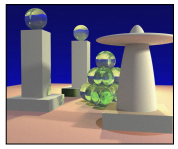
Advanced Computer Graphics (Fall 2009)

CS 283, Lecture 11: Monte Carlo Integration

Ravi Ramamoorthi

http://inst.eecs.berkeley.edu/~cs283


Acknowledgements and many slides courtesy:
Thomas Funkhouser, Szymon Rusinkiewicz and Pat Hanrahan

Motivation

Rendering = integration

- Reflectance equation: Integrate over incident illumination
- Rendering equation: Integral equation

Many sophisticated shading effects involve integrals

- Antialiasing
- Soft shadows
- Indirect illumination
- Caustics

Example: Soft Shadows

$$E(x) = \int_{H^2} L_i(x,\omega)\cos\theta\, d\omega$$

Challenges

- **Visibility and blockers**
- **Varying light distribution**
- **Complex source geometry**

Source: Agrawala. Ramamoorthi, Heirich, Moll, 2000

Monte Carlo

- Algorithms based on statistical sampling and random numbers
- Coined in the beginning of 1940s. Originally used for neutron transport, nuclear simulations
  - Von Neumann, Ulam, Metropolis, …
- Canonical example: 1D integral done numerically
  - Choose a set of random points to evaluate function, and then average (expectation or statistical average)

Monte Carlo Algorithms

Advantages

- Robust for complex integrals in computer graphics (irregular domains, shadow discontinuities and so on)
- Efficient for high dimensional integrals (common in graphics: time, light source directions, and so on)
- Quite simple to implement
- Work for general scenes, surfaces
- Easy to reason about (but care taken re statistical bias)

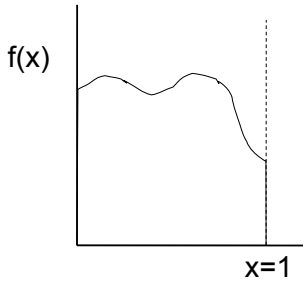
Disadvantages

- Noisy
- Slow (many samples needed for convergence)
- Not used if alternative analytic approaches exist (but those are rare)

Outline

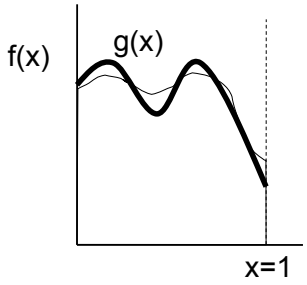
- Motivation
- *Overview, 1D integration*
- Basic probability and sampling
- Monte Carlo estimation of integrals

Integration in 1D



Slide courtesy of
Peter Shirley

We can approximate



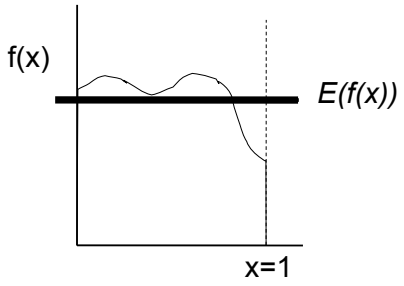
Standard integration methods like trapezoidal rule and Simpsons rule

Advantages:
- Converges fast for ***smooth*** integrands
- Deterministic

Disadvantages:
- Exponential complexity in many dimensions
- Not rapid convergence for discontinuities

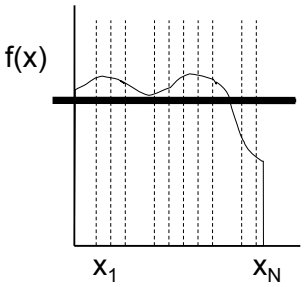
Slide courtesy of
Peter Shirley

Or we can average



Slide courtesy of
Peter Shirley

Estimating the average

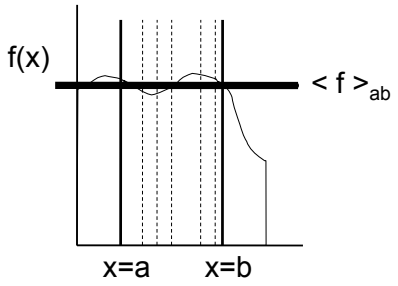


Monte Carlo methods (random choose samples)

Advantages:
- Robust for discontinuities
- Converges reasonably for large dimensions
- Can handle complex geometry, integrals
- Relatively simple to implement, reason about

Slide courtesy of
Peter Shirley

Other Domains

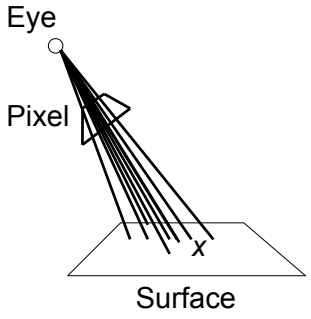


Slide courtesy of
Peter Shirley

Multidimensional Domains

Same ideas apply for integration over …

- Pixel areas
- Surfaces
- Projected areas
- Directions
- Camera apertures
- Time
- Paths



Outline

- Motivation
- Overview, 1D integration
- *Basic probability and sampling*
- Monte Carlo estimation of integrals

Random Variables

- Describes possible outcomes of an experiment
- In discrete case, e.g. value of a dice roll [x = 1-6]
- Probability p associated with each x (1/6 for dice)
- Continuous case is obvious extension

Expected Value

- Expectation

- For Dice example:

Continuous Probability Distributions

PDF $p(x)$

$$p(x) \geq 0$$

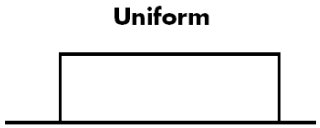
CDF $P(x)$

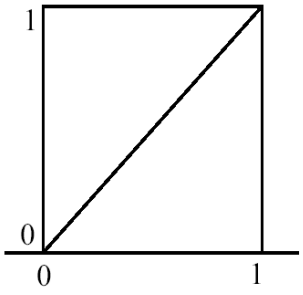
$$P(x) = \int_0^x p(x)dx$$

$$P(x) = \Pr(X < x) \qquad P(1) = 1$$

$$\Pr(\alpha \leq X \leq \beta) = \int_\alpha^\beta p(x)\,dx$$

$$= P(\beta) - P(\alpha)$$

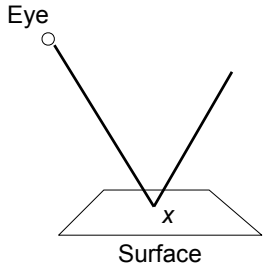




Sampling Techniques

Problem: how do we generate random points/ directions during path tracing?

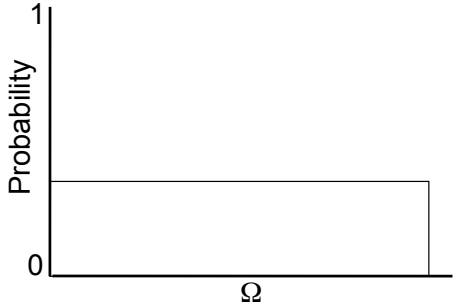
- Non-rectilinear domains
- Importance (BRDF)
- Stratified



Generating Random Points

Uniform distribution:

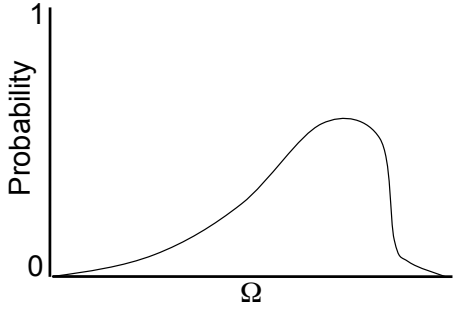
- Use random number generator



Generating Random Points

Specific probability distribution:

- Function inversion
- Rejection
- Metropolis



Common Operations

Want to ***sample*** probability distributions

- Draw samples distributed according to probability
- Useful for integration, picking important regions, etc.

Common distributions

- Disk or circle
- Uniform
- Upper hemisphere for visibility
- Area luminaire
- Complex lighting like an environment map
- Complex reflectance like a BRDF

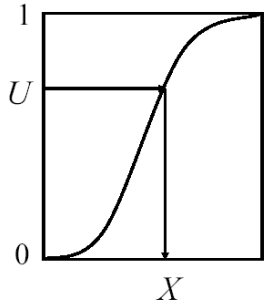
Sampling Continuous Distributions

Cumulative probability distribution function

$$P(x) = \Pr(X < x)$$

Construction of samples

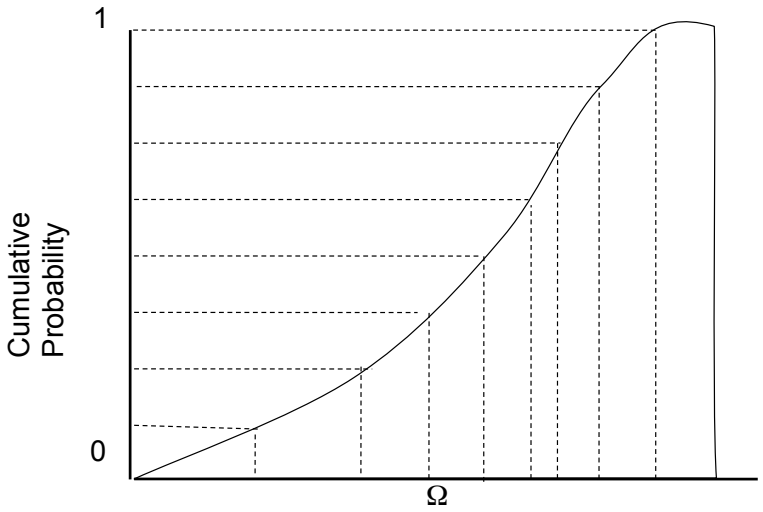
Solve for $X=P^{-1}(U)$



Must know:

1. **The integral of** $p(x)$
2. **The inverse function** $P^{-1}(x)$

Generating Random Points



Example: Power Function

Assume

$$p(x) = (n+1)x^n$$

$$\int_0^1 x^n dx = \frac{x^{n+1}}{n+1}\Big|_0^1 = \frac{1}{n+1}$$

$$P(x) = x^{n+1}$$

$$X \sim p(x) \Rightarrow X = P^{-1}(U) = \sqrt[n+1]{U}$$

Trick

$$Y = \max(U_1, U_2, \cdots, U_n, U_{n+1})$$

$$\Pr(Y < x) = \prod_{i=1}^{n+1} \Pr(U < x) = x^{n+1}$$

Sampling a Circle

Unit radius circle

$$A = \int_0^{2\pi}\int_0^1 r\, dr\, d\theta = \int_0^1 r\, dr \int_0^{2\pi} d\theta = \left(\frac{r^2}{2}\right)\Big|_0^1 \theta\Big|_0^{2\pi} = \pi$$

$$p(r,\theta)\, dr\, d\theta = \frac{1}{\pi} r\, dr\, d\theta \Rightarrow p(r,\theta) = \frac{r}{\pi}$$

$$p(r,\theta) = p(r)p(\theta)$$

$$p(\theta) = \frac{1}{2\pi}$$

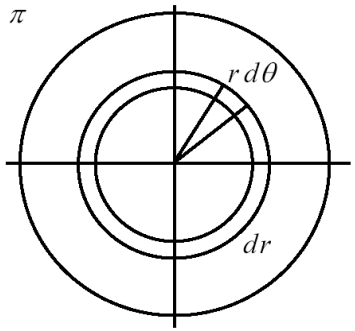
$$P(\theta) = \frac{1}{2\pi}\theta$$

$$\theta = 2\pi U_1$$

$$p(r) = 2r$$

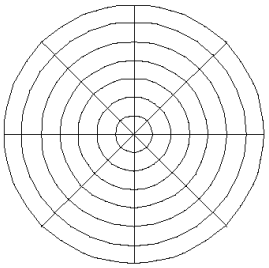
$$P(r) = r^2$$

$$r = \sqrt{U_2}$$



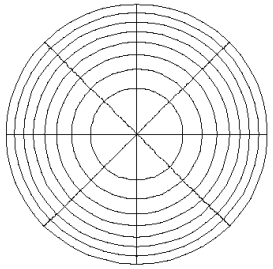
Sampling a Circle

WRONG ≠ Equi-Areal



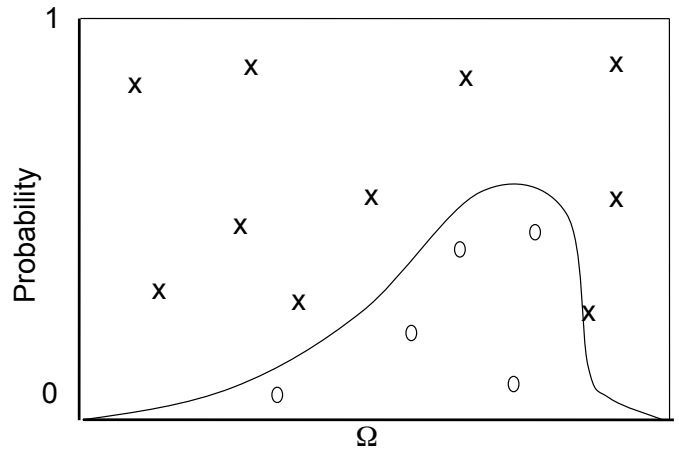
$$\theta = 2\pi U_1$$
$$r = U_2$$

RIGHT = Equi-Areal



$$\theta = 2\pi U_1$$
$$r = \sqrt{U_2}$$

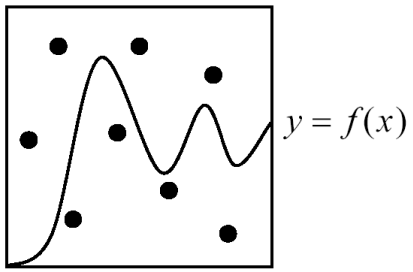
Rejection Sampling



Rejection Methods

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

$$= \iint_{y<f(x)} dx\,dy$$



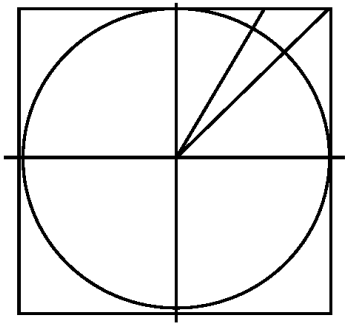
Algorithm

Pick U_1 **and** U_2

Accept U_1 **if** $U_2 < f(U_1)$

Wasteful? Efficiency = Area / Area of rectangle

Sampling a Circle: Rejection



```
do {
   X=1-2*U1
   Y=1-2*U2
while( X2+ Y2 >1 )
```

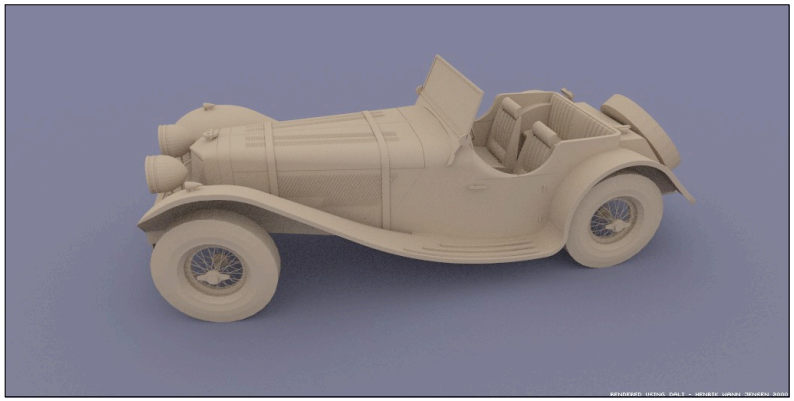
May be used to pick random 2D directions

Circle techniques may also be applied to the sphere

Outline

- Motivation
- Overview, 1D integration
- Basic probability and sampling
- *Monte Carlo estimation of integrals*

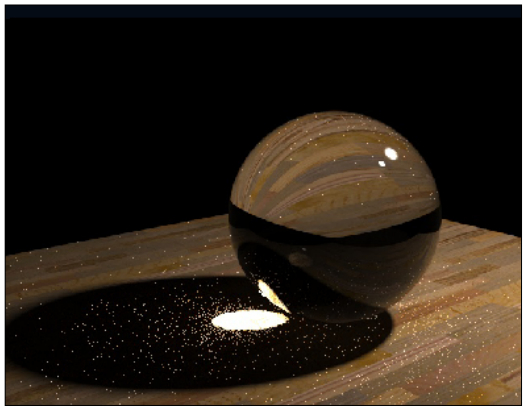
Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

Motivation for rendering in graphics: Covered in detail in next lecture

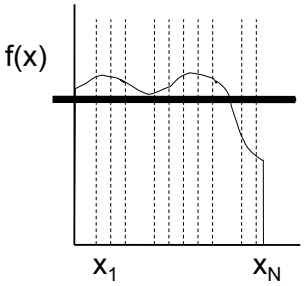
Monte Carlo Path Tracing



1000 paths/pixel

Jensen

Estimating the average



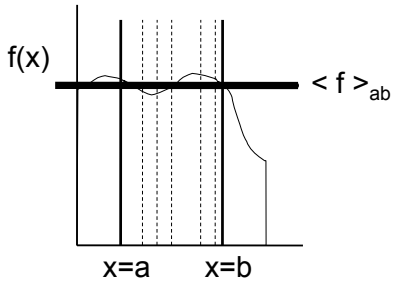
Monte Carlo methods (random choose samples)

Advantages:

- Robust for discontinuities
- Converges reasonably for large dimensions
- Can handle complex geometry, integrals
- Relatively simple to implement, reason about

Slide courtesy of Peter Shirley

Other Domains



Slide courtesy of
Peter Shirley

More formally

Definite integral $I(f) \equiv \int_0^1 f(x)\,dx$

Expectation of f $E[f] \equiv \int_0^1 f(x)\,p(x)\,dx$

Random variables $X_i \sim p(x)$

$Y_i = f(X_i)$

Estimator $F_N = \frac{1}{N}\sum_{i=1}^{N} Y_i$

Unbiased Estimator

$$E[F_N] = I(f)$$

$$E[F_N] = E[\frac{1}{N}\sum_{i=1}^{N} Y_i]$$

$$= \frac{1}{N}\sum_{i=1}^{N} E[Y_i] = \frac{1}{N}\sum_{i=1}^{N} E[f(X_i)]$$

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

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

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

Properties

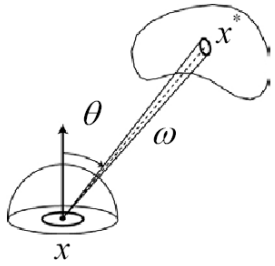
$$E[\sum_i Y_i] = \sum_i E[Y_i]$$

$$E[aY] = aE[Y]$$

Assume uniform probability distribution for now

Direct Lighting – Directional Sampling

$$E(x) = \int_\Omega L(x,\omega)\cos\theta\, d\omega$$



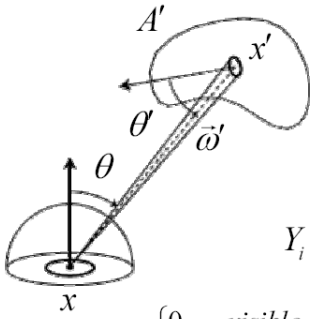
Ray intersection $x^*(x,\omega)$

Sample ω **uniformly by** Ω

$$Y_i = L(x^*(x,\omega_i), -\omega_i)\cos\theta\, 2\pi$$

Direct Lighting – Area Sampling

$$E(x) = \int_{\Omega} L_i(x,\omega)\cos\theta\, d\omega = \int_{A'} L_o(x',\omega')V(x,x')\frac{\cos\theta\cos\theta'}{|x-x'|^2}dA'$$



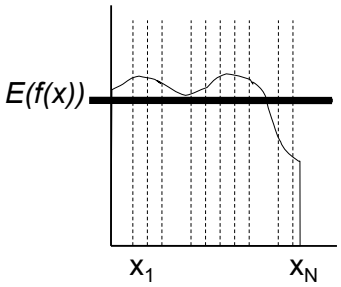
Ray direction $\omega' = x - x'$

Sample x' uniformly by A'

$$Y_i = L_o(x_i',\omega_i')\,V(x,x_i')\frac{\cos\theta\,\cos\theta_i'}{|x-x_i'|^2}A$$

$$V(x,x') = \begin{cases} 0 & \neg visible \\ 1 & visible \end{cases}$$

Variance



Variance

Definition

$$V[Y] \equiv E[(Y - E[Y])^2]$$
$$= E[Y^2 - 2YE[Y] + E[Y]^2]$$
$$= E[Y^2] - E[Y]^2$$

Properties

$$V[\sum_i Y_i] = \sum_i V[Y_i]$$

$$V[aY] = a^2 V[Y]$$

Variance decreases with sample size

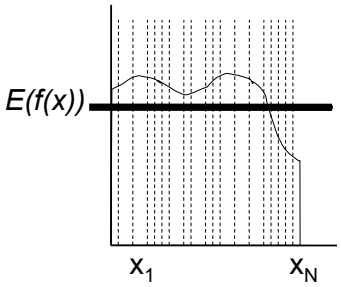
$$V[\frac{1}{N}\sum_{i=1}^{N} Y_i] = \frac{1}{N^2}\sum_{i=1}^{N} V[Y_i] = \frac{1}{N} V[Y]$$

Variance for Dice Example?

- Work out on board (variance for single dice roll)

Exercise for students at home.

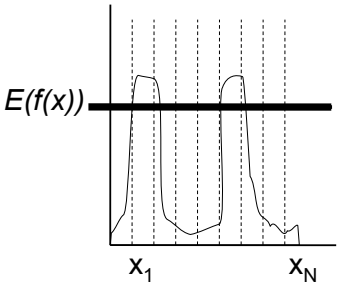
Variance



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

Efficiency measure

$$Efficiency \propto \frac{1}{Variance \bullet Cost}$$

Techniques

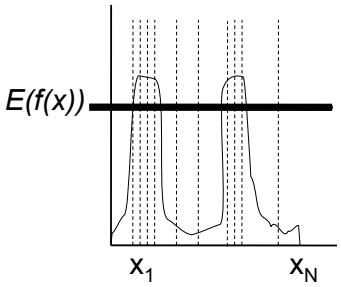
- **Importance sampling**
- **Sampling patterns: stratified, ...**

Variance Reduction Techniques

- Importance sampling
- Stratified sampling

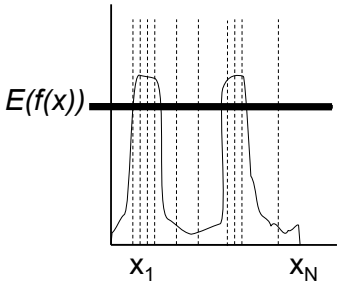
Importance Sampling

Put more samples where f(x) is bigger



Importance Sampling

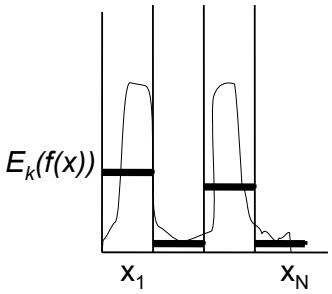
- This is still unbiased

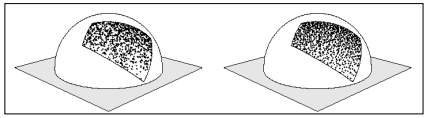


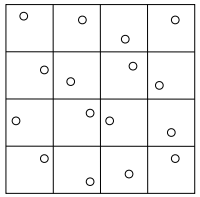
for all N

Stratified Sampling

- Estimate subdomains separately

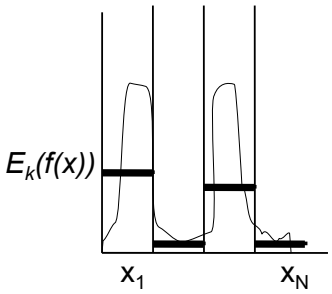






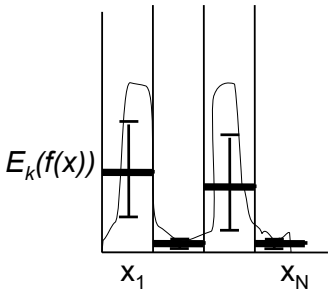
Stratified Sampling

- This is still unbiased



Stratified Sampling

- Less overall variance if less variance in subdomains



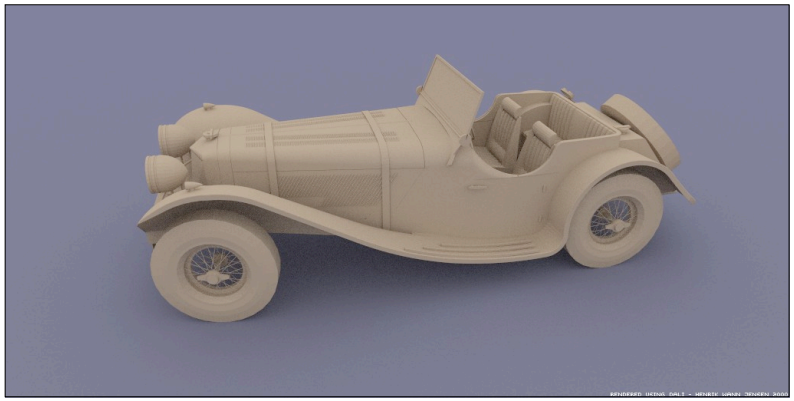
More Information

- Veach PhD thesis chapter (linked to from website)
- Course Notes (links from website)
  - *Mathematical Models for Computer Graphics*, Stanford, Fall 1997
  - *State of the Art in Monte Carlo Methods for Realistic Image Synthesis*, Course 29, SIGGRAPH 2001

Motivation

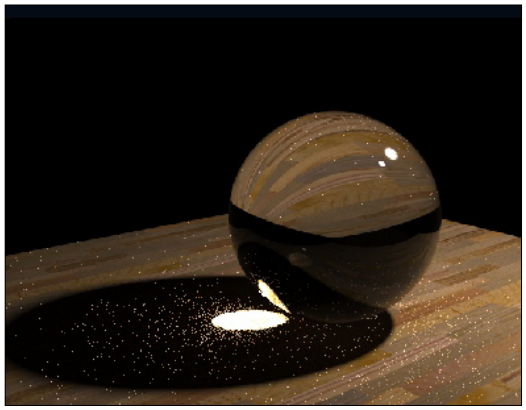
- General solution to rendering and global illumination
- Suitable for a variety of general scenes
- Based on Monte Carlo methods
- Enumerate all paths of light transport

Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

Monte Carlo Path Tracing



1000 paths/pixel

Jensen

Monte Carlo Path Tracing

Advantages

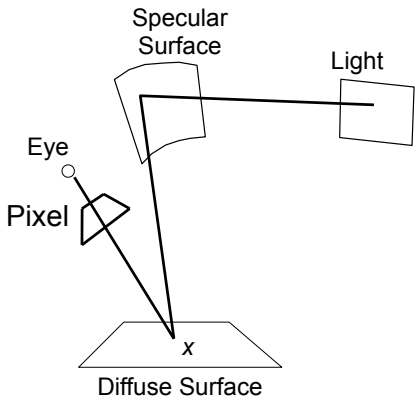
- Any type of geometry (procedural, curved, ...)
- Any type of BRDF (specular, glossy, diffuse, ...)
- Samples all types of paths (L(SD)*E)
- Accuracy controlled at pixel level
- Low memory consumption
- Unbiased - error appears as noise in final image

Disadvantages (standard Monte Carlo problems)

- Slow convergence (square root of number of samples)
- Noise in final image

Monte Carlo Path Tracing

Integrate radiance
for each pixel
by sampling paths
randomly



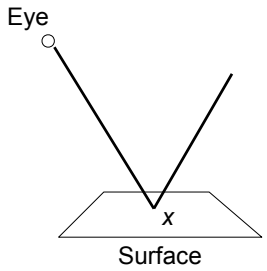
Simple Monte Carlo Path Tracer

- *Step 1: Choose a ray (u,v,θ,ϕ) [per pixel]; 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 and reduce variance?

- Importance sampling (e.g. by BRDF)
- Stratified sampling



Outline

- Motivation and Basic Idea
- *Implementation of simple path tracer*
- Variance Reduction: Importance sampling
- Other variance reduction methods
- Specific 2D sampling techniques

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average over paths

- Choose a ray with *p*=camera, *d*=(θ,ϕ) within pixel
- Pixel color += (1/n) * TracePath(*p*, *d*)

TracePath(*p*, *d*) returns (r,g,b) [and calls itself recursively]:

- Trace ray (*p*, *d*) to find nearest intersection *p'*
- Select with probability (say) 50%:
  - Emitted:
    return 2 * (Le_{red}, Le_{green}, Le_{blue}) // 2 = 1/(50%)
  - Reflected:
    generate ray in random direction *d'*
    return 2 * $f_r(d \rightarrow d') * (n \cdot d')$ * TracePath(*p'*, *d'*)

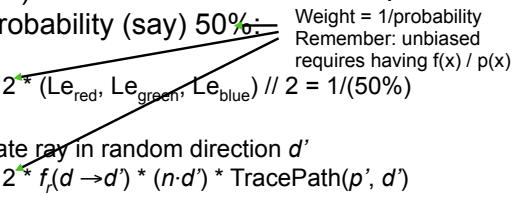
Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with *p*=camera, *d*=(θ,ϕ) within pixel
- Pixel color += (1/n) * TracePath(*p*, *d*)

TracePath(*p*, *d*) returns (r,g,b) [and calls itself recursively]:

- Trace ray (*p*, *d*) to find nearest intersection *p'*
- Select with probability (say) 50%:
  - Emitted:
    return 2 * (Le_{red}, Le_{green}, Le_{blue}) // 2 = 1/(50%)
  - Reflected:
    generate ray in random direction *d'*
    return 2 * $f_r(d \rightarrow d') * (n \cdot d')$ * TracePath(*p'*, *d'*)



Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

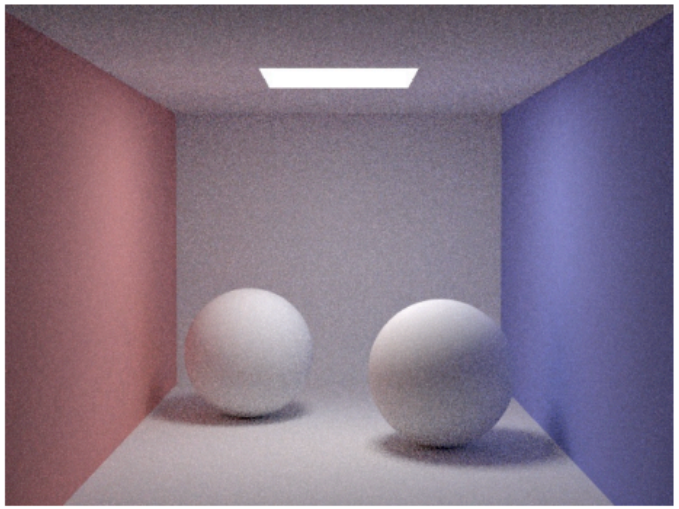
- Choose a ray with *p*=camera, *d*=(θ,ϕ) within pixel
- Pixel color += (1/n) * TracePath(*p*, *d*)

TracePath(*p*, *d*) returns (r,g,b) [and calls itself recursively]:

- Trace ray (*p*, *d*) to find nearest intersection *p'*
- Select with probability (say) 50%:
  - Emitted:
    return 2 * (Le_{red}, Le_{green}, Le_{blue}) // 2 = 1/(50%)
  - Reflected:
    generate ray in random direction *d'*
    return 2 * $f_r(d \rightarrow d')$ * $(n \cdot d')$ * TracePath(*p'*, *d'*)

Path terminated when Emission evaluated

Path Tracing



10 paths / pixel

Arnold Renderer (M. Fajardo)

- Works well diffuse surfaces, hemispherical light

From CS 283(294) last year



Daniel Ritchie and Lita Cho

Advantages and Drawbacks

- Advantage: general scenes, reflectance, so on
  - By contrast, standard recursive ray tracing only mirrors
- This algorithm is *unbiased*, but horribly inefficient
  - Sample “emitted” 50% of the time, even if emitted=0
  - Reflect rays in random directions, even if mirror
  - If light source is small, rarely hit it
- Goal: improve efficiency without introducing bias
  - Variance reduction using many of the methods discussed for Monte Carlo integration last week
  - Subject of much interest in graphics in 90s till today

Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- *Variance Reduction: Importance sampling*
- Other variance reduction methods
- Specific 2D sampling techniques

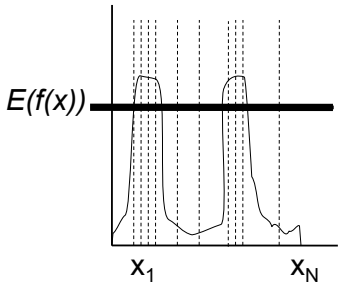
Importance Sampling

- Pick paths based on energy or expected contribution
  - More samples for high-energy paths
  - Don't pick low-energy paths
- At "macro" level, use to select between reflected vs emitted, or in casting more rays toward light sources
- At "micro" level, importance sample the BRDF to pick ray directions
- Tons of papers in 90s on tricks to reduce variance in Monte Carlo rendering

Importance Sampling

Can pick paths however we want, but contribution weighted by 1/probability

- Already seen this division of 1/prob in weights to emission, reflectance



Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with *p*=camera, *d*=(θ,ϕ) within pixel
- Pixel color += (1/n) * TracePath(*p*, *d*)

TracePath(*p*, *d*) returns (r,g,b) [and calls itself recursively]:

- Trace ray (*p*, *d*) to find nearest intersection *p'*
- Select with probability (say) 50%:
  - Emitted:
    return 2 * (Le_{red}, Le_{green}, Le_{blue}) // 2 = 1/(50%)
  - Reflected:
    generate ray in random direction *d'*
    return 2 * $f_r(d \rightarrow d') * (n \cdot d')$ * TracePath(*p'*, *d'*)

Importance sample Emit vs Reflect

TracePath(*p*, *d*) returns (r,g,b) [and calls itself recursively]:

- Trace ray (*p*, *d*) to find nearest intersection *p'*
- If Le = (0,0,0) then p_{emit}= 0 else p_{emit}= 0.9 (say)
- If random() < p_{emit} then:
  - Emitted:
    return (1/ p_{emit}) * (Le_{red}, Le_{green}, Le_{blue})
  - Else Reflected:
    generate ray in random direction *d'*
    return (1/(1- p_{emit})) * $f_r(d \rightarrow d') * (n \cdot d')$ * TracePath(*p'*, *d'*)

Importance sample Emit vs Reflect

TracePath(p, d) returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- If Le = (0,0,0) then p_{emit}= 0 else p_{emit}= 0.9 (say)
- If random() < p_{emit} then:
  - Emitted:
    return (1/ p_{emit}) * (Le_{red}, Le_{green}, Le_{blue})
  - Else Reflected:
    generate ray in random direction d'
    return (1/(1- p_{emit})) * $f_r(d \rightarrow d')$ * $(n \cdot d')$ * TracePath(p', d')

Can never be 1 unless Reflectance is 0

Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- Variance Reduction: Importance sampling
- *Other variance reduction methods*
- Specific 2D sampling techniques

More variance reduction

- Discussed “macro” importance sampling
  - Emitted vs reflected
- How about “micro” importance sampling
  - *Shoot rays towards light sources in scene*
  - Distribute rays according to BRDF

One Variation for Reflected Ray

- Pick a light source
- Trace a ray towards that light
- Trace a ray anywhere except for that light
  - Rejection sampling
- Divide by probabilities
  - 1/(solid angle of light) for ray to light source
  - (1 – the above) for non-light ray
  - Extra factor of 2 because shooting 2 rays

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 * Lr(light sources)
   weight *= reflectance(r)
   Choose new ray r' ~ BRDF pdf(r)
       Go to Step 2.
```

Monte Carlo Extensions

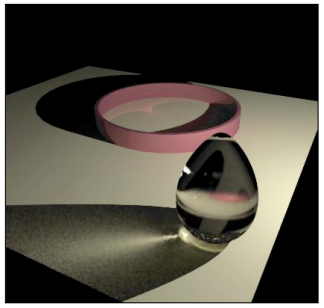
Unbiased
- Bidirectional path tracing
- Metropolis light transport

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

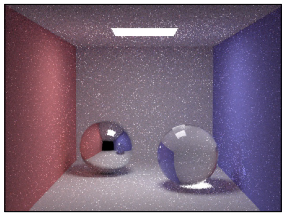
Biased, but consistent
- Noise filtering
- Adaptive sampling
- Irradiance caching

Heinrich

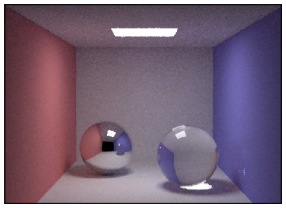
Monte Carlo Extensions

Unbiased
- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent
- Noise filtering
- Adaptive sampling
- Irradiance caching



Unfiltered



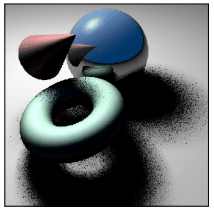
Filtered

Jensen

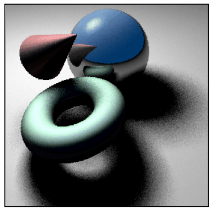
Monte Carlo Extensions

Unbiased
- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent
- Noise filtering
- Adaptive sampling
- Irradiance caching



Fixed



Adaptive

Ohbuchi

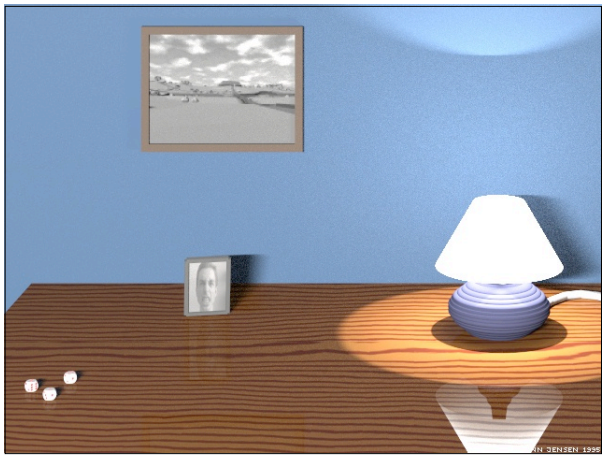
Monte Carlo Extensions

Unbiased
- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent
- Noise filtering
- Adaptive sampling
- Irradiance caching

Jensen

Monte Carlo Path Tracing Image



2000 samples per pixel, 30 computers, 30 hours Jensen

Outline

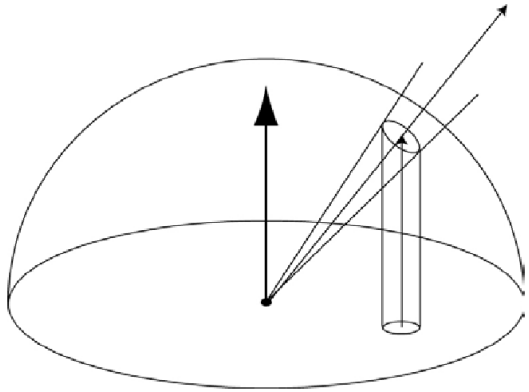
- Motivation and Basic Idea
- Implementation of simple path tracer
- Variance Reduction: Importance sampling
- Other variance reduction methods
- *Specific 2D sampling techniques*

2D Sampling: Motivation

- Final step in sending reflected ray: sample 2D domain
- According to projected solid angle
- Or BRDF
- Or area on light source
- Or sampling of a triangle on geometry
- Etc.

Sampling Projected Solid Angle

Generate cosine weighted distribution



Sampling Upper Hemisphere

- Uniform directional sampling: how to generate random ray on a hemisphere?
- Option #1: rejection sampling
  - Generate random numbers (x,y,z), with x,y,z in –1..1
  - If $x^2+y^2+z^2 > 1$, reject
  - Normalize (x,y,z)
  - If pointing into surface (ray dot n < 0), flip

Sampling Upper Hemisphere

- Option #2: inversion method
  - In polar coords, density must be proportional to sin θ((remember *d*(solid angle) = sin θ *d*θ *d*ϕ)
  - Integrate, invert $\rightarrow$ $\cos^{-1}$
- So, recipe is
  - Generate ϕ in 0..2π
  - Generate *z* in 0..1
  - Let $\theta = \cos^{-1} z$
  - (*x*,*y*,*z*) = (sin θ cos ϕ, sin θ sin ϕ, cos θ)

BRDF Importance Sampling

- Better than uniform sampling: importance sampling
- Because you divide by probability, ideally probability $\propto f_r * \cos \theta_i$

BRDF Importance Sampling

- For cosine-weighted Lambertian:
  - Density = $\cos \theta \sin \theta$
  - Integrate, invert $\rightarrow \cos^{-1}(\text{sqrt})$
- So, recipe is:
  - Generate ϕ in $0..2\pi$
  - Generate z in $0..1$
  - Let $\theta = \cos^{-1}(\text{sqrt}(z))$

BRDF Importance Sampling

- Phong BRDF: $f_r \propto \cos^n\alpha$ where α is angle between outgoing ray and ideal mirror direction
- Constant scale = $k_s(n+2)/(2\pi)$
- Can't sample this times cos θ_i
  - Can only sample BRDF itself, then multiply by cos θ_i
  - That's OK – still better than random sampling

BRDF Importance Sampling

- Recipe for sampling specular term:
  - Generate *z* in 0..1
  - Let $\alpha = \cos^{-1}(z^{1/(n+1)})$
  - Generate ϕ_α in 0..2π
- This gives direction w.r.t. ideal mirror direction
- Convert to (x,y,z), then rotate such that *z* points along mirror dir.

Summary

- Monte Carlo methods robust and simple (at least until nitty gritty details) for global illumination
- Must handle many variance reduction methods in practice
- Importance sampling, Bidirectional path tracing, Russian roulette etc.
- Rich field with many papers, systems researched over last 10 years