Computer Graphics

-Radiosity-

Oliver Bimber

Course Schedule

Type	Date	Time	Room	Topic	Comment
C1	01.03.2016	13:45-15:15	HS 18	Introduction and Course Overview	Conference
C2	15.03.2016	13:45-15:15	HS 18	Transformations and Projections	Easter Break
C3	05.04.2016	13:45-15:15	HS 18	Raster Algorithms and Depth Handling	
C4	12.04.2016	13:45-15:15	HS 18	Local Shading and Illumination	
C5	19.04.2016	13:45-15:15	HS 18	Texture Mapping Basics	
C6	26.4.2016	13:45-15:15	HS 18	Advanced Texture Mapping & Graphics Pipelines	
C7	03.05.2016	13:45-15:15	HS 18	Intermediate Exam	
C8	09.05.2016	17:15-18:45	HS 18	Global Illumination I: Raytracing	
C9	10.05.2016	13:45-15:15	HS 18	Global Illumination II: Radiosity	Conference / Holiday
C10	31.05.2016	13:45-15:15	HS 18	Volume Rendering	
C11	07.06.2016	13:45-15:15	HS 18	Scientific Data Visualization	
C12	14.06.2016	13:45-15:15	HS 18	Curves and Surfaces	
C13	21.06.2016	13:45-15:15	HS 18	Basics of Animation	
C14	28.06.2016	13:45-15:15	HS 18	Final Exam	
C15	04.10.2016	13:45-15:15	TBA	Retry Exam	

NEXT ICG LAB TALK:
10. MAY 2016, 4:00PM

Institute of Computer Graphics

Dr. Bettina Heise

Johannes Kepler University Linz

(Low) Coherent Imaging: Current State, Trends and Perspectives

Science Park 2

Room S2 059

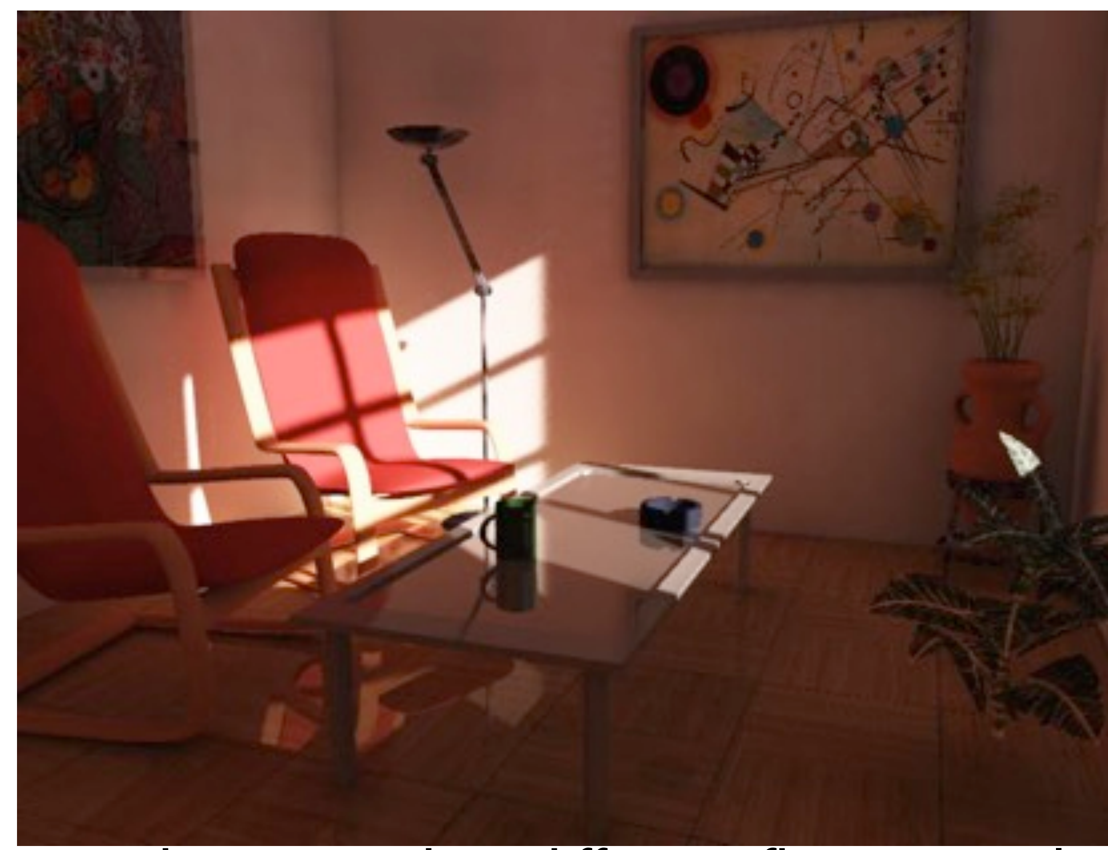
JKU

For more information about our talks visit
http://www.cg.jku.at/talks/invited

The Problem with Ray Tracing

- Ray tracing is one of the first global illumination techniques
- It handles perfect specular reflection and transmissions well
- Realistic diffuse reflection effects are difficult or costly to achieve on a ray basis with special ray tracing variants, like distribution ray tracing or path tracing
- Radiosity solves exactly this problem
- It is an efficient method for rendering realistic scattering / diffuse reflection effects
- But it is not efficient for rendering anything else (such as specular reflections or refraction, etc.)
- Thus, radiosity and ray tracing are complementary

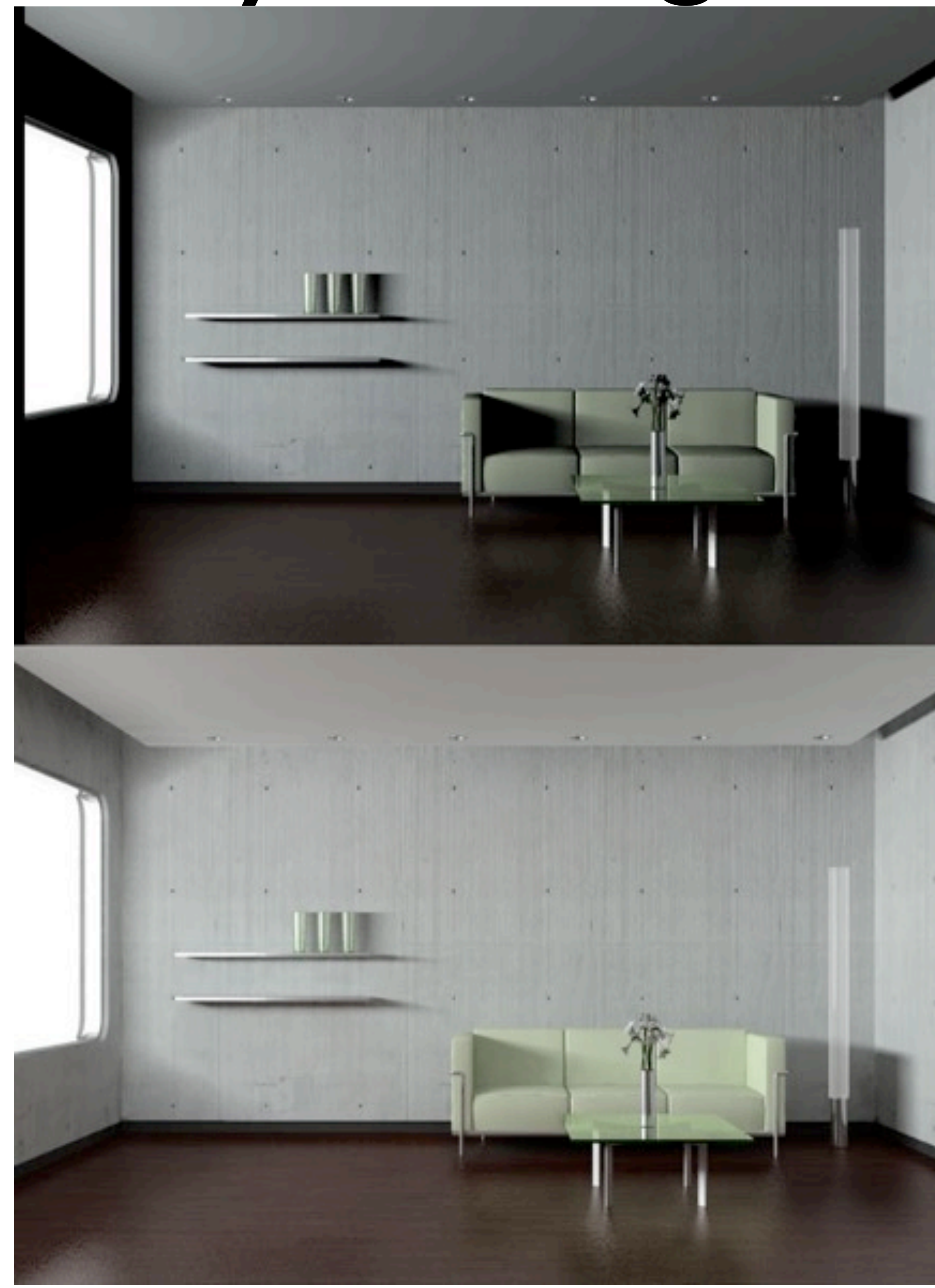
ray tracing renders specular reflections, refractions and hard shadows efficiently



radiosity renders diffuse reflections, color bleeding and soft shadows efficiently

The Problem with Ray Tracing

- Ray tracing is one of the first global illumination techniques
- It handles perfect specular reflection and transmissions well
- Realistic diffuse reflection effects are difficult or costly to achieve on a ray basis with special ray tracing variants, like distribution ray tracing or path tracing
- Radiosity solves exactly this problem
- It is an efficient method for rendering realistic scattering / diffuse reflection effects
- But it is not efficient for rendering anything else (such as specular reflections or refraction, etc.)
- Thus, radiosity and ray tracing are complementary



ray tracing only (top) and
ray tracing + radiosity (bottom)

The Problem with Ray Tracing

- Ray tracing is one of the first global illumination techniques
- It handles perfect specular reflection and transmissions well
- Realistic diffuse reflection effects are difficult or costly to achieve on a ray basis with special ray tracing variants, like distribution ray tracing or path tracing
- Radiosity solves exactly this problem
- It is an efficient method for rendering realistic scattering / diffuse reflection effects
- But it is not efficient for rendering anything else (such as specular reflections or refraction, etc.)
- Thus, radiosity and ray tracing are complementary

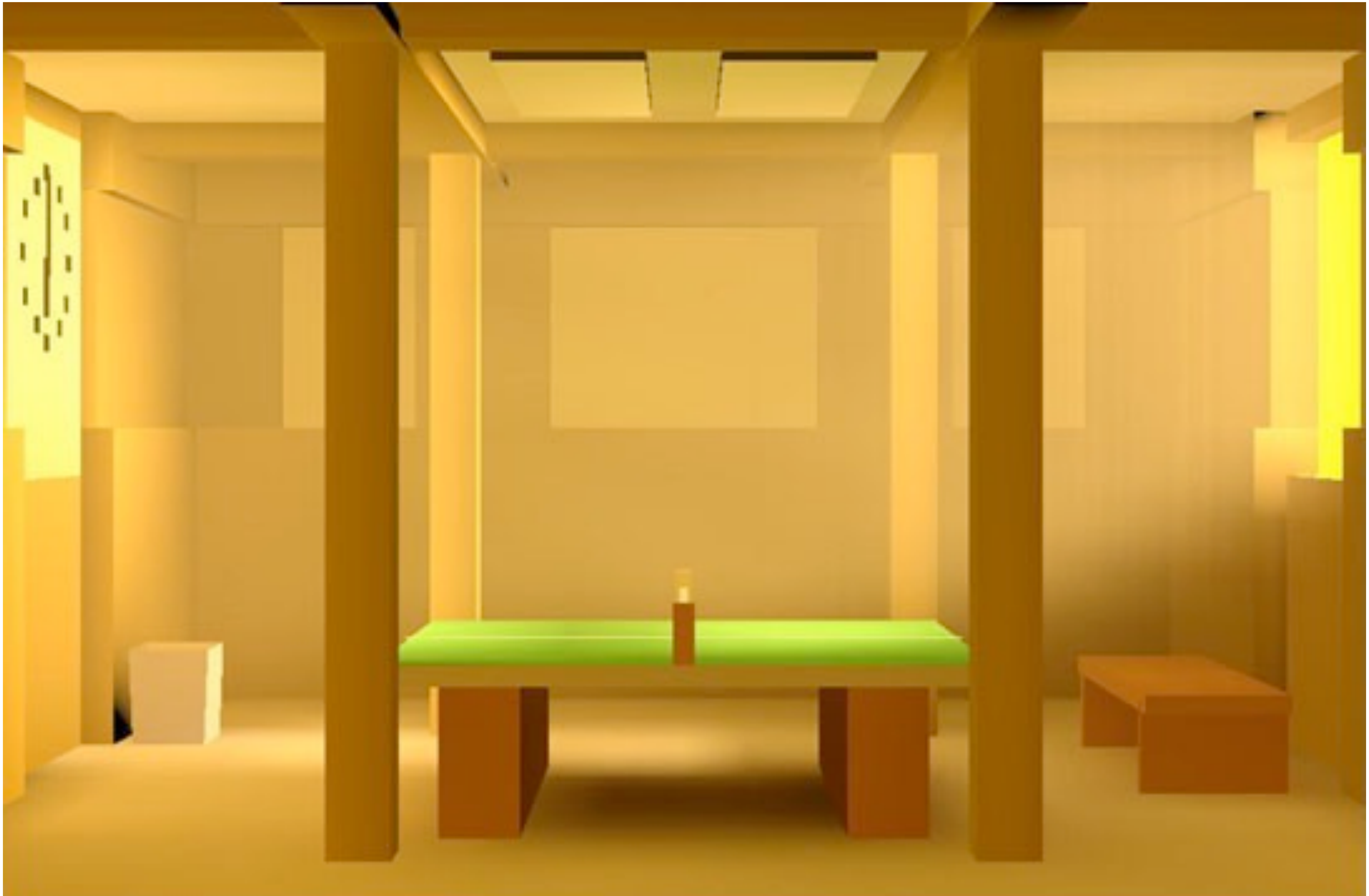
ray tracing fails if diffuse inter-reflections are dominant for scene lighting

radiosity solves this problem

Radiosity Examples

Radiosity Examples

Radiosity Examples



Radiosity Examples

Radiosity Examples

Radiosity Examples

Radiosity Examples

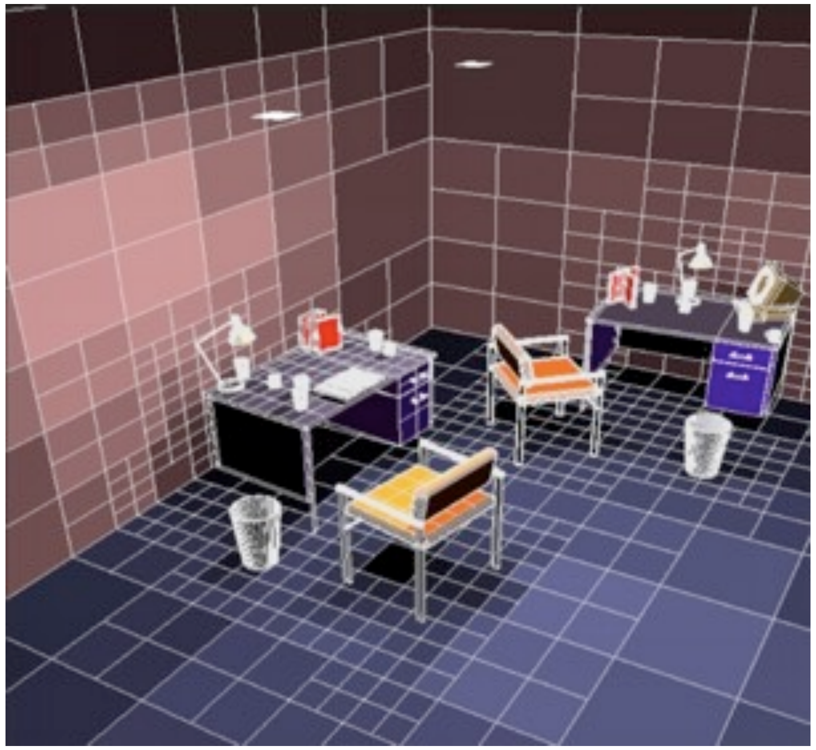
Radiosity Examples

Radiosity Examples

Radiosity Examples

Radiosity

- Radiosity is based on the principles of heat transfer and is predominantly applied for indoor scenes, where diffuse inter-reflections are dominant
- The scene is divided into surface patches (uniformly or non-uniformly)
- Even light sources are approximated with patches
- In radiosity, we compute the transfer of light energy from patch to patch
- The cost of the algorithm is $O(n^2)$ for n patches
- Thus, while ray tracing is an image space algorithm, radiosity is computed in object space



scene is divided into patches for which the radiosity solution are computed

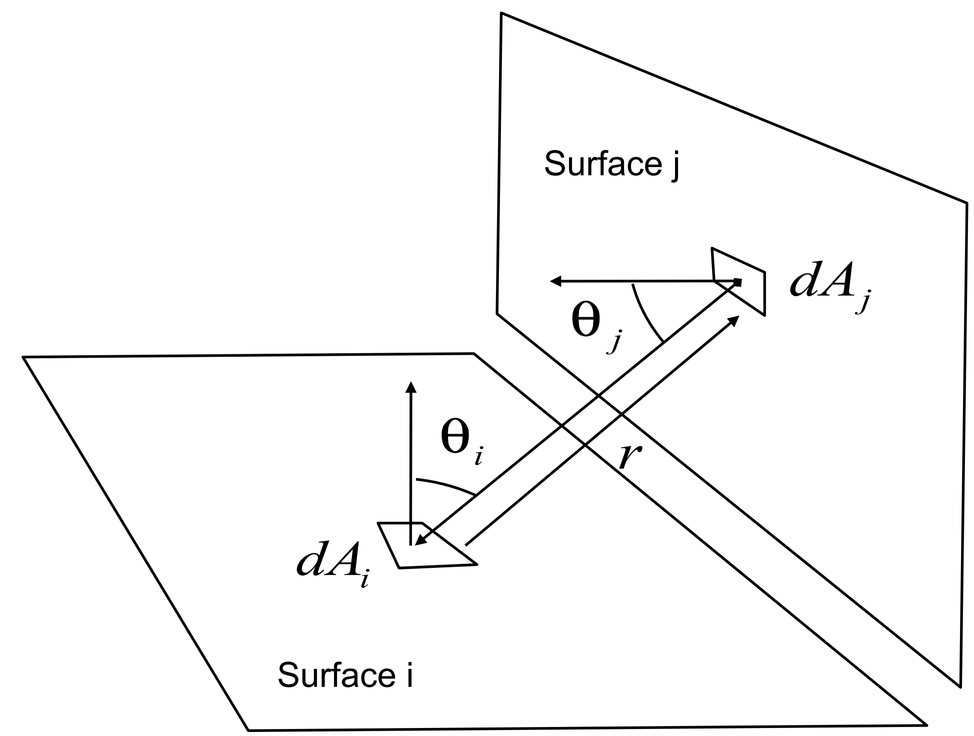
the result is then interpolated

Surface-to-Surface Form Factor

- The surface-to-surface form factor is the radiative energy leaving the source (unit) area and strikes the destination (unit) area, divided by PI (see next slide)
- Form factors between surfaces are reciprocal (duality of light transport: energy transfer is equal in both directions, if properties of patches are swapped)
- What do we do for larger (non-unit area) surfaces?

$$F_{ji} dA_j = F_{ij} dA_i$$

form factors are reciprocal



$$F_{ji} dA_i dA_j = \frac{\cos(\theta_i)\cos(\theta_j)}{\pi r^2}$$

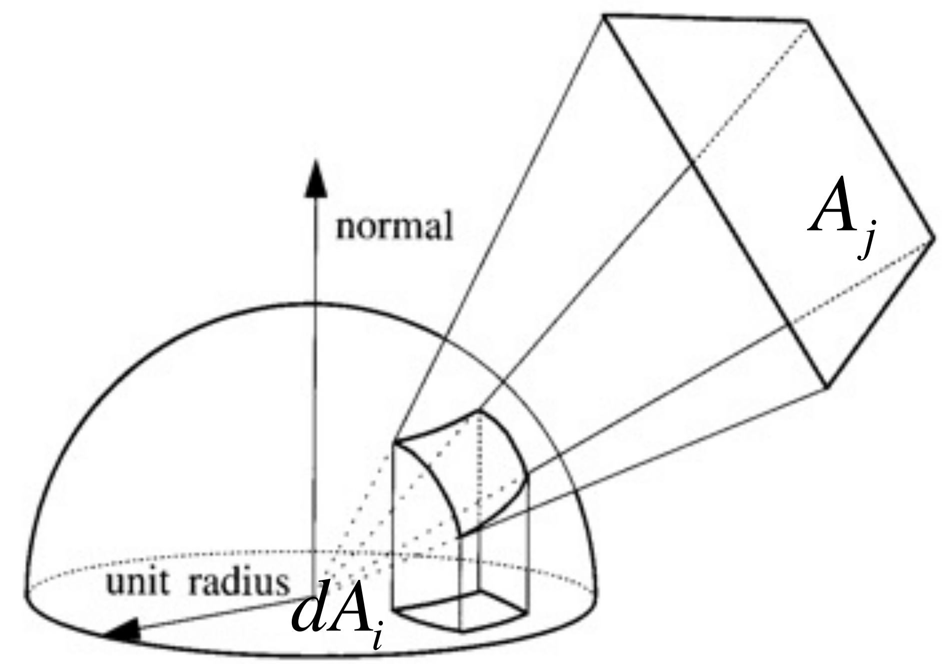
form factor between differential areas dA_i and dA_j

The Nusselt Analog

- Nusselt developed a geometric analog which allows the simple and accurate calculation of the form factor between a surface and a point on a second surface
- The Nusselt analog involves placing a hemispherical projection body, with unit radius, at a point on a surface
- The second surface is spherically projected onto the projection body, then cylindrically projected onto the base of the hemisphere
- The form factor for differential area dA_i equals the area projected on the base of the hemisphere divided by the area of the base of the hemisphere (which is PI for r=1)
- Some surfaces might not be directly visible by others, because they are blocked (e.g., occlusion)
- To handle this, we add a binary visibility factor V_{ji}
- As we will see later, V_{ji} zeros rows and columns in our equation system (the diagonal remains)

$$F_{ji} dA_i A_j = \int_{A_j} \frac{\cos(\theta_i)\cos(\theta_j)}{\pi r^2} dA_j$$

form factor between differential area dA_i and surface A_j

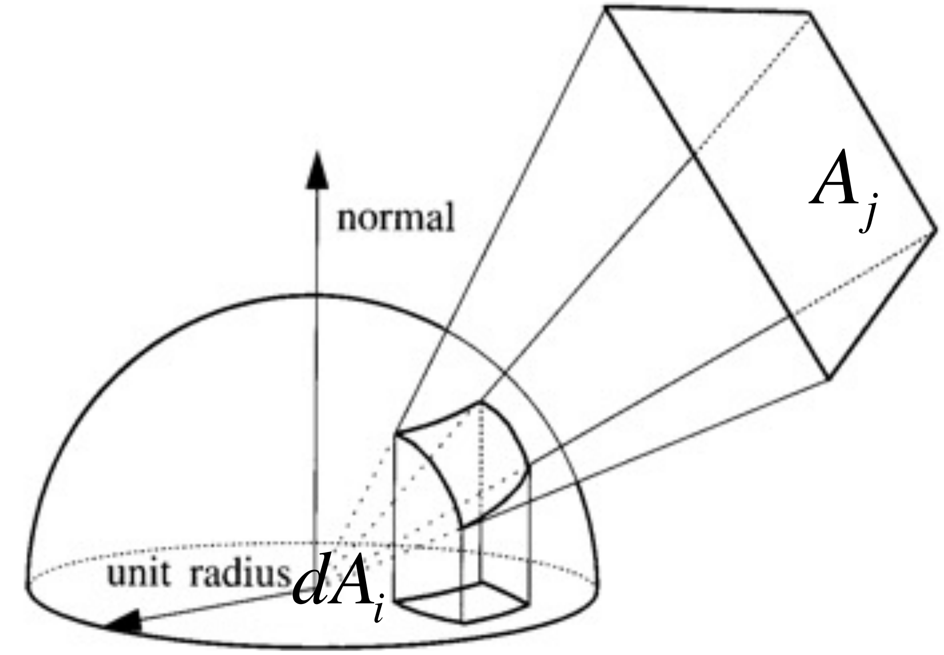


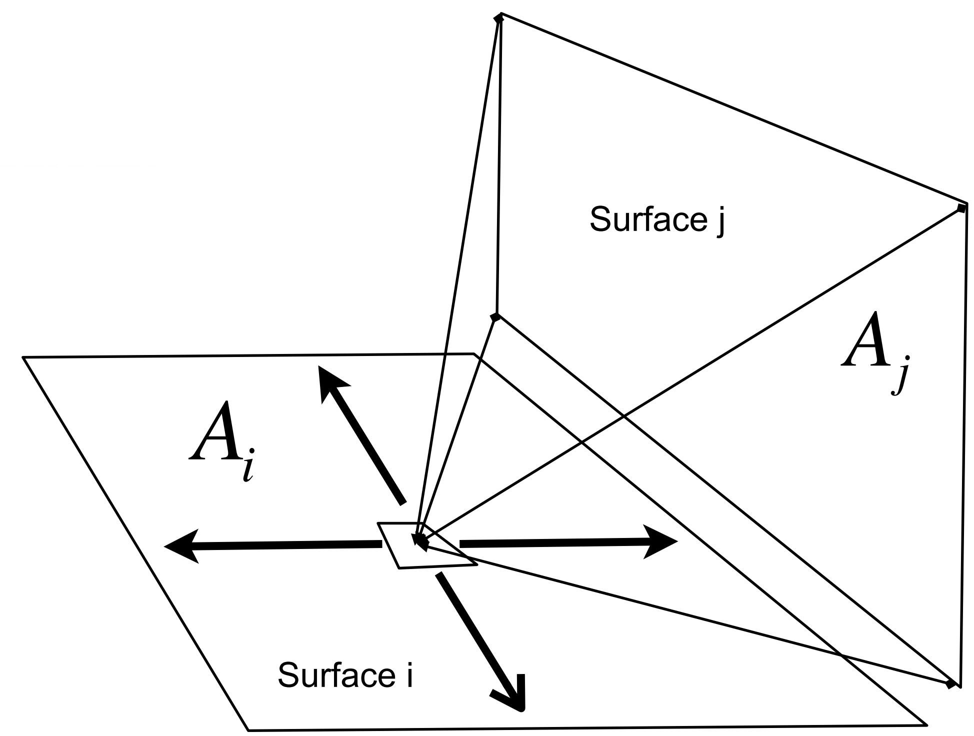
$$F_{ji} dA_i A_j = \int_{A_j} \frac{\cos(\theta_i)\cos(\theta_j)}{\pi r^2} V_{ji} dA_j$$

form factor between differential areas dA_i and surface A_j with binary visibility factor V_{ji}

Surface-to-Surface Form Factor

- The projection of A_j on the base of the hemisphere is not equal for all points on the surface A_i
- Therefore, the exact form factor between two surfaces A_i and A_j must be determined through a double integral
- Integrating over both surfaces, however, is computationally far too expensive
- How can we determine the form factors efficiently in CG now?



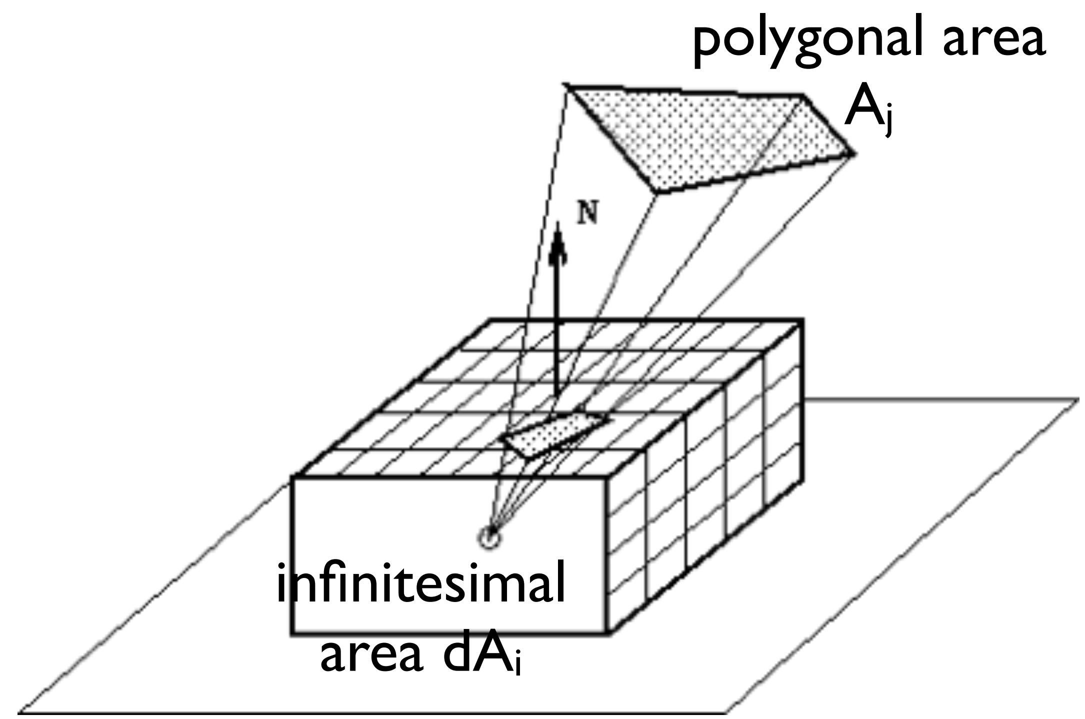


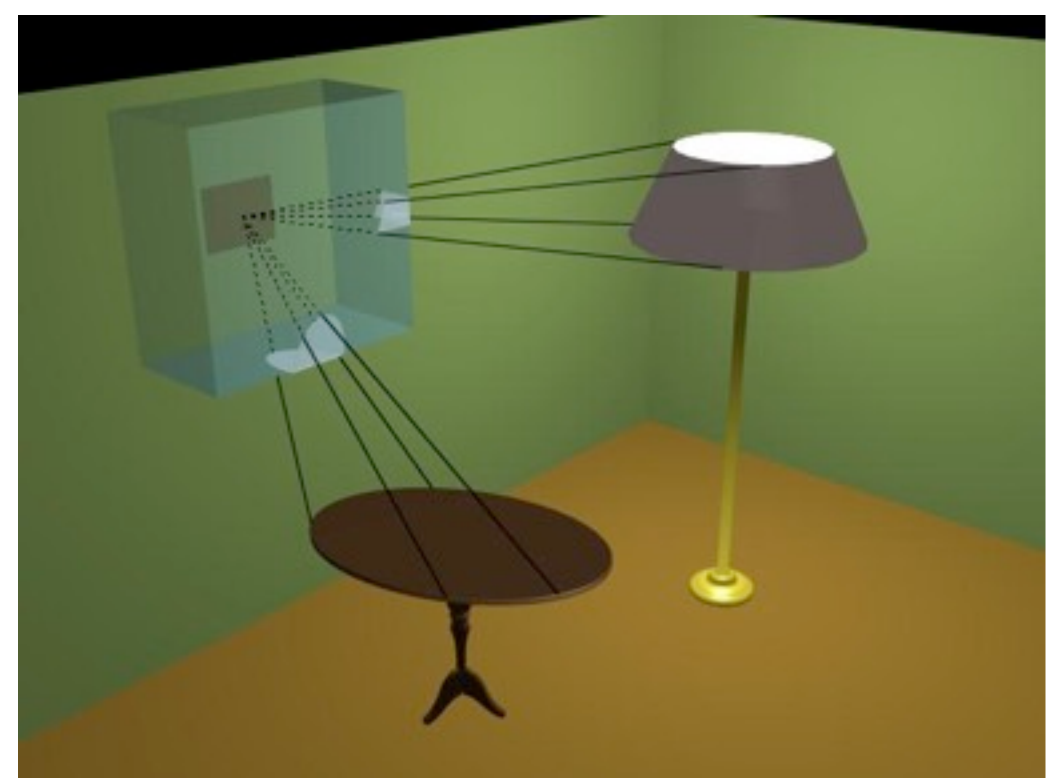
$$F_{ji}A_iA_j = F_{ji} = \frac{1}{A_i}\int_{A_i}\int_{A_j}\frac{\cos(\theta_i)\cos(\theta_j)}{\pi r^2}V_{ji}\,dA_j\,dA_i$$

overall form factor between areas A_i and A_j

The Hemicube Method

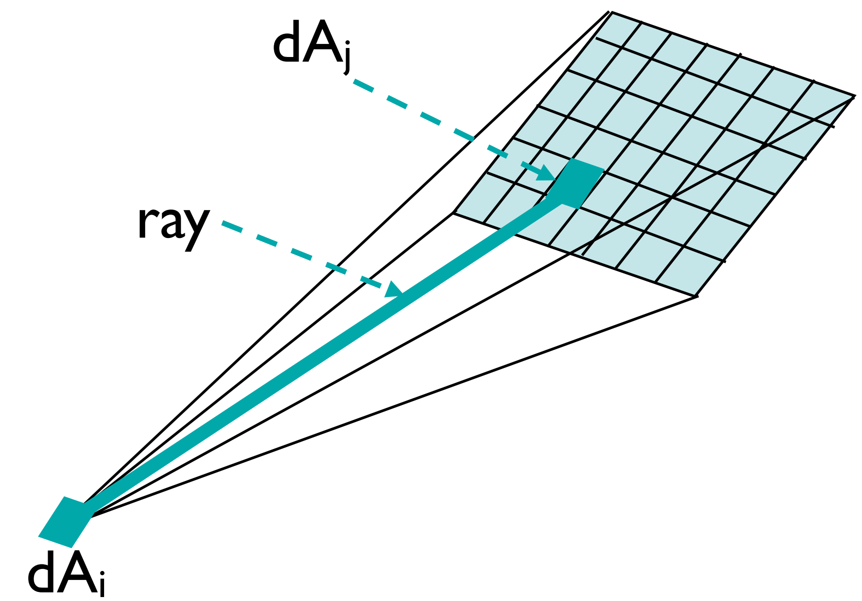
- The hemicube method is an approximation of Nusselt‘s analog
- It determines the form factor between a point dA_i and a polygon A_j
- The hemicube approximates the hemisphere because the flat projection planes are computationally less expensive
- It is centered on top of dA_i
- For convenience, a cube with 1 unit high and with a top face of 2 x 2 is used (i.e., side faces are 2 wide by 1 high)
- The cube is subdivided into cells with a defined resolution (e.g., 512 by 512 for the top)
- Then project all scene polygons onto cube cells (projection onto mutually orthogonal planes, including Z-buffering=visibility) and store patch index at corresponding cell (for fast look-up)
- The form factor of a polygon is the sum of covered cell areas (sometimes weighted differently for top and sides)





Area Sampling

- The advantage of the hemicube method is, that it is compatible with CG techniques
- Thus, it can be implemented on the graphics card (simple projections onto planes)
- The disadvantage is, that errors appear due to discrete approximation (e.g., aliasing errors through sampling, visibility errors, proximity errors) - this depends on the cell resolution (if large, then this becomes inefficient)
- An alternative is called area sampling
- Area sampling is slower than hemicube, but it is as accurate as desired for each polygon
- Polygon resolution can be adaptively changed
- Area sampling is the preferred technique today (hemicube was the original method)
- Now we have the form factors - but how to solve the chicken-and-egg problem of radiosity?



subdivide A_j into small pieces dA_j
for all dA_j
    cast ray dA_j-dA_i to determine V_{ji}
    if visible
        compute FdA_idA_j
        sum up FdA_iA_j +=FdA_idA_j
    endif
endfor

Radiosity Theory

- Radiosity B is defined as the energy per unit area leaving a surface patch per unit time
- It is the sum of reflected R and directly emitted E light at a surface patch
- Thus: radiosity x area = emitted energy + reflected energy
- This is obviously a recursive problem - how to solve it?

radiosity of A_i

reflectance of A_i

$$\underbrace{B_i A_i}_{\text{total energy leaving } A_i} = \underbrace{E_i A_i}_{\text{emitted energy at } A_i} + \underbrace{R_i \int_j \overbrace{B_j F_{ji} A_j}^{\text{energy transfered from all patches } A_j \text{ to } A_i}}_{\text{reflected energy at } A_i}$$

Radiosity Theory

- Radiosity B is defined as the energy per unit area leaving a surface patch per unit time
- It is the sum of reflected R and directly emitted E light at a surface patch
- Thus: radiosity x area = emitted energy + reflected energy
- This is obviously a recursive problem - how to solve it?
- Consider the reciprocal relationship of the form factor allows for canceling the area (by dividing both sides through A_i)
- And then we can discretize by replacing integration through summation
- What do we have now?

$$\underbrace{\overset{\text{radiosity of } A_i}{B_i} A_i}_{\text{total energy leaving } A_i} = \underbrace{E_i A_i}_{\text{emitted energy at } A_i} + \underbrace{\overset{\text{reflectance of } A_i}{R_i} \int_j \overbrace{B_j F_{ji} A_j}^{\text{energy transfered from all patches } A_j \text{ to } A_i}}_{\text{reflected energy at } A_i}$$

$$F_{ji} A_j = \boxed{F_{ij} A_i}$$

$$B_i A_i = E_i A_i + R_i \int_j B_j \boxed{F_{ij} A_i}$$

$$B_i = E_i + R_i \int_j B_j F_{ij}$$

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

Radiosity Theory

- We have a linear equation!
- Such an equation exists for each of the n surface patches
- Considering them all simultaneously leads us to a linear equation system
- Solving this equation system for B is normally done iteratively since it is quite large (i.e. direct methods like Gaussian elimination is not efficient)
  - Direct methods with $O(n^3)$ like Gaussian elimination (in most cases not possible, since too large)
  - Gathering using iterative methods with $O(n^2)$ like Gauss-Seidel, Jacobi (can be done on GPU, see www.gpgpu.org)
  - Shooting using iterative methods with $O(n^2)$ like Southwell

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

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

$$\begin{array}{cccc}
\text{known} & & & \text{known} \\
\begin{bmatrix} 1-R_1F_{11} & -R_1F_{12} & \cdots & -R_1F_{1n} \\ -R_2F_{21} & 1-R_2F_{22} & \cdots & -R_2F_{2n} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ -R_nF_{n1} & -R_nF_{n2} & \cdots & 1-R_nF_{nn} \end{bmatrix} &
\begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ B_n \end{bmatrix} & = &
\begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_n \end{bmatrix} \\
 & \text{unknown} & &
\end{array}$$

$$M_{rf} \cdot b = e$$

$$b = e \cdot M_{rf}^{-1}$$

Gathering vs. Shooting

- Gathering
  - The light leaving a patch is computed by gathering the light from the rest of the environment
  - Initially: B=E (if patch is emitting)
  - One step updates one patch by gathering contributions from other patches
  - This has to be repeated iteratively until we converge at a solution

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

$$\begin{bmatrix} X \\ \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} X \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} X & X & X & X & X \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{bmatrix} X \\ X \\ X \\ X \\ X \end{bmatrix}$$

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

gathering

- Shooting
  - We shoot light from a single patch to update the rest of the environment
  - Initially: B=E (if patch is emitting)
  - One step updates all patches with unshot energy from one patch (can be sorted: shoot high energy first)
  - This has to be repeated iteratively until we converge at a solution

$$B_i = E_i + R_i B_j F_{ij}$$

$$\begin{bmatrix} X \\ X \\ X \\ X \\ X \end{bmatrix} = \begin{bmatrix} X \\ X \\ X \\ X \\ X \end{bmatrix} + \begin{bmatrix} & X & & & \\ & X & & & \\ & X & & & \\ & X & & & \\ & X & & & \end{bmatrix} \begin{bmatrix} \\ X \\ \\ \\ \\ \end{bmatrix}$$

$$B_i = E_i + (R_i F_{ij}) B_j$$

shooting

Neumann Series

- Let's take a look at the radiosity equation in vector-matrix form again
- If we solve it in a slightly different way, we see, that it can be decomposed in a Neumann series
- This would be equivalent to an iterative solution, as every term of the Neuman series (in this case) represents one additional light bounce
  - Using I (the identity matrix) we can compute the emission (E) of every patch
  - Using A, we can compute the first bounce (= direct reflection), I+A gives is the sum of emission and direct reflection
  - Using A^2, gives us the second light bounce, $I+A+A^2$ leads to the sum of emission, 1st and 2nd bounces
  - ...
- This is useful if you are interested in computing only the first n bounces, or if you are interested only in the contribution of bounce i

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

in vector-matrix form:

$$\begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_n \end{bmatrix} + \begin{bmatrix} R_1F_{11} & R_1F_{12} & \dots & R_1F_{1n} \\ R_2F_{21} & R_2F_{22} & \dots & R_2F_{2n} \\ . & . & \dots & . \\ . & . & \dots & . \\ R_nF_{n1} & R_nF_{n2} & \dots & R_nF_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ B_n \end{bmatrix}$$

$$B = E + AB$$

solving for B:

$$B = (I - A)^{-1} E$$

Neumann series:

$$(I - A)^{-1} = I + A + A^2 + A^3 + \dots.$$

Progressive Radiosity

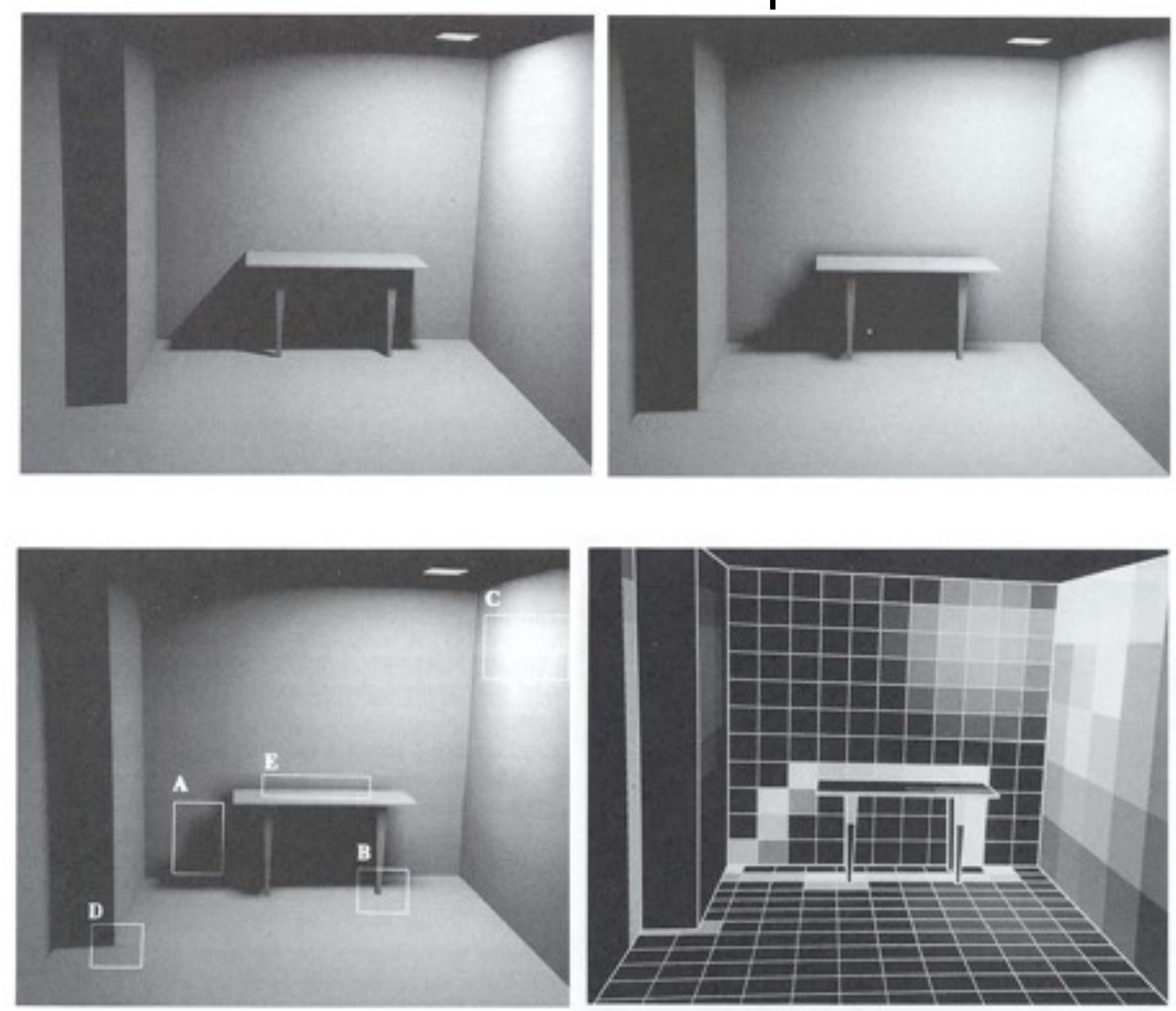
- The advantage of iterative solutions is, that they immediately provide results that are enhanced progressively with each iteration
- We can condition the progression differently
- For instance, we can progressively refine as long as the scene does change, and until our solution converges (i.e., image does not change much)
- With each displayed frame we get an enhanced image
- Thus the frame-rate is inversely proportional to the time required for one iteration

Patch Subdivision

- Subdividing the scene into a uniform patch structure leads to artifacts if patch resolution is too small
  - Especially shadow regions appear blocky and discontinuous

reference image

result with low resolution uniform patch mesh



Error Image

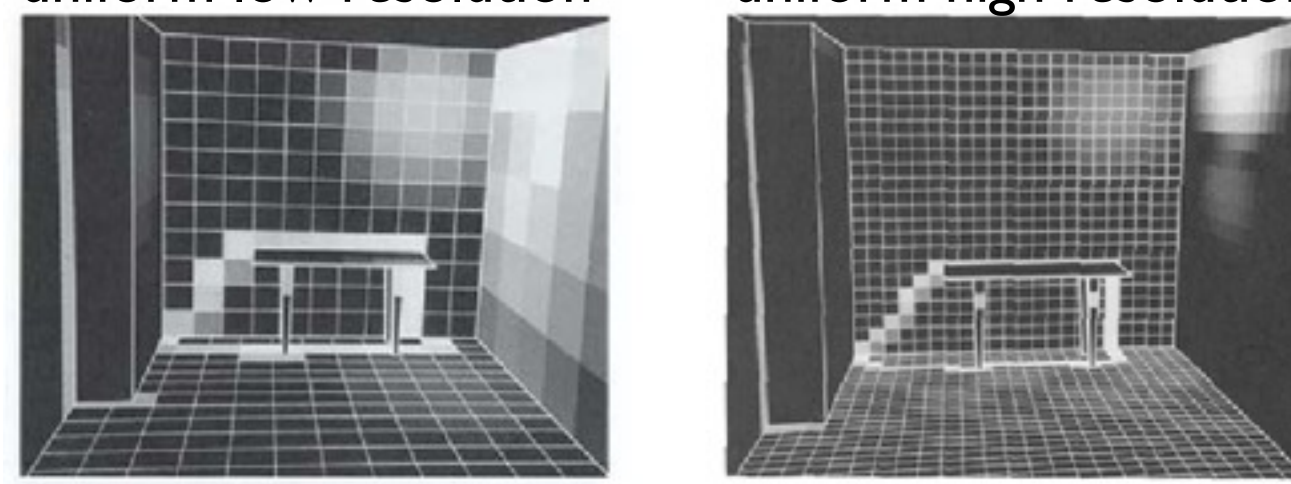
A. Blocky shadows
B. Missing features
C. Mach bands
D. Inappropriate shading discontinuities
E. Unresolved discontinuities

Patch Subdivision

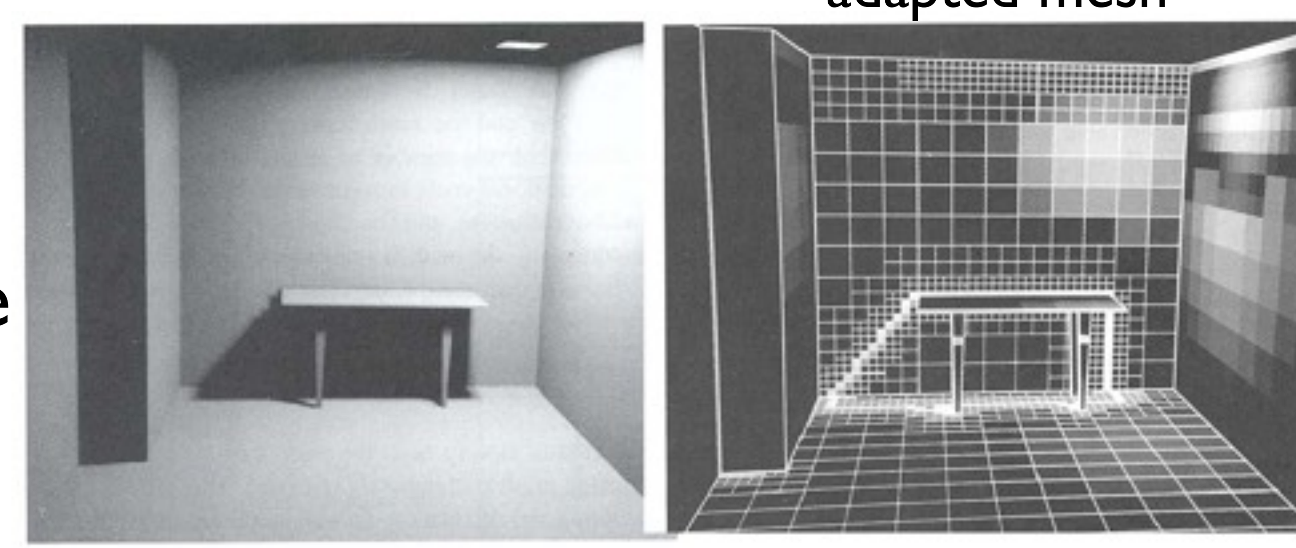
- Subdividing the scene into a uniform patch structure leads to artifacts if patch resolution is too small
  - Especially shadow regions appear blocky and discontinuous
- Off course we can increase the geometric resolution
  - But this will significantly slow down the performance
- A better solution is to locally adapt the patch mesh
  - Increase resolution where necessary, and leave it coarse where possible

uniform low resolution

uniform high resolution

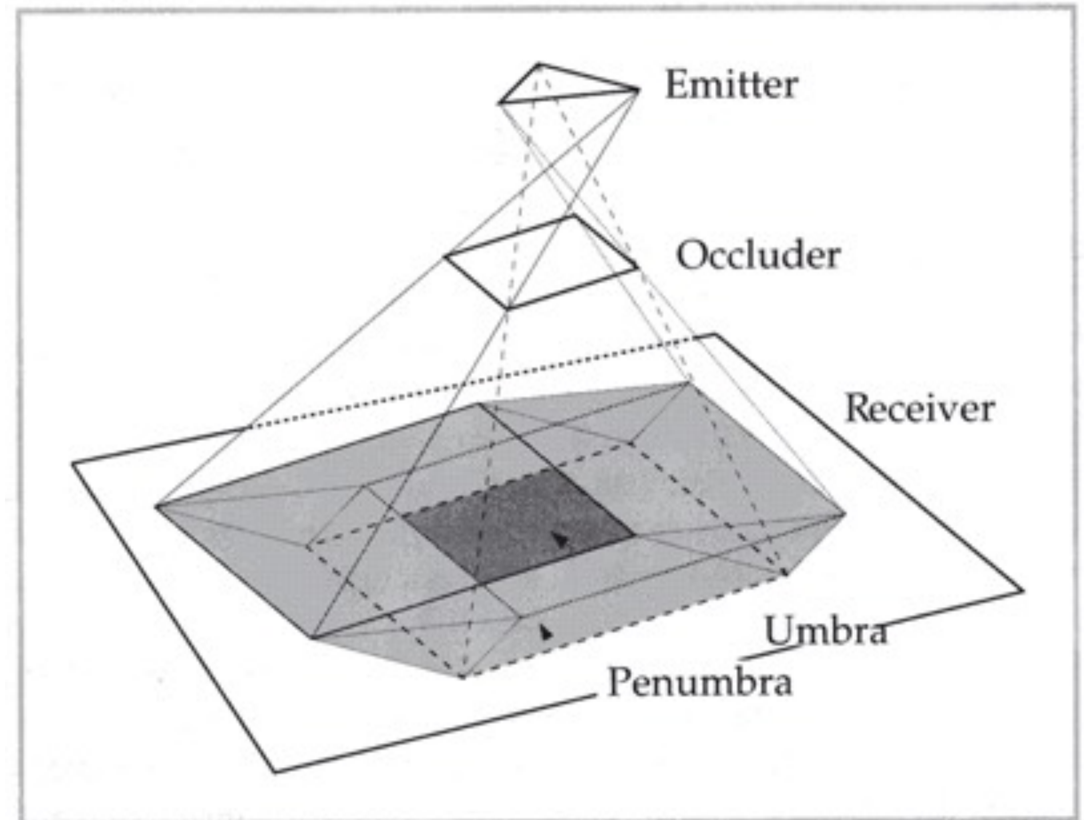


adapted mesh

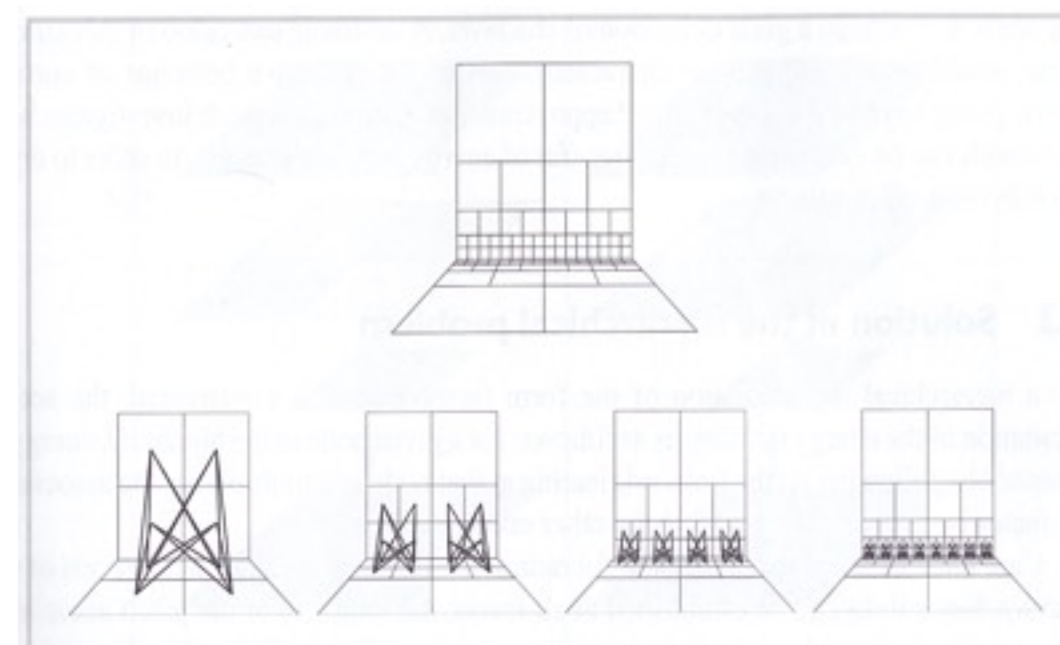


Meshing Strategies

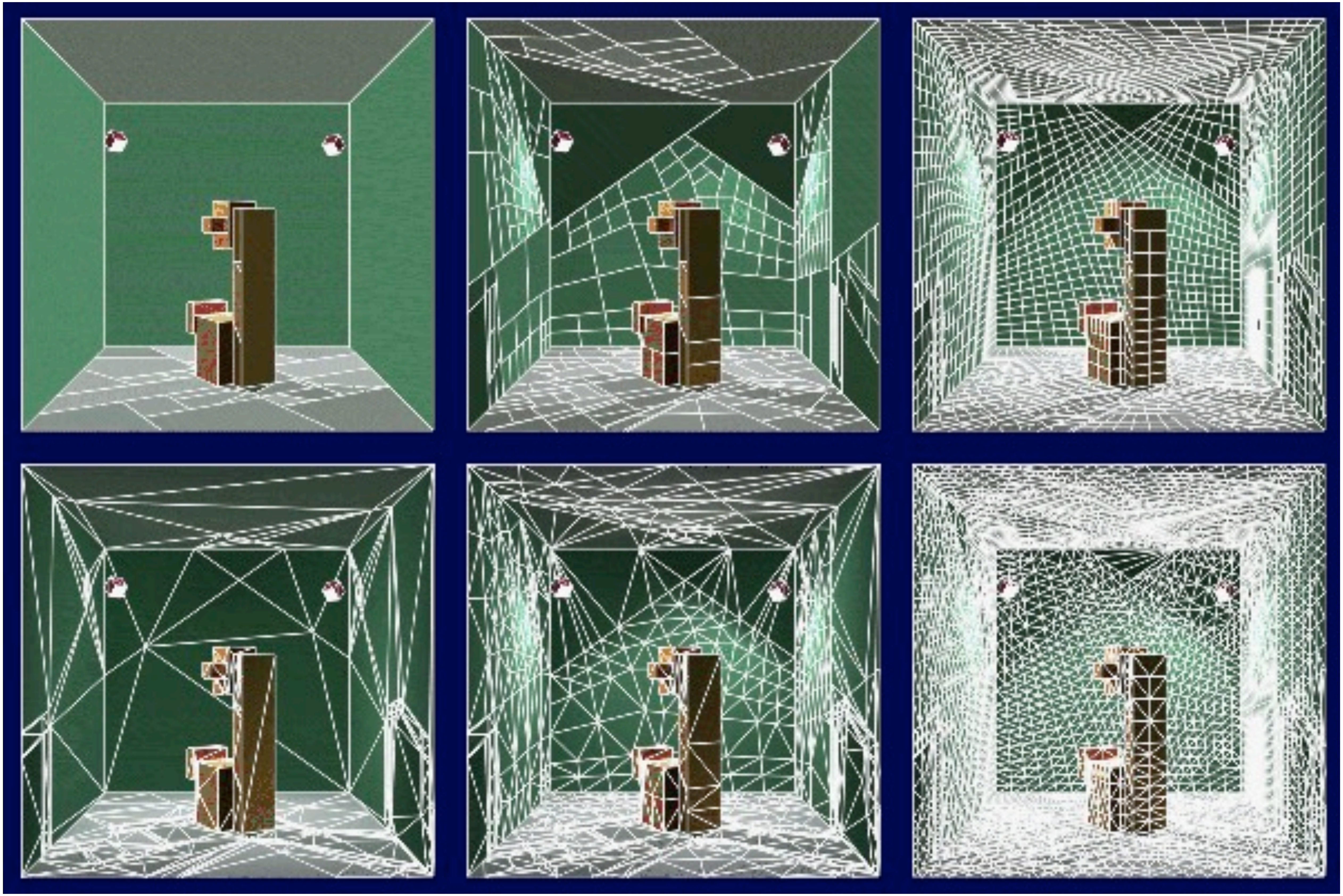
- Discontinuity meshing predicts discontinuities (e.g., in shadow regions) before the radiosity solution is computed
    - The mesh is refined at strong discontinuities and remains coarse at low discontinuities
- Adaptive meshing refines a coarse version as the radiosity solution progresses
    - If the sub-patches of the next higher resolution of a patch have strong discontinuities, then subdivide



discontinuity meshing


hierarchical meshing

Meshing Strategies



Reconstruction

- To summarize radiosity:
    1. Mesh surface into patches (uniformly or **adapted**)
    2. Compute form factors between patches (hemicube or **area sampling**)
    3. Solve linear equation system (direct or **iterative**)
    4. Reconstruct solution (interpolating and blending low resolution radiosity solution with high resolution scene image)
- Note, that radiosity is view independent - it does not have to be recomputed if camera changes (only if scene changes)
- Radiosity on GPU?
    - Sure: realtimeradiosity.com

low resolution radiosity solution

low resolution radiosity solution interpolated and bended with high resolution scene image

Course Schedule

Type	Date	Time	Room	Topic	Comment
C1	01.03.2016	13:45-15:15	HS 18	Introduction and Course Overview	Conference
C2	15.03.2016	13:45-15:15	HS 18	Transformations and Projections	Easter Break
C3	05.04.2016	13:45-15:15	HS 18	Raster Algorithms and Depth Handling	
C4	12.04.2016	13:45-15:15	HS 18	Local Shading and Illumination	
C5	19.04.2016	13:45-15:15	HS 18	Texture Mapping Basics	
C6	26.4.2016	13:45-15:15	HS 18	Advanced Texture Mapping & Graphics Pipelines	
C7	03.05.2016	13:45-15:15	HS 18	Intermediate Exam	
C8	09.05.2016	17:15-18:45	HS 18	Global Illumination I: Raytracing	
C9	10.05.2016	13:45-15:15	HS 18	Global Illumination II: Radiosity	Conference / Holiday
C10	31.05.2016	13:45-15:15	HS 18	Volume Rendering	
C11	07.06.2016	13:45-15:15	HS 18	Scientific Data Visualization	
C12	14.06.2016	13:45-15:15	HS 18	Curves and Surfaces	
C13	21.06.2016	13:45-15:15	HS 18	Basics of Animation	
C14	28.06.2016	13:45-15:15	HS 18	Final Exam	
C15	04.10.2016	13:45-15:15	TBA	Retry Exam	

Thank You!