EDA101 : Advanced Shading and Rendering

Stochastic Path Tracing and Image-based lighting

Michael Doggett
This is what we want:

Courtesy of Henrik Wann Jensen

Courtesy of Paul Debevec

Images courtesy of Illuminate Labs
Isn’t Whitted-style ray tracing enough?

• Does not give truly realistic images

• Why?
  – Does not solve the entire rendering equation
  – Miss several phenomenon:
    • Caustics
    • Indirect illumination (color bleeding, ambient lighting)
    • Soft shadows

• This lecture will present Global Illumination techniques: path tracing, image-based lighting and photon mapping
Light transport notation

Useful tool for thinking about global illumination (GI)

• Follow light paths
• The endpoints of straight paths can be:
  – L: light source
  – E: the eye
  – S: a specular reflection
  – D: a diffuse reflection
  – G: semi-diffuse (glossy) reflection

• Regular expressions can be used:
  – (K)+: one or more of K
  – (K)*: zero or more of K
  – (K)?: zero or one of K (book notation: K^{0..1})
  – (K | M): a K or an M event

Notation after Paul Heckbert, SIGGRAPH 90
Examples of light transport notation

- Path A: LDDDE
- Path B: LSDSDE
Light transport notation: why?

• The ultimate goal is to simulate all light paths: \( L(S|G|D)\times E \)

• Using this notation, we can find what Whitted ray tracing can handle:
  – \( LDS\times E \) | \( LS\times E = LD(S)\times E \)
    • Or if we include glossy surfaces: \( LD(G|S)\times E \)
  – This is clearly not \( L(S|G|D)\times E \)!
Radiance

- **Flux (radiant power):** $\Phi$ [W]
  - The total energy that flows from/to/through a surface
- **Irradiance:** $E = \frac{d\Phi}{dA}$ [W/m$^2$]
  - Incoming radiant power per unit surface area
- **Radiosity (radiant exitance)**
  - Same as irradiance but outgoing
- **Radiance:** $L(x \rightarrow \Theta)$
  - $L(x \rightarrow \Theta)$ – 5 dimensions: position ($x$) and direction ($\Theta$)
  - Important: captures "appearance" of objects
  - Flux per unit projected area ($E$) per unit solid angle

$$L = \frac{d^2\Phi}{d\omega dA^\perp} = \frac{d^2\Phi}{d\omega dA \cos \theta}$$

- Incoming flux spread out over larger area for big angles, the cosine term compensates for this
Rendering Equation

\[ L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + L_r(x \rightarrow \Theta) \]

\( L \) is radiance = most important quantity
  - captures "appearance" of objects in a scene.
  - what we store in a pixel.

Energy conservation - total outgoing radiance at a point is the sum of the **emitted** and **reflected** radiance. Read section 2.3.1 (skip "Transport Theory" if you want)

\[ L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta)L(x \leftarrow \Psi)\cos(N_x, \Psi)d\omega_\Psi \]

The equation above is the **Rendering Equation** : *most important equation in graphics*

It is a Fredholm equation of the second kind: \( L \) is both to the left, and inside the integral
Rendering Equation - $L_r$

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(x, \Psi \leftarrow \Theta) L(x \leftarrow \Psi) \cos(N_x, \Psi) d\omega_{\Psi}$$

Why not sample light sources separately? They give most of the appearance (usually)!

$$L_r(x \rightarrow \Theta) = L_{direct} + L_{indirect}$$

Direct illumination + Indirect illumination
Direct Illumination: $L_{\text{direct}}$

Some definitions:

$y = r(x, \Psi)$  
Is the closest positive intersection along ray that starts at $x$ and has direction $\Psi$

$V(x,y)$ is the **visibility** function: is 0 if $x$ occluded from $y$, otherwise 1.

Approach: integrate over the surface area instead of over the hemisphere (for full derivation, see section 2.6.2-2.6.3)

$L_{\text{direct}} = \int_A f_r(x, \Theta \leftarrow xy) L_e(y \rightarrow yx) V(x,y) G(x,y) dA_y$

The geometry term:

$G(x, y) = \frac{\cos(N_x, xy) \cos(N_y, yx)}{r_{xy}^2}$

$r_{xy} = \| x - y \|$  

$G(x,y)V(x,y)$ is often called "radiance transfer"
Indirect Illumination

\[ L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) L(x \leftarrow \Psi) \cos(N_x, \Psi) d\omega_\Psi \]

\[ L_r(x \rightarrow \Theta) = L_{direct} + L_{indirect} \]

\[ L_{indirect} = \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) L_i(x \leftarrow \Psi) \cos(N_x, \Psi) d\omega_\Psi \]

The incoming radiance at \( x \) in direction \( \Psi \) is the same as the outgoing radiance from \( y \) in direction \(-\Psi\)

\[ L_i(x \leftarrow \Psi) = L_r(r(x, \Psi) \rightarrow -\Psi) \]

\( y = r(x, \Psi) \) the ray-casting operation, finds the closest positive intersection along ray that starts at \( x \) and has direction \( \Psi \)
GI illustrated

Direct illumination + Indirect illumination

Recursion Illustrated:
(dashed lines are shadow rays)
\( L \) \(_{\text{indirect}} \)

- Change from integrating Cartesian coordinate (solid angle) to (hemi-)spherical coordinates:
  
  \[ x = r \sin \theta \cos \varphi \]
  \[ y = r \sin \theta \sin \varphi \]
  \[ z = r \cos \theta \]

- \( J \) is the Jacobian of the coordinate transform

\[
J = \begin{vmatrix}
\frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial \varphi} \\
\frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \varphi} \\
\frac{\partial z}{\partial r} & \frac{\partial z}{\partial \theta} & \frac{\partial z}{\partial \varphi}
\end{vmatrix} = \sin \theta
\]

\[
\int f(\Theta)d\omega_\Theta = \int_0^{2\pi} \int_0^{\pi/2} f(\varphi, \theta)Jd\theta d\varphi
\]

In programming assignments, we will still use Cartesian...
Monte Carlo Recap

\( X \) is stochastic random variable, drawn from PDF \( p(x) \)

\[
E[X] = \int xp(x)dx
\]

\[
E[f(X)] = \int f(x) p(x)dx
\]

How evaluate the integral of \( f(x) \) when \( x \) is drawn from PDF \( p(x) \)?

\[
\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}. \quad x_i \text{ drawn from PDF}
\]

\[
E[\langle I \rangle] = E\left[ \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \right] = \frac{1}{N} \sum_{i=1}^{N} E\left[ \frac{f(x_i)}{p(x_i)} \right]
\]

\[
= \frac{1}{N} \int \frac{f(x)}{p(x)} p(x) = \int f(x) dx = I
\]
Practical direct illumination using MC

\[ L_{direct} = \int_A f_r(x, \Theta \leftrightarrow \vec{x} \vec{y}) L_e(y \rightarrow \vec{y} \vec{x}) V(x, y) G(x, y) dA_y \]

\[ \approx \frac{1}{N_s} \sum_{i=1}^{N_s} f_r(x, \Theta \leftrightarrow x\vec{y}_i) L_e(y_i \rightarrow \vec{y}_i \vec{x}) V(x, y_i) G(x, y_i) \frac{p(y_i)}{p(y)} \]

The PDF, \( p(y) \) may be uniform, i.e., 1/A

See Figure 5.7 for pseudo code.
Russian roulette

• We need to be able to terminate recursion
• Bias is the error (possibly) introduced with some technique: $E[I_{MC}] - I$ (mean of MC minus real solution)
• Avoid incorrect lighting contributions (bias)
  – Light bounces around infinitely...
  – How to avoid infinite path lengths without bias?

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

\[ I_{RR} = \int_0^P \frac{1}{P} f(x/P) \, dx, \quad P \leq 1 \]

\[
< I_{RR} > = \begin{cases} 
  \frac{1}{P} f(x/P), & \text{if } x \leq P \\
  0, & \text{if } x > P 
\end{cases}
\]

\[
E[< I_{RR} >] = (1 - P) \cdot 0 + P \cdot \frac{E[I]}{P} = E[I]
\]
Russian roulette for indirect illumination

- That is, we can stop recursion with a probability of $\alpha = 1 - P$, where $\alpha$ is the absorption probability.

- See figure 5.4 for pseudocode.

- If photon not absorbed, diffuse or specular?
  - Random number $r$ from $[0,1]$
  - If ($r < \rho_d$) then shoot diffuse ray
  - Else if ($r < \rho_d + \rho_s$) then shoot specular ray
  - Else absorb

- Use average reflectance
  - E.g., $\rho_d = (\rho_{d,\text{red}} + \rho_{d,\text{green}} + \rho_{d,\text{blue}})/3$
  - Or special reflectivity parameter of material
    - As done in programming assignments 1 & 2
Practical indirect illumination

• How to choose random direction on the hemisphere?
  – Uniformly distributed

\[
\int_{\Omega} f(\Theta) d\Omega = \int_{0}^{2\pi} \int_{0}^{\pi/2} f(\varphi, \theta) \sin \theta d\theta d\varphi
\]

  I.e., choose random \( \theta \) and \( \varphi \), and then use

  The \( \sin \theta \) to correct for non-uniform distribution

  Not efficient!

• Instead:

\[
\varphi = 2\pi r_1 \quad r_1, r_2 \text{ are random numbers in } [0,1]
\]

\[
\theta = \arccos(r_2)
\]

\[
\Psi_i = (\varphi, \theta)
\]

\[
L_{\text{indirect}} = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \Psi_i \leftrightarrow \Theta)L_r(r(x, \Psi_i) \rightarrow -\Psi_i)\cos(N_x, \Psi_i)}{p(\Psi_i)}
\]

PDF for \( p(\Psi_i) \) is \( 1/2\pi \)
Importance Sampling

• Choose the PDF proportional to a factor in the equation that we want to integrate

• For rendering equation, this can be:
  – \( \cos(\Psi_i, N_x) \)
    • pages 65-66, and page 137 ("Cosine Sampling")
    • Included in programming assignment 2
  – BRDF
  – Incident radiance
  – Combination...
Path tracing algorithm

• Shoot rays through each pixel
  – Compute direct illumination
  – Compute indirect illumination (recursive)
    • Use Russian roulette to terminate recursion without bias
    • Use Hemispherical Monte Carlo sampling

• Important for path tracing:
  – Shoot **one** ray to the light sources
  – Shoot **one** ray over the hemisphere
  – Reason: do not want to spend more rays that are deeper down in the paths
Path tracing example

From Kevin Beason’s GI in 99 lines of C++  
http://www.kevinbeason.com/smallpt/

© 2008 Tomas Akenine-Moller
Image-Based Lighting

• Enables very realistic lighting
  – In a simple way

• Not only primary light sources illuminate a point

• Basic idea:
  – Capture an image of the environment
  – Each pixel in this image is treated as a light source
  – During rendering, use these lights!
Example images

Images courtesy of Paul Debevec
More example images

Images courtesy of Henrik Wann Jensen

© 2008 Tomas Akenine-Möller
Capturing environment maps (light probes)

• Many ways:
  – Take a photograph of a highly reflective sphere
  – Fish-eye lens: 2 images needed
  – Special device: rotating image sensors

• Need high resolution images!
  – 3 bytes not enough for RGB $\rightarrow$ 3 floats

• Why?
  – Because the range of "brightness" values is larger than [0,255]
Example of image with high dynamic range

- Lowest intensity is about 20
- Highest intensity is about 12,000
- Dynamic range is 1:600!
High Dynamic Range Images

• Dynamic range of image:
  – brightest region/dimmest region

• Very briefly:
  – A digital camera uses a CCD (charge coupled device) array as a sensor. However, these values are mapped, using a non-linear function in order to display them on screen
  – To recover the entire range of the image, take several photos of the same scene with different exposure times
  – Non-linear function: from an overdetermined system using SVD
  – Use function to recover HDR image (one float per RGB)
  – There are also cameras that can take 12 bits per color component directly

• For more details: Debevec & Malik, "Recovering High Dynamic Range Radiance Maps from Photographs," SIGGRAPH 97, pp. 369-378.

© 2008 Tomas Akenine-Möller
Example of image with different exposure times

Images courtesy of Paul Debevec
How to capture an image of the environment?

• Use simplest method: image of highly reflective sphere
  – We get an image of the entire environment, if we can assume that the camera is infinitely far away (or if the sphere is very small)
Examples of environment maps

Dynamic Range: 1:200,000

Images courtesy of Paul Debevec
How to use environment map

Environment map

Often need many rays!

May need varying number of rays depending on frequency content of env. map

Note: in path tracing, we only shoot one ray at every intersection point, but many rays through each pixel

© 2008 Tomas Akenine-Möller
IBL: some notes

• Even simpler to code with a cube map of the environment
  – Transform sphere map into cube map once before rendering
  – Debevec uses angular map (can use sphere map too)
  – That article is compulsory literature for this course!!
  – Check out light probe gallery + HDR shop
• How to do it fast?
  – Probably takes weeks to implement though...
Photon mapping
State-of-the-art in GI

- Developed by Henrik Wann Jensen (started 1993)
- A clever two-pass algorithm:
  - 1: Shoot photons from light sources, and let them bounce around in the scene, and store them where they land
  - 2: "Ray tracing"-like pass from the eye, but gather the photons from the previous pass
- Advantages:
  - Fast
  - Handles arbitrary geometry (as do path tracing)
  - All global illumination effects can be seen
  - Little noise
The first pass: Photon tracing

- Store illumination as points (photons) in a "photon map" data structure.
- In the first pass: photon tracing
  - Emit photons from light sources
  - Trace them through scene
  - Store them in photon map data structure
- More details:
  - When a photon hits a surface (that has a diffuse component), store the photon in photon map
  - Then use Russian roulette to find out whether the photon is absorbed or reflected
  - If reflected, the shoot photon in new random direction

© 2008 Tomas Akenine-Möller
Photon tracing

- Should not store photon at specular surfaces, because these effects are view dependent
  - only diffuse effect is view independent
- Some diffuse photons are absorbed, some are scattered further
- A photon = the incoming illumination at a point

This type of marker is a stored photon

© 2008 Tomas Akenine-Möller 35
The photon map data structure

• Keep them in a separate (from geometry) structure
• Store all photons in kD-tree
  – Essentially an axis-aligned BSP tree, but we must alter splitting axis: x,y,z,x,y,z,x,y,z, etc.
  – Each node stores a photon
  – Needed because the algorithm needs to locate the $n$ closest photons to a point
• A photon:
  – float x, y, z;
  – char power[4];  // essentially RGB, with more accuracy
  – char phi, theta;  // compact representation of incoming direction
  – short flag;  // used by KD-tree (stores which plane to split)
• Create balanced KD-tree – simple, done once.
• Photons are stored linearly in memory:
  – Parent node at index: $p$, left child at: $2p$, right child: $2p + 1$
What does it look like?

- Stored photons displayed:
Density estimation

- The density of the photons indicate how much light that point receives
- Radiance is the term for what we display at a pixel
- Rather complex derivation skipped (see Jensen’s book) …
- Reflected radiance at point $x$:

$$L(x, \omega) \approx \frac{1}{\pi r^2} \sum_{1}^{n} f_r(x, \omega_p \leftrightarrow \omega) \Phi_p(x, \omega_p)$$

  - $L$ is radiance in $x$ in the direction of $\omega$
  - $r$ is radius of expanded sphere
  - $\omega_p$ is the direction of the stored photon
  - $\Phi_p$ is the stored power of the photon
  - $f_r$ is the BRDF
Two-pass algorithm

• Already said:
  – 1) Photon tracing, to build photon maps
  – 2) Rendering from the eye using photon maps

• Pass 1:
  – Use two photon maps
  – A caustics photon map (for caustics)
    • Reflected or refracted via a surface to a diffuse surface
    • Light transport notation: LS+D
  – A global photon map (for all illumination)
    • All photons that landed on diffuse surfaces
    • L(S | D)*D
Caustic map and global map

Caustic map:
- Send photons only towards reflective and refractive surfaces
  - Caustics is a high frequency component of illumination
  - Therefore, need many photons to represent accurately

Global map - assumption: illumination varies more slowly
Pass 2: Rendering using the photon map

• Render from the eye using a modified ray tracer
  – A number of rays are sent per pixel
  – For each ray evaluate four terms
    • **Direct illumination** (light reaches a surface directly from light source)... may need to send many rays to area lights. Done using standard ray tracing.
    • **Specular reflection** (also evaluated using ray tracing, possibly, with many rays sent around the reflection direction)
    • **Caustics**: use caustics photon map
    • **Indirect illumination** (multiple diffuse reflections) (gives color bleeding): use the photon map for reflected rays.
Images of the four components

- These together solves the entire rendering equation!
Indirect illumination: Use the photon map

• To evaluate indirect illumination at point $p$:
  – Send several random rays out from $p$, and grow spheres at contacts
  – May need several hundreds of rays to get good results.
Locate n closest photons  (After Henrik Wann Jensen)

// locate n closest photons around point "pos"
// call with "locate_photons(1)", i.e., with the root as in argument
locate_photons(p)
{
  if(2p+1 < number of photons in photon map structure)
  {
    // examine child nodes
    delta=signed distance to plane of node n
    if(delta<0)
    {
      // we’re to the "left" of the plane
      locate_photons(2p);
      if(delta*delta < d*d)
        locate_photons(2p+1);  //right subtree
    }
    else
    {
      // we’re to the "right" of the plane
      locate_photons(2p+1);
      if(delta*delta < d*d)
        locate_photons(2p);   // left subtree
    }
  }
  delta=real distance from photon p to pos
  if(delta*delta < d*d)
  {
    // photon close enough?
    insert photon into priority queue h
    d=distance to photon in root node of h
  }
}

// think of it as an expanding sphere, that stops expanding when n closest
// photons have been found
More literature

• Great book on global illumination and photon mapping:

• Check Henrik’s website:
  – [http://graphics.ucsd.edu/~henrik/](http://graphics.ucsd.edu/~henrik/)
  – "Global Illumination using Photon Maps"
    Henrik Wann Jensen
    In *"Rendering Techniques '96"*. Eds. X. Pueyo and P. Schröder. Springer-Verlag, pages 21-30, 1996
Inspiration...

- Subsurface scattering
- Participating media
- Diffraction: CDs etc
- Different forms of BRDFs
- Acceleration algorithms

Images courtesy of Henrik Wann Jensen
Reading list

• Advanced Global Illumination
  – Section 2.6
  – Section 3.5.1
  – Section 5.1-5.6
  – Section 7.2, 7.6
  – Section 8.2.1
  – Appendix B.3

  – Download from www.debevec.org