The problem
A solution
Hidden surface removal

★ For each pixel $\pi$, shoot a ray $\rho$ from the viewpoint to the center of $\pi$.

★ If $\rho$ does not intersect any object, color $\pi$ with the background color.

★ Otherwise, compute the first object $O$ intersected by $\rho$ and the first intersection point $\sigma$.

★ Compute the color at $\sigma$ using the reflection model.

★ Draw $\pi$ with the computed color.

★ Each pixel is colored only once.

★ Computing $\sigma$ is expensive!
Ray Tracing: Reflection
Ray Tracing: Shadow Generation
Ray Tracing: Refraction

[Diagram of ray tracing with vectors and grid]

CPS124, 296: Computer Graphics Raytracing Page 6
Extend the standard ray tracing to handle shadows, reflection, and refraction.

Shoot secondary rays recursively to calculate shadows, reflection, and refraction.

For each pixel \( \pi \) on the screen, do the following:

- **Primary ray (\( \rho_P \)):** Ray emanating from the viewer to the center of \( \pi \).
- If \( \rho_P \) doesn’t hit any object, render \( \pi \) with the background color.
- Suppose the first intersect point of \( \rho \) and an object is \( p \).
Secondary Rays

- Shoot the following *secondary* rays from \( p \):
  - **Shadow ray** \((\rho_L)\): Shoot a ray along \( p \vec{L} \).
  - **Reflection ray** \((\rho_R)\): If the object has reflectance property (e.g., mirror), shoot a ray in direction \( R \).
  - **Refraction ray** \((\rho_T)\): If the object is transparent, shoot a ray in direction \( T \).

- If \( \rho_R, \rho_T \) hit an object, shoot secondary rays from there as above.
- Apply distance attenuation to the intensity of secondary rays.
No object is hit.

Light source is hit.

Reached a cut-off depth.

Creates a ray tree; evaluate in bottom-up fashion.

\[ I_\lambda = k_a I_{a\lambda} O_{a\lambda} + \sum_{i=1}^{k} S_i f_{att} I_{L_i\lambda} \left[ k_d \cdot O_{d\lambda} (\mathbf{N} \cdot \mathbf{L}_i) + k_s (\mathbf{R} \cdot \mathbf{V})^n \right] + k_R I_{R\lambda} + k_T I_{t\lambda} \]
Pros and Cons

★ Better illumination model.
★ Prone to numerical instability.
★ Very expensive.

Efficiency Issues:

★ Ray object intersection: Use object hierarchy, spatial decomposition techniques (oct trees, BSP’s).
★ Adaptive tree depth
★ Reflection maps
★ Light buffer
Distributed Ray Tracing

★ Handles antialiasing.

★ Divide pixel into subpixels.

★ Choose pixels at random (under some given distribution).

★ Divide each pixel into a grid; *jitter* the centers of the grid randomly within the grid cell.

★ Instead of uniform sampling, use weighted sampling, e.g., distribution of subpixel depends on light intensity.

★ Shoot different rays at slightly different times.