**Mies Courtyard House with Curved Elements**
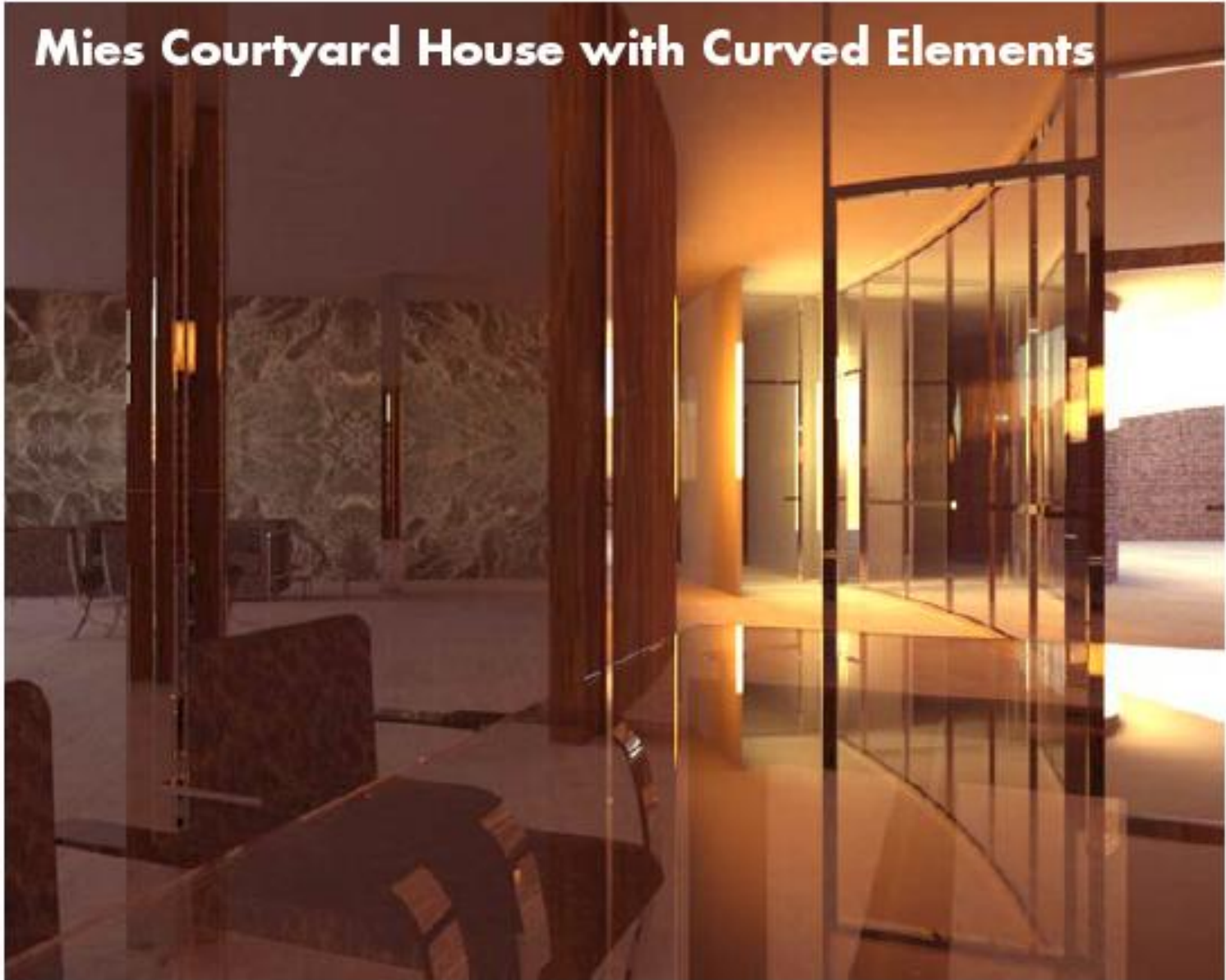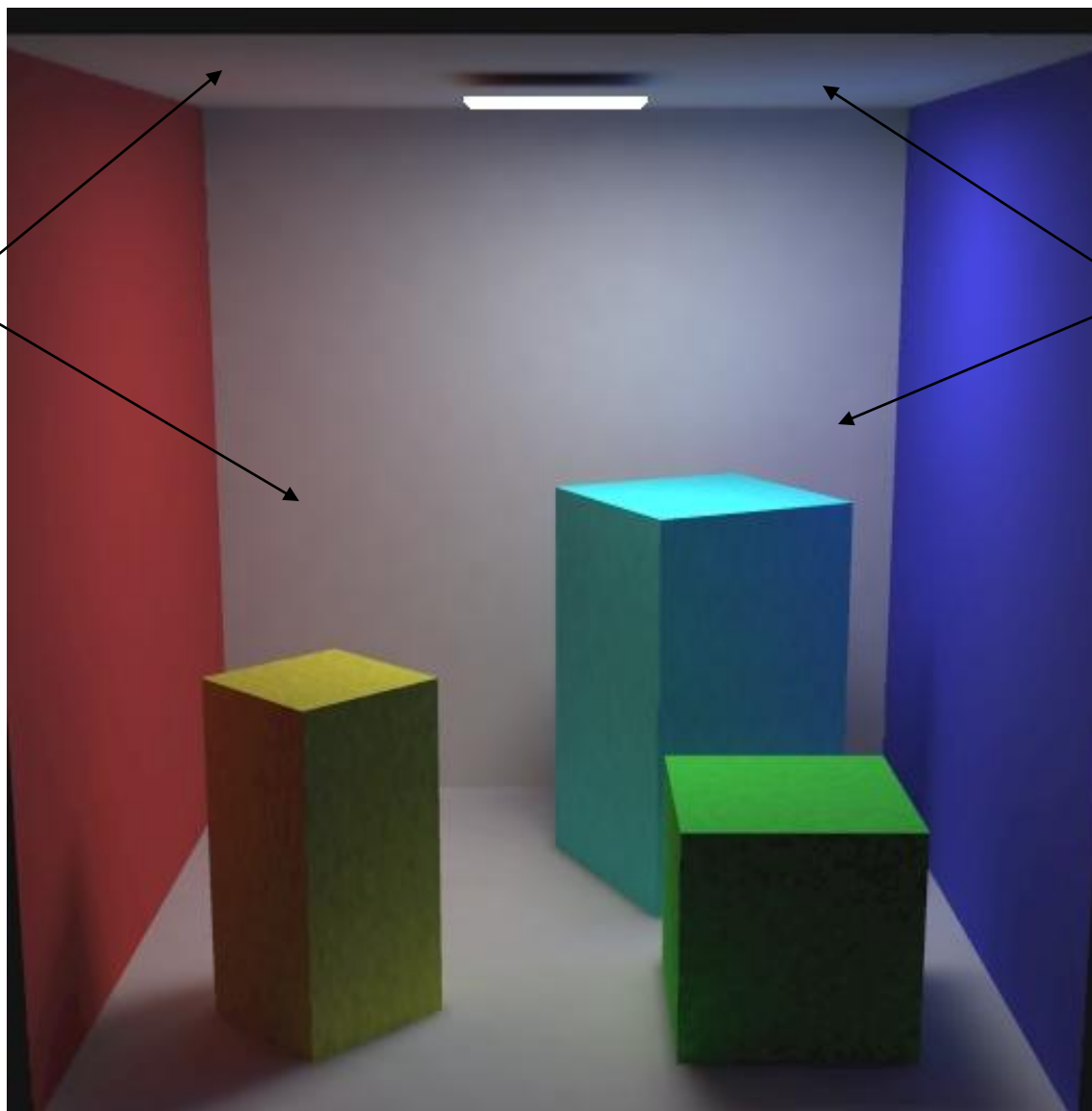
**Modeling: Stephen Duck; Rendering: Henrik Wann Jensen**

# Radiosity

# Cornell Box



red hue

blue hue

# Lighting Effects



Hard Shadows

Soft Shadows

Caustics

Indirect Illumination

The ambient lighting in the upper-right image is approximated by a constant value. This is typical of most scanline algorithms. The middle and lower-left images were rendered with a ray tracing global illumination algorithm.

The middle image was rendered with no ambient light calculations. The lower-left image was rendered with several levels of diffuse re-reflection to give a better approximation of the ambient light in this scene.

# Phong Shading
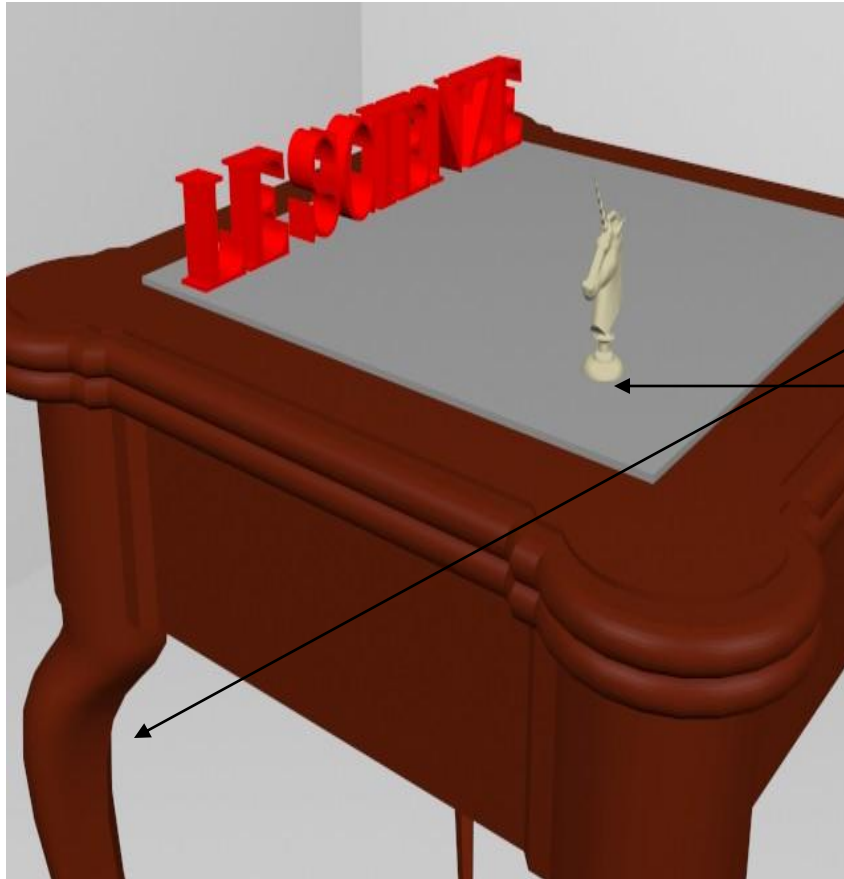
Plastic looking scene

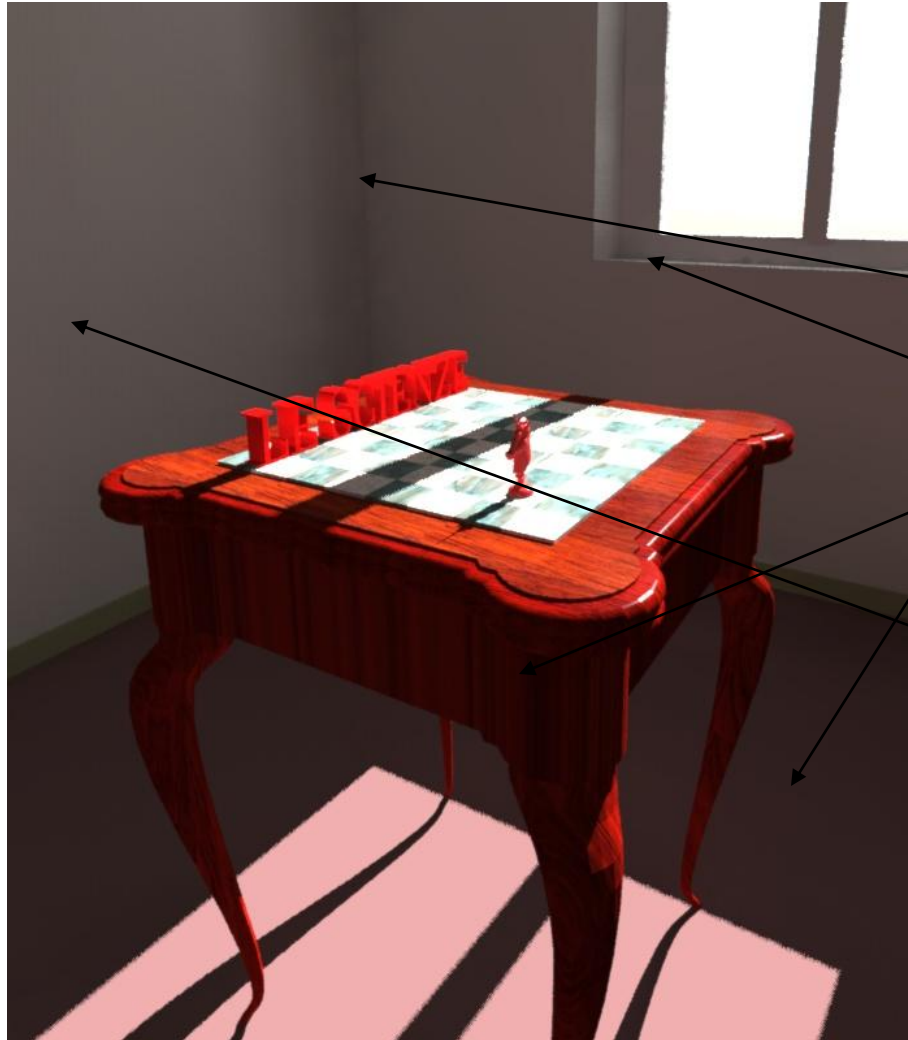- no object interactions

- no shadows

# Ray Tracing



Scene doesn't look realistic enough.

- where is the corner of room?

- is window flush with wall?

- is the carpet and wood supposed to be this dark?

# Radiosity – today's topic



Indirect lighting affects realism.

- room has a corner

- window has depth

- carpet and wood on table is lighter

- walls look more pink

# Planar piecewise constancy assumption

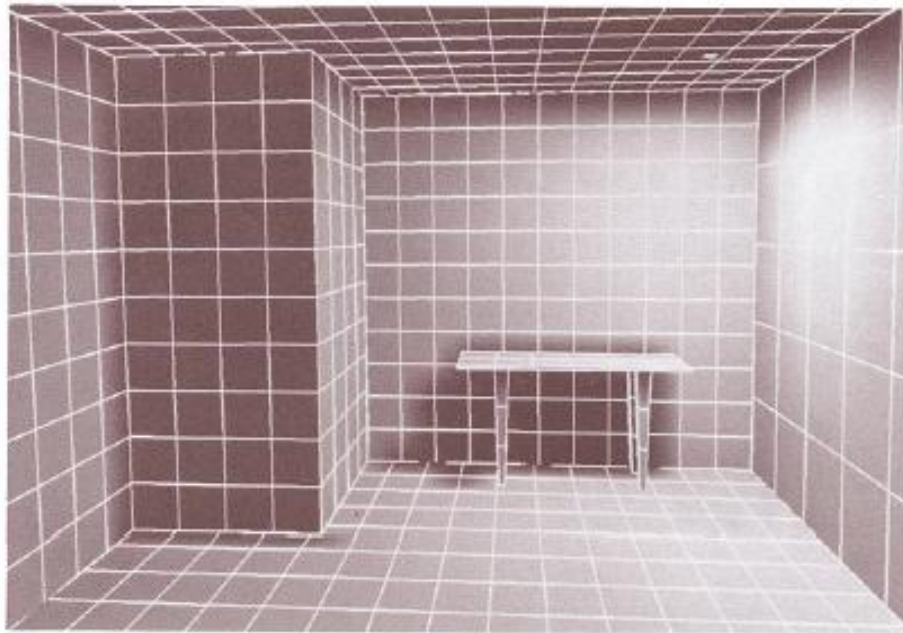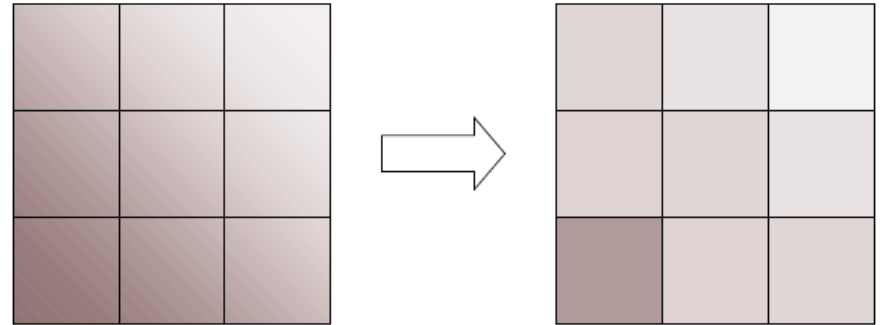- Subdivide scene into small "uniform" polygons



**Table in room sequence from Cohen and Wallace**

# Diffuse Interreflections - Radiosity

- Consider lambertian surfaces and sources.

- Radiance independent of viewing direction.

- Consider total power leaving per unit area of a surface.

- Can simulate soft shadows and color bleeding
  from diffuse surfaces.

- Used abundantly in heat transfer literature

# Irradiance, Radiosity

- Irradiance E is the power **received** per unit surface area
  - Units: $W/m^2$

- Radiosity
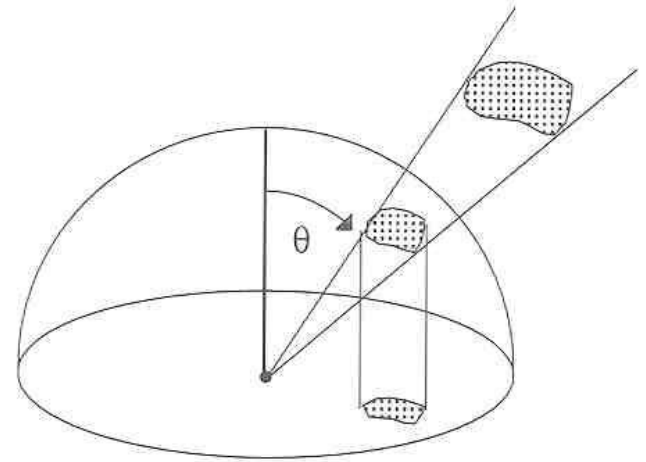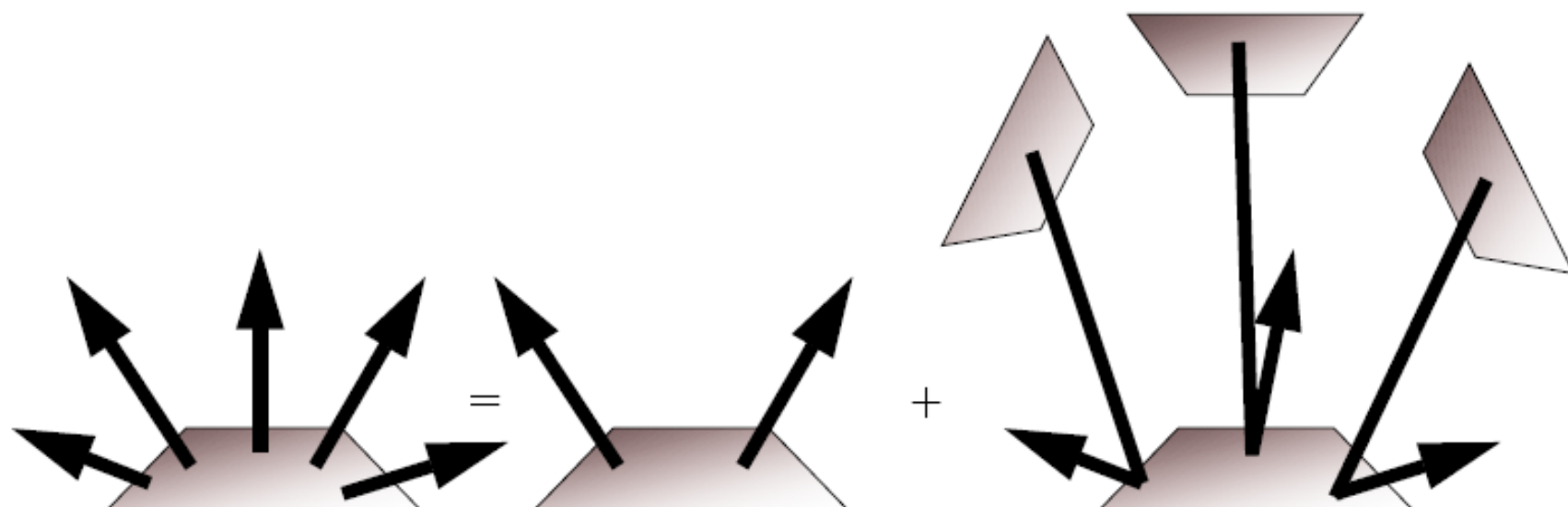  - Power per unit area **leaving** the surface (like irradiance)

**Figure 2.8:** *Projection of differential area.*

# Conservation of Energy



Emitted power = self-emitted power + received & reflected power

# Power Equation

- Power from each polygon:

$$\forall i : \Phi_i = \Phi_{ei} + \rho_i \sum_{j=1}^{N} \Phi_j \ F(j \rightarrow i)$$

- Linear System of Equations:

  - $\Phi_i$ : power of patch i (unknown)
  - $\Phi_{e,i}$ : emission of patch i (known)
  - $\rho_I$ : reflectivity of patch i (known)
  - $F(j \rightarrow i)$: form-factor (coefficients of matrix)

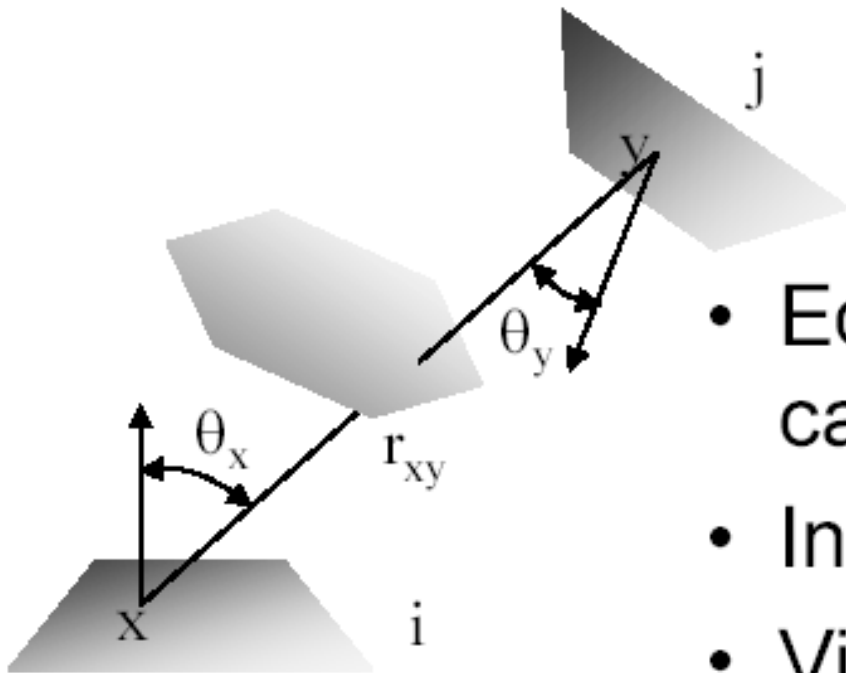# Form Factor

- $F_{j \to i}$ = the fraction of power emitted by j, which is received by i

- <u>Area</u>
  - if i is smaller, it receives less power

- <u>Orientation</u>
  - if i faces j, it receives more power

- <u>Distance</u>
  - if i is further away, it receives less power
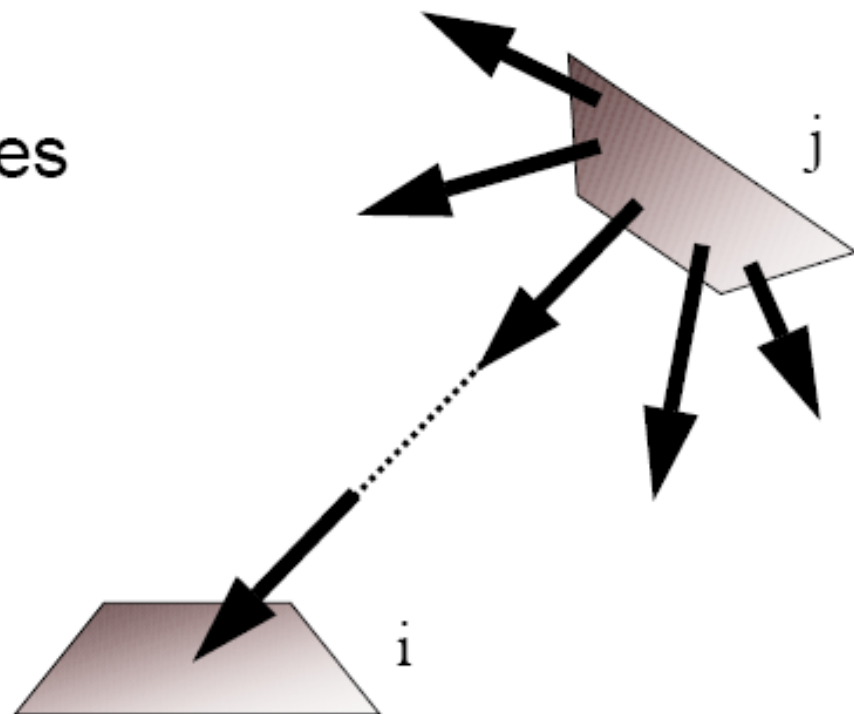
# Form Factor

$$F(j \to i) = \frac{1}{A_j} \int_{A_i} \int_{A_j} \frac{\cos\theta_x \cos\theta_y}{\pi r_{xy}^2} V(x, y)\, dA_y\, dA_x$$



- Equations for special cases (polygons)
- In general hard problem
- Visibility makes it harder

# Form Factors Invariant

$$F(j \to i) = \frac{1}{A_j} \int_{A_i} \int_{A_j} \frac{\cos \theta_x \cos \theta_y}{\pi \, r_{xy}^2} \, V(x, y) \, dA_y \, dA_x$$

$$F(i \to j) = \frac{1}{A_i} \int_{A_j} \int_{A_i} \frac{\cos \theta_x \cos \theta_y}{\pi \, r_{xy}^2} \, V(x, y) \, dA_x \, dA_y$$

$$F(i \to j) A_i = F(j \to i) A_j$$

# Form Factor Computation

$$F(j \rightarrow i) = \frac{1}{A_j} \int_{A_i} \int_{A_j} \frac{\cos \theta_x \cos \theta_y}{\pi \, r_{xy}^2} \, V(x, y) \, dA_y \, dA_y$$

- Schroeder and Hanrahan derived an analytic expression for polygonal surfaces.

- In general, computing double integral is hard.

- Use Monte Carlo Integration.

# Power →Radiosity

$$\Phi_i = \Phi_{e,i} + \rho_i \sum_{j=1}^{N} \Phi_j F(j \rightarrow i)$$

Divide by $A_i$

$$\frac{\Phi_i}{A_i} = \frac{\Phi_{e,i}}{A_i} + \rho_i \sum_{j=1}^{N} \frac{\Phi_j F(j \rightarrow i)}{A_i}$$

$$B_i = B_{e,i} + \rho_i \sum_{j=1}^{N} \frac{\Phi_j \dfrac{F(i \rightarrow j)A_i}{A_j}}{A_i}$$

$$B_i = B_{e,i} + \rho_i \sum_{j=1}^{N} \frac{\Phi_j F(i \rightarrow j)}{A_j}$$

$$B_i = B_{e,i} + \rho_i \sum_{j=1}^{N} B_j F(i \rightarrow j)$$

# Linear System of Radiosity Equations

$$\forall \text{patches i:} \qquad B_i = B_{ei} + \rho_i \sum_j F_{i \to j} B_j$$

$$\begin{bmatrix} 1 - \rho_1 F_{1 \to 1} & -\rho_1 F_{1 \to 2} & \cdots & -\rho_1 F_{1 \to n} \\ -\rho_2 F_{2 \to 1} & 1 - \rho_2 F_{2 \to 2} & \cdots & -\rho_2 F_{2 \to n} \\ \cdots & \cdots & \cdots & \cdots \\ -\rho_n F_{n \to 1} & -\rho_n F_{n \to 2} & \cdots & 1 - \rho_n F_{n \to n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \cdots \\ B_n \end{bmatrix} = \begin{bmatrix} B_{e1} \\ B_{e2} \\ \cdots \\ B_{en} \end{bmatrix}$$

Known      Unknown      Known

- Matrix Inversion to Solve for Radiosities.

# Iterative approaches

- Jacobi iteration

- Start with initial guess for energy distribution (light sources)

- Update radiosity/power of all patches based on the previous guess

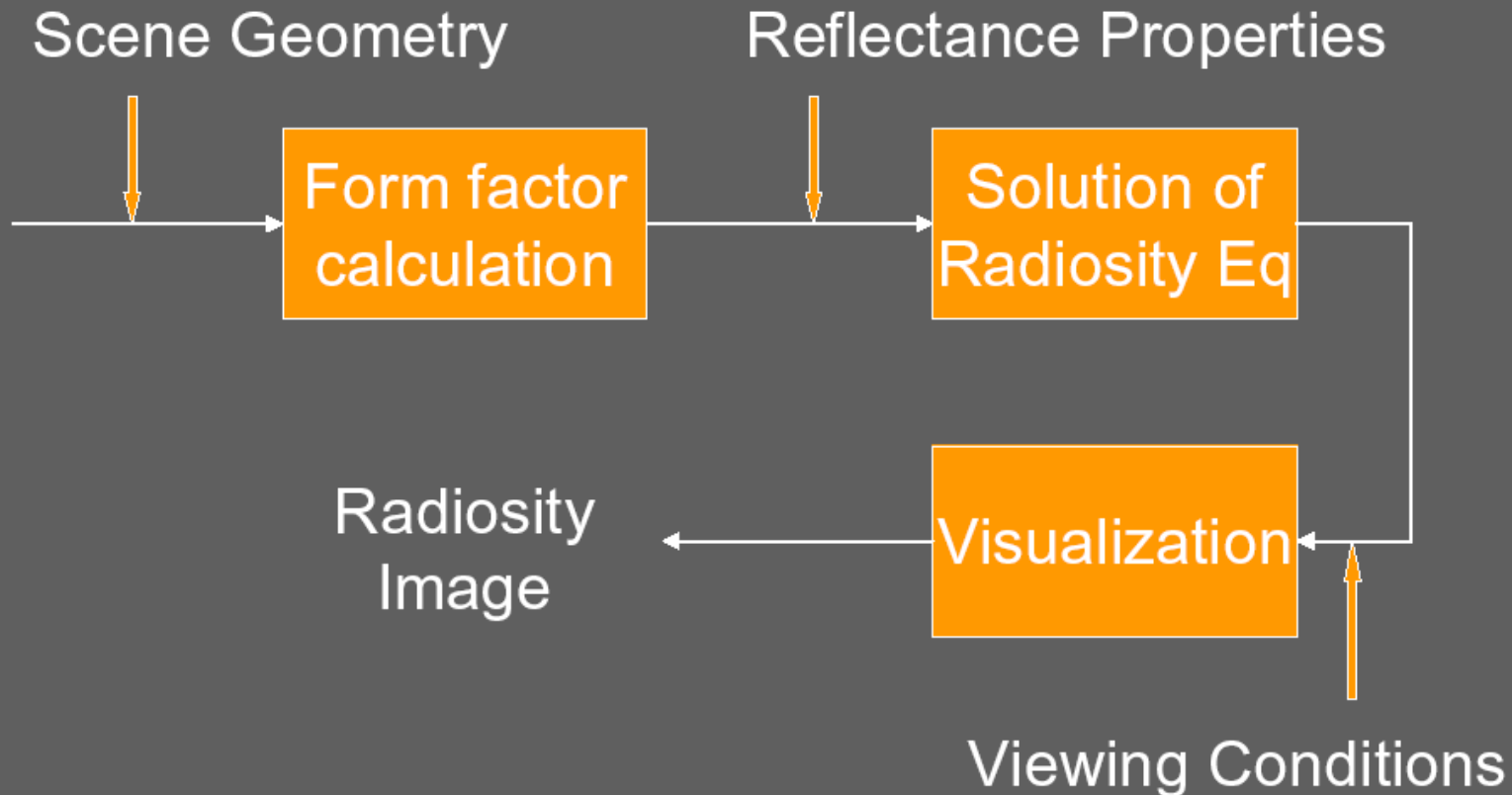$$B_i = B_{e,i} + \rho_i \sum_{j=1}^{N} B_j F(i \rightarrow j)$$

new value          old values

- Repeat until converged

# Radiosity "Pipeline"

# Being Smart about Form Factors

Form factors depend only on scene geometry.  If geometry is constant, they only need to be calculated once.

Solution of the radiosity system is independent of viewing conditions, so if only the viewer position changes, it only needs to be solved once—can walk around the scene in real-time after it's initially generated

# Being Smart about Form Factors

Form factors are complicated.  Full numeric approximation of these is expensive—many special cases may be solved analytically.

Because we assume that radiosity is constant across a patch, two patches are typically assumed to be fully inter-visible or not at all inter-visible.  That means that patches have to be small enough to resolve shadows and other complexities

# How to perform visibility testing?

Two basic methods, both of which have aliasing problems:

- Raycasting (typically slow)

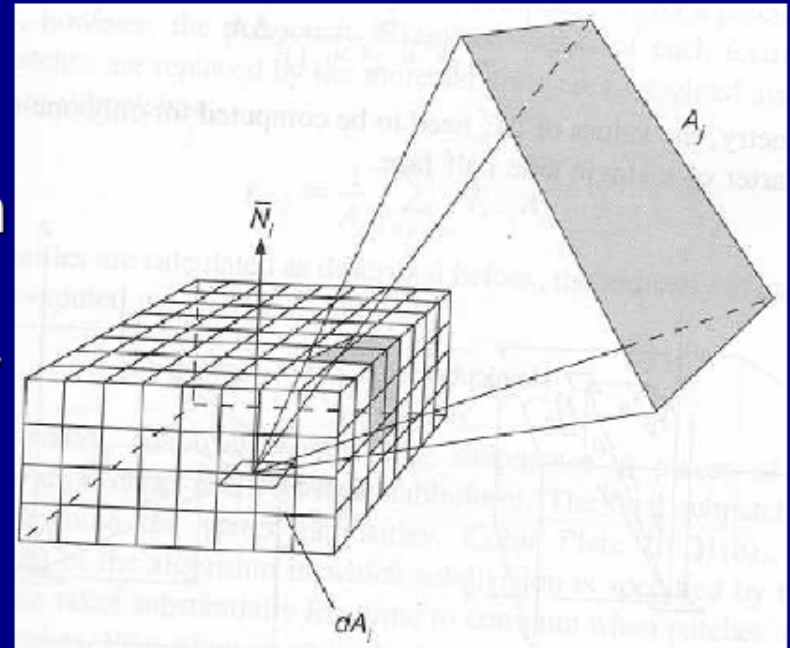- Hemicube method (z-buffer exploit)

Anti-aliasing may be performed in both cases

# Hemicube Visibility Testing

Render the entire scene from the perspective of the center of the current patch
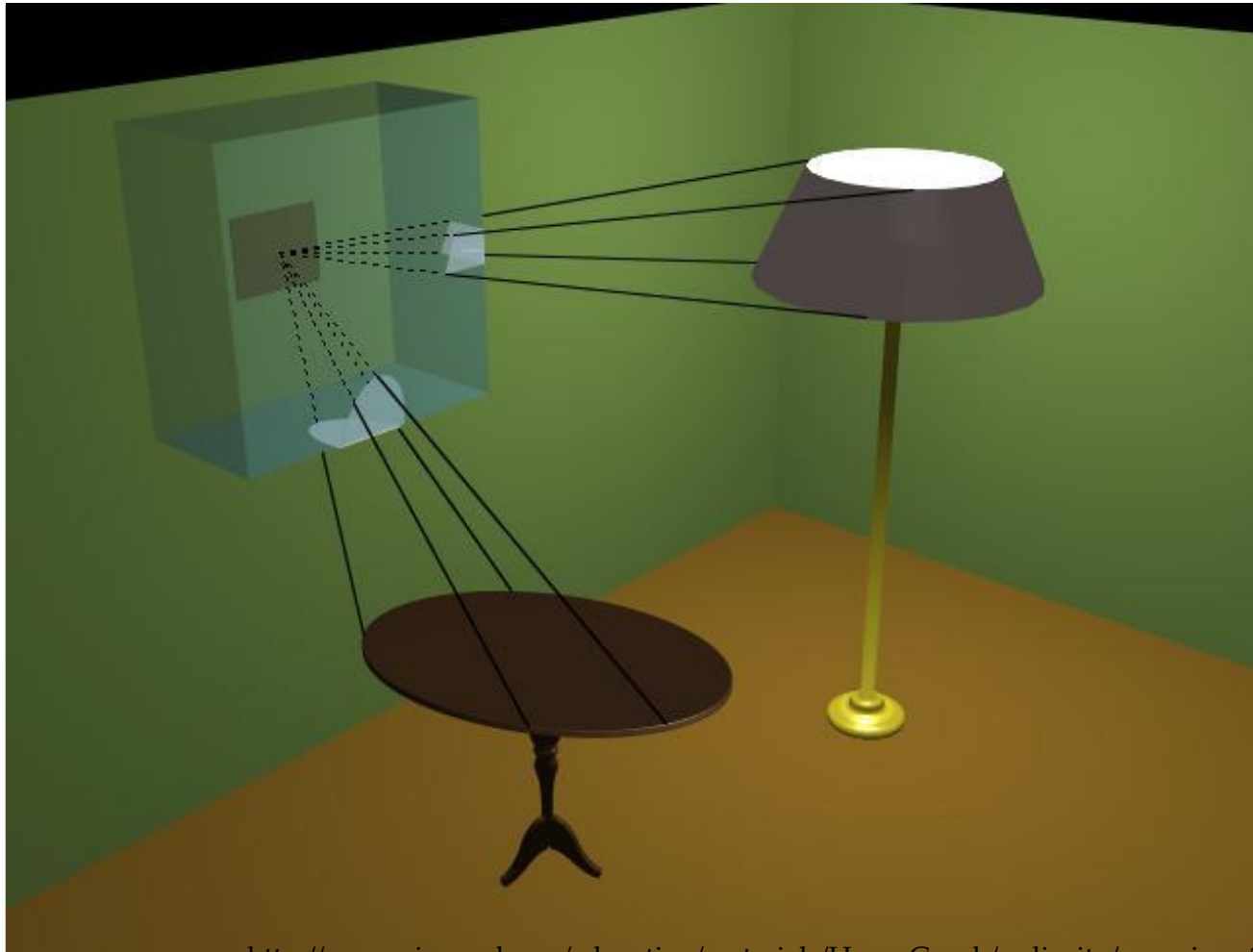
Rather than color, store patch identifiers, using the z-buffer to determine visibility
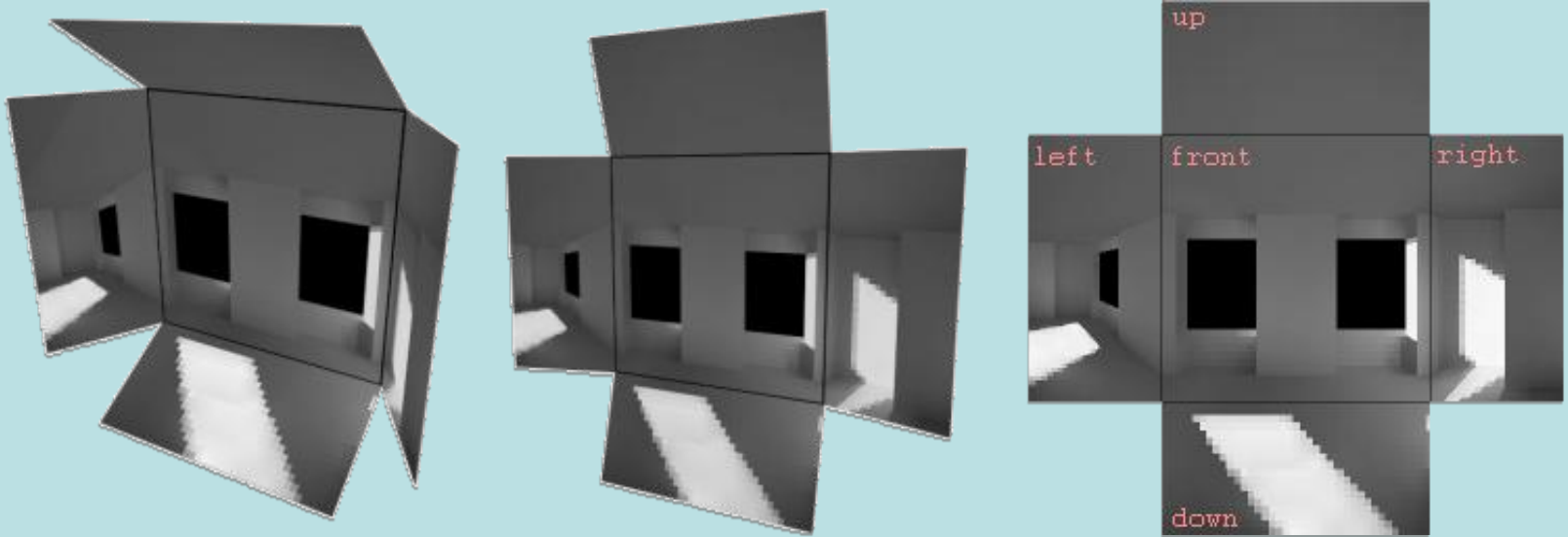
Takes advantage of graphics hardware
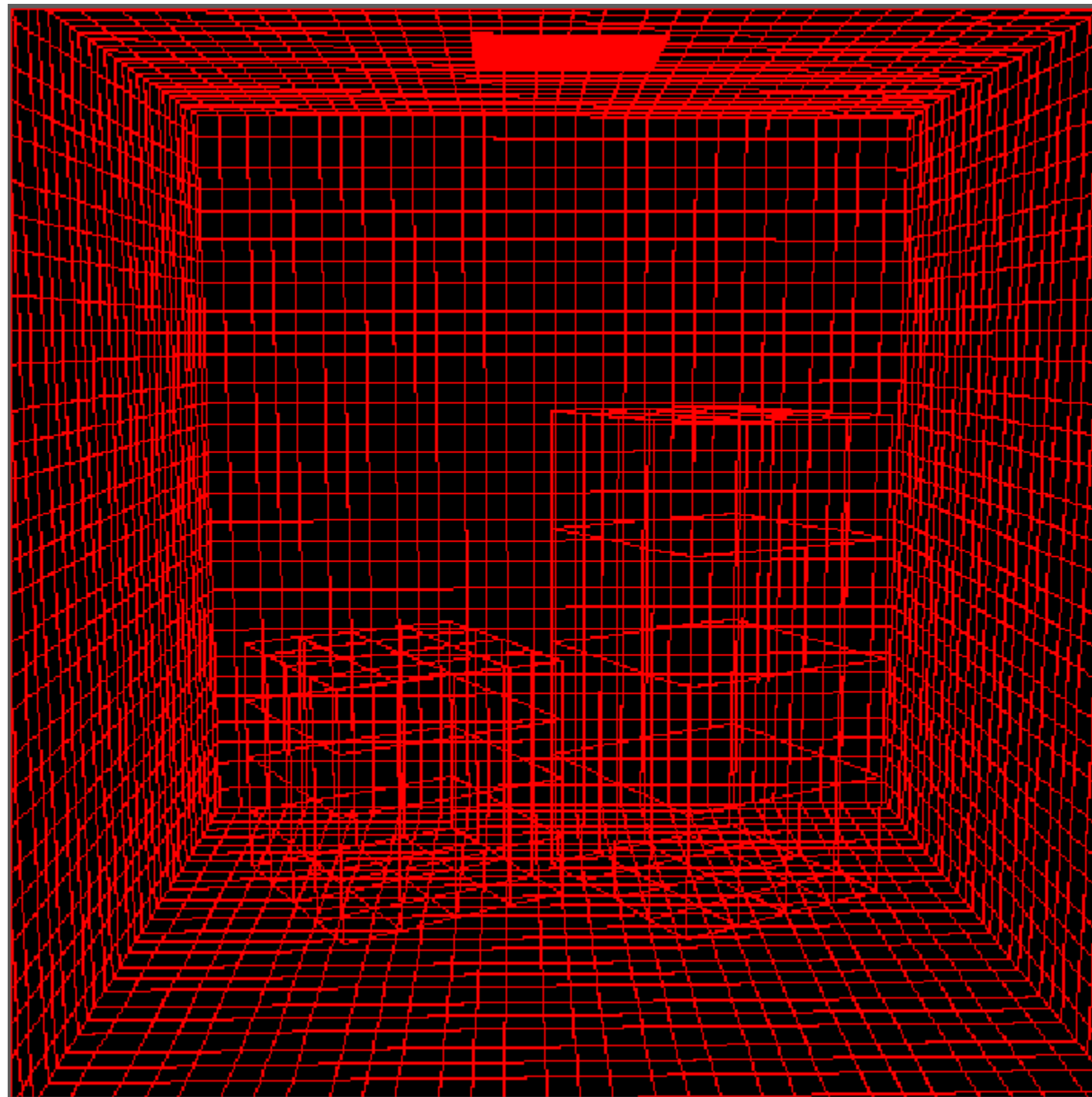


R. Ramamoorthi

# Hemicube in Action
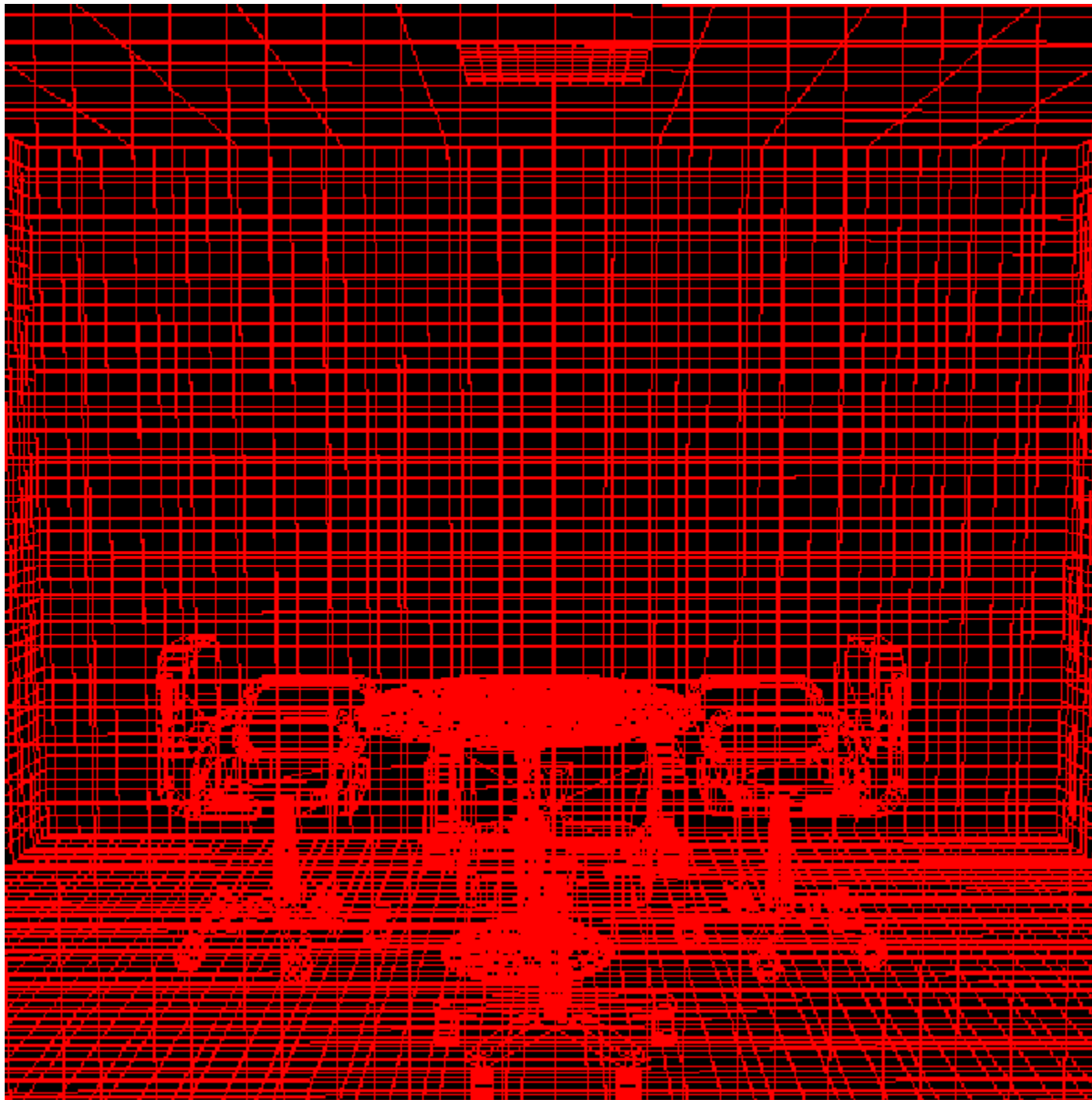
# Hemicube in Action



http://en.wikipedia.org/wiki/File:Hemicube_Unfold.gif

Wireframe

- Classical Approach

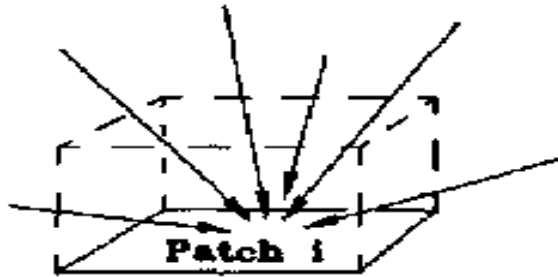- No Interpolation

Wireframe

- Classical Approach

- Low Res

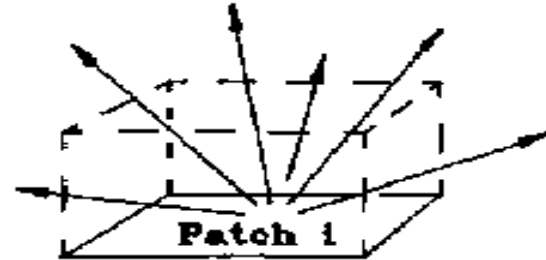- Classical Approach

- High Res

- More accurate

- Classical Approach

- High Res

- Interpolated

# Progressive Solution



GATHERING            vs.            SHOOTING

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

For all j:

$$B_j = B_j + B_i (\rho_j F_{ji})$$

where: $F_{ji} = F_{ij} A_i / A_j$

Figure 1: Gathering vs. Shooting

A Progressive Refinement Approach to Fast Radiosity Image Generation, Cohen et al 88

## PROGRESSIVE SOLUTION

The above images show increasing levels of global diffuse illumination. From left to right:  0 bounces, 1 bounce, 3 bounces.

# Sample Scenes

# Sample Scenes



**From Cohen, Chen, Wallace and Greenberg 1988**

# Sample Scenes

# Sample Scenes

# Sample Scenes

# Radiosity  Summary

Classic radiosity = finite element method

Assumptions

- Diffuse reflectance
- Usually polygonal surfaces

Advantages

- Soft shadows and indirect lighting
- View independent solution
- Precompute for a set of light sources
- Useful for walkthroughs

# Review: Local vs. Global Illumination

- Global illumination: **Ray tracing**
  - Realistic specular reflection/transmission
  - Simplified diffuse reflection*
- Global illumination: **Radiosity**
  - Realistic diffuse reflection
  - Diffuse-only: No specular interaction*

indirect          direct          both

# Radiosity Examples



http://www.autodesk.com/us/lightscape/examples/html/index.htm

11

# Raytracing Examples



© Steve Unger 1993

http://www.povray.org/

12

# Raytracing Examples

18

http://www.povray.org/

# Radiosity Examples



http://www.autodesk.com/us/lightscape/examples/html/index.htm
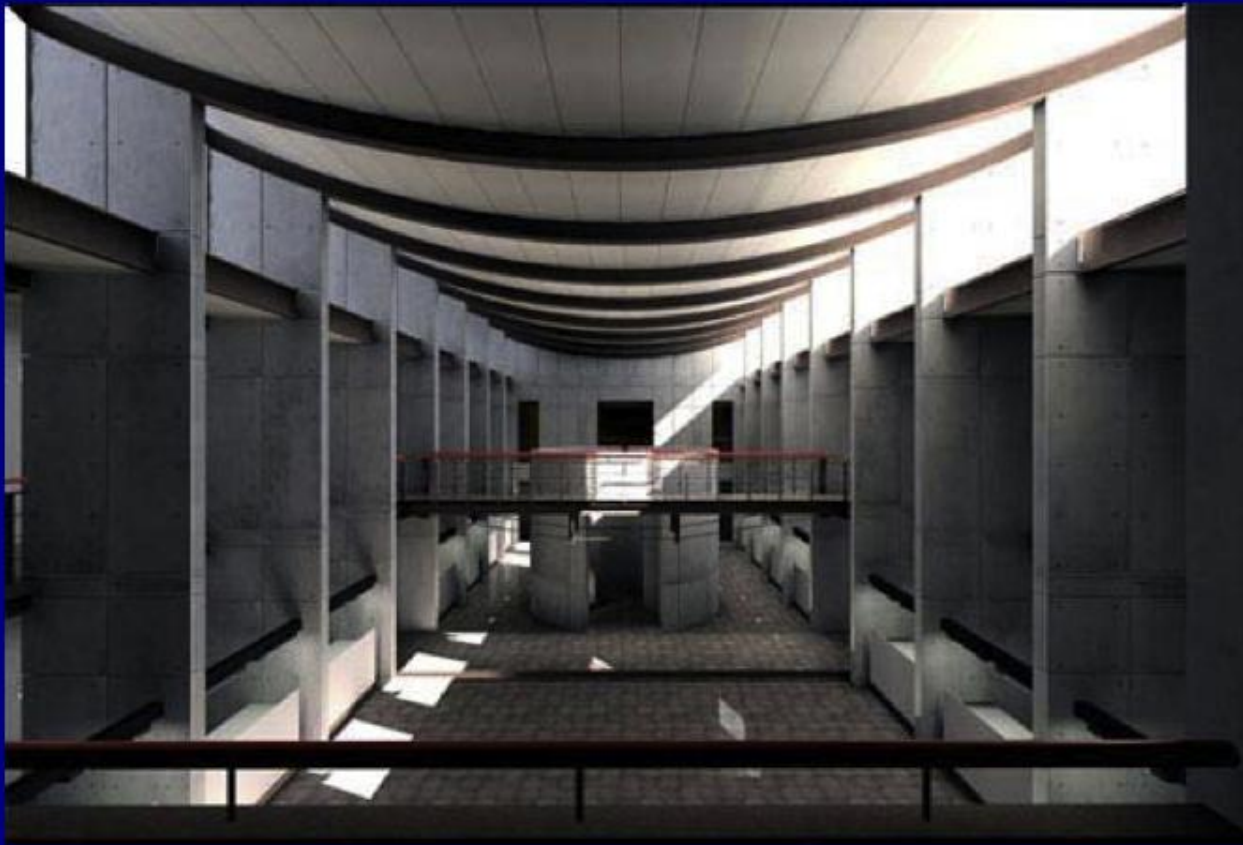
# Image vs. Object Space

- Image space: Ray tracing
  - Trace backwards from viewer
  - View-dependent calculation
  - Result: rasterized image (pixel by pixel)
- Object space: Radiosity
  - Assume only diffuse-diffuse interactions
  - View-independent calculation
  - Result: 3D model, color for each surface patch
  - Can render with OpenGL

**Doug James**

# A Better Idea: The Best of Both Worlds

Combine radiosity and raytracing

Goal: Represent four forms of light transport:

- – Diffuse -> Diffuse
- – Diffuse -> Specular
- – Specular -> Diffuse
- – Specular -> Specular

Two-pass approach, one for each method

# First Pass:  Enhanced Radiosity

## Diffuse -> Diffuse

Normal diffuse reflection model

Diffuse transmission (translucent objects) – requires modified form factor

## Specular -> Diffuse

Specular transmission (transparent objects, e.g. windows) – involves extended form factor

Specular reflection (reflective objects, e.g. mirrors) – create actual "mirror image" environment with copies of all patches.  Expensive!

# Enhanced Radiosity - Evaluation

- Only accounts for a single specular reflection (try creating "mirror image" environments for two mirrors facing each other)

- Accurate diffuse model

- Equations solved as in the classical method

- Still viewer-independent
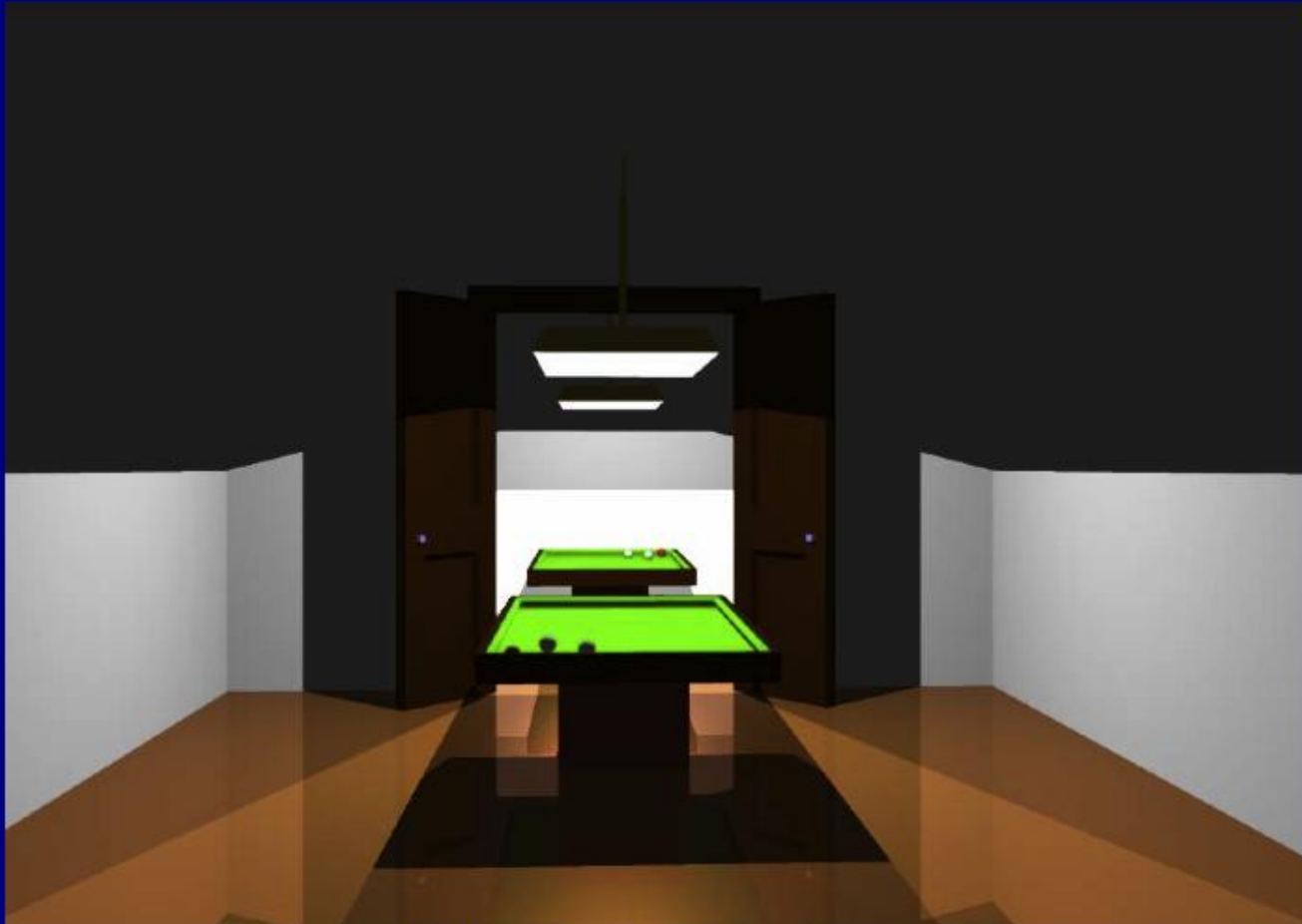
# Second Pass: Enhanced Raytracing

- ## Specular -> Specular
  - Reflection and transmission as in classical method
- ## Diffuse -> Specular
  - Use the radiosity calculated in the first pass
  - Integrate incoming light over a hemisphere (or hemicube), or approximate with a tiny frustum in the direction of reflection
  - Recurse if visible surface is specular
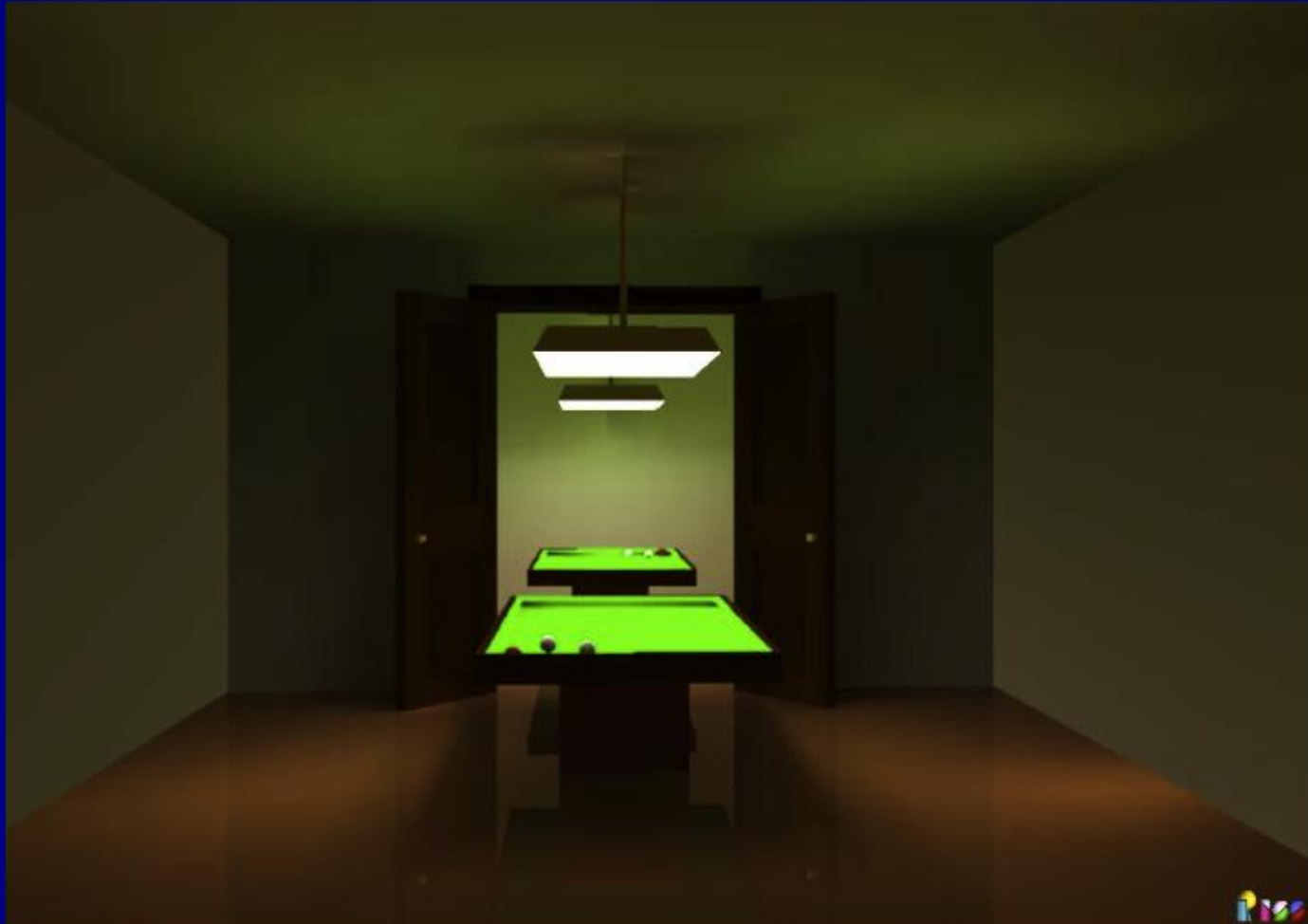
# First Pass Result

http://www.cg.tuwien.ac.at/research/rendering/rays-radio/

# Second Pass Result
## (radiosity info. not yet used, just raytracing)

http://www.cg.tuwien.ac.at/research/rendering/rays-radio/

# Combined (Final) Result

# Two-Pass Global Illumination: Evaluation

Very expensive.  Takes the cost of radiosity added to the cost of raytracing and then throws even more calculations into the mix

Many approximations remain, particularly in specular -> diffuse and diffuse -> specular transport