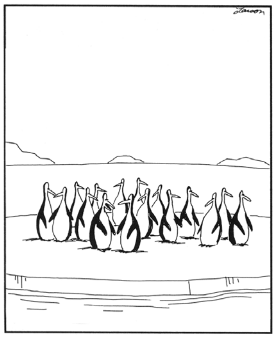
Monte-Carlo Ray Tracing



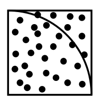
"Well, that's an interesting bit of trivia — I guess I do *only dream in black and white."*

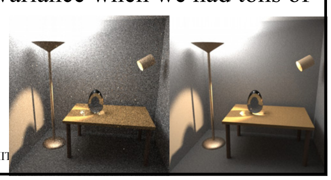
Last Time?

- Two perspective on antialiasing:
  - Signal processing
  - Integration
- Supersampling, multisampling
- Jittering, Poisson disk
- Adaptive supersampling
- Monte-Carlo integration
- Probabilities: discrete, continuous
  - Expected value, variance

Goal today

- Prove that Monte-Carlo integration works
  - Notion of expected value
- Prove its convergence speed
  - Error: notion of variance
- For all this, we need to study what happens to expected value and variance when we had tons of random samples
- Apply to ray tracing



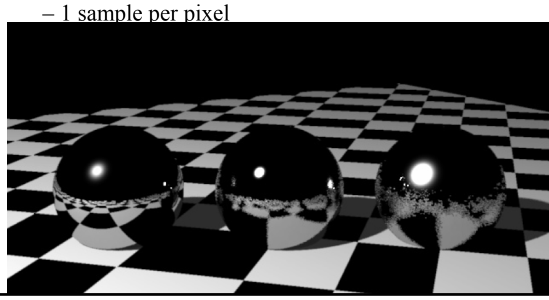


Today's lecture

- Expected value and variance
- Analysis of Monte-Carlo integration
- Monte-Carlo in graphics
- Importance sampling
- Stratified sampling
- Global illumination
- Advanced Monte-Carlo rendering

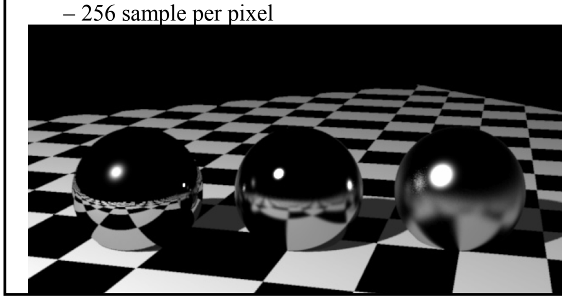
A little bit of eye candy for motivation

- Glossy material rendering
- Random reflection rays around mirror direction
  - 1 sample per pixel



A little bit of eye candy for motivation

- Glossy material rendering
- Random reflection rays around mirror direction
  - 256 sample per pixel



Back to work

- Let us study how expected value and variance behave

MIT EECS 6.837

Expected value

$$E[x] = \int_{-\infty}^{\infty} xp(x)dx$$

$$E[f(x)] = \int_{-\infty}^{\infty} f(x)p(x)dx$$

- Expected value is linear

$E[f_1(x) + a\, f_2(x)] = E[f_1(x)] + a\, E[f_2(x)]$

MIT EECS 6.837

Variance

$$\sigma^2 = E[(x - E[x])^2] = \int_{-\infty}^{\infty} (x - E[x])^2 p(x)dx$$

- Measure of deviation from expected value
- Expected value of square difference (MSE)
- Standard deviation σ: square root of variance (notion of error, RMS)

MIT EECS 6.837

Variance

$$\sigma^2 = E[(x - E[x])^2] = E[x^2] - (E[x])^2$$

- Proof:

$$\sigma^2 = E[(x - E[x])^2]$$
$$= E[x^2 - 2xE[x] + E[x]^2]$$

- Note that E[x] is a constant. By linearity of E we have:

$$\sigma^2 = E[x^2] - (2E[x])E[x] + (E[x])^2$$
$$\sigma^2 = E[x^2] - (E[x])^2$$

MIT EECS 6.837

Non-linearity of variance

$$\sigma^2 = E[(x - E[x])^2] = \int_{-\infty}^{\infty} (x - E[x])^2 p(x)dx$$

- Variance is not linear !!!!
- $\sigma^2[ax] = a^2\, \sigma^2[x]$

MIT EECS 6.837

Non-linearity of variance

- Consider two random variable x_1 and x_2

$$\sigma^2(x_1 + x_2) = E[(x_1 + x_2)^2] - (E[x_1 + x_2])^2$$
$$= E[x_1^2 + 2x_1x_2 + x_2^2] - (E[x_1] + E[x_2])^2$$
$$= E[x_1^2] + 2E[x_1x_2] + E[x_2^2] - E[x_1]^2 - 2E[x_1]E[x_2] - E[x_2]^2$$
$$= \sigma^2[x_1] + \sigma^2[x_2] + 2E[x_1x_2] - 2E[x_1]E[x_2]$$

- We define the covariance

$Cov[x_1,x_2] = E[x_1x_2] - E[x_1]\, E[x_2]$

$\sigma^2[x_1+x_2] = \sigma^2[x_1] + \sigma^2[x_2] + 2\, Cov[x_1,x_2]$

MIT EECS 6.837

Non-linearity of variance, covariance

- Consider two random variable x_1 and x_2
- We define the covariance
  $Cov[x_1,x_2] = E[x_1 x_2] - E[x_1] E[x_2]$
  - Tells how much they are big at the same time
  - **Null if variables are independent**

$$\sigma^2[x_1+x_2] = \sigma^2[x_1] + \sigma^2[x_2] + 2\, Cov[x_1,x_2]$$

Recap

- Expected value is linear
  - $E[ax_1+bx_2]=aE[x_1]+bE[x_2]$
- Variance is not
- For two independent variables
  - $\sigma^2[x_1+x_2]=\sigma^2[x_1]+\sigma^2[x_2]$
  - If not independent, needs covariance
- $\sigma^2[ax]=a^2\sigma^2[x]$

Questions?

- Image from the ARNOLD Renderer by Marcos Fajardo

Today's lecture

- Expected value and variance
- Analysis of Monte-Carlo integration
- Monte-Carlo in graphics
- Importance sampling
- Stratified sampling
- Global illumination
- Advanced Monte-Carlo rendering

Monte Carlo integration

- Function *f(x)* of $x \in [a\ b]$
- We want to compute $I = \int_a^b f(x)dx$
- Consider a random variable *x*
- If *x* has uniform distribution, *I=E[f(x)]*
  - By definition of the expected value

Sum of Random Variables

- Use *N* independent identically-distributed (IID) variables x_i
  - Share same probability (uniform here)
- Define $F_N = \frac{1}{N}\sum_{j=1}^{n} f(x_i)$
- By linearity of the expectation:
  $E[F_N] = E[f(x)]$

Study of variance

$$\sigma^2[F_N] = \sigma^2\left[\sum_{j=1}^{n} \frac{f(x_i)}{N}\right]$$

- Recall $\sigma^2[x+y] = \sigma^2[x] + \sigma^2[y] + 2\,\mathrm{Cov}[x,y]$
  - We have independent variables: Cov[xi, xj]=0 if $i \neq j$
- $\sigma^2[ax] = a^2\,\sigma^2[x]$

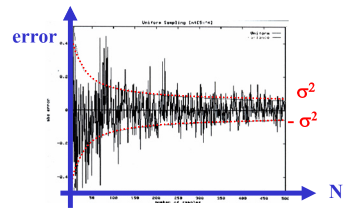
$$\sigma^2[F_N] = \frac{\sigma^2[f(x)]}{N}$$

- i.e. stddev σ (error) decreases by $\sqrt{N}$

Example

$$I = \int_0^1 5x^4 dx$$

- We know it should be 1.0
- In practice with uniform samples:



Monte Carlo integration with proba

- Consider N random samples over domain **with probability *p(x)***
- Define estimator ⟨ I ⟩ as:

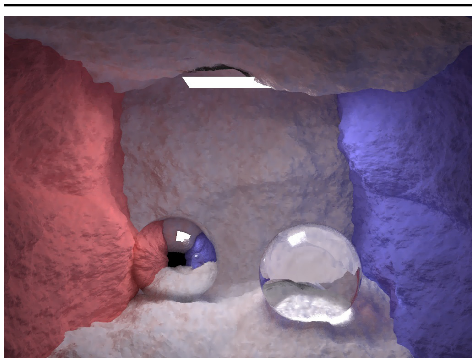
$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

- Probability *p* allows us to sample the domain more smartly

Monte-Carlo Recap

- Expected value is the integrand
  - Accurate “on average”
- Variance decrease in 1/N
  - Error decreases in $1/\sqrt{n}$
- Good news:
  - Math are over for today
  - OK, it’s bad news if you like math (and you should)

Questions?



Advantages of MC Integration

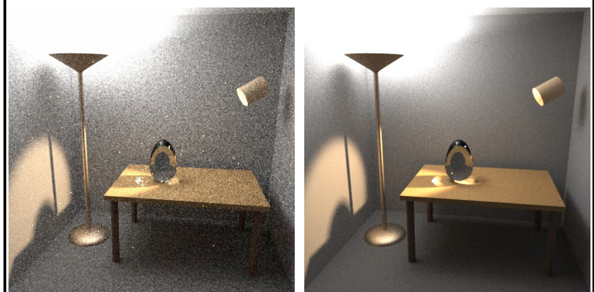
- Few restrictions on the integrand
  - Doesn’t need to be continuous, smooth, ...
  - Only need to be able to evaluate at a point
- Extends to high-dimensional problems
  - Same convergence
- Conceptually straightforward
- Efficient for solving at just a few points

Disadvantages of MC

- Noisy
- Slow convergence
- Good implementation is hard
  - Debugging code
  - Debugging maths
  - Choosing appropriate techniques

Questions?

- Images by Veach and Guibas



Naïve sampling strategy — Optimal sampling strategy

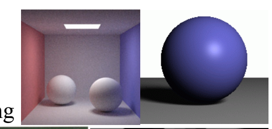
Today’s lecture

- Expected value and variance
- Analysis of Monte-Carlo integration
- Monte-Carlo in graphics
- Importance sampling
- Stratified sampling
- Global illumination
- Advanced Monte-Carlo rendering

What can we integrate?

$$\int\int\int\int\int L(x,y,t,u,v)\,dx\,dy\,dt\,du\,dv$$

- Pixel: antialiasing
- Light sources: Soft shadows
- Lens: Depth of field
- Time: Motion blur
- BRDF: glossy reflection
- Hemisphere: indirect lighting

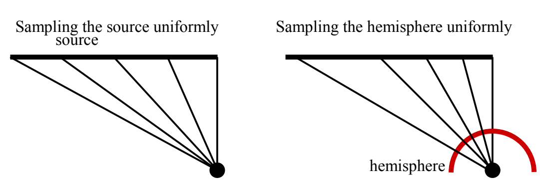


Domains of integration

- Pixel, lens (Euclidean 2D domain)
- Time (1D)
- Hemisphere
  - Work needed to ensure uniform probability
- Light source
  - Same thing: make sure that the probabilities and the measures are right.

Example: Light source

- Integrate over surface or over angle
- Be careful to get probabilities and integration measure right!
  - More in 6.839



Questions?

- Image by Henrik

Today's lecture

- Expected value and variance
- Analysis of Monte-Carlo integration
- Monte-Carlo in graphics
- Importance sampling
- Stratified sampling
- Global illumination
- Advanced Monte-Carlo rendering

Important issues in MC rendering

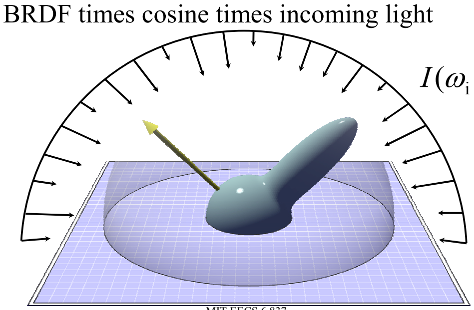
Reduce variance!

- Choose a smart probability distribution
- Choose smart sampling patterns

And of course, cheat to make it faster without being noticed

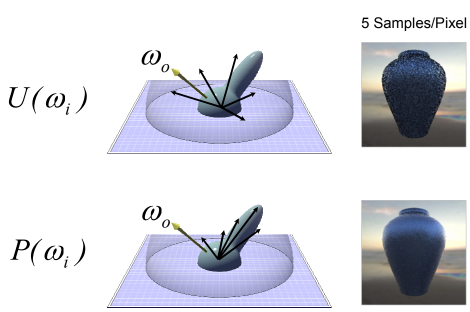
Example: Glossy rendering

- Integrate over hemisphere
- BRDF times cosine times incoming light



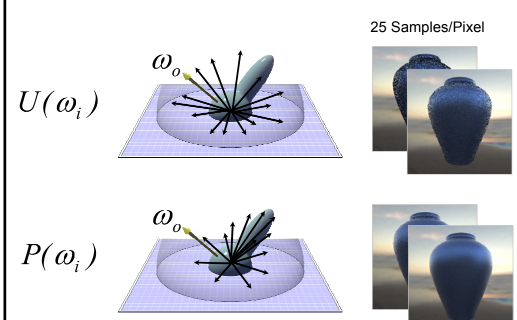
Slide courtesy of Jason Lawrence

Sampling a BRDF



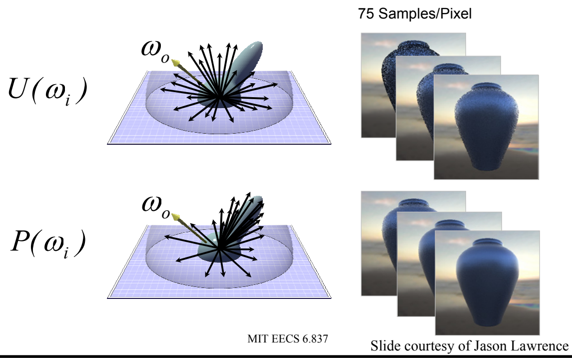
Slide courtesy of Jason Lawrence

Sampling a BRDF



Slide courtesy of Jason Lawrence

Sampling a BRDF



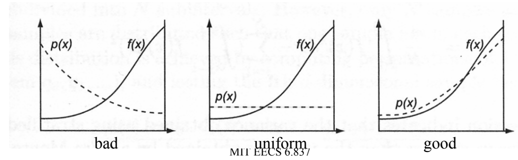
MIT EECS 6.837

Slide courtesy of Jason Lawrence

Importance sampling

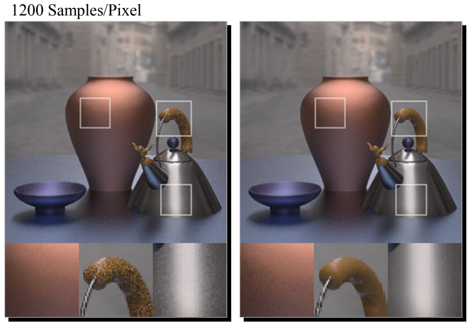
$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

- Choose p wisely to reduce variance
  - p that resembles f
  - Does not change convergence rate (still sqrt)
  - But decreases the constant



MIT EECS 6.837

Questions?



Traditional importance function

Better importance by Lawrence et al.

MIT EECS 6.837

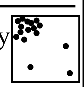
Today's lecture

- Expected value and variance
- Analysis of Monte-Carlo integration
- Monte-Carlo in graphics
- Importance sampling
- Stratified sampling
- Global illumination
- Advanced Monte-Carlo rendering

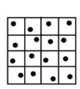
MIT EECS 6.837

Stratified sampling

- With uniform sampling, we can get unlucky
  - E.g. all samples in a corner



- To prevent it, subdivide domain Ω into non-overlapping regions Ω_i
  - Each region is called a stratum



- Take one random samples per Ω_i

MIT EECS 6.837

Example

- Borrowed from Henrik Wann Jensen

$f(x) = e^{\sin(3x^2)}$

N	I
1	2.75039
10	1.9893
100	1.79139
1000	1.75146
10000	1.77313
100000	1.77862

Unstratified
$O(1/\sqrt{N})$

$f(x) = e^{\sin(3x^2)}$

N	I
1	2.70457
10	1.72858
100	1.77925
1000	1.77606
10000	1.77610
100000	1.77610

Stratified
$O(1/N)$

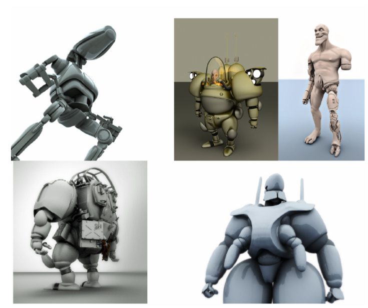
MIT EECS 6.837

Stratified sampling - bottomline

- Cheap and effective
- Typical example: jittering for antialiasing
  - Signal processing perspective: better than uniform because less aliasing (spatial patterns)
  - Monte-Carlo perspective: better than random because lower variance (error for a given pixel)

Questions?

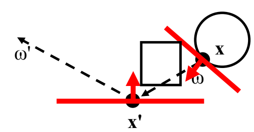
- Image from the ARNOLD Renderer by Marcos Fajardo



Today's lecture

- Expected value and variance
- Analysis of Monte-Carlo integration
- Monte-Carlo in graphics
- Importance sampling
- Stratified sampling
- Global illumination
- Advanced Monte-Carlo rendering

The Rendering Equation

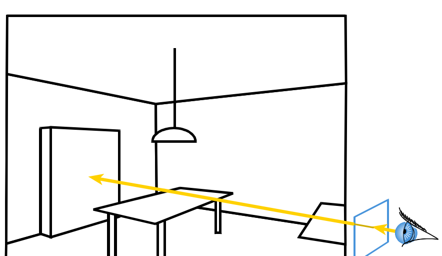


$$L(x',\omega') = E(x',\omega') + \int \rho_{x'}(\omega,\omega')L(x,\omega)G(x,x')V(x,x')\,dA$$

emission — BRDF — Incoming light — Geometric term — visibility

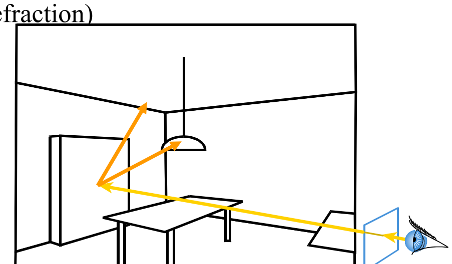
Ray Casting

- Cast a ray from the eye through each pixel



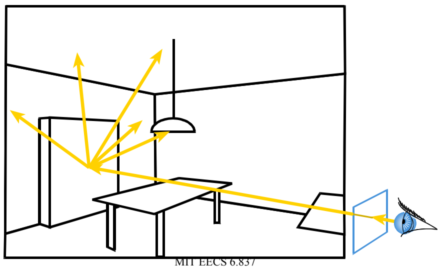
Ray Tracing

- Cast a ray from the eye through each pixel
- Trace secondary rays (light, reflection, refraction)



Monte-Carlo Ray Tracing

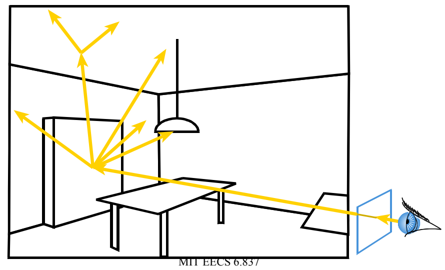
- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
  - Accumulate radiance contribution



MIT EECS 6.837

Monte-Carlo Ray Tracing

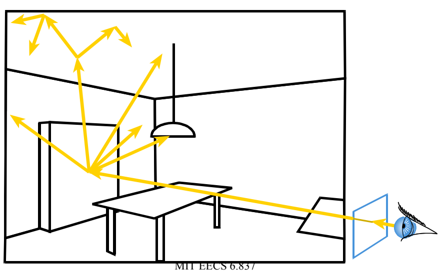
- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse



MIT EECS 6.837

Monte-Carlo

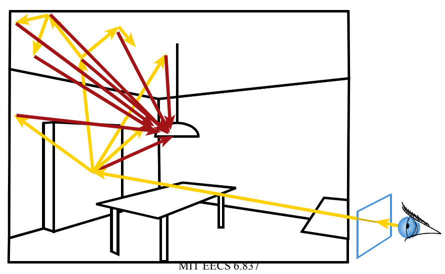
- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse



MIT EECS 6.837

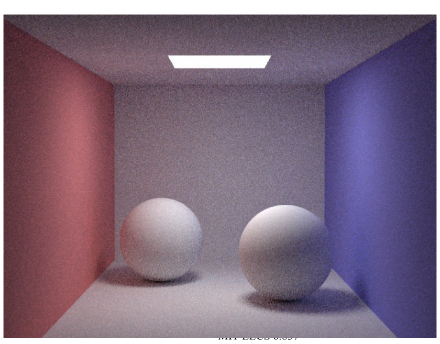
Monte-Carlo

- Systematically sample primary light



MIT EECS 6.837

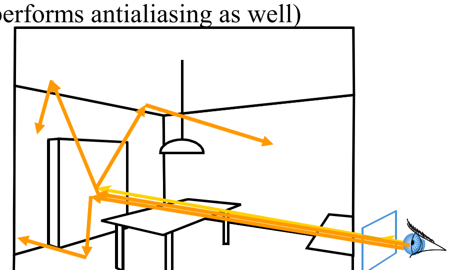
Results



MIT EECS 6.837

Monte Carlo Path Tracing

- Trace only one secondary ray per recursion
- But send many primary rays per pixel
- (performs antialiasing as well)

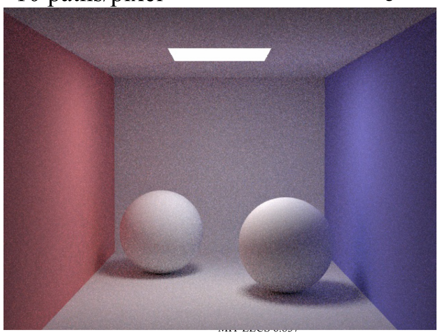


MIT EECS 6.837

Results

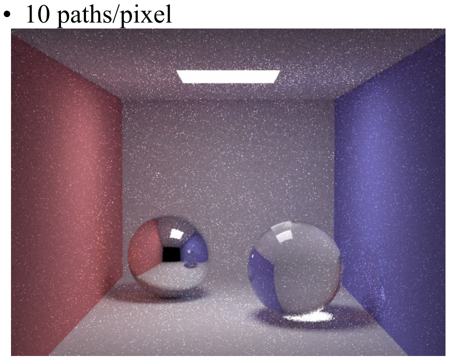
- 10 paths/pixel

Think about it : we compute an infinite-dimensional integral with 10 samples!!!



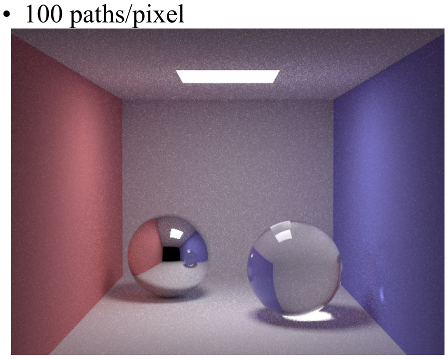
Results: glossy

- 10 paths/pixel



Results: glossy

- 100 paths/pixel

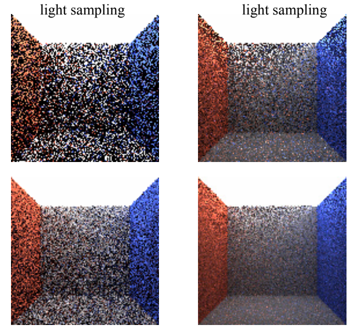


Importance of sampling the light

Without explicit light sampling | With explicit light sampling

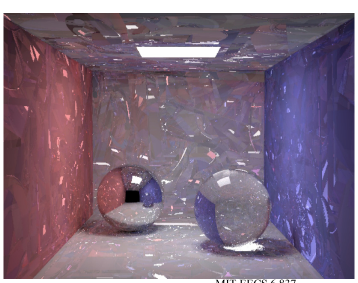
1 path per pixel

4 path per pixel

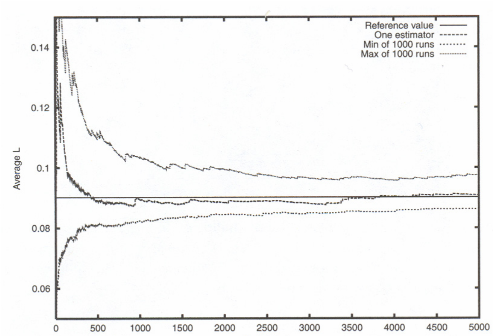


Why use random numbers?

- Fixed random sequence
- We see the structure in the error



Convergence speed

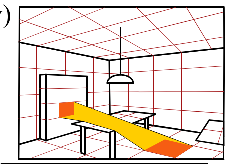


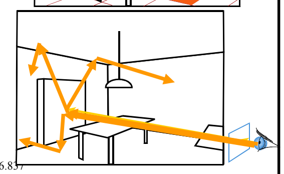
Questions?

- Vintage path tracing by Kajyia

Radiosity vs. Monte Carlo

- We have an integral equation on an infinite space
- Finite elements (Radiosity)
  - Project onto finite basis of functions
  - Linear system
  - View-independent (no angular information)
- Monte Carlo
  - Probabilistic sampling
  - View-dependent (but angular information)





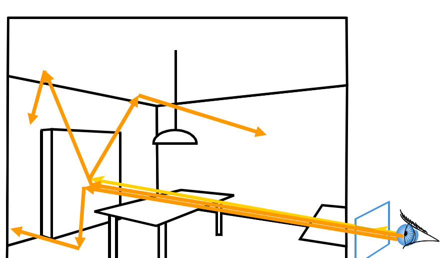
Questions?

Today's lecture

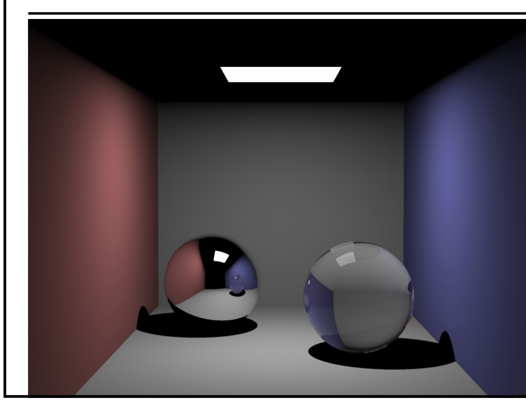
- Expected value and variance
- Analysis of Monte-Carlo integration
- Monte-Carlo in graphics
- Importance sampling
- Stratified sampling
- Global illumination
- Advanced Monte-Carlo rendering

Path Tracing is costly

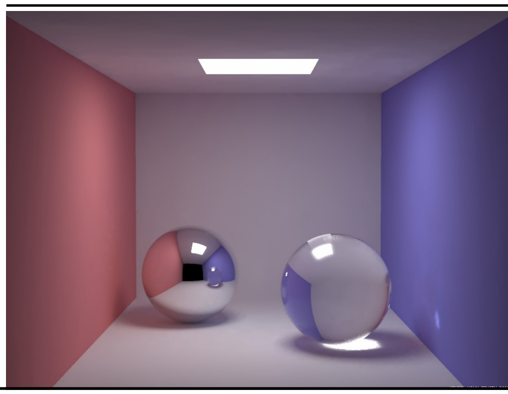
- Needs tons of rays per pixel



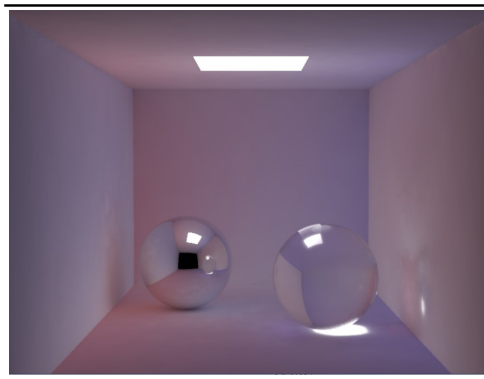
Direct illumination



Global Illumination

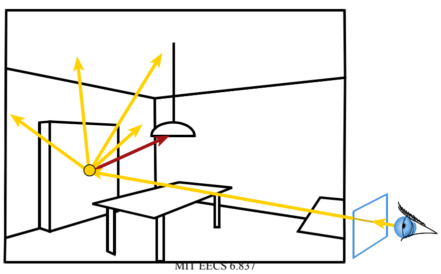


Indirect illumination: smooth



Irradiance cache

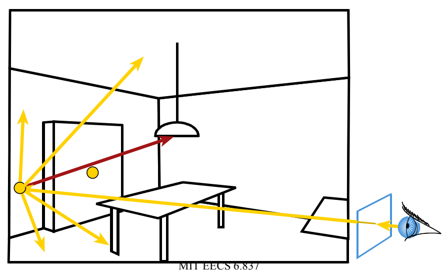
- The indirect illumination is smooth



MIT EECS 6.837

Irradiance cache

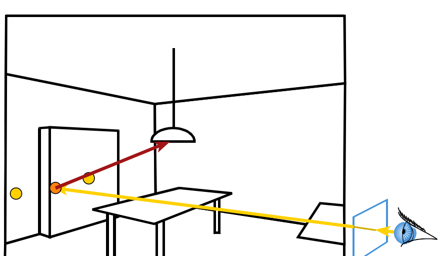
- The indirect illumination is smooth



MIT EECS 6.837

Irradiance cache

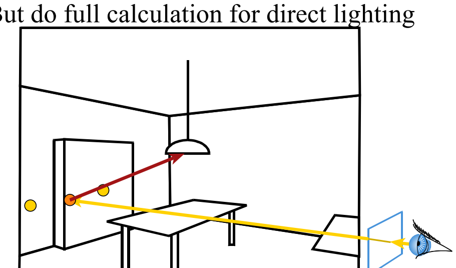
- The indirect illumination is smooth
- Interpolate nearby values



MIT EECS 6.837

Irradiance cache

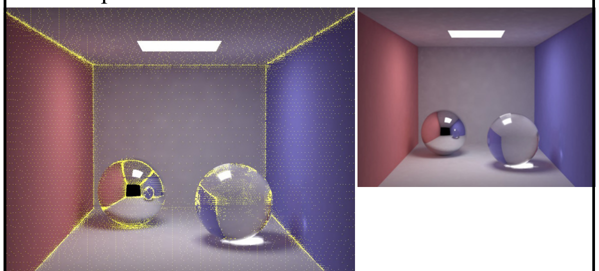
- Store the indirect illumination
- Interpolate existing cached values
- But do full calculation for direct lighting



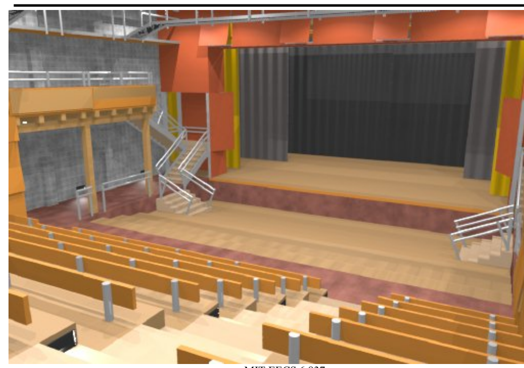
MIT EECS 6.837

Irradiance caching

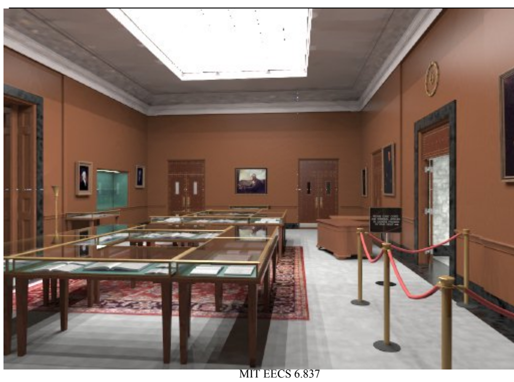
- Yellow dots:
  computation of indirect diffuse contribution



Radiance software by Greg Ward

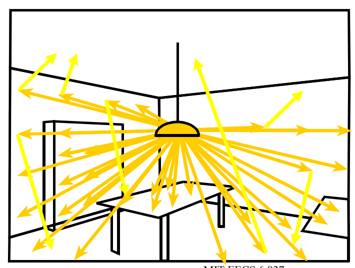


Questions?



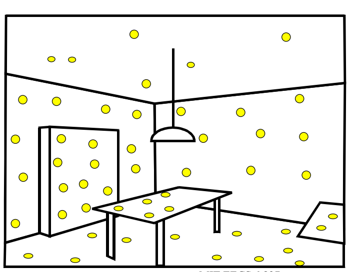
Photon mapping

- Preprocess: cast rays from light sources
- Store photons



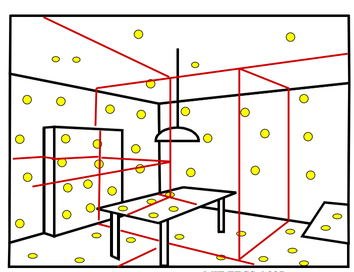
Photon mapping

- Preprocess: cast rays from light sources
- Store photons (position + light power + incoming direction)



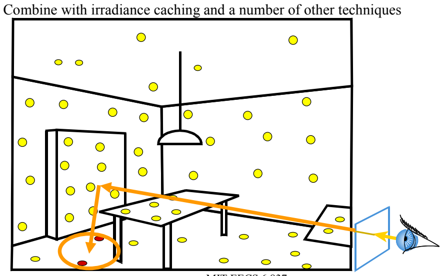
Photon map

- Efficiently store photons for fast access
- Use hierarchical spatial structure (kd-tree)



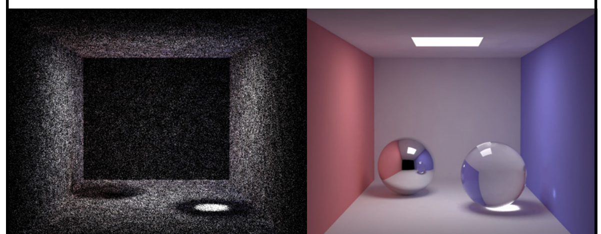
Photon mapping - rendering

- Cast primary rays
- For secondary rays
  - reconstruct irradiance using adjacent stored photon
  - Take the k closest photons
- Combine with irradiance caching and a number of other techniques



MIT EECS 6.837

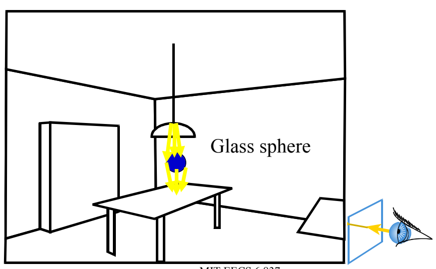
Photon map results



MIT EECS 6.837

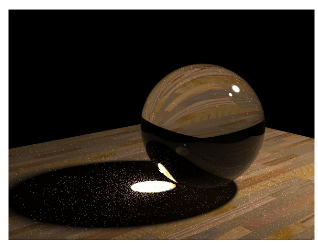
Photon mapping - caustics

- Special photon map for specular reflection and refraction



MIT EECS 6.837

- 1000 paths/pixel



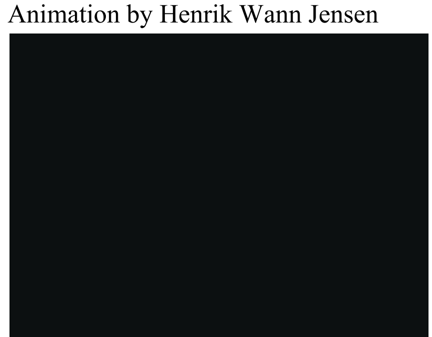
MIT EECS 6.837

- Photon mapping

MIT EECS 6.837

Photon mapping

- Animation by Henrik Wann Jensen



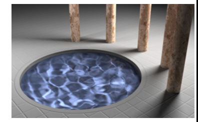
MIT EECS 6.837

Questions?

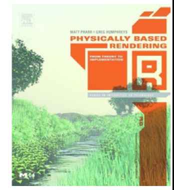
- Image by Henrik

References

- 6.839!
- Eric Veach's PhD dissertation
  http://graphics.stanford.edu/papers/veach_thesis/



- Physically Based Rendering
  by Matt Pharr, Greg Humphreys



References