Global Illumination

Light Paths

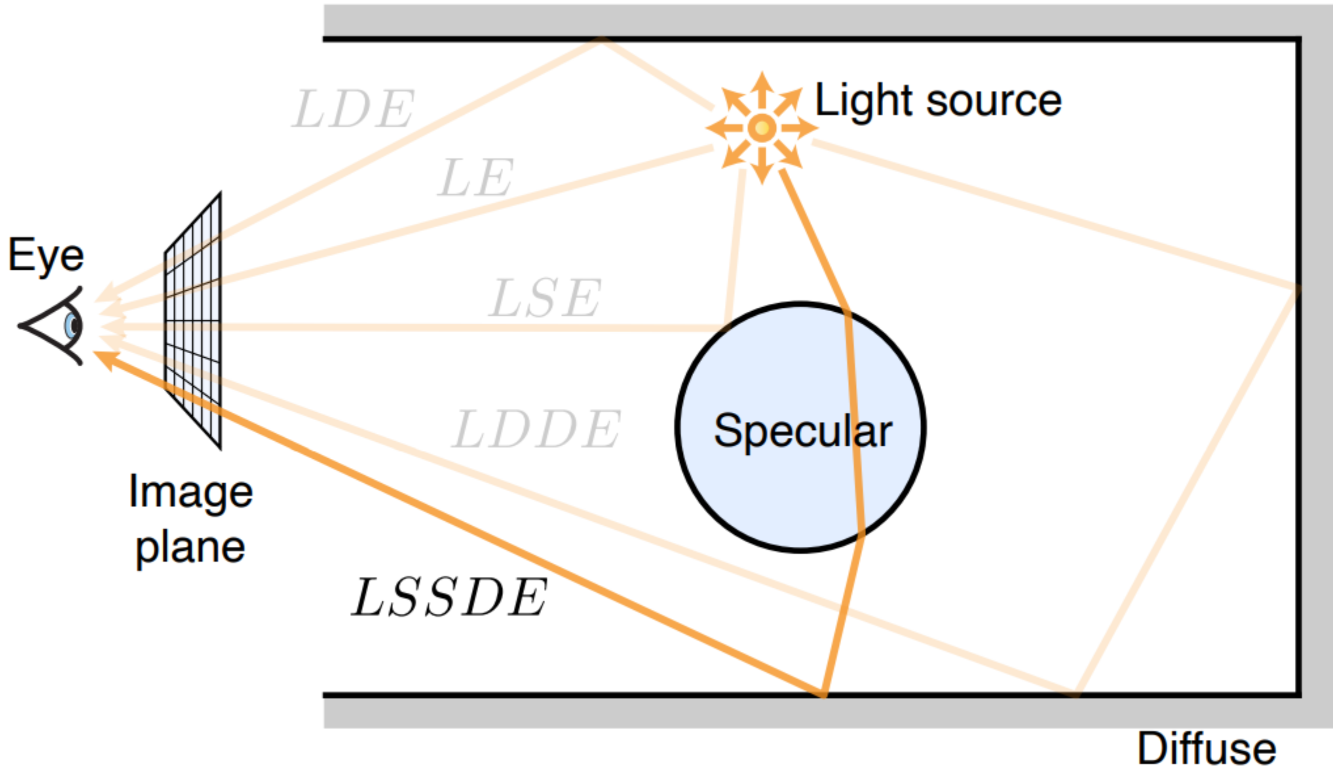
A light path is a chain of linear segments joining at event "vertices". An event is a surface interaction(reflection or refraction) or source/eye.

Heckbert's Classification

Heckbert's classification distinguishes the vertices

- L: a light source
- E: the eye/camera
- S: a specular reflection
- D: a diffuse reflection

Example:



with the following expression type syntax

- $K+$: one or more of event K
- $K*$: zero or more of event K
- $K?$: zero or one K events
- $(K|H)$: a K or an H event
  we could have more detailed or powerful description.
- Direction illumination: $L(D|S)E$

- Indirect illumination: $L(D|S)(D|S)+E$
- Classical (Whitted-style) ray tracing: $LDS*E$
- Full global illumination: $L(D|S)*E$
  - diffuse inter-reflections: $LDD+E$
  - caustics: $LS+DE$

Recursive Rendering Equation

At this point, we don't consider the participating media, thus the radiance is constant along the ray. To construct a recursive relation between rays, we need a ray tracing function:

$$\mathbf{x}'=r(\mathbf{x},\vec{\omega})$$

which gives the point of surface interaction of a ray defined by $\mathbf{x}$ and $\vec{\omega}$. Besides we know that the incident ray at one point is the outgoing ray of the previous interaction point. Thus

$$L_i(\mathbf{x},\vec{\omega})=L_o(r(\mathbf{x},\vec{\omega}),-\vec{\omega})$$

The rendering equation thus becomes recursive

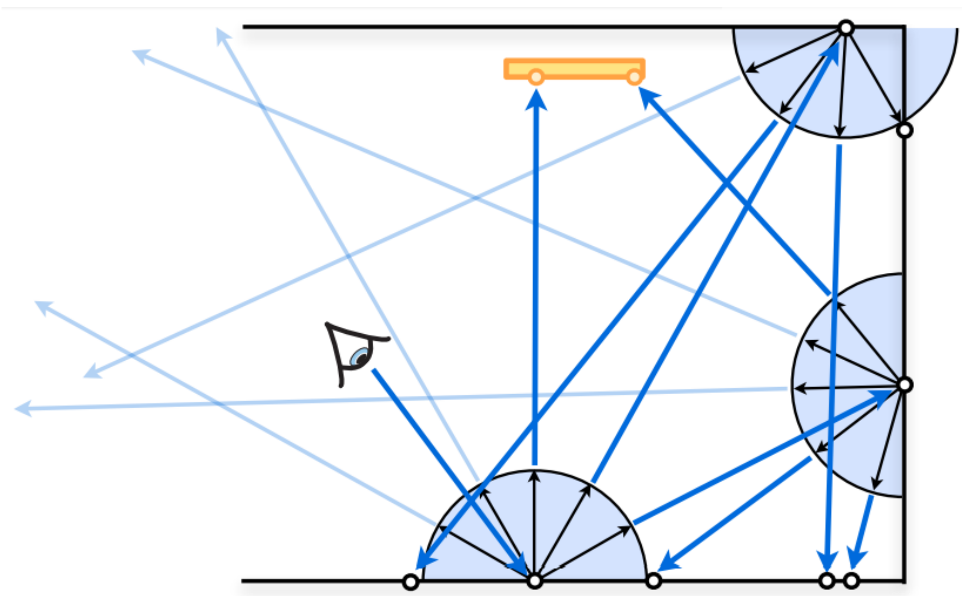
$$L_o(\mathbf{x},\vec{\omega})=L_e(\mathbf{x},\vec{\omega})+\int_{H^2} f_r\left(\mathbf{x},\vec{\omega}',\vec{\omega}\right)L_o\left(r\left(\mathbf{x},\vec{\omega}'\right),-\vec{\omega}'\right)\cos\theta' d\vec{\omega}'$$

since there are only outgoing functions on both sides, we could drop the "o" subscript.

$$L(\mathbf{x},\vec{\omega})=L_e(\mathbf{x},\vec{\omega})+\int_{H^2} f_r\left(\mathbf{x},\vec{\omega}',\vec{\omega}\right)L\left(r\left(\mathbf{x},\vec{\omega}'\right),-\vec{\omega}'\right)\cos\theta' d\vec{\omega}'$$

Solving the Rendering Equation

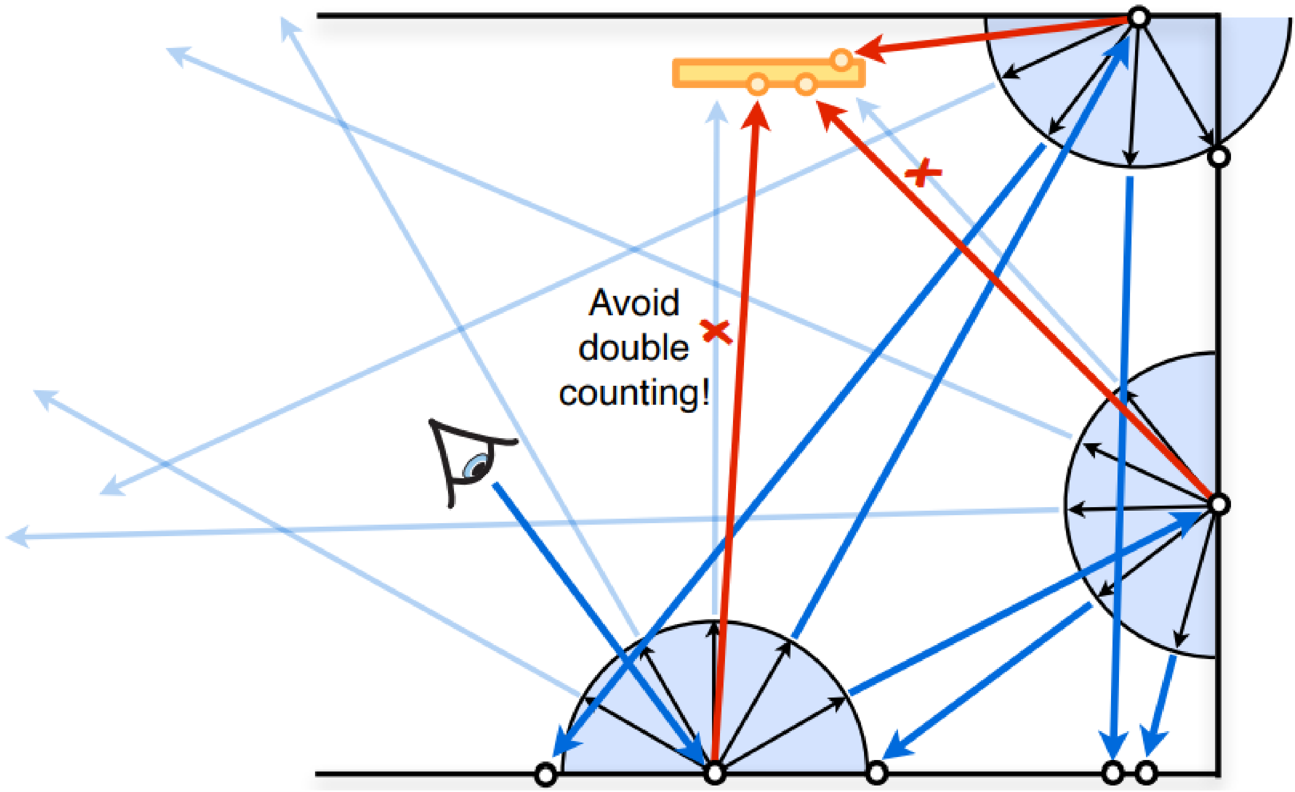
Recursive Monte Carlo Ray Tracing



$$L(\mathbf{x},\vec{\omega})\approx L_e(\mathbf{x},\vec{\omega})+\frac{1}{N}\sum_{i=1}^{N}\frac{f_r\left(\mathbf{x},\vec{\omega}',\vec{\omega}\right)L\left(r\left(\mathbf{x},\vec{\omega}'\right),-\vec{\omega}'\right)\cos\theta'}{p\left(\vec{\omega}'\right)}$$

Recursively calling Monte Carlo Integration will work but is not efficient. We could partition the integrand to make a better approximation. One way is to estimate direct illumination separately from indirect illumination, then add the two:

$$L(\mathbf{x},\vec{\omega}) = L_e(\mathbf{x},\vec{\omega}) + L_d(\mathbf{x},\vec{\omega}) + L_i(\mathbf{x},\vec{\omega})$$



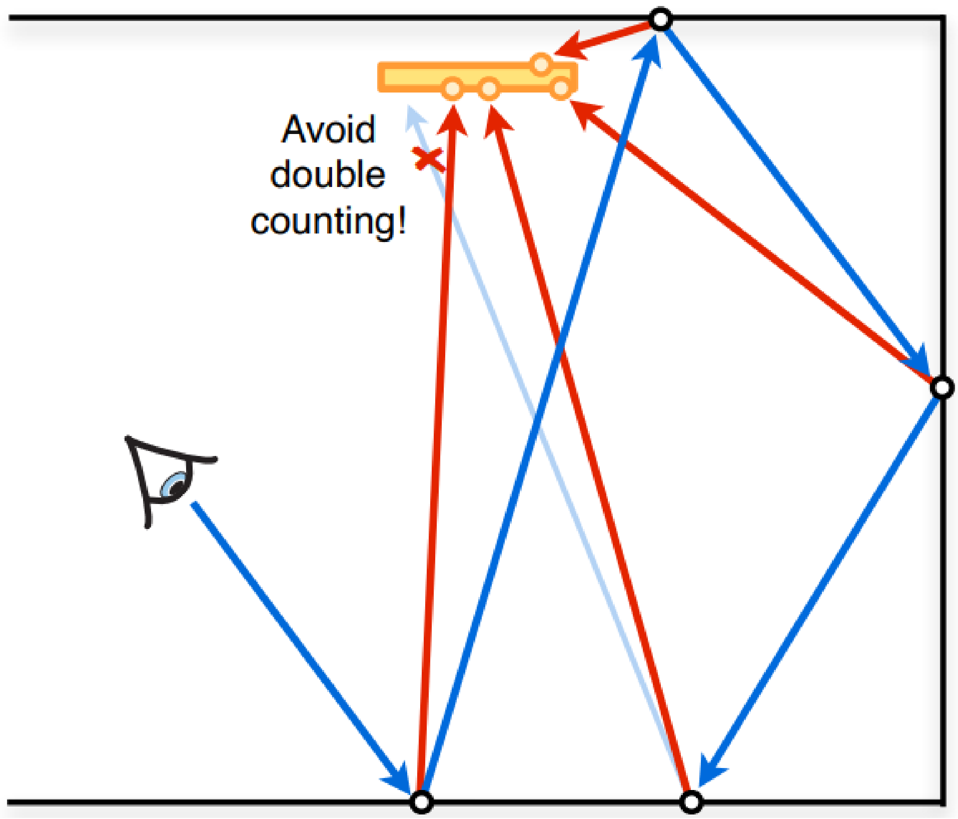

```C++
color shade (ray)
{
        if ray intersects emitter
                Le += light from emitter
        Ld = 0 // Direct
        Li = 0 // Indirect
        for light in all lights // direct illumination
                if no occlusion
                        Ld += brdf weighted radiance;
        for 1:N sample rays // indirect illumination
                do // avoid double counting
                        ω = random direction in hemisphere above n;
                while ray(x, w) hits a emitter
                Li += brdf * shade(ray(x, ω)) * cos() / pdf();
        return Le + Ld + Li;
}
```

Note that the above code does not have a termination condition and will trace indefinitely the ray.

However, the sampled rays after several bounces will grow geometrically which may cause a disaster. We will see 3 methods to solve this problem.

Path Tracing

Set the number of sampled incident rays N at each interaction point to be 1, i.e. sample one direct ray and one indirect ray at each interaction point.




```C++
// Simple PT
color shade (ray)
{
	if ray intersects emitter
		Le += light from emitter
	Ld = 0 // Direct
	Li = 0 // Indirect
	// direct illumination
	for light in all lights
		if no occlusion
			Ld += brdf weighted radiance;

	// indirect illumination
	do // avoid double counting
		ω = random direction in hemisphere above n;
	while ray(x, w) hits a emitter
	Li += brdf * shade(ray(x, ω)) * cos() / pdf();

	return Le + Ld + Li;
}
```

Russian Roulette

One strategy to terminate path tracing

1. Choose some termination probability $q \in (0, 1)$
2. Generate a random number ξ

$$F' = \begin{cases} \frac{F}{1-q} & \xi > q \\ 0 & \text{otherwise} \end{cases}$$

where F is any quantity. In this way, the expected value is

$$\mathbb{E}[F'] = (1-q) \cdot \left(\frac{\mathbb{E}[F]}{1-q} \right) + q * 0 = \mathbb{E}[F]$$

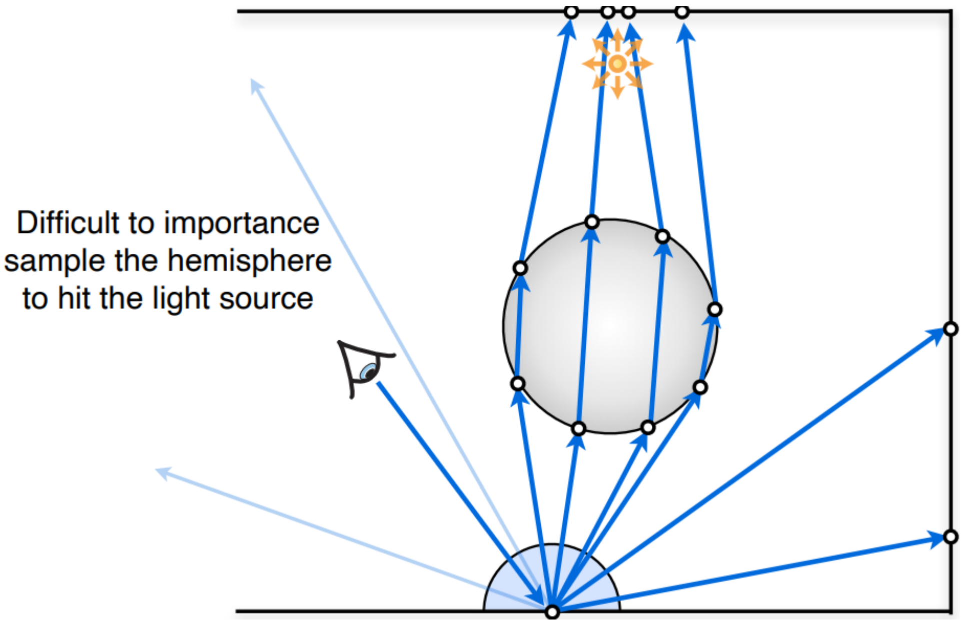
C++

```
// Russian Roulette PT
color shade (ray)
{
        if ray intersects emitter
                Le += light from emitter
        Ld = 0 // Direct
        Li = 0 // Indirect
        // direct illumination
        for light in all lights
                if no occlusion
                        Ld += brdf weighted radiance;

        // indirect illumination
        if rand() > q
                do // avoid double counting
                        ω = random direction in hemisphere above n;
                while ray(x, w) hits a emitter
                Li += brdf * shade(ray(x, ω)) * cos() / pdf();

        return Le + Ld + Li/(1-q);
}
```

Path Tracing have problems handling caustics because it's difficult to importance sample the hemisphere to hit the light source.



Light Tracing

The total radiance contributing to pixel j is

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

where $W_e(\mathbf{x}, \vec{\omega})$ is the response of the sensor at film location $\mathbf{x}$ to radiance arriving from direction $\vec{\omega}$, which is often referred to as *emitted importance*.

Radiance & Importance Duality

Radiance:

- emitted from light sources
- describes amount of light traveling within a differential beam

Importance:

- emitted from sensors
- describes the response of the sensor to radiance traveling within a differential beam

$$\begin{aligned}
I_j &= \int_{A_{\text{film}}} \int_{H^2} W_e(\mathbf{x}, \vec{\omega}) L_i(\mathbf{x}, \vec{\omega}) \cos\theta d\vec{\omega} d\mathbf{x} \\
&= \int_{A_{\text{film}}} \int_{A} W_e(\mathbf{x}, \mathbf{y}) G(\mathbf{x}, \mathbf{y}) L_o(\mathbf{y}, \mathbf{x}) d\mathbf{y} d\mathbf{x} \\
&= \int_{A_{\text{film}}} \int_{A} \int_{A_{\text{light}}} W_e(\mathbf{x}, \mathbf{y}) G(\mathbf{x}, \mathbf{y}) f(\mathbf{y}, \mathbf{z}, \mathbf{x}) G(\mathbf{y}, \mathbf{z}) L_e(\mathbf{z}, \mathbf{y}) d\mathbf{z} d\mathbf{y} d\mathbf{x} \\
&= \int_{A_{\text{light}}} \int_{A} \int_{A_{\text{film}}} W_e(\mathbf{x}, \mathbf{y}) \underbrace{G(\mathbf{y}, \mathbf{x}) f(\mathbf{y}, \mathbf{x}, \mathbf{z}) G(\mathbf{z}, \mathbf{y})}_{\text{symemetric functions}} L_e(\mathbf{z}, \mathbf{y}) d\mathbf{x} d\mathbf{y} d\mathbf{z} \\
&= \int_{A_{\text{light}}} \int_{A} W_o(\mathbf{y}, \mathbf{z}) G(\mathbf{z}, \mathbf{y}) L_e(\mathbf{z}, \mathbf{y}) d\mathbf{y} d\mathbf{z} \\
&= \int_{A_{\text{light}}} \int_{H^2} W_i(\mathbf{z}, \vec{\omega}) L_e(\mathbf{z}, \vec{\omega}) \cos\theta d\vec{\omega} d\mathbf{z}
\end{aligned}$$

By defining

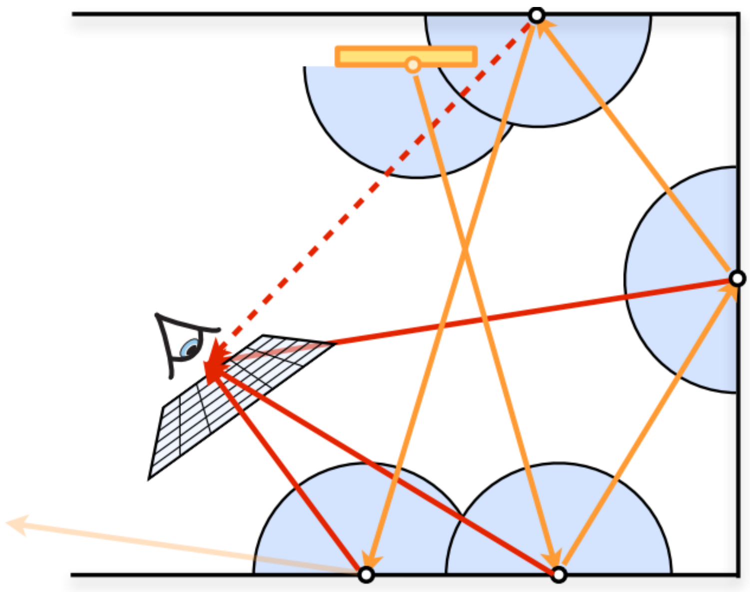
$$W_o(\mathbf{y}, \mathbf{z}) = W_e(\mathbf{x}, \mathbf{y}) G(\mathbf{y}, \mathbf{x}) f(\mathbf{y}, \mathbf{z}, \mathbf{x})$$

we have the duality of radiance and importance. In summary,

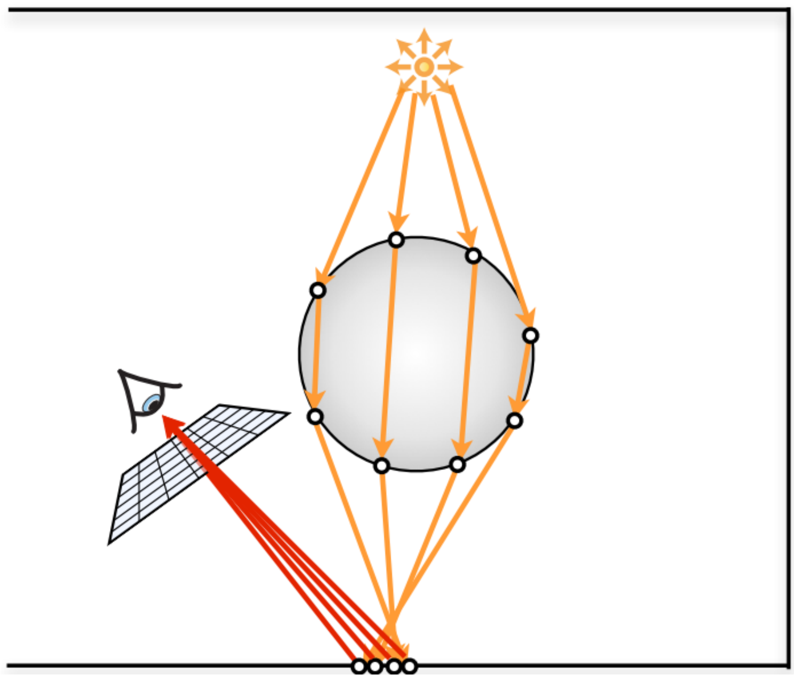
$$\begin{aligned}
I_j &= \overbrace{\int_{A_{\text{film}}} \int_{H^2} \overbrace{W_e(\mathbf{x}, \vec{\omega})}^{\text{emitted importance}} \overbrace{L_i(\mathbf{x}, \vec{\omega})}^{\text{incident radiance}} \cos\theta d\vec{\omega} d\mathbf{x}}^{\text{Path tracing}} \\
&= \underbrace{\int_{A_{\text{light}}} \int_{H^2} \underbrace{W_i(\mathbf{z}, \vec{\omega})}_{\text{incident importance}} \underbrace{L_e(\mathbf{z}, \vec{\omega})}_{\text{emitted radiance}} \cos\theta d\vec{\omega} d\mathbf{z}}_{\text{Light tracing}}
\end{aligned}$$

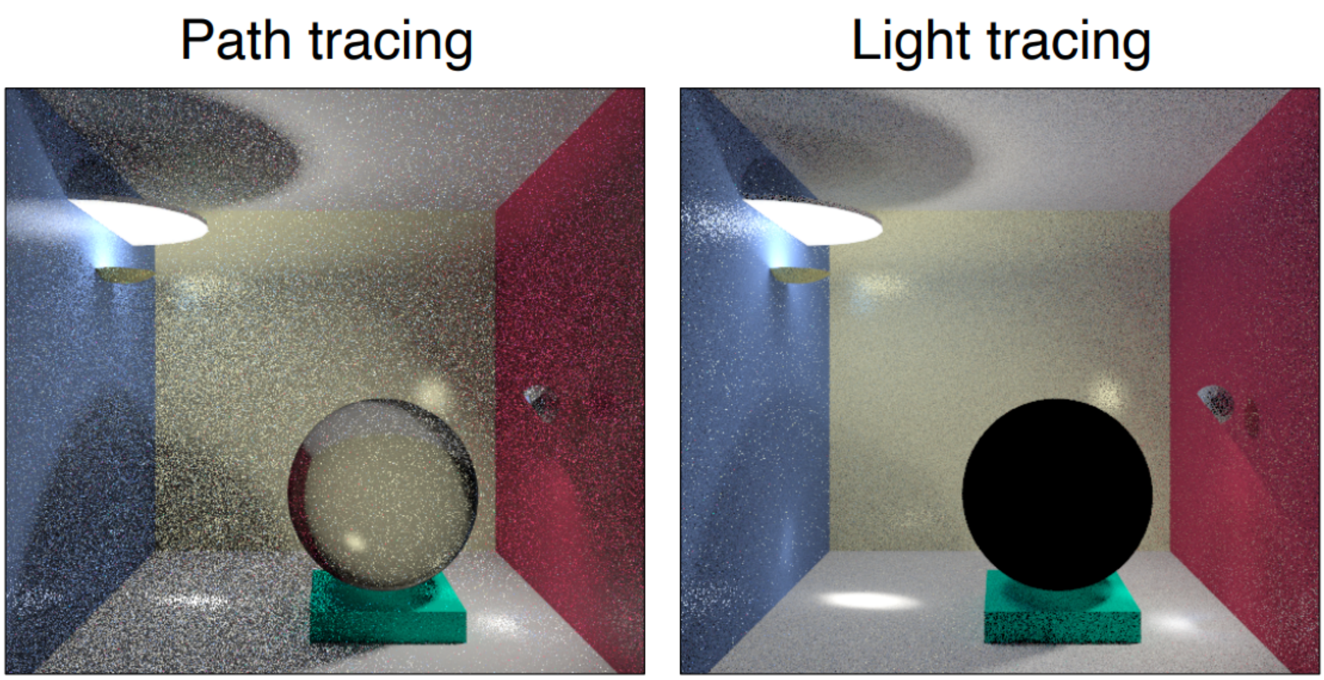
The light tracing steps are similar to path tracing:

- Shoot multiple paths from light sources
- Search/Query for importance
- Connect to the image using next event estimation (a.k.a. shadow rays in PT)



Light tracing is very useful for dealing with caustics:





Images courtesy of F. Suykens

Path Integral Framework

If we write explicitly the rendering equation without the recursive term, it would be like this

$$
\begin{aligned}
I_j &= \int_A \int_A W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) G\left(\mathbf{x}_0, \mathbf{x}_1\right) L_o\left(\mathbf{x}_1, \mathbf{x}_0\right) d\mathbf{x}_1 d\mathbf{x}_0 \\
&= \underbrace{\iint_A W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) L_e\left(\mathbf{x}_1, \mathbf{x}_0\right) G\left(\mathbf{x}_0, \mathbf{x}_1\right) d\mathbf{x}_1 d\mathbf{x}_0}_{\text{Emission: } LE} \\
&+ \underbrace{\iiint_A W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) L_e\left(\mathbf{x}_2, \mathbf{x}_1\right) G\left(\mathbf{x}_0, \mathbf{x}_1\right) f\left(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_0\right) G\left(\mathbf{x}_1, \mathbf{x}_2\right) d\mathbf{x}_2 d\mathbf{x}_1 d\mathbf{x}_0 + \cdots}_{\text{Direct illumination(3 vertices): } L(D|S)E} \\
&+ \underbrace{\int \cdots \int_A W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) L_e\left(\mathbf{x}_k, \mathbf{x}_{k-1}\right) G\left(\mathbf{x}_0, \mathbf{x}_1\right) \prod_{j=1}^{k-1} f\left(\mathbf{x}_j, \mathbf{x}_{j+1}, \mathbf{x}_{j-1}\right) G\left(\mathbf{x}_j, \mathbf{x}_{j+1}\right) d\mathbf{x}_k \cdots d\mathbf{x}_0 + \cdots}_{(k-1)\text{-bounce illumination } (k+1 \text{ vertices })}
\end{aligned}
$$

Define

- the space of all paths with k segments

$$\mathcal{P}_k = \{\overline{\mathbf{x}} = \mathbf{x}_0, \ldots, \mathbf{x}_k; \mathbf{x}_0, \ldots, \mathbf{x}_k \in A\}$$

- the throughput of path $\overline{\mathbf{x}}_k$

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

- the path space, i.e. the space of all paths of all lengths

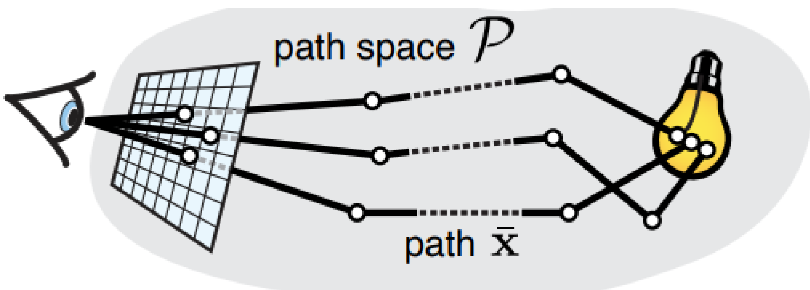
$$\mathcal{P} = \cup_{k=1}^{\infty} \mathcal{P}_k$$

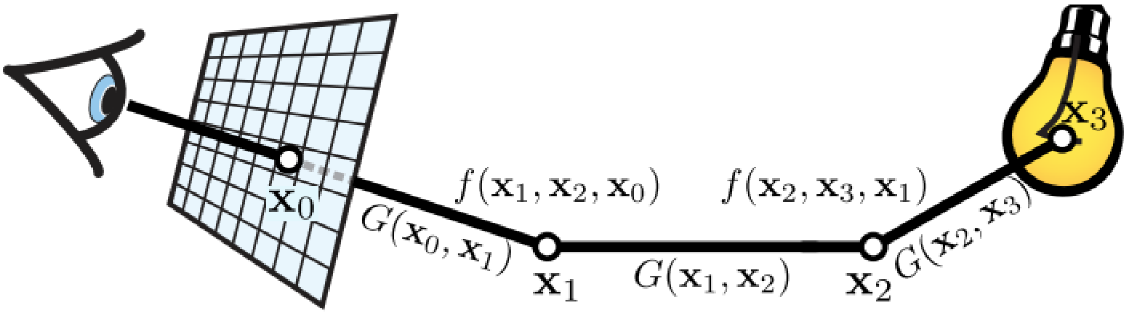
  Then the rendering equation becomes

$$
\begin{aligned}
I_j &= \int_A \int_A W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) G\left(\mathbf{x}_0, \mathbf{x}_1\right) L_o\left(\mathbf{x}_1, \mathbf{x}_0\right) d\mathbf{x}_1 d\mathbf{x}_0 \\
&= \int_{\mathcal{P}_1} W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) L_e\left(\mathbf{x}_1, \mathbf{x}_0\right) T\left(\overline{\mathbf{x}}_1\right) d\overline{\mathbf{x}}_1 \\
&+ \int_{\mathcal{P}_2} W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) L_e\left(\mathbf{x}_2, \mathbf{x}_1\right) T\left(\overline{\mathbf{x}}_2\right) d\overline{\mathbf{x}}_2 + \cdots \\
&+ \int_{\mathcal{P}_k} W_e\left(\mathbf{x}_0, \mathbf{x}_1\right) L_e\left(\mathbf{x}_k, \mathbf{x}_{k-1}\right) T\left(\overline{\mathbf{x}}_k\right) d\overline{\mathbf{x}}_k + \cdots
\end{aligned}
$$

which gives some kind of uniform expression

$$I_j = \underbrace{\int_{\mathcal{P}} W_e(\mathbf{x}_0, \mathbf{x}_1) L_e(\mathbf{x}_k, \mathbf{x}_{k-1}) T(\overline{\mathbf{x}})\, d\overline{\mathbf{x}}}_{\text{global ilumination(all paths of all lengths)}}$$



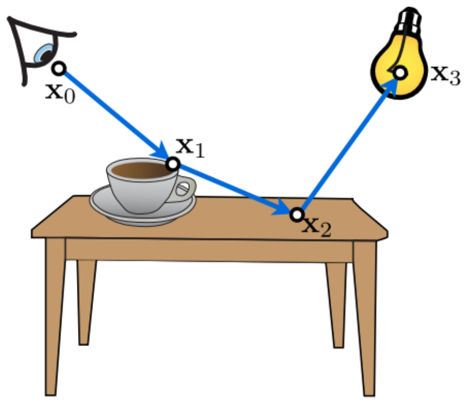
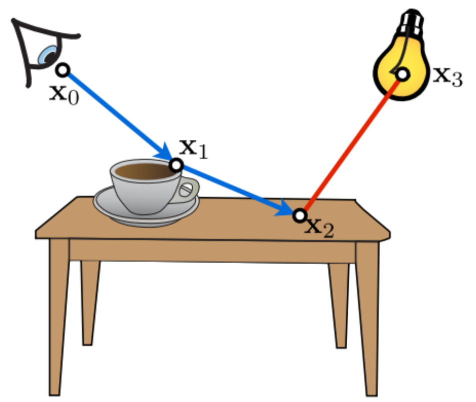
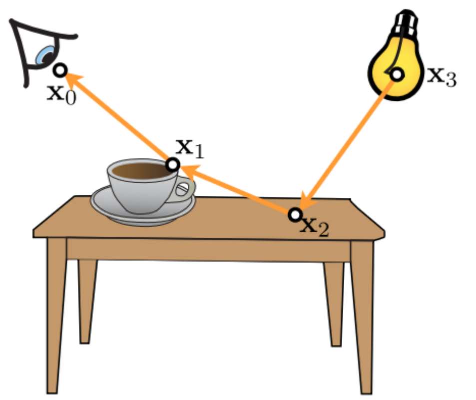


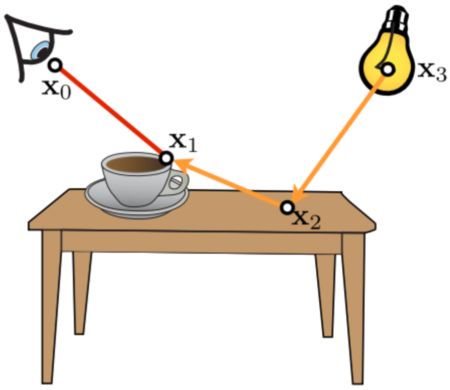
The integral is evaluated by Monte Carlo estimator

$$I_j \approx \frac{1}{N} \sum_{i=1}^{N} \frac{W_e\left(\mathbf{x}_{i,0}, \mathbf{x}_{i,1}\right) L_e\left(\mathbf{x}_{i,k}, \mathbf{x}_{i,k-1}\right) T\left(\overline{\mathbf{x}}_i\right)}{p\left(\overline{\mathbf{x}}_i\right)}$$

where $p(\overline{\mathbf{x}}) = p(\mathbf{x}_0, \ldots, \mathbf{x}_k)$ is the joint PDF of path vertices.

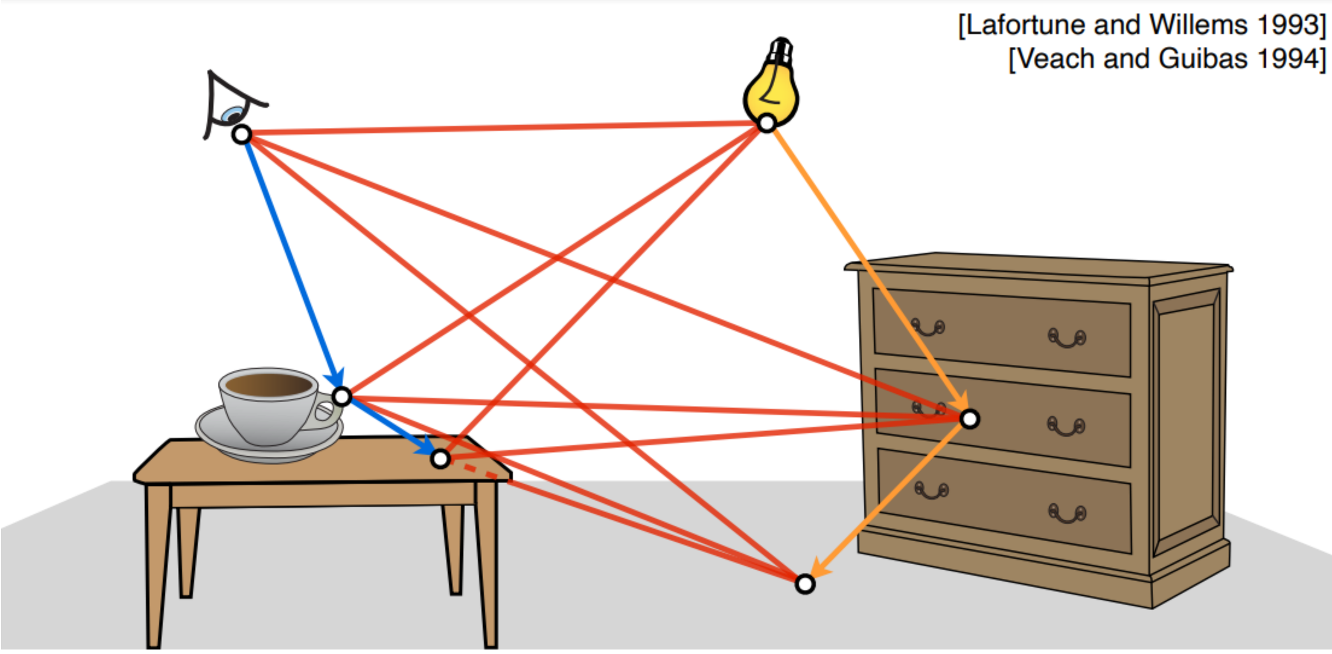
Still, the process could be more efficient with NEE(next event estimation)

Tracing type	**Example**	**Path pdf**
Path tracing		$p(\overline{\mathbf{x}}) = p(\mathbf{x}_0)p(\mathbf{x}_1 \mid \mathbf{x}_0)p(\mathbf{x}_2 \mid \mathbf{x}_0\mathbf{x}_1)p(\mathbf{x}_3 \mid \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2)$
Path tracing with NEE		$p(\overline{\mathbf{x}}) = p(\mathbf{x}_0)p(\mathbf{x}_1 \mid \mathbf{x}_0)p(\mathbf{x}_2 \mid \mathbf{x}_0\mathbf{x}_1)p(\mathbf{x}_3)$
Light tracing		$p(\overline{\mathbf{x}}) = p(\mathbf{x}_0 \mid \mathbf{x}_3\mathbf{x}_2\mathbf{x}_1)p(\mathbf{x}_1 \mid \mathbf{x}_3\mathbf{x}_2)p(\mathbf{x}_2 \mid \mathbf{x}_3)p(\mathbf{x}_3)$

Tracing type	Example	Path pdf
Light tracing with NEE		$p(\overline{\mathbf{x}}) = p(\mathbf{x}_0)p(\mathbf{x}_1 \mid \mathbf{x}_3\mathbf{x}_2)p(\mathbf{x}_2 \mid \mathbf{x}_3)p(\mathbf{x}_3)$

Bidirectional Path Tracing

We could combine light tracing and path tracing using bidirectional path tracing

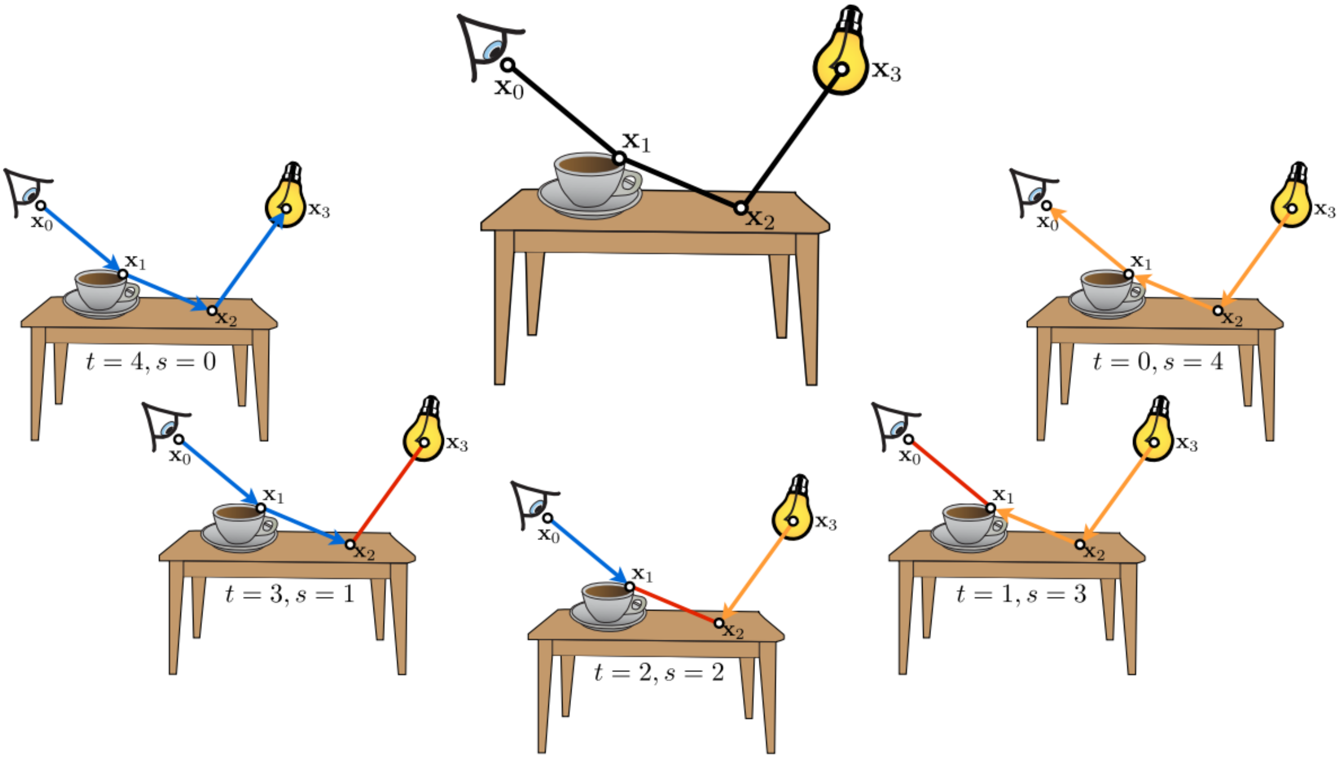


Steps:

- Sample a light path with length s
- Sample a camera path with length t
- Connect vertices in both paths to construct subpaths

Remark:

- Every path (formed by connecting camera sub-path to light sub-path) with k vertices can be constructed using $k+1$ strategies
- For a particular path length, all strategies estimate the same integral
- Each strategy has a different PDF, i.e. each strategy has different strengths and weaknesses



Bidirectional path tracing is not good at simulate scenes containing $LS + DS + E$ paths.