Ray Tracing

CSCI 4239/5239
Advanced Computer Graphics
Spring 2025

What is it?

- Method for rendering a scene using the concept of optical rays bouncing off objects
  - More realistic
  - Reflections
  - Shadows

How does it work?

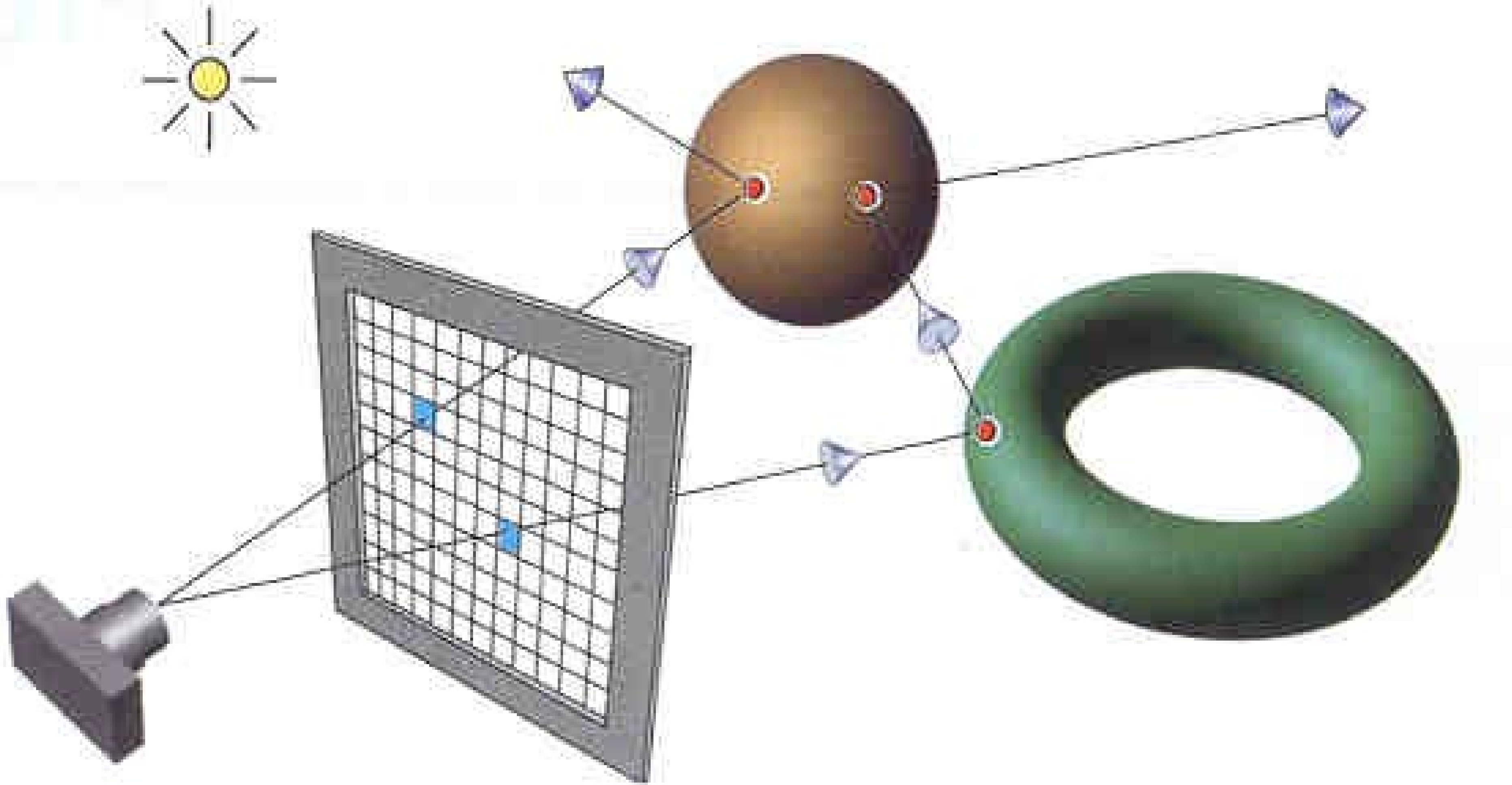


Figure 1. The ray-tracing process.

Sources

- Ray Tracing from the Ground Up
  - Kevin Suffern
  - Excellent tutorial
  - Some working examples
  - http://www.raytracegroundup.com/
- nVidia
- Intel
- PBRT (Physically Based Ray Tracing)

Interactive Ray Tracing

- True ray tracing is VERY compute intensive
- Global problem – scene complexity adds effort
- Generally there is no upper limit to computation
- Solutions are generally software based
  - Dedicated hardware provides 3-5x speedup
  - http://www.caustic.com/
  - OpenRL
  - Maya Plugins
- Compare nVidia RTX

nVidia Quadra Plex 1920x1024@30fps

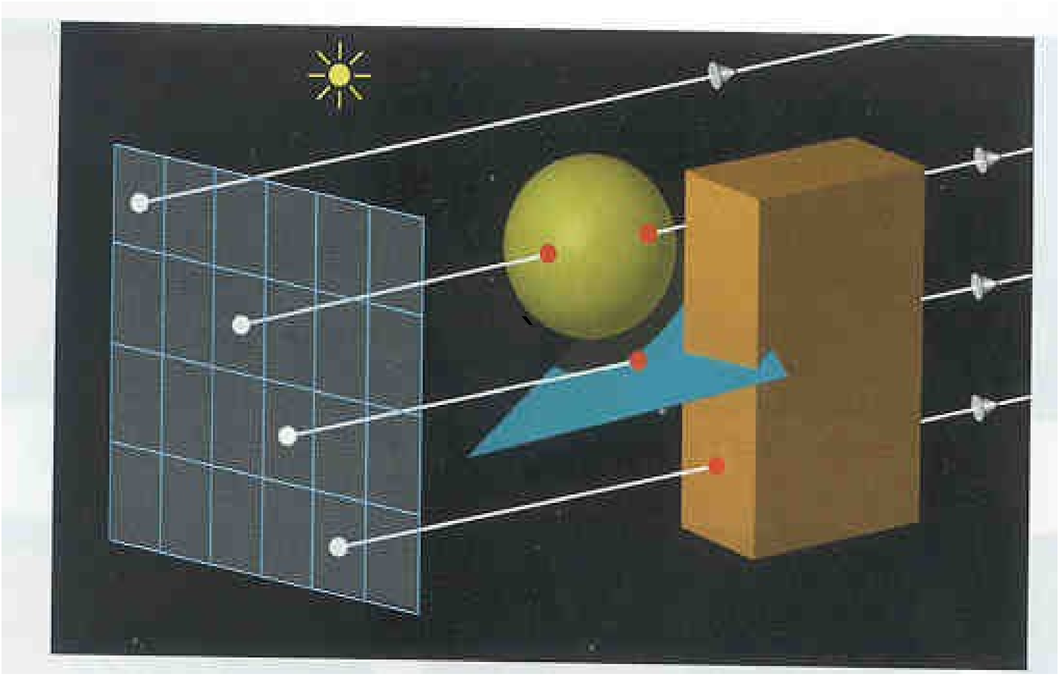
nVidia Quadra Plex 1920x1024@30fps

How is it Done?

- Scene Description Language
  - Defines objects in scene
    - Geometry and properties
  - Lights
  - Eye position
- Determine color of individual pixels using ray tracing algoritms
  - Very hard to do real time

How ray tracing works

- Define scene and view
  - objects
  - lights
  - eye
- For each pixel
  - Shoot ray from pixel
  - Find nearest hit
  - Use object properties and lights to calculate color, or set to black if no hits



True Global Ray Tracing

- Light can bounce many times
  - Color changes at each bounce
  - Each bounce attenuates light
  - Light scatters in complex ways
  - Objects block light
- This simple scene took 2 CPU years to render
  - Cornell Box
  - Area light and three boxes

Efficiency and Complexity

- Most ray tracers written in C++
  - Object Oriented paradigm for objects, rays, colors
  - Good efficiency/readability trade-off
- Efficiency is a HUGE deal
  - Pushing the envelope of hardware
  - Algorithm is global by definition
- Recursion and complexity
  - Need clean interface on objects

What is a Ray?

- **p** = **o** + *t* **d**
- Types of rays
  - Primary rays
  - Secondary rays
  - Shadow rays
  - Light rays

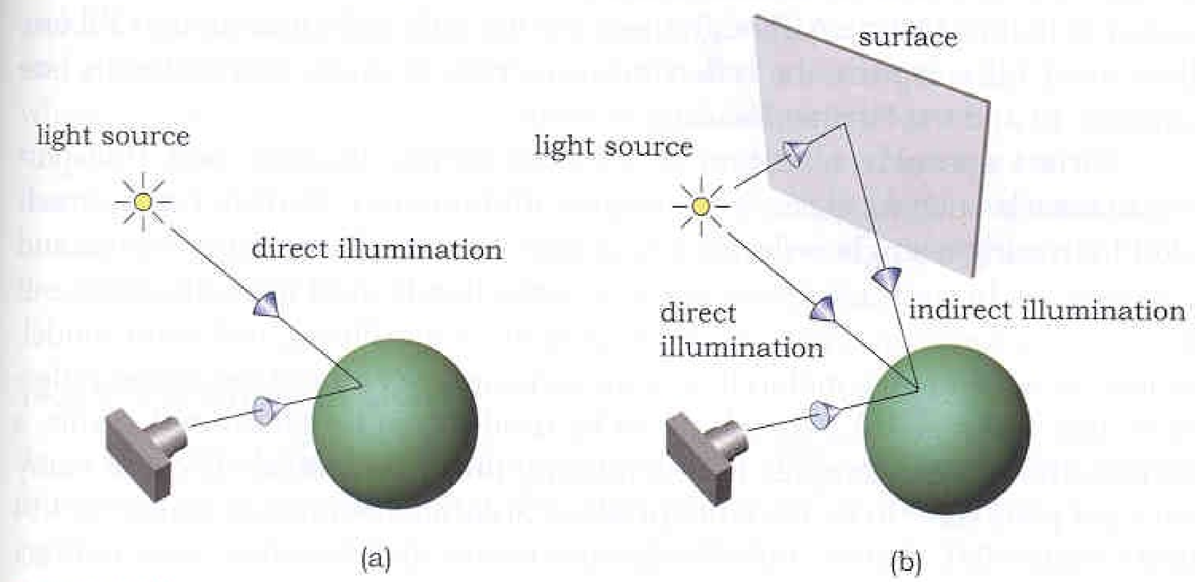


Figure 14.2. (a) Direct illumination hits the surface of an object directly from a light source; (b) indirect illumination hits a surface after being reflected from at least one other surface.

- Rays are one directional

Intersections

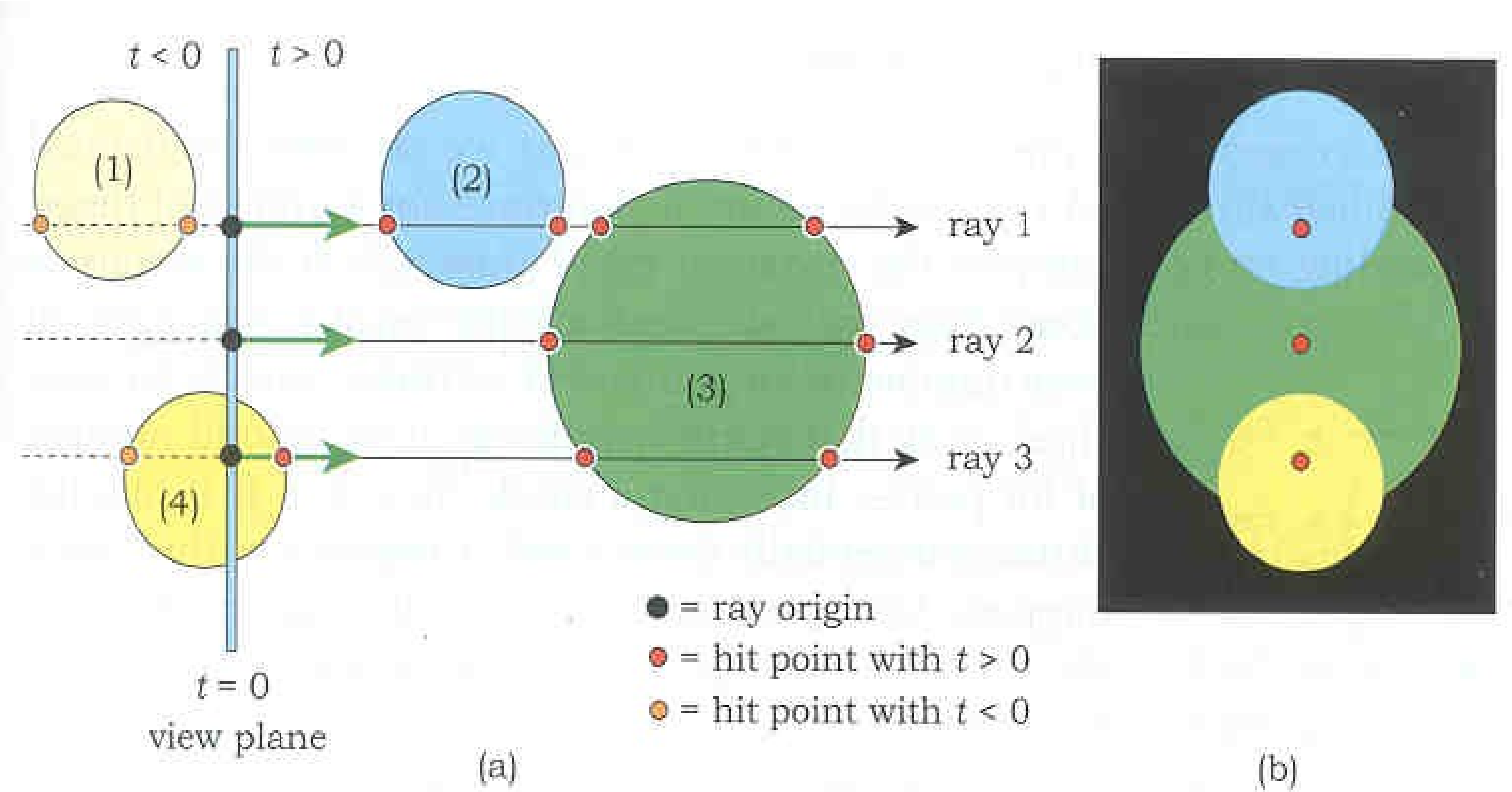


Figure 3.4. (a) Rays and their intersections with spheres; (b) ray-traced image of the spheres.

Intersecting a Sphere

- Simplest 3D object
  - Center
  - Radius
- Smooth normal
- Intersections
  - none
  - once
    - tangent
    - internal
  - twice

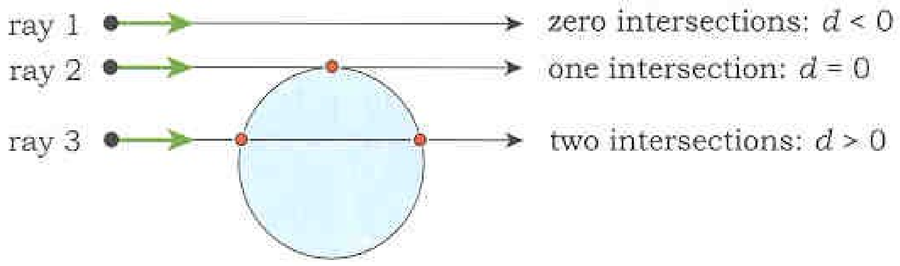


Figure 3.7. Ray-sphere intersections.

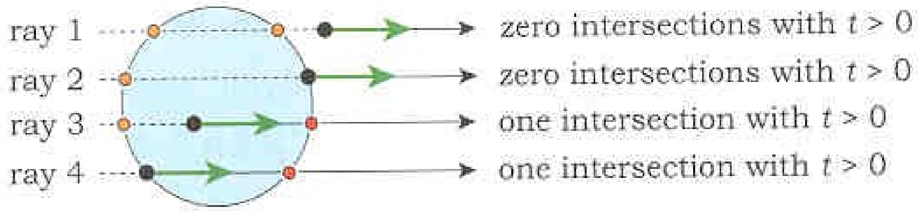


Figure 3.8. Further ray-sphere intersections.

Implicit Surfaces

- General
  - $f(x,y,z) = 0$
- Plane: Point **a** and Normal **n**
  - $(p-a) \bullet n = 0$
- Sphere
  - $(\mathbf{p}-\mathbf{a}) \bullet (\mathbf{p}-\mathbf{a}) - r^2 = 0$
- Triangle
  - Limit plane

Interaction between Lights and Objects

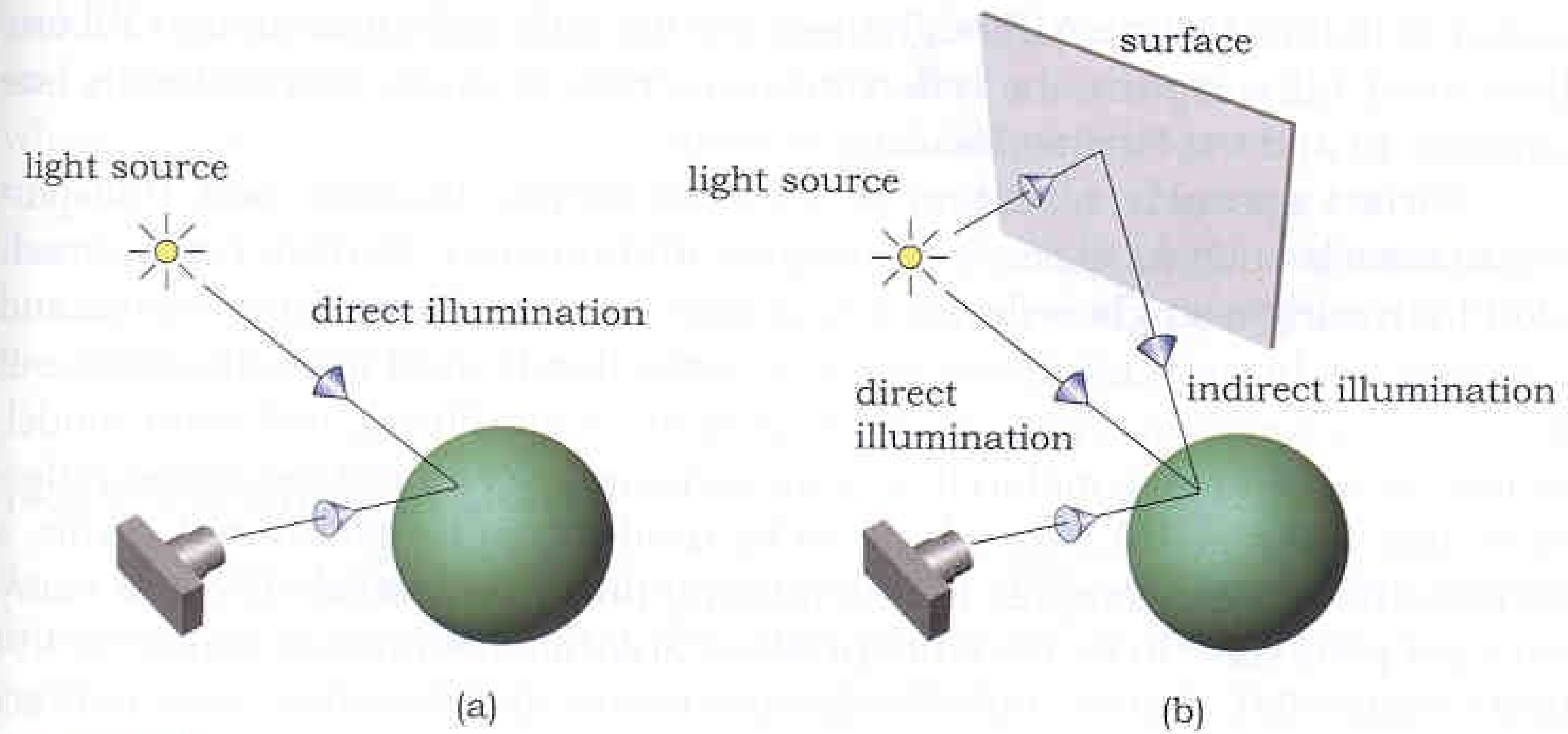


Figure 14.2. (a) Direct illumination hits the surface of an object directly from a light source; (b) indirect illumination hits a surface after being reflected from at least one other surface.

Bouncing Rays from Surfaces

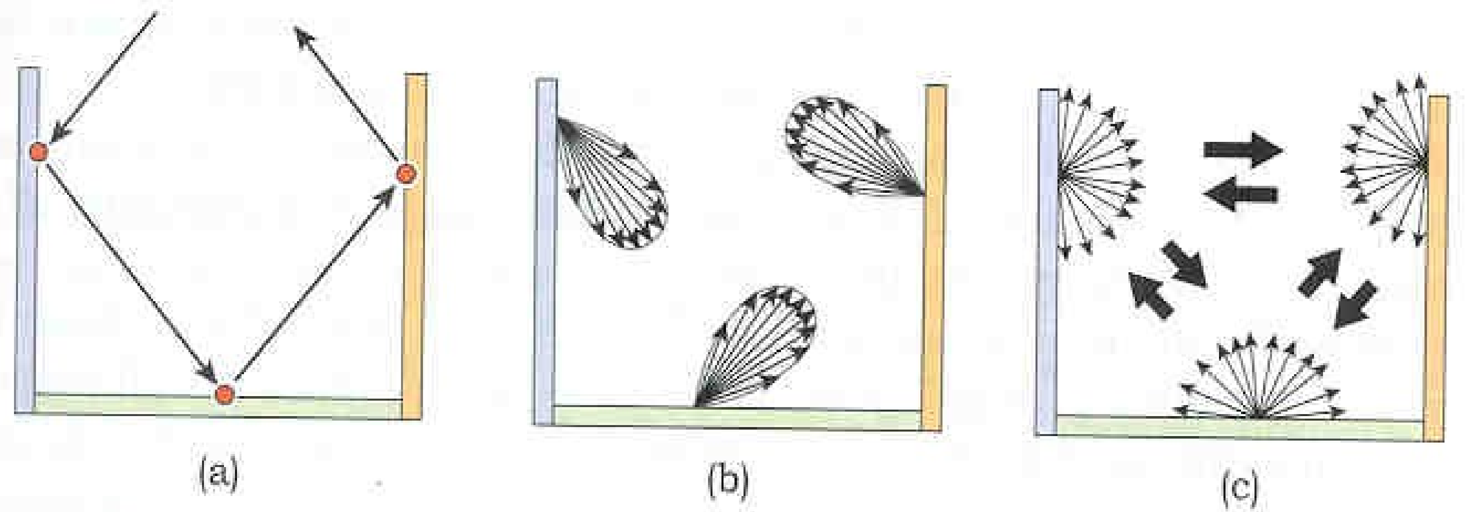


Figure 14.4. (a) Mirror reflection can be modeled by tracing a single reflected ray at each hit point; (b) modeling glossy specular light transport between surfaces requires many rays to be traced per pixel; (c) modeling perfect diffuse light transport between surfaces also requires many rays to be traced per pixel.

Light Reflection

- Diffuse (Lambertian) reflection
  - Intensity Factor N•L

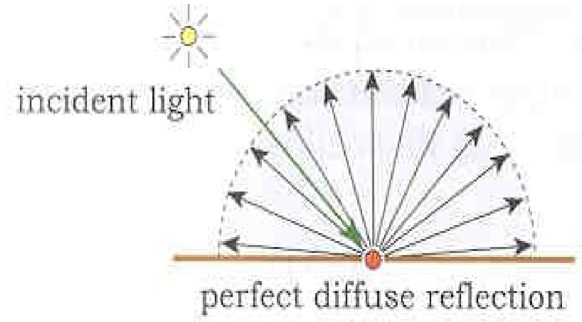


Figure 13.6. Light being scattered from a perfectly diffuse surface.

- Specular reflection
  - R = 2(N•L)N-L
  - Intensity Factor

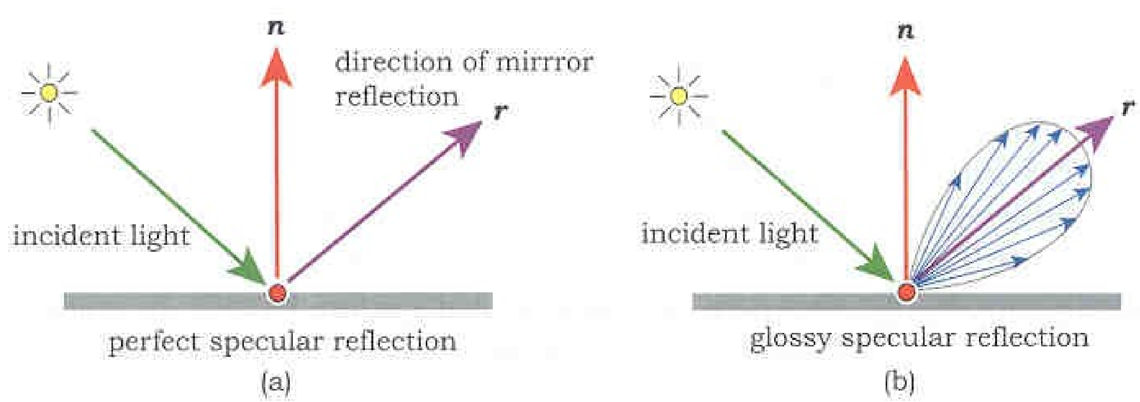


Figure 14.3. (a) Perfect specular reflection; (b) glossy specular reflection.

Specular Reflected Light

- Assume the ray (from the eye) hits objects 1,2,3,... with reflection coefficients $\alpha_1, \alpha_2, \alpha_3, \ldots$
- Specular Reflection Color

$$\alpha_1(C_1 + \alpha_2(C_2 + \alpha_3(C_3 + \ldots)))$$

$$= \alpha_1 C_1 + \alpha_1\alpha_2 C_2 + \alpha_1\alpha_2\alpha_3 C_3 + \ldots$$

- Since light is assumed to be linearly additive, just keep track of α and add light along successive bounces of the ray
- White specular means α can be a scalar