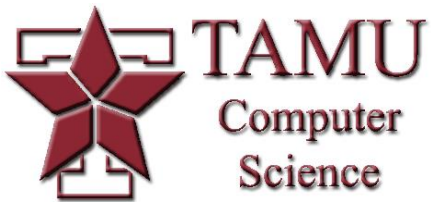
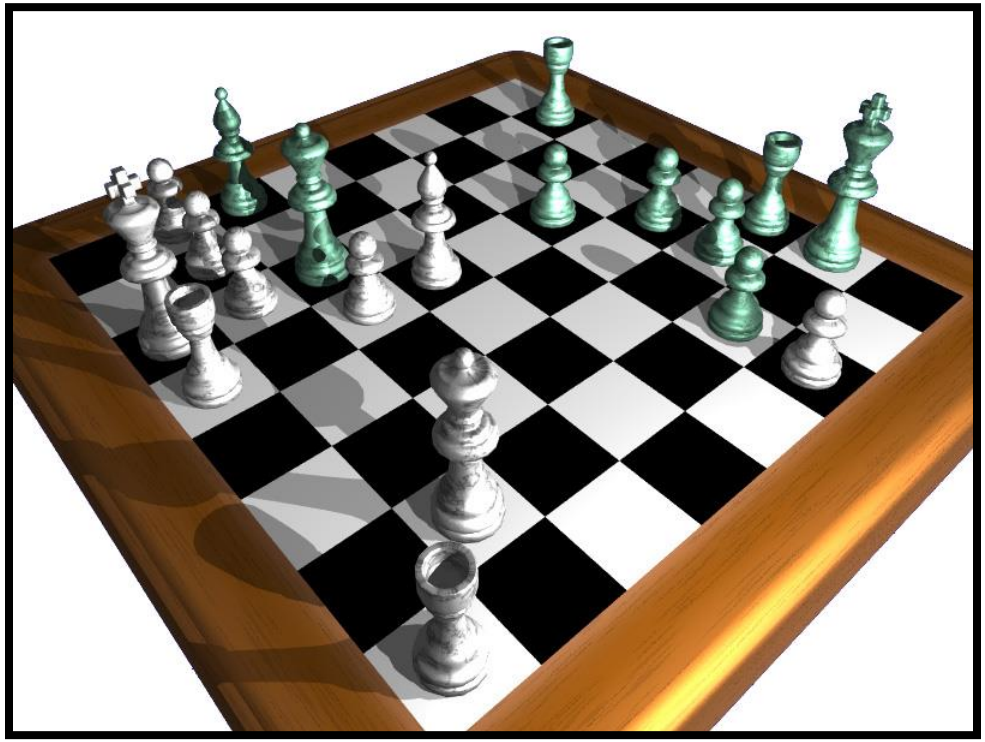
Ray Tracing

Dr. Scott Schaefer



Ray Tracing

- Provides rendering method with
  - Refraction/Transparent surfaces
  - Reflective surfaces
  - Shadows



Ray Tracing

- Provides rendering method with
  - Refraction/Transparent surfaces
  - Reflective surfaces
  - Shadows

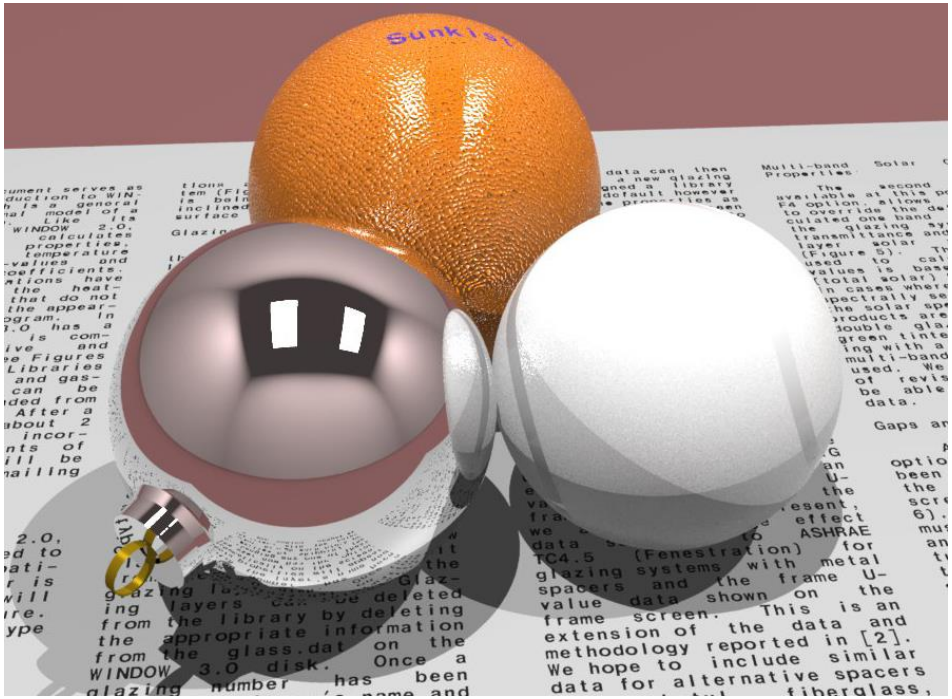


Image taken from http://radsite.lbl.gov/radiance/book/img/plate10.jpg

Ray Tracing

- Provides rendering method with
  - Refraction/Transparent surfaces
  - Reflective surfaces
  - Shadows

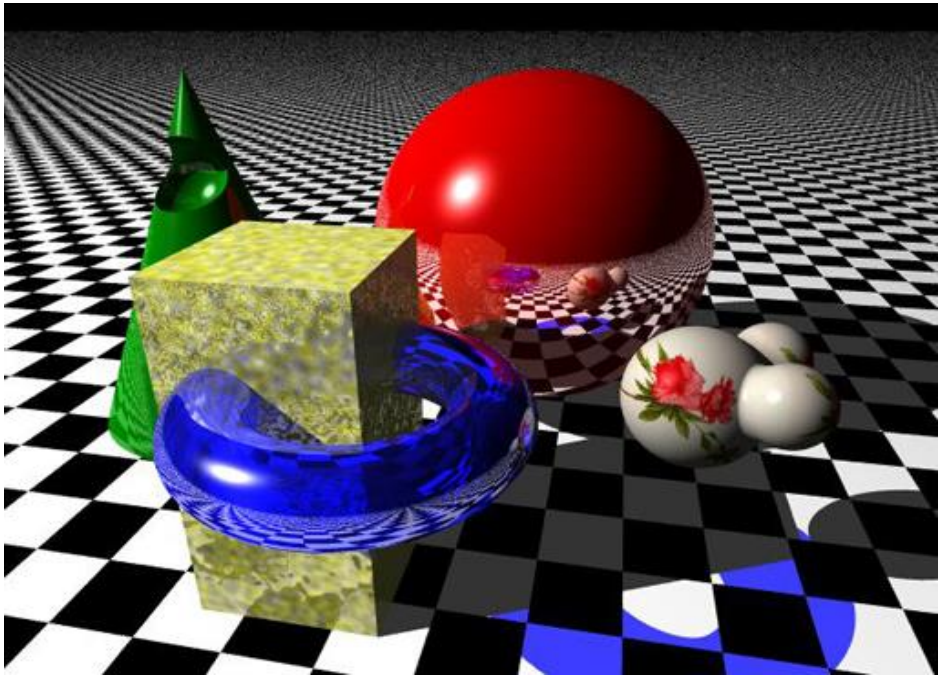
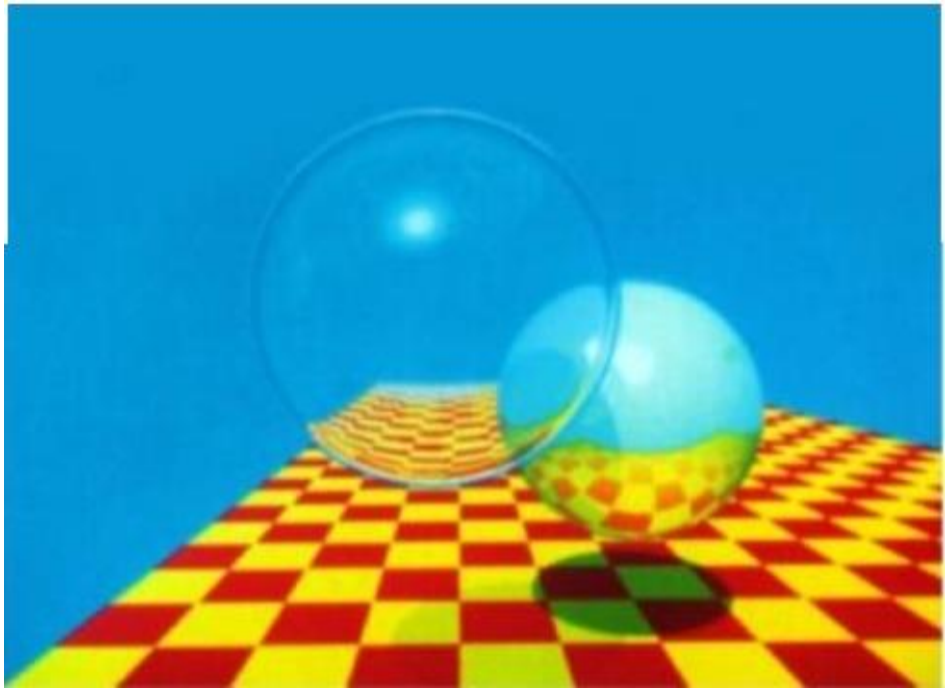


Image taken from http://www.tjhsst.edu/~dhyatt/superap/samplex.jpg

Ray Tracing

- Provides rendering method with
  - Refraction/Transparent surfaces
  - Reflective surfaces
  - Shadows

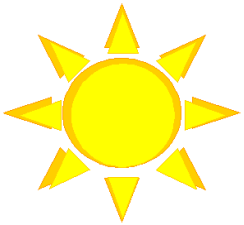


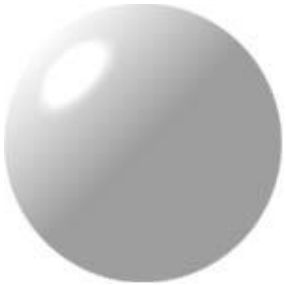
Essential Information for Ray Tracing

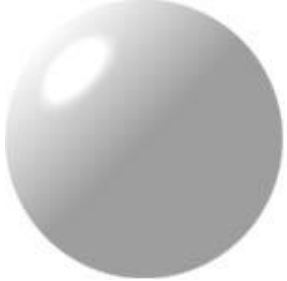
- Eye point
- Screen position/orientation
- Objects
  - Material properties
  - Reflection/Refraction coefficients
  - Index of refraction
- Light sources

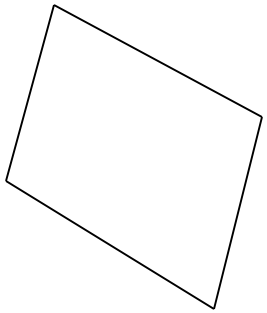
Recursive Ray Tracing

- For each pixel
  - Intersect ray from eye through pixel with all objects in scene
  - Find closest (positive) intersection to eye
  - Compute lighting at intersection point
  - Recur for reflected and refracted rays (if necessary)

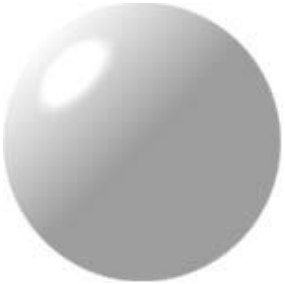




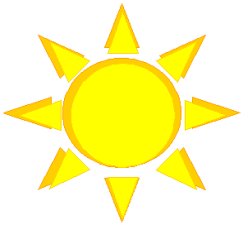


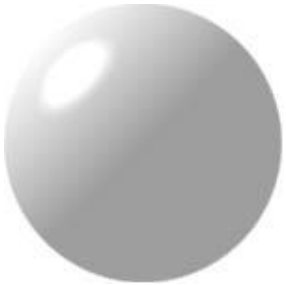


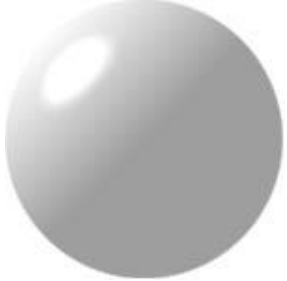
screen

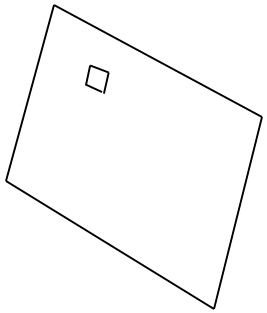


eye

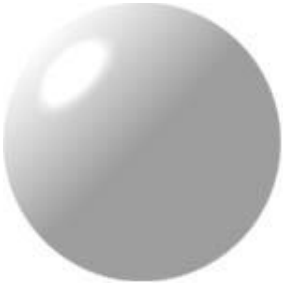




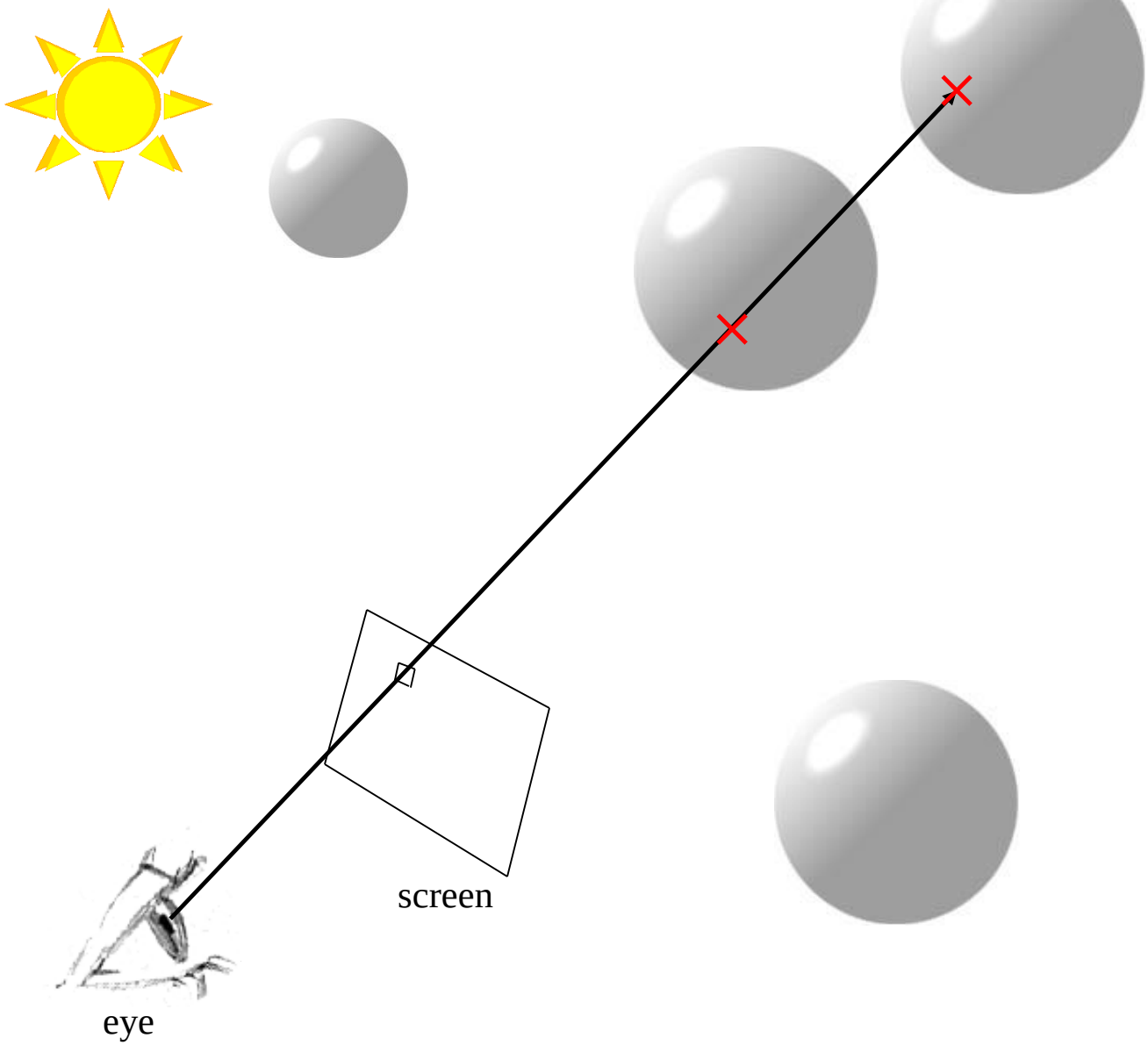


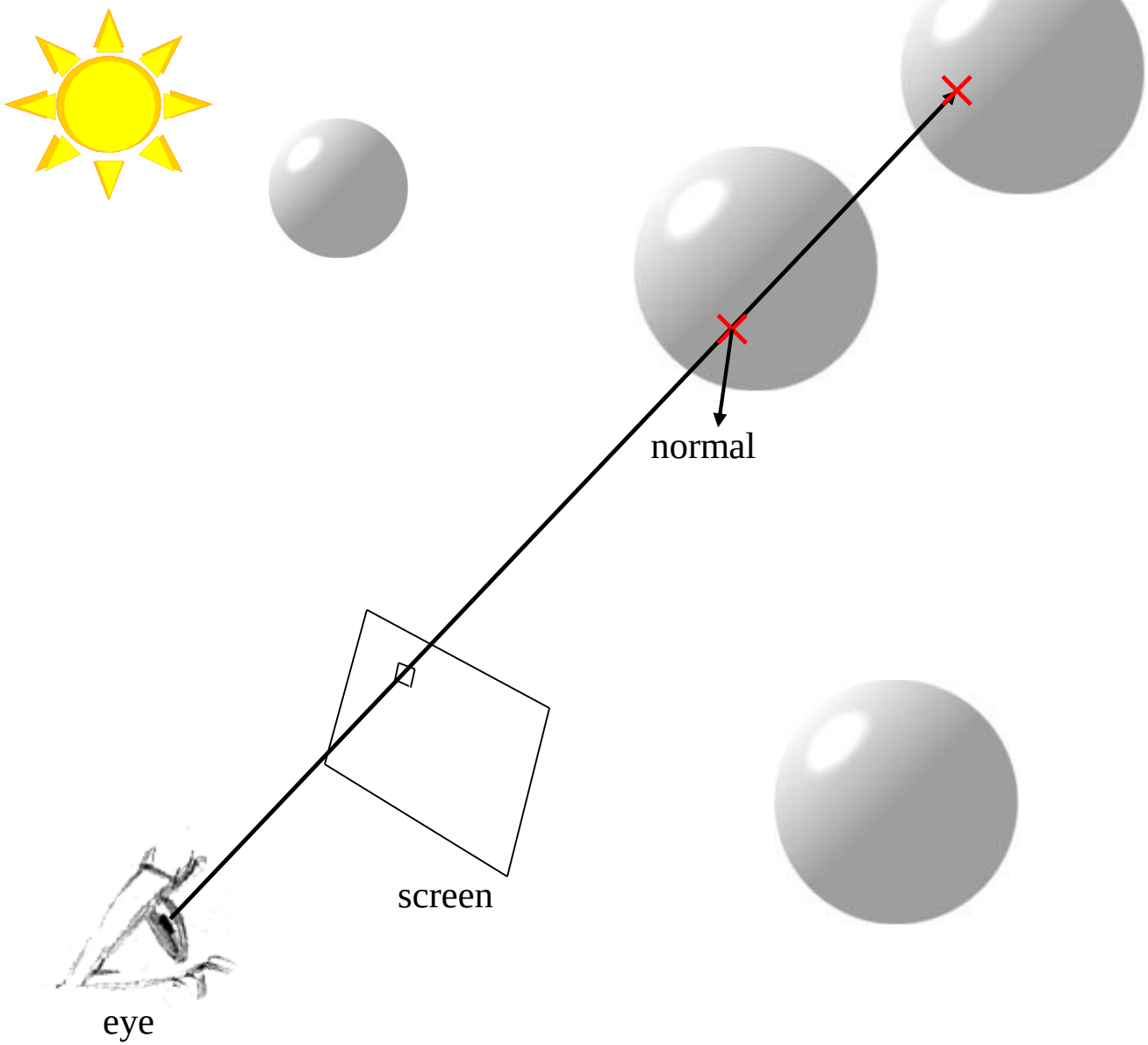


screen



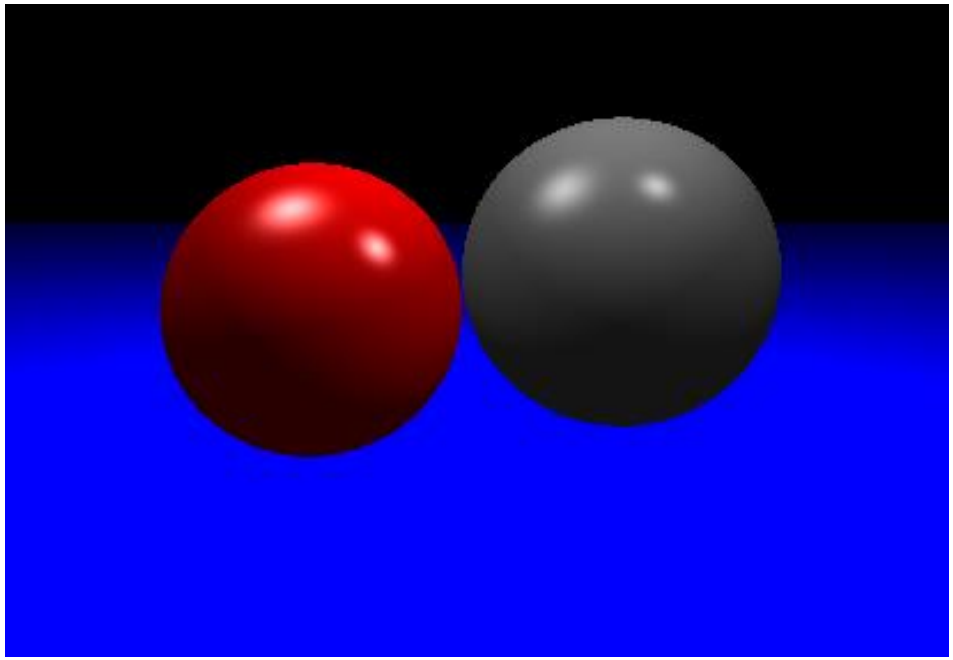
eye


screen
eye


normal
screen
eye

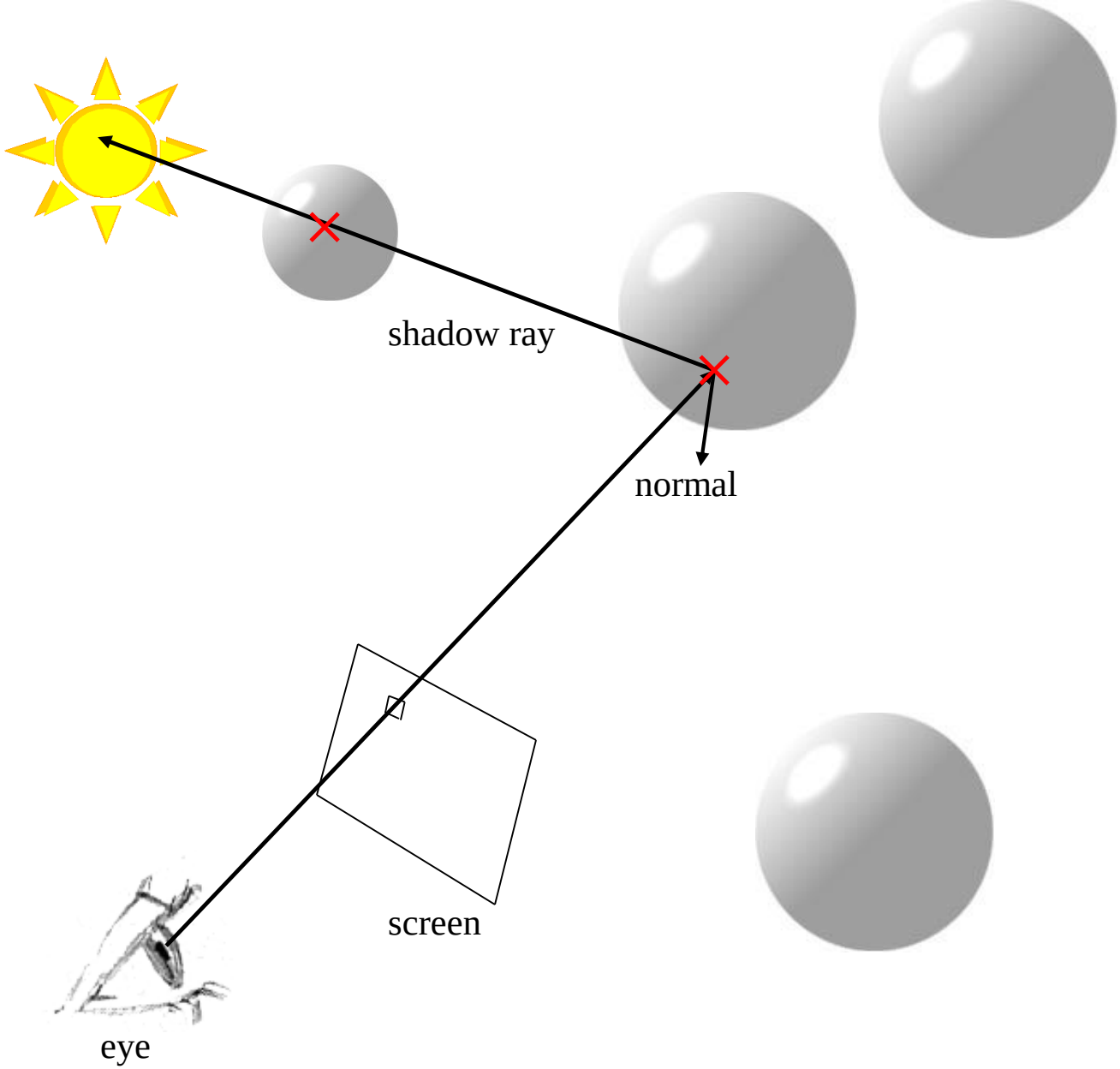
Ray Casting

- Removes hidden surfaces
- Per-pixel lighting computations



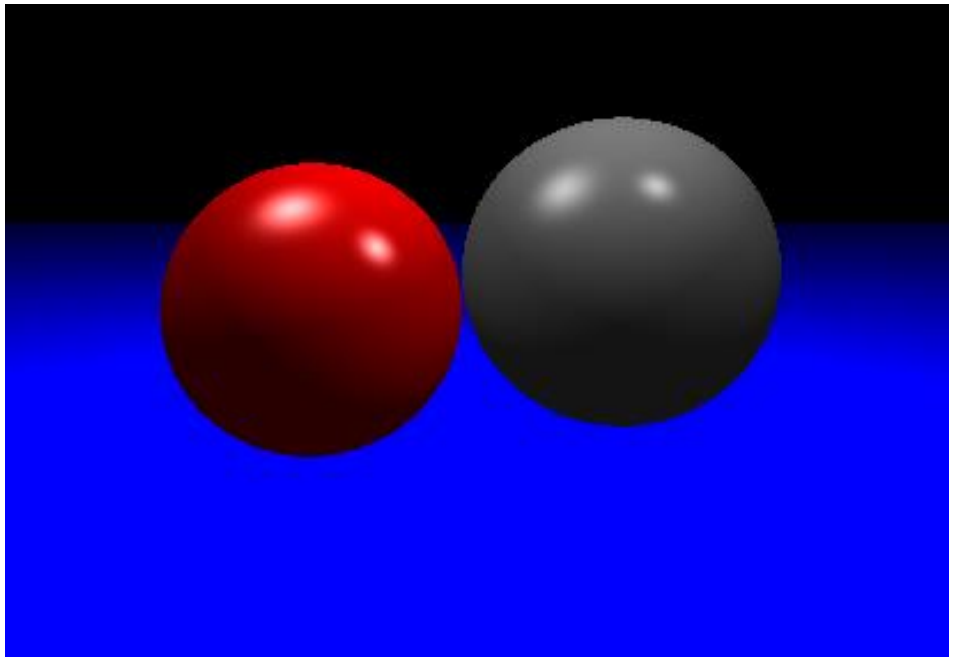
Shadows

- Cast a **virtual ray** to each light source
- If ray hits an opaque object before the light, then omit contribution of that light
- If ray hits a semi-transparent object, scale the contribution of that light and continue to look for intersections
- Note: objects may be self-shadowing!!!


shadow ray
normal
screen
eye

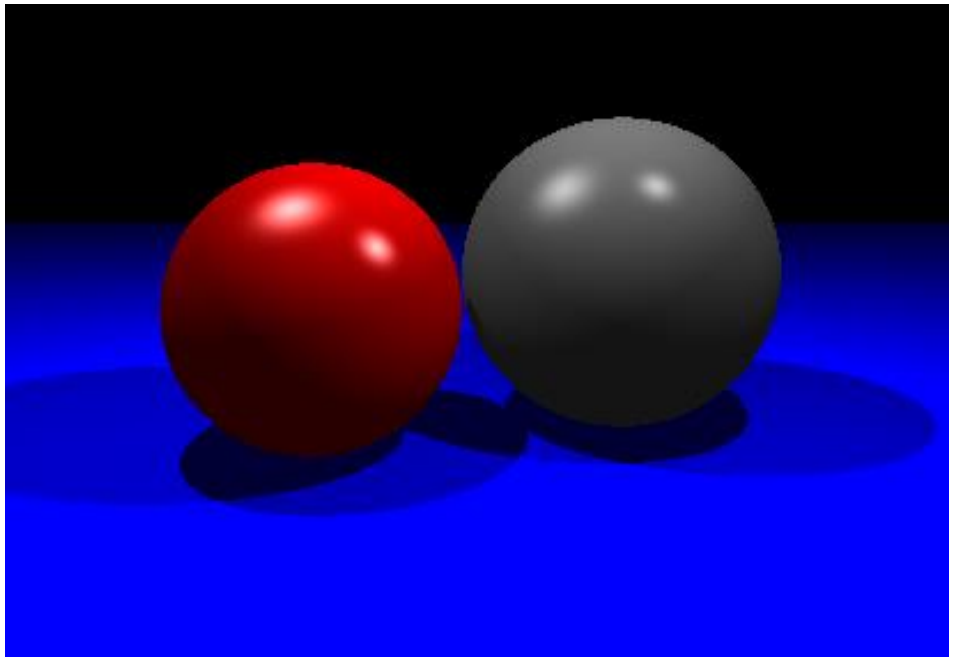
Shadows

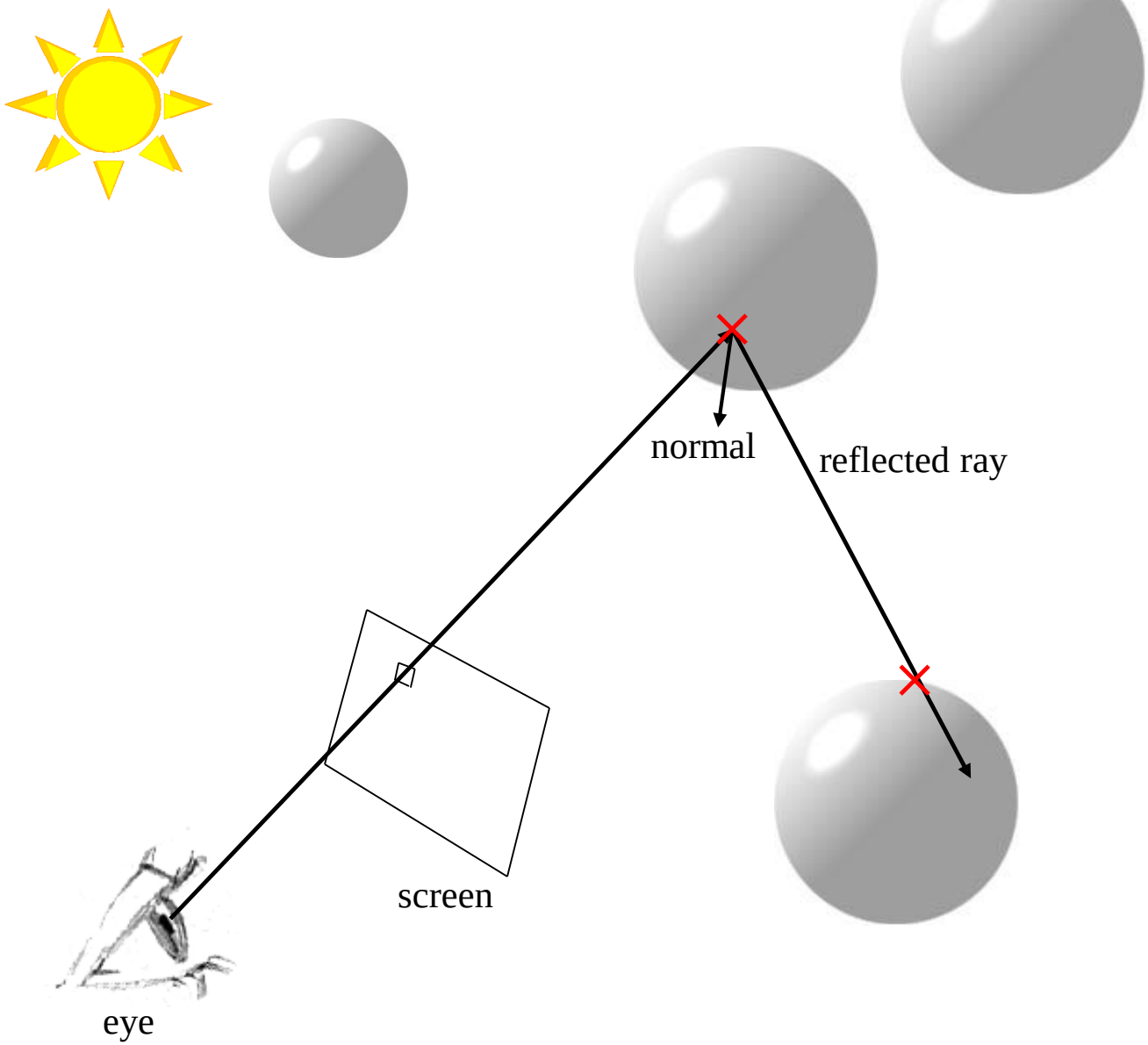
$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right)$$

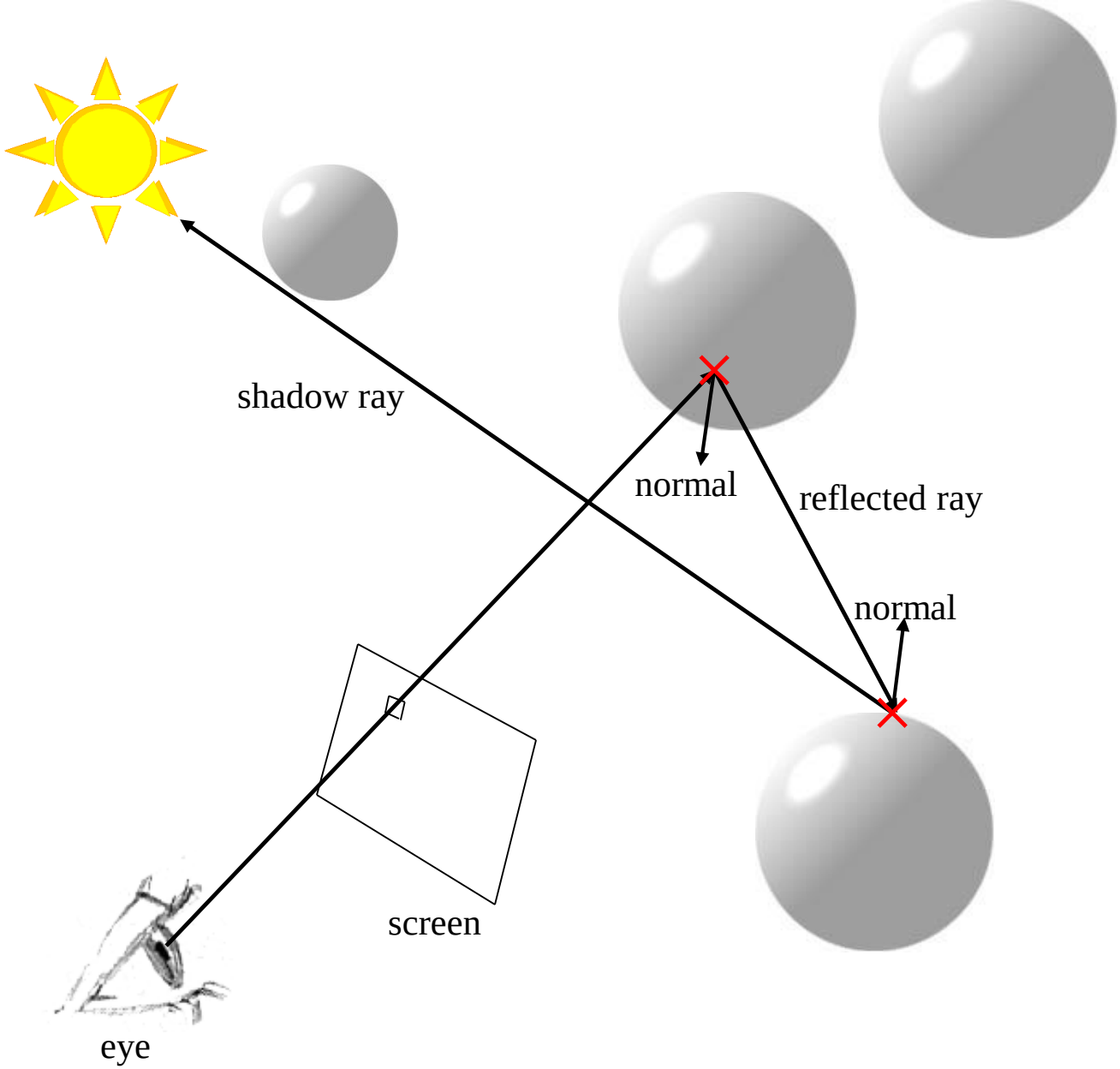


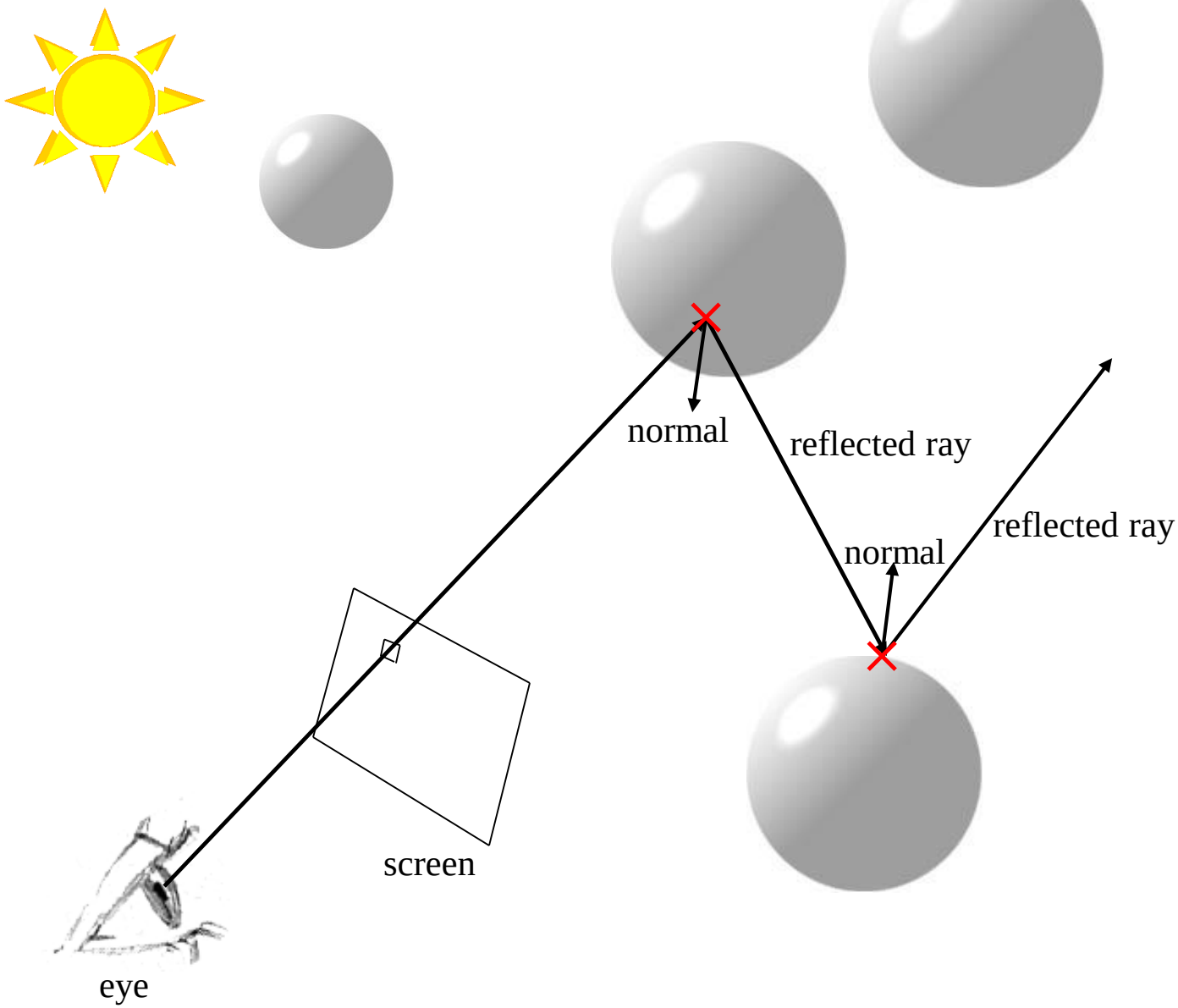
Shadows

$$I = k_a A + \sum_i If\left(shadow, 0, C_i\left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n\right)\right)$$




normal
reflected ray
screen
eye

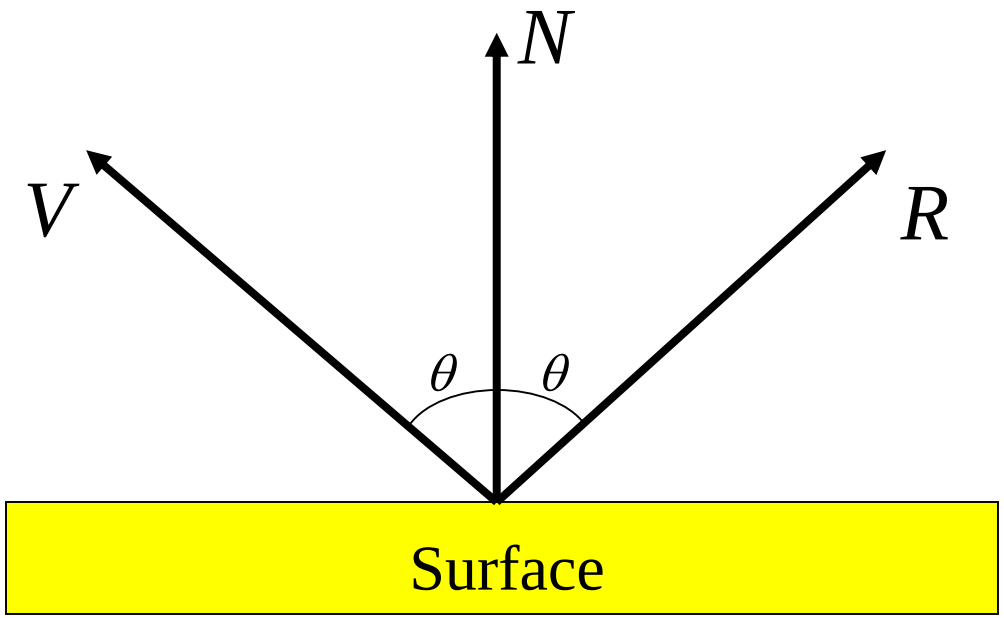

shadow ray
normal
reflected ray
normal
screen
eye

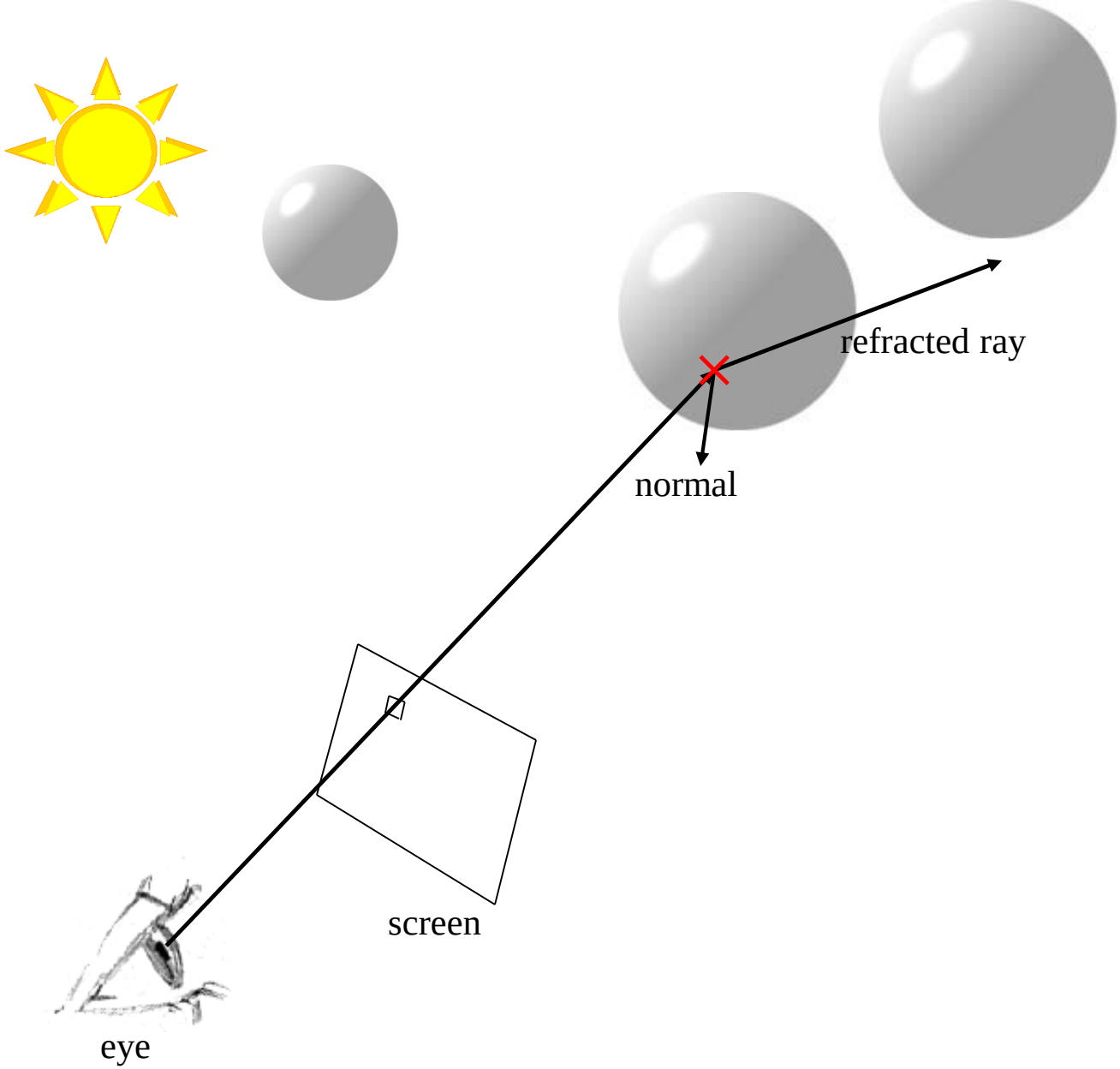

normal
reflected ray
normal
reflected ray
screen
eye

Reflection

- Mirror-like/Shiny objects

$$R = 2(V \cdot N)N - V$$




refracted ray
normal
screen
eye

Refraction

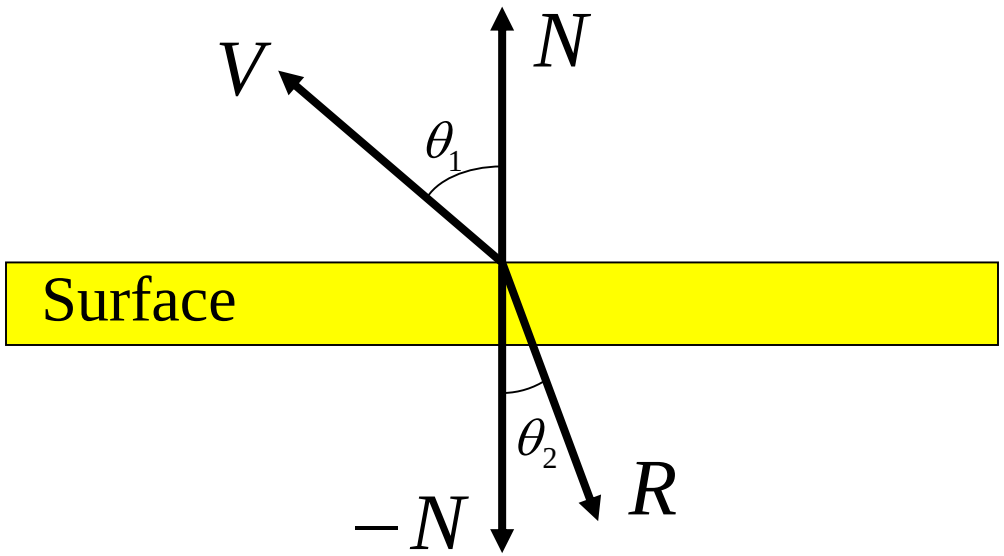
- Bending of light caused by different speeds of light in different medium
- Each (semi-)transparent object has an index of refraction n_i or phase velocity of light c_i
- Use Snell's law to find refracted vector

Image taken from http://hyperphysics.phy-astr.gsu.edu/hbase/geoopt/refr2.html

Snell's Law

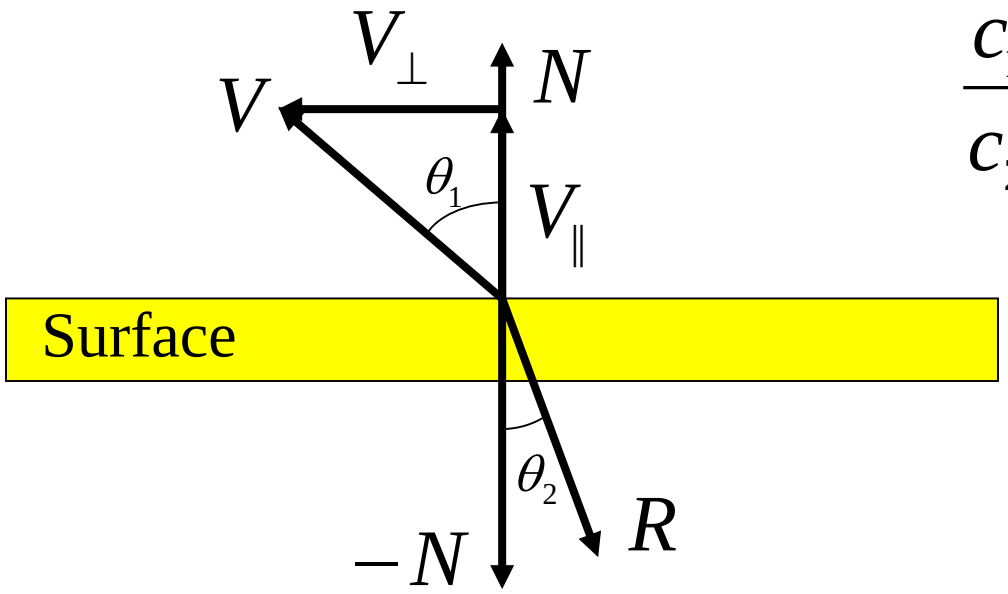
c=speed of light in vacuum

$$n_1 = \frac{c}{c_1}$$



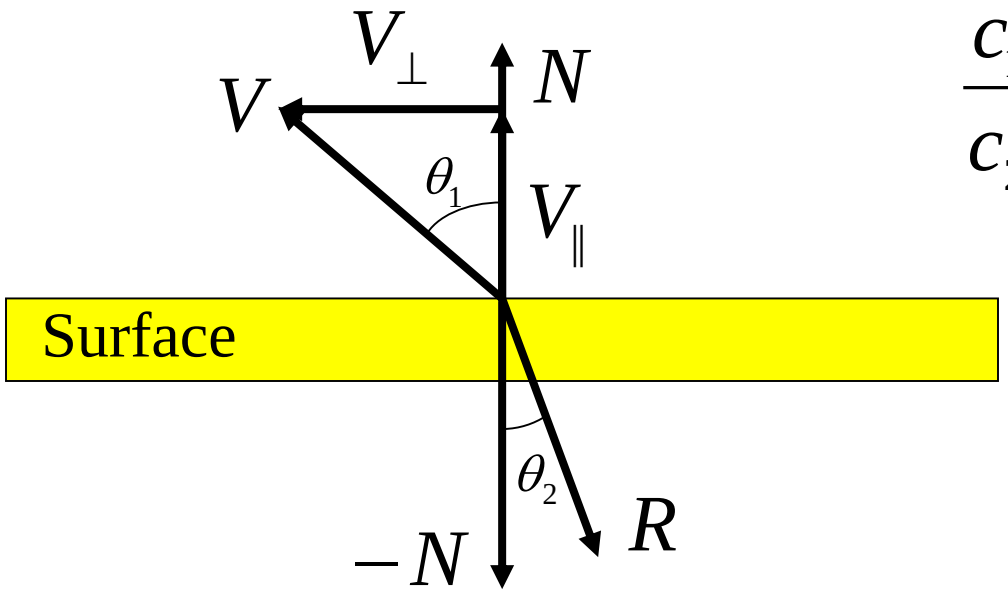
$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

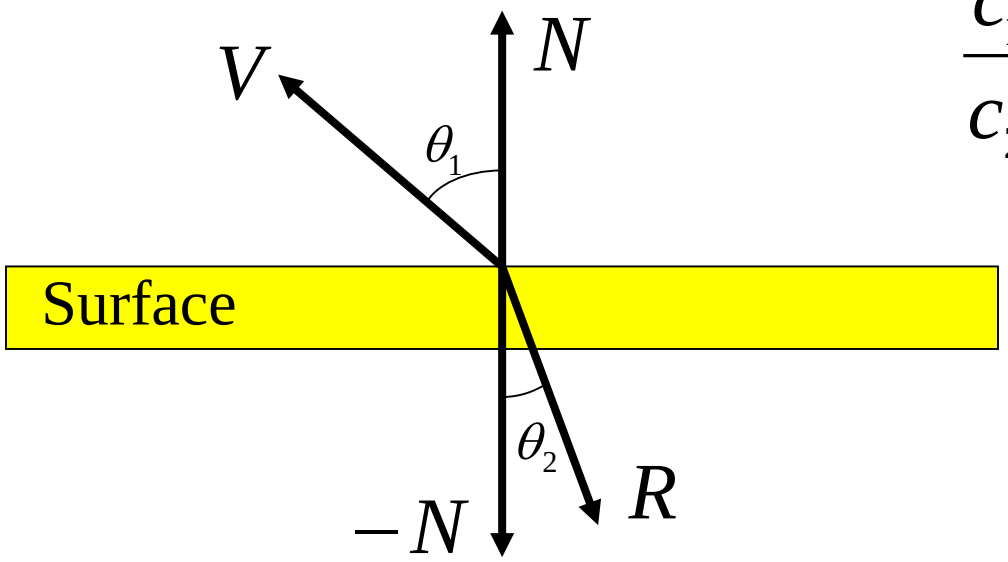
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \cos(\theta_2)(-N) + \sin(\theta_2)\left(\frac{-V_\perp}{|V_\perp|}\right)$$

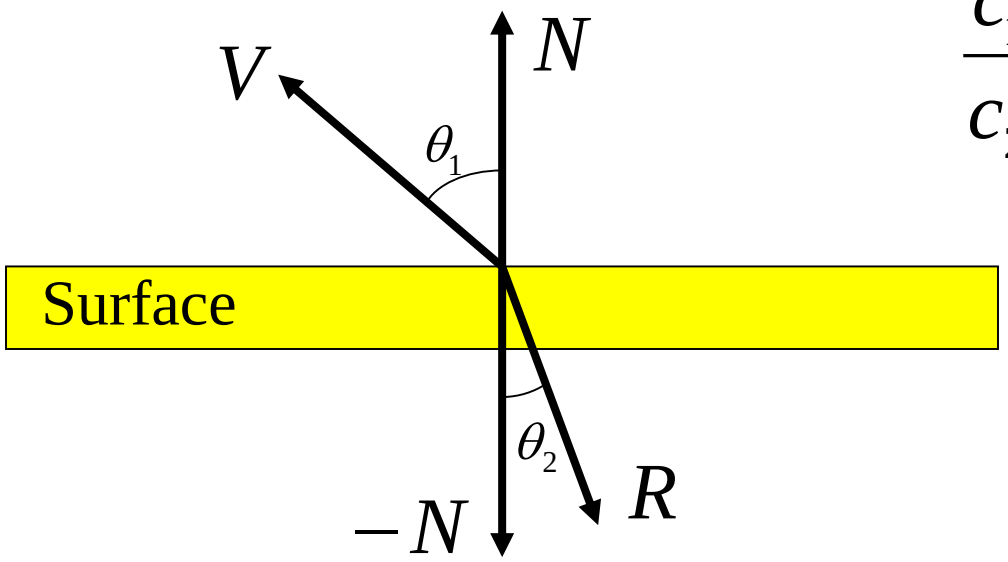
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \cos(\theta_2)(-N) + \sin(\theta_2)\left(\frac{(V \cdot N)N - V}{|V_\perp|}\right)$$

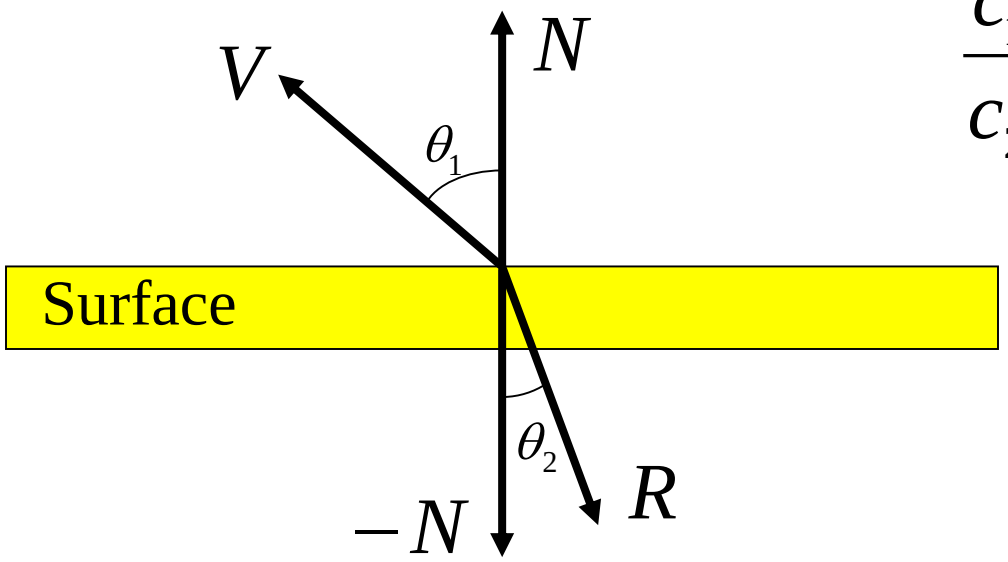
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \cos(\theta_2)(-N) + \sin(\theta_2)\left(\frac{(V \cdot N)N - V}{\sin(\theta_1)}\right)$$

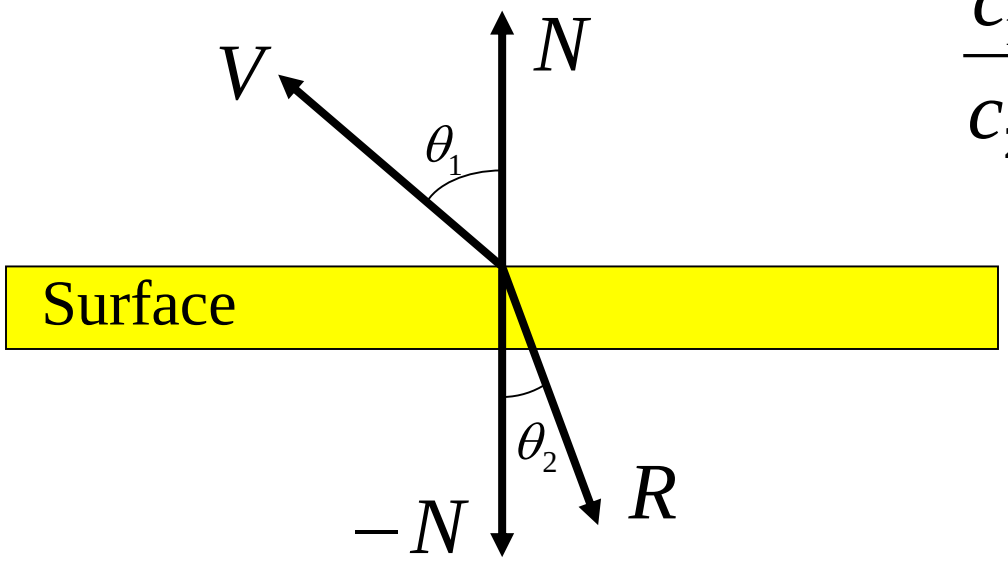
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \cos(\theta_2)(-N) + \frac{c_2}{c_1}((V \cdot N)N - V)$$

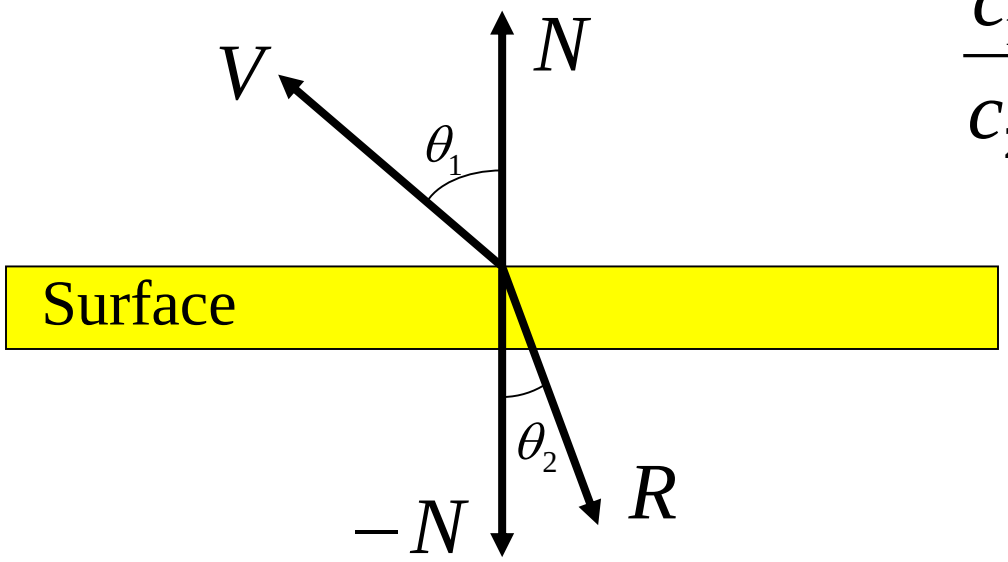
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \sqrt{1-\sin(\theta_2)^2}\,(-N) + \frac{c_2}{c_1}((V \cdot N)N - V)$$

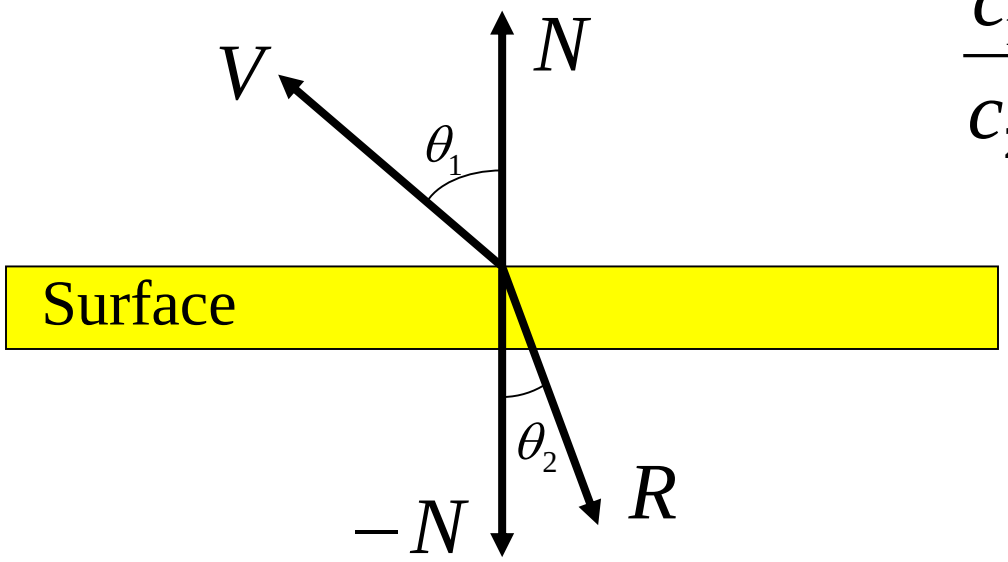
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \sqrt{1 - \sin(\theta_1)^2 \frac{{c_2}^2}{{c_1}^2}}(-N) + \frac{c_2}{c_1}((V \cdot N)N - V)$$

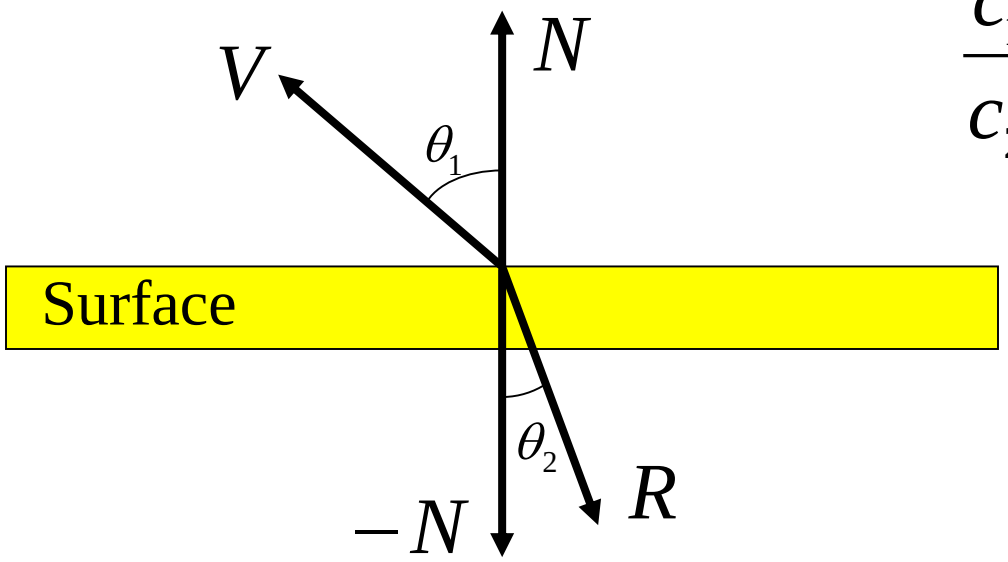
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \sqrt{c_1^{\,2} - c_2^{\,2}\sin(\theta_1)^2}\,\frac{1}{c_1}(-N) + \frac{c_2}{c_1}((V \cdot N)N - V)$$

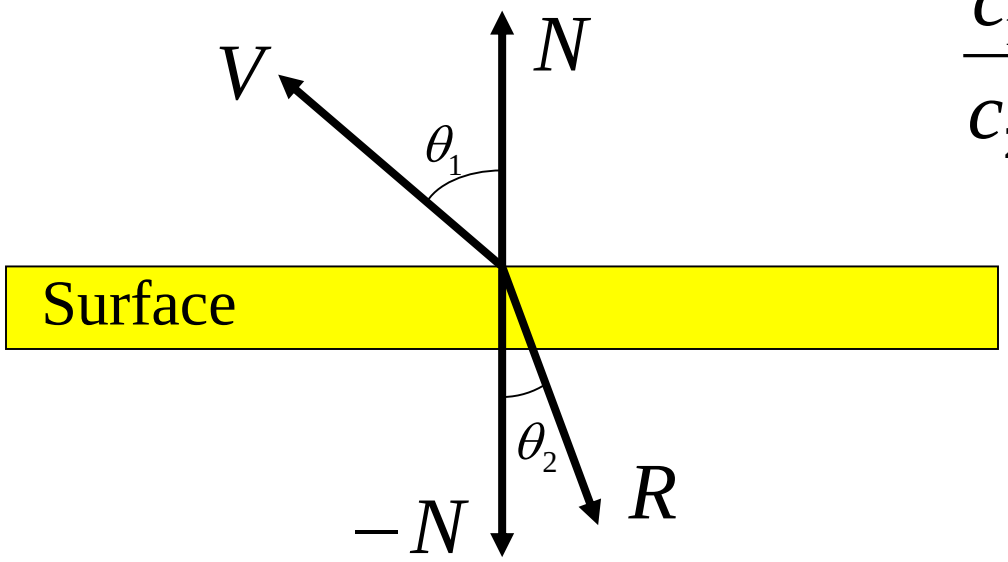
Snell's Law



$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \sqrt{c_1^{\,2} - c_2^{\,2}(1-(V\cdot N)^2)}\,\frac{1}{c_1}(-N) + \frac{c_2}{c_1}((V\cdot N)N - V)$$

Total Internal Reflection



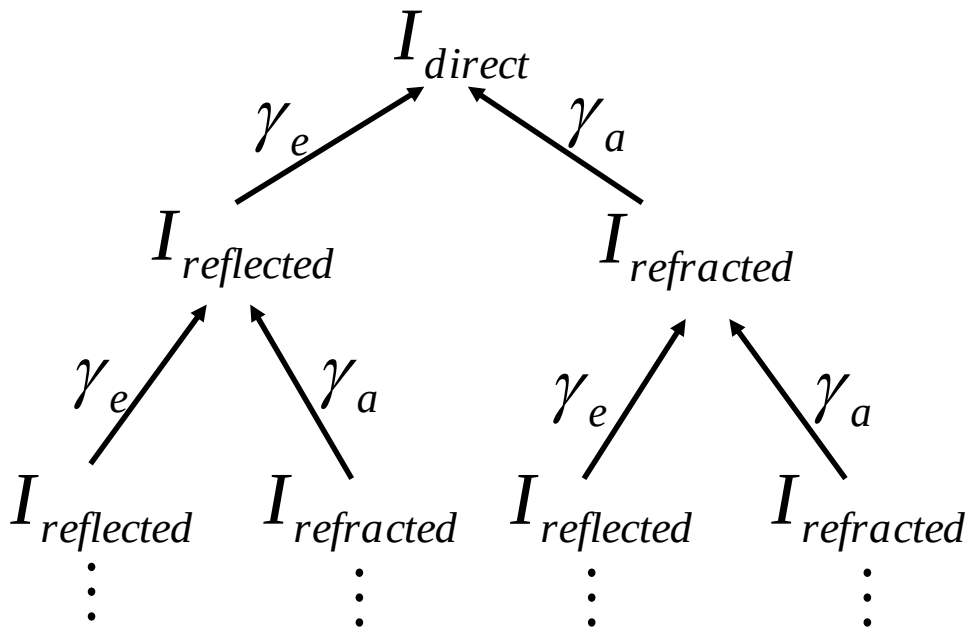
$$\frac{c_1}{c_2} = \frac{\sin(\theta_1)}{\sin(\theta_2)}$$

$$R = \sqrt{c_1^{\,2} - c_2^{\,2}(1-(V\cdot N)^2)}\,\frac{1}{c_1}(-N) + \frac{c_2}{c_1}((V\cdot N)N - V)$$

Recursive Ray Tracing

$$I = I_{direct} + \gamma_e I_{reflected} + \gamma_a I_{refracted}$$

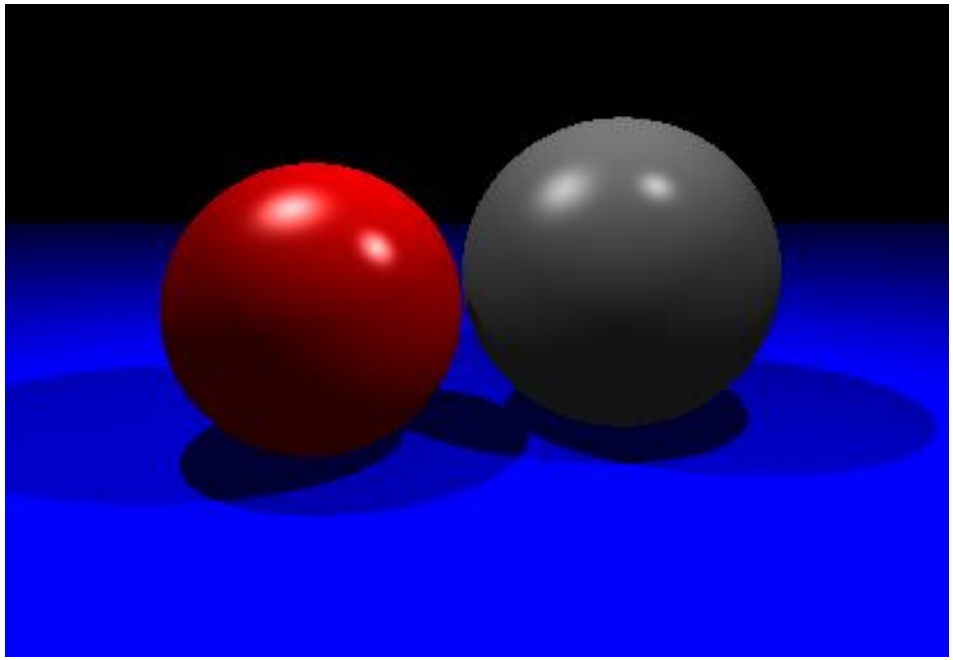
$$I_{direct} = I_{ambient} + I_{diffuse} + I_{specular}$$



Truncate at finite depth!

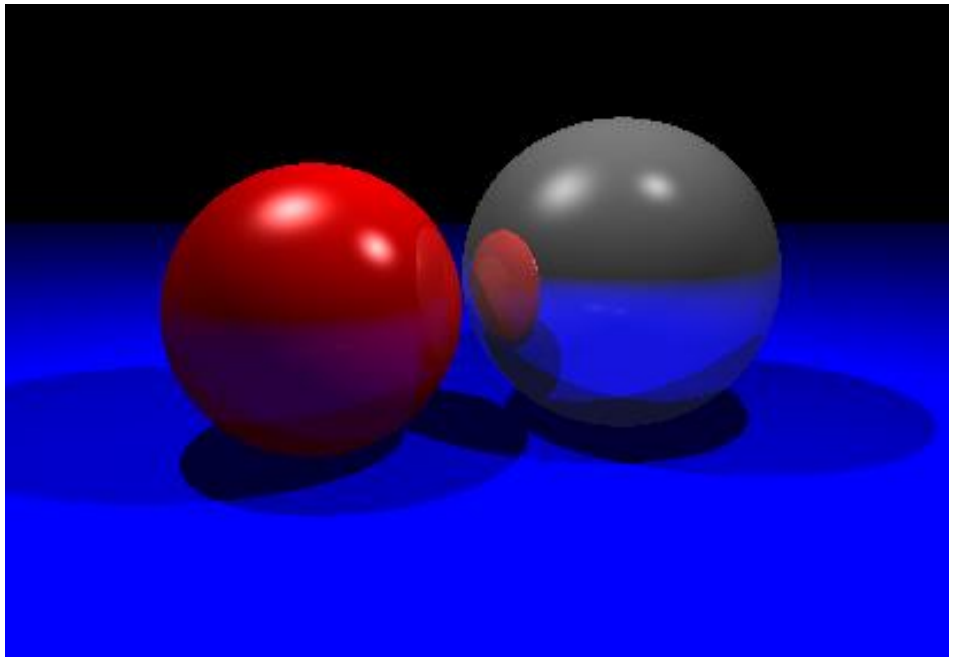
Recursive Ray Tracing

- Recur for reflective/transparent objects



Recursive Ray Tracing

- Recur for reflective/transparent objects



Optimizations

- Lots of rays to cast!
- Ray-Surface intersections are expensive

- Associate with each object
  - Bounding box in 3-space
- If ray doesn’t intersect box, then ray doesn’t intersect object

Parallel Processing

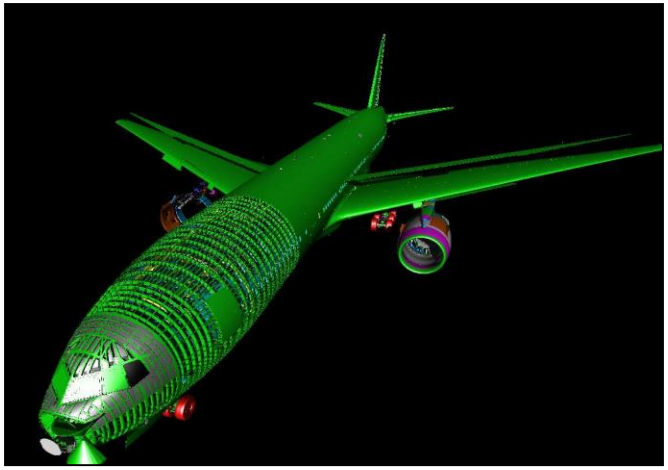
- Ray tracing is a trivially parallel algorithm!
  - Cast rays in parallel
  - Cast reflection, refraction, shadow rays in parallel
  - Calculate ray/surface intersections independently in parallel

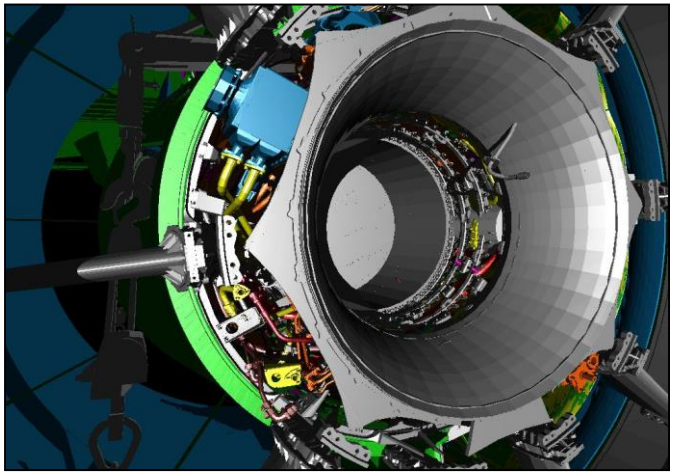
Ray Tracing: Special Effects

copyright Newline Cinema

Ray Tracing: Video Games

Ray Tracing: Massive Models





Extensions of Ray Tracing

- Only considers totally specular interactions
  - **rays either reflect perfectly or refract perfectly**
- Ray traced scenes don't show “color bleed”