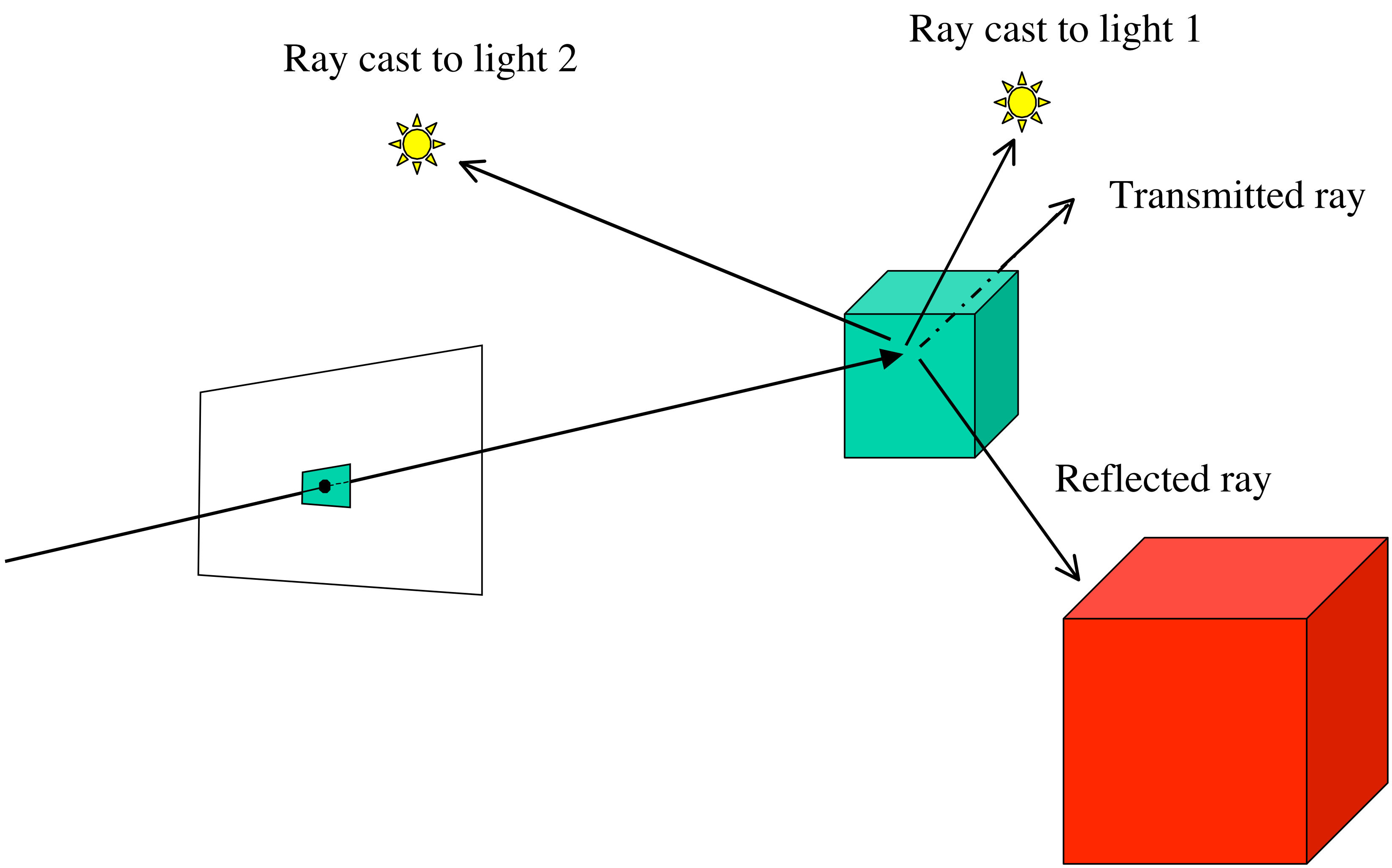
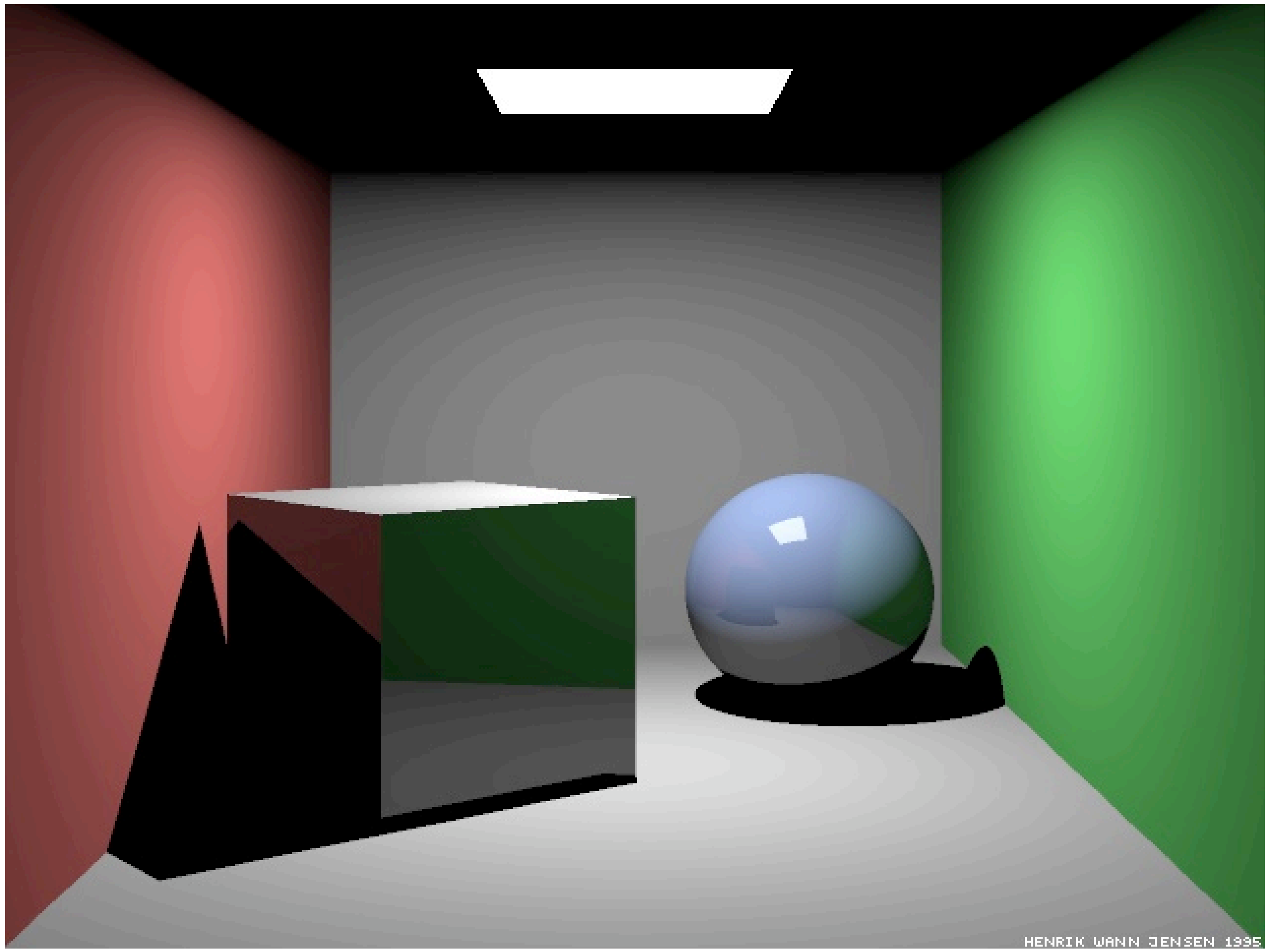
Recursive ray tracing

H&B, page 597





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

PCKTWTCH by Kevin Odhner, POVRay

6Z4.JPG - A Philco 6Z4 vacuum tube by Steve Anger

Issues

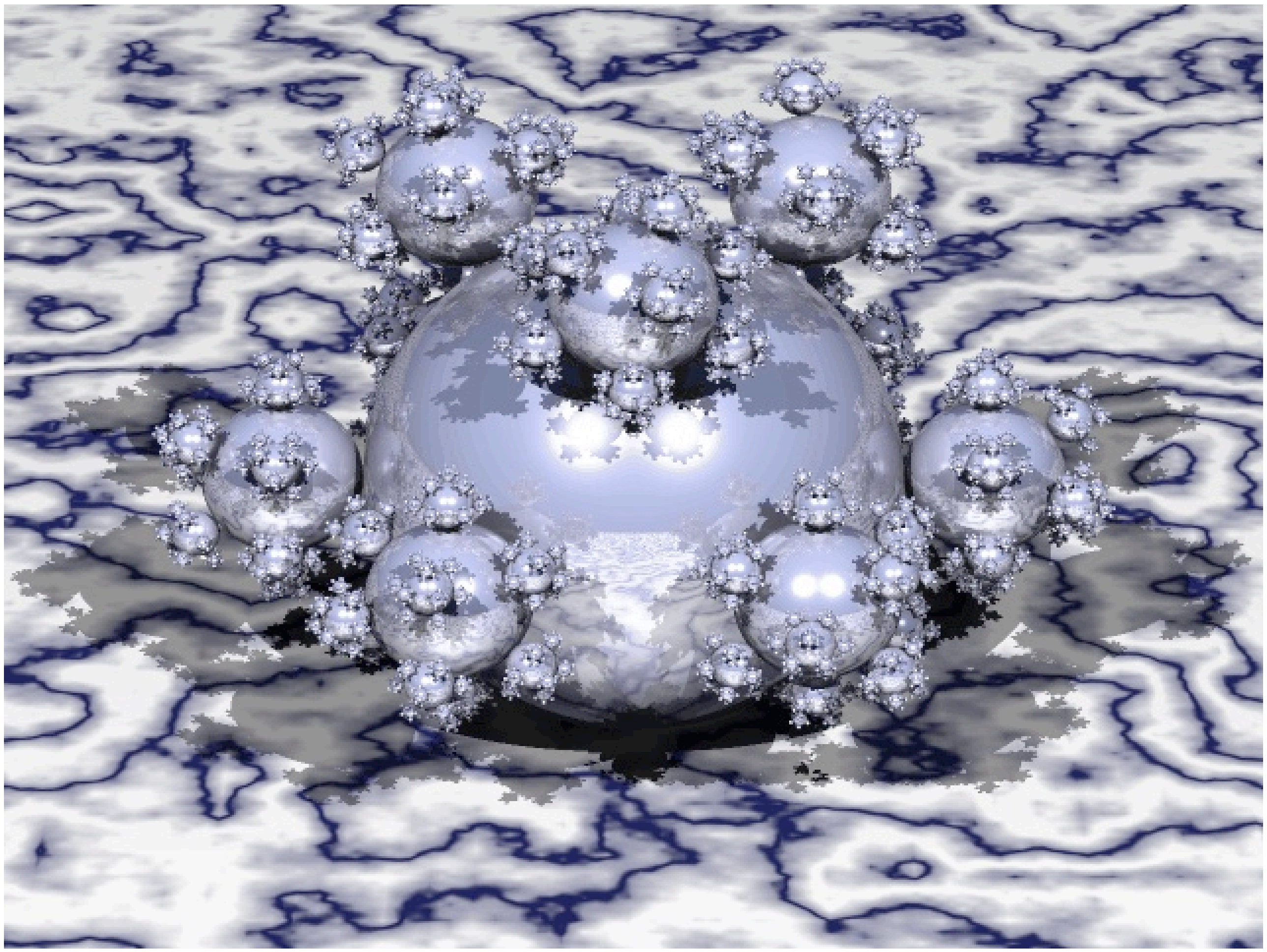
- Sampling (aliasing)
- Very large numbers of objects
  - Need making intersections efficient, exclude as much as possible using clever data structures
- Surface detail
  - bumps, texture, etc.
- Illumination effects
  - Caustics, specular to diffuse transfer
- Camera models

Sampling

- Simplest ray-tracer is one ray per pixel
  - This gives aliasing problems
- Solutions
  - Cast multiple rays per pixel, and use a weighted average
  - Rays can be on a uniform grid
  - It turns out to be better if they are “quite random” in position
    - “hard-core” Poisson model appears to be very good
    - different patterns of rays at each pixel

Efficiency - large numbers of objects

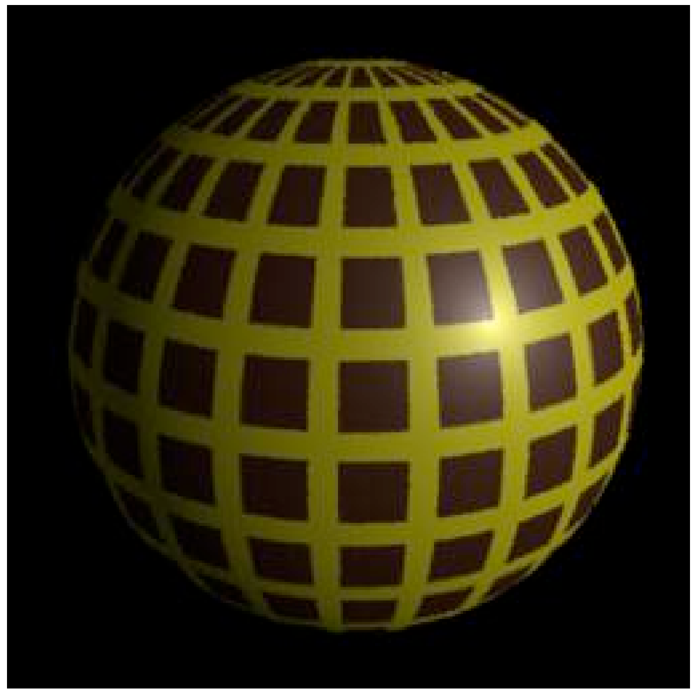
- Construct a space subdivision hierarchy of some form to make it easier to tell which objects a ray might intersect
- Uniform grid
  - easy, but many cells
- Bounding Spheres
  - easy intersections first
- Octtree
  - rather like a grid, but hierarchical
- BSP tree



500,000 spheres, Henrik Jensen, http://www.gk.dtu.dk/~hwj

Surface detail

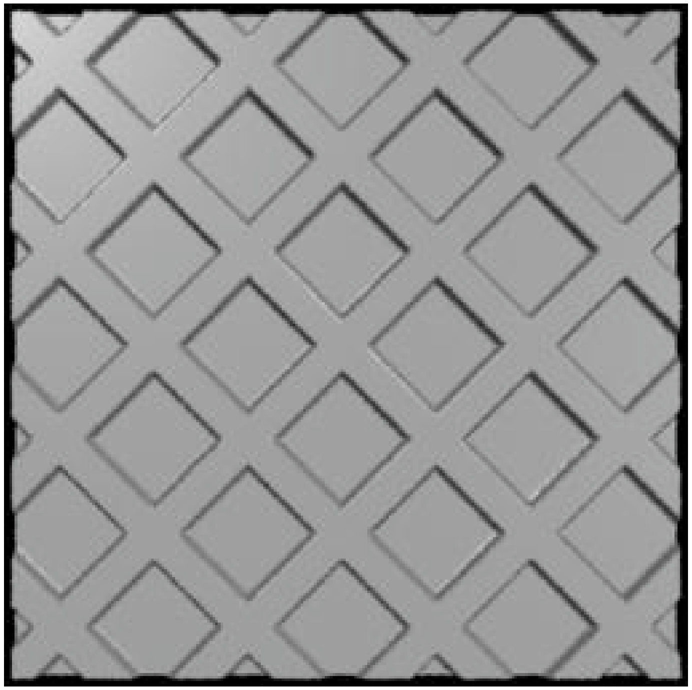
- Knowing the intersection point gives us a position in intrinsic coordinates on the surface
  - e.g. for a triangle, distance from two of three bounding planes
- This is powerful - we could attach some effect at that point
- Texture maps:
  - Make albedo (or color) a function of position in these coordinates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

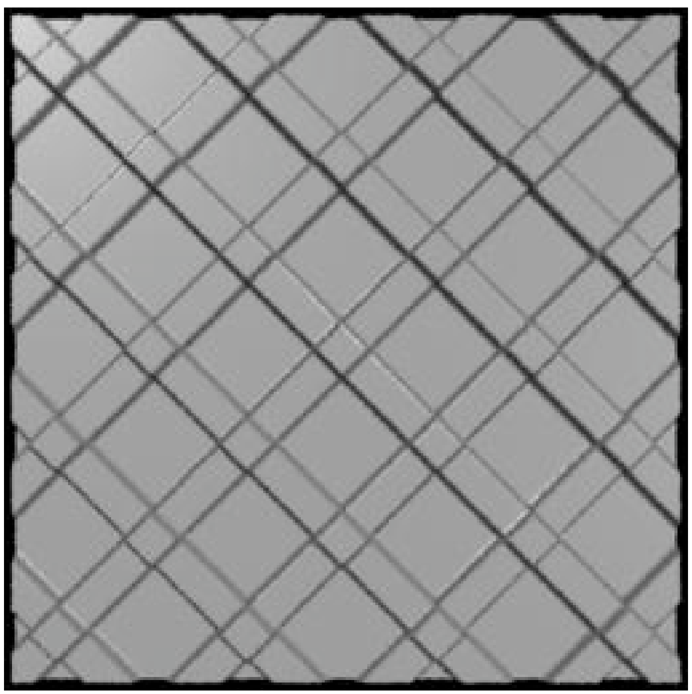


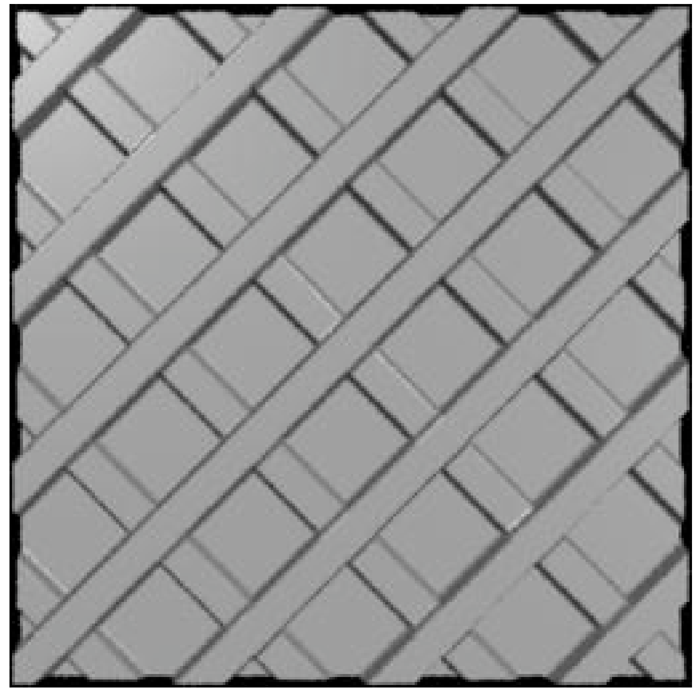
From RmanNotes: http://www.cgrg.ohio-state.edu/~smay/RManNotes/index.html

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)



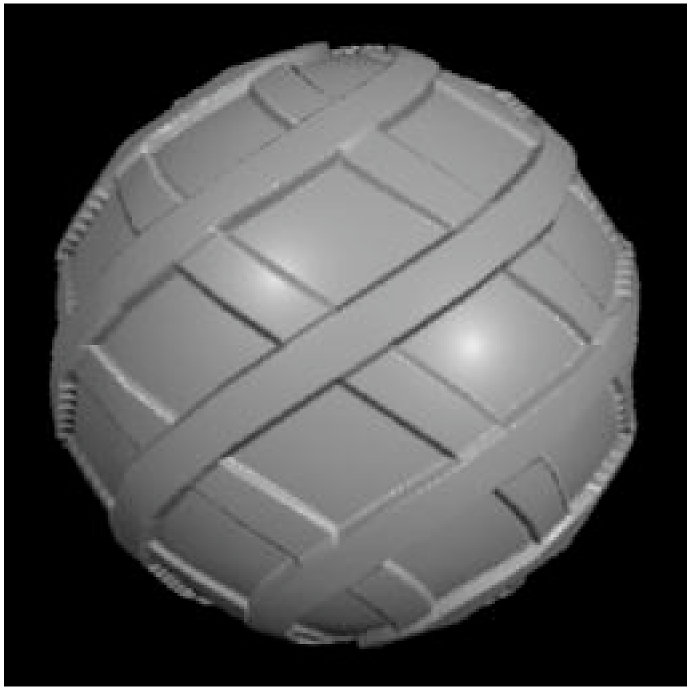




From RmanNotes
http://www.cgrg.ohiostate.edu/~smay/RManNotes/index.html

Surface detail, III

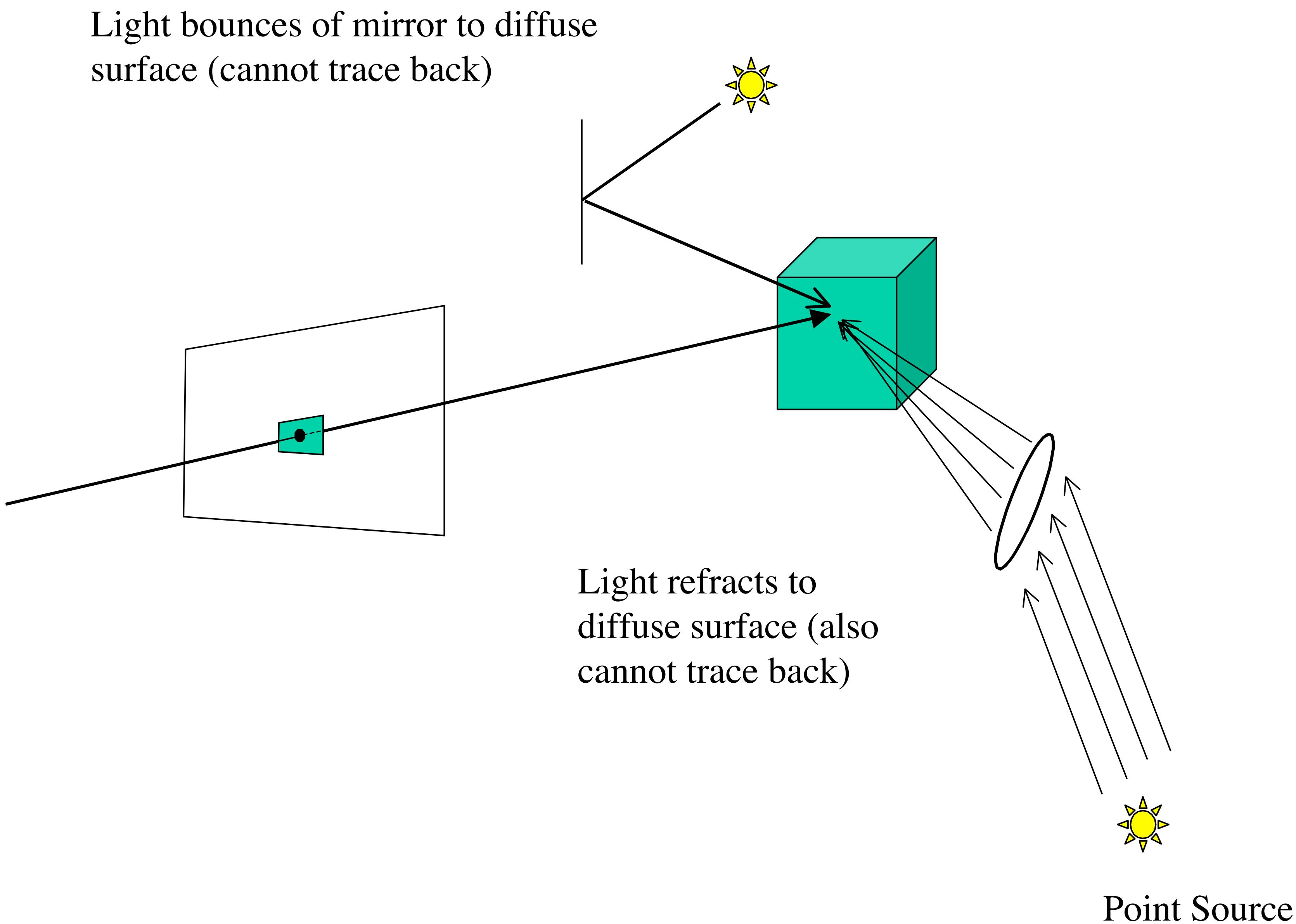
- A more expensive trick is to have a map which includes **displacements** as well
- Must be done **before** visibility



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

Illumination effects

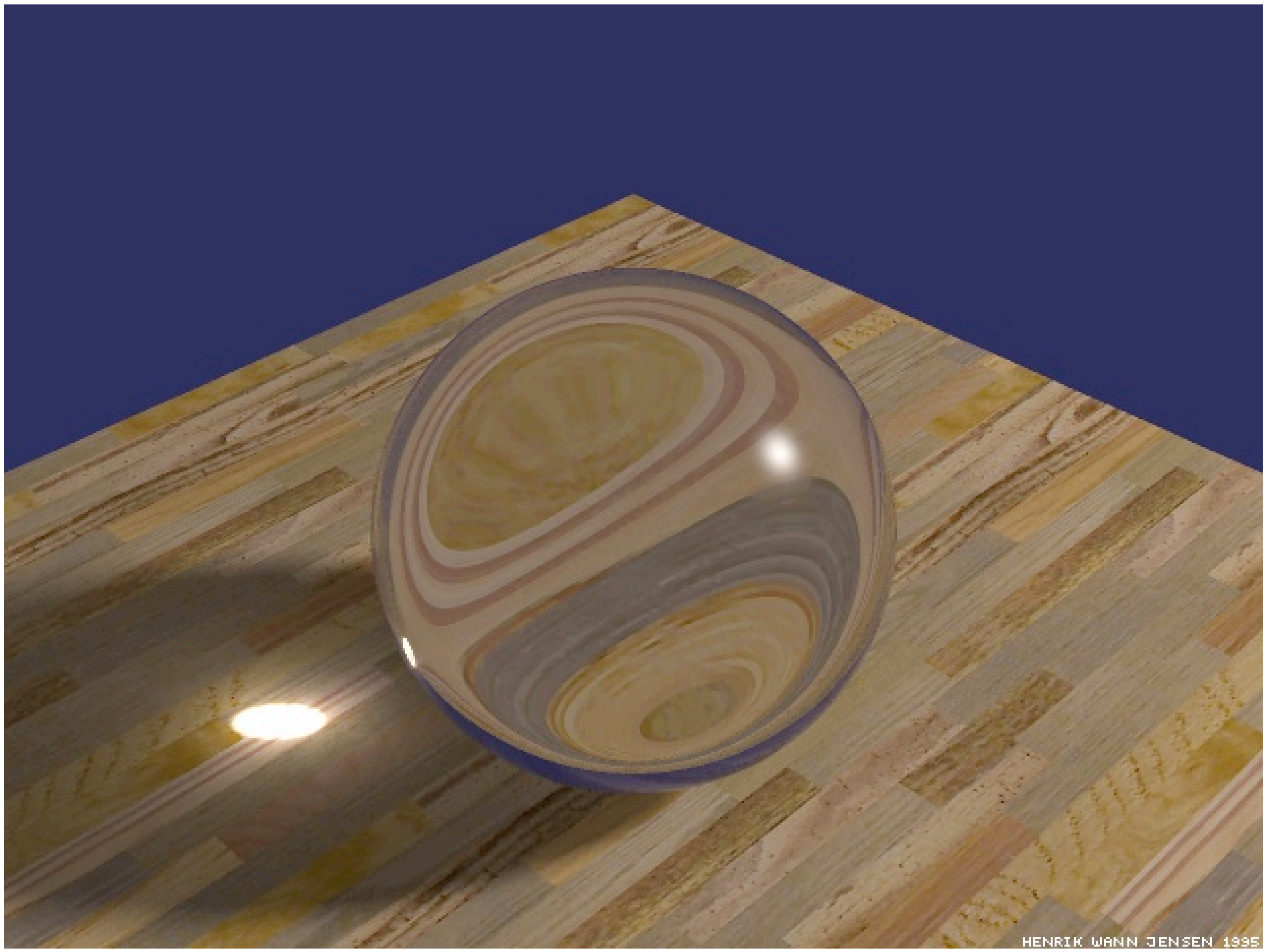
- Caustics:
  - refraction or reflection causes light to be “collected” in some regions.
- Specular-> diffuse transfer
  - source reflected in a mirror
- Can’t render this by tracing rays from the eye - how do they know how to get back to the source?


Light bounces of mirror to diffuse
surface (cannot trace back)
Light refracts to
diffuse surface (also
cannot trace back)
Point Source

Illumination effects (cont)

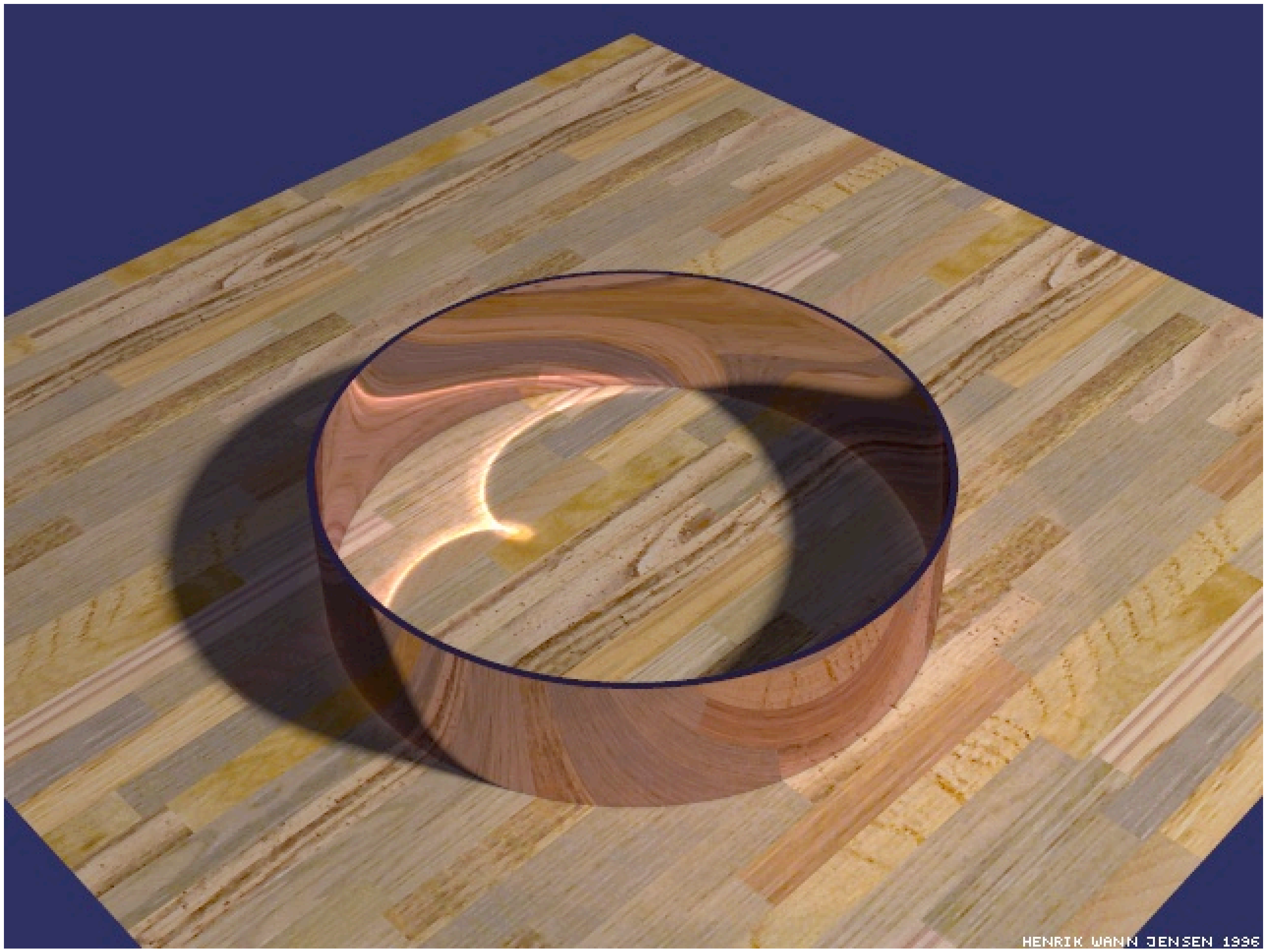
- To get the effect of light reflected and refracted from sources onto diffuse surfaces, we can trace rays **from** the light **to** the first diffuse surface
  - leave a note that illumination has arrived - an illumination map, or photon map
  - sometimes referred to as the forward ray
  - now retrieve 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

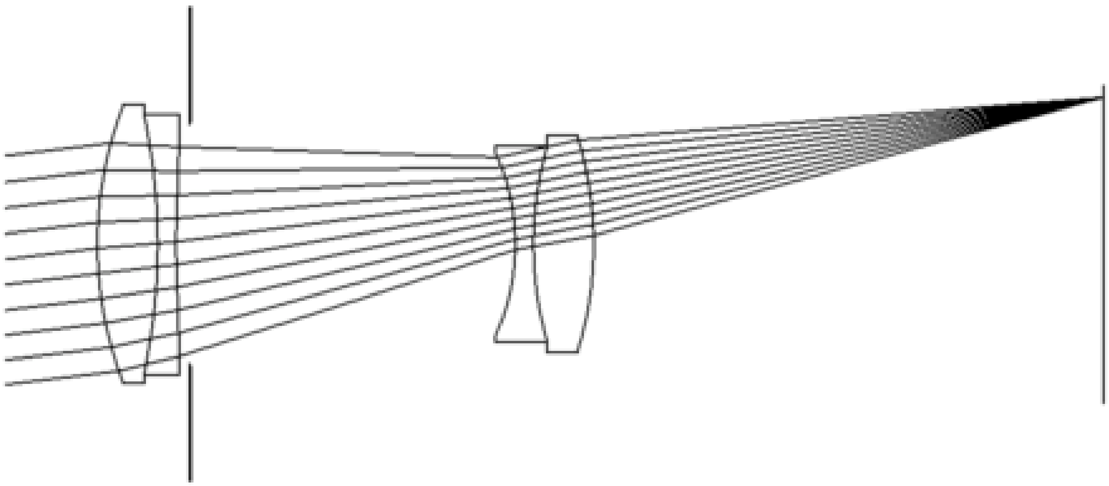
Henrik Jensen, http://www.gk.dtu.dk/~hwj

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

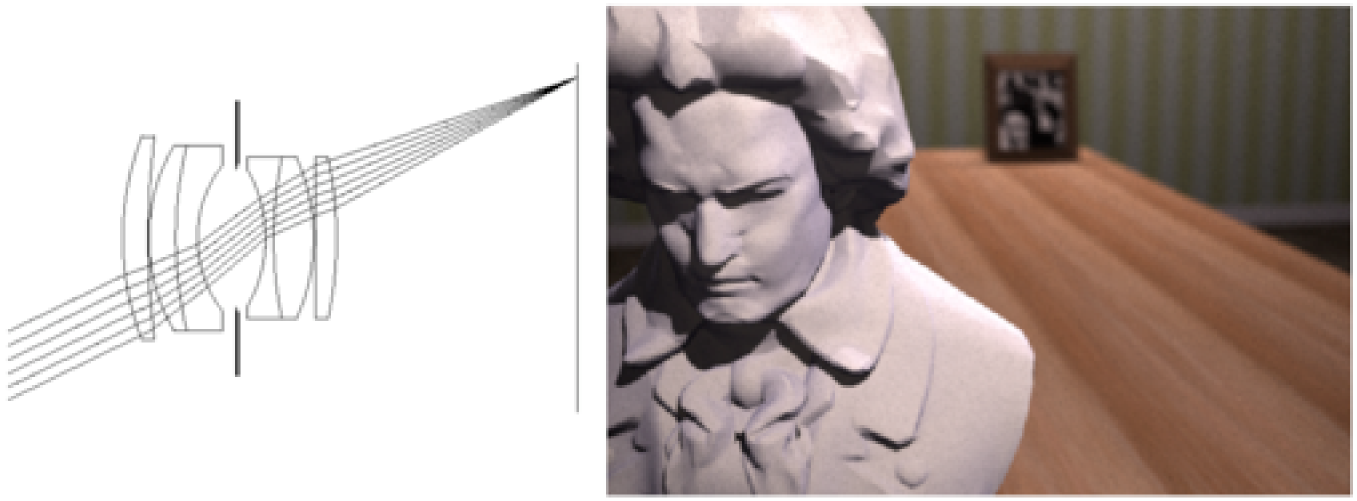
We can go further and introduce a more interesting or realistic camera model

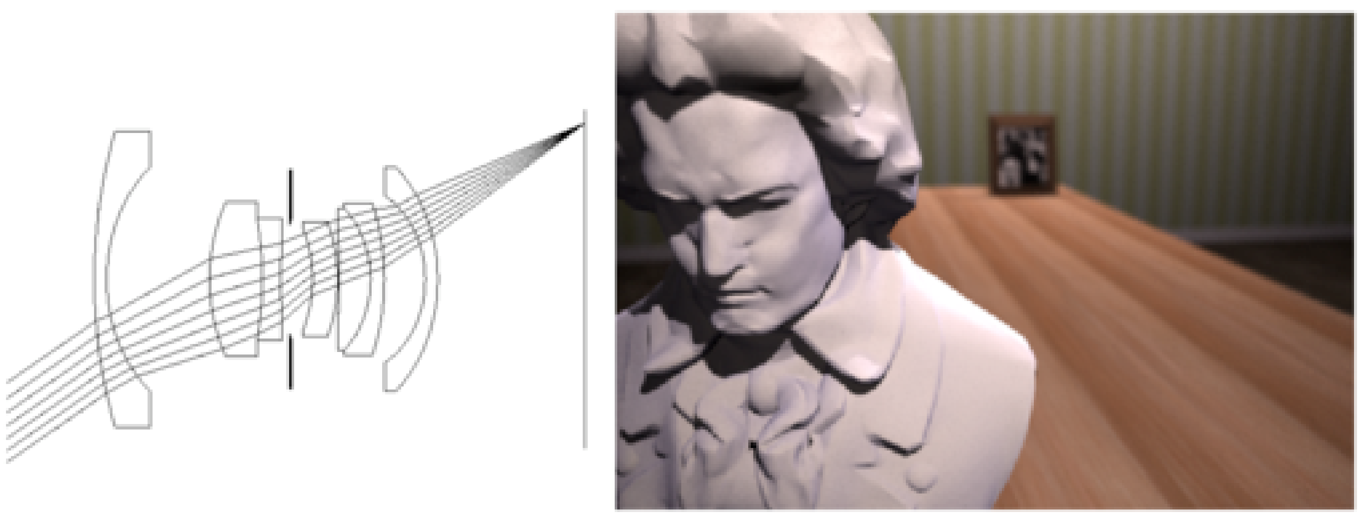
from
A Realistic Camera Model for Computer Graphics
Craig Kolb, Don Mitchell, and Pat Hanrahan
Computer Graphics (Proceedings of SIGGRAPH '95), ACM SIGGRAPH, 1995, pp. 317-324



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