# Ray Tracing

# Features

Advantages

- model higher order specular reflection and transmission
- model visible surface detection, complex shadow effects and transparency
- highly realistic for shiny objects

Limitations

- does not model diffuse effects
- produce “hard shadow” (i.e. boundary of shadow is sharp)
- more computationally intensive than ray casting

# Ray Casting (fig. 9-26)

- retrace the light paths of the rays that arrive at the pixel

- for each pixel, send a ray from PRP that goes through the pixel
- find all intersections of the ray with the surfaces
- the nearest intersections is the visible part of the surface for that pixel

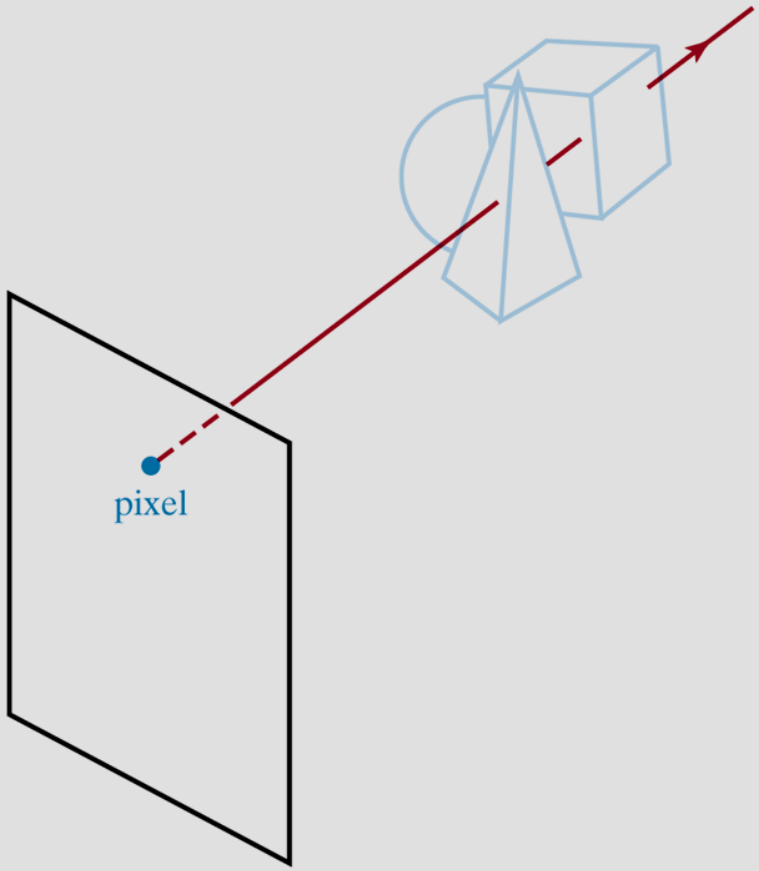
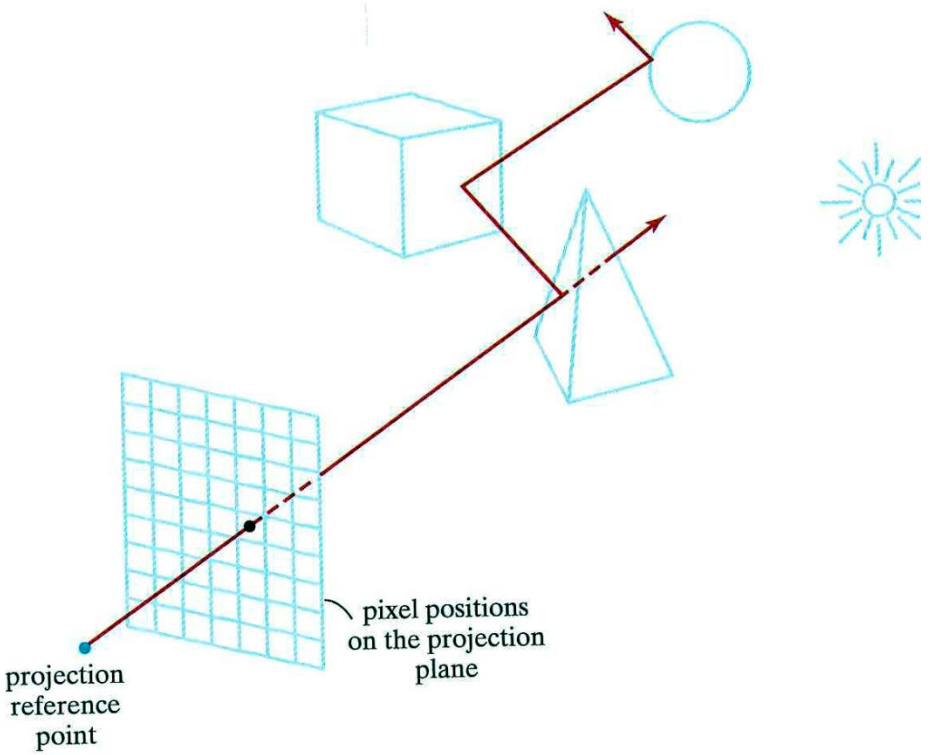



Figure 9-26

A ray along the line of sight from a pixel position through a scene.

# Ray Tracing

- extension of *ray casting*. Ray tracing continues to bounce the ray around the scene, collecting intensity contributions



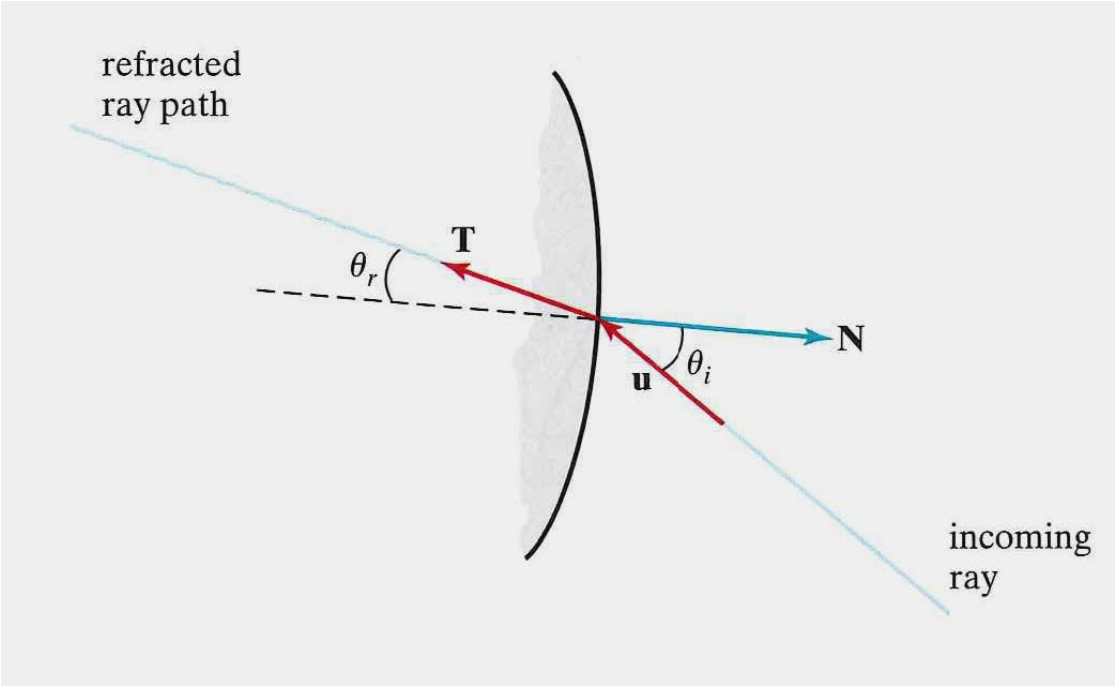
# Reflection and refraction rays

- reflect the ray off the surface along the specular path R
- R is calculated in the usual way (see Lecture 7)
- if the surface is semi-transparent, also send a ray through the surface in the refraction direction T.

- reflection and refraction rays are called *secondary rays*.

# Modelling transparency

$$u = -L$$




$$T = \frac{\eta_i}{\eta_r} u - (\cos\theta_r - \frac{\eta_i}{\eta_r} \cos\theta_i) N$$

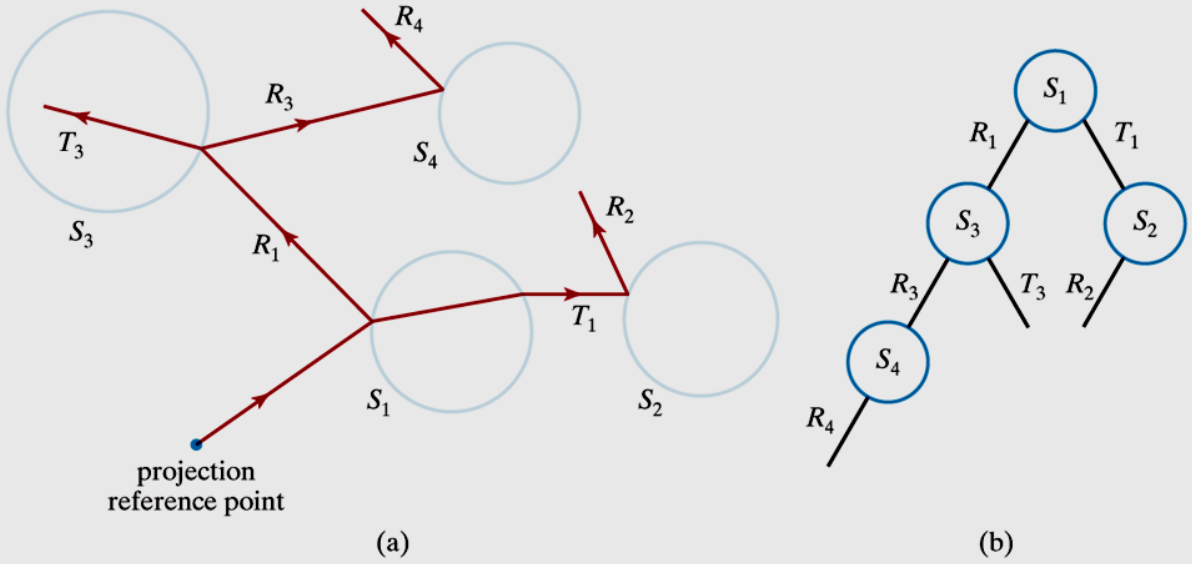



Figure 10-55

The reflection and refraction paths for a pixel ray traveling through a scene are shown in (a), and the corresponding binary ray-tracing tree is given in (b).

# Binary ray tracing tree

- recursively apply to each secondary ray, generate a binary *ray-tracing tree* (Fig. 10-55)
- a secondary ray terminates if

  i) it does not hit any surface or

  ii) it hits a light source
- ray tracing tree terminates if it reaches a pre-set maximum depth

# Calculate intensity at each node

- Intensity at each node is calculated in the usual way, i.e.,

$$I = k_a I_a + \sum_{i=1}^{n} I_{li}[k_d(N \cdot L_i) + k_s(V \cdot R_i)^{n_s}]$$

- Incorporate shadow effect
  - "multiple shadows", "shadow within shadow" can be modelled
  - The path along L is referred to as the *shadow ray*. The idea is to let $I_{li} = 0$ if there is an object between the surface and light source i

# Attenuate light intensity at each edge

- surface intensity from each node in the tree is attenuated by the distance from the “parent surface” (next node up the tree) and added to the intensity of the parent surface
- light attenuates according to inverse square law $1/d^2$
- Include constant and linear term as well for realism

$$f(d) = \begin{cases} 1.0 & \text{if light source at infinity} \\ \dfrac{1}{a_0 + a_1 d + a_2 d^2} & \text{if otherwise} \end{cases}$$

# Calculating pixel intensity

- At the bottom (terminal nodes) of tree. If a secondary ray

  i) hits a light source, assign the intensity of the source

  ii) does not hit any surface, assign the intensity of the background

- Start at the bottom and proceed to the top, sum up the attenuated intensity at each node (i.e., *reverse trace the light paths*)

- pixel intensity = Intensity at the root node of the tree

# References

- Ray tracing: Ch. 21-1
- Distance attenuation function: Ch. 17-1