Ray Tracing

Courtesy of Prof. Rasmussen

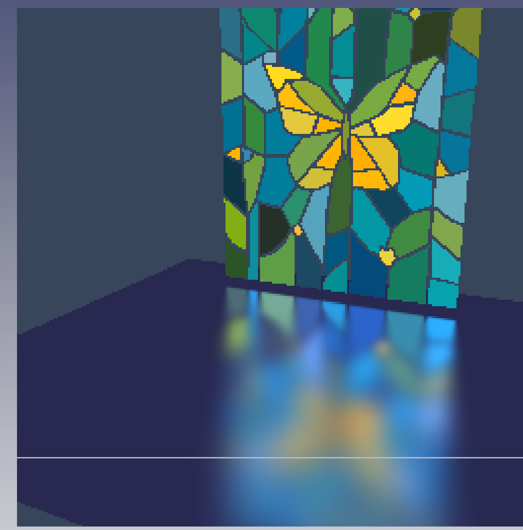
University of Delaware

Outline

- Recursive rays
  - Reflection
  - Refraction
- HW #3 details

Ray "tracing" for more realism

- Ray casting does not account for two important visual phenomena:
  - Mirror-like surfaces should **reflect** other objects in scene
  - Transparent surfaces should **refract** scene objects behind them



courtesy of J. Arvo

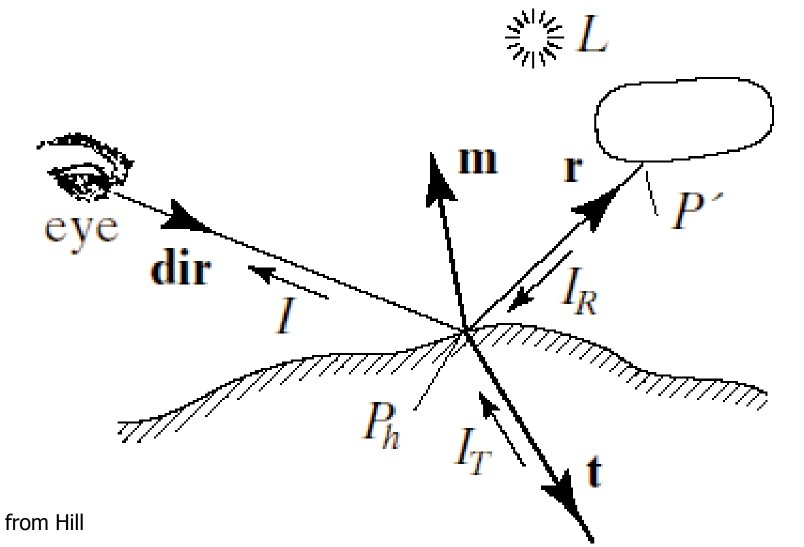
Glossy reflection

Refraction

UNIVERSITY OF DELAWARE

Ray Tracing

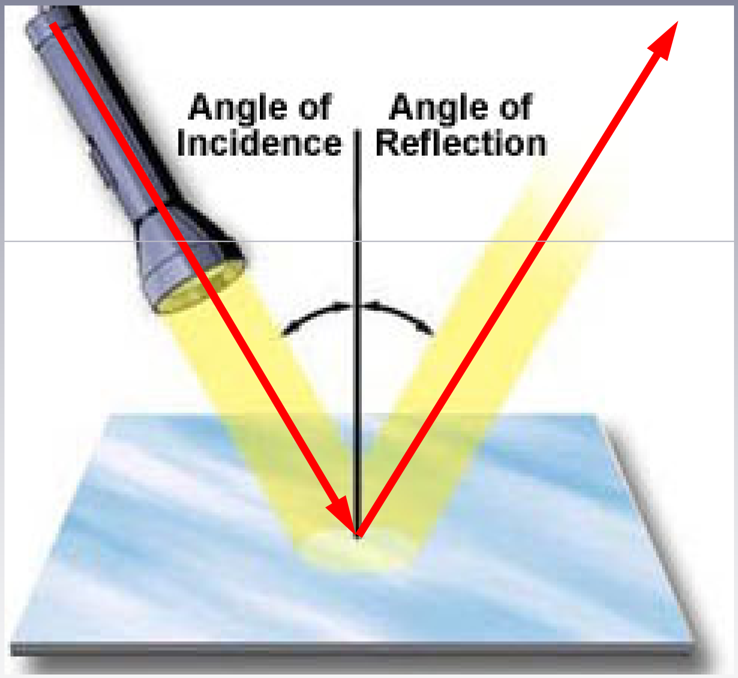
- Model: Perceived color at point **p** is an additive combination of local illumination (e.g., Phong), reflection, and refraction effects
- Compute reflection, refraction contributions by **tracing** respective rays back from **p** to surfaces they came from and evaluating local illumination at those locations
- Apply operation **recursively** to some maximum depth to get:
  - Reflections of reflections of ...
  - Refractions of refractions of ...
  - And of course mixtures of the two



from Hill

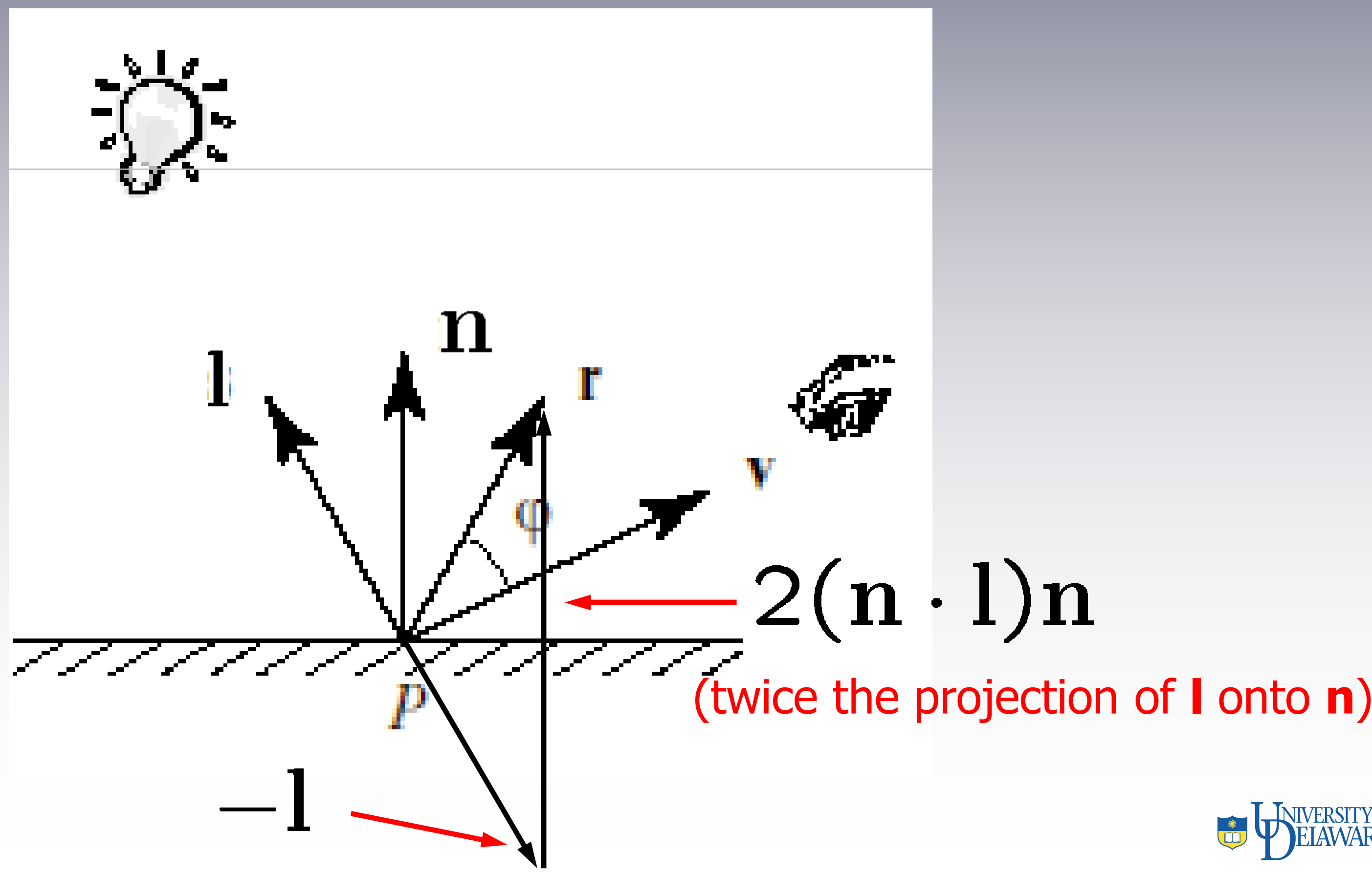
Reflections

incident ray **v** reflected ray **r**



Review: Reflectance direction for Phong model

- We calculated **r** from normal **n**, light direction **l** via: $\mathbf{r} = 2(\mathbf{n}\cdot\mathbf{l})\mathbf{n} - \mathbf{l}$

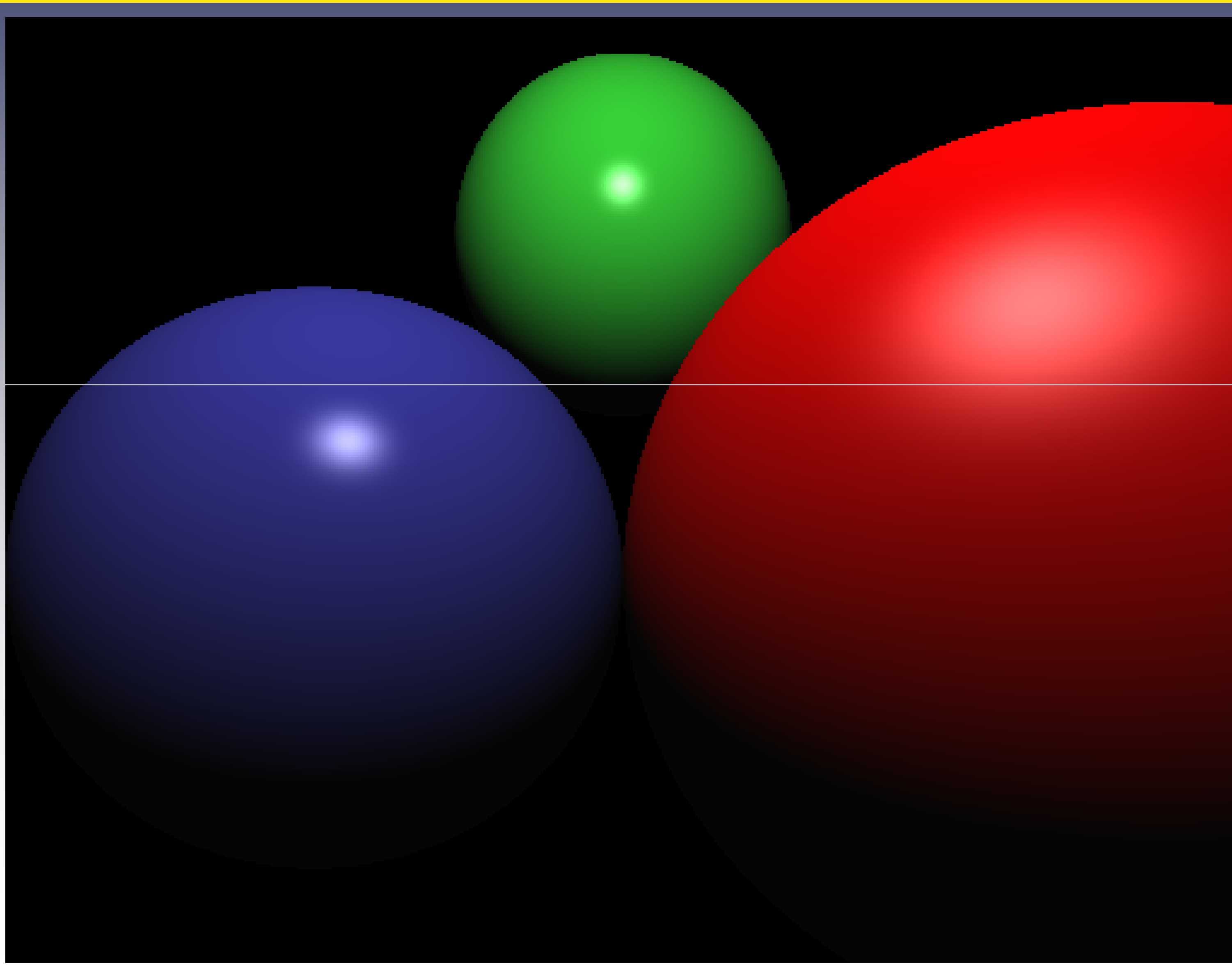


Ray Tracing Reflection Formula

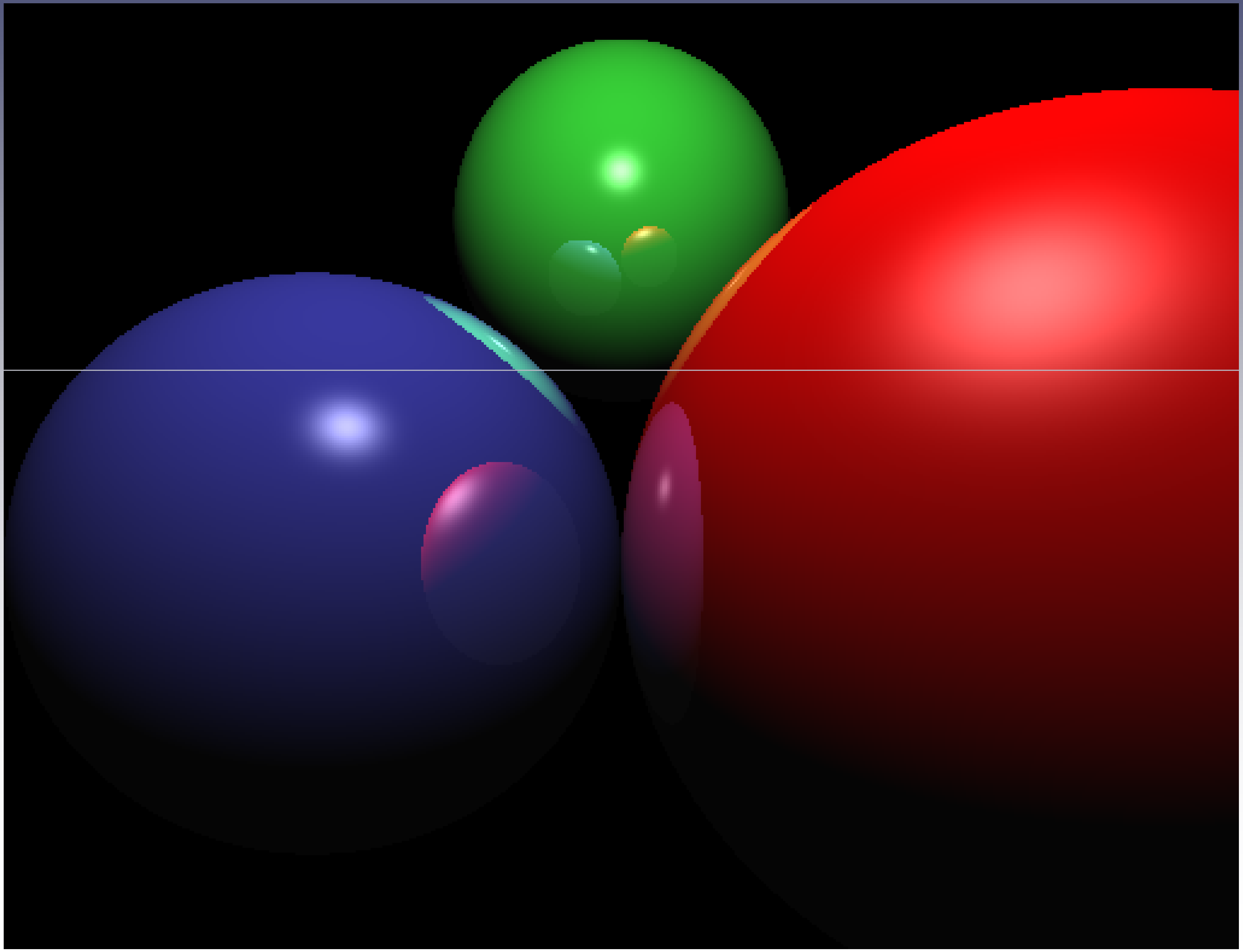
- The formula used for Phong illumination is not what we want here because our incident ray **v** is pointing **in** toward the surface, whereas the light direction **l** was pointed **away** from the surface
- So just negate the formula to get:

$$\mathbf{r} = \mathbf{v} - 2(\mathbf{n} \cdot \mathbf{v})\mathbf{n}$$

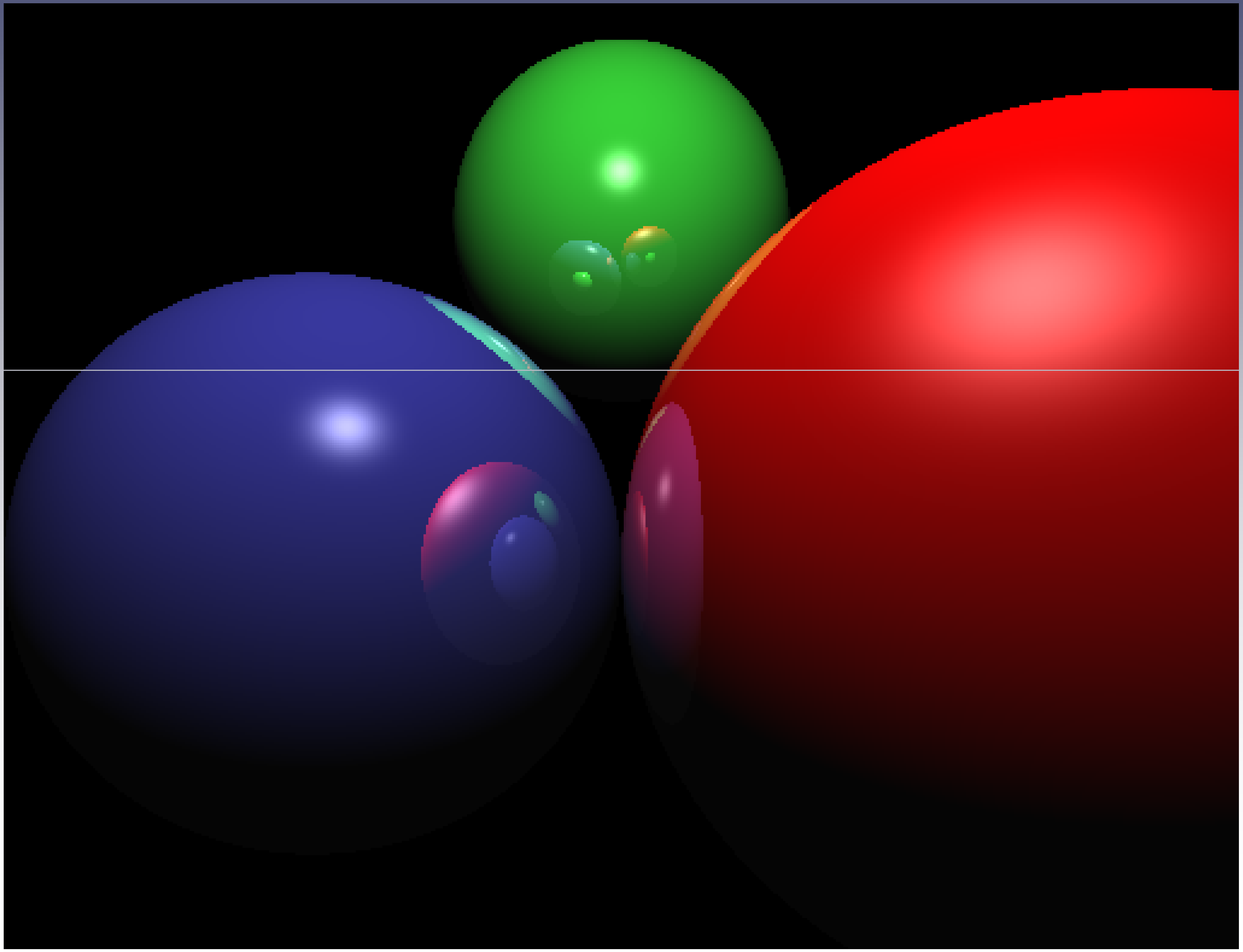
Example: Reflections at depth = 0



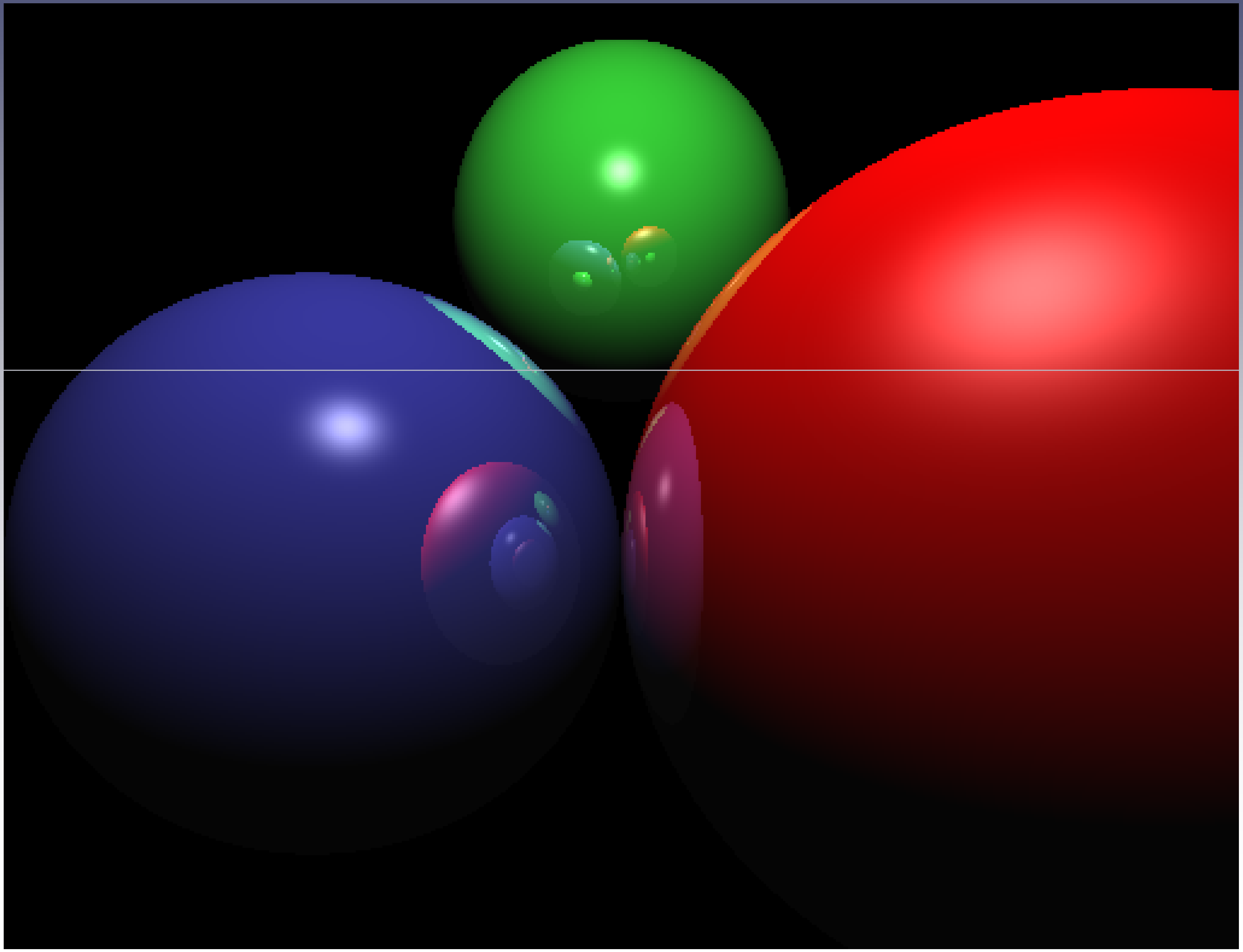
Example: Reflections at depth = 1



Example: Reflections at depth = 2



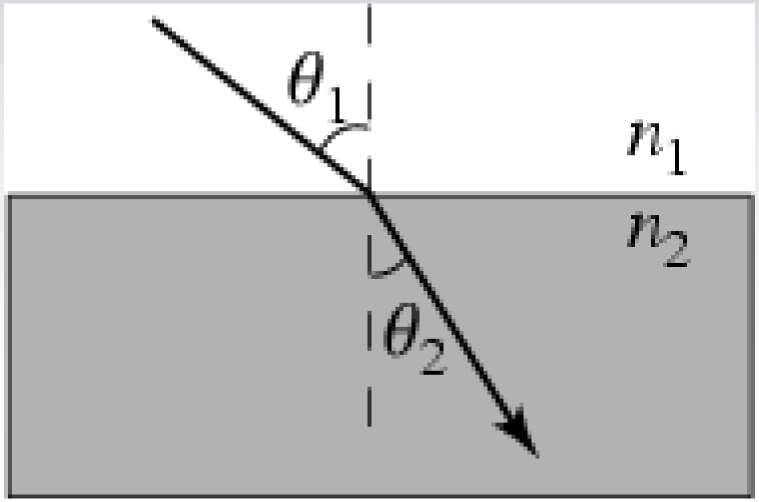
Example: Reflections at depth = 3



Refraction

- Definition: Bending of light ray as it crosses interface between media (e.g., air → glass or vice versa)
- Index of refraction (IOR) *n* for a medium: Ratio of speed of light in vacuum to that in medium (wavelength-dependent ⇒ prisms)
  - By definition, $n \geq 1$
  - Examples: n_{air} (1.00) < n_{water} (1.33) < n_{glass} (1.52)

courtesy of Wolfram



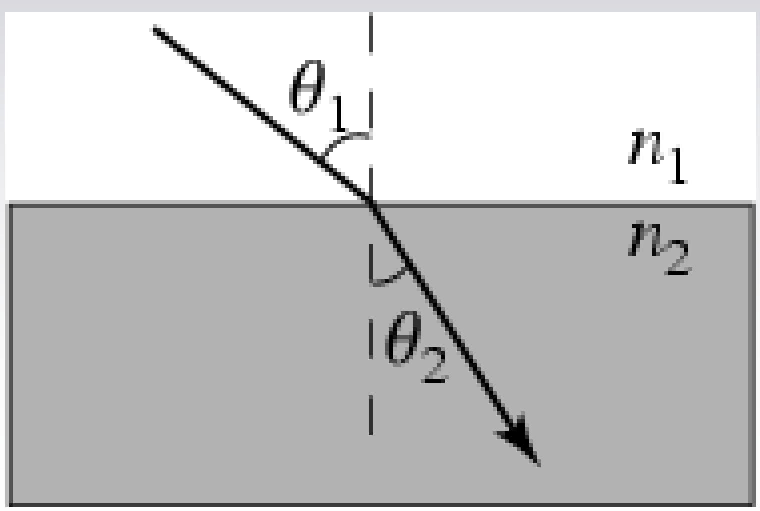
θ_1: Angle of incidence

θ_2: Angle of refraction

Snell's Law

- The relationship between the angle of incidence and the angle of refraction is given by:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$



courtesy of Wolfram

Snell's Law: Implications

- Since θ ≈ sin θ over the range [0, π/2] and the angle of refraction is given by

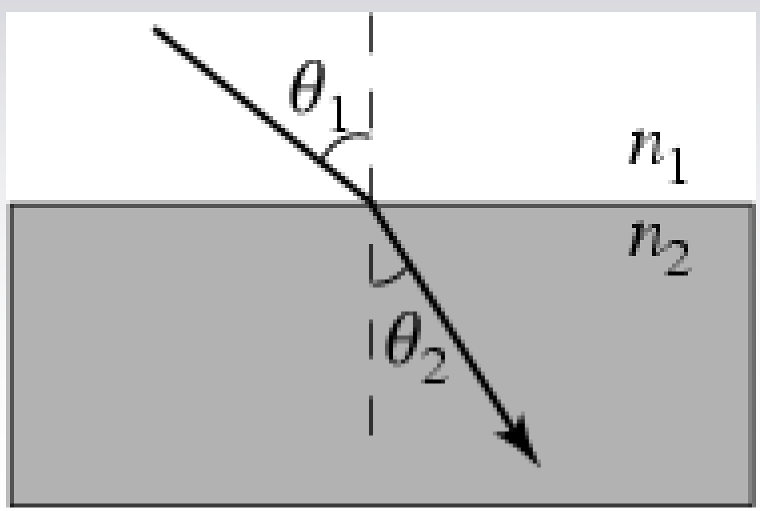
$$\sin\theta_2 = \frac{n_1}{n_2}\sin\theta_1$$

we can infer the following from their IORs:

$n_1 < n_2 \Rightarrow \theta_2 < \theta_1$ and $n_1 > n_2 \Rightarrow \theta_2 > \theta_1$

divergence

convergence



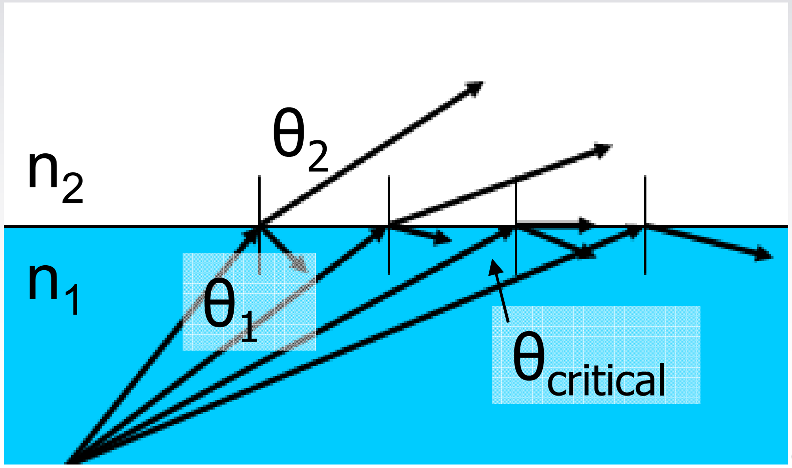
courtesy of Wolfram

So $n_1 < n_2$
in this image
(like air to water)

Refraction: Critical Angle

- Snell's law $n_1 \sin\theta_1 = n_2 \sin\theta_2$ says that $n_1 > n_2$ $\Rightarrow \theta_2 > \theta_1$ (e.g., water to air), but biggest angle θ_2 that exiting ray can be bent is $\pi/2$ (along tangent to the surface)
- Thus, no light escapes—all light is reflected internally—for θ_1 greater than or equal to the **critical angle** of:

$$\theta_{critical} = \sin^{-1}\left(\frac{n_2}{n_1}\right)$$

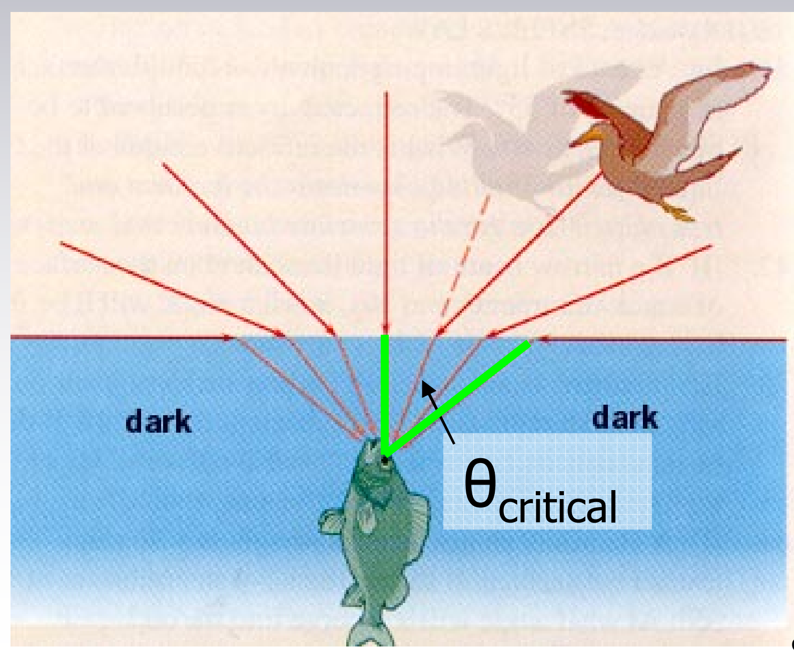


courtesy of G. Kessler

Critical Angle: Example

- Going from water (IOR = 1.33) to air (IOR = 1.00), we have:

$$\theta_{critical} = \sin^{-1}\left(\frac{1.00}{1.33}\right) \approx 48.75^\circ$$



courtesy of J. Alward

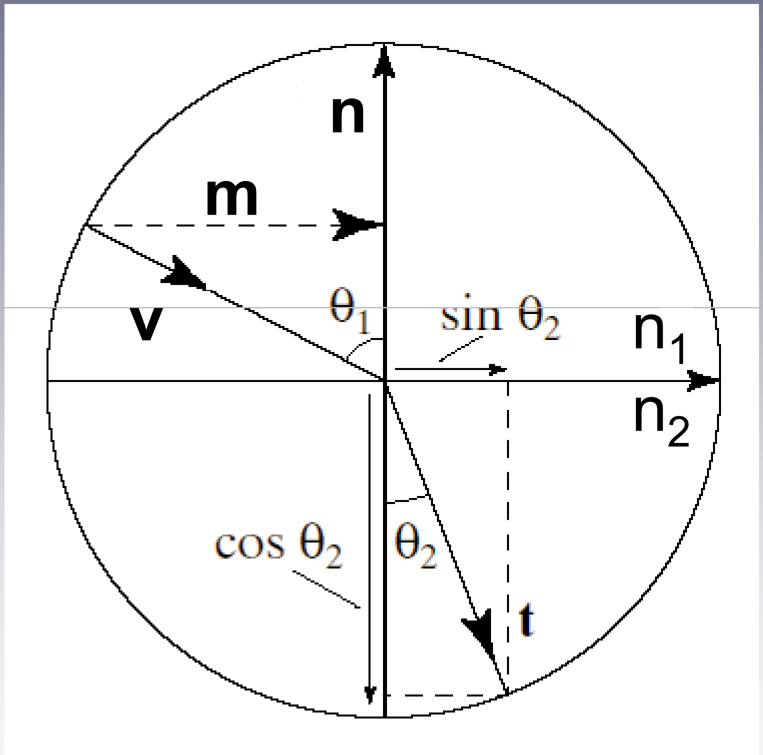
Computing the Transmission Direction t

$$n = \frac{n_1}{n_2}$$

$$c_1 = \cos\theta_1 = -\mathbf{v}\cdot\mathbf{n}$$

$$c_2 = \cos\theta_2 = \sqrt{1 - n^2(1 - c_1^2)}$$

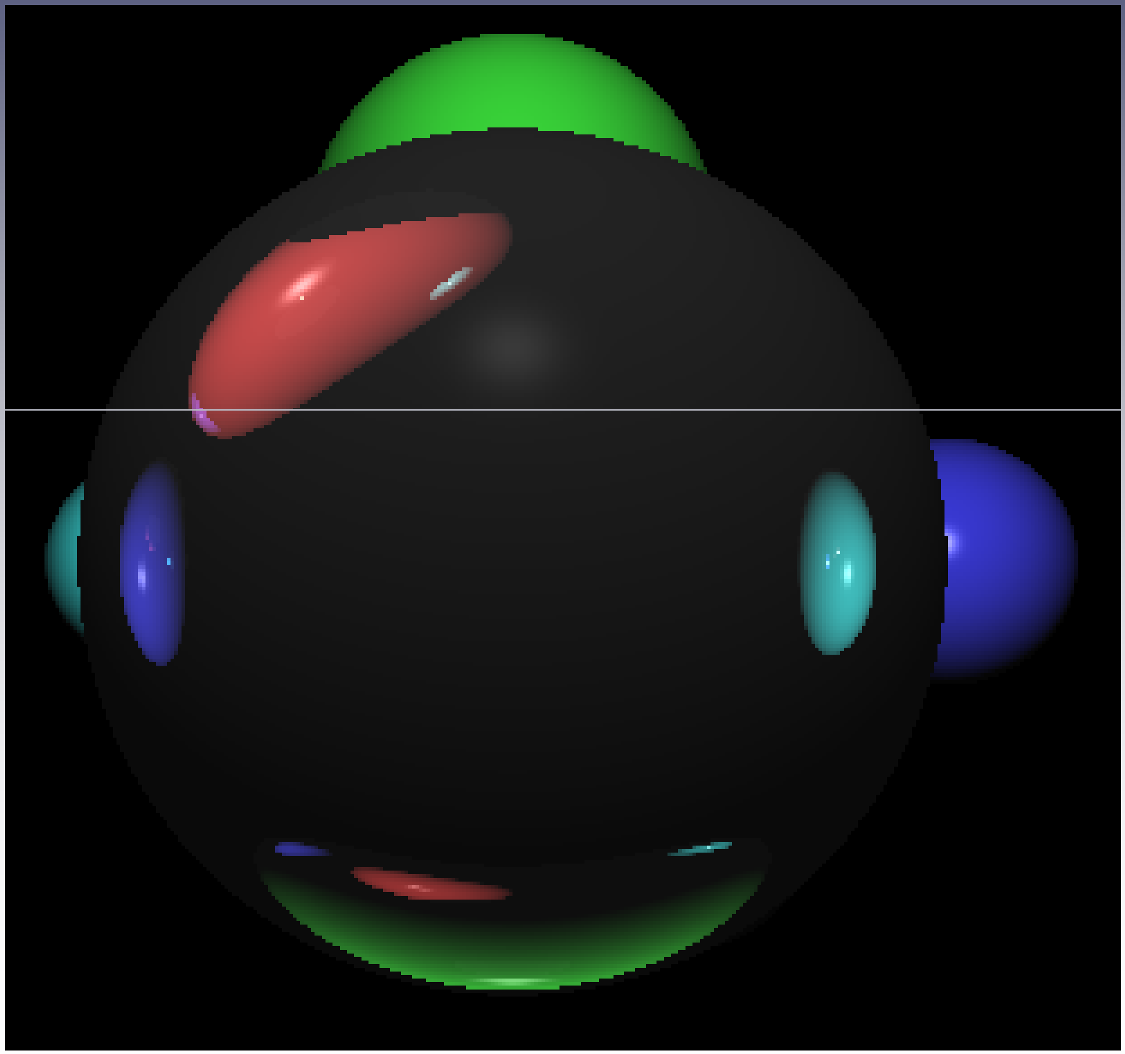
$$\mathbf{t} = n\mathbf{v} + (nc_1 - c_2)\mathbf{n}$$



adapted from Hill

Total internal reflection happens when the term in the square root above isn't positive, which is when $n^2(1 - c_1^2) \geq 1$

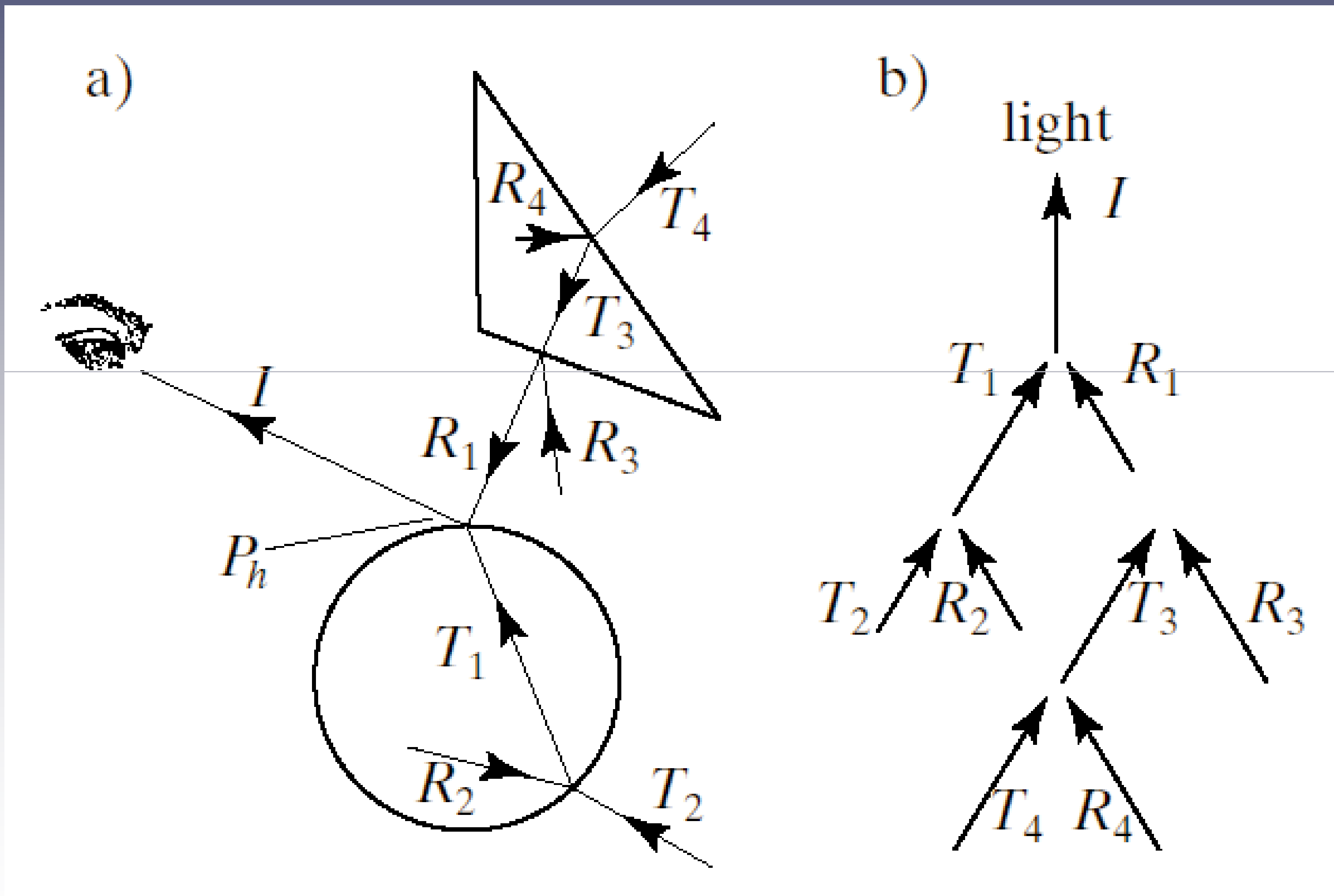
Example: Refraction



Ray Tracing Example (with texture mapping)

courtesy of J. Lee

Ray Tracing: Recursion



from Hill

Anatomy of `shade_ray()`

Local Phong illumination

Figure out reflected/ refracted ray direction and recurse

```
Color3 Scene :: shade(Ray& r)
{
   Get the first hit, and build hitInfo h
   Shape* myObj = (Shape*)h.hitObject; //pointer to the hit object
   Color3 color.set(the emissive component);
   color.add(ambient contribution);
   get the normalized normal vector m at the hit point
   for(each light source)
       add the diffuse and specular components
   // now add the reflected and transmitted components

   if(r.recurseLevel == maxRecursionLevel)
       return color; // don't recurse further

   if(hit object is shiny enough) // add any reflected light
   {
       get reflection direction
       build reflected ray, refl
       refl.recurseLevel = r.recurseLevel + 1;
       color.add(shininess * shade(refl));
   }
   if(hit object is transparent enough)
   {
       get transmited direction
       build transmitted ray, trans
       trans.recurseLevel = r.recurseLevel + 1;
       color.add(transparency * shade(trans));
   }
   return color;
}
```

make sure all channels are clamped to [0, 1] range

HW #3 (due May 4)

- Basic requirements
  - Complete shade_ray_**diffuse**()
  - Complete shade_ray_local(), which adds **specular and shadow** effects
  - Complete **reflection** component of shade_ray_recursive()
  - Add **sphere intersection testing** in intersect_ray_sphere()
  - Scene complexity and creativity
- Grad student requirements
  - Add support for **refraction** in shade_ray_recursive()
  - Add some version of adaptive supersampling, glossy reflection, ambient occlusion, or another advanced **distributed-ray** technique
  - Implement **bounding spheres** around objects to speed intersection calculations