Monte Carlo Path Tracing

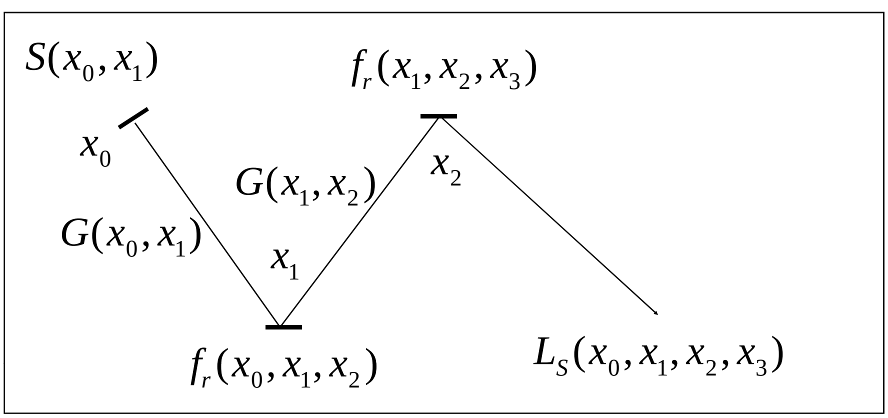
Today

- Path tracing
- Random walks and Markov chains
- Eye vs. light ray tracing
- Bidirectional ray tracing

Next

- Irradiance caching
- Photon mapping

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 Transport

Integrate over all paths

$$L(x_{k-1}, x_k)$$

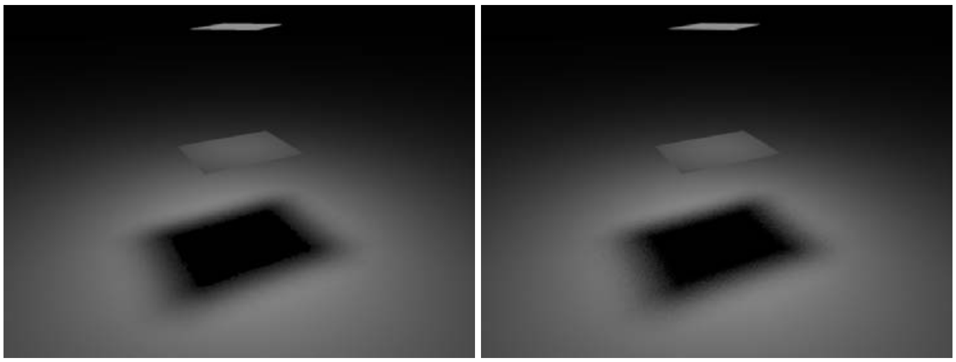
$$= \sum_{k=1}^{\infty} \int_{M^2} \cdots \int_{M^2} L_S(x_0, \cdots, x_{k-2}, x_{k-1}, x_k) dA(x_0) \cdots dA(x_{k-2})$$

Questions

- How to sample space of paths

Path Tracing

Penumbra: Trees vs. Paths



4 eye rays per pixel
16 shadow rays per eye ray

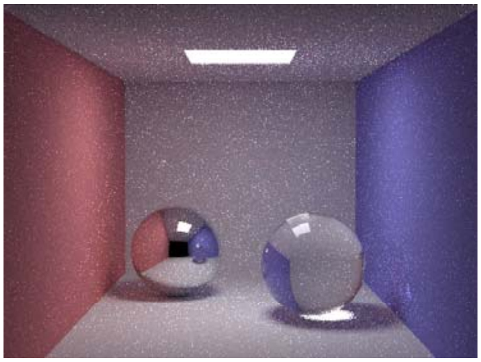
64 eye rays per pixel
1 shadow ray per eye ray

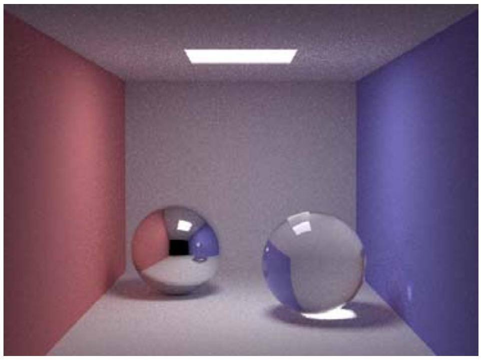
Path Tracing: From Camera

```
Step 1. Choose a camera ray r given the
   (x,y,u,v,t) sample
      weight = 1;
Step 2. Find ray-surface intersection
Step 3.
   if light
      return weight * Le();
   else
      weight *= reflectance(r)
      Choose new ray r' ~ BRDF pdf(r)
      Go to Step 2.
```

M. Fajardo Arnold Path Tracer

Cornell Box: Path Tracing





10 rays per pixel **100 rays per pixel**

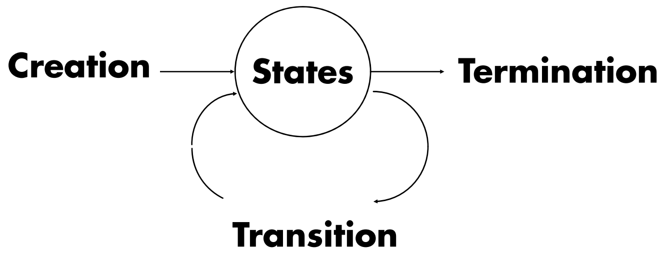
From Jensen, Realistic Image Synthesis Using Photon Maps

Path Tracing: Include Direct Lighting

```
Step 1. Choose a camera ray r given the
   (x,y,u,v,t) sample
     weight = 1;
Step 2. Find ray-surface intersection
Step 3.
     weight += Lr(light sources)
     Choose new ray r' ~ BRDF pdf(r)
     Go to Step 2.
```


Discrete Random Walk

Discrete Random Process



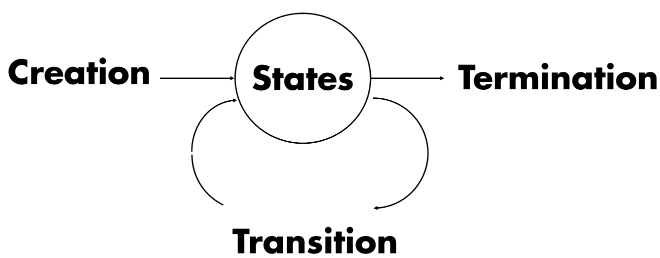
Assign probabilities to each process

p_i^0 : probability of creation in state i

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

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

Discrete Random Process

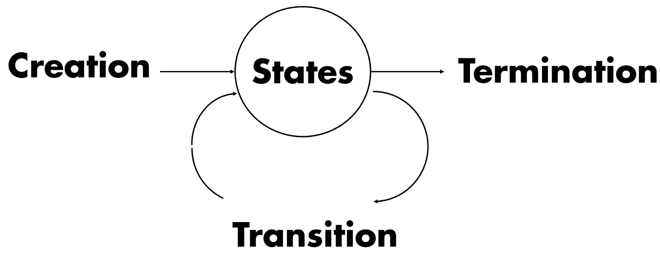


Equilibrium number of particles in each state

$$P_i = \sum_j p_{i,j} P_j + p_i^0 \qquad M_{i,j} = p_{i,j}$$

$$P = MP + p^0$$

Discrete Random Walk



1. Generate random particles from sources.
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

Define a random variable on the space of paths

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

Probability: $P(\alpha_k)$

Estimator: $W(\alpha_k)$

Expectation:

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

Monte Carlo Algorithm

Define a random variable on the space of paths

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

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

Estimator

Count the number of particles terminating in state *j*

$$E[W_j] = \sum_{k=1}^{\infty} \sum_{i_k} \cdots \sum_{i_1} (p^0_{i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} p^*_{i_k}) \frac{\delta_{i_k,j}}{p^*_j}$$
$$= \left[p^0\right]_j + \left[Mp^0\right]_j + \left[M^2 p^0\right]_j \cdots$$

Equilibrium Distribution of States

Total probability of being in states P

$$P = \left(I + M + M^2 + \cdots\right) p^0$$

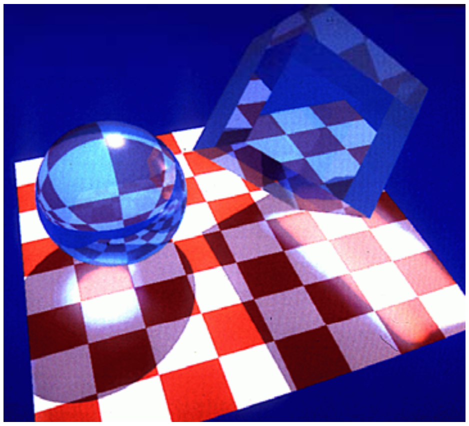
Note that this is the solution of the equation

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

Thus, the discrete random walk is an unbiased estimate of the equilibrium number of particles in each state

Light Ray Tracing

Examples



Backward ray tracing, Arvo 1986

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
   w = 1;
Step 2. Trace ray to find surface intersect
Step 3. Interaction
```

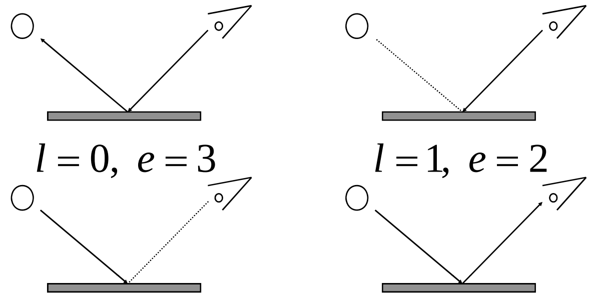
Path Tracing: From Lights

```
Step 1. Choose a light ray
Step 2. Find ray-surface intersection
Step 3. Interaction
  u = rand()
  if u < reflectance
    Choose new ray r ~ BRDF
    goto Step 2
  else
    terminate on surface; deposit energy
```


Bidirectional Path Tracing

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

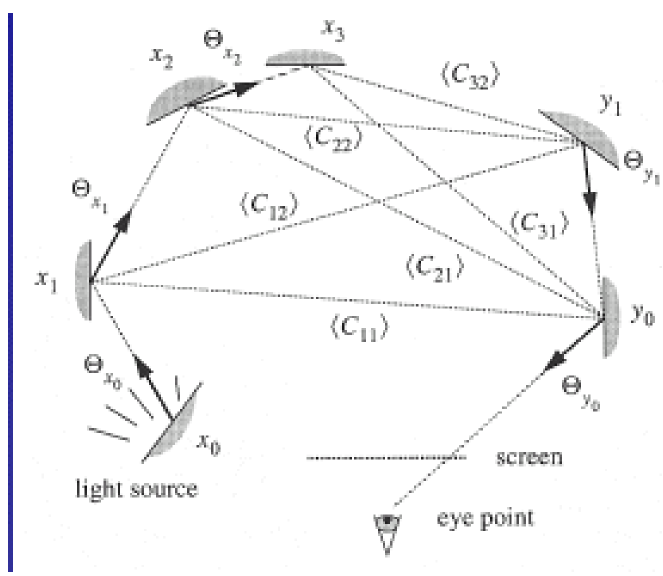
Same amount of time

Bidirectional path tracing
25 rays per pixel

Path tracing
56 rays per pixel

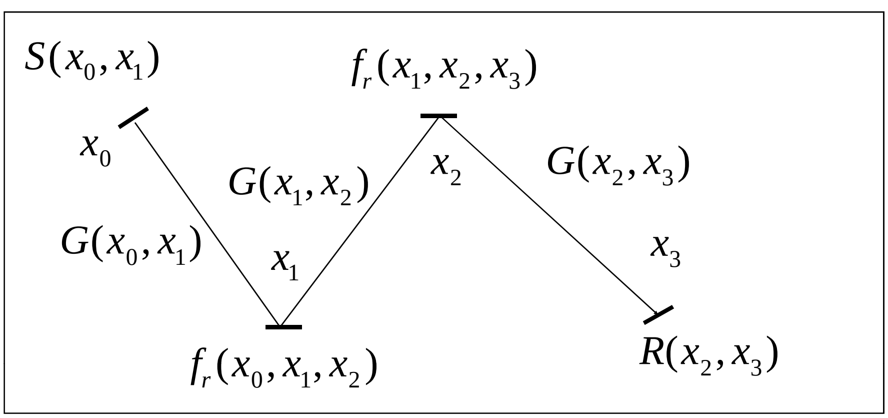
From Veach and Guibas

Generating All Paths



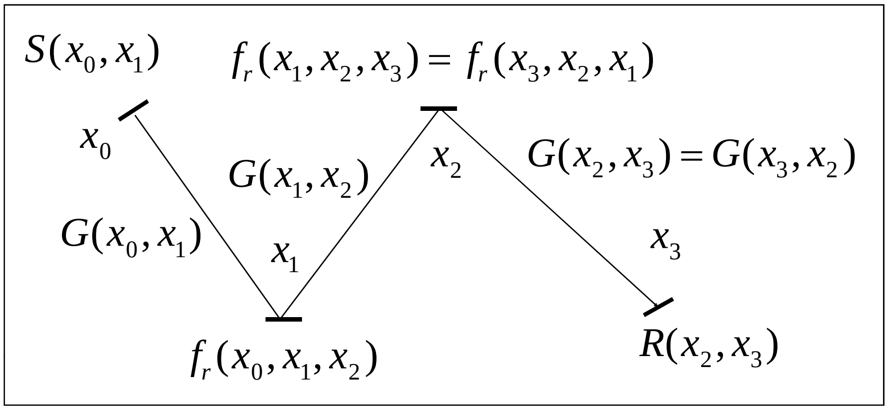
Adjoint Formulation

Symmetric Light Path



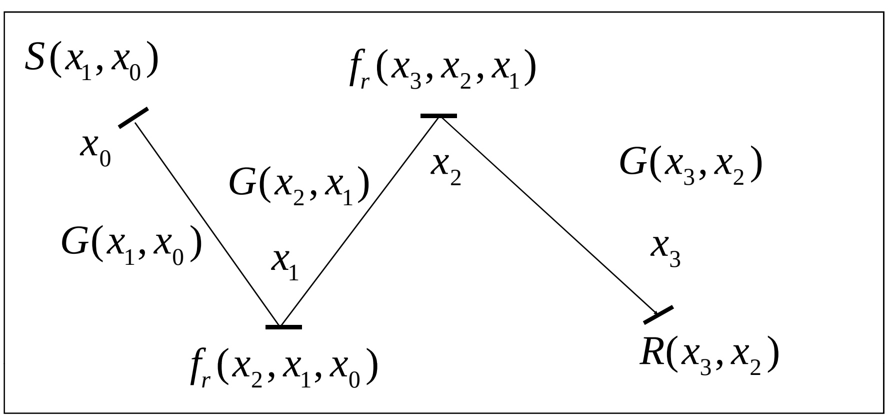
$$M = S(x_0, x_1) G(x_o, x_1) f_r(x_0, x_1, x_2) G(x_1, x_2) f_r(x_1, x_2, x_3) G(x_2, x_3) R(x_2, x_3)$$

Symmetric Light Path



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

Symmetric Light Path



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

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]