Monte Carlo Path Tracing

Today

- Path tracing starting from the eye
- Path tracing starting from the lights
- Which direction is best?
- Bidirectional ray tracing
- Random walks and Markov chains

Next

- Irradiance caching
- Photon mapping

The Rendering Equation

$$L(x,\omega) = L_e(x,\omega) + \int_{H^2} f_r(x,\omega' \to \omega)\, L(x^*(x,\omega'),-\omega') \cos\theta' \, d\omega'$$

$$L = L_e + K \circ L$$

Solving the Rendering Equation

Rendering Equation

$$L = L_e + K \circ L$$

$$(I - K) \circ L = L_e$$

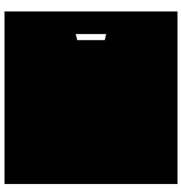
Solution

$$L = (I - K)^{-1} \circ L_e$$

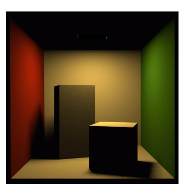
$$= (I + K + K^2 + K^3 + \cdots) \circ L_e$$

$$= (I + K(I + K(I + K \cdots))) \circ L_e$$

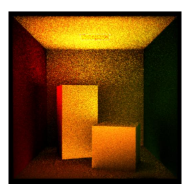
Successive Gathers



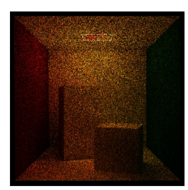
L_e



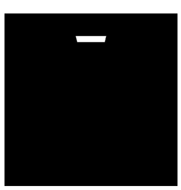
$K \circ L_e$



$K \circ K \circ L_e$

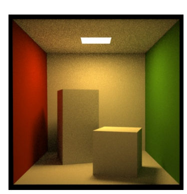


$K \circ K \circ K \circ L_e$

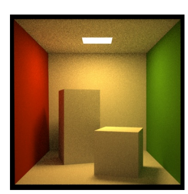


L_e

$L_e + K \circ L_e$

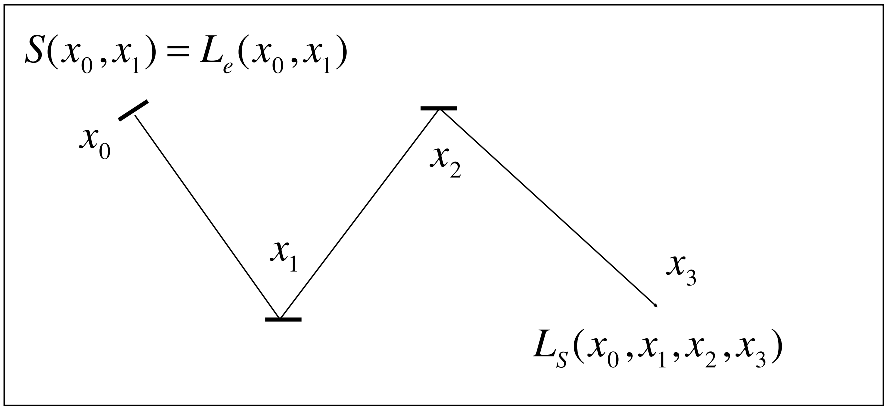


$L_e + \cdots K^2 \circ L_e$

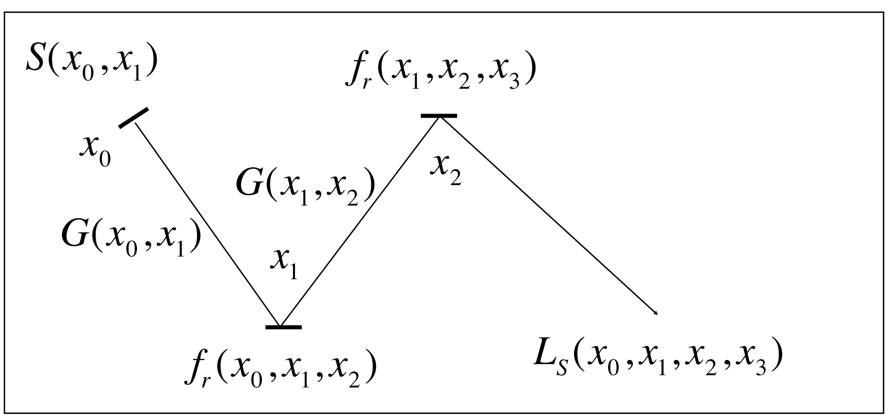


$L_e + \cdots K^3 \circ L_e$

Light Path



Light Path



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

Solving the Rendering Equation

One path

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

Solution is the integral over all paths

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

Solve using Monte Carlo Integration

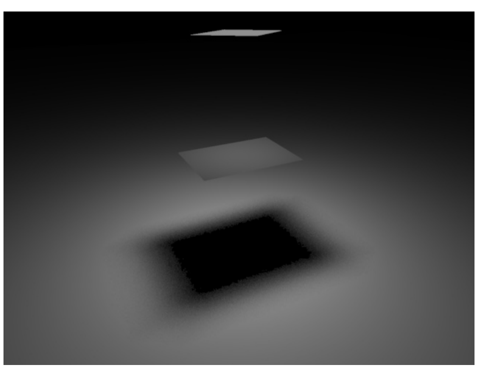
Question: How to generate a random path?

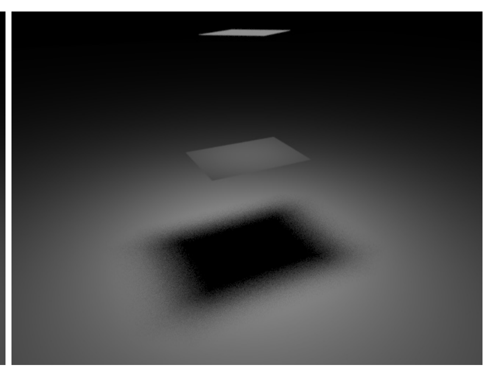
Path Tracing from the Eye

Path Tracing: From Camera

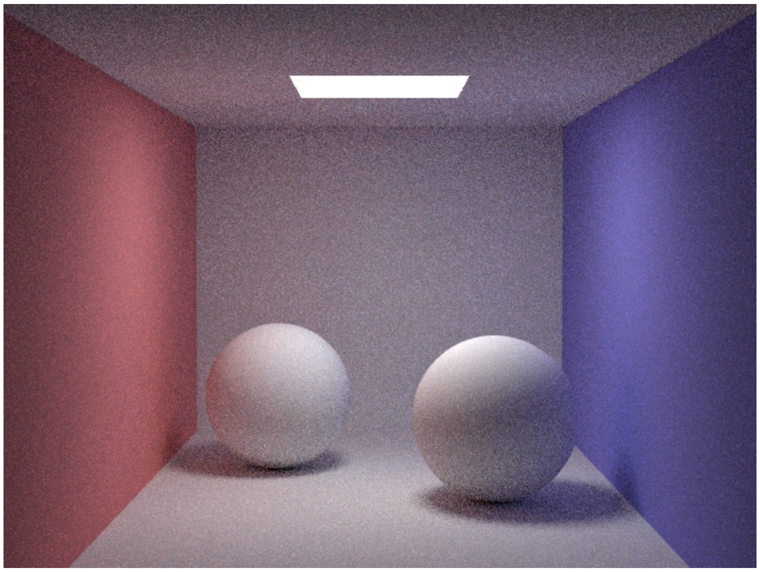
```
Step 1. Choose a camera ray r given the
   (x, y, u, v, t ) sample
      weight = 1;
Step 2. Find ray-surface intersection
Step 3.
   if hit light
      return weight * Le(r);
   else
      weight *= reflectance(r)
      Choose new ray r' ~ BRDF(O|I)
      Go to Step 2.
```


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

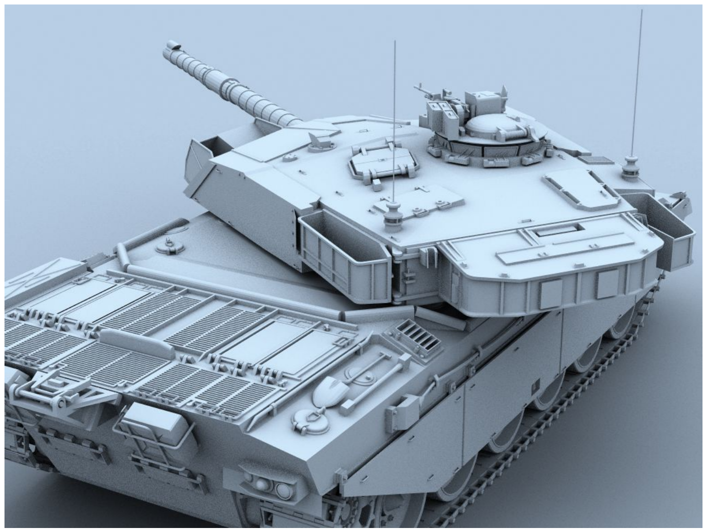
Path Tracing



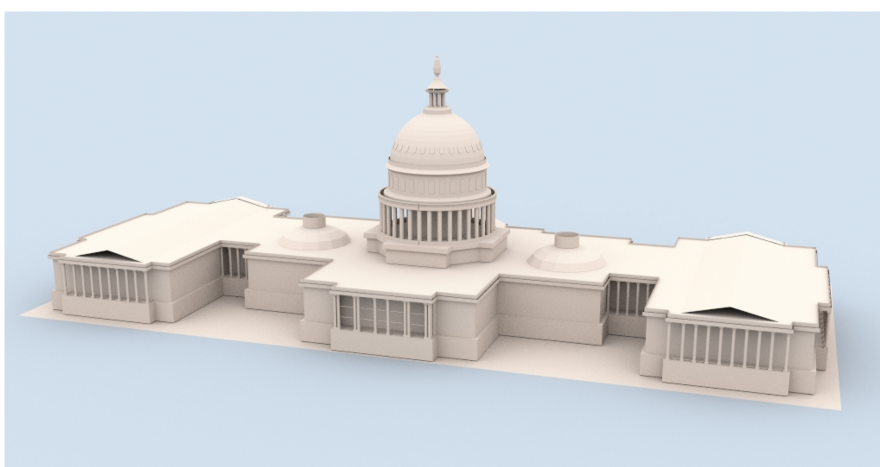
10 paths / pixel

M. Fajardo Arnold Path Tracer

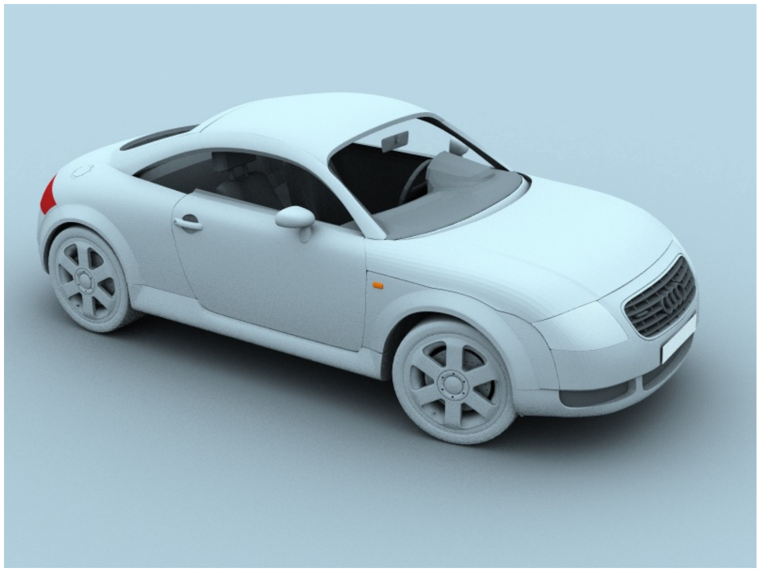
M. Fajardo Arnold Path Tracer



M. Fajardo Arnold Path Tracer

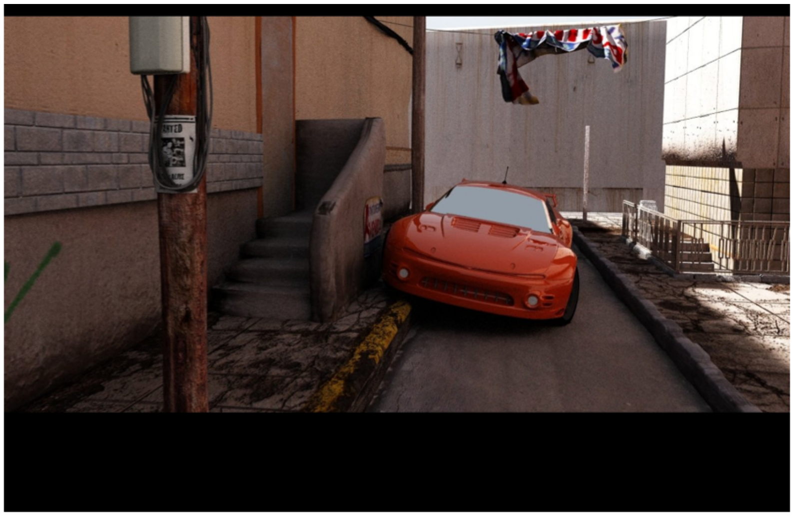


M. Fajardo Arnold Path Tracer



M. Fajardo Arnold Path Tracer

M. Fajardo Arnold Path Tracer



Street scene 1

1536x654, 16 paths/pixel, 2 bounces, 250,000 faces, 18 min., dual PIII-800

How Many Bounces?

Avoid paths that carry little energy

- Terminate when the weight is low

Photons with similar power is a good thing

- Think of importance sampling
- Integrand is f(x)/p(x) which is constant

Russian Roulette

Terminate photon with probability *p*

Adjust weight of the result by 1/(1-p)

$$E(X) = p \cdot 0 + (1-p)\frac{E(X)}{1-p} = E(X)$$

Intuition:

Reflecting from a surface with R=.5

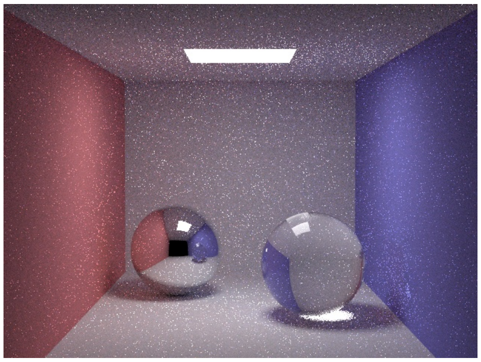
100 incoming photons with power 2 W

1. **Reflect 100 photons with power 1 W**
2. **Reflect 50 photons with power 2 W**

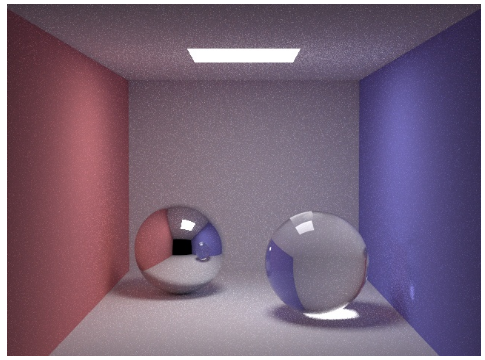
Path Tracing: Include Direct Lighting

```
Step 1. Choose a camera ray r given the
   (x,y,u,v,t) sample
     weight = 1;
     L = 0
Step 2. Find ray-surface intersection
Step 3.
     L += weight * Sum (fr * Le(light))
     weight *= reflectance(x)
     Choose new ray r' ~ BRDF pdf(r)
     Go to Step 2.
```

Variance Decreases with N



10 rays per pixel



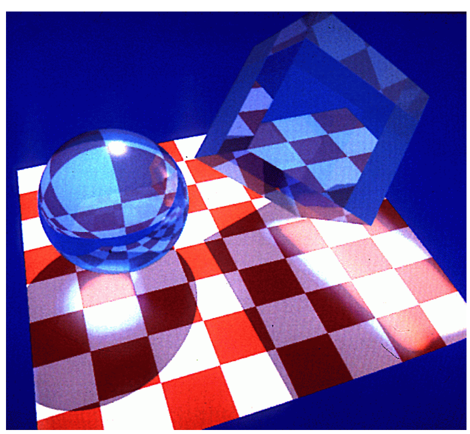
100 rays per pixel

From Jensen, Realistic Image Synthesis Using Photon Maps

Fixed Sampling (Not Random Enough)

Light Ray Tracing

Early Example [Arvo, 1986]



"Backward" ray tracing

Path Tracing: From Lights

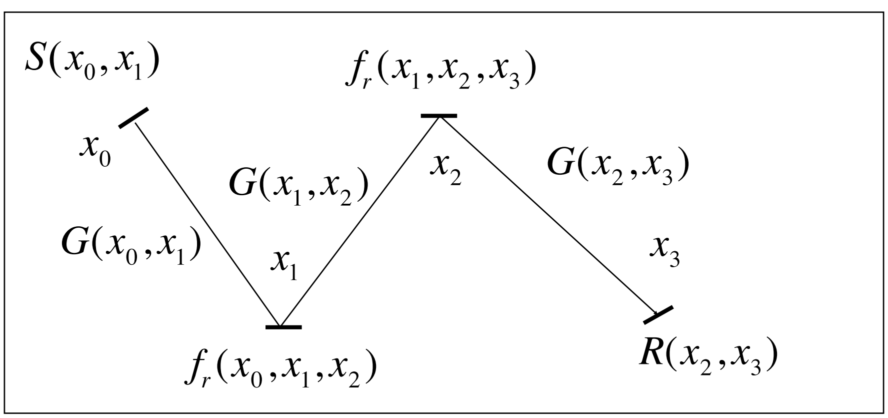
```
Step 1. Choose a light ray.
   Choose a ray from the light source
   distribution function
   x ~ p(x)
   d ~ p(d|x)
   r = (x, d)
   weight = Φ;
```


Path Tracing: From Lights

```
Step 1. Choose a light ray
Step 2. Find ray-surface intersection
Step 3. Reflect or transmit
   u = Uniform()
   if u < reflectance(x)
      Choose new direction d ~ BRDF(O|I)
      goto Step 2
   else u < reflectance(x)+transmittance(x)
      Choose new direction d ~ BTDF(O|I)
      goto Step 2
   else // absorption=1-reflectance-transmittance
      terminate on surface; deposit energy
```

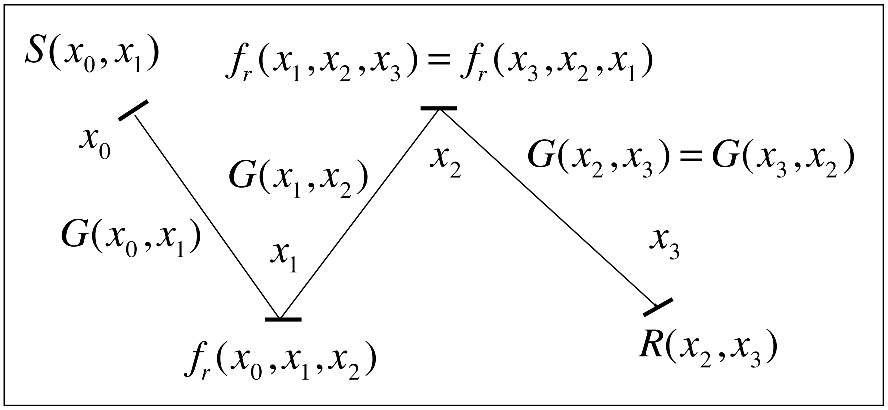
Bidirectional Path Tracing

Symmetric Light Path



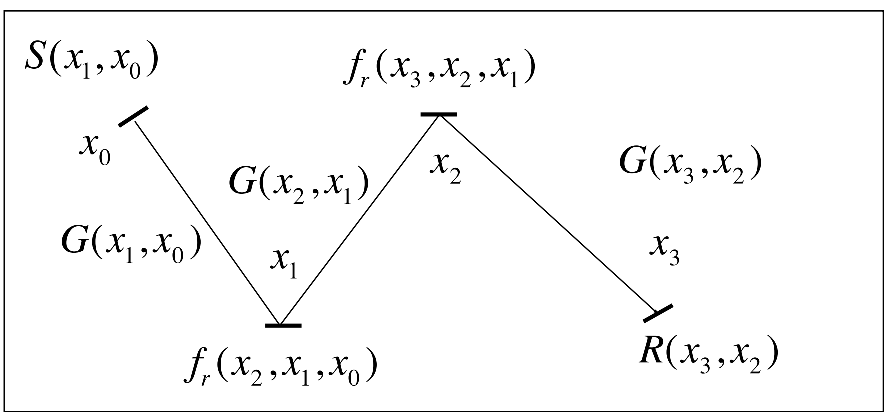
$$M = S(x_0, x_1) G(x_o, x_1) f_r(x_0, x_1, x_2) G(x_1, x_2) f_r(x_1, x_2, x_3) G(x_2, x_3) R(x_2, x_3)$$

Symmetric Light Path



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

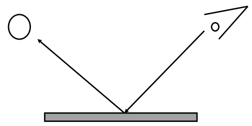
Symmetric Light Path



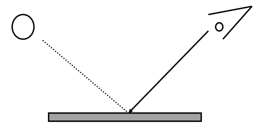
$$M = R(x_3,x_2)G(x_3,x_2)f_r(x_3,x_2,x_1)G(x_2,x_1)f_r(x_2,x_1,x_0)G(x_1,x_0)S(x_1,x_0)$$

Bidirectional Ray Tracing

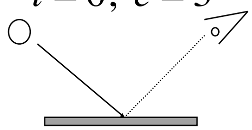
$$k = l + e$$



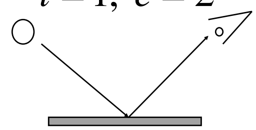
$l = 0,\ e = 3$



$l = 1,\ e = 2$



$l = 2,\ e = 1$



$l = 3,\ e = 0$

$$k = 3$$

Path Pyramid

$k = 3$

$k = 4$

$k = 5$

$k = 6$

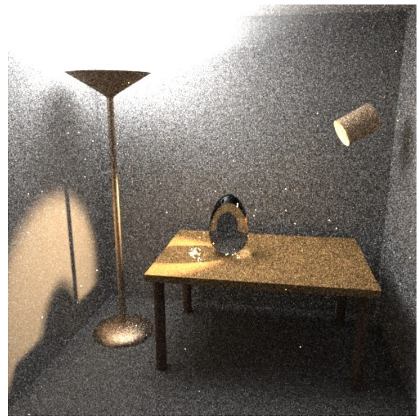
$(l = 2, e = 1)$

$(l = 5, e = 1)$

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

Which Direction?

Solve a linear system $Mx = b$

Solve for a single x_i**?**

Solve the reverse equation

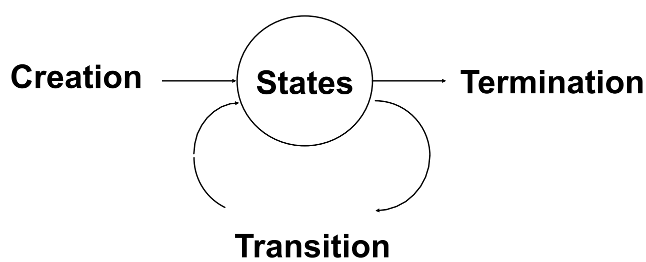
- **Source** x_i
- **Estimator** $<(x_i + Mx_i + M^2 x_i + \cdots), b>$

More efficient than solving for all the unknowns

[von Neumann and Ulam]

Discrete Random Walk

Discrete Random Process



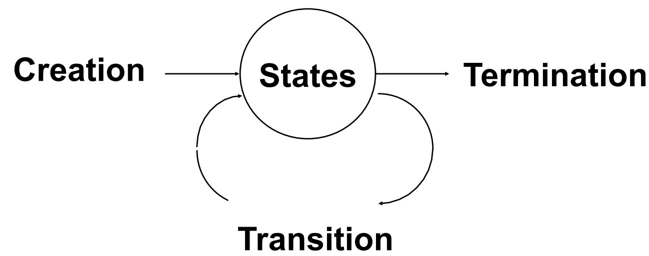
Assign probabilities to each process

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

Discrete Random Process



Equilibrium number of particles in each state

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

$$P = MP + p^0$$

Equilibrium Distribution of States

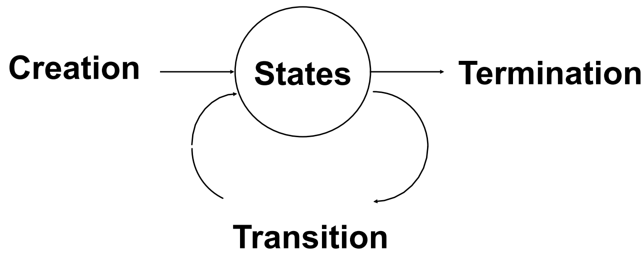
Total probability of being in states P

Solve this equation

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

$$P = (I - M)^{-1} p^0$$
$$= \left(I + M + M^2 + \cdots\right) p^0$$

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*

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

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