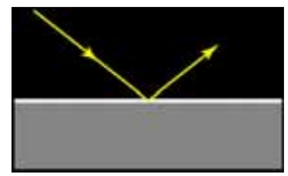
Dielectrics and Distribution in Ray Tracing

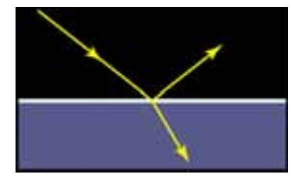
CS 465 Lecture 22

Specular reflection

- Smooth surfaces of pure materials have ideal specular reflection (said this before)
  - Metals (conductors) and dielectrics (insulators) behave differently
- Reflectance (fraction of light reflected) depends on angle



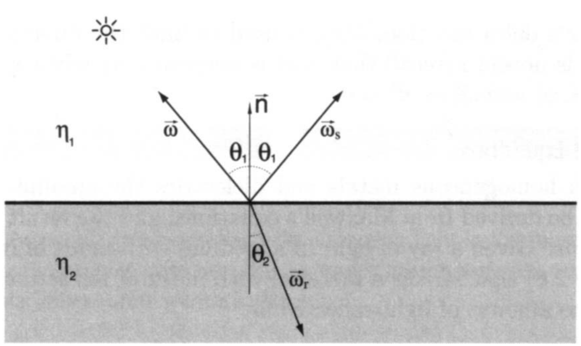
metal



dielectric

Snell's Law

- Tells us where the refracted ray goes



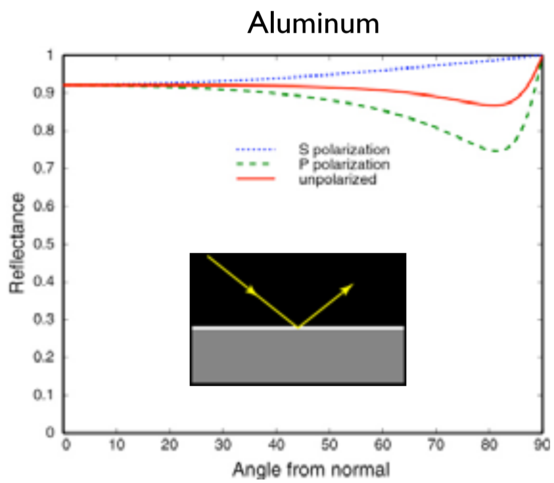
$$\eta_1 \sin\theta_1 = \eta_2 \sin\theta_2$$

Ray tracing dielectrics

- Like a simple mirror surface, use recursive ray tracing
- But we need two rays
  - One reflects off the surface (same as mirror ray)
  - The other crosses the surface (computed using Snell's law)
    - Doesn't always exist (total internal reflection)
- Splitting into two rays, recursively, creates a ray tree
  - Very many rays are traced per viewing ray
  - Ways to prune the tree
    - Limit on ray depth
    - Limit on ray attenuation

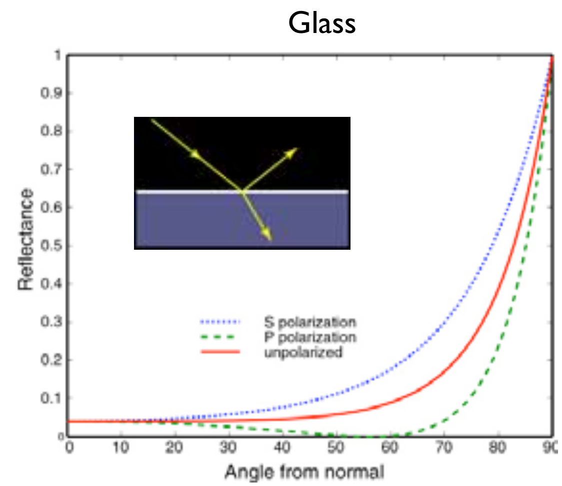
Specular reflection from metal

- Reflectance does depend on angle
  - but not much
  - safely ignored in basic rendering



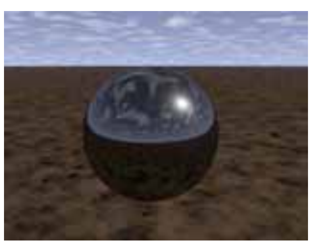
Specular reflection from glass

- Dependence on angle is dramatic!
  - about 4% at normal incidence
  - always 100% at grazing
  - remaining light is transmitted
- This is important for proper appearance

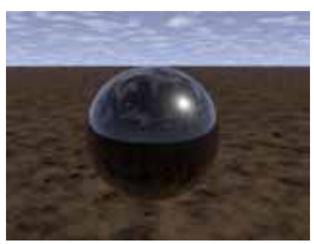


Fresnel reflection

- Black glazed sphere
  - reflection from glass surface
  - transmitted ray is discarded



constant reflectance



Fresnel reflectance

Fresnel's formulas

- They predict how much light reflects from a smooth interface between two materials
  - usually one material is empty space

$$F_p = \frac{\eta_2 \cos\theta_1 - \eta_1 \cos\theta_2}{\eta_2 \cos\theta_1 + \eta_1 \cos\theta_2}$$

$$F_s = \frac{\eta_1 \cos\theta_1 - \eta_2 \cos\theta_2}{\eta_1 \cos\theta_1 + \eta_2 \cos\theta_2}$$

$$R = \frac{1}{2}\left(F_p^2 + F_s^2\right)$$

  - R is the fraction that is reflected
  - $(1 - R)$ is the fraction that is transmitted

Schlick's approximation

- For graphics, a quick hack to get close with less computation:

$$\tilde{R} = R_0 + (1 - R_0)(1 - \cos\theta)^5$$

- R_0 is easy to compute:

$$F_p = \frac{\eta_2 - \eta_1}{\eta_2 + \eta_1}$$

$$F_s = \frac{\eta_1 - \eta_2}{\eta_1 + \eta_2}$$

$$R_0 = \left(\frac{\eta_2 - \eta_1}{\eta_2 + \eta_1}\right)^2$$

[Josh Wills | 2003 UCSD Rendering Competition]

Fresnel reflection

[Mike Hill & Gaain Kwan | Stanford cs348 competition 2001]

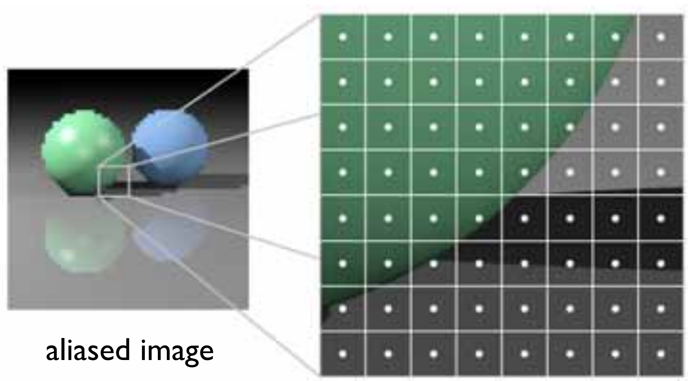
Basic ray tracing

- Many advanced methods build on the basic ray tracing paradigm
- Basic ray tracer: one sample for everything
  - one ray per pixel
  - one shadow ray for every point light
  - one reflection ray, possibly one refraction ray, per intersection

Making use of multiple samples

- Most advanced methods for ray tracing use multiple rays to gather more information
- Simplest example: image-plane supersampling
  - recall discussion of antialiasing
  - ray tracing gives you direct access to the continuous image
  - we know point sampling causes aliasing, and we want area sampling (box filtering being the simplest example)
  - simulate area sampling by averaging many sub-pixel samples

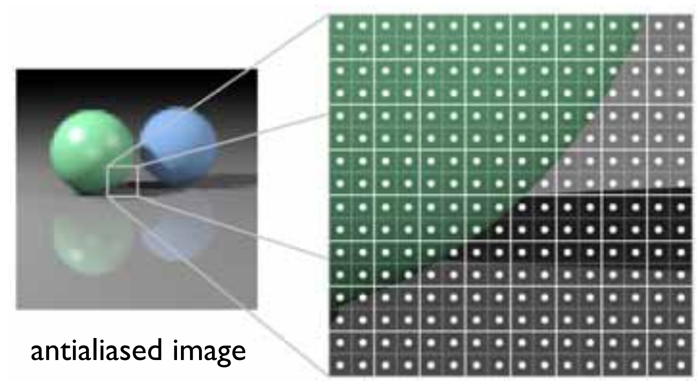
Antialiasing in ray tracing



aliased image

one sample per pixel

Antialiasing in ray tracing



antialiased image

four samples per pixel

Blurring to improve images

- Antialiasing is one example of blurring to improve images
  - problem: sharp edges in the image
  - solution: blur in image space
- This is one example of the general case of discontinuities in the simple model that do not look right in images
  - perfectly sharp shadow edges
  - perfectly clear mirror reflections
  - infinitely tiny point camera
  - perfectly frozen instant in time (in animation)

Basic ray traced image

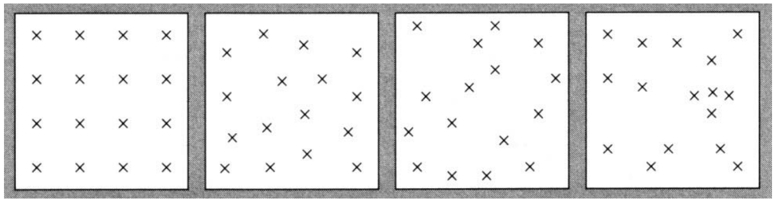
[Glassner 89]

Making use of multiple samples

- Once we're tracing multiple rays per pixel to blur in image space (antialiasing), we can use the multiple samples to blur in other domains
  - blur shadows with distributed shadow rays
  - blur reflections with distributed reflection rays
  - camera lens blur with distributed eye points
  - motion blur with distributed ray times

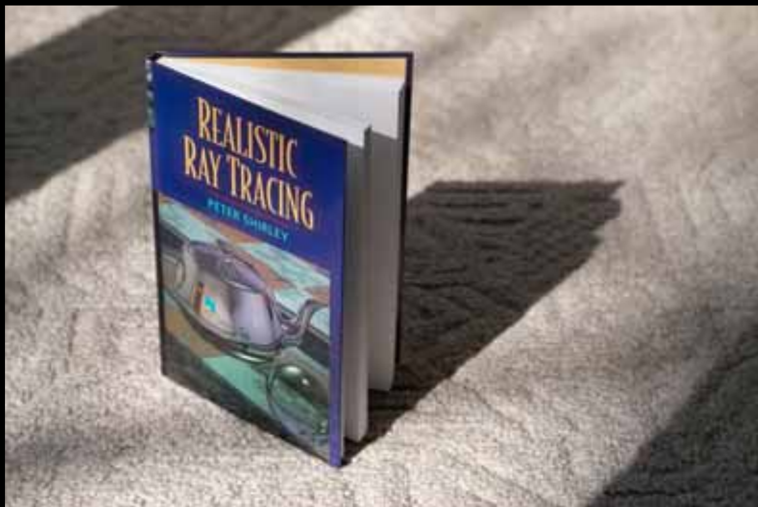
Adding randomness

- Regular sampling, even at sub-pixel spacing, can still lead to artifacts
- Randomness lets you replace artifacts with noise, which is often less bothersome.
- In antialiasing, jitter the sample positions randomly in image space

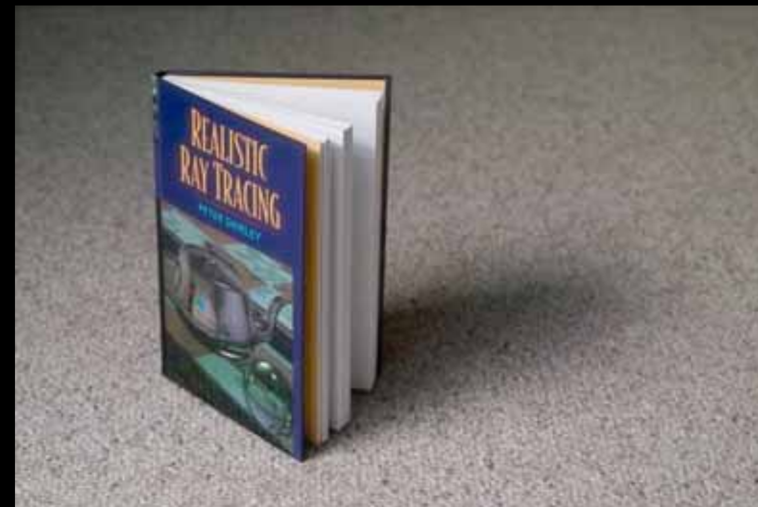


[Shirley 2002]

Hard shadows

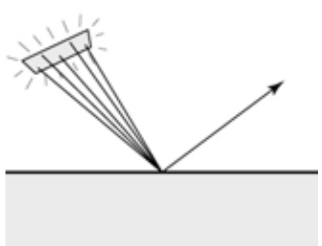


Soft shadows



Creating soft shadows

- Hard shadows caused by point lights
- For soft shadows use area lights
  - and each shadow ray gets a different point on the light
- Choosing samples
  - general principle: start with uniform in square

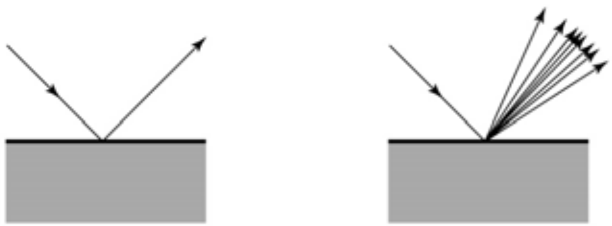


Glossy reflection

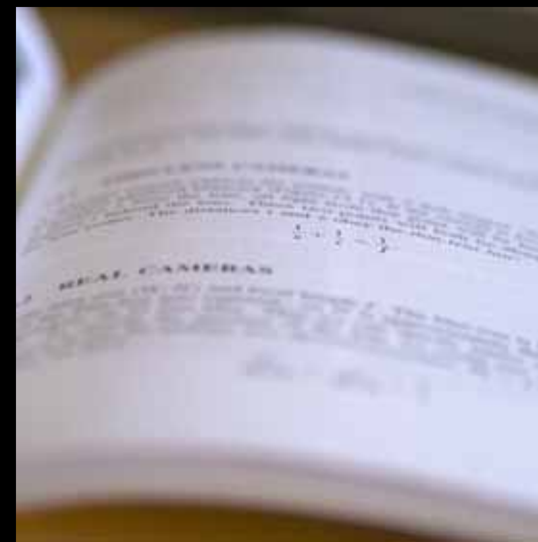
[Lafortune et al. 97]

Creating glossy reflections

- Cause: not all “mirror” reflected rays go the same way
- Jitter the reflected rays
  - Not exactly in mirror direction; add a random offset
  - Can work out math to match Phong exactly
  - Can do this by jittering the normal if you want

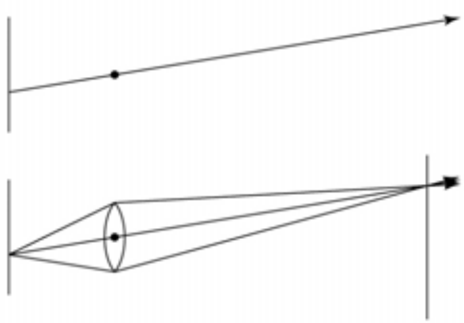


Depth of field



Depth of field

- In reality comes from averaging over the lens
  - how camera optics work
- In rendering, make eye rays come from random points
  - always going toward a point on the focus plane



Motion blur

- Caused by finite shutter times
  - strobing without blur
- Introduce time as a variable throughout the system
  - object are hit by rays according to their position at a given time
- Then generate rays with times distributed over shutter interval

Cook, Porter, Carpenter 1984