Monte Carlo Integration

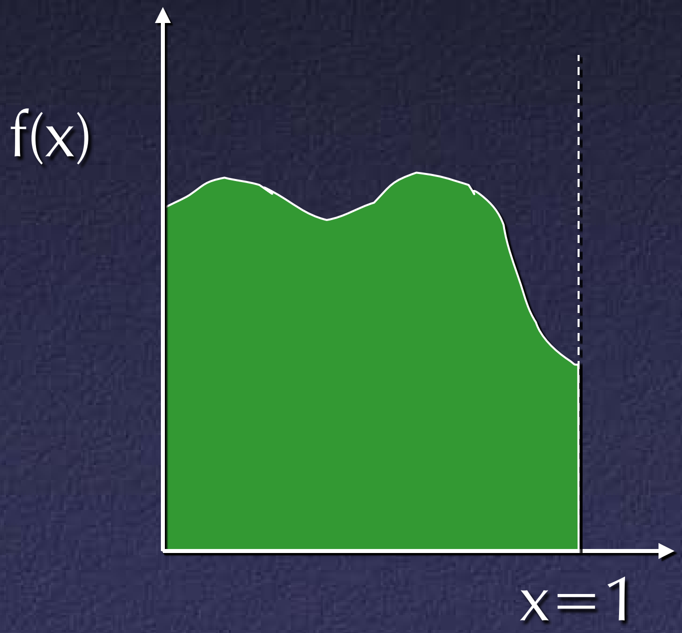
COS 323

Integration in *d* Dimensions?

- Trapezoidal rule in d dimensions?
  - In 1D: 2 points
  - In 2D: 4 points (corners of a square)
  - In general: 2^d points
- Exponential growth in # of points for a fixed order of method
  - "Curse of dimensionality"
- Other problems, e.g. non-rectangular domains

Rethinking Integration in 1D

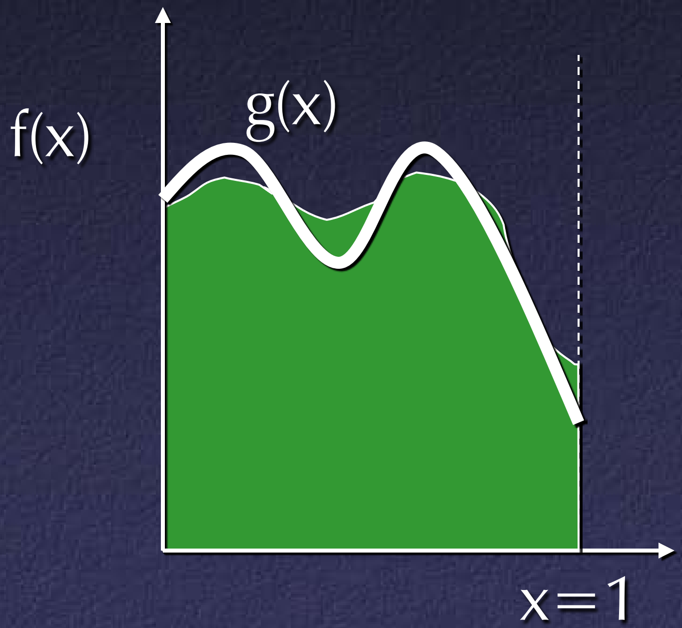
$$\int_0^1 f(x)\,dx = ?$$



Slide courtesy of
Peter Shirley

We Can Approximate…

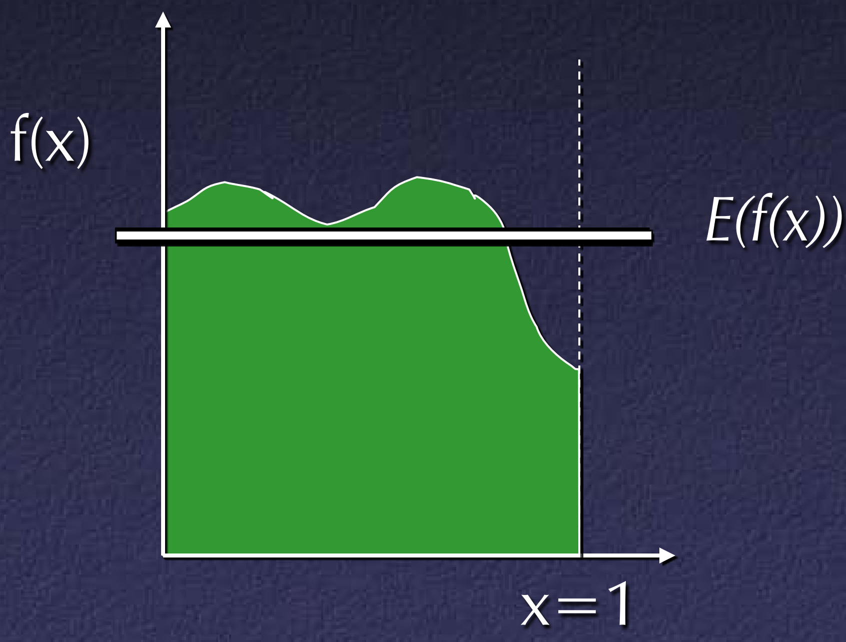
$$\int_0^1 f(x)dx = \int_0^1 g(x)dx$$



Slide courtesy of
Peter Shirley

Or We Can Average

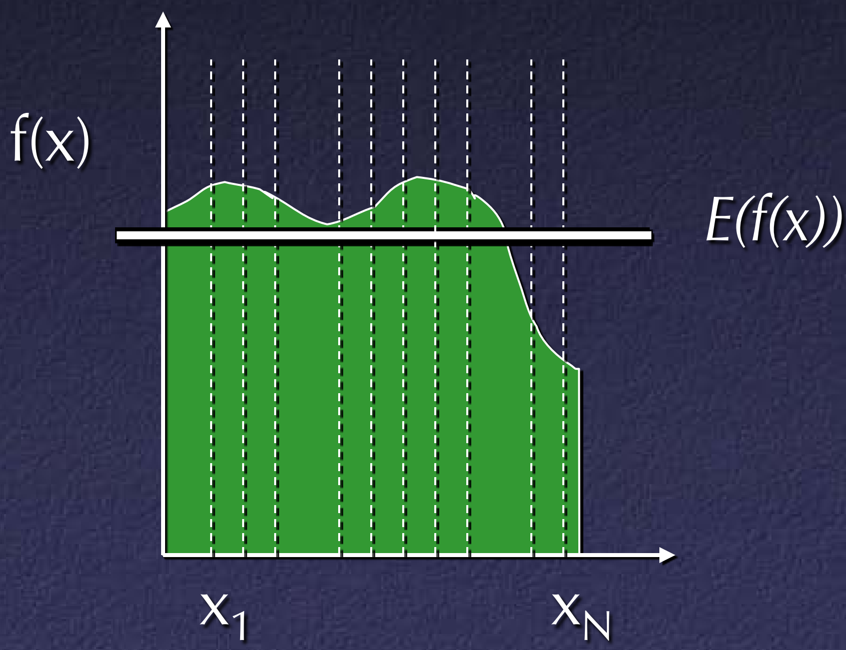
$$\int_0^1 f(x)dx = E(f(x))$$



Slide courtesy of
Peter Shirley

Estimating the Average

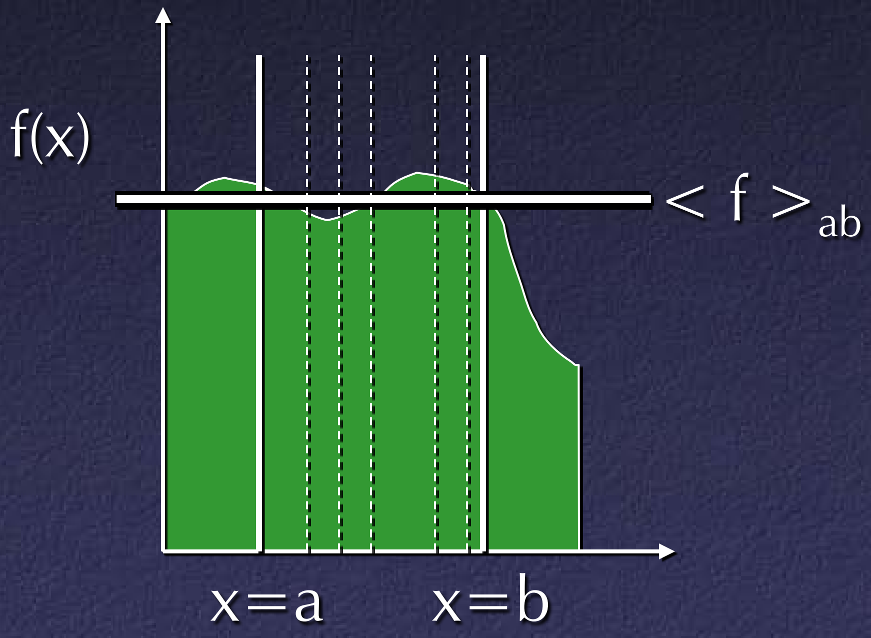
$$\int_0^1 f(x)dx = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$



Slide courtesy of
Peter Shirley

Other Domains

$$\int_a^b f(x)dx = \frac{b-a}{N}\sum_{i=1}^{N} f(x_i)$$



Slide courtesy of
Peter Shirley

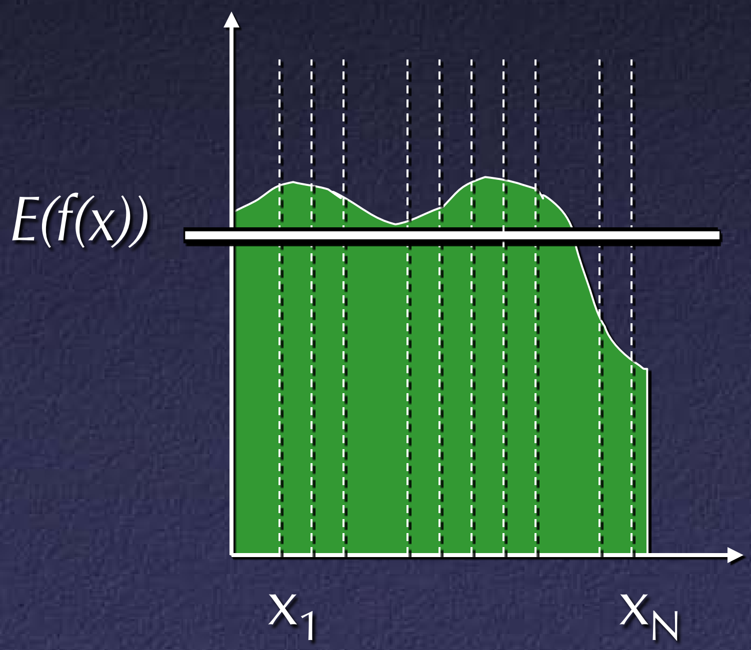
"Monte Carlo" Integration

- No "exponential explosion" in required number of samples with increase in dimension
- (Some) resistance to badly-behaved functions

Le Grand Casino de Monte-Carlo

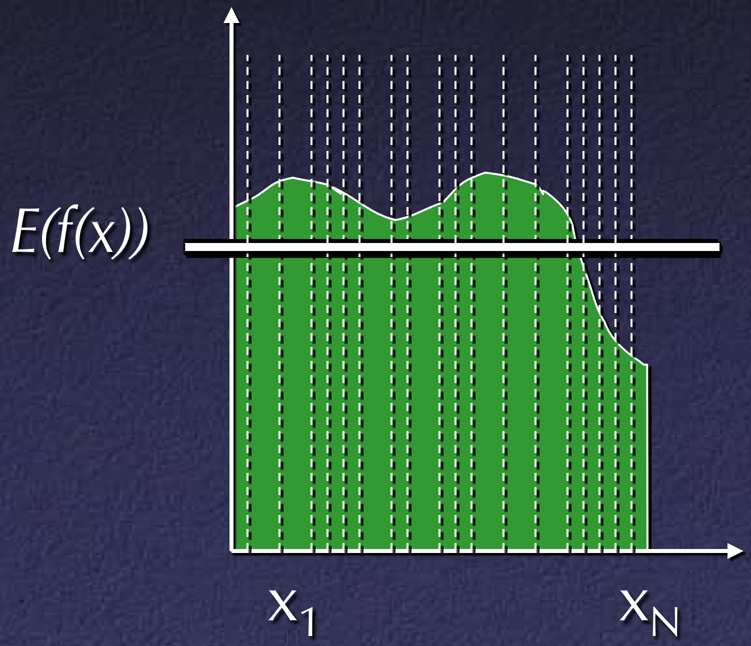
Variance

$$Var\left[f(x)\right] = \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - E(f(x))]^2$$



Variance

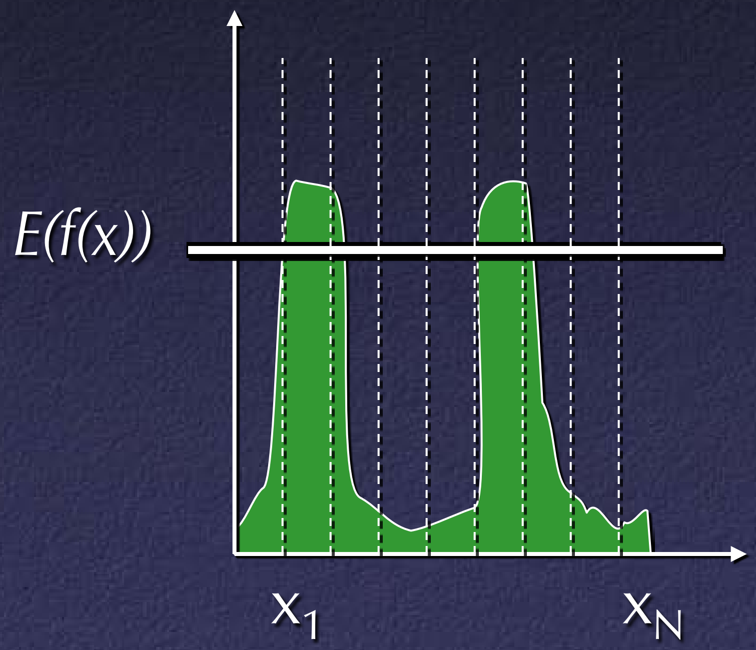
$$Var\left[E(f(x))\right] = \frac{1}{N} Var\left[f(x)\right]$$



Variance decreases as 1/N
Error decreases as 1/sqrt(N)

Variance

- Problem: variance decreases with 1/N
  - Increasing # samples removes noise slowly

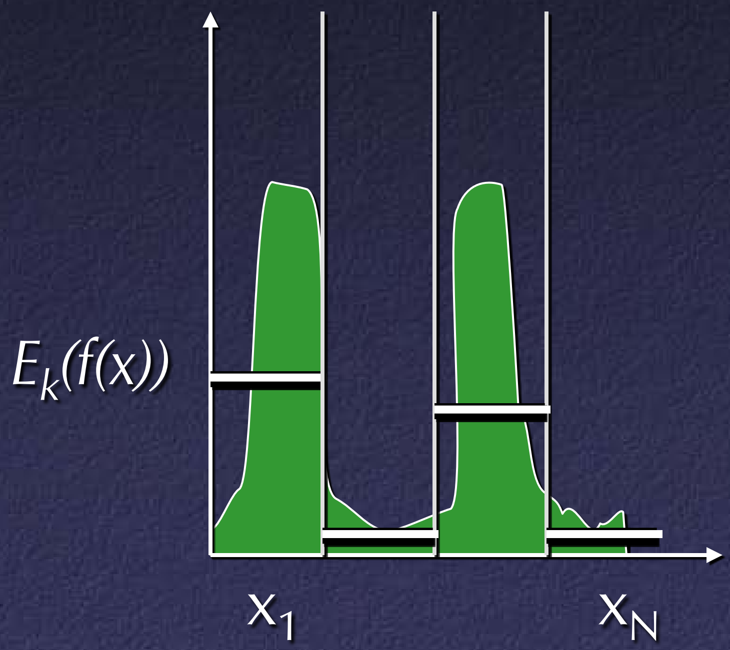


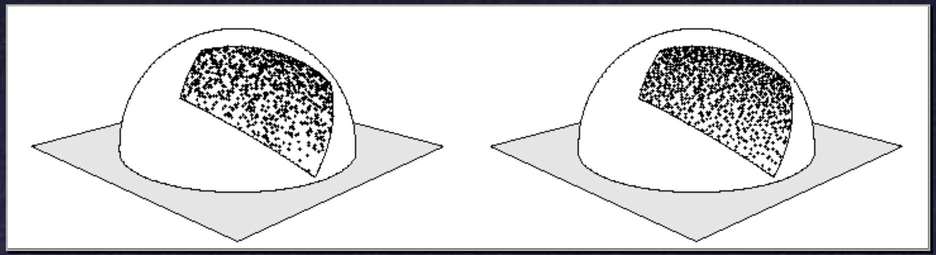
Variance Reduction Techniques

- Problem: variance decreases with 1/N
  - Increasing # samples removes noise slowly
- Variance reduction:
  - Stratified sampling
  - Importance sampling

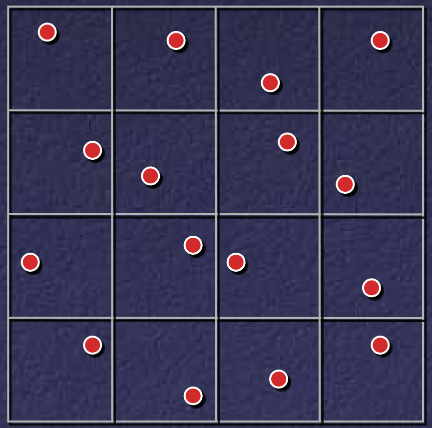
Stratified Sampling

- Estimate subdomains separately





Arvo

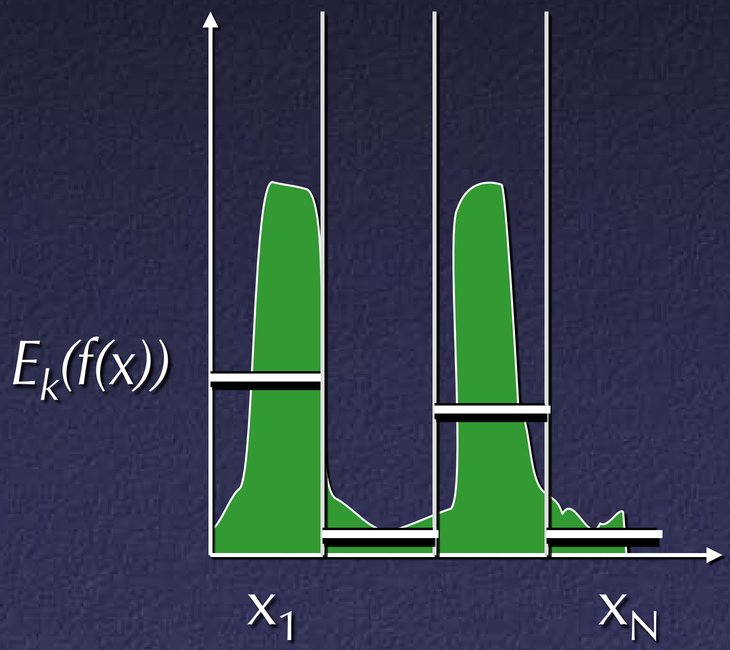


Stratified Sampling

- This is still unbiased

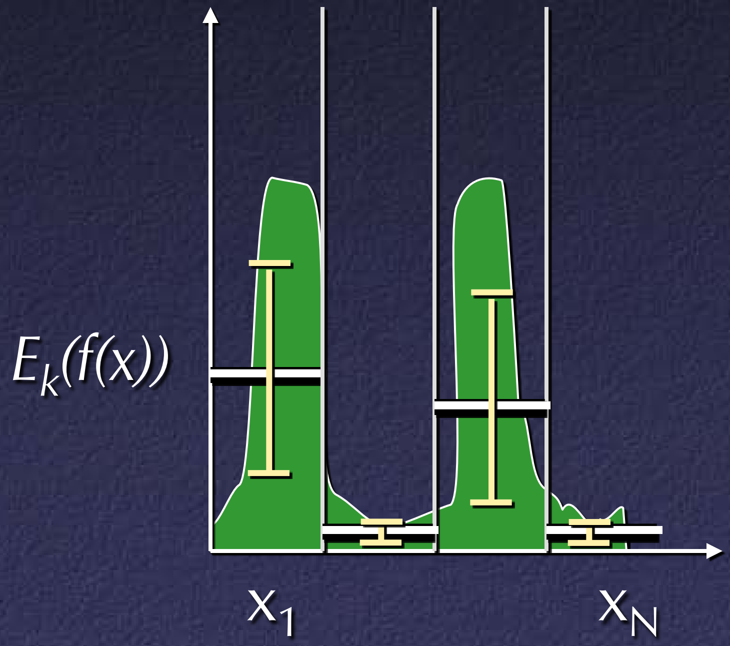
$$F_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

$$= \frac{1}{N}\sum_{k=1}^{M} N_i F_i$$



Stratified Sampling

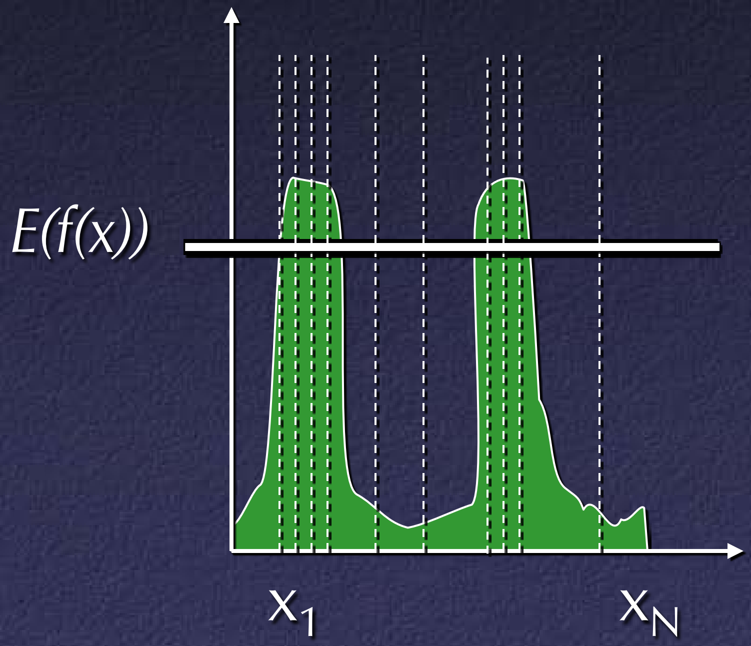
- Less overall variance if less variance in subdomains



$$Var[F_N] = \frac{1}{N^2} \sum_{k=1}^{M} N_i Var[F_i]$$

Importance Sampling

- Put more samples where f(x) is bigger

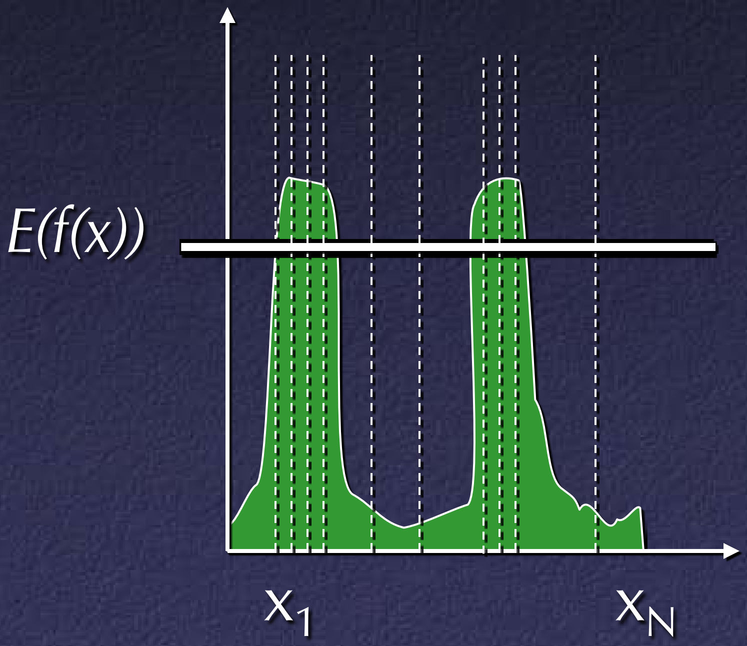


$$\int_{\Omega} f(x)dx = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

$$Y_i = \frac{f(x_i)}{p(x_i)}$$

Importance Sampling

- This is still unbiased



$$E[Y_i] = \int_\Omega Y(x)\,p(x)\,dx$$

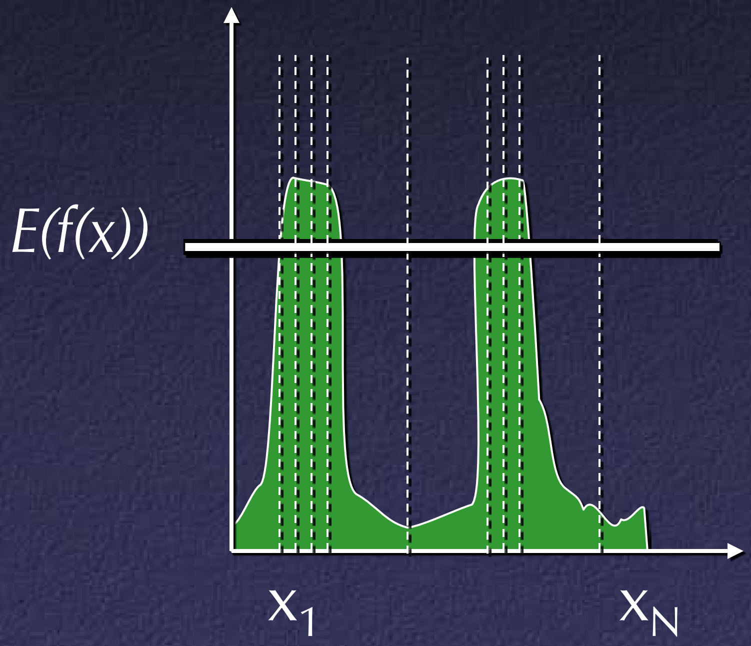
$$= \int_\Omega \frac{f(x)}{p(x)}\,p(x)\,dx$$

$$= \int_\Omega f(x)\,dx$$

for all N

Importance Sampling

- Zero variance if p(x) ~ f(x)



$$p(x) = cf(x)$$

$$Y_i = \frac{f(x_i)}{p(x_i)} = \frac{1}{c}$$

$$Var(Y) = 0$$

Less variance with better importance sampling

Generating Random Points

- Uniform distribution:
  - Use pseudorandom number generator

Pseudorandom Numbers

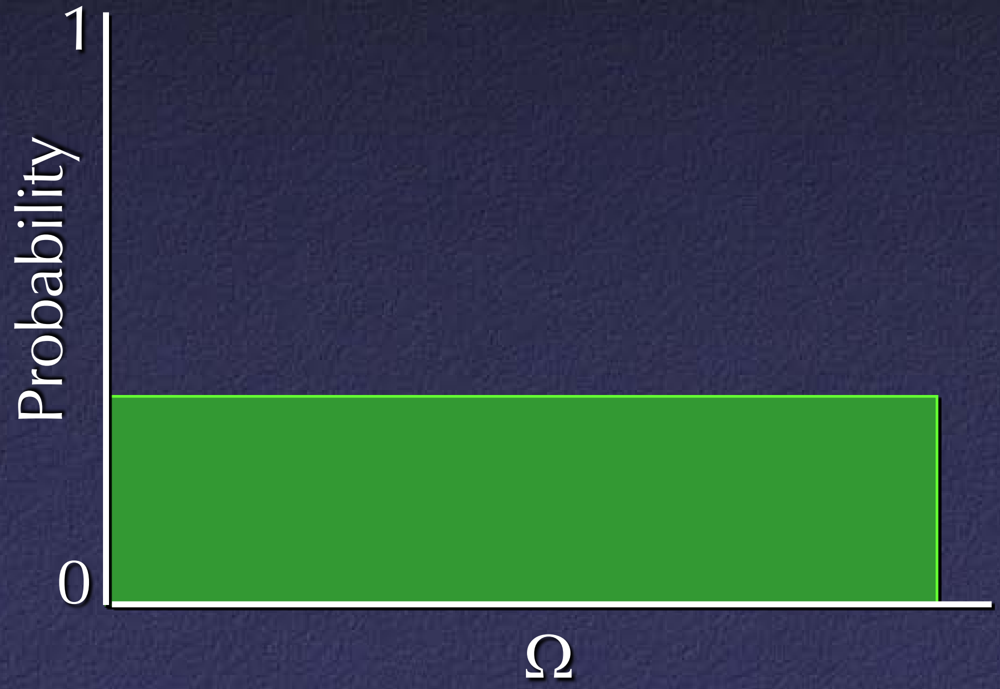
- Deterministic, but have statistical properties resembling true random numbers
- Common approach: each successive pseudorandom number is function of previous
- Linear congruential method: $x_{n+1} = (ax_n + b) \bmod c$
  - Choose constants carefully, e.g.
    a = 1664525
    b = 1013904223
    c = $2^{32} - 1$

Pseudorandom Numbers

- To get floating-point numbers in [0..1), divide integer numbers by $c + 1$
- To get integers in range [u..v], divide by (c+1)/(v–u+1), truncate, and add u
  - Better statistics than using modulo (v–u+1)
  - Only works if u and v small compared to c

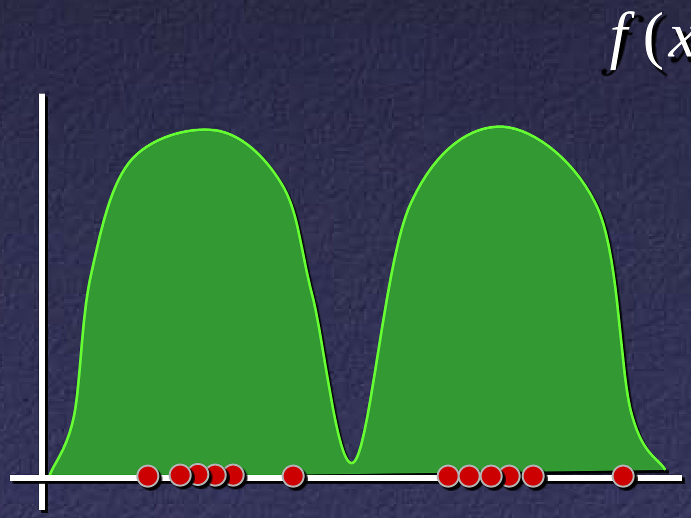
Generating Random Points

- Uniform distribution:
  - Use pseudorandom number generator



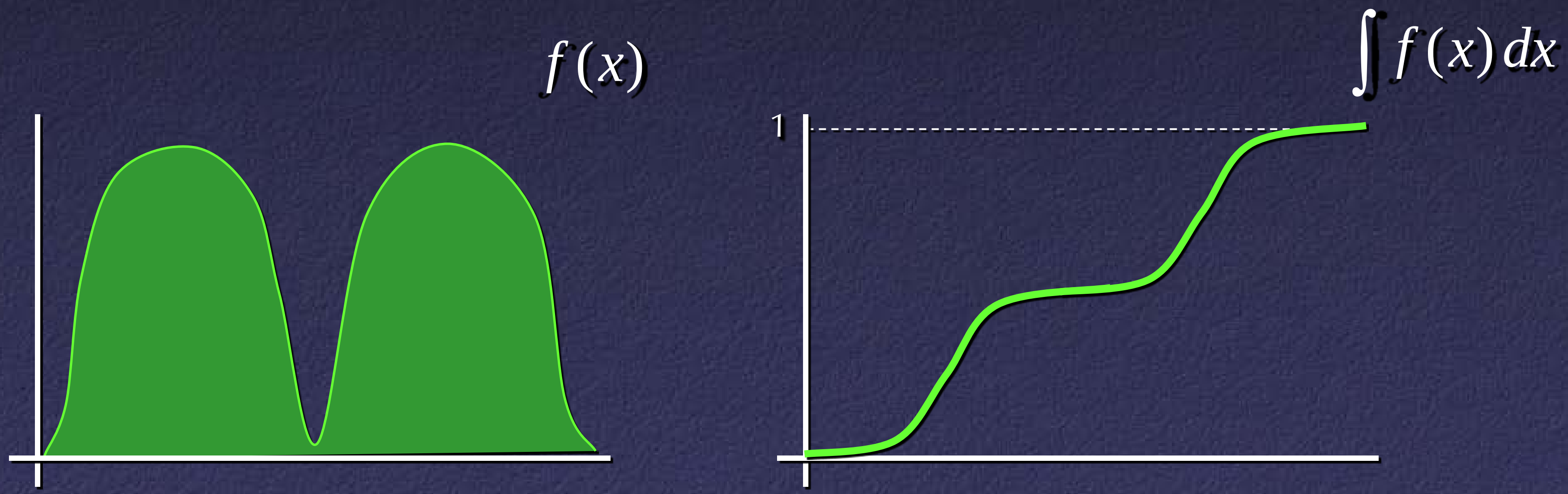
Importance Sampling

- Specific probability distribution:
  - Function inversion
  - Rejection



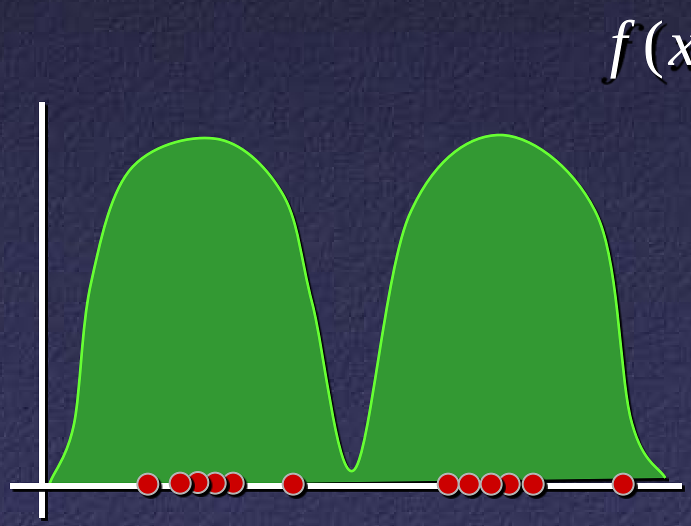
Importance Sampling

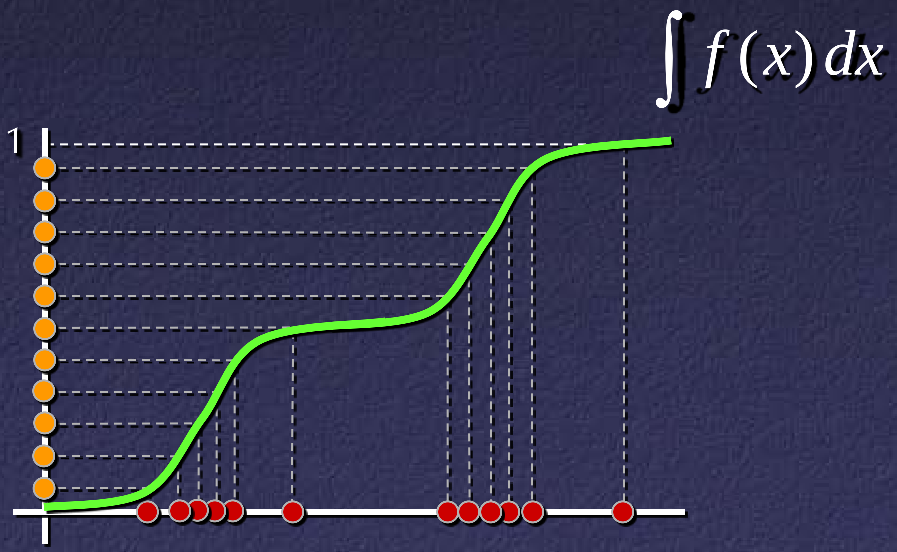
- "Inversion method"
  - Integrate *f*(*x*): Cumulative Distribution Function



Importance Sampling

- "Inversion method"
  - Integrate *f*(*x*): Cumulative Distribution Function
  - Invert CDF, apply to uniform random variable

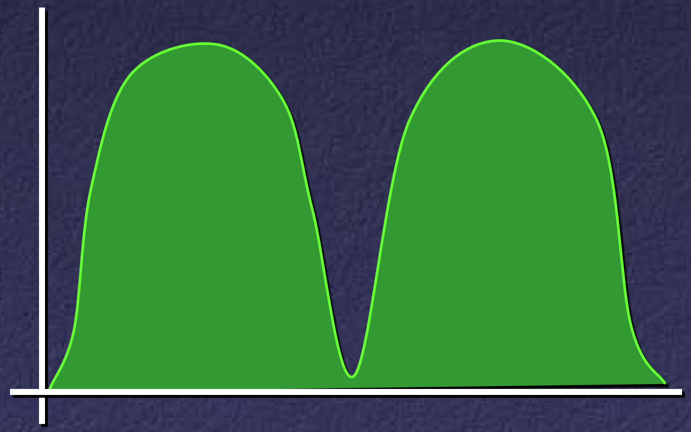




Importance Sampling

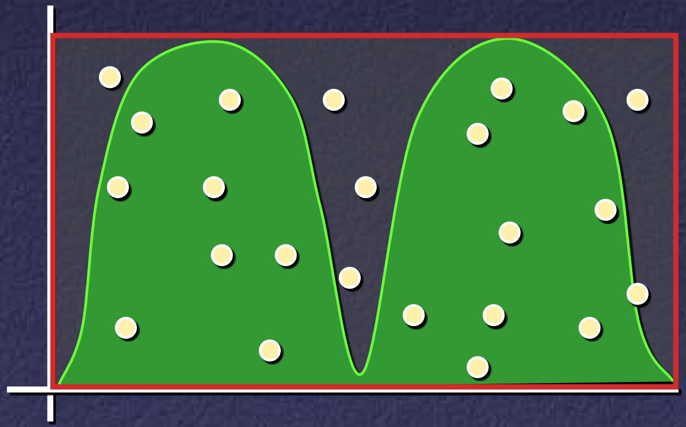
- Specific probability distribution:
  - Function inversion
  - Rejection

$f(x)$



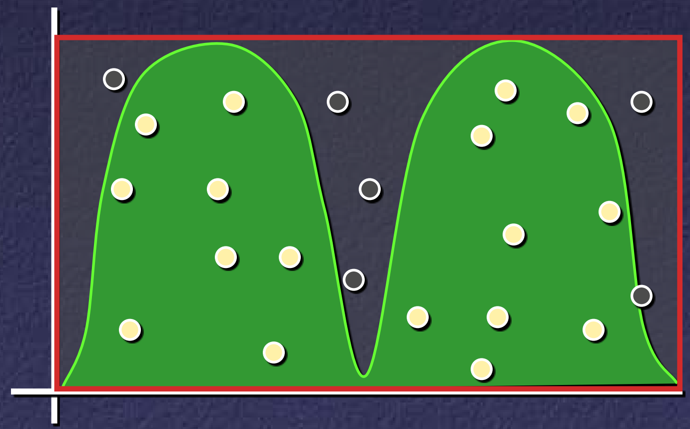
Generating Random Points

- "Rejection method"
  - Generate random (x,y) pairs, y between 0 and max(f(x))



Generating Random Points

- "Rejection method"
  - Generate random (x,y) pairs, y between 0 and max(f(x))
  - Keep only samples where y < f(x)

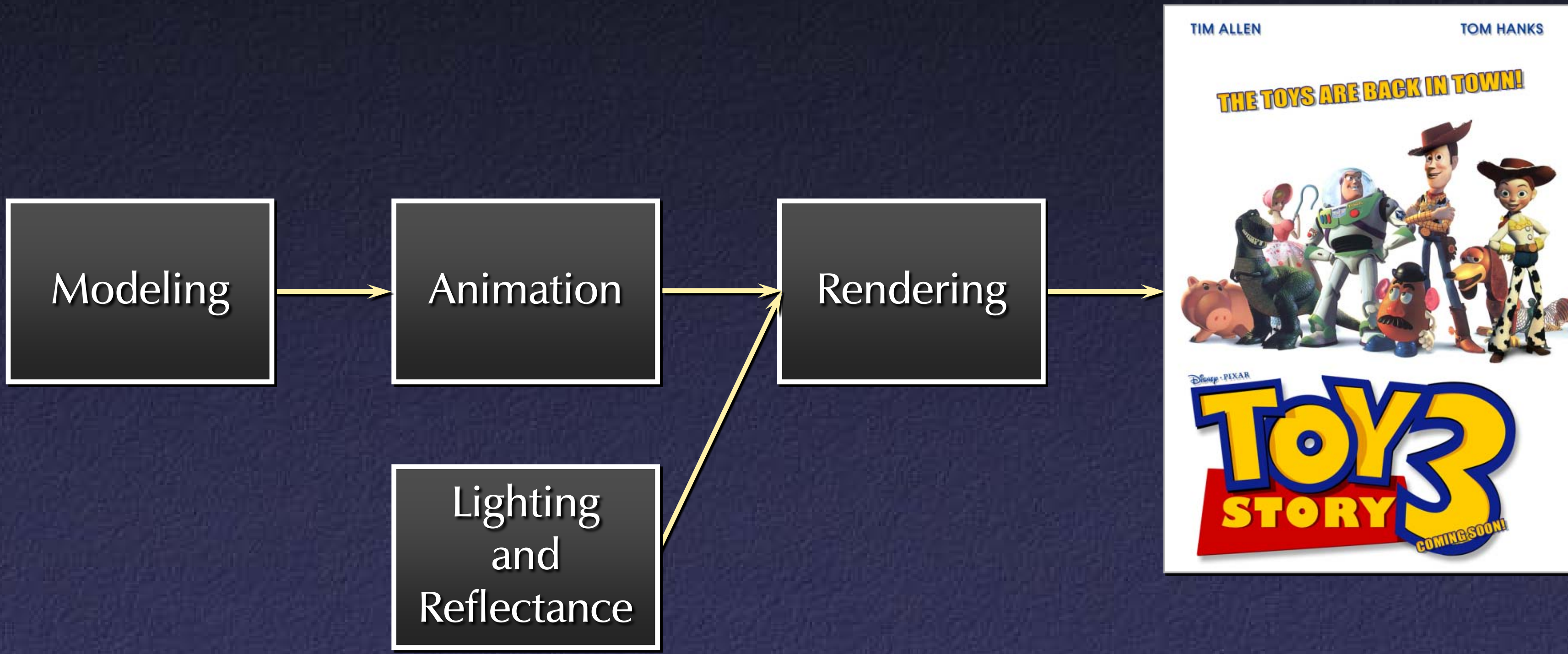


Monte Carlo in Computer Graphics

or, Solving Integral Equations for Fun and Profit

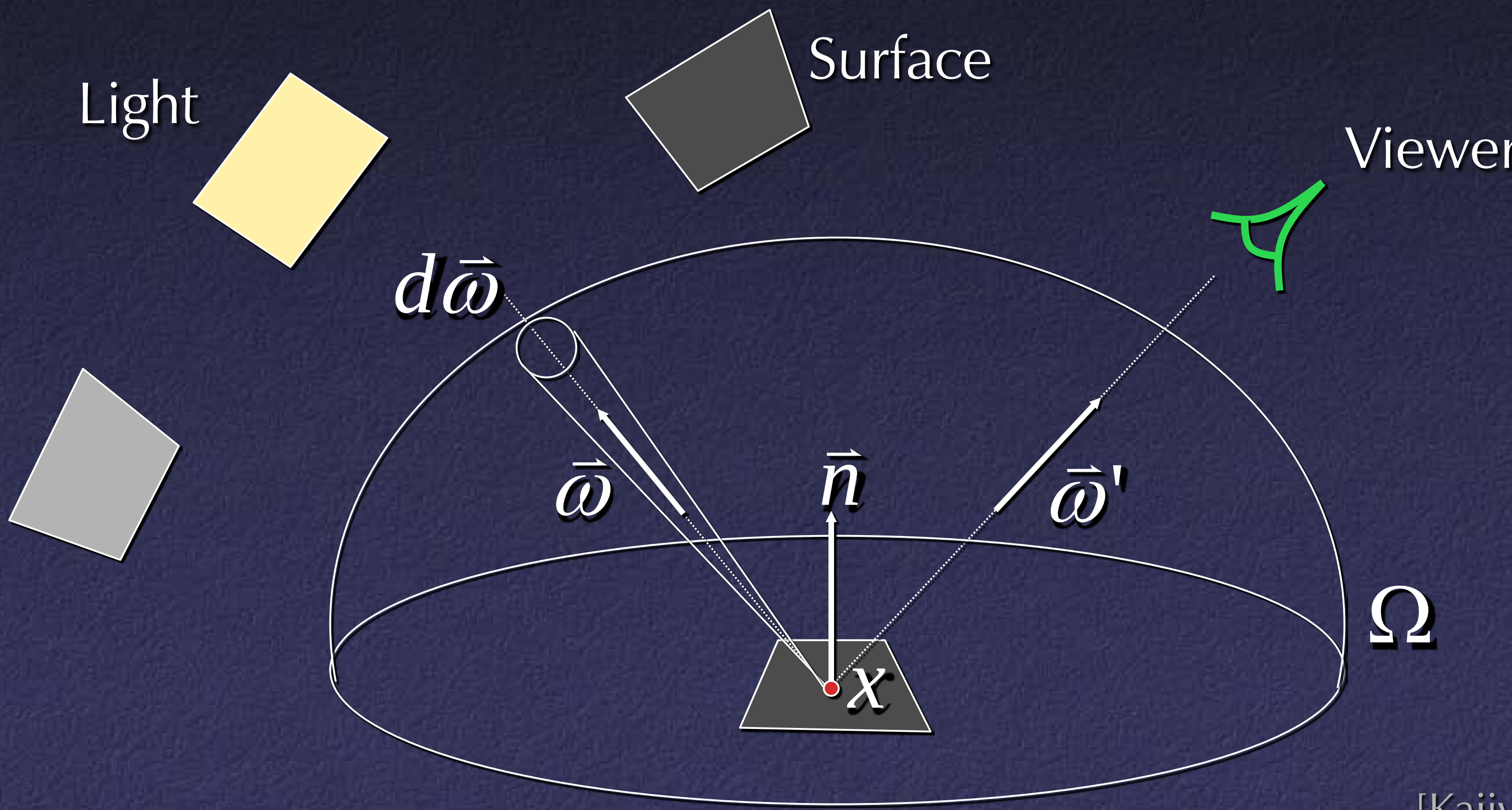
or, Ugly Equations, Pretty Pictures

Computer Graphics Pipeline



Rendering Equation

$$L_o(x,\vec{\omega}') = L_e(x,\vec{\omega}') + \int_{\Omega} L_i(x,\vec{\omega})\, f_r(x,\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{n})\, d\vec{\omega}$$



[Kajiya 1986]

Rendering Equation

$$L_o(x,\vec{\omega}') = L_e(x,\vec{\omega}') + \int_{\Omega} L_i(x,\vec{\omega})\, f_r(x,\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{n})\,d\vec{\omega}$$

- This is an *integral equation*
- Hard to solve!
  - Can't solve this in closed form
  - Simulate complex phenomena

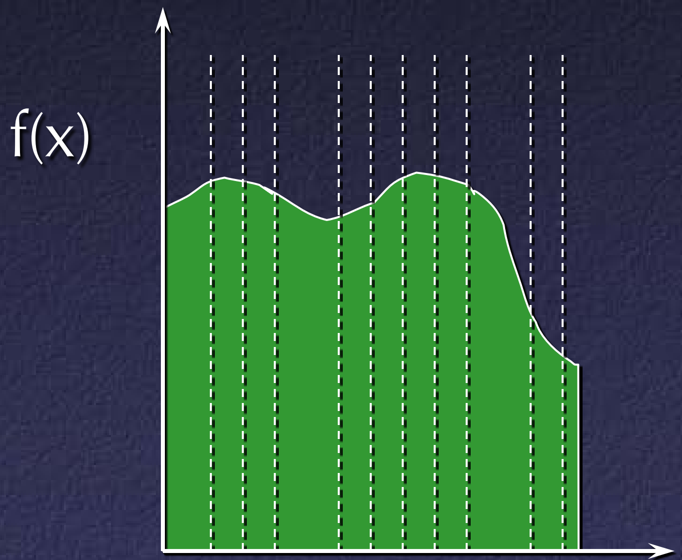
Heinrich

Rendering Equation

$$L_o(x,\vec{\omega}') = L_e(x,\vec{\omega}') + \int_{\Omega} L_i(x,\vec{\omega})\, f_r(x,\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{n})\,d\vec{\omega}$$

- This is an *integral equation*
- Hard to solve!
  - Can't solve this in closed form
  - Simulate complex phenomena

Monte Carlo Integration

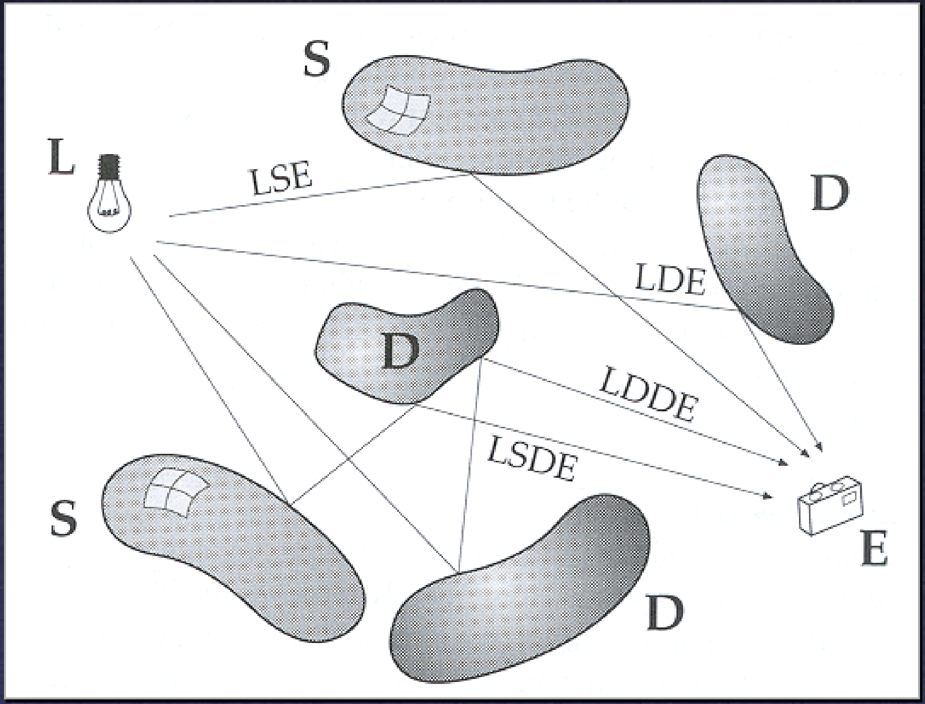


$$\int_0^1 f(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Shirley

Monte Carlo Path Tracing

Estimate integral
for each pixel
by random sampling

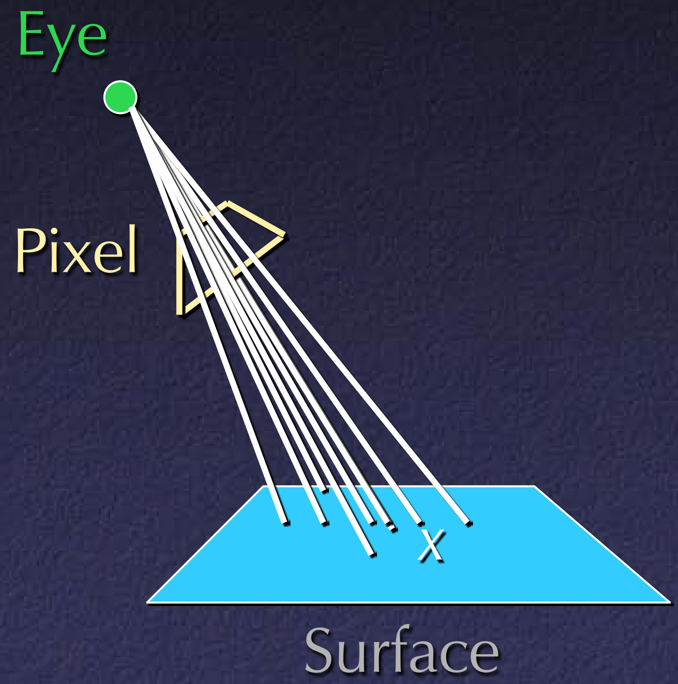


Monte Carlo Global Illumination

- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Monte Carlo Global Illumination

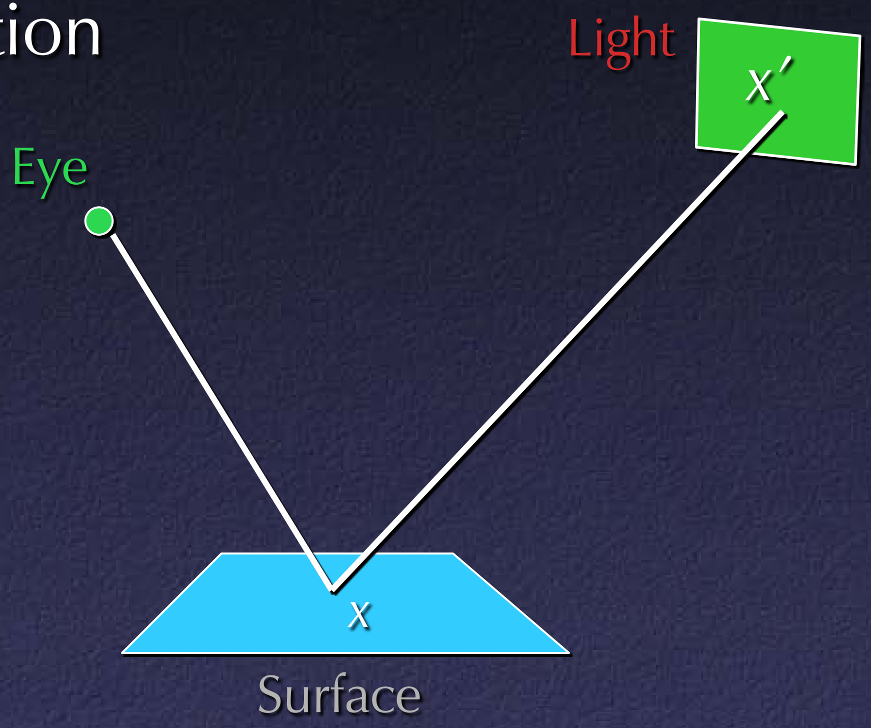
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



$$L_P = \int_S L(x \to e)\,dA$$

Monte Carlo Global Illumination

- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



$$L(x,\vec{w}) = L_e(x,x \to e) + \int_S f_r(x,x' \to x, x \to e) L(x' \to x) V(x,x') G(x,x') dA$$

Monte Carlo Global Illumination

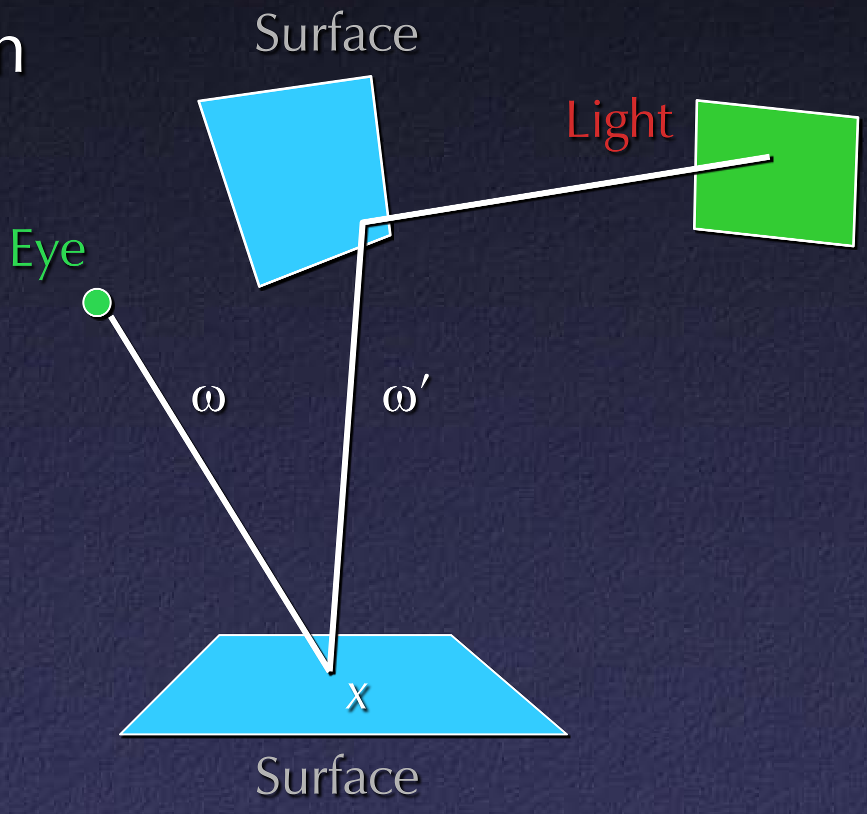
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Herf

$$L(x,\vec{w}) = L_e(x, x \rightarrow e) + \int_S f_r(x, x' \rightarrow x, x \rightarrow e) L(x' \rightarrow x) V(x,x') G(x,x') dA$$

Monte Carlo Global Illumination

- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_{\Omega} f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}' \bullet \vec{n}) d\vec{w}$$

Monte Carlo Global Illumination

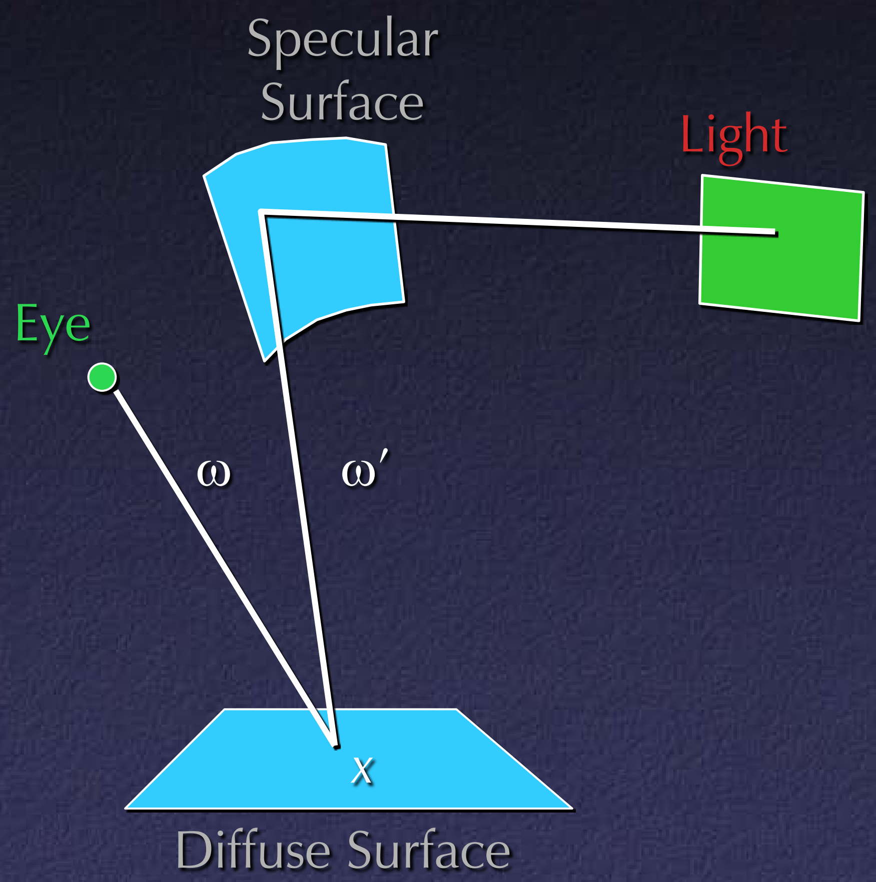
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Debevec

$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_\Omega f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}' \bullet \vec{n}) d\vec{w}$$

Monte Carlo Global Illumination

- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_{\Omega} f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}' \bullet \vec{n}) d\vec{w}$$

Monte Carlo Global Illumination

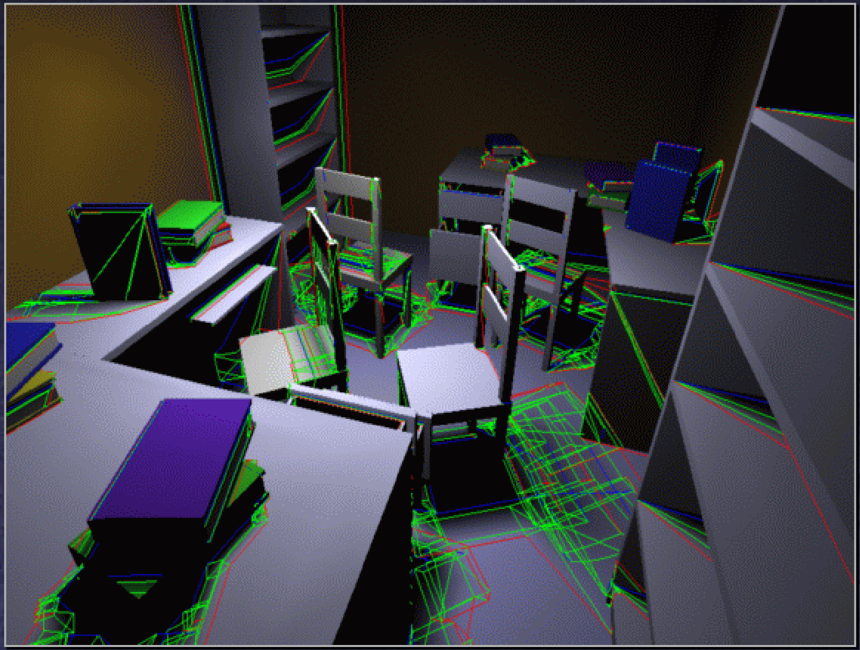
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Jensen

$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_{\Omega} f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}' \bullet \vec{n}) d\vec{w}$$

Challenge

- Rendering integrals are difficult to evaluate
  - Multiple dimensions
  - Discontinuities
    - Partial occluders
    - Highlights
    - Caustics



Drettakis

$$L(x,\vec{w}) = L_e(x,x \rightarrow e) + \int_S f_r(x,x' \rightarrow x, x \rightarrow e) L(x' \rightarrow x) V(x,x') G(x,x') dA$$

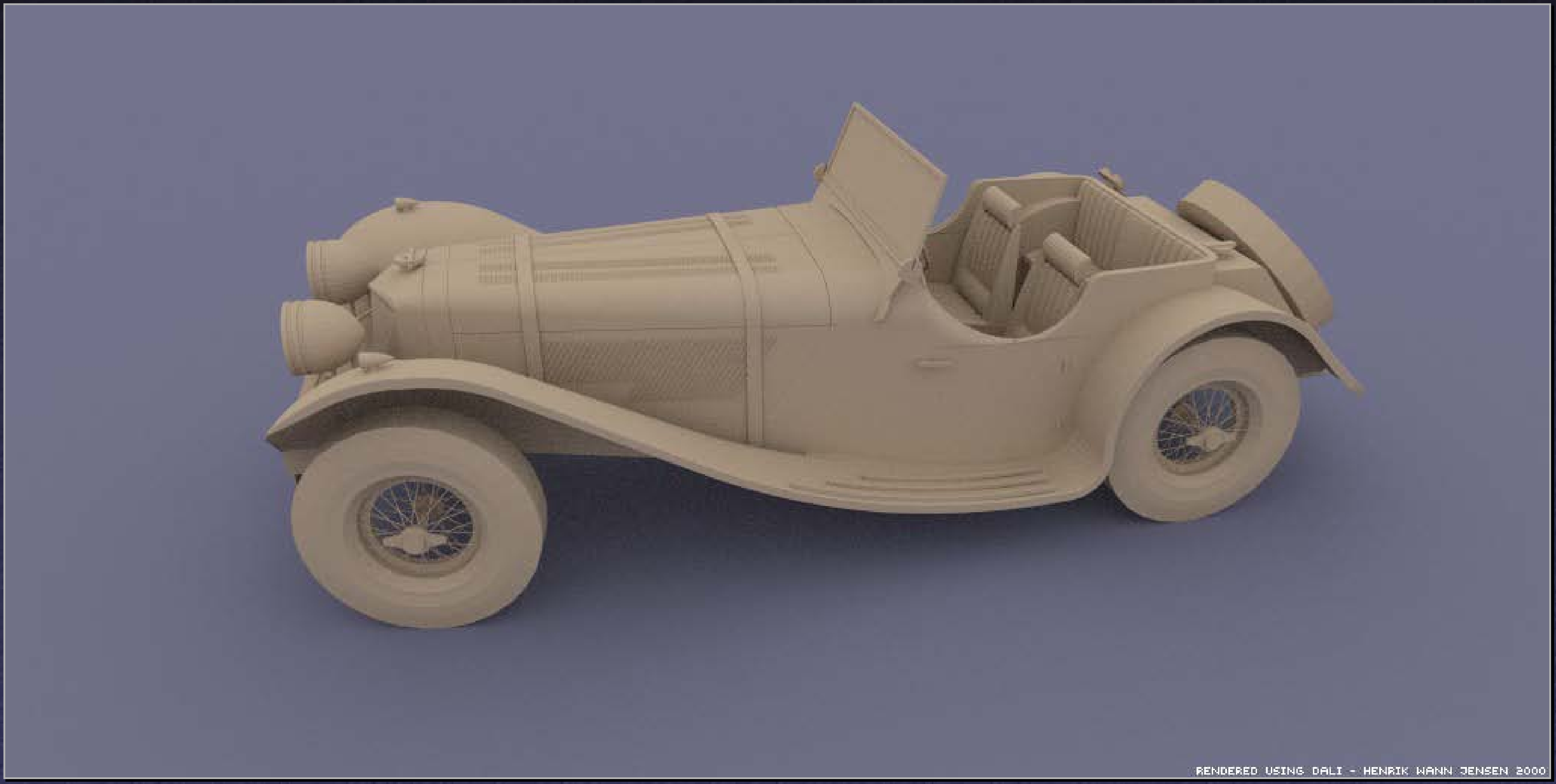
Challenge

- Rendering integrals are difficult to evaluate
  - Multiple dimensions
  - Discontinuities
    - Partial occluders
    - Highlights
    - Caustics

Jensen

$$L(x,\vec{w}) = L_e(x,x \rightarrow e) + \int_S f_r(x,x' \rightarrow x, x \rightarrow e) L(x' \rightarrow x) V(x,x') G(x,x') dA$$

Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

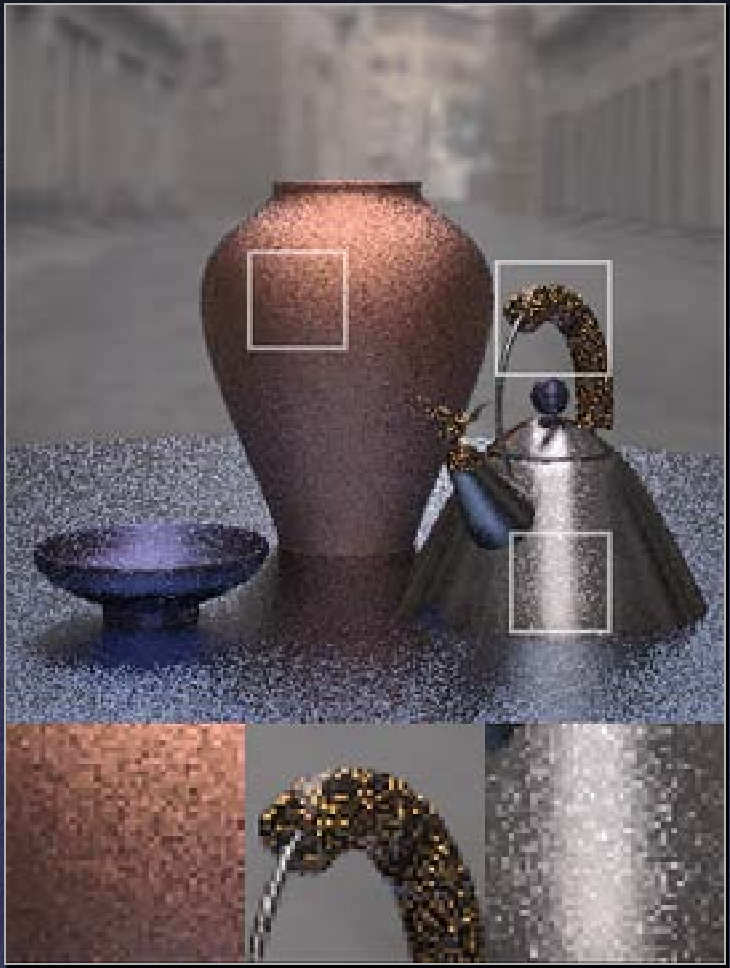
Monte Carlo Path Tracing

1000 paths/pixel

Jensen

Monte Carlo Path Tracing

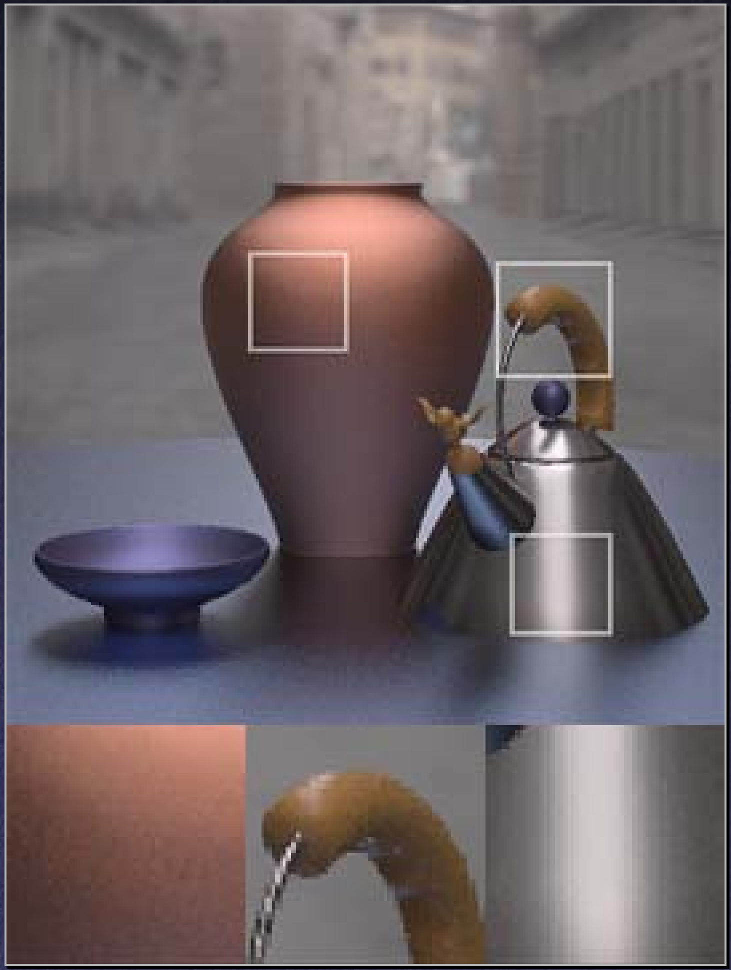
- Drawback: can be noisy unless *lots* of paths simulated
- 40 paths per pixel:



Lawrence

Monte Carlo Path Tracing

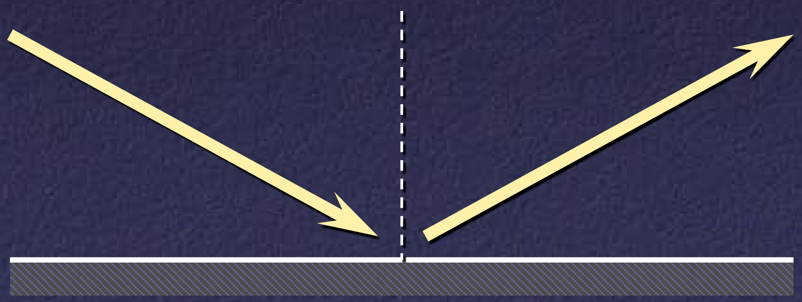
- Drawback: can be noisy unless *lots* of paths simulated
- 1200 paths per pixel:



Lawrence

Reducing Variance

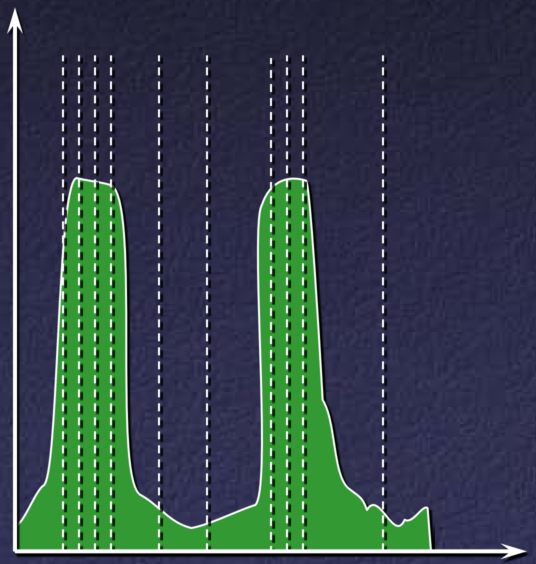
- Observation: some paths more important (carry more energy) than others
  - For example, shiny surfaces reflect more light in the ideal "mirror" direction



- Idea: put more samples where f(x) is bigger

Importance Sampling

- Idea: put more samples where f(x) is bigger

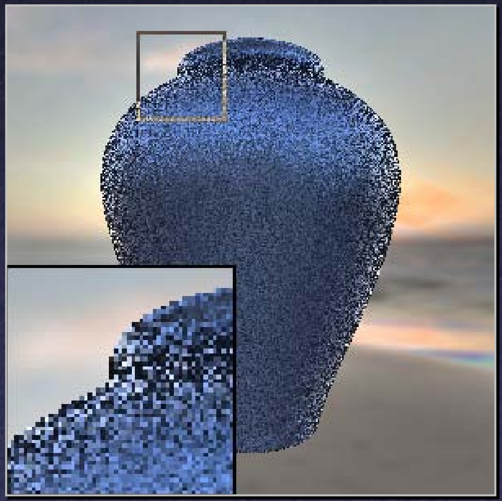


$$\int_0^1 f(x)dx = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

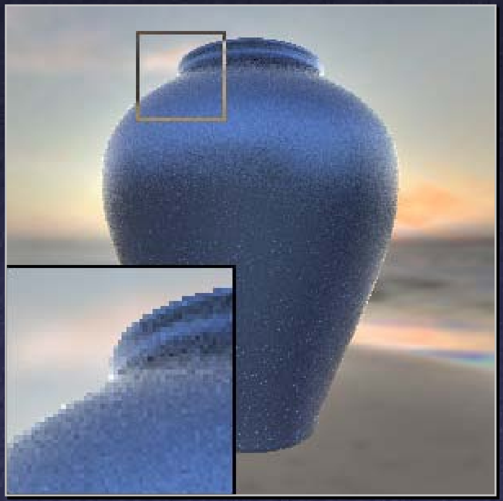
$$Y_i = \frac{f(x_i)}{p(x_i)}$$

Effect of Importance Sampling

- Less noise at a given number of samples



Uniform random sampling



Importance sampling

- Equivalently, need to simulate fewer paths for some desired limit of noise