Monte Carlo Integration

COS 323

Last time

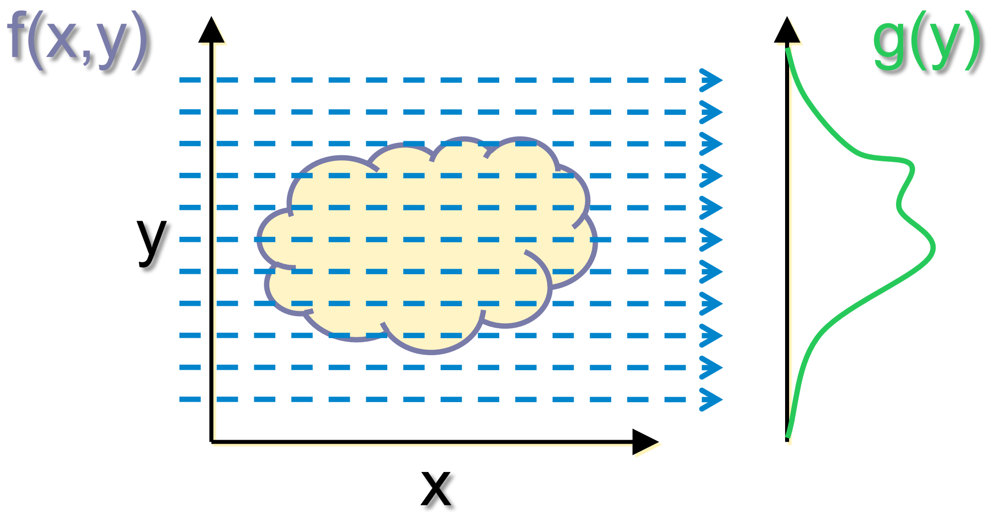
- Interpolatory Quadrature
  - Review formulation; error analysis
- Newton-Cotes Quadrature
  - Midpoint, Trapezoid, Simpson's Rule
- Error analysis for trapezoid, midpoint rules
- Richardson Extrapolation / Romberg Interpolation
- Gaussian Quadrature
- Class WILL be held on Tuesday 11/22

Today

- Monte Carlo integration
- Random number generation
- Cool examples from graphics

Integration in *d* Dimensions?

- One option: nested 1-D integration



$$\int\int f(x,y)\,dx\,dy = \int g(y)\,dy$$

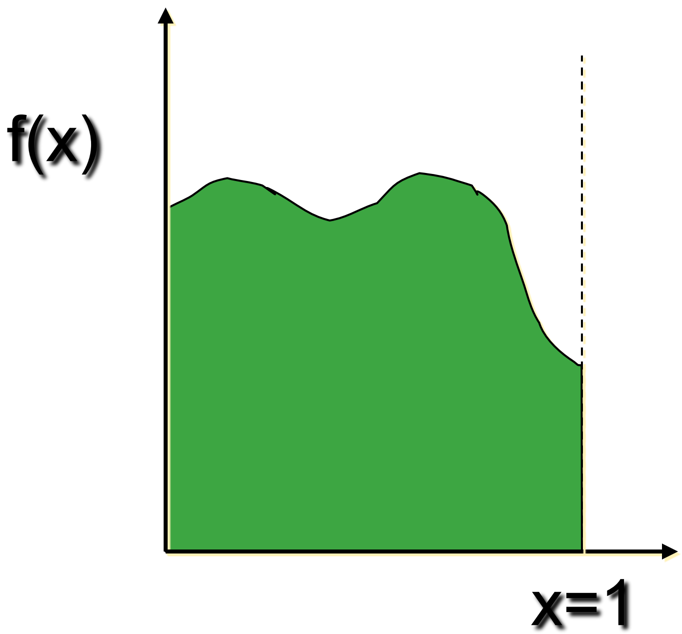
Evaluate the latter numerically, but each “sample” of g(y) is itself a 1-D integral, done numerically

Integration in *d* Dimensions?

- Midpoint rule in *d* dimensions?
  - In 1D: (b-a)/h points
  - In 2D: (b-a)/h^2 points
  - In general: O(1/h^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

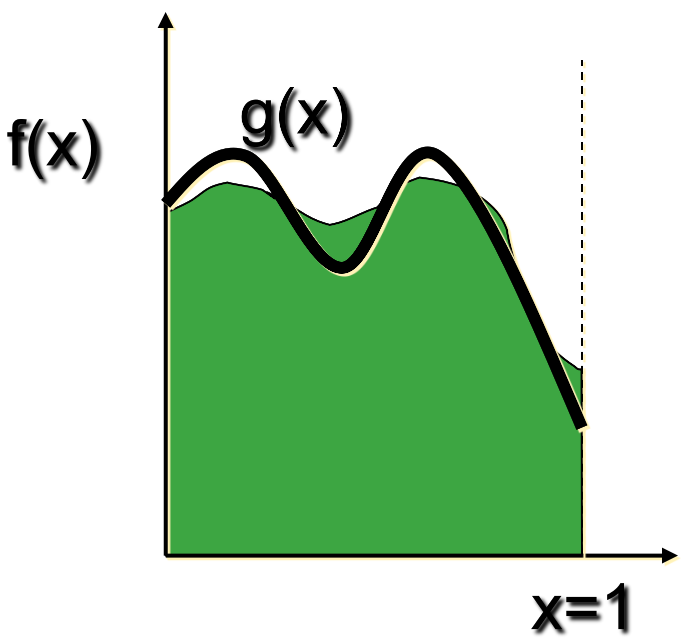
$$\int_0^1 f(x)dx = ?$$



Slide courtesy of
Peter Shirley

We Can Approximate…

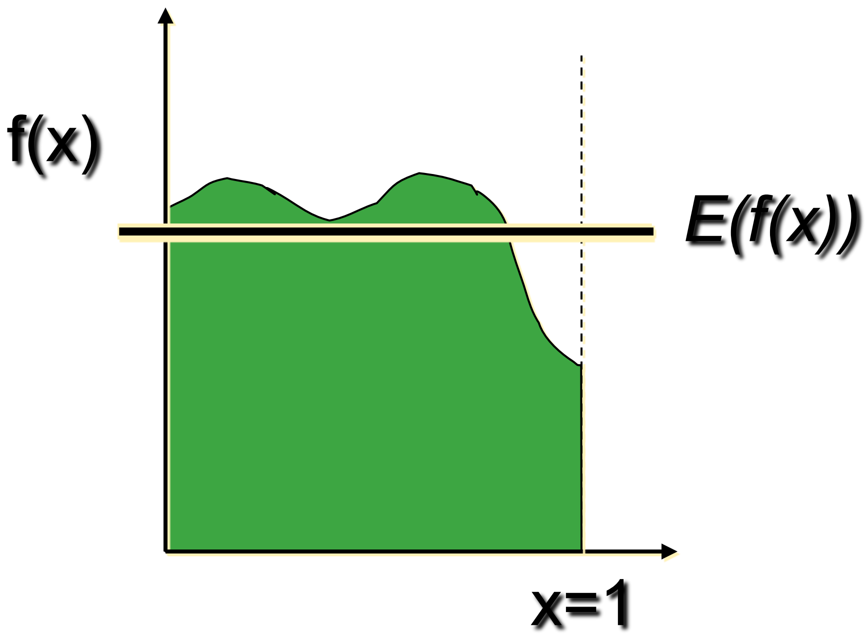
$$\int_0^1 f(x)dx = \int_0^1 g(x)dx$$



Slide courtesy of
Peter Shirley

Or We Can Average

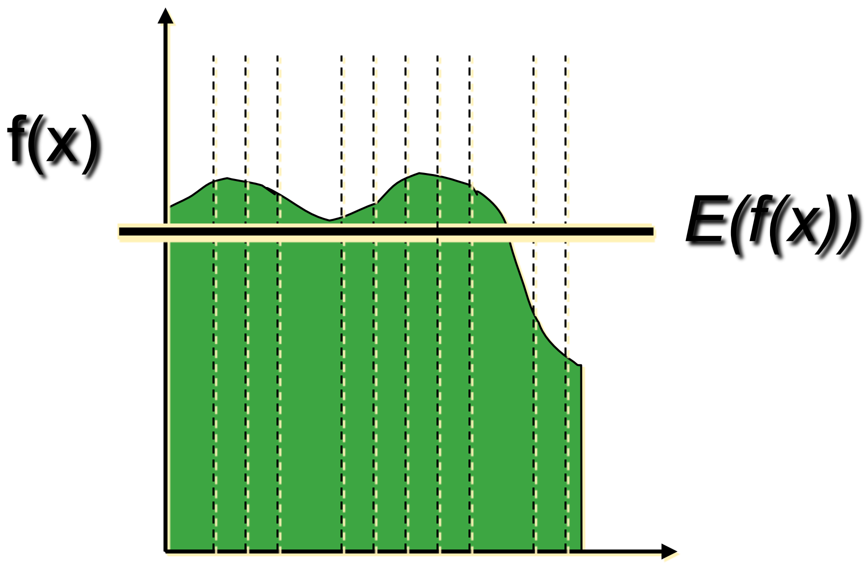
$$\int_0^1 f(x)dx = E(f(x))$$



Slide courtesy of
Peter Shirley

Estimating the Average

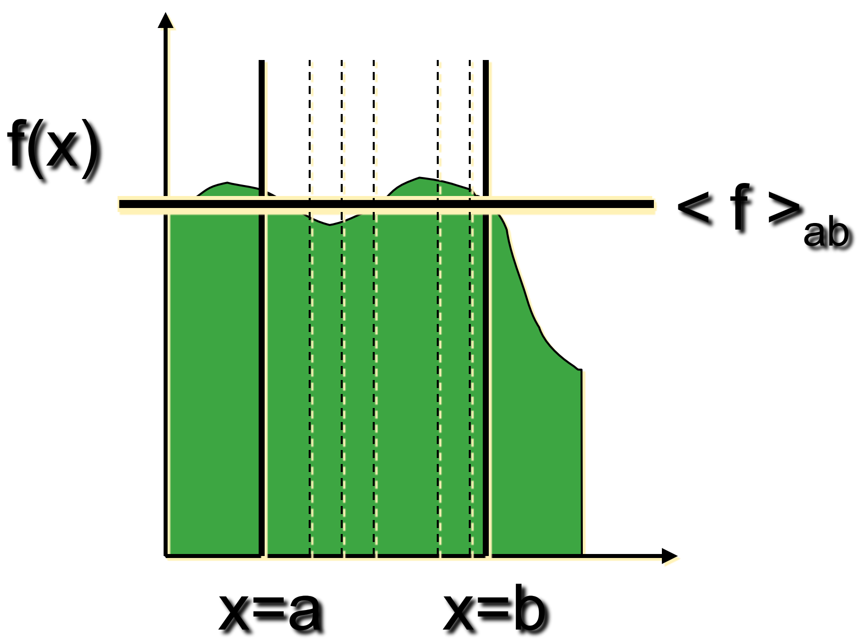
$$\int_0^1 f(x)dx \cong \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$



Slide courtesy of
Peter Shirley

Other Domains

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



Slide courtesy of
Peter Shirley

"Monte Carlo" Integration

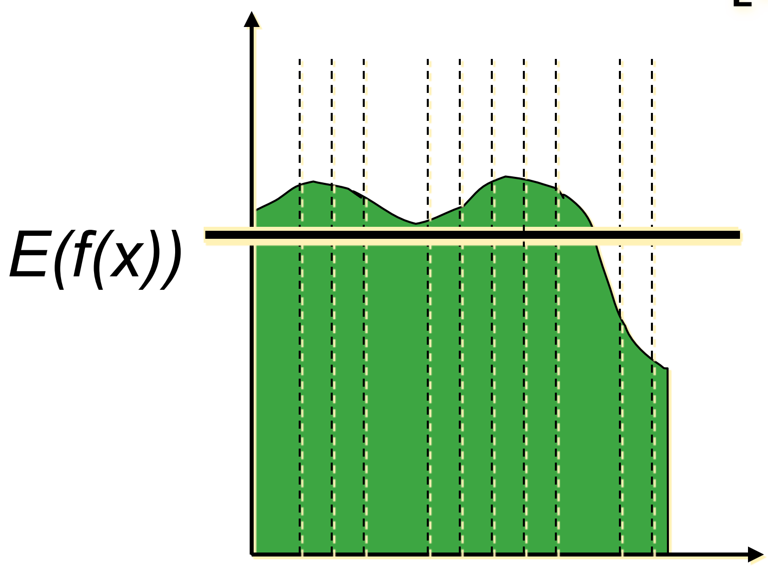
- No "exponential explosion" in required number of samples with increase in dimension
- (Some) resistance to badly-behaved functions

Le Grand Casino de Monte-Carlo

Variance

Sample variance of f :

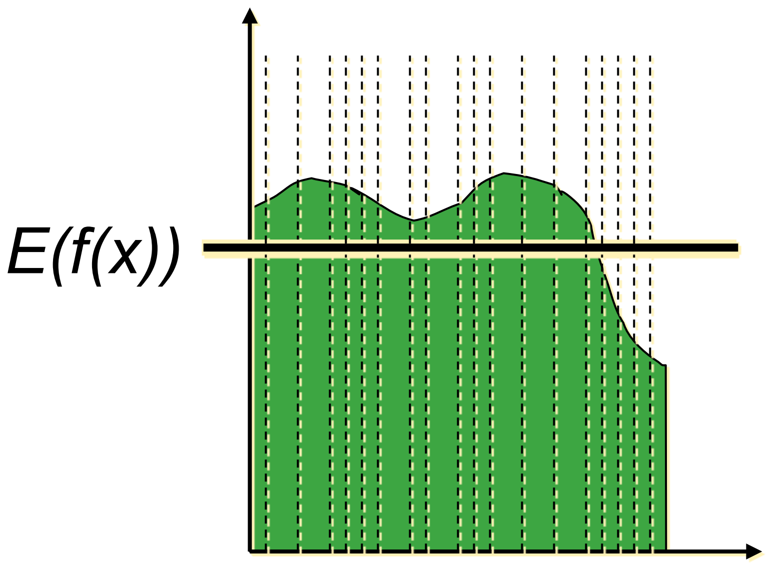
$$Var\left[f(x)\right] = \frac{1}{N-1}\sum_{i=1}^{N}[f(x_i) - E(f(x))]^2$$



Variance

Sample variance of f :

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



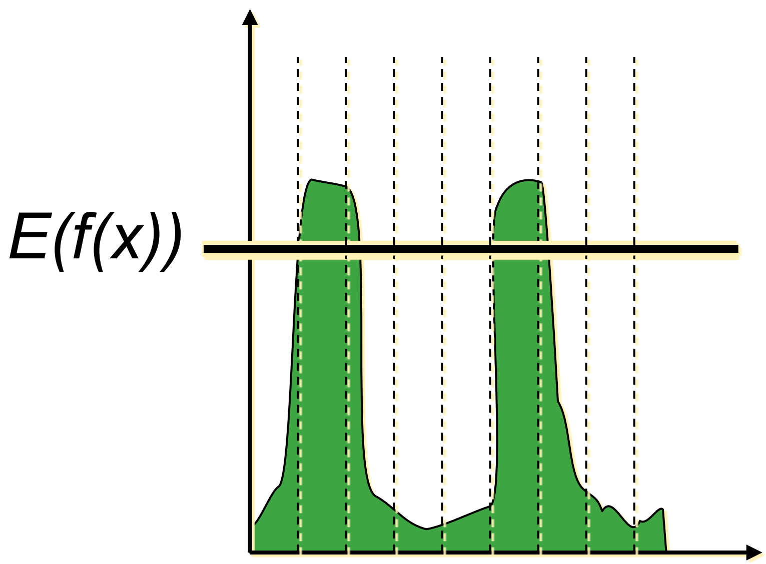
$$Var\left[\sum_{i=1}^{N} Y_i\right] = \sum_{i=1}^{N} Var\left[Y_i\right]$$

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

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

Variance

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

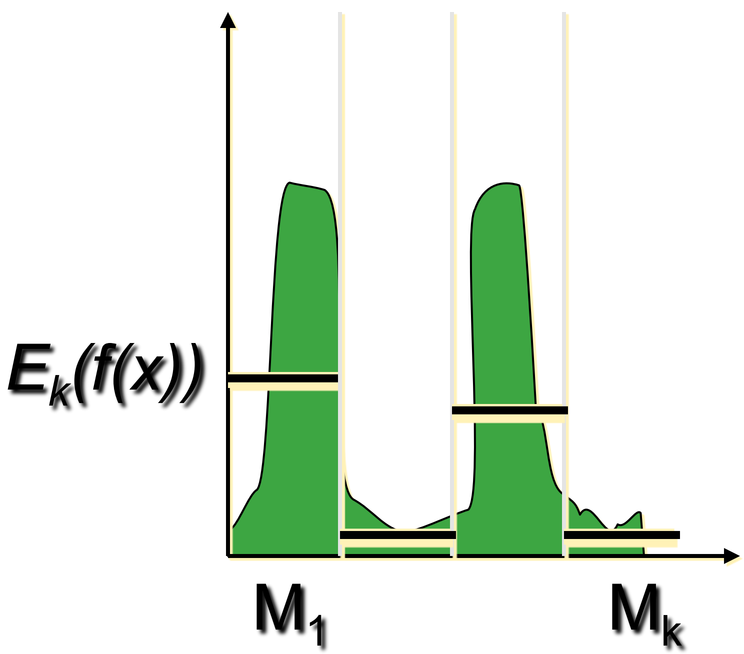


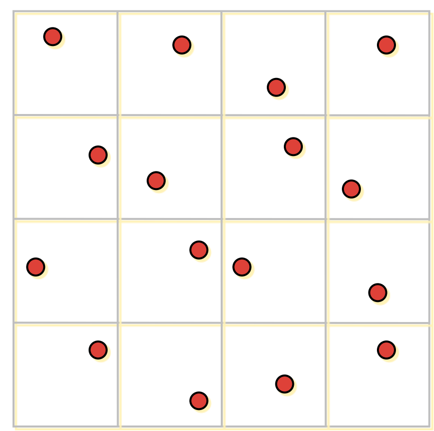
Variance Reduction Techniques

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

Stratified Sampling

- Estimate subdomains separately

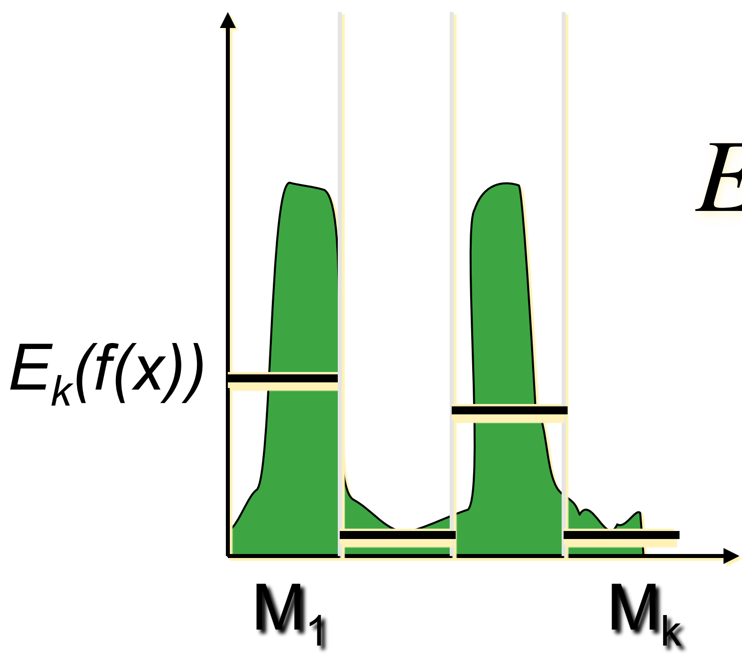




Can do this recursively!

Stratified Sampling

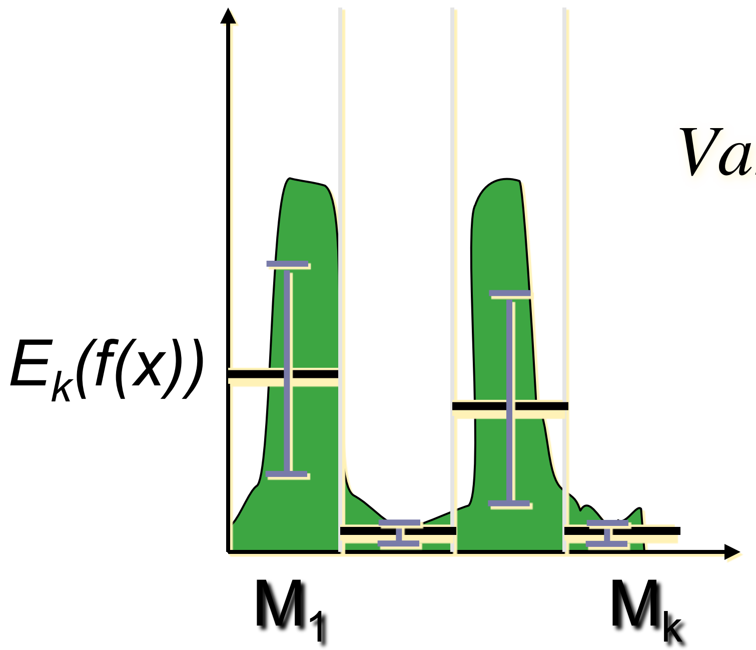
- This is still unbiased



$$E = \sum_{j=1}^{k} \frac{vol(M_j)}{N_j} \sum_{n=1}^{N_j} f(x_{jn})$$

Stratified Sampling

- Less overall variance if less variance in subdomains

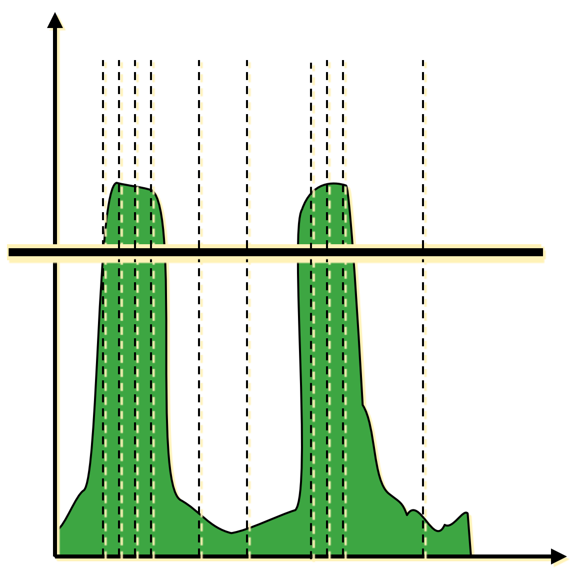


$$Var[E] = \sum_{j=1}^{k} \frac{vol(M_j)^2}{N_j} Var[f(x)]\Big|_{M_j}$$

Total variance minimized when number of points in each subvolume M_j proportional to error in M_j.

Importance Sampling

- Put more samples where f(x) is bigger



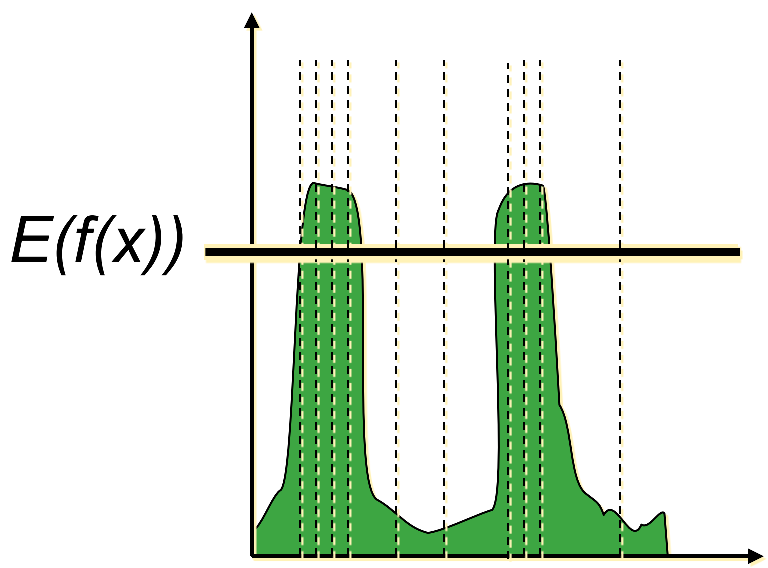
$$\int_{\Omega} f(x)dx = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

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

and x_i drawn from $P(x)$

Importance Sampling

- This is still unbiased



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

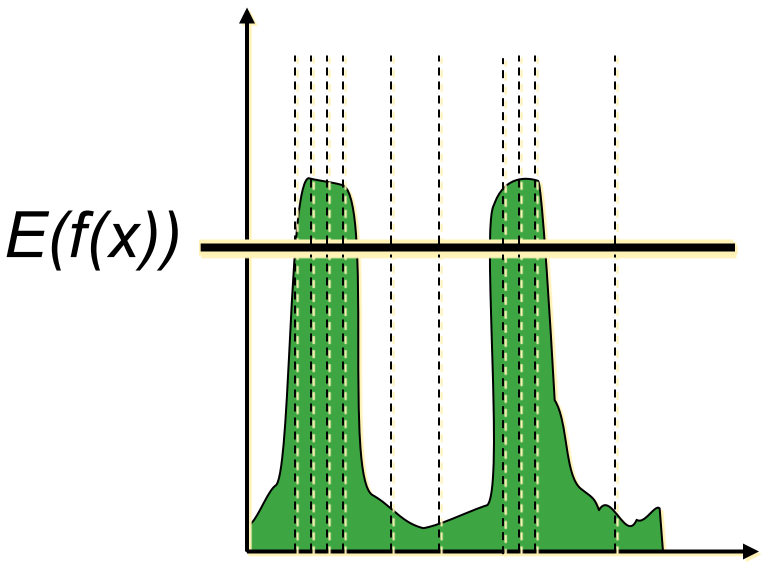
$$= \int_\Omega \frac{f(x)}{p(x)}\,p(x)\,dx$$

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

for all N

Importance Sampling

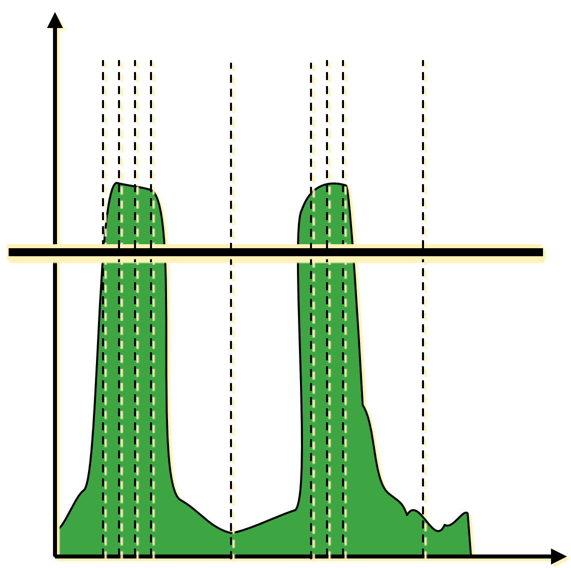
- Variance depends on choice of p(x):



$$Var(E) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{f(x_n)}{p(x_n)}\right)^2 - E^2$$

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

Random number generation

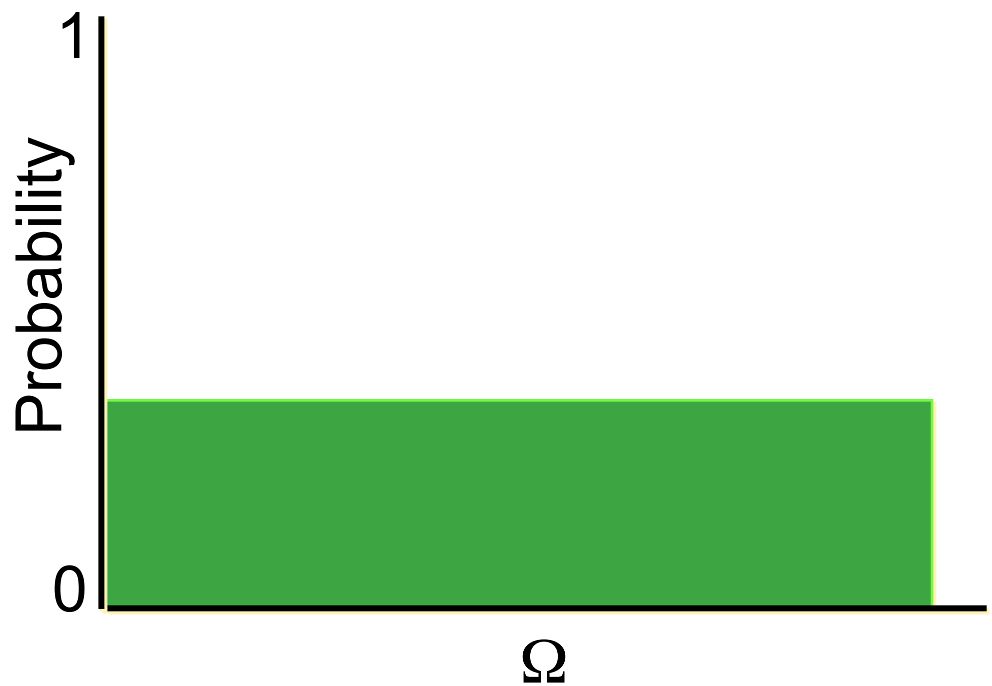
True random numbers

- http://www.random.org/

10101111 00101011 10111000 11110110 10101010 00110001 01100011 00010001
00000011 00000010 00111111 00010011 00000101 01001100 10000110 11100010
10010100 10000101 10000011 00000100 00111011 10111000 00110000 11001010
11011101 11101111 00100010 10101011 00100110 10101111 00001011 10110100
00011100 00001111 11001001 11001100 01111101 10000100 10111000 01101011
01101011 01111101 11001010 11101110 11101110 00100010 10110100 01001000
11010111 11011011 11100100 01010010 10111101 01011010 01001110 01110000
00100010 11000111 01010000 10110011 01001011 00110001 01011100 10001111
11111000 10101011 01011011 01010000 01101111 00011001 00000011 00110000
10000001 00000110 11010011 00011110 11101101 00000011 00100110 01010011
11010111 10010001 10000111 01010010 01101010 00100101 10011111 01000111
10101001 01100001 01010011 01001000 11010110 01111110 11010011 01110110
00000001 01001110 00011001 00111001

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

Desirable properties

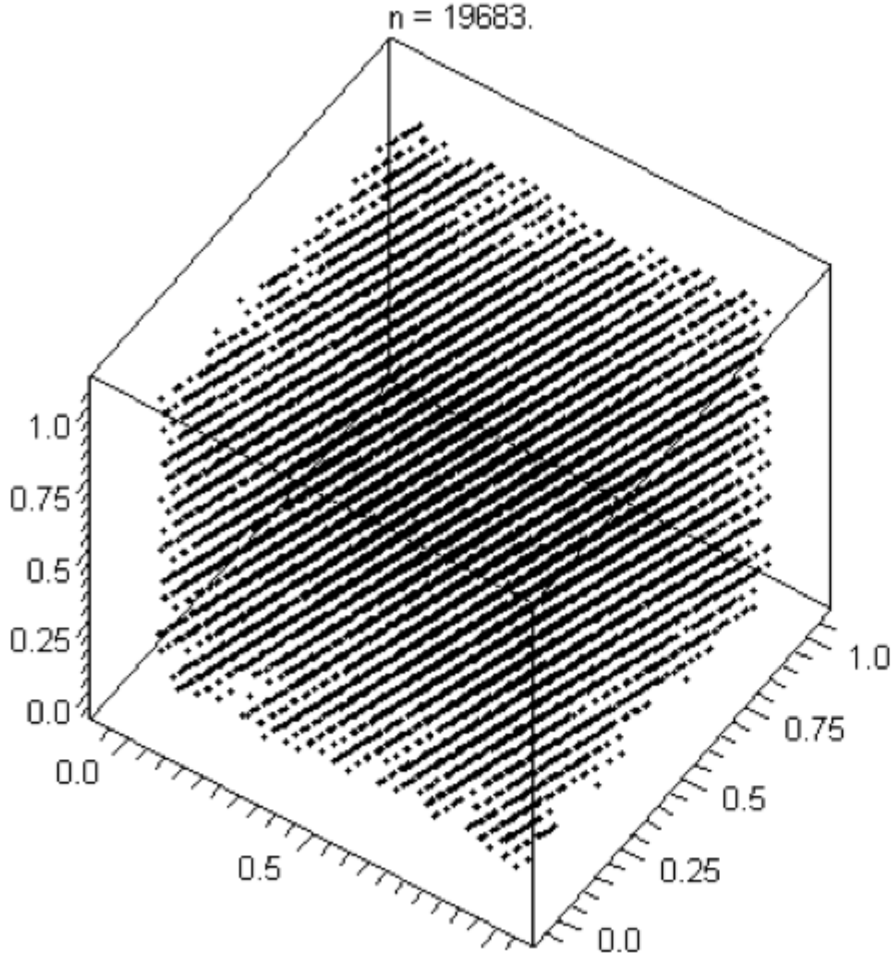
- Random pattern: Passes statistical tests (e.g., can use chi-squared)
- Long period: Go as long as possible without repeating
- Efficiency
- Repeatability: Produce same seuqence if started with same initial conditions
- Portability

Linear Congruential Methods

$$x_{n+1} = (ax_n + b) \bmod c$$

- Choose constants carefully, e.g.
  a = 1664525
  b = 1013904223
  c = $2^{32} - 1$
- Results in integer in [0, c)
- Not suitable for MC: e.g. exhibit serial correlations

Problem with LCGs



No higher resolution available.

Fibonacci Generators

- Takes form $x_n = x_{n-j} - x_{n-k}$
- Standard choices of j, k: e.g., (7, 10), (31, 63), (168, 521)
- Proper initialization is important and hard
- Built-in correlation!
- Not totally understood in theory (need statistical tests to evaluate)

Seeds

- Why?

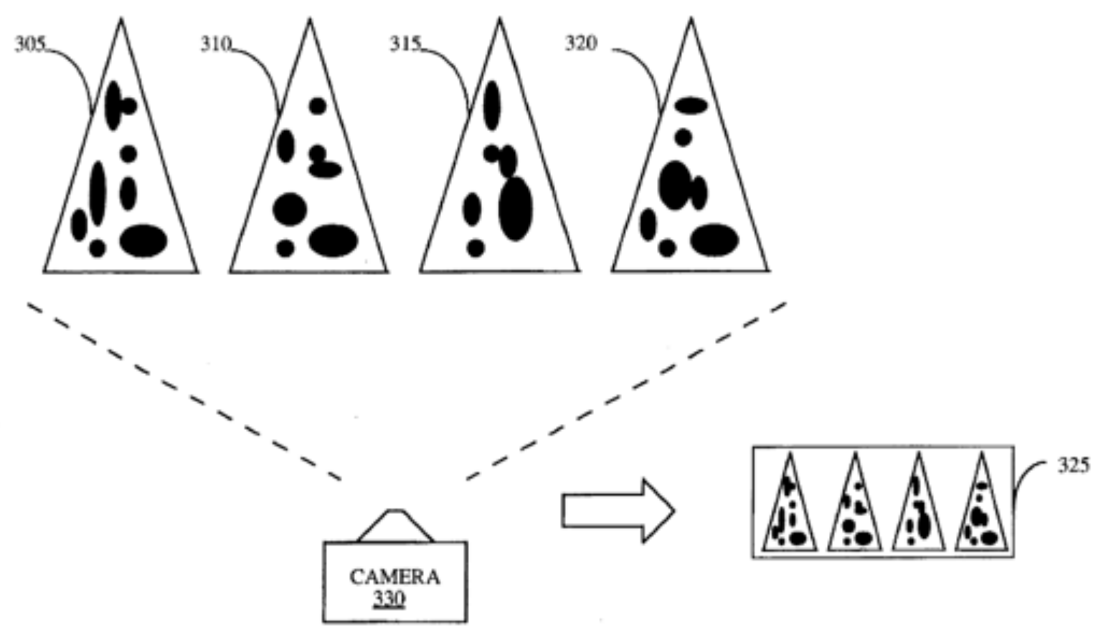


FIG. 3

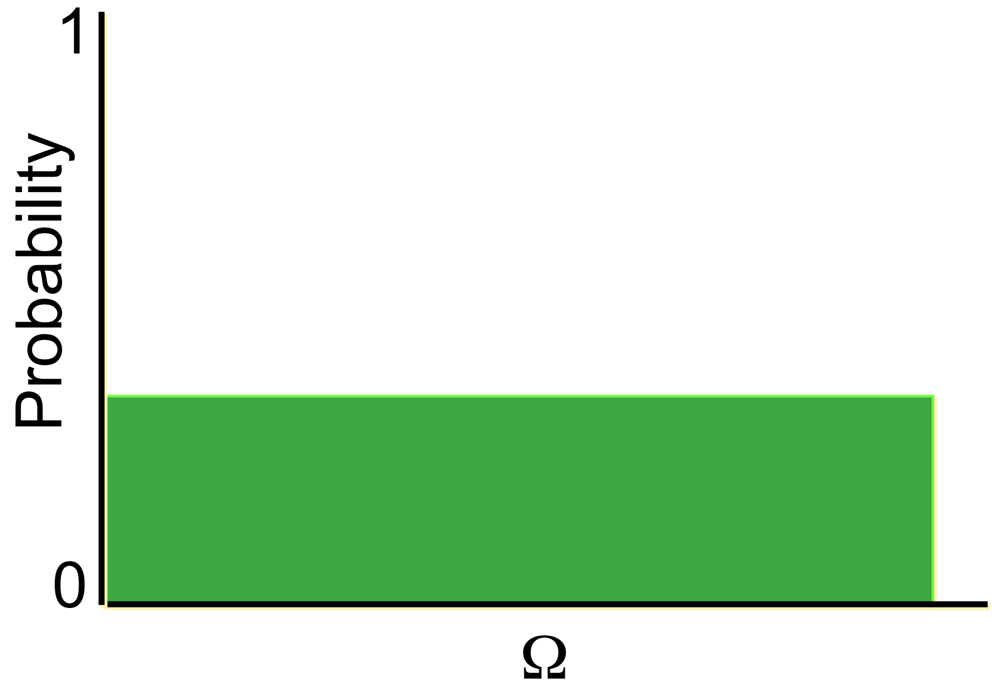
- http://www.google.com/patents/about/5732138_Method_for_seeding_a_pseudo_rand.html?id=ou0gAAAAEBAJ

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

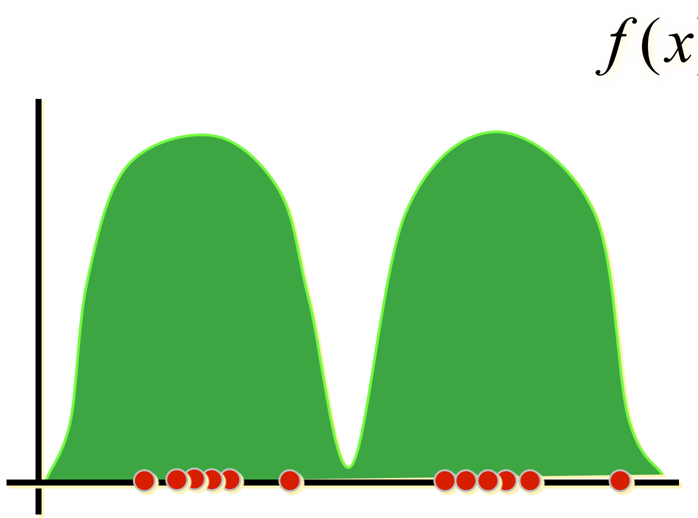
Generating Random Points

- Uniform distribution:
  - Use pseudorandom number generator



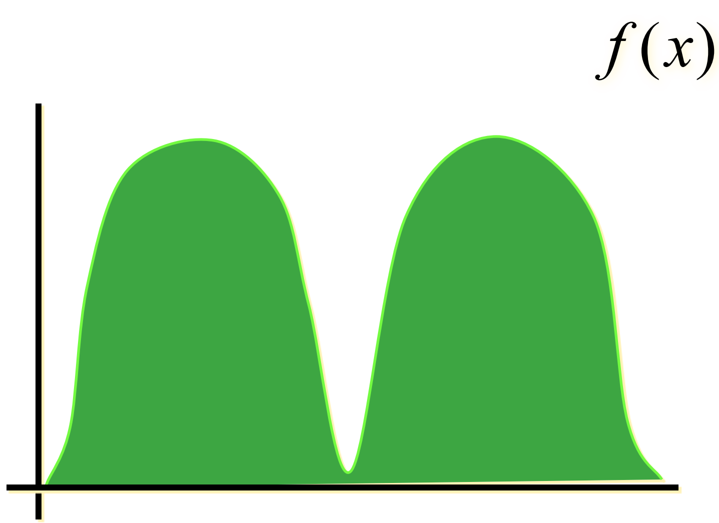
Sampling from a non-uniform distribution

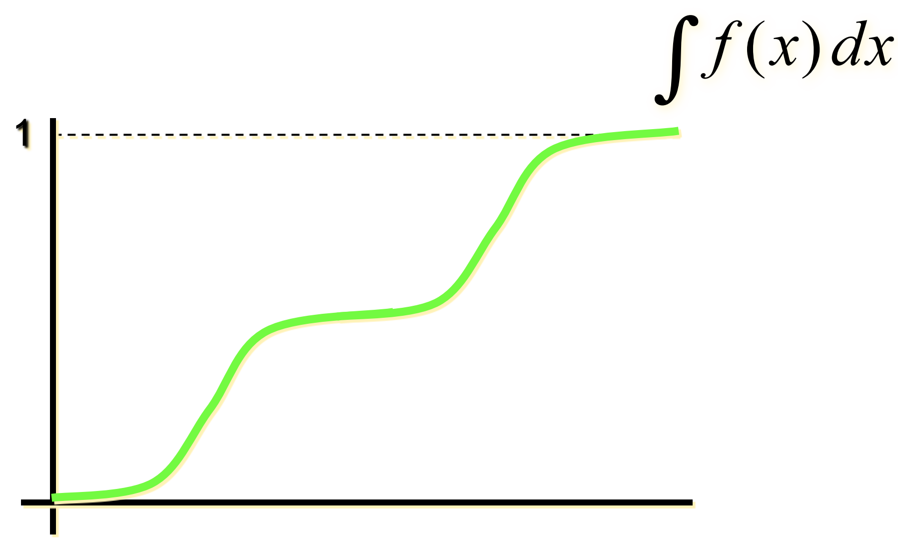
- Specific probability distribution:
  - Function inversion
  - Rejection



Sampling from a non-uniform distribution

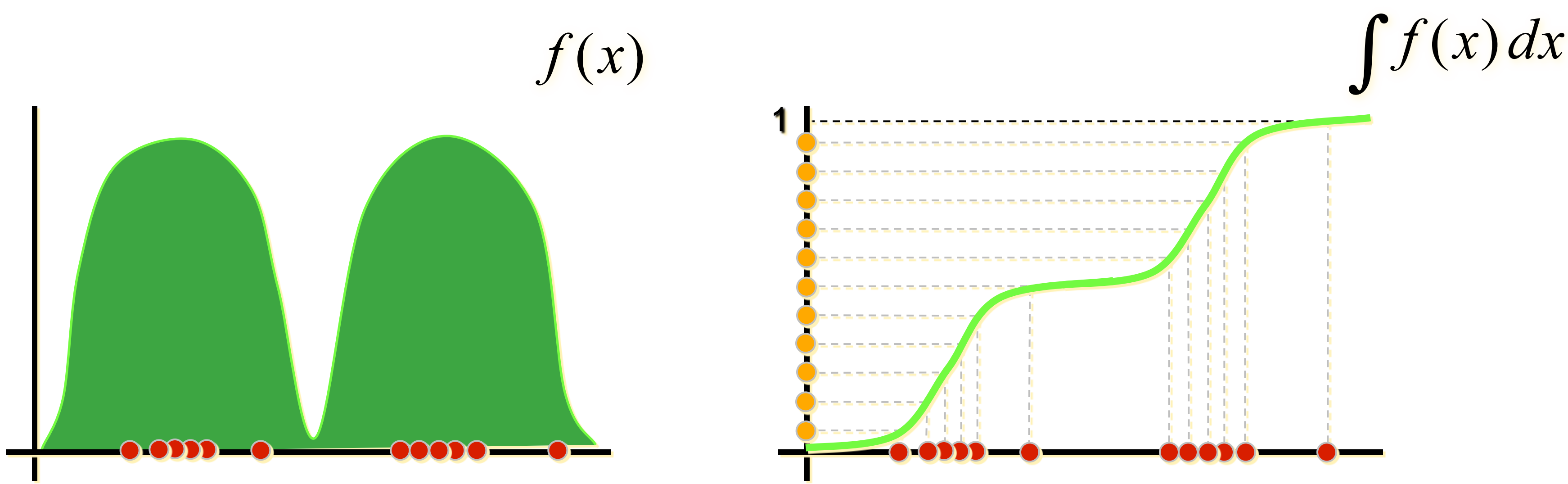
- "Inversion method"
  - Integrate $f(x)$: Cumulative Distribution Function





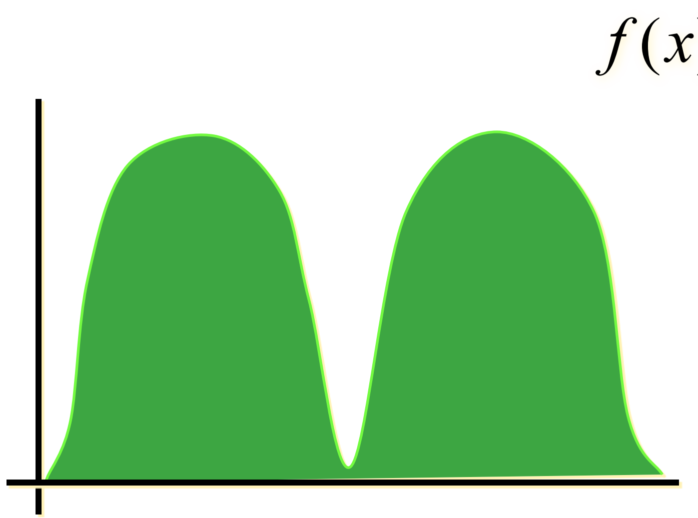
Sampling from a non-uniform distribution

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



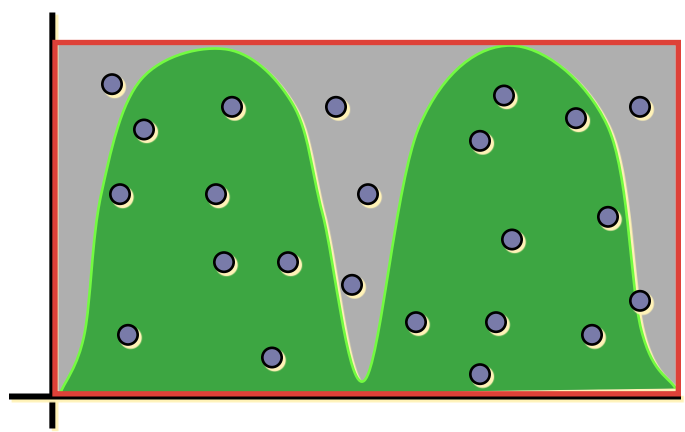
Sampling from a non-uniform distribution

- Specific probability distribution:
  - Function inversion
  - Rejection



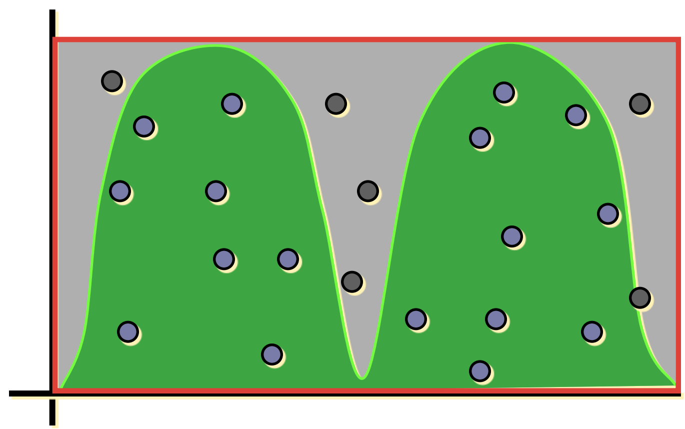
Sampling from a non-uniform distribution

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



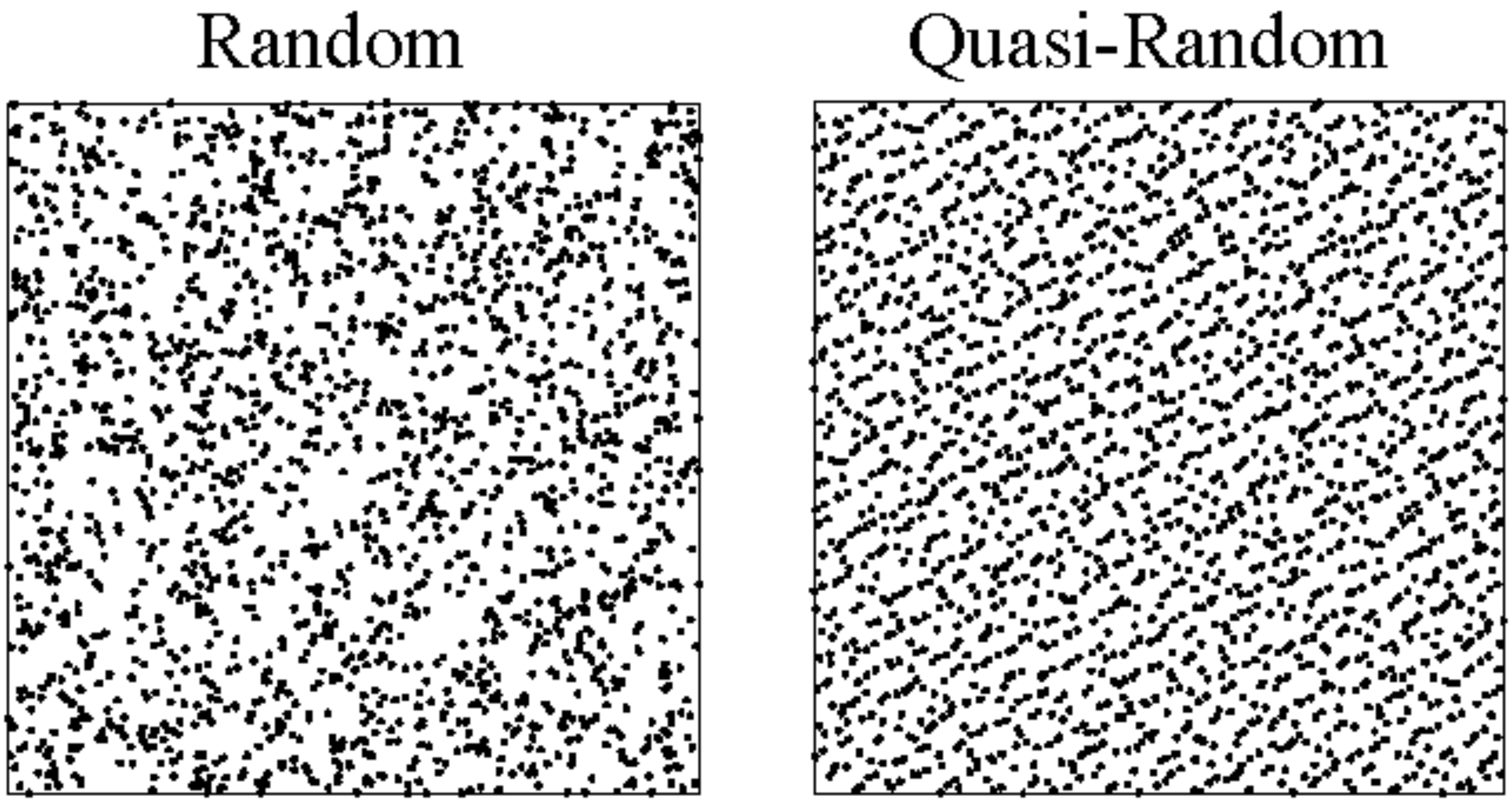
Sampling from a non-uniform distribution

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

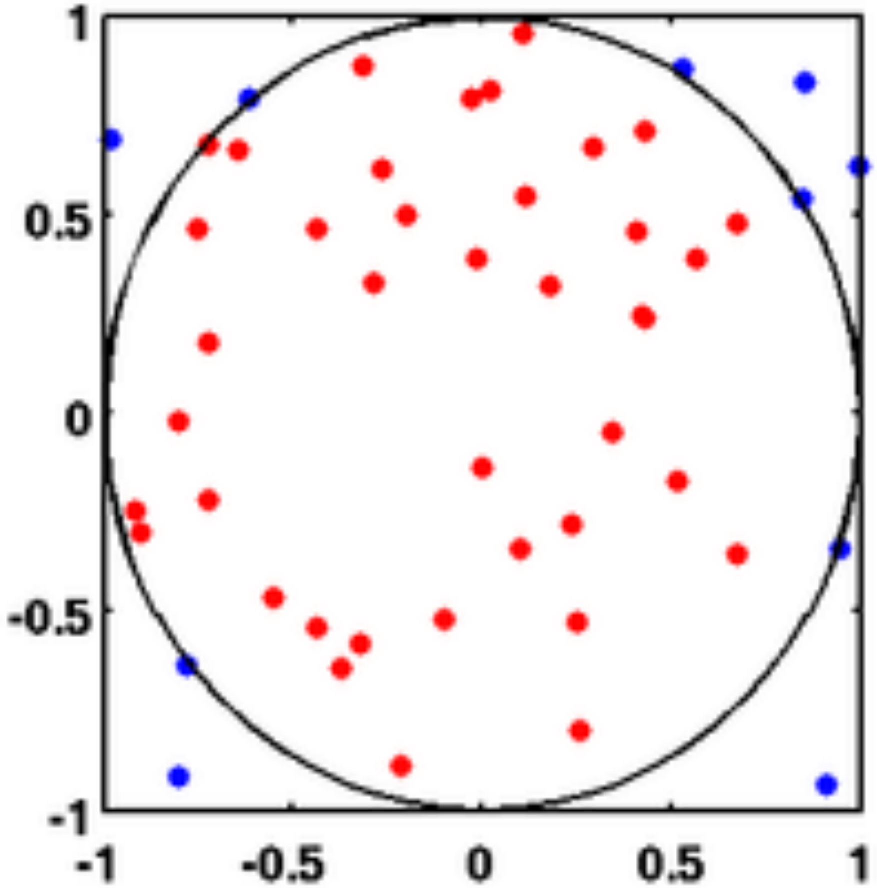


Doesn't require cdf: Can use directly for importance sampling.

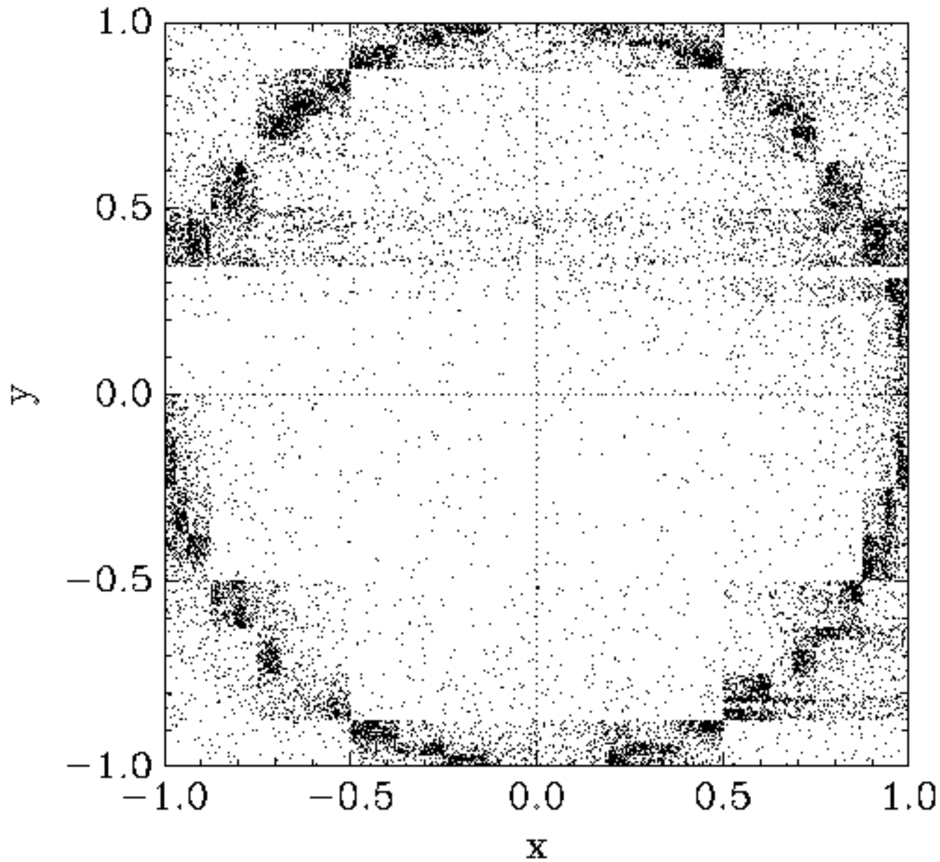
Quasi-Random Sampling



Example: Computing pi



With Stratified Sampling

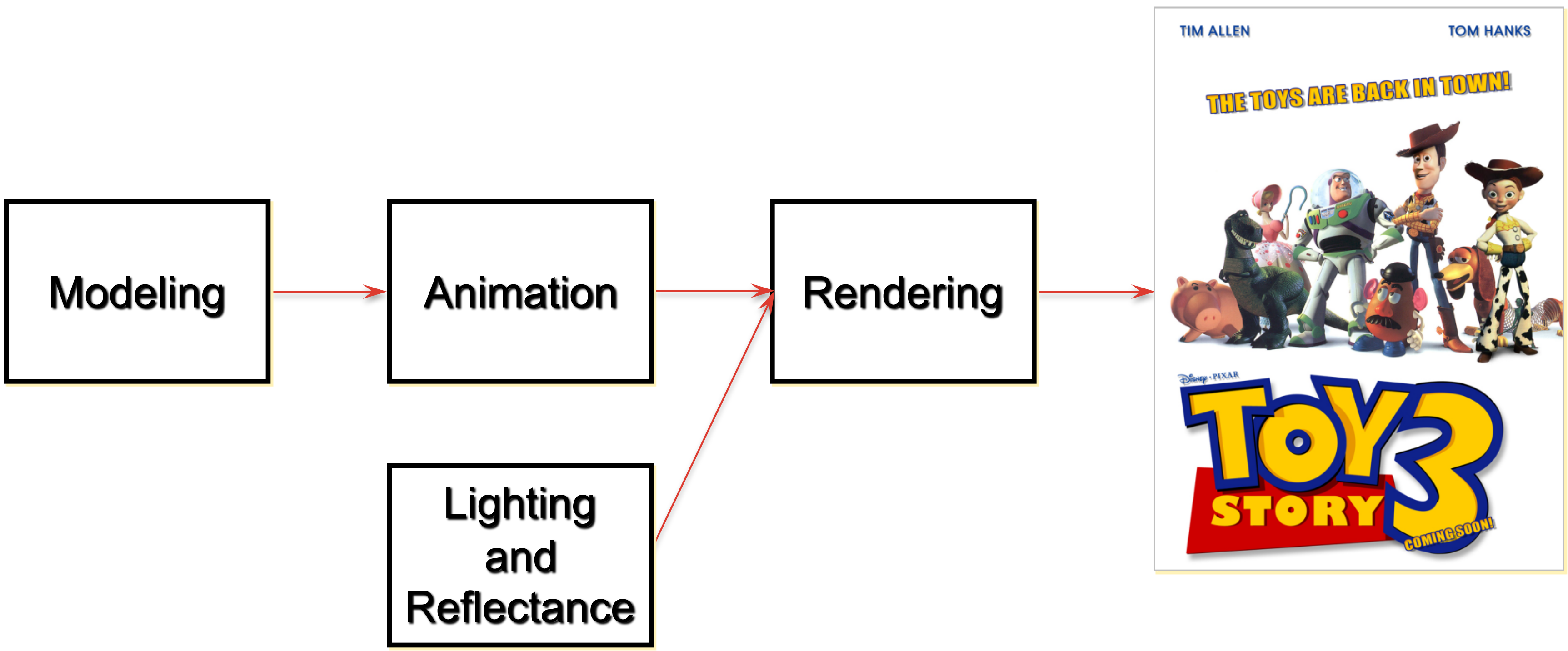


Monte Carlo in Computer Graphics

or, Solving Integral Equations for Fun and Profit

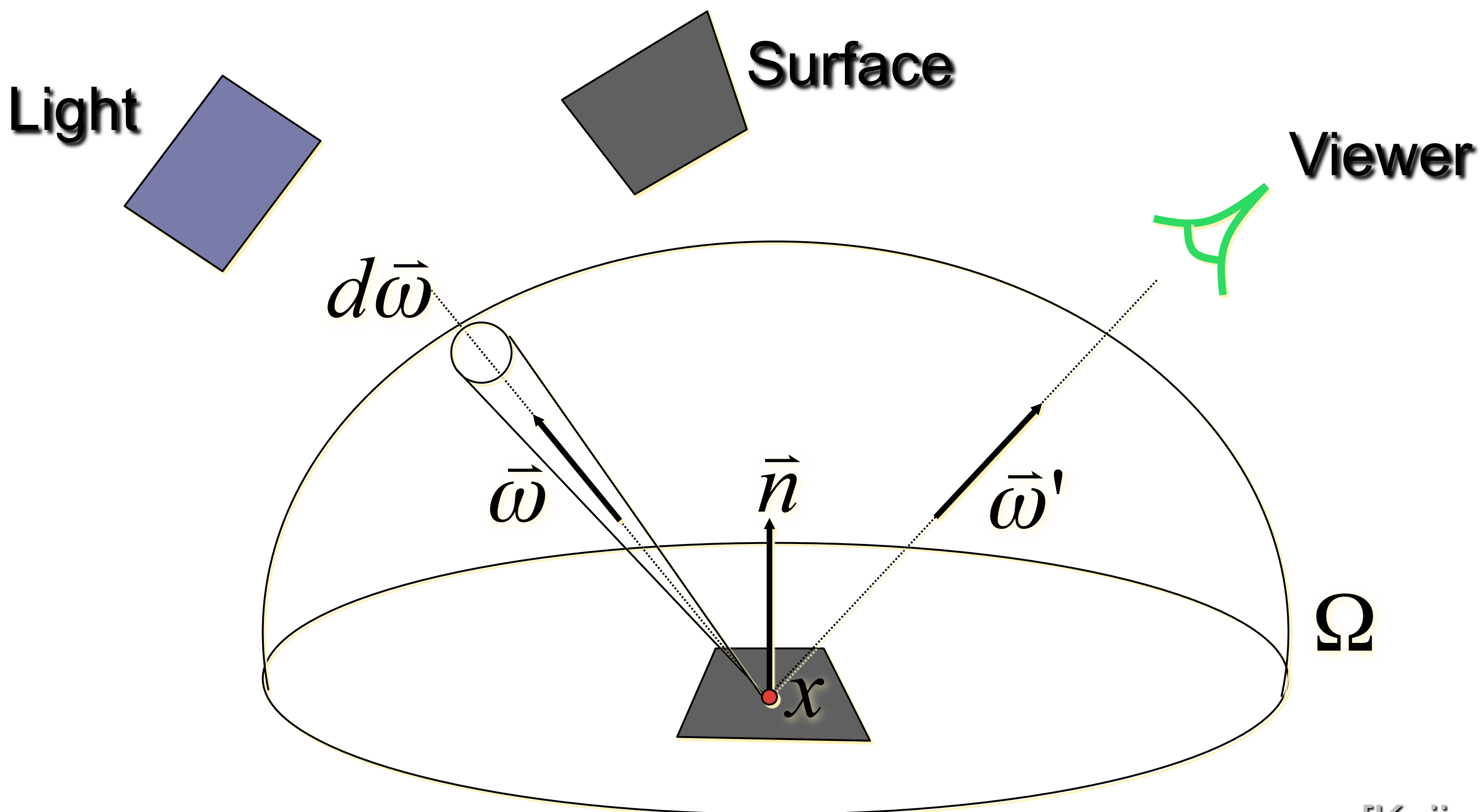
or, Ugly Equations, Pretty Pictures

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

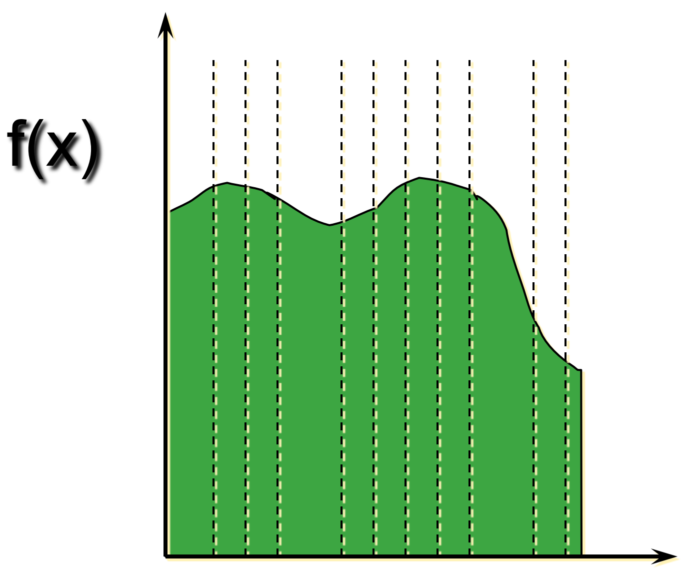
Rendering Equation

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

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

Jensen

Monte Carlo Integration

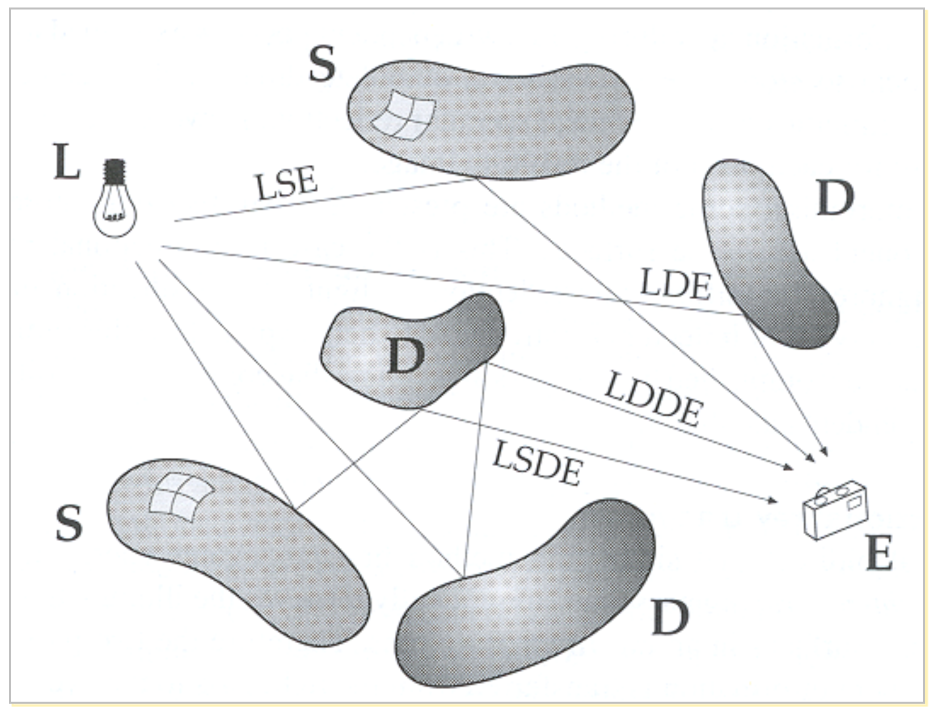


$$\int_0^1 f(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

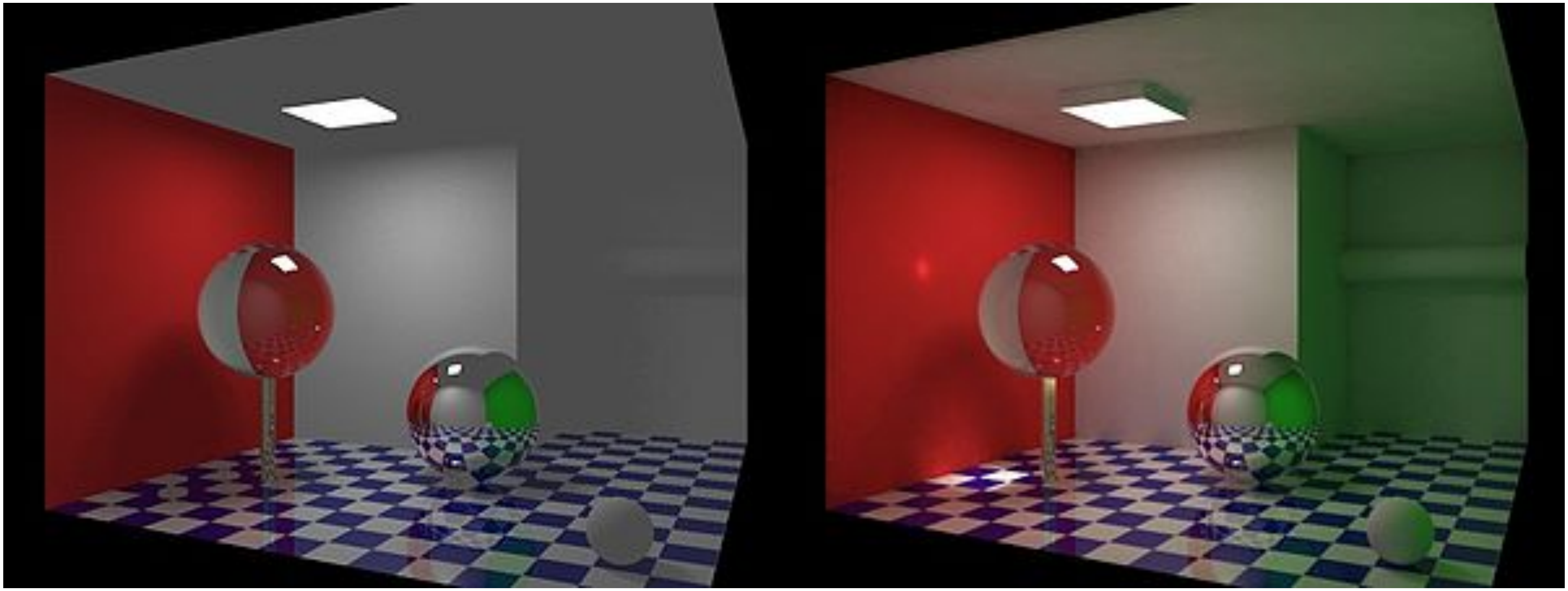
Shirley

Monte Carlo Path Tracing

Estimate integral
for each pixel
by random sampling



Global Illumination



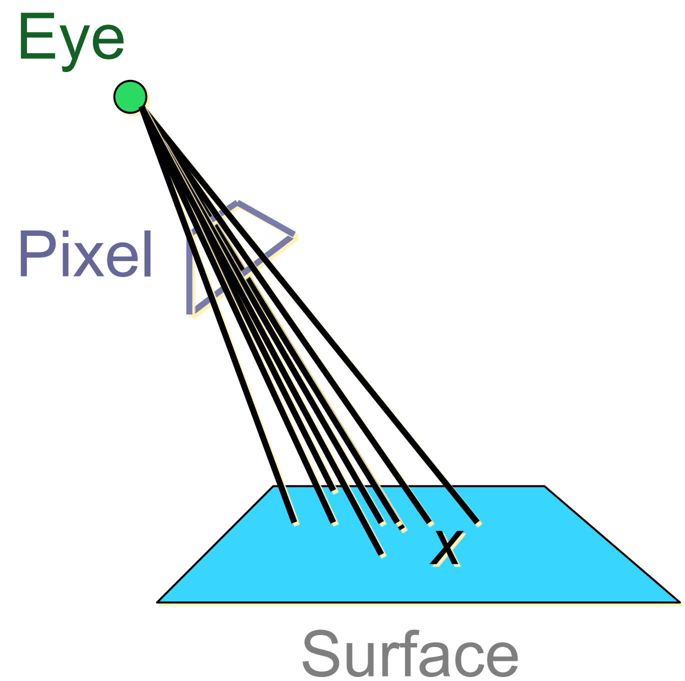
From Grzegorz Tanski, Wikipedia

Monte Carlo Global Illumination

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

Monte Carlo Global Illumination

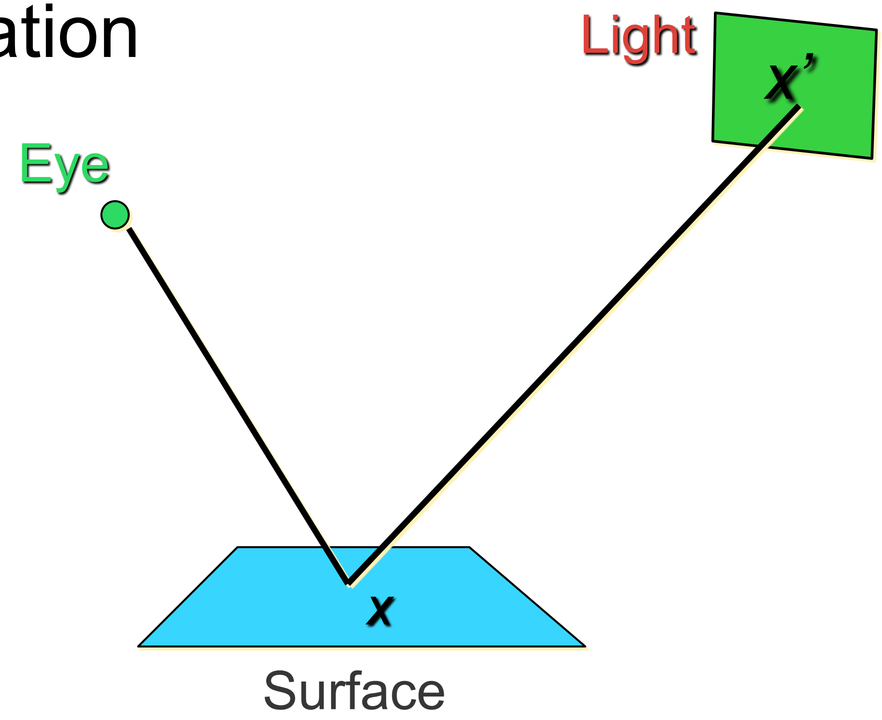
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



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

Monte Carlo Global Illumination

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



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

Monte Carlo Global Illumination

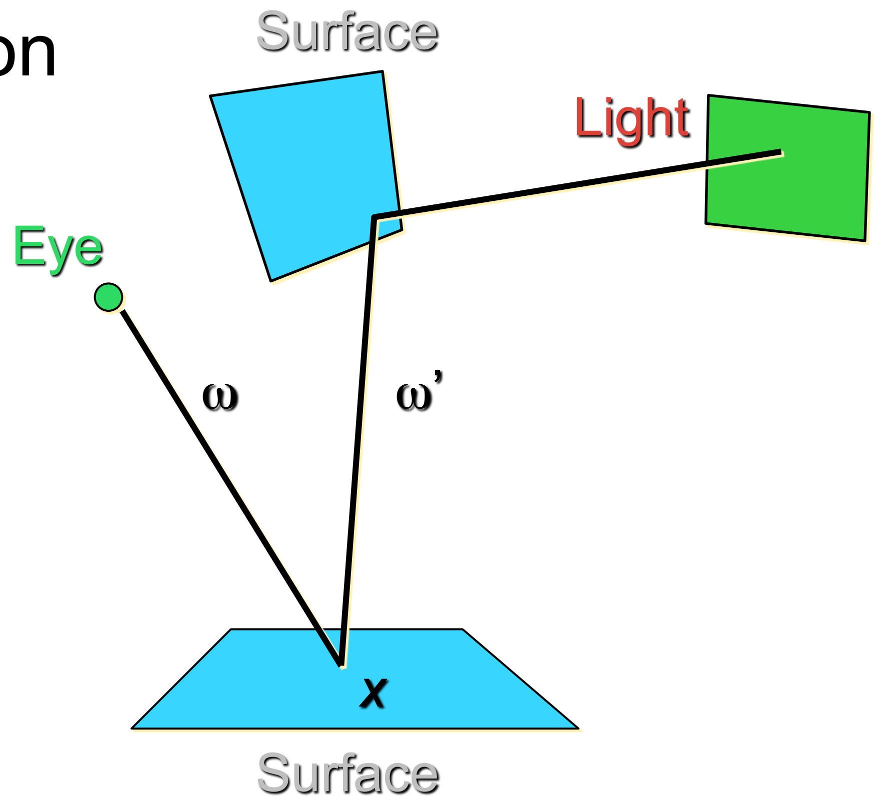
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

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

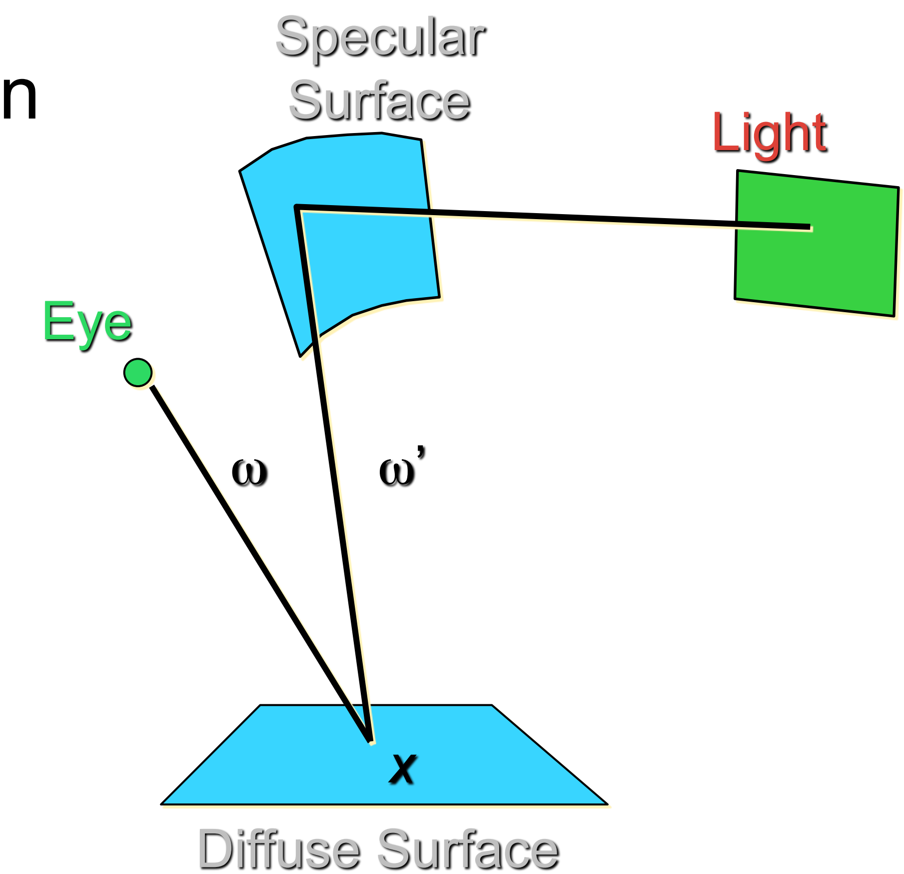
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

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

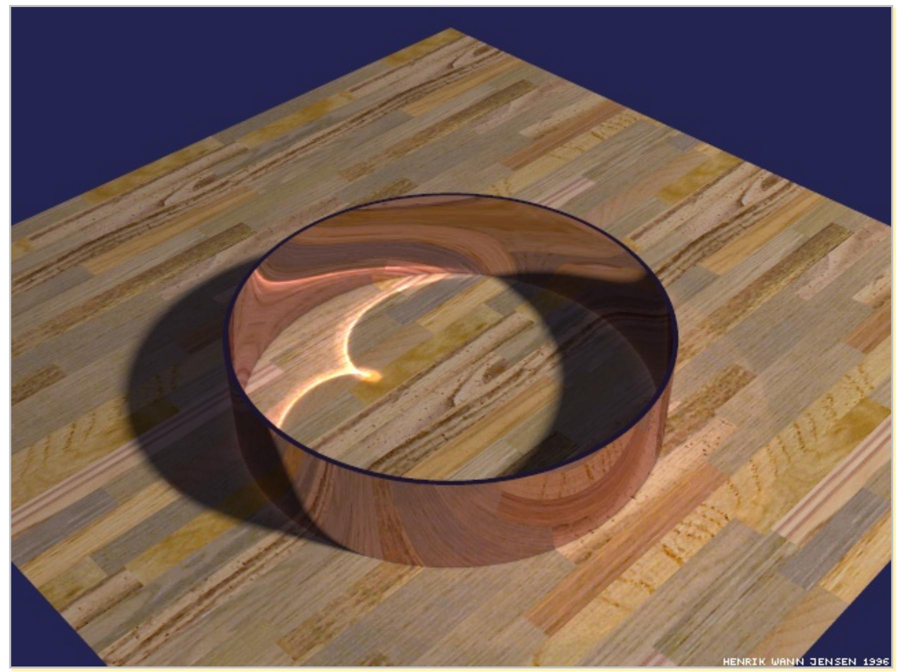
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



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

Monte Carlo Global Illumination

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

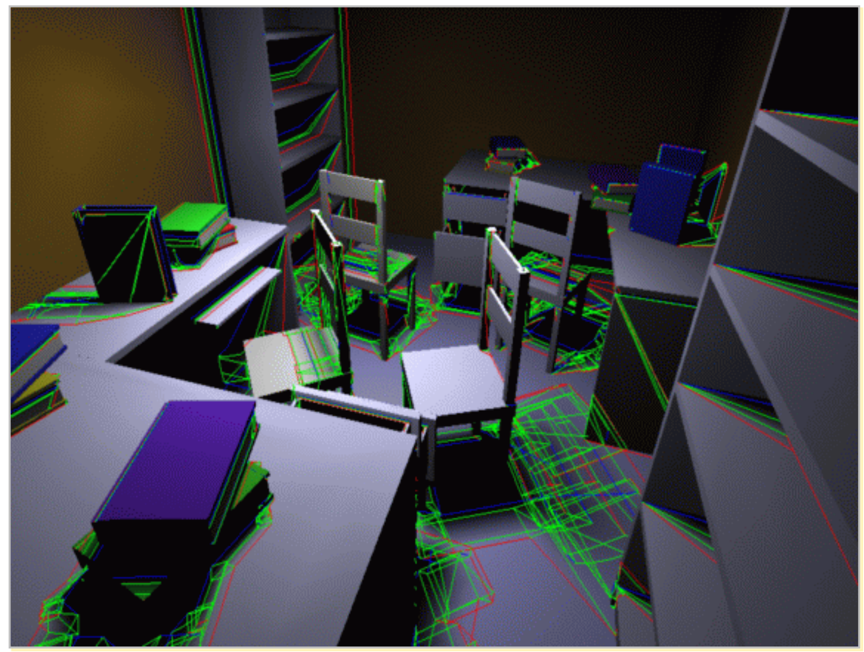


Jensen

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

Challenge

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



Drettakis

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

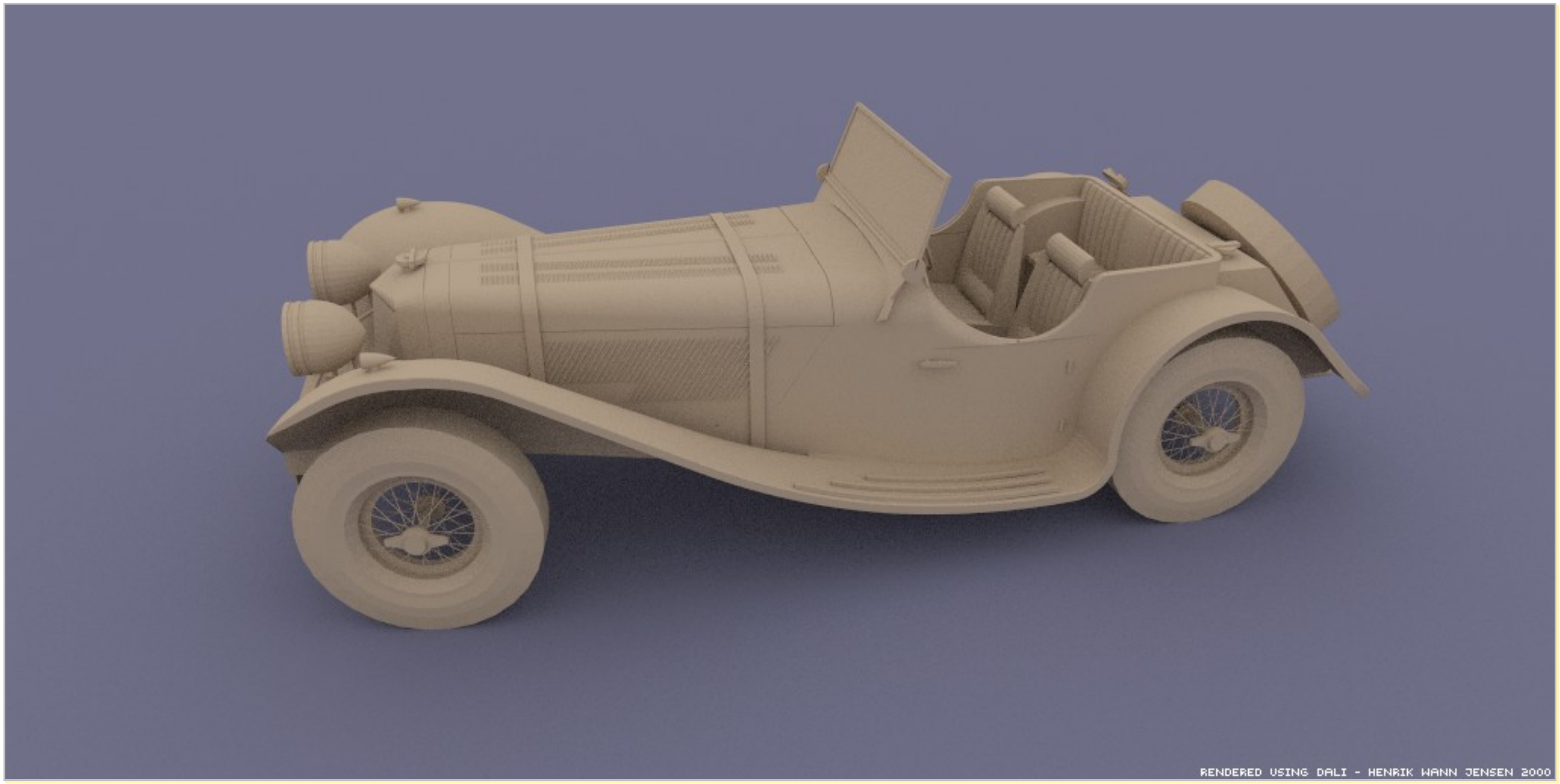
Challenge

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

Jensen

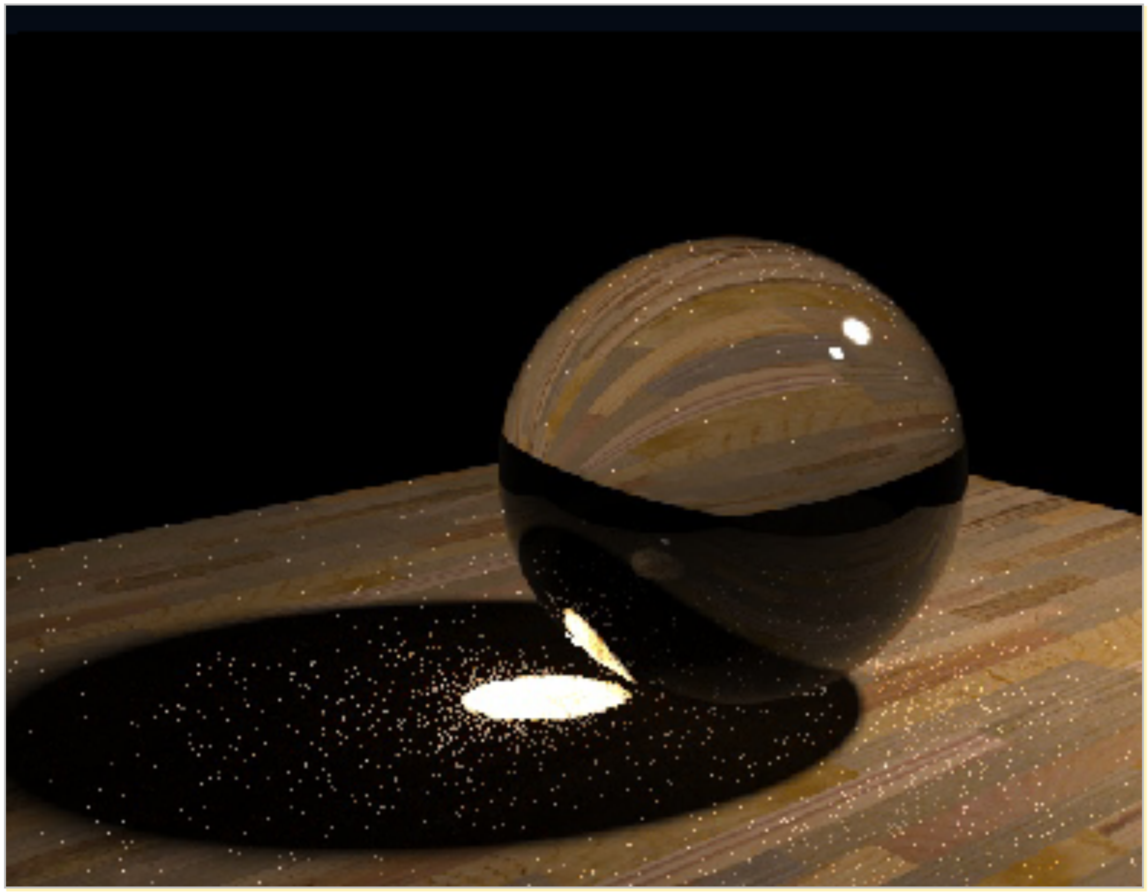
$$L(x,\vec{w}) = L_e(x,x \rightarrow e) + \int_S f_r(x,x' \rightarrow x, x \rightarrow e) L(x' \rightarrow x) V(x,x') G(x,x') dA$$

Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

Monte Carlo Path Tracing

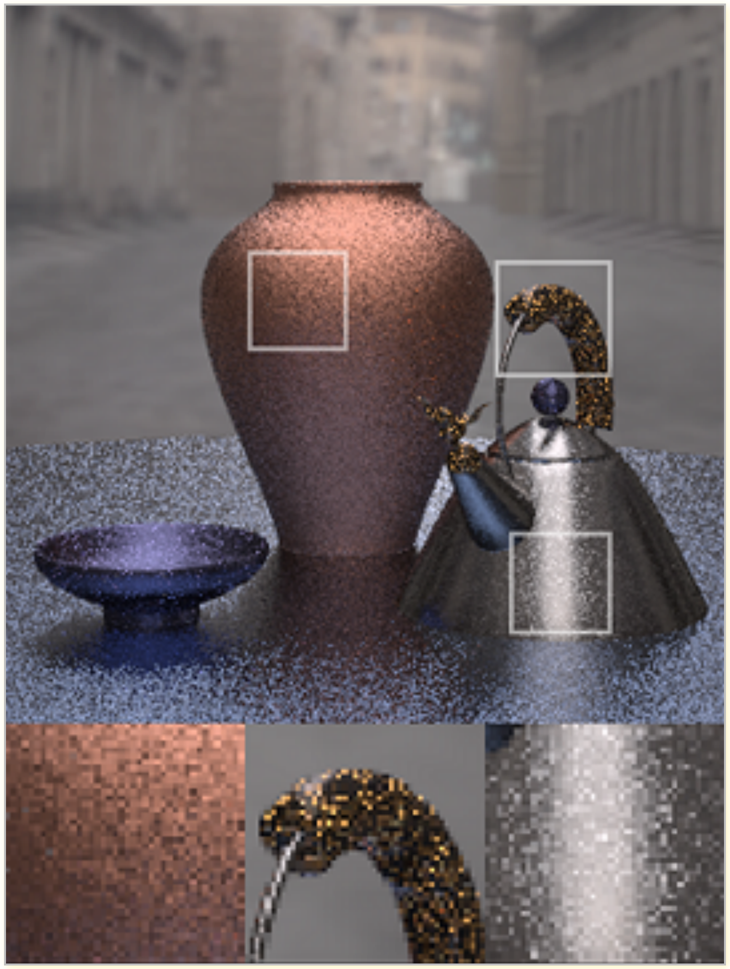


1000 paths/pixel

Jensen

Monte Carlo Path Tracing

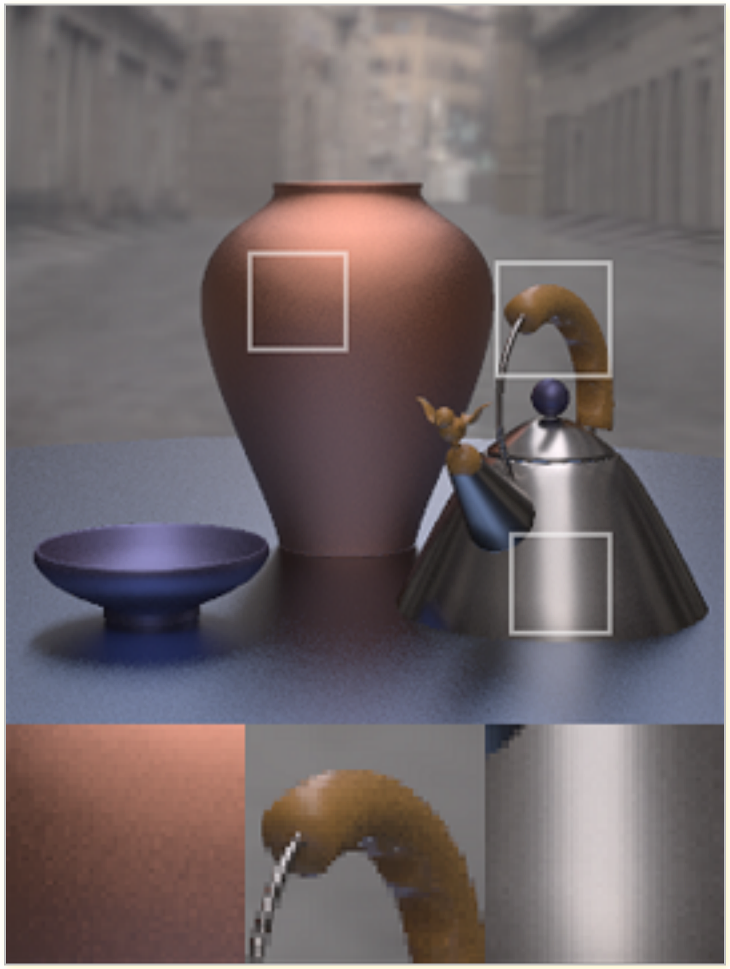
- Drawback: can be noisy unless *lots* of paths simulated
- 40 paths per pixel:



Lawrence

Monte Carlo Path Tracing

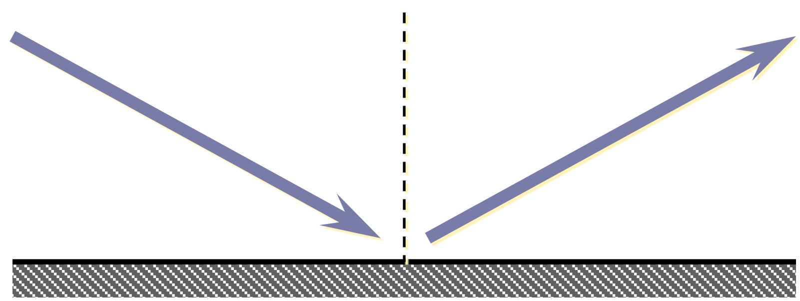
- Drawback: can be noisy unless *lots* of paths simulated
- 1200 paths per pixel:



Lawrence

Reducing Variance

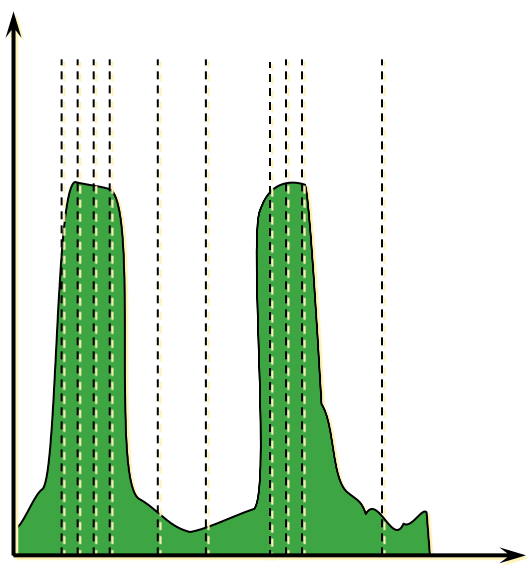
- Observation: some paths more important (carry more energy) than others
  - For example, shiny surfaces reflect more light in the ideal “mirror” direction



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

Importance Sampling

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

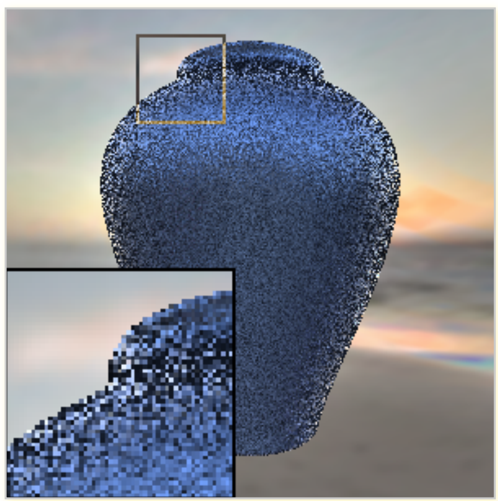


$$\int_0^1 f(x)dx = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

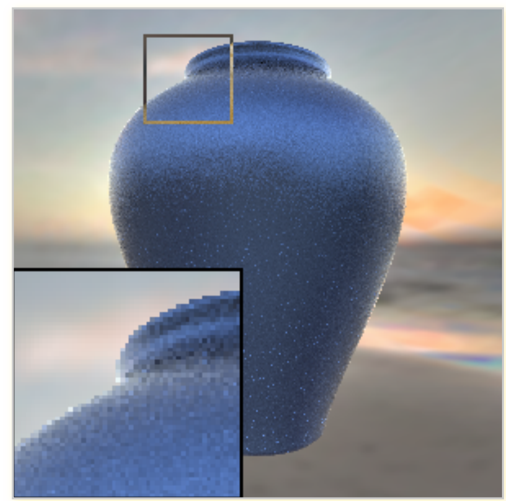
$$Y_i = \frac{f(x_i)}{p(x_i)}$$

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

Other examples

- http://www.jposhea.org/projects/monte/

- More information on Monte Carlo:

http://arxiv.org/abs/hep-ph/0006269/

- A review of Monte Carlo Ray Tracing Methods:

http://www.cg.tuwien.ac.at/~balazs/PAPERS/CESCG97/mcrt.html

Matlab functions

- rand, randi, randn (normal)
- rng: Configure your random number generator!
- Quasi-random: haltonset, sobolset