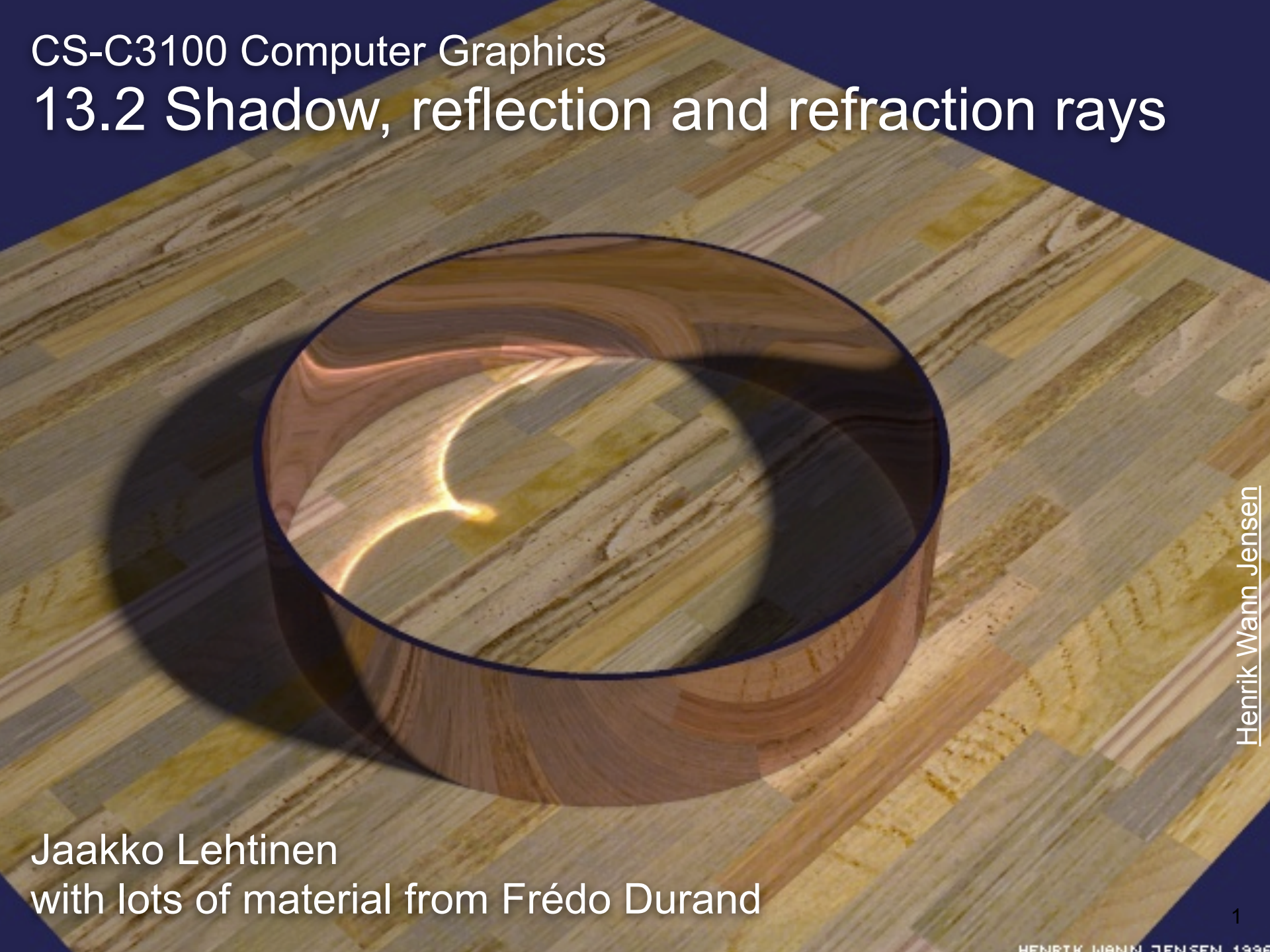
CS-C3100 Computer Graphics

13.2 Shadow, reflection and refraction rays



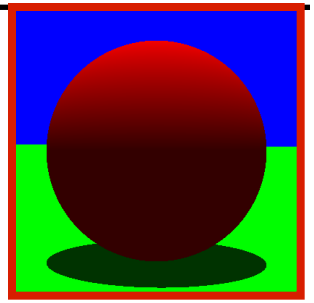
Jaakko Lehtinen
with lots of material from Frédo Durand

In This Video

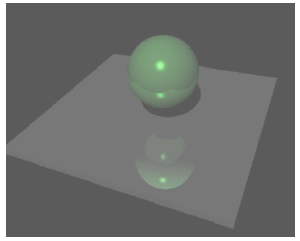
- Using the ray tracer for..
  - Shadows
    - With an important optimization
  - Reflections
    - Only perfect mirrors for now
  - Refractions
    - The Snell-Descartes law and refractive indices

Using a Ray Tracer for..

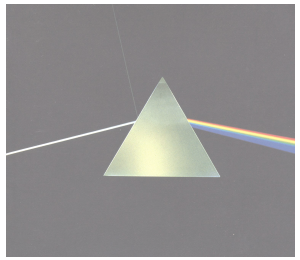
- Shadows



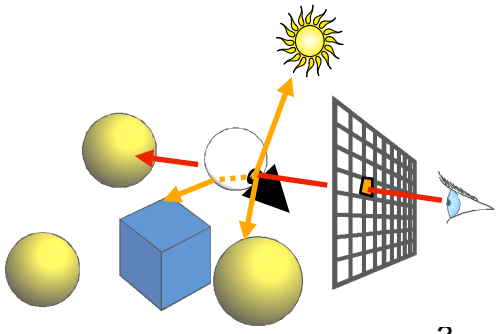
- Reflection



- Refraction



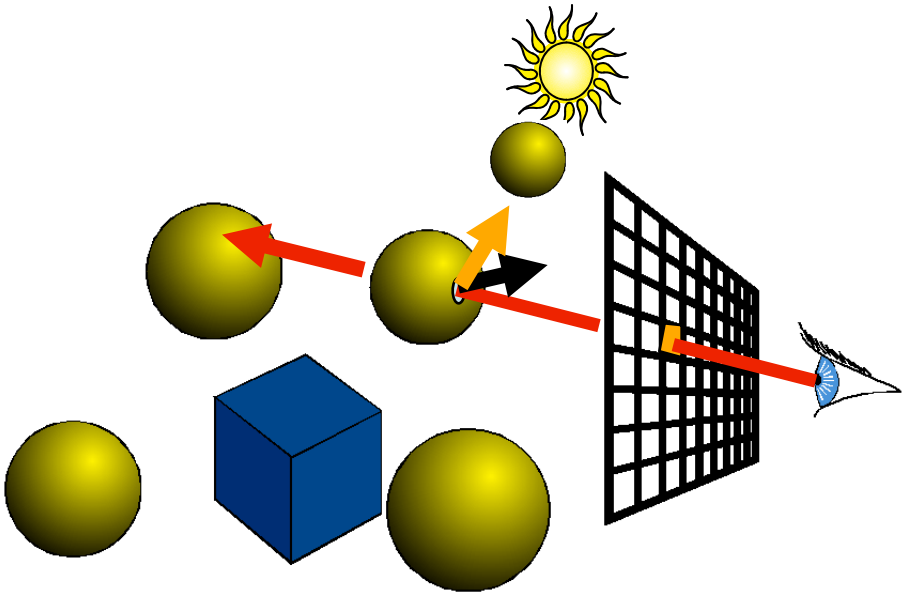
- Recursive Ray Tracing
  - "Hall of mirrors"



How Can We Add Shadows?

$$I = k_d \max(0, \boldsymbol{L} \cdot \boldsymbol{N}) V(\boldsymbol{L})$$

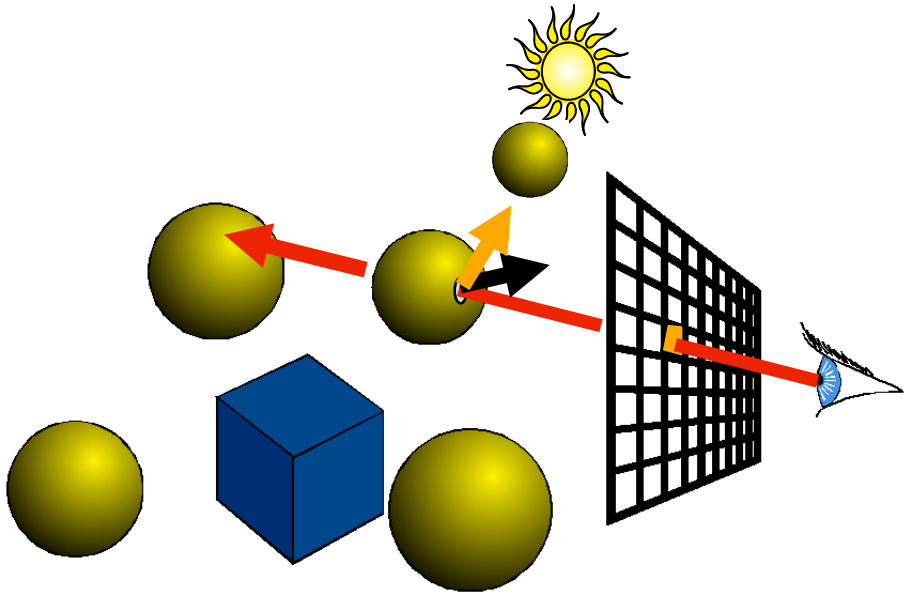
direct lighting

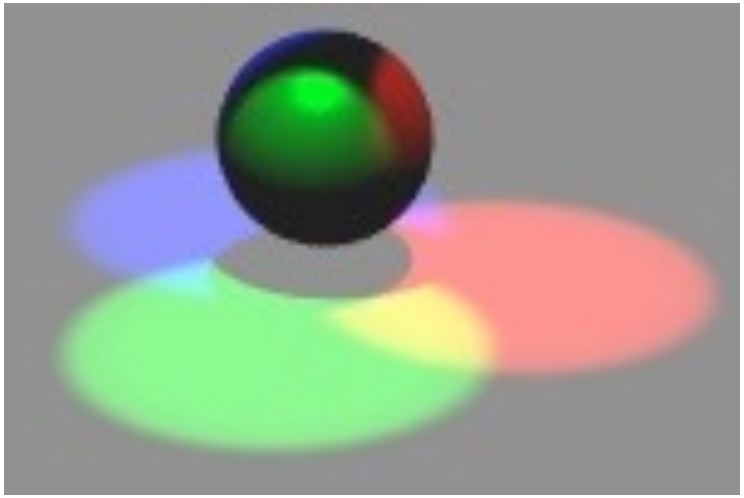


How Can We Add Shadows?

```
color = ambient*hit->getMaterial()->getDiffuseColor()
for every light
   Ray ray2(hitPoint, directionToLight)
   Hit hit2(distanceToLight, NULL, NULL)
   For every object
      object->intersect(ray2, hit2, 0)
   if (hit2->getT() = distanceToLight)
      color += hit->getMaterial()->Shade
                (ray, hit, directionToLight, lightColor)
return color
```

ambient = k_a
diffuseColor = k_d





Problem: Self-Shadowing

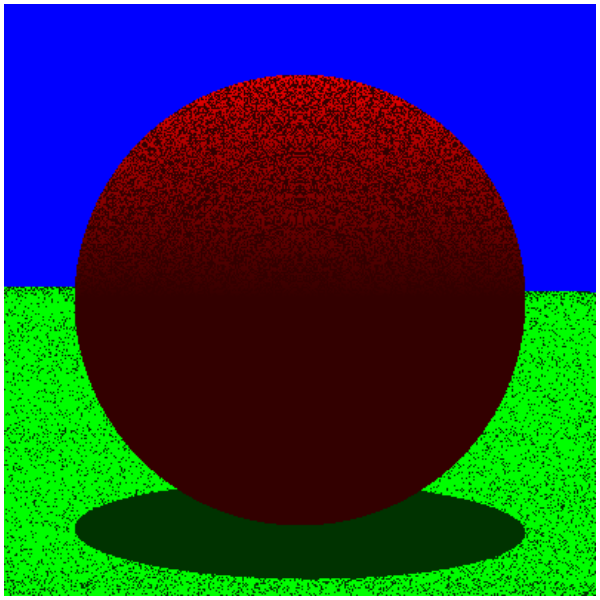
```
color = ambient*hit->getMaterial()->getDiffuseColor()
for every light
   Ray ray2(hitPoint, directionToLight)
   Hit hit2(distanceToLight, NULL, NULL)
   For every object
      object->intersect(ray2, hit2, 0)
   if (hit2->getT() = distanceToLight)
      color += hit->getMaterial()->Shade
               (ray, hit, directionToLight, lightColor)
return color
```

Problem: Self-Shadowing

```
color = ambient*hit->getMaterial()->getDiffuseColor()
for every light
   Ray ray2(hitPoint, directionToLight)
   Hit hit2(distanceToLight, NULL, NULL)
   For every object
      object->intersect(ray2, hit2, epsilon)
   if (hit2->getT() = distanceToLight)
      color += hit->getMaterial()->Shade
               (ray, hit, directionToLight, lightColor)
return color
```

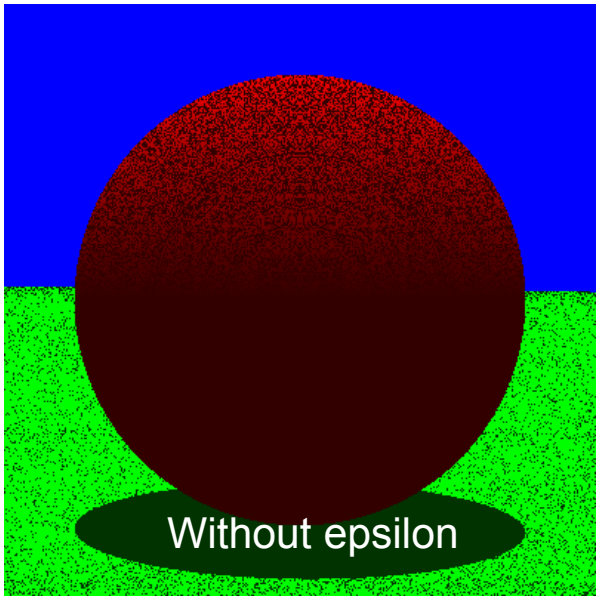
Problem: Self-Shadowing

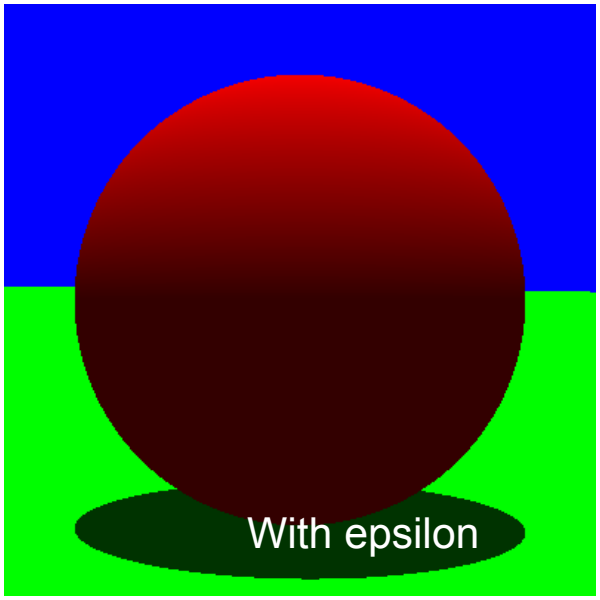
```
color = ambient*hit->getMaterial()->getDiffuseColor()
for every light
   Ray ray2(hitPoint, directionToLight)
   Hit hit2(distanceToLight, NULL, NULL)
   For every object
      object->intersect(ray2, hit2, epsilon)
   if (fabs(hit2->getT()-distanceToLight) < epsilon)
      color += hit->getMaterial()->Shade
                (ray, hit, directionToLight, lightColor)
return color
```



Problem: Self-Shadowing

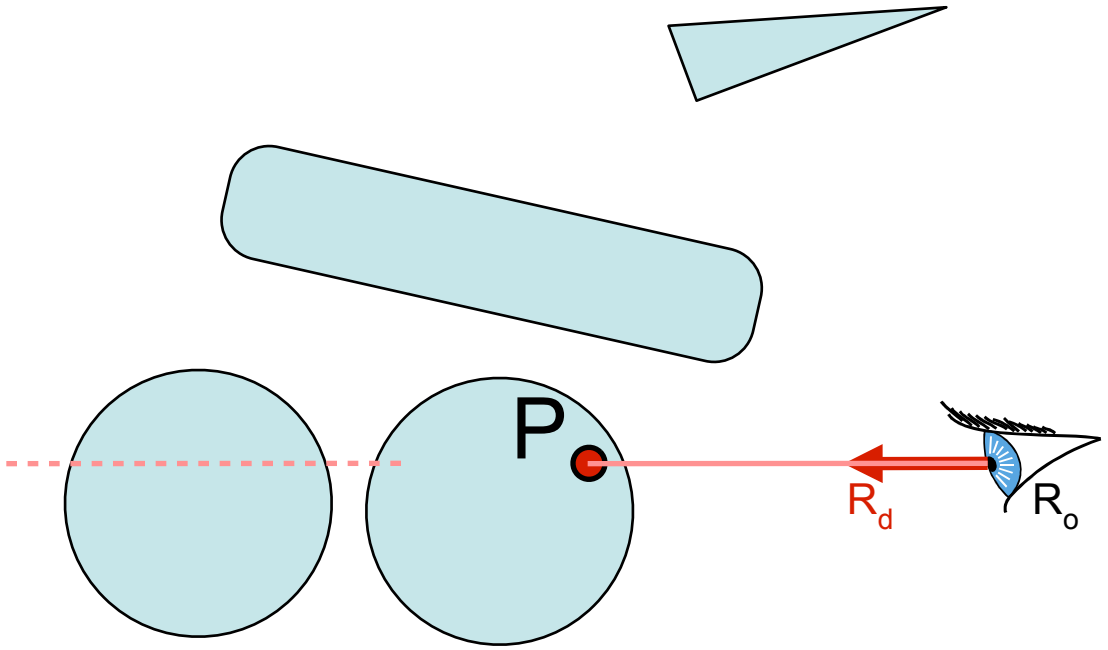
```
color = ambient*hit->getMaterial()->getDiffuseColor()
for every light
   Ray ray2(hitPoint, directionToLight)
   Hit hit2(distanceToLight, NULL, NULL)
   For every object
      object->intersect(ray2, hit2, epsilon)
   if (fabs(hit2->getT()-distanceToLight) < epsilon)
      color += hit->getMaterial()->Shade
                (ray, hit, directionToLight, lightColor)
return color
```





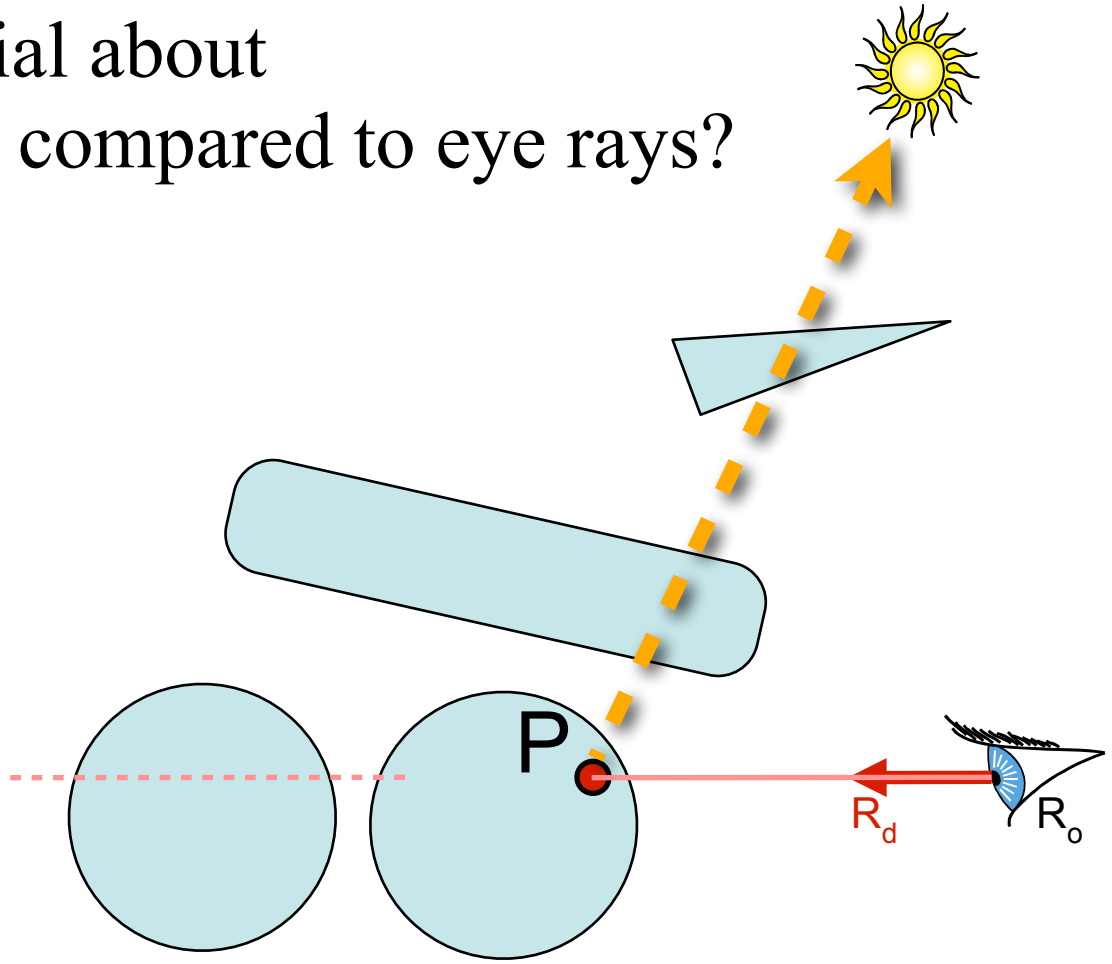
Let's Think About Shadow Rays

- What's special about shadow rays compared to eye rays?



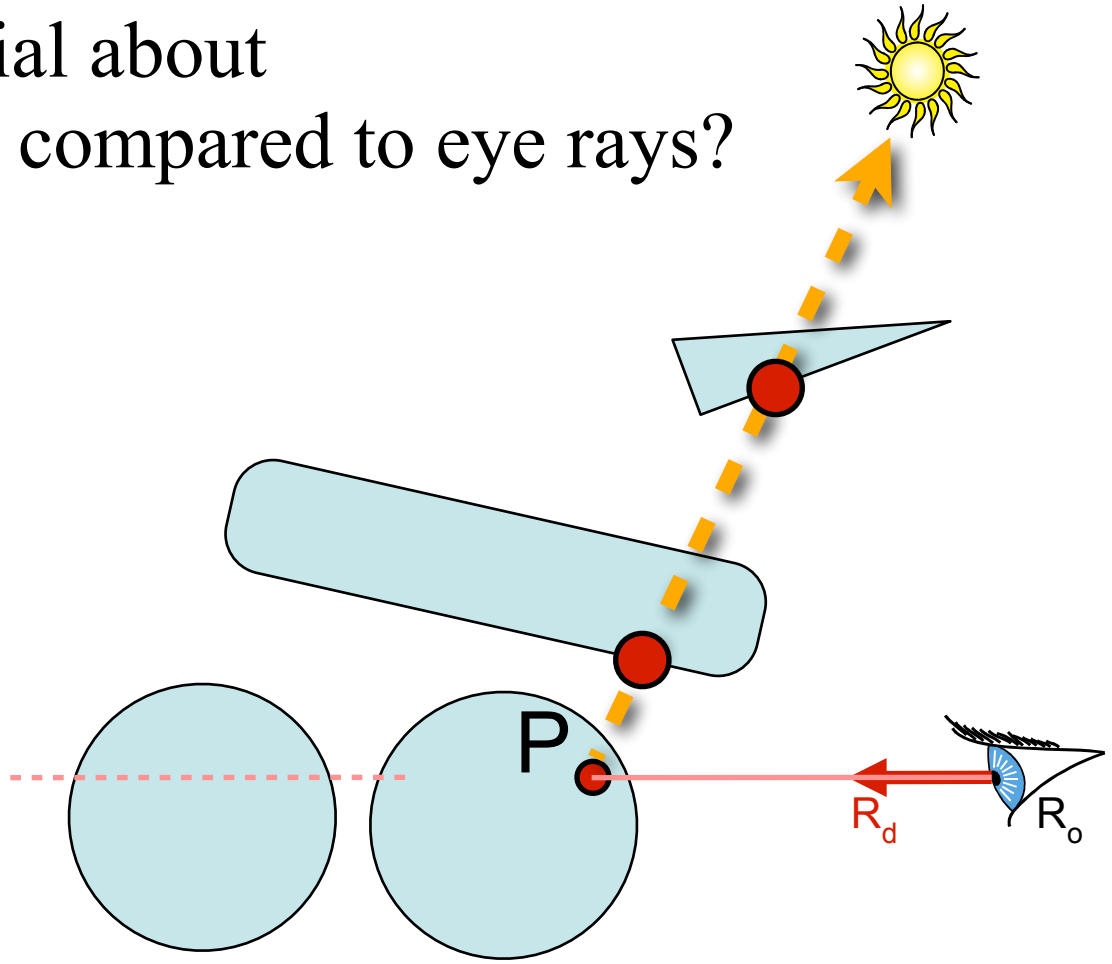
Let's Think About Shadow Rays

- What's special about shadow rays compared to eye rays?



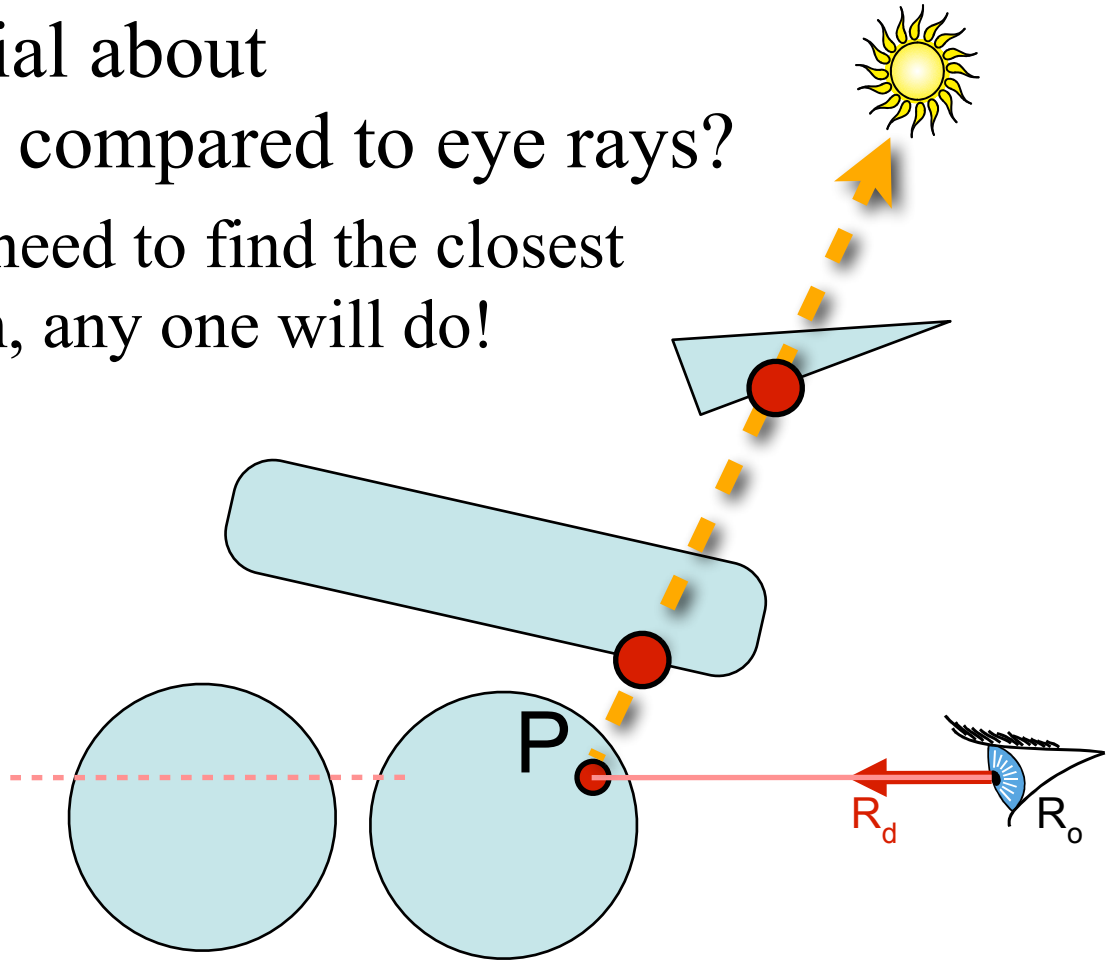
Let's Think About Shadow Rays

- What's special about shadow rays compared to eye rays?

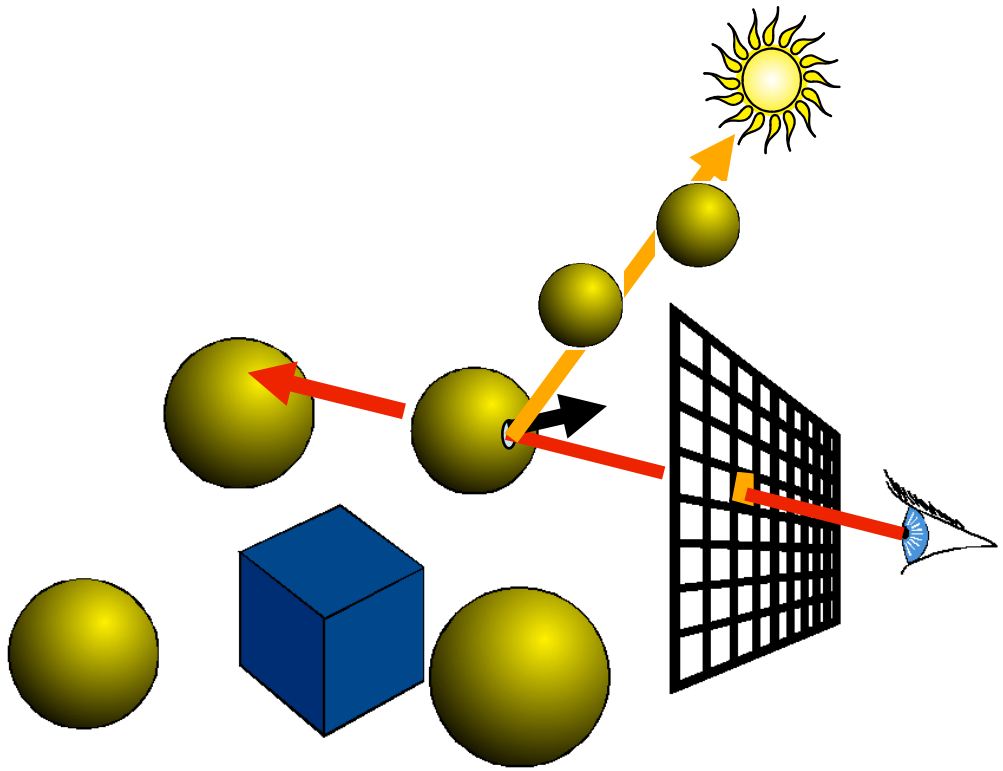


Let's Think About Shadow Rays

- What's special about shadow rays compared to eye rays?
  - We do not need to find the closest intersection, any one will do!

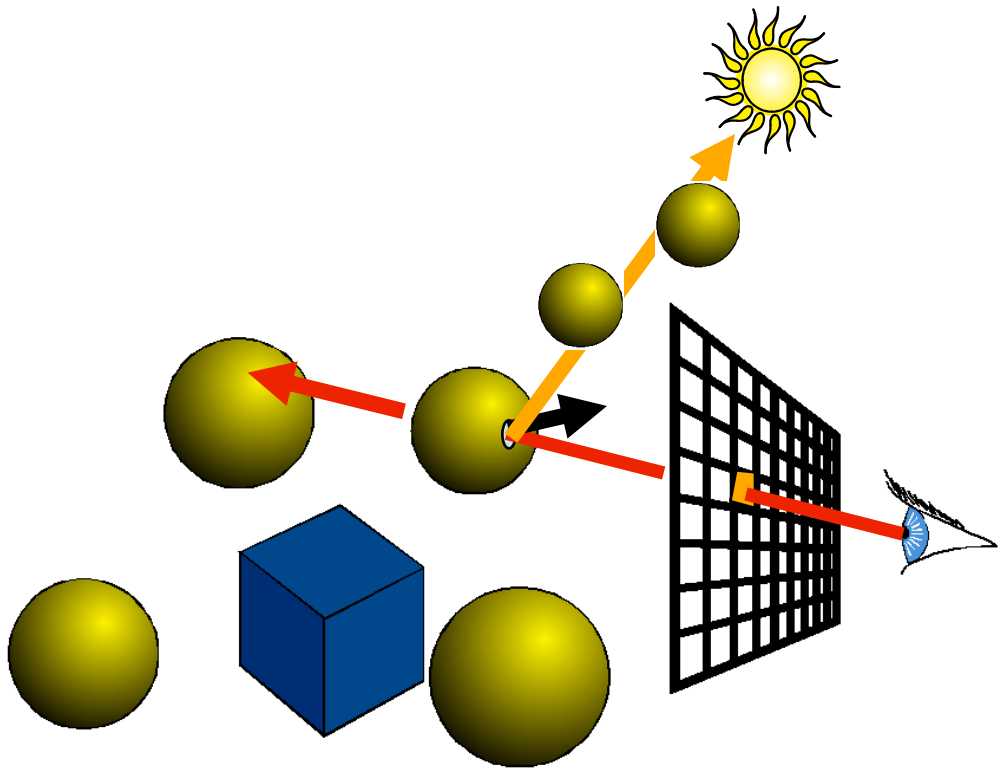


Shadow Optimization



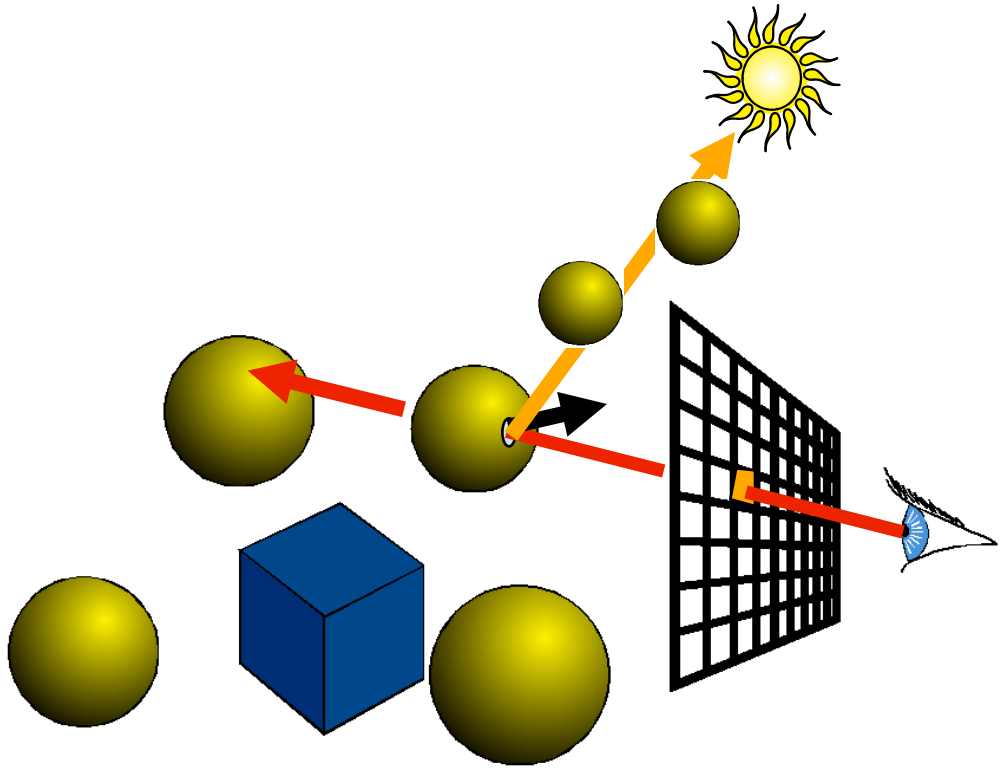
Shadow Optimization

- We only want to know whether there is an intersection, *not* which one is closest



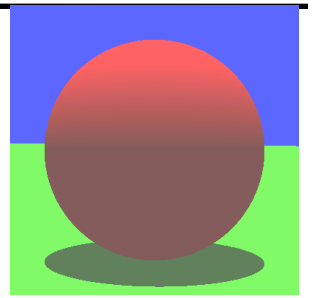
Shadow Optimization

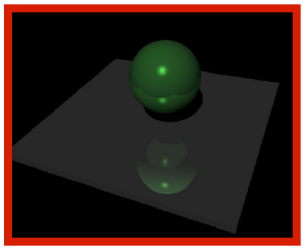
- We only want to know whether there is an intersection, *not* which one is closest
- Special routine `Object3D::intersectShadowRay()`
  - Stops at first intersection found

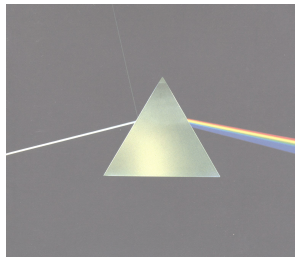


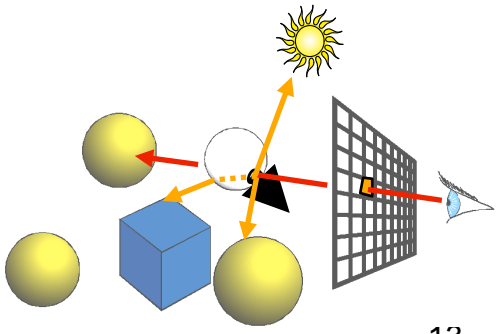
Using a Ray Tracer for..

- Shadows
- Reflection
- Refraction
- Recursive Ray Tracing

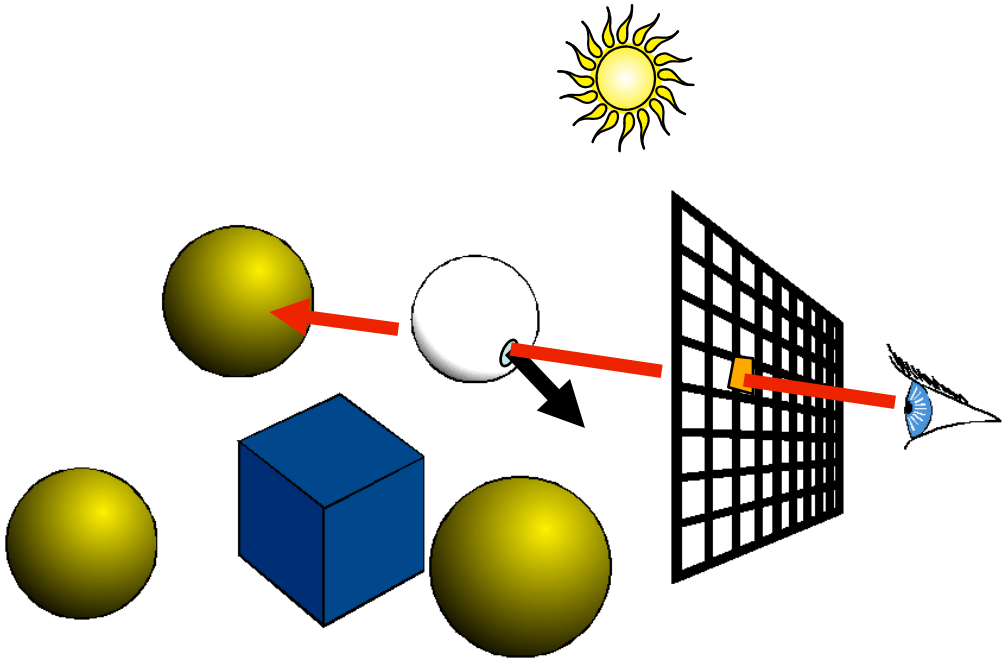


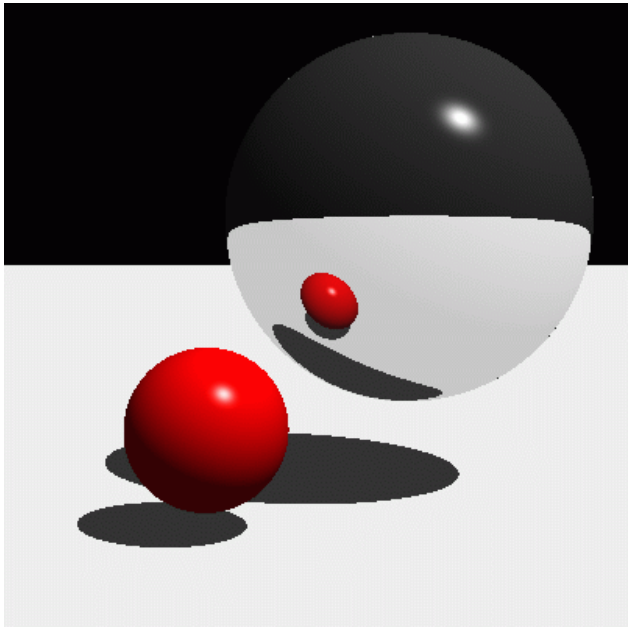






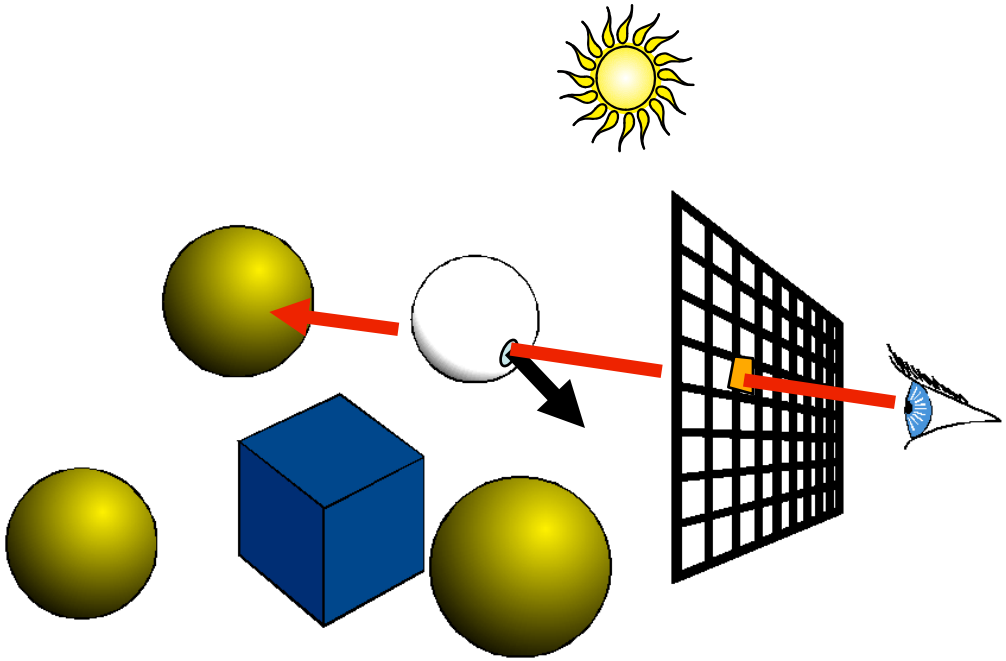
Mirror Reflection

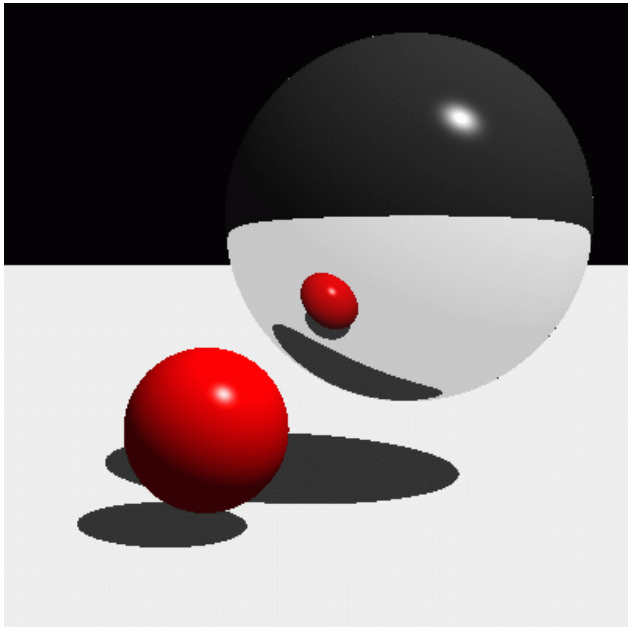




Mirror Reflection

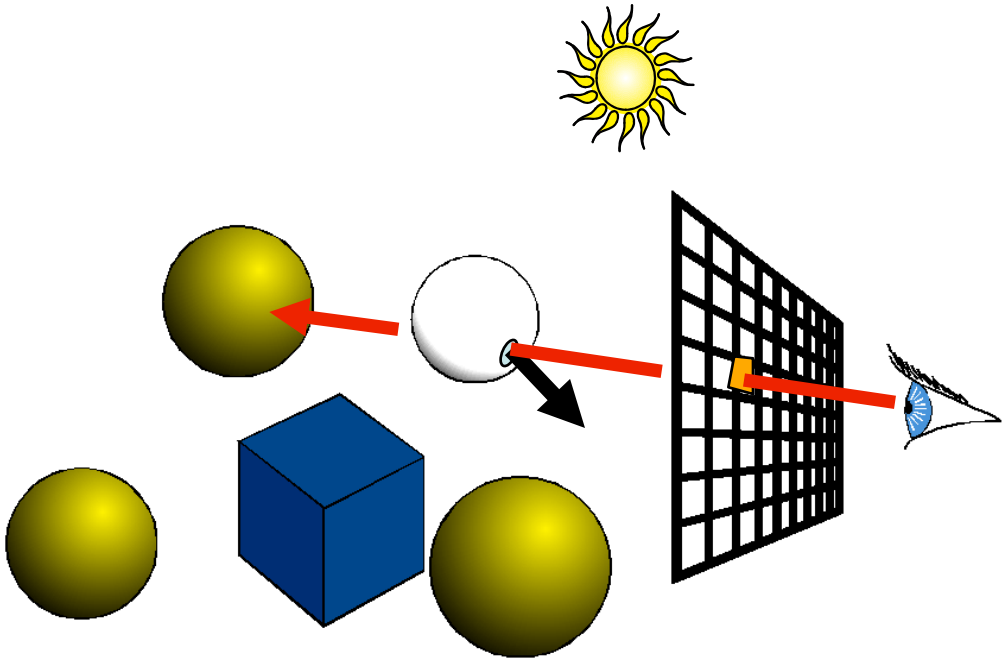
- Cast ray symmetric with respect to the normal

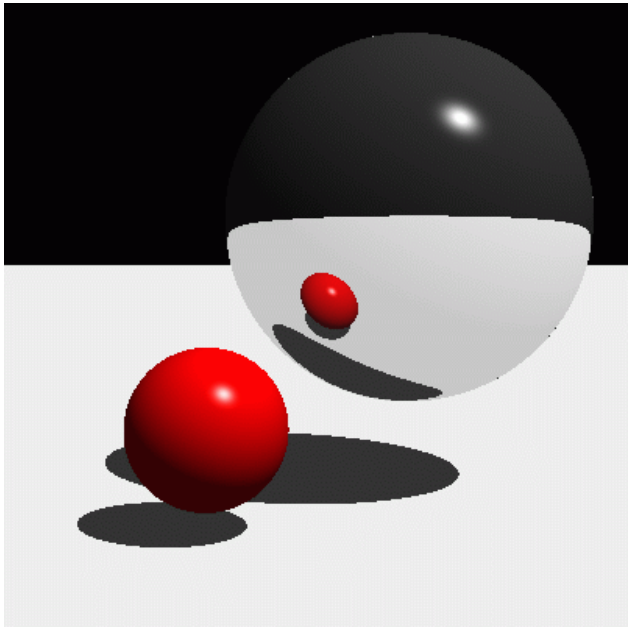




Mirror Reflection

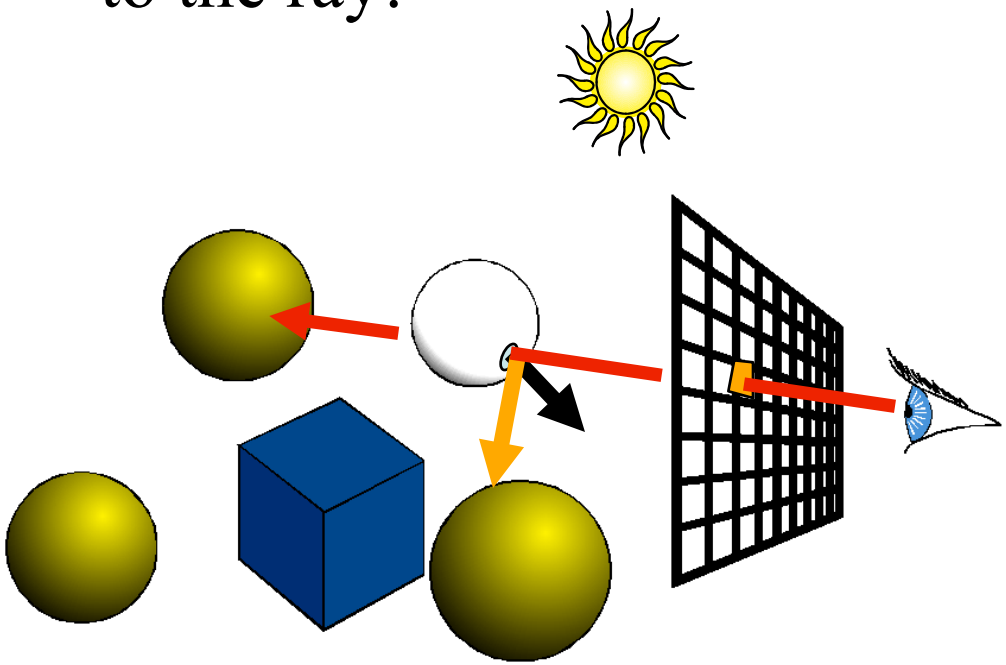
- Cast ray symmetric with respect to the normal
- Multiply by reflection coefficient k_s (color)

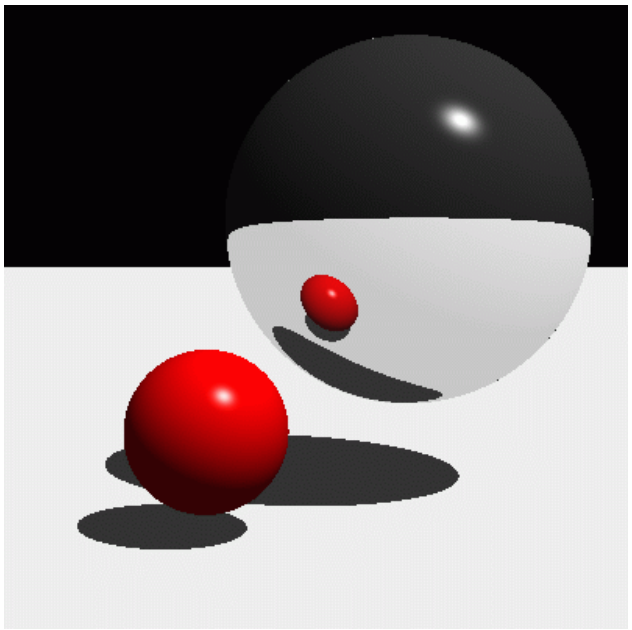




Mirror Reflection

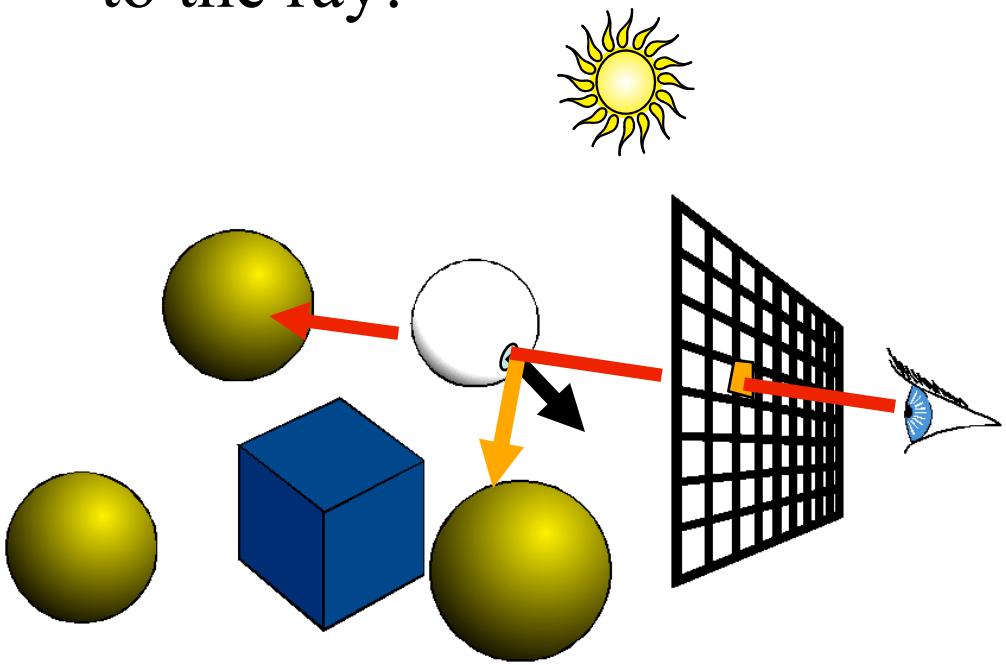
- Cast ray symmetric with respect to the normal
- Multiply by reflection coefficient k_s (color)
- Don't forget to add epsilon to the ray!

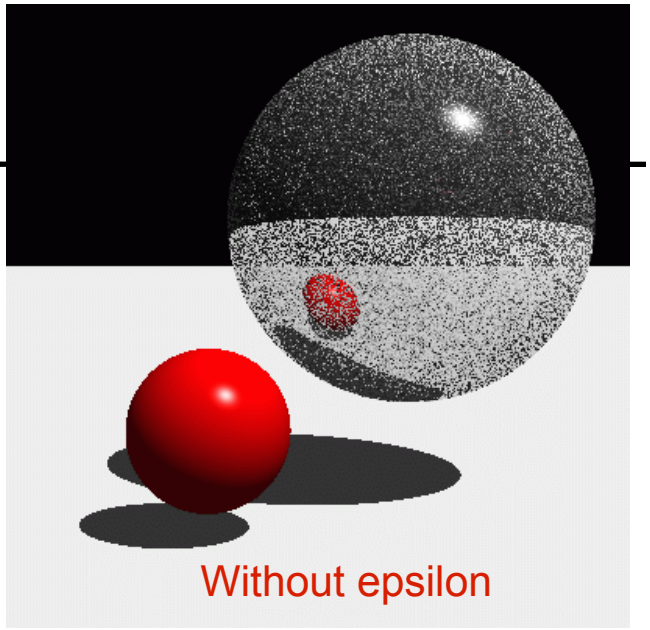


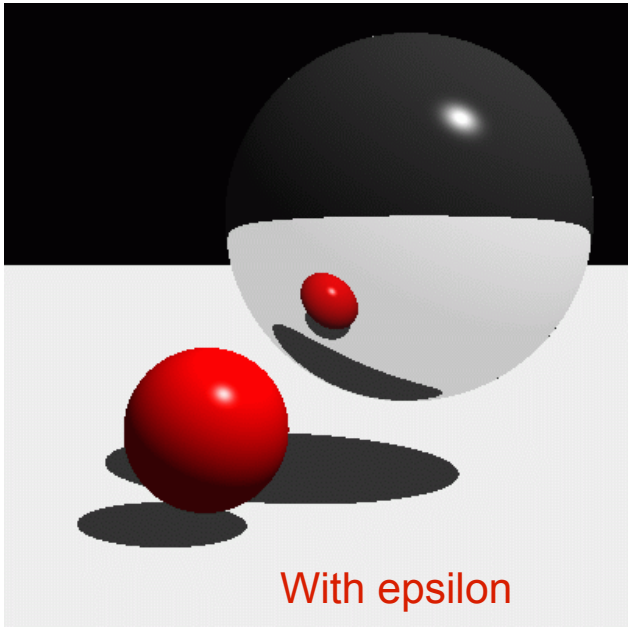


Mirror Reflection

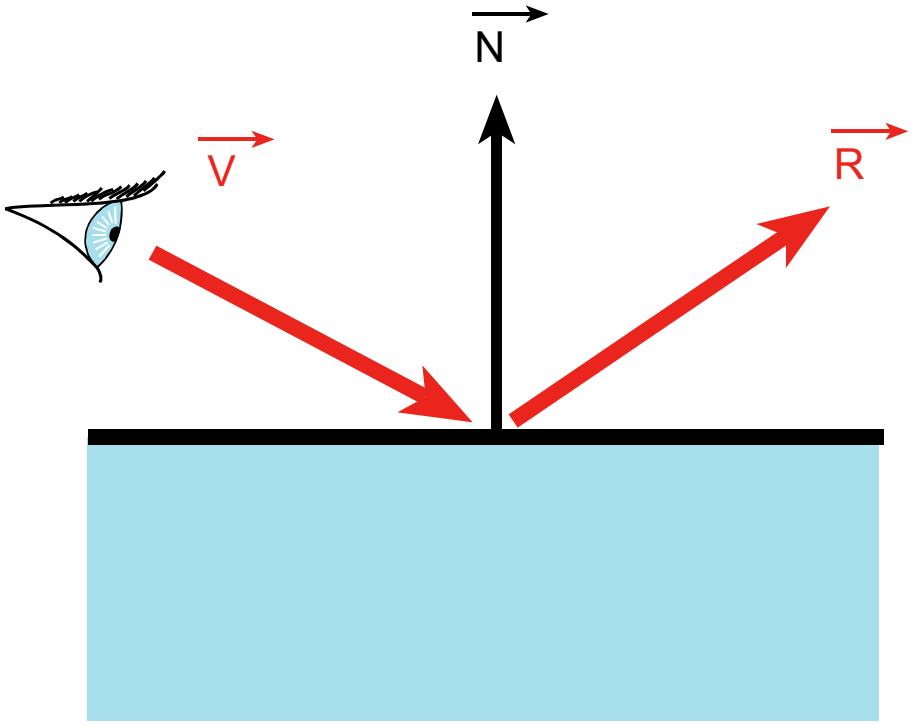
- Cast ray symmetric with respect to the normal
- Multiply by reflection coefficient k_s (color)
- Don't forget to add epsilon to the ray!





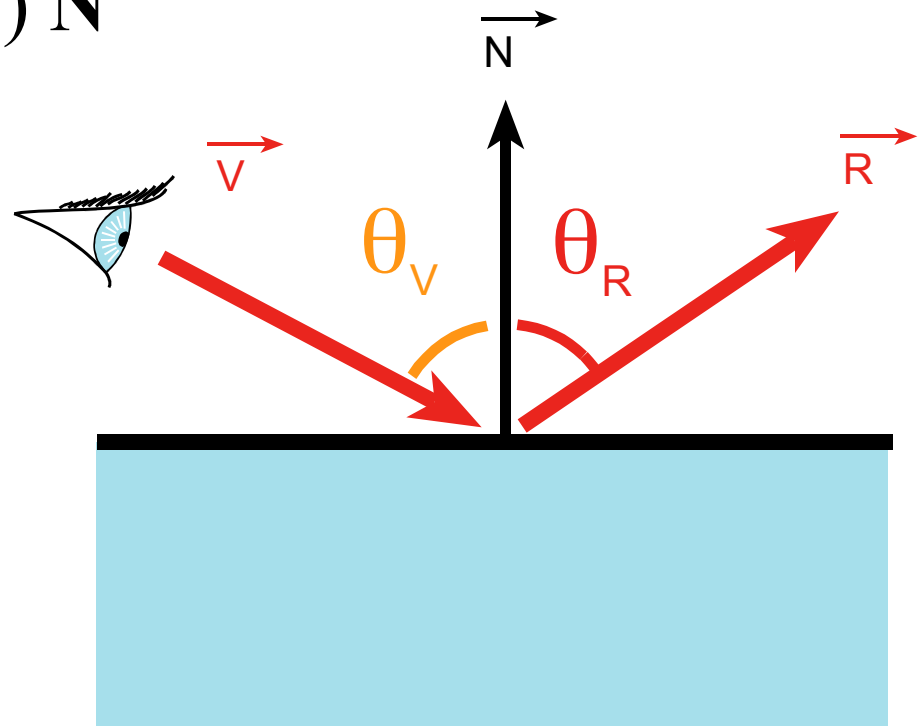


Perfect Mirror Reflection



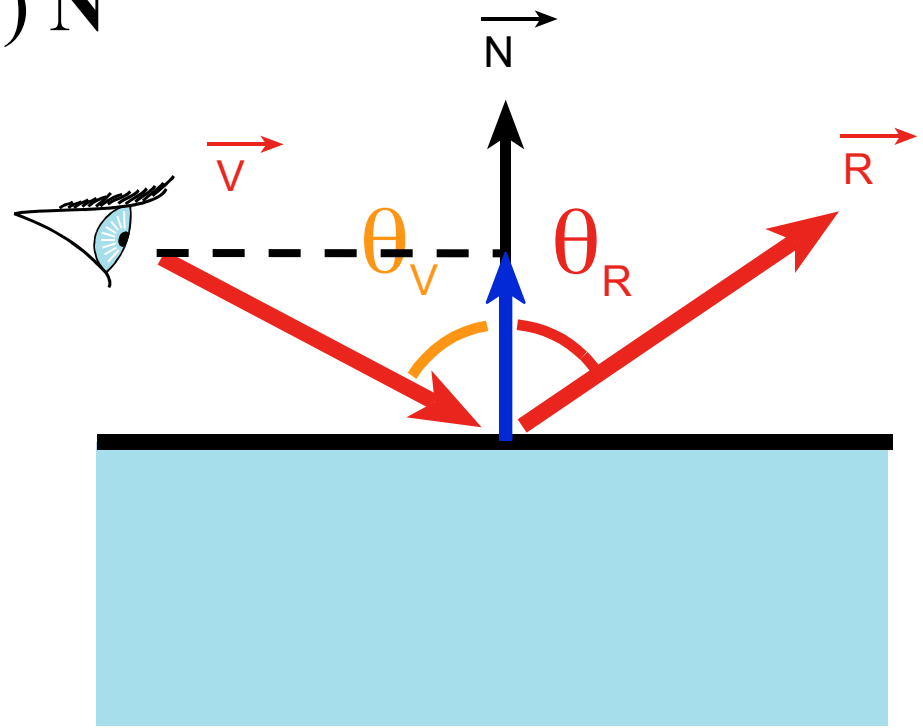
Perfect Mirror Reflection

- Reflection angle = view angle
  - Normal component is negated
  - Remember particle collisions?
- $\mathbf{R} = \mathbf{V} - 2\,(\mathbf{V} \cdot \mathbf{N})\,\mathbf{N}$



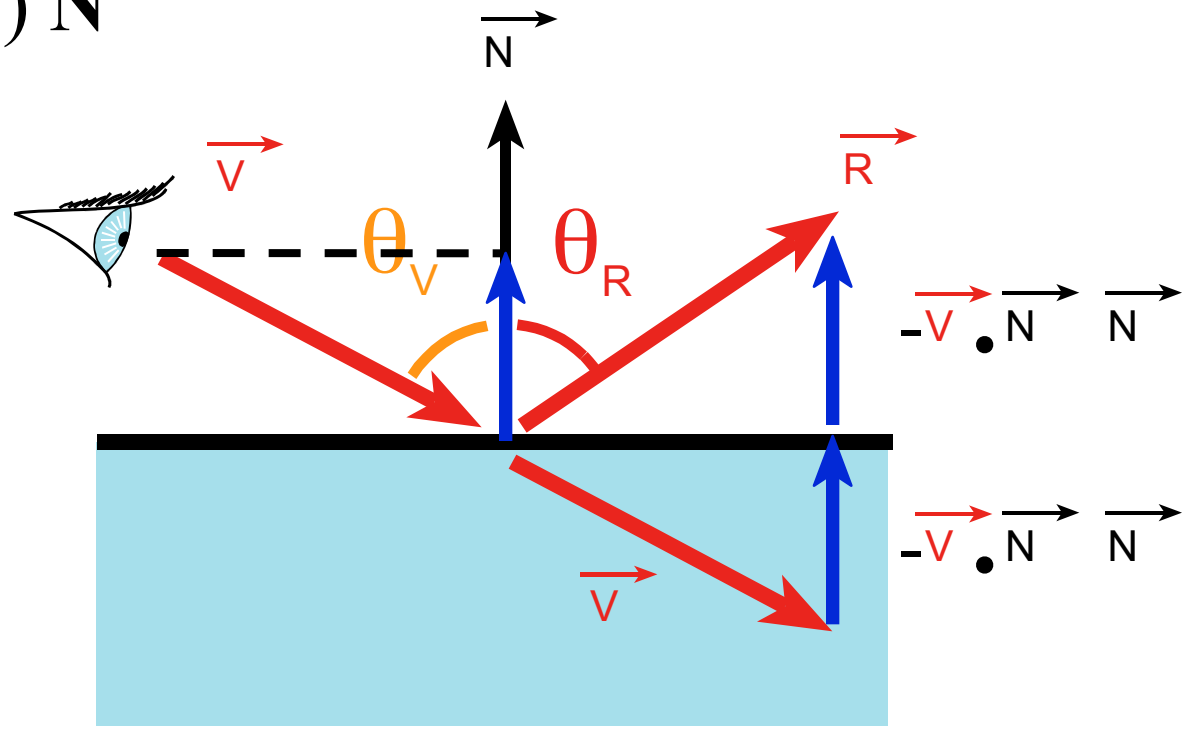
Perfect Mirror Reflection

- Reflection angle = view angle
  - Normal component is negated
  - Remember particle collisions?
- $\mathbf{R} = \mathbf{V} - 2\,(\mathbf{V} \cdot \mathbf{N})\,\mathbf{N}$



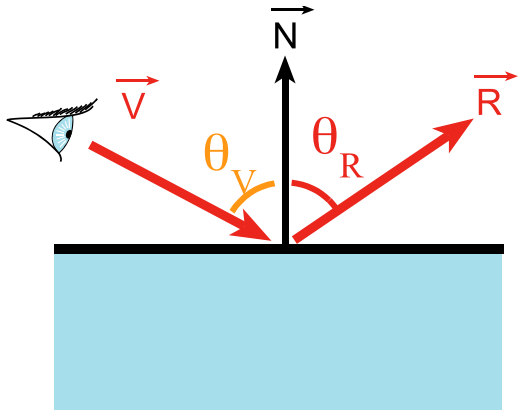
Perfect Mirror Reflection

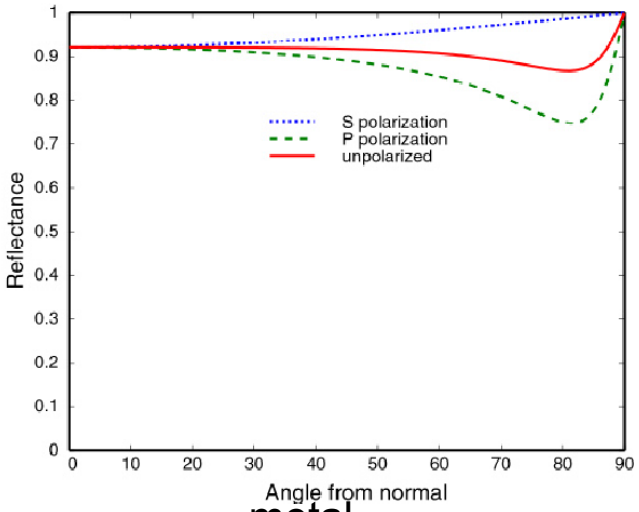
- Reflection angle = view angle
  - Normal component is negated
  - Remember particle collisions?
- $\mathbf{R} = \mathbf{V} - 2\,(\mathbf{V} \cdot \mathbf{N})\,\mathbf{N}$



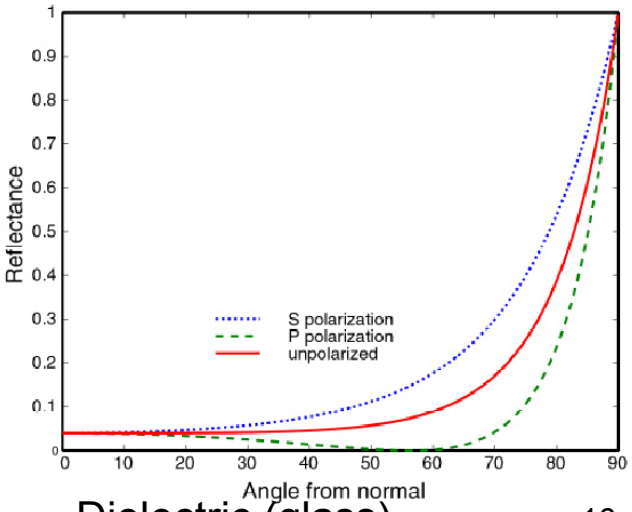
Amount of Reflection

- Traditional ray tracing (hack)
  - Constant k_s
- More realistic
  - Fresnel reflection term (more reflection at grazing angle)
  - Schlick's approximation: $R(\theta)=R_0+(1-R_0)(1-\cos\theta)^5$
- Fresnel makes a big difference!





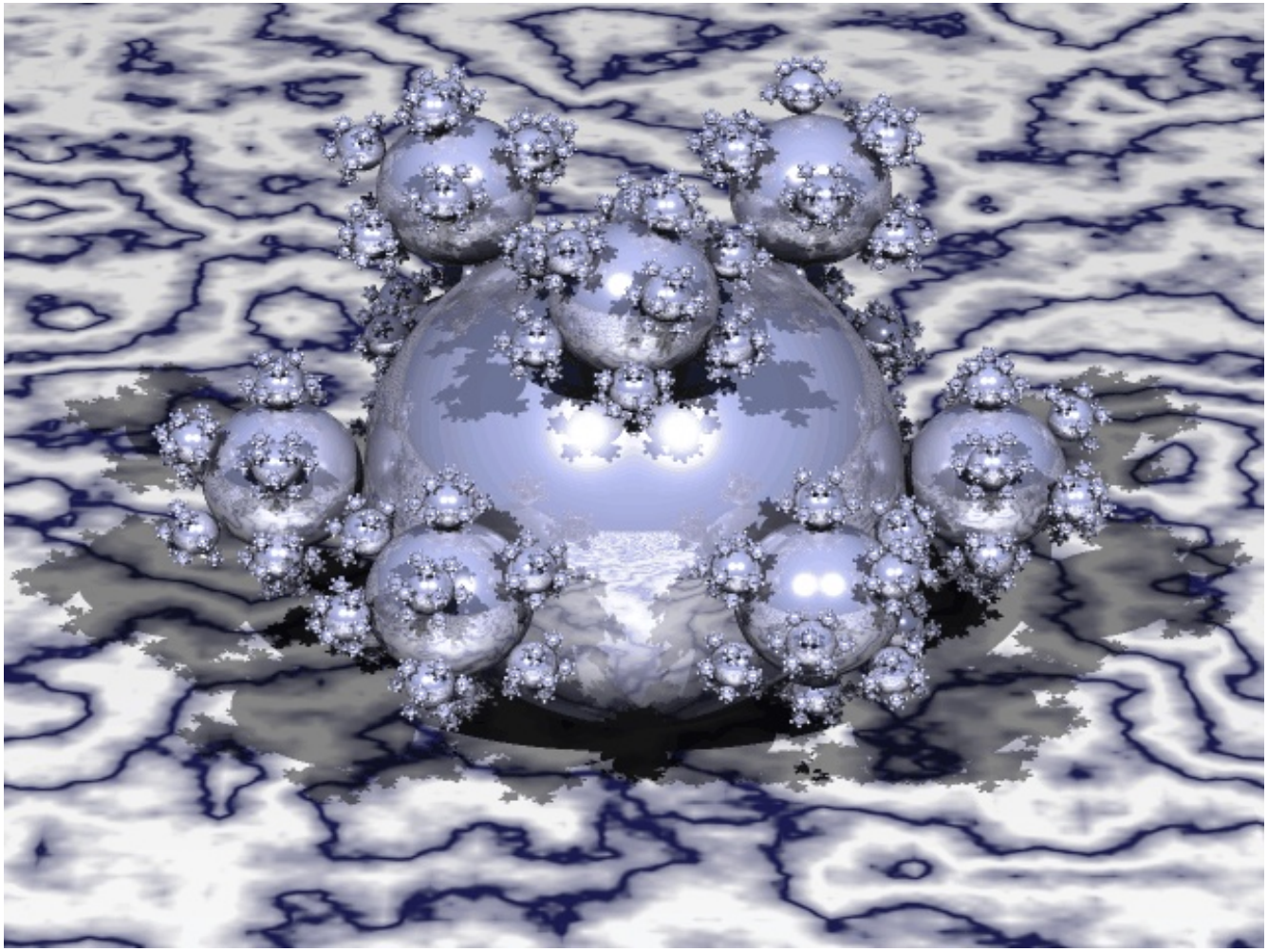
metal



Dielectric (glass)

Questions?

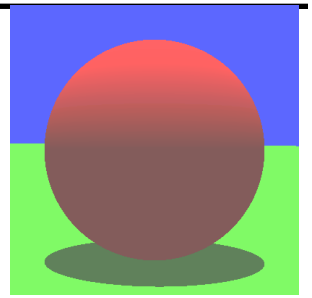
"Sphereflake" fractal

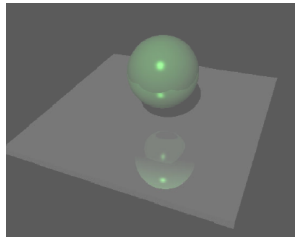


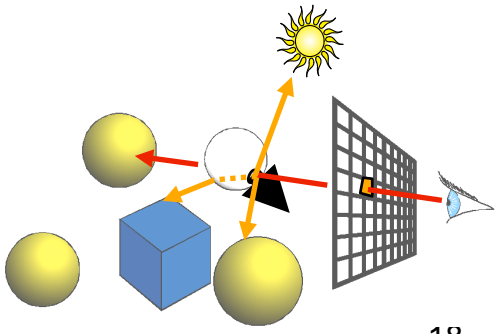
Henrik Wann Jensen

Using a Ray Tracer for..

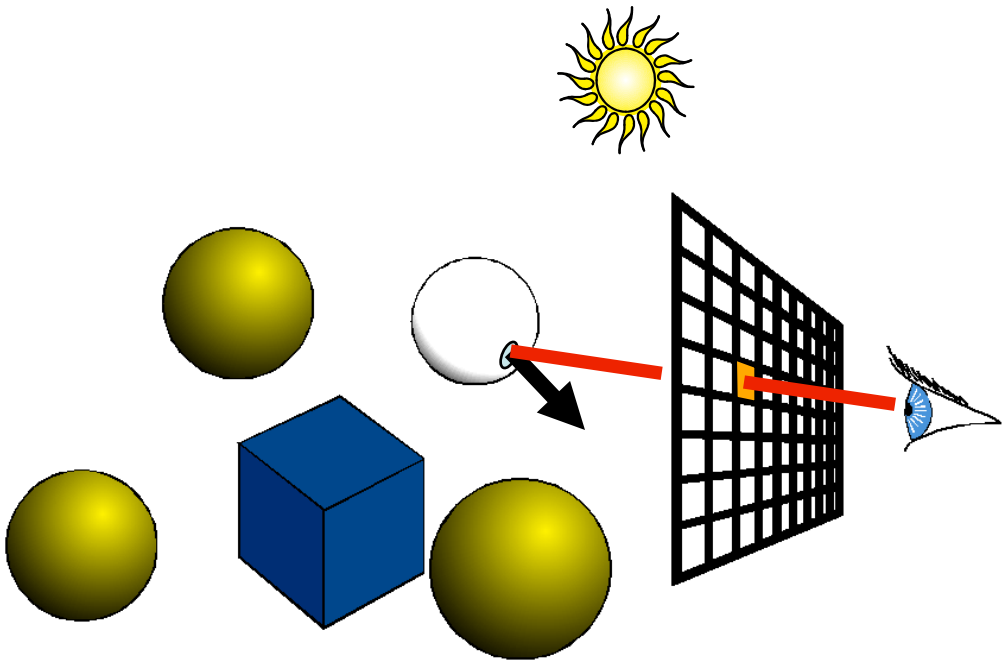
- Shadows
- Reflection
- Refraction
- Recursive Ray Tracing





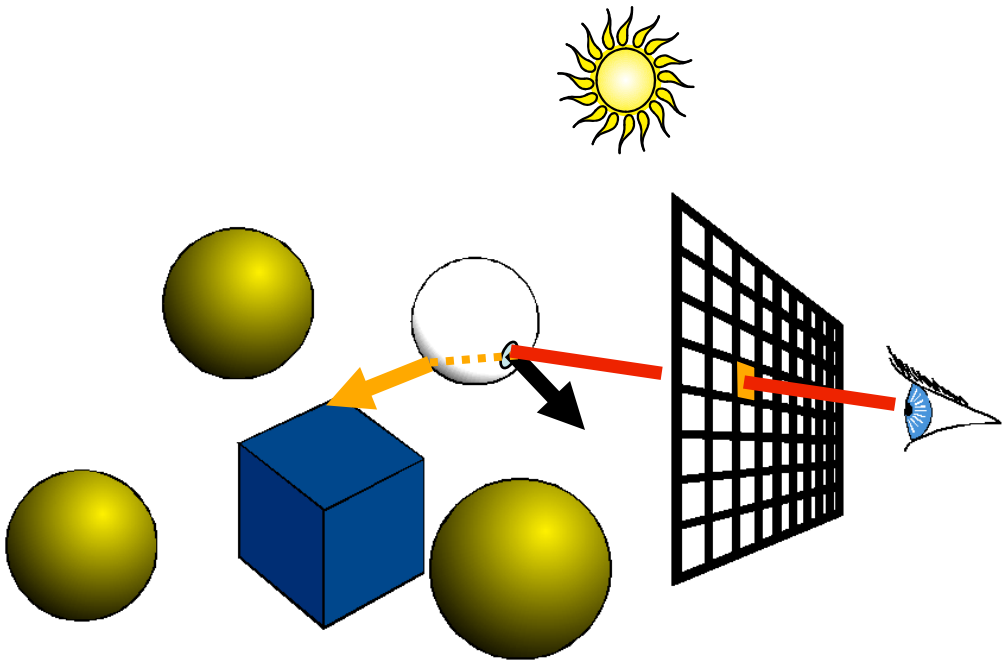


Transparency (Refraction)

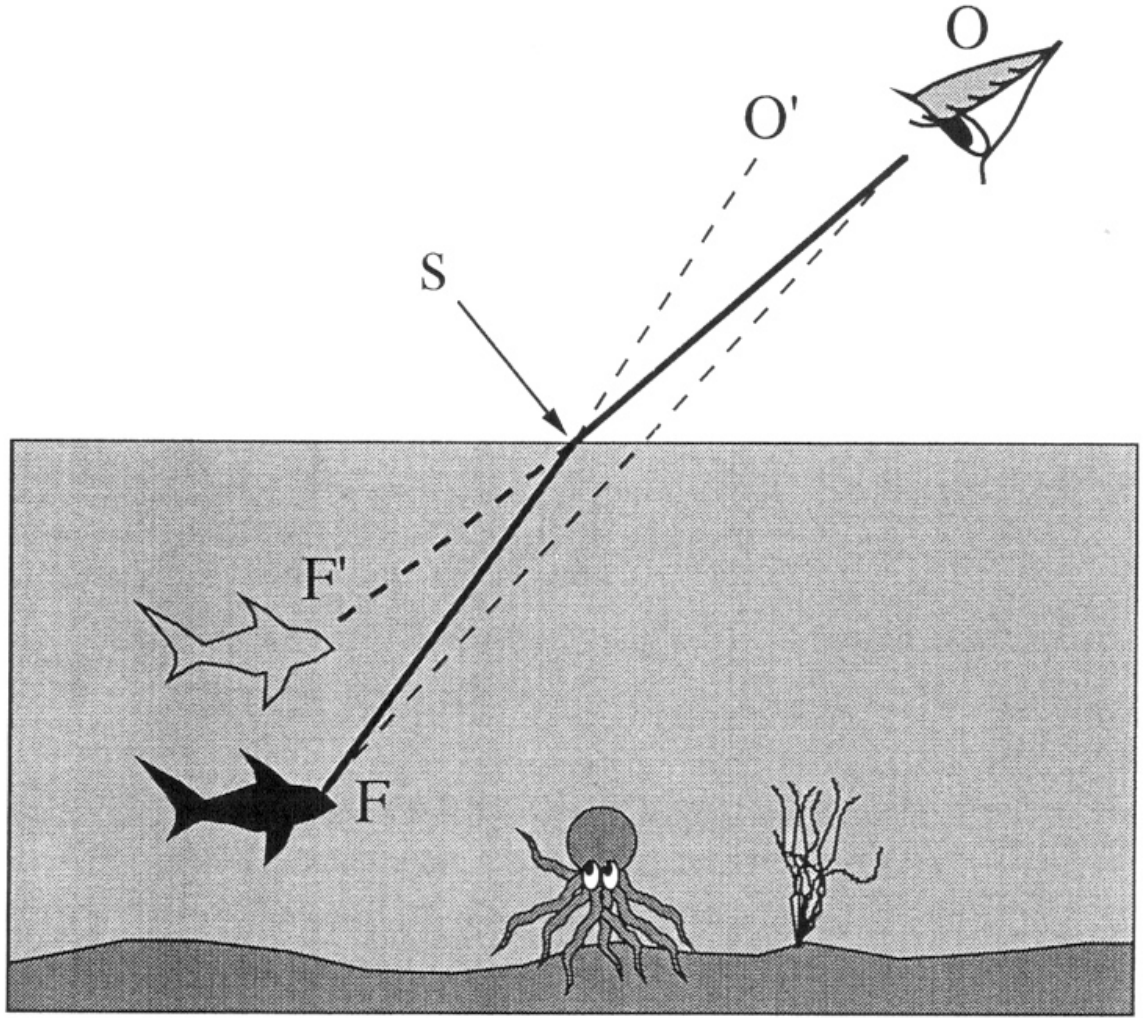


Transparency (Refraction)

- Cast ray in refracted direction
- Multiply by transparency coefficient k_t (color)

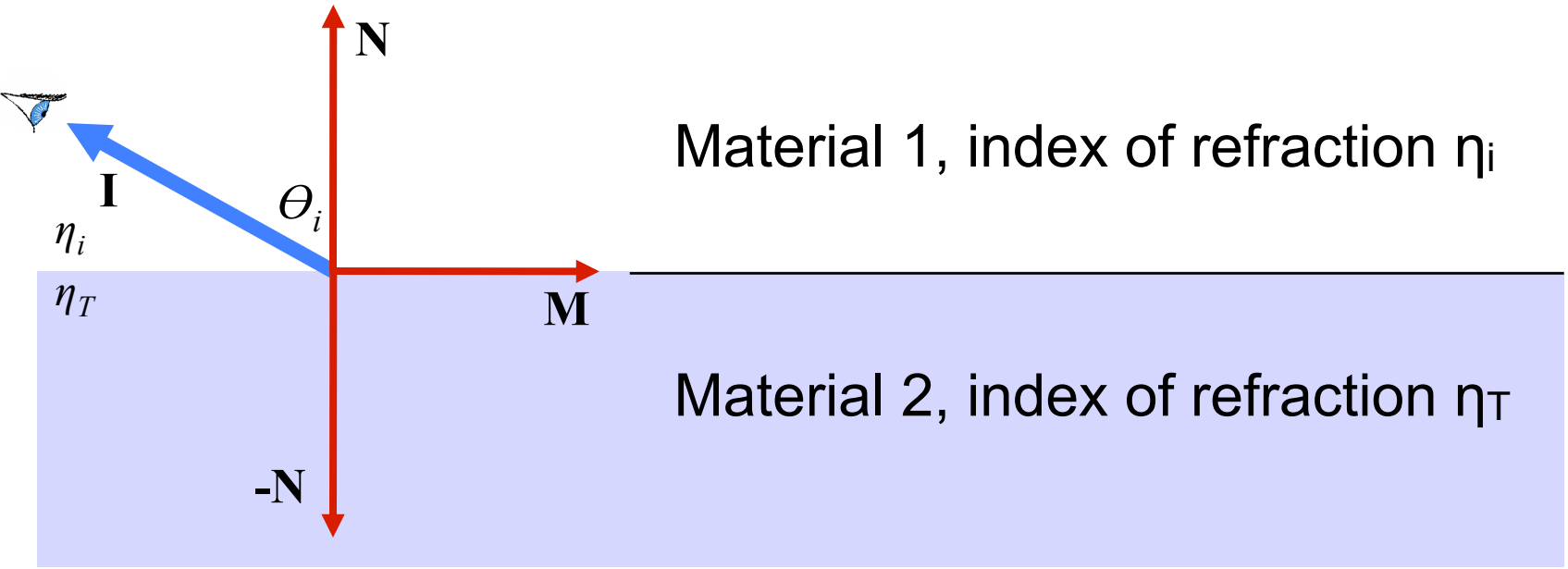


Qualitative Refraction

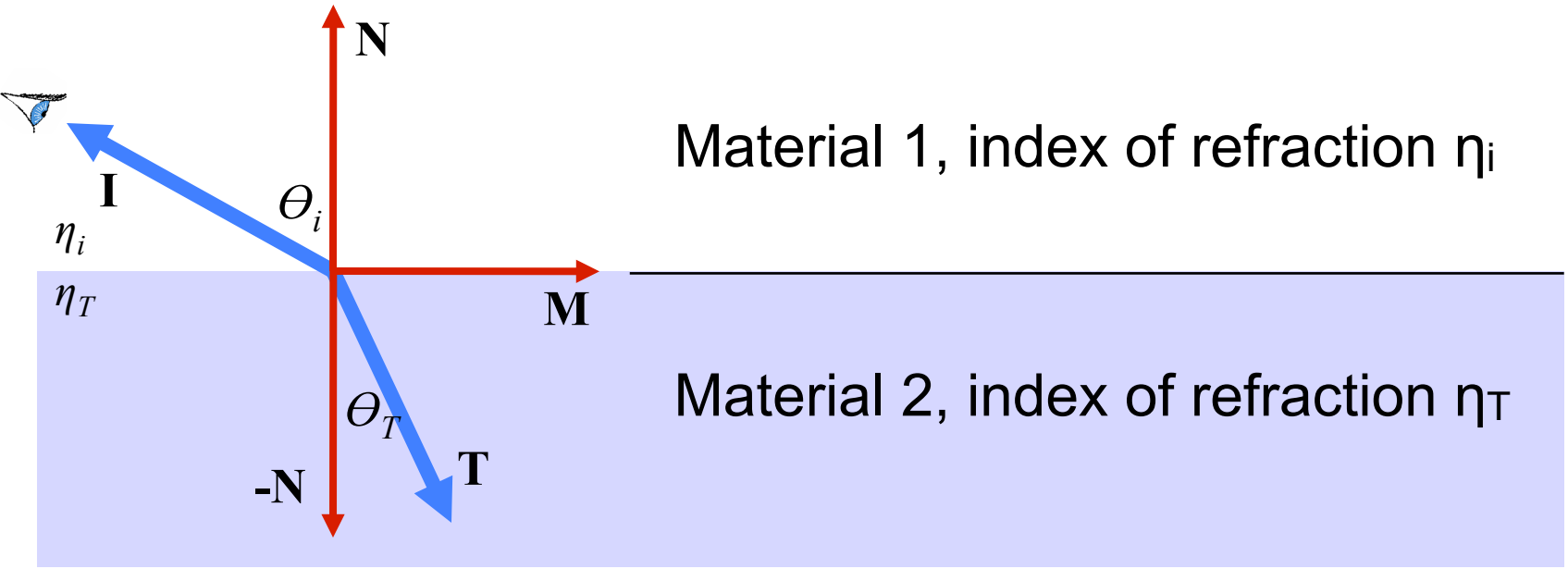


From “Color and Light in Nature” by Lynch and Livingston

Refraction

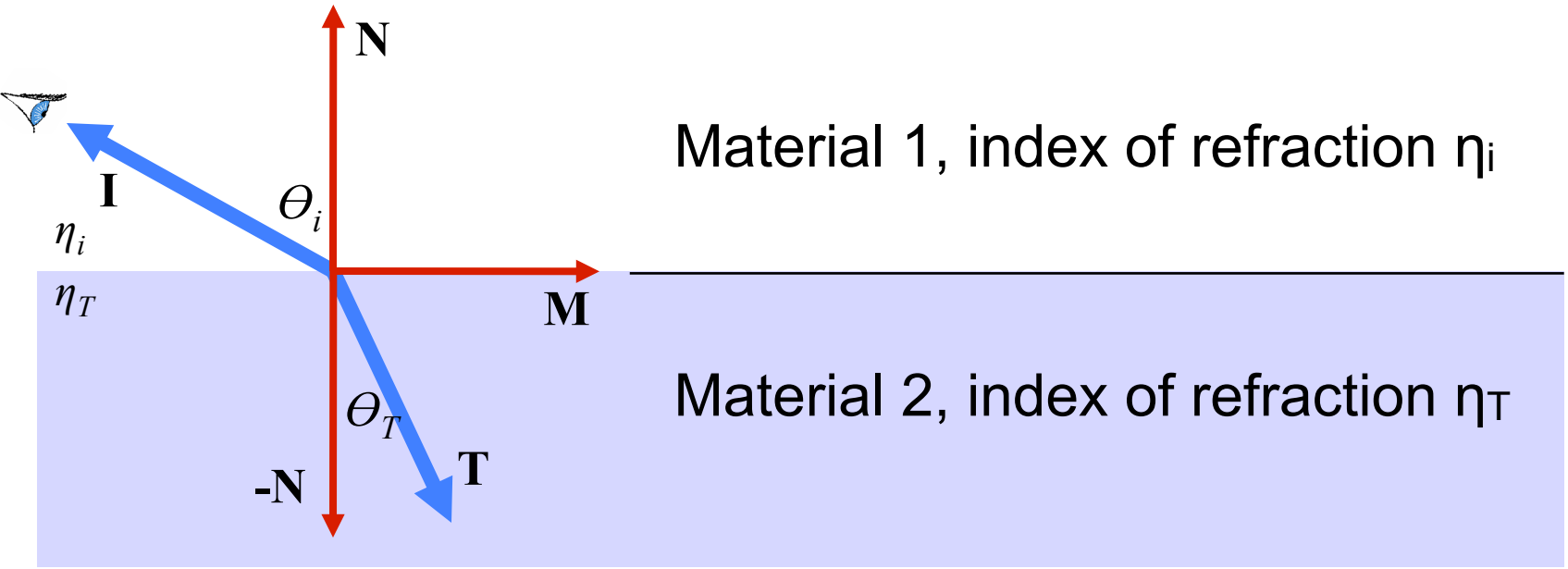

N
I
η_i
Θ_i
η_T
M
-N
Material 1, index of refraction η_i
Material 2, index of refraction η_T

Refraction



Refracted direction **T**?

Refraction



Snell-Descartes Law:

$$\frac{\sin\Theta_i}{\sin\Theta_T} = \frac{\eta_T}{\eta_i} = \eta_r$$

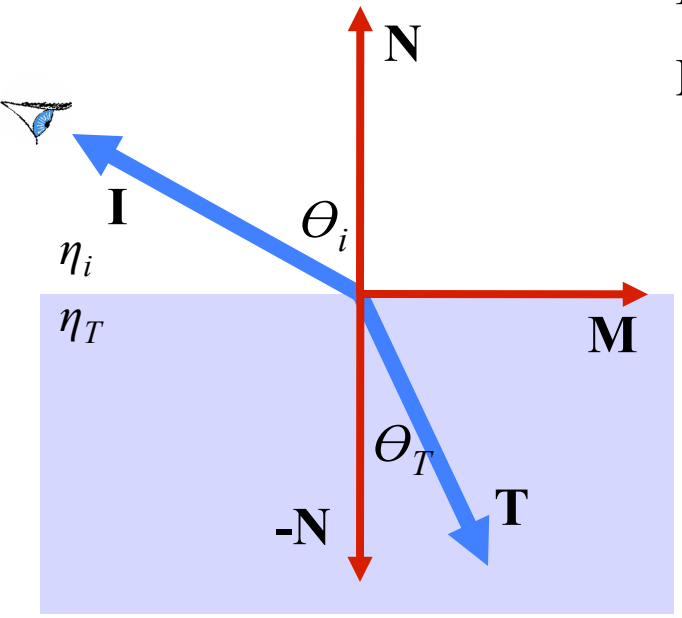
Refracted direction **T**?

Relative index of refraction

Refraction

$$\mathbf{I} = \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i$$

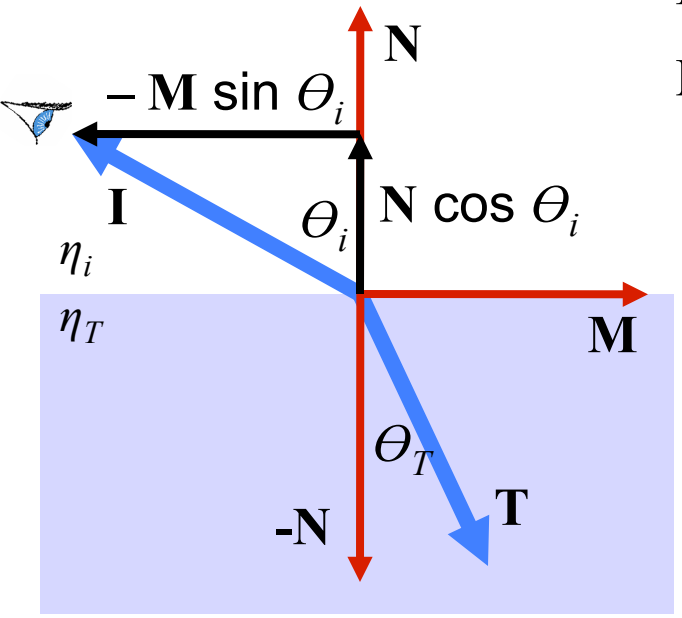
$$\mathbf{M} = (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i$$



Snell-Descartes Law:

$$\frac{\sin\Theta_i}{\sin\Theta_T} = \frac{\eta_T}{\eta_i} = \eta_r$$

Refraction



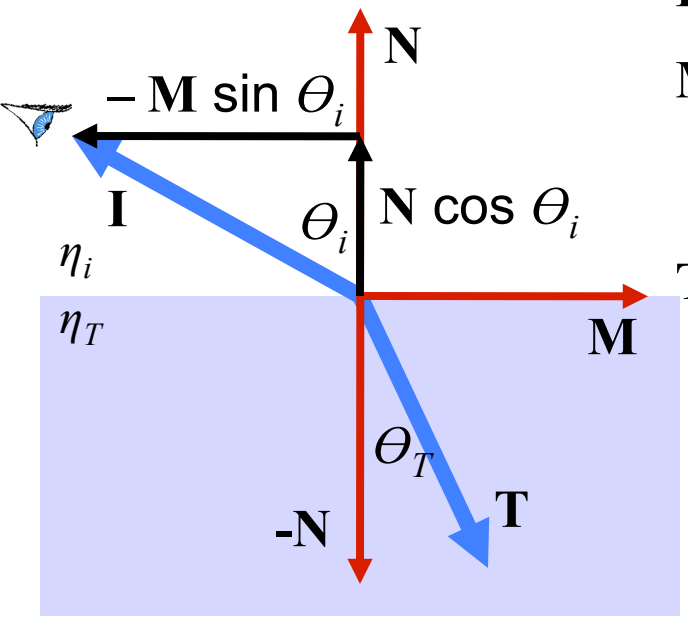
$$\mathbf{I} = \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i$$

$$\mathbf{M} = (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i$$

Snell-Descartes Law:

$$\frac{\sin\Theta_i}{\sin\Theta_T} = \frac{\eta_T}{\eta_i} = \eta_r$$

Refraction



$$\mathbf{I} = \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i$$

$$\mathbf{M} = (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i$$

$$\mathbf{T} = -\mathbf{N}\cos\Theta_T + \mathbf{M}\sin\Theta_T \qquad \textit{Plug M}$$

$$= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\sin\Theta_T / \sin\Theta_i$$

$$= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\,\eta_r \qquad \textit{let's get rid of the cos \& sin}$$

$$= [\,\eta_r\cos\Theta_i - \cos\Theta_T\,]\,\mathbf{N} - \eta_r\mathbf{I}$$

$$= [\,\eta_r\cos\Theta_i - \sqrt{1-\sin^2\Theta_T}\,]\,\mathbf{N} - \eta_r\mathbf{I}$$

$$= [\,\eta_r\cos\Theta_i - \sqrt{1-\eta_r^2\sin^2\Theta_i}\,]\,\mathbf{N} - \eta_r\mathbf{I}$$

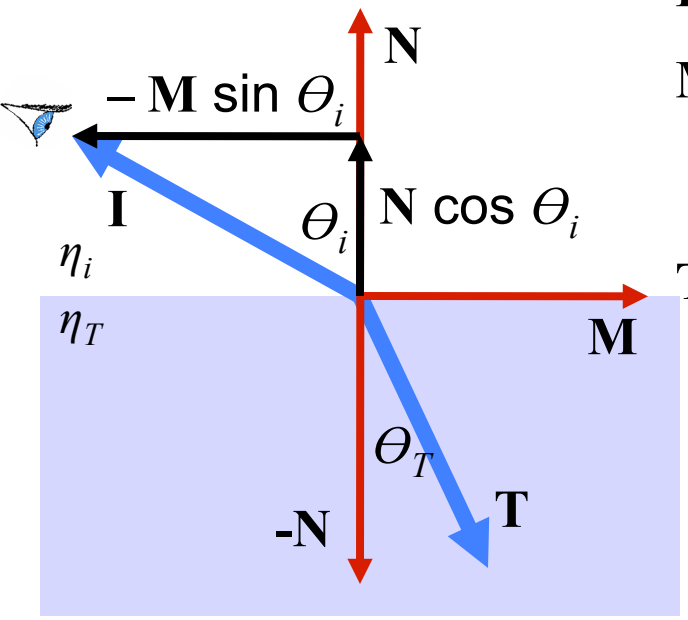
$$= [\,\eta_r\cos\Theta_i - \sqrt{1-\eta_r^2(1-\cos^2\Theta_i)}\,]\,\mathbf{N} - \eta_r\mathbf{I}$$

$$= [\,\eta_r(\mathbf{N}\cdot\mathbf{I}) - \sqrt{1-\eta_r^2(1-(\mathbf{N}\cdot\mathbf{I})^2)}\,]\,\mathbf{N} - \eta_r\mathbf{I}$$

Snell-Descartes Law:

$$\frac{\sin\Theta_i}{\sin\Theta_T} = \frac{\eta_T}{\eta_i} = \eta_r$$

Refraction



$$\mathbf{I} = \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i$$

$$\mathbf{M} = (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i$$

$$\mathbf{T} =$$

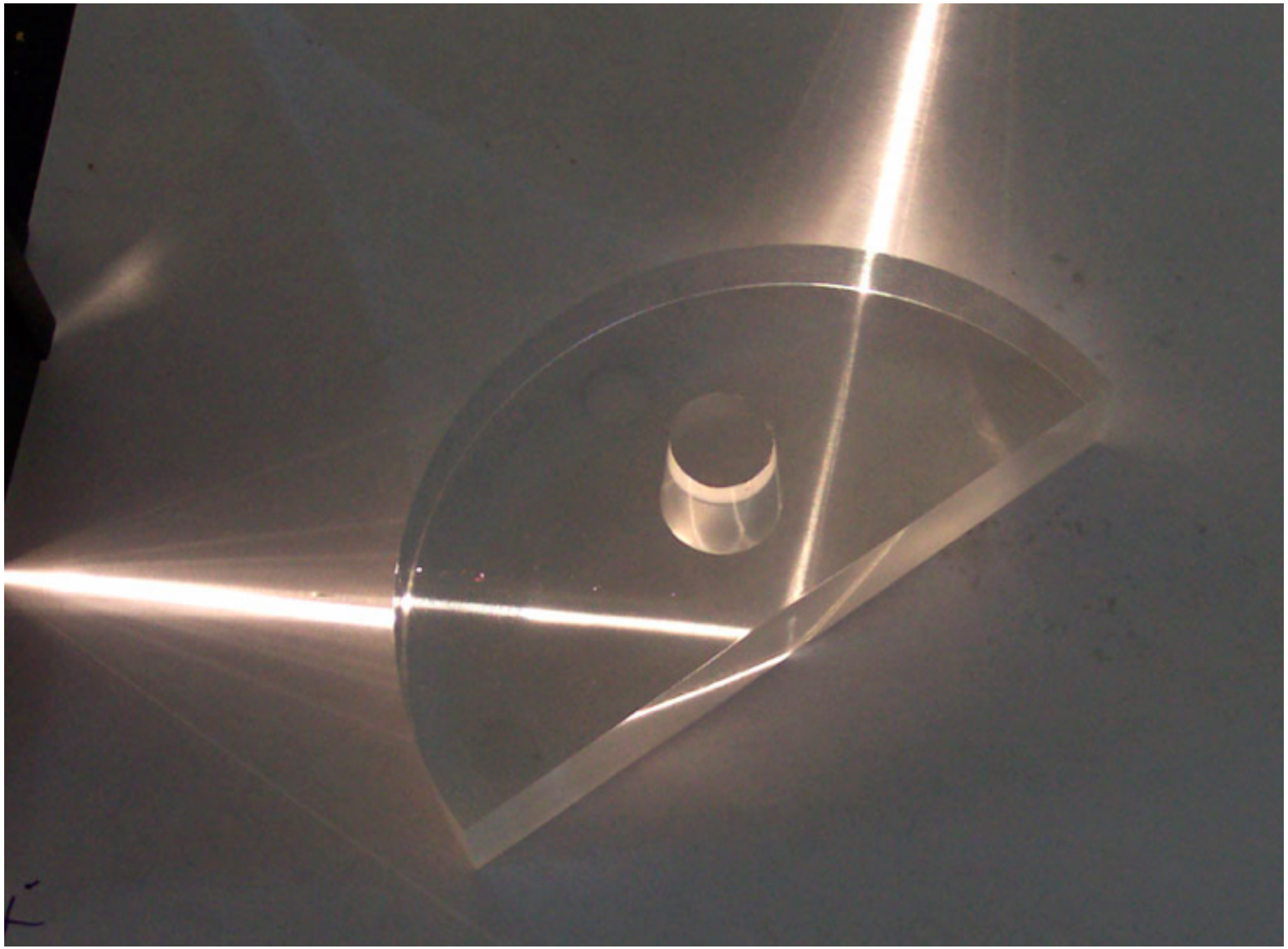
$$= [\, \eta_r\,(\mathbf{N}\cdot\mathbf{I}) - \sqrt{1 - \eta_r^2\,(1 - (\mathbf{N}\cdot\mathbf{I})^2)}\,]\,\mathbf{N} - \eta_r\,\mathbf{I}$$

- **Total internal reflection** when the square root is imaginary (no refraction, just reflection)
- Don't forget to normalize **T!**

Snell-Descartes Law:

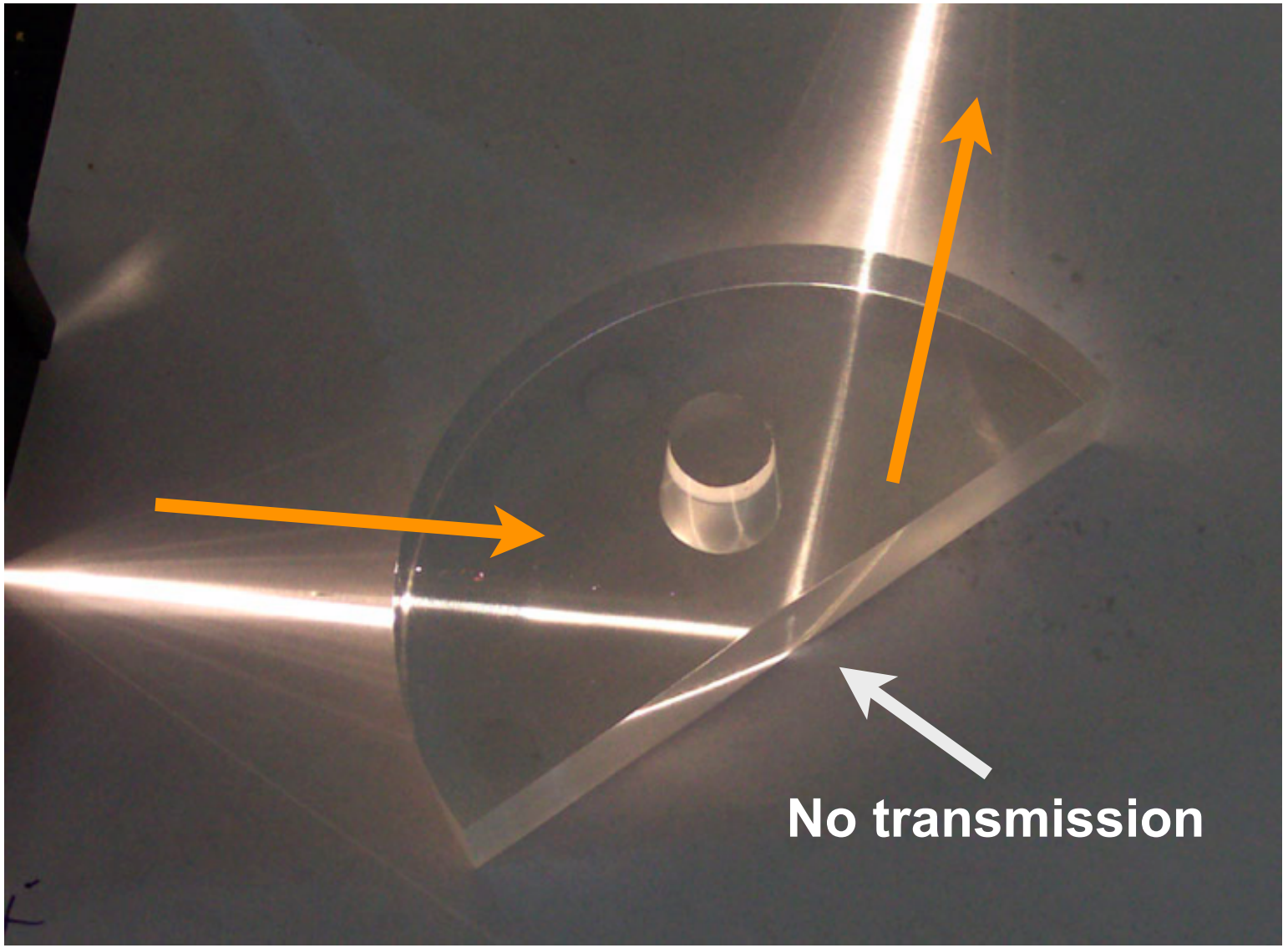
$$\frac{\sin\Theta_i}{\sin\Theta_T} = \frac{\eta_T}{\eta_i} = \eta_r$$

Total Internal Reflection



Wikipedia user Fir0002

Total Internal Reflection



Wikipedia user Fir0002

Total Internal Reflection

Fig. 3.7A The optical manhole. From under water, the entire celestial hemisphere is compressed into a circle only 97.2° across. The dark boundary defining the edges of the manhole is not sharp due to surface waves. The rays are analogous to the crepuscular type seen in hazy air, Section 1.9. (Photo by D. Granger)

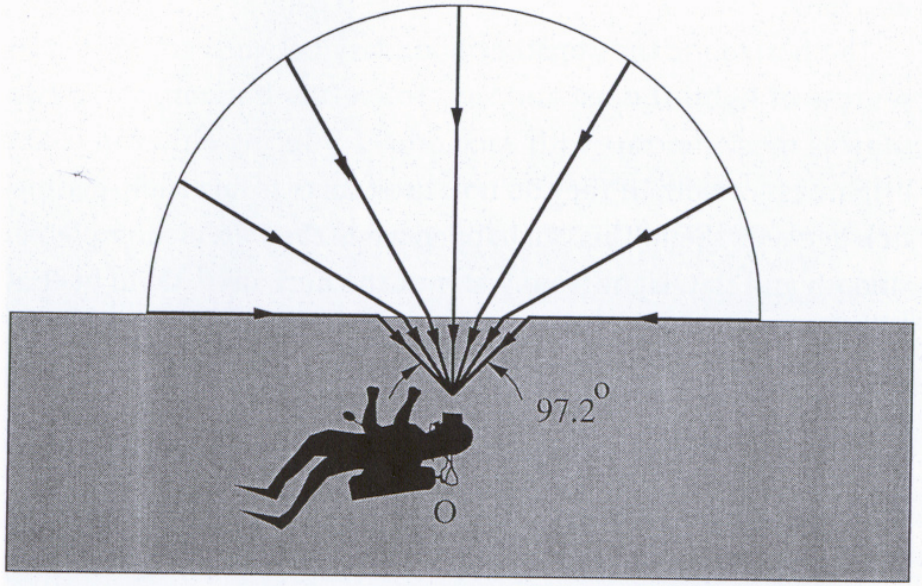
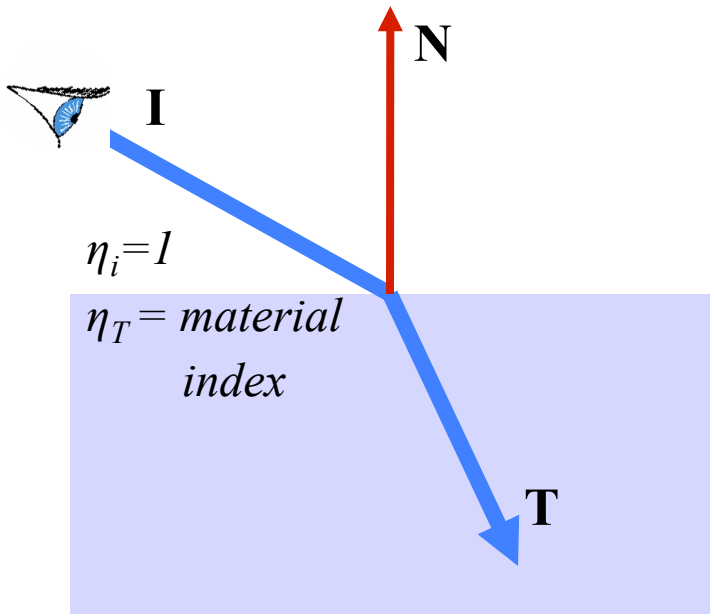


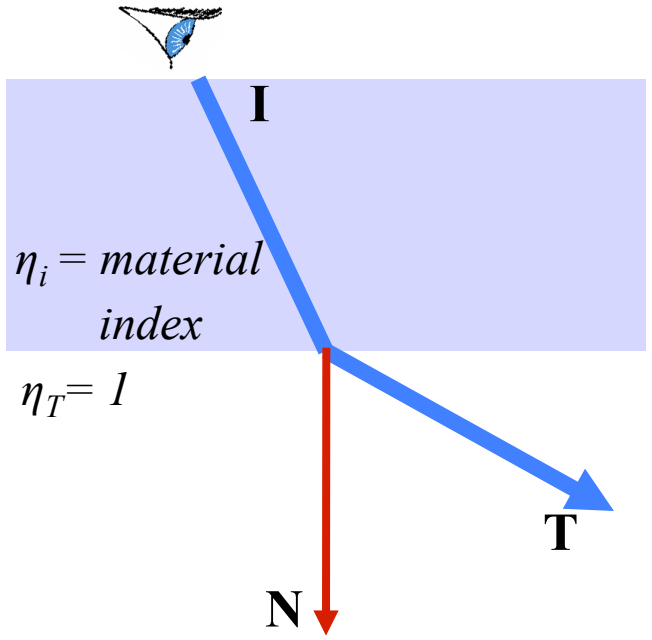
Fig. 3.7B The optical manhole. Light from the horizon (angle of incidence = 90°) is refracted downward at an angle of 48.6°. This compresses the sky into a circle with a diameter of 97.2° instead of its usual 180°.

From "Color and Light in Nature" by Lynch and Livingston

Refraction & Sidedness of Objects

- Make sure you know whether you're entering or leaving the transmissive material:





- Note: We won't ask you to trace rays through intersecting transparent objects **:-)**

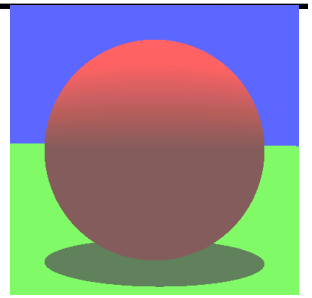
Cool Refraction Demo

- Enright, D., Marschner, S. and Fedkiw, R., SIGGRAPH 2002

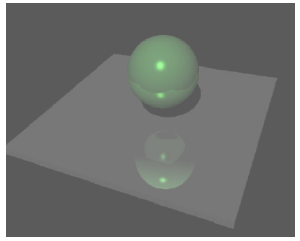
- Physical fluid simulation combined with ray traced rendering

Using a Ray Tracer for..

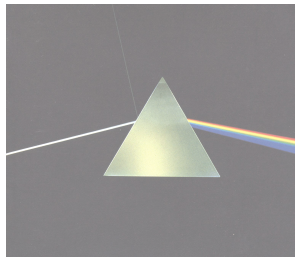
- Shadows



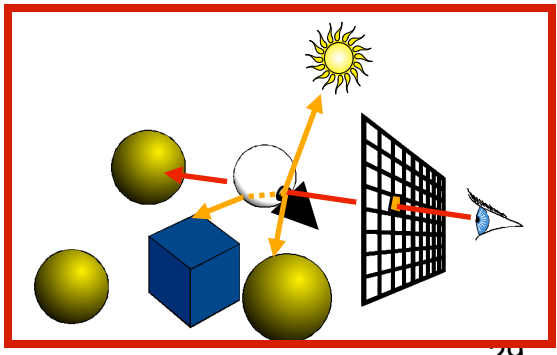
- Reflection



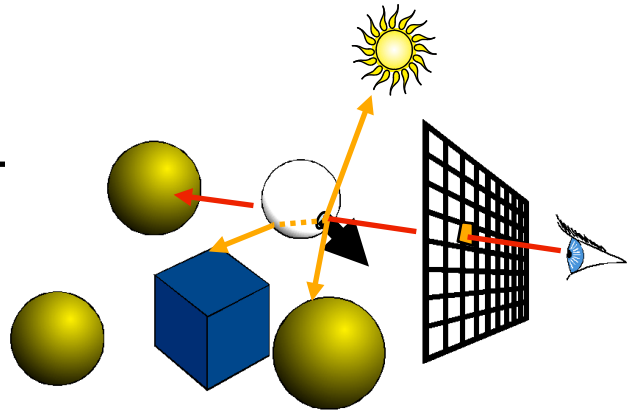
- Refraction



- Recursive Ray Tracing

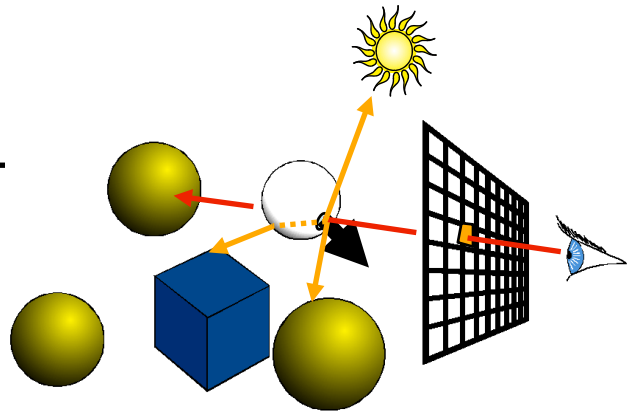


Recap: Ray Tracing



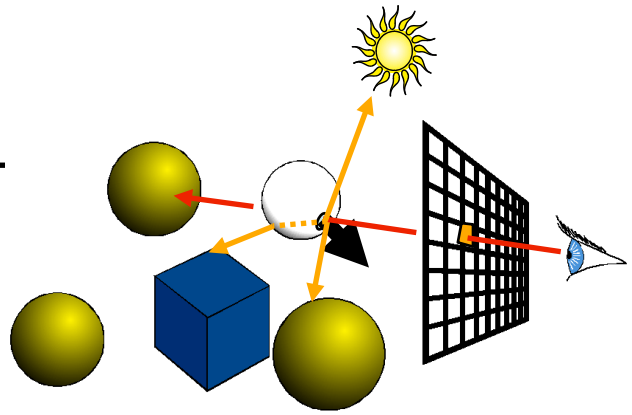
Recap: Ray Tracing

```
trace ray
```



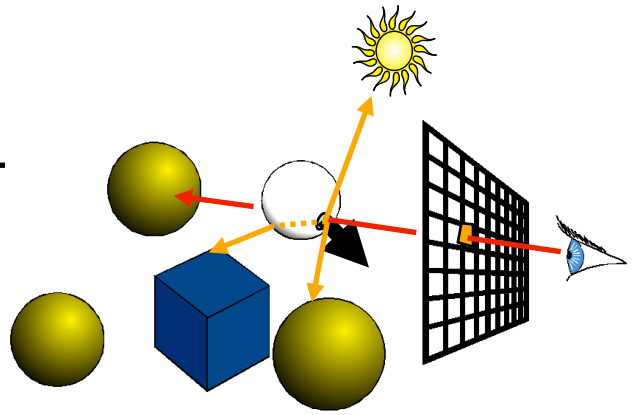
Recap: Ray Tracing

```
trace ray
    Intersect all objects
```



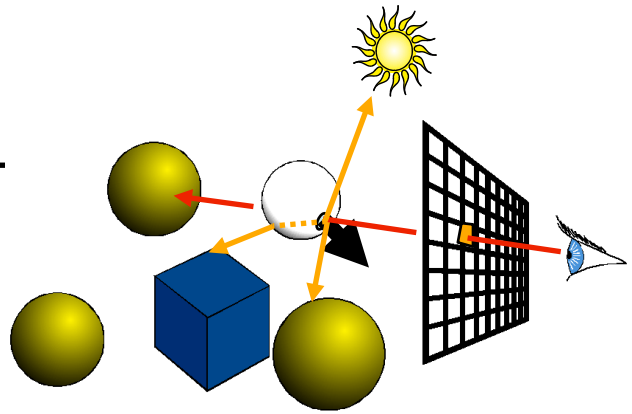
Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
```



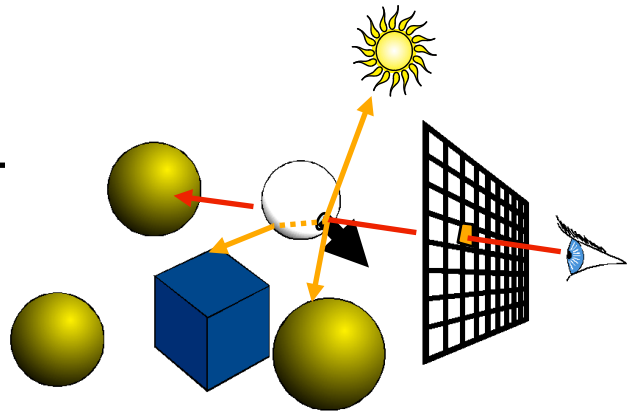
Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
```



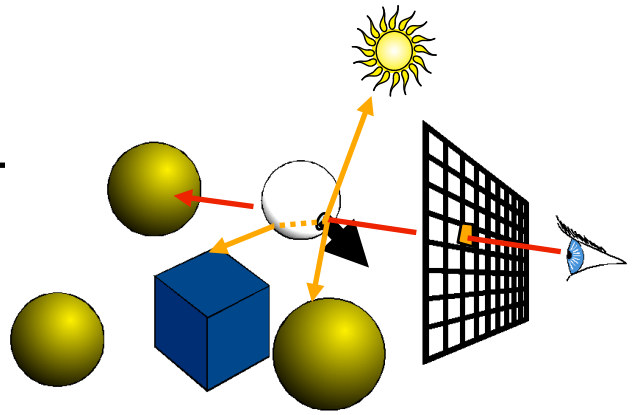
Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
```



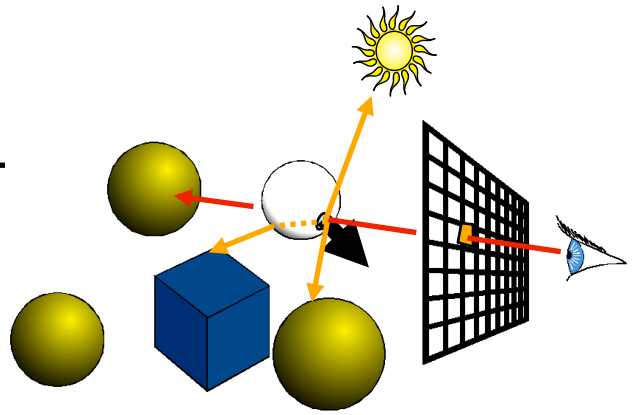
Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
```

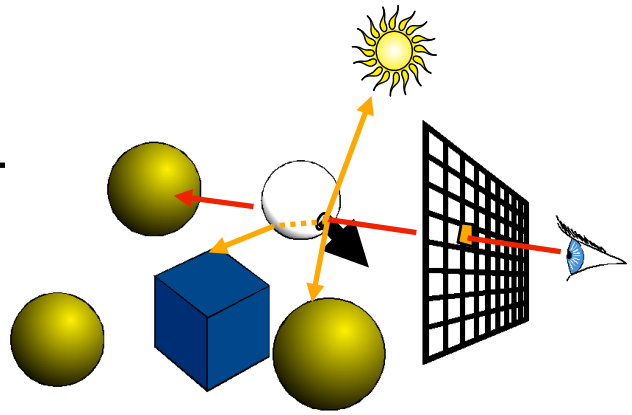


Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
```

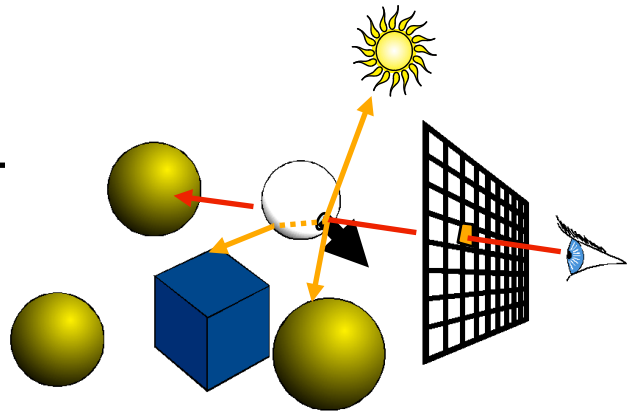


Recap: Ray Tracing



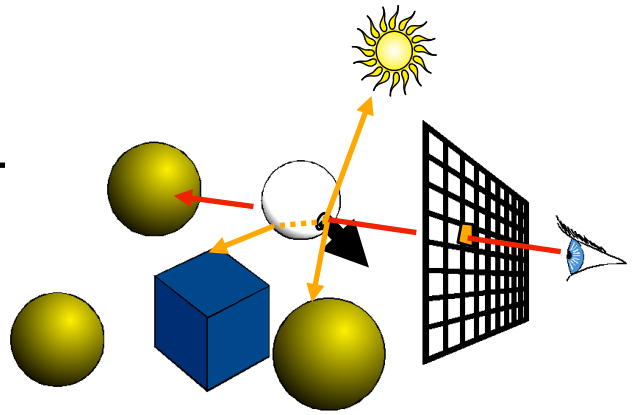
```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
```

Recap: Ray Tracing



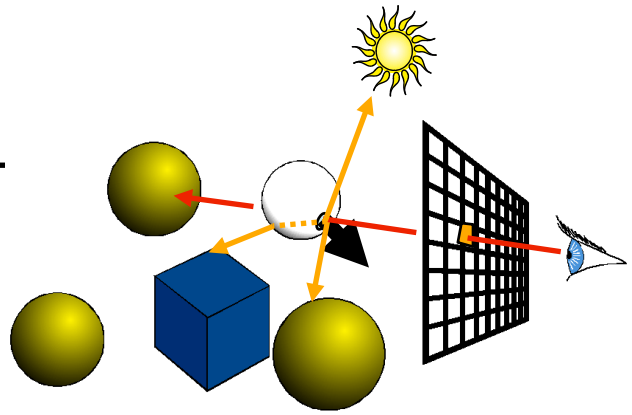
```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
```

Recap: Ray Tracing



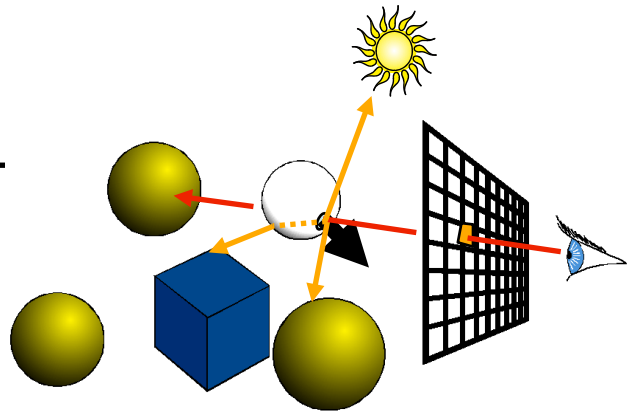
```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
```

Recap: Ray Tracing



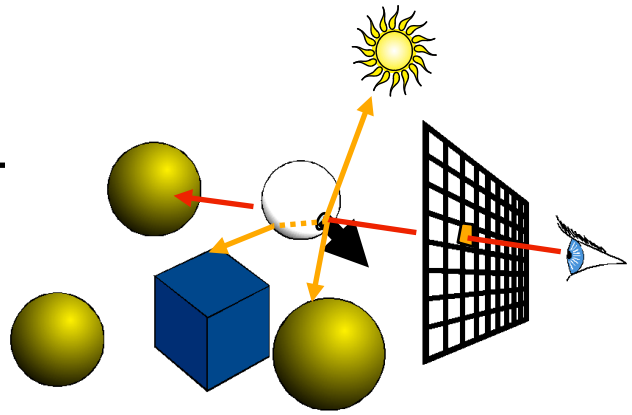
```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
```

Recap: Ray Tracing



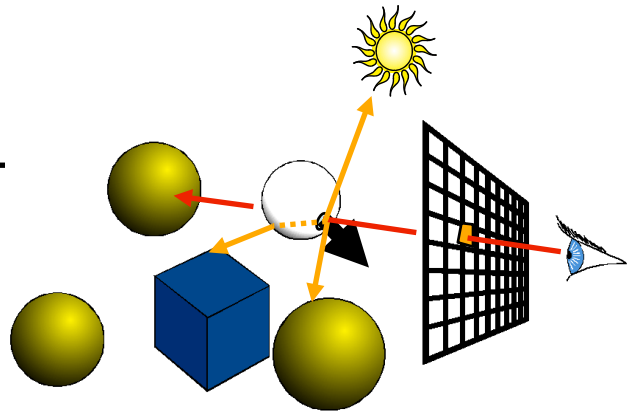
```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```

Recap: Ray Tracing



```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```

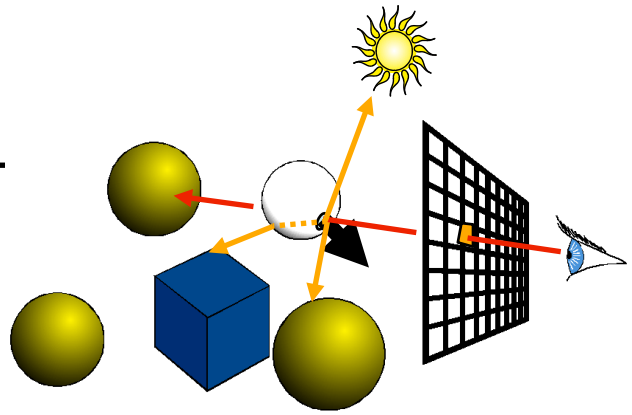
Recap: Ray Tracing



```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```

- ***Does it ever end?***

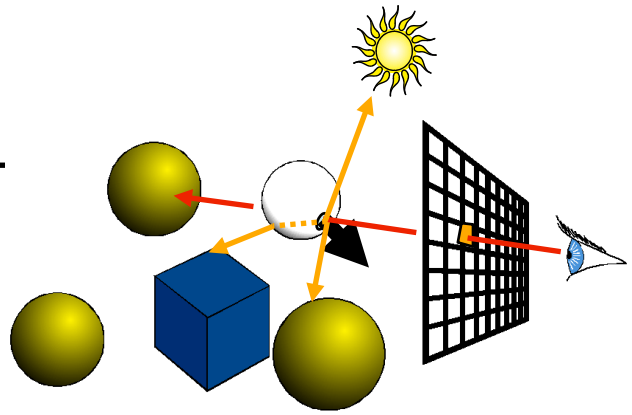
Recap: Ray Tracing



```
trace ray
   Intersect all objects
   color = ambient term
   For every light
      cast shadow ray
      color += local shading term
   If mirror
      color += color_refl *
               trace reflected ray
   If transparent
      color += color_trans *
               trace transmitted ray
```

- ***Does it ever end?***

Recap: Ray Tracing

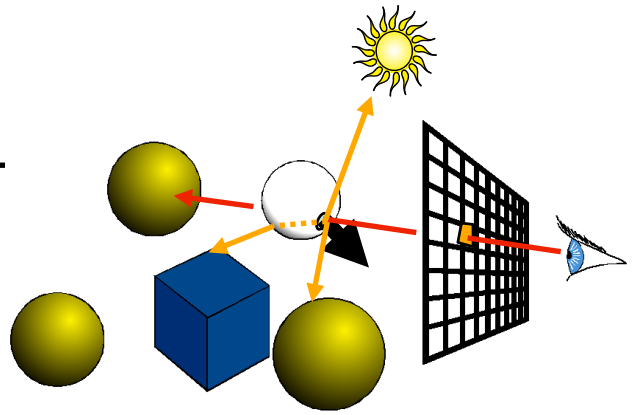


```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```

- ***Does it ever end?***

Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```



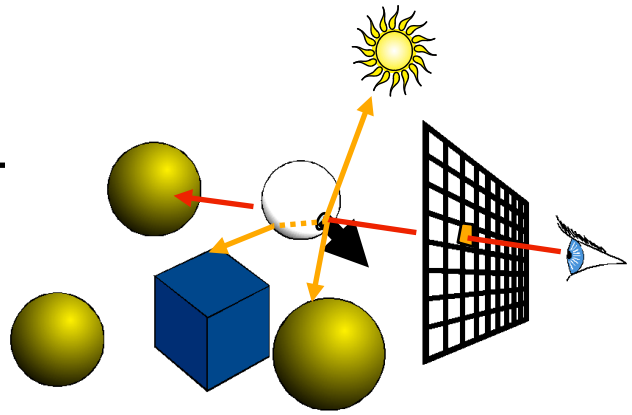
Stopping criteria:

- ***Does it ever end?***

Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```

- ***Does it ever end?***



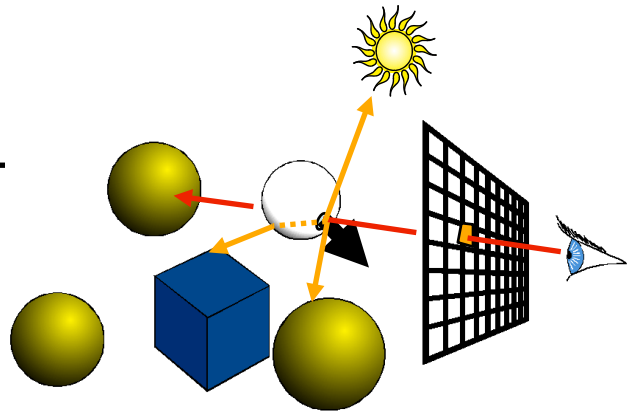
Stopping criteria:

- Recursion depth
  - Stop after a number of bounces

Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```

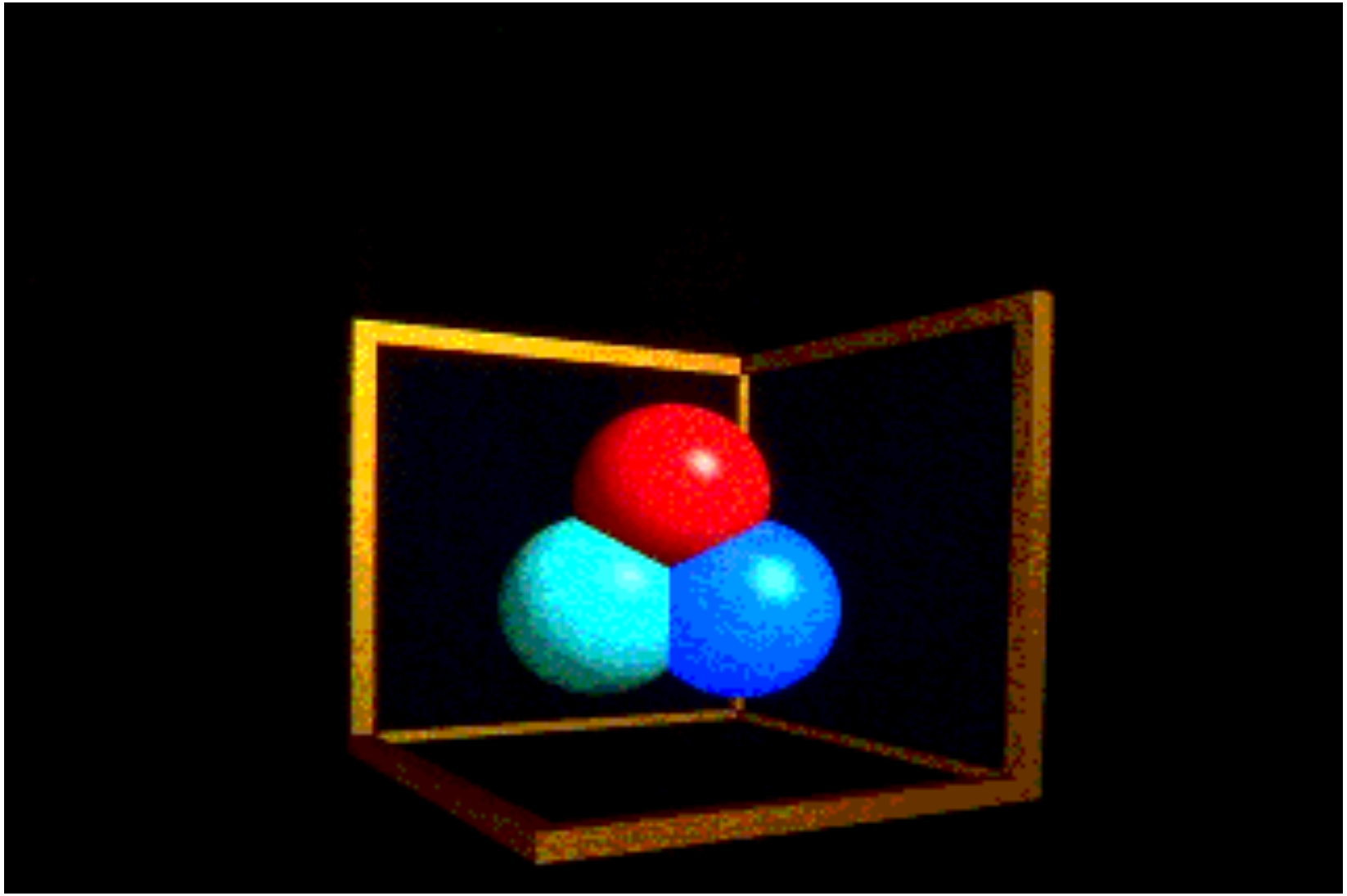
- ***Does it ever end?***



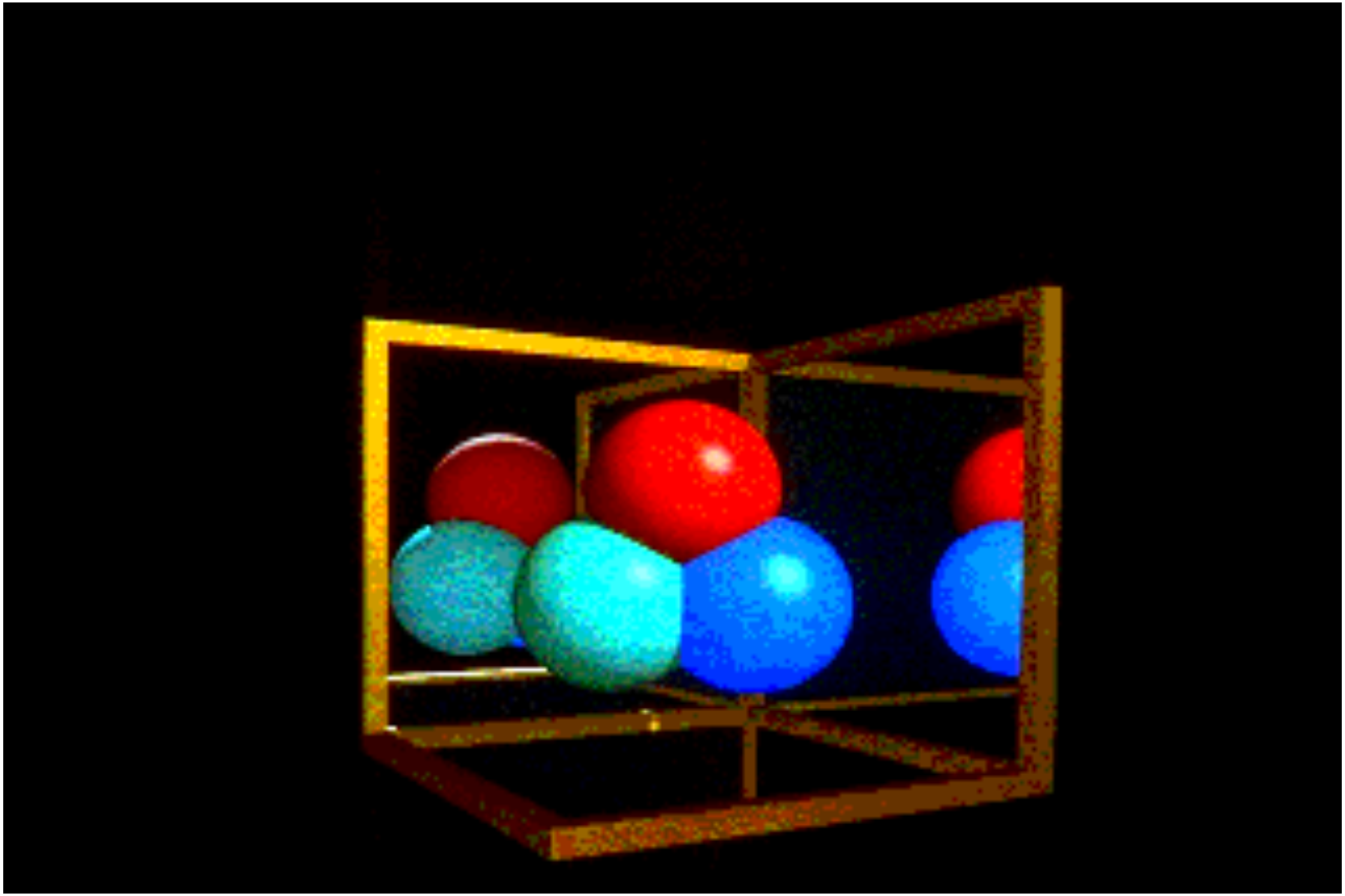
Stopping criteria:

- Recursion depth
  - Stop after a number of bounces
- Ray contribution
  - Stop if reflected / transmitted contribution becomes too small

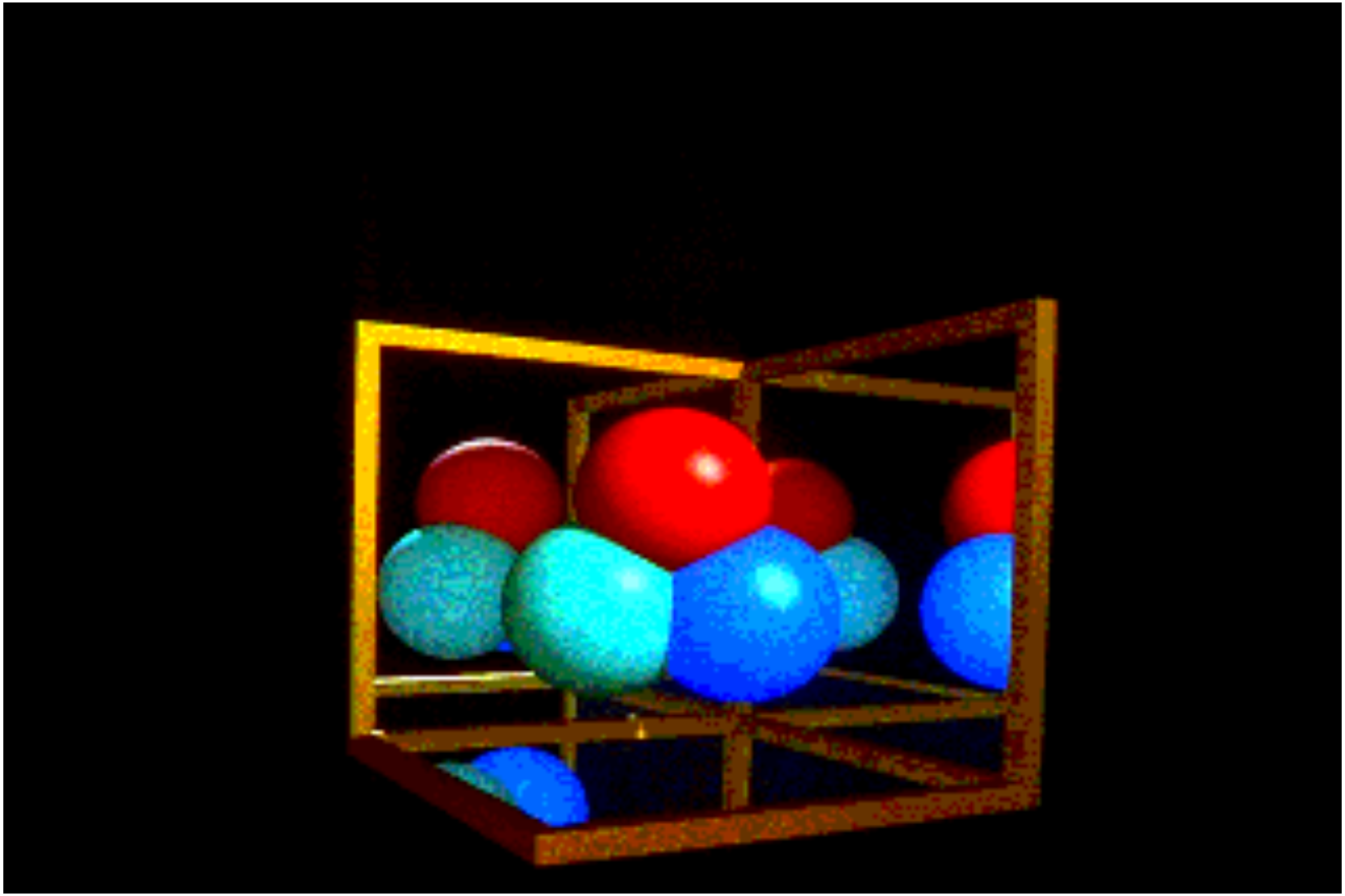
Recursion For Reflection: None



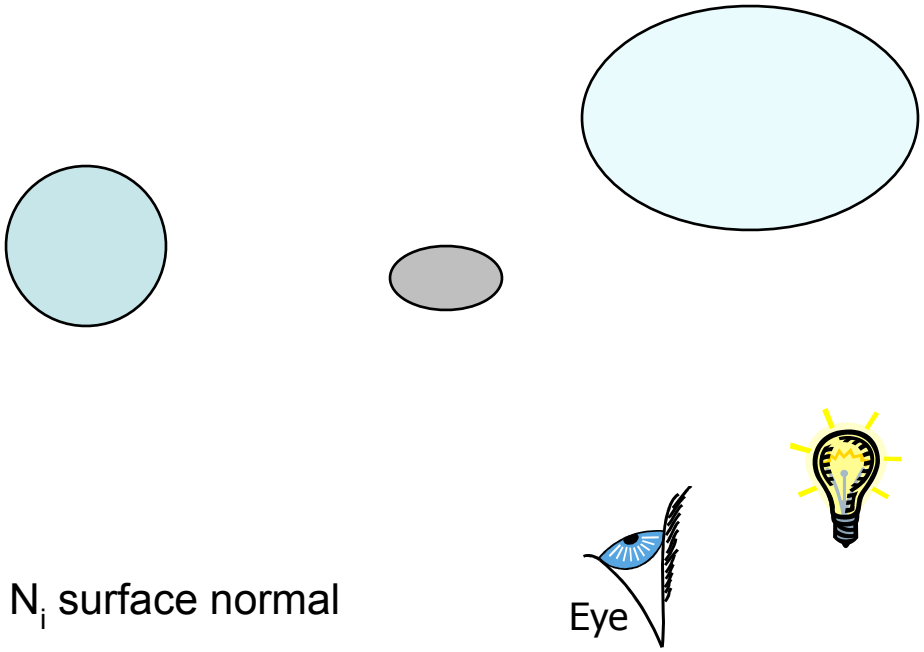
Recursion For Reflection: 1



Recursion For Reflection: 2



The Ray Tree



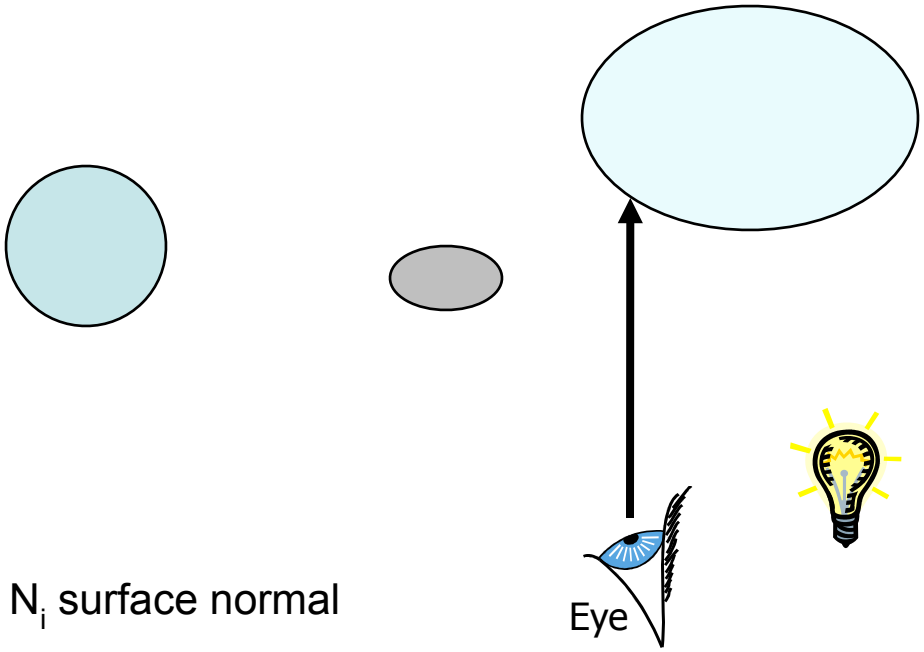
N_i surface normal

R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree



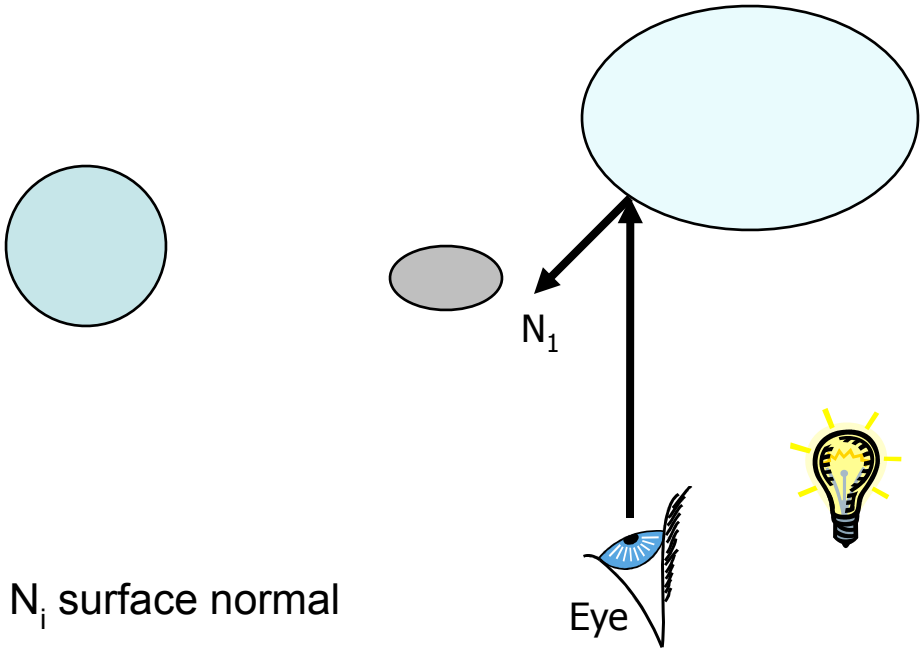
N_i surface normal

R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree



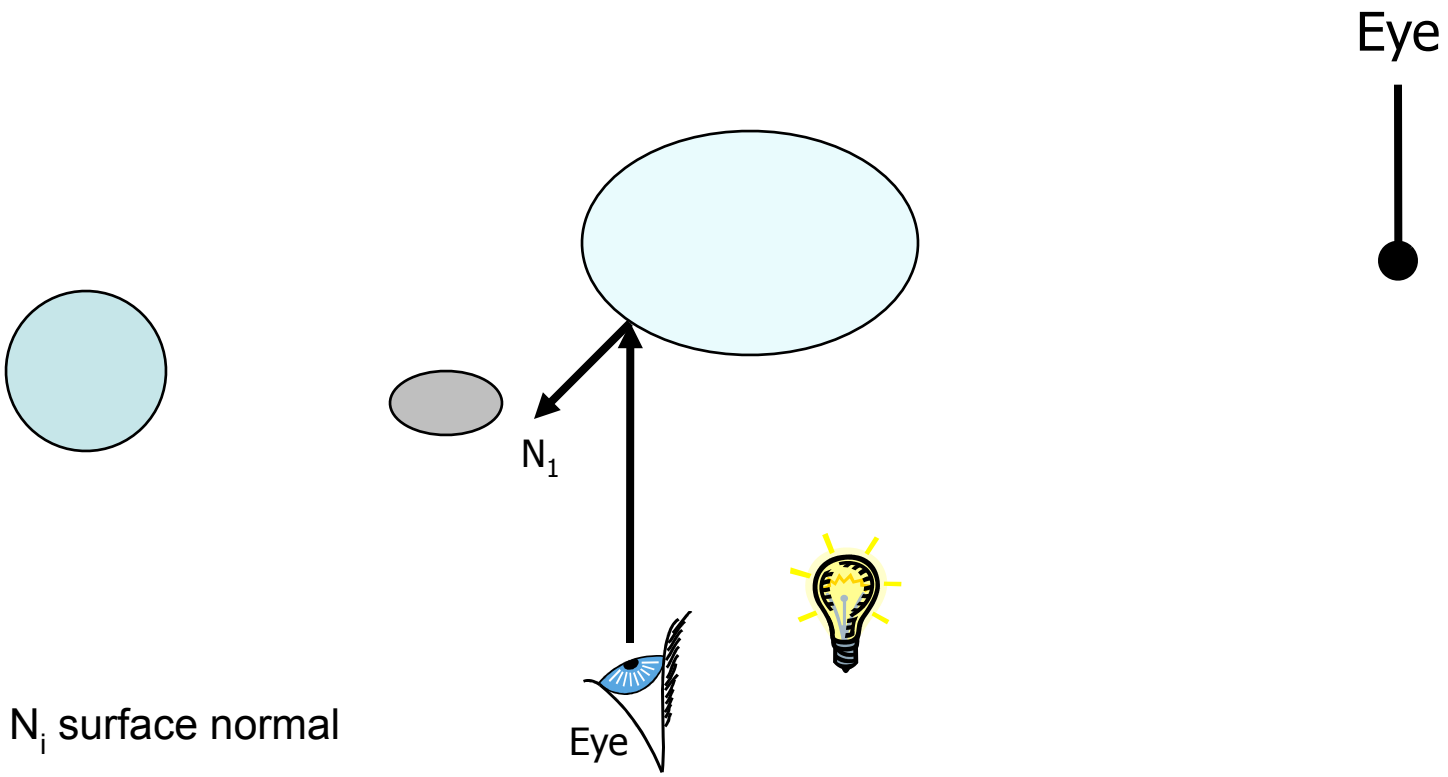
N_i surface normal

R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree



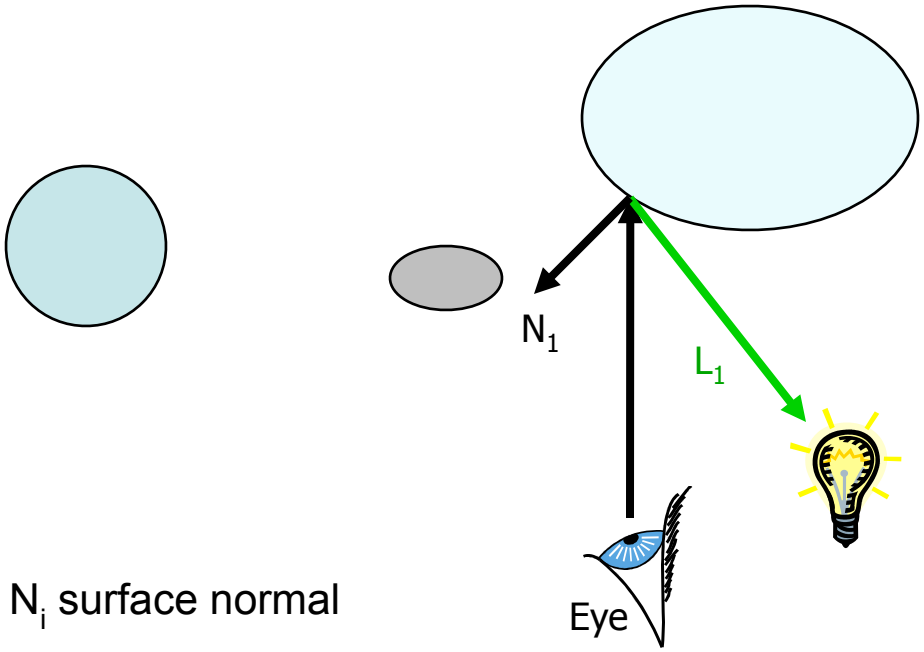
N_i surface normal

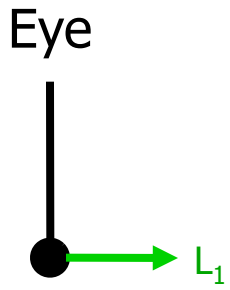
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





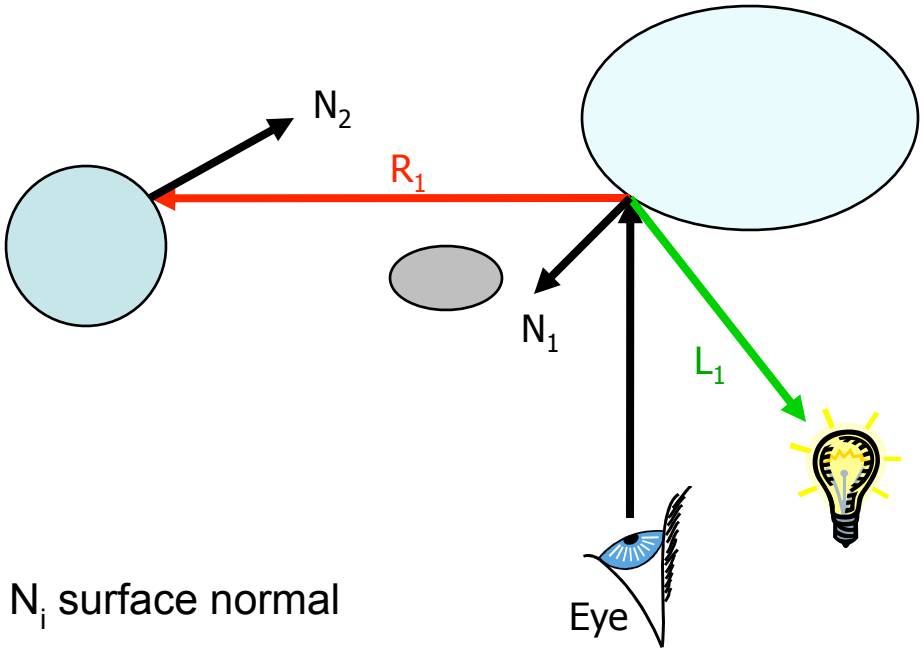
N_i surface normal

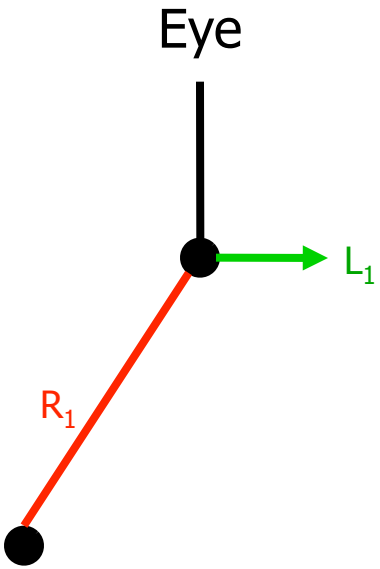
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





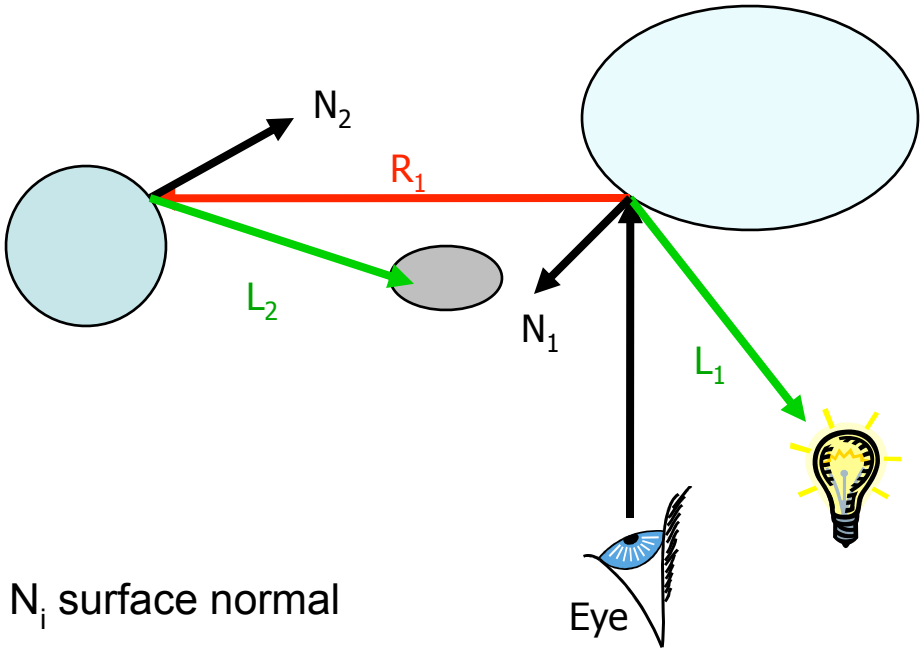
N_i surface normal

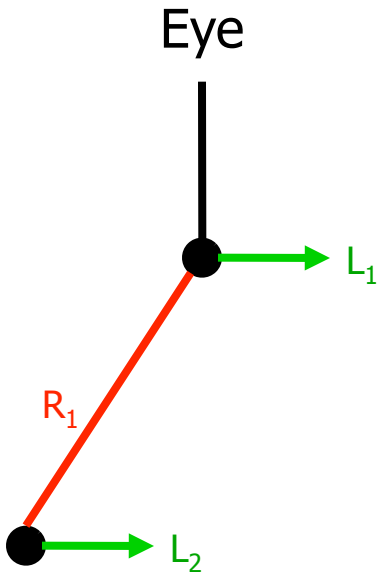
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





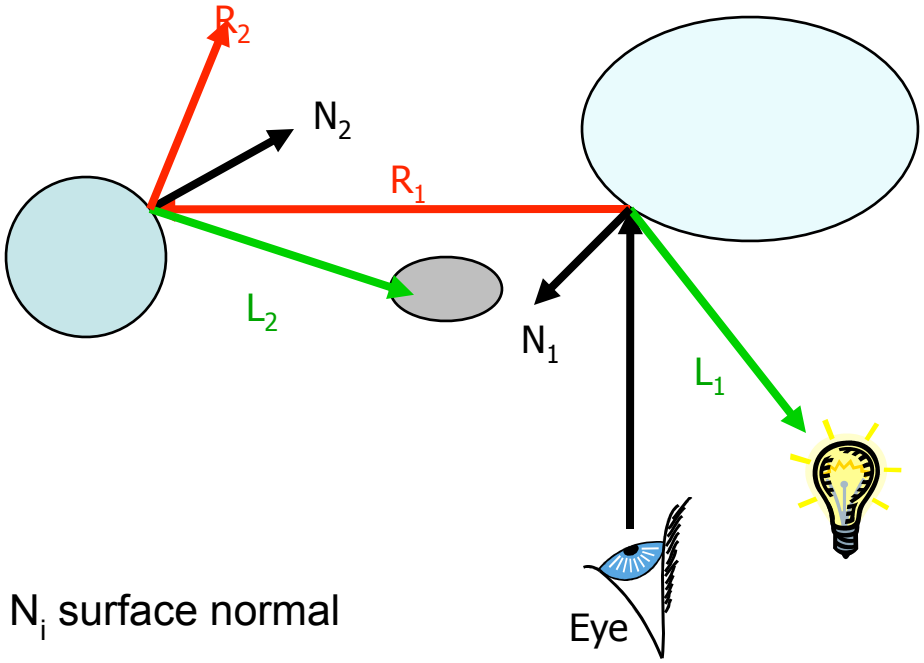
N_i surface normal

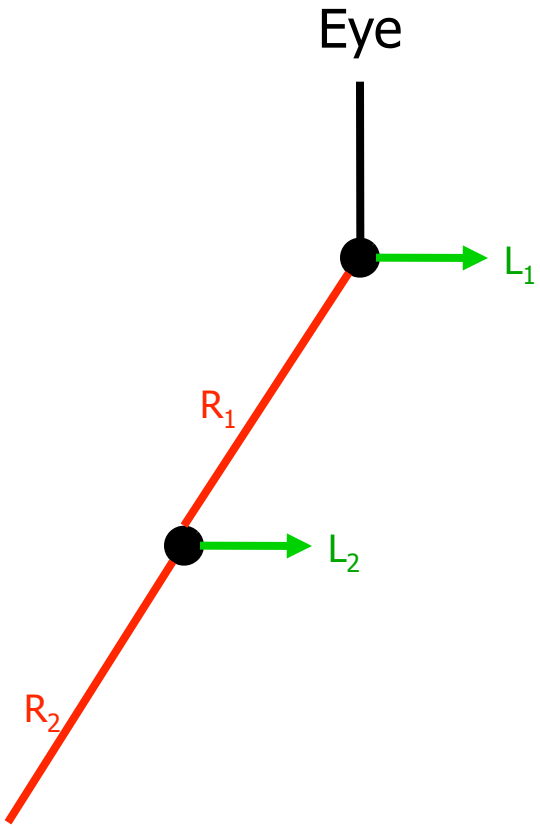
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





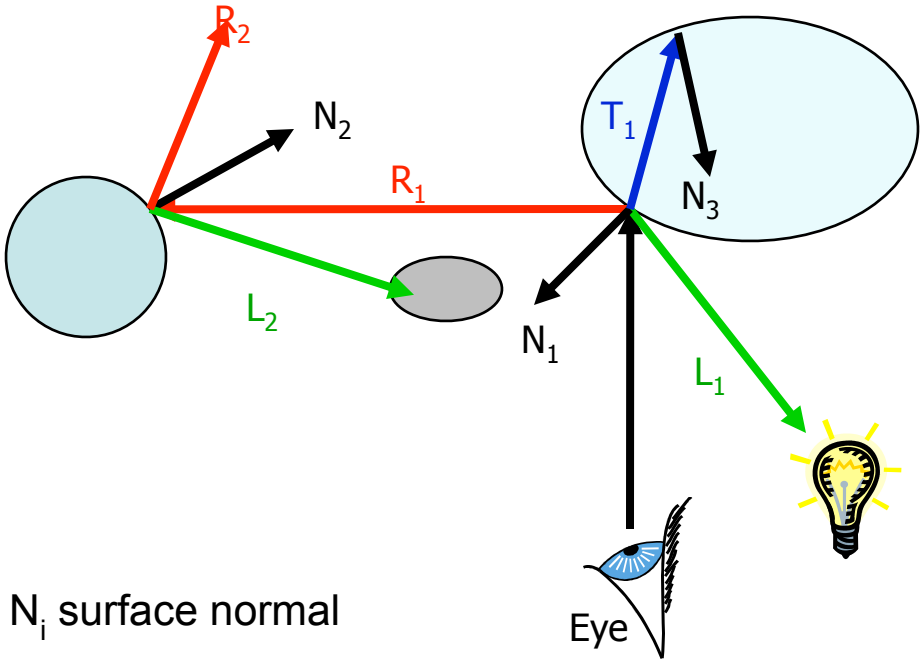
N_i surface normal

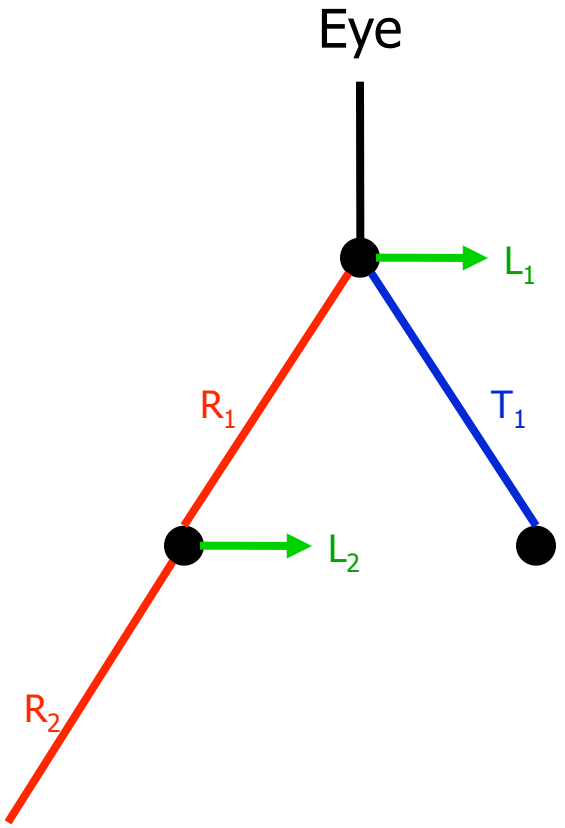
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





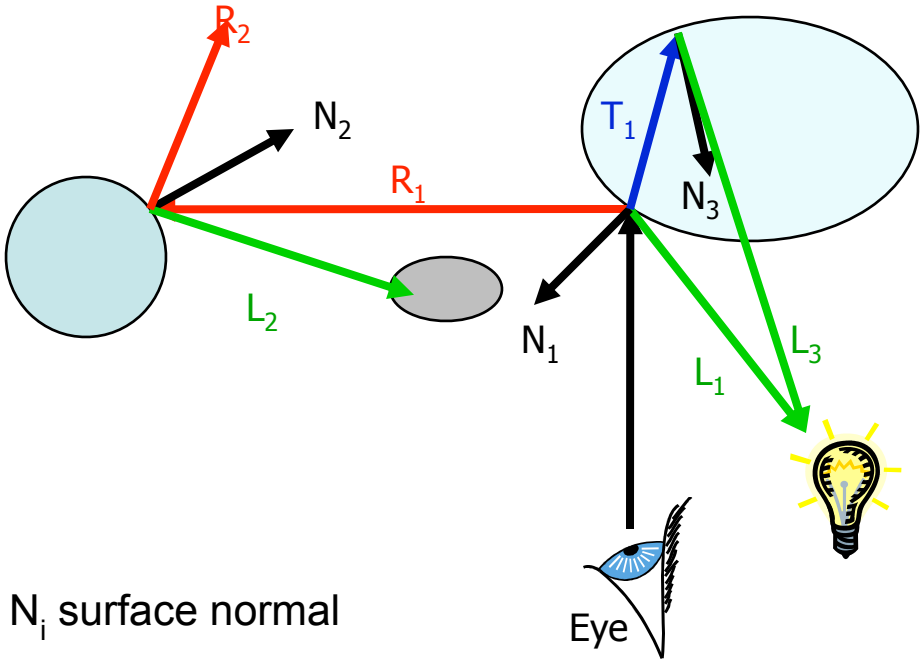
N_i surface normal

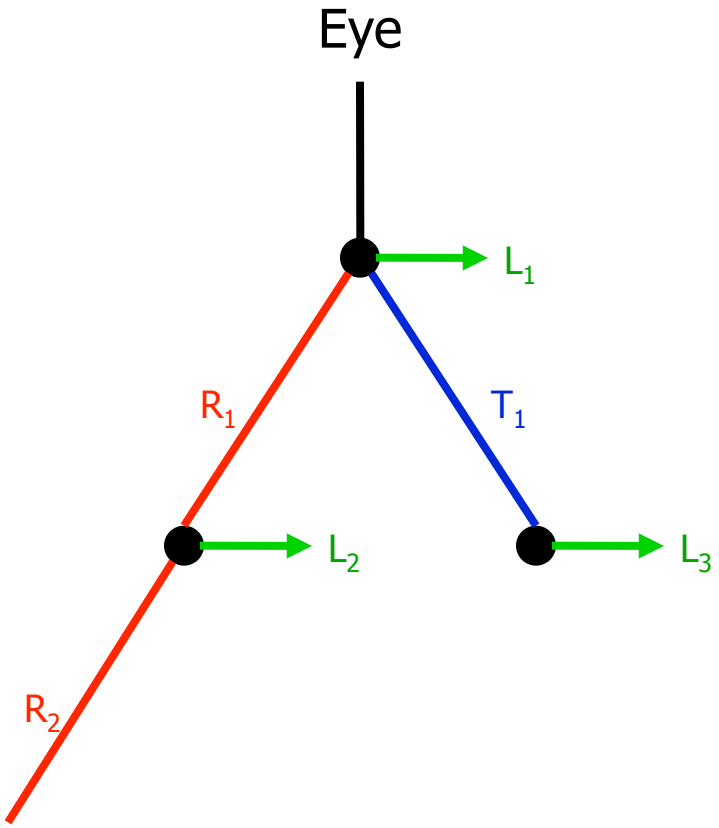
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





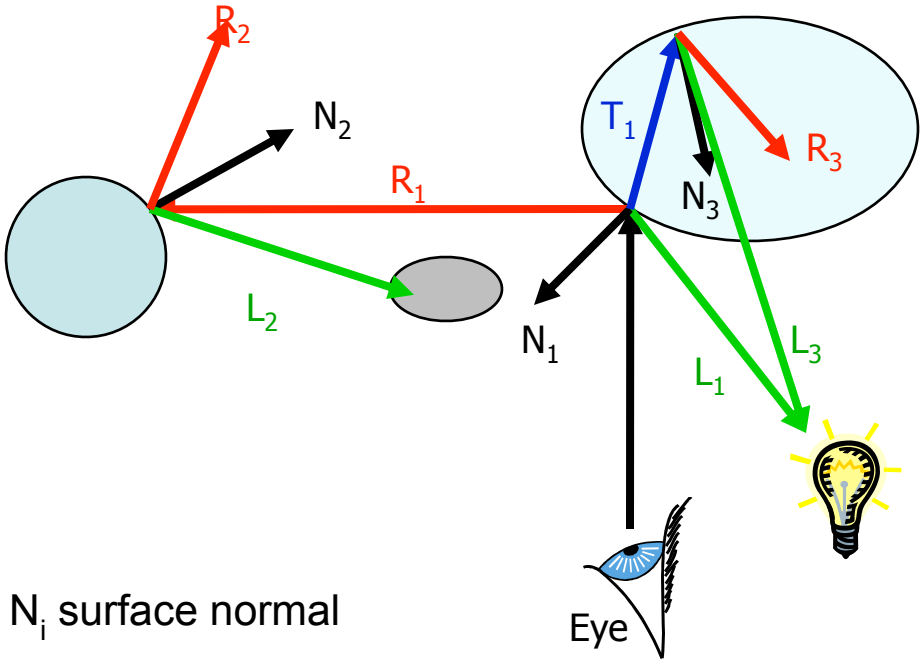
N_i surface normal

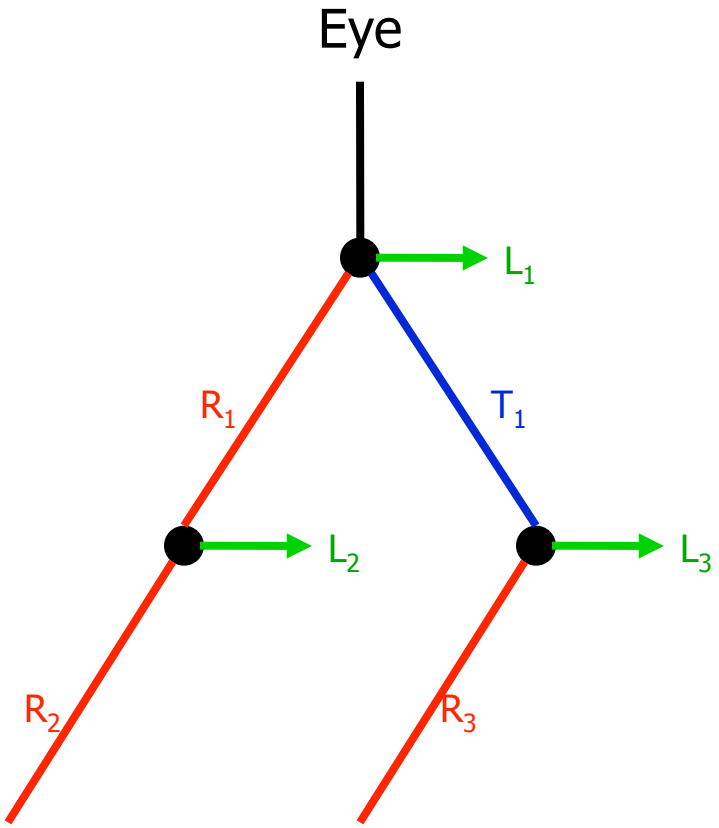
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





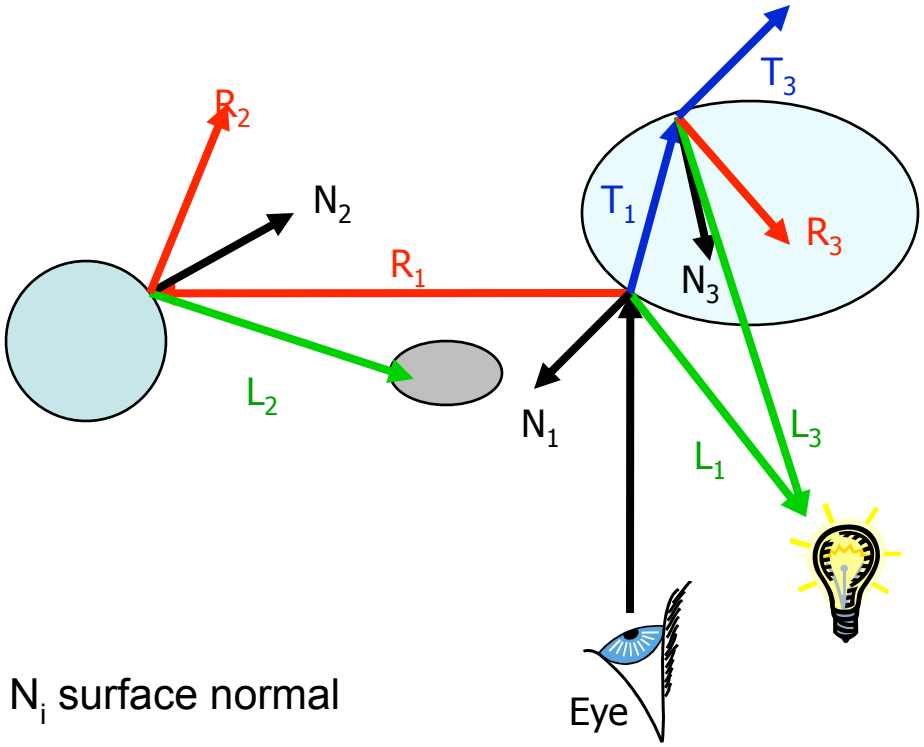
N_i surface normal

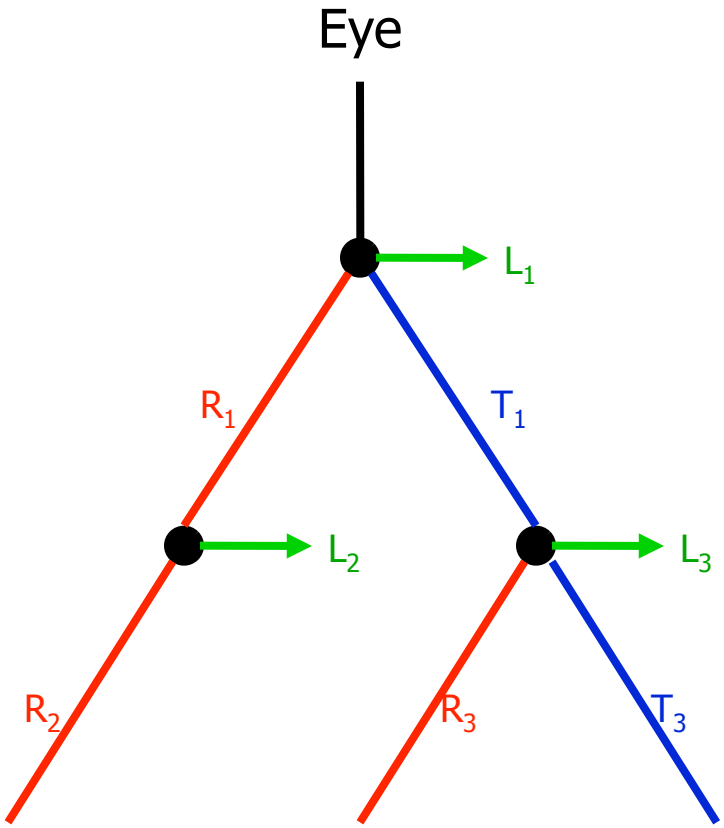
R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree





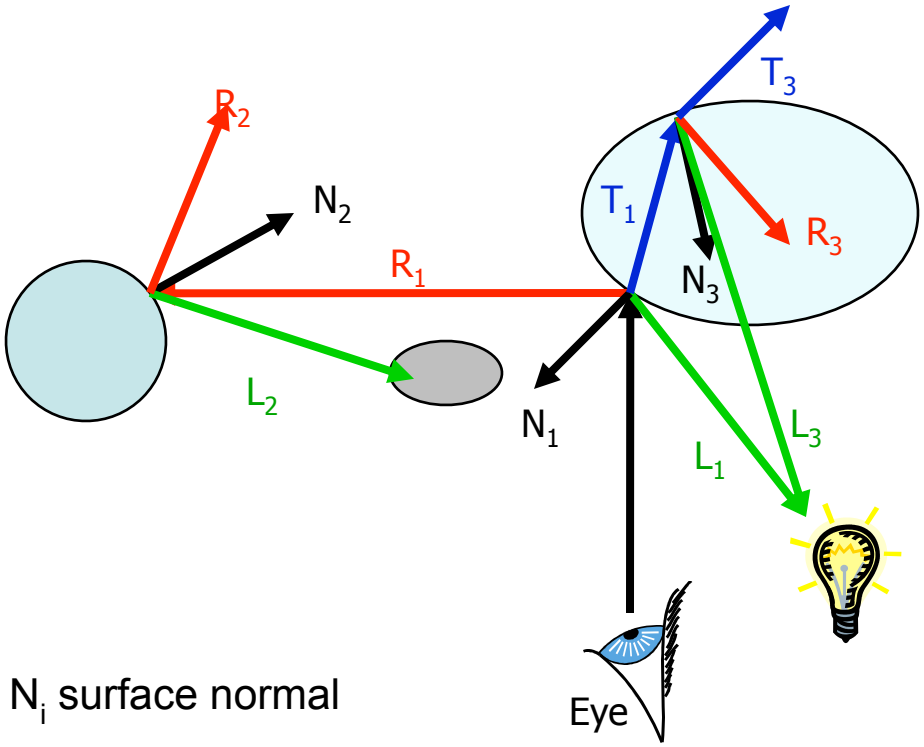
N_i surface normal

R_i reflected ray

L_i shadow ray

T_i transmitted (refracted) ray

The Ray Tree

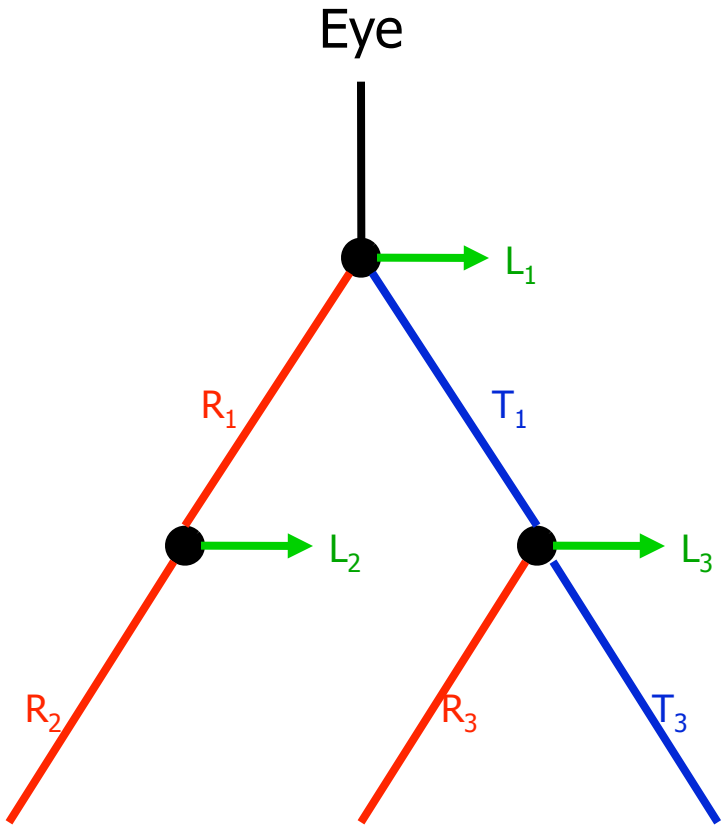


N_i surface normal

R_i reflected ray

L_i shadow ray

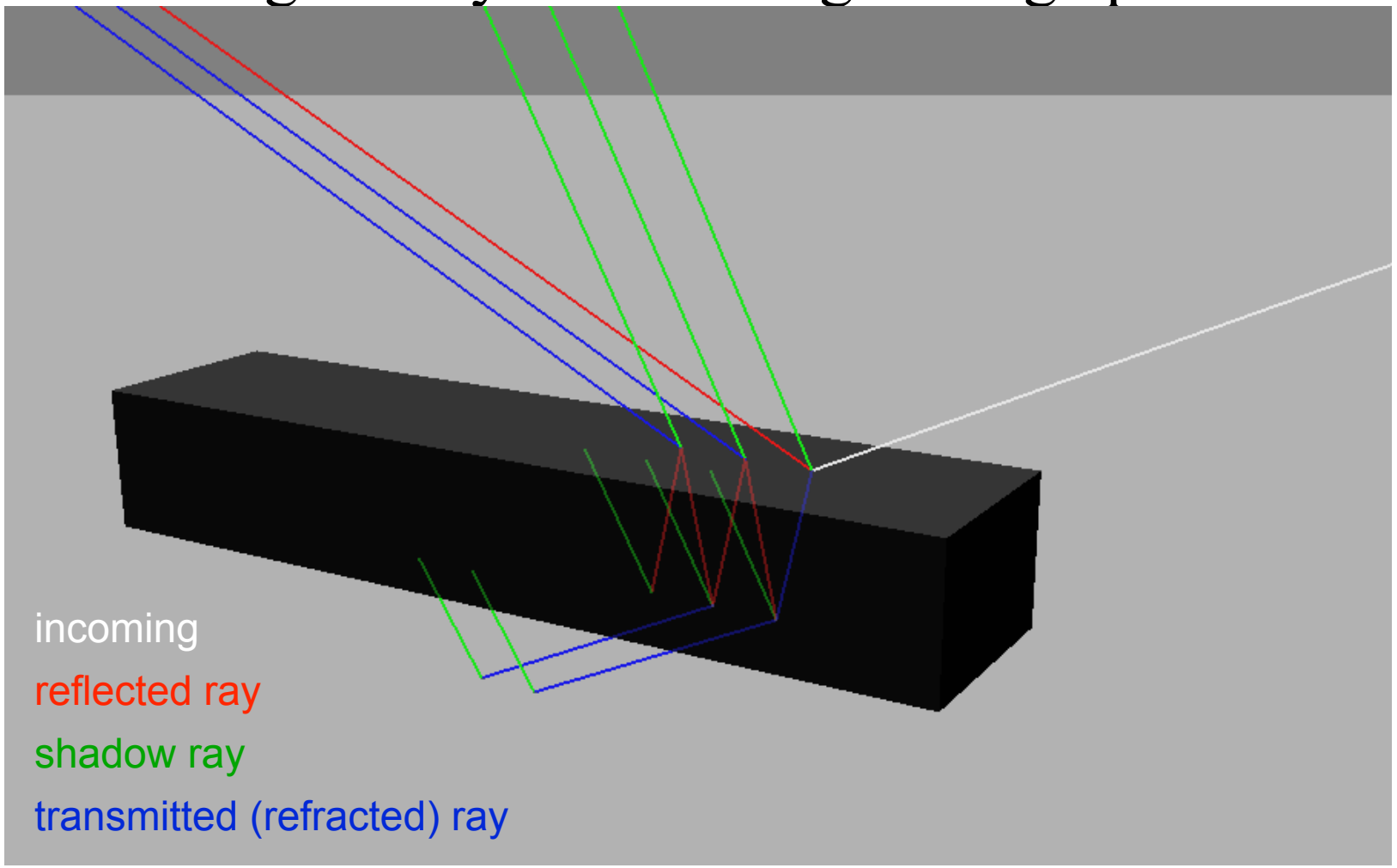
T_i transmitted (refracted) ray



Complexity?

Ray tree

- Visualizing the ray tree for single image pixel



Ray tree

This gets pretty complicated pretty fast!

- Visualizing the ray tree for single image pixel