Overview

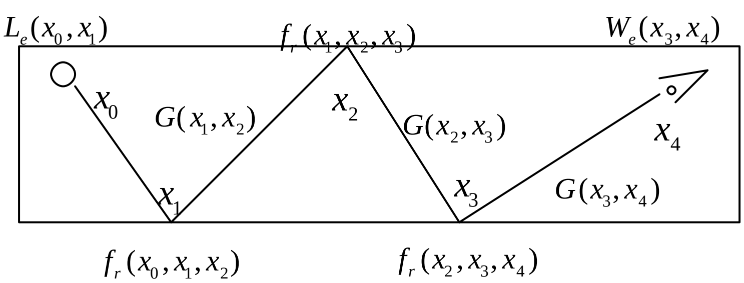
Tuesday

- Direct lighting
- Sampling distributions and shapes
- Importance and multiple importance sampling

Today

- Markov chains and path tracing
- Forward and backward tracing (adjoint equations)
- Bidirectional ray tracing
- Density estimation and photon tracing

Light Paths



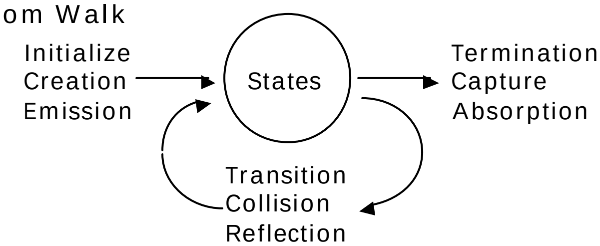
$$M = \sum_{k=1}^{\infty} \int_{M^2} \int_{M^2} \cdots \int_{M^2} L_e(x_0, x_1) G(x_o, x_1) f_r(x_0, x_1, x_2) \cdots f_r(x_{k-2}, x_{k-1}, x_k) G(x_{k-1}, x_k) W_e(x_{k-1}, x_k)\, dA_0 dA_1 \ldots dA_k$$

Light transport: Integrate over paths with k bounces

- Sample space of paths
- Find good estimators

Particle Simulation

Discrete Random Walk



von Neumann and Ulam; Forsythe and Leibler (‘50)

1. Generate random particle paths from source (receiver).
2. Count how many terminate in state $\boldsymbol{i}$.

Wasow (‘52)

1. Generate random particle paths from source (receiver).
2. Count how many pass through state $\boldsymbol{i}$.

Markov Process

Assign probabilities to each process

- Creation: p_i^0 : probability of particle being created in state i
- Transition: $p_{i,j}$: probability of transition from state $i \to j$
- Termination: p_i^* : probability of termination in state i

$$p_i^* = 1 - \sum_j p_{i,j}$$

Compute steady state probability of being in state $\boldsymbol{i}$

$$P_i^0 = p_i^0$$

$$P_i^1 = \sum_j p_{j,i} P_j^0$$

$$\vdots$$

$$P_i^n = \sum_j p_{j,i} P_j^{n-1}$$

$$P_i = \sum_{k=0}^{\infty} P_i^k$$

But this is the solution of

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

Random Walk Algorithm

Define a random variable on the space of paths

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

Expectation: $E[W] = \sum_a P(a)W(a) = \sum_{k=1}^{\infty} \sum_{a_k} P(a_k)W(a_k)$

Count the number of particles terminating in state j

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

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

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

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

Collision Estimator

Count the number of particles passing through state j

Weight: $W_j(a_k) = \sum_{m=1}^{k} d_{i_m,j}$

$$E[W_j] = \sum_{k=1}^{\infty} \sum_{i_k} \cdots \sum_{i_1} (p^0_{i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} p^*_{i_k}) \left(\sum_{m=1}^{k} d_{i_m,j} \right)$$

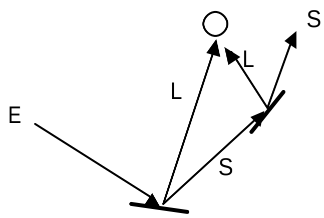
$$= \sum_{m=1}^{\infty} \sum_{i_m} \cdots \sum_{i_1} (p^0_{i_1} p_{i_1,i_2} \cdots p_{i_{m-1},i_m}) d_{i_m,j} \underbrace{\left(\sum_{k=m}^{\infty} \sum_{i_{m+1}} \cdots \sum_{i_k} p_{i_m,i_{m+1}} \cdots p_{i_{k-1},i_k} p^*_{i_k} \right)}_{\text{Probability that particle terminates} = 1.0}$$

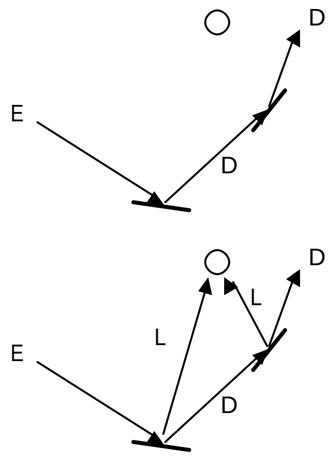
Is this better?

$$V[W^c_j] \le V[W^a_j] \quad \text{iff} \quad p^*_j \le \frac{\Pr[j \text{ never returns to } j]}{2 - \Pr[j \text{ never returns to } j]}$$

Path Tracing

Classic ray tracing	Path Tracing
[Whitted 1980]	[Kajiya 1986]





Bias towards the lights ...
Careful to not count twice!

Classic Path Tracing: Random Walk

1. Generate particle

$$w_0 = \frac{L_e(x_0, x_1)G(x_0, x_1)}{p_0(x_0, x_1)}$$

2. Test for termination

$$q_i \quad \text{probability of continuation}$$

$$q_i^* \quad \text{probability of termination}$$

3. Find intersection

$$w_{i+1} = w_i \frac{1}{q_i} \frac{f_r(x_{i-1}, x_i, x_{i+1})G(x_i, x_{i+1})}{p_i(x_i, x_{i+1})}$$

4. Repeat 2

Classic Path Tracing

This process yields a set of path samples

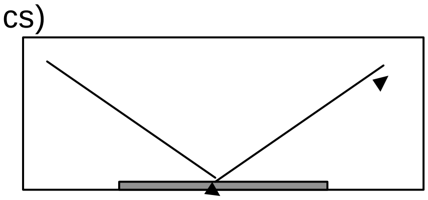
Final estimator

$$w_i = \frac{L_e(x_0, x_1) G(x_0, x_1)}{p_0(x_0, x_1)} W_e(x_{i-1}, x_i) \prod_{j=1}^{i-1} \frac{1}{q_j} \frac{f_r(x_{j-1}, x_j, x_{j+1}) G(x_j, x_{j+1})}{p_j(x_j, x_{j+1})}$$

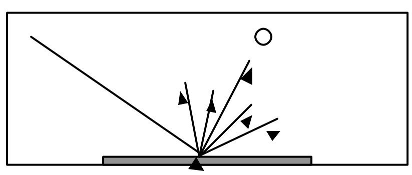
$$E[M] = \frac{1}{N} \sum_{i=1}^{N} w_i$$

Delta Functions

Mirrors (Caustics)



Point lights



LS*DS*E particularly problematic

Russian Roulette and Splitting

Previous estimators just counted particles

- Termination causes noise

Instead weight the particles by probabilities

- Weighting better, but terminate those with small weights

Russian Roulette

If (***w*** < ***wc***)

terminate with probability ***1-wc***

if survive, ***w=w/wc***

E[w]=(1-wc)*0+wc*(w/wc)=w

Splitting

if (***w*** > ***wm***)

split into n particles with ***w=w/n***

Forward=Backward Estimate

Inner product / Estimated quantity

$$KL = S$$

$$K^+L^+ = R = S^+$$

Justification for eye ray tracing

$$< S^+, L > = < K^+L^+, L > = < L^+, KL > = < L^+, S >$$

Self-adjoint

$$K(y,x) = K(x,y) \quad \Rightarrow \quad K^+ = K$$

CS348B Lecture 14 Pat Hanrahan, Spring 2000

Adjoint Equations

Estimated quantity

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

Original equation

$$Kg = \int K(x,y) g(y) dy$$

Estimated quantity

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

Adjoint equation

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

Three Consequences

1. Forward estimate equal backward estimate
2. Solve for small subset of the answer
3. Importance sampling paths

Example

Solve a linear system

$$Mx = b$$

Solve for a single x_i?

von Neumann and Ulam: Solve the adjoint equation

Source $\quad x_i$

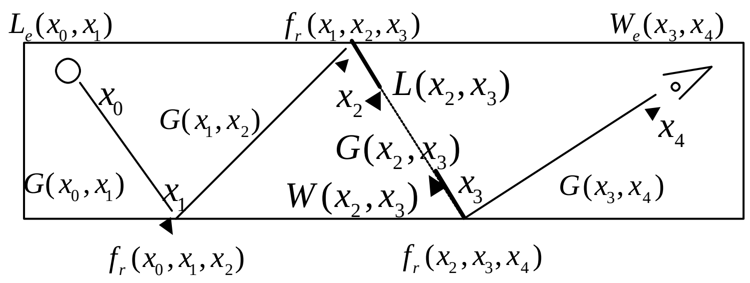
Estimator $\quad < (x_i + Mx_i + M^2 x_i + \cdots), b >$

More efficient than solving the whole system of equations

Applicable to image synthesis! Don’t solve if not seen

“Importance”

Importance is the adjoint solution



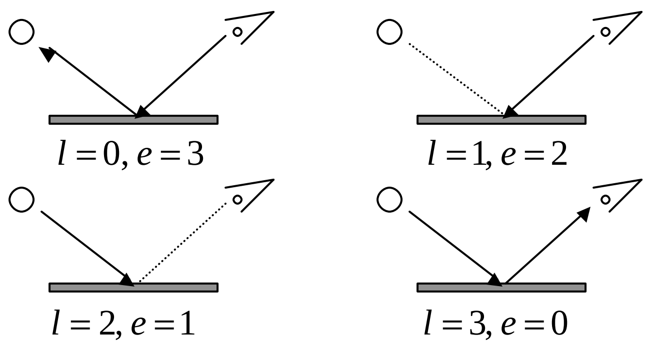
$$L(x_m, x_{m+1}) = \int_{M^2} \int_{M^2} \dots \int_{M^2} L_e(x_0, x_1) G(x_0, x_1) f_r(x_0, x_1, x_3) \cdots G(x_{m-1}, x_m) f_r(x_{m-1}, x_m, x_{m+1}) dA_0 dA_1 \dots dA_{m-1}$$

$$W(x_m, x_{m+1}) = \int_{M^2} \int_{M^2} \dots \int_{M^2} f_r(x_m, x_{m+1}, x_{m+2}) G(x_{m+1}, x_{m+2}) \cdots f_r(x_m, x_{m+1}, x_{m+2}) G(x_{k-1}, x_k) W_e(x_{k-1}, x_k)\, dA_{m+2} dA_{m+3} \dots dA_k$$

$$M = \int_{M^2} \int_{M^2} L(x_m, x_{m+1}) G(x_m, x_{m+1}) W(x_m, x_{m+1})\, dA_m dA_{m+1}$$

Bidirectional Ray Tracing

$k = l + e - 1$



Path Pyramid

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

$k = 4$

$k = 5$

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

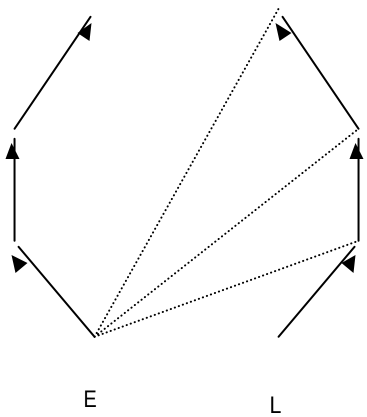
l From Veach and Guibas e

Comparison

Bidirectional ray tracing Path tracing

From Veach and Guibas

Tree of Paths



Efficiently generate a collection of bidirectional paths