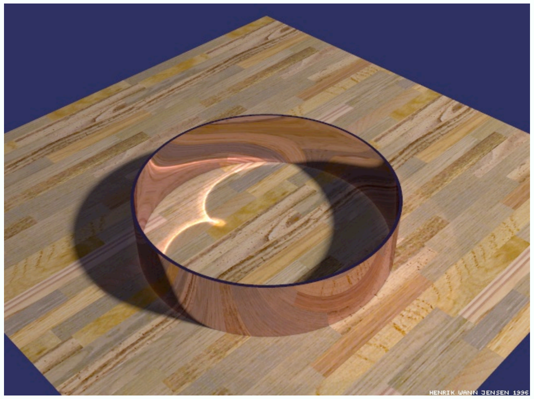
Rendering Area Sources

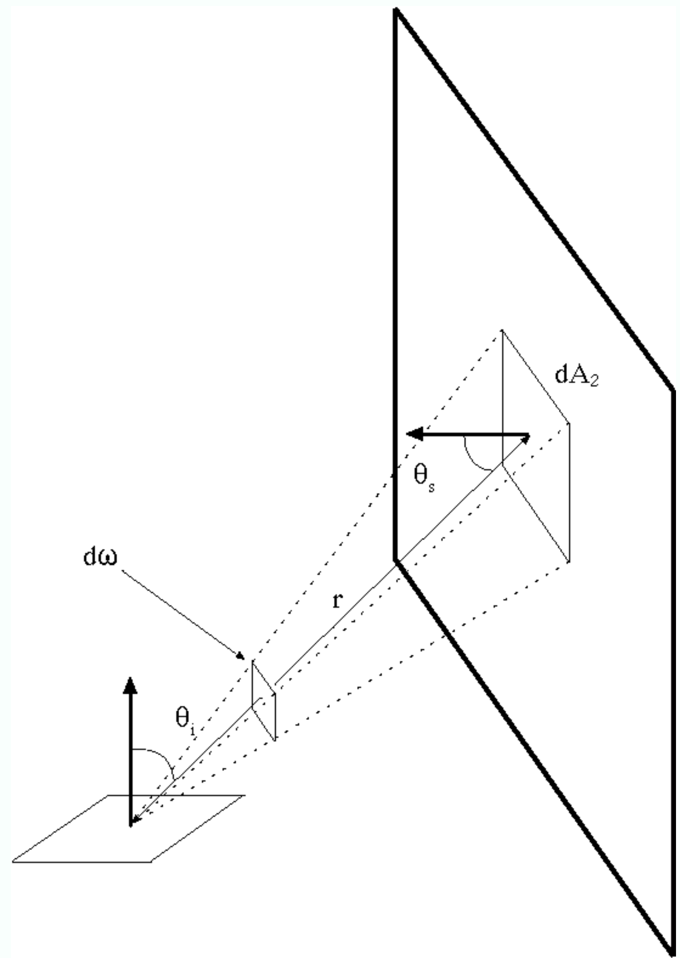
D.A. Forsyth

Point source model is unphysical

- Because
  - imagine source surrounded by
    - big sphere, radius R
    - small sphere, radius r
  - each point on each sphere gets exactly the same “brightness”!
  - This means our unit of brightness either
    - depends on the distance to the source, which is weird
    - does not conserve energy, which is worse
- We do this because, in practice, pictures look OK
  - we’re doing graphics, not physical simulation
  - but better physics will lead to better pictures
    - Area sources
    - Caustics



Area sources



- Examples: diffuser boxes, white walls.
- The intensity at a point due to an area source is obtained by adding up the contribution of source elements

Area Source Shadows

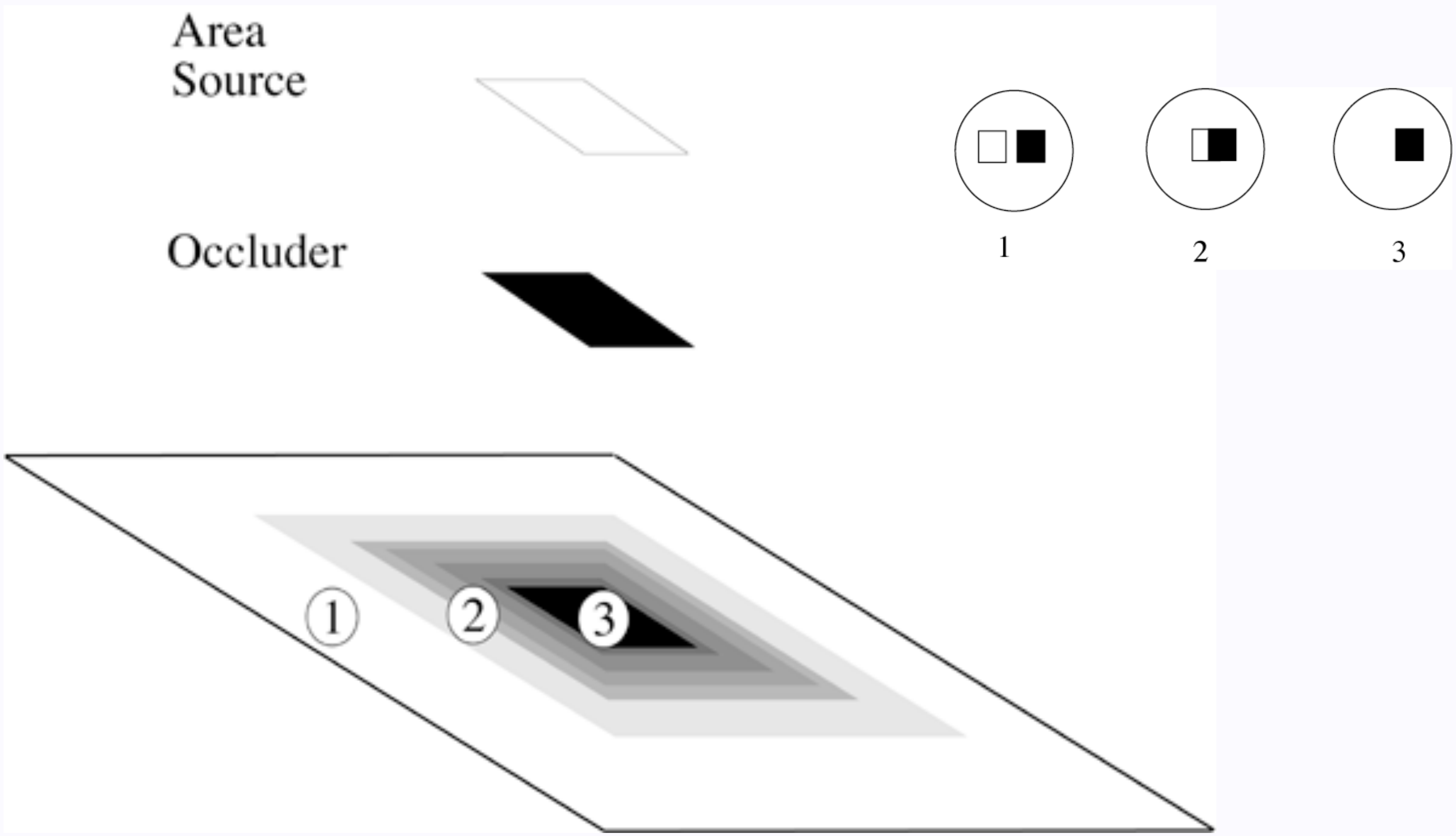


Photo by Reiner Kraft

Terminology

- In the real world, most surfaces have light leaving them
  - otherwise you couldn't see them
- Luminaires
  - surfaces that "produce light"
  - i.e. some (not all) of the light leaving them is internally generated
    - chemistry, physics, etc.
    - eg lightbulb, diffuser box, etc.
  - notice we might model a lampshade as a luminaire
    - even though the light comes from the bulb

Terminology

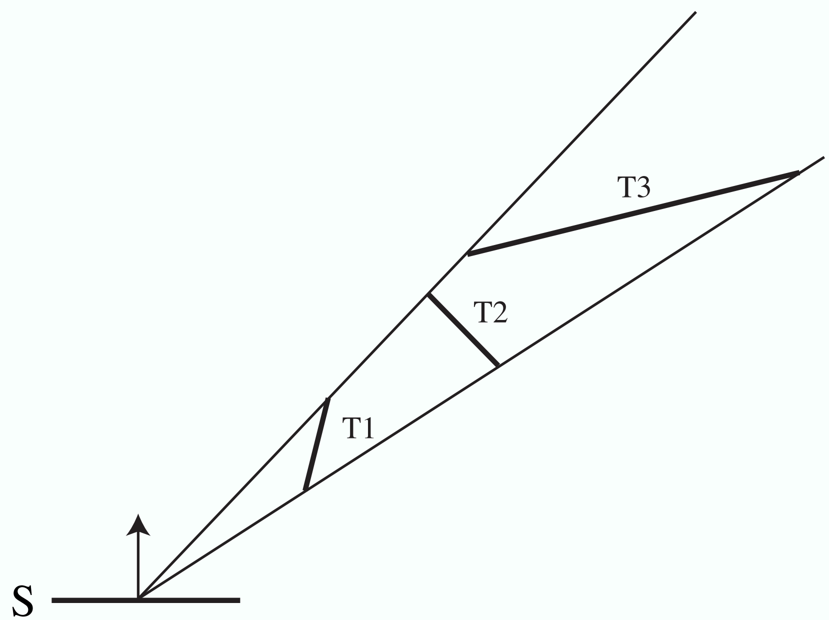
- Radiosity -
  - total power emitted by a surface, per unit area, irrespective of direction
  - contains terms due to reflection and due to emitted light
    - eg diffuser box
  - appropriate unit for describing intensity of diffuse surfaces
  - Usually written B(x)

Terminology

- Exitance
  - total internally generated power emitted by a surface, per unit area, irrespective of direction
  - non-zero only for luminaires
    - things that make light internally
  - appropriate unit for describing intensity of diffuse luminaires
  - Usually written E(x)

Foreshortening

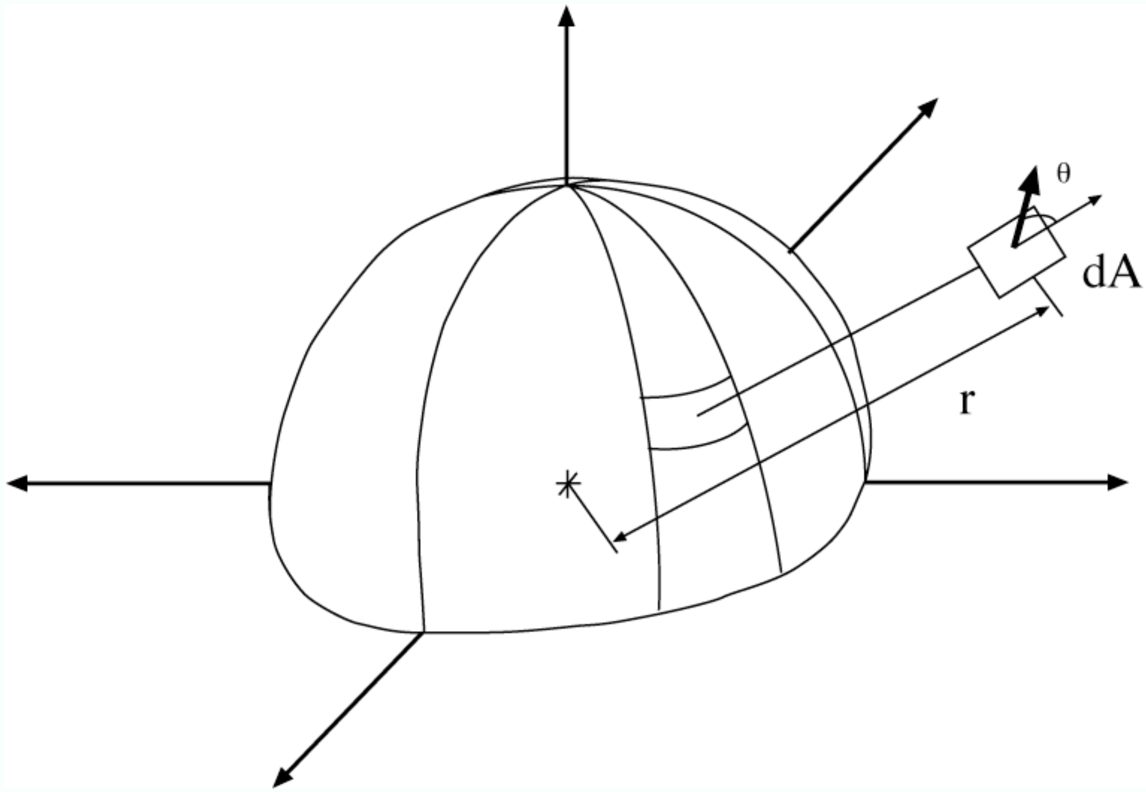
- Imagine two patches, S (source) and T (target)
  - Power=(Energy/Time) travels from source to target
- All S knows about T is
  - what T looks like at S
    - so any target that looks the same at S should get the same power
    - T1, T2, T3, must receive the same total power from S



Solid Angle

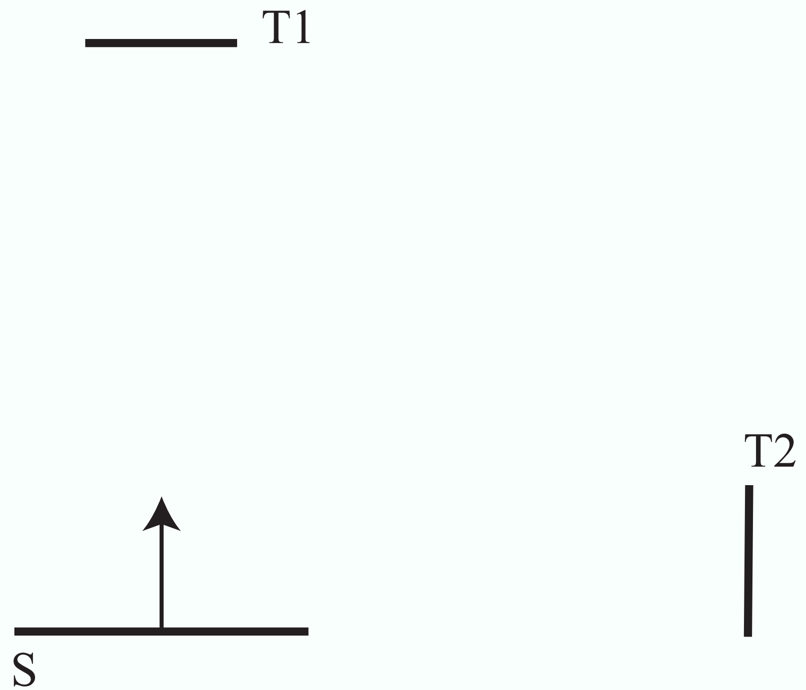
- By analogy with angle (in radians)
- The solid angle subtended by a patch area dA is given by

$$d\omega = \frac{dA\cos\vartheta}{r^2}$$



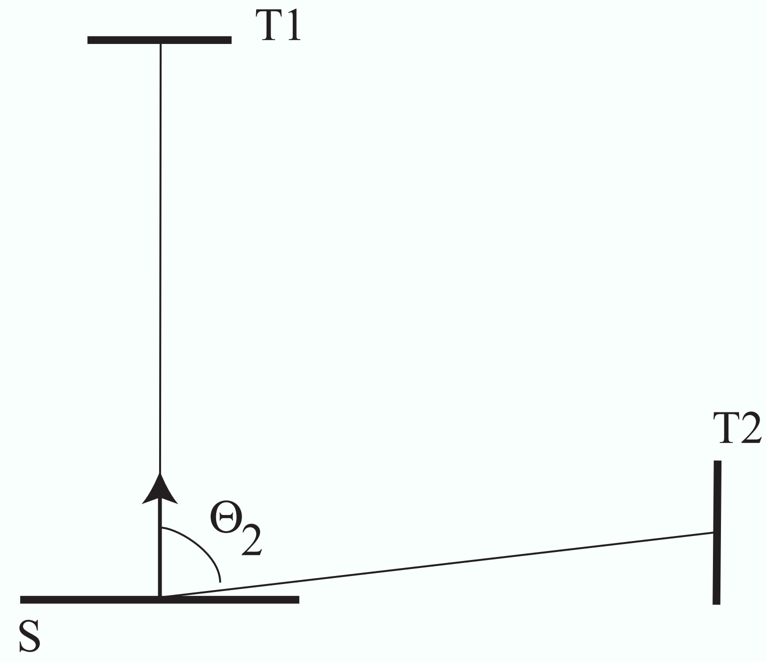
Foreshortening

- Power received from S depends on how big S looks to T
  - in the drawing, T1 sees “big” S, T2 sees “small” s
  - T1 must get more



Apparent area

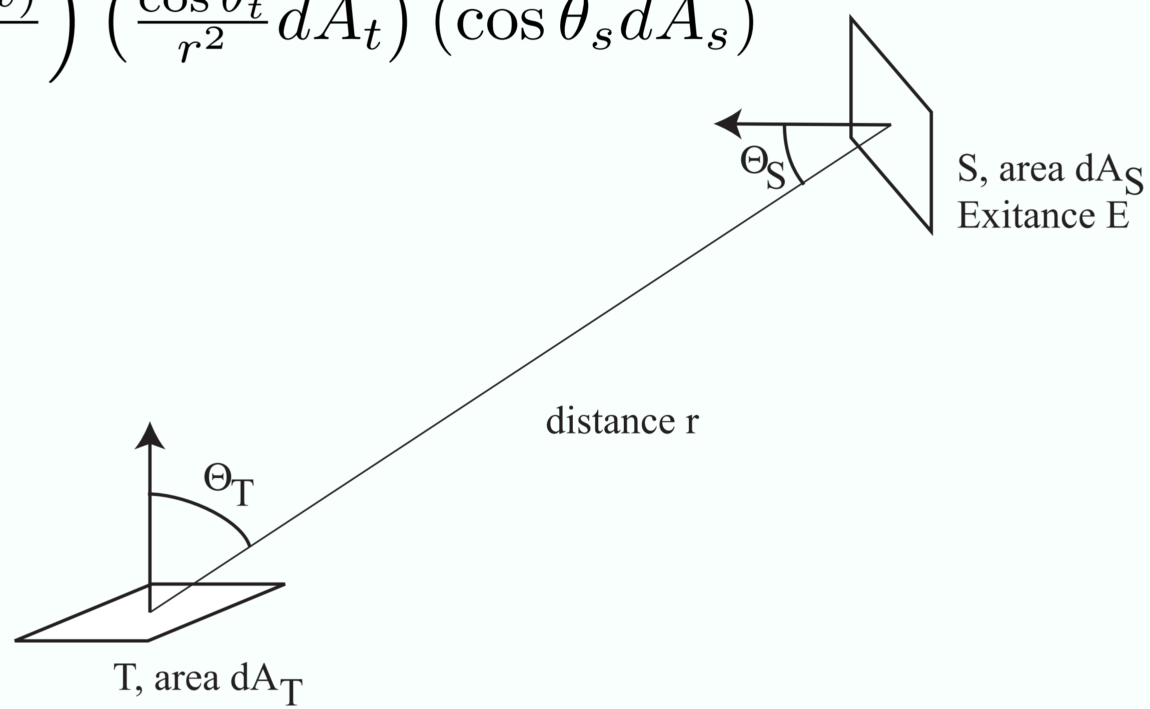
- S has area dA
- T1 sees an S with area $\mathrm{dA}\ \cos\theta_1$
- T2 sees an S with area $\mathrm{dA}\ \cos\theta_2$



Transmitted power

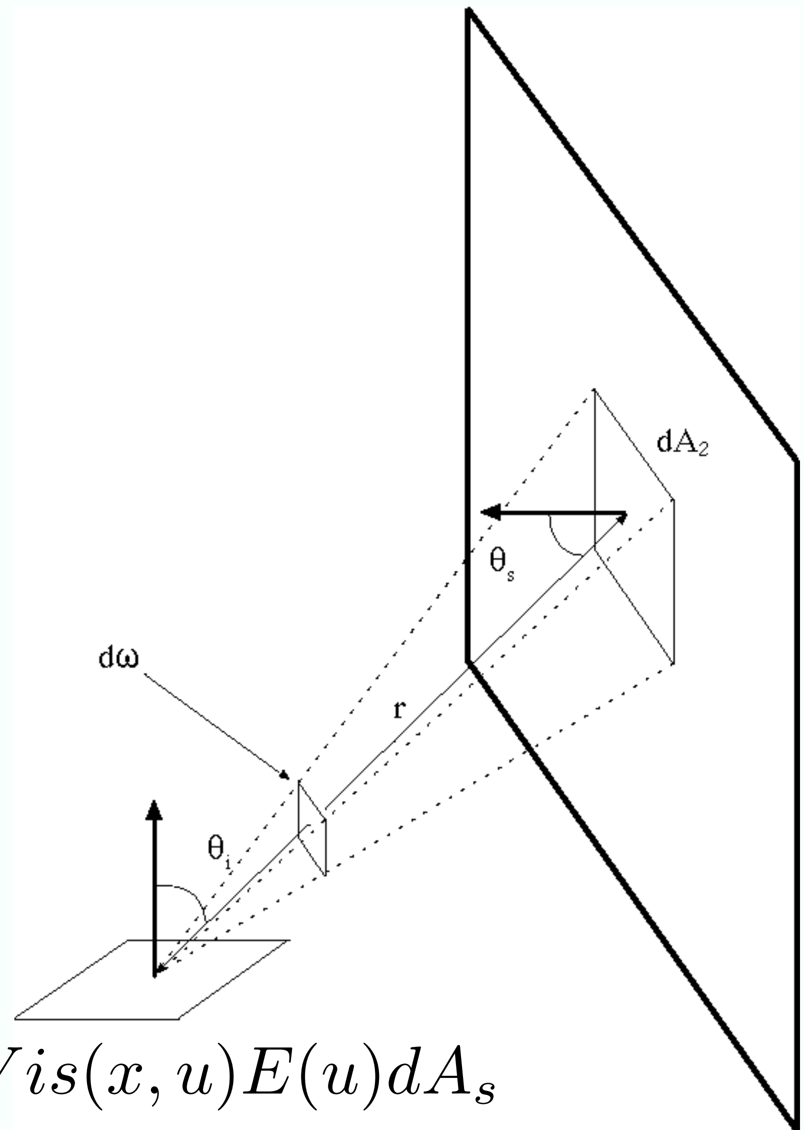
- (power density at S)(solid angle) (apparent area of S)

$$\left(\frac{E(x)}{\pi}\right)\left(\frac{\cos\theta_t}{r^2}dA_t\right)\left(\cos\theta_s dA_s\right)$$



Radiosity due to an area source

- Assume
  - source has B=E
  - receiver has E=0
    - doesn't reflect
  - Vis(x, u)=1 if x can see u
    - 0 otherwise
- Derive later
  - what do we do with it?



$$B(x) = \rho(x) \int_S \frac{\cos\theta_i \cos\theta_s}{\pi r^2} Vis(x,u) E(u) dA_s$$

Natural algorithm for area sources

- Recall:

if $x_i \sim p(x)$

then $\frac{1}{N}\sum_{i=1}^{N} f(x_i) \rightarrow \int f(x)p(x)dx$

$$B(x) = \rho(x)\int_S \frac{\cos\theta_i \cos\theta_s}{\pi r^2} Vis(x,u)E(u)dA_s$$

- Draw random samples u_i on source.
- Let p(u) be uniform on the area source
  - p(u)=1/A
- then use:

$$\rho(x)\frac{A}{N}\sum_i \frac{\cos\theta_i \cos\theta_s}{\pi r^2} Vis(x,u_i)E(u_i)$$

Shading with an area source

- Now we compute the diffuse term at each point by
  - cast N rays to area source
  - for each ray, test shadow (this is Vis(x, u))
  - compute average
- Options, comments
  - shadow cache, ray speedup becomes a big deal
  - different random points on area source for each shading point
  - same points on area source for each shading point
    - noise is now correlated, but slightly faster
  - stratified sampling of area source

Texture maps and ray tracing

- Simple cases first
  - Assume surface is parametric
  - intersection point yields parameters
  - query texture map at parameter points for texture value
- Issue:
  - curved surface distorts texture map

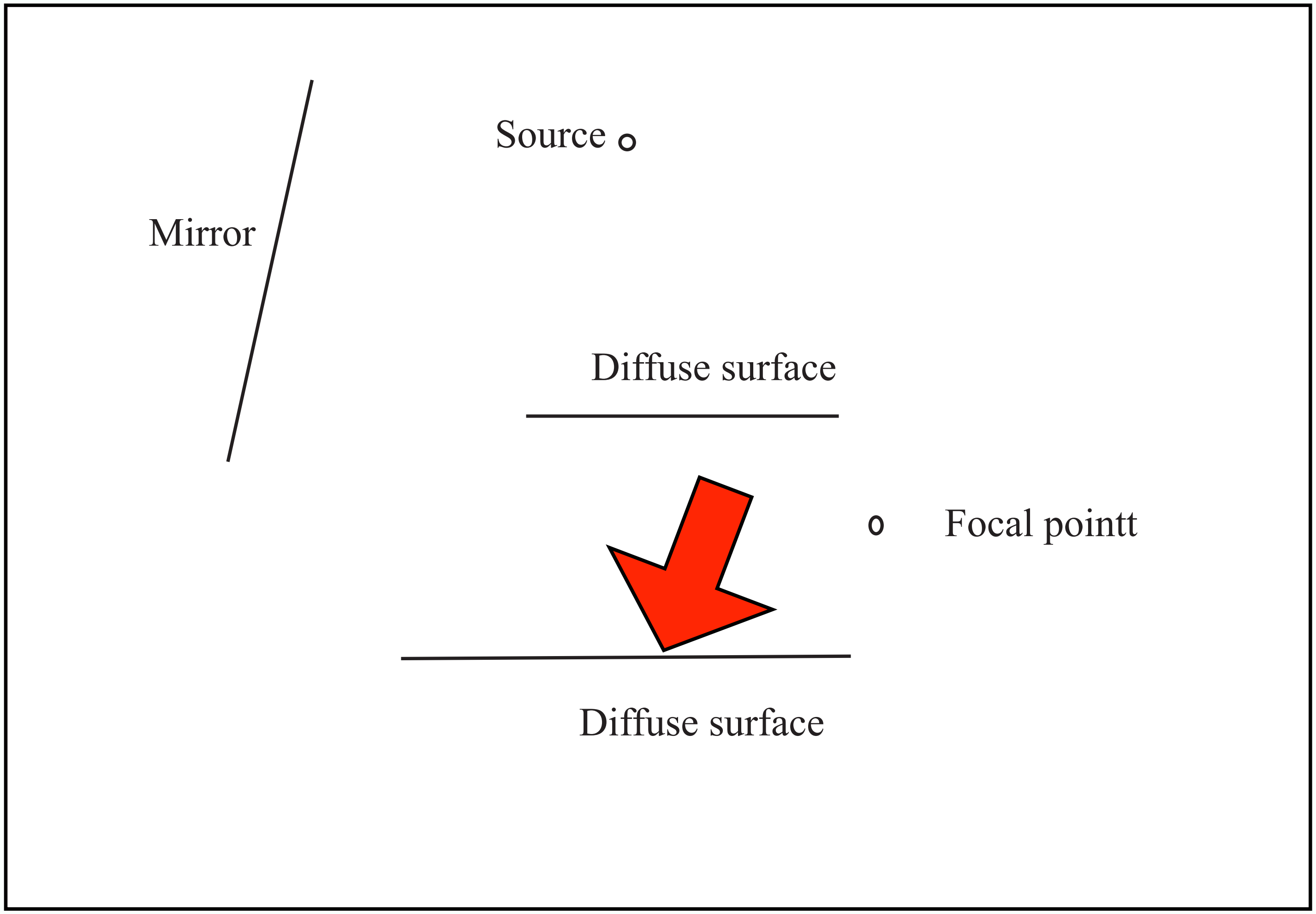
Texture maps for triangles

- Triangle has 3 vertices
  - parametric form:
    - s (p1-p0)+ t(p2-p0)+p2
    - procedure:
      - find intersection point
      - compute s, t by solving linear system
      - albedo at this point is albedo map (s, t)
- Texture map distorts
  - we can fix this if we know q0, q1, q2
    - coordinates of triangle on texture map plane

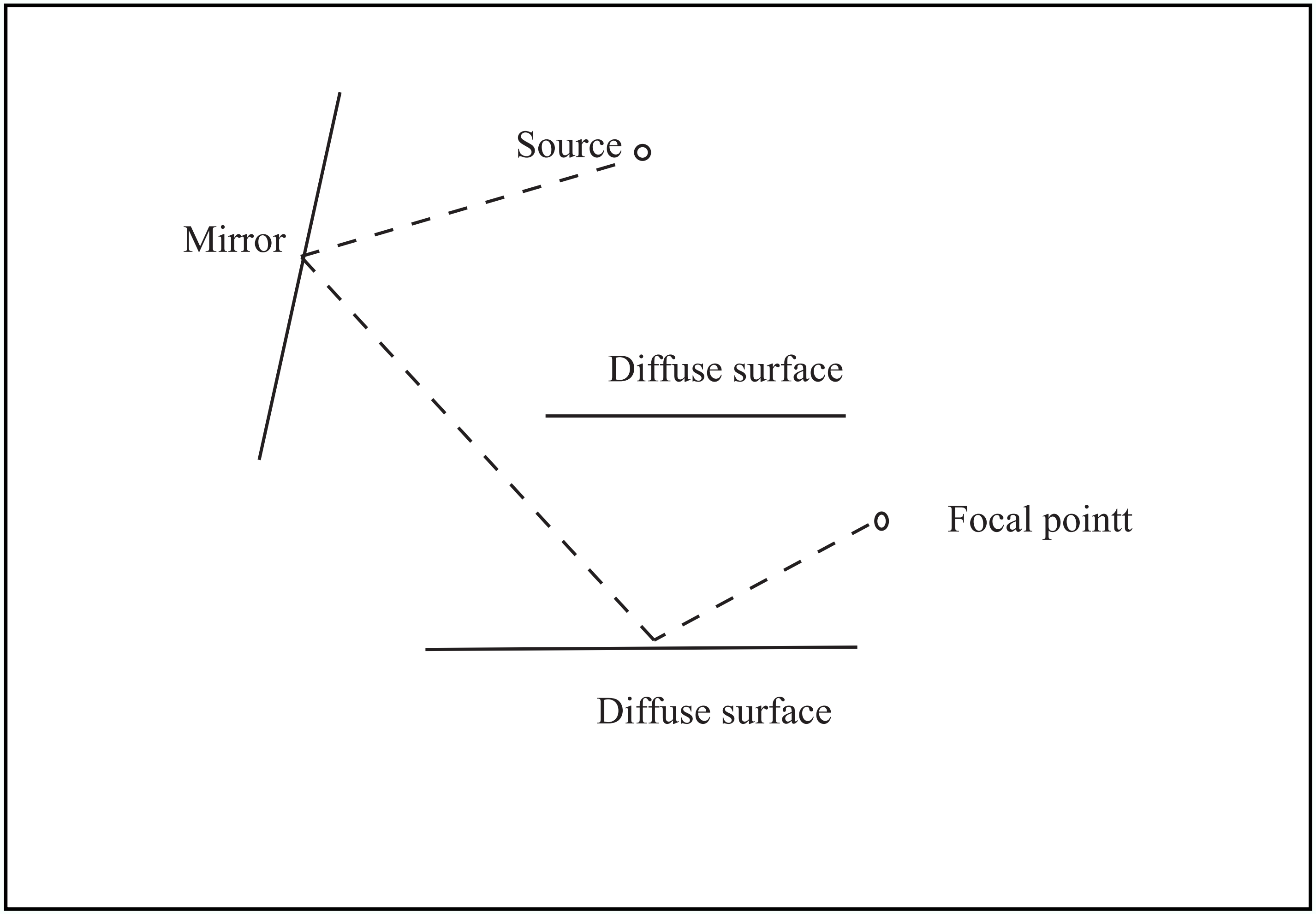
Texture maps for spheres

- Parametric sphere
  - R(cos s cos t, cos s sin t, sin s)
  - We need to know
    - where the north pole is
      - yields s from intersection point
    - where some point on the equator is
      - yields t from intersection point
  - this s, t map distorts area
      - adjust map to account for distortions

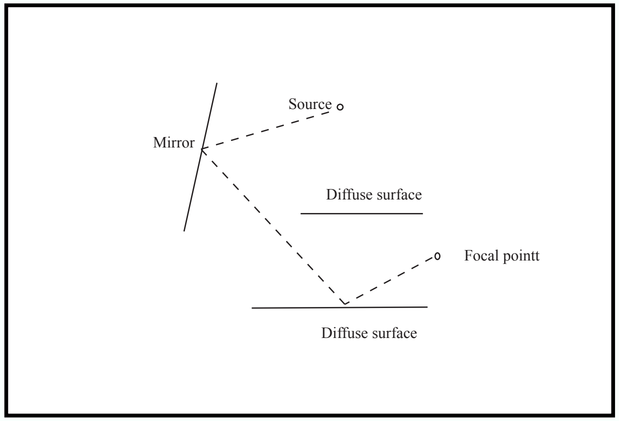
Specular - diffuse transfer



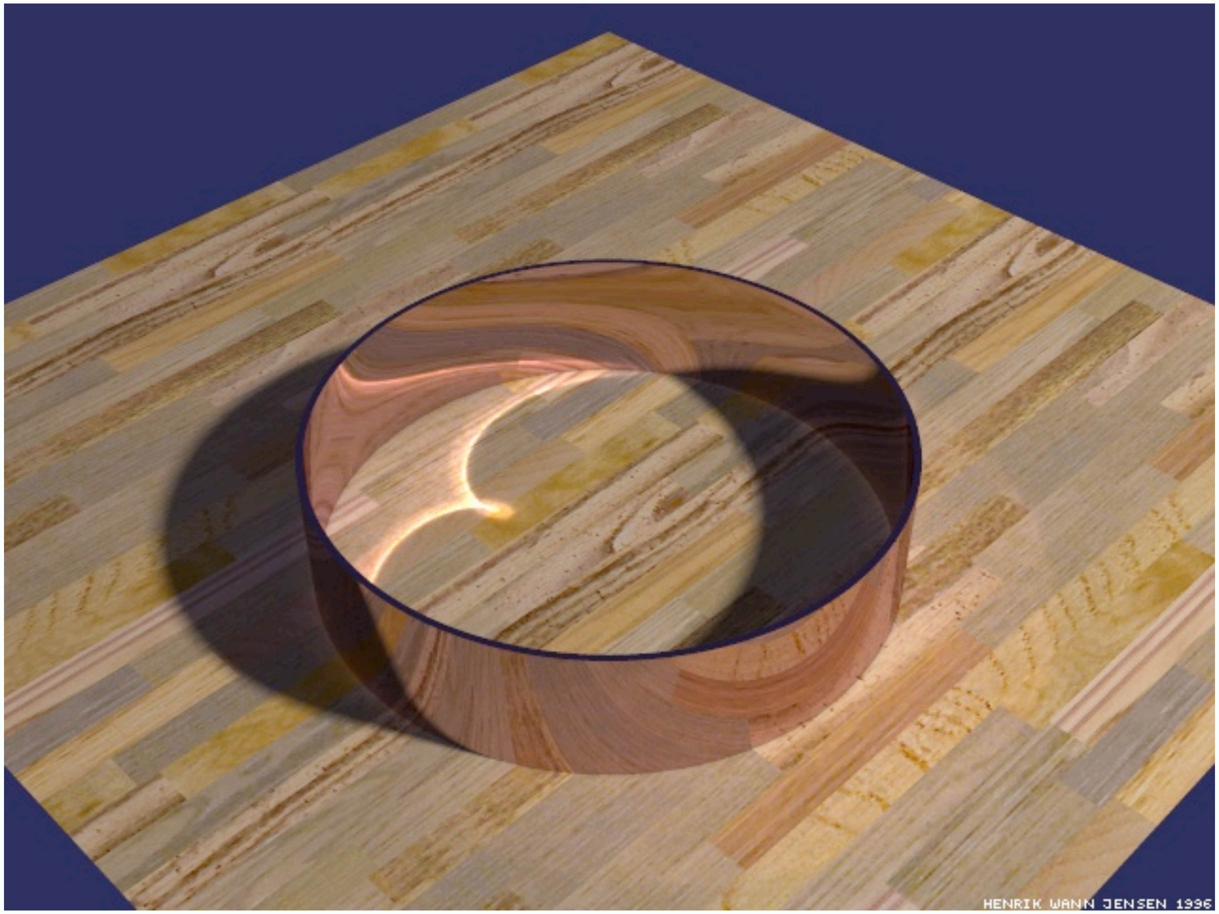
Specular-diffuse transfer



Specular-diffuse transfer

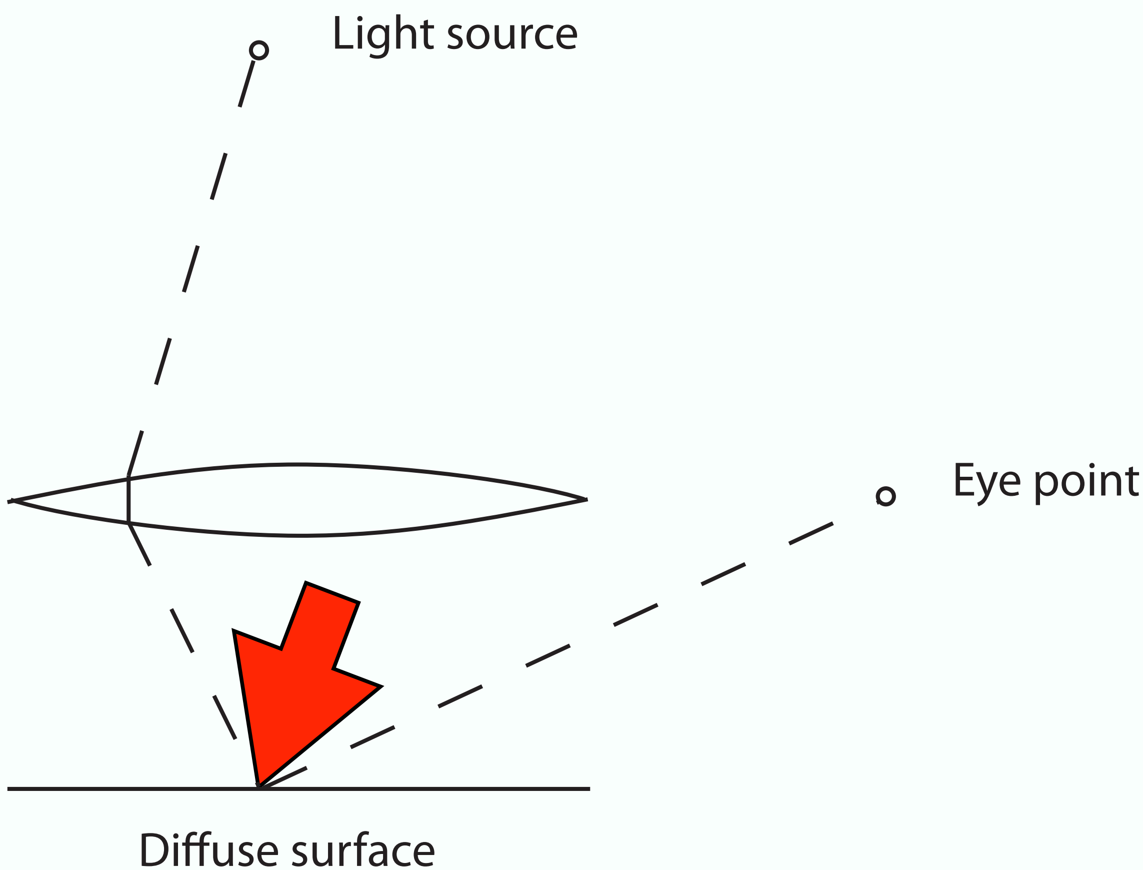


- Hard to render, because it is hard to find the ray leaving the patch that finds the light source

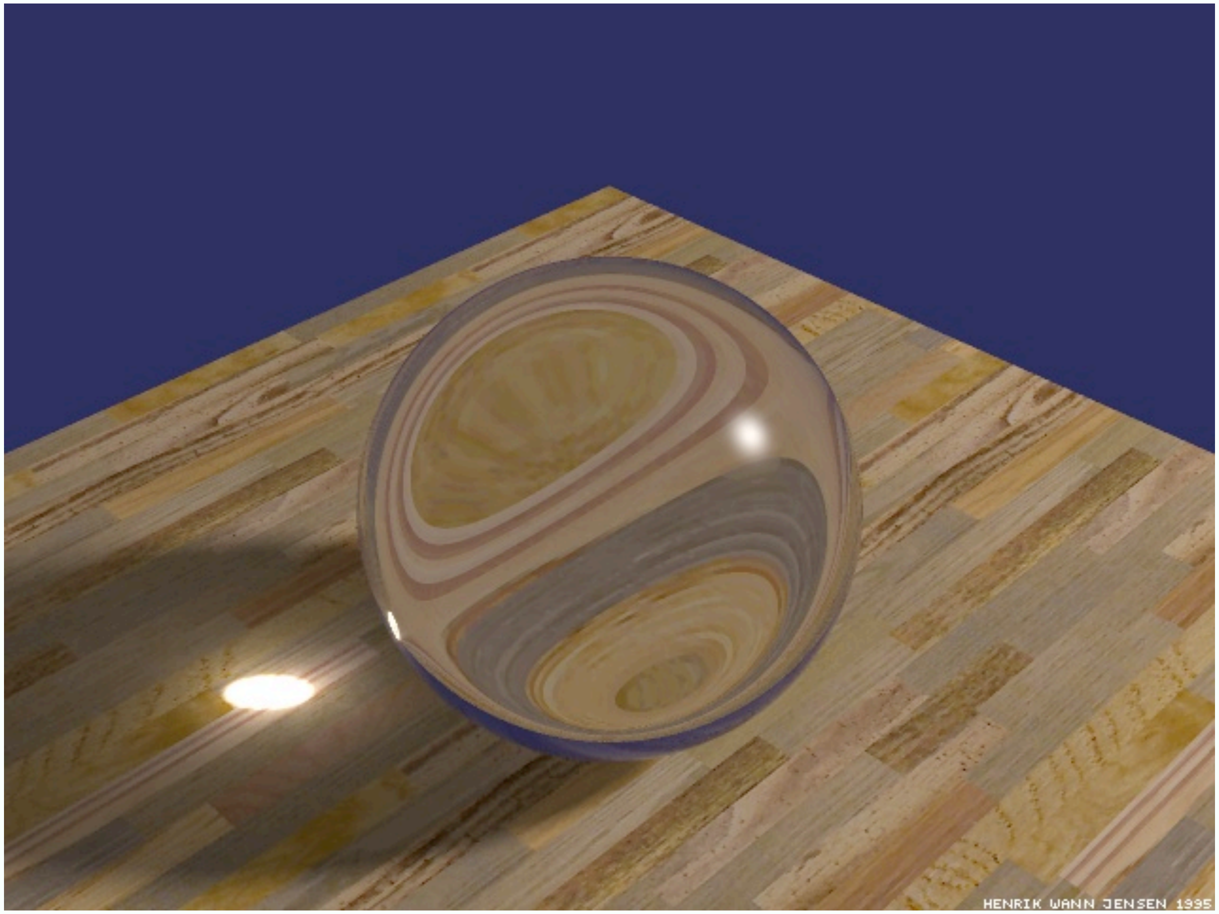


Specular-diffuse transfer creates important effects;
curved surfaces can collect light into caustics

Transmissive - diffuse transfer

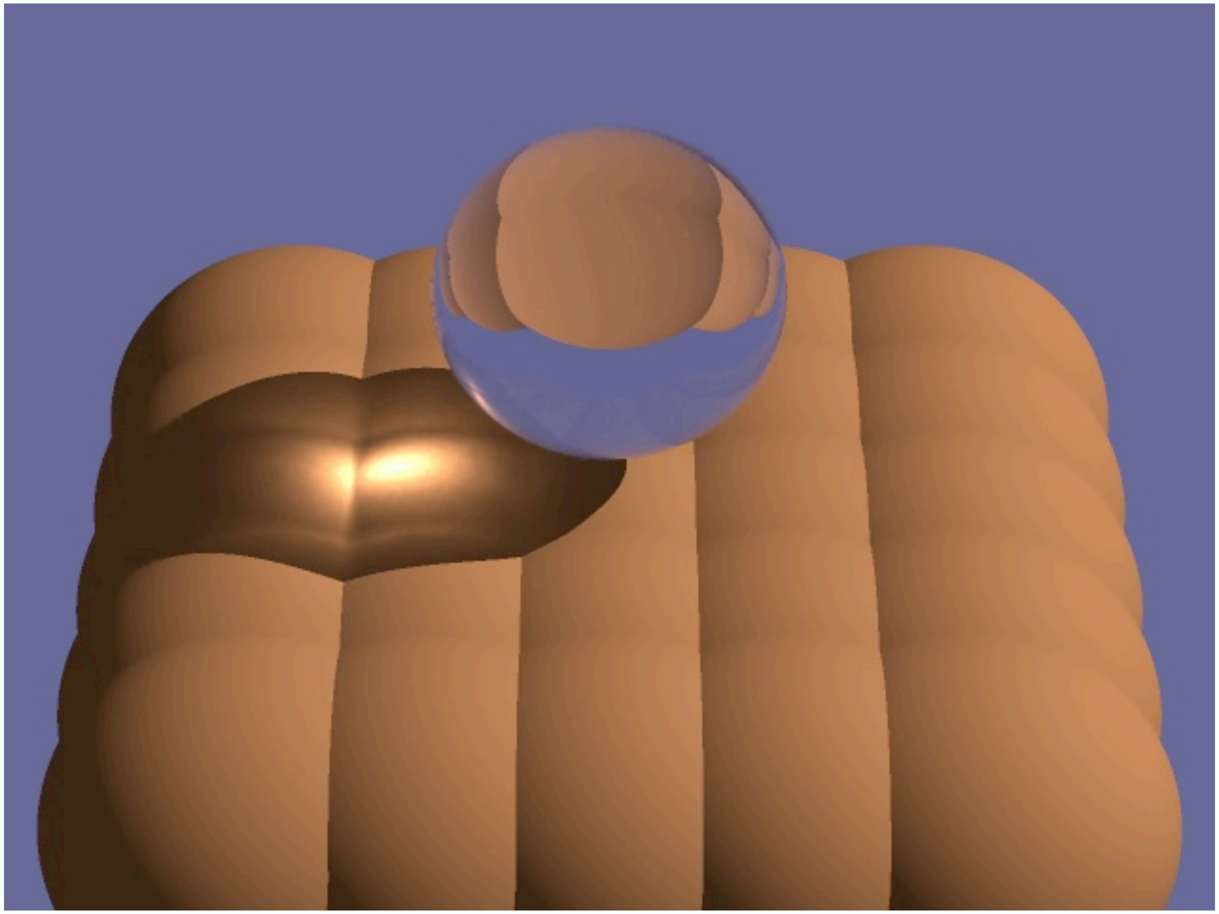


Again, hard to find the ray leaving the patch that finds the light source

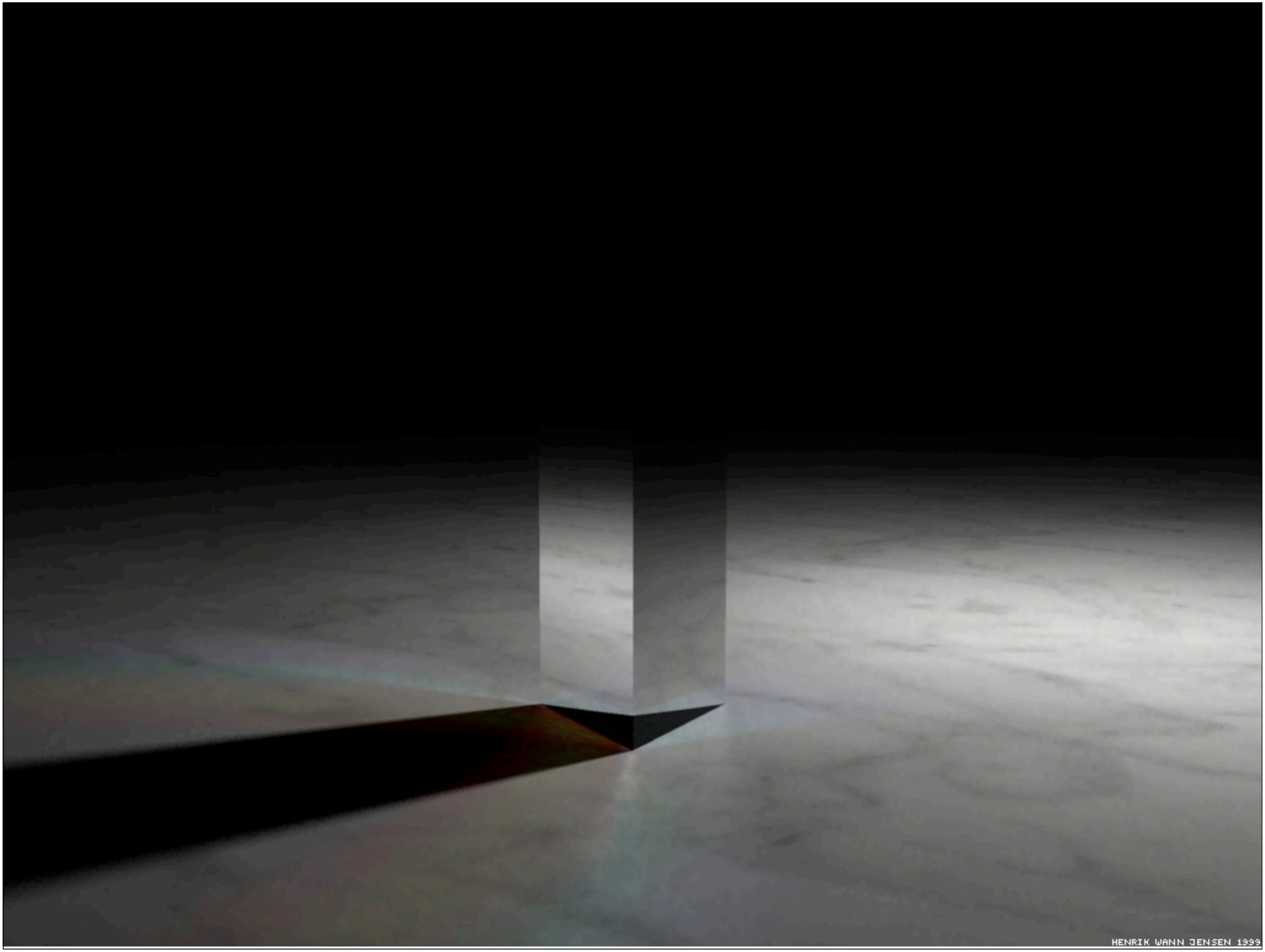


Transmissive-diffuse transfer creates important effects;
curved surfaces can collect light into caustics

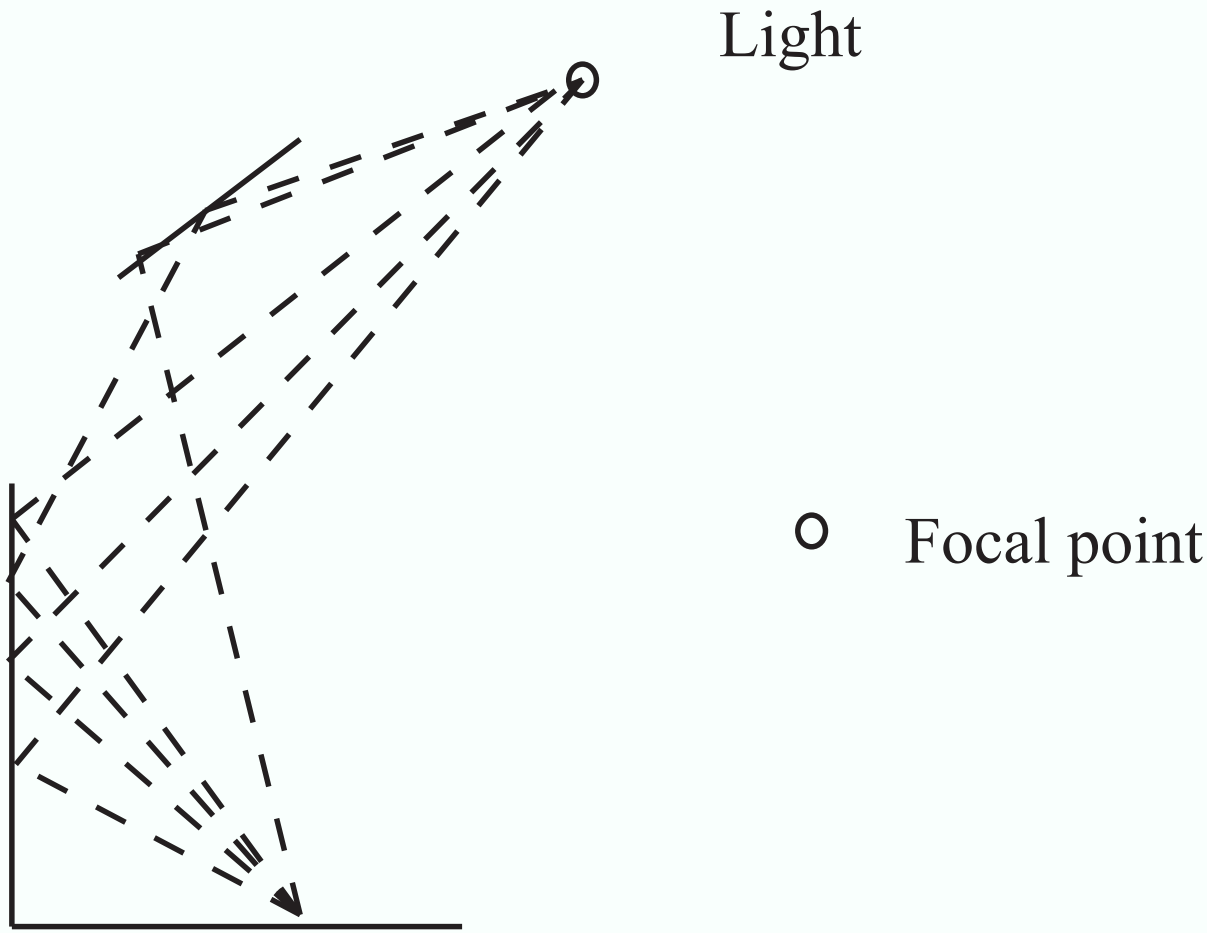
Transmissive-diffuse transfer creates important effects;
curved surfaces can collect light into caustics



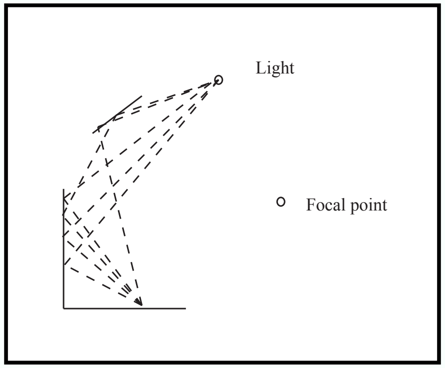
Trasmissive - diffuse transfer creates important effects;
curved surfaces can collect light into caustics


HENRIK WANN JENSEN 1999

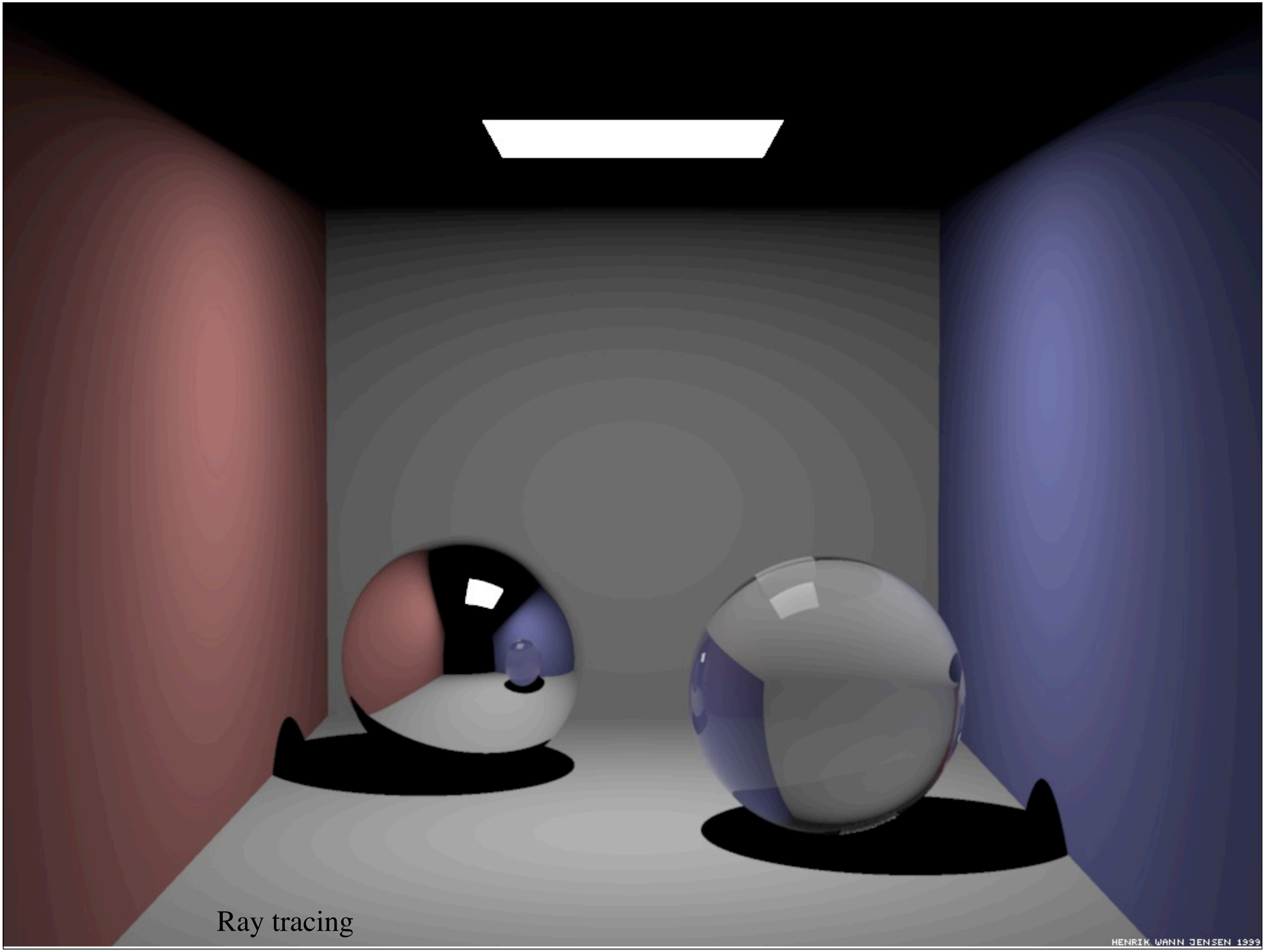
Diffuse-diffuse transfer



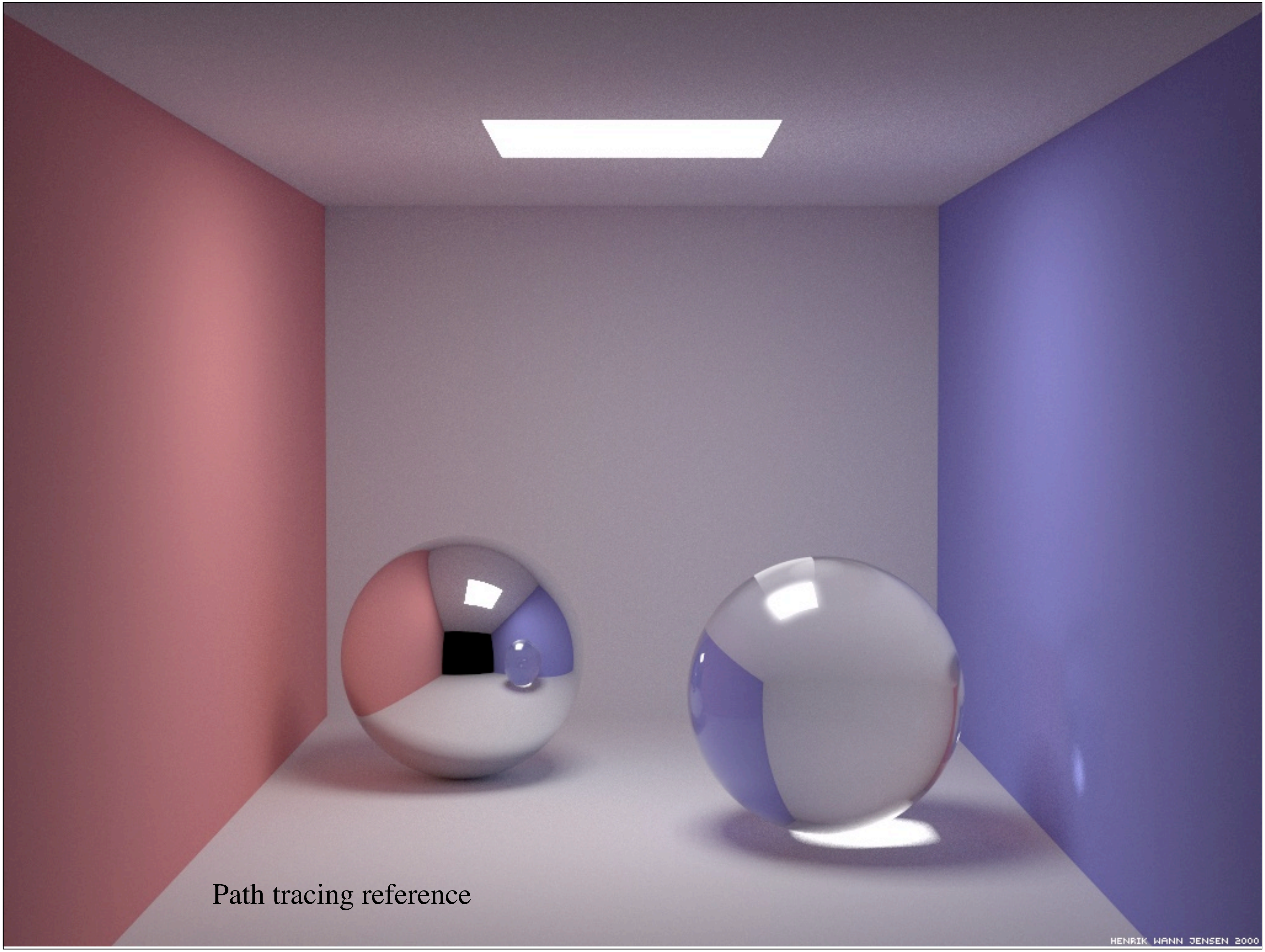
Diffuse-diffuse transfer



- Again, hard to render because
  - many paths are important, even more are not
    - most do not reach the light
  - we don't know how to find the important ones



Ray tracing



Path tracing reference

Rendering specular-diffuse transfer

- We must now cast rays from the light
  - these rays undergo specular reflection
    - but we don't compute shading - these rays model photons
  - when they arrive at a diffuse surface, they stop and leave a record
    - for the moment, in a map like a texture map
- Then cast rays from the eye
  - shading computation:
    - old shading computation + contribution from map
- Practical questions
  - which rays do we cast from the light?
    - (towards specular objects)