Global Illumination

CS334

Daniel G. Aliaga
Department of Computer Science
Purdue University

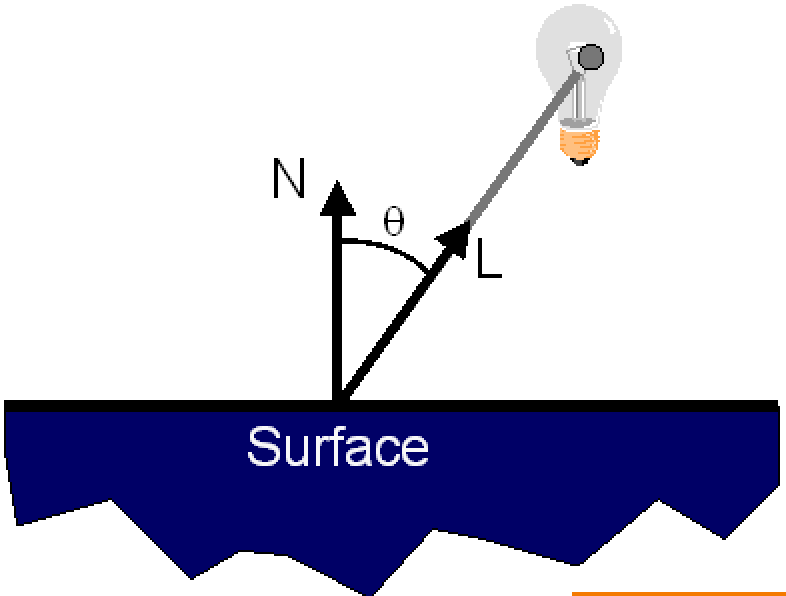
Recall: Lighting and Shading

- Light sources
  - Point light
    - Models an omnidirectional light source (e.g., a bulb)
  - Directional light
    - Models an omnidirectional light source at infinity
  - Spot light
    - Models a point light with direction

- Light model
  - Ambient light
  - Diffuse reflection
  - Specular reflection

Recall: Lighting and Shading

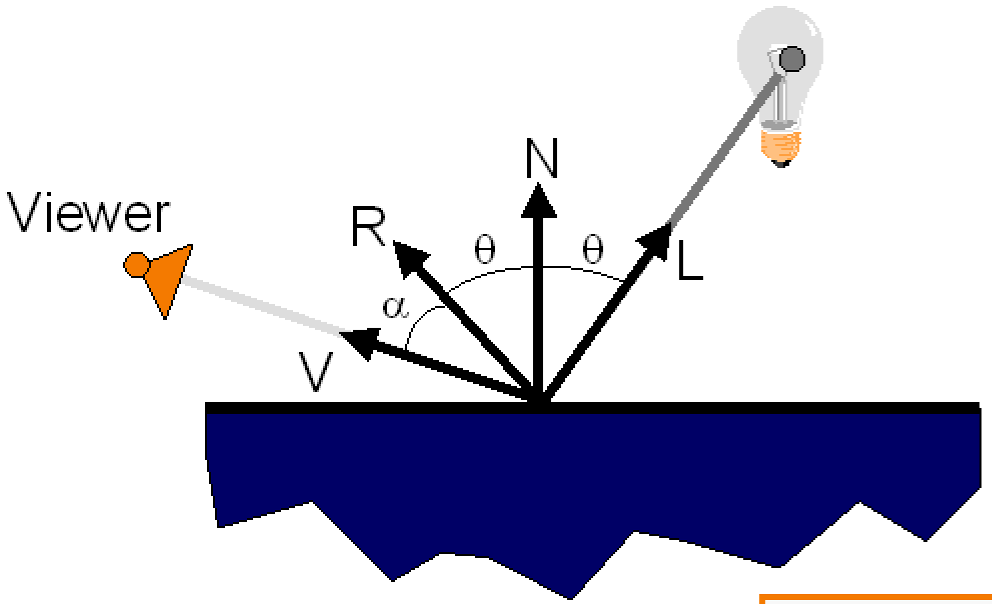
- Diffuse reflection
  - Lambertian model



$$I_D = K_D (N \bullet L) I_L$$

Recall: Lighting and Shading

- Specular reflection
  - Phong model



$$\boxed{I_S = K_S (V \bullet R)^n I_L}$$

Recall: Lighting and Shading

- Well….there is much more

For example…

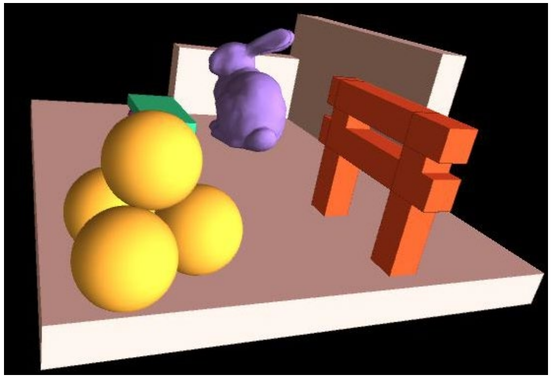
- Reflection -> Bidirectional Reflectance Distribution Functions (BRDF)
- Diffuse, Specular -> Diffuse Interreflection, Specular Interreflection
- Color bleeding
- Transparency, Refraction
- Scattering
  - Subsurface scattering
  - Through participating media
- And more!

Illumination Models

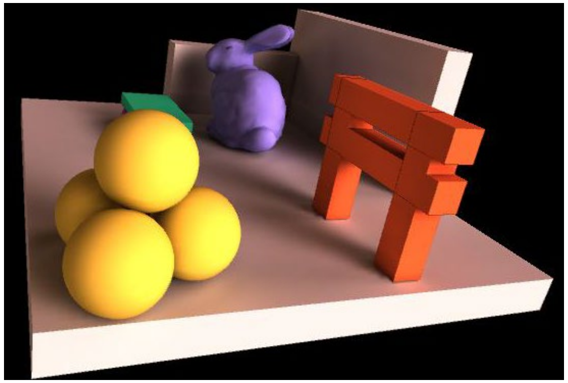
- So far, you considered mostly local (direct) illumination
  - Light directly from light sources to surface
  - No shadows (actually is a global effect)
- Global (indirect) illumination: multiple bounces of light
  - Hard and soft shadows
  - Reflections/refractions (you kinda saw already)
  - Diffuse and specular interreflections

Welcome to Global Illumination

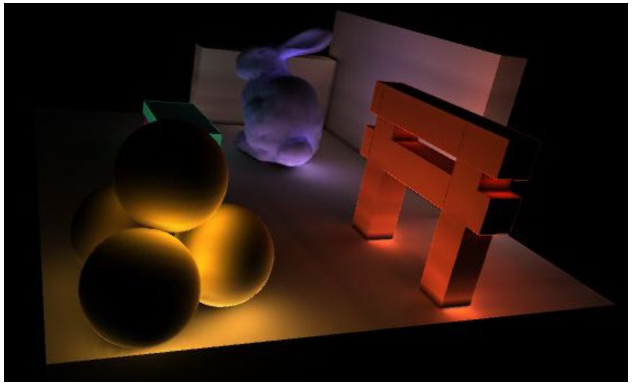
- *Direct illumination* + *indirect illumination*; e.g.
  - Direct = reflections, refractions, shadows, …
  - Indirect = diffuse and specular inter-reflection, …



direct illumination



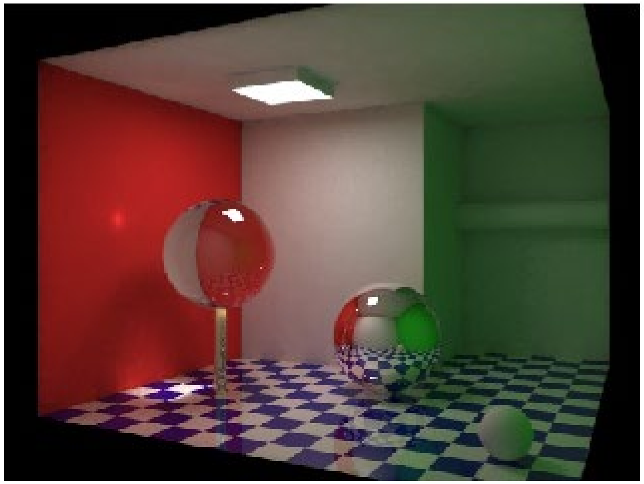
with global illumination



only diffuse inter-reflection

Global Illumination

- *Direct illumination* + *indirect illumination*; e.g.
  - Direct = reflections, refractions, shadows, …
  - Indirect = diffuse and specular inter-reflection, …



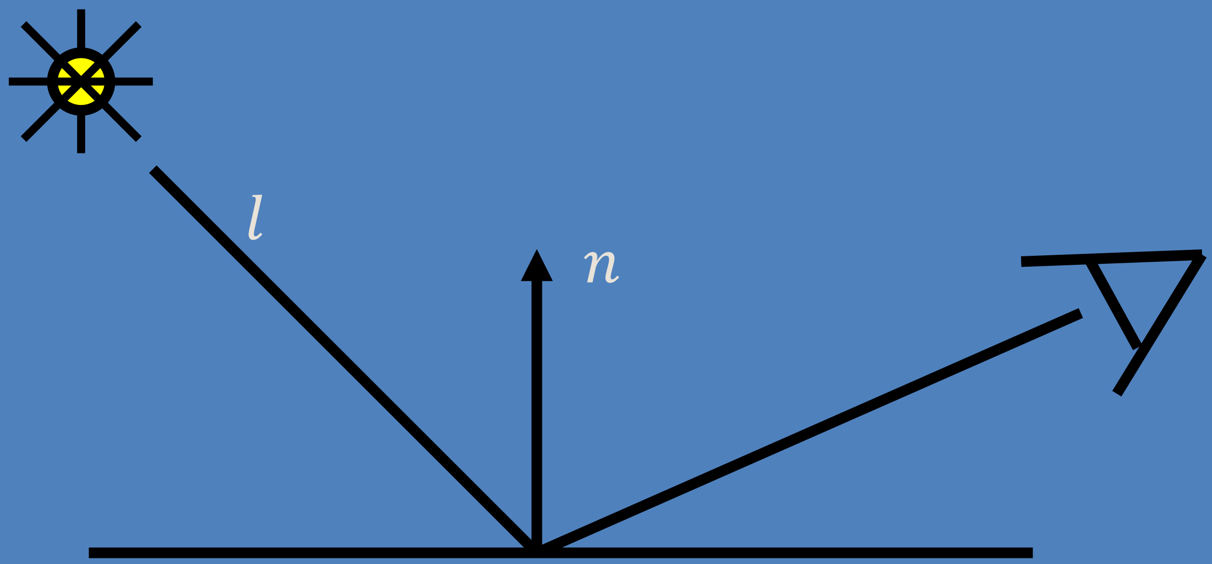
Reflectance Equation

- Lets start with the diffuse illumination equation and generalize…
- Define the all encompassing reflectance equation…
- Then specialize to the subset called the rendering equation…

Reflectance Equation

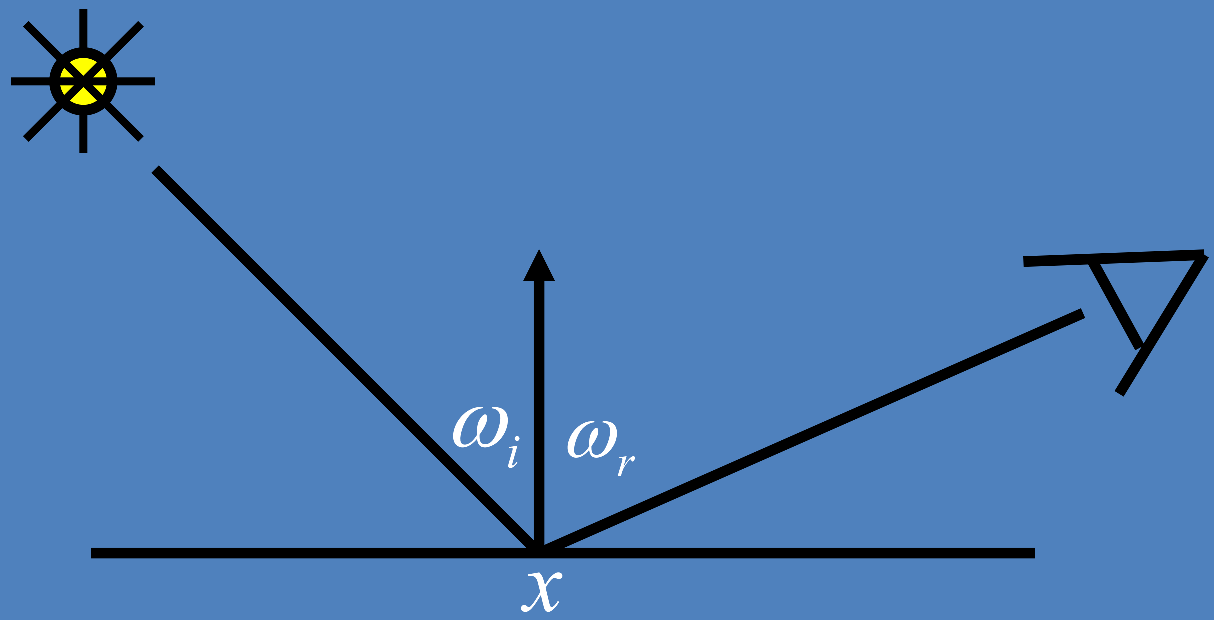
diffuse_illumination = $0 \quad + \quad I_L \qquad K_D \qquad l \cdot n$

Reflectance Equation



diffuse_illumination = $0 \quad + \quad I_L \quad K_D \quad l \cdot n$

Reflectance Equation

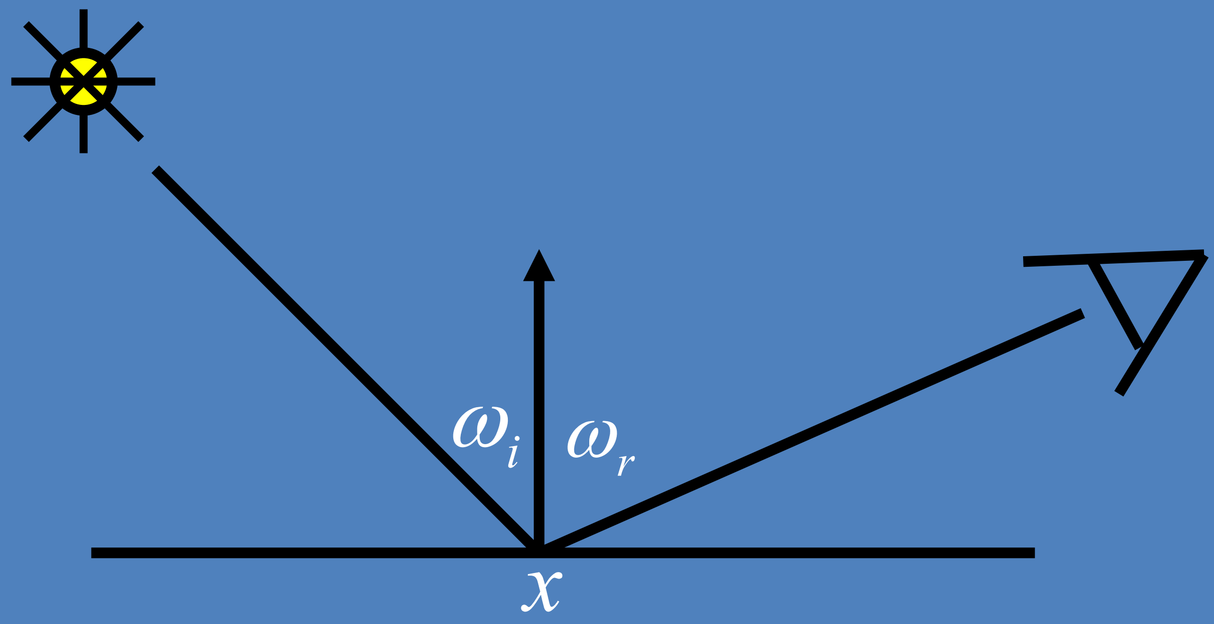


$$L_r(x,\omega_r) = L_e(x,\omega_r) + L_i(x,\omega_i)\, f(x,\omega_i,\omega_r)(\omega_i \; n)$$

diffuse_illumination = $0 \quad + \quad I_L \quad K_D \quad l \cdot n$

[Slides with help from Pat Hanrahan and Henrik Jensen]

Reflectance Equation

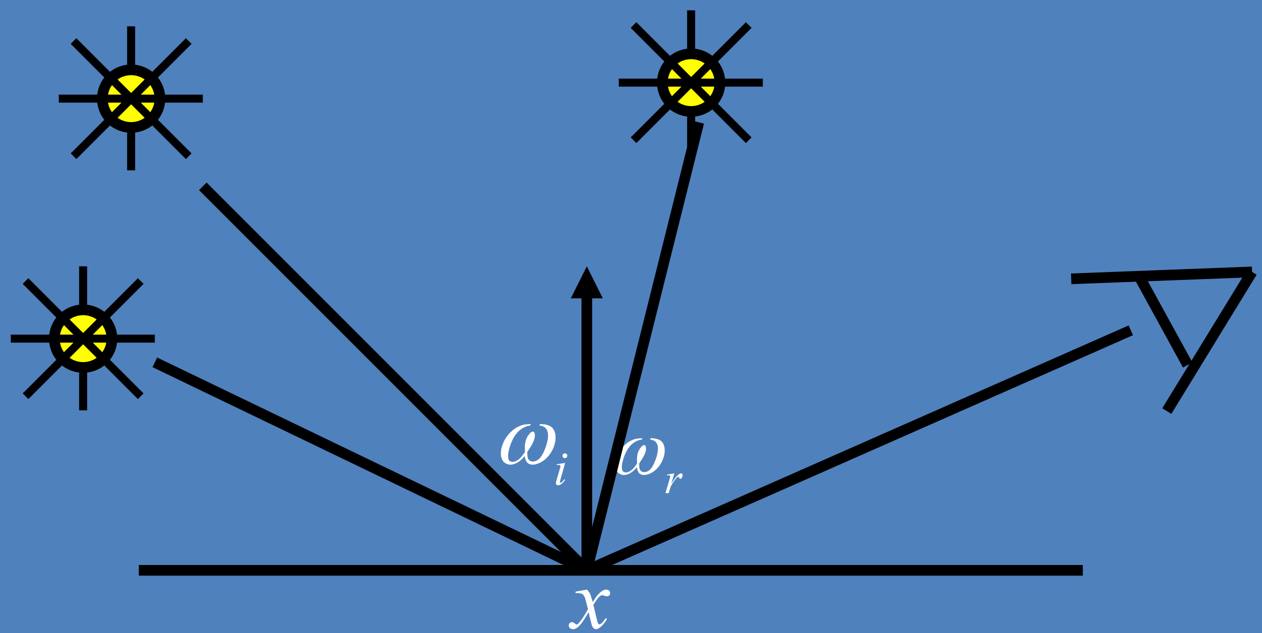


$$L_r(x,\omega_r) = L_e(x,\omega_r) + L_i(x,\omega_i)\, f(x,\omega_i,\omega_r)(\omega_i \bullet n)$$

Reflected Light (Output Image)	Emission	Incident Light (from light source)	BRDF	Cosine of Incident angle

[Slides with help from Pat Hanrahan and Henrik Jensen]

Reflectance Equation

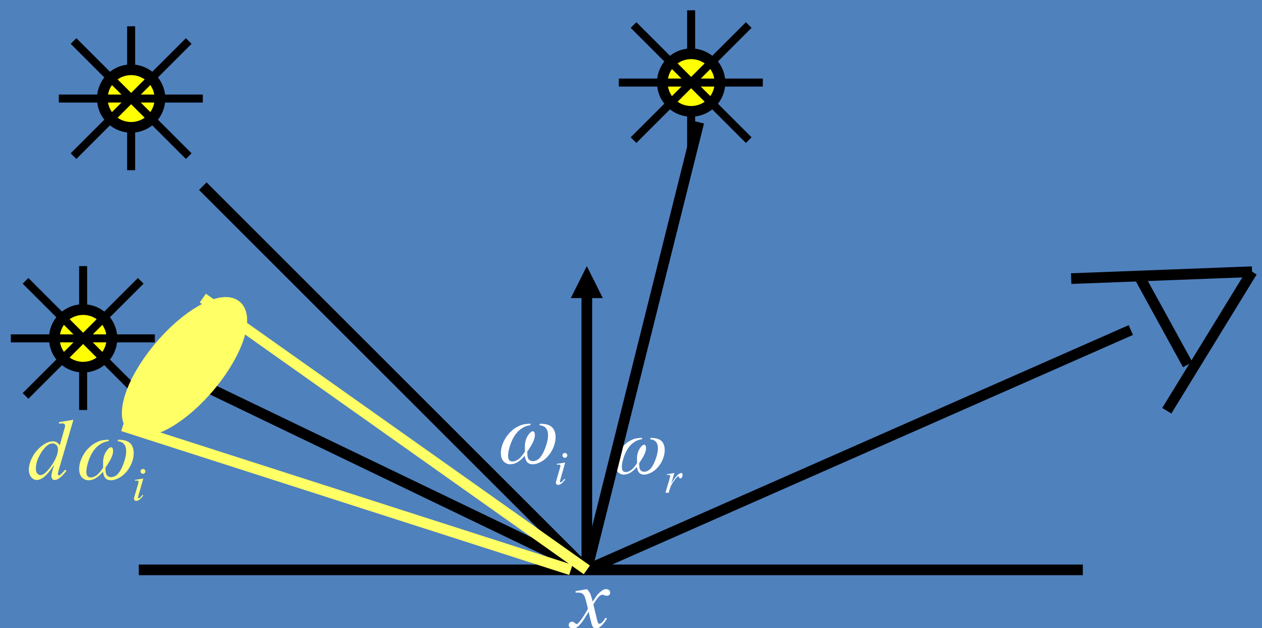


Sum over all light sources

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \sum L_i(x,\omega_i)\, f(x,\omega_i,\omega_r)(\omega_i \bullet n)$$

Reflected Light (Output Image) | Emission | Incident Light (from light source) | BRDF | Cosine of Incident angle

Reflectance Equation



Replace sum with integral

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i)\, f(x,\omega_i,\omega_r)\cos\theta_i \, d\omega_i$$

Reflected Light (Output Image)

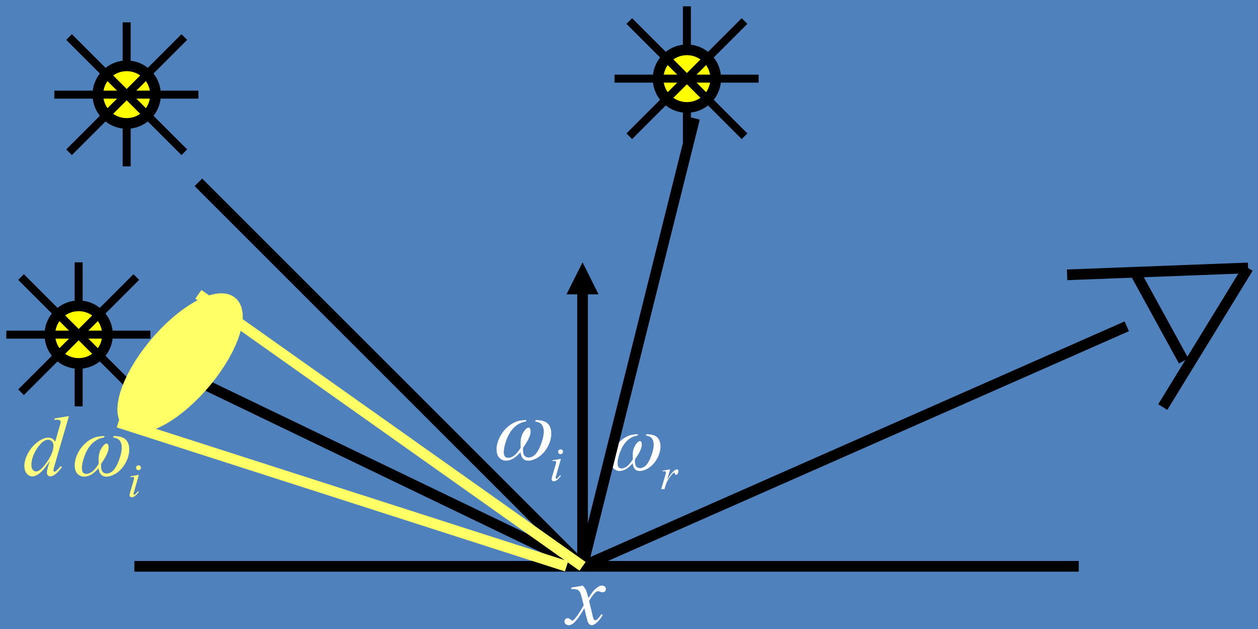
Emission

Incident Light (from light source)

BRDF

Cosine of Incident angle

Reflectance Equation



$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i)\, f(x,\omega_i,\omega_r)\cos\theta_i\, d\omega_i$$

The Challenge

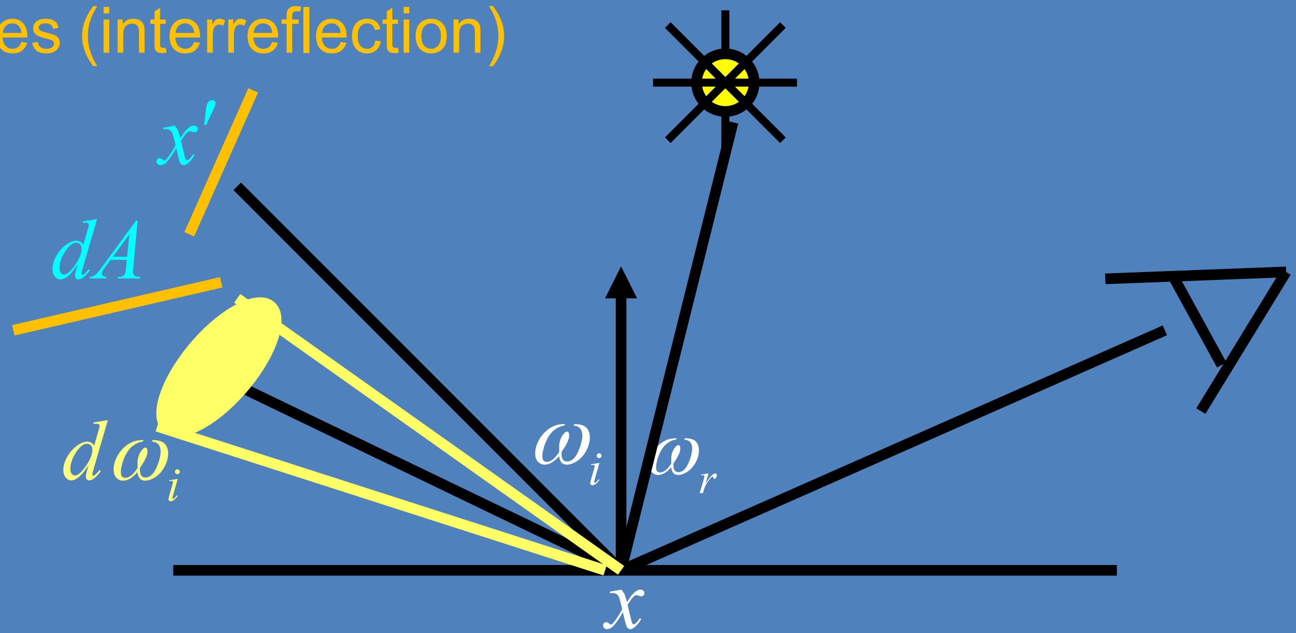
$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_\Omega L_i(x,\omega_i) f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

- Computing reflectance equation requires knowing the incoming radiance from surfaces

- …But determining incoming radiance requires knowing the reflected radiance from surfaces

Global Illumination

Surfaces (interreflection)



$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_r(x',-\omega_i)\, f(x,\omega_i,\omega_r)\cos\theta_i\, d\omega_i$$

Reflected Light (Output Image)	Emission	Reflected Light (from prev surface)	BRDF	Cosine of Incident angle

Rendering Equation (Kajiya 1986)

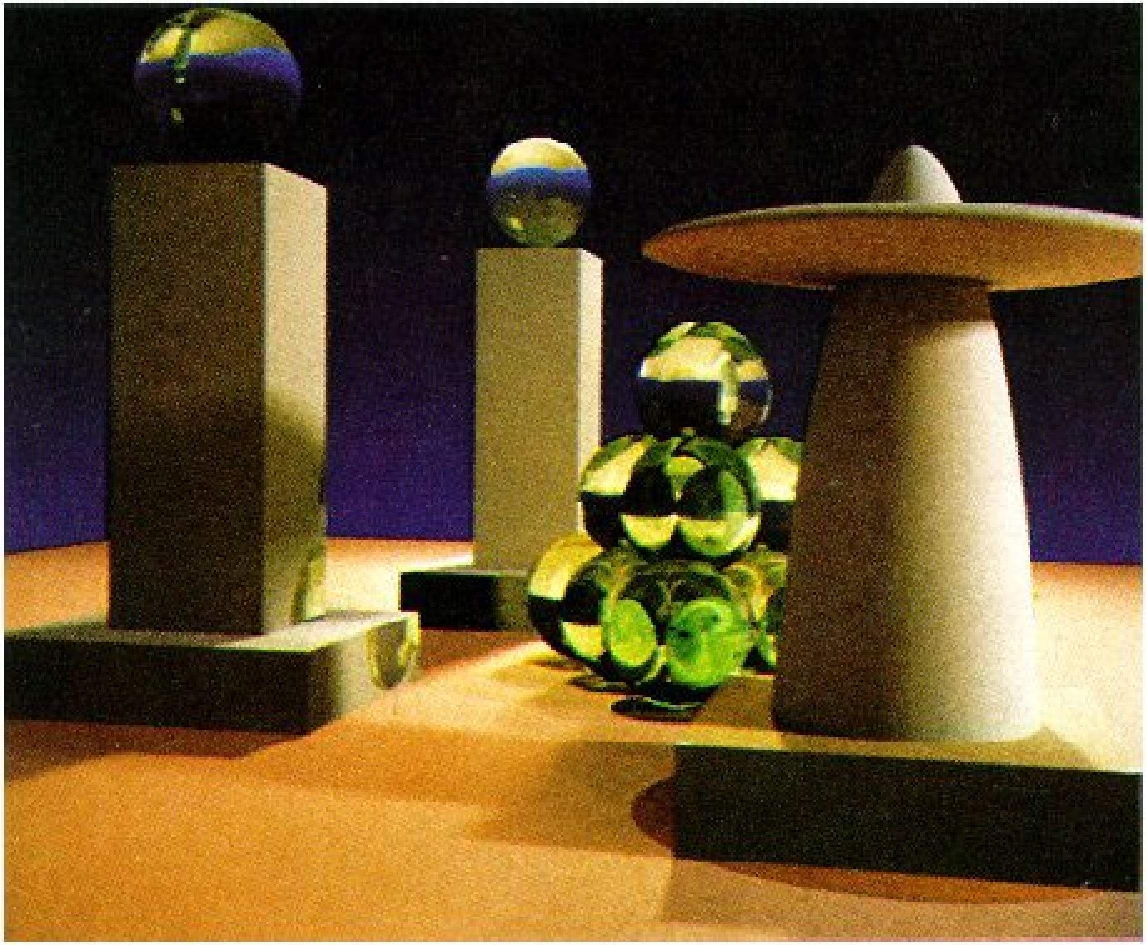
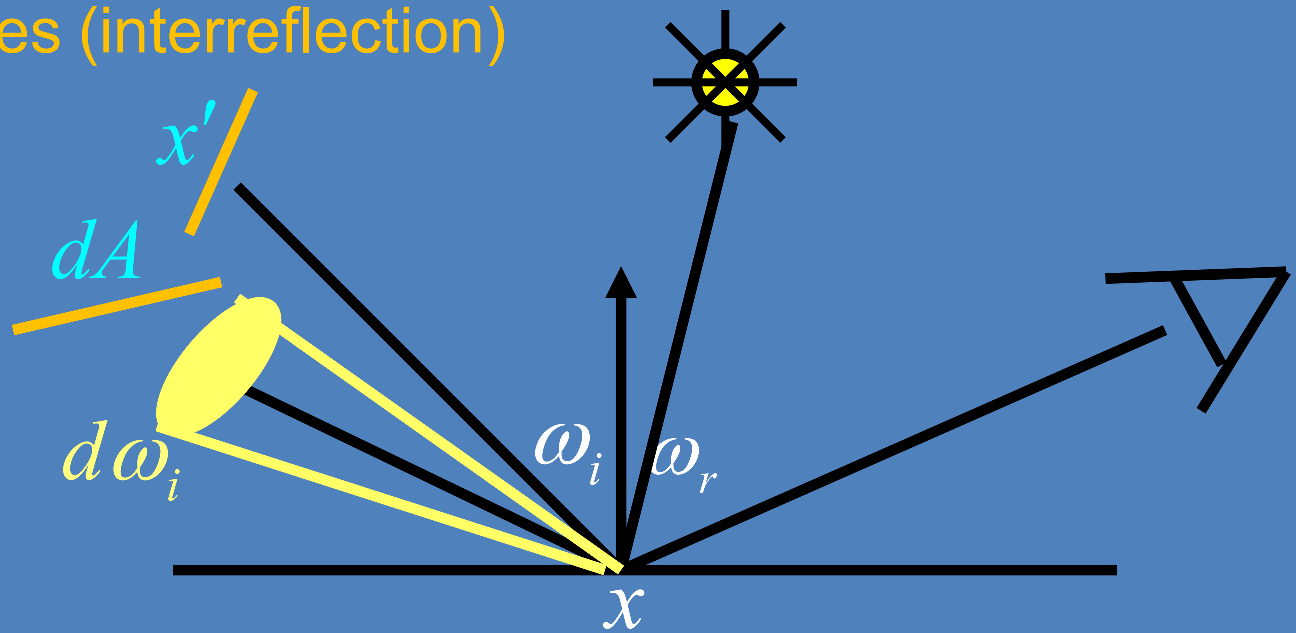


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

Rendering Equation

Surfaces (interreflection)



$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_\Omega L_r(x',-\omega_i)\, f(x,\omega_i,\omega_r)\cos\theta_i\, d\omega_i$$

Reflected Light (Output Image)	Emission	Reflected Light	BRDF	Cosine of Incident angle
UNKNOWN	KNOWN	UNKNOWN	KNOWN	KNOWN

Rendering Equation

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_\Omega L_r(x',-\omega_i)\, f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

Reflected Light (Output Image)	Emission	Reflected Light	BRDF	Cosine of Incident angle
UNKNOWN	KNOWN	UNKNOWN	KNOWN	KNOWN

After applying to simple math and simplifications, it turns we can approximately express the above as

$$L = E + KL$$

L, E are vectors,

K is the light transport matrix

Rendering as a Linear Operator…

$$L = E + KE + K^2E + K^3E + \ldots$$

Emission directly
From light sources

Direct Illumination
on surfaces

Global Illumination
(One bounce indirect)
[Mirrors, Refraction]

(Two bounce indirect)
[Caustics, etc…]

Ray Tracing

$$L = E + KE + K^2E + K^3E + \ldots$$

Emission directly
From light sources

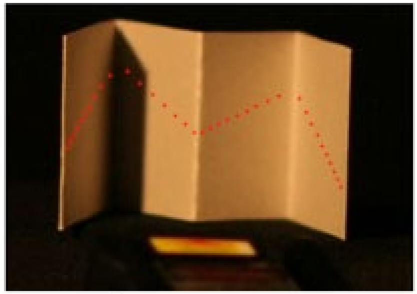
Direct Illumination
on surfaces

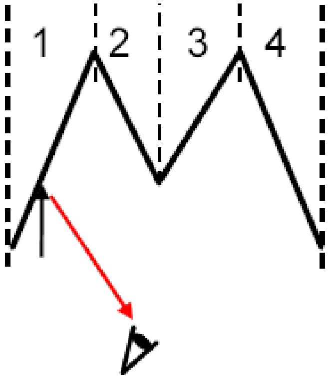
Global Illumination
(One bounce indirect)
[Mirrors, Refraction]

(Two bounce indirect)
[Caustics, etc…]

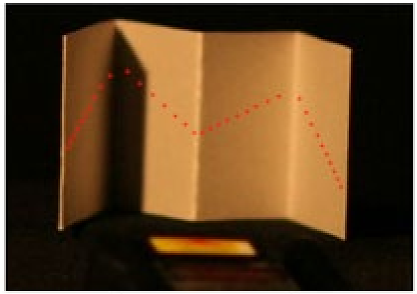
OpenGL
Shading

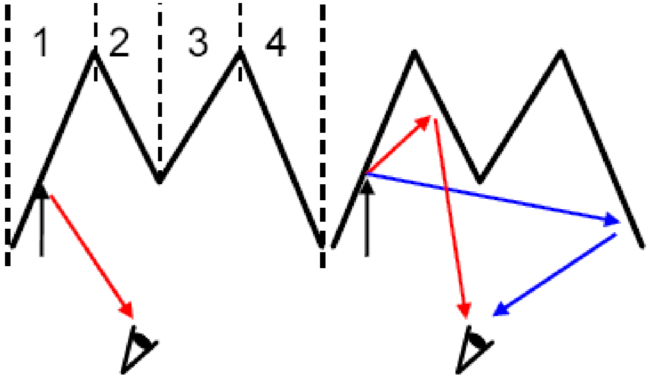
Example



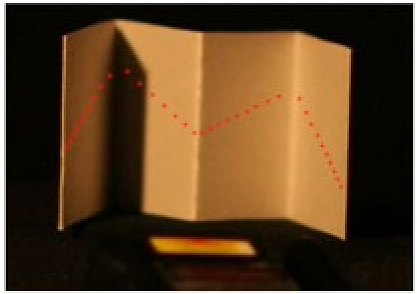


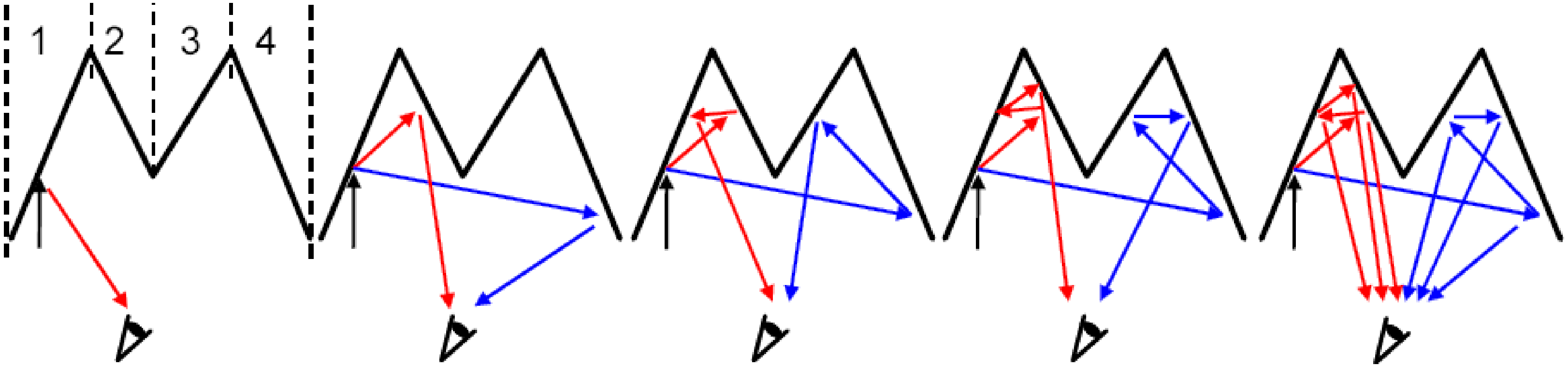
Example





Example





Example

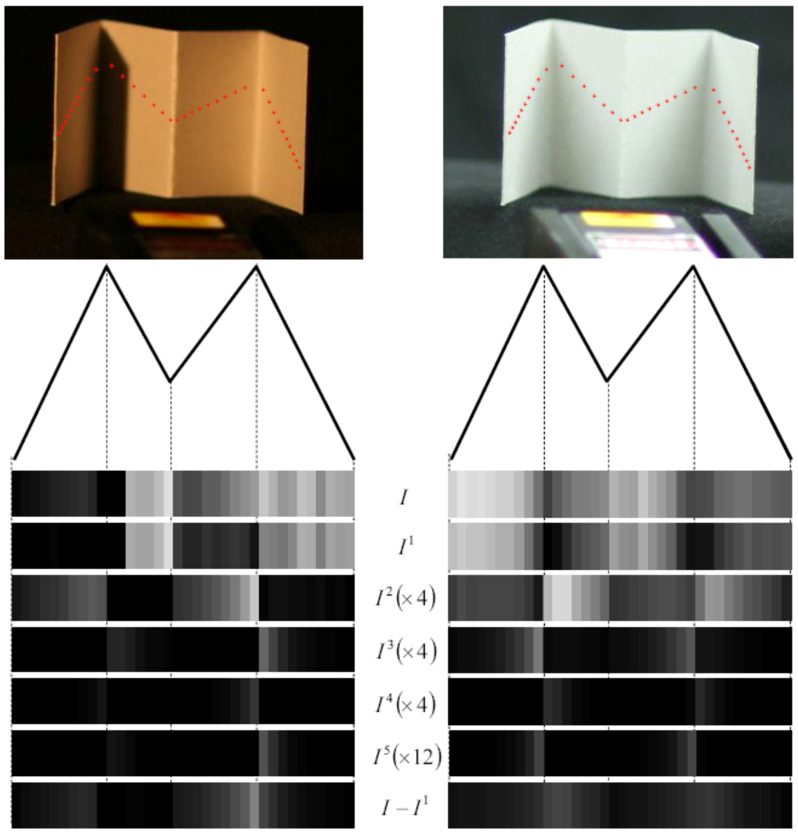


Figure 6: Inverse light transport applied to images I captured under unknown illumination conditions. I is decomposed into direct illumination I^1 and subsequent n-bounce images I^n, as shown. Observe that the interreflections have the effect of increasing brightness in concave (but not convex) junctions of the "M". Image intensities are scaled linearly, as indicated.

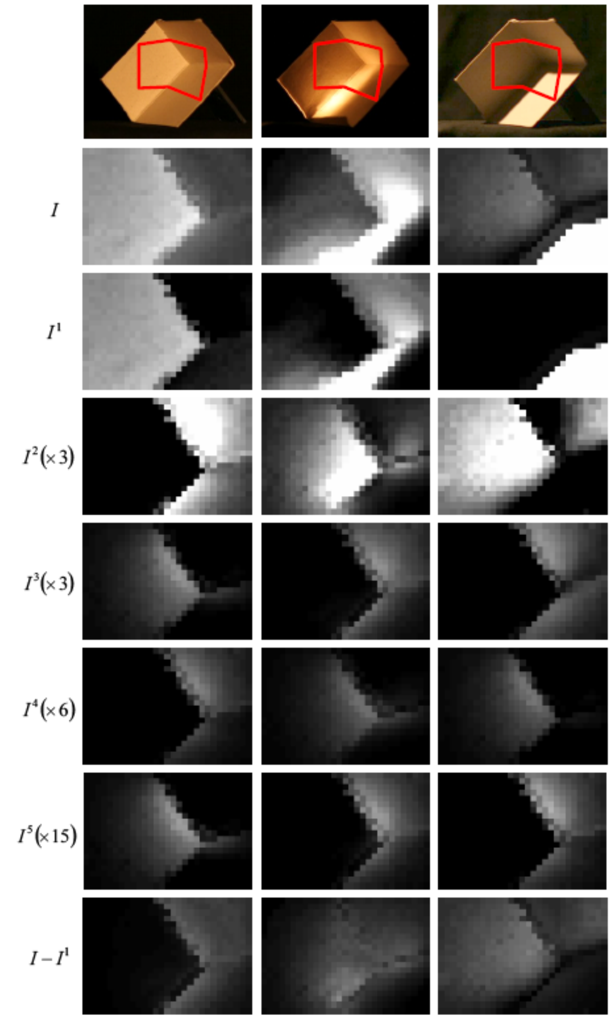


Figure 9: *Inverse light transport applied to images captured under unknown illumination conditions: input images I are decomposed into direct illumination I^1, 2- to 5-bounce images I^2–I^5, and indirect illuminations $I - I^1$.*

Rendering Equation and Global Illumination Topics

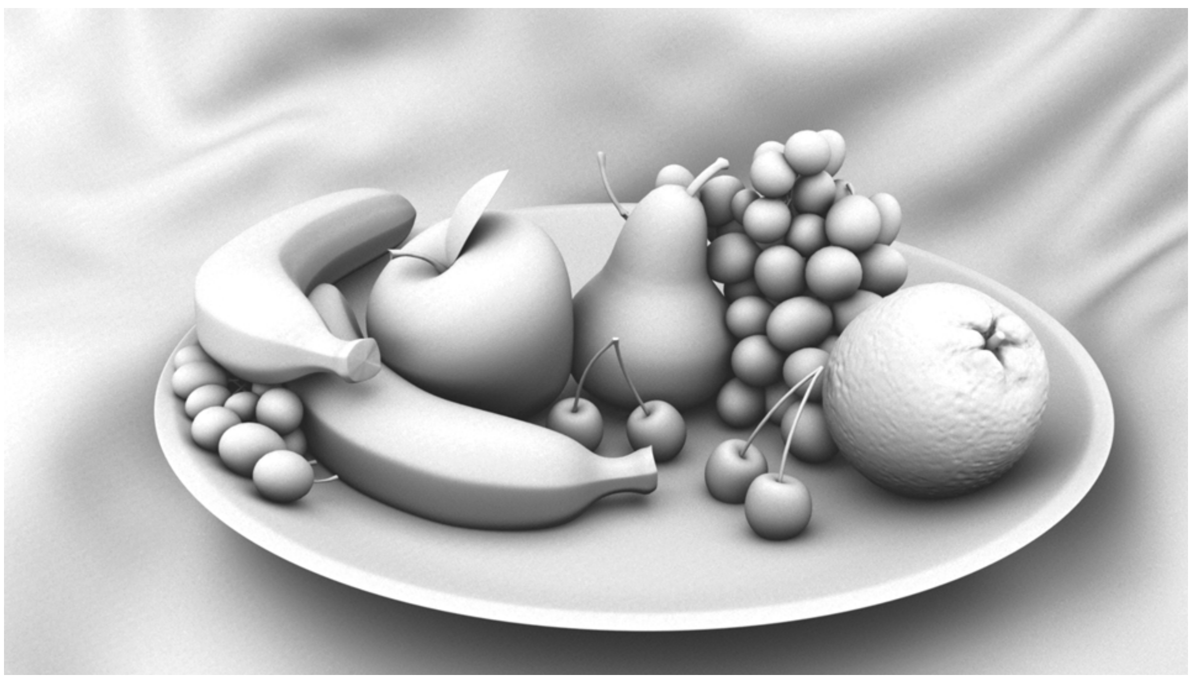
- Local-approximations to Global Illumination
  - Diffuse/Specular
  - Ambient Occlusion
- Global Illumination Algorithms
  - Ray tracing
  - Path tracing
  - Radiosity
- Bidirectional Reflectance Distribution Functions (BRDF)

Rendering Equation and Global Illumination Topics

- Local-approximations to Global Illumination
  - Diffuse/Specular
  - **Ambient Occlusion**
- Global Illumination Algorithms
  - Ray tracing
  - Path tracing
  - Radiosity
- Bidirectional Reflectance Distribution Functions (BRDF)

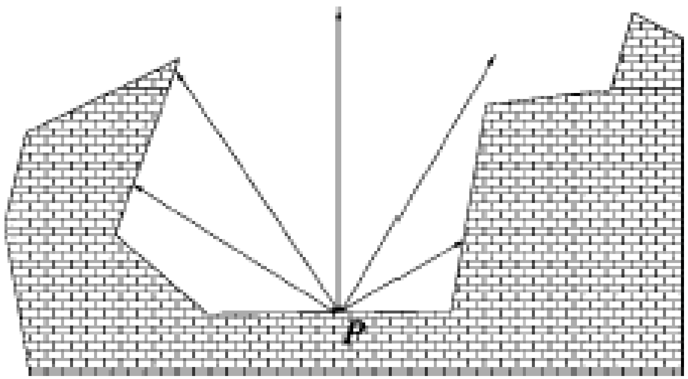
Ambient Occlusion

- It is a lighting technique to increase the realism of a 3D scene by a “cheap” imitation of global illumination



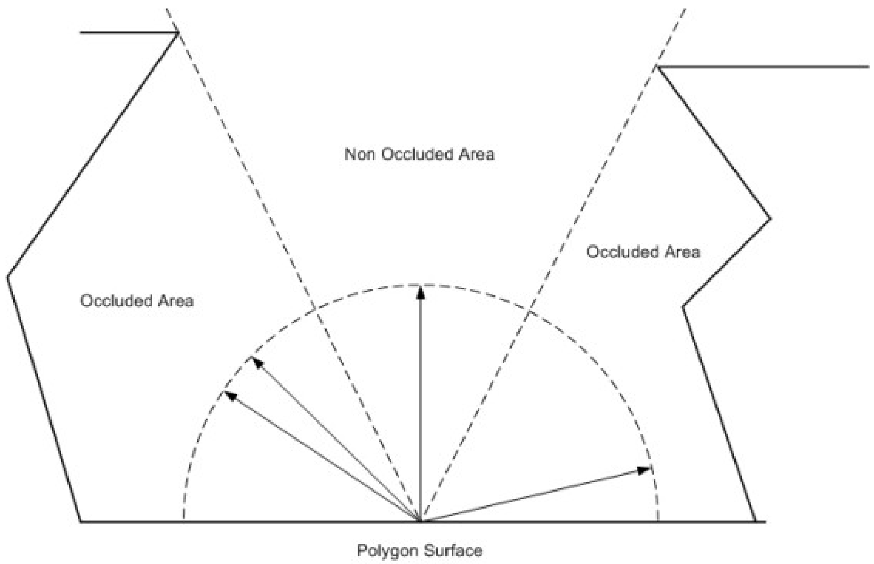
History

- In 1998, Zhukov introduced *obscurances* in the paper "An Ambient Light IlluminationModel."
- The effect of obscurances : we just need to evaluate the *hiddenness* or occlusion of the point by considering the objects around it.



Occlusion Factor/Map

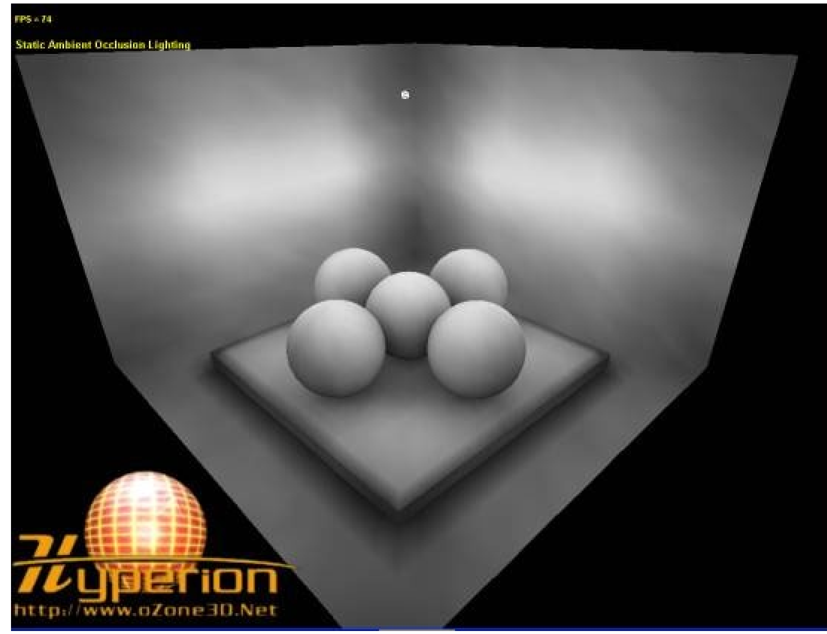
- Shooting rays outwards
- Determine the occlusion factor at p as a percentage; e.g., $occ(p) \in [0,1]$



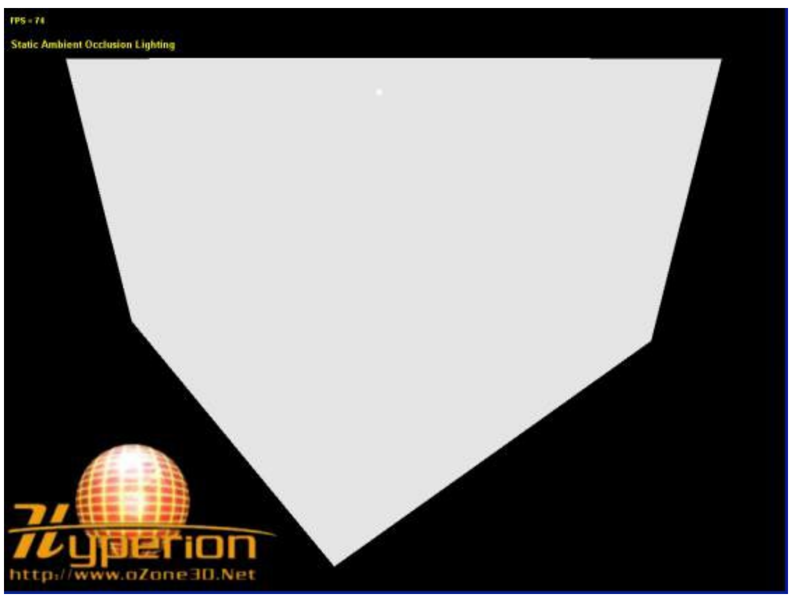
Ambient Occlusion in a Phong Illumination Model

$$I = I_a + I_d + I_s$$

$$I_a = IA \cdot occ(p)$$



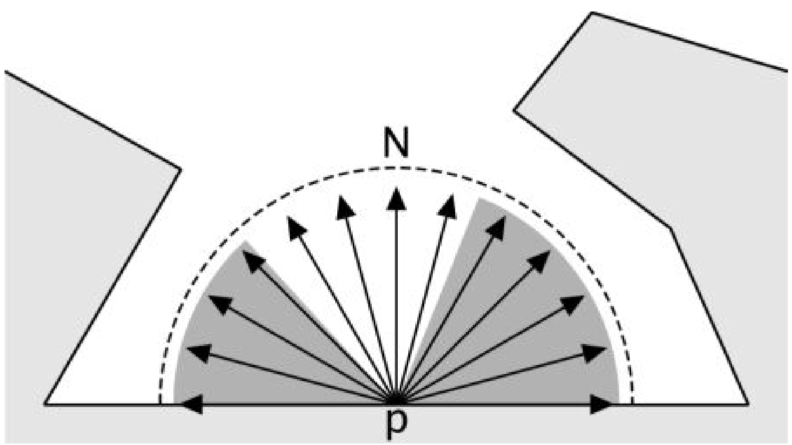
Modulate the intensity by an occlusion factor



Constant ambient intensity rendering

Inside-Looking-Out Approach: Ray Casting

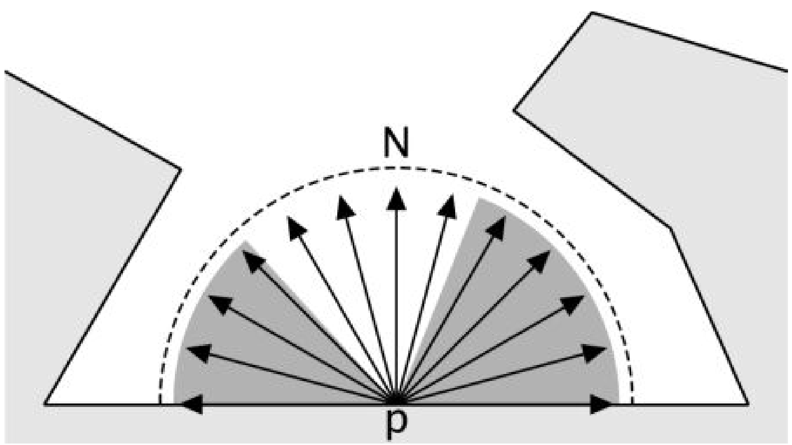
- Cast rays from ***p*** in uniform pattern across the hemisphere.
- Each surface point is shaded by a ratio of ray intersections to number of original samples.
- Subtracting this ratio from 1 gives us dark areas in the occluded portions of the surface.



e.g.: Cast 13 rays
9 intersections, so
occ(p)= ?

Inside-Looking-Out Approach: Ray Casting

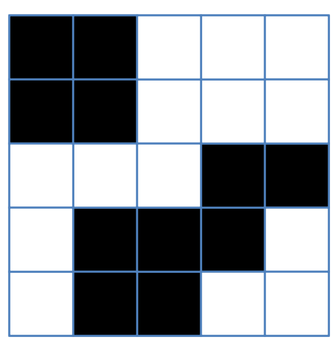
- Cast rays from ***p*** in uniform pattern across the hemisphere.
- Each surface point is shaded by a ratio of ray intersections to number of original samples.
- Subtracting this ratio from 1 gives us dark areas in the occluded portions of the surface.



e.g.: Cast 13 rays
9 intersections, so
occ(p)=4/13;
⇨ Color * 4/13

Inside-Looking-Out Approach: Hardware Rendering

- Render the view at low-res from ***p*** toward normal ***N***
- Rasterize black geometry against a white background
- Take the (cosine-weighted) average of rasterized fragments.



11 black fragments
⇨ Color * 14/25

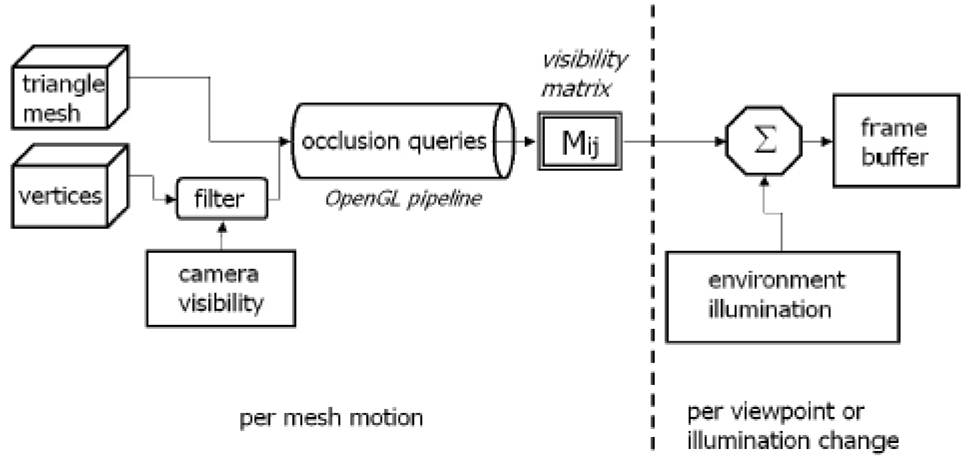
Comments

- Potentially huge pre-computation time per scene
- Stores occlusion factor as vertex attributes
  - Thus needs a dense sampling of vertices
- Variations on sampling method
  - "Inside-out" algorithm
  - "outside-in" alternative (not explained)

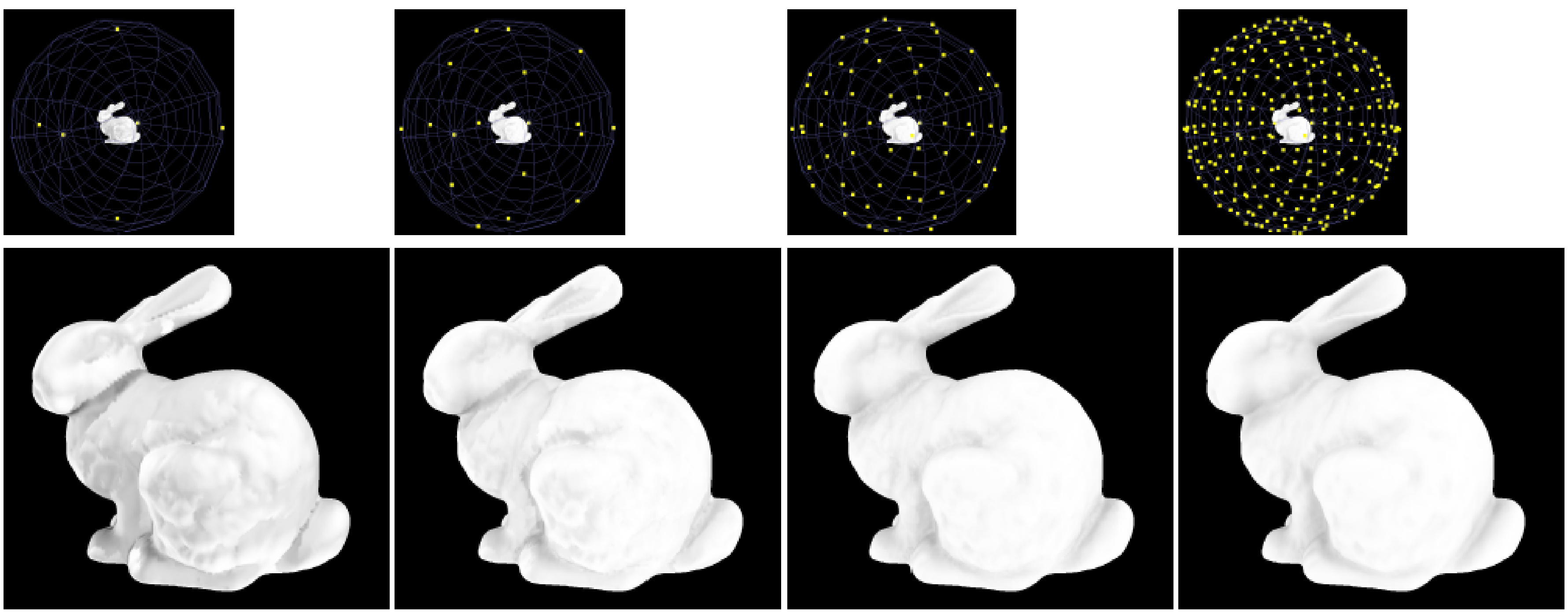
Outside-Looking-In Approach

- What would you do?

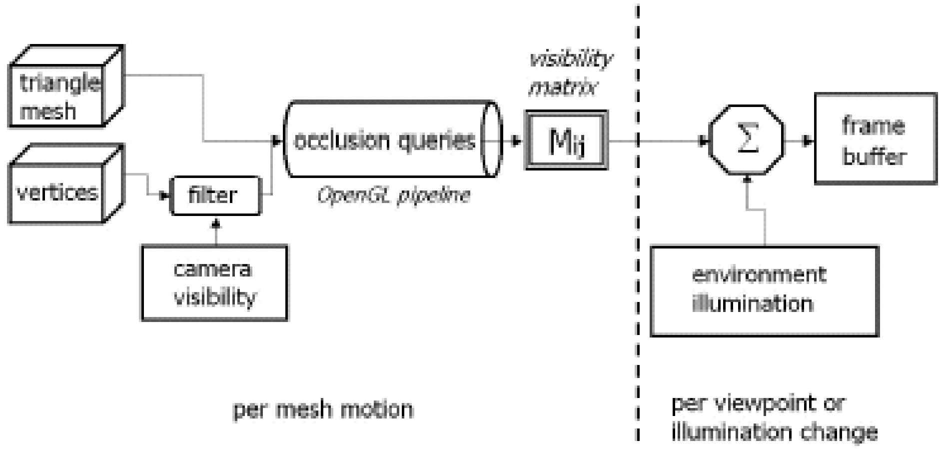
Outside-Looking-In: One option is [Sattler et. al 2004]



$$c_i = \sum_{j=1}^{k} M_{ij} I_j$$



enable orthographic projection
disable framebuffer
for all light directions j **do**
set camera at light direction $\mathbf{l}_j$
render object into depth buffer with polygon offset
for all vertices i **do**
begin query i
render vertex i
end query i
end for
for all vertices i **do**
retrieve result from query i
if result is "visible" **then**
$M_{ij} = \mathbf{n}_i \cdot \mathbf{l}_j$
end if
end for
end for



$$M_{ij} = \begin{cases} \mathbf{n}_i \cdot \mathbf{l}_j & : \text{ vertex visible} \\ 0 & : \text{ vertex invisible} \end{cases}$$

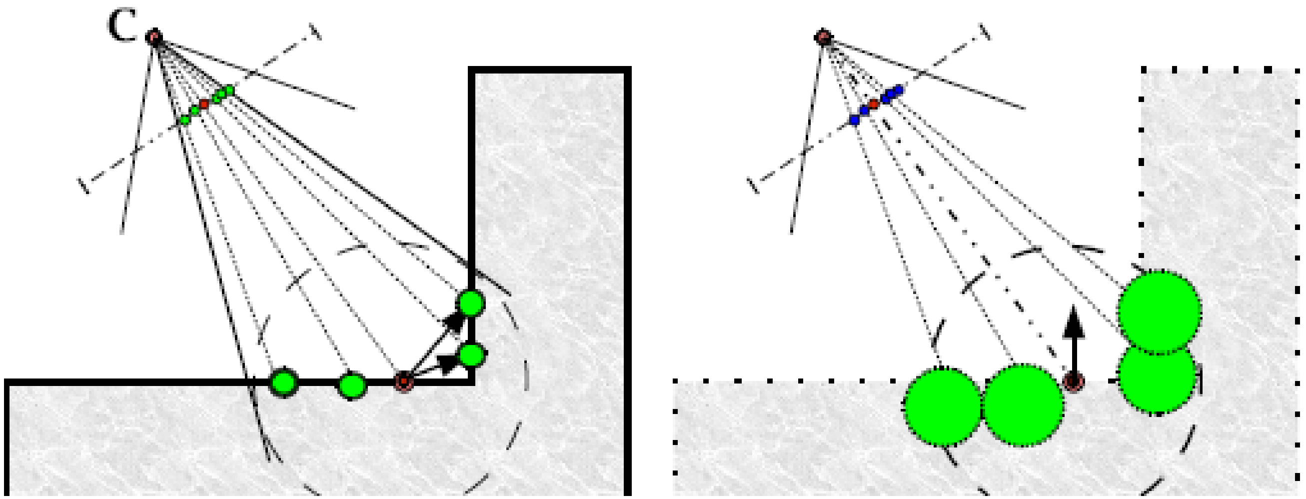
$$c_i = \sum_{j=1}^{k} M_{ij} I_j$$

[Sattler et al. 2004]

- For each light on the light sphere
- Take the depth map (for occlusion query)
- Use occlusion query to determine the visibility matrix

Another option: Screen-Based AO

- SHANMUGAM, P., AND ARIKAN, O. 2007. Hardware Accelerated Ambient Occlusion Techniques on GPUs. In Proceedings of ACM Symposium in Interactive 3D Graphics and Games, ACM.



Screen-Based AO

Screen-Based AO

- What would you do?

Rendering Equation and Global Illumination Topics

- Local-approximations to Global Illumination
  - Diffuse/Specular
  - Ambient Occlusion
- Global Illumination Algorithms
  - Ray tracing
  - Path tracing
  - **Radiosity**
- Bidirectional Reflectance Distribution Functions (BRDF)

Radiosity

- Radiosity, inspired by ideas from heat transfer, is an application of a finite element method to solving the rendering equation for scenes with purely diffuse surfaces.
- The main idea of the method is **to store illumination values on the surfaces of the objects, as the light is propagated starting at the light sources.**

[Radiosity slides heavily based on Dr. Mario Costa Sousa, Dept. of of CS, U. Of Calgary]

Radiosity

- Calculating the overall light propagation within a scene, for short **global illumination** is a very difficult problem.
- With a standard ray tracing algorithm, this is a very time consuming task, since a huge number of rays have to be shot.

Radiosity (Computer Graphics)

- Assumption #1: surfaces are diffuse emitters and reflectors of energy, emitting and reflecting energy uniformly over their entire area.

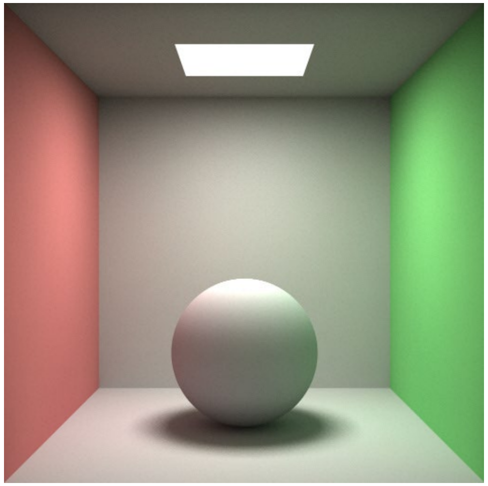
- Assumption #2: an equilibrium solution can be reached; that all of the energy in an environment is accounted for, through absorption and reflection.

- Also viewpoint independent: the solution will be the same regardless of the viewpoint of the image.

Radiosity

- Equation:

$$B_i = E_i + \rho_i \sum B_j F_{ij}$$




Ray Tracing


Radiosity

The Radiosity Equation

$$B_i = E_i + \rho_i \sum B_j F_{ij}$$

Radiosity of surface i

Emissivity of surface i

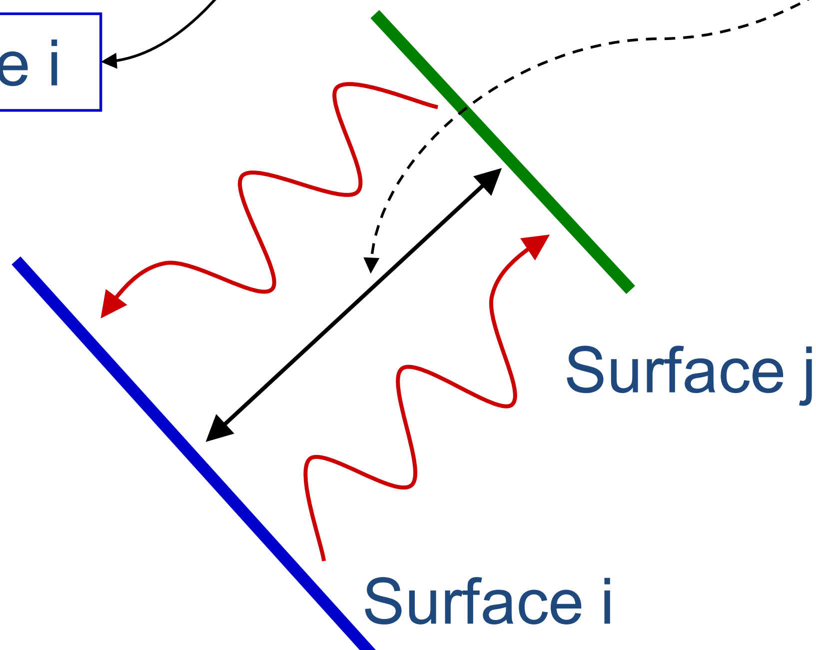
Reflectivity of surface i

will <u>absorb</u> a certain percentage of light energy which strikes the surface

Form Factor of surface j relative to surface i

accounts for the <u>physical relationship</u> between the two surfaces

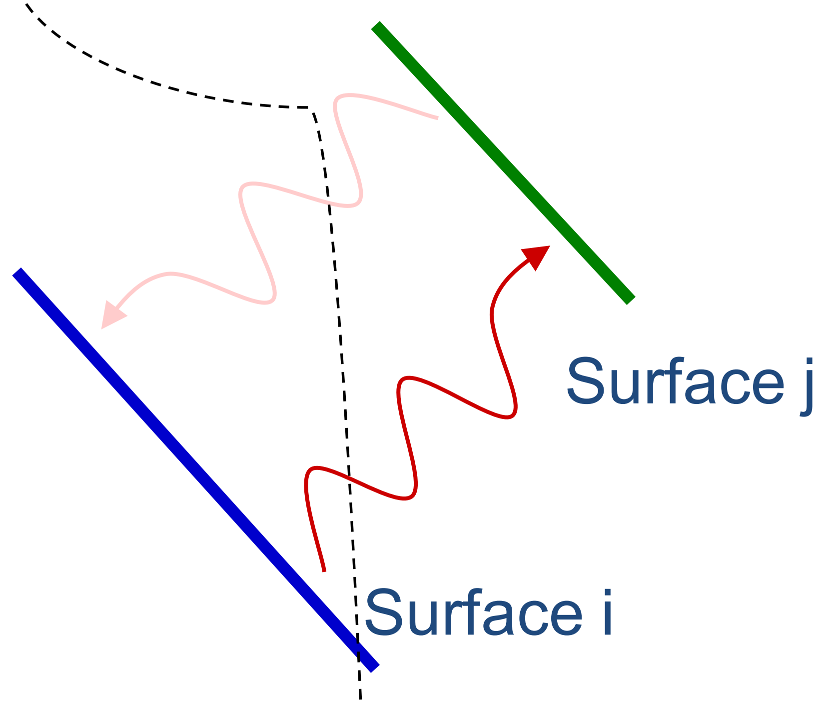
Radiosity of surface j



The Radiosity Equation

$$B_i = \boxed{E_i} + \rho_i \sum B_j F_{ij}$$

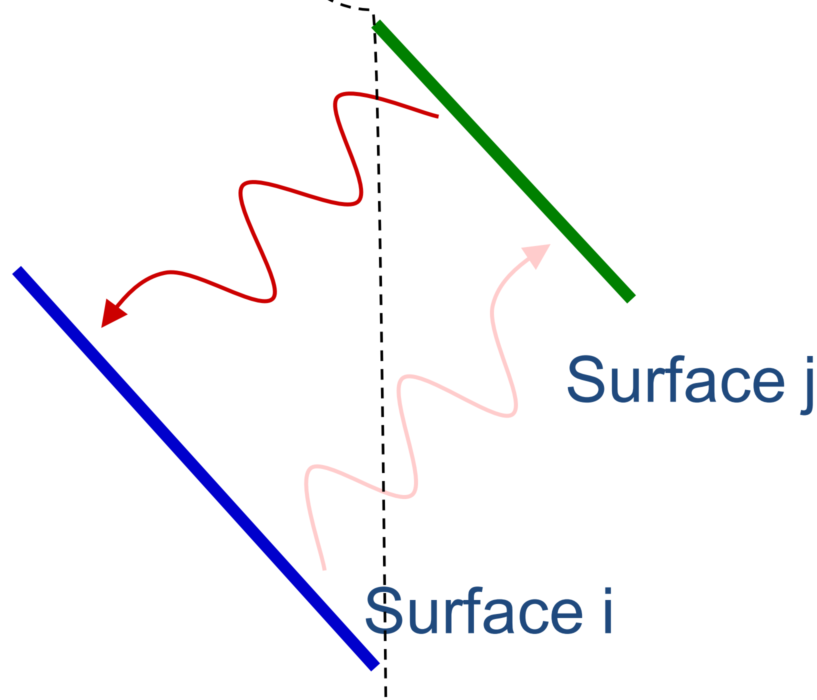
Energy emitted by surface i



The Radiosity Equation

$$B_i = E_i + \rho_i \boxed{\sum B_j F_{ij}}$$

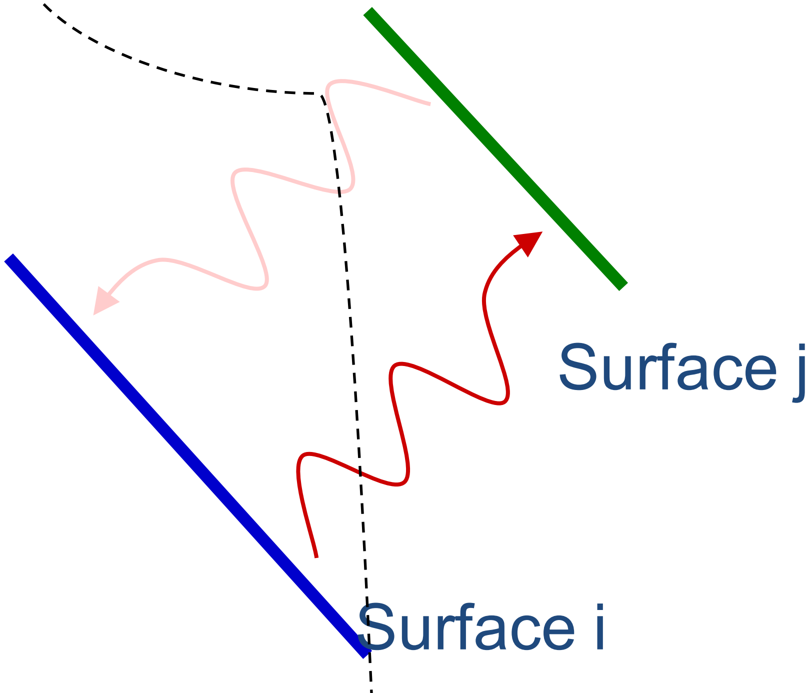
Energy reaching surface i from other surfaces



The Radiosity Equation

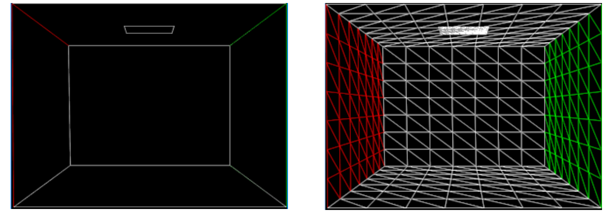
$$B_i = E_i + \boxed{\rho_i \sum B_j F_{ij}}$$

Energy reflected by surface i

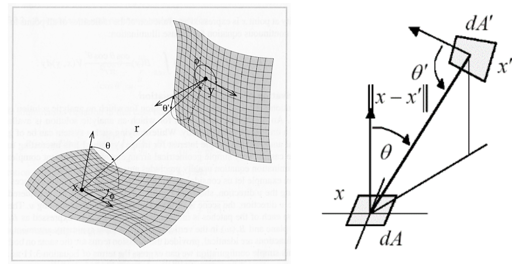


Classic Radiosity Algorithm

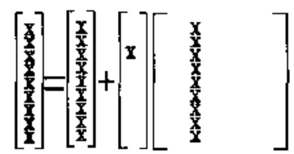
Mesh Surfaces into Elements



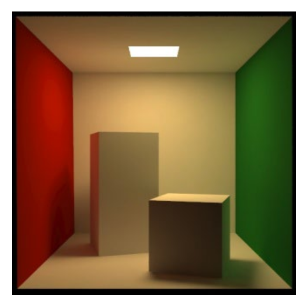
Compute Form Factors Between Elements



Solve for Radiosities



Reconstruct and Display Solution



Solving for radiosity solution

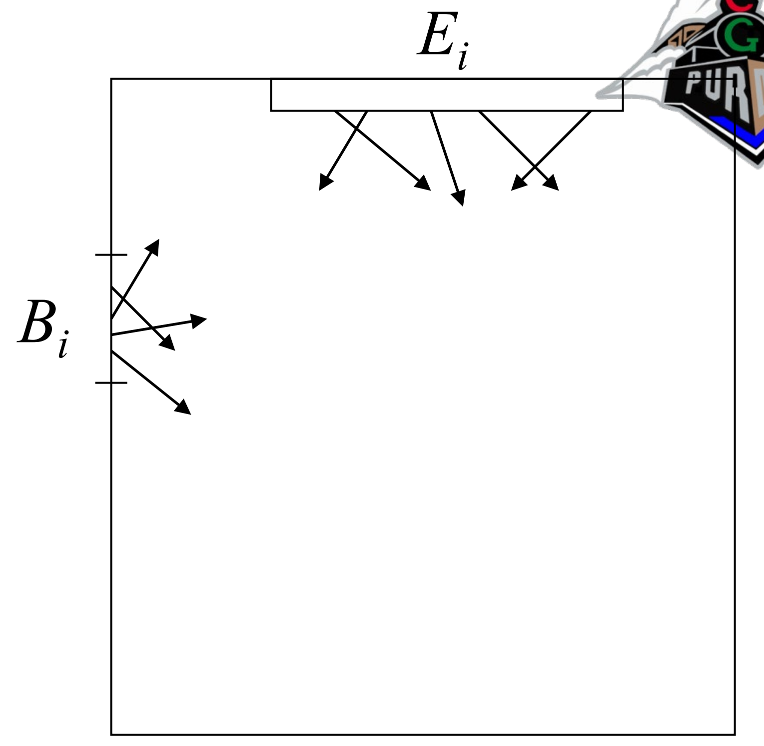
- $\boxed{\text{The "Full Matrix" Radiosity Algorithm}}$
- Gathering & Shooting

Radiosity Matrix

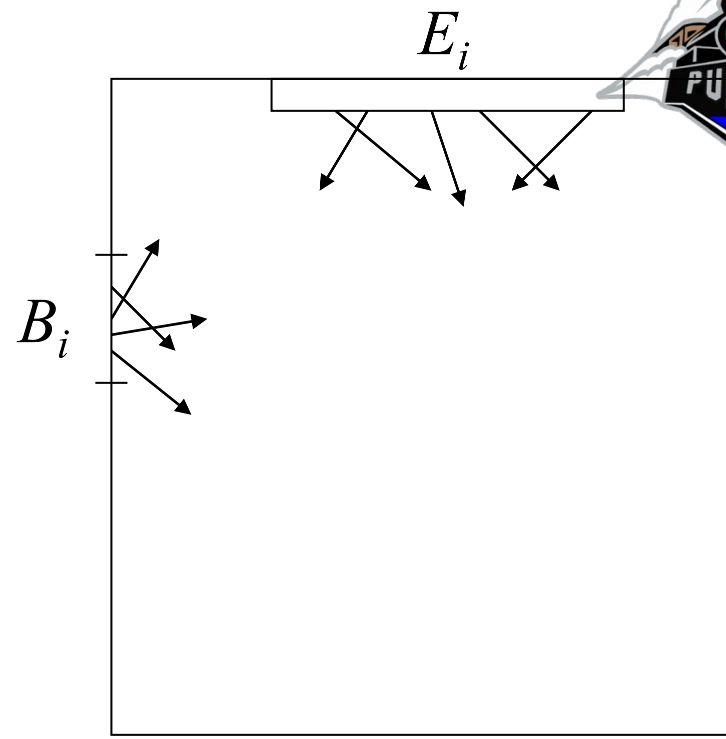
$$\boxed{B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j}$$

What is the matrix form?
(like "Ax=b")

$$B_i - \rho_i \sum_{j=1}^{n} F_{ij} B_j = E_i$$



Radiosity Matrix



$$\boxed{B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j}$$

$$B_i - \rho_i \sum_{j=1}^{n} F_{ij} B_j = E_i$$

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

Radiosity Matrix

- The "full matrix" radiosity solution calculates the form factors between each pair of surfaces in the environment, then forms a series of simultaneous linear equations.

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

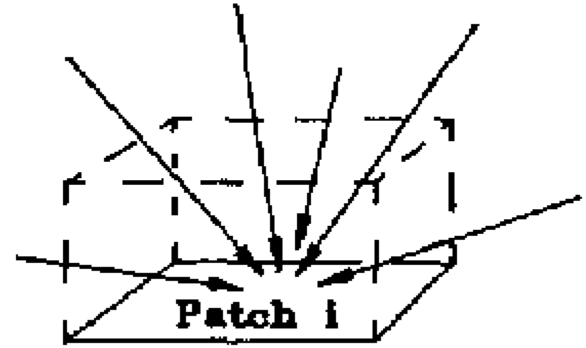
- This matrix equation is solved for the "B" values, which can be used as the final intensity (or color) value of each surface.

Solving for radiosity solution

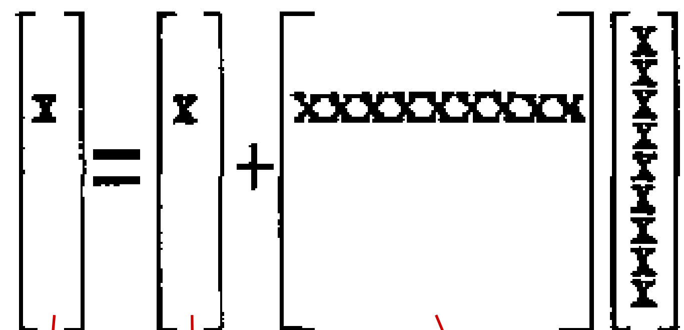
- The "Full Matrix" Radiosity Algorithm
- Gathering & Shooting

Gathering

- In a sense, the light leaving patch i is determined by *gathering* in the light from the rest of the environment



GATHERING



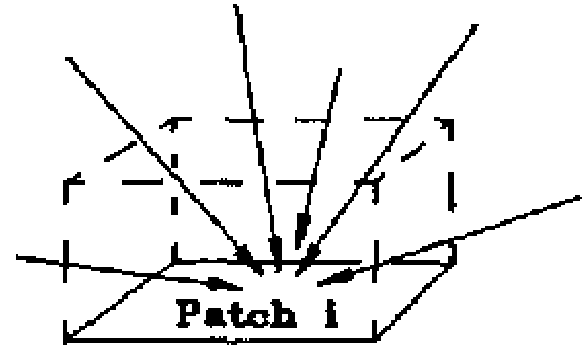
$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$$

$$B_i = E_i + \sum_{j=1}^{n} \left(\rho_i F_{ij}\right) B_j$$

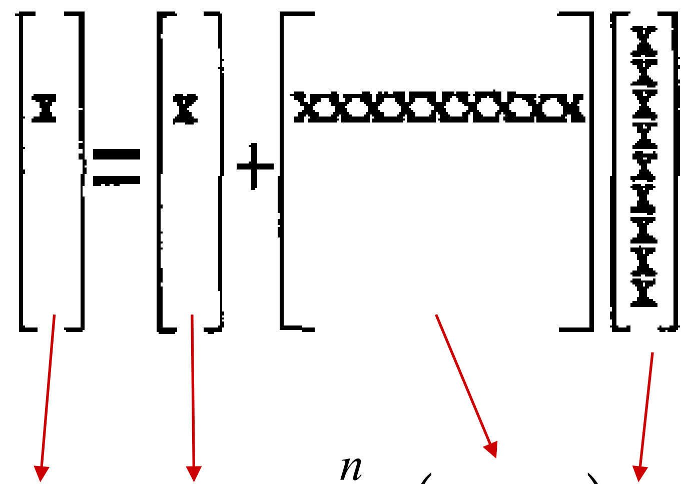
$$B_i \ due \ to \ B_j = \rho_i B_j F_{ij}$$

Gathering

- <u>Gathering light</u> through a hemi-cube allows <u>one patch</u> radiosity to be updated.

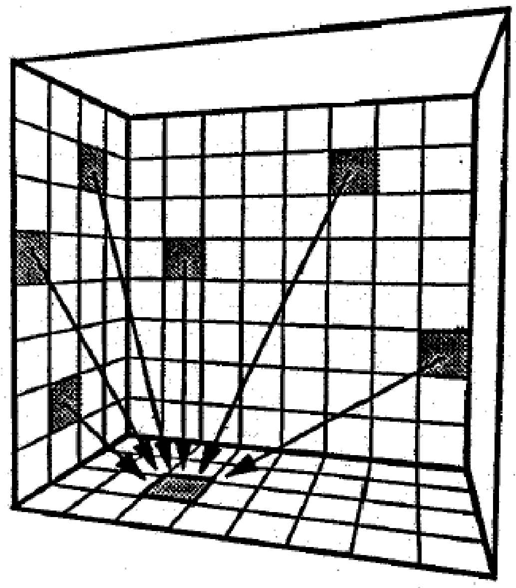


GATHERING



$$B_i = E_i + \sum_{j=1}^{n} \left(\rho_i F_{ij} \right) B_j$$

Gathering



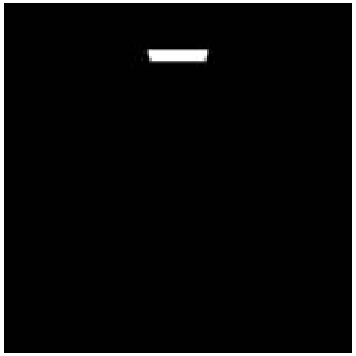
```
for(i=0; i<n; i++)
  B[i] = Be[i];

while( !converged ) {
  for(i=0; i<n; i++) {
    E[i] = 0;
    for(j=0; j<n; j++)
      E[i] += F[i][j]*B[j];
    B[i] = Be[i]+rho[i]*E[i];
  }
}
```

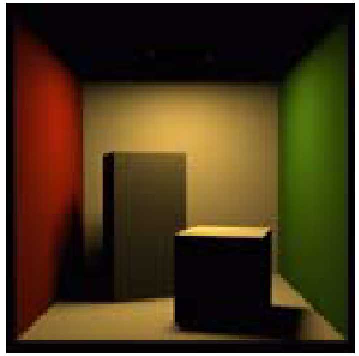
Row of F times B

Calculate one row of F and discard

Successive Approximation

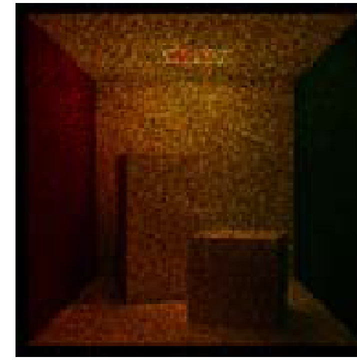


L_e

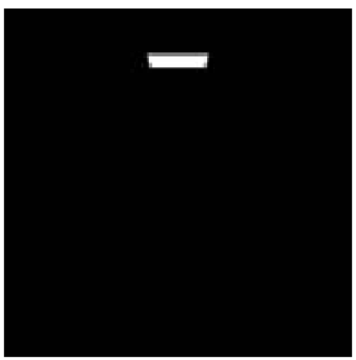


$K \circ L_e$

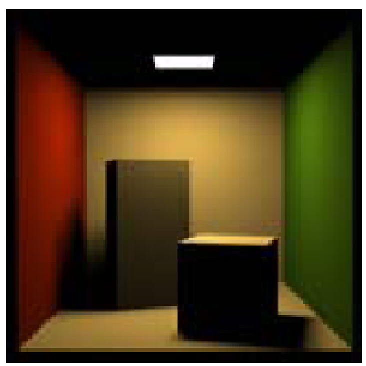
$K \circ K \circ L_e$



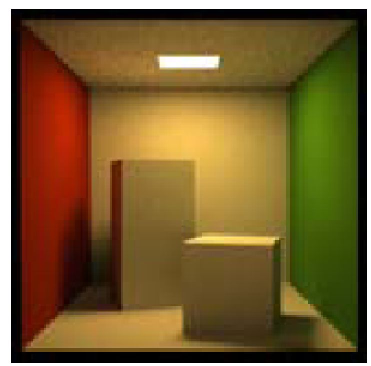
$K \circ K \circ K \circ L_e$



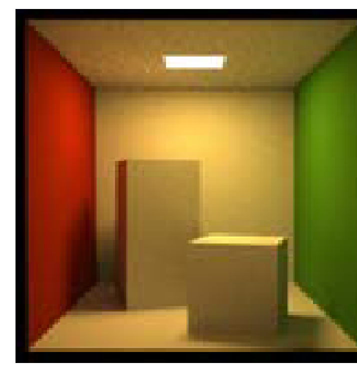
L_e



$L_e + K \circ L_e$



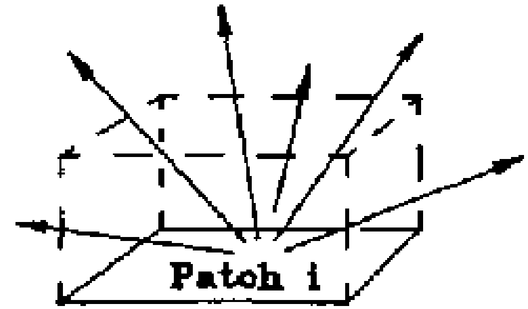
$L_e + \cdots K^2 \circ L_e$



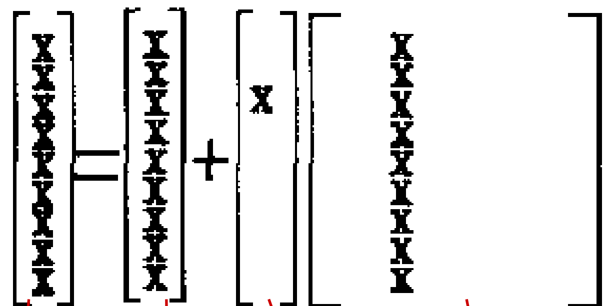
$L_e + \cdots K^3 \circ L_e$

Shooting

- <u>Shooting light</u> through a single hemi-cube allows <u>the whole environment's radiosity values</u> to be updated simultaneously.



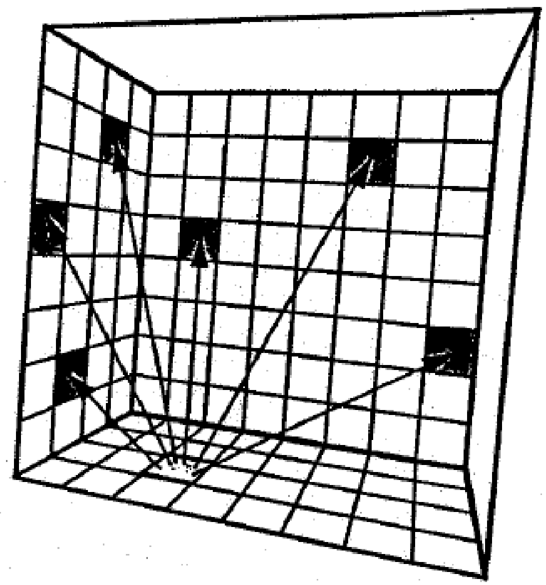
SHOOTING



For all j $\Longrightarrow$ $B_j = B_j + B_i\left(\rho_j E_{ji}\right)$

where $F_{ji} = \dfrac{F_{ij} A_i}{A_j}$

Shooting

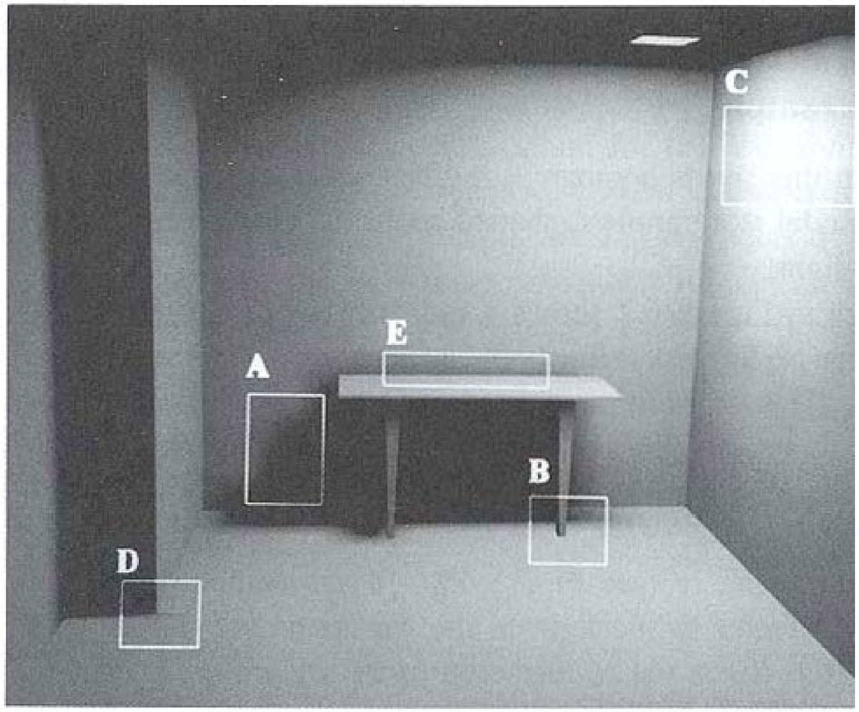


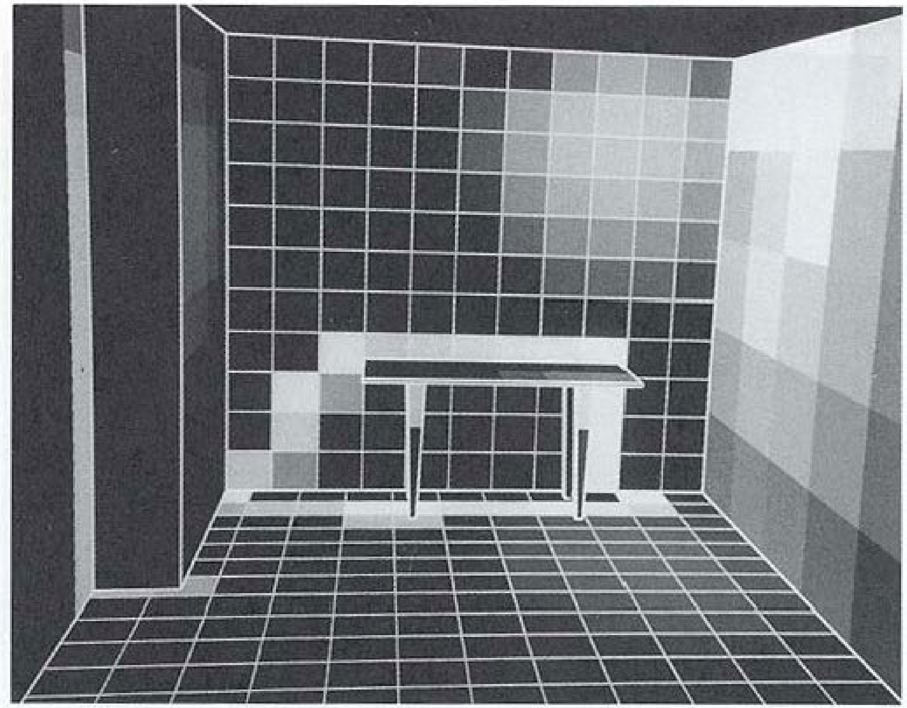
```
for(i=0; i<n; i++) {
  B[i] = dB[i] = Be[i];
  while( !converged ) {
    set i st dB[i] is the largest;
    for(j=0;j<n;j++)
      if(i!=j) {
        db =rho[j]*F[j][i]*dB[i];
        dB[j] += db;
        B[j] += db;
      }
    dB[i]=0;
  }
}
```

Brightness order

Column of F times B

Artifacts



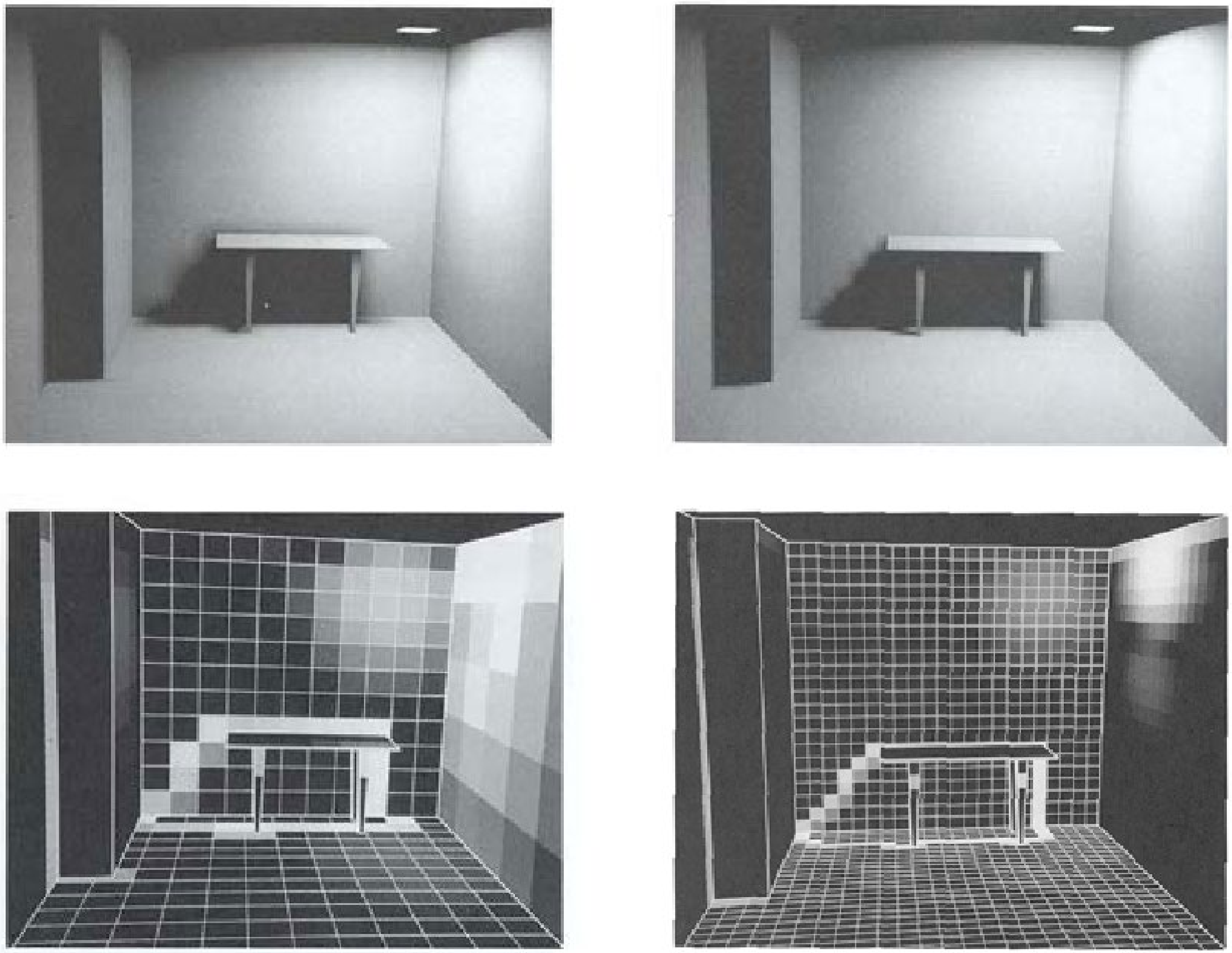


Error Image

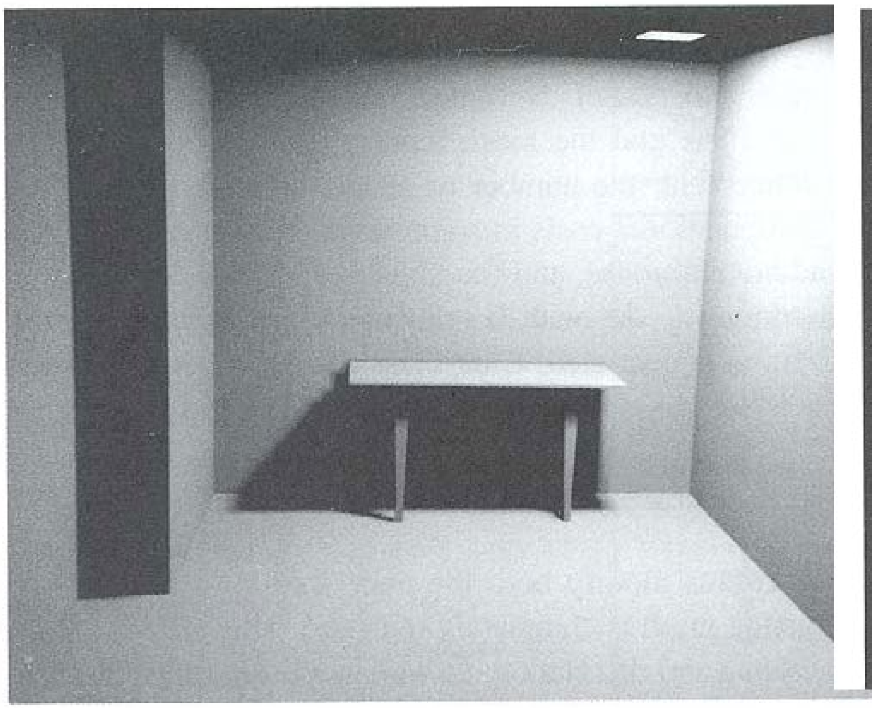
A. Blocky shadows
B. Missing features
C. Mach bands
D. Inappropriate shading discontinuities
E. Unresolved discontinuities

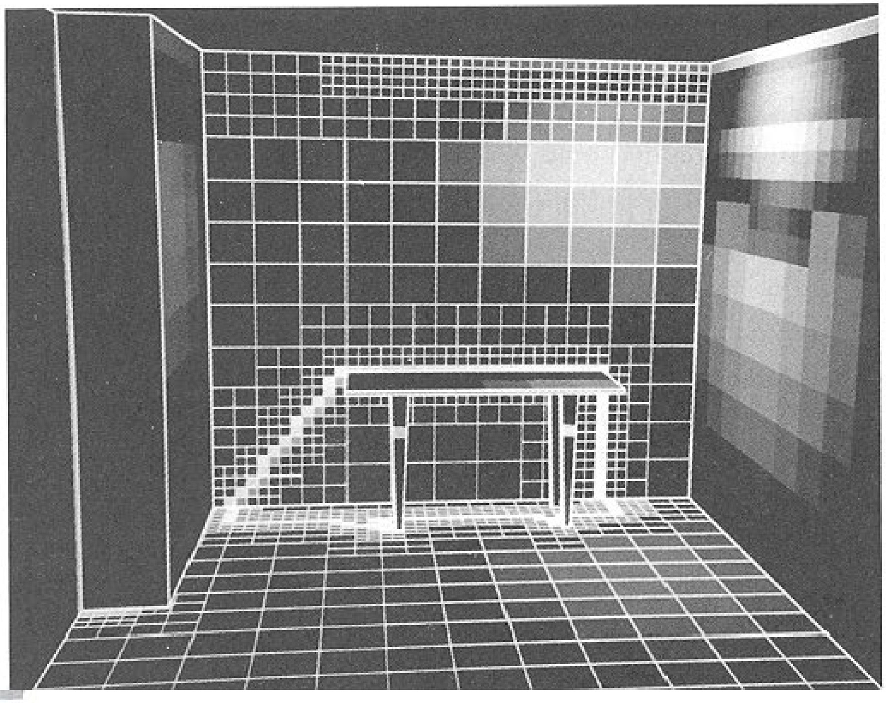
What can you do?

Increase Resolution



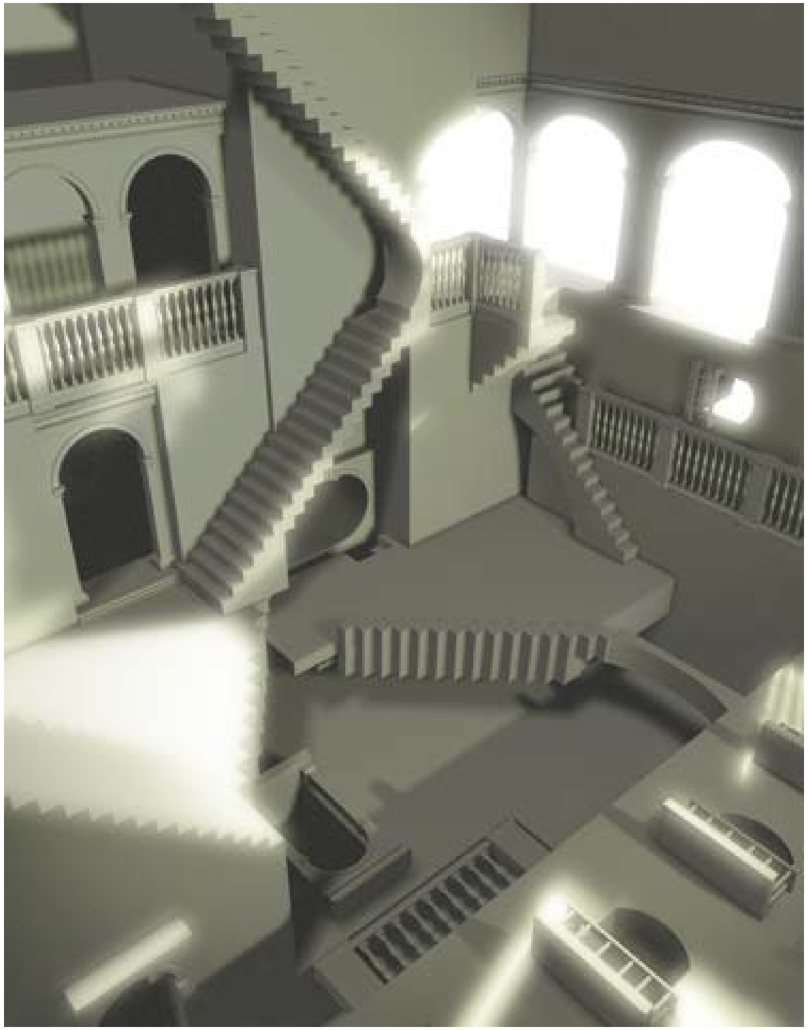
Adaptively Mesh





e.g., Discontinuity Meshing

More examples…



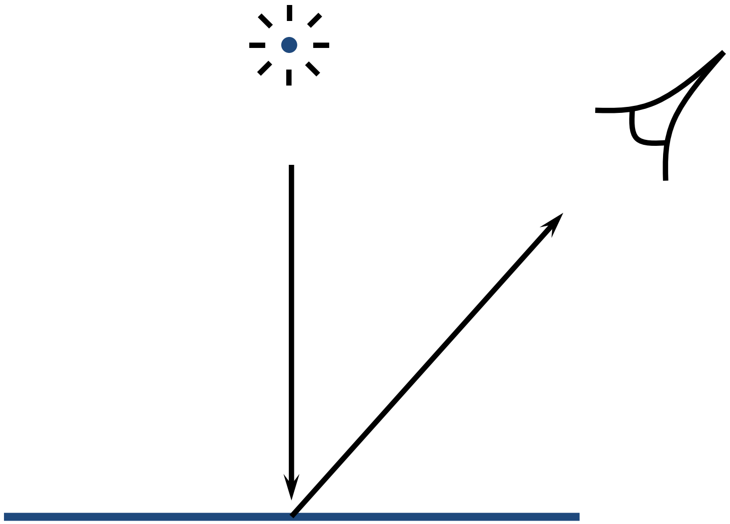
Rendering Equation and Global Illumination Topics

- Local-approximations to Global Illumination
  - Diffuse/Specular
  - Ambient Occlusion
- Global Illumination Algorithms
  - Ray tracing
  - Path tracing
  - Radiosity
- **Bidirectional Reflectance Distribution Functions (BRDF)**

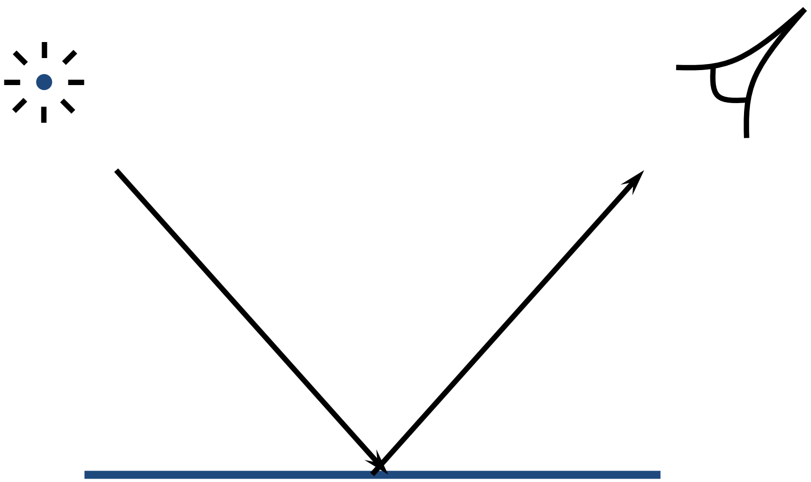
Measuring BRDFs

- BRDF is 4-dimensional, though simpler measurements (0D/1D/2D/3D) are often useful

Measuring Reflectance



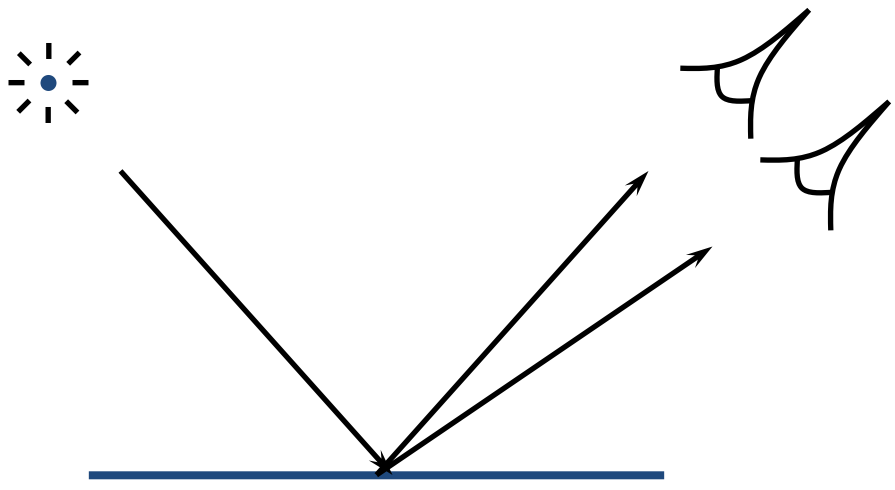
0º/45º
Diffuse Measurement



45º/45º
Specular Measurement

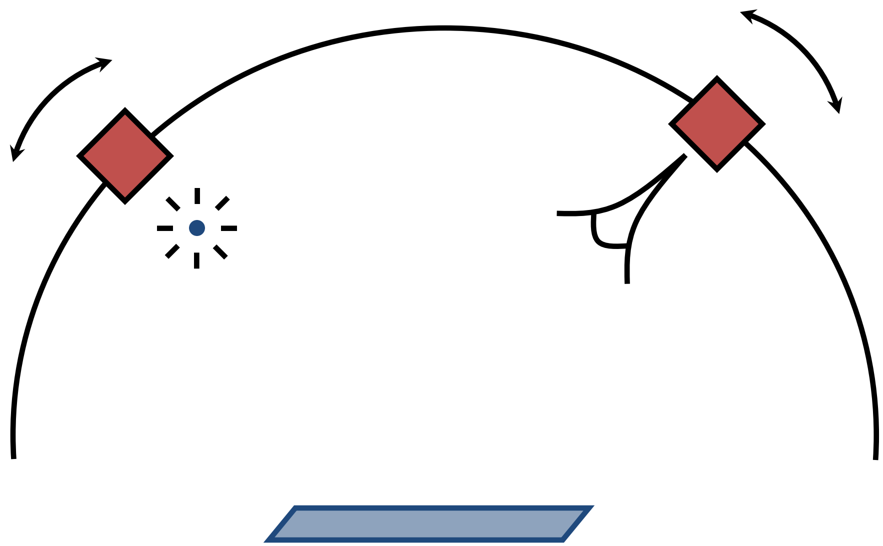
Gloss Measurements

- “Haze” is the width of a specular peak



BRDF Measurements

- Next step up: measure over a 1- or 2-D space



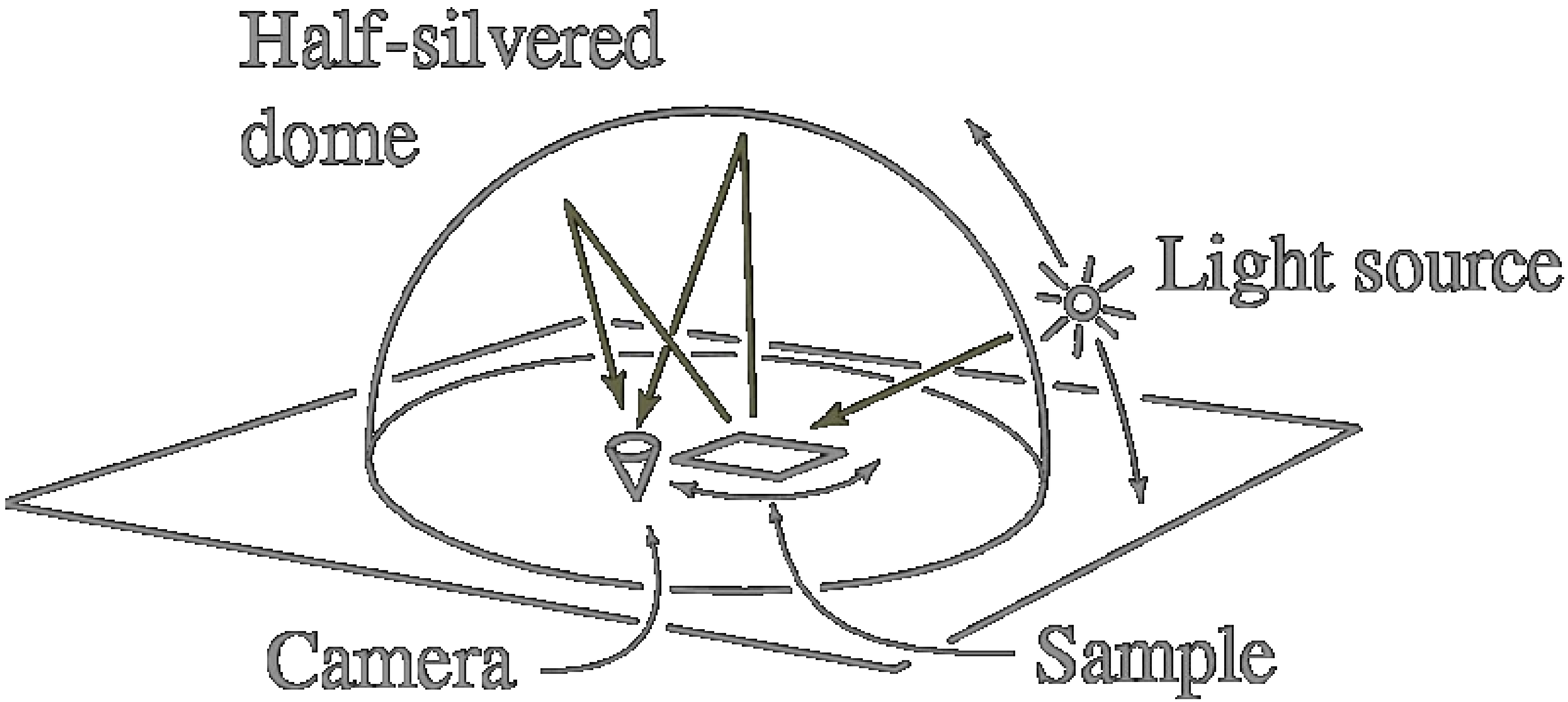
Gonioreflectometers

- Or a 4D space

Image-Based BRDF Measurement

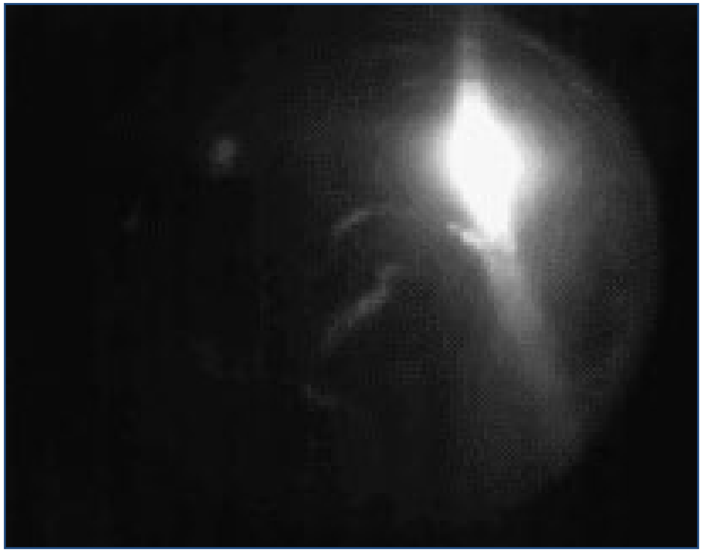
- A camera acquires with each picture a 2D image of sampled measurements
  - Requires mapping light angles to camera pixels

Ward's BRDF Measurement Setup



Ward's BRDF Measurement Setup

- Each picture captures light from a hemisphere of angles



Measurement

SIGGRAPH2005

- 20-80 million reflectance measurements per material
- Each tabulated BRDF entails 90x90x180x3=4,374,000 measurement bins