Illumination Algorithms

OpenGL

- **Only point/distant light sources / not area**
- **Assumes objects are not "shadowed"**
- **Doesn't handle interreflection**

Ray Tracing

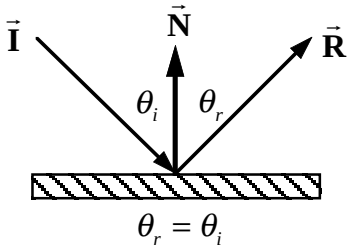
- **Basic ray tracing algorithm**
- **Reflection and refraction**
- **Ray-object intersections**
  - **Plane**
  - **Polygon**
  - **Sphere**
- **Acceleration schemes**

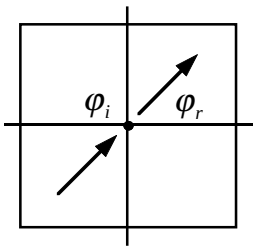
Copyright © Pat Hanrahan

Classic Ray Tracing Algorithm

Copyright © Pat Hanrahan

Law of Reflection



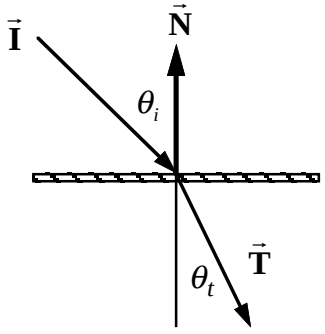


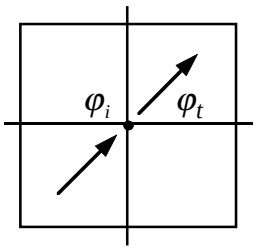
$$\vec{\mathbf{R}} + (-\vec{\mathbf{I}}) = 2\cos\theta\,\vec{\mathbf{N}} = -(\vec{\mathbf{I}} \bullet \vec{\mathbf{N}})\vec{\mathbf{N}}$$

$$\vec{\mathbf{R}} = \vec{\mathbf{I}} - (\vec{\mathbf{I}} \bullet \vec{\mathbf{N}})\vec{\mathbf{N}}$$

Copyright © Pat Hanrahan

Snell's Law

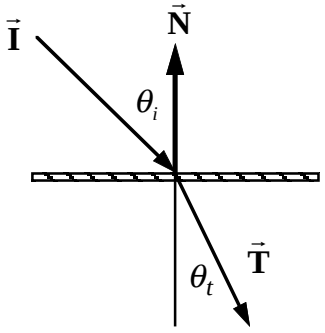




$$n_i \sin\theta_i = n_t \sin\theta_t \qquad n_i\,\vec{\mathbf{N}} \times \vec{\mathbf{T}} = n_t\,\vec{\mathbf{N}} \times \vec{\mathbf{I}}$$

Copyright © Pat Hanrahan

Law of Refraction



$$\vec{\mathbf{N}} \times \vec{\mathbf{T}} = \mu \vec{\mathbf{N}} \times \vec{\mathbf{I}} \qquad \mu = n_i / n_t$$

$$\vec{\mathbf{N}} \times (\vec{\mathbf{T}} - \mu \vec{\mathbf{I}}) = 0 \qquad \vec{\mathbf{T}} = \mu \vec{\mathbf{I}} + \gamma \vec{\mathbf{N}}$$

$$\vec{\mathbf{T}}^2 = 1 = \mu^2 + \gamma^2 + 2\mu\gamma \vec{\mathbf{I}} \bullet \vec{\mathbf{N}}$$

$$\gamma = -\mu \vec{\mathbf{I}} \bullet \vec{\mathbf{N}} \pm \left\{1 - \mu^2 (1 - \left(\vec{\mathbf{I}} \bullet \vec{\mathbf{N}}\right)^2)\right\}^{1/2}$$

$$= \mu \cos\theta_i \pm \left\{1 - \mu^2 \sin^2 \theta_i\right\}^{1/2}$$

$$= \mu \cos\theta_i \pm \cos\theta_t$$

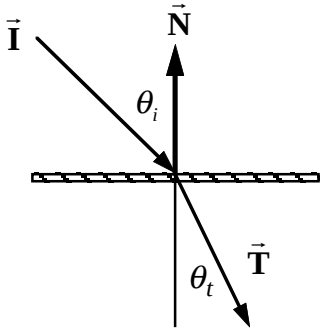
$$= \mu \cos\theta_i - \cos\theta_t \qquad \leftarrow \gamma = \mu - 1$$

Total internal reflection:

$$1 - \mu^2 (1 - \left(\vec{\mathbf{I}} \bullet \vec{\mathbf{N}}\right)^2) < 0$$

Copyright © Pat Hanrahan

Law of Refraction



$$\vec{\mathbf{N}} \times \vec{\mathbf{T}} = \mu \vec{\mathbf{N}} \times \vec{\mathbf{I}} \qquad \mu = n_i / n_t$$

Total internal reflection:

Copyright © Pat Hanrahan

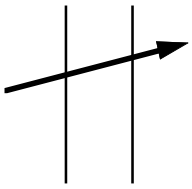
Ray-Plane Intersection

Ray: $\vec{\mathbf{P}} = \vec{\mathbf{O}} + t\vec{\mathbf{D}}$

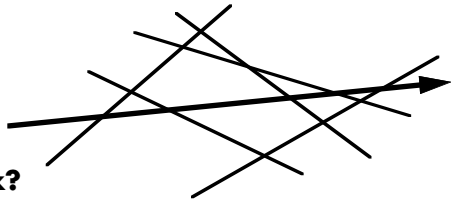
Plane: $\vec{\mathbf{P}} \bullet \vec{\mathbf{N}} + d = 0$

$ax + by + cz + d = 0$

Ray-Slab



Ray-ConvexPolyhedra



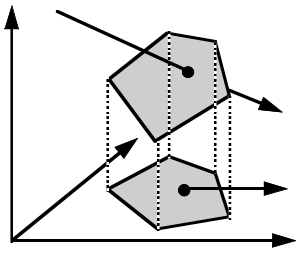
Ray-Box?

Ray-Zonahedra?

Copyright © Pat Hanrahan

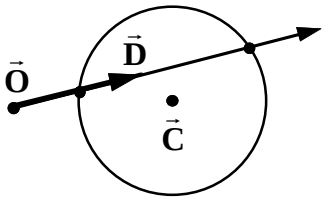
Ray-Polygon Intersection

1. **Find intersection with plane of support**
2. **Test whether point is inside 3d polygon**
   a. **Project onto xy plane (actually use max N coord.)**
   b. **Test whether point is inside 2d polygon**



Copyright © Pat Hanrahan

Ray-Sphere Intersection



Ray: $\vec{\mathbf{P}} = \vec{\mathbf{O}} + t\vec{\mathbf{D}}$

Sphere: $\left(\vec{\mathbf{P}} - \vec{\mathbf{C}}\right)^2 - R^2 = 0$

$$\left(\vec{\mathbf{O}} - t\vec{\mathbf{D}} - \vec{\mathbf{C}}\right)^2 - R^2 = 0$$

$$at^2 + bt + c = 0$$

$$a = \vec{\mathbf{D}}^2 = 1$$

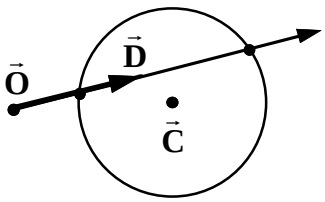
$$b = 2(\vec{\mathbf{O}} - \vec{\mathbf{C}}) \bullet \vec{\mathbf{D}}$$

$$c = (\vec{\mathbf{O}} - \vec{\mathbf{C}})^2 - R^2$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Copyright © Pat Hanrahan

Ray-Sphere Intersection

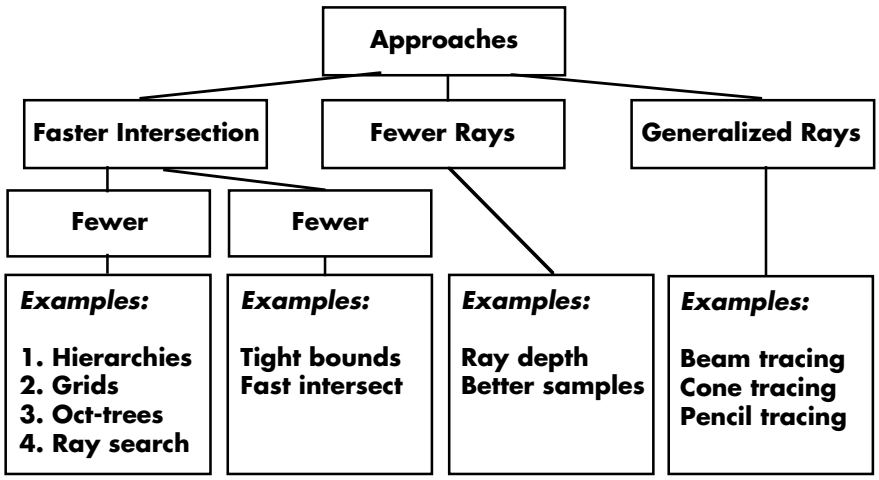


Ray: $\vec{\mathbf{P}} = \vec{\mathbf{O}} + t\vec{\mathbf{D}}$

Sphere: $\left(\vec{\mathbf{P}} - \vec{\mathbf{C}}\right)^2 - R^2 = 0$

Copyright © Pat Hanrahan

Ray Tracing Acceleration Techniques



Copyright © Pat Hanrahan

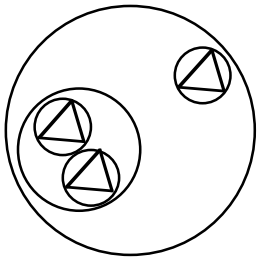
Hierarchical Bounding Volumes

Create of tree of bounding volumes

- **Children are contained within parent**
- **Creation preprocess**
  - **From model hierarchy**
  - **Automatically**

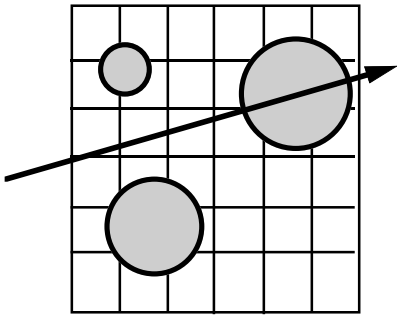
Search

```
Test parent
if intersect
   recurse on children
     in ray order
```



Copyright © Pat Hanrahan

Grids



Fujimoto *et al.*

1. Preprocess scene

Place object in cell if overlaps cell

2. Traverse grid

3D line drawing

What resolution?

Copyright © Pat Hanrahan

Photon Paths

Copyright © Pat Hanrahan

To Be Continued ...

Copyright © Pat Hanrahan