Monte Carlo Path Tracing

COS 526, Fall 2016

Tom Funkhouser

Slides from Rusinkiewicz, Shirley

Outline

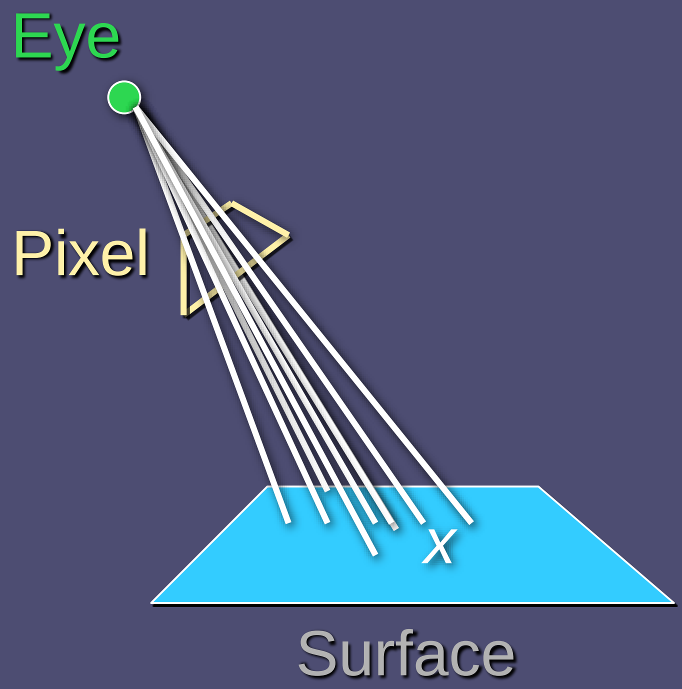
- Motivation
- Monte Carlo integration
- Variance reduction techniques
- Monte Carlo path tracing
- Sampling techniques
- Conclusion

Motivation

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

Motivation

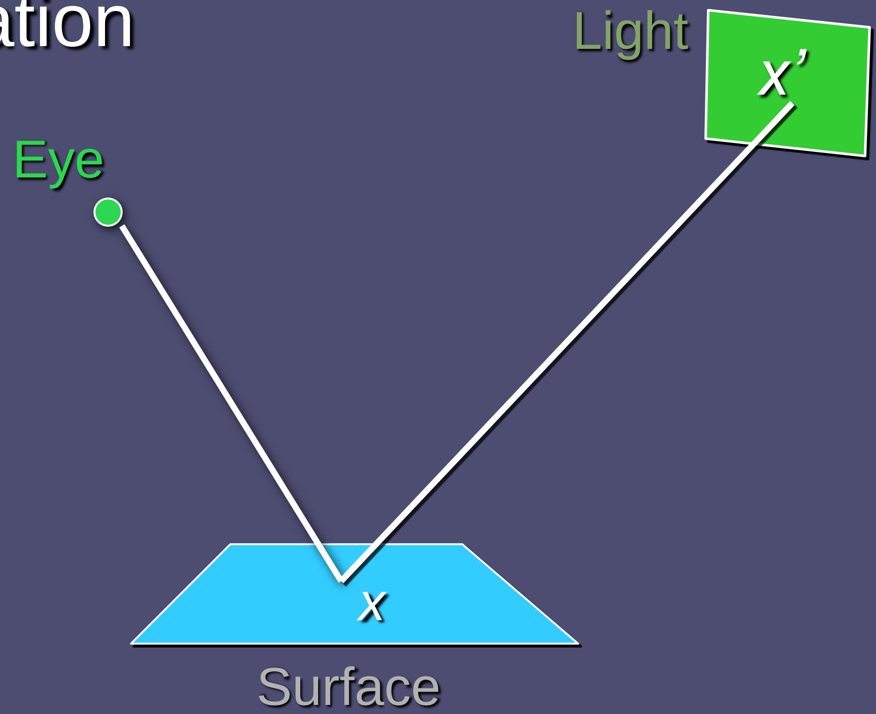
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



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

Motivation

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



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

Motivation

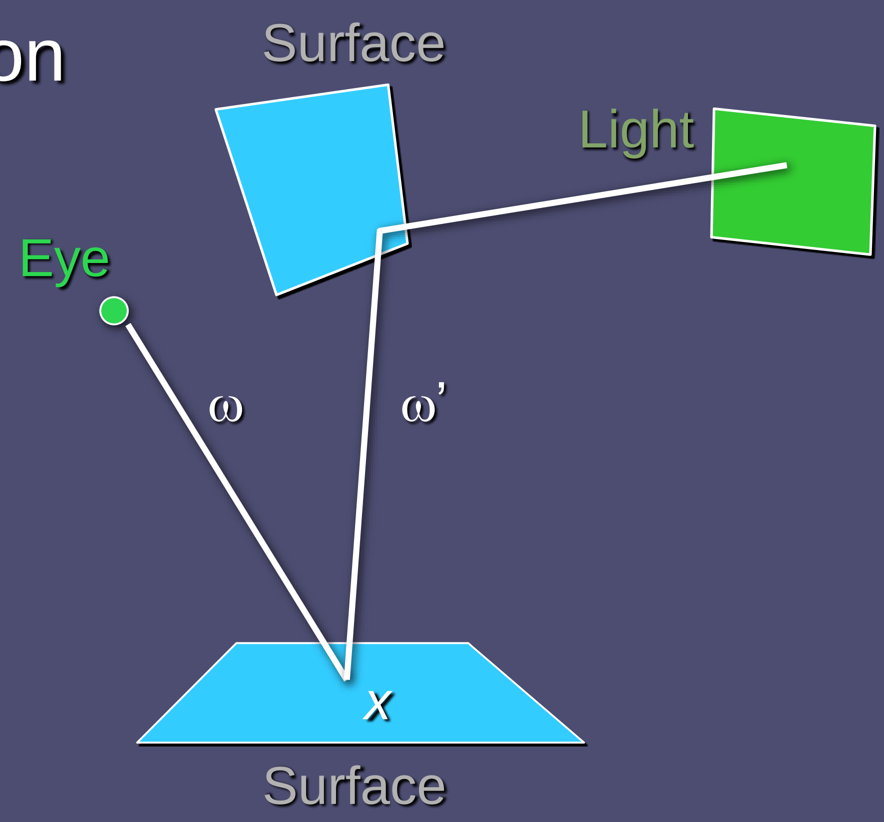
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Herf

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

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

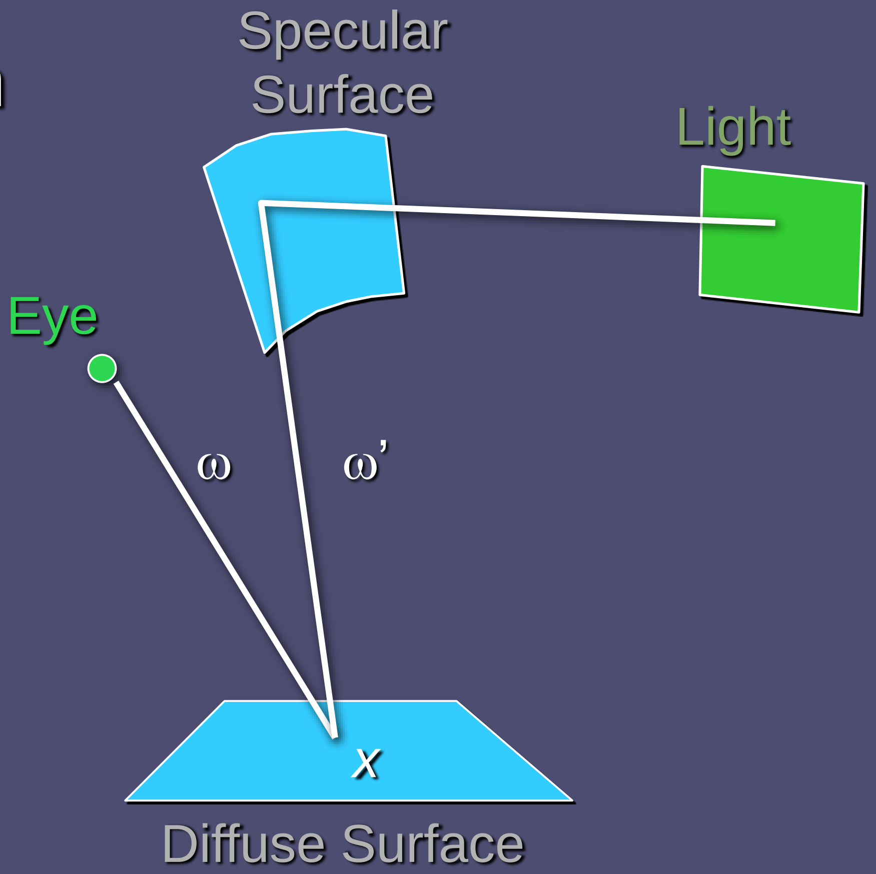
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

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

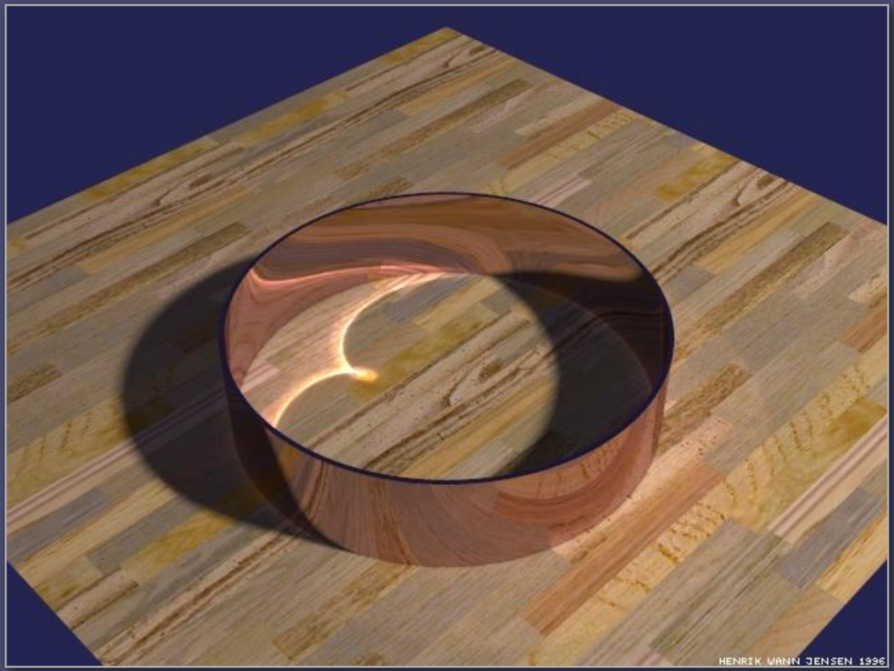
- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics



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

Motivation

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

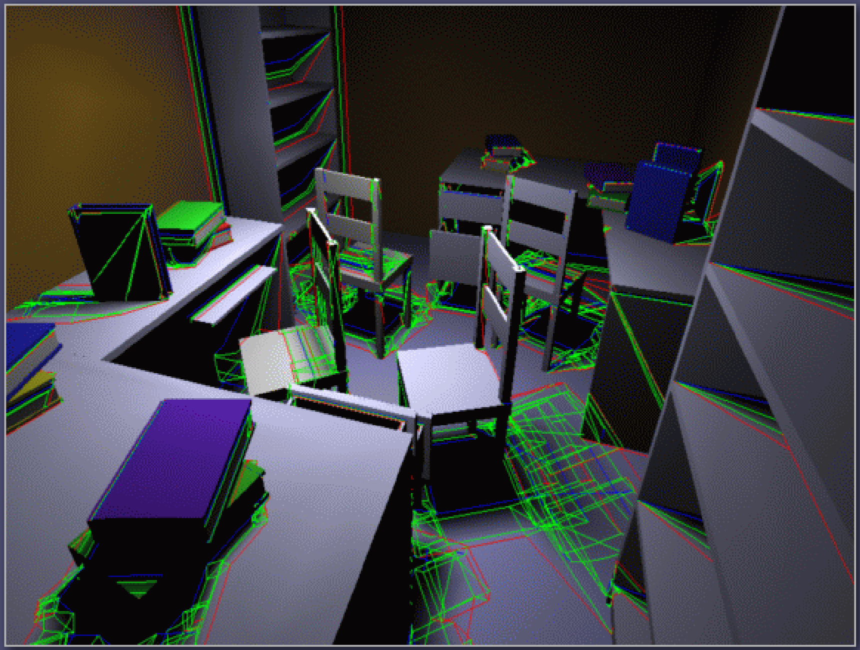


Jensen

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

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

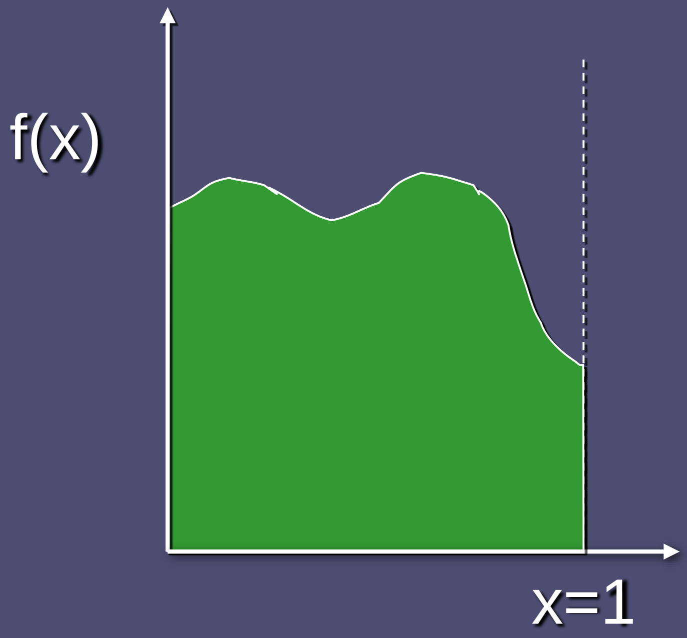
$$L(x,\vec{w}) = L_e(x,x \to e) + \int_S f_r(x,x' \to x, x \to e) L(x' \to x) V(x,x') G(x,x') dA$$

Outline

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

Integration in 1D

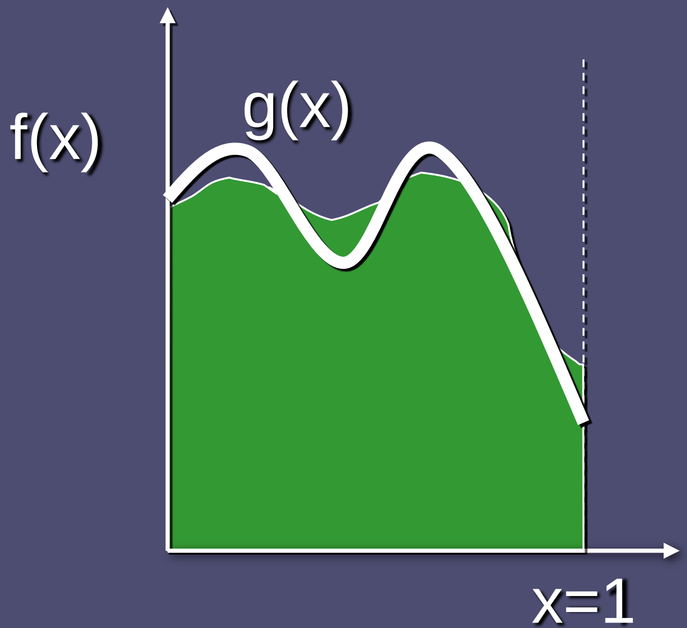
$$\int_0^1 f(x)dx = ?$$



Slide courtesy of
Peter Shirley

We can approximate

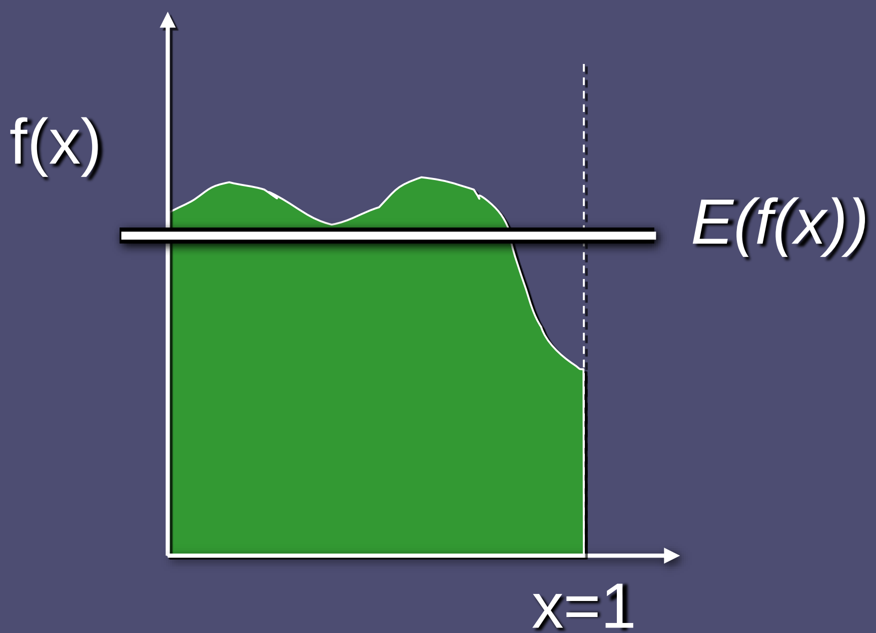
$$\int_0^1 f(x)dx = \int_0^1 g(x)dx$$



Slide courtesy of
Peter Shirley

Or we can average

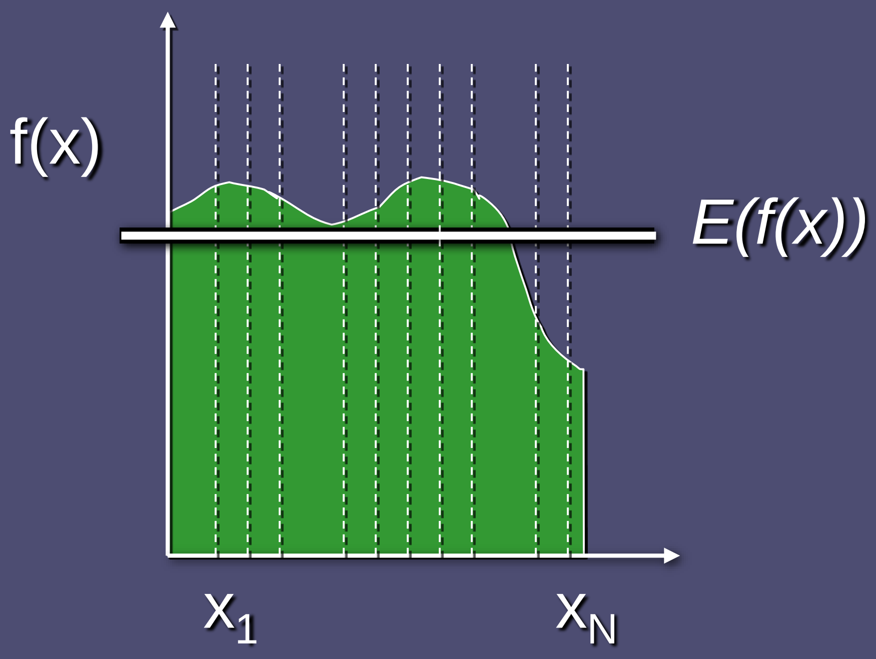
$$\int_0^1 f(x)dx = E(f(x))$$



Slide courtesy of
Peter Shirley

Estimating the average

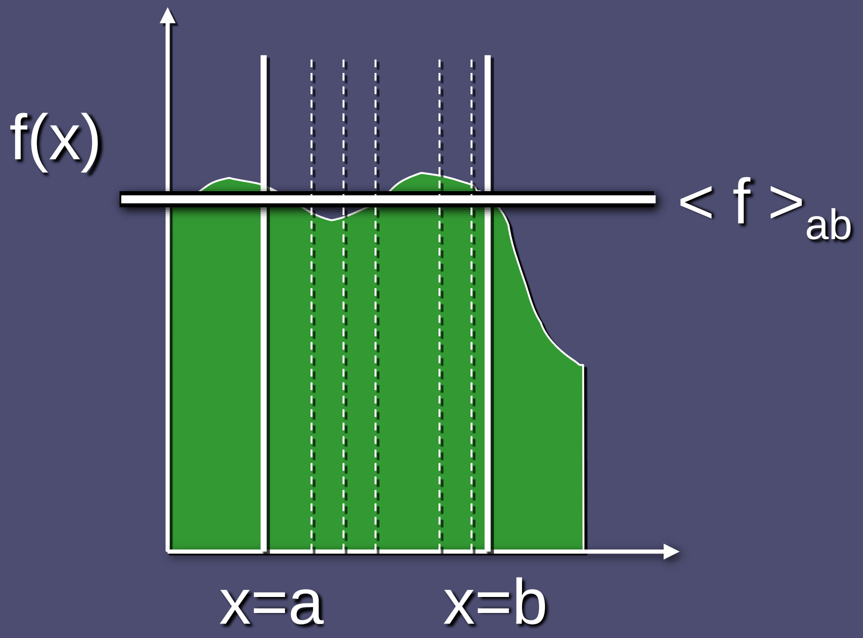
$$\int_0^1 f(x)dx = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$



Slide courtesy of
Peter Shirley

Other Domains

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

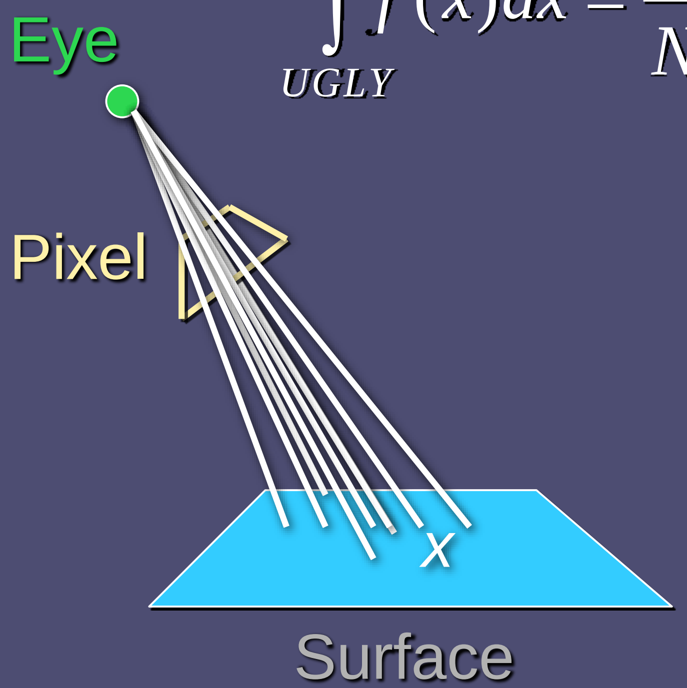


Slide courtesy of
Peter Shirley

Multidimensional Domains

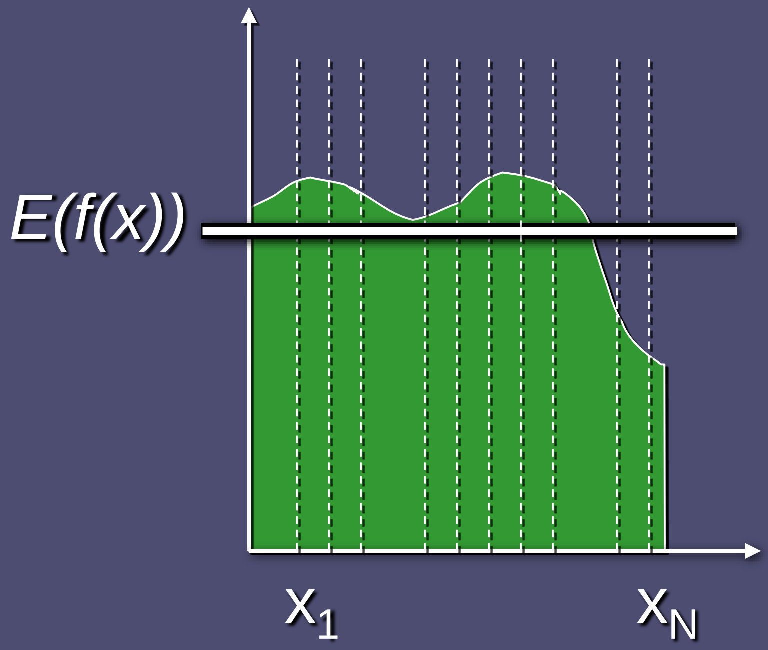
- Same ideas apply for integration over …
  - Pixel areas
  - Surfaces
  - Projected areas
  - Directions
  - Camera apertures
  - Time
  - Paths

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

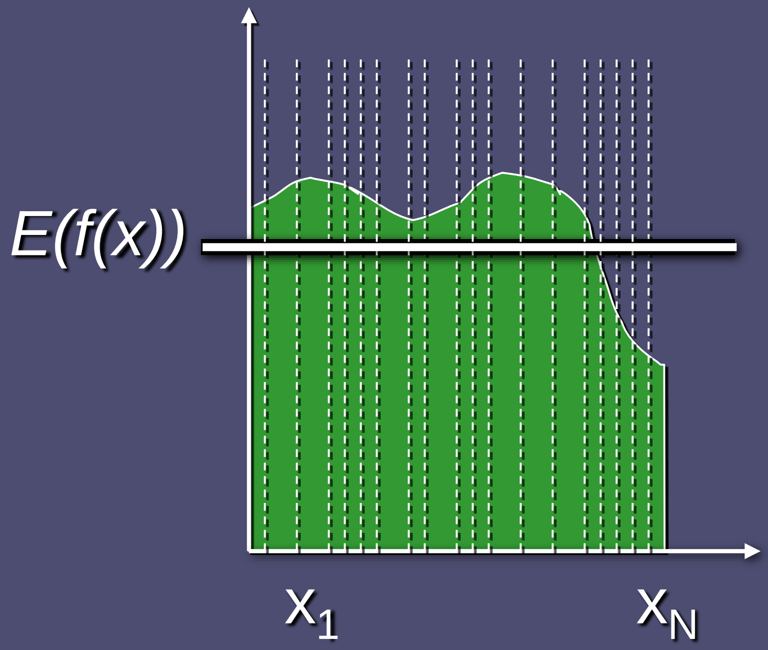


Efficiency?

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



Efficiency?



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

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

Outline

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

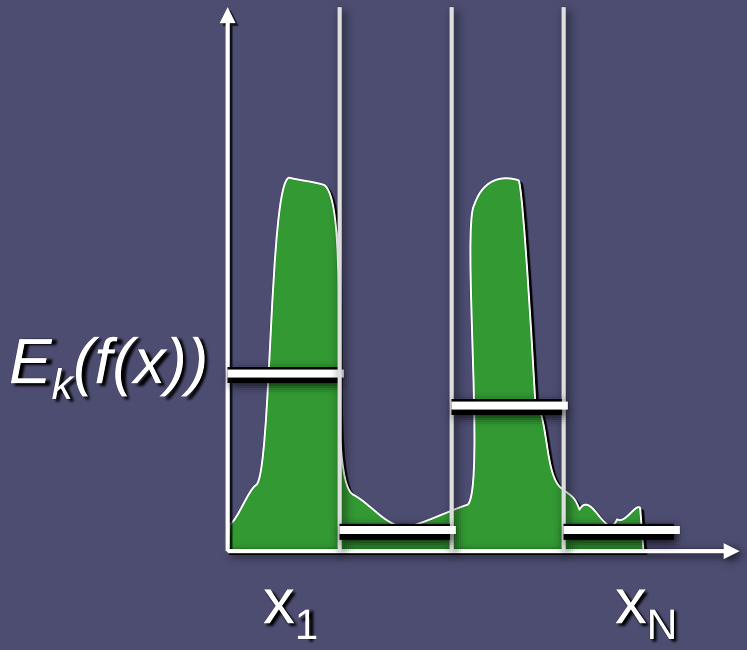
Variance Reduction Techniques

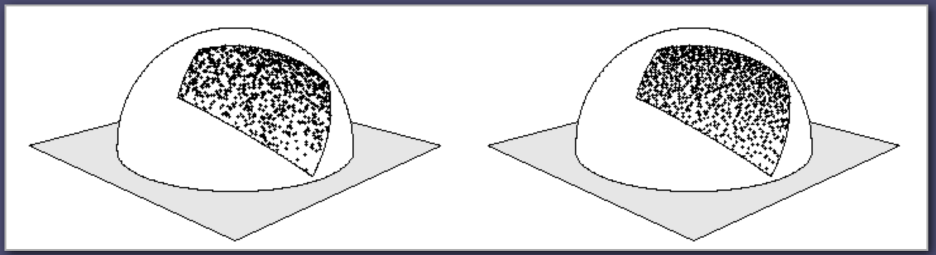
- Stratified sampling
- Importance sampling
- Metropolis sampling
- Quasi-random

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

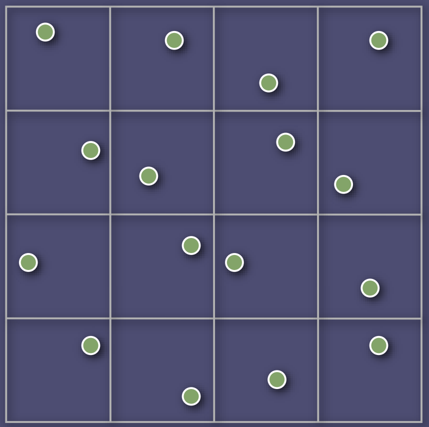
Stratified Sampling

- Estimate subdomains separately





Arvo

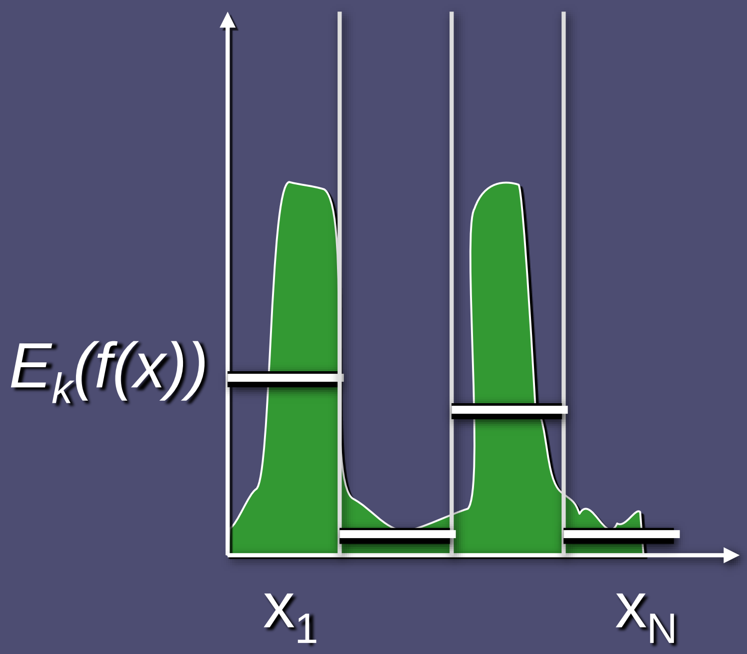


Stratified Sampling

- This is still unbiased

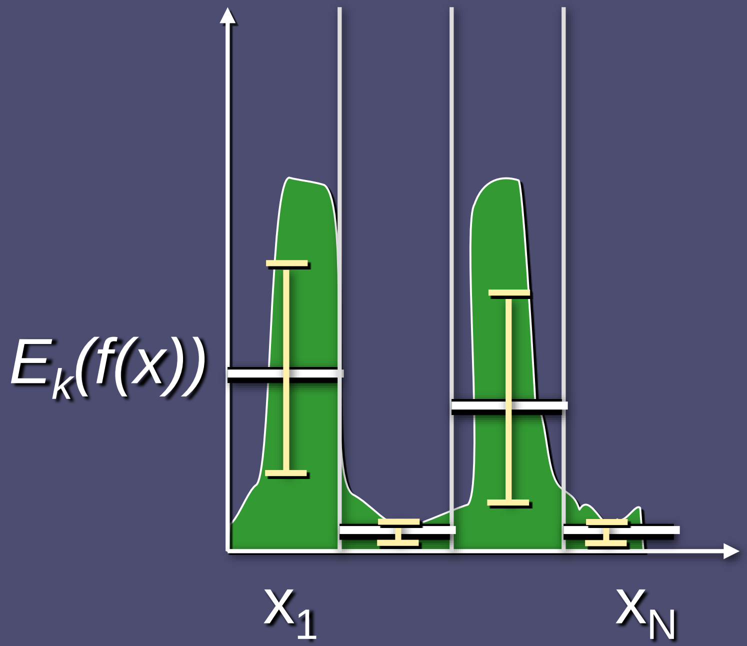
$$F_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

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



Stratified Sampling

- Less overall variance if less variance in subdomains



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

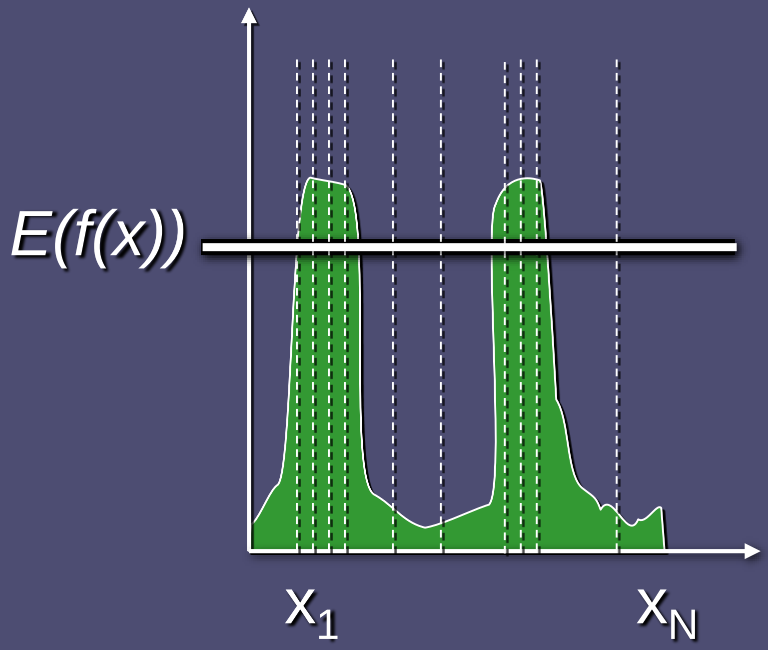
Variance Reduction Techniques

- Stratified sampling
- Importance 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

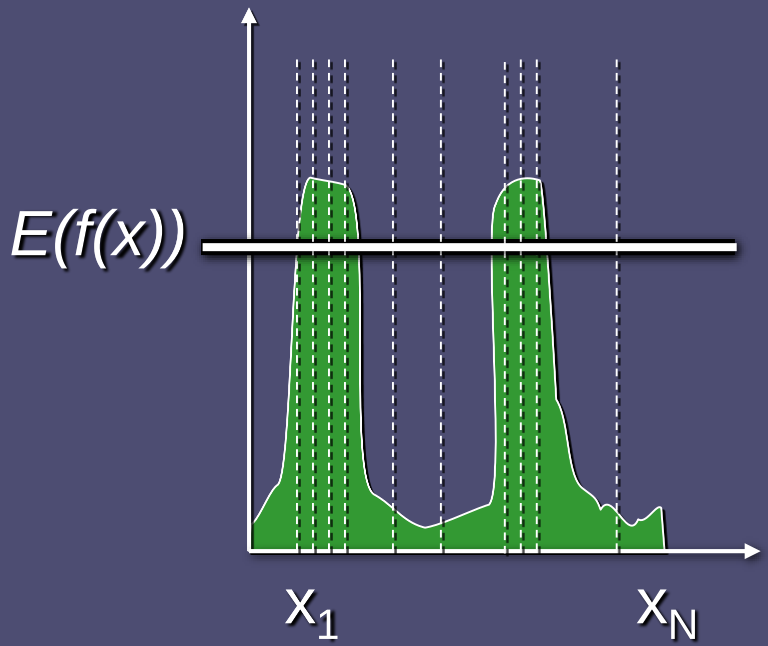


$$\int_{\Omega} f(x)dx = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

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

Importance Sampling

- This is still unbiased



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

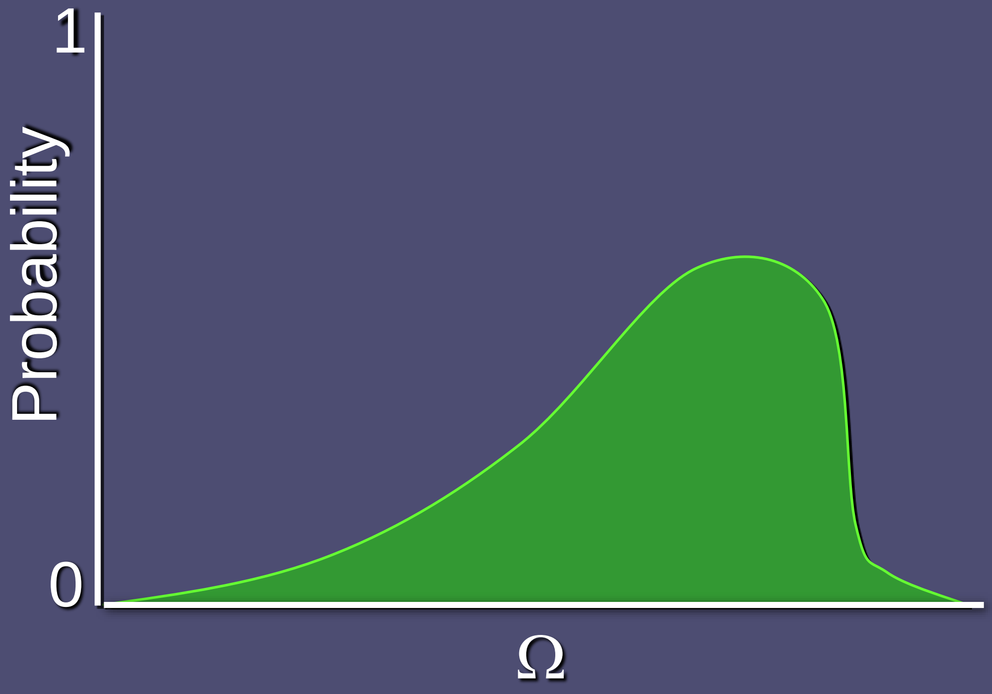
$$= \int_\Omega \frac{f(x)}{p(x)}\,p(x)\,dx$$

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

for all N

Importance Sampling

- How do we draw samples with probability proportional to function value?

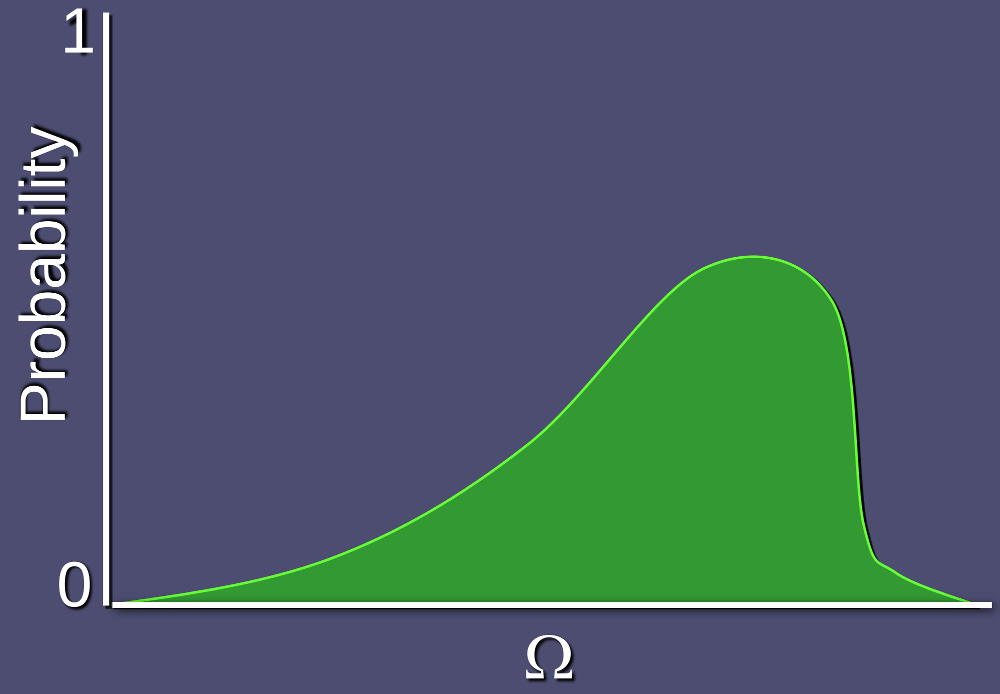


Importance Sampling

- Sampling uniform distribution:
  - Use random number generator

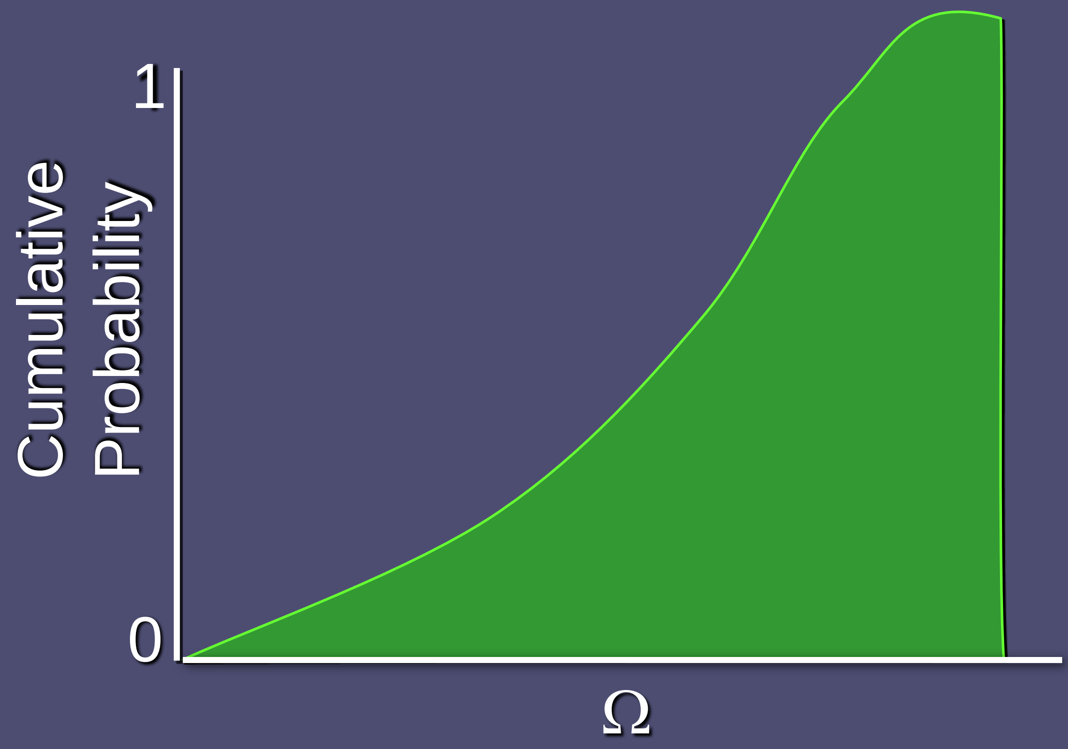
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



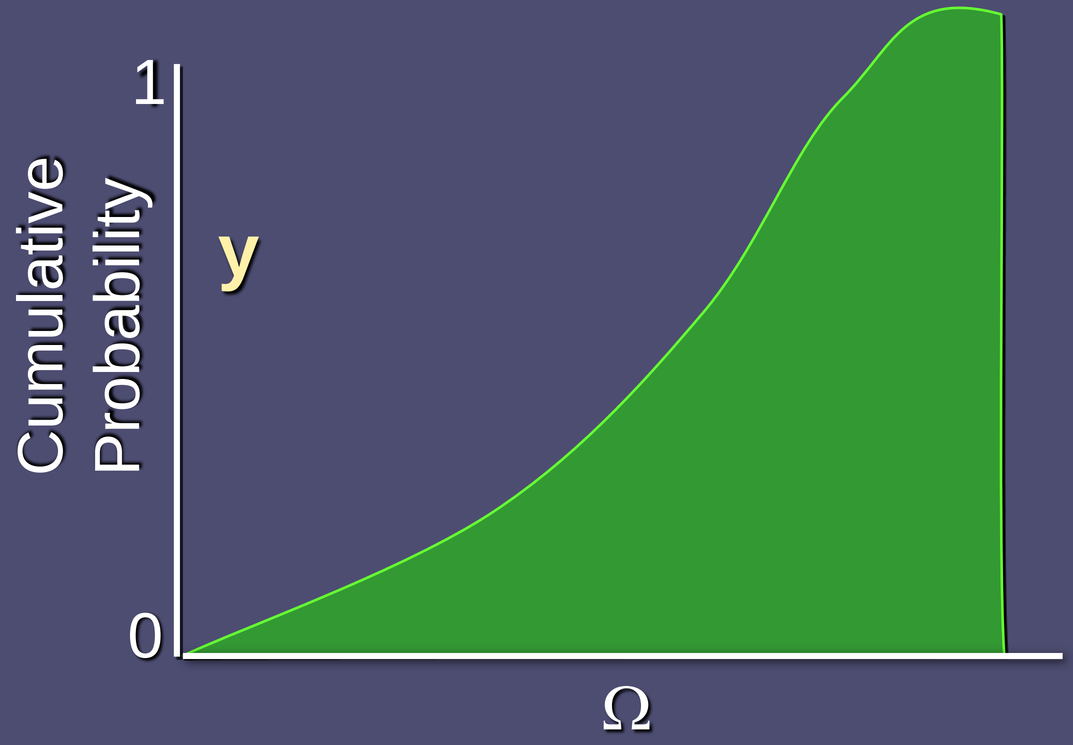
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



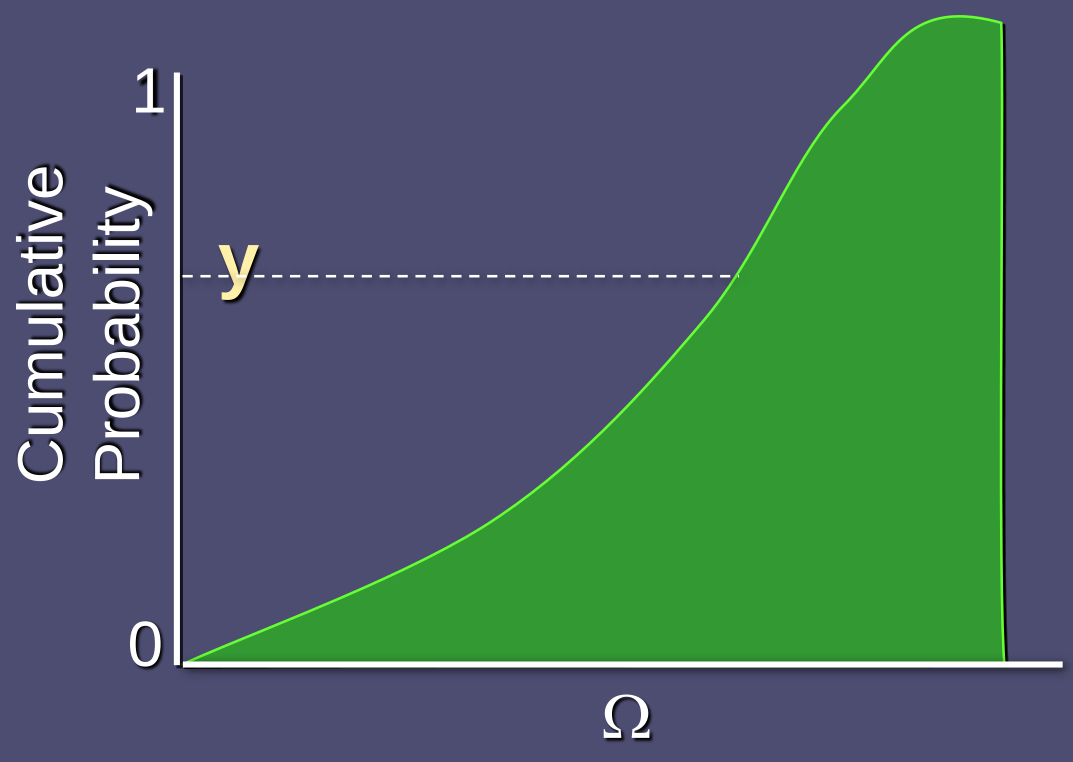
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



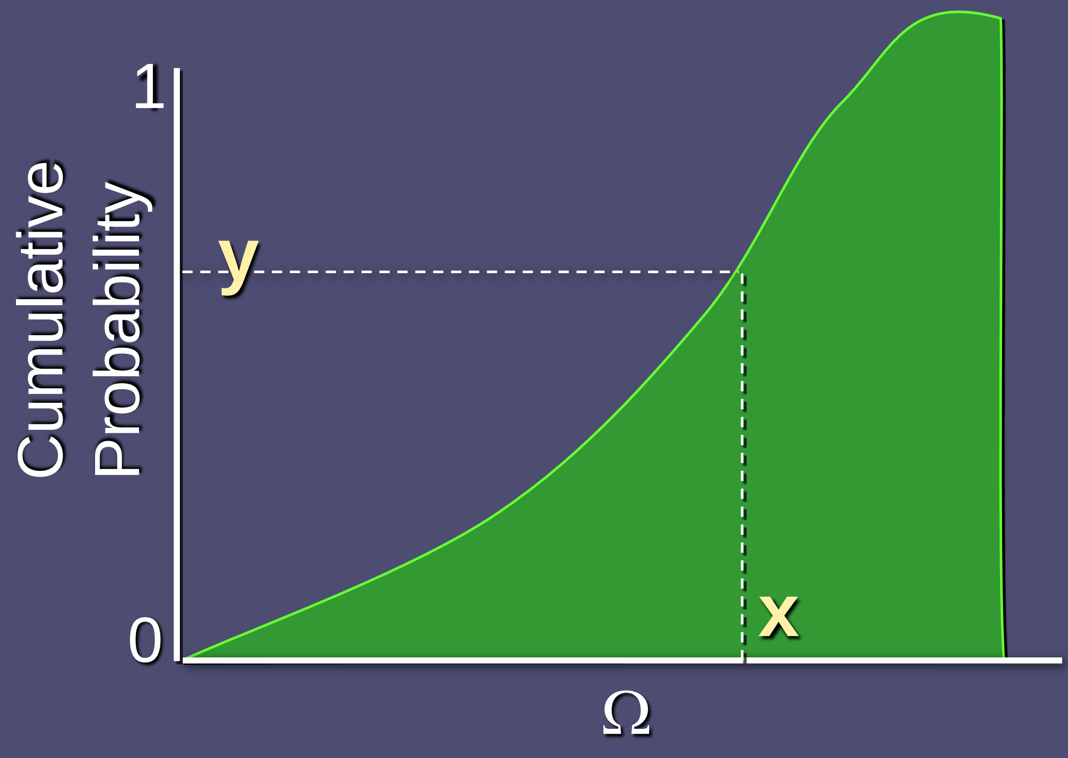
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



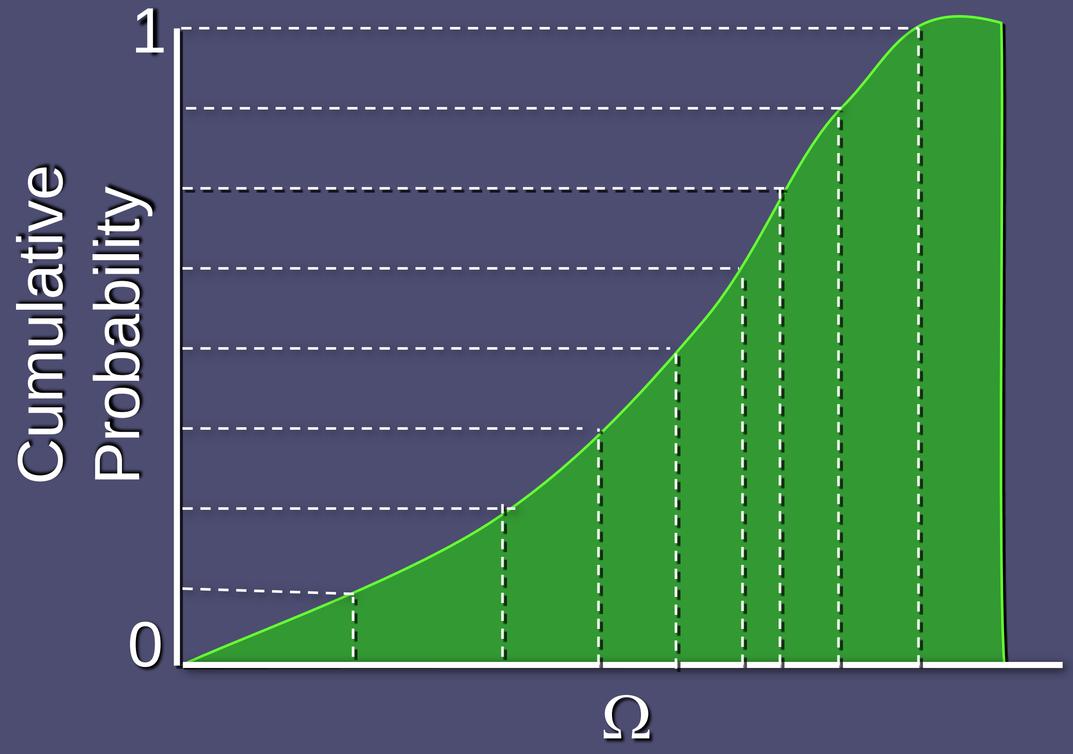
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



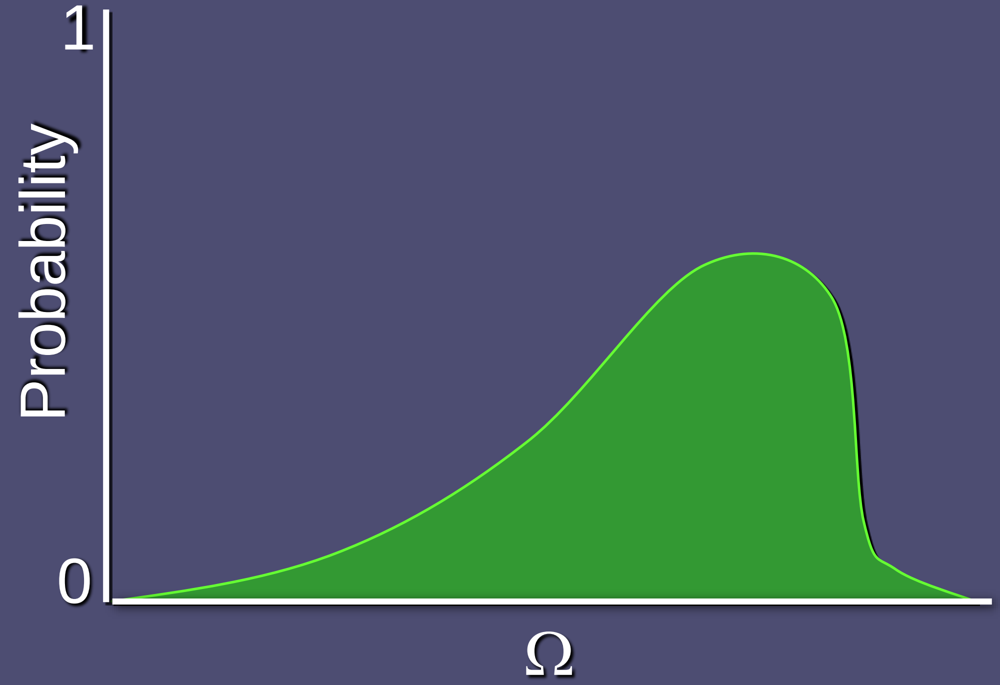
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



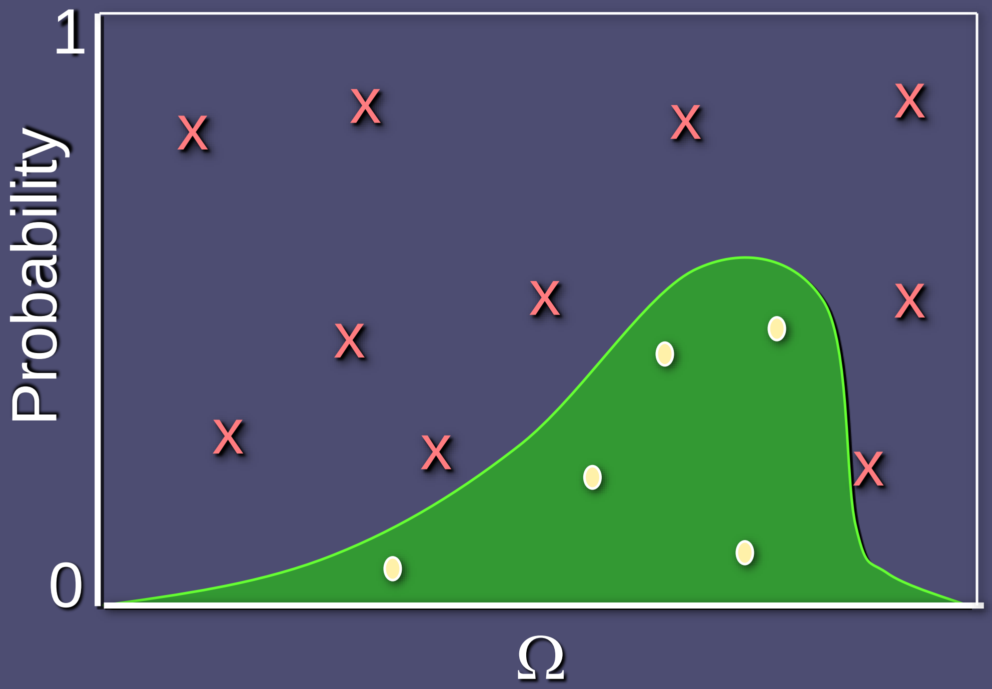
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



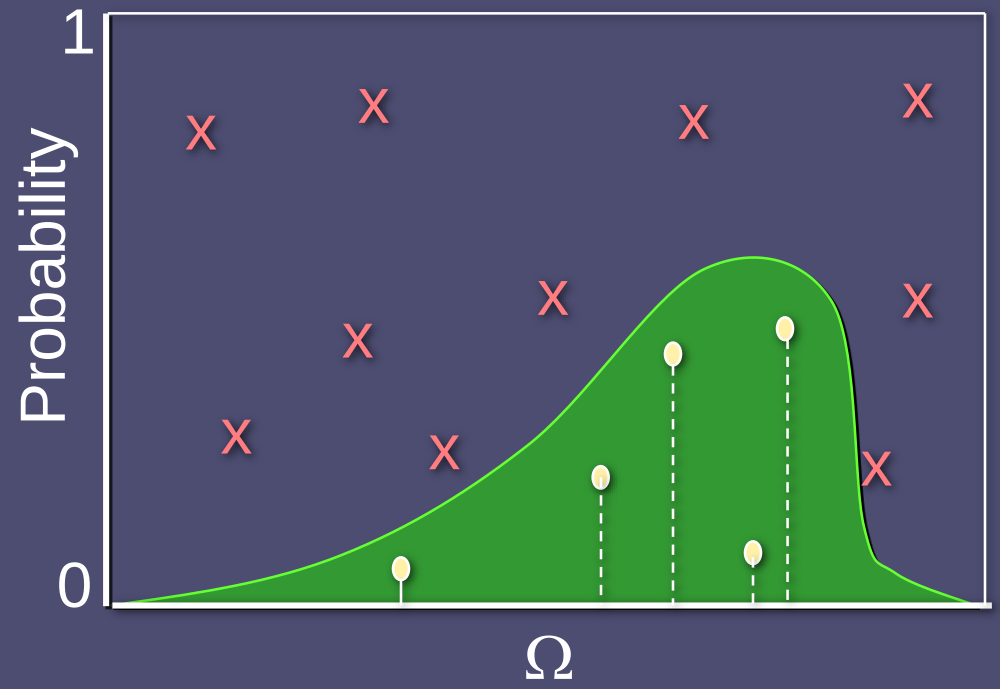
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection



Combining Multiple PDFs

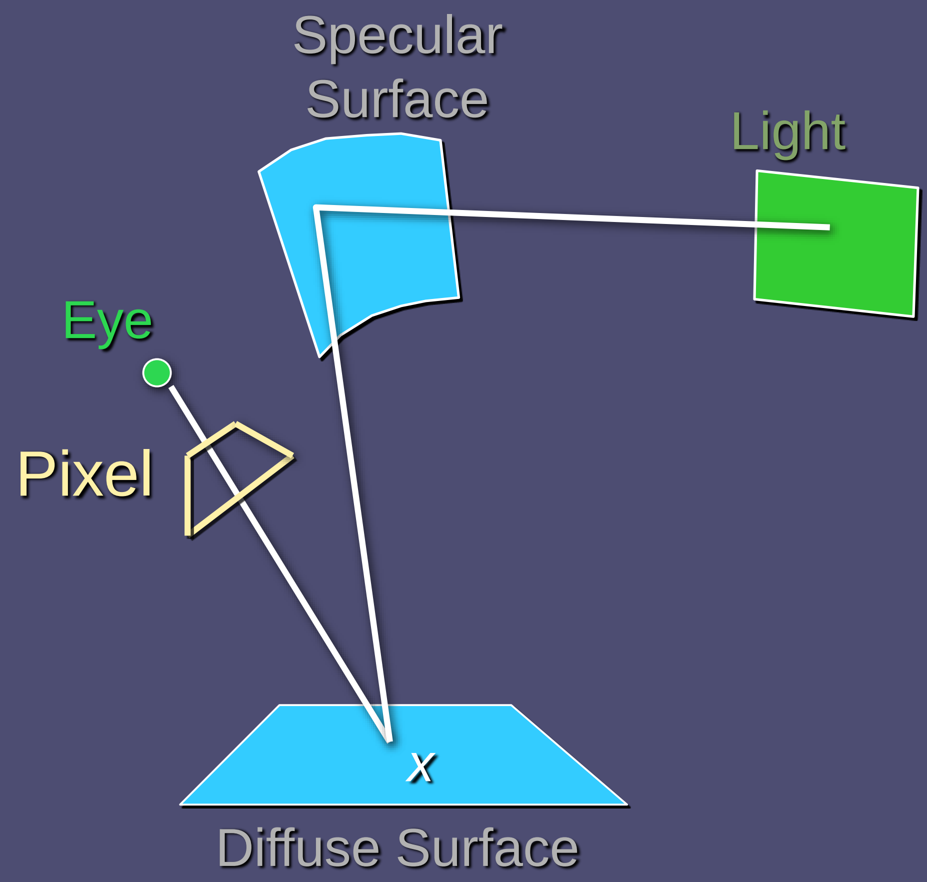
- Balance heuristic
  - Use combination of samples generated for each PDF
  - Number of samples for each PDF chosen by weights
  - Near optimal

Outline

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

Monte Carlo Path Tracing

- Integrate radiance for each pixel by sampling paths randomly

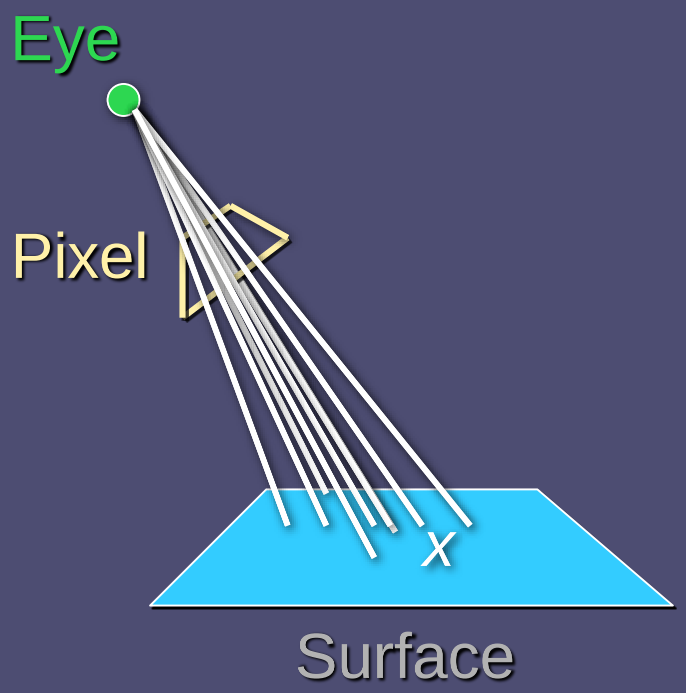


$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_{\Omega} f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}' \bullet \vec{n}) d\vec{w}$$

Monte Carlo Path Tracer

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

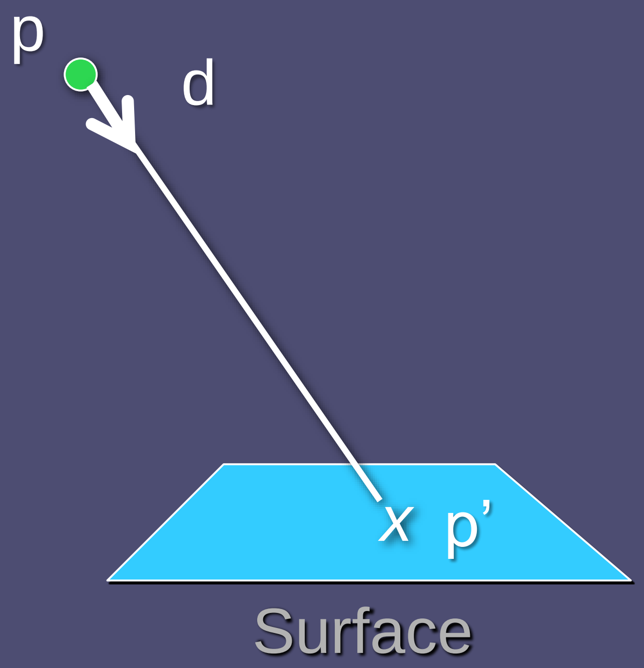
- Use stratified sampling to select rays within each pixel



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

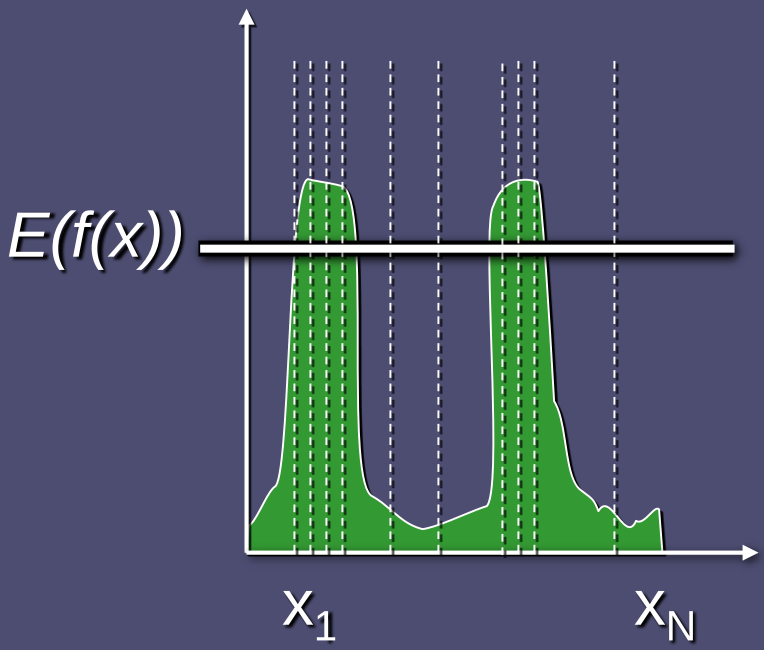
TracePath

- TracePath(*p*, *d*) returns (r,g,b):
  - Trace ray (*p*, *d*) to find nearest intersection *p'*
  - Sample radiance leaving p' towards p



TracePath

- Can sample radiance however we want, but contribution weighted by 1/probability



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

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

TracePath

- TracePath(p, d) returns (r,g,b):
  - Trace ray (p, d) to find nearest intersection p'
  - If random() < p_{emit} then
    - Emitted:

      return $(1/\ p_{emit}) * (Le_{red}, Le_{green}, Le_{blue})$
    - Reflected:

      generate ray in random direction d'

      return $(1/\ (1-p_{emit})) * f_r(d \rightarrow d') * (n \cdot d') *$ TracePath(p', d')

TracePath

- TracePath(p, d) returns (r,g,b):
  - Trace ray (p, d) to find nearest intersection p'
  - If Le = (0,0,0) then $p_{emit} = 0$
    else if $f_r = (0,0,0)$ then $p_{emit} = 1$
    else $p_{emit} = .9$
  - If random() < p_{emit} then
    - Emitted:

      return $(1/\ p_{emit}) * (Le_{red}, Le_{green}, Le_{blue})$
    - Reflected:

      generate ray in random direction d'

      return $(1/\ (1-p_{emit})) * f_r(d \rightarrow d') * (n \cdot d') *$ TracePath(p', d')

TracePath

- Reflected case:
  - Pick a light source
  - Trace a ray towards that light
  - Trace a ray anywhere except for that light
    - Rejection sampling
  - Divide by probabilities
    - p_{light} = 1/(solid angle of light) for ray to light source
    - (1 – the above) for non-light ray

TracePath

- TracePath(p, d) returns (r,g,b):
  - Trace ray (p, d) to find nearest intersection p'
  - If Le = (0,0,0) then $p_{emit} = 0$
    else if $f_r = (0,0,0)$ then $p_{emit} = 1$
    else $p_{emit} = .9$
  - If random() < p_{emit} then
    - Emitted:

      return $(1/ p_{emit}) * (Le_{red}, Le_{green}, Le_{blue})$
    - Reflected:

      generate ray in random direction d' towards a light

      $L_r = (1/2 * p_{light}) * f_r(d \to d') * (n \cdot d') *$ TracePath(p', d')

      generate ray in random direction d' not towards the light

      $L_r += (1/2*(1-p_{light})) * f_r(d \to d') * (n \cdot d') *$ TracePath(p', d')

      return $(1/ (1-p_{emit})) * L_r$

Reflected Ray Sampling

- Uniform directional sampling:
  how to generate random ray on hemisphere?

Reflected Ray Sampling

- 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

Reflected Ray Sampling

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

BRDF Importance Sampling

- Better than uniform sampling:
  importance sampling
- Because you divide by probability, ideally:
  probability $\propto f_r * \cos\theta_i$
- [Lafortune, 1994]:

$$f_r(x, \vec{\omega}_i, \vec{\omega}_o) = k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \alpha$$

BRDF Importance Sampling

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

BRDF Importance Sampling

- Phong BRDF: $f_r \propto \cos^n \alpha$ where α is angle between outgoing ray and ideal mirror direction
- Constant scale = $k_s(n+2)/(2\pi)$
- Ideally we would sample this times $\cos\, \theta_i$
  – Difficult!
  – Easier to sample BRDF itself, then multiply by $\cos\, \theta_i$
  – 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 $0..2\pi$
- This gives direction w.r.t. ideal mirror direction

BRDF Importance Sampling

- Recipe for combining terms:
  - r = random()
  - If ($r < k_d$) then
    - d' = sample diffuse direction
    - weight = $1/k_d$
  - else if ($r < k_d + k_s$) then
    - d' = sample specular direction
    - weight = $1/k_s$
  - else
    - terminate ray

Recap

- TracePath(p, d) returns (r,g,b):
  - Trace ray (p, d) to find nearest intersection p'
  - If Le = (0,0,0) then $p_{emit} = 0$
    else if f_r = (0,0,0) then $p_{emit} = 1$
    else $p_{emit} = .9$
  - If random() < p_{emit} then
    - Emitted:
      return $(1/ p_{emit}) * (Le_{red}, Le_{green}, Le_{blue})$
    - Reflected:
      generate ray in random direction d' towards a light
      $L_r = (1/2 * p_{light}) * f_r(d \rightarrow d') * (n \cdot d') *$ TracePath(p', d')

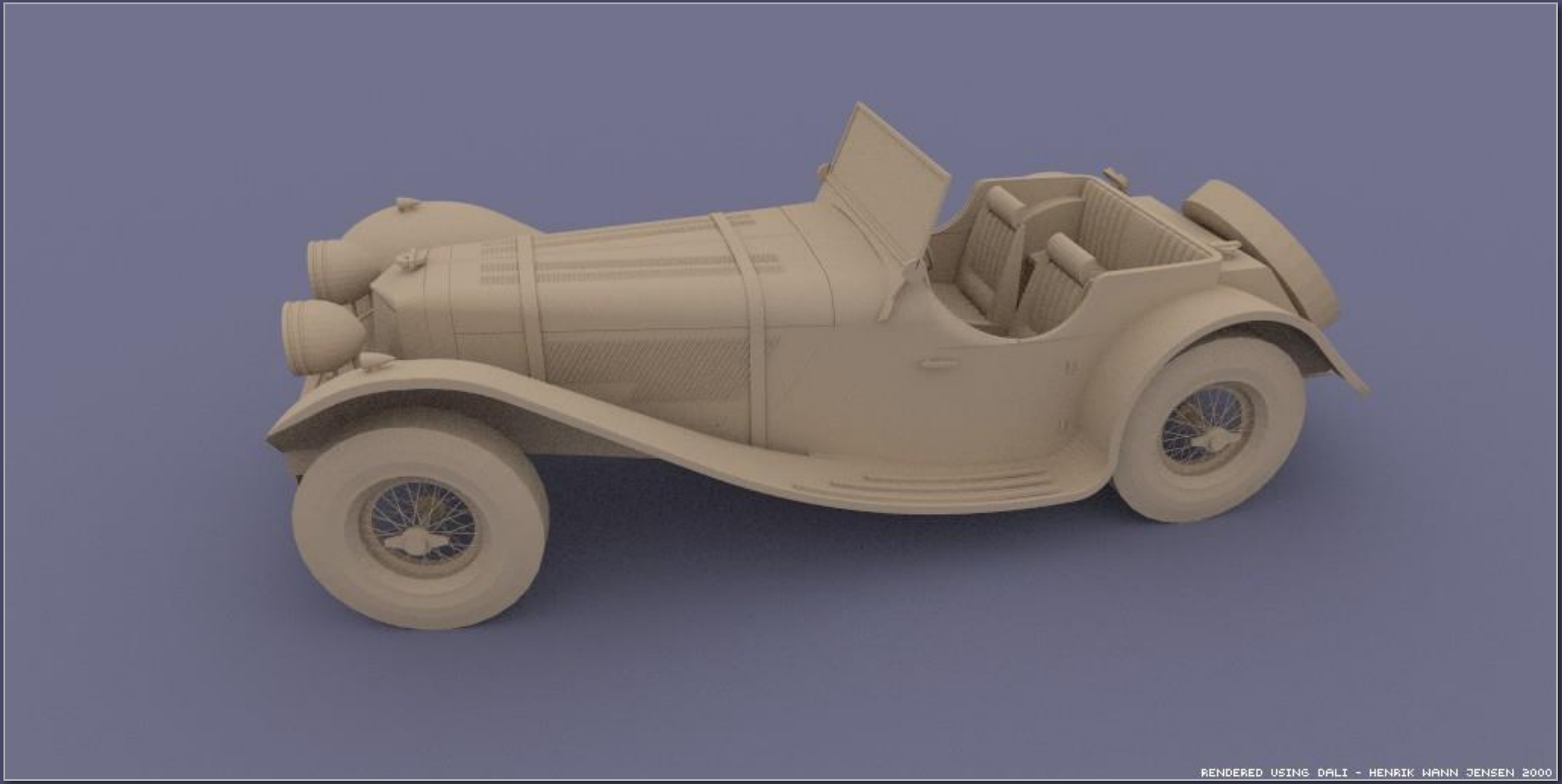
      generate ray in random direction d' not towards the light
      L_r += $(1/2*(1-p_{light})) * f_r(d \rightarrow d') * (n \cdot d') *$ TracePath(p', d')

      return $(1/ (1-p_{emit})) * L_r$

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

Monte Carlo Path Tracing

1000 paths/pixel

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