Light Transport Simulation:
Advanced Global Illumination

Ing. Michal Vlnas, ivlnas@fit.vutbr.cz

Department of Computer Graphics and Multimedia

FIT VUT

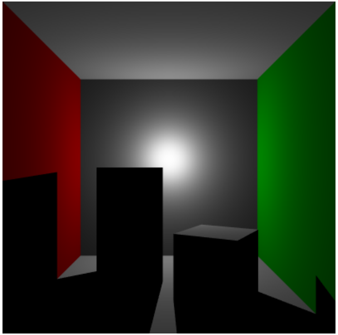
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

4.11.2024

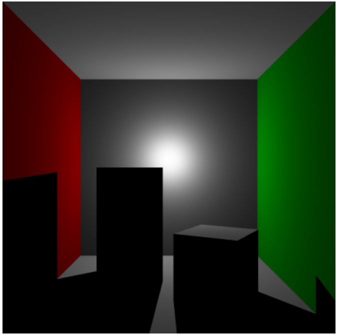
What is real and what is not real?


味玉
12m
味玉
●豚
らーめん
味玉
味玉
Paloma
6.0

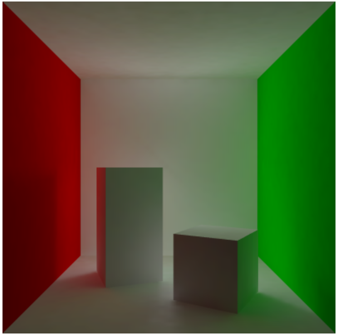

HOSHIMAKI
GIORNO
XTRA COFFEE
練乳入り
スイカ ソーダ
レヴ の お茶
デリース MILK TEA



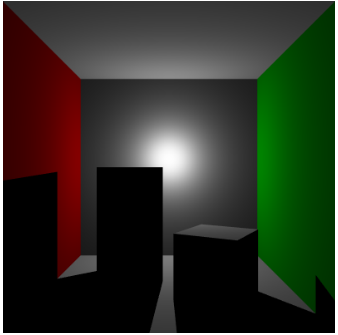
(a) Direct Illumination



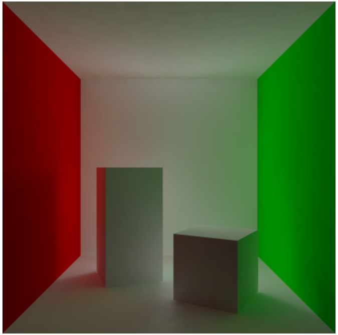
(a) Direct Illumination



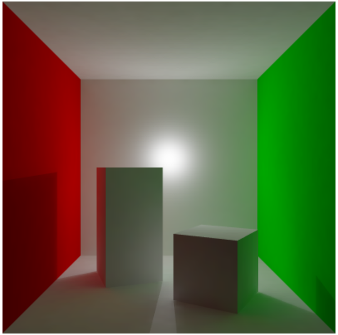
(b) Indirect Illumination



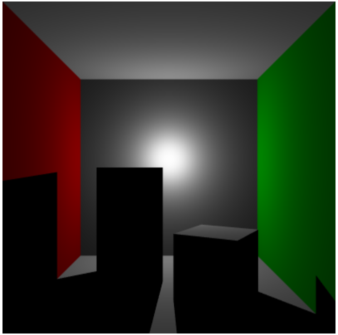
(a) Direct Illumination

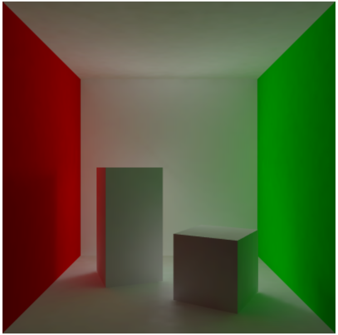


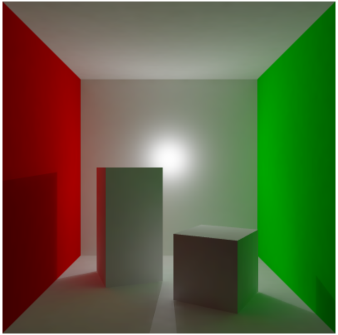
(b) Indirect Illumination



(c) Global Illumination


(a) Direct Illumination


(b) Indirect Illumination


(c) Global Illumination

Global Illumination + Physically Based Shading = (Photo)Realistic Rendering or Physically Based Rendering

Outline

1. Radiometry overview and BRDF
2. Rendering Equation to Light Transport Equation
3. Light Transport methods

1st part – Radiometry overview and BRDF

Radiometry overview I.

Radiometry is defined for all kinds of electromagnetic waves, however photometry utilizes only the visible light spectrum, according to the human perception.

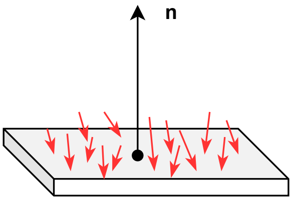
radiometry	eng. name	symbol	unit	photometry	eng. name	symbol	unit
zářivá energie	radiant energy	Q_e	J	světelné množství	luminous energy	Q	lm · s
zářivost	radiant intensity	I_e	$\mathrm{W \cdot sr^{-1}}$	svítivost	luminous intensity	I	cd (kandela)
zářivý tok	radiant flux	ϕ_e	W	světelný tok	luminous flux	ϕ	lm (lumen)
intenzita ozáření	irradiance	E_e	$\mathrm{W \cdot m^{-2}}$	osvětlení	illuminance	E	lx (lux)
zář	radiance	L_e	$\mathrm{W \cdot sr^{-1} \cdot m^{-2}}$	jas	luminance	L	$\mathrm{cd \cdot m^{-2}}$
expozice	radiant exposure	H_e	$\mathrm{J \cdot m^{-2} = W \cdot s \cdot m^{-2}}$	osvit	luminous exposure	H	lx · s
intenzita vyzařování	radiant exitance	M_e	$\mathrm{W \cdot m^{-2}}$	intenzita světlení	luminous exitance	M	$\mathrm{lm \cdot m^{-2}}$
radiozita	radiosity	J_e	$\mathrm{W \cdot m^{-2}}$	-	-	-	-

- **Note:** in GI, subscript „e“ is usually omitted, however it is the only sign that differs them from photometry.
- **Note:** in most cases, subscript „e“ denotes an emitted radiance (e. g. light sources, derived from radiant intensity).
- Each quantity has a spectral variant.
- Light Transport Simulation utilizes Radiometry.

Radiant flux (zářivý tok)

Radiant flux is an emitted, reflected, transmitted or received radiant energy Q_e per unit frequency t.

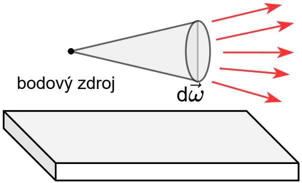
$$\Phi = \frac{\mathrm{d}Q_e}{\mathrm{d}t}\ [\mathrm{W} = \mathrm{J} \cdot \mathrm{s}^{-1}]$$



Radiant intensity (zářivost)

Radiant intensity is the emitted, reflected, transmitted or received radiant flux per solid angle.

$$I(\vec{\omega}) = \frac{\mathrm{d}\Phi(\vec{\omega})}{\mathrm{d}\vec{\omega}}\ [\mathrm{W} \cdot \mathrm{sr}^{-1}]$$

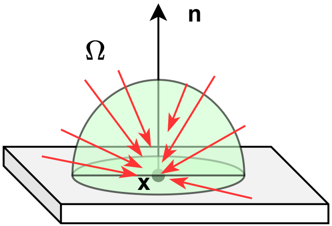


Radiometry overview III.

Irradiance (intenzita ozáření)

Irradiance is the radiant flux received by surface on unit area A. In practice, it is measured in a point $\mathbf{x}$ on the surface.

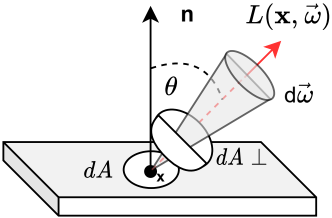
$$E(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A(\mathbf{x})} [\mathrm{W} \cdot \mathrm{m}^{-2}]$$



Radiance (zář)

Radiance is the emitted, reflected, transmitted or received radiant flux per solid angle $\vec{\omega}$ per unit area A.

$$L(\mathbf{x}, \vec{\omega}) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \vec{\omega})}{\mathrm{d}\vec{\omega}\, \mathrm{d}A(\mathbf{x})\, (\mathbf{n} \cdot \vec{\omega})} [\mathrm{W} \cdot \mathrm{sr}^{-1} \cdot \mathrm{m}^{-2}]$$

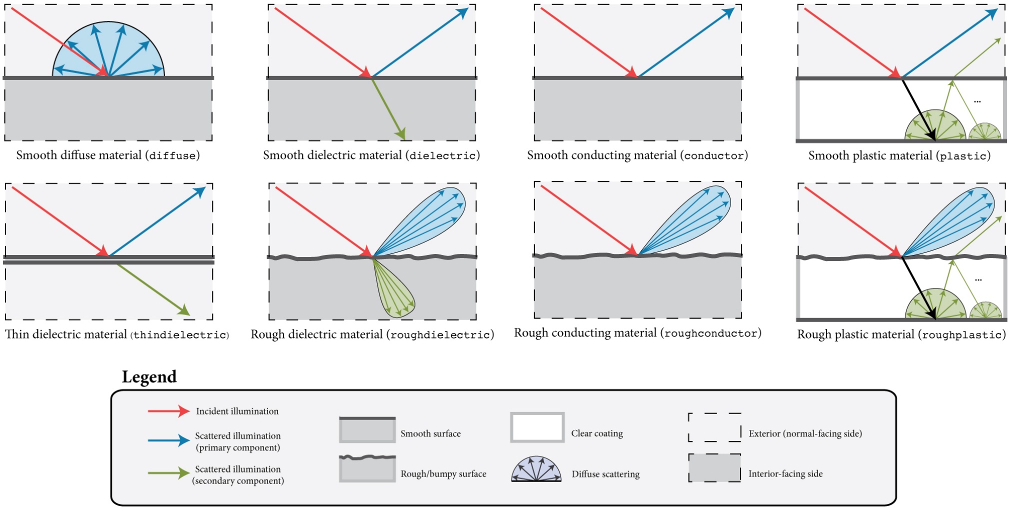


Bidirectional Reflectance Distribution Function: function takes an incoming light direction $\vec{\omega_i}$, and outgoing direction $\vec{\omega_o}$, and returns the ratio of radiance exiting along $\vec{\omega_o}$ to the incident radiance on the surface from direction $\vec{\omega_i}$.

$$f_r(\mathbf{x}, \vec{\omega_i} \to \vec{\omega_o}) = \frac{\mathrm{d}L_r(\mathbf{x}, \vec{\omega_o})}{\mathrm{d}E(\mathbf{x})} = \frac{\mathrm{d}L_r(\mathbf{x}, \vec{\omega_o})}{L_i(\mathbf{x}, \vec{\omega_i}) \cdot \cos(\theta_i)\, \mathrm{d}\vec{\omega_i}}$$

- Mathematical description of surface light interaction properties.
- Intuition: the probability density that photon incoming the surface from direction $\vec{\omega_i}$ will be reflected in direction $\vec{\omega_o}$.
- Also often referenced as BSDF.

BRDF examples



(a) Oren-Nayar BRDF

(b) Torrance-Sparrow BRDF

Figure: BRDF examples

2nd part – Rendering Equation to Light Transport Equation

Rendering equation (Kajiya 1986) in solid angle form:

$$L(\mathbf{x}, \vec{\omega_o}) = L_e(\mathbf{x}, \vec{\omega_o}) + \int_\Omega L(\mathbf{x}', -\vec{\omega_i}) f_r(\mathbf{x}, \vec{\omega_i} \to \vec{\omega_o}) \cdot \cos(\theta_i) \mathrm{d}\vec{\omega_i} \qquad (\mathrm{sr}^{-1}) \qquad (1)$$

On the contrary, rendering equation in three-point area form:

$$L(\mathbf{x}' \to \mathbf{x}'') = L_e(\mathbf{x}' \to \mathbf{x}'') + \int_M L(\mathbf{x} \to \mathbf{x}') \cdot f_r(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') \cdot G(\mathbf{x} \leftrightarrow \mathbf{x}')\, \mathrm{d}A(\mathbf{x}) \qquad (2)$$

Note: arrow convention follows light direction

Rendering equation (Kajiya 1986) in solid angle form:

$$L(\mathbf{x}, \vec{\omega_o}) = \underbrace{L_e(\mathbf{x}, \vec{\omega_o})}_{\text{emission}} + \int_{\Omega} L(\mathbf{x}', -\vec{\omega_i}) f_r(\mathbf{x}, \vec{\omega_i} \to \vec{\omega_o}) \cdot \cos(\theta_i) \mathrm{d}\vec{\omega_i} \tag{1}$$

On the contrary, rendering equation in three-point area form:

$$L(\mathbf{x}' \to \mathbf{x}'') = L_e(\mathbf{x}' \to \mathbf{x}'') + \int_{M} L(\mathbf{x} \to \mathbf{x}') \cdot f_r(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') \cdot G(\mathbf{x} \leftrightarrow \mathbf{x}') \, \mathrm{d}A(\mathbf{x}) \tag{2}$$

Note: arrow convention follows light direction

Rendering equation (Kajiya 1986) in solid angle form:

$$L(\mathbf{x},\vec{\omega_o}) = L_e(\mathbf{x},\vec{\omega_o}) + \int_{\Omega} \underbrace{L(\mathbf{x}', -\vec{\omega_i})}_{\text{incoming radiance}} f_r(\mathbf{x},\vec{\omega_i} \to \vec{\omega_o}) \cdot \mathbf{cos}(\theta_i)\mathbf{d}\vec{\omega_i} \quad (1)$$

On the contrary, rendering equation in three-point area form:

$$L(\mathbf{x}' \to \mathbf{x}'') = L_e(\mathbf{x}' \to \mathbf{x}'') + \int_M L(\mathbf{x} \to \mathbf{x}') \cdot f_r(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') \cdot G(\mathbf{x} \leftrightarrow \mathbf{x}')\,\mathrm{d}A(\mathbf{x}) \quad (2)$$

Note: arrow convention follows light direction

Rendering equation (Kajiya 1986) in solid angle form:

$$L(\mathbf{x}, \vec{\omega_o}) = L_e(\mathbf{x}, \vec{\omega_o}) + \int_\Omega L(\mathbf{x}', -\vec{\omega_i}) \underbrace{\boxed{f_r(\mathbf{x}, \vec{\omega_i} \rightarrow \vec{\omega_o})}}_{\text{BRDF coefficient}} \cdot \cos(\theta_i) \mathrm{d}\vec{\omega_i} \quad (1)$$

On the contrary, rendering equation in three-point area form:

$$L(\mathbf{x}' \rightarrow \mathbf{x}'') = L_e(\mathbf{x}' \rightarrow \mathbf{x}'') + \int_M L(\mathbf{x} \rightarrow \mathbf{x}') \cdot f_r(\mathbf{x} \rightarrow \mathbf{x}' \rightarrow \mathbf{x}'') \cdot G(\mathbf{x} \leftrightarrow \mathbf{x}') \, \mathrm{d}A(\mathbf{x}) \quad (2)$$

Note: arrow convention follows light direction

Rendering equation (Kajiya 1986) in solid angle form:

$$L(\mathbf{x},\vec{\omega_o}) = L_e(\mathbf{x},\vec{\omega_o}) + \int_\Omega L(\mathbf{x}',-\vec{\omega_i}) f_r(\mathbf{x},\vec{\omega_i}\to\vec{\omega_o}) \cdot \underbrace{\boxed{\cos(\theta_i)}}_{\text{weakening factor}} \, \mathrm{d}\vec{\omega_i} \tag{1}$$

On the contrary, rendering equation in three-point area form:

$$L(\mathbf{x}'\to\mathbf{x}'') = L_e(\mathbf{x}'\to\mathbf{x}'') + \int_M L(\mathbf{x}\to\mathbf{x}') \cdot f_r(\mathbf{x}\to\mathbf{x}'\to\mathbf{x}'') \cdot G(\mathbf{x}\leftrightarrow\mathbf{x}') \, \mathrm{d}A(\mathbf{x}) \tag{2}$$

Note: arrow convention follows light direction

Rendering equation (Kajiya 1986) in solid angle form:

$$L(\mathbf{x}, \vec{\omega_o}) = L_e(\mathbf{x}, \vec{\omega_o}) + \int_{\Omega} L(\mathbf{x}', -\vec{\omega_i}) f_r(\mathbf{x}, \vec{\omega_i} \to \vec{\omega_o}) \cdot \cos(\theta_i) \mathrm{d}\vec{\omega_i} \tag{1}$$

On the contrary, rendering equation in three-point area form:

$$L(\mathbf{x}' \to \mathbf{x}'') = L_e(\mathbf{x}' \to \mathbf{x}'') + \int_M L(\mathbf{x} \to \mathbf{x}') \cdot f_r(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') \cdot G(\mathbf{x} \leftrightarrow \mathbf{x}') \, \mathrm{d}A(\mathbf{x}) \tag{2}$$

Note: arrow convention follows light direction

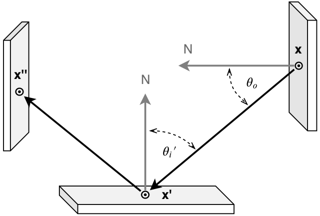


Figure: Three point rendering equation geometry

It remains to define so-called geometric term:

$$G(\mathbf{x} \leftrightarrow \mathbf{x}') = V(\mathbf{x} \leftrightarrow \mathbf{x}') \cdot \frac{|\cos(\theta_o) \cdot \cos(\theta_i')|}{\|\mathbf{x} - \mathbf{x}'\|^2} \tag{3}$$

But recursion is bad. . .

$$
\begin{aligned}
L = L_e(x) + \int_M L \,\mathrm{d}A &= \\
= L_e(x) + \int_M \left(L_e(x) + \int_M L \,\mathrm{d}A \right) \mathrm{d}A &= \\
= L_e(x) + \int_M \left(L_e(x) + \int_M \left(L_e(x) + \int_M L \,\mathrm{d}A \right) \mathrm{d}A \right) \mathrm{d}A &= \dots
\end{aligned} \tag{4}
$$

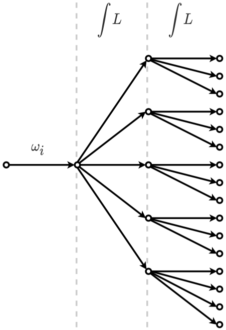
Intuition: We would like to have a single equation which tells us the intensity I of pixel j ...

But recursion is bad...

$$
\begin{aligned}
L &= L_e(x) + \int_M L \,\mathrm{d}A = \\
&= L_e(x) + \int_M \left(L_e(x) + \int_M L \,\mathrm{d}A \right) \mathrm{d}A = \\
&= L_e(x) + \int_M \left(L_e(x) + \int_M \left(L_e(x) + \int_M L \,\mathrm{d}A \right) \mathrm{d}A \right) \mathrm{d}A = \dots
\end{aligned} \tag{4}
$$

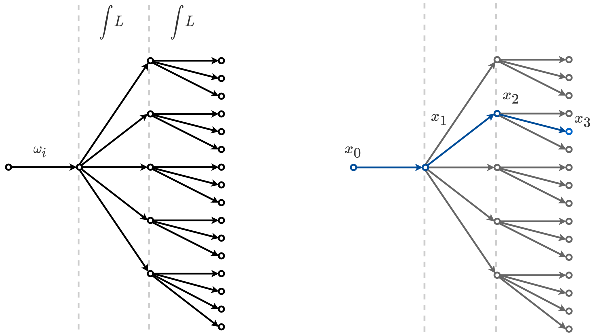
Intuition: We would like to have a single equation which tells us the intensity I of pixel j ...
... therefore, we introduce intensity measurement equation:

$$
I_j = \int_{M \times M} W_e^j(\mathbf{x} \to \mathbf{x}')L(\mathbf{x} \to \mathbf{x}')G(\mathbf{x} \leftrightarrow \mathbf{x}')\mathrm{d}A(\mathbf{x})\mathrm{d}A(\mathbf{x}') \tag{5}
$$

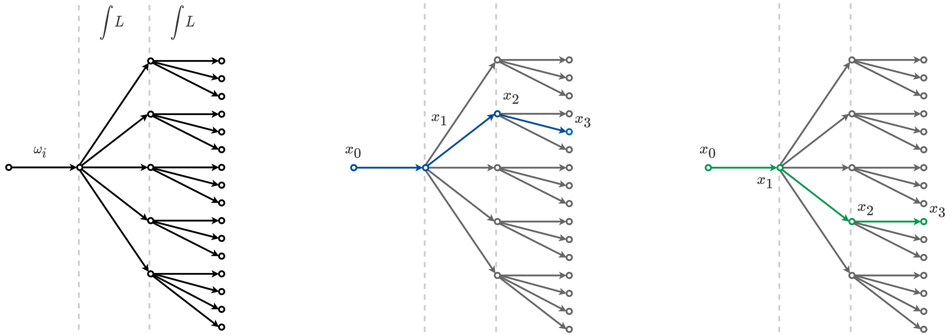


(a) Recursive trajectory principle

Light transport equation IV. – path integral



(a) Recursive trajectory principle (b) Path integral



(a) Recursive trajectory principle (b) Path integral (c) Path integral

Path integral: instead of using a **recursive trajectory principle** (like in rendering equation), path integral uses an **integral over all trajectories**. (Veach 1998)

$$I_j = \underbrace{\int_\Omega}_{\text{all paths}} \underbrace{f_j(\bar{x})}_{\text{throughput function}} \,\mathrm{d}\omega(\mathbf{x}) \tag{6}$$

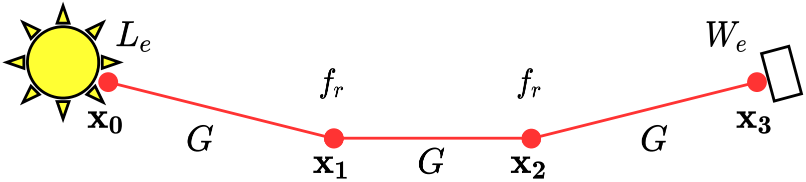
Imagine integrating the whole recursion tree using rendering equation, the result will be the **same** as integrating over all paths in the tree, if you use correct throughput function.

Using recursive expansion of integral from equation 2 into k dimension wide integral, with omitting the emissive part, and inserting into equation 5, together with applying path integral theorem, the result is following: (Veach 1998)

$$I_j = \sum_{k=1}^{\infty} \int_{M^{k+1}} L_e(\mathbf{x}_0 \to \mathbf{x}_1) \cdot G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \cdot W_e^j(\mathbf{x}_{k-1} \to \mathbf{x}_k)$$
$$\left(\prod_{i=1}^{k-1} f_r(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}) \cdot G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \right) \mathrm{d}A(\mathbf{x}_0) \ldots \mathrm{d}A(\mathbf{x}_k) \quad (7)$$

$$f_j(\bar{x}) = L_e(\mathbf{x}_0 \to \mathbf{x}_1) \cdot G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \cdot f_r(\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2) \cdot G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \cdot f_r(\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3) \cdot G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3) \cdot W_e^j(\mathbf{x}_2 \to \mathbf{x}_3) \tag{8}$$

An example of integrand, constructed with path $\bar{x} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$



Monte Carlo integration

A function $f : \Omega \to \mathbb{R}$ can be estimated using Monte Carlo method with N random samples as:

$$I = \int_\Omega f(x)\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}, \tag{9}$$

where x_i is a random variable with probability density function $p(x_i)$.

3rd part – Light Transport methods

unidirectional	*bidirectional*	*hybrids*
Particle Tracing	**Bidirectional Path Tracing**	Metropolis Light Transport
Path Tracing	Photon Mapping and variants	Energy Redistribution Path Tracing
	Vertex Connections and Merging	
	Virtual Point Lights	

Most of the methods are based on ray tracing principle

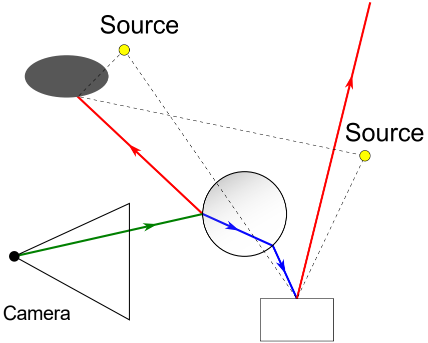
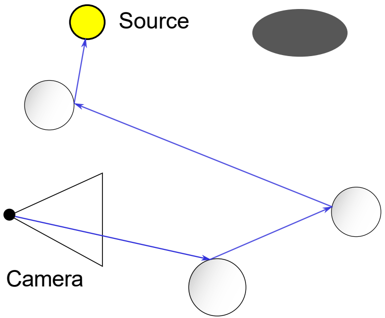
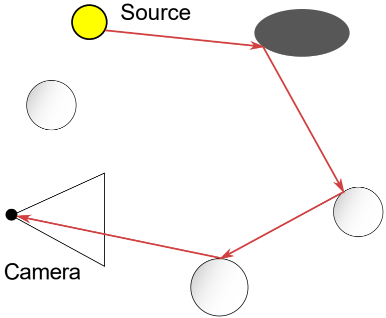


Figure: Ray tracing

... which is **NOT** global illumination method

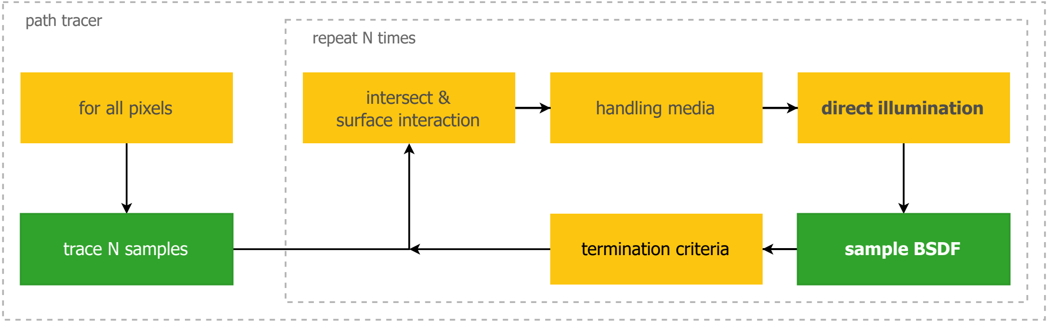


(a) **path tracing** – backward tracing from camera to the light source



(b) **light tracing** – forward tracing from light source to the camera

Figure: Forward/backward tracing comparison

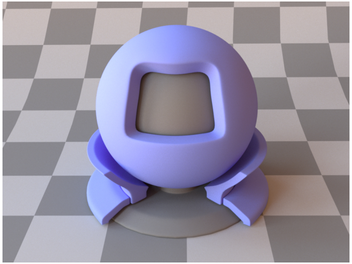


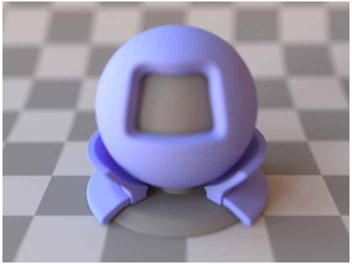
- Quasi Path Tracing also present in modern games: Cyberpunk, Witcher 3, ...
- NVIDIA SDK for real-time PT: `https://developer.nvidia.com/rtx/path-tracing`

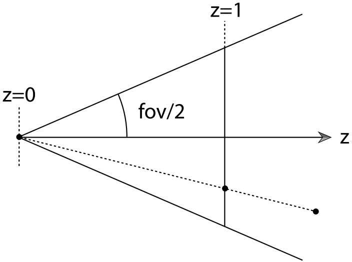
Generalized algorithm:

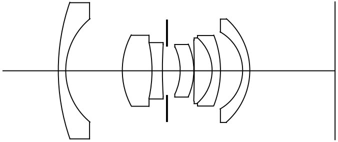
1. Sample sensor (film)
2. Sample lens
3. Cast a ray

- Pinhole camera has infinitely small aperture, infinite depth of field.
- Realistic lens – tracing ray through optical system.

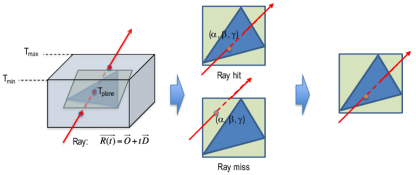








1. Intersect with scene (traverse BVH, bounding box, kD, etc.)



(a) Stage 1: Ray-plane intersection test (b)) Stage 2: Barycentric test (c)) Stage 3: Exact hit point calculation

2. If no hit, try to shade using surrounding lights
3. Check for direct light source hit
4. Prepare BSDF

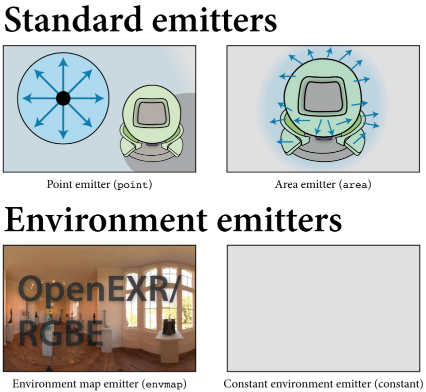
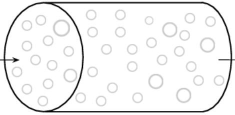


Image taken from Mitsuba 3 doc.

- Except of surface interaction, there can be medium interaction
- Participating medium is a black box for solving volume mediums or subsurface scattering (fog, human skin, wax, liquids, ...)



1. Scatter inside medium.
2. Solve Radiative Transfer Equation
3. Return to path tracer

- RTE can be solved with Ray Marching inside medium

$$(\vec{\omega}\cdot\nabla)L(\mathbf{x}\to\vec{\omega}) = \underbrace{-\underbrace{\sigma_a(\mathbf{x})L(\mathbf{x}\to\vec{\omega})}_{absorption} - \underbrace{\sigma_s(\mathbf{x})L(\mathbf{x}\to\vec{\omega})}_{out-scattering}}_{extinction} + \underbrace{\sigma_a(\mathbf{x})L_e(\mathbf{x}\to\vec{\omega})}_{emission} + \underbrace{\sigma_s(\mathbf{x})L_i(\mathbf{x}\to\vec{\omega})}_{in-scattering}$$

- Shadow ray from each path vertex towards a random light source.
- Only for **NON**-pure specular surfaces.
- How to handle the ray orientation?
- How to evaluate weight w?

Algorithm:

1. Pick random light source with selection PDF p_s.
2. Sample emitter with PDF p_{em}
3. Check visibility
4. Eval contribution: $th \cdot f_s \cdot L_e \cdot w(p_s, p_{em}, p_{fs})$

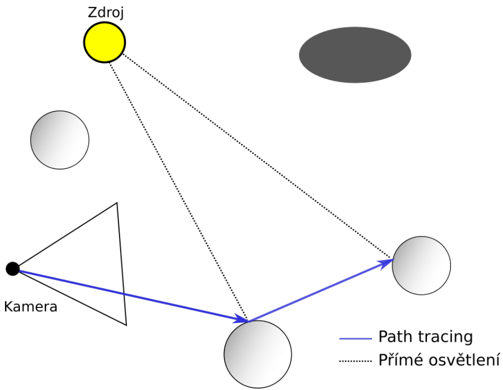
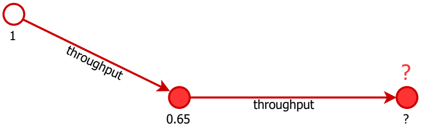


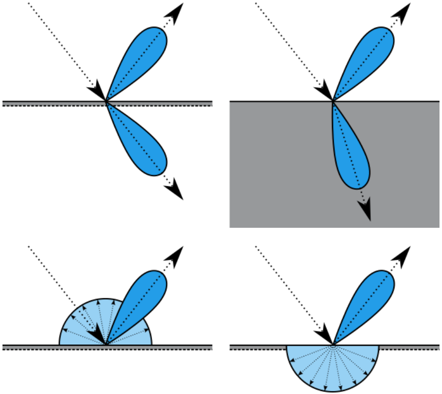
Figure: Path tracing with direct illumination

... also called Next Event Estimation (NEE)

1. Pick any BSDF component to sample.
2. Compute BSDF value f_s
3. Sample new outgoing direction $\vec{\omega_o}$ with PDF p_{fs}.
4. Update throughput: $th = th \cdot f_s$.

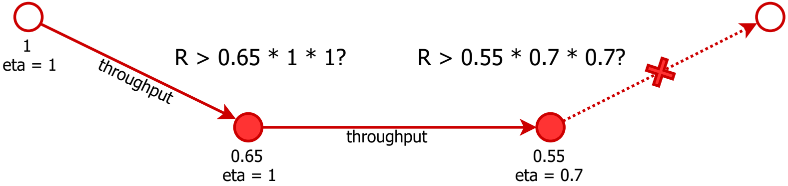
- Possible interactions:
  - reflect (mirror, glossy, dielectrics, plastics)
  - refract (dielectrics, plastics)
  - scatter (matte, plastics)
- Materials can be smooth or rough (microfacet models)
- **Importance sample direction with respect to BSDF shape.**

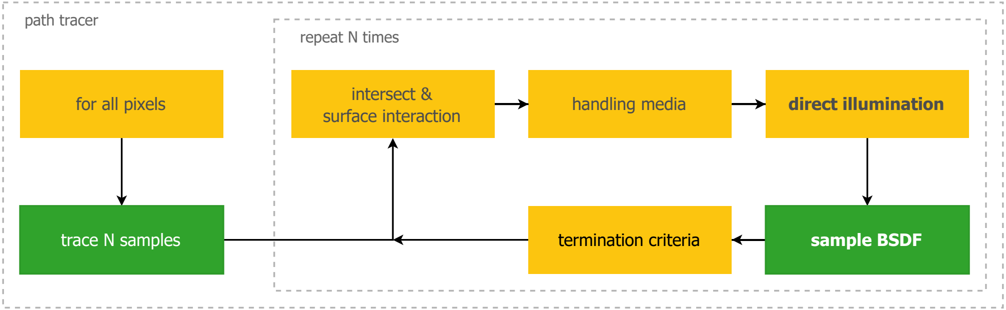




```
// Russian roulette
if (rec.depth++ >= rrDepth) {
    Float q = std::min(throughput.max() * eta * eta, 0.95f);
    if (rec.nextSample1D() > q)
        break;
    throughput /= q;
}
```

- How to prevent too long paths? E.g due to total internal reflection.
- Fixed path length
- Russian roulette
- The lower throughput, the higher termination chance





- Similar approach can be applied to particle/light tracing.
- We can achieve faster convergence with De-noisers, Path Guiding, ...

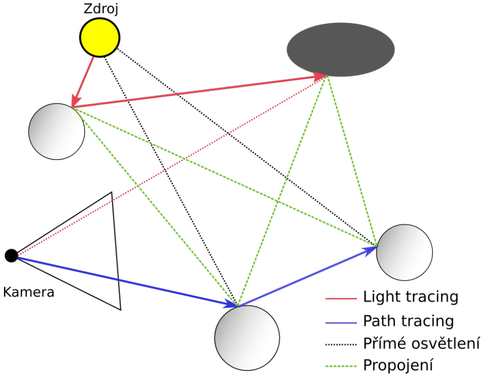
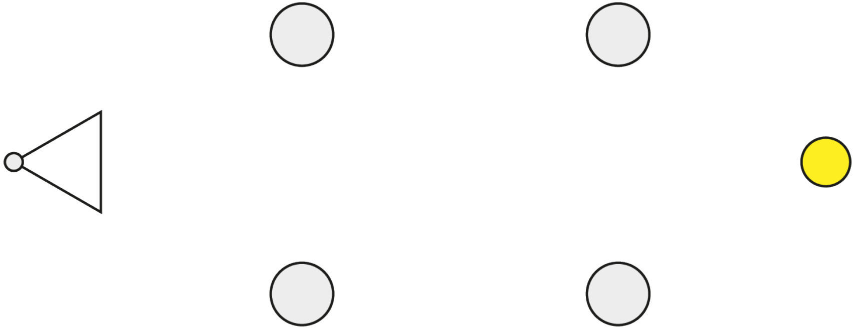


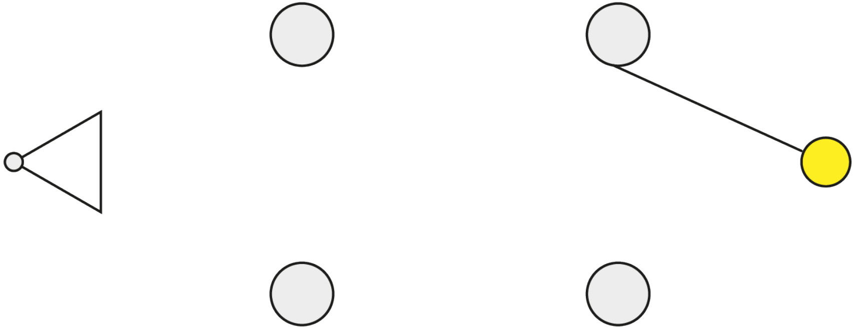
Figure: **Bidirectional Path Tracing**

Bidirectional path tracing (Veach 1998) is considered to be one of the most powerful "basic" algorithms.

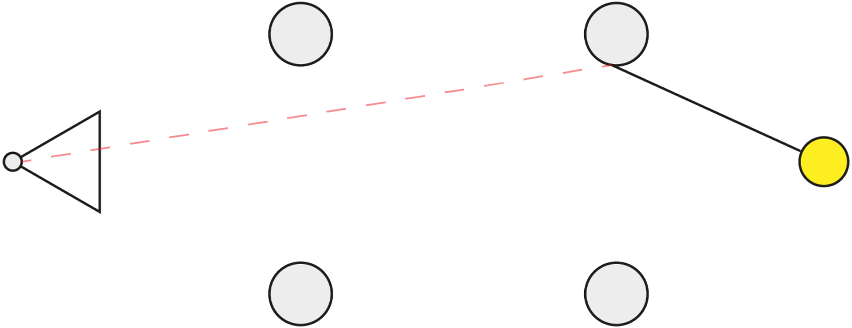
Bidirectional path tracing



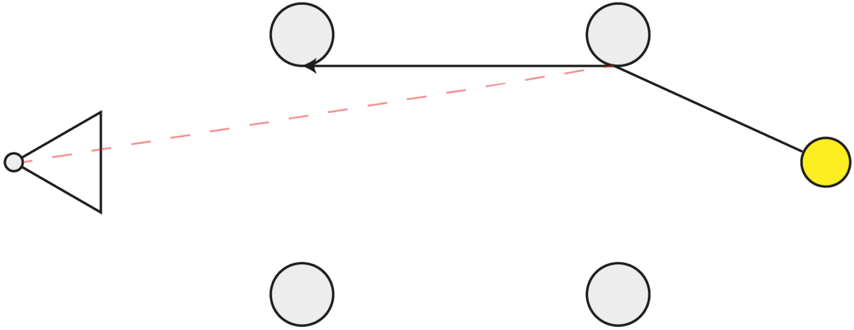
Bidirectional path tracing



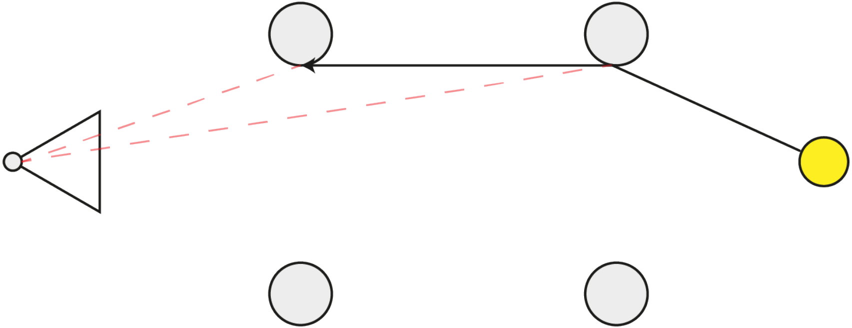
Bidirectional path tracing

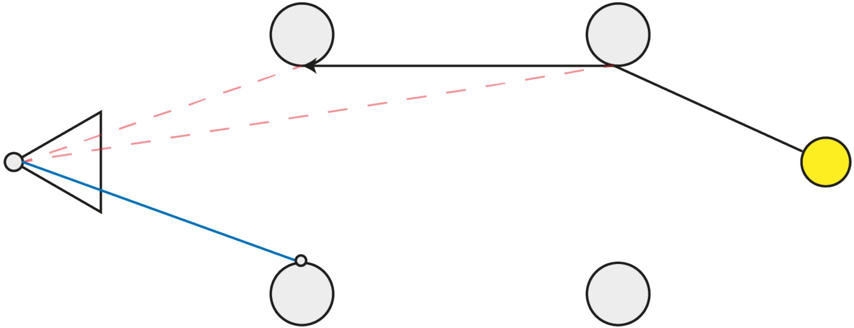


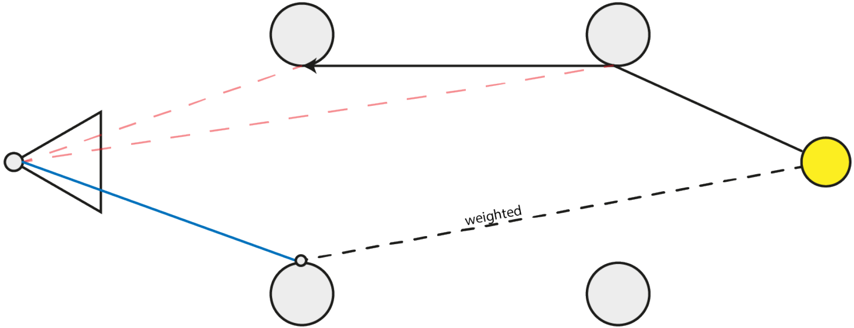
Bidirectional path tracing



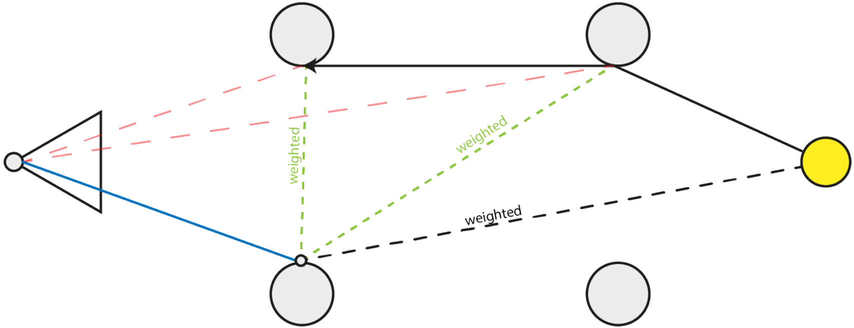
Bidirectional path tracing

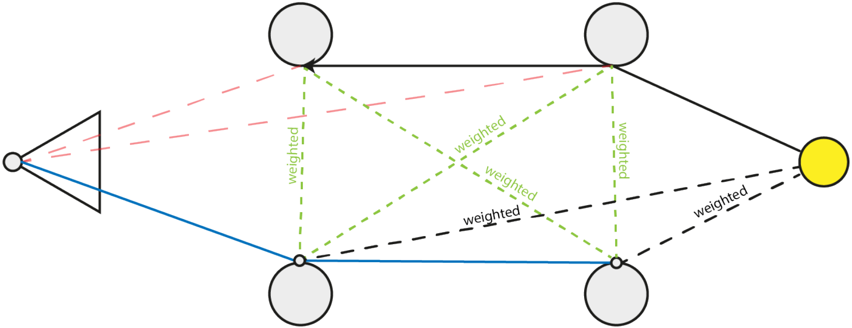


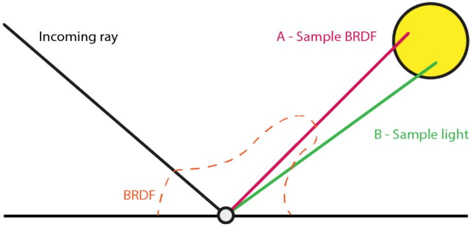



weighted

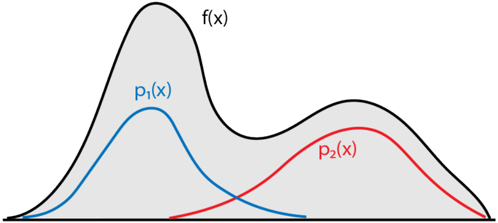
Bidirectional path tracing




weighted
weighted
weighted
weighted
weighted
weighted



(a) Two possible sampling techniques



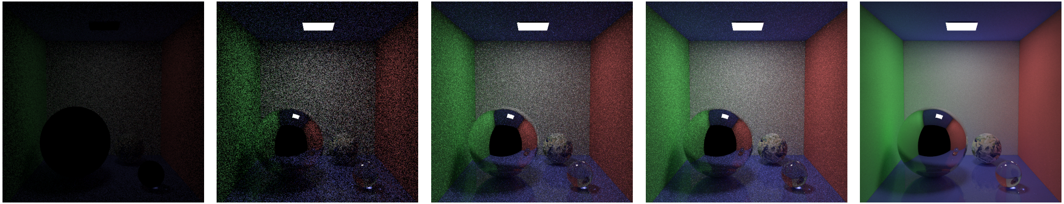
(b) General MIS case

MIS is a powerful noise reduction tool:

$$w_s(\mathbf{x}) = \frac{n_s \cdot p_s(\mathbf{x})}{\sum_i n_i \cdot p_i(\mathbf{x})} \tag{10}$$

This strategy is called balance heuristic.

PT vs BDPT



(a) LT (b) PT (c) PT + DI (d) BDPT (e) BDPT + MIS

Figure: PT vs BDPT, all images uses x60 supersampling

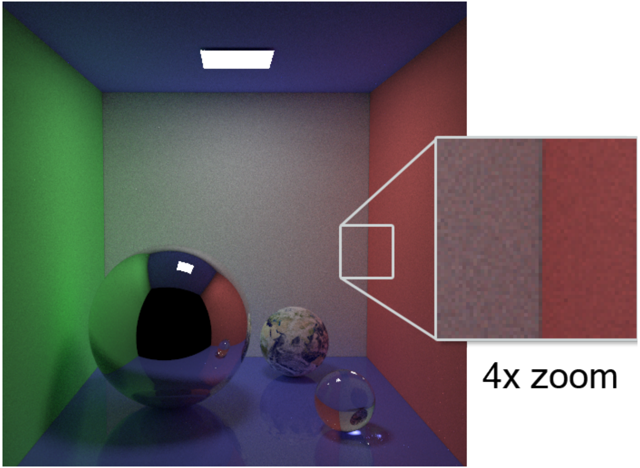


Figure: BDPT + MIS

PT vs BDPT

BDPT is more powerfull than PT but ...

- Quite complicated to implement.
- PT is more simple for mass parallel acceleration on GPU and more friendly.
- Modern rendering systems are based on PT + NEE (Next-Event-Estimation) + denoise + path guiding + post-processing.
- Multiple Importance Sampling (MIS) is applicable also on PT.

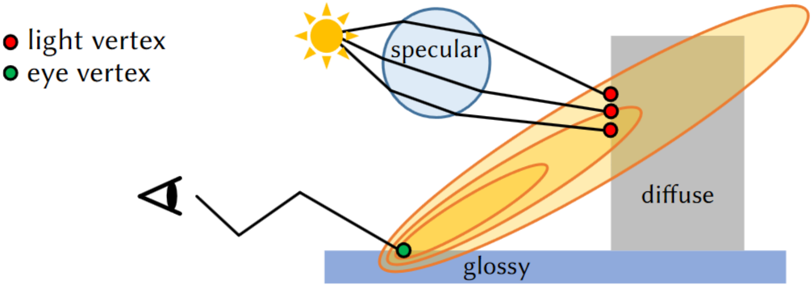


Figure: Unconnected GDS path

Unbiased algorithms like PT or BDPT can't efficiently handle SDS, GDS, SGD, GDG paths (S – Specular, G – Glossy, D – Diffuse)

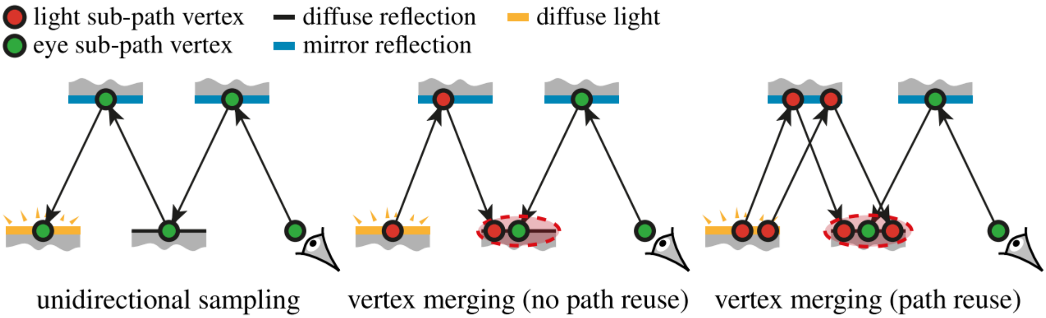
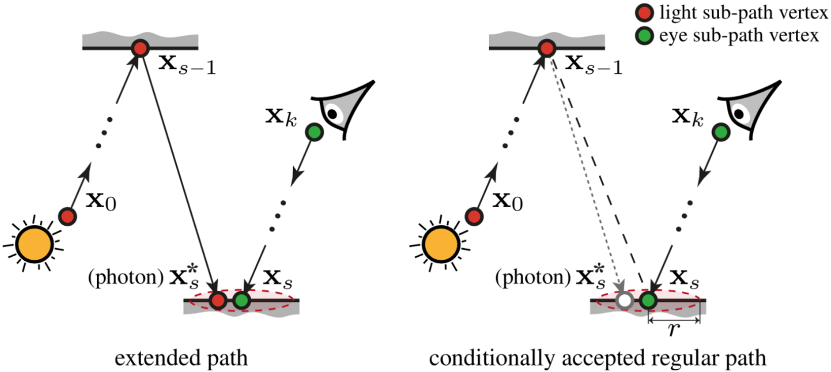


Figure: **Left:** unidirectional sampling. **Middle:** vertex merging. **Right:** light path re-using.

Combination of BDPT and Photon Mapping (Georgiev et al. 2012), via multiple importance sampling in path space domain.

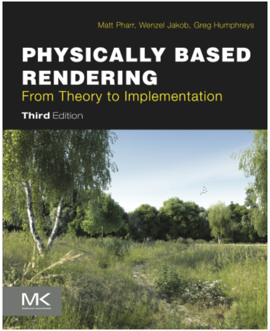


- Requires spatial structure with range-search support.
- Original implementation uses hash grid.

Open source and scientific implementations

- Physically Based Rendering (Pharr, Jakob, and Humphreys 2016)
- Mitsuba 3 (Jakob et al. 2022)

Most of the scientific papers are implemented in one of these.



Summary I.

What we discussed today:

- Basics to Radiometry and BRDF/BSDF
- Rendering equation and Light Transport Equation
- Stochastic solution to RE/LTE
- Path Tracing
- Bidirectional Path Tracing and Multiple Importance Sampling
- Vertex Connections and Merging

Summary II.

Where else to look ...

- Photon mapping and its variants (PM, PPM, SPPM)
- Metropolis light transport (MLT)
- Energy redistribution path tracing (PT + MLT)
- ...
- Acceleration methods (Path guiding, Hierarchical Russian roulette, Quasi-Monte Carlo)
- Differentiable rendering
- Spectral rendering
- Signed Distance Fields in LTS

Sources

- Georgiev, Iliyan et al. (Nov. 2012). "Light Transport Simulation with Vertex Connection and Merging". In: ACM Trans. Graph. 31.6, 192:1–192:10. ISSN: 0730-0301.
- Jakob, Wenzel et al. (2022). Mitsuba 3 renderer. Version 3.0.1. https://mitsuba-renderer.org.
- Kajiya, James T. (Aug. 1986). "The Rendering Equation". In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques. Vol. 20. SIGGRAPH '86 4. New York, NY, USA: ACM, pp. 143–150. ISBN: 0-89791-196-2.
- Pharr, Matt, Wenzel Jakob, and Greg Humphreys (2016). Physically Based Rendering: From Theory to Implementation. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 0128006455.
- Veach, Eric (1998). "Robust Monte Carlo Methods for Light Transport Simulation". PhD thesis. Stanford, CA, USA. ISBN: 0591907801.

Thank you for attention