Computer Graphics: 10 - Radiosity

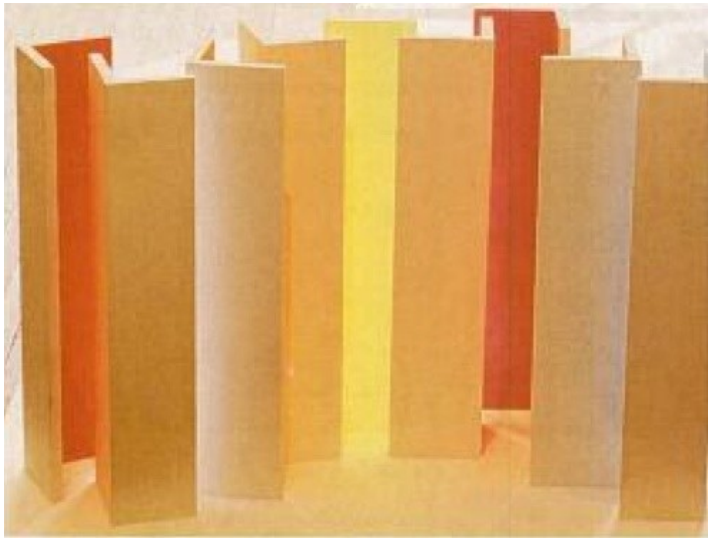
Prof. Dr. Charles A. Wüthrich,
Fakultät Medien, Medieninformatik
Bauhaus-Universität Weimar
caw AT medien.uni-weimar.de

A pretty raytraced picture

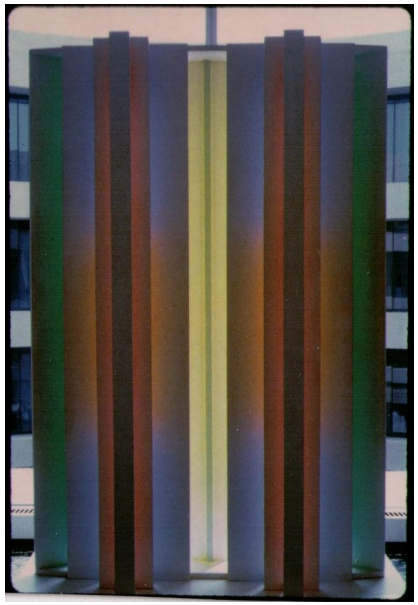
Courtesy Martin Moeck, Siemens Lighting, 1994

The limits of raytracing

- Abstract artist John Ferren and his “Construction in Wood: Daylight Experiment (Façade)” (1968)
- Wooden scultures: front white, back painted different colours
- One of them set up in front of an unshaded window,
  - so that sunlight reflects the fluorescent paint on the back sides of the wood slats
  - onto the white paint on the front sides,
  - tricking your eye into imagining that the light comes from fluorescent bulbs.



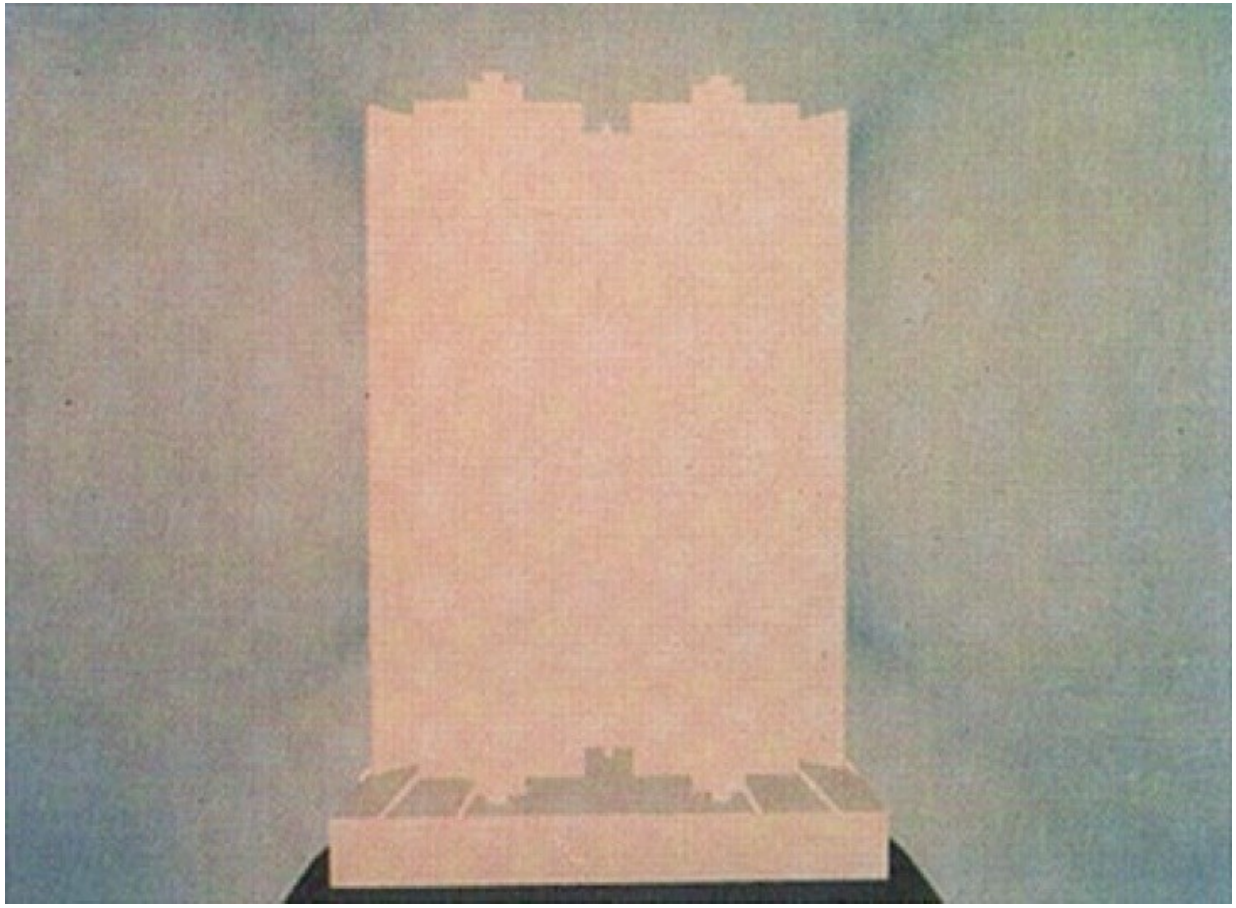
Copyright 1989, The New Yorker



Courtesy, Jerry Scharf

The limits of raytracing

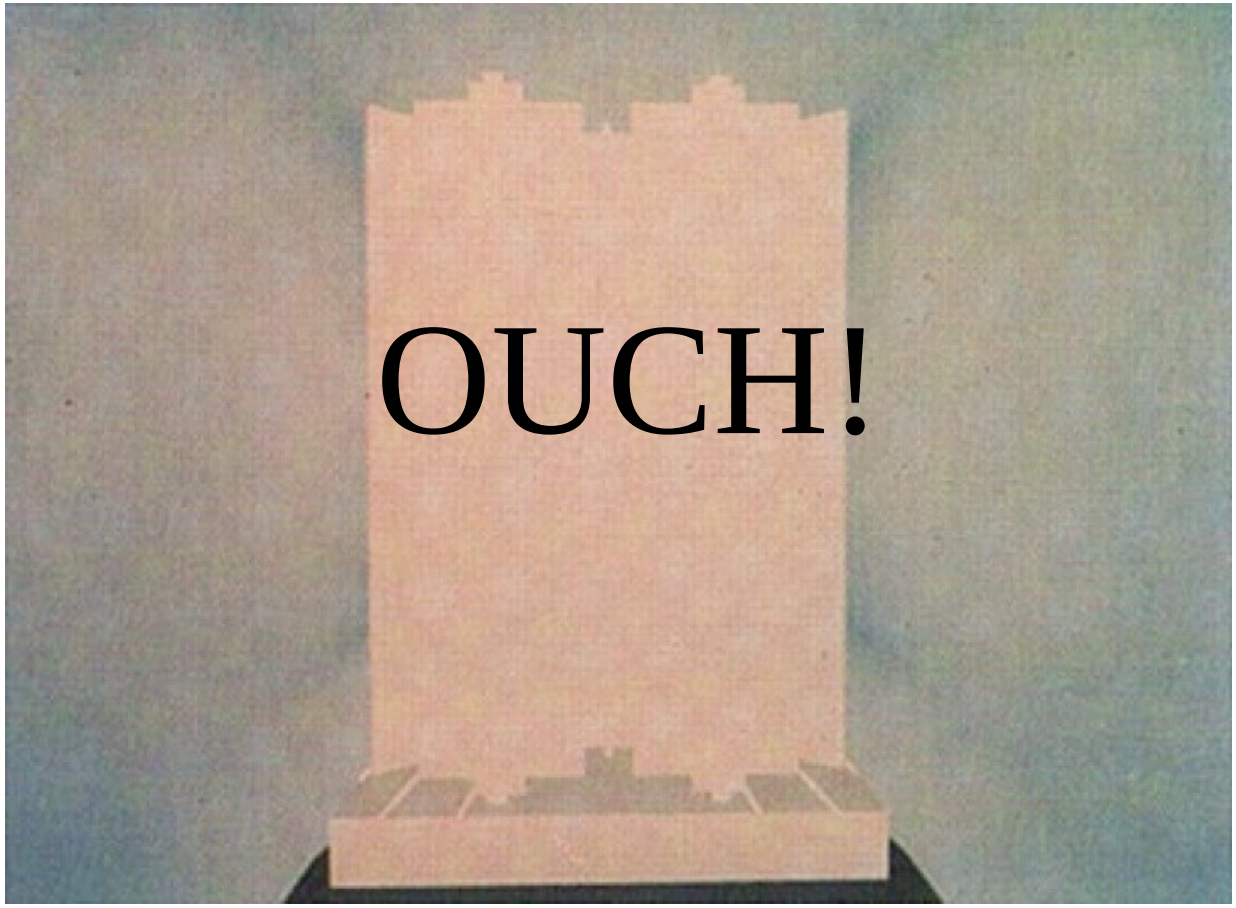
- Raytraced rendering:



Copyright (1993), A. M. Cohen, J. Wallace, Radiosity and Realistic Image Synthesis, Morgan Kaufman

The limits of raytracing

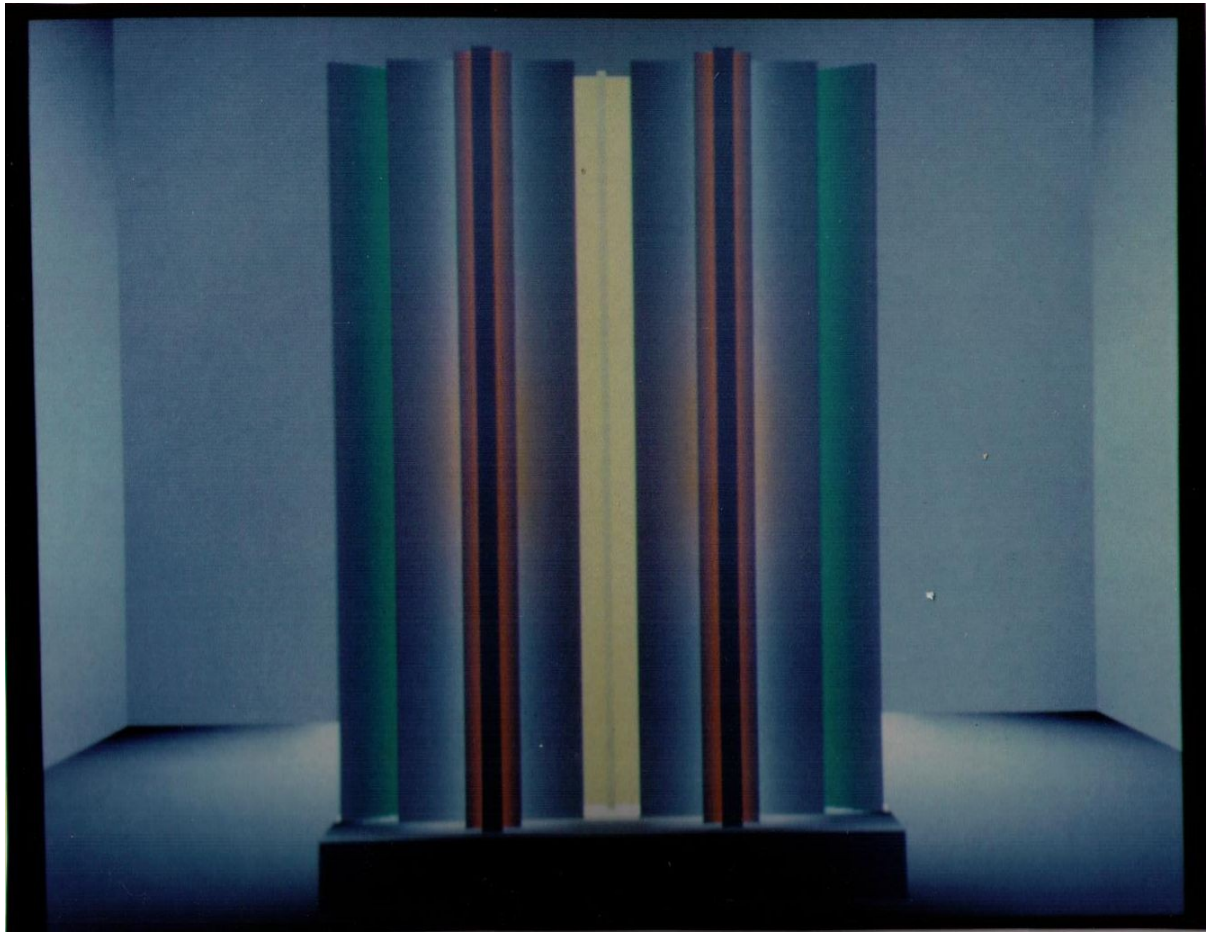
- Raytraced rendering:



Copyright (1993), A. M. Cohen, J. Wallace, Radiosity and Realistic Image Synthesis, Morgan Kaufman

The limits of raytracing

- But we want THIS!!!!



The limits of raytracing

- The sculpture is worst-case scenario for ray tracing:
  - All the surfaces facing the viewer are white
  - Color bleeding is only due to indirect diffuse-diffuse interreflection
  - Classical ray tracing paints surfaces black:
    - no light percolates from the windows to any of the surfaces.
    - Thus they get the white color of the ambient light component

Courtesy, Jerry Scharf

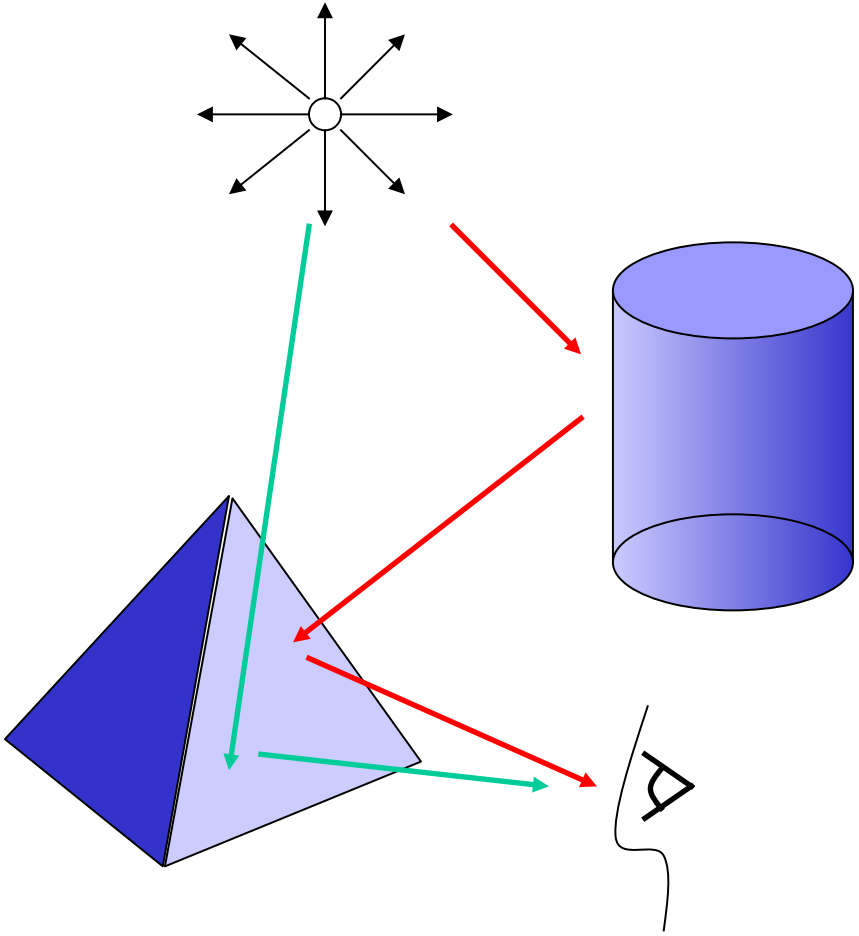
The limits of raytracing

- Geometrical optics is not enough for simulating light reflections
- Diffuse reflection needs a new model
- But how?

Courtesy, Jerry Scharf

Global Illumination Models

- Let us go back to the definition of global illumination models:
- Light reflected by a surface is dependent
  - on the surface itself,
  - the direct light sources, and
  - light which is reflected by the other surfaces on the environment towards the current surface (Reflections)
- Plus emitted light from light surfaces
- Kajiya introduced an equation describing this:



Kajiyas Rendering Equation

$$I(x,x') = g(x,x')\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

- James T. Kajiya, Siggraph '86

- *x, x', x''* : Points in the environment
- *I(x,x')* : Light Intensity from x' to x

Kajiyas Rendering Equation

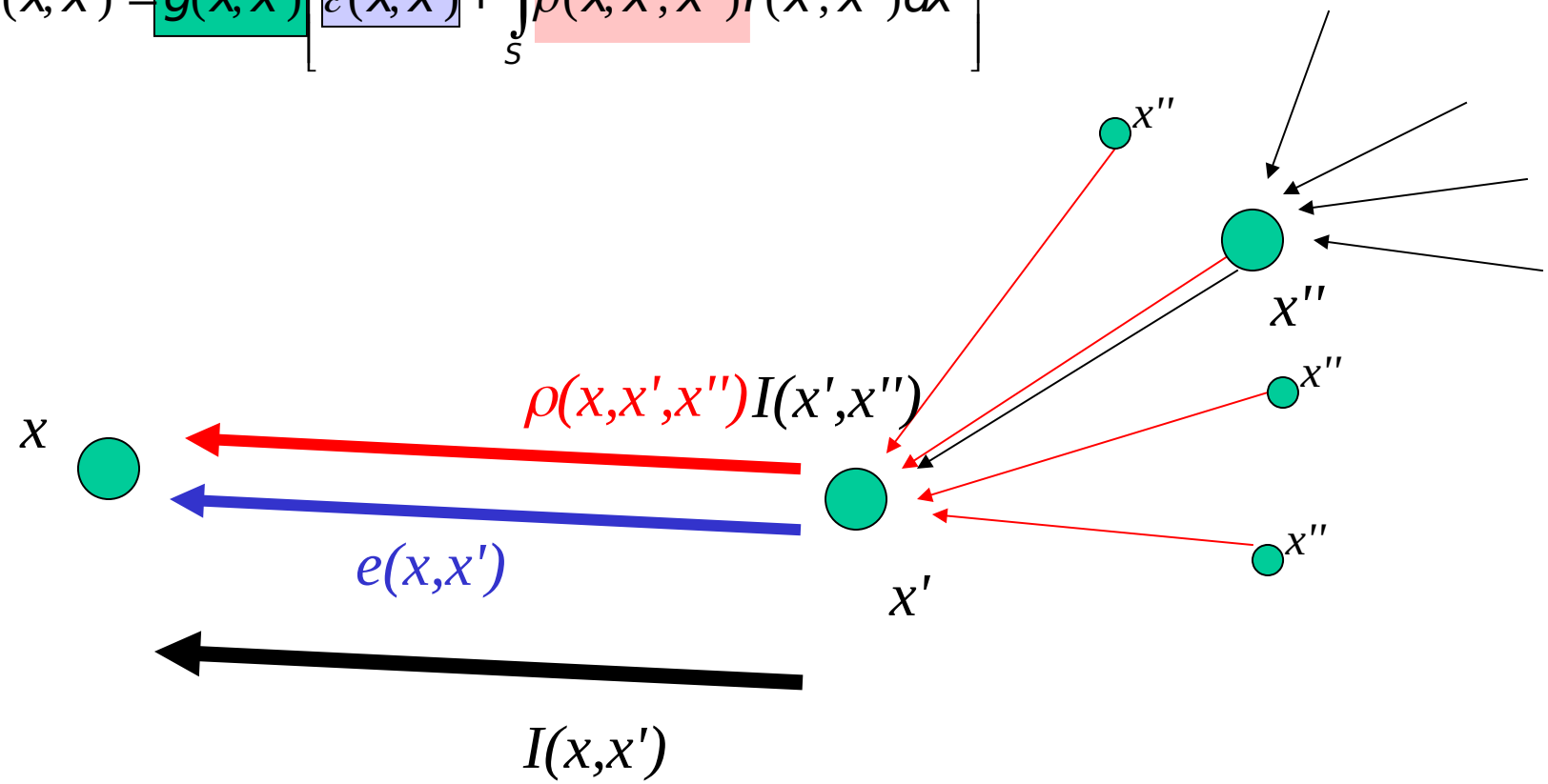
$$I(x,x') = g(x,x')\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')\, I(x',x'')\, dx''\right]$$

Where

- $g(x,x')$: Visibility term (geometry factor)
  - $g(x,x')=0$ if x,x' mutually invisible else $g= 1/d(x,x')^2$
- $\varepsilon(x,x')$: Light emitted directly from x' to x
- $\rho(x,x',x'')$: Reflection coefficient
  - Intensity arriving in x, that has been originated at x'', and reflected through x'
- The integral is made on all surfaces in the environment

Kajiyas Rendering Equation

$$I(x,x') = g(x,x')\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

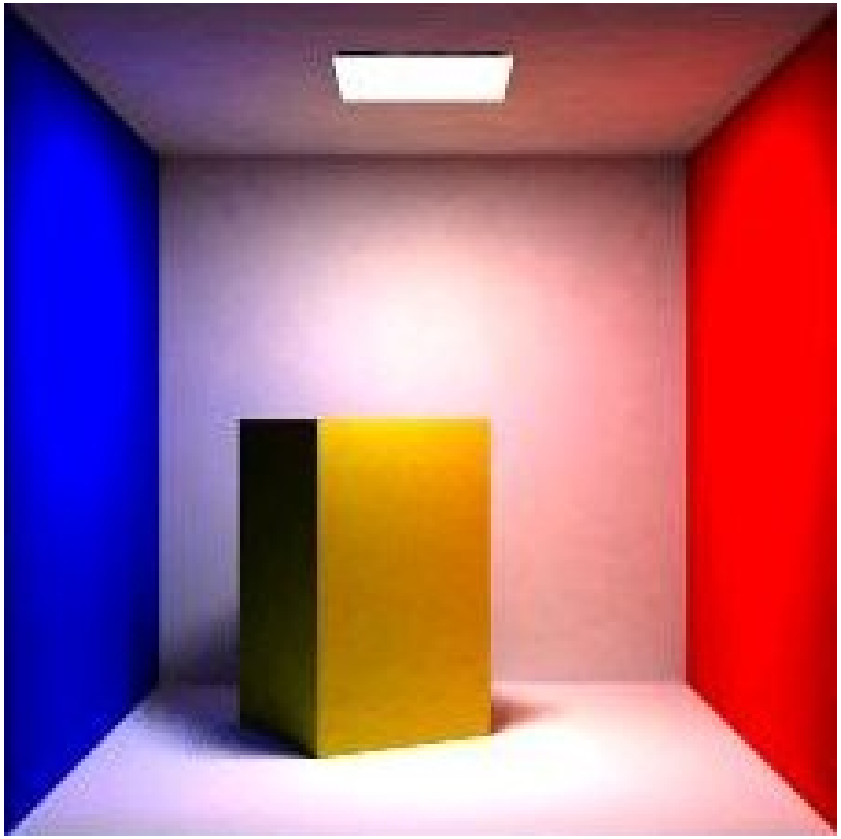


Kajiyas Rendering Equation

- Notes:
  - $g(x,x')*\varepsilon(x,x')$ codes visibility information.
    If x=Viewpoint it is hidden surface computations
  - The rendering equation is computationally very complex, the integral extends to all surfaces in the environment
  - In „partecipating media“, such as foggy environments, the integral is done on all points of the volume considered
  - All Illumination Methods are in some ways solutions to the Kajiya's equation

A step forward in rendering

- The solution for the bleeding problem was developed at Cornell University in 1985
- This is a rendering of the famous Cornell Box, the first experiment made to validate graphics algorithms perceptually.



Reasoning on the rendering equation

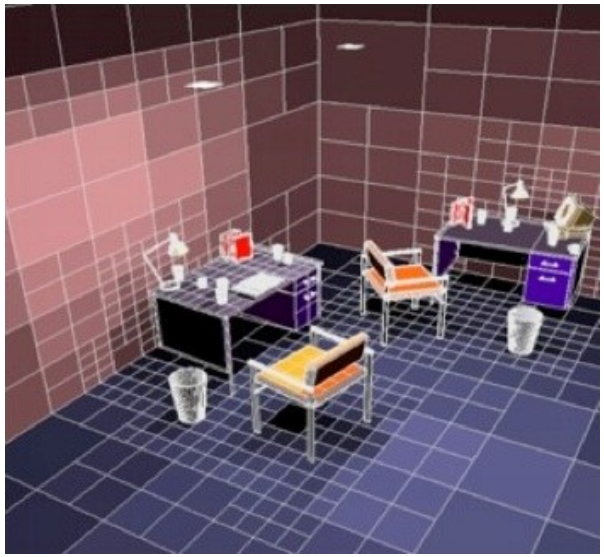
- To simulate this latest effect, one has to resort to Kajiya‘s reendering equation, and try to solve the equation itself
- Remember Kajiya´s equation:

$$I(x,x') = g(x,x')\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

  Posed in these terms, the equation is practically unsolvable
- However, one can use finite elements methods to approximate the equation

Finite elements methods

- Given a complex differential equation system, one method to solve it is through applying a finite elements method.
- By finite element one indicates that the space for which the solution is computed is subdivided in finite elements
- On each of such elements, the solution is approximated as being a simple function, such as a constant or linear function
- In the case of a radiosity method, the 3D space surfaces are subdivided into patches
- After computation, results are interpolated to get smooth result



What is radiosity?

- The radiosity of a surface is the rate at which energy leaves that surface
- It is measured in Energy per unit time per unit area
- Radiosity is built up by two components:
  - Energy emited by the surface
  - Energy reflected from other surfaces

- Originally, the method stems from modeling heat transfer between surfaces in a closed environment
- The same techniques can be used to compute the transfer of radiant energy between surfaces for Computer Graphics
- One can compute the intensity of radiant energy arriving at a surface
- This incoming energy is used to compute shading of the surface

The radiosity equation pro patch

- Let the 3D world be constituted by N patches
- The radiosity OUTGOING from a patch *i* can be expressed by the following equation

$$B_i = \underbrace{E_i}_{\text{Emitted energy by patch i}} + \underbrace{\rho_i \underbrace{\sum_{\substack{j=1,\ldots,N \\ j \neq i}} B_j F_{ij}}_{\text{Energy reaching patch i}}}_{\text{Reflected energy leaving patch i}}$$

- Where
  - B_i: radiosity of surface patch *i*
  - E_i: Emitted radiosity from patch *i*
  - ρ_i: Reflectivity of patch i
  - B_j: radiosity of surface patch *j*
  - F_{ij}: Form factor of patch *j* relative to patch *i*

Form factors

- A form factor F_{ij} is the fraction of energy that leaves the surface F_j and reaches F_i
- As such it is dimension-less, and a pure number
- It does NOT include surface information, nor energy
- The form factor is not easy to compute: in general it involves integral computation

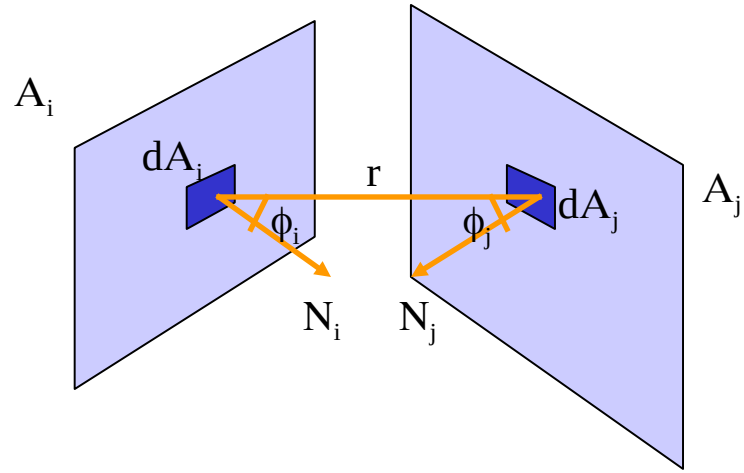
Form factors

- Suppose we have two surfaces A_i and A_j in 3D space.
- Take two infinitesimal elements on them, dA_i and DA_j
- Then $$F_{dA_i dA_j} = \frac{\cos\phi_i \cos\phi_j}{\pi |r^2|}$$

  where
  - r: ray length between dA_i & dA_j
  - ϕ_i: Angle between normal to dA_i and r
  - ϕ_j: Angle between normal to dA_j and r

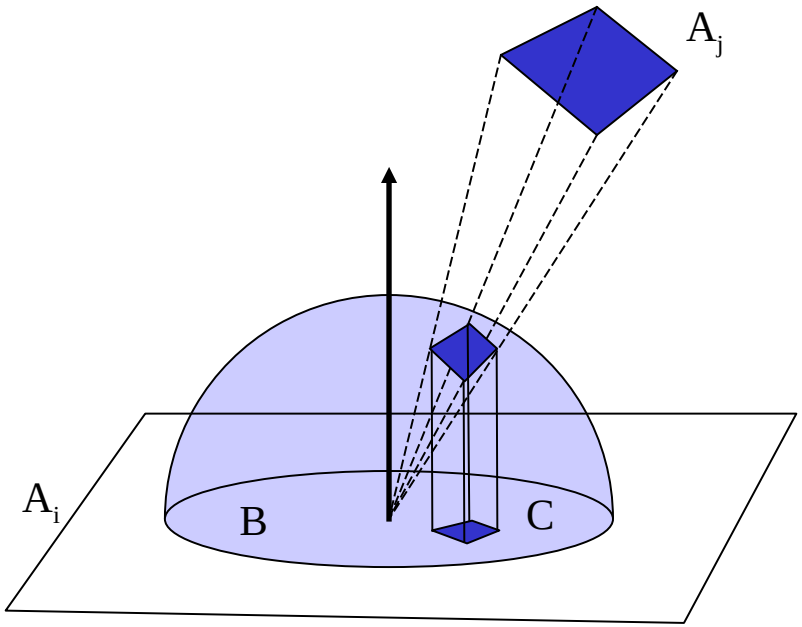
- Integrating this on the surfaces A_i and A_j one obtains the form factor between A_i and A_j:

$$F_{A_i A_j} = F_{ij} = \frac{1}{A_i} \iint_{A_i A_j} \frac{\cos\phi_i \cos\phi_j}{\pi |r^2|} dA_i dA_j$$



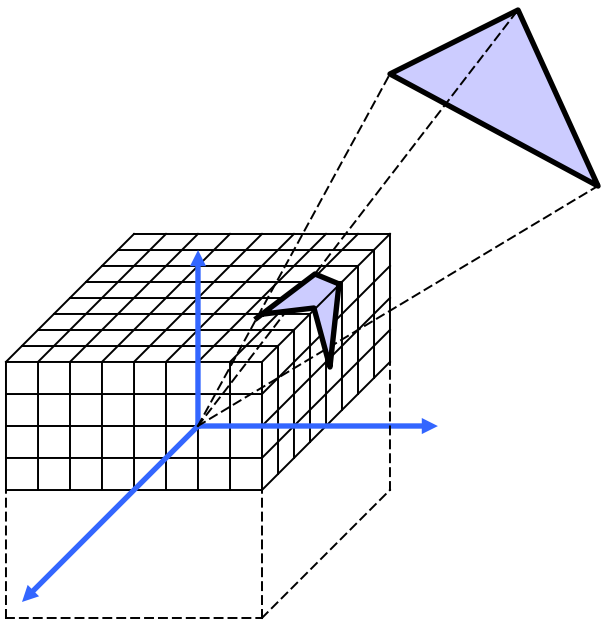
The Nusselt analog

- Computing form factors directly is not easy, already for simple surfaces the equations become complicated
- Nusselt proposed an analog to differential form factors
- The surface A_j is projected on a unit hemisphere centred at A_i
- The projected element is then projected again on the base
- The form factor equals to the projected area divided by the area of the base (C/B)



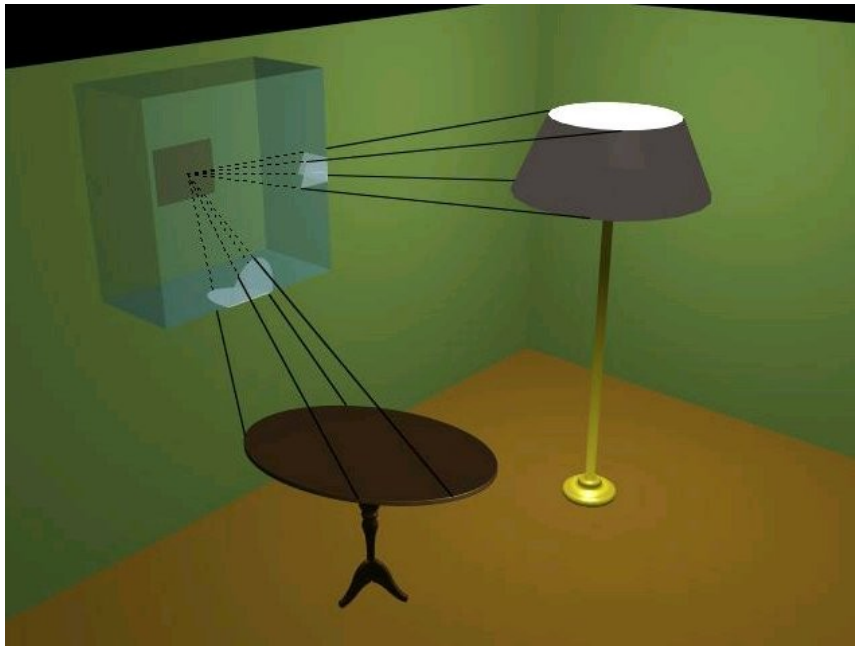
The Hemicube approximation

- To make such computations feasible the hemicube method is used
- A unit hemicube is laid around the surface A_i
- For each cell of the surface of the hemicube, its contribution to the form factor is precomputed (these are called *delta form factors* or DFF)
- The surface A_j is projected to the hemicube
- DFFs of the cells covered are summed to obtain the FF approximation
- Notice that the projections can be done in hardware



Hemicube approximation

- A typical radiosity algorithm projects all other surfaces to the hemicube of a surface, marking hemicube cells with the label of the closest surface
- From the marked list of the cells, and their corresponding contributions, one can compute the FF of the different surfaces with that surface
- Large surfaces are subdivided into smaller pieces to improve precision of the computations



Courtesy Scott Owen, Siggraph

The radiosity matrix system

- Remember the equation for the radiosity of a patch:

$$B_i = E_i + \rho_i \sum_{\substack{j=1,\ldots,N \\ j \neq i}} B_j F_{ij}$$

- In a closed 3D environment, the radiosity equation for each patch form a system of N equations (one for each patch) in the unknown N radiosities B_i
  - Note that the Bi appearing in the equation for each patch are also N
- Such equations have to be solved simultaneously to compute the energy equilibrium in the environment
- This leads to a huge linear system, which can be solved through normal numerical methods

The radiosity matrix system

- The system then looks like this:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_1 F_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

- Where:
  - B_i: radiosity of surface patch *i*
  - E_i: Emitted radiosity from patch *i*
  - ρ_i: Reflectivity of patch i
  - F_{ij}: Form factor from patch i to patch *j*
- The system is solved for B values, which can be used as intensities (or color values) at each surface

The radiosity matrix system

- Note that the size of the linear system is determined by the number of patches in the scene (often millions of patches)
- The linear equation can be solved by using trad. linear system solution numerical methods (such as Gauss/Seidel)
- Such methods are iterative methods, and compute solutions from iteration to iteration, each one of which is closer to the exact solution
- Pure numerical solutions have however some problems:
  - the approximate solutions are optimized for convergence, not for viewing
  - one has to compute ALL N^2 form factors at the beginning
- There is a way to make the solution converge in a „visually appealing“ way, so that the process can be interrupted if the image is „badly taken“

Progressive radiosity

- Yelds intermediate results at lower computation and storage costs
- Converges too to the exact solution BUT it can be halted when the desirable approximation is reached
- At each step, the method computes the form factors between ONE surface and all other surfaces (N-complexity)
- Then the next radiosity solution is computed
- This allows to „shoot energy from ONE surface in the environment“ per step

```
FOREACH iteration
  select a surface i
  Calculate Fij for each surface j
  FOREACH surface j
    update radiosity of surface j
    update emission of surface j
  set emission of surface i to 0
```

- Each step equals to „shooting“ energy“ from a single surface, and storing it in the environment patches as new emitted and reflected energy
- Various strategies exist on which patch to choose to shoot its energy from

Progressive radiosity

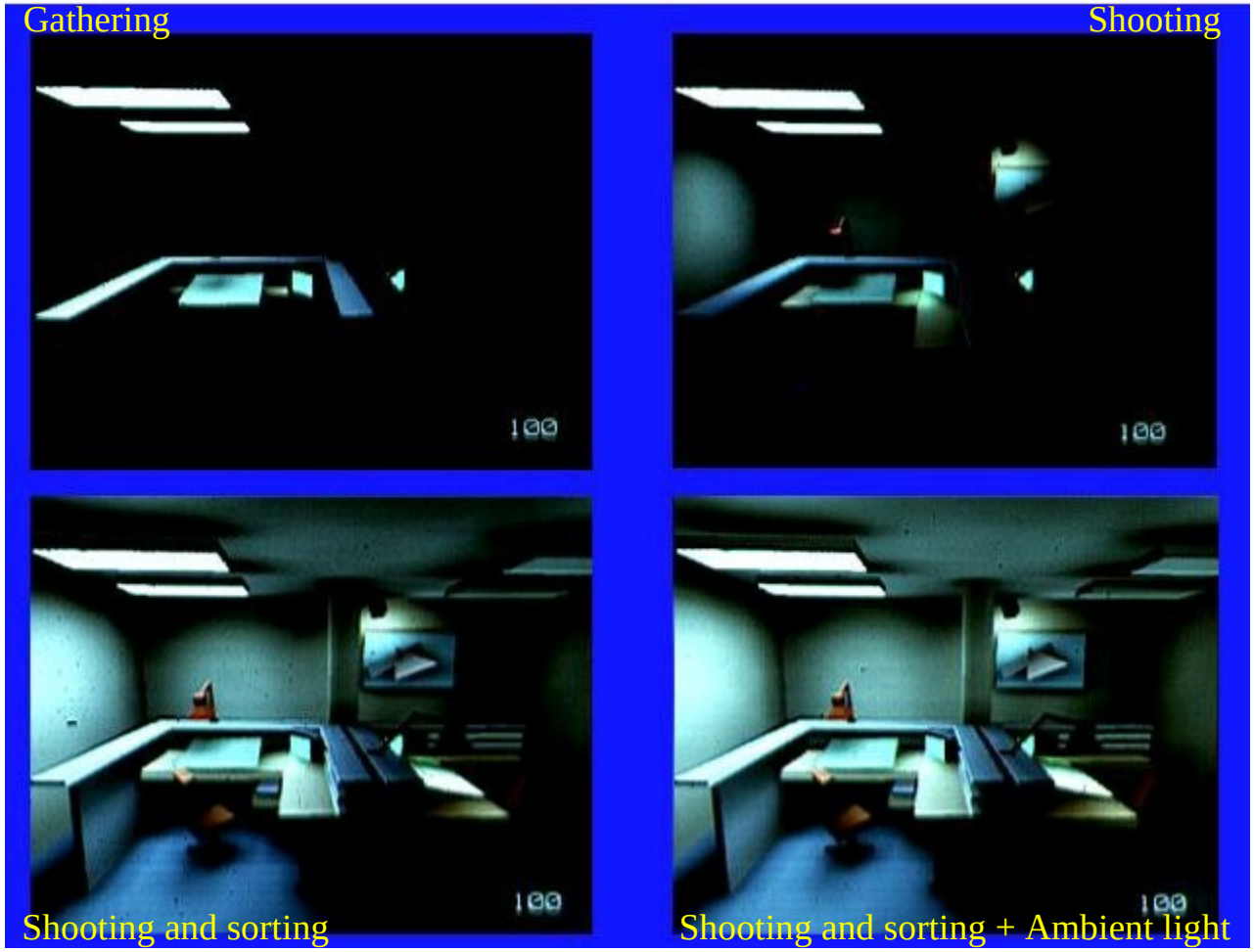
- As the number of iterations increases, the resulting image gets better
- Light spreads slowly in the environment
- Look at how the colour of the walls contributes to the room colour

Variations of progressive radiosity

- Gathering:
  - Choose „base“ surf. arbitrarily
  - Collect light energy from all other surf. in envir., attenuated by the calculated form factors, and update the "base" surface
- Shooting:
  - Distribute light energy from the "base" surface to all other surfaces in the envir., attenuated by the calculated form factors.
- Shooting and sorting:
  - first calculate surface with the greatest amount of unshot light energy
  - then use this surf. as the "base" surface in the "shooting" variant.

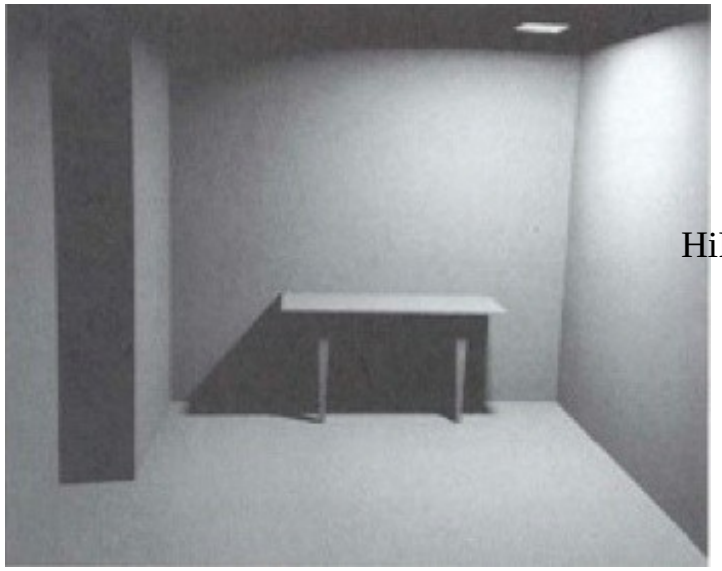
- In addition, an initial "ambient" term can be approximated for the environment and adjusted at each iteration, gradually replaced by the true ambient contribution to the rendered image.
- The "shooting and sorting" method is the most desirable, as it finds the surface with the greatest potential contribution to the intensity solution and updates all other surfaces in the environment with its energy.

Variations of progressive radiosity



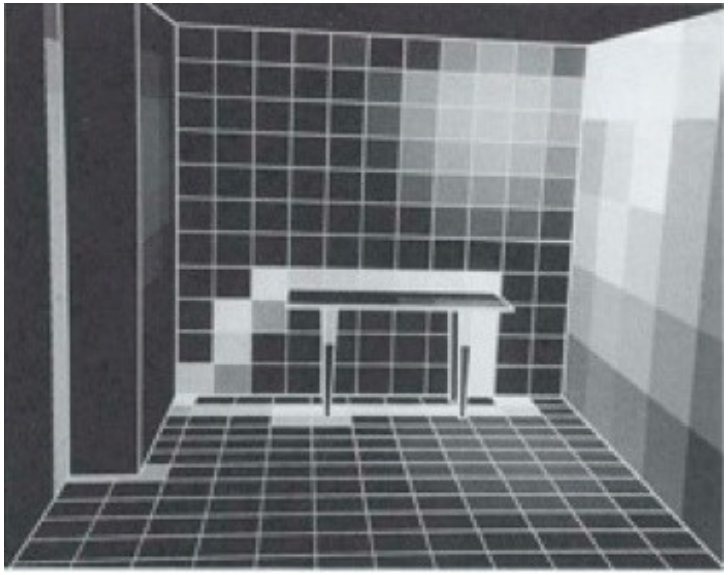
Mesh refinement

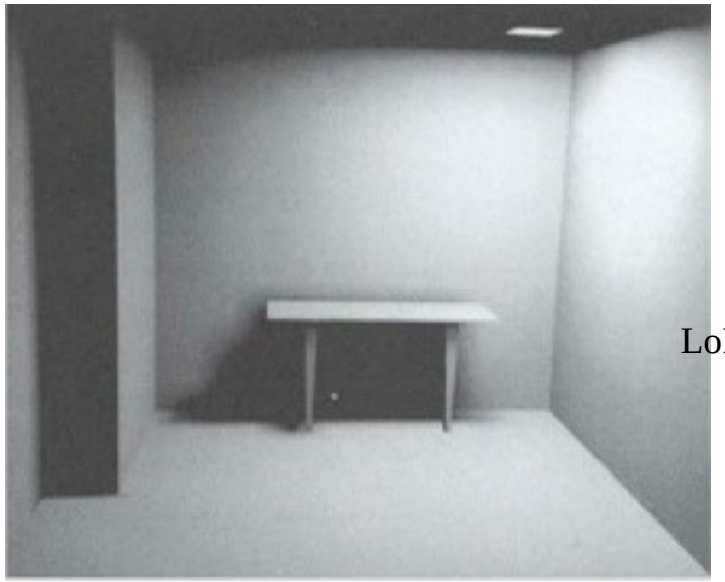
- A uniform patch subdivision leads to artifacts
  - A single value is computed for a patch ⇒ visible patch structure
  - Blocky shadows, missing features, mach bands, shading discontinuities, discontinuities



HiRes

Difference

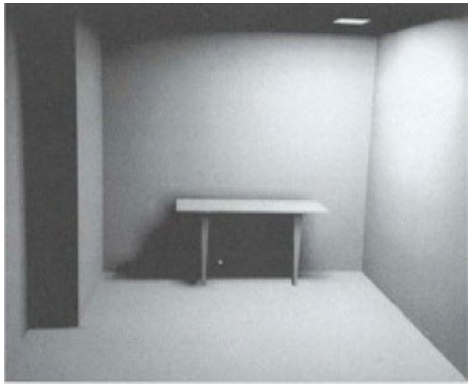




LoRes

Mesh refinement

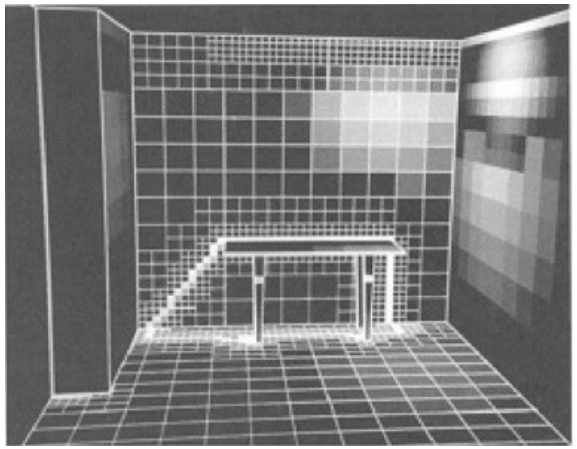
- Increasing resolution helps only partially:
  - It increases also complexity where not necessary
- Solution here: adapt the mesh

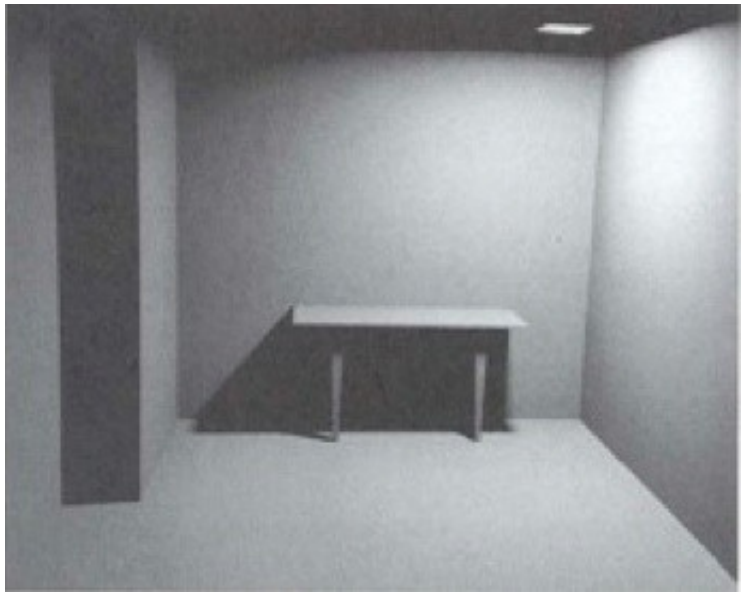


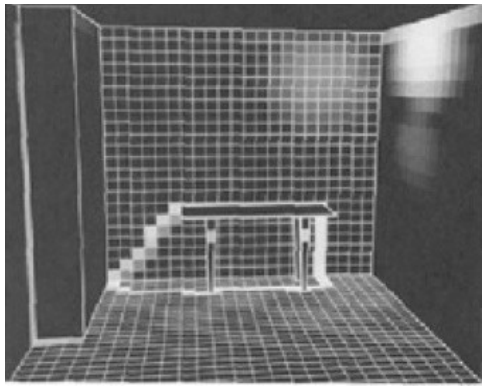
LoRes

HiRes

Adaptive patches

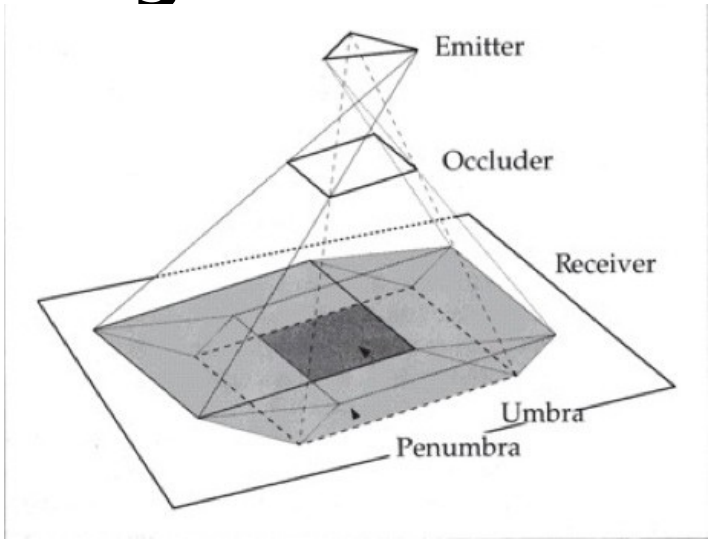




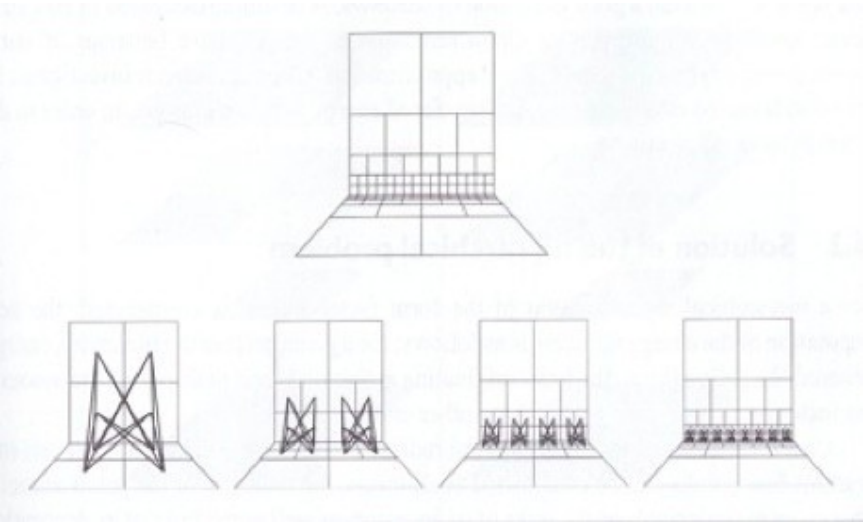


Refinement strategies

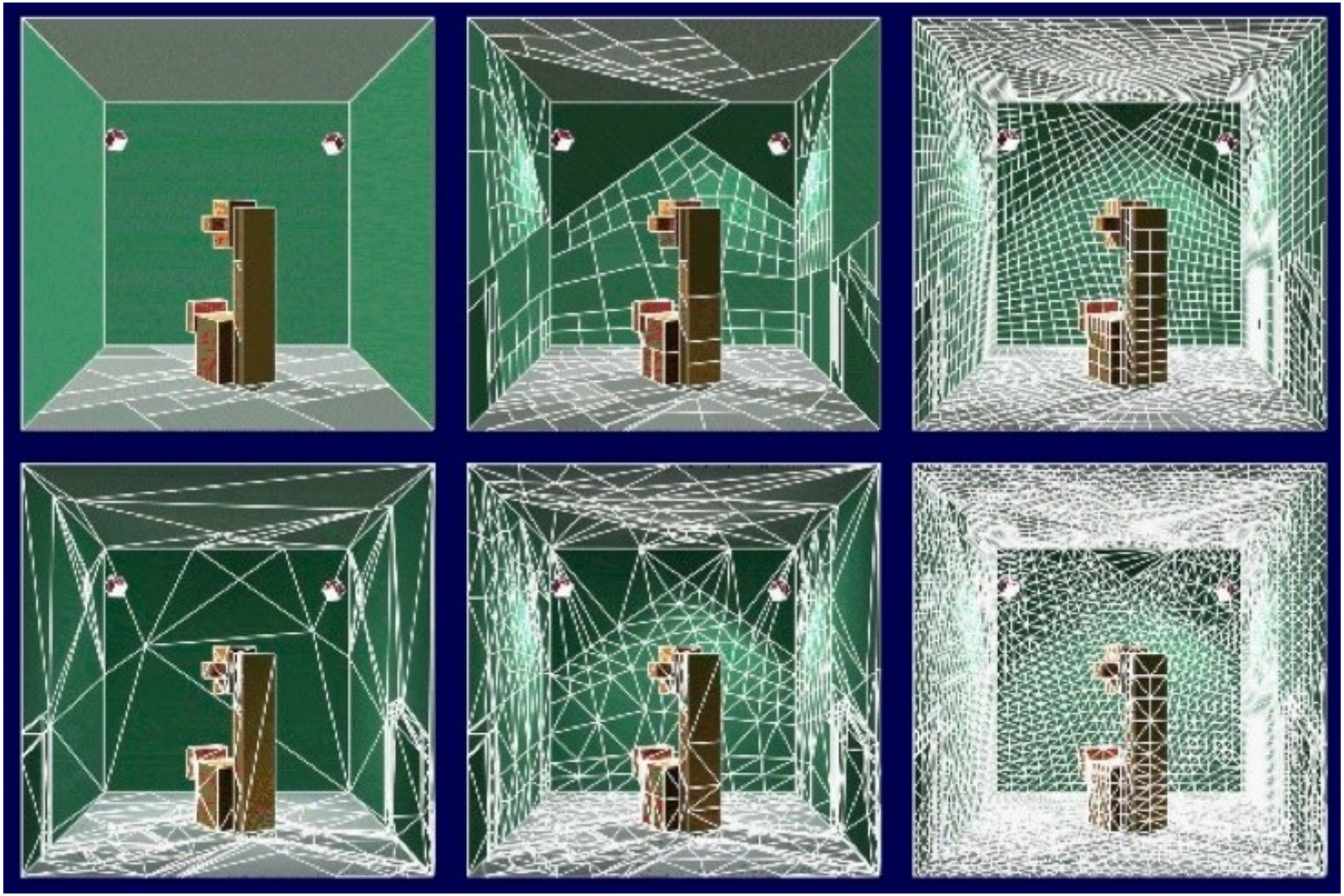
- A priori discontinuity prediction
  - e.g. along shadows
  - Optimal shadow patch
- Adaptive meshing
  - Refine if neighboring patches have very different radiosity values



- Adaptive meshing
  - Refine mesh while solution is being computed
  - Hierarchical computations



Refinement strategies



Two pass methods

- Radiosity computations very expensive
- Usually, radiosity is used at low resolution:
  - Compute low-res radiosity
  - Add to hi-res image by interpolating and blending (*two pass rendering*)
    - Local illumination
    - raytracing
- Radiosity is view independent
  ⇒ compute only once

Partecipating media

- Smoke, dust or vapour can influence the ditribution of light in 3D space by scattering it and partially absorbing it
- To simulate this, light is sent through a 3D volume representing the partecipating medium which
  - Attenuates the light
  - Adds to the illumination solution through illumination of the partecipating medium
- Form factors computations have to be extended to include partecipating volumes („zonal method“)

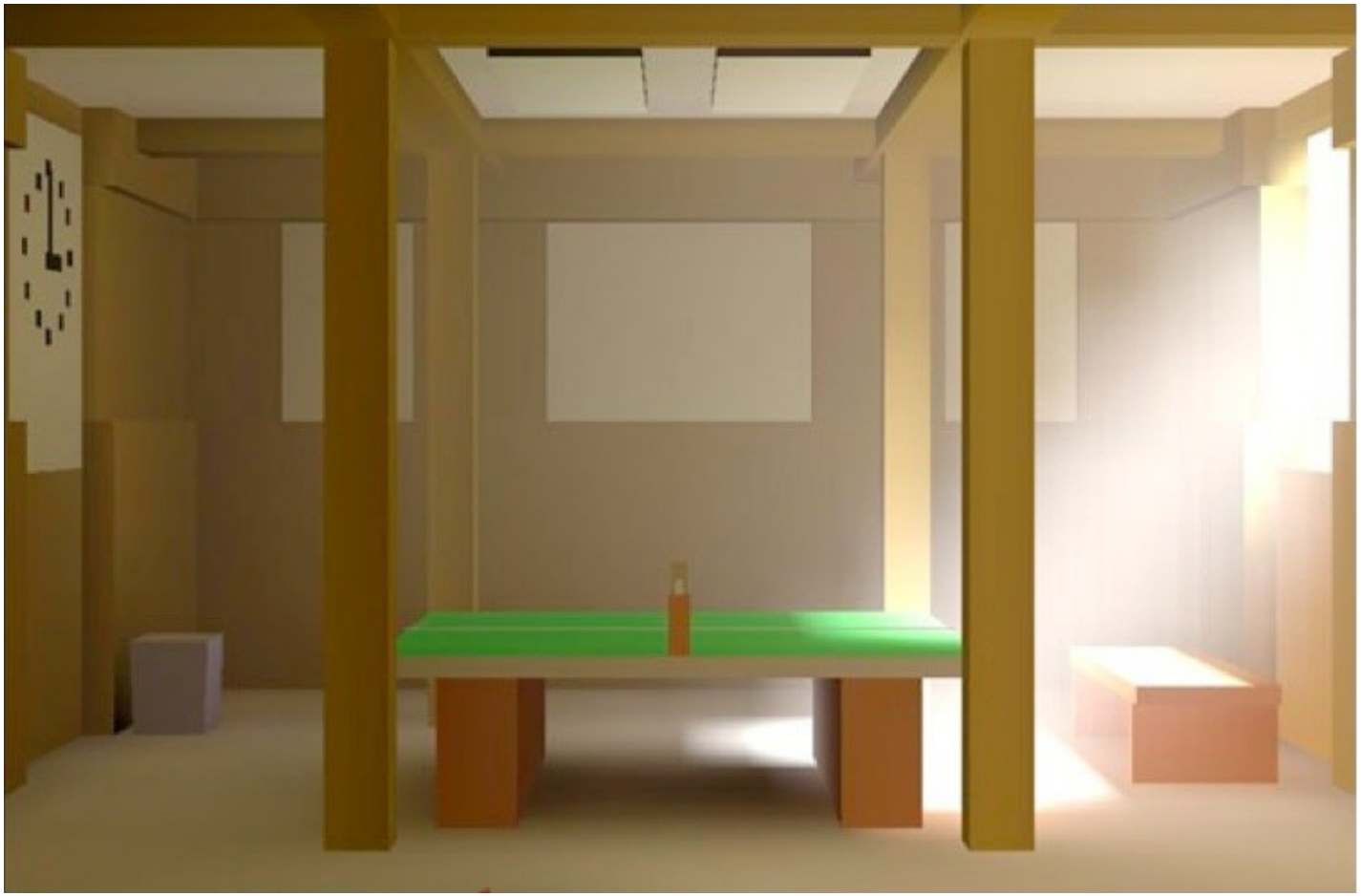
Copyright Holly Rushmeier 1987, Cornell University Program of Computer Graphics

Summary

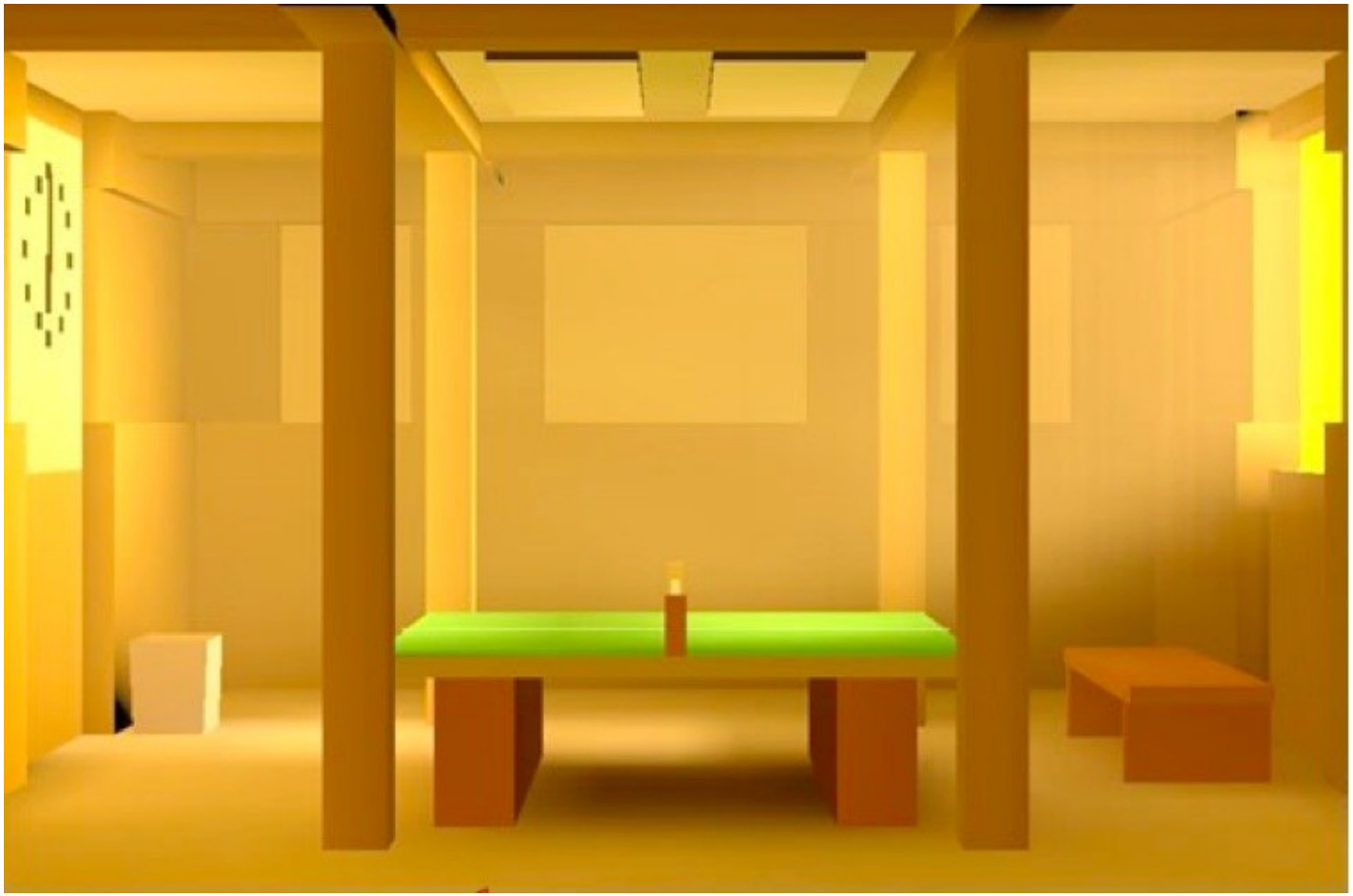
- Advantages of radiosity methods:
  - Photorealistic image quality
  - Accurate simulation of energy transfer
  - Soft shadows and diffuse interreflections
- Disadvantages
  - High computational and storage costs
  - Environment (model) must be preprocessed to have similar patch shapes
  - Non diffuse reflections have to be extra computed

Picture gallery

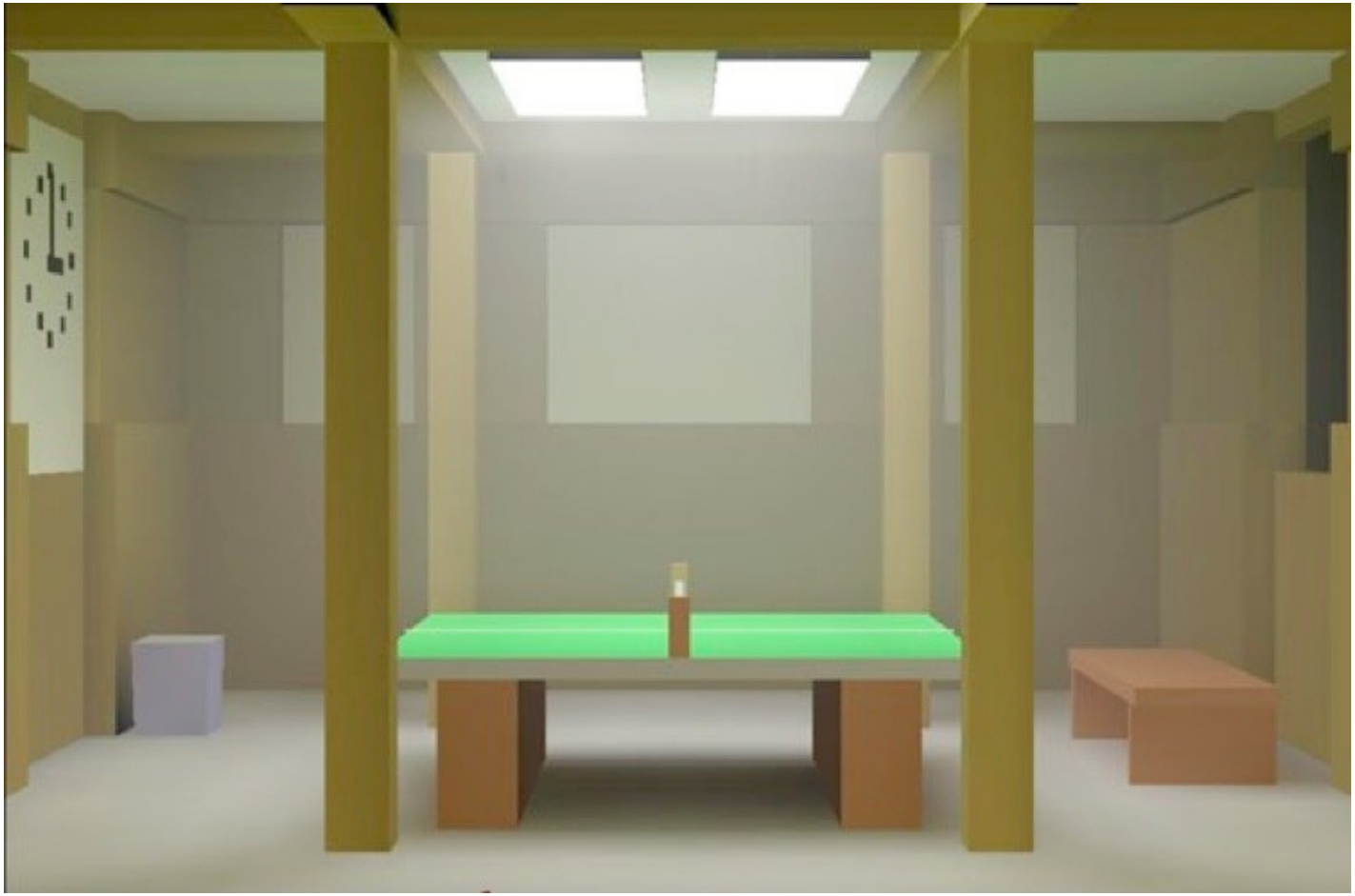
Picture gallery



Picture gallery



Picture gallery



Picture gallery

Copyright Dan Baum, Siggraph

Picture gallery

Copyright Dan Baum, Siggraph

Picture gallery

Picture gallery

Copyright 1991 Eric Haines,
The Hewlett-Packard Company

Picture gallery

Picture gallery

Picture gallery

End

Credits: Pictures partially from Siggraph tutorial slides

```
+++ Ende - The end - Finis - Fin - Fine +++ Ende - The end - Finis - Fin - Fine +++
```