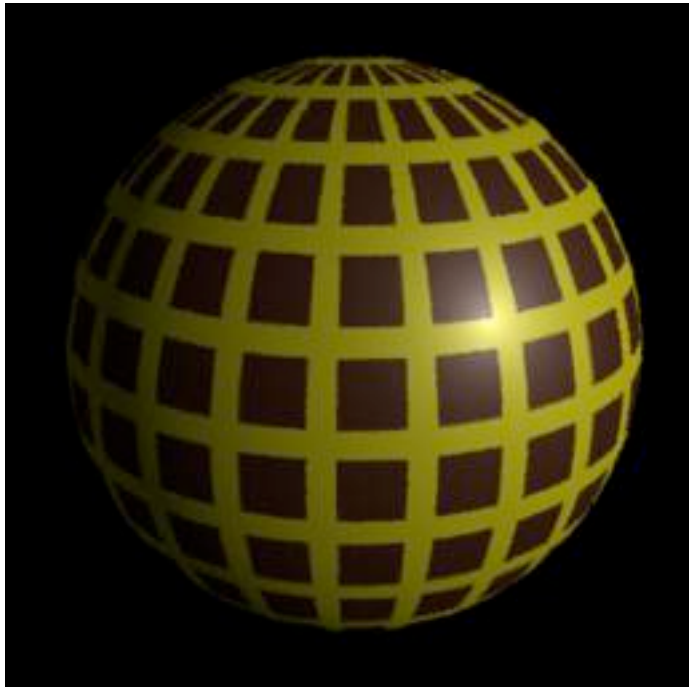
Surface detail

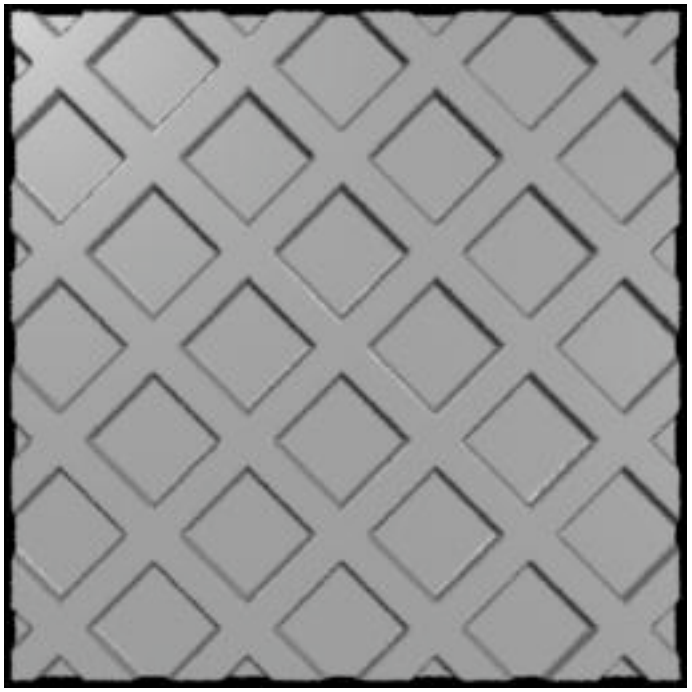
- Knowing the intersection point gives us a position in intrinsic coordinates on the surface
  - e.g. for a triangle, distance from two of three bounding planes
- This is powerful - we could attach some effect at that point
- Texture maps:
  - Make albedo (or color) a function of position in these coordinates
  - Rendering: when intersection is found, compute coordinates and get albedo from a map
  - This is not specific to ray-tracing

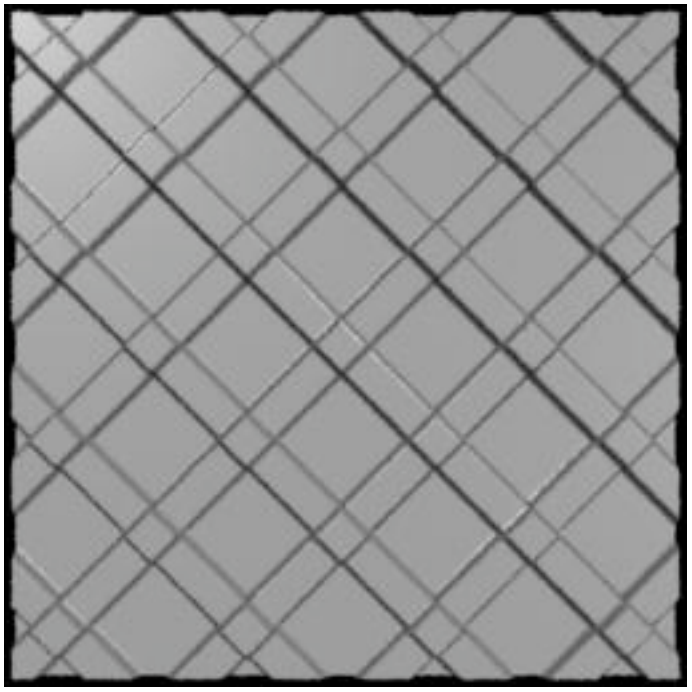


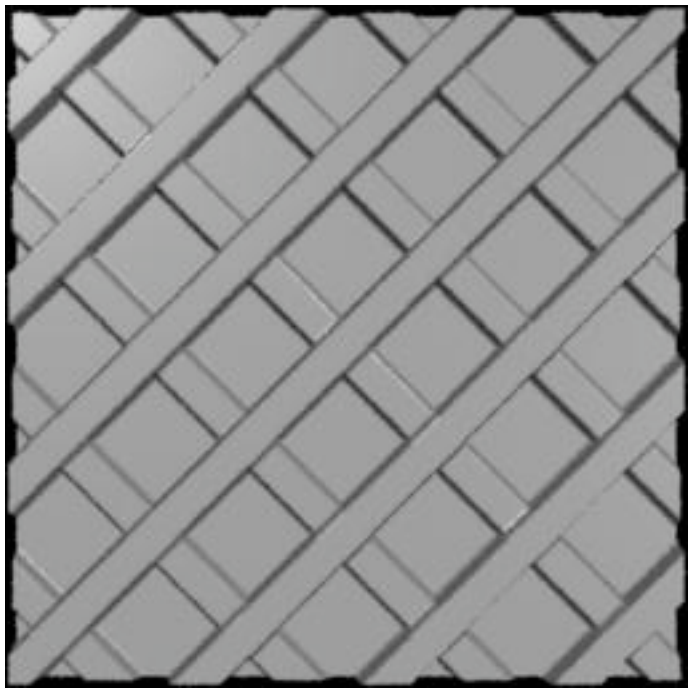
From RmanNotes
http://www.cgrg.ohio-state.edu/~smay/RManNotes/index.html

Surface detail, II

- Bumps
  - we assume that the surface has a set of bumps on it
    - e.g. the pores on an orange
  - these bumps are at a fine scale, so don't really affect the point of intersection, but do affect the normal
  - strategy:
    - obtain normal from "bump function"
    - shade using this modified normal
    - notice that some points on the surface may be entirely dark
    - bump maps might come from pictures (like texture maps)



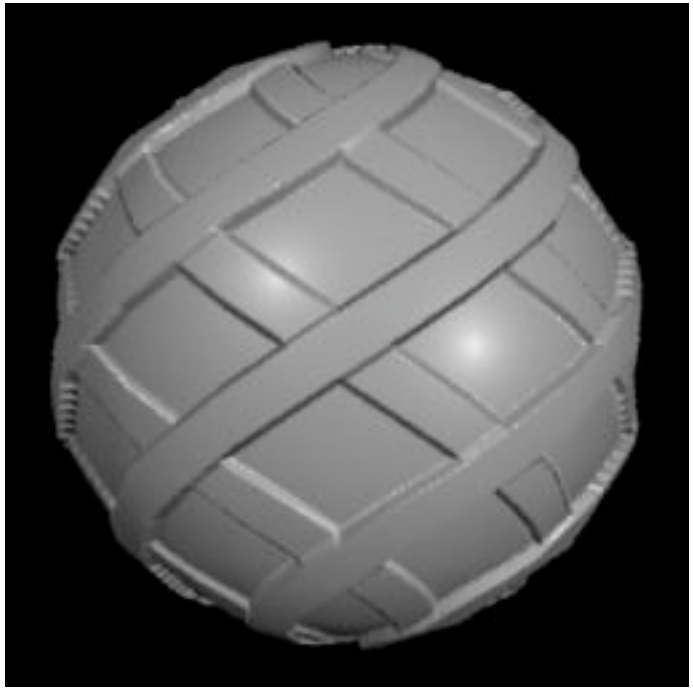




From RmanNotes
http://www.cgrg.ohio-state.edu/~smay/RManNotes/index.html

Surface detail, III

- A more expensive trick is to have a map which includes **displacements** as well
- Must be done **before** visibility

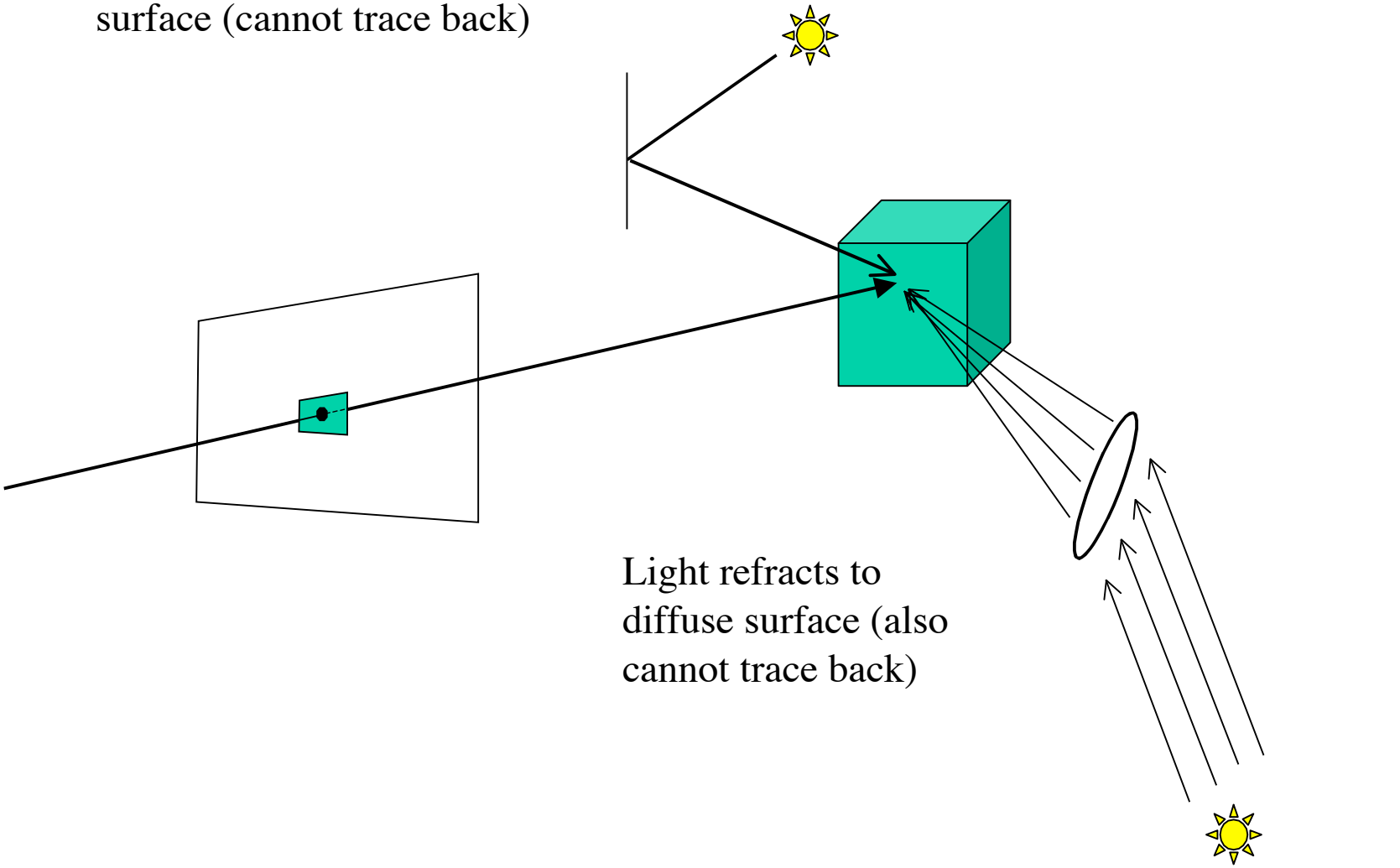


From RmanNotes
http://www.cgrg.ohio-state.edu/~smay/RManNotes/index.html

Illumination effects

- Caustics:
  - refraction or reflection causes light to be “collected” in some regions.
- Specular-> diffuse transfer
  - source reflected in a mirror
- Can’t render this by tracing rays from the eye - how do they know how to get back to the source?

Light bounces of mirror to diffuse surface (cannot trace back)



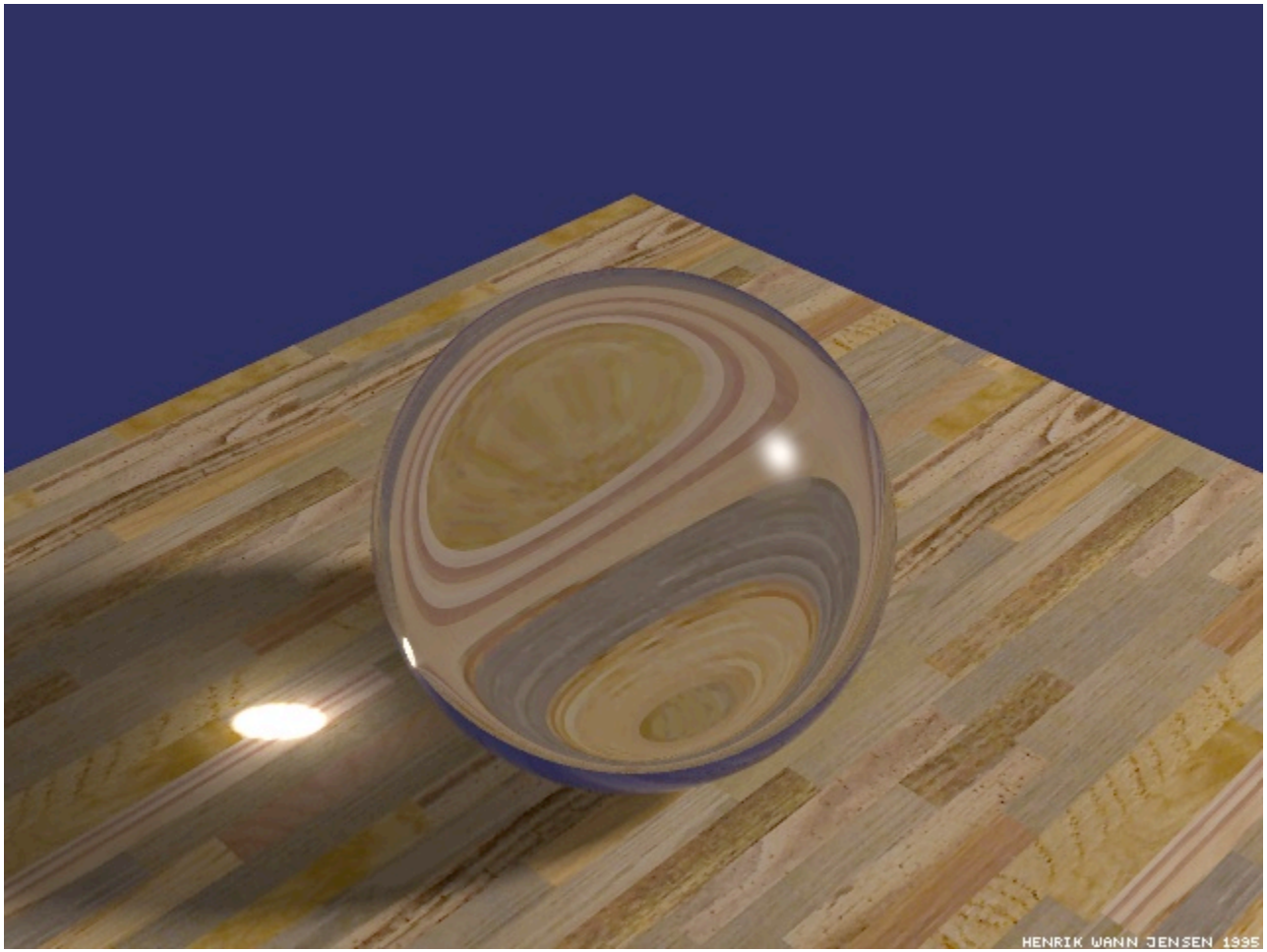
Light refracts to diffuse surface (also cannot trace back)

Point Source

Illumination effects (cont)

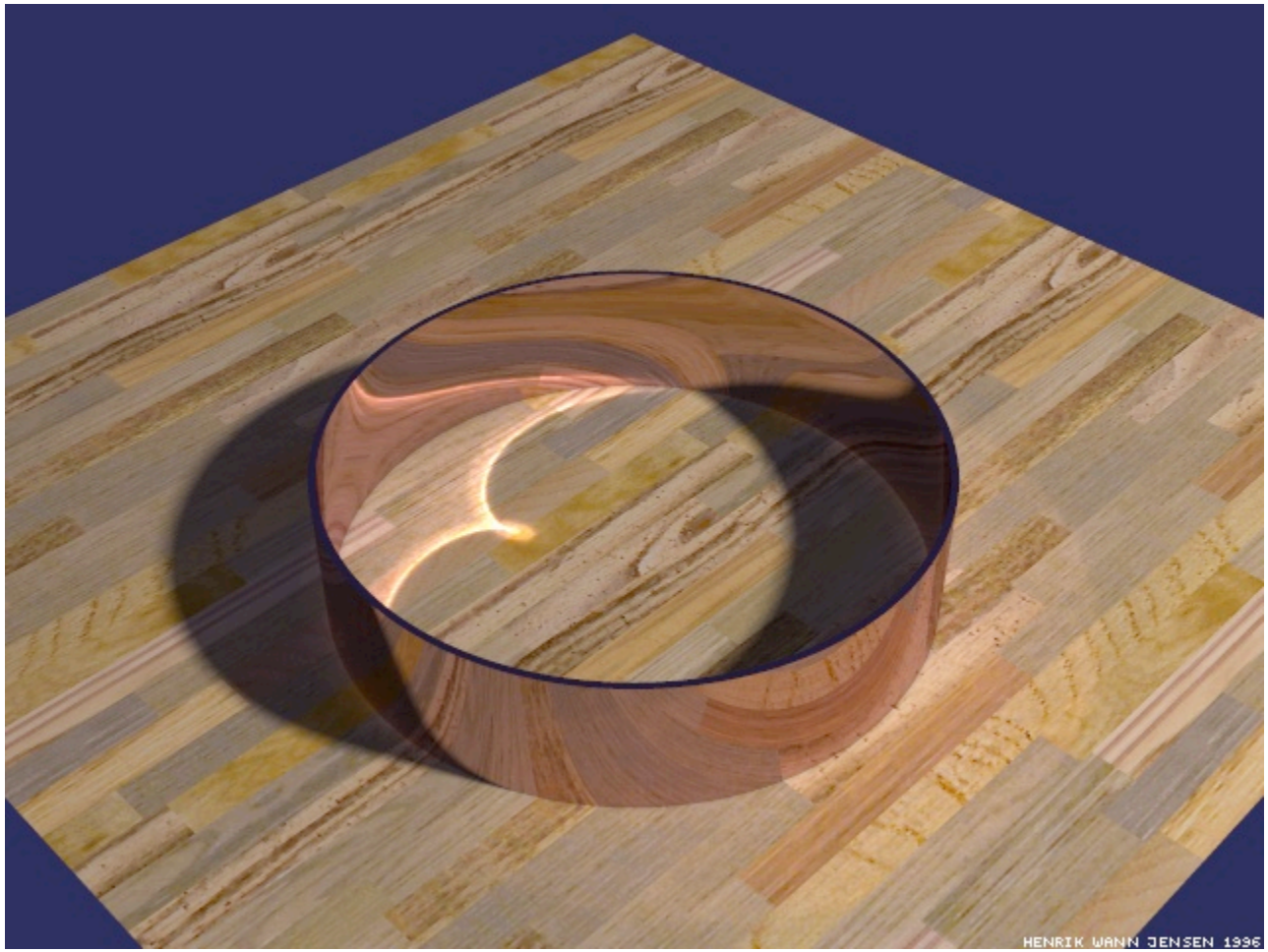
- To get the effect of light reflected and refracted from sources onto diffuse surfaces, trace rays from the light to the first diffuse surface
  - leave a note that illumination has arrived - an illumination map, or photon map
  - sometimes referred to as the forward ray
  - now retrieve this note by tracing eye rays
- Issues
  - efficiency (why trace rays to things that might be invisible?)
  - aliasing (rays are spread out by, say, curved mirrors)

Refraction caustic



Henrik Jensen, http://www.gk.dtu.dk/~hwj

Reflection caustic



Henrik Jensen, http://www.gk.dtu.dk/~hwj

Refraction caustics

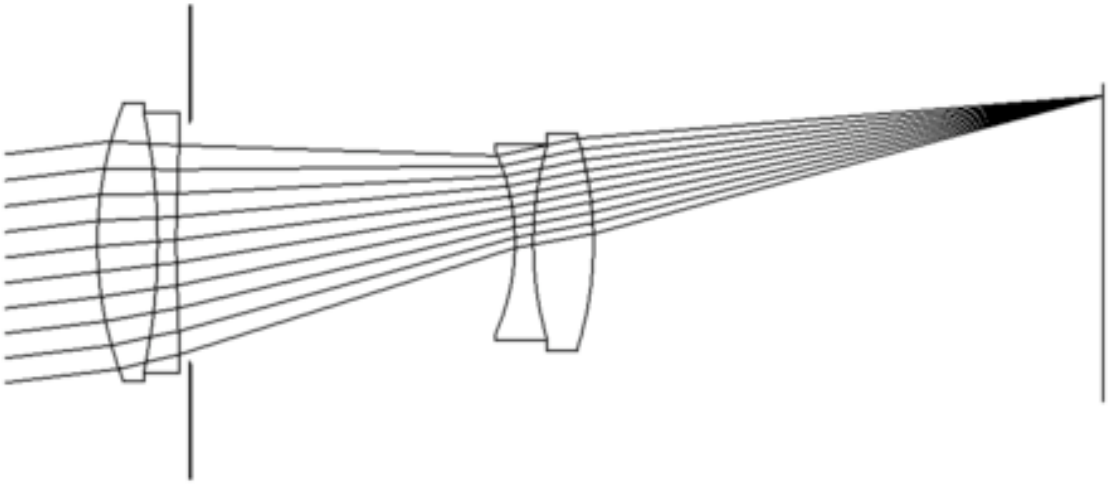
Henrik Jensen, http://www.gk.dtu.dk/~hwj

Lens Effects

Note that a ray tracer very elegantly deals with the projection geometry that we struggled with in earlier lectures which was based on a very simple and “ideal” camera model

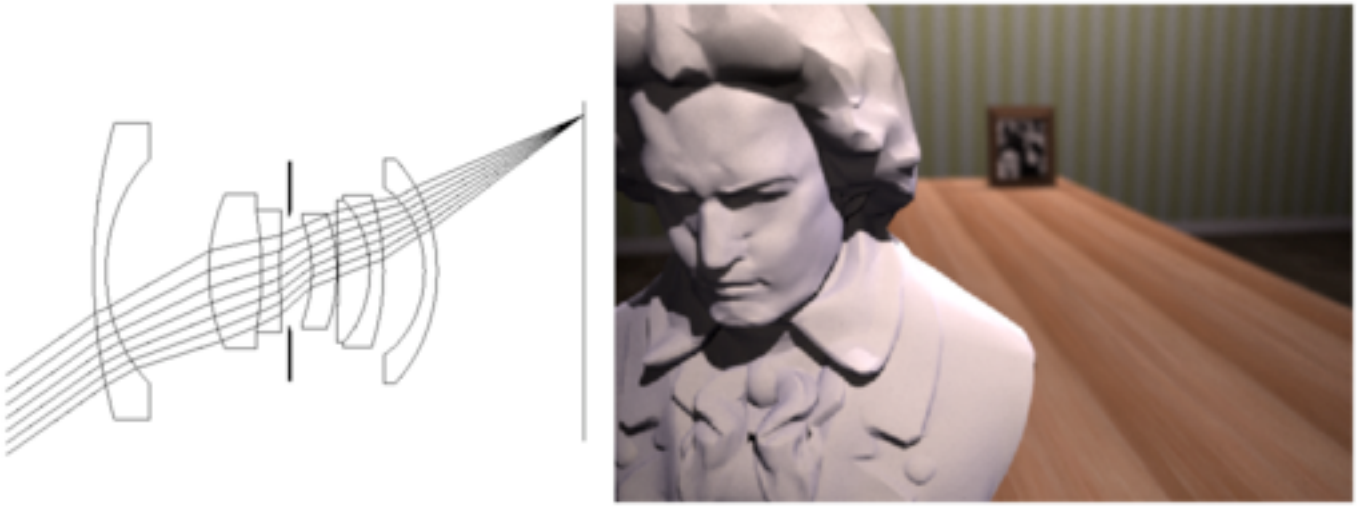
We can go further and introduce a more interesting or realistic camera model

from
A Realistic Camera Model for Computer Graphics
Craig Kolb, Don Mitchell, and Pat Hanrahan
Computer Graphics (Proceedings of SIGGRAPH '95), ACM SIGGRAPH, 1995, pp. 317-324



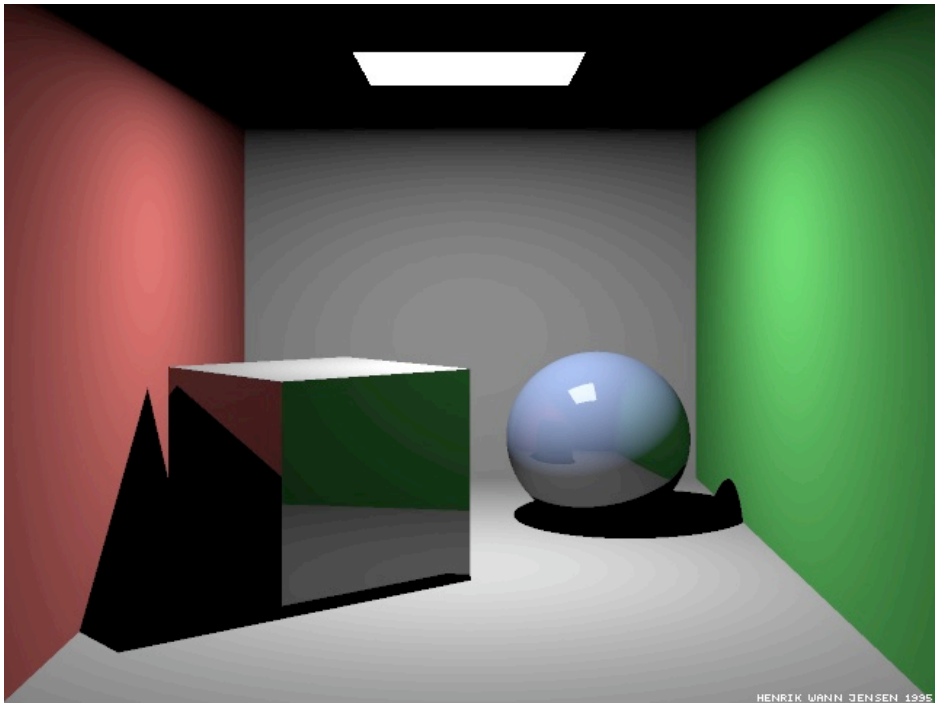
Note limited depth of field, just like a real lens

from
A Realistic Camera Model for Computer Graphics
Craig Kolb, Don Mitchell, and Pat Hanrahan
Computer Graphics (Proceedings of SIGGRAPH '95), ACM SIGGRAPH, 1995, pp. 317-324

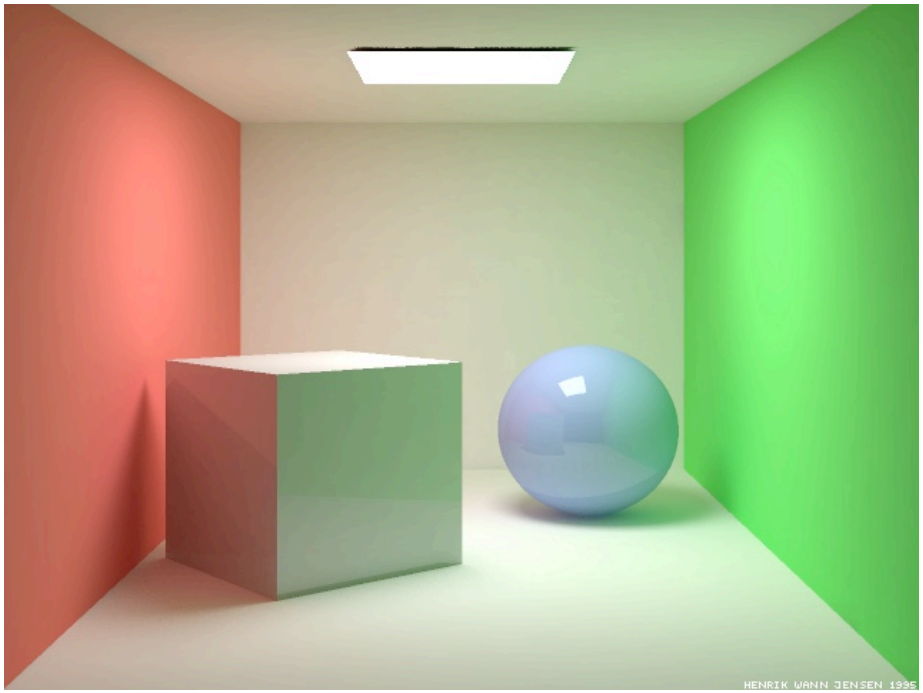


from
A Realistic Camera Model for Computer Graphics
Craig Kolb, Don Mitchell, and Pat Hanrahan
Computer Graphics (Proceedings of SIGGRAPH '95), ACM SIGGRAPH, 1995, pp. 317-324

Radiosity



Ray-traced Cornell box, due to Henrik Jensen, http://www.gk.dtu.dk/~hwj



Radiosity Cornell box, due to Henrik Jensen, http://www.gk.dtu.dk/~hwj, rendered with ray tracer

Radiosity

Want to capture the basic effect that surfaces illuminate each other

Again, following every piece of light from a diffuse reflector is impractical--but combinations of brute force and clever hacks can be done

Another approach: Radiosity methods

Radiosity

Think of the "world" as a bunch of patches. Some are sources, (and reflect), some just reflect. Each sends light towards all the others.

Consider one color band at a time (some of the computation is shared among bands).

Each surface, i, *radiates* reflected light, B_i

Each surface, *emits* light E_i (if it is not a source, this is 0).

Denote the albedo of surface i as ρ_i

Radiosity equation

$$B_i = E_i + \rho_i \sum_j F_{j \to i} B_j \frac{A_j}{A_i}$$

The form factor $F_{j \to i}$

is the fraction of light leaving dA_j arriving at dA_i taking into account orientation and obstructions

Useful relation

$$A_i F_{i\to j} = A_j F_{j\to i}$$

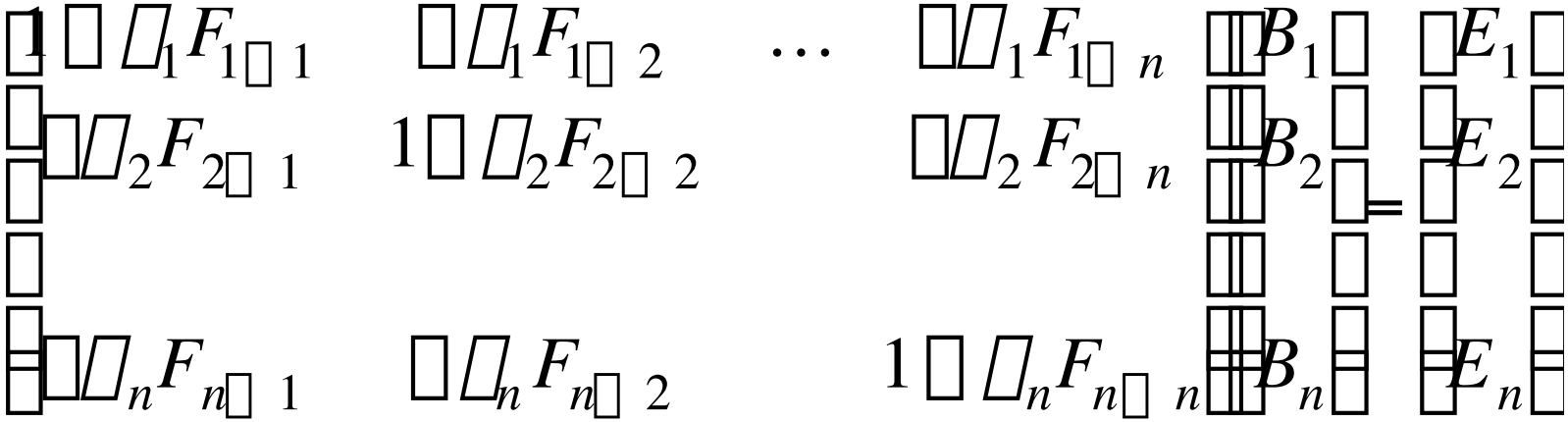
The equation now becomes

$$B_i = E_i + \rho_i \sum_j F_{i\to j} B_j$$

Rearrange to get

$$B_i - \rho_i \sum_j F_{i\to j} B_j = E_i$$

In matrix form

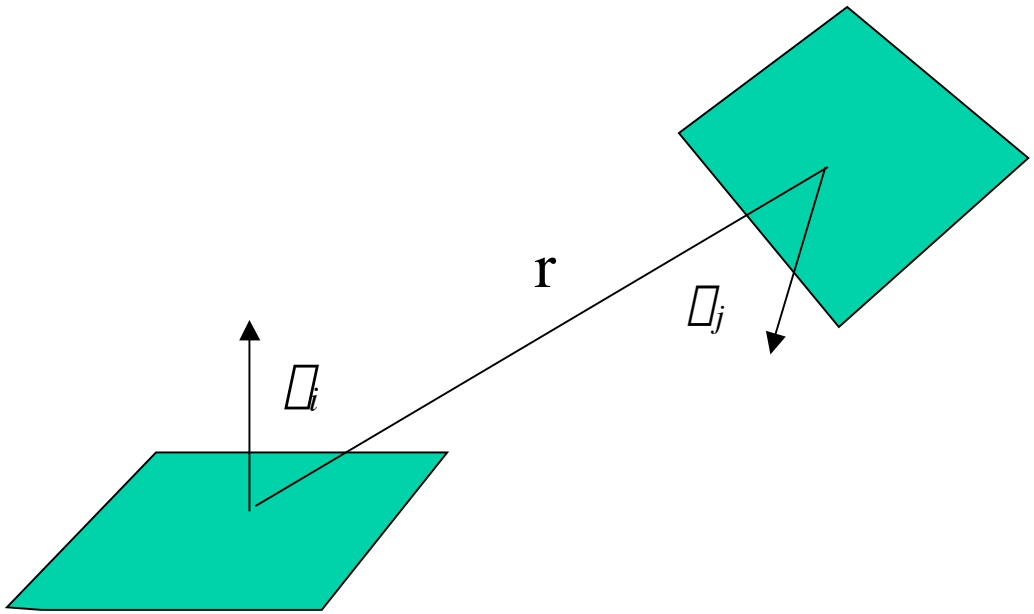


So, in theory, we just compute the Bi's by solving this (large!) matrix equation.

Optional

The fun part: Computing the $F_{i\to j}$

Without obstruction $dF_{dj\to di} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j$



Optional

Fancy methods exist for of computing and/or approximating storing form factors (e.g. hemisphere and hemicube methods)

Hemicube: For a given patch, project other patches onto a cube surrounding it, which is an appropriately scaled version of the projection onto the hemisphere (use cube trick because planar projection is faster).

Hemicube projections can be tagged for distance (like Z-buffer) to deal with for occlusion.

Iterative Solution (gather)

The matrix equation can be solved iteratively (Gauss-Siedel method) starting with a crude estimate of the solution.

More intuitively, consider an estimate $\hat{B}_j$ for the B_j and plug them into our equation to re-estimated them.

$$B_i = E_i + \rho_i \sum_j F_{i\,j} \hat{B}_j$$

This is sometimes referred to as “gather”, as we update the brightness of each patch in turn by gathering light from the other patches.

Iterative Solution (cast)

Iteration has the additional benefit that we can provide intermediate, approximate, solutions while the user waits.

But, in the previous version, each patch gets better in turn. Better to have all patches get a little better on each iteration.

This leads to the alternative approach where energy is cast from each patch in turn to update the others with:

$$B_j \quad due \;\; to \;\; B_i \;\; is \;\; p_j B_i F_{i\square j} \frac{A_i}{A_j}$$

A second advantage is that the casting can be done in order of brightness, which is obviously helpful.