CMSC 427: Global Illumination Models
(Guest Lecturer: Dave Mount)

Reading: Sect. 10.12-10.14 in Hearn & Baker.
Overview:
- Global Illumination Models
- Rendering Equation
- Radiosity
- Photon Mapping

Summary of Illumination Models

You have seen:

Local illumination model: Phong
- Ambient
- Diffuse
- Specular
- Point light sources
- **No**: shadows, inter-object light reflection.

"More global" illumination model: Ray Tracing
- Shadows
- Area light sources (via distributed ray tracing)
- **No**: inter-object light reflection

Tradeoffs:
- Improvements in **fidelity** come at the expense of **computational complexity**.

Ray Tracing

Ray tracing: More accurate than Phong, but not without its own limitations.

Strengths:
- Easy to implement.
- General and extensible.
- Better handling of global issues (shadows, reflection, etc.).

Shortcomings:

Optimizing not easy: Involves non-trivial data structures.

Not truly global: Relies on the Phong illumination model to compute illumination at each point. Ignores inter-object light reflection.

Too "Specular": Ray-traced images are best for highly specular objects (glass and metalic balls), but specular reflection is not common in typical real-world scenes.

Chapter 14, Slide 3

Global Illumination and Light Transport

Global Illumination and Light Transport:

Describe the flow of light energy in a scene, including **inter-object reflections**.

The Rendering Equation:

The theoretical basis for **light energy transport**.

Conservation of Energy:
- **Global conservation**:
  - Assumes a **closed environment**.
  - Energy input = energy output.
  - Rest is converted to **heat**.
- **Local conservation**:
  - Incident energy must be either **reflected** or **absorbed**

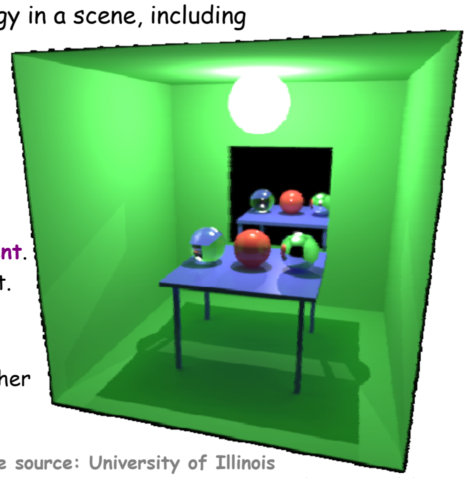


Image source: University of Illinois

Chapter 14, Slide 4

The Rendering Equation

Rendering Equation: (Kajiya 86)

- Describes the **flow of light energy** throughout a scene, assuming that all objects of a scene (not just light sources) may **reflect light**.
- It relates the **light energy** $L_o(\mathbf{x}, \mathbf{w})$ that is reflected outwards from a point **x** in direction **w** as a function of:
  - **emitted light energy** $L_e(\mathbf{x}, \mathbf{w})$ (if this object is a light source), and
  - the **total light energy** $L_i(\mathbf{x}, \mathbf{w}')$ **received** at **x** from all other directions **w**′ which is then **reflected** outwards in direction **w**.
- There are a number of **variants**, depending on the assumptions made.
- **Radiance form** of the rendering equation:

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

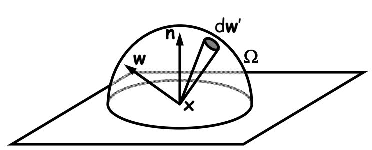
We will explain this next.

The Rendering Equation

Rendering Equation: What's what.

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

- **x** is a **surface point**. **n** is the **normal**. **w** is a **unit vector** (direction).
- $L_o(\mathbf{x}, \mathbf{w})$ is the light energy **reflected outwards** from point **x** in direction **w**.
- $L_e(\mathbf{x}, \mathbf{w})$ is the light energy **emitted** at **x** in direction **w**.
- Ω denotes the **hemisphere** above the surface patch at **x**. The integral is taken over all differential directional elements d**w**′ on Ω.
- $L_i(\mathbf{x}, \mathbf{w})$ is the **incoming light energy** incident on **x** arriving from direction **w**′.
- $f_r(\mathbf{x}, \mathbf{w}', \mathbf{w})$ is the **fraction** of light energy arriving at **x from** direction **w**′ , that is reflected **to** direction **w**. (In general this depends on **w** and **w**′.)
- The (**w**′ · **n**) term captures the **attenuation** of arriving light, similar to Lambert's law. (The bigger the angle, the less the energy per unit area.)



The Rendering Equation

Is this the Holy Grail?

- A **perfect implementation** of this rule would result in accurate lighting.
- Virtually all illumination models only provide an **approximation**.

Can we solve the rendering equation? Not practical for real-world scenes:

- Need to model the **bidirectional reflectance term**, $f_r(\mathbf{x}, \mathbf{w}', \mathbf{w})$. Not hard for pure diffuse and specular reflectors, but harder for real-world materials. (**BRDF**s)
- The **incoming light term** $L_i(\mathbf{x}, \mathbf{w})$ requires that we determine what is visible from $\mathbf{x}$ in direction $\mathbf{w}$, which would involve **hidden surface removal**, from every point in the scene.
- This is **not just one equation**. The outgoing light from each point affects the incoming light to all other points. This is a huge **system of integral equations**, one variable for each point and each direction about that point.

Computational Approaches:

Path Tracing: Attempts to trace all light rays in a scene.

Photon Mapping: Deposits light energy on surfaces for later collection.

Radiosity: Simulation of light transport under a steady-state assumption, assuming diffuse reflection.

Radiosity

Radiosity: A method for implementing a **global illumination model**.

- Simulates lighting due to **inter-object reflections**.
- Principally for **diffuse surfaces** (that is, Lambertian reflectors).
- Models **view-independent illumination**.
- Can generate **diffuse/soft shadows**, **color bleeding**.

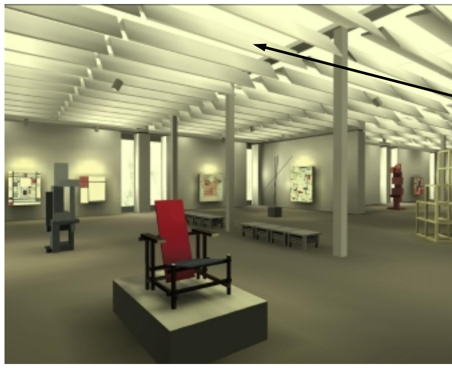
Color bleeding on walls from floor.

Image source: Lightscape Inc.

Radiosity: Basic Elements

Basic Elements of Radiosity:

- Based on **conservation of light energy**.
- Assumes **area light sources**.



Most light comes from baffles in the ceiling.

Image source: Cornell University

Radiosity: Basic Elements

Definition: **Radiosity** is the rate at which energy leaves a surface either through:

- **Emission** or
- **Reflection**

Computational Approach:

- Model the scene as a set of **small surface patches**, each assumed to have constant (but unknown) radiosity.
- Set up a **linear system** (based on a straightforward approximation to the rendering equation) that relates the radiosity of each patch to some function of its **surrounding patches**.
- **Solve** this linear system (by standard numerical methods) to determine the radiosity of each surface patch.
- **Render** the scene, using these radiosity values.

The Radiosity Equation

Radiosity Equation:

- Expresses the **radiosity** of each **differential patch** of a surface patch i in terms of the radiosity's of **surrounding patches**:

$$B_i dA_i = E_i dA_i + \rho_i \int_j B_j F_{ji} dA_j$$

Energy emitted by patch i. ($E_i dA_i$)

Total energy incoming to patch i. ($\rho_i \int_j B_j F_{ji} dA_j$)

- where:
  - B_i: the **radiosity** for patch i.
  - dA_i: a **differential area** element on patch i.
  - E_i: rate of **emission** of patch i (for light sources).
  - ρ_i: **reflectivity** (fraction of incoming light that is reflected) for patch i.
  - F_{ij}: a dimensionless term, called the **form factor**, which indicates the fraction of energy leaving patch j that arrives directly at patch i. (More about this later.)
- **Diffuseness assumption**: reflectivity does not depend on direction.
- This represents **three equations**, one for red, green, and blue, but we will just assume **monochromatic** light for now.

Chapter 14, Slide 11

Form Factor

Form Factor: F_{ji} is the **fraction** of energy leaving patch j that arrives (directly) at patch i.

It accounts for:

- **Distance** between surfaces (and attenuation due to this).
- Relative **sizes** of the patches.
- Relative **orientations** of the patches.
- **Visibility** of the patches (that is, the lack of occluding objects).
- We will discuss this in greater detail later.

Global Knowledge: Computing form factors requires **complete knowledge** of the scene geometry.

Chapter 14, Slide 12

Discretization

Discretization:
- To avoid dealing with integral equations, we discretize the scene into **small patch elements** of **constant radiosity**. Replace $dA_i \rightarrow A_i$.

$$B_i dA_i = E_i dA_i + \rho_i \int_j B_j F_{ji} dA_j \quad \rightarrow \quad B_i A_i = E_i A_i + \rho_i \sum_j B_j F_{ji} A_j \qquad (1)$$

Reciprocity relationship of form factors:
- By symmetry we have: $F_{ij} A_i = F_{ji} A_j$, and therefore $F_{ij} = F_{ji} A_j / A_i$.
- Dividing Eq.(1) by A_i, we have:

$$B_i = E_i + \rho_i \sum_j B_j F_{ij} \qquad \forall i$$

- Rearranging terms yields (for all surface patches i):

$$B_i - \rho_i \sum_j B_j F_{ij} = E_i \qquad \forall i$$

- We assume we are given E_i and ρ_i and can compute F_{ij}. What remains is a system of **linear equations** in the **variables** B_i. (**Next slide.**)

Chapter 14, Slide 13

Radiosity System of Equations

Final Radiosity System:
- In summary, we have:

$$B_i - \rho_i \sum_j B_j F_{ij} = E_i \qquad \forall i$$

- We know everything but the B_i's, the radiosity values.

Matrix form:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$$

Solving the System: This is a **very large** system.
- Solving directly (e.g., by Gauss elimination) is **expensive** ($O(n^3)$ time).
- In practice we take advantage of the structure of the matrix (strictly diagonally dominant) to approximate the solution by **iterative methods** (e.g., Gauss-Seidel).

Chapter 14, Slide 14

Final Rendering

Drawing the Final Scene:

- **View Independence**: Radiosities are **view independent**. Although this computation is expensive, we may **do it only once** (assuming lights do not change).
- **Radiosity as color**: No need for additional lighting computations (unless you need specularity). Simply **color each patch with its radiosity**, after multiplying by the object's surface color.
- **Smoothing**: Simply rendering the patches this way produces "**tiled**" appearance. Instead interpolate the radiosity values at each vertex and smooth using **Gourad shading**.

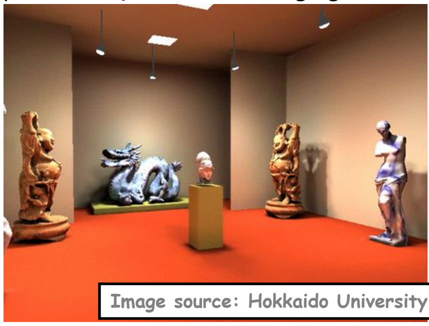


Image source: Hokkaido University

Computing Form Factors

How to compute the **Form Factor** F_{ij}?

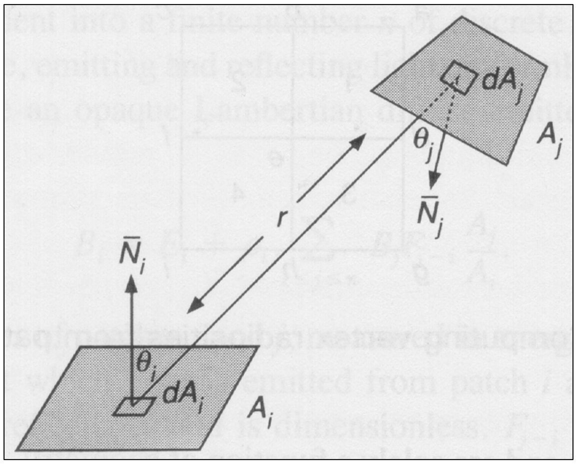


Image courtesy, Foley, van Dam, Feiner, Hughes

Computing Form Factors

How to compute **Form Factor** F_{ij}?

- Recall that F_{ij} is the **fraction** of energy **leaving patch i** that **arrives directly at patch j**.
- Consider two **patches** A_i and A_j, and let dA_i and dA_j be **differential elements** on these patches.
- Let r be the **distance** between the patches.
  - The fraction of energy falls off as $1/r^2$.
- Let θ_i and θ_j be the **angles** between the line joining the centers of dA_i and dA_j and the respective **surface normals**.
  - The energy is highest if the surfaces face each other directly, and otherwise falls off as $(\cos\theta_i \cdot \cos\theta_j)$ (by Lambert's law).
- Let H_{ij} be **visibility flag**: $H_{ij} = 1$ if dA_i and dA_j are visible and 0 otherwise.
- **Bottom line**:

$$F_{ij} \leftarrow \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij}\, dA_i dA_j$$

Chapter 14, Slide 17

Computing Form Factors

We can interpret the form factor F_{ij} **geometrically** by:

- project the patch A_j onto a **unit hemisphere** centered about dA_i, to account for the $(\cos\theta_j)/r^2$ term, and
- **project onto the plane**, to account for the $\cos\theta_i$ term.
- The resulting area is **proportional** to F_{ij}.
- Unfortunately, this is still **not easy** to compute, but it suggests… **(next slide)**

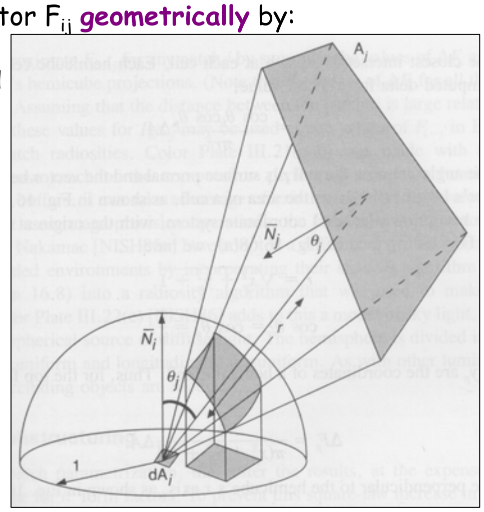


Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 14, Slide 18

Hemicube Approach for Form Factors

Hemicube Approximation:

- We can use **graphics hardware** to **speed up** the integration.
- Render the entire scene onto the 5 sides of a **hemicube** centered at dA_i.
- **Sum** the contributions from each **pixel** to get the form factors.
- Some **further processing** is needed to capture the $(\cos\theta_i \cdot \cos\theta_j)$ and $1/r^2$ terms.

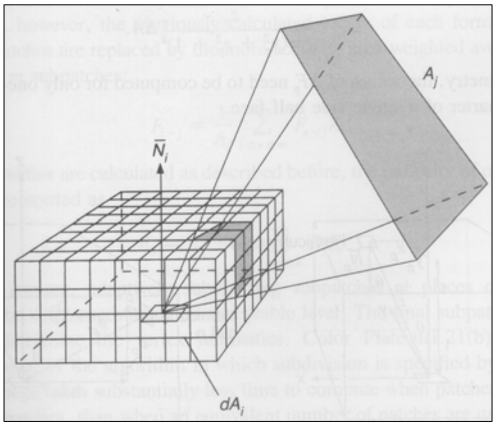


Image courtesy, Foley, van Dam, Feiner, Hughes

Progressive Refinement Radiosity

Progressive Refinement Radiosity:

- Solving the radiosity system takes a **long time**. This approach computes an **approximate solution** of the system of linear equations $\mathbf{A} \cdot \mathbf{b} = \mathbf{e}$.

Basic Idea:

- Identify the **brightest patch** in the environment and "**shoot**" (i.e., distribute) its energy to the **other patches** that can see it.
- This is equivalent to computing **only those rows** of the form-factor matrix A that correspond to the **brightest patches**.
- Repeat on the **next brightest**, and so on. (Note that the brightness of patches changes with each shoot.)

Performance:

- In practice, this approach results in a **fast convergence** to the solution without computing all the rows of **A**.

Progressive Refinement Radiosity

For each patch i, let

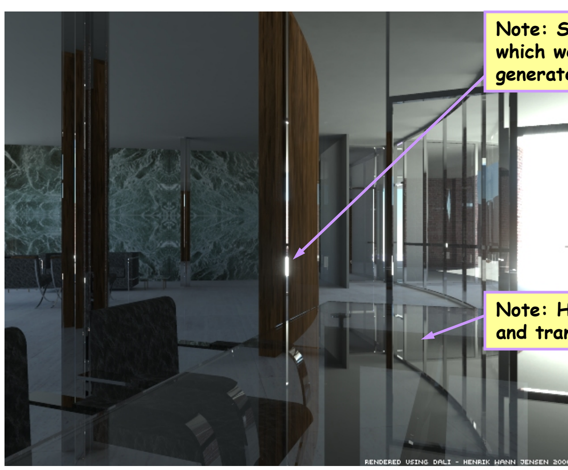
- Let B_i be its (current) radiosity.
- Let ΔB_i be its increase in radiosity.

Algorithm: Let ε be a small constant > 0.

```
for (all i) { Bi ← ΔBi ← Ei }      // initial radiosity is the emission value
while ( max ( ΔBi ) > ε ) {         // stop when values converge
    select patch i with maximum un-shot energy ΔBi
    for each patch j do {           // shoot energy from patch i to j
        ΔR ← ρj ΔBi Fij Ai/Aj       // energy contribution from patch i to j
        ΔBj ← ΔBj + ΔR              // increase unshot energy for patch j
        Bj ← Bj + ΔR                // accumulated energy at patch j
    }
    ΔBi ← 0                         // patch i is now "shot-out"
}
```


Photon-Mapping

Photon mapping: another approach to global illumination.



Note: Specular highlights, which were not easy to generate using radiosity.

Note: Handles reflection and transparency.

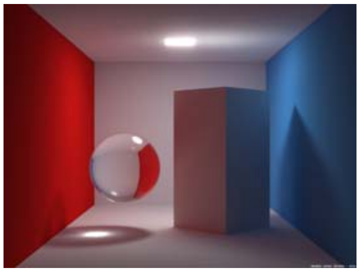
Standard Photon Map

Photon Mapping: A number of advantages over radiosity.

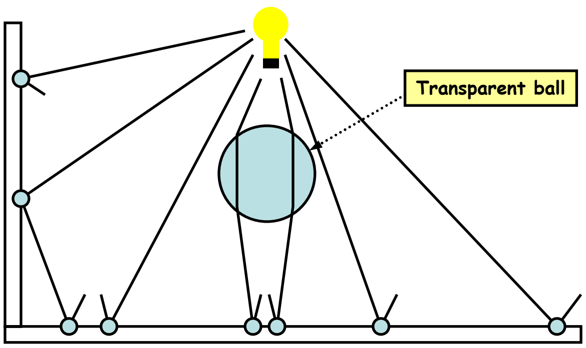
- Can deal with **non-diffuse reflective surfaces** (specular).
- Can deal with light transmission through **reflective** and **transparent objects** (and caustics).
- Can deal with **curved object geometries**.

Basic Algorithm: Two-pass approach:

- **Photon trace**: Simulate propagation of photons from light source onto surfaces.
- **Rendering**: Draw the objects using illumination information from the photon trace.



Photon Trace

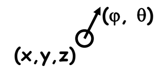


- For 100 photons emitted from 100W source, each photon initially carries 1W.
- Propagate this radiant flux through scene using **Monte-Carlo** (randomized sampling) **methods**.

Estimating incident flux

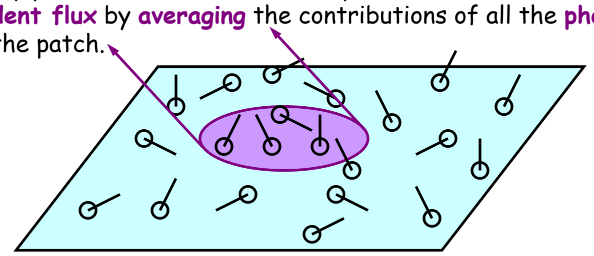
Photon Representation: Each photon stores:

- **Location** (x, y, z): where the photon finally hits.
- **Power** (color and brightness)
- **Incident direction** (φ, θ): for determining specular reflection.



Incident Flux: The amount of energy passing through a region in some direction.

- At any patch of surface and for any direction, we can estimate the **incident flux** by **averaging** the contributions of all the **photons** that hit the patch.



Chapter 14, Slide 25

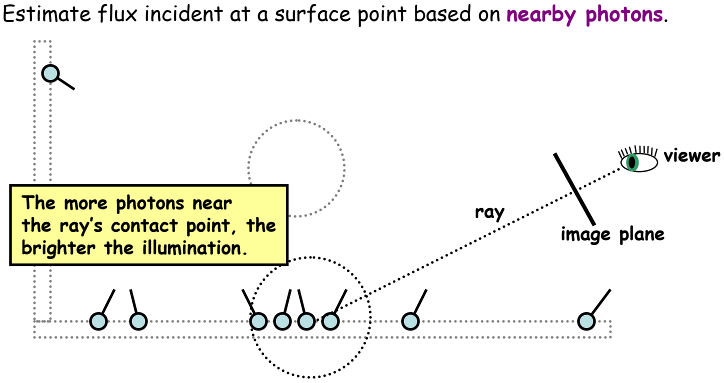
Photon Storage and Rendering

Photon Storage:

- Store photon information in a 3-d point **kd-tree**, called the **photon map**.
- Photon storage is **decoupled** from object geometry. They just float in space.

Render: (e.g., by ray tracing)

- Estimate flux incident at a surface point based on **nearby photons**.



Chapter 14, Slide 26

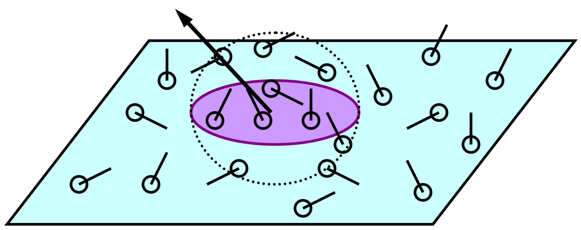
Radiance Estimate

Radiance Estimate : At a surface point (x, y, z).

Grow ball: about (x, y, z) until it contains a reasonable number of photons.

Estimate surface area: Compute intersection of the ball with the surface with plane to **estimate area** of surface patch.

Radiance: The **total contribution** of the **photons** in the ball **divided by** the patch **surface area** gives the final radiance (brightness).



Summary

Summary:

- Global Illumination Models
- Rendering Equation
- Radiosity
- Photon mapping