

Light Reflection and Advanced Shading

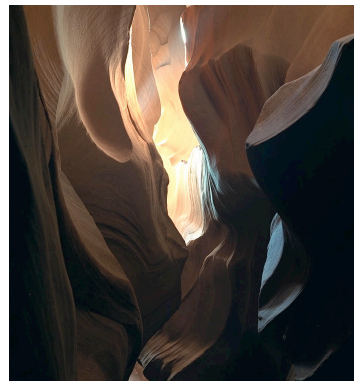
CS 4620

Visual cues to 3D geometry

- size (perspective)
- occlusion
- shading

Shading

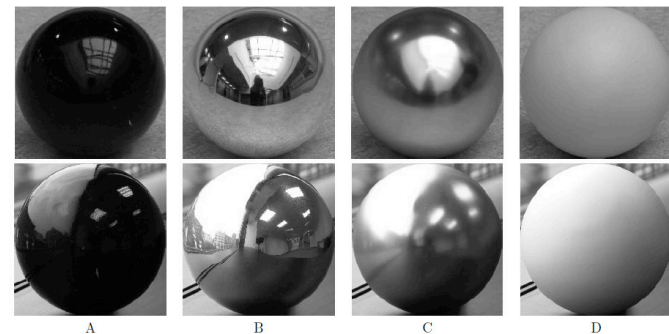
- Variation in observed color across an object
 - strongly affected by lighting
 - present even for homogeneous material
- caused by how a material reflects light
 - depends on
 - geometry
 - lighting
 - material
 - therefore gives cues to all 3



[Philip Greenspun]

Recognizing materials

- Human visual system is quite good at understanding shading



[Dror, Adelson, & Willsky]

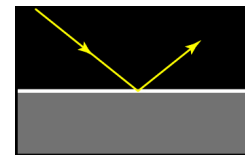
Shading for Computer Graphics

- Need to compute an image
 - of particular geometry
 - under particular illumination
 - from a particular viewpoint
- Basic question: how much light reflects from an object toward the viewer?

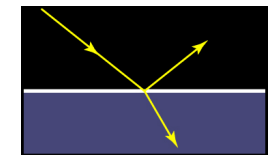
Simple materials



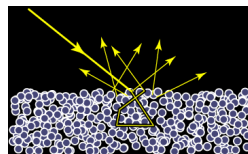
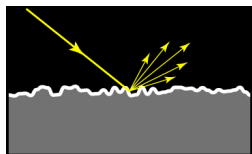
metal



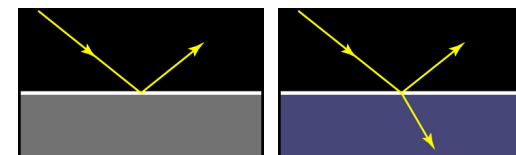
dielectric



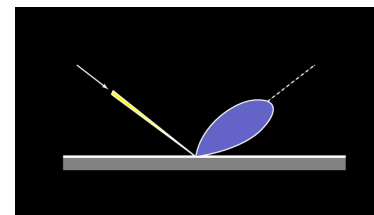
Adding microgeometry



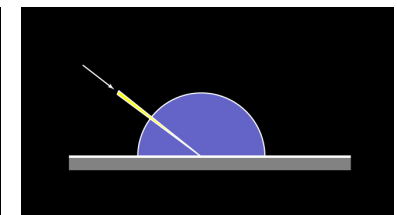
Classic reflection behavior



ideal specular (Fresnel)



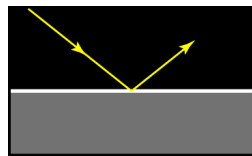
rough specular



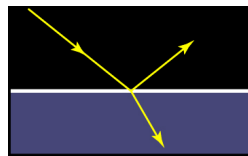
Lambertian

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

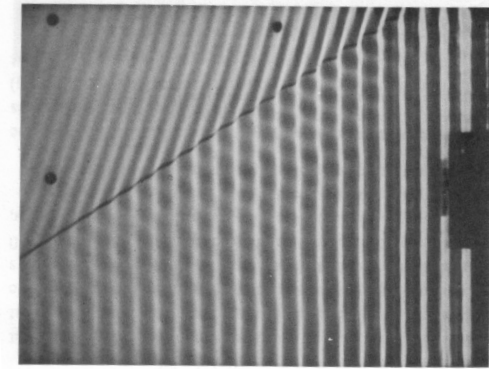


metal



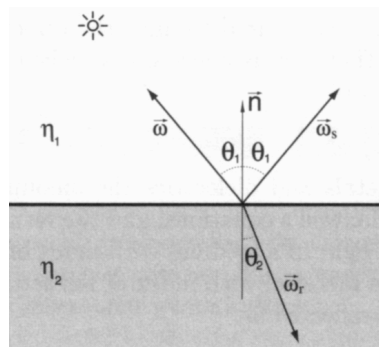
dielectric

Refraction at boundary of media



Snell's Law

- Tells us where the refracted ray goes
 - a.k.a. transmission vector
- Computation
 - ratio of sines is ratio of in-plane components
 - project to surface; scale by eta ratio; recompute normal-direction component
 - total internal reflection



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

Example values of n :
 air: 1.00;
 water: 1.33–1.34;
 window glass: 1.51;
 optical glass: 1.49–1.92;
 diamond: 2.42.

Computing the Transmission Vector, t

$$n \sin \theta = n_t \sin \phi.$$

Computing the sine of an angle between two vectors is usually not as convenient as computing the cosine, which is a simple dot product for the unit vectors such as we have here. Using the trigonometric identity $\sin^2 \theta + \cos^2 \theta = 1$, we can derive a refraction relationship for cosines:

$$\cos^2 \phi = 1 - \frac{n^2 (1 - \cos^2 \theta)}{n_t^2}.$$

Note that if n and n_t are reversed, then so are θ and ϕ as shown on the right of Figure 13.1.

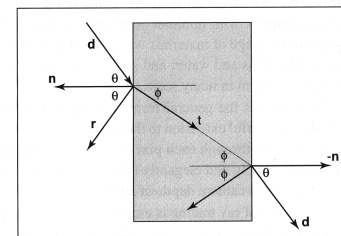


Figure 13.1. Snell's Law describes how the angle ϕ depends on the angle θ and the refractive indices of the object and the surrounding medium.

Computing the Transmission Vector, \mathbf{t}

$$\mathbf{t} = \sin \phi \mathbf{b} - \cos \phi \mathbf{n}.$$

Since we can describe \mathbf{d} in the same basis, and \mathbf{d} is known, we can solve for \mathbf{b} :

$$\begin{aligned}\mathbf{d} &= \sin \theta \mathbf{b} - \cos \theta \mathbf{n}, \\ \mathbf{b} &= \frac{\mathbf{d} + \mathbf{n} \cos \theta}{\sin \theta}.\end{aligned}$$

This means that we can solve for \mathbf{t} with known variables:

$$\begin{aligned}\mathbf{t} &= \frac{n(\mathbf{d} + \mathbf{n} \cos \theta)}{n_t} - \mathbf{n} \cos \phi \\ &= \frac{n(\mathbf{d} - \mathbf{n}(\mathbf{d} \cdot \mathbf{n}))}{n_t} - \mathbf{n} \sqrt{1 - \frac{n^2(1 - (\mathbf{d} \cdot \mathbf{n})^2)}{n_t^2}}.\end{aligned}$$

Note that this equation works regardless of which of n and n_t is larger. An immediate question is, "What should you do if the number under the square root is negative?" In this case, there is no refracted ray and all of the energy is reflected. This is known as *total internal reflection*, and it is responsible for much of the rich appearance of glass objects.

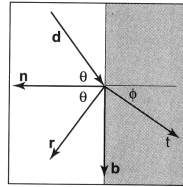


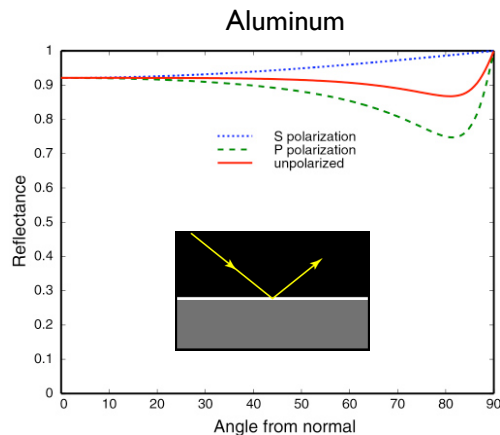
Figure 13.2. The vectors \mathbf{n} and \mathbf{b} form a 2D orthonormal basis that is parallel to the transmission vector \mathbf{t} .

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

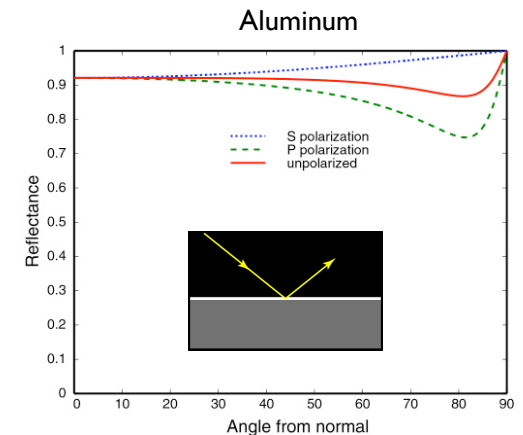
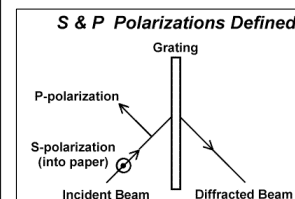
- Reflectance does depend on angle
 - but not much
 - safely ignored in basic rendering



Specular reflection from metal

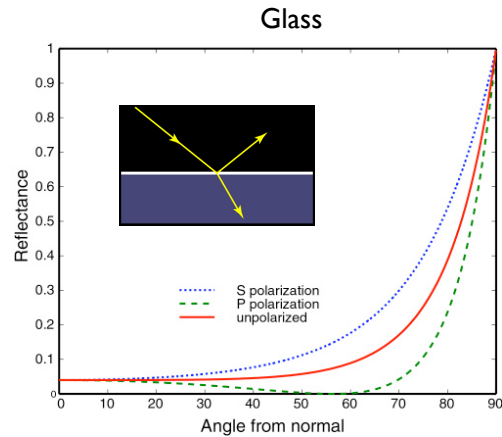
- Reflectance does depend on angle
 - but not much
 - safely ignored in basic rendering

FYI...



Specular reflection from glass/water

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

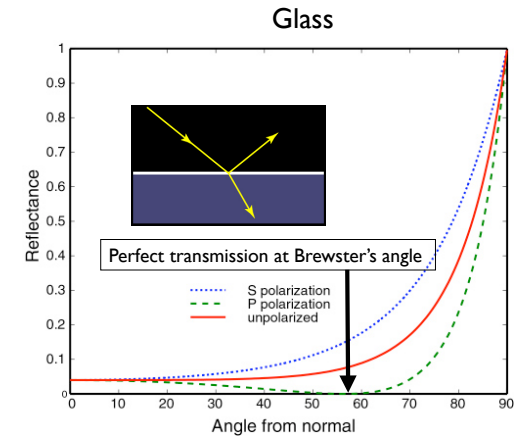


Cornell CS4620 Fall 2010

© 2010 Doug James • 16

Specular reflection from glass/water

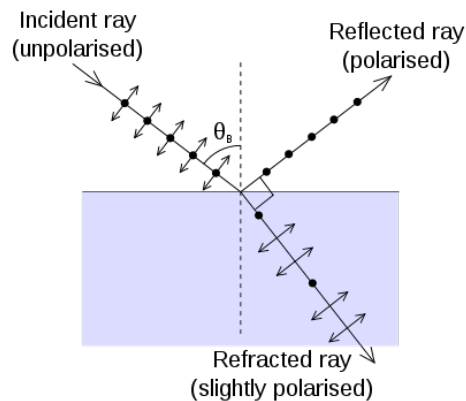
- Dependence on angle is dramatic!
 - about 4% at normal incidence
 - 100% at grazing
 - remaining light is transmitted
- Important for proper appearance



Cornell CS4620 Fall 2010

© 2010 Doug James • 16

Brewster's Angle

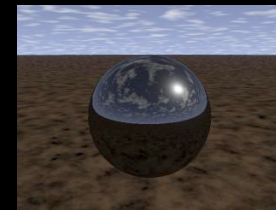


Cornell CS4620 Fall 2010

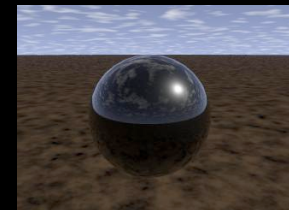
© 2010 Doug James • 17

Fresnel reflection

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



Constant reflectance



Fresnel reflectance

Cornell CS4620 Fall 2010

© 2010 Doug James •

Fresnel's formulas

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

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

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

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

- 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:

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

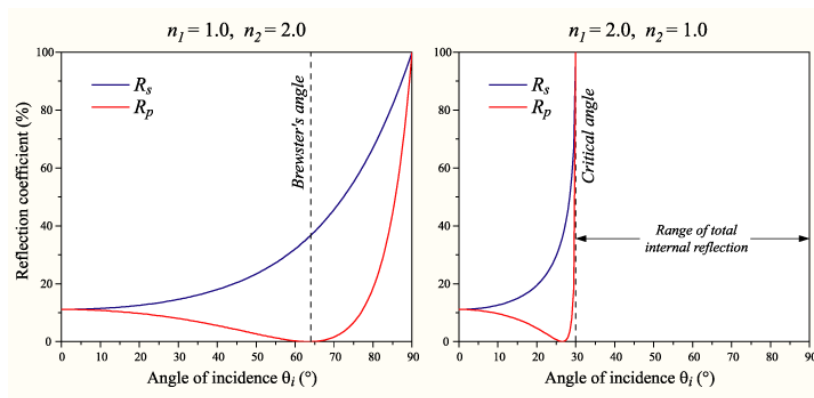
- R_0 is easy to compute:

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

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

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

Fresnel reflection



<http://en.wikipedia.org/wiki/File:Fresnel2.png>

Fresnel reflection



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



Hmm, ... how do we render beer?

Cornell CS4620 Fall 2010

© 2010 Doug James •

Hmm, ... how do we render beer?



Beer's Law!

Light-carrying ray intensity I is attenuated with distance traveled x according to $\frac{dI}{dx} = -bI$, $b \geq 0$

Therefore, the attenuated per-channel intensity is

$$I(x) = I(0)e^{-bx}, \quad x > 0$$

Evaluated on a per-channel basis, e.g., the color of white light after traveling unit distance is

$$\mathbf{I}(1) = (e^{-b_R}, e^{-b_G}, e^{-b_B})$$

$$\xrightarrow{\ln} (b_R, b_G, b_B) = -\ln(\mathbf{I}(1))$$

Cornell CS4620 Fall 2010

© 2010 Doug James •

Cornell CS4620 Fall 2010

© 2010 Doug James •

Not just for beer...

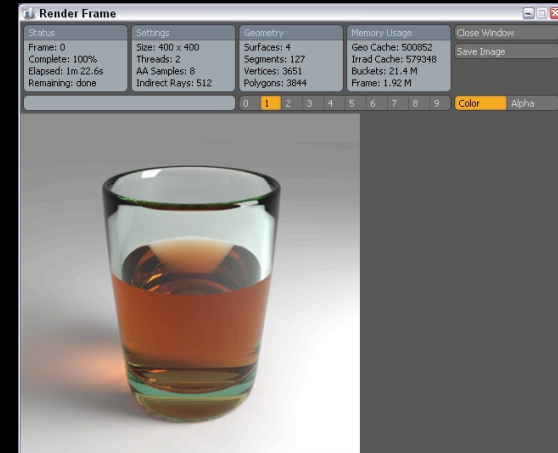


[Josh Willis | 2003 UCSD Rendering Competition]

Cornell CS4620 Fall 2010

© 2010 Doug James •

Not just for beer...



Modo 202 render

Cornell CS4620 Fall 2010

© 2010 Doug James •

Shader for Transparent Objects (pp. 306-7)

To add transparent materials to our code, we need a way to determine when a ray is going “into” an object. The simplest way to do this is to assume that all objects are embedded in air with refractive index very close to 1.0, and that surface normals point “out” (toward the air). The code segment for rays and dielectrics with these assumptions is:

```

if (p is on a dielectric) then
    r = reflect(d, n)
    if (d · n < 0) then
        refract(d, n, t)
        c = -d · n
        kr = kg = kb = 1
    else
        kr = exp(-art)
        kg = exp(-agt)
        kb = exp(-abt)
        if refract(d, -n, 1/n, t) then
            c = t · n
        else
            return k * color(p + tr)
    R0 = (n - 1)2 / (n + 1)2
    R = R0 + (1 - R0)(1 - c)5
    return k(R color(p + tr) + (1 - R) color(p + tt))
    
```

Cornell CS4620 Fall 2010

© 2010 Doug James • 28

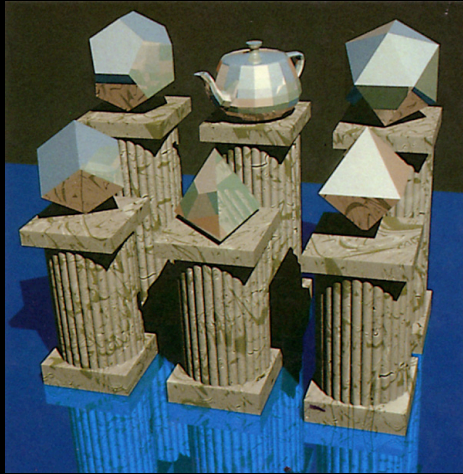
Basic ray tracing

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

Cornell CS4620 Fall 2010

© 2010 Doug James • 29

Basic ray traced image

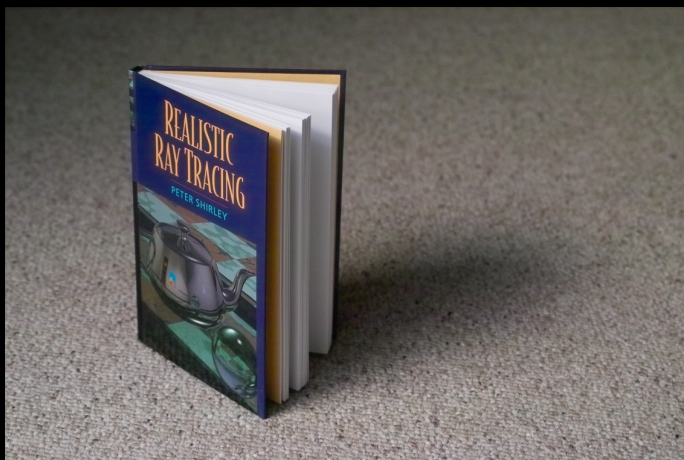


[Glasner 89]

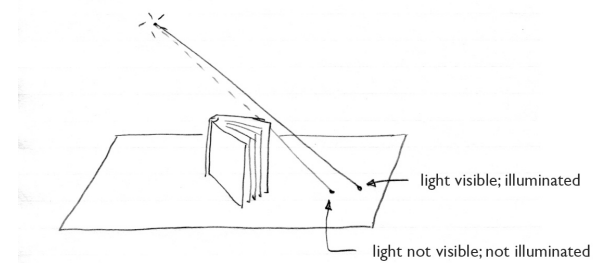
Discontinuities in basic RT

- Perfectly sharp object silhouettes in image
 - leads to aliasing problems (stair steps)
- Perfectly sharp shadow edges
 - everything looks like it's in direct sun
- Perfectly clear mirror reflections
 - reflective surfaces are all highly polished
- Perfect focus at all distances
 - camera always has an infinitely tiny aperture
- Perfectly frozen instant in time (in animation)
 - motion is frozen as if by strobe light

Soft shadows

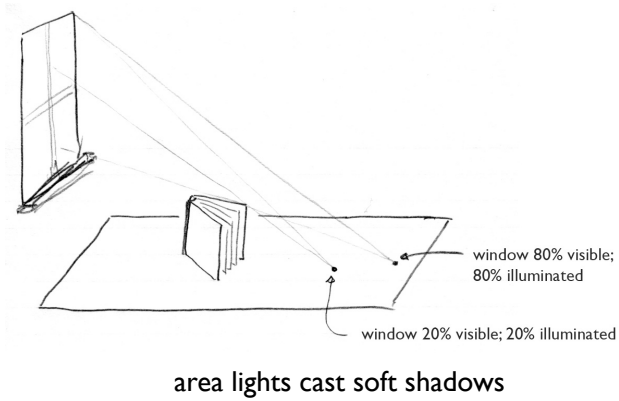


Cause of soft shadows



point lights cast hard shadows

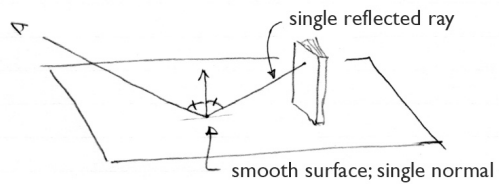
Cause of soft shadows



Glossy reflection

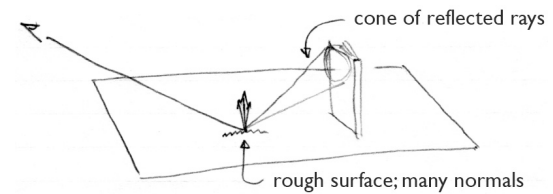


Cause of glossy reflection



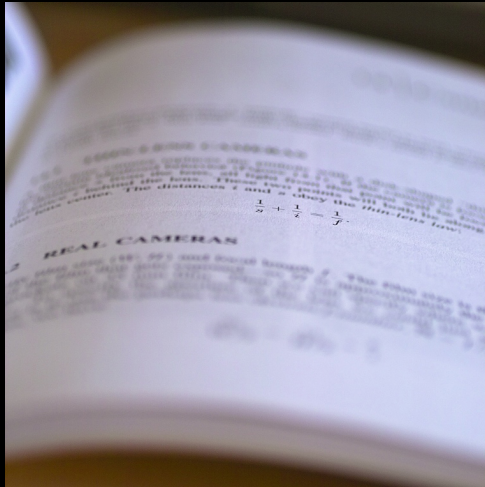
smooth surfaces produce sharp reflections

Cause of glossy reflection

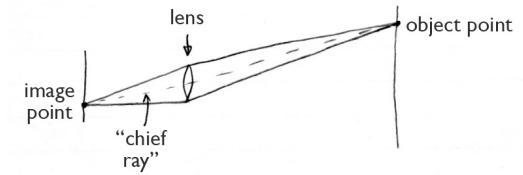


rough surfaces produce soft (glossy) reflections

Depth of field



Cause of focusing effects



what lenses do (roughly)

Cause of focusing effects

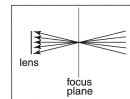
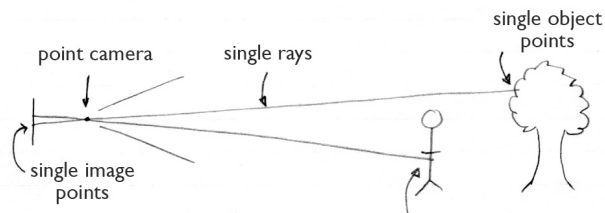
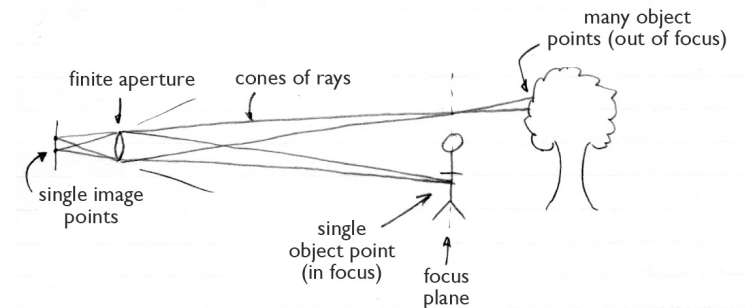


Figure 13.15. The lens averages over a cone of directions that hit the pixel location being sampled.



point aperture produces always-sharp focus

Cause of focusing effects



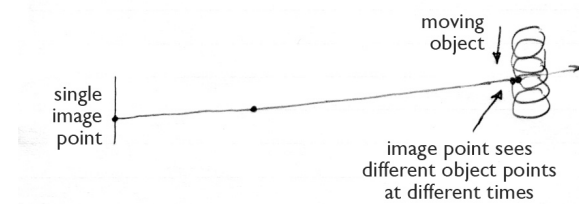
finite aperture produces limited depth of field

Motion blur



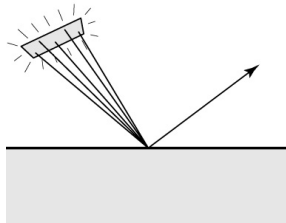
[Cook, Porter, Carpenter 1984]

Cause of motion blur



Creating soft shadows

- For area lights: use many shadow rays
 - and each shadow ray gets a different point on the light
- Choosing samples
 - general principle: start with uniform in square



Creating soft shadows

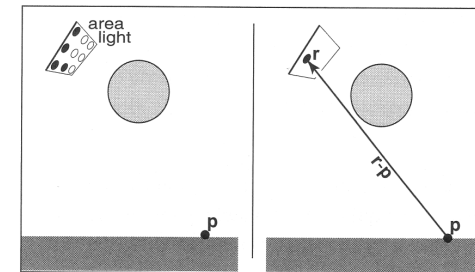
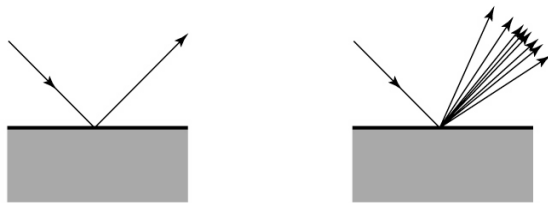


Figure 13.13. Left: an area light can be approximated by some number of point lights; four of the nine points are visible to p so it is in the penumbra. Right: a random point on the light is chosen for the shadow ray, and it has some chance of hitting the light or not.

Creating glossy reflections

- 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



Creating glossy reflections

To choose \mathbf{r}' , we again sample a random square. This square is perpendicular to \mathbf{r} and has width a which controls the degree of blur. We can set up the square's orientation by creating an orthonormal basis with $\mathbf{w} = \mathbf{r}$ using the techniques in Section 2.4.6. Then, we create a random point in the 2D square with side length a centered at the origin. If we have 2D sample points $(\xi, \xi') \in [0, 1]^2$, then the analogous point on the desired square is

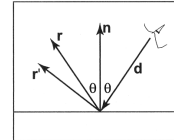


Figure 13.18. The reflection ray is perturbed to a random vector \mathbf{r}' .

$$u = -\frac{a}{2} + \xi a,$$

$$v = -\frac{a}{2} + \xi' a.$$

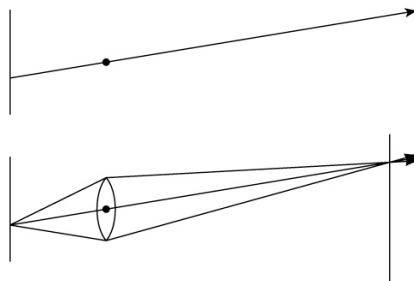
Because the square over which we will perturb is parallel to both the \mathbf{u} and \mathbf{v} vectors, the ray \mathbf{r}' is just

$$\mathbf{r}' = \mathbf{r} + u\mathbf{u} + v\mathbf{v}.$$

Note that \mathbf{r}' is not necessarily a unit vector and should be normalized if your code requires that for ray directions.

Depth of field

- Make eye rays start at random points on aperture
 - always going toward a point on the focus plane



Depth of field

- Make eye rays start at random points on aperture
 - always going toward a point on the focus plane

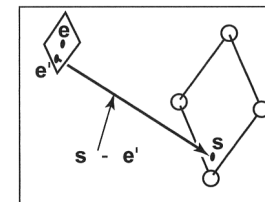


Figure 13.17. To create depth-of-field effects, the eye is randomly selected from a square region.

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

$$T = T_0 + \xi(T_1 - T_0)$$

Generating samples

- A complicated question in general
- Basic idea: start with random points in a square
- Monte Carlo methods—see 600-level graphics courses