## Monte Carlo Path Tracing

**Today**

- Path tracing
- Random walks and Markov chains
- Adjoint equations
- Light ray tracing
- Bidirectional ray tracing

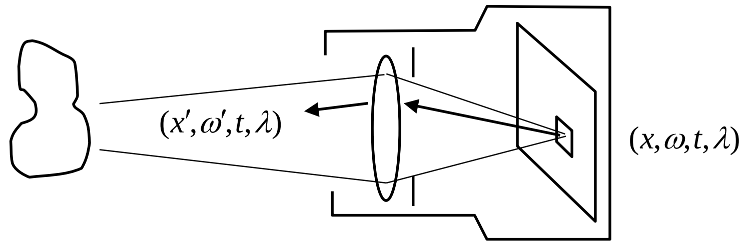
**Next – Henrik Wann Jensen**

- Irradiance caching
- Photon mapping



# Path Tracing

## The Measurement Equation




$$R=\iint\iint P(x,\lambda')S(x,\omega,t)L(T(x,\omega,\lambda),t,\lambda)\; d\vec{A}\bullet d\vec{\omega}\,dt\,d\lambda$$

| | |
|---|---|
| **Pixel response** | $P(x,\lambda)$ |
| **Lens optics** | $(x',\omega')=T(x,\omega,\lambda)$ |
| **Shutter** | $S(x,\omega,t)$ |
| **Scene radiance** | $L(x,\omega,t,\lambda)$ |



## Light Transport

**Integrate over all paths of all lengths**

$$L(x_{k-1},x_k)$$
$$=\sum_{k=1}^{\infty}\int_{M^2}\cdots\int_{M^2} L_S(x_0,\cdots,x_{k-2},x_{k-1},x_k)dA(x_0)\cdots dA(x_{k-2})$$

**Questions**

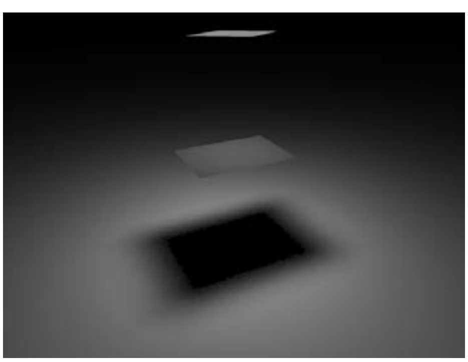
- **How to sample space of paths**
- **Find good estimators**

## Path Tracing: From Camera

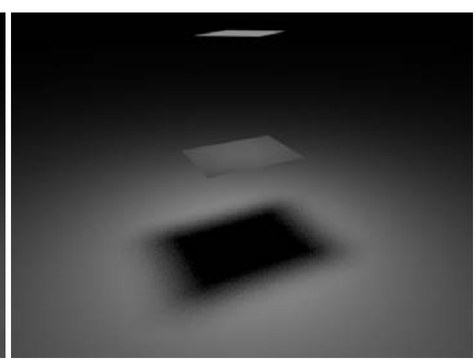
```
Step 1. Choose a ray given (x,y,u,v,t)
     weight = 1;
Step 2. Trace ray
Step 3. Randomly choose to compute Le or Lr
   Step 3a. If Le,
     return weight * Le;
   Step 3b. If Lr,
     weight = weight * reflectance;
     Choose new ray ~ BRDF pdf
     Go to Step 2.
```



## Penumbra: Trees vs. Paths

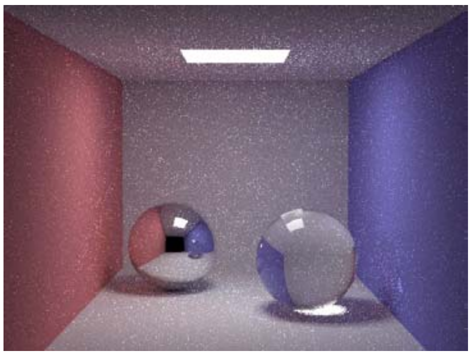


4 eye rays per pixel
16 shadow rays per eye ray

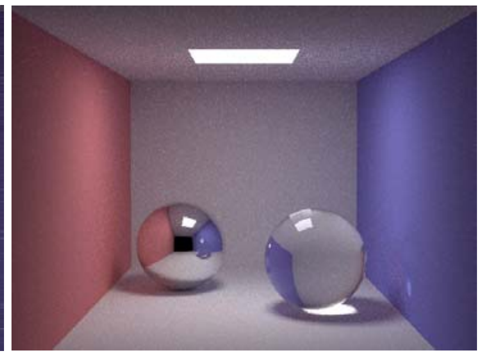


64 eye rays per pixel
1 shadow ray per eye ray

## Cornell Box: Path Tracing



10 rays per pixel



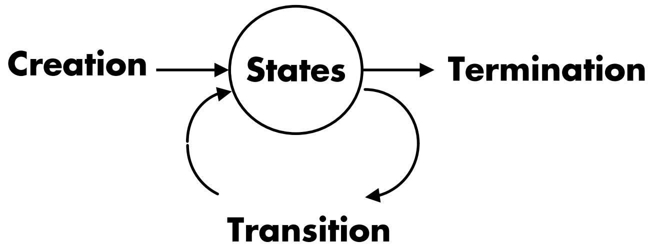
100 rays per pixel

From Jensen, Realistic Image Synthesis Using Photon Maps



# Discrete Random Walk

## Discrete Random Process




**Assign probabilities to each process**

$p_i^0$ : probability of creation in state $i$

$p_{i,j}$ : probability of transition from state $i \rightarrow j$

$p_i^*$ : probability of termination in state $i$ $\quad p_i^* = 1 - \sum_j p_{i,j}$



## Markov Chain

**Probability of being in state *i* after *n* transitions**

$$P_j^0 = p_j^0 \qquad P^0 = p^0$$

$$P_j^1 = p_j^0 + \sum_i p_{i,j} P_i^0 \qquad P^1 = p^0 + MP^0$$

$$\vdots \qquad\qquad M_{i,j} = p_{i,j}$$

$$P_j^n = p_j^0 + \sum_i p_{i,j} P_i^{n-1} \qquad P^n = p^0 + MP^{n-1}$$

## Equilibrium Distribution of States

**Total probability of being in state $i$**

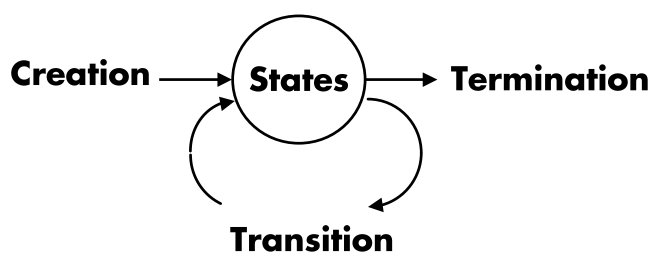
$$P_i = \sum_{k=0}^{\infty} P_i^k = \left(I + M + M^2 + \cdots\right) p^0$$

**But this is the solution of the matrix equation**

$$(I - M)P = p^0$$



## Monte Carlo Algorithm




1. **Generate random particle paths from source.**
2. **Undertake a discrete random walk.**
3. **Count how many terminate in state $i$**

**[von Neumann and Ulam; Forsythe and Leibler; 1950s]**

## Monte Carlo Algorithm: Analysis

**Define a random variable on the space of paths**

**Path:** $\alpha_k = (i_1, i_2, \ldots, i_k)$

**Probability:** $P(\alpha_k) = p_{i_1}^0 \times p_{i_1,i_2} \cdots p_{i_{k-1},i_k} \times p_{i_k}^*$



## Monte Carlo Algorithm

**Define a random variable on the space of paths**

**Path:** $\alpha_k = (i_1, i_2, \ldots, i_k)$

**Probability:** $P(\alpha_k) = p_{i_1}^0 \times p_{i_1,i_2} \cdots p_{i_{k-1},i_k} \times p_{i_k}^*$

**Expectation:**

$$E[W] = \sum_{k=1}^{\infty} \sum_{\alpha_k} P(\alpha_k) W(\alpha_k)$$

## Estimator

**Count the number of particles terminating in state *j***

**Estimator:** $W_j(\alpha_k) = \frac{\delta_{i_k,j}}{p^*_{i_k}}$

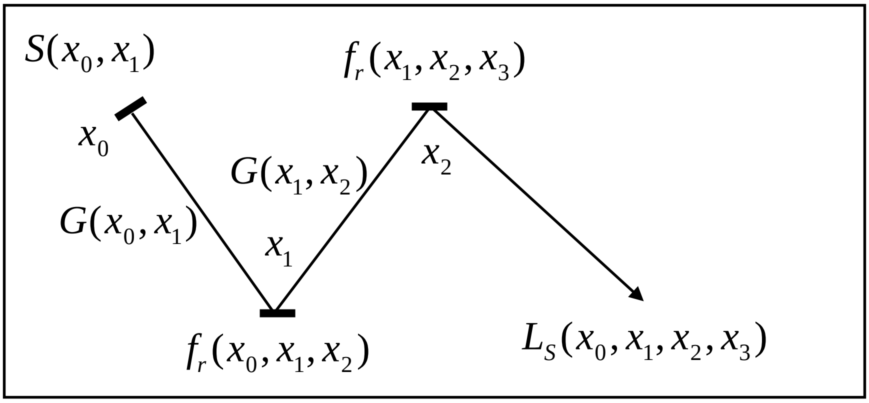
**Unbiased:** $E[W_j] = \sum_{k=1}^{\infty} \sum_{i_k} \cdots \sum_{i_1} (p^0_{i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} p^*_{i_k}) \frac{\delta_{i_k,j}}{p^*_j}$

$$= \left[ p^0 \right]_j + \left[ M p^0 \right]_j + \left[ M^2 p^0 \right]_j \cdots$$



# Adjoint Formulation

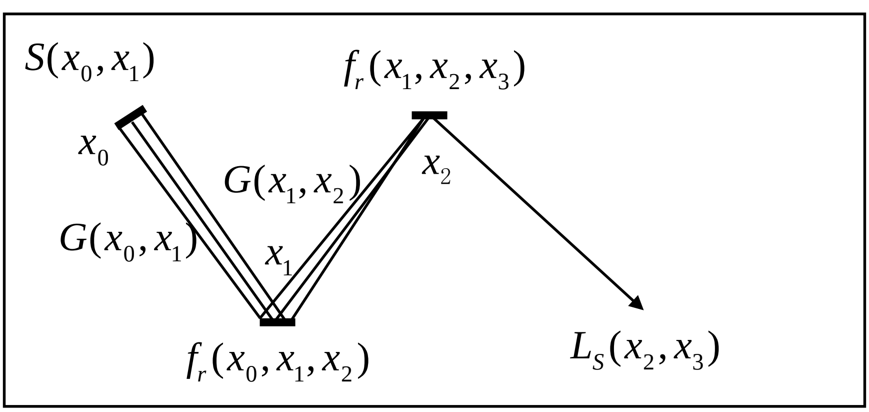
# Light Path




$$L_S(x_0, x_1, x_2, x_3) = S(x_0, x_1) G(x_o, x_1) f_r(x_0, x_1, x_2) G(x_1, x_2) f_r(x_1, x_2, x_3)$$

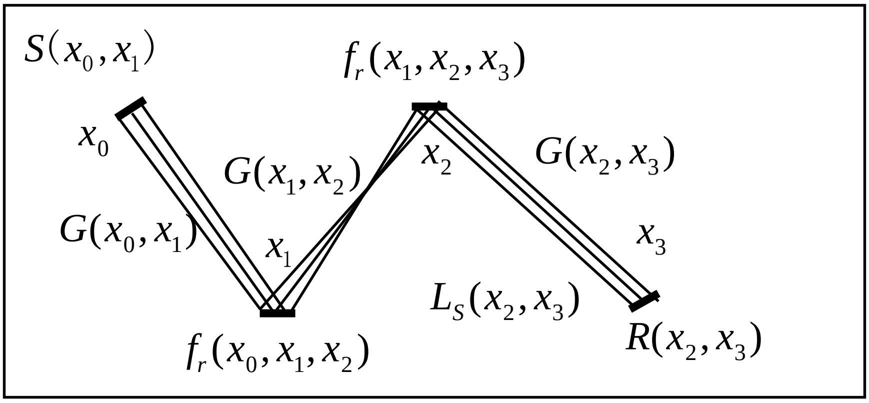


# Light Paths




$$L_S(x_2, x_3) = \int_{M^2} \int_{M^2} L_S(x_0, x_1, x_2, x_3) dA(x_0) dA(x_1)$$

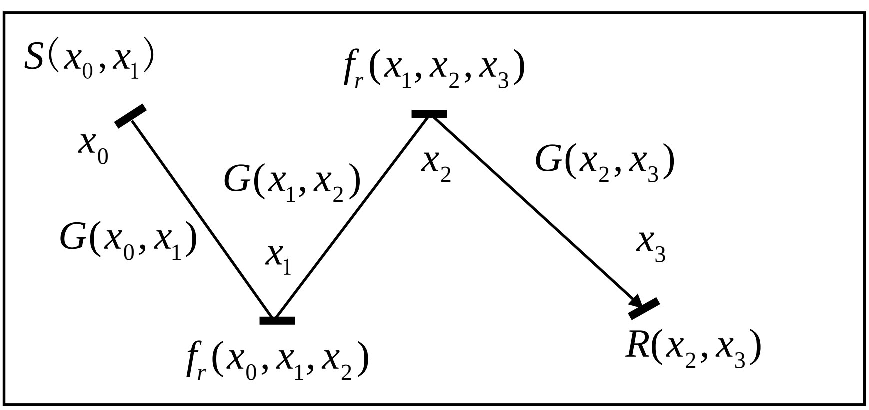
## Measurement




$$M = \int_{M^2} \int_{M^2} L_S(x_2, x_3) G(x_2, x_3) R(x_2, x_3)\, dA(x_2)\, dA(x_3)$$

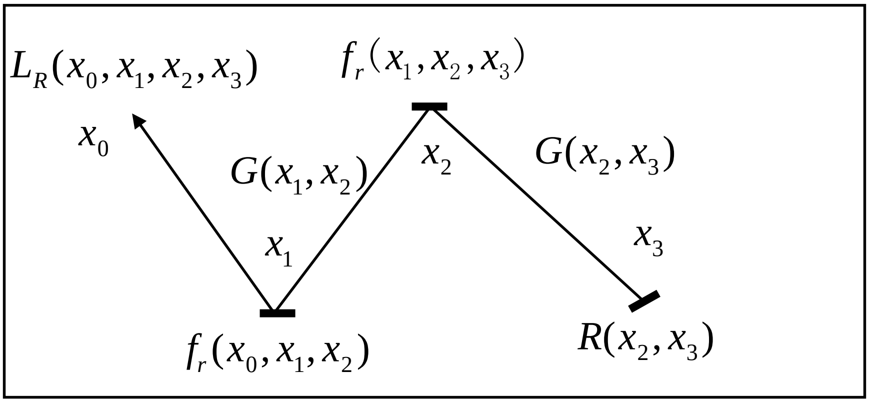


## Symmetric Light Path




$$M = S(x_0, x_1) G(x_o, x_1) f_r(x_0, x_1, x_2) G(x_1, x_2) f_r(x_1, x_2, x_3) G(x_2, x_3) R(x_2, x_3)$$

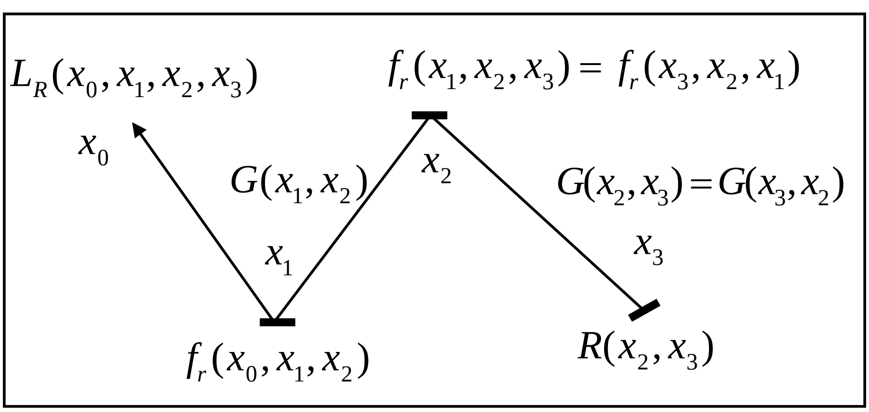
## Symmetry




$$L_R(x_0,x_1,x_2,x_3) = f_r(x_0,x_1,x_2)G(x_1,x_2)f_r(x_1,x_2,x_3)G(x_2,x_3)R(x_2,x_3)$$

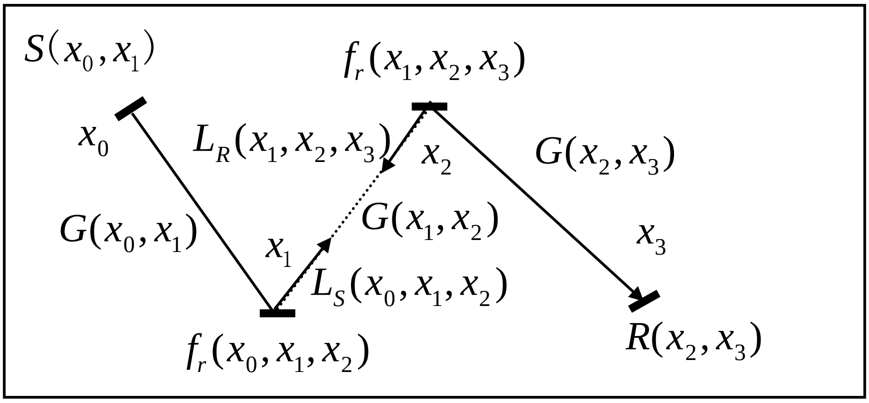


## Symmetry




$$\begin{aligned} L_R(x_0,x_1,x_2,x_3) &= f_r(x_0,x_1,x_2)G(x_1,x_2)f_r(x_1,x_2,x_3)G(x_2,x_3)R(x_2,x_3) \\ &= R(x_3,x_2)G(x_3,x_2)f_r(x_3,x_2,x_1)G(x_2,x_1)f_r(x_2,x_1,x_0) \end{aligned}$$

# Importance




$$M = \int_{M^2} \int_{M^2} L_S(x_0, x_1, x_2) G(x_1, x_2) L_R(x_1, x_2, x_3)\, dA(x_1) dA(x_2)$$

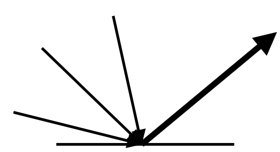


# Adjoint Equations

**Original equation**

$$K \circ f = \int K(x, y) f(y)\, dy$$

**Forward direction**

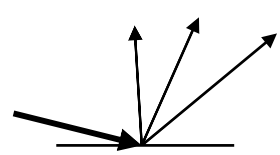


**Out-Scatter**

**Adjoint equation**

$$K^+ \circ f = \int K(x, y) f(x)\, dx$$

**Backward direction**



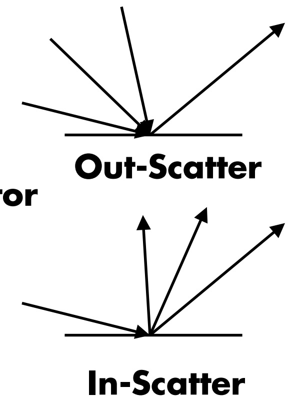
**In-Scatter**

## Self-Adjoint Equations

**Self-adjoint**

$$K^{+} = K$$

**Forward = backward operator**






## Forward = Backward Estimate

$$< f, g > = \int f(x) g(x)\, dx$$

$$\begin{aligned} < f, K \circ g > &= \int f(x) \left( \int K(x,y) g(y)\, dy \right) dx \\ &= \int \left( \int f(x) K(x,y) dx \right) g(y)\, dy \\ &= < K^{+} \circ f, g > \end{aligned}$$

## Three Consequences

1. Forward estimate equal backward estimate
   - May use forward or backward ray tracing

2. Adjoint solution
   - Importance sampling paths

3. Solve for small subset of the answer



## Example: Linear Equations

Solve a linear system $Mx = b$

Solve for a single $x_i$?

Solve the adjoint equation

Source $x_i$

Estimator $<(x_i + Mx_i + M^2 x_i + \cdots), b>$

More efficient than solving for all the unknowns [von Neumann and Ulam]

# Light Ray Tracing

## Path Tracing: From Lights

```
Step 1. Choose a light ray
Step 2. Trace ray
Step 3. Randomly decide whether to absorb or
   reflect the ray ~ reflectance
   if reflected
      goto Step 2
```

## Path Tracing: From Lights

Step 1. Choose a light ray

Choose a light source according to the light source power distribution function.

Choose a ray from the light source radiance (area) or intensity (point) distribution function

weight = 1;

Step 2. Trace ray

Step 3. Randomly decide whether to absorb or reflect the ray

if reflected

goto Step 2



## Path Tracing: From Lights

Step 1. Choose a light ray

Step 2. Trace ray

Step 3. Randomly decide whether to absorb or reflect the ray ~ reflectance

Step 3a. If reflected,

Randomly scatter ~ BRDF pdf

Go to Step 2.

Step 3b. If absorbed at the camera,

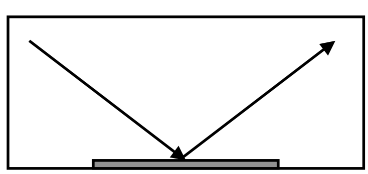
Record weight at x, y

Go to Step 1;

# Bidirectional Ray Tracing

## Delta Functions

**Mirrors (Caustics)**



**Eye ray tracing: ES*DL**

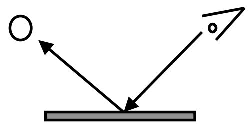
**Light ray tracing: LS*DE**

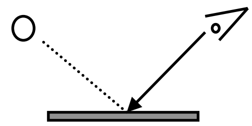
**ES*DS*L problematic (virtual point light source)**

**Even more problematic is LS*DS*DS*E**

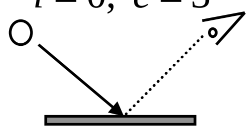
## Bidirectional Ray Tracing

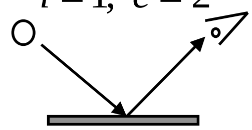
$k = l + e$





$l = 0,\ e = 3$ $l = 1,\ e = 2$





$l = 2,\ e = 1$ $l = 3,\ e = 0$

$k = 3$



## Path Pyramid

$k=3$ $(l=2, e=1)$

$k=4$

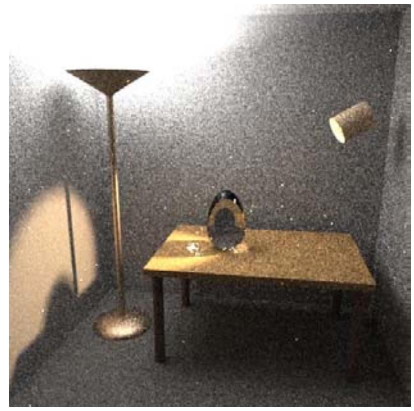
$k=5$

$k=6$ $(l=5, e=1)$

$l$ **From Veach and Guibas** $e$

## Comparison

**Bidirectional ray tracing**



**Path tracing**

**From Veach and Guibas**