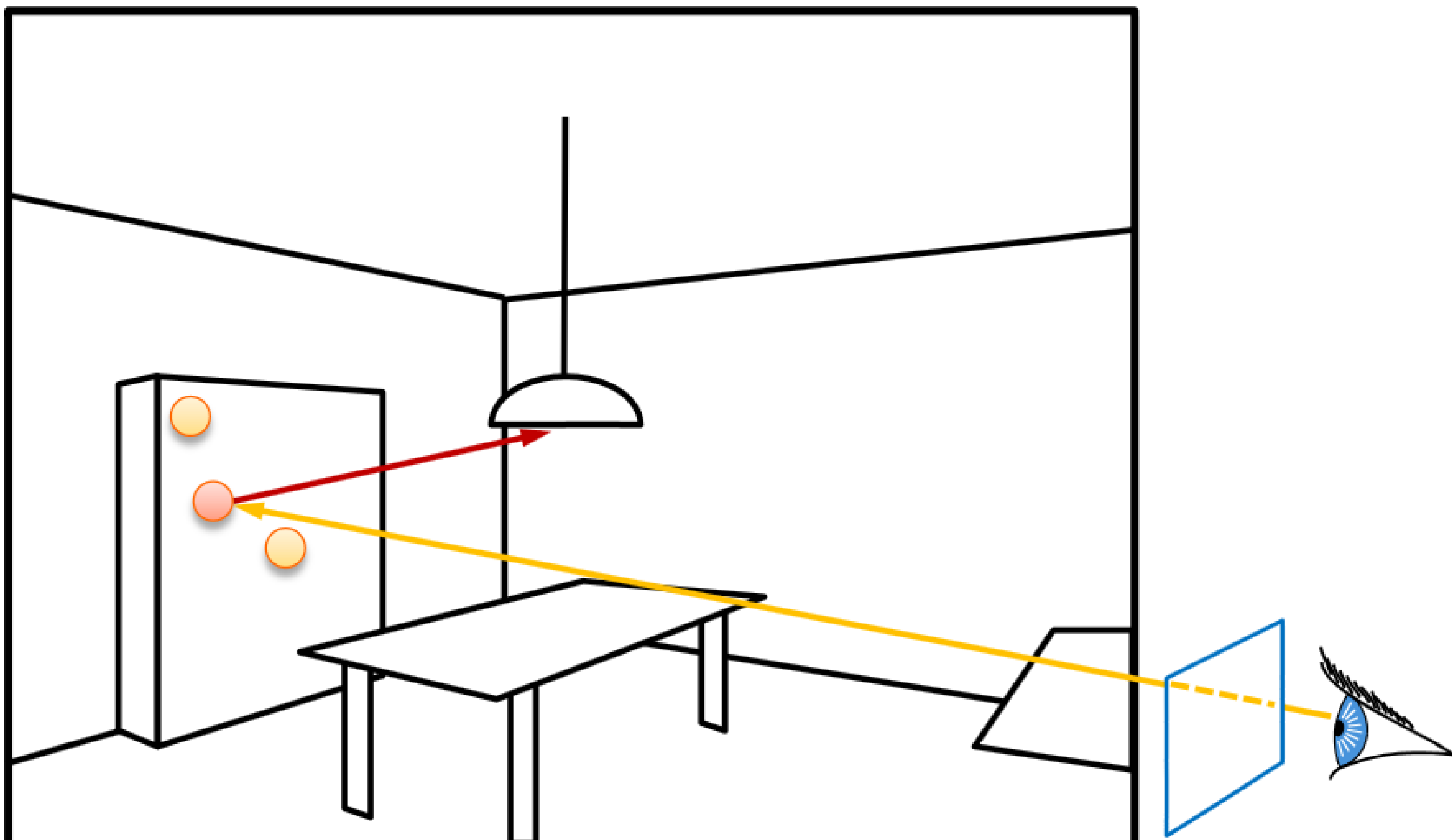
Irradiance caching

- Indirect lighting is mostly smooth
- Interpolate nearby values



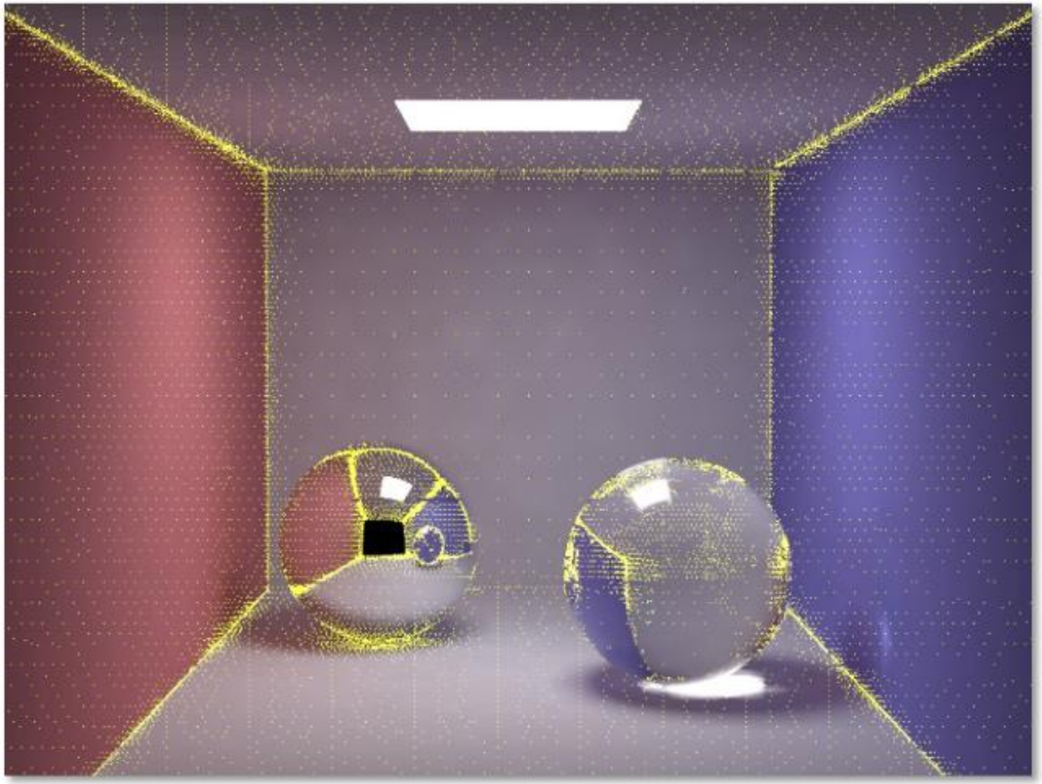
Irradiance caching

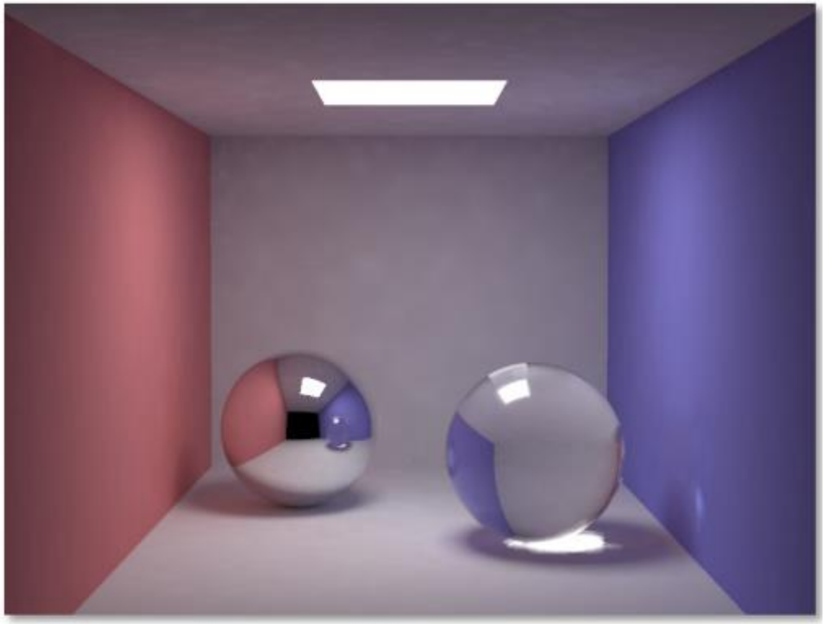
- Indirect lighting is mostly smooth
- Interpolate nearby values
- But compute full direct lighting



Irradiance caching

- Indirect lighting is mostly smooth
- Interpolate nearby values
- But compute full direct lighting



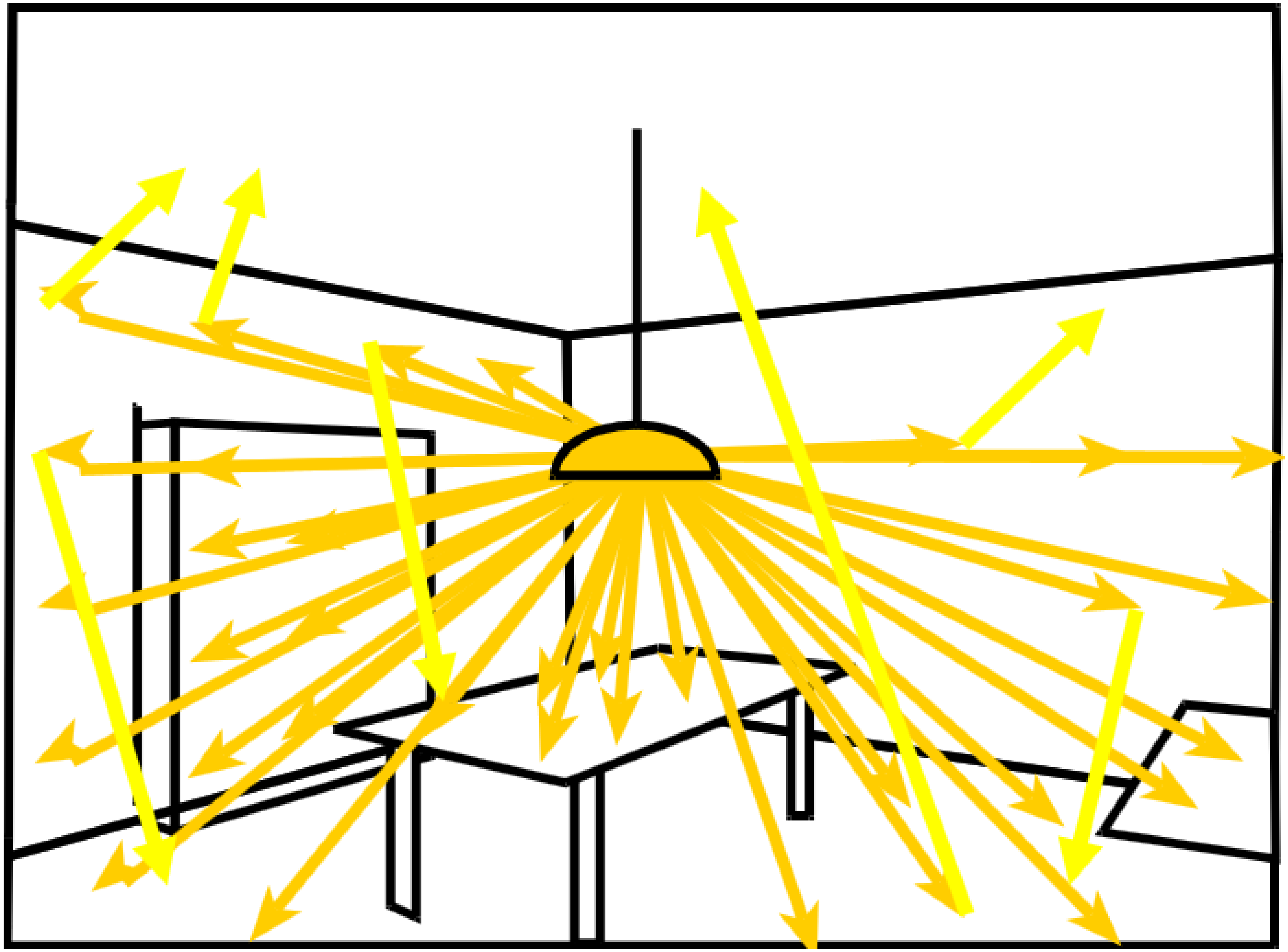


Irradiance caching

- Introduced by Ward [92]
- Lots of extensions
  - Gradient [92], SH [05], Hessian [12],…
- Complementary to photon mapping

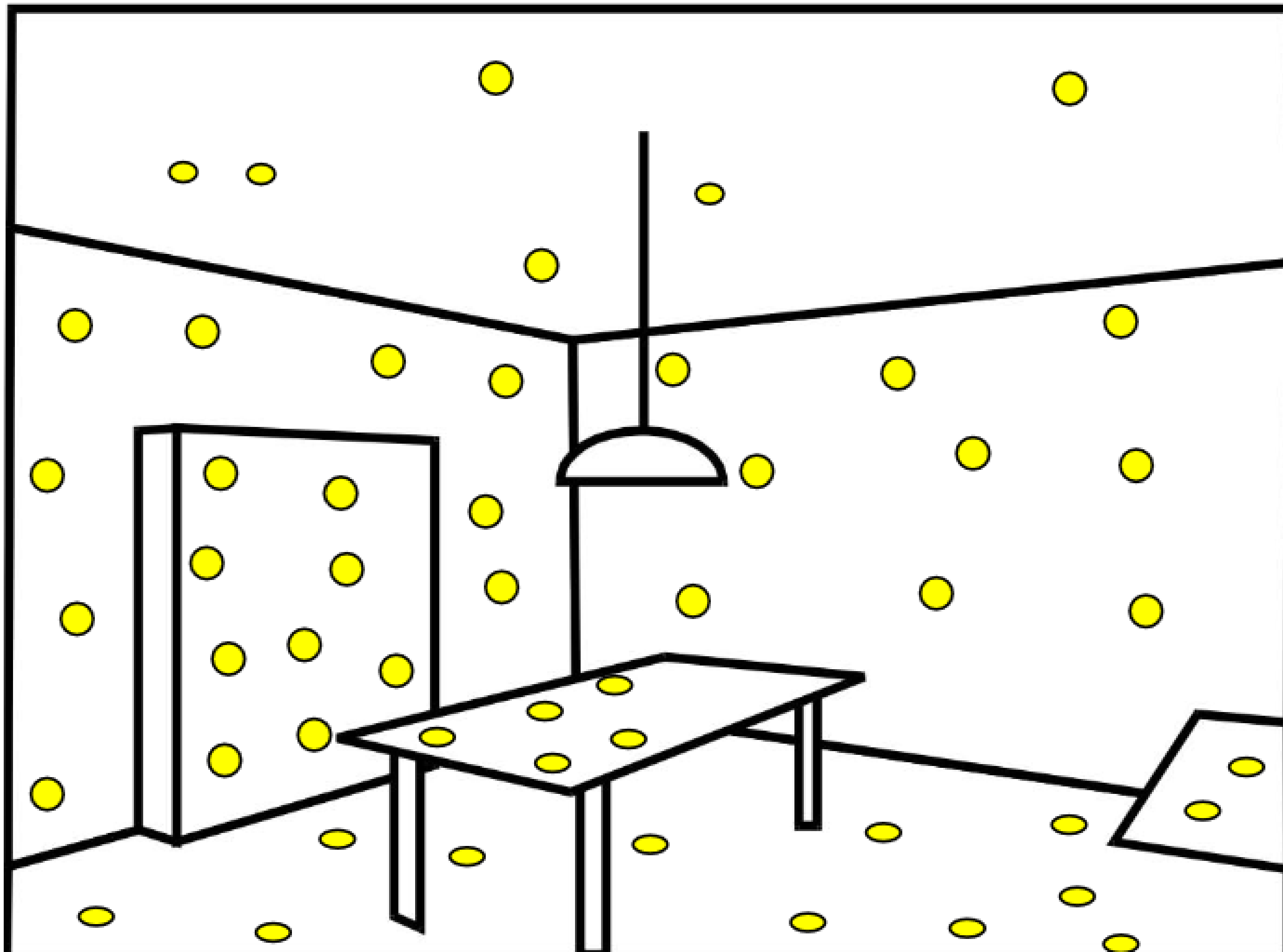
Photon mapping

- Preprocess: cast rays from light sources, and bounce randomly
- Store photons: position, power, direction



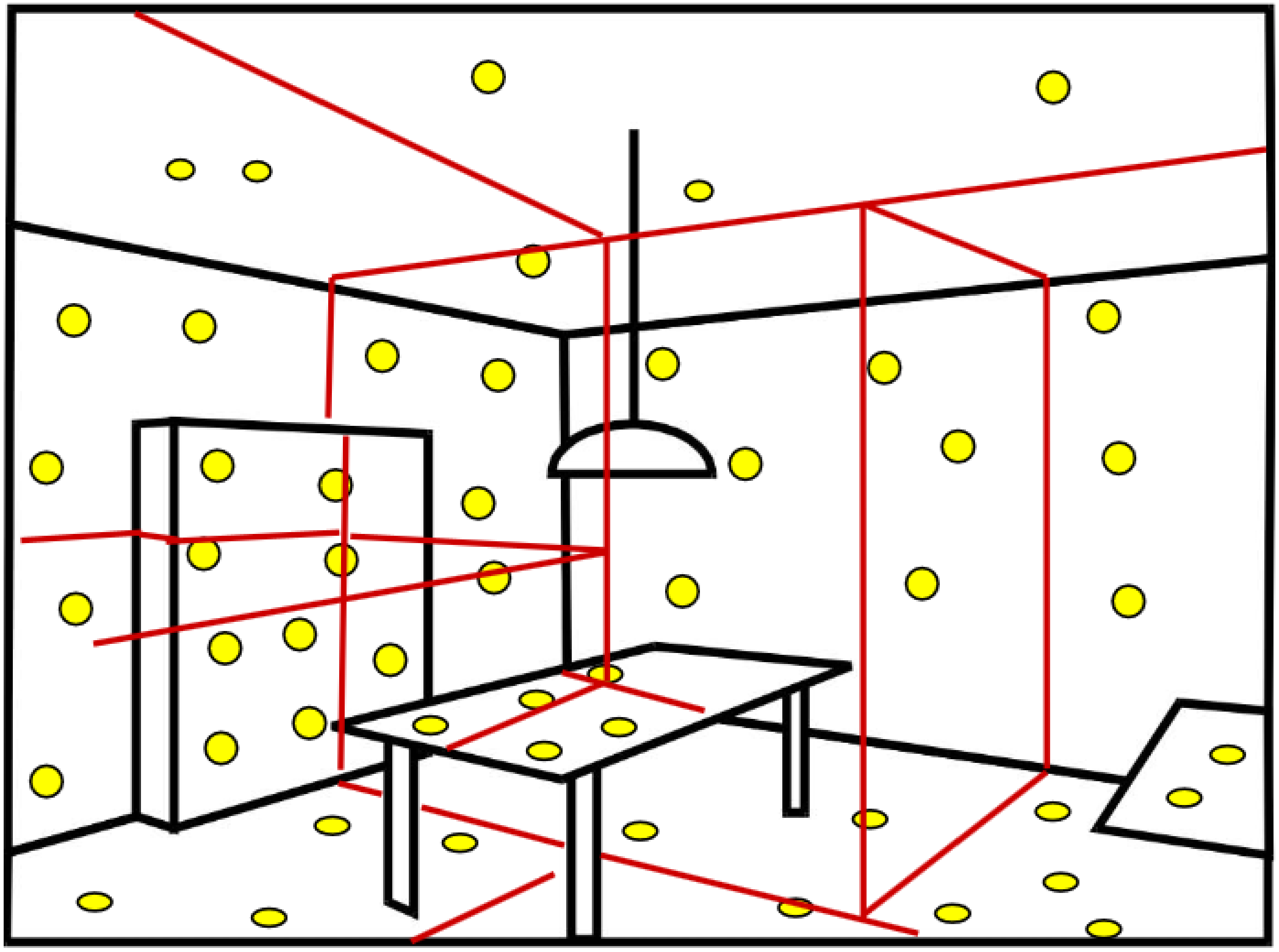
Photon mapping

- Preprocess: cast rays from light sources, and bounce randomly
- Store photons: position, power, direction



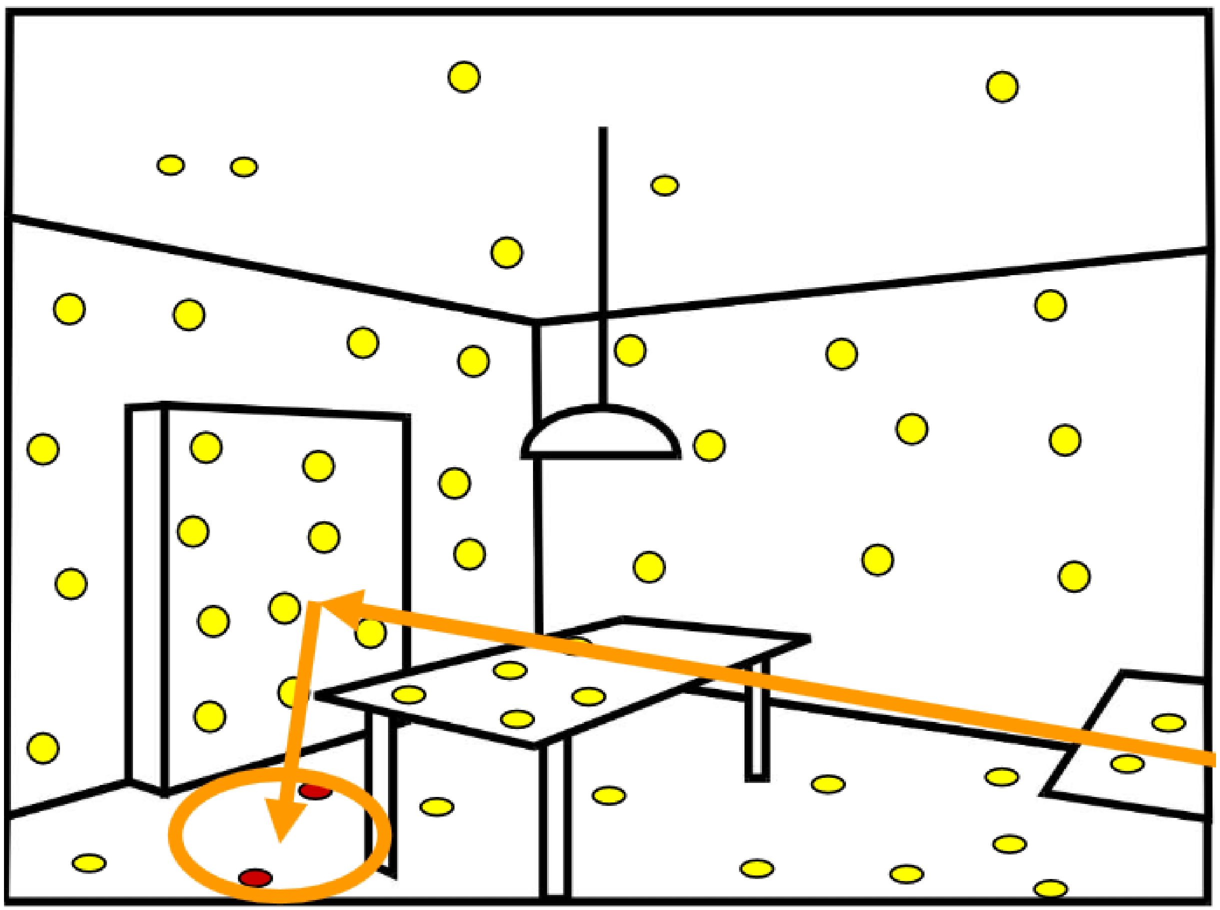
Photon mapping

- Efficiently store photons for fast access
- Use spatial hierarchical data structure (kd-tree)



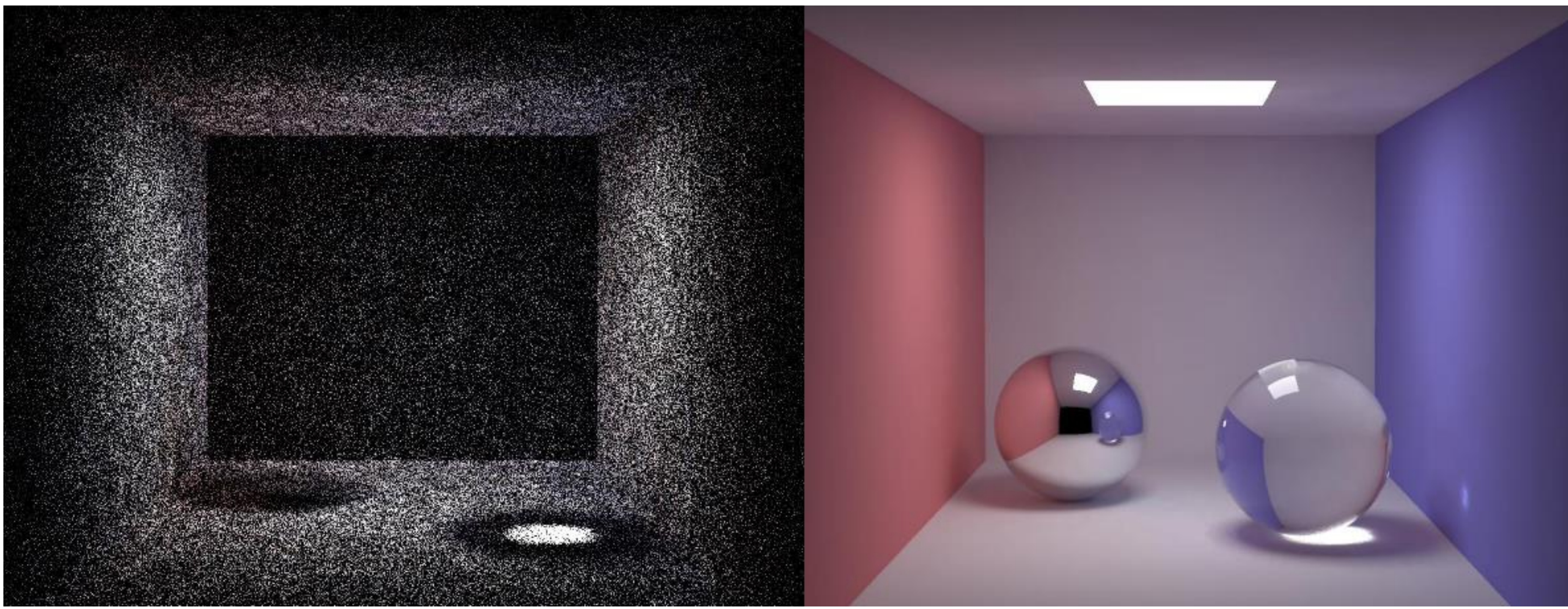
Photon mapping (rendering)

- Cast primary rays
- Secondary rays
  - Reconstruct irradiance using adjacent stored photons (k-closest photons)
- Combine with irradiance caching or other technique…



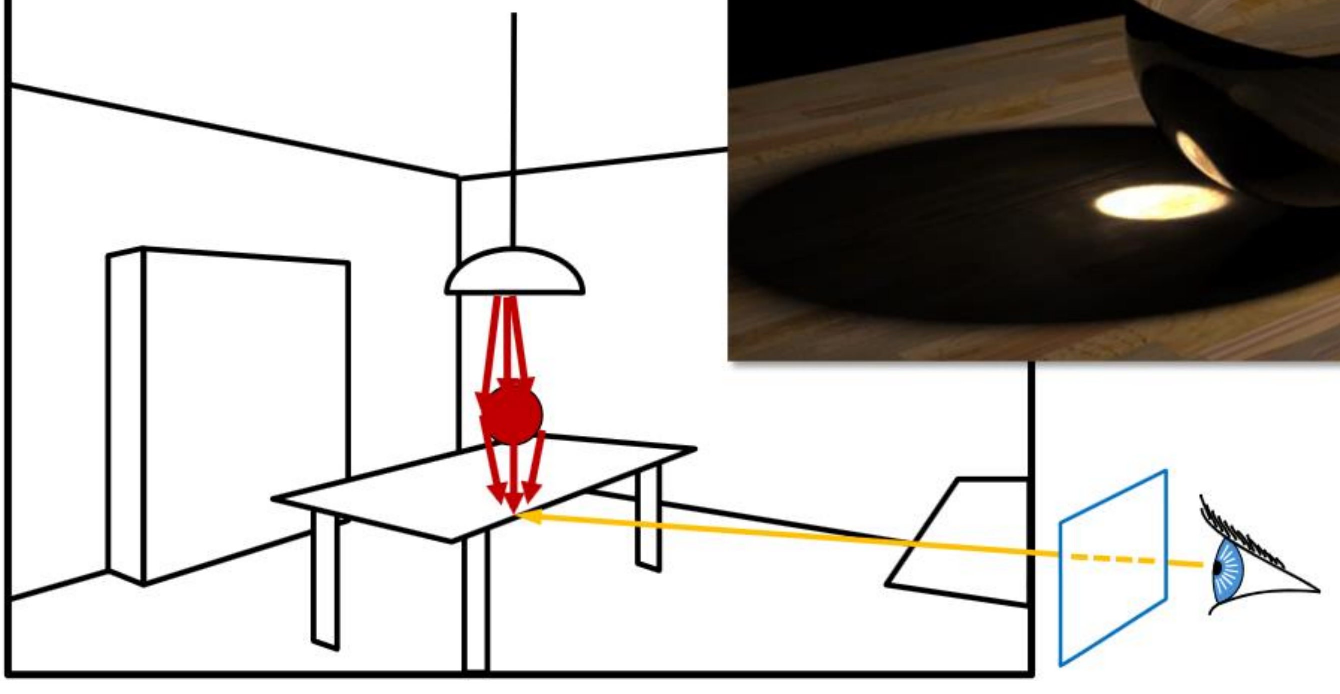
Photon mapping (rendering)

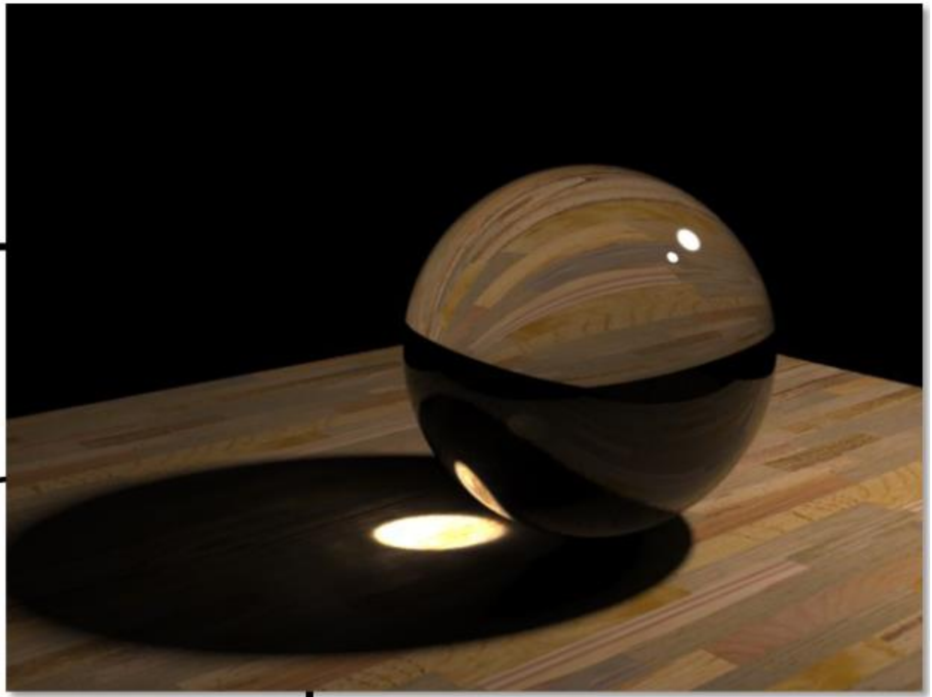
- Cast primary rays
- Secondary rays
  - Reconstruct irradiance using adjacent stored photons (k-closest photons)
- Combine with irradiance caching or other technique…



Photon mapping (rendering)

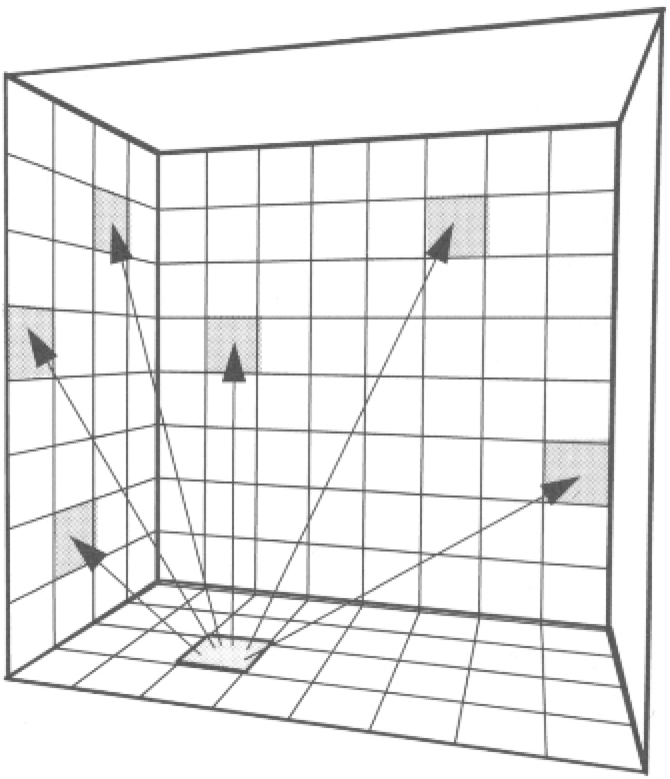
- Special photon map for caustics LS+D

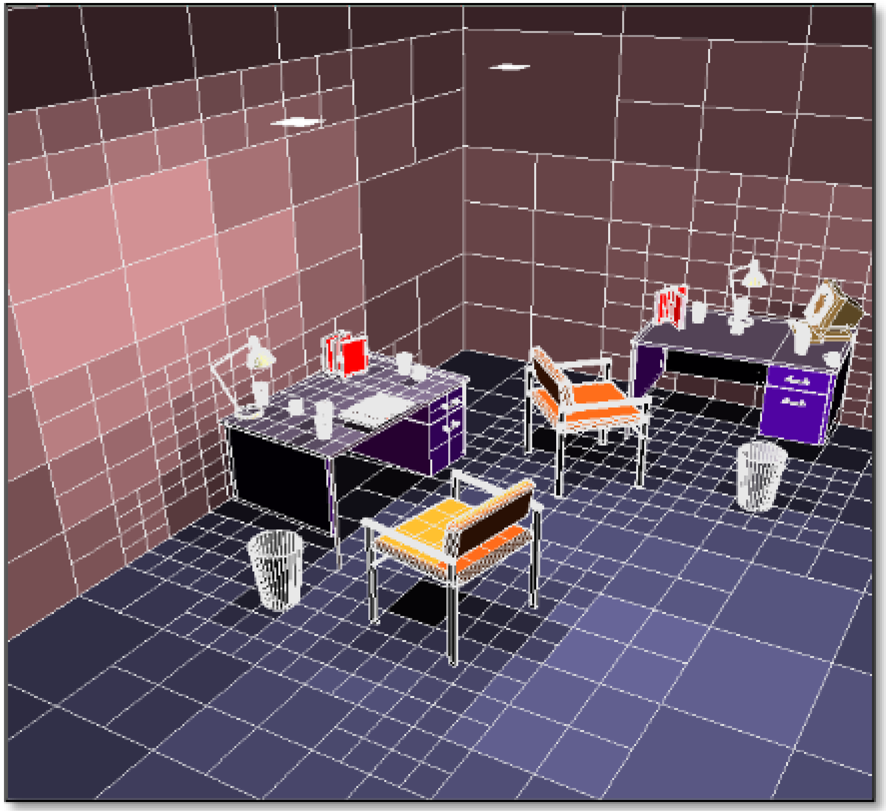




Radiosity

- Diffuse hypothesis
- Discretize scene in patchs





Radiosity

- Convert rendering equation in the form:

$$B_i = B_{ei} + f_i \sum_j F_{ij} B_j$$

- Bi = radiosity of patch i

- Fij = form factor : $F_{ij} = \int_{A_i} \int_{A_j} v(\mathbf{x},\mathbf{x}') \frac{\cos(\theta)\cos(\theta')}{\pi r^2} \, d\mathbf{x} d\mathbf{x}'$

Radiosity

- Matricial equation

$$\begin{bmatrix} B_1 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} B_{e,1} \\ \vdots \\ B_{e,n} \end{bmatrix} + \begin{bmatrix} f_1 F_{1,1} & \cdots & f_1 F_{1,n} \\ \vdots & \ddots & \vdots \\ f_n F_{n,1} & \cdots & f_n F_{n,n} \end{bmatrix} \begin{bmatrix} B_1 \\ \vdots \\ B_n \end{bmatrix}$$

Radiosity

- Matricial equation

$$\begin{bmatrix} B_1 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} B_{e,1} \\ \vdots \\ B_{e,n} \end{bmatrix} + \begin{bmatrix} f_1 F_{1,1} & \cdots & f_1 F_{1,n} \\ \vdots & \ddots & \vdots \\ f_n F_{n,1} & \cdots & f_n F_{n,n} \end{bmatrix} \begin{bmatrix} B_1 \\ \vdots \\ B_n \end{bmatrix}$$

- - Long / memory / discontinuities (geom) / diffuse only
- + view independant / handle complex scenes /

- Mostly used in architecture and video games (light maps)

Radiosity

- More recent methods
  - Precomputed radiance transfer
  - Many lights (VPLs) and extensions

References

- MIT:
  - http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/
- Standford:
  - http://candela.stanford.edu/cs348b-14/doku.php
- Siggraph:
  - http://blog.selfshadow.com/publications/s2014-shading-course/
  - http://blog.selfshadow.com/publications/s2013-shading-course/
- Image synthesis & OpenGL:
  - http://romain.vergne.free.fr/blog/?page_id=97
- Path tracing and global illum:
  - http://www.graphics.stanford.edu/courses/cs348b-01/course29.hanrahan.pdf
  - http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/realistic_raytracing.html
- GLSL / Shadertoy:
  - https://www.opengl.org/documentation/glsl/
  - https://www.shadertoy.com/
  - http://www.iquilezles.org/
- http://fileadmin.cs.lth.se/cs/Education/EDAN30/lectures/L2-rt.pdf
- http://csokavar.hu/raytrace/imm6392.pdf
- http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/realistic_raytracing.html