Monte Carlo Integration for Image Synthesis

Thomas Funkhouser

Princeton University

C0S 526, Fall 2002

Main Sources

- Books
  - Realistic Ray Tracing, Peter Shirley
  - Realistic Image Synthesis Using Photon Mapping, Henrik Wann Jensen
- Theses
  - Robust Monte Carlo Methods for Light Transport Simulation, Eric Veach
  - Mathematical Models and Monte Carlo Methods for Physically Based Rendering, Eric La Fortune
- Course Notes
  - Mathematical Models for Computer Graphics, Stanford, Fall 1997
  - State of the Art in Monte Carlo Methods for Realistic Image Synthesis, Course 29, SIGGRAPH 2001

Outline

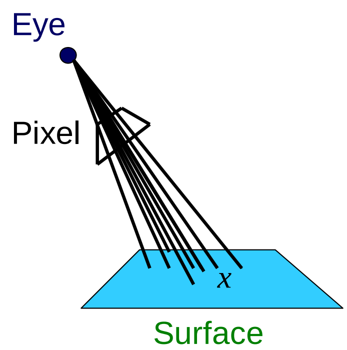
- Motivation
- Monte Carlo integration
- Monte Carlo path tracing
- Variance reduction techniques
- Sampling techniques
- Conclusion

Motivation

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

Motivation

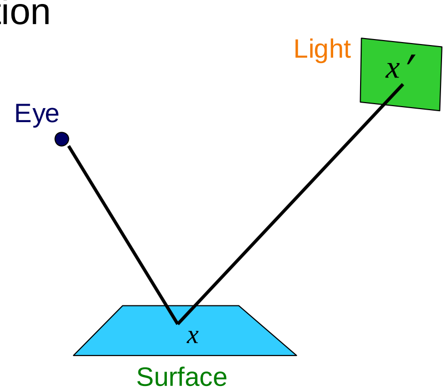
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



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

Motivation

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



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

Motivation

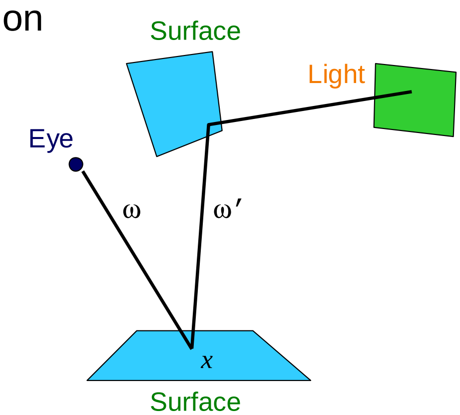
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Herf

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

Motivation

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



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

Motivation

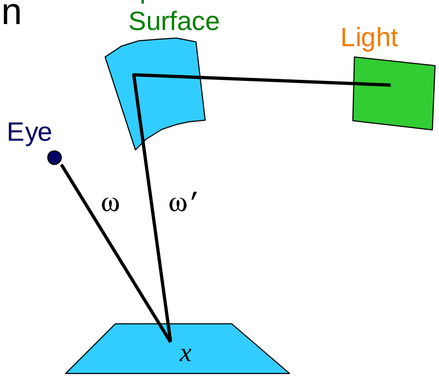
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Debevec

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

Motivation

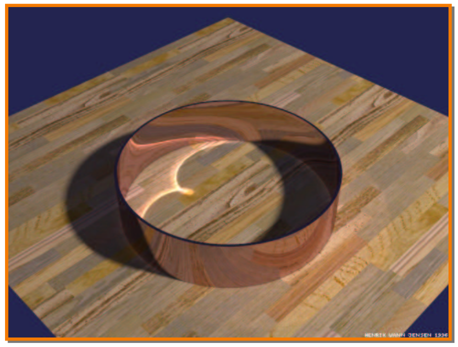
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



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

Motivation

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

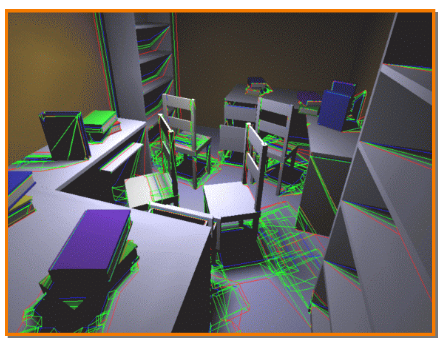


Jensen

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

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

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

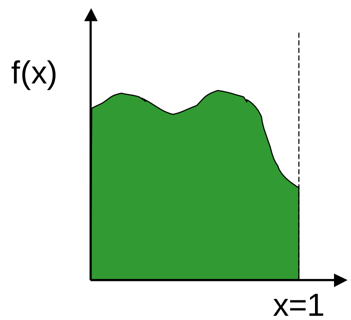
Outline

- Motivation
- Monte Carlo integration
- Monte Carlo path tracing
- Variance reduction techniques
- Sampling techniques
- Conclusion

Integration in 1D

Slide courtesy of
Peter Shirley

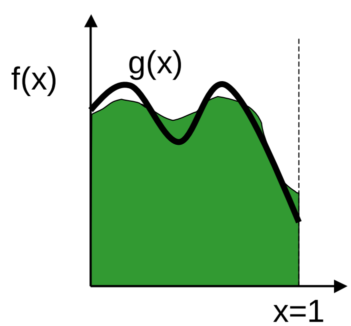
$$\int_0^1 f(x)dx = ?$$



We can approximate

Slide courtesy of
Peter Shirley

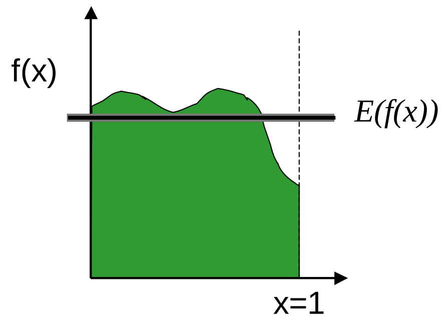
$$\int_0^1 f(x)dx = \int_0^1 g(x)dx$$



Or we can average

Slide courtesy of Peter Shirley

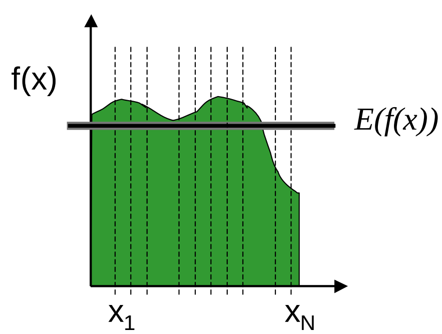
$$\int_0^1 f(x)dx = E(f(x))$$



Estimating the average

Slide courtesy of Peter Shirley

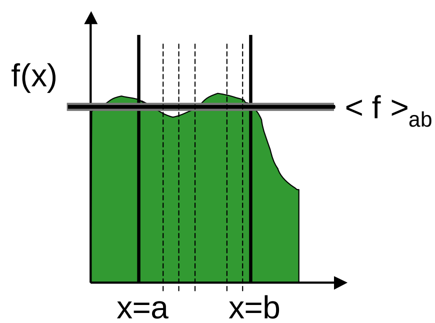
$$\int_0^1 f(x)dx = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$



Other Domains

Slide courtesy of Peter Shirley

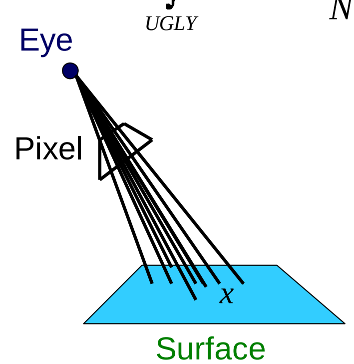
$$\int_a^b f(x)dx = \frac{b-a}{N}\sum_{i=1}^{N} f(x_i)$$



Multidimensional Domains

- Same ideas apply for integration over …
  - Pixel areas
  - Surfaces
  - Projected areas
  - Directions
  - Camera apertures
  - Time
  - Paths

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

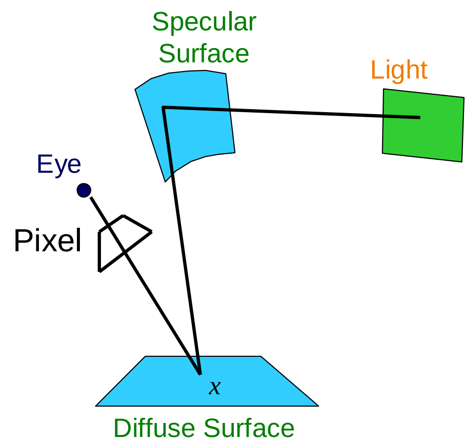


Outline

- Motivation
- Monte Carlo integration
- Monte Carlo path tracing
- Variance reduction techniques
- Sampling techniques
- Conclusion

Monte Carlo Path Tracing

- Integrate radiance for each pixel by sampling paths randomly



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

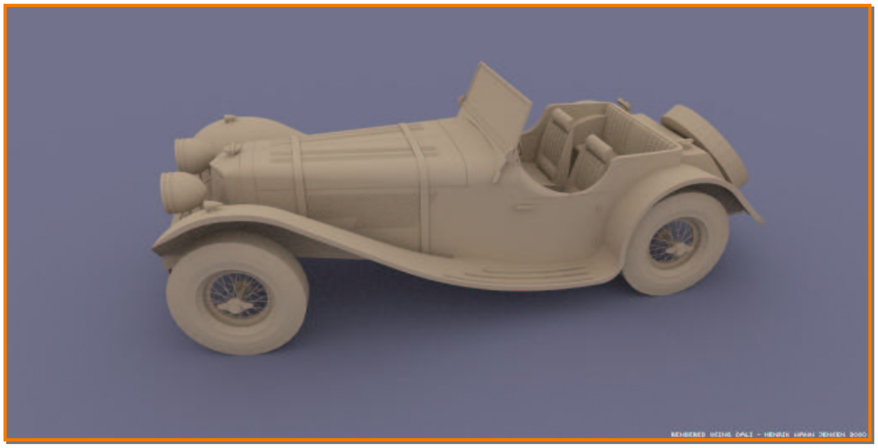
Simple Monte Carlo Path Tracer

- *Step 1: Choose a ray (x, y), (u, v), t; weight = 1*
- *Step 2: Trace ray to find intersection with nearest surface*
- *Step 3: Randomly decide whether to compute emitted or reflected light*
- *Step 3a: If emitted, return weight * Le*
- *Step 3b: If reflected,*
- *weight *= reflectance*
- *Generate ray in random direction*
- *Go to step 2*

Monte Carlo Path Tracing

- Advantages
  - Any type of geometry (procedural, curved, ...)
  - Any type of BRDF (specular, glossy, diffuse, ...)
  - Samples all types of paths (L(SD)*E)
  - Accuracy controlled at pixel level
  - Low memory consumption
  - Unbiased - error appears as noise in final image
- Disadvantages
  - Slow convergence
  - Noise in final image

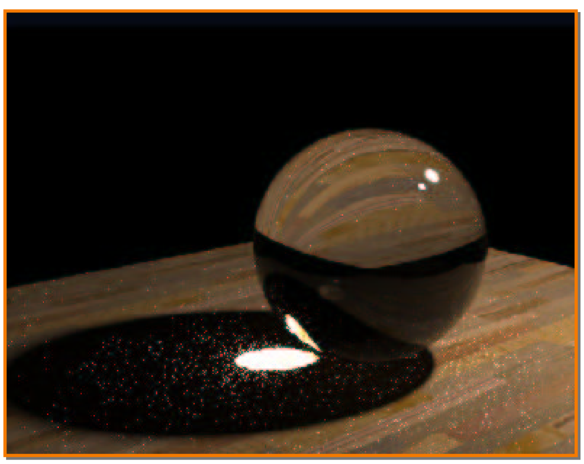
Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

Jensen

Monte Carlo Path Tracing

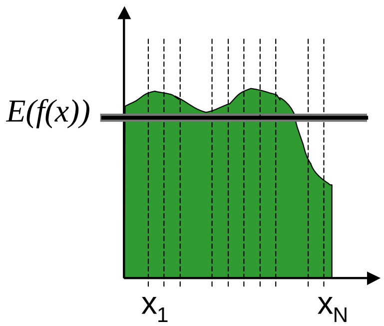


1000 paths/pixel

Jensen

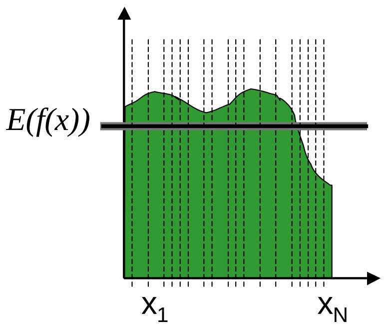
Variance

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



Variance

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



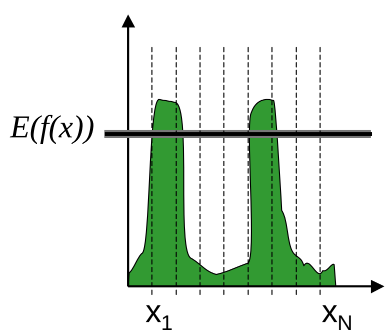
Variance decreases with 1/N
Error decreases with 1/sqrt(N)

Outline

- Motivation
- Monte Carlo integration
- Monte Carlo path tracing
- Variance reduction techniques
- Sampling techniques
- Conclusion

Variance

- Problem: variance decreases with 1/N



More samples removes noise SLOWLY

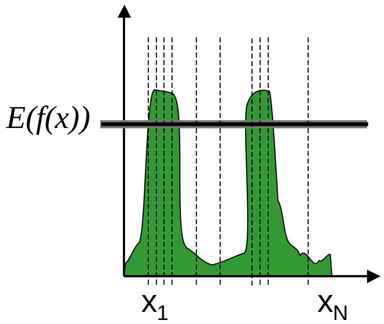
Variance Reduction Techniques

- Importance sampling
- Stratified sampling
- Metropolis sampling
- Quasi-random

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

Importance Sampling

- Put more samples where f(x) is bigger

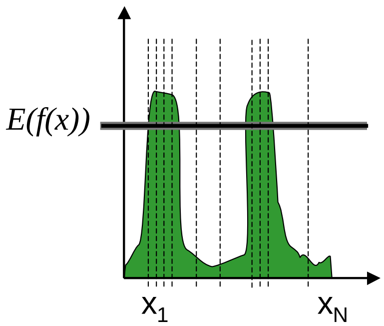


$$\int_\Omega f(x)dx = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

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

Importance Sampling

- This is still "unbiased"



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

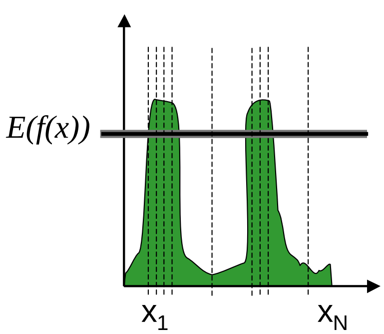
$$= \int_\Omega \frac{f(x)}{p(x)}\,p(x)dx$$

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

for all N

Importance Sampling

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



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

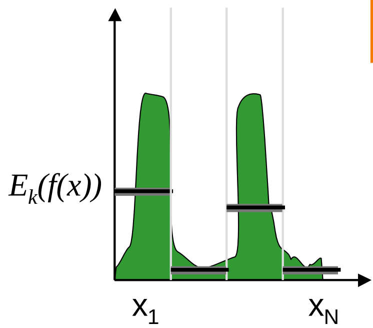
$$Y_i = \frac{f(x_i)}{p(x_i)} = \frac{1}{c}$$

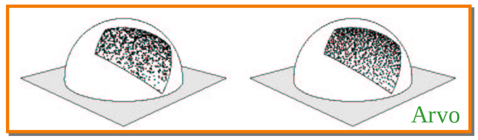
$$Var(Y) = 0$$

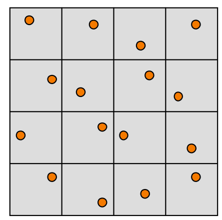
Less variance with better importance sampling

Stratified Sampling

- Estimate subdomains separately



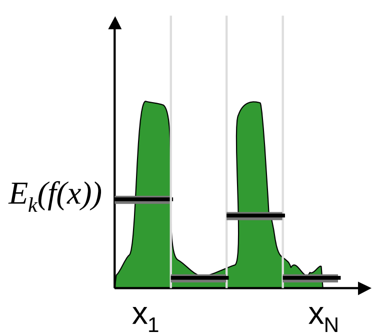




Stratified Sampling

- This is still unbiased

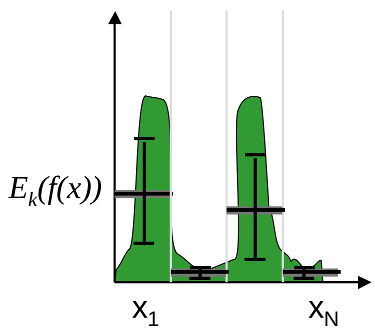
$$F_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$
$$= \frac{1}{N}\sum_{k=1}^{M} N_i F_i$$



Stratified Sampling

- Less overall variance if less variance in subdomains

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



Outline

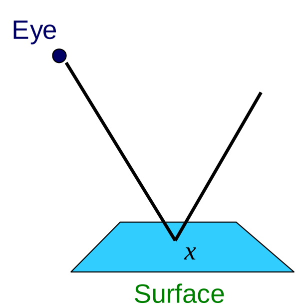
- Motivation
- Monte Carlo integration
- Monte Carlo path tracing
- Variance reduction techniques
- Sampling techniques
- Conclusion

Basic Monte Carlo Path Tracer

- ***Step 1: Choose a ray (x, y), (u, v), t***
- ***Step 2: Trace ray to find intersection with nearest surface***
- ***Step 3: Randomly decide whether to compute emitted or reflected light***
- ***Step 3a: If emitted, return weight * Le***
- ***Step 3b: If reflected,***
- ***weight *= reflectance***
- ***Generate ray in random direction***
- ***Go to step 2***

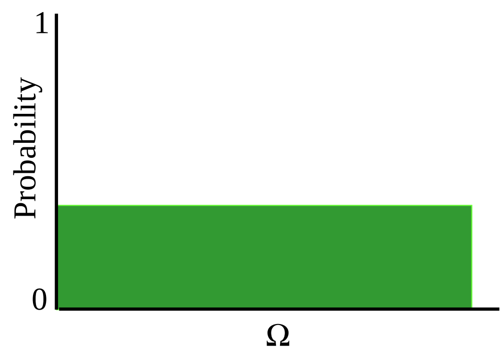
Sampling Techniques

- Problem: how do we generate random points/directions during path tracing?
  - Non-rectilinear domains
  - Importance (BRDF)
  - Stratified



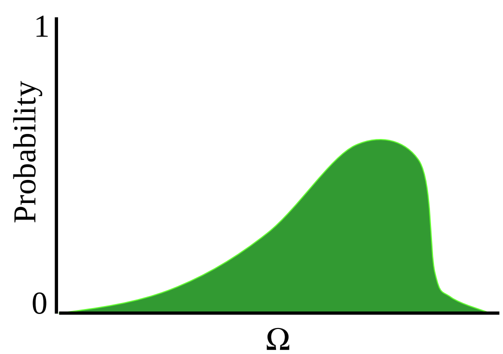
Generating Random Points

- Uniform distribution:
  - Use random number generator



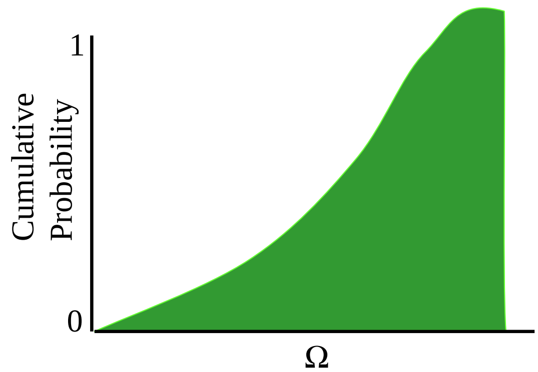
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



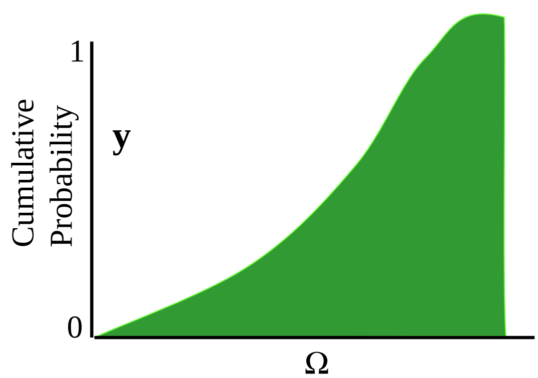
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



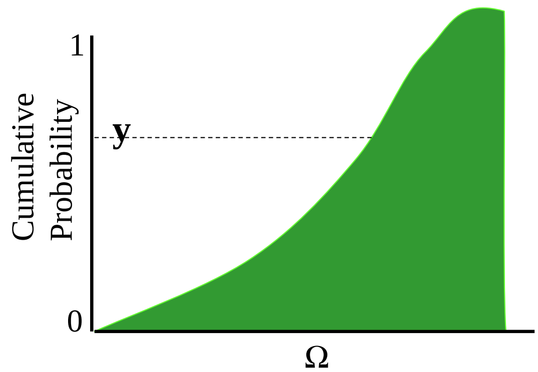
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



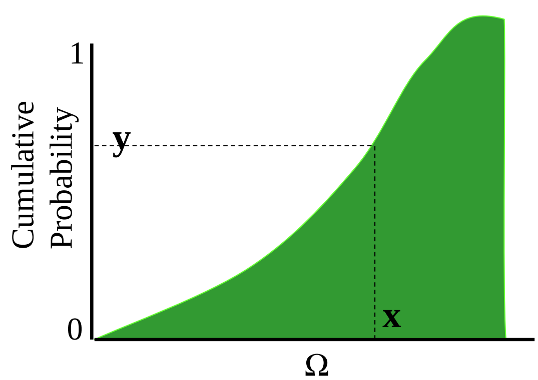
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



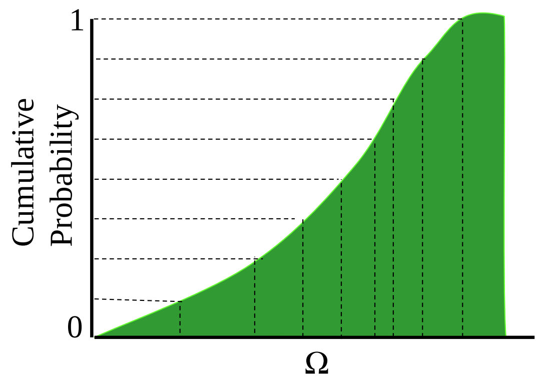
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



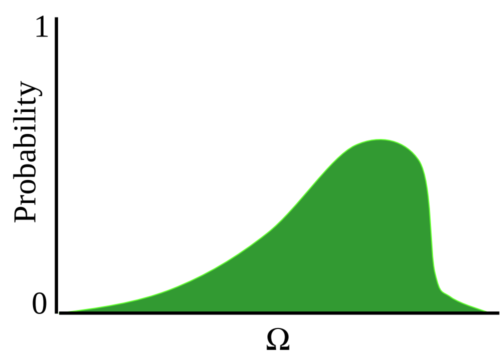
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



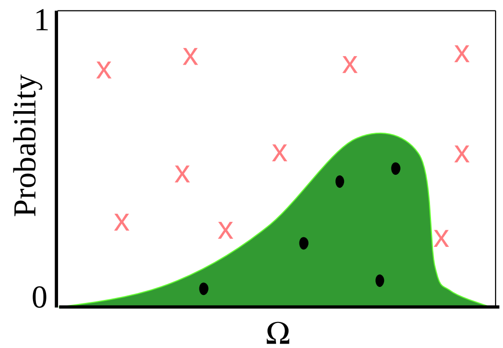
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



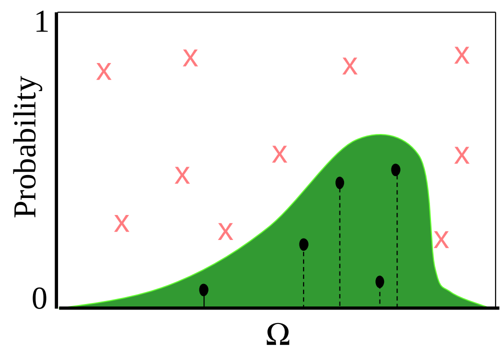
Generating Random Points

- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



Generating Random Points

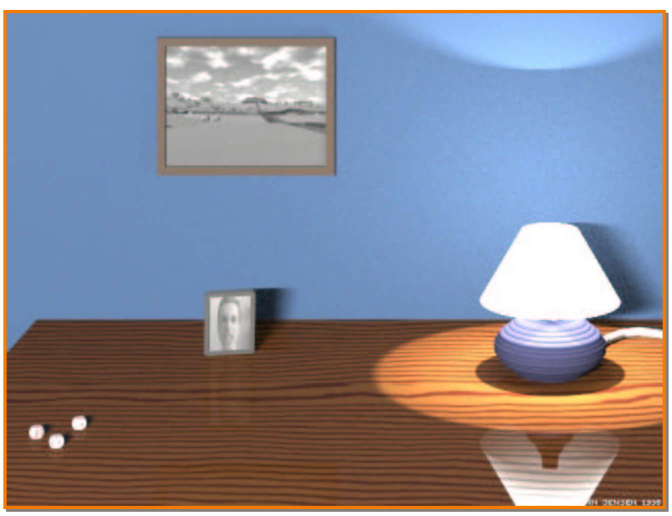
- Specific probability distribution:
  - Function inversion
  - Rejection
  - Metropolis



Combining Multiple PDFs

- Balance heuristic
  - Use combination of samples generated for each PDF
  - Number of samples for each PDF chosen by weights
  - Near optimal

Monte Carlo Path Tracing Image



2000 samples per pixel, 30 SGIs, 30 hours

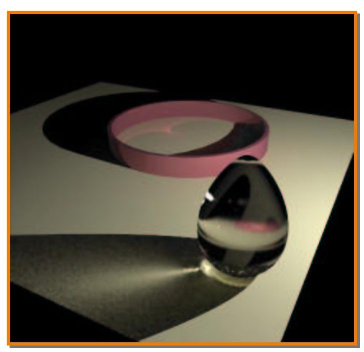
Jensen

Monte Carlo Extensions

- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching

Monte Carlo Extensions

- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching



RenderPark

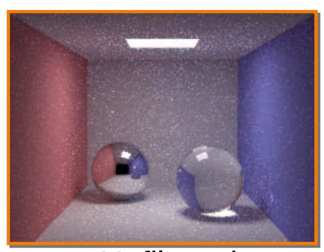
Monte Carlo Extensions

- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching

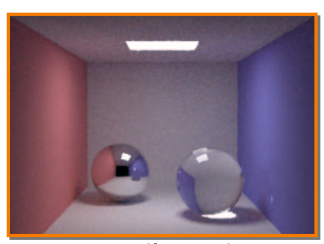
Heinrich

Monte Carlo Extensions

- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching



Unfiltered



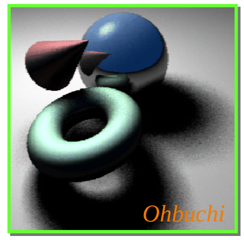
Filtered

Jensen

Monte Carlo Extensions

- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching

Fixed



Adaptive

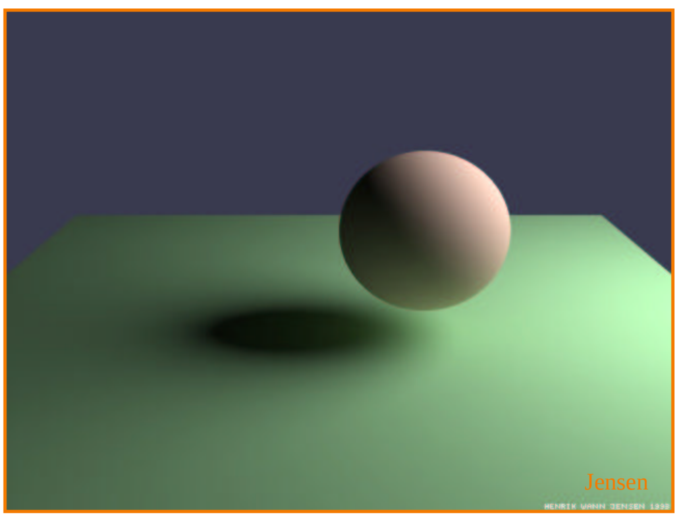
Monte Carlo Extensions

- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching

Summary

- Monte Carlo Integration Methods
  - Very general
  - Good for complex functions with high dimensionality
  - Converge slowly (but error appears as noise)
- Conclusion
  - Preferred method for difficult scenes
  - Noise removal (filtering) and
    irradiance caching (photon maps)
    used in practice

Programming Assignment #1



More Information

- Books
  - Realistic Ray Tracing, Peter Shirley
  - Realistic Image Synthesis Using Photon Mapping, Henrik Wann Jensen
- Theses
  - Robust Monte Carlo Methods for Light Transport Simulation, Eric Veach
  - Mathematical Models and Monte Carlo Methods for Physically Based Rendering, Eric La Fortune
- Course Notes
  - Mathematical Models for Computer Graphics, Stanford, Fall 1997
  - State of the Art in Monte Carlo Methods for Realistic Image Synthesis, Course 29, SIGGRAPH 2001