Monte Carlo Integration for Image Synthesis

COS 526, Fall 2010

Tom Funkhouser

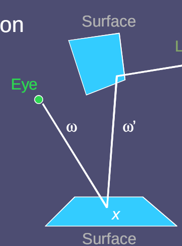
Slides from Rusinkiewicz, Shirley

Outline

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

Motivation

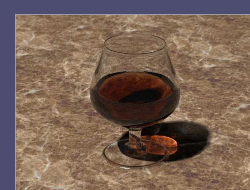
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



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

Challenge

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



Jensen

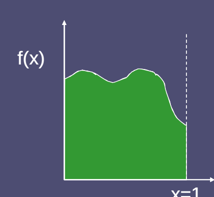
$$L(x,\vec{w}) = L_e(x, x \rightarrow e) + \int_S f_r(x, x' \rightarrow x, x \rightarrow e) L(x' \rightarrow x) V(x,x') G(x,x') dA$$

Outline

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

Integration in 1D

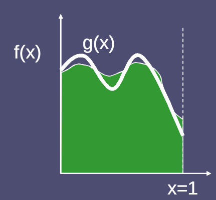
$$\int_0^1 f(x)dx = ?$$



Slide courtesy of Peter Shirley

We can approximate

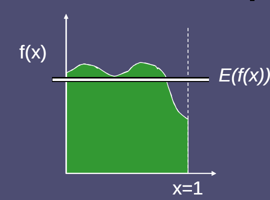
$$\int_0^1 f(x)\,dx = \int_0^1 g(x)\,dx$$



Slide courtesy of Peter Shirley

Or we can average

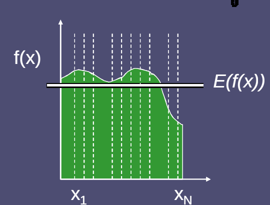
$$\int_0^1 f(x)\,dx = E(f(x))$$



Slide courtesy of Peter Shirley

Estimating the average

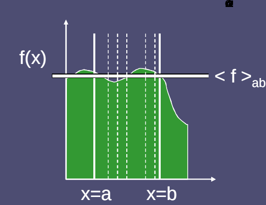
$$\int_0^1 f(x)\,dx = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$



Slide courtesy of Peter Shirley

Other Domains

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

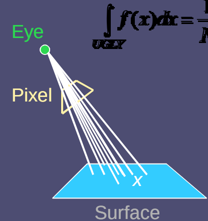


Slide courtesy of Peter Shirley

Multidimensional Domains

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

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

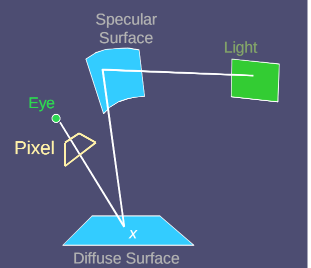


Outline

- 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

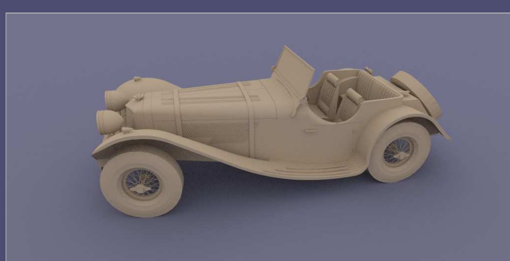


$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_{\Omega} f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}' \bullet \vec{n}) d\vec{w}$$

Simple Monte Carlo Path Tracer

- *Step 1:* *Choose a ray p=camera, d=(θ,ϕ); assign weight = 1*
- *Step 2:* *Trace ray to find intersection with nearest surface*
- *Step 3:* *Randomly choose between emitted and 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



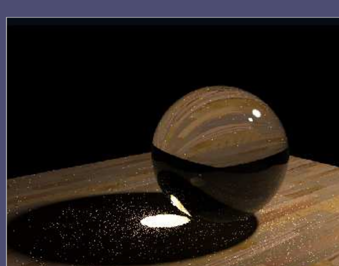
Big diffuse light source, 20 minutes

Jensen

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

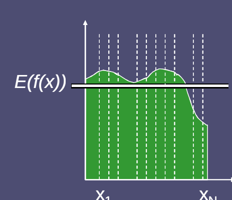


1000 paths/pixel

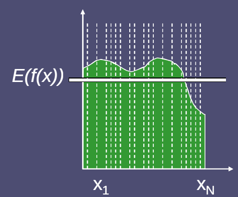
Jensen

Variance

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



Variance



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

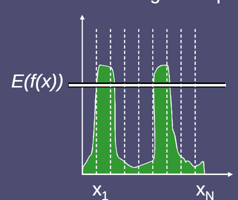
Variance decreases as 1/N
Error decreases as 1/sqrt(N)

Outline

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

Variance

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



Variance Reduction Techniques

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

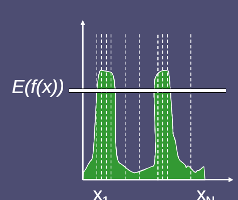
$$\int_0^1 f(x)dx = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

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

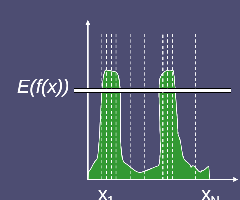
- This is still unbiased

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

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

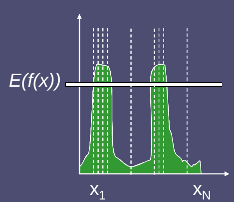
$$= \int_\Omega f(x)dx$$

for all N



Importance Sampling

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

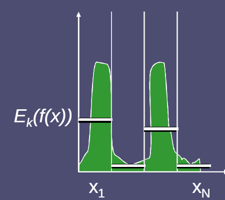


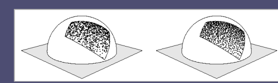
$$p(x) = cf(x)$$
$$Y_i = \frac{f(x_i)}{p(x_i)} = \frac{1}{c}$$
$$Var(Y) = 0$$

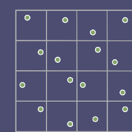
Less variance with better importance sampling

Stratified Sampling

- Estimate subdomains separately



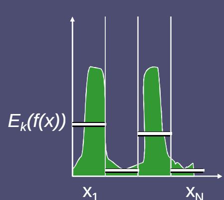




Stratified Sampling

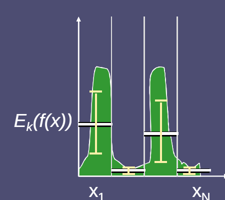
- This is still unbiased

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



Stratified Sampling

- Less overall variance if less variance in subdomains



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

Outline

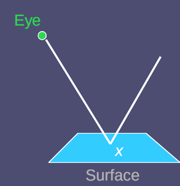
- Motivation
- Monte Carlo integration
- Monte Carlo path tracing
- Variance reduction techniques
- Sampling techniques
- Conclusion

Simple Monte Carlo Path Tracer

- *Step 1: Choose a ray (u,v,θ,φ); assign weight = 1*
- *Step 2: Trace ray to find intersection with nearest surface*
- *Step 3: Randomly choose between emitted and reflected light*
  - *Step 3a: If emitted,*
    *return weight * Le*
  - *Step 3b: If reflected,*
    *weight *= reflectance*
    Generate ray in random direction
    Go to step 2

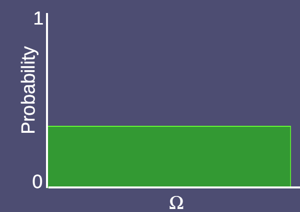
Sampling Techniques

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



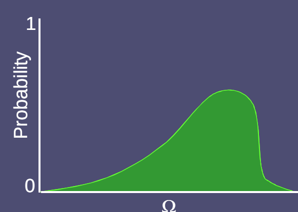
Generating Random Points

- Uniform distribution:
  - Use random number generator



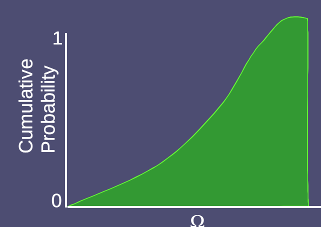
Generating Random Points

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



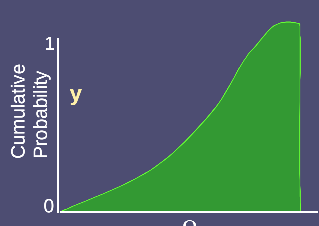
Generating Random Points

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



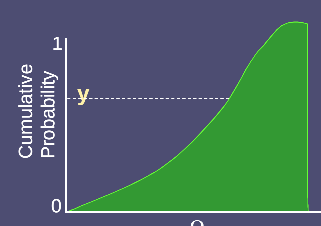
Generating Random Points

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



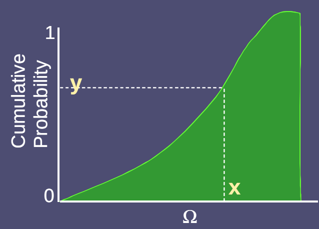
Generating Random Points

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



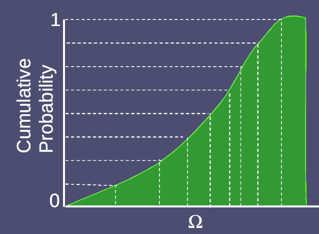
Generating Random Points

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



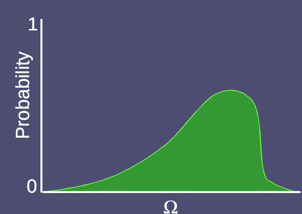
Generating Random Points

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



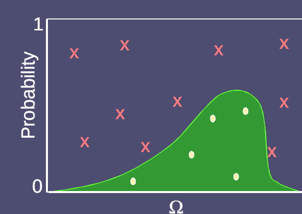
Generating Random Points

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



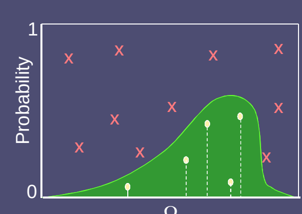
Generating Random Points

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



Generating Random Points

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



Combining Multiple PDFs

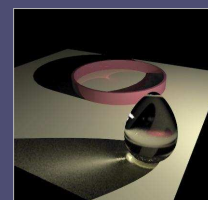
- Balance heuristic
  - Use combination of samples generated for each PDF
  - Number of samples for each PDF chosen by weights
  - Near optimal

Monte Carlo Extensions

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

Monte Carlo Extensions

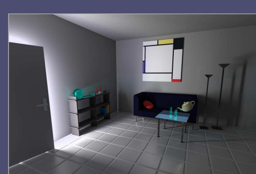
- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching



RenderPark

Monte Carlo Extensions

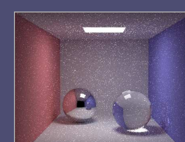
- Unbiased
  - Bidirectional path tracing
  - Metropolis light transport
- Biased, but consistent
  - Noise filtering
  - Adaptive sampling
  - Irradiance caching



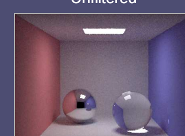
Heinrich

Monte Carlo Extensions

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



Unfiltered

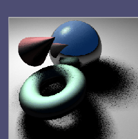


Filtered

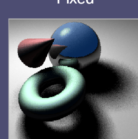
Jensen

Monte Carlo Extensions

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



Fixed



Adaptive

Ohbuchi

Monte Carlo Extensions

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

Jensen

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

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