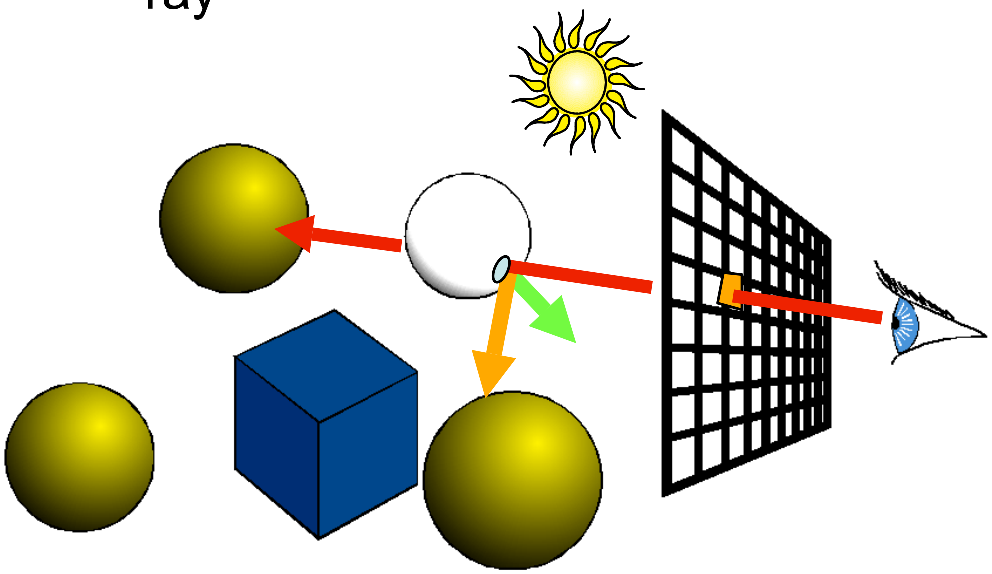
CSE 681

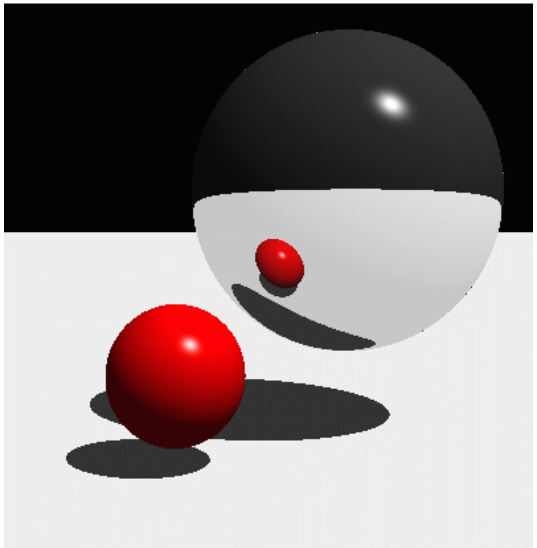
Reflection and Refraction

Mirror Reflection

- Shiny Objects
  - Perfect mirror in the extreme case
  - Assume this for now – mathematics easy for this nice case

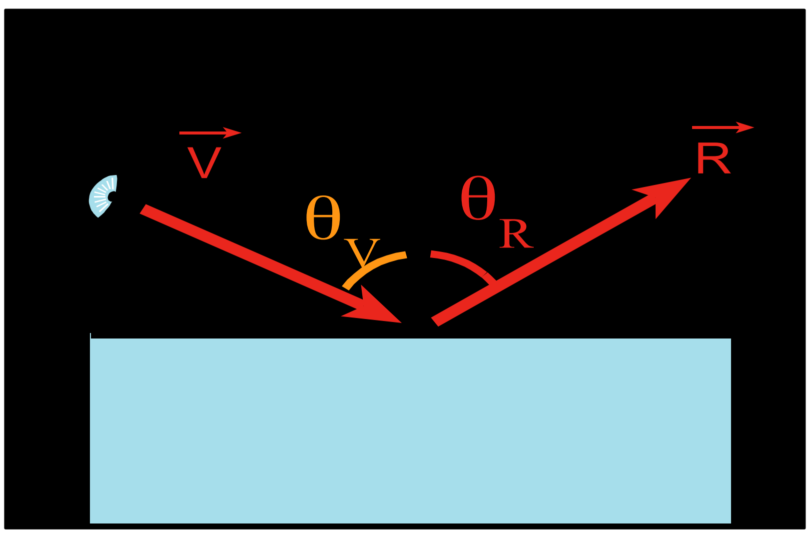
- Reflect the view vector about the normal and cast a reflection ray





Reflection Ray

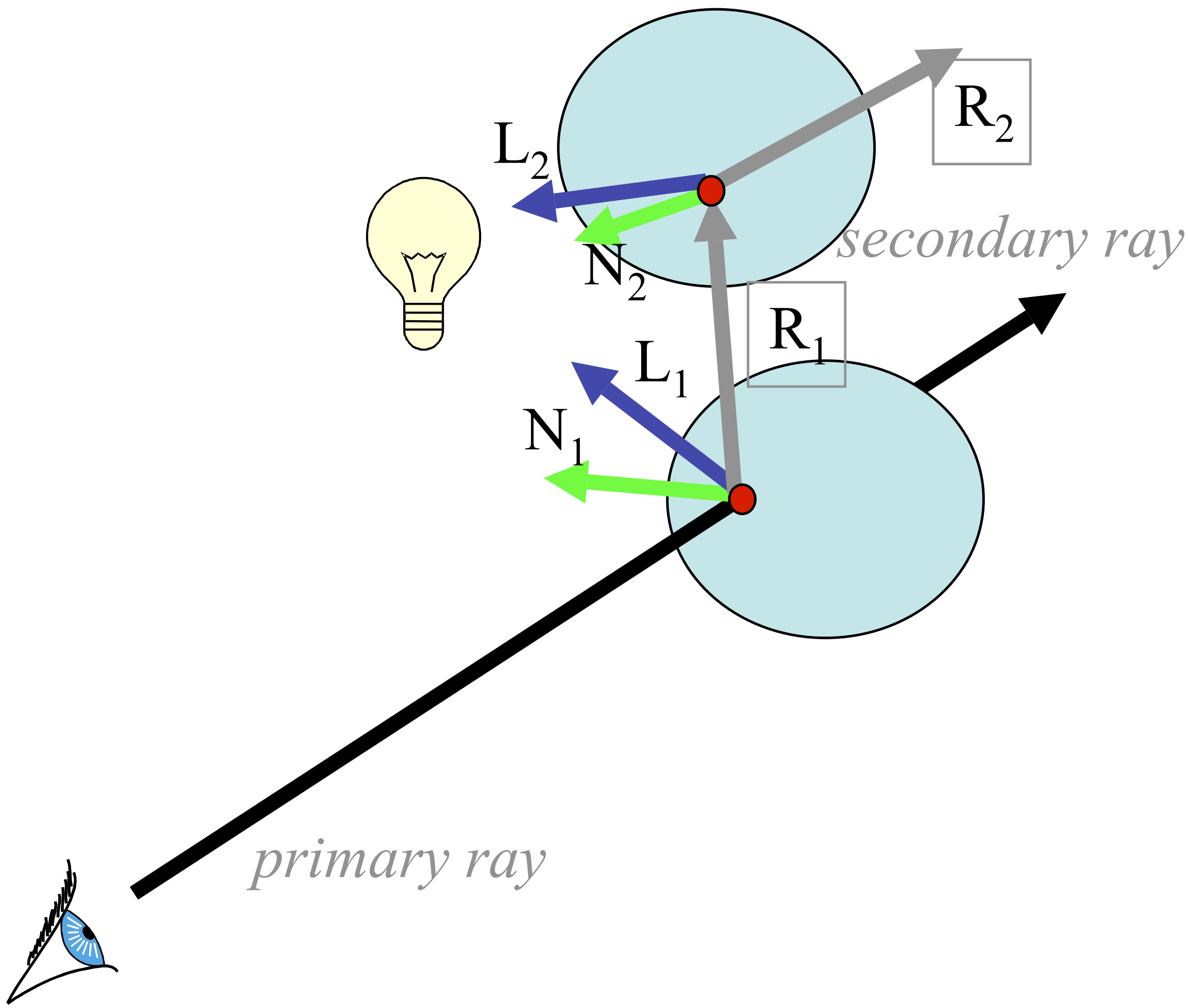
- Reflect the view vector about the normal, call it R
  - We already know how to solve this



$$\vec{R} = -2\,(\vec{V}.\vec{N})\,\vec{N} + \vec{V}$$

(note that $\vec{V}$ is pointing to the object)

Ray tracing With Reflection



Raytracing With Reflection

```
Color shade( ray, recursionDepth )
{
    c = background  color;
    intersectFlag = FALSE;
    for each object
        intersectFlag = intersect ( ray, p );

    if intersectFlag is TRUE
        c = ambient;
        for each light source
            compute reflective ray R (or H);
            c += diffuse;
            c += specular components;
        if ( recursionDepth < MAXRECURSION)
            if (object is shiny)
                compute reflection of the ray, R1;
                c += Ks * shade( R1, recursionDepth + 1 );
    return c;
}
```

Keep track of the depth of the ray tree

First illuminate the point

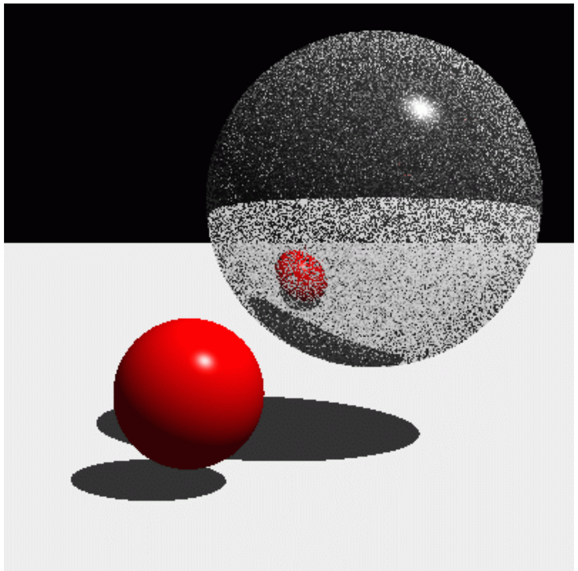
Don't spawn reflection ray past the pre-chosen tree depth

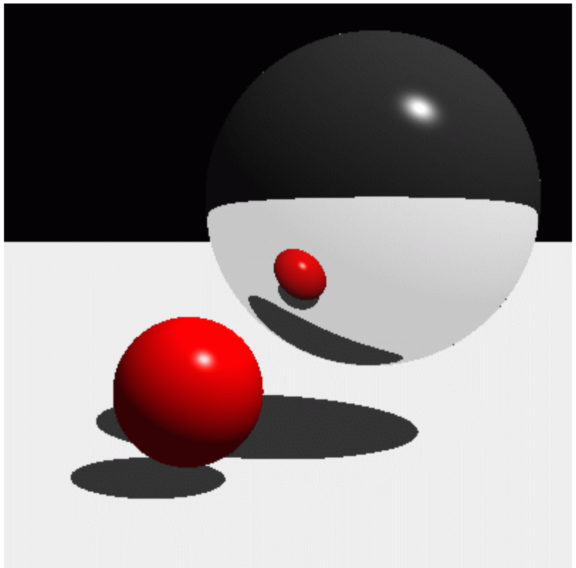
R1 reflects the view vector
Add the returned color with a shininess factor

Reflection Problem

- Same issue as the shadow ray - self intersection
- How shall we perturb the intersection point?
  – In the direction of the reflection vector

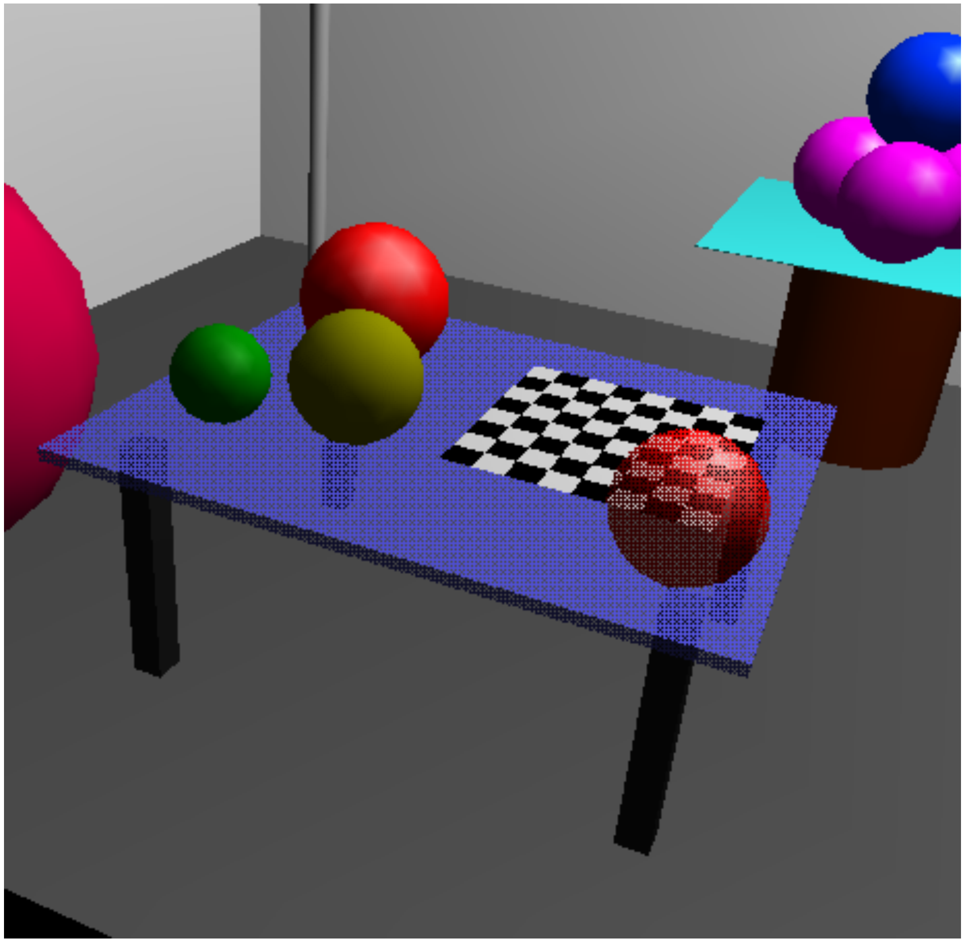
No epsilon





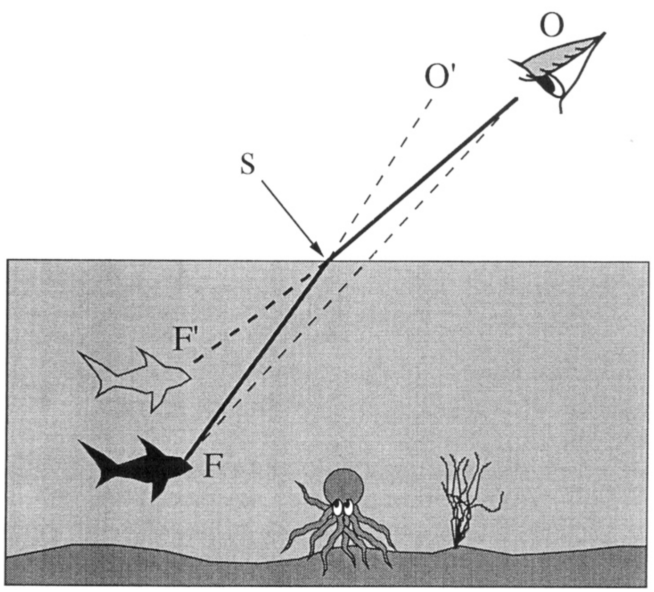
With epsilon

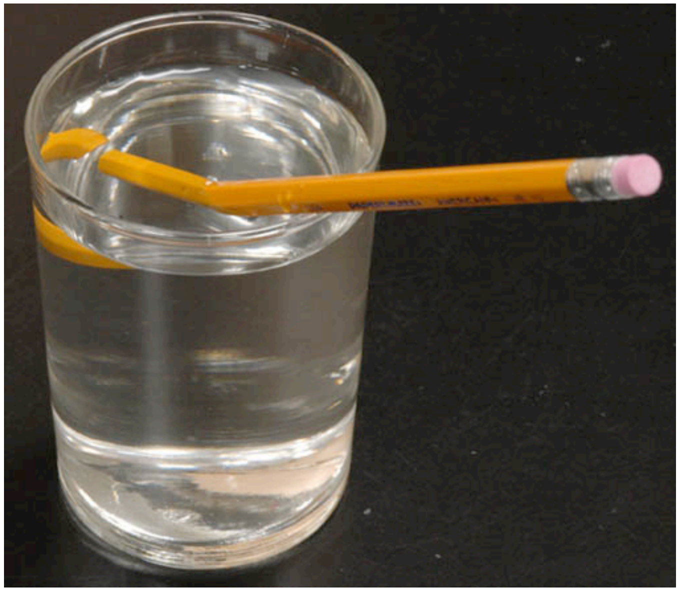
Transparency



Refraction

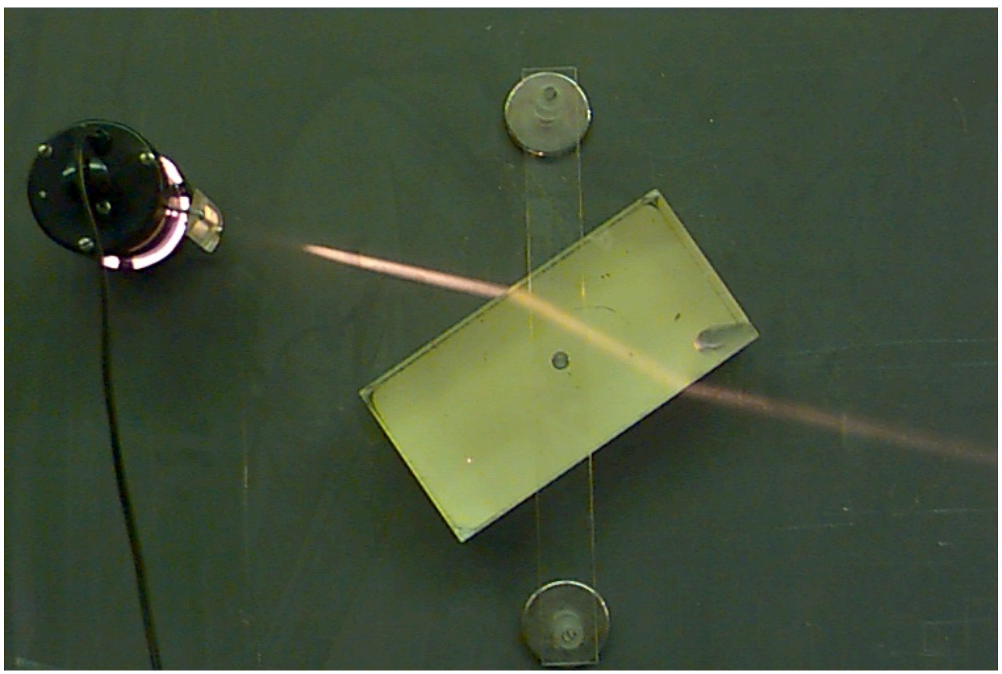
- Transparency depends upon the **refractive** properties of the material
  - Light bends through some materials

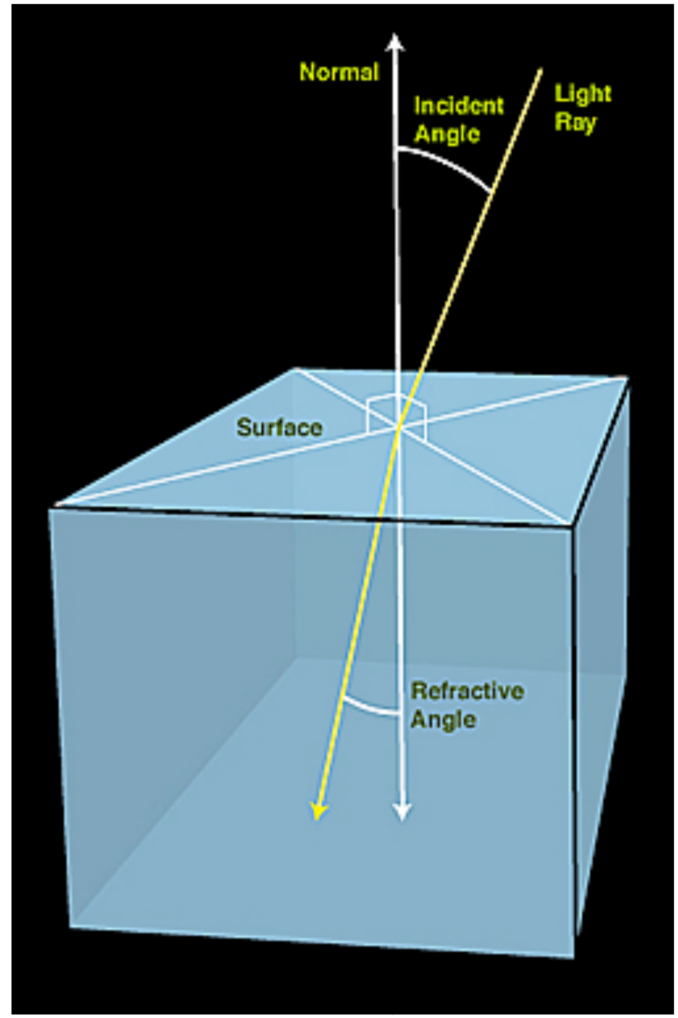




Snell's Law

Willebrord Snell (Dutch Physicist) determined how light refracts through a medium in 1621





Snell's Law

The angle at which the light bends is described by the following relationship:

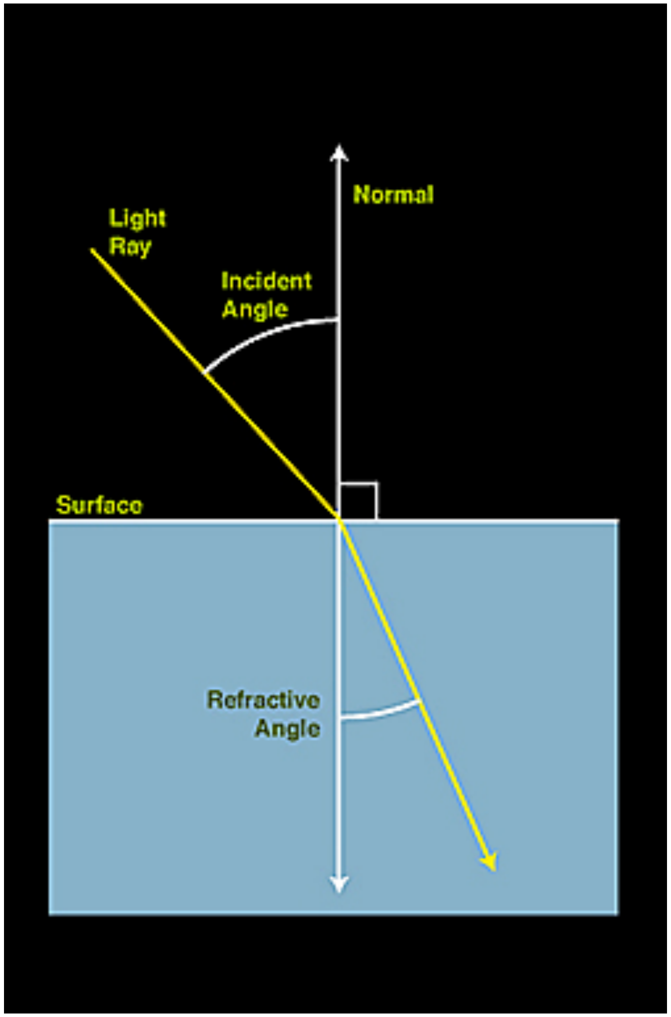
$$\frac{\sin\theta_t}{\sin\theta_i} = \frac{\eta_i}{\eta_t} = \eta_r$$

θ_i **: Incident angle**

θ_t **: Refractive angle**

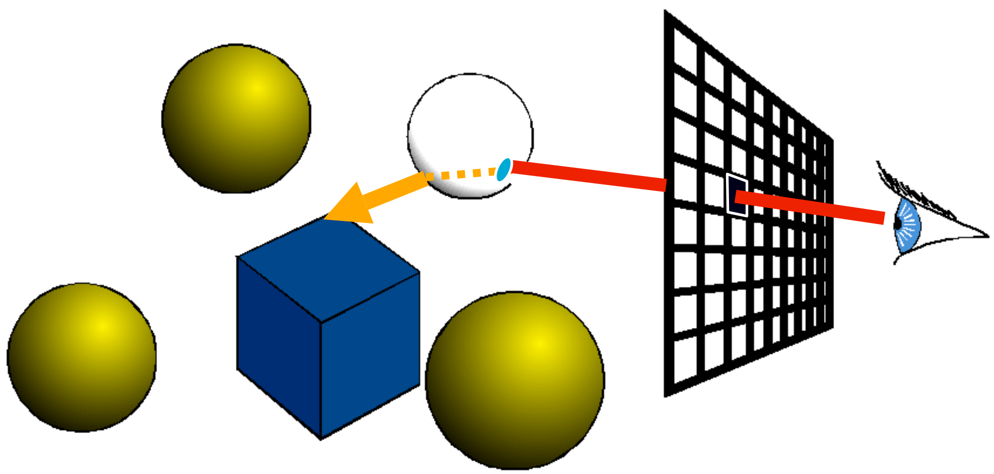
η_r **: Refractive index of the medium light is entering**

η_i **: Refractive index of the medium light is leaving**



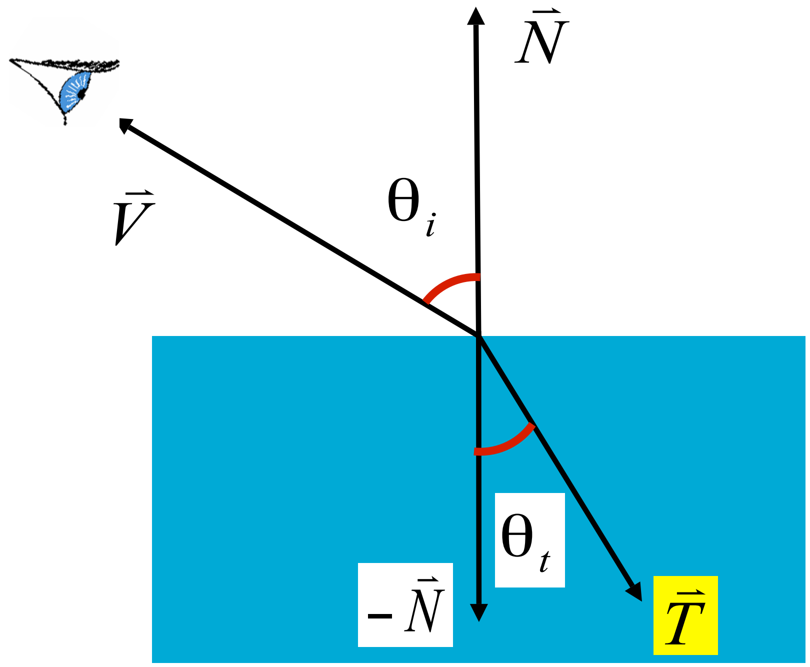
Refraction (Ray Tracing)

- Since we are following rays from the eye to the light, we "refract" using the view vector
  - Cast a secondary ray (refractive ray)
  - Incorporate the color from this ray into our shading



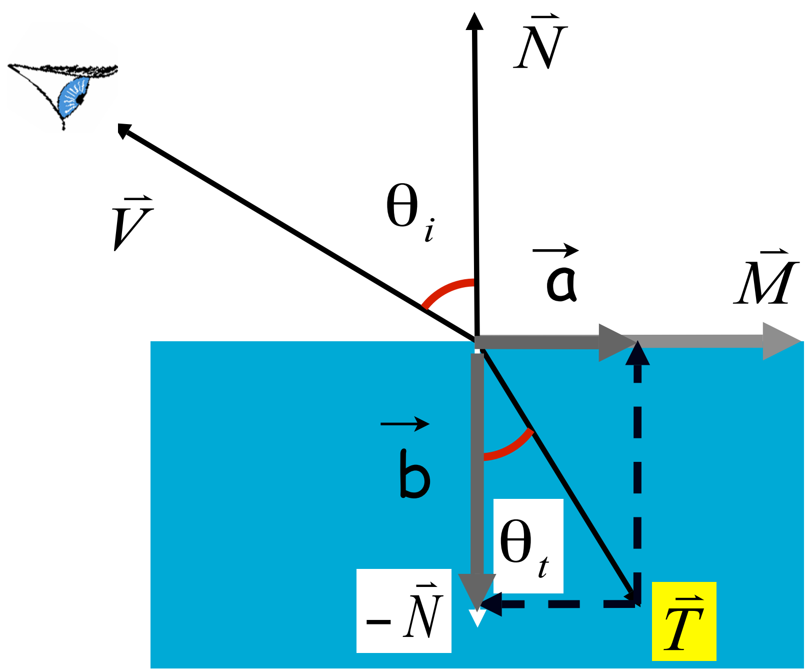
Refraction

- How do we compute the refraction vector T?



Geometric Construction

- Project T onto M to get $\vec{a}$
- Project T onto -N to get $\vec{b}$

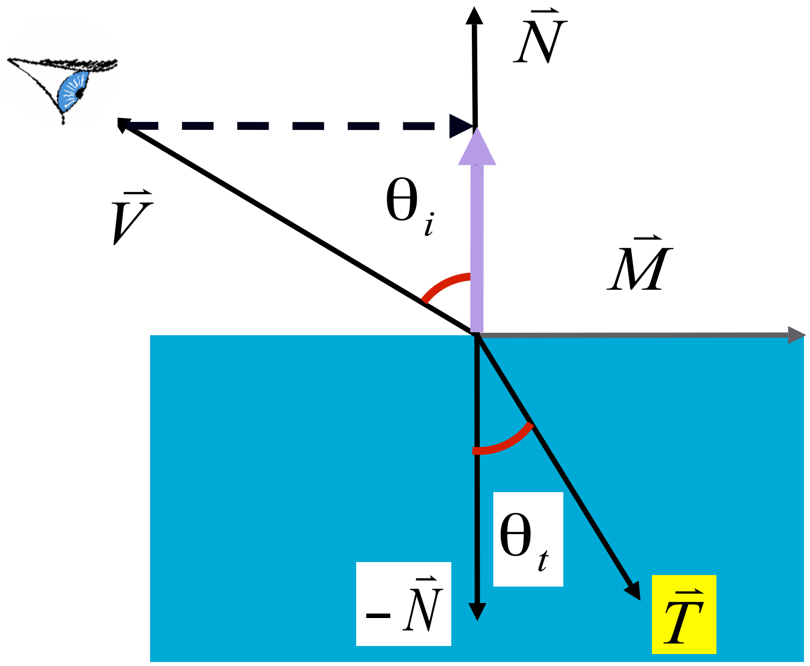


$$\vec{T} = \vec{a} + \vec{b}$$

$$\vec{T} = \sin\theta_t \vec{M} - \cos\theta_t \vec{N}$$

Geometric Construction

- Assume all given vectors are unit vectors
- Define vector M that is ⊥ to N



$$\vec{M} = \frac{(\vec{N}\cos\theta_i - \vec{V})}{\sin\theta_i}$$

Normalization

Geometric Construction

- Algebra …

$$\vec{T} = \sin\theta_t \vec{M} - \cos\theta_t \vec{N}$$

$$\vec{M} = \frac{(\vec{N}\cos\theta_i - \vec{V})}{\sin\theta_i}$$

$$\vec{T} = \frac{\sin\theta_t}{\sin\theta_i}(\vec{N}\cos\theta_i - \vec{V}) - \cos\theta_t \vec{N}$$

$$\boxed{\frac{\sin\theta_t}{\sin\theta_i} = \frac{\eta_i}{\eta_t} = \eta_r}$$

$$\vec{T} = (\eta_r \cos\theta_i - \cos\theta_t)\vec{N} - \eta_r \vec{V}$$

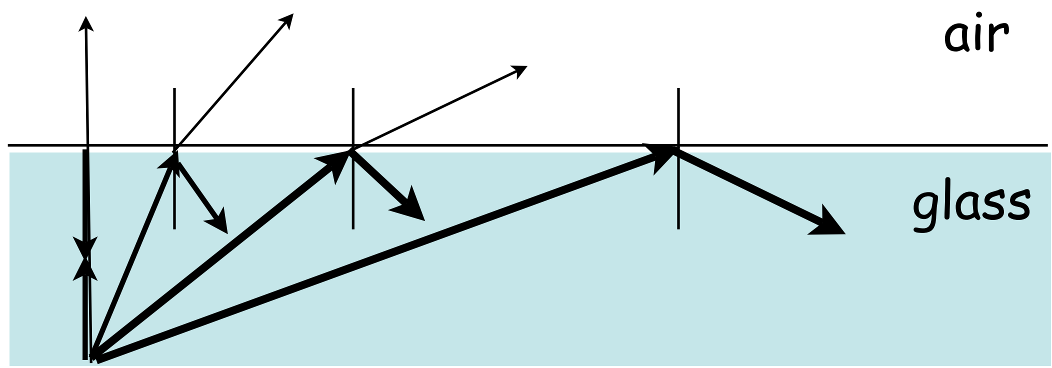
$$\cos\theta_i = \vec{N} \bullet \vec{V}$$

$$\cos\theta_t = \sqrt{1-\sin^2\theta_t} = \sqrt{1-\eta_r^2 \sin^2\theta_i} = \sqrt{1-\eta_r^2(1-(\vec{N}\bullet\vec{V})^2)}$$

$$\boxed{\vec{T} = \left(\eta_r(\vec{N}\bullet\vec{V}) - \sqrt{1-\eta_r^2(1-(\vec{N}\bullet\vec{V})^2)}\right)\vec{N} - \eta_r \vec{V}}$$

Total Internal Reflection

- In the equation above, if the term inside the sqrt is negative, then there is no refraction. This is called total internal refraction
- This happens when light tries to pass from a dense medium (glass) to a less-dense medium (air) at a shallow angle



Raytracing Algorithm
Shadows, Reflection, and Refraction

```
Color shade( ray, recursionDepth )
{
    Initialize pixel color to background
    Intersect all objects
    If an intersection is found
        Initialize to ambient shading
        For each light
            Shoot shadow ray
            If not in shadow add diffuse and specular
        If ( recursionDepth < MAXRECURSION)
            If (object is shiny)
                Trace reflection ray
            If (object is transparent)
                Trace transmittive (refractive) ray
    Return color
}
```

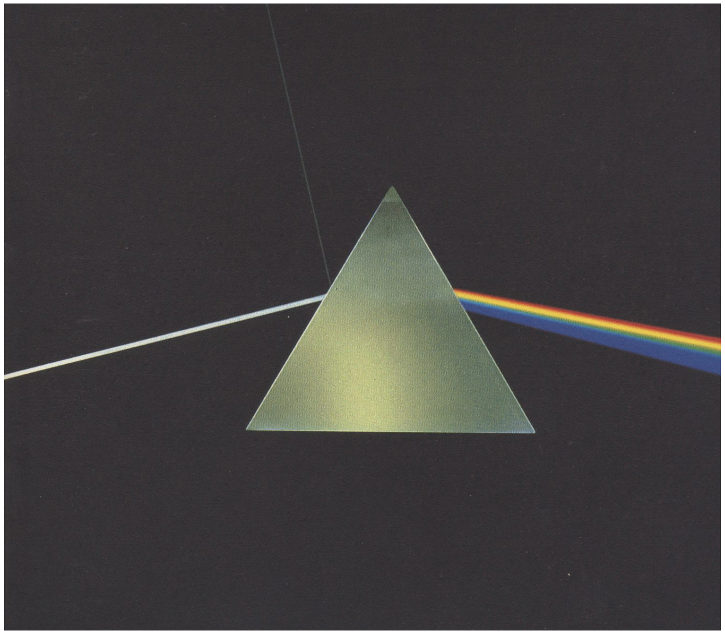
Raytracing Pseudocode
Shadows, Reflection, and Refraction

```
Color shade( ray, recursionDepth )
{
    c = background  color;
    intersectFlag = FALSE;
    for each object
            intersectFlag = intersect ( ray, p );

    if intersectFlag is TRUE
            c = ambient;
            for each light source
                shadowFlag = intersectShadowRay ( p );
                if  shadowFlag is FALSE
                    compute reflective ray R (or H);
                    c += diffuse;
                    c += specular components;
            if  ( recursionDepth < MAXRECURSION)
                if (object is shiny)
                    compute reflection of  the ray, R1;
                    c += Ks * shade( R1, recursionDepth + 1 );
                if (object is transparent)
                    compute transmittance ray T;
                    c +=  Kt * shade( R1, recursionDepth + 1 );  //Kt is the amount of  light allowed to go through
    return c;
}
```

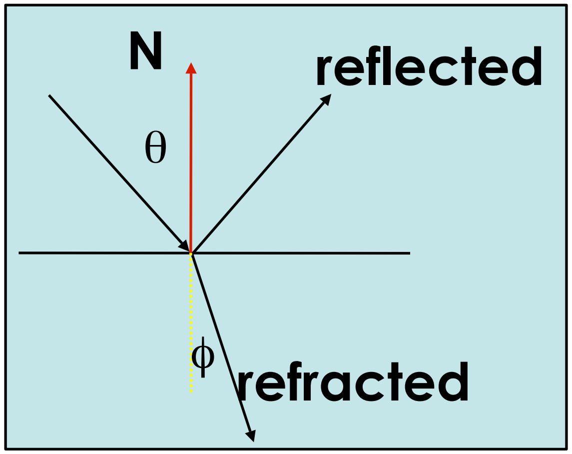
Wavelength

- Refraction (and reflection) is wavelength-dependent
  - Remember Newton's experiment
  - Where rainbow's come from
  - So, compute separately for each component (i.e., r, g, b)



Reflection and Refraction

- Light is partially reflected and refracted
- Refractive Index: $\eta = \sin\phi / \sin\theta$
- Depends on the object material



Fresnel Equation

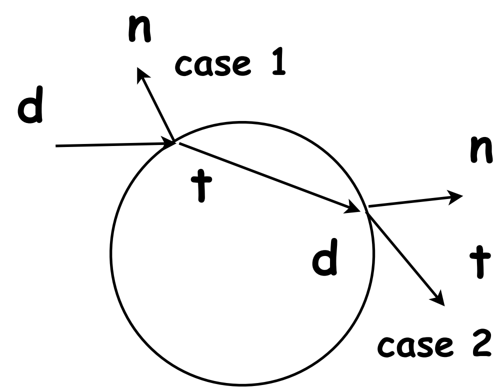
- The reflectance R of a dielectric depends on the refractive index η of the material and the angle of incidence θ
  - dielectric: a transparent material that refracts light
- $R_0 = (\eta - 1)^2 / (\eta + 1)^2$, for $\theta = 0$
  - Depends on the material
  - For example, burnished copper has roughly:

$$R_{0,blue} = 0.1,\ R_{0,green} = 0.2,\ R_{0,red} = 0.5$$

Schlick's Approximation

- $R_\theta = R_0 + (1 - \cos \theta)^5 (1 - R_0)$, for $0 < \theta < 90$
  - As θ increases, R_θ increases
  - $R_{90} = 1$, the light is tangential to the surface
  - θ is always the larger of the internal and external angles relative to the normal (e.g. always the angle in air)

Incorporating Fresnel Effect



if (p is on a dielectric) then // p is the ray hit point

r = reflect(**d**,**n**) // calculate reflection ray **r**

C**r =** shade(**p,r**)**;**

if (**d**.**n** <0) then // going into medium: **case 1**

refract(**d**,**n**,η,**t)**; // t is the transmitted ray

C**t** = shade(**p**,**t**);

cos = **-d.n**

else //going out of medium: **case 2**

if (refract(**d**,**-n**, 1/η,**t**) = true) then // no total internal refraction

C**t** = shade(**p**,**t**)

cos = **t**.**n**

else

return C**r** // just the reflection color

R0 = $(\eta-1)^2 / (\eta+1)^2$

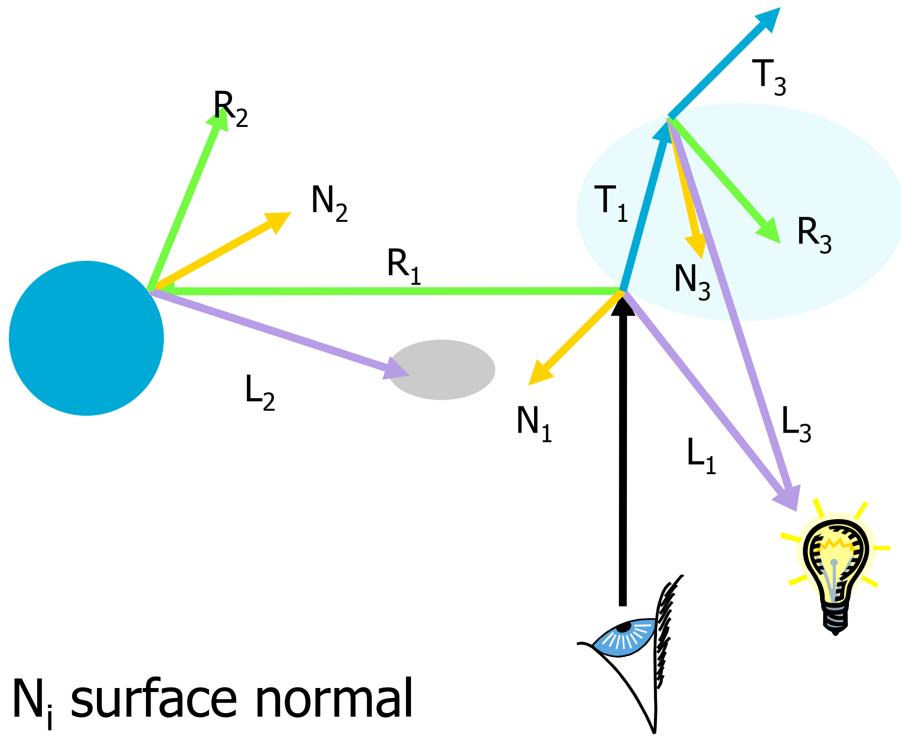
R = R0 + (1-R0)$(1-\cos)^5$

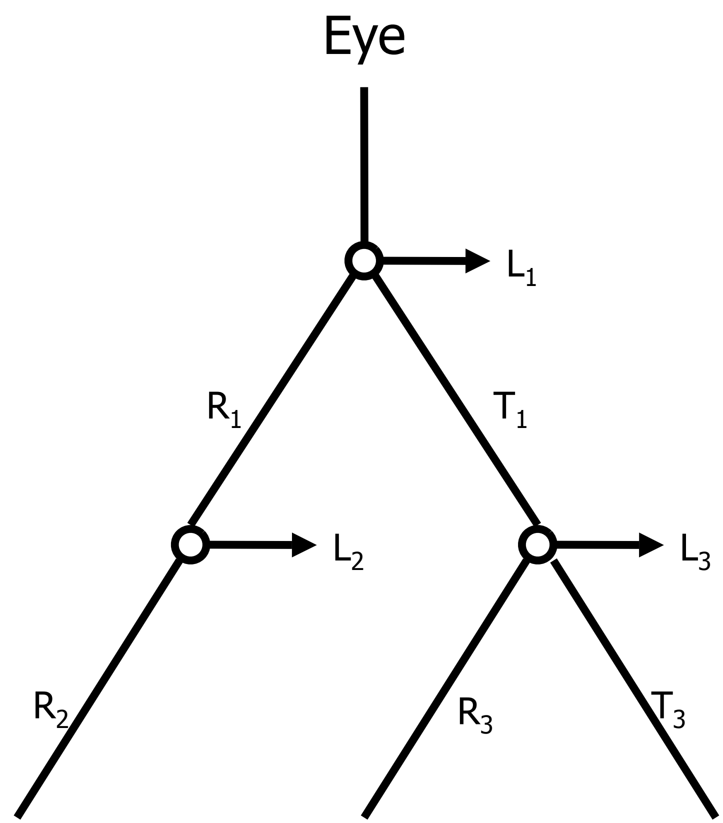
return R*C**r** + (1-R) * C**t**

Fresnel Reflectance

Lafortune et al.

The Ray Tree – Recursion!





N_i surface normal

R_i reflected ray

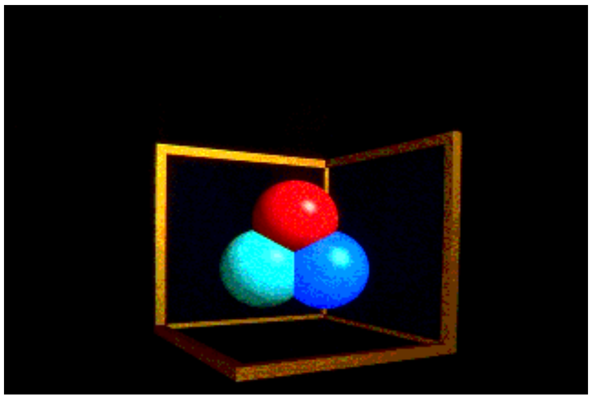
L_i shadow ray

T_i transmitted (refracted) ray

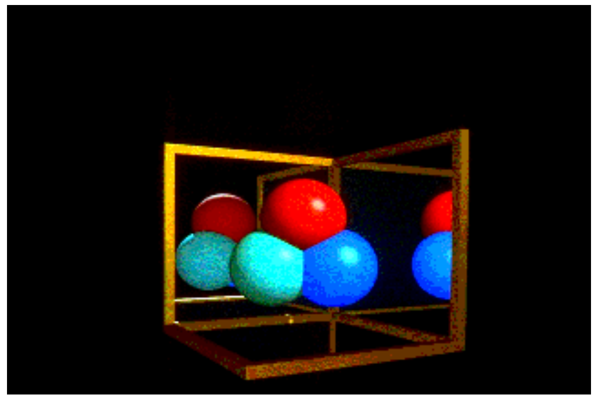
Infinite Recursion

- Stopping criteria
  - Recursion depth
    - Stop after some bounces
  - Ray contribution
    - Stop if transparency/transmitted attenuation becomes too small
  - Usually do both

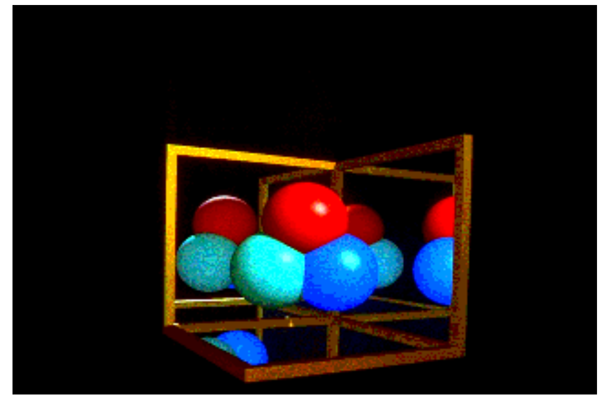
Recursion for Reflection



0 recursion



1 recursion



2 recursions