Metropolis Virtual Point Light Rendering

Sébastien Speierer

EPFL

Spring 2016

Realistic Graphic Lab

Professor W. Jakob

Outline

- Rendering equation
- VPL method
- Metropolis-Hastings algorithm
- Implementation
- Results
- Questions

Rendering equation

Rendering - the equation

$$L(x' \to x'') = \boxed{L_e(x' \to x'')} + \boxed{\int_{M^2} L(x \to x')G(x \to x')f(x \to x' \to x'')\,\partial A(x)}$$

direct lighting

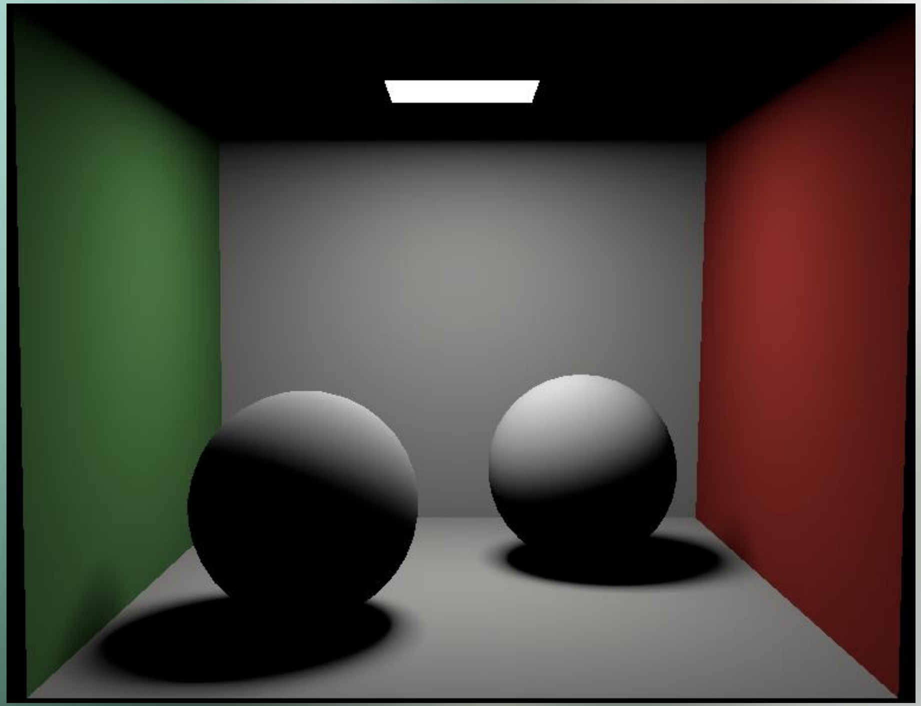
indirect lighting

- $L(x' \to x'')$: incoming radiance at x'' from x'
- $L_e(x' \to x'')$: emitted radiance at x' in the direction of x''
- $f(x \to x' \to x'')$: BRDF at x'
- M^2: scene surface
- $G(x \to x')$: geometric term between x and x'

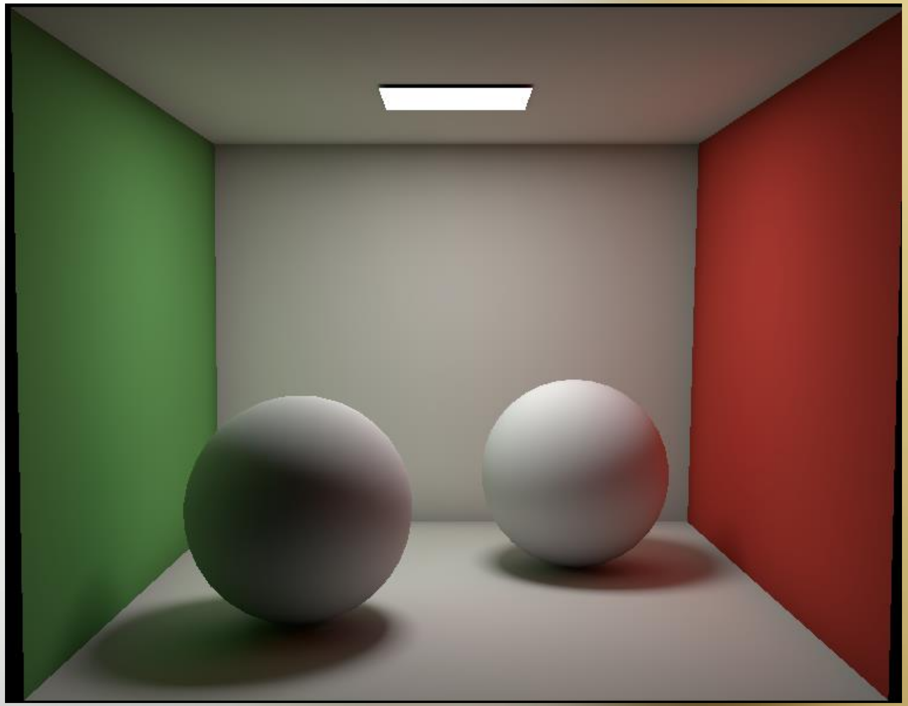
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Rendering - global illumination

$$L(x' \rightarrow x'') = L_e(x' \rightarrow x'') + L_i\,(x' \rightarrow x'')$$



direct lighting



direct + indirect lighting

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

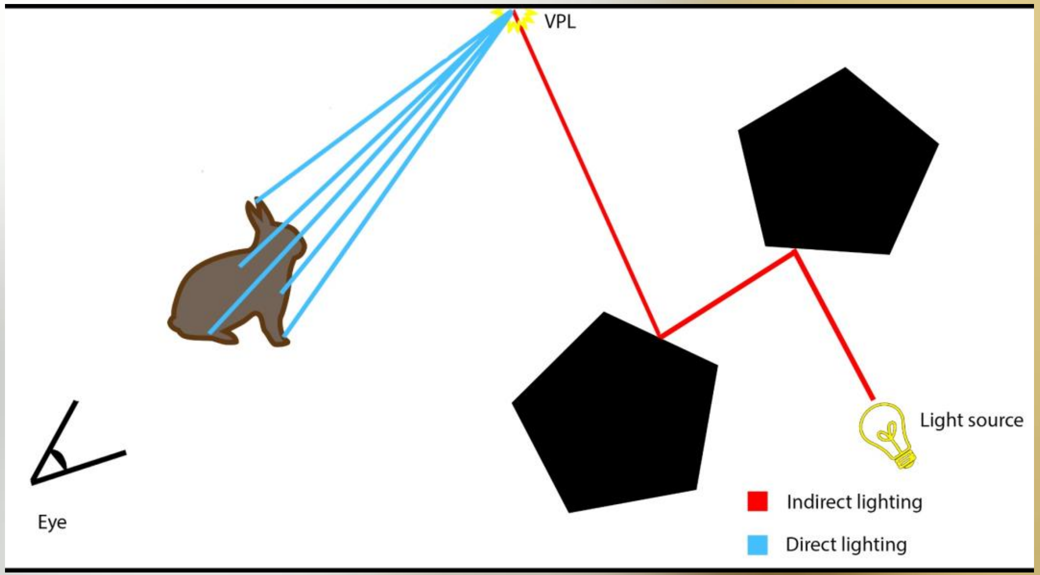
VPL method

VPL method - what does it do?

- solve the rendering equation
- use GPU
- progressive
- fast
- user real-time interaction
- converge to the correct solution

VPL method - what is a VPL?

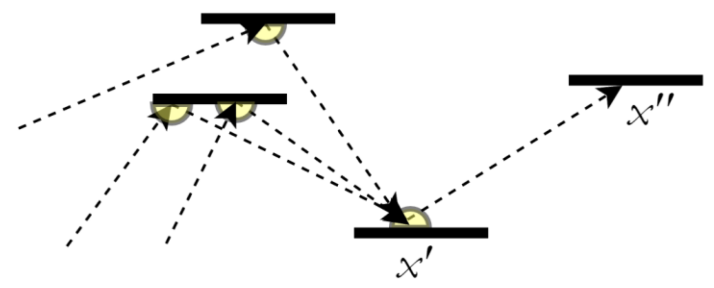
- Virtual Point Light
- similar to a photon in photon mapper
- point on a surface in the scene
- incoming radiance field along a path
- use them to render with a GPU
- generate thousands of them



ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

VPL method - the equation (path space)

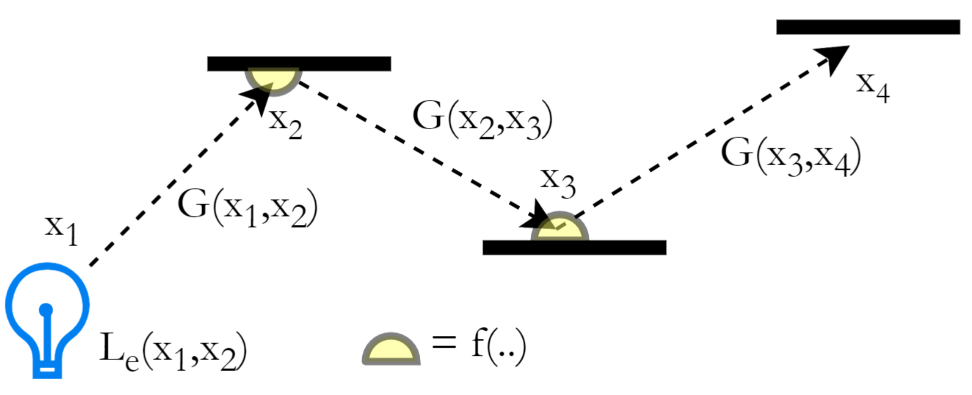
$$L(x' \to x'') = L_e(x', x'') + \sum_{k=1}^{\infty} \int_{P_k^{x'}} L(\bar{\boldsymbol{x}}) f(x_{k-1}, x', x'') \partial \bar{\boldsymbol{x}}$$



$$\int_{P_k^x} L(\bar{\boldsymbol{x}}) f(x_{k-1}, x, x') d\bar{\boldsymbol{x}} = \int_{M^2} \dots \int_{M^2} f(x_{k-1}, x, x') S(x_1, \dots, x_k)\, \partial A(x_1) \dots \partial A(x_{k-1})$$

$$S(x_1, \dots, x_k) = L_e(x_1, x_2) \prod_{i=1}^{k-1} G(x_i, x_{i+1}) \prod_{j=1}^{k-2} f(x_j, x_{j+1}, x_{j+2})$$

- P is the multi-dimensional space of all possible paths
- with $\bar{\boldsymbol{x}} \in P$, x_i for $1 \le i \le k$ its ith points
- $P_k^x = \{\bar{\boldsymbol{x}} : \bar{\boldsymbol{x}} \in P \text{ and } x_k = x\}$ and length k



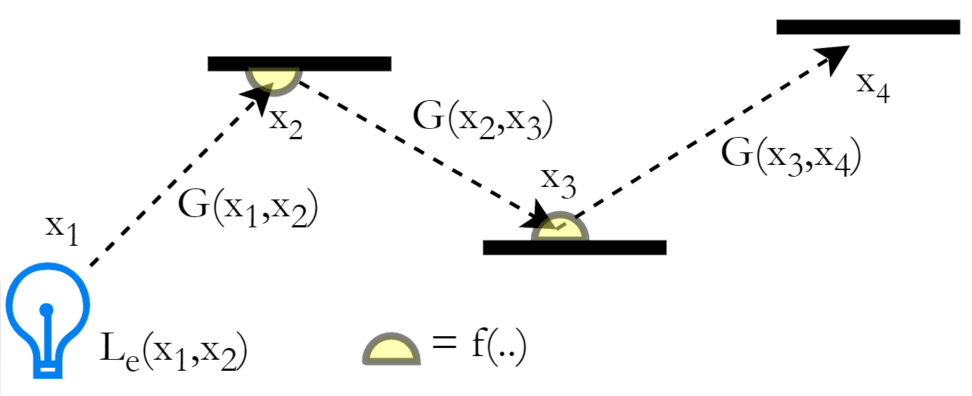
VPL method - the equation (path space)

$$L(x' \to x'') = L_e(x', x'') + \sum_{k=1}^{\infty} \int_{M^2} G(x, x') f(x, x', x'') \int_{P_{k-1}^{x}} L(\overline{\boldsymbol{x}}) f(x_{k-1}, x, x') \partial \overline{\boldsymbol{x}} \; \partial A(x)$$

$$\int_{P_k^x} L(\overline{\boldsymbol{x}}) f(x_{k-1}, x, x') d\overline{\boldsymbol{x}} = \int_{M^2} \dots \int_{M^2} f(x_{k-1}, x, x') S(x_1, \dots, x_k) \, \partial A(x_1) \dots \partial A(x_{k-1})$$

$$S(x_1, \dots, x_k) = L_e(x_1, x_2) \prod_{i=1}^{k-1} G(x_i, x_{i+1}) \prod_{j=1}^{k-2} f(x_j, x_{j+1}, x_{j+2})$$

- P is the multi-dimensional space of all possible paths
- with $\overline{\boldsymbol{x}} \in P$, x_i for $1 \leq i \leq k$ its ith points
- $P_k^x = \{\overline{\boldsymbol{x}}: \overline{\boldsymbol{x}} \in P \; and \; x_k = x\}$ and length k

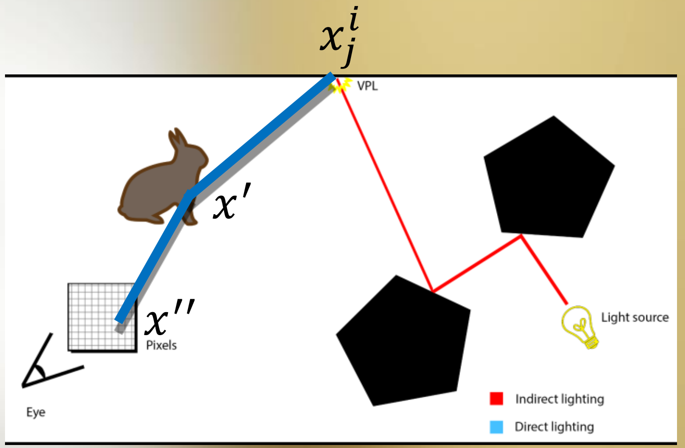


VPL method - Monte Carlo approach

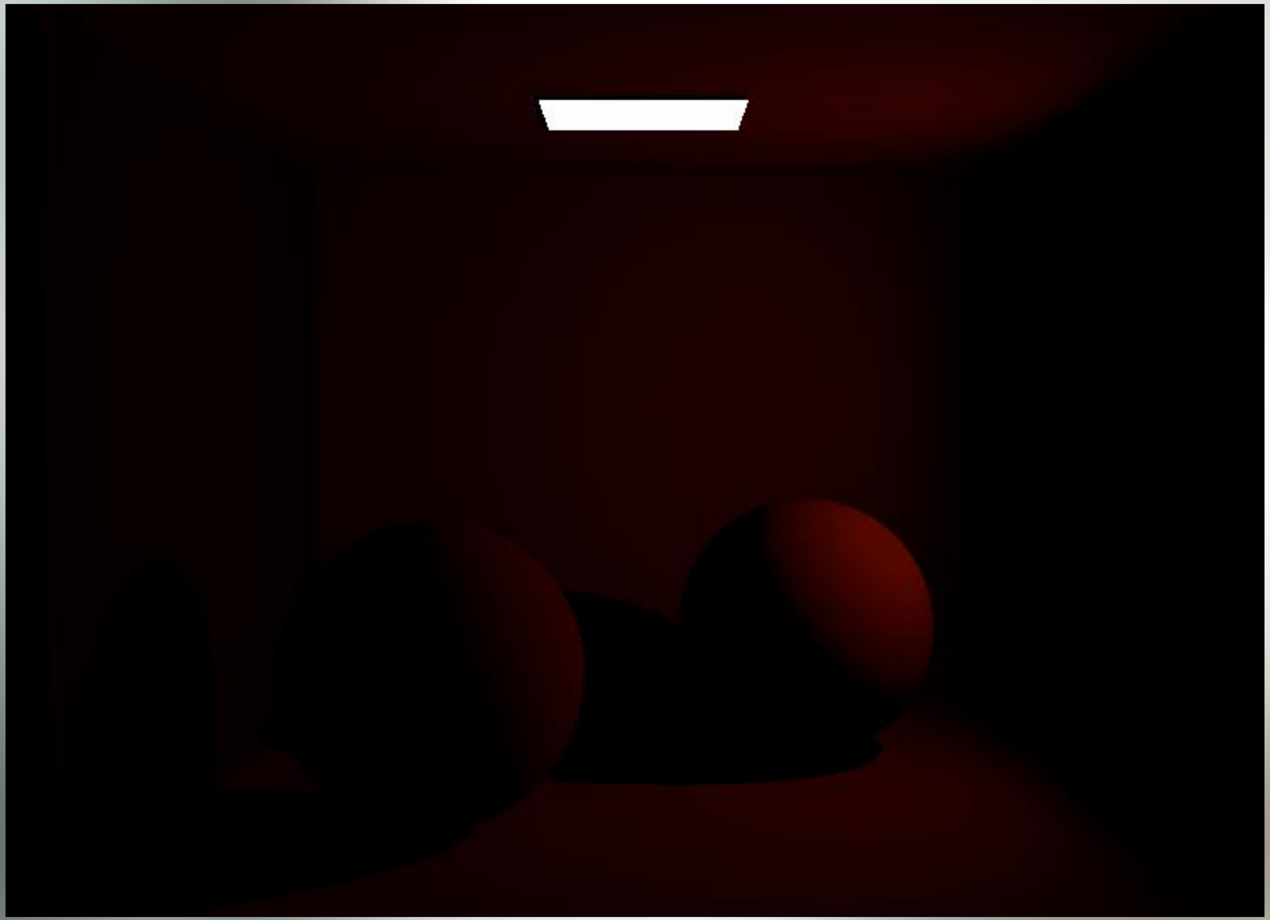
$$L(x' \to x'') = L_e(x', x'') + \sum_{k=1}^{\infty} \int_{M^2} G(x, x') f(x, x', x'') \int_{P_{k-1}^{x}} f(x_{k-1}, x, x') L(\bar{\boldsymbol{x}}) \partial \bar{\boldsymbol{x}} \, \partial A(x)$$

$$L_i(x' \to x'') \approx \frac{1}{N} \frac{1}{M} \underbrace{\sum_{k=1}^{\infty} \sum_{i=1}^{N} \sum_{j=1}^{M}}_{\text{accumulation}} \underbrace{G(x_j^i, x') f(x_j^i, x', x'') f(x_{k-1}^i, x_j^i, x')}_{\text{GPU}} \underbrace{\frac{1}{p(x_j^i)} \frac{L(\bar{\boldsymbol{x}}_i)}{p(\bar{\boldsymbol{x}}_i)}}_{\text{path tracer}}$$

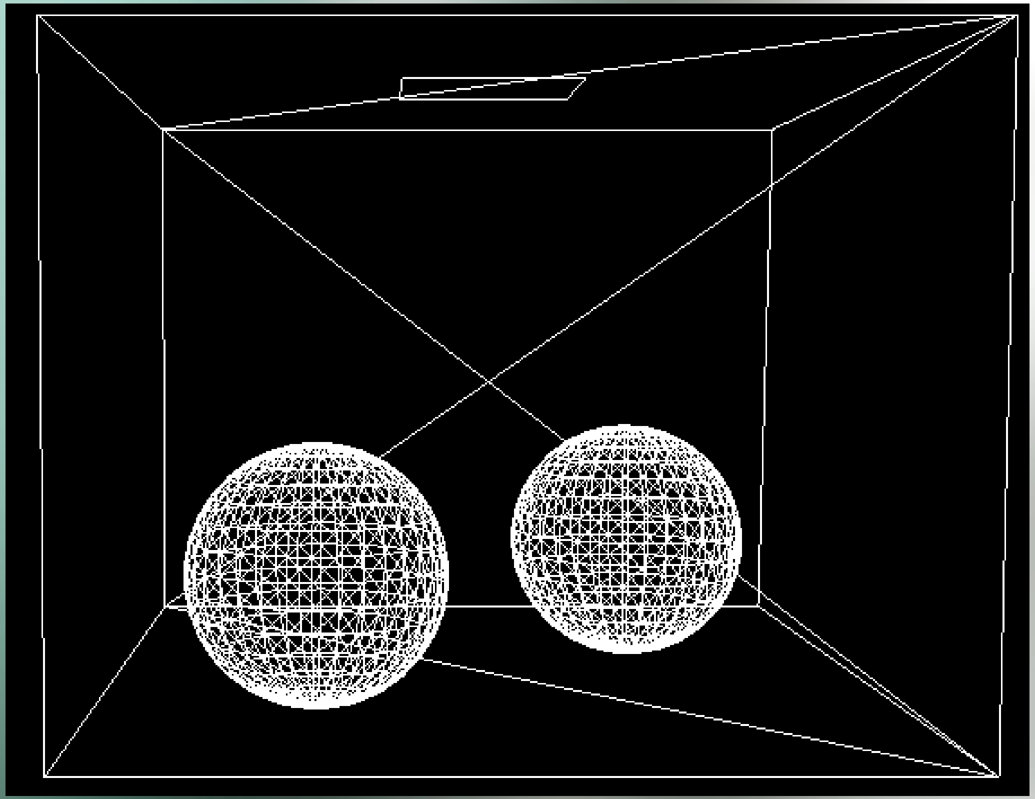
$$\Delta L_i(x' \to x'') = \underbrace{G(x_j^i, x') f(x_j^i, x', x'') f(x_{k-1}^i, x_j^i, x')}_{\text{GPU}} \underbrace{\frac{1}{p(x_j^i)} \frac{L(\bar{\boldsymbol{x}}_i)}{p(\bar{\boldsymbol{x}}_i)}}_{\text{path tracer}}$$



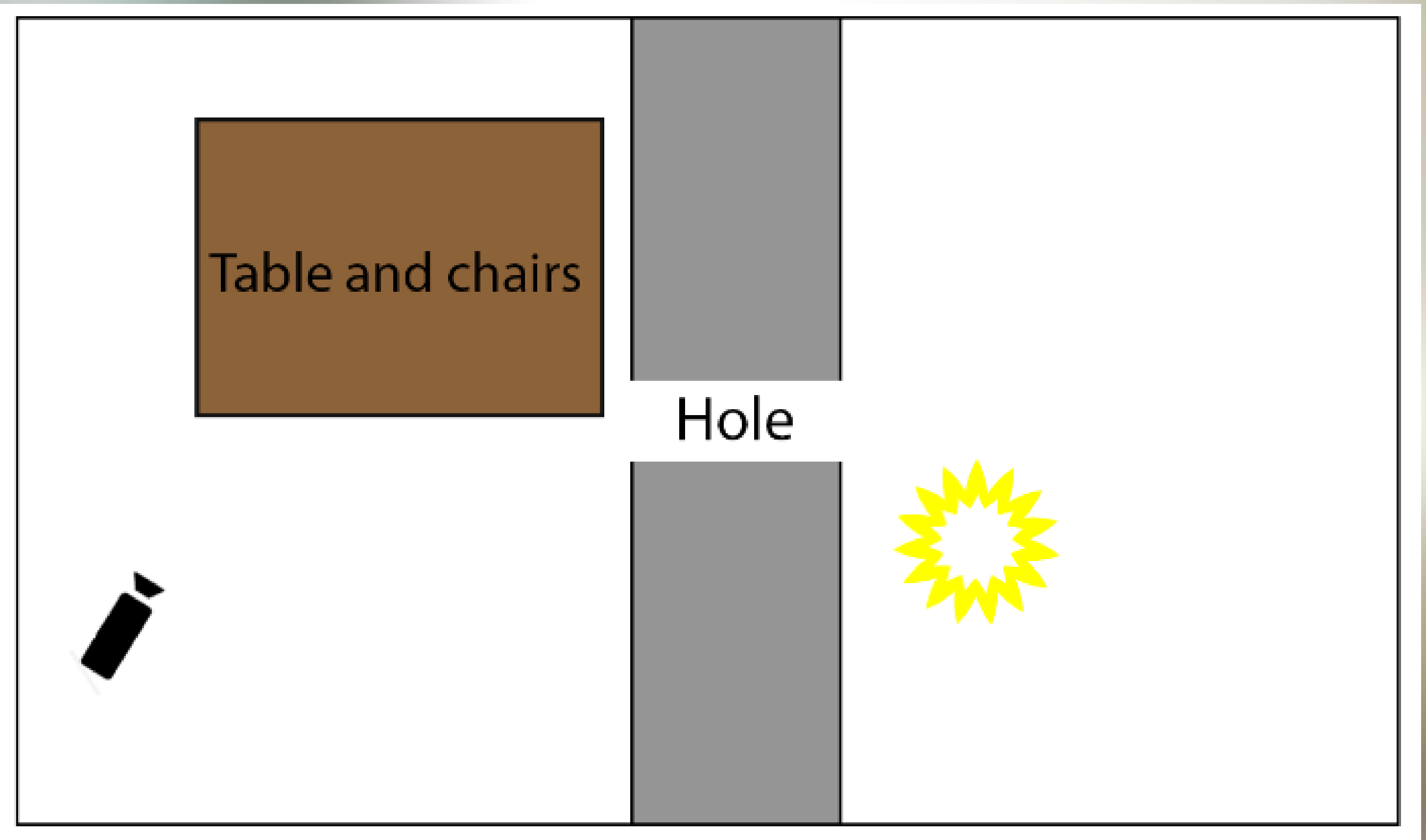
VPL method - accumulation of VPLs



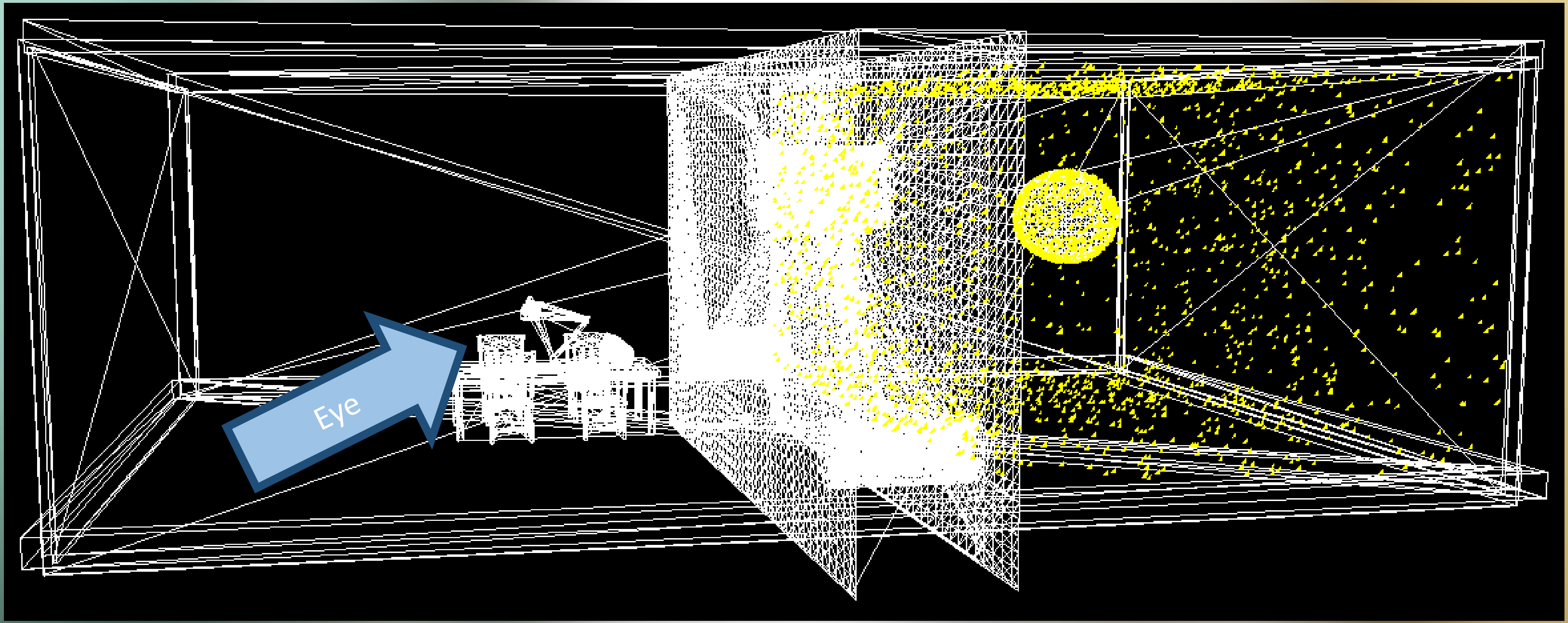
VPL method - Virtual Point Lights (VPL)



VPL method - issue



VPL method - issue (2)



Metropolis-Hastings algorithm

Metropolis-Hastings - the idea

- generate samples $\boldsymbol{X}$ distributed proportionally to their contribution

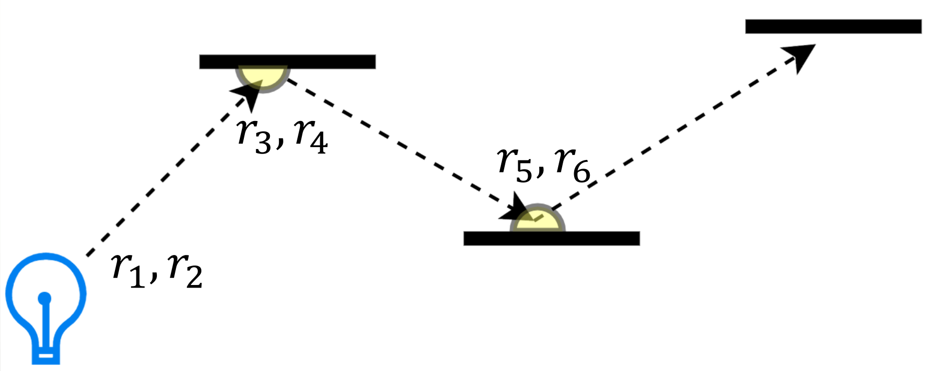
$$p(\boldsymbol{X}) = \frac{L(\boldsymbol{X})}{\int_P L(\boldsymbol{X}) \partial \boldsymbol{X}}$$

- iteratively produce statistically correlated samples
- next sample being dependent on the current sample
- accept / reject based on the acceptance probability

$$a(\boldsymbol{X} \rightarrow \boldsymbol{X}') = \min(1, \frac{L(\boldsymbol{X\prime})}{L(\boldsymbol{X})})$$

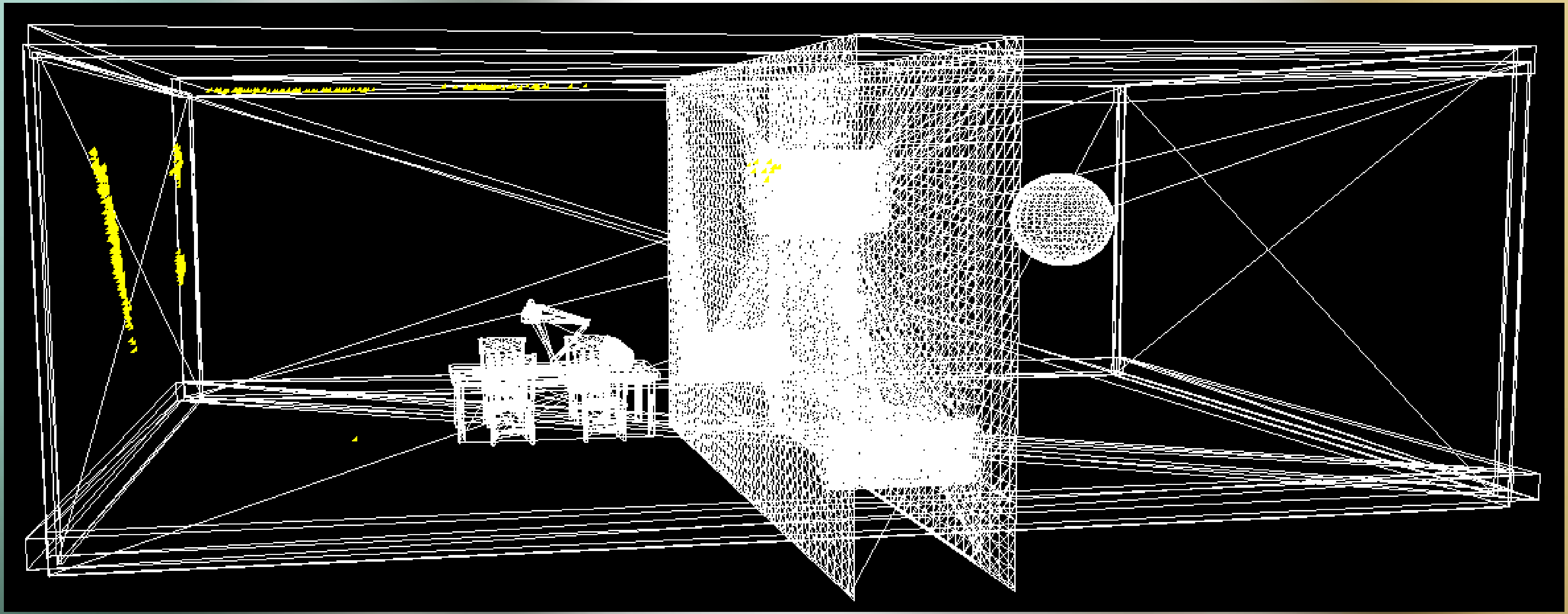
Metropolis-Hastings - path creation

- A path as a vector $\boldsymbol{X}$ of N real numbers r_n

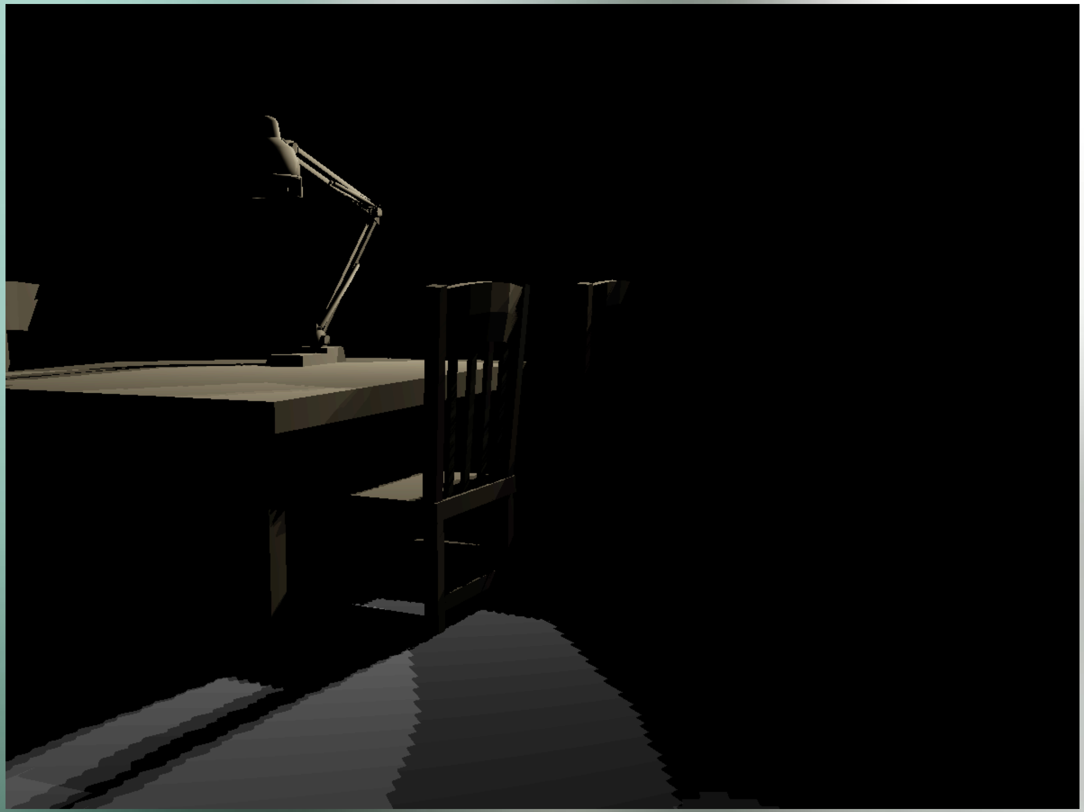


- Mutations:
  - Probability p of taking a large mutation
  - Large: $r'_n = \mathcal{N}(0,1)$
  - Small: $r'_n = \mathcal{N}(r_n, \sigma^2)$

Metropolis-Hastings - results



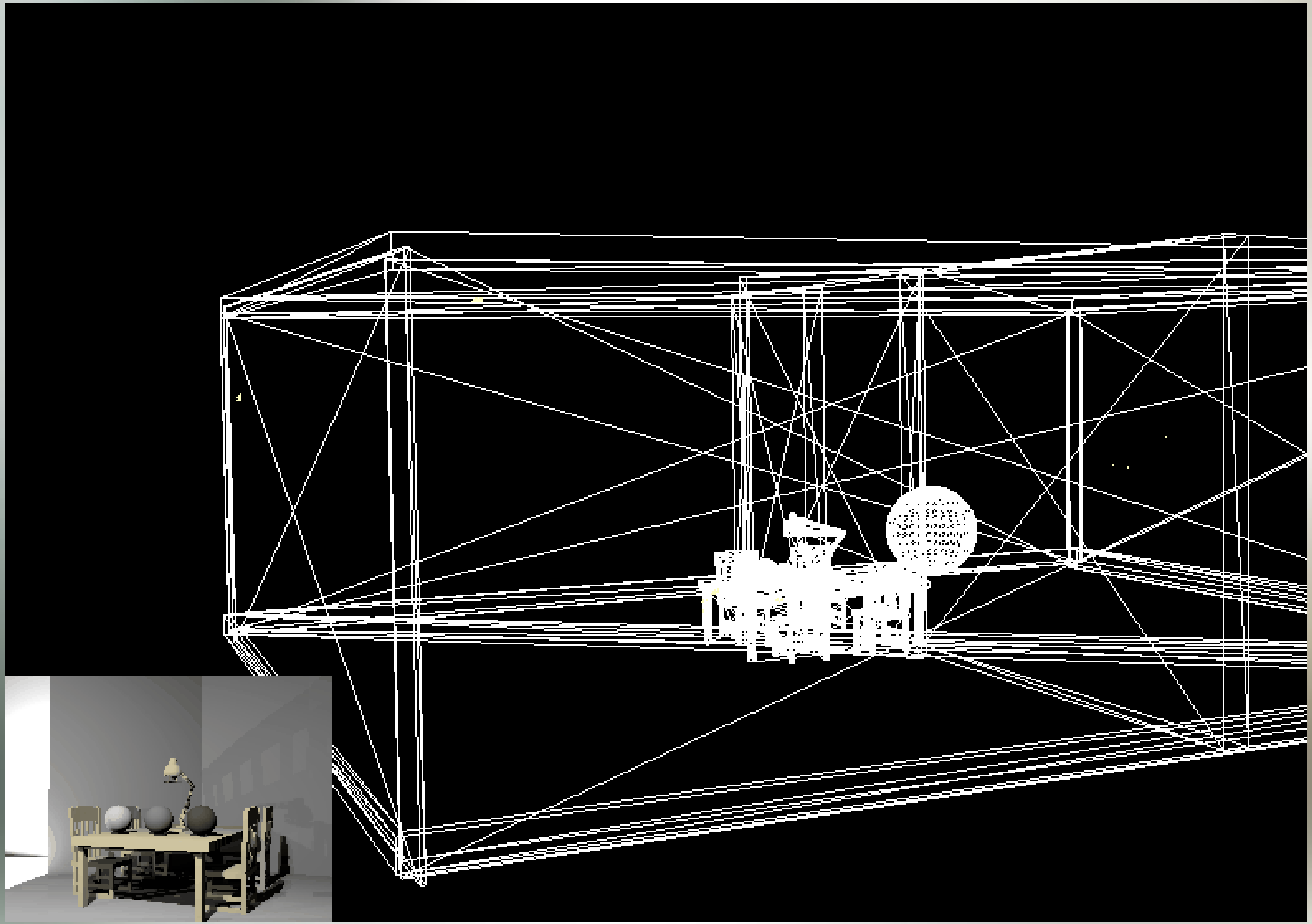
Metropolis-Hastings - results (2)



standard method – 3000 VPLs

metropolis method – 3000 VPLs

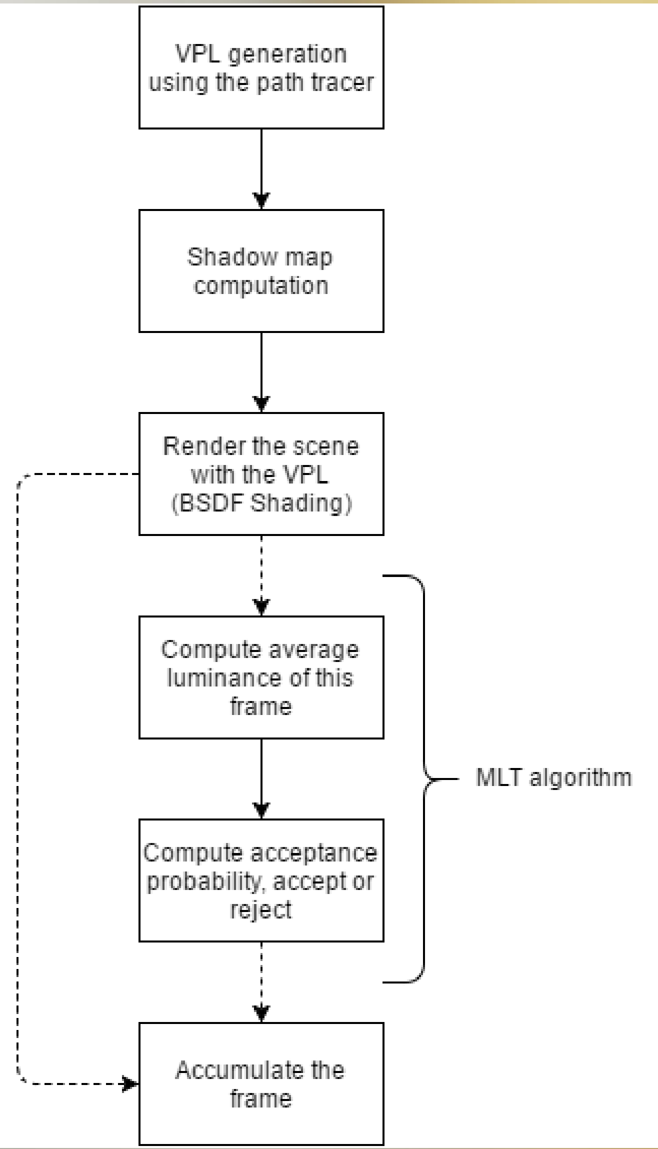
Metropolis-Hastings - results (3)



Implementation

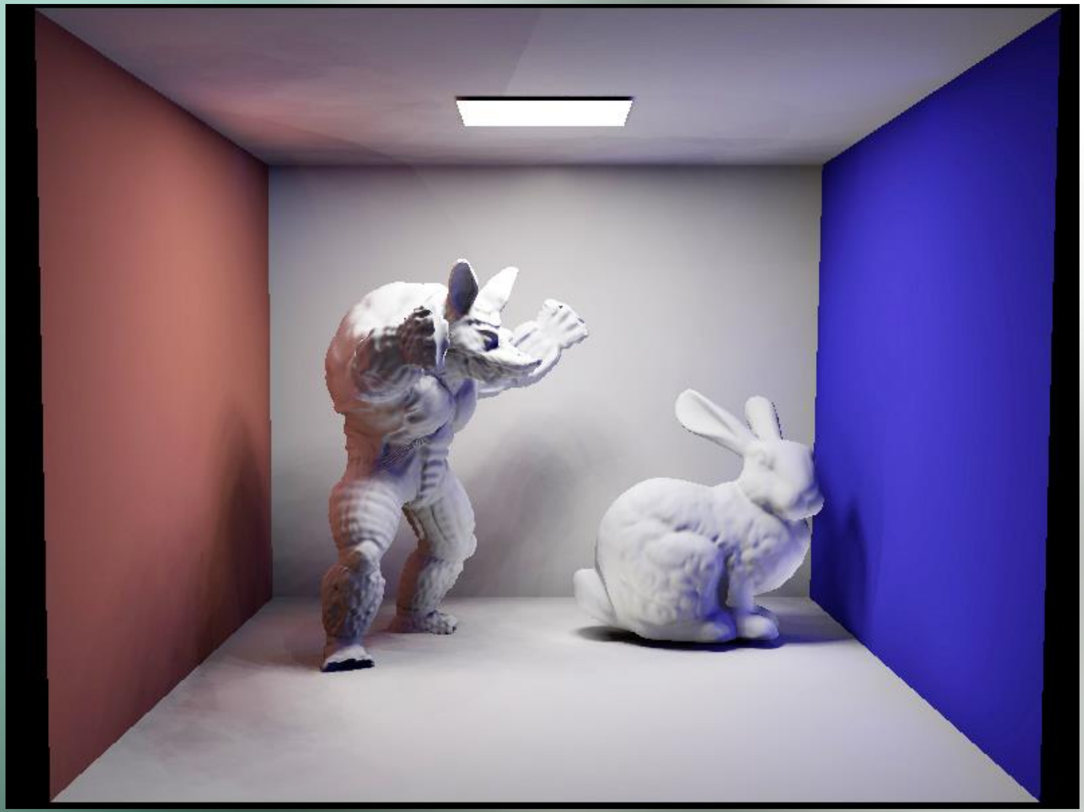
Implementation - pipeline

- Generate VPL
- Compute shadow map
- Render the scene
- Compute average frame illuminance
- Accept / reject VPL
- Accumulate the frame

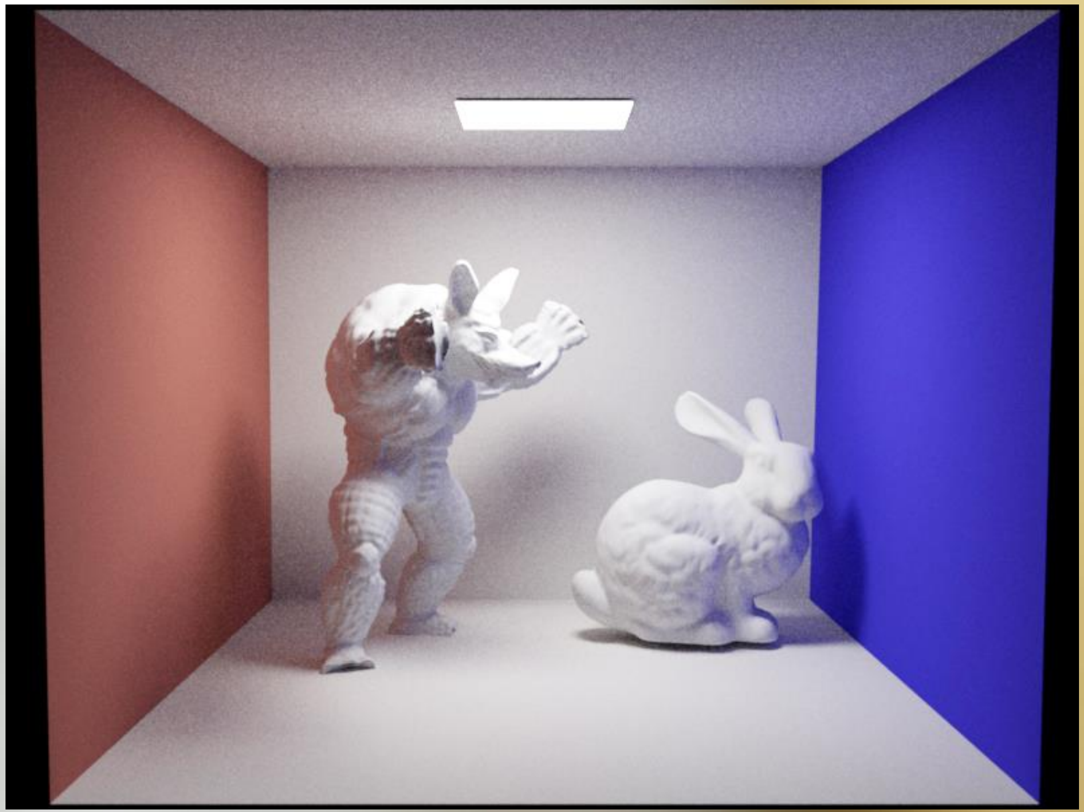


Results

Results - high resolution meshes

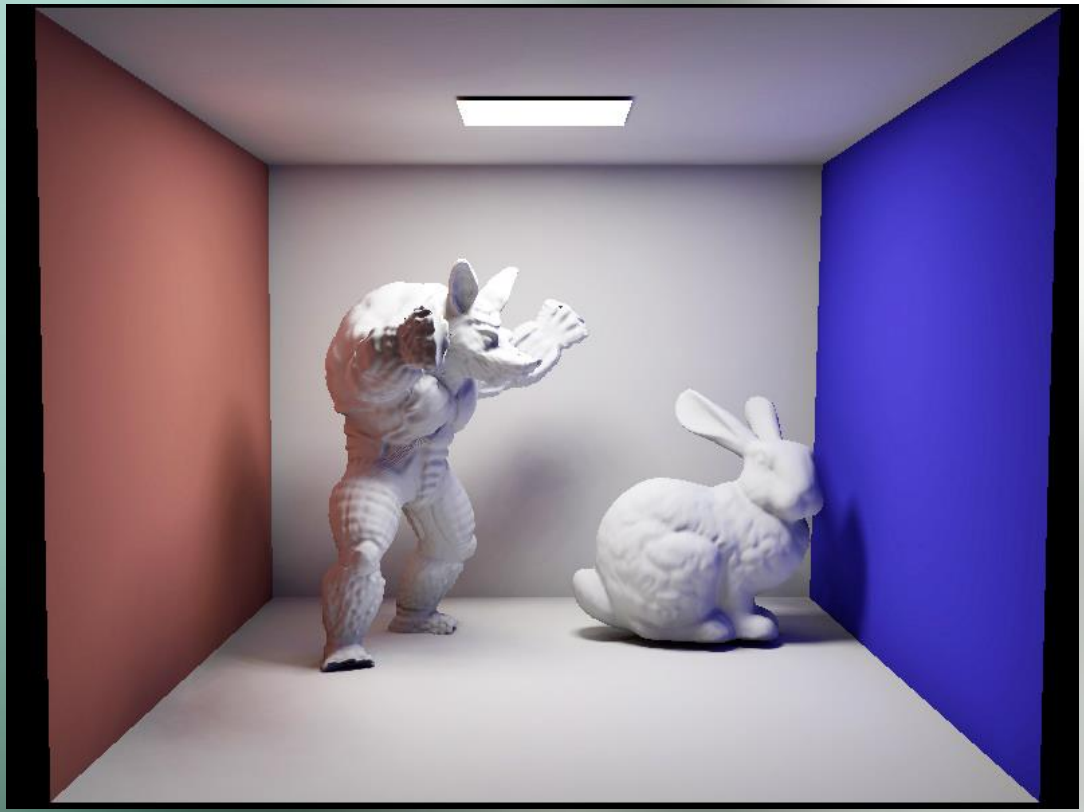


Our method – 155 VPLs – 0.5 seconds

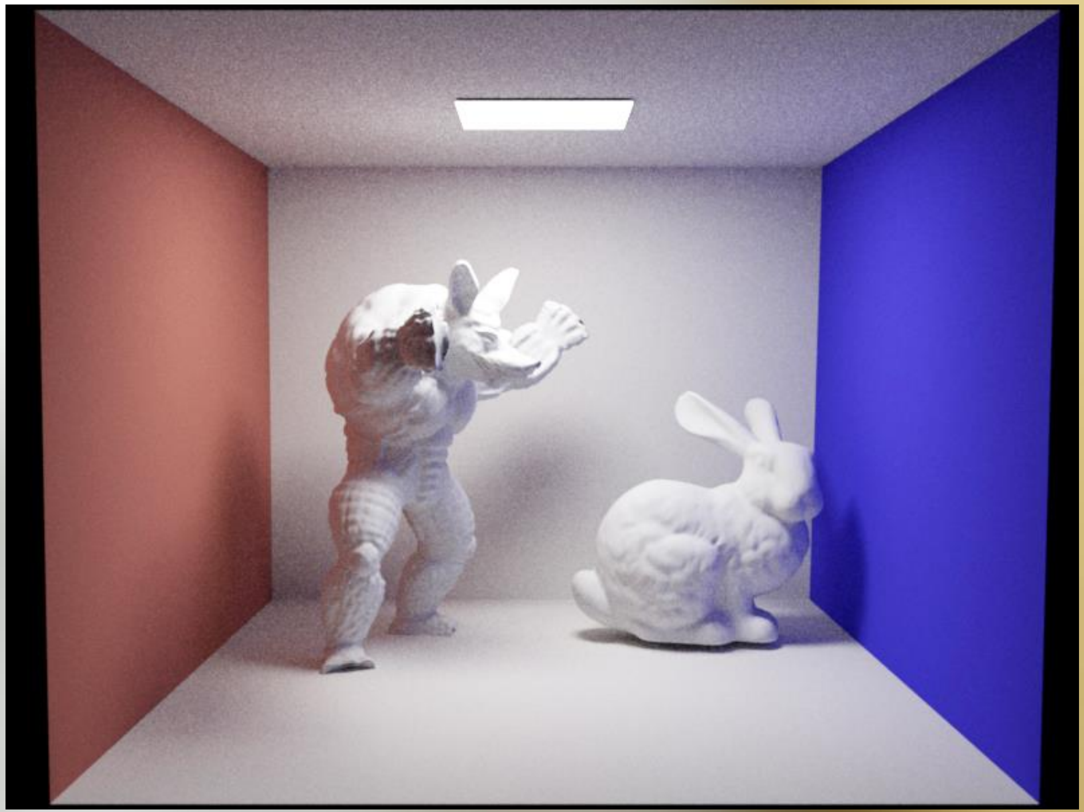


Path tracer – 128 samples – 410 seconds

Results - high resolution meshes (2)

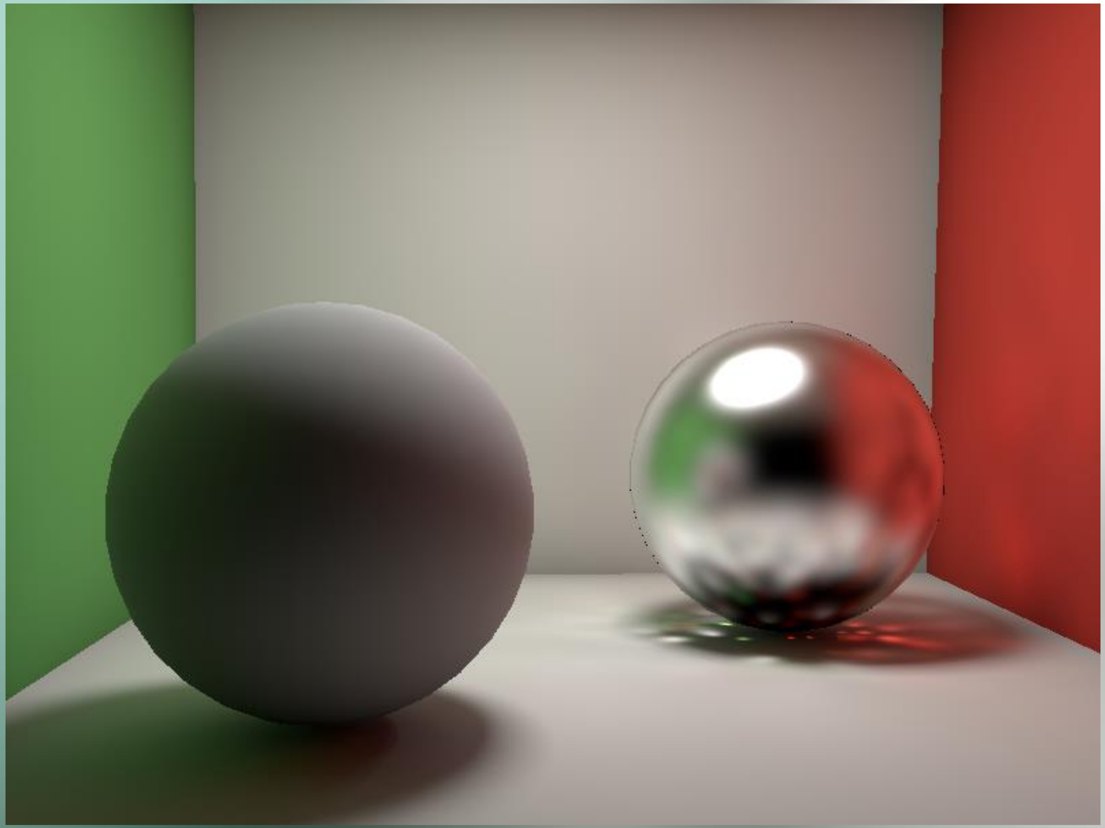


Our method – 880 VPLs – 10 seconds

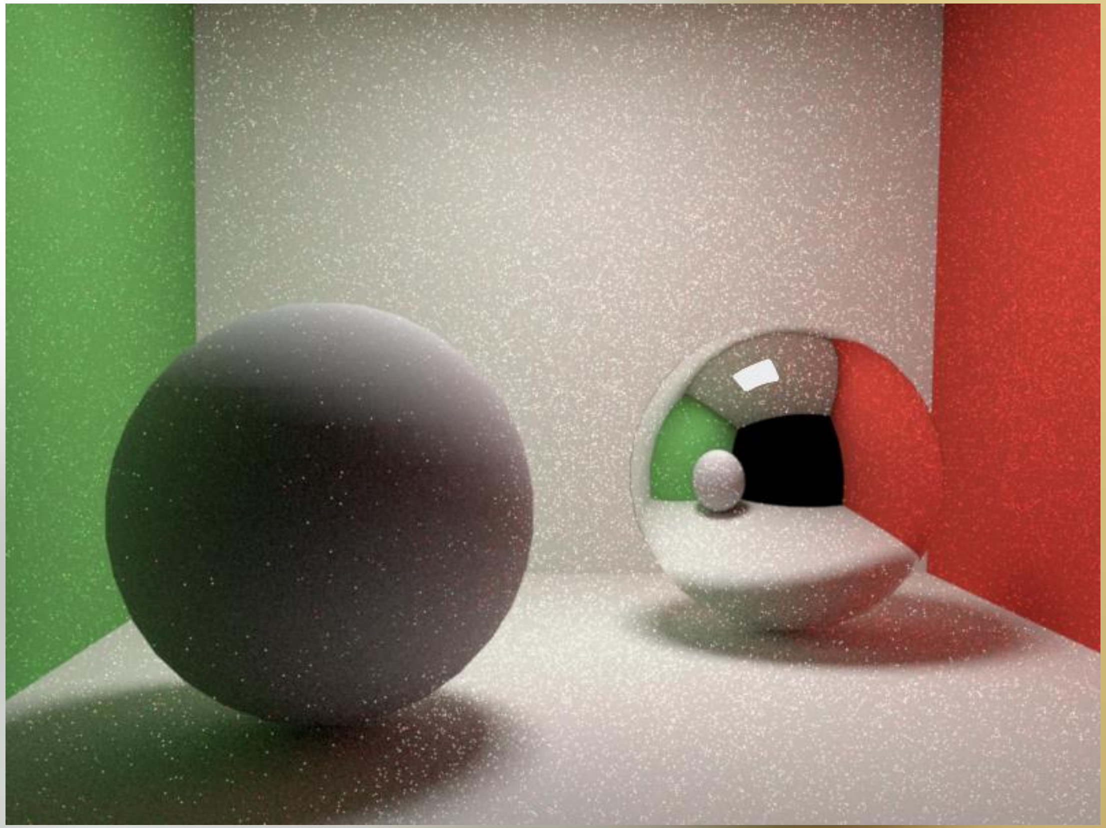


Path tracer – 128 samples – 410 seconds

Results - specular object



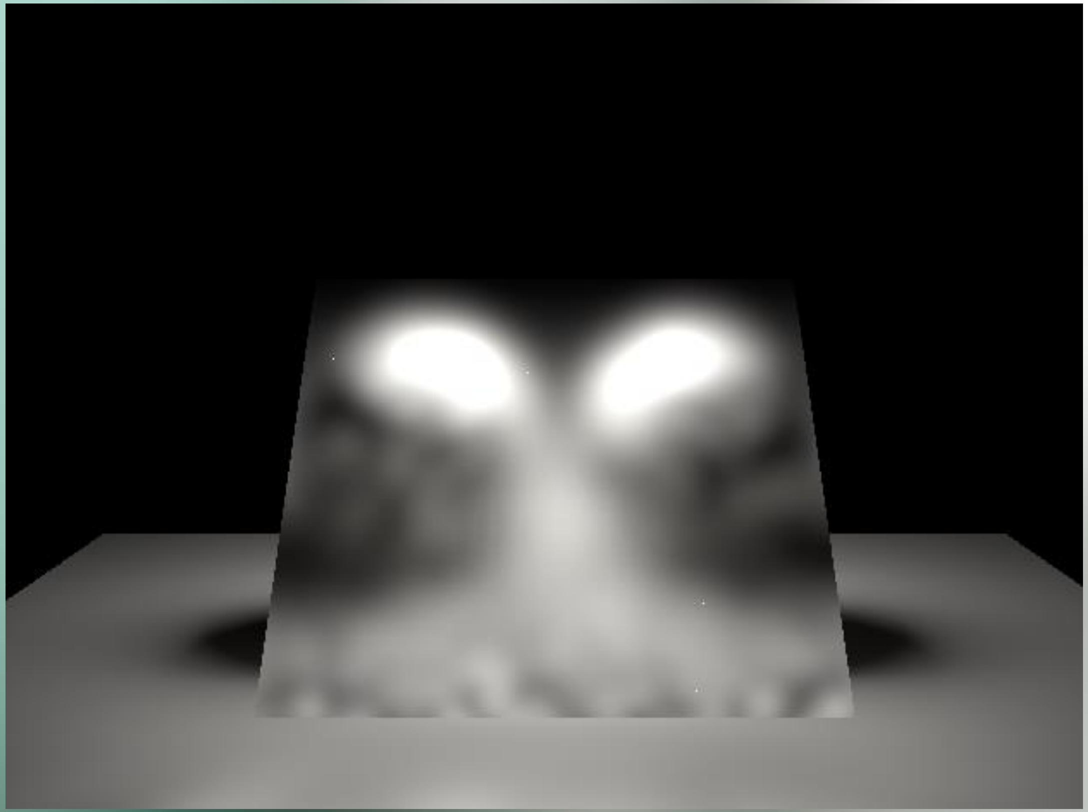
Our method – 20000 VPLs – 40 seconds



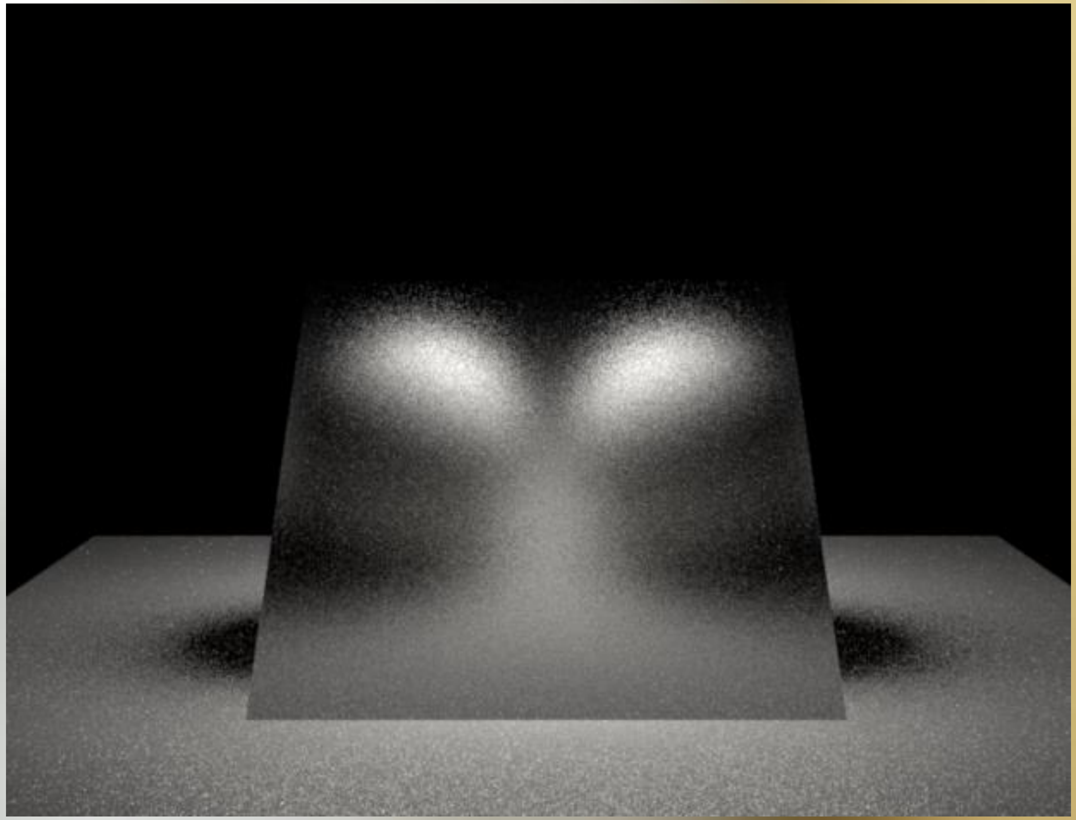
Path tracer – 64 samples – 200 seconds

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Results - rough dielectric object



Our method – 10000 VPLs – 15 seconds



Path tracer – 32 samples – 25 seconds

Future work

- More mutation types that handle different lighting problems
  - Caustics
  - Specular materials
  - Rough dielectric materials
- Vulkan implementation

Thank you