



# Lighting (cont'd)

CS 355: Interactive Graphics and Image Processing

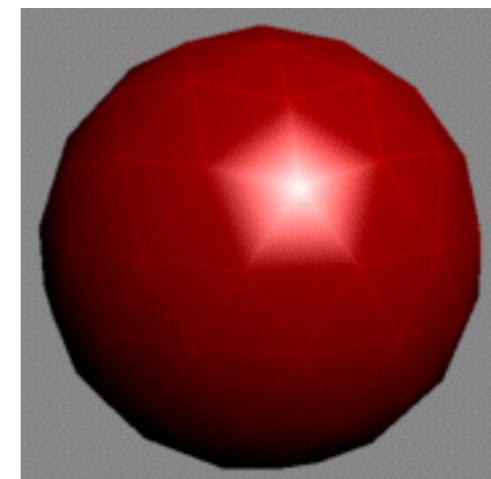
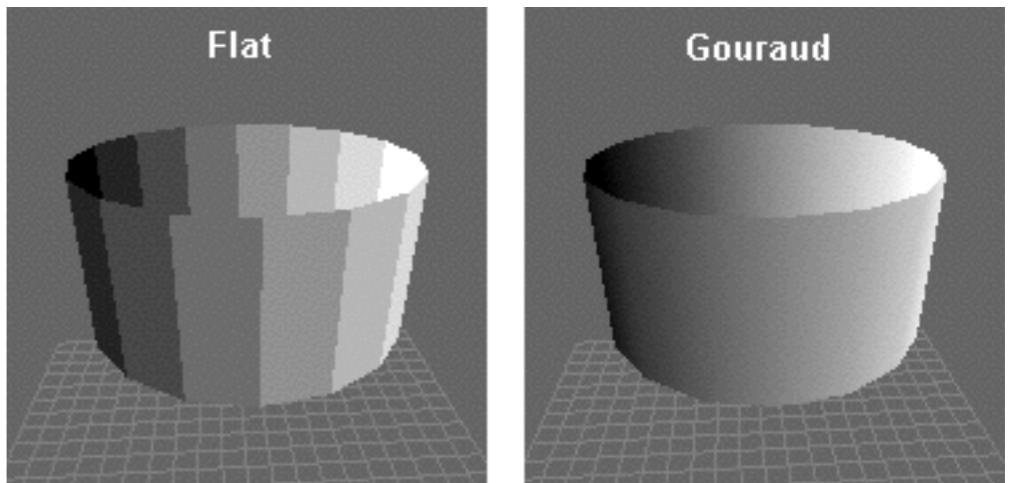
# Flat Shading

- The entire polygon has the same normal, so it's all colored the same
- Leads to “flat shading”



# Gouraud Shading

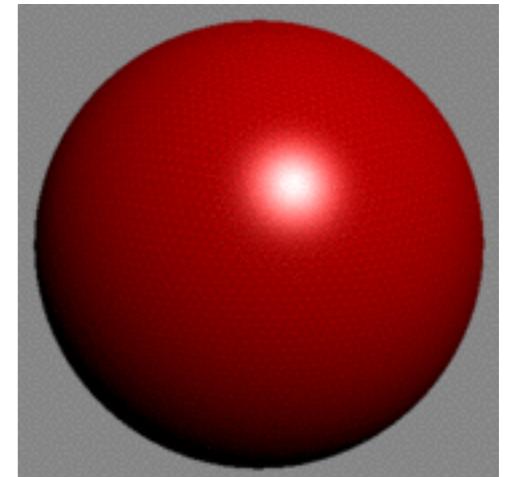
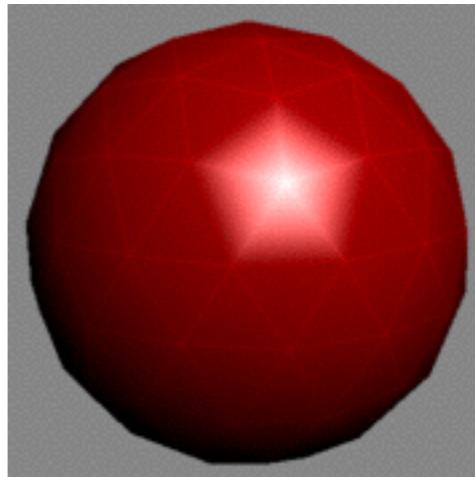
- Simple idea:
  - Compute shading at edges/vertices and interpolate
  - Works well for diffuse
  - Doesn't work as well for specular



Henri Gouraud, 1971

# Phong Shading

- Key idea:
  - Instead of interpolating the shading, *interpolate the normals*
  - Compute shading on a per-pixel basis using interpolated normal



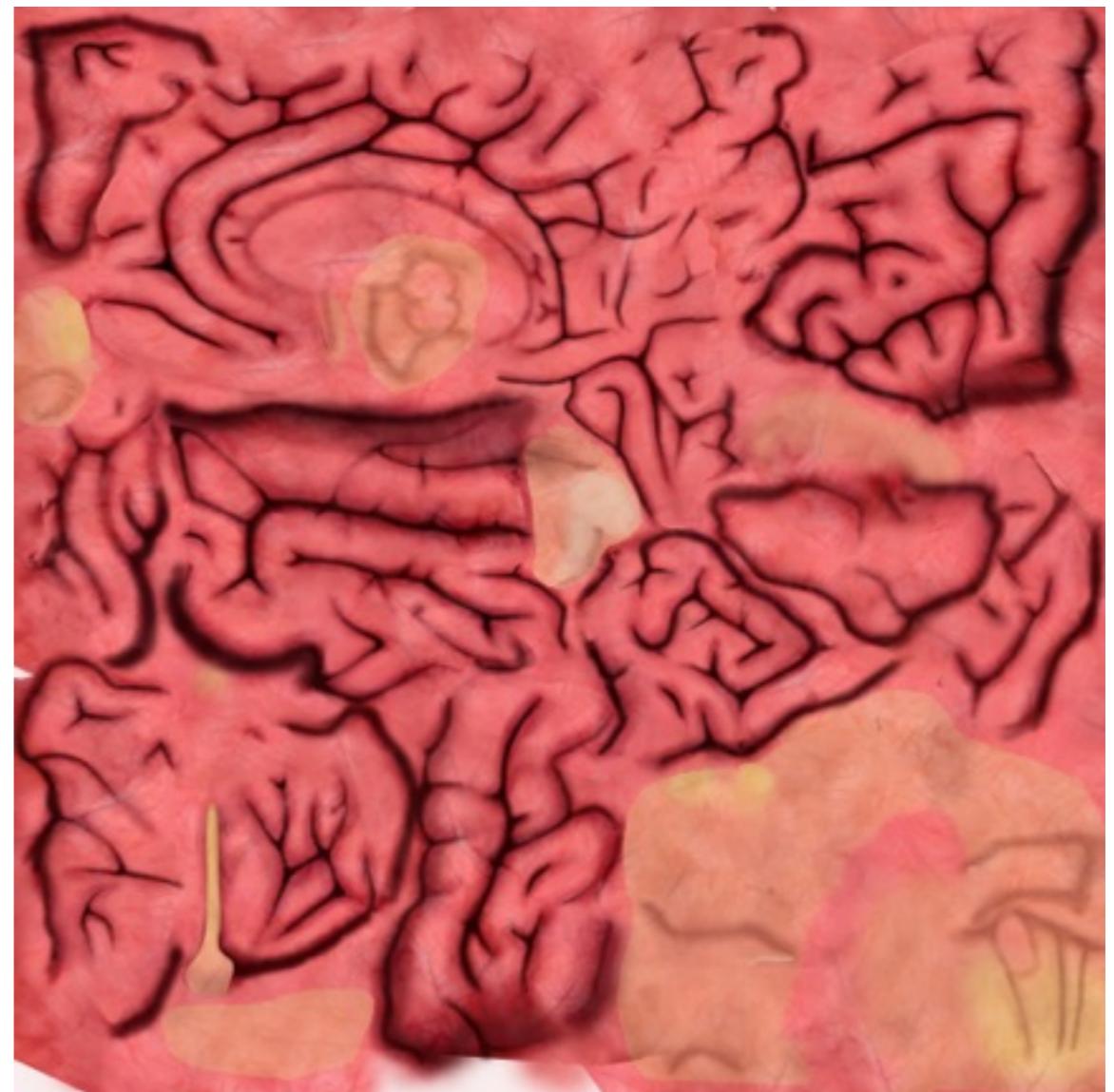
Bui Tan Phong, 1973

# Texture Maps



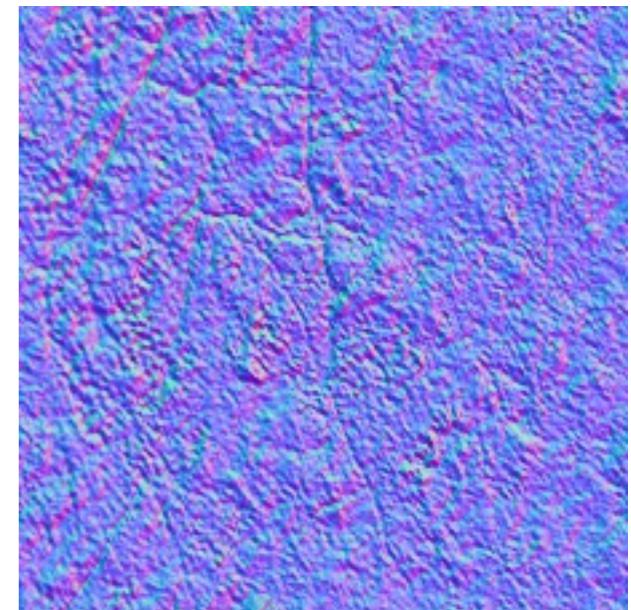
# Texture Maps

- Reflectance across a surface varies with position
- Map each vertex  $(x,y,z)$  to a point  $(u,v)$  in an image
- Interpolate across the polygon to interpolate the corresponding  $(u,v)$  positions in the image



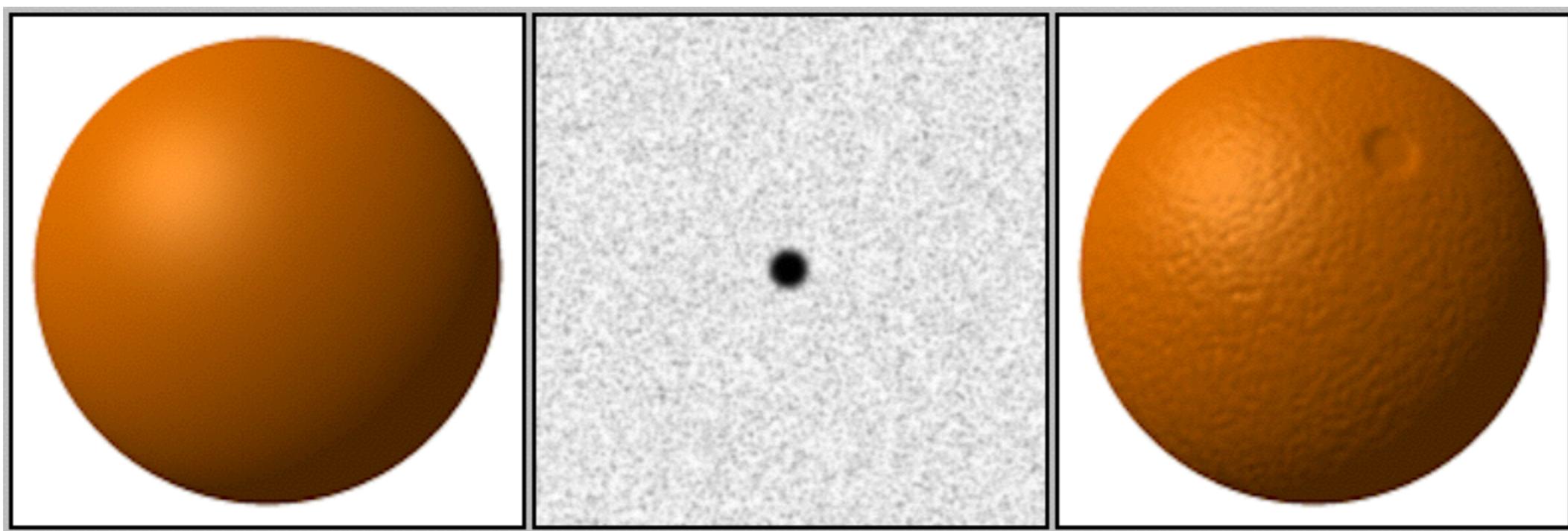
# Bump Maps

- Texture maps are only reflectance
- They don't respond to changes in the lighting ("painted on")
- Idea: use a *normal map*

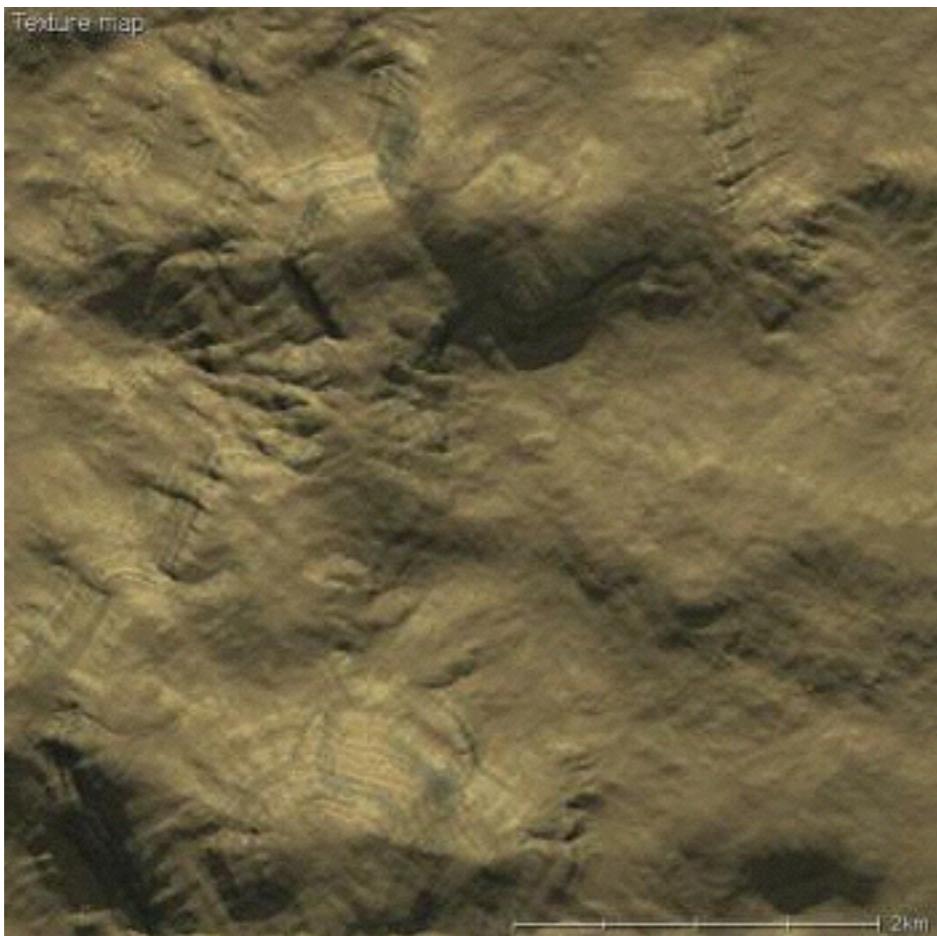


Jim Blinn, 1978

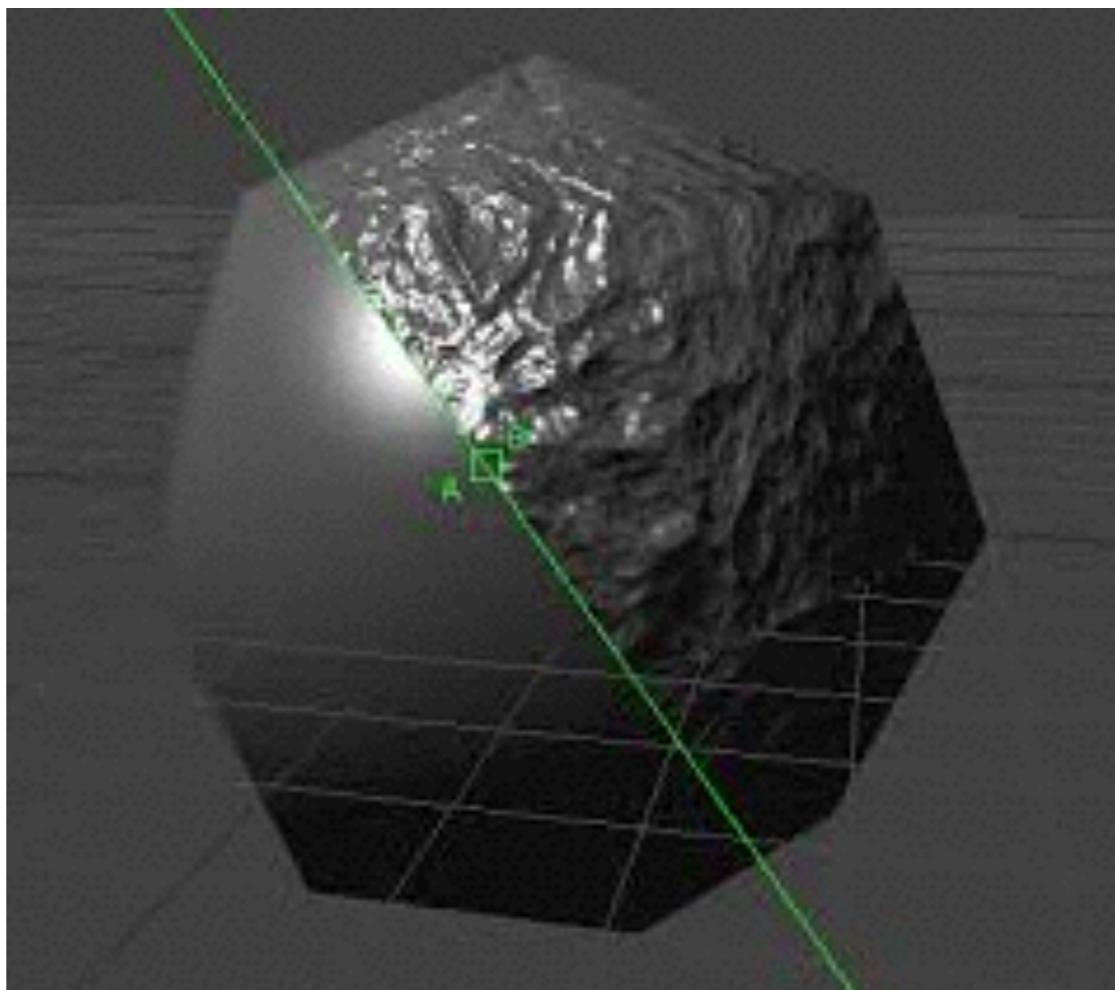
# Bump Maps



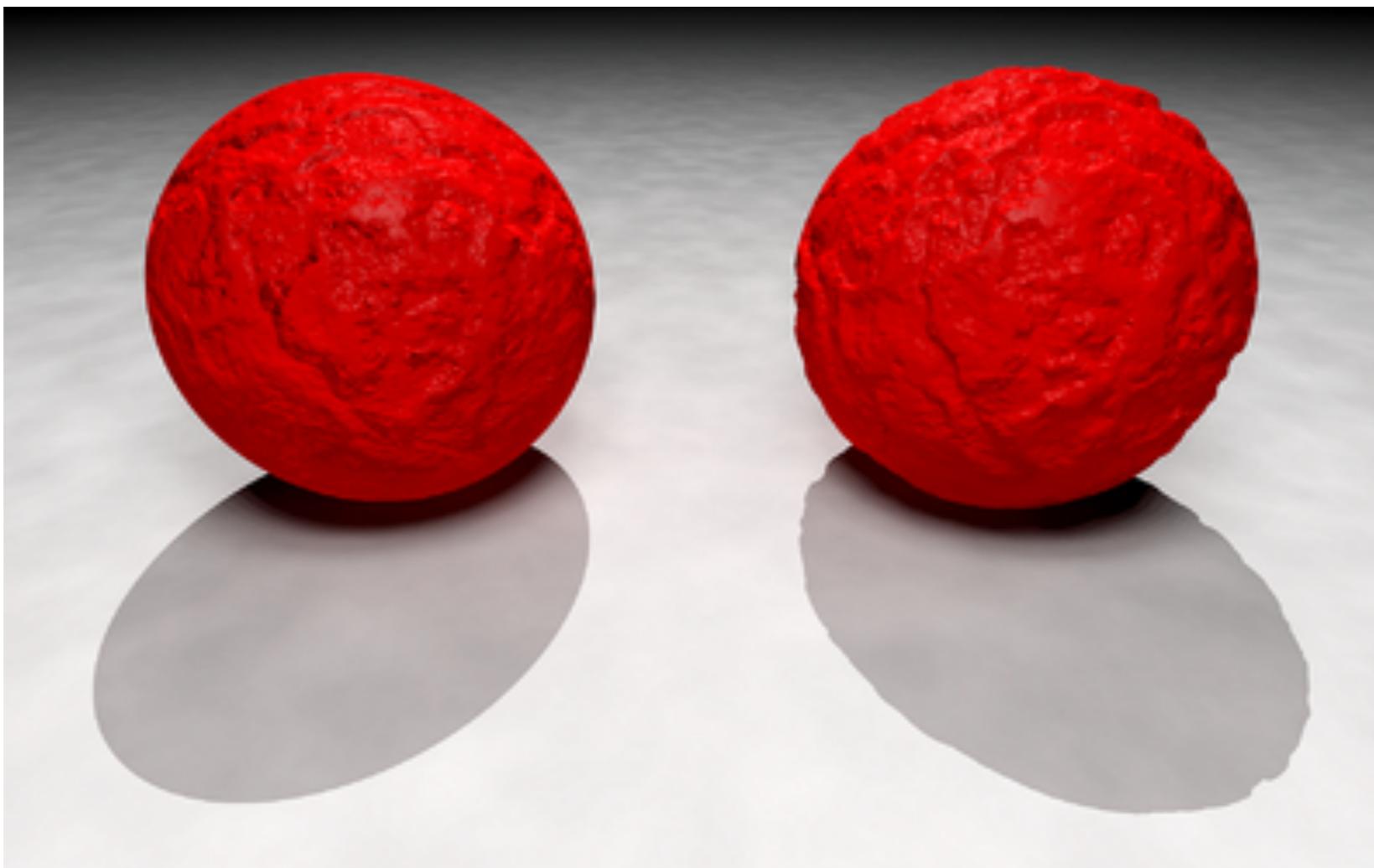
# Bump Mapping



# Bump Maps



# Bump Maps



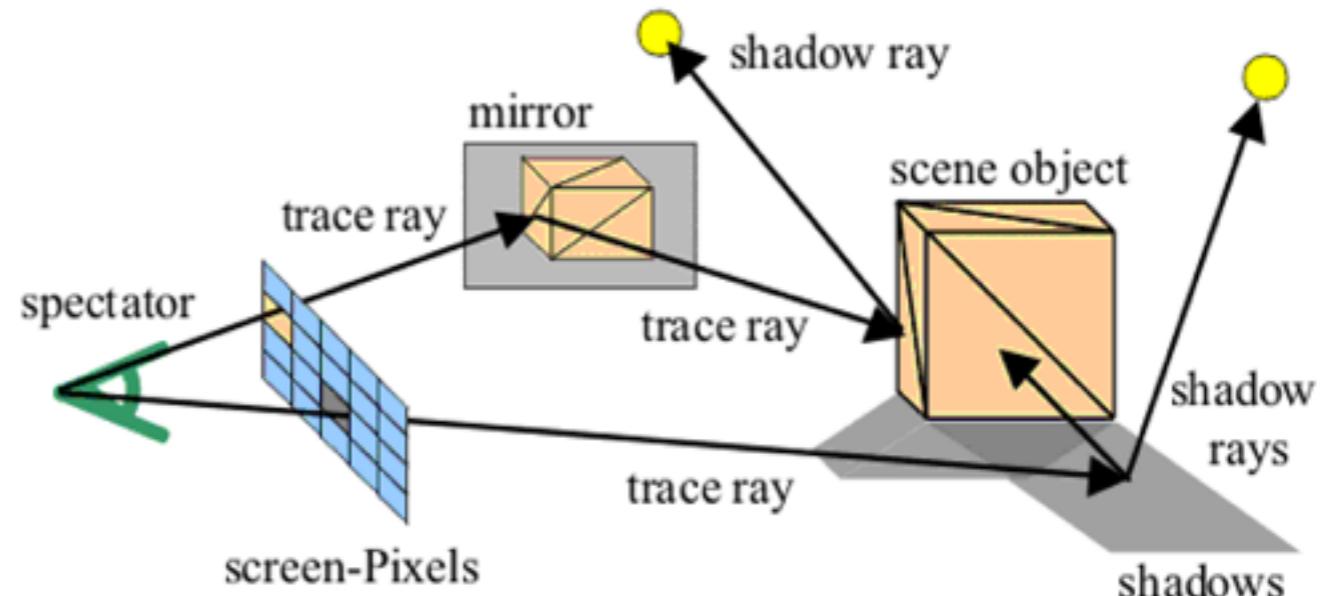
# Ray Tracing

- What about things like
  - Reflections?
  - Refraction through transparent materials?



# Ray Tracing

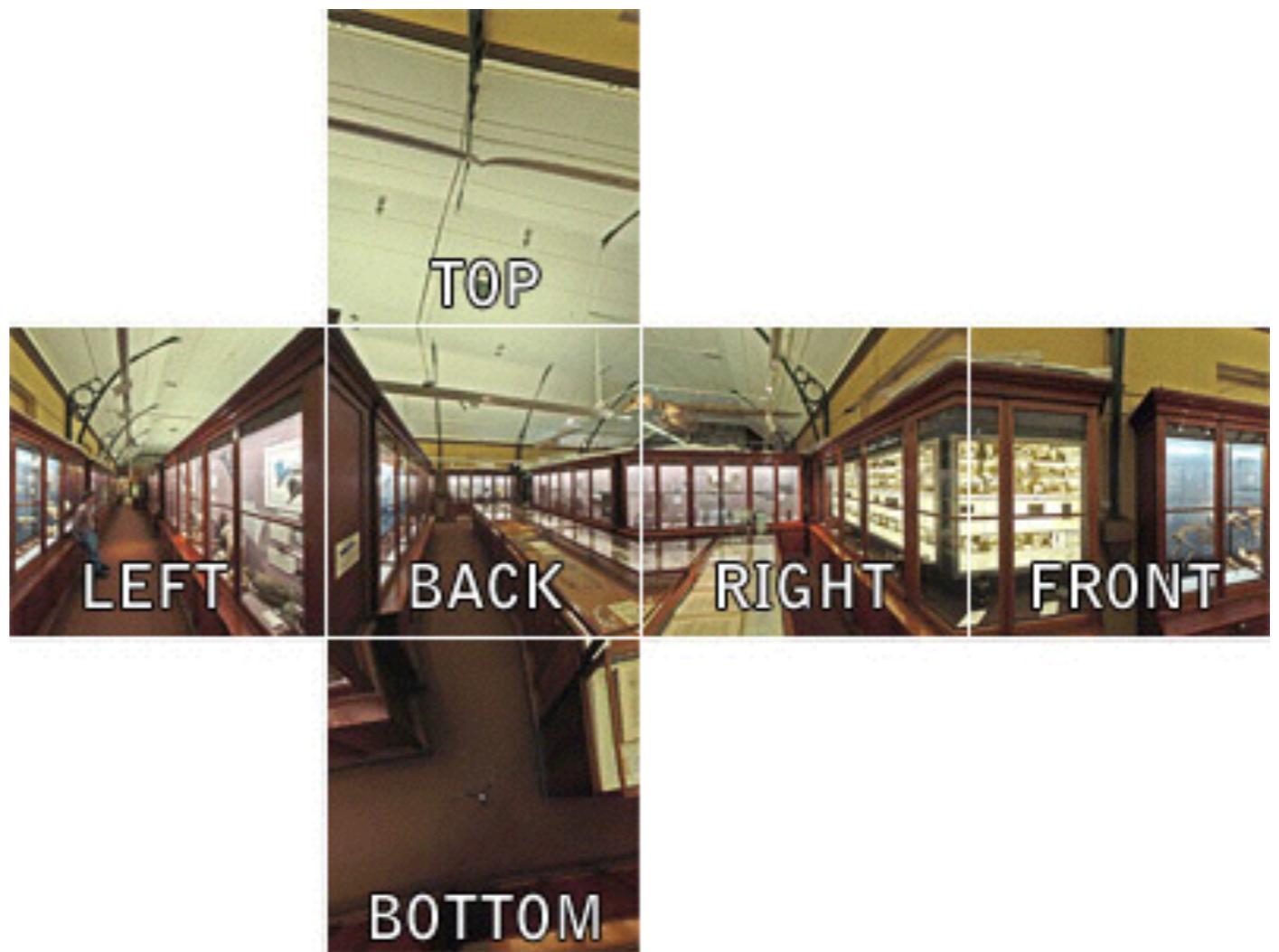
- Shoot a ray out through the pixel on the imaging plane
- When it hits something, **recursively** shoot more rays:
  - Towards light source(s) — direct lighting
  - Reflection ray (if applicable)
  - Refraction ray (if applicable)



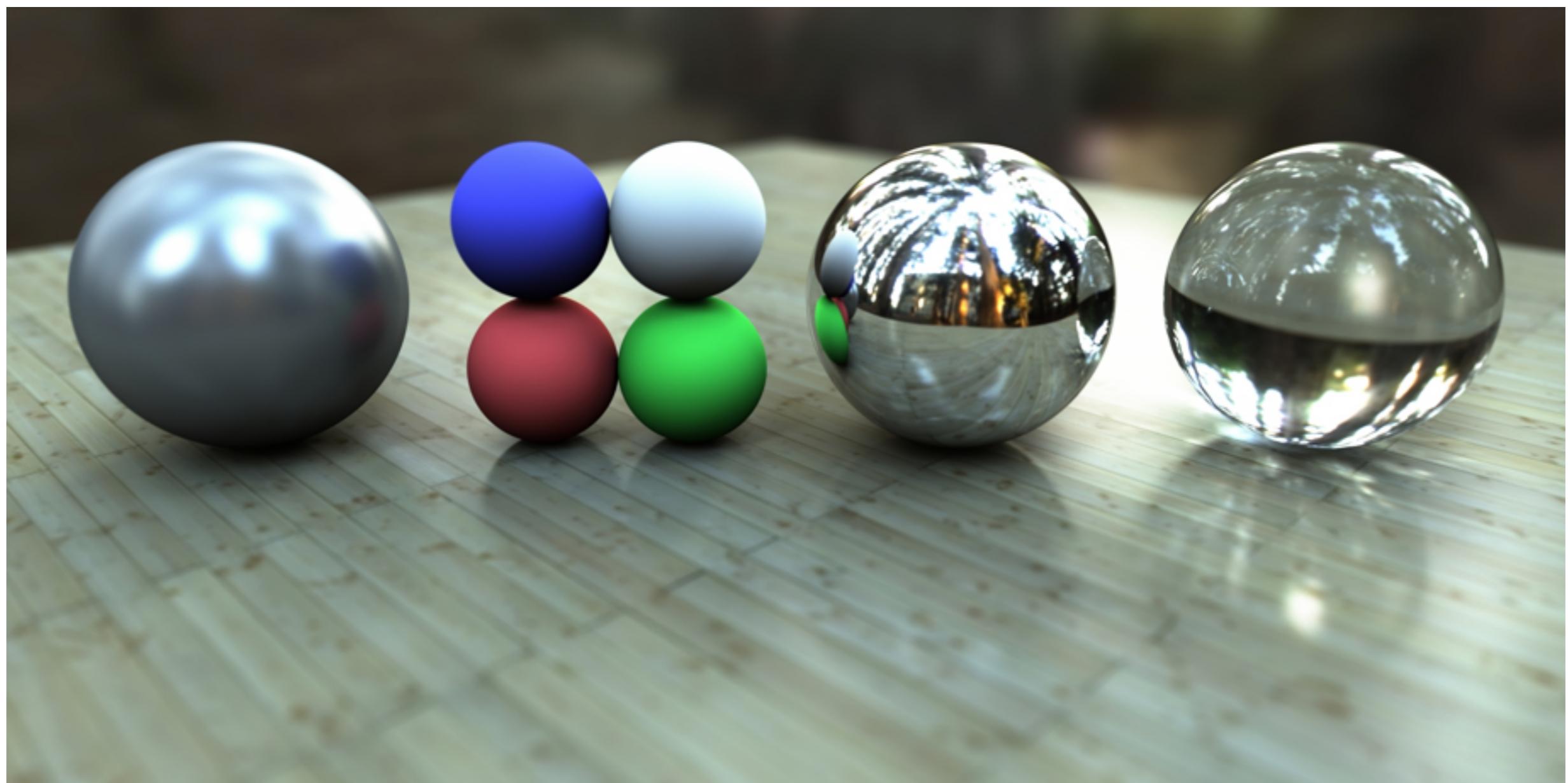
Turner Whitted, 1980

# Environment Maps

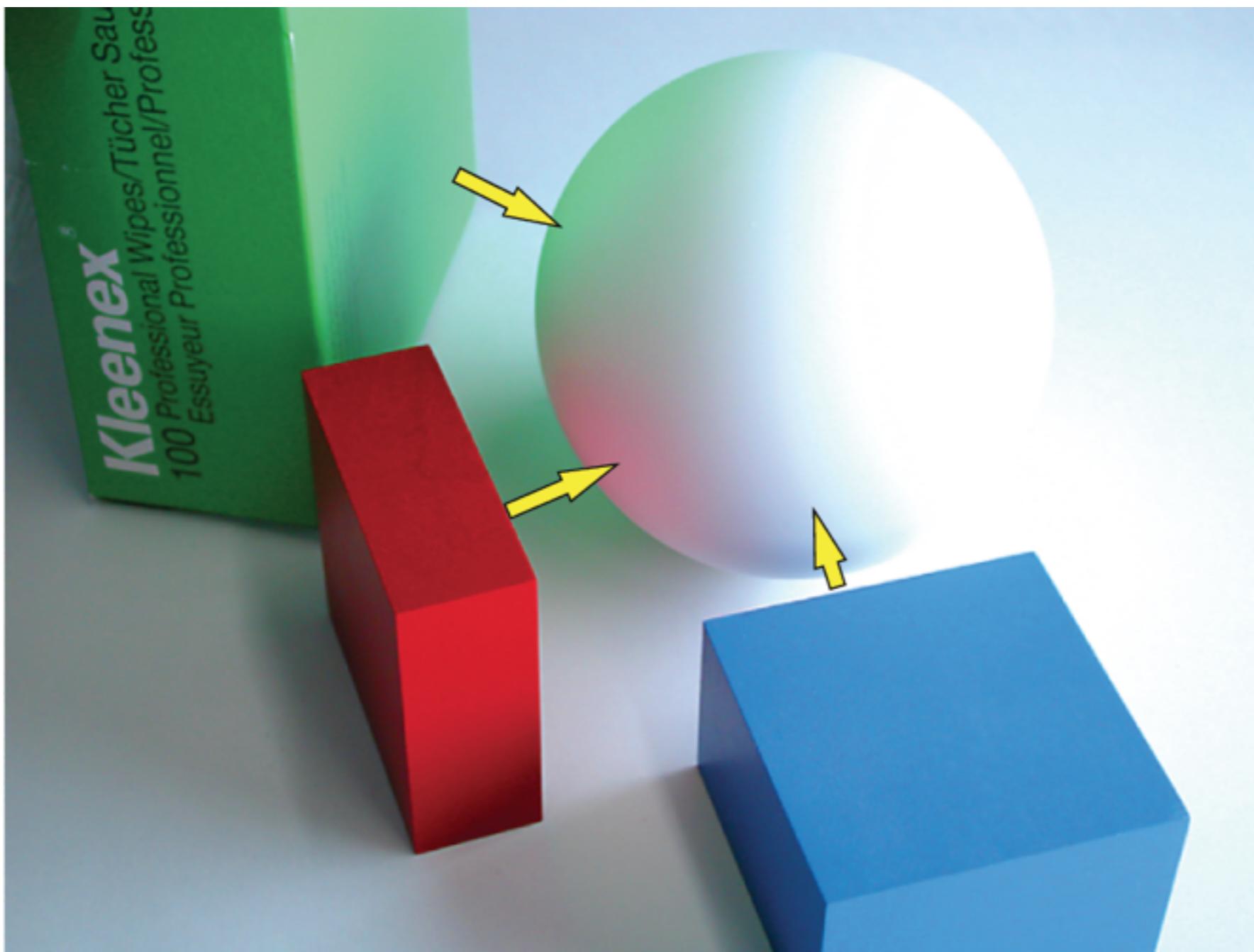
- What about rays that exit the scene?
- Can wrap the entire scene in an *environment map*
- Square, spherical, etc.



# Environment Maps

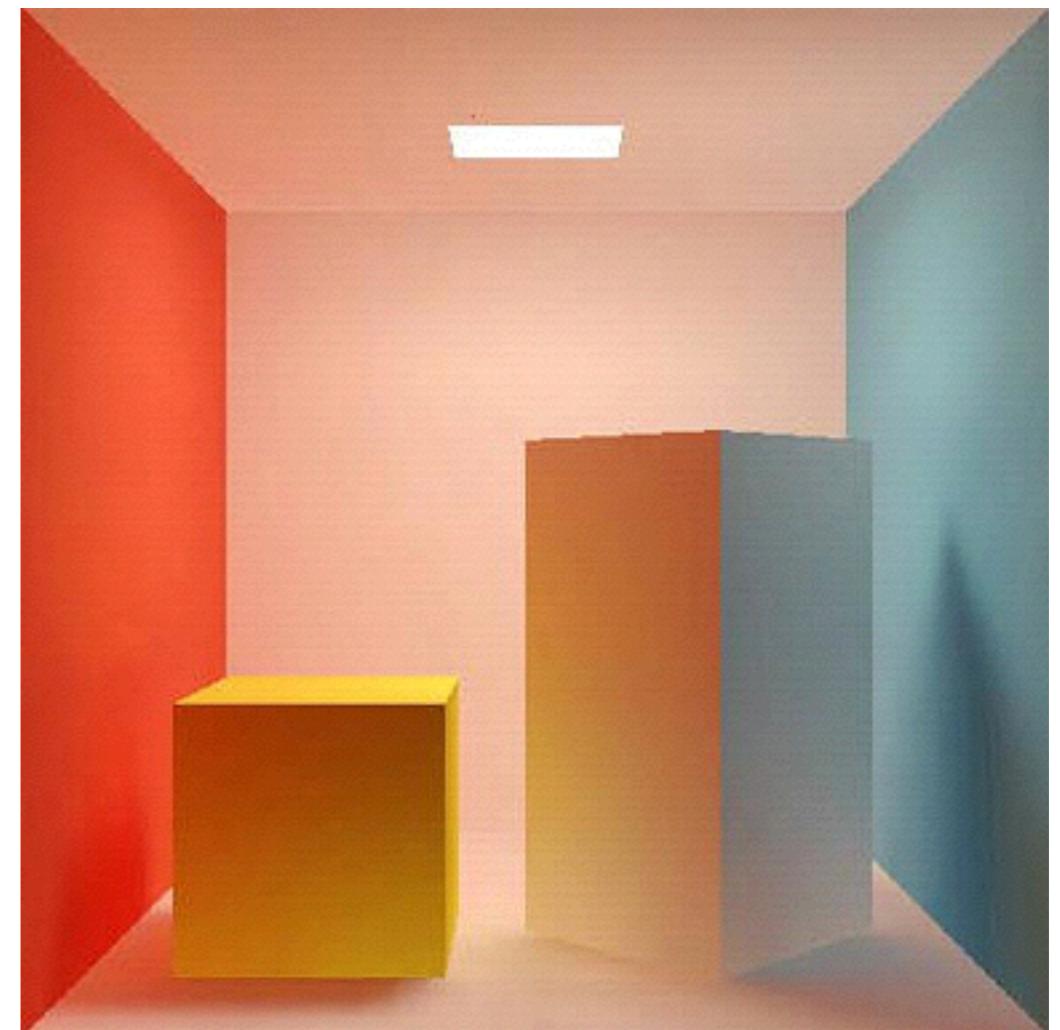


# Diffuse Interreflections



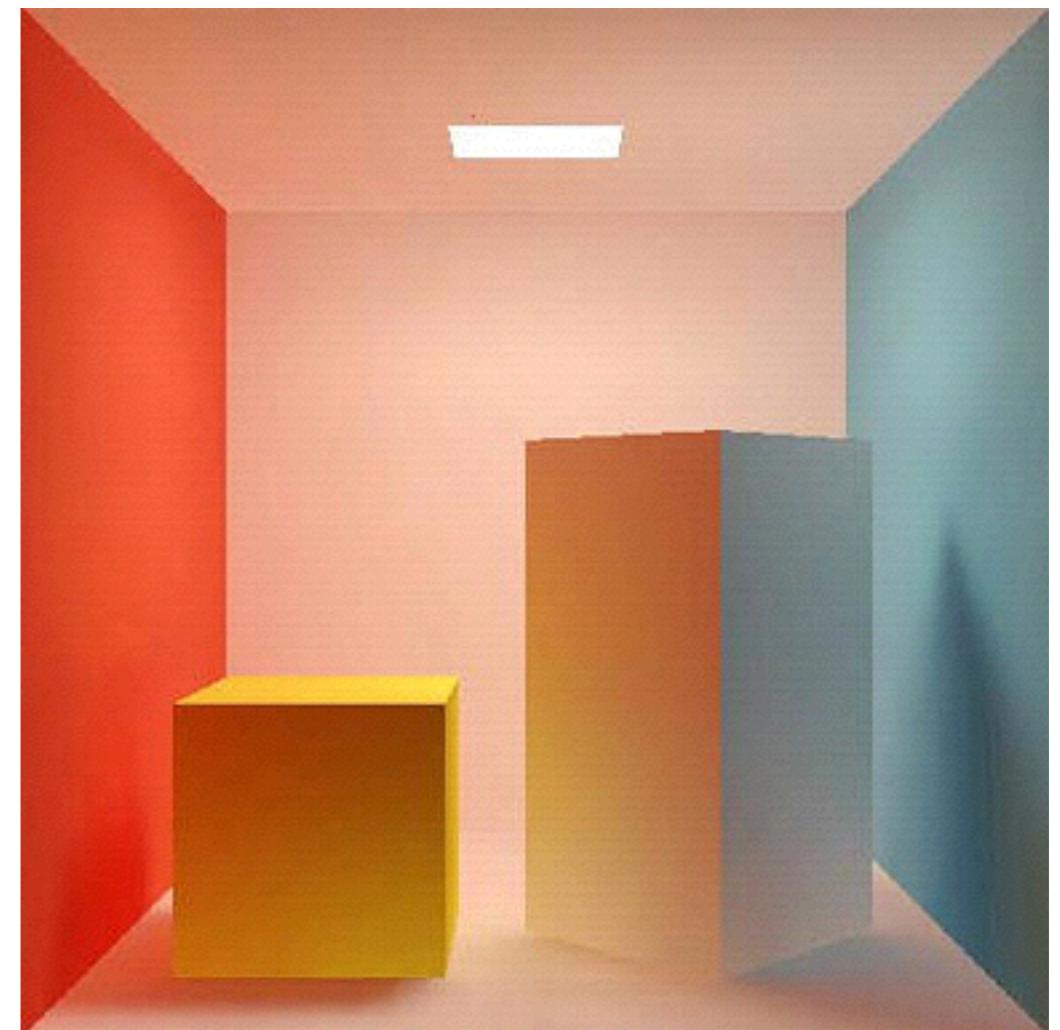
# Radiosity

- Math involves integrals over *all angles* of light coming *into* and *off* of a surface
- Approximated using *importance sampling*
- Effectively traces lots and lots of rays — compute intensive



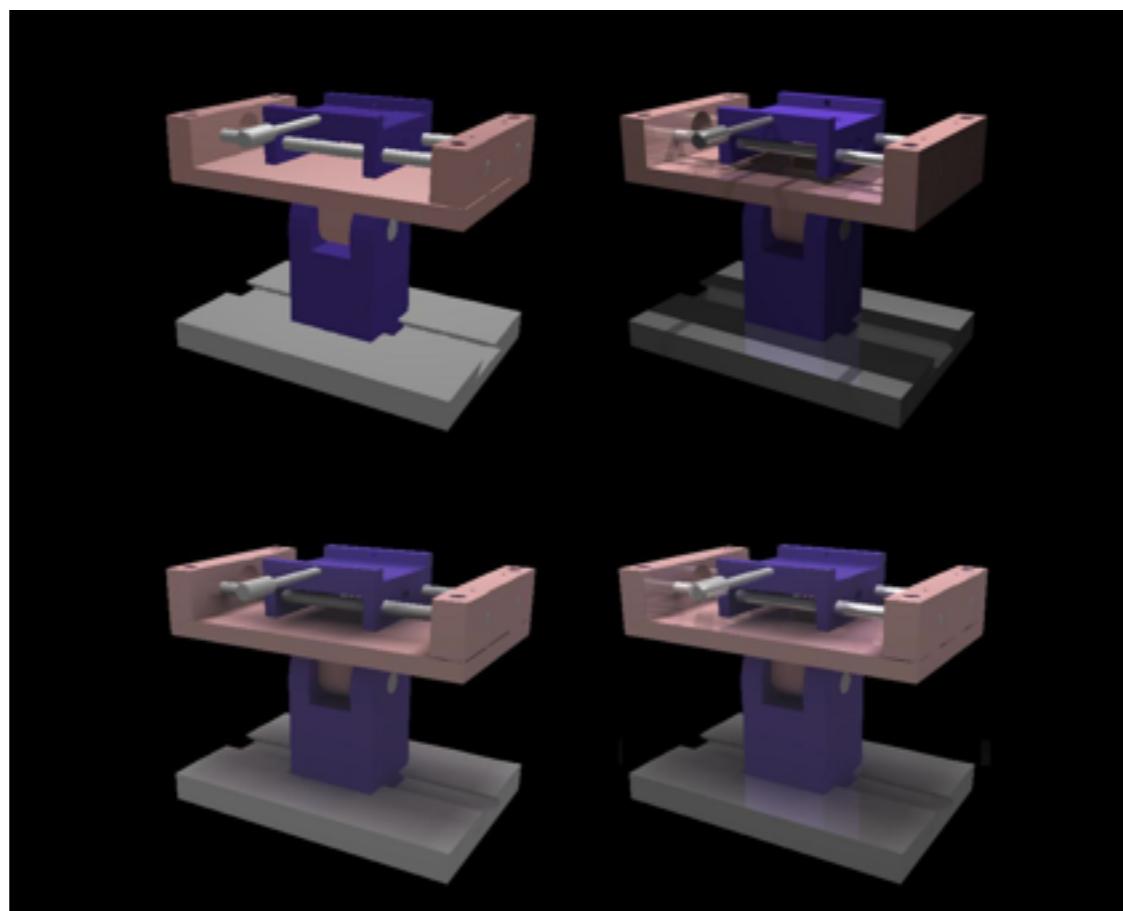
# Radiosity

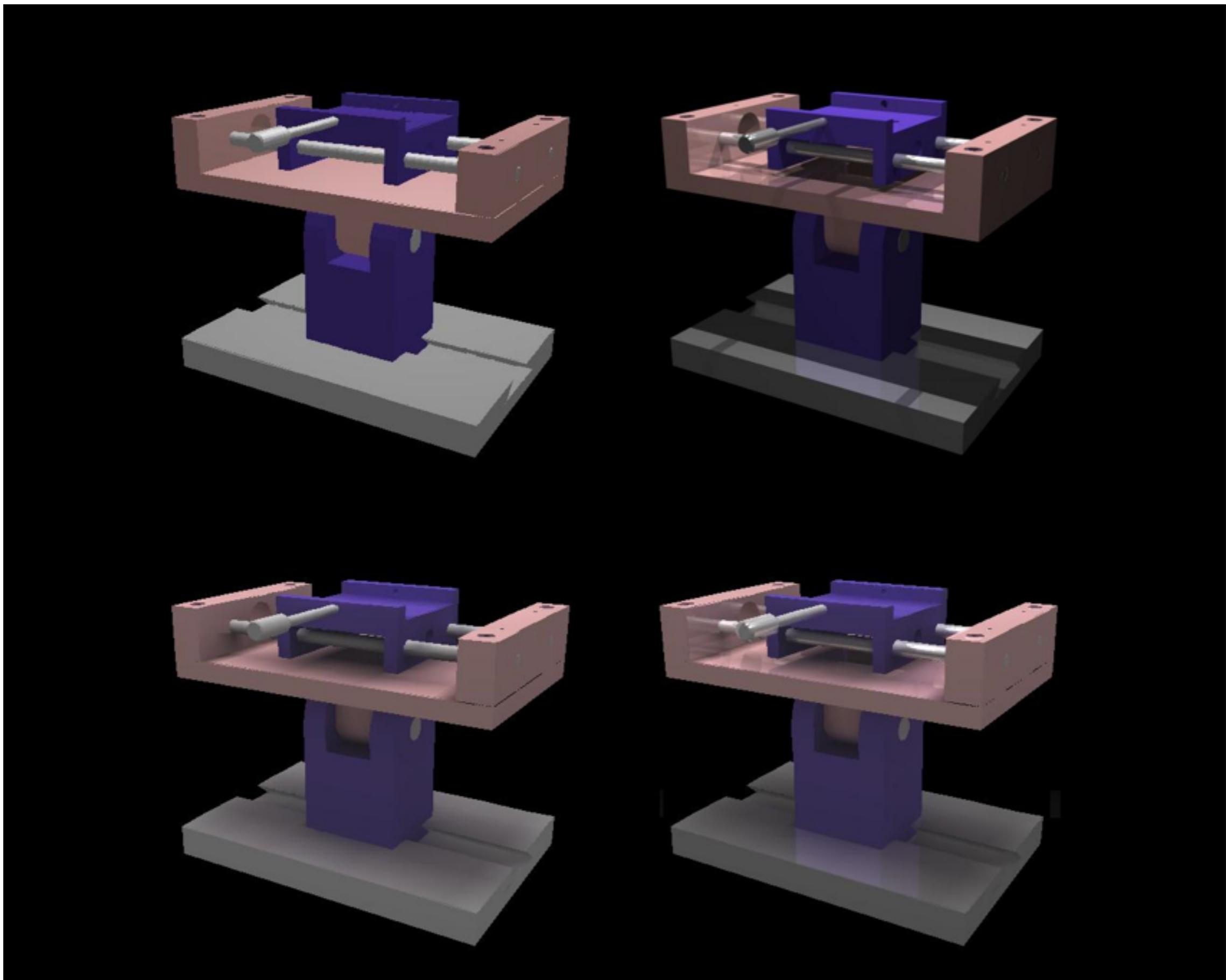
- Because radiosity involves diffuse light, it is independent of the viewpoint
- Can precompute ahead of time if objects and light sources *do not change*
- But if something other than the camera moves...



# Combining Approaches

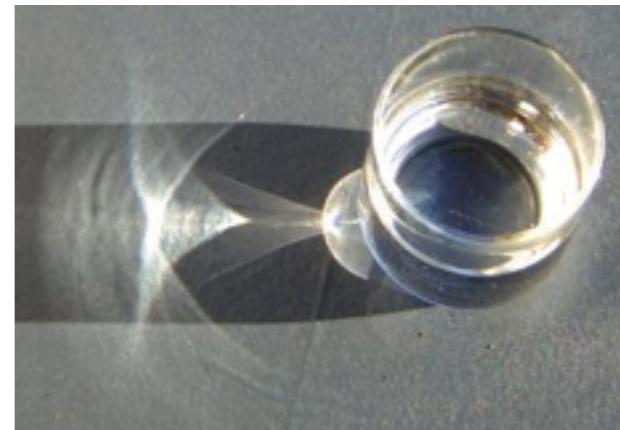
- Ray tracing is good for specular reflections, refraction, etc.
- Radiosity is good for diffuse interreflections
- Can combine methods to get both





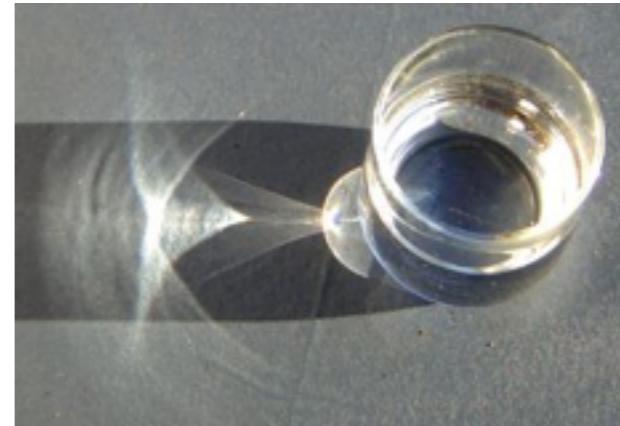
# Forward Rendering

- Some phenomena can't be modeled well by going "backward" from the camera
  - ray tracing
  - radiosity
- Some require "forward" rendering of the light from the light source
- Example: "caustics"



# Forward Rendering

- Forward or hybrid methods:
  - Path tracing
  - Metropolis light transport
  - Photon mapping
- All try to *approximate* the “rendering equation”  
(Jim Kajiya, 1986)



# Coming up...

- Cameras and image acquisition revisited
- Image processing