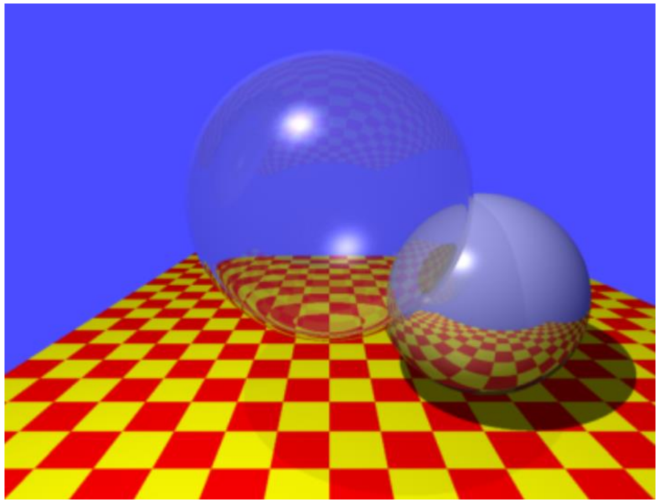
Raytracing
III – Reflection & Transmission

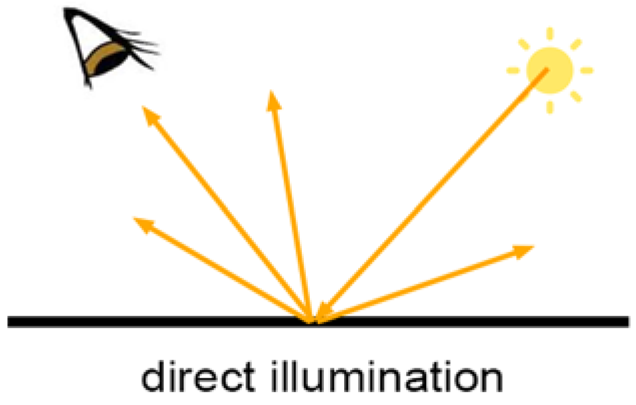


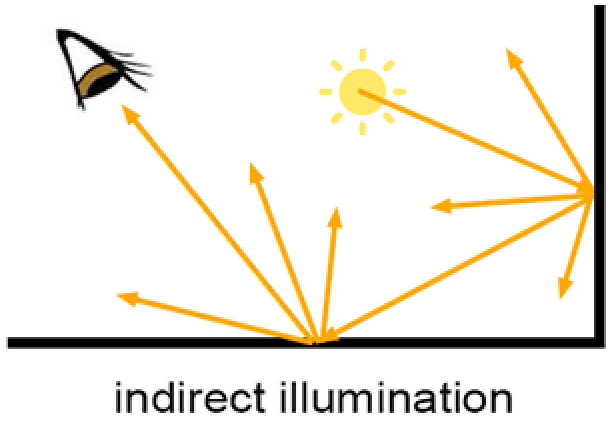
Stefan BORNHOFEN

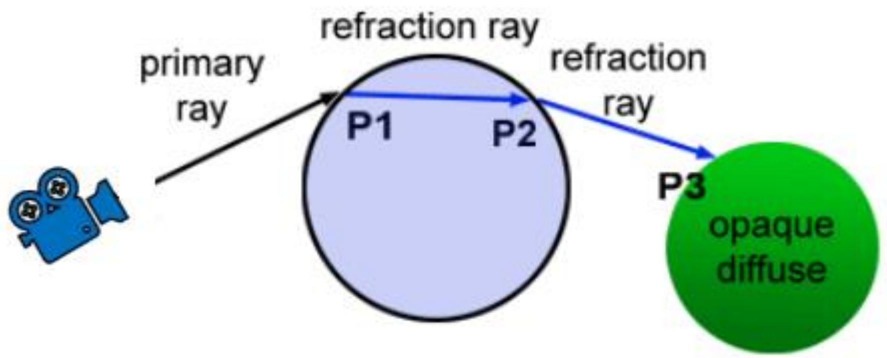
Global Illumination

- **Direct illumination** determines the color and quantity of light that reaches a surface from a light source, but ignores light that may arrive at the surface from other directions.
- **Indirect illumination** comes from light bouncing off a surface and hitting other objects.
- **Global illumination** is the process of computing all light, both direct and indirect, on visible surfaces in a scene.

*Theoretically, the scattering at each intersection point generates an infinite number of new rays that should be traced. In a classical **Whitted-style raytracer**, we only follow the **perfectly transmitted and reflected rays**, and use a shading model to compute direct illumination.*



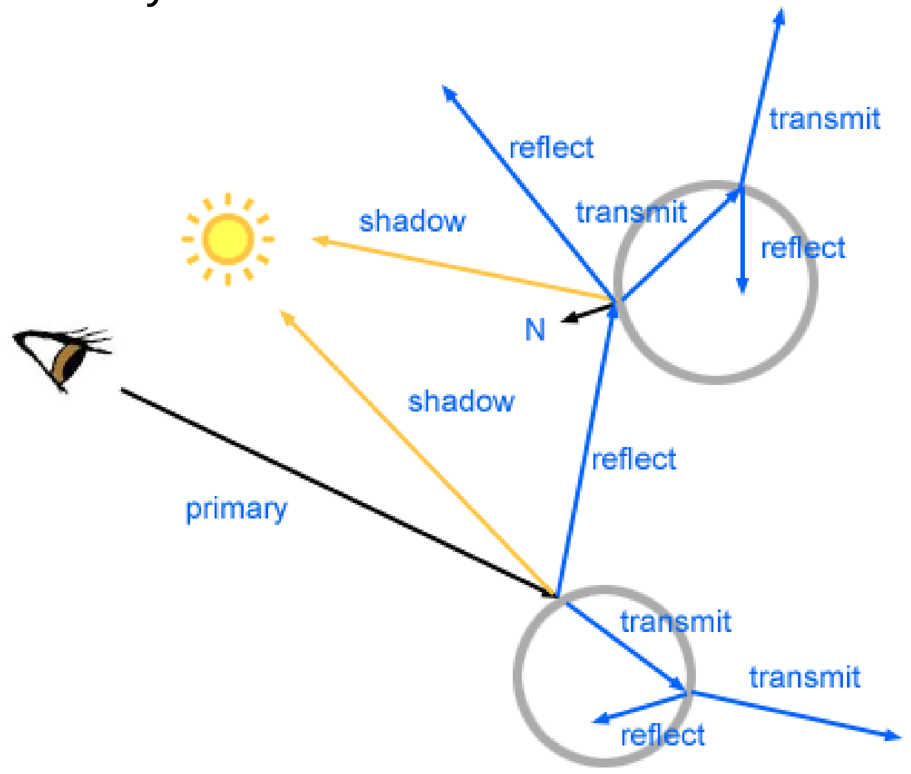




From Raycasting to Raytracing

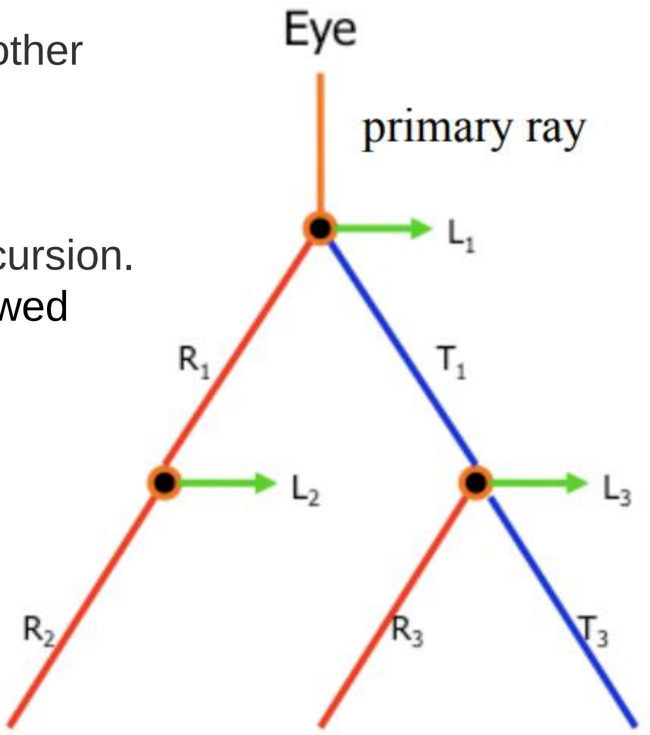
When rays intersect with an object, they may generate more rays. Because applying the raytracing algorithm at one point can involve applying the same algorithm at additional points, raytracing is a **recursive algorithm**.
The number of rays increases exponentially.

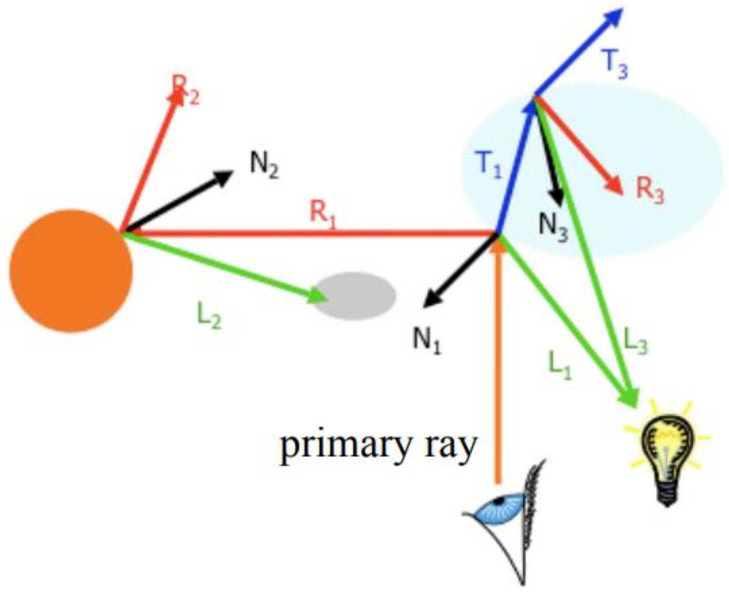
1. Trace a **primary ray** to detect the first visible surface.
2. At the intersection point, trace **secondary rays**:
   - Shadow ray
   - Reflection ray (recursion)
   - Refraction ray (recursion)



Ray Tree

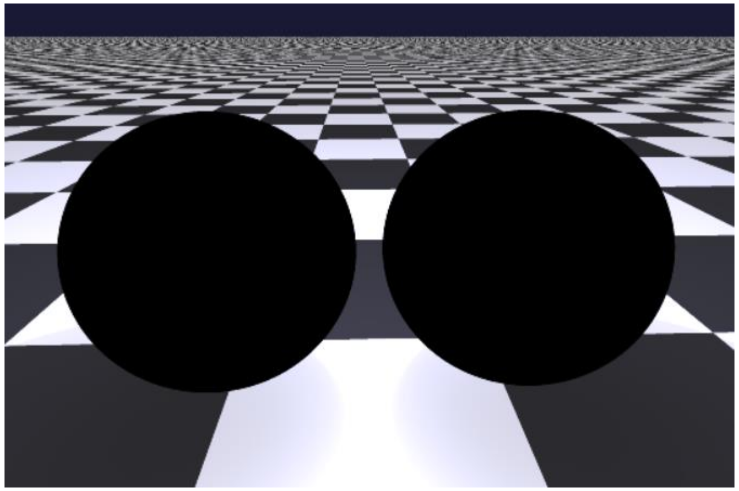
- The secondary rays spawned by the primary ray or other secondary rays can be represented as a tree.
  - Nodes = intersection points
  - Edges = secondary rays
- Each level in the tree corresponds to one level of recursion.
- Raytracers typically put a limit on the number of allowed recursions:
  - **Fixed**: stop if recursion depth > max
  - **Adaptive**: stop if contribution of ray to final pixel color < threshold

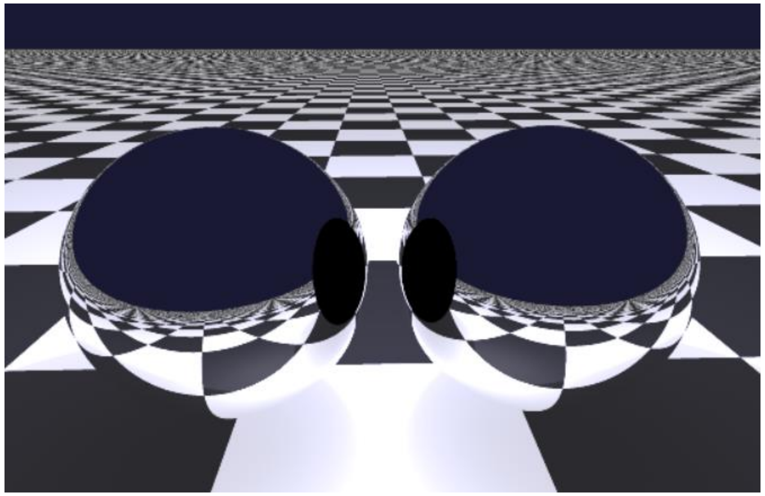


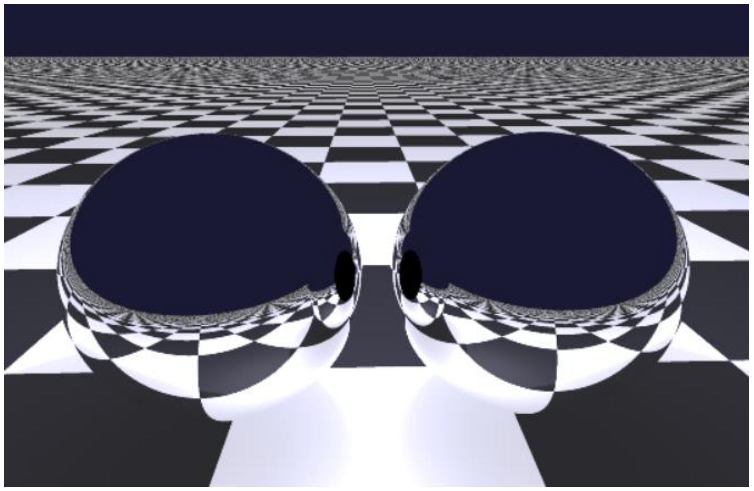


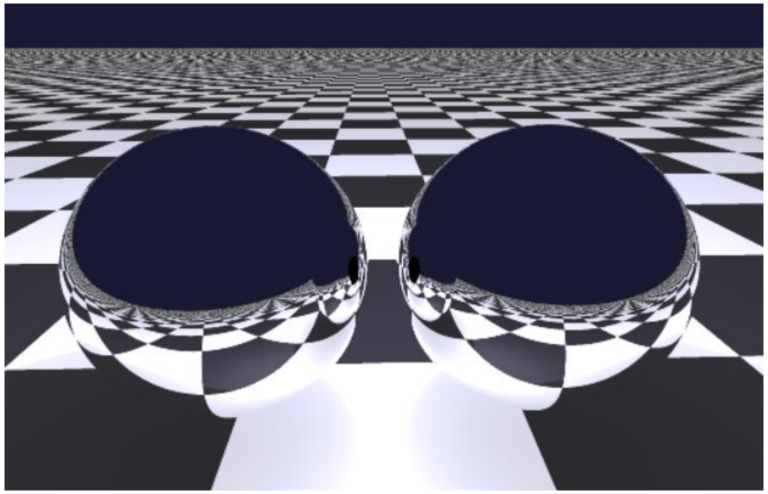
R_i reflected ray
L_i shadow ray
T_i transmitted (refracted) ray
} secondary rays

Recursion Depth


Depth = 0


Depth = 1


Depth = 2


Depth = 3

Recursive Raytracer Algorithm

```
trace (ray)
  compute nearest intersection
  compute shadow ray
  get cReflect and cRefract of the material
  color = (1-cReflect-cRefract) * shading
  if (recursion limit not reached)
    if (cReflect>0)
      color += cReflect * trace (reflected ray)
    if (cRefract>0)
      color += cRefract * trace (refracted ray)
```

GLSL does not support recursions ☹

You need to unroll the recursion to its maximum depth:

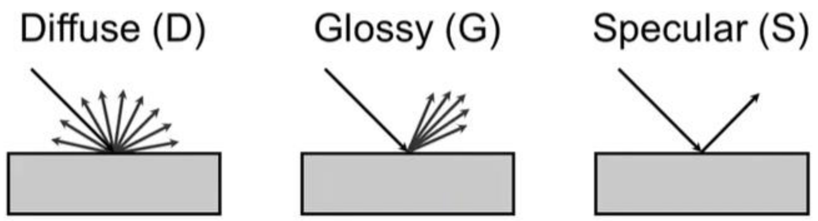
```
trace (ray)
       shade
       trace (reflected ray)
       trace (refracted ray)
```

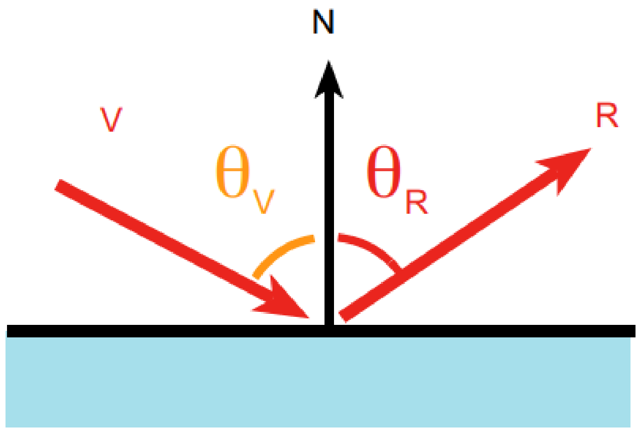
becomes

```
trace (ray)
       shade
       trace1 (reflected ray)
       trace1 (refracted ray)
trace1 (ray)
       shade
       trace2 (reflected ray)
       trace2 (refracted ray)
trace2 (ray) // max depth
       shade
```

Reflection

- We only consider perfect specular reflection.
- Just like the shadow rays, don't for get to add a bias to the origin of the reflected rays or you may encounter surface acne.

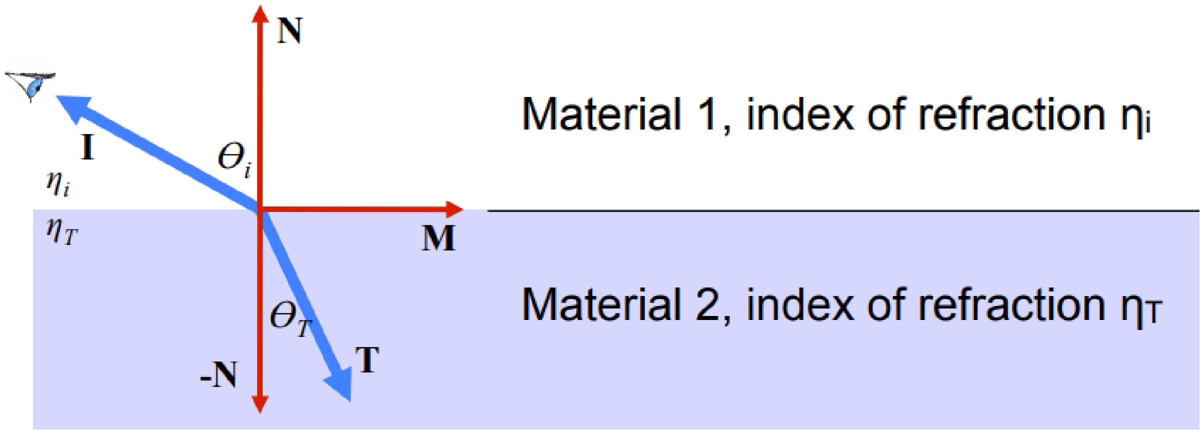




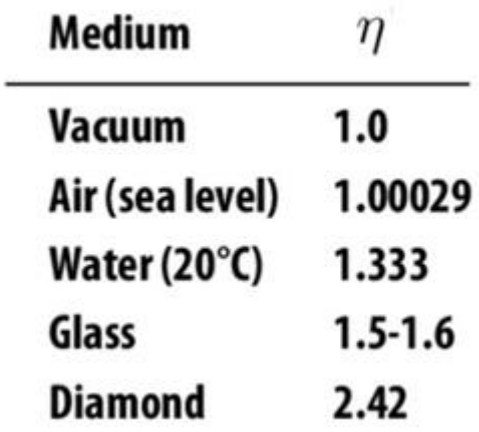
$$\mathbf{R} = \mathbf{V} - 2\,(\mathbf{V} \cdot \mathbf{N})\,\mathbf{N}$$

Refraction

- When light crosses the boundary between two different transparent media it undergoes refraction.
- The ray is bent at the boundary or interface between the two mediums. This change of direction is caused by a change in speed.
- The new direction of the ray depends on the incident angle and the **refractive index** n.



N
I
θ_i
η_i
η_T
M
θ_T
T
-N
Material 1, index of refraction η_i
Material 2, index of refraction η_T

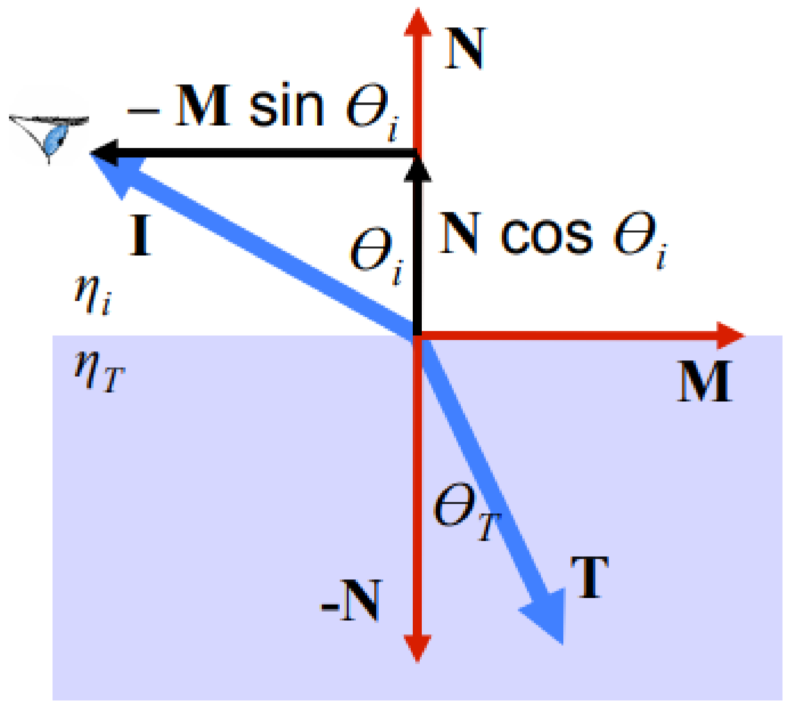


Medium	η
Vacuum	1.0
Air (sea level)	1.00029
Water (20°C)	1.333
Glass	1.5-1.6
Diamond	2.42

Snell's Law

The relationship between the angle of incidence and the angle of refraction is:

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

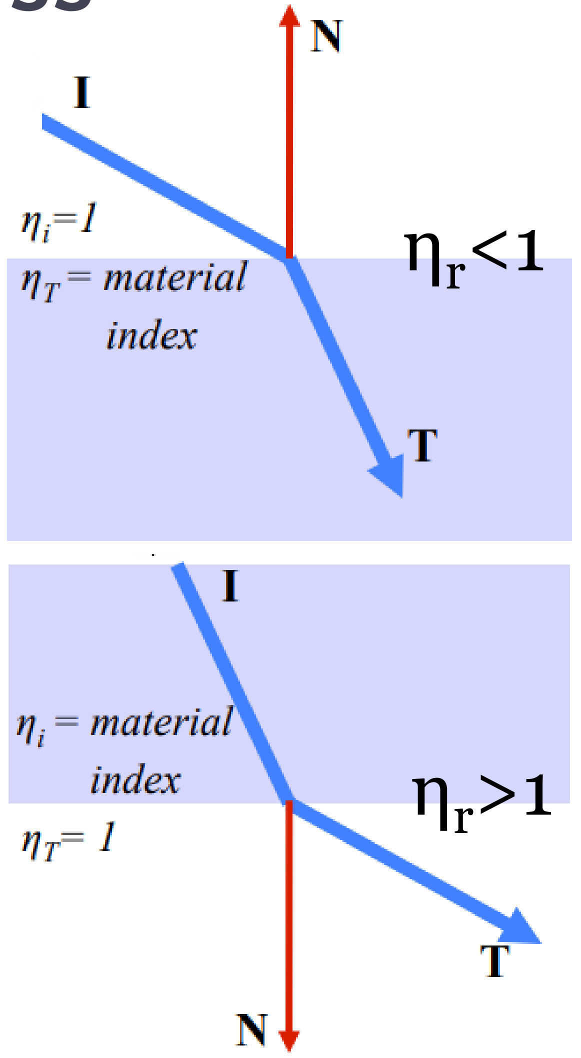


$$\begin{aligned}
\mathbf{I} &= \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i \\
\mathbf{M} &= (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i \\
\mathbf{T} &= -\mathbf{N}\cos\Theta_T + \mathbf{M}\sin\Theta_T \\
&= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\sin\Theta_T / \sin\Theta_i \\
&= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\,\eta_r \\
&= [\,\eta_r\cos\Theta_i - \cos\Theta_T\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{1-\sin^2\Theta_T}\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{1-\eta_r^2\sin^2\Theta_i}\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{1-\eta_r^2(1-\cos^2\Theta_i)}\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r(\mathbf{N}\cdot\mathbf{I}) - \sqrt{1-\eta_r^2(1-(\mathbf{N}\cdot\mathbf{I})^2)}\,]\,\mathbf{N} - \eta_r\mathbf{I}
\end{aligned}$$

Refraction and Sidedness

- When light enters a denser material, i.e. with a higher refractive index, it bends towards the normal line.
- When light enters a material with a lower refractive index, it bends away from the normal line.

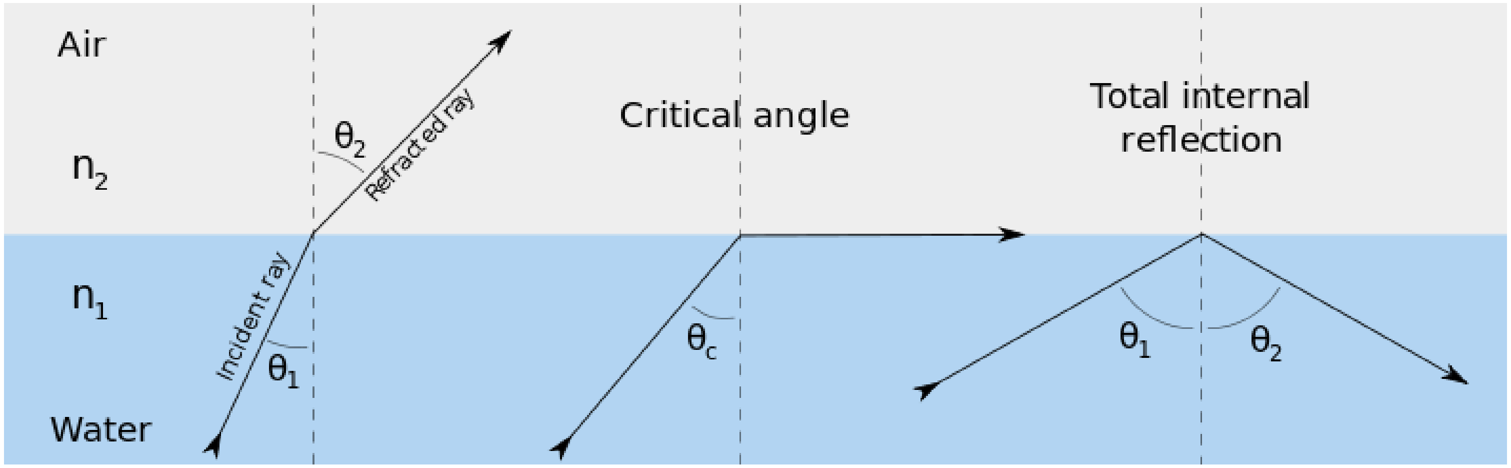
In your raytracer, you need to check if your ray is entering or leaving the transmissive object. Look at the orientation of the surface normal. If you leave the object, return the normal and invert the refraction index.



Total Internal Reflection

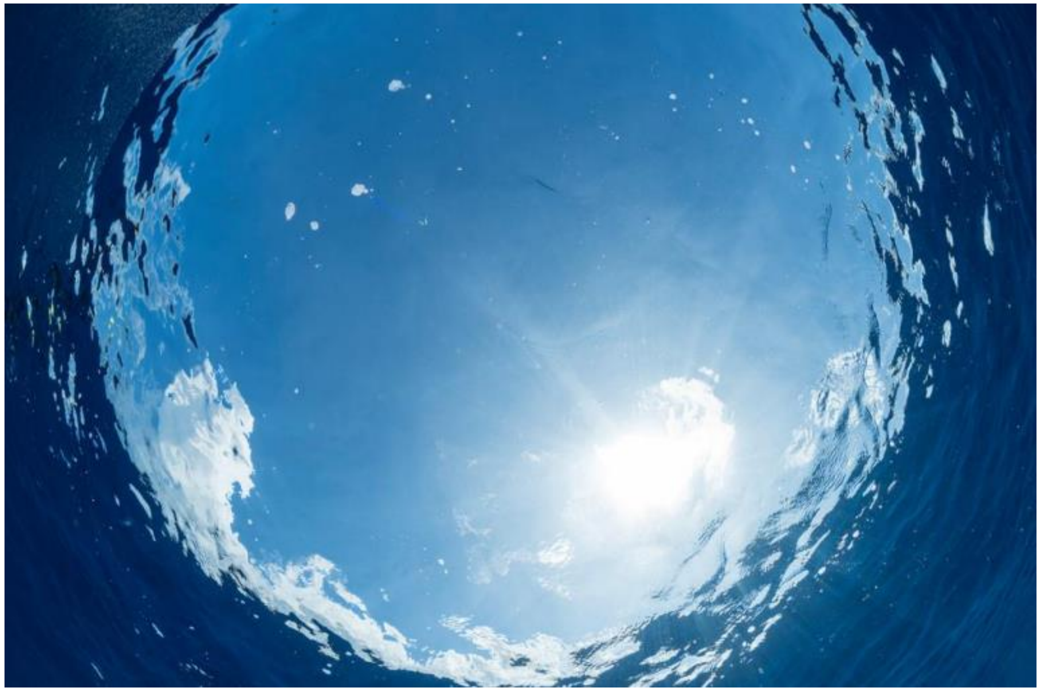
- While exiting a denser material, Snell's law has no solutions for near-grazing angles (n_r>1 and the refracted ray T has an imaginary square root).
- In this case, instead of refracting out of the material, the light reflects entirely back into the material.
- The **critical angle** is the angle of incidence for which the angle of refraction is exactly 90 degrees.

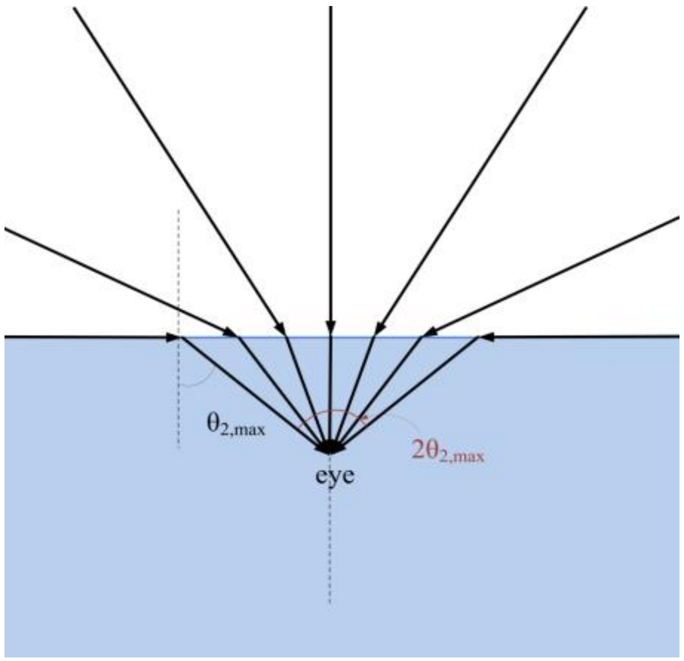
The GLSL built-in function "refract" returns a zero-length vector in the case of total internal reflection. You need to manually check for this case.



Snell’s window

Snell's window is a phenomenon by which an underwater viewer sees everything above the surface through a cone of light. The area outside Snell's window shows a reflection of the underwater world by total internal reflection.



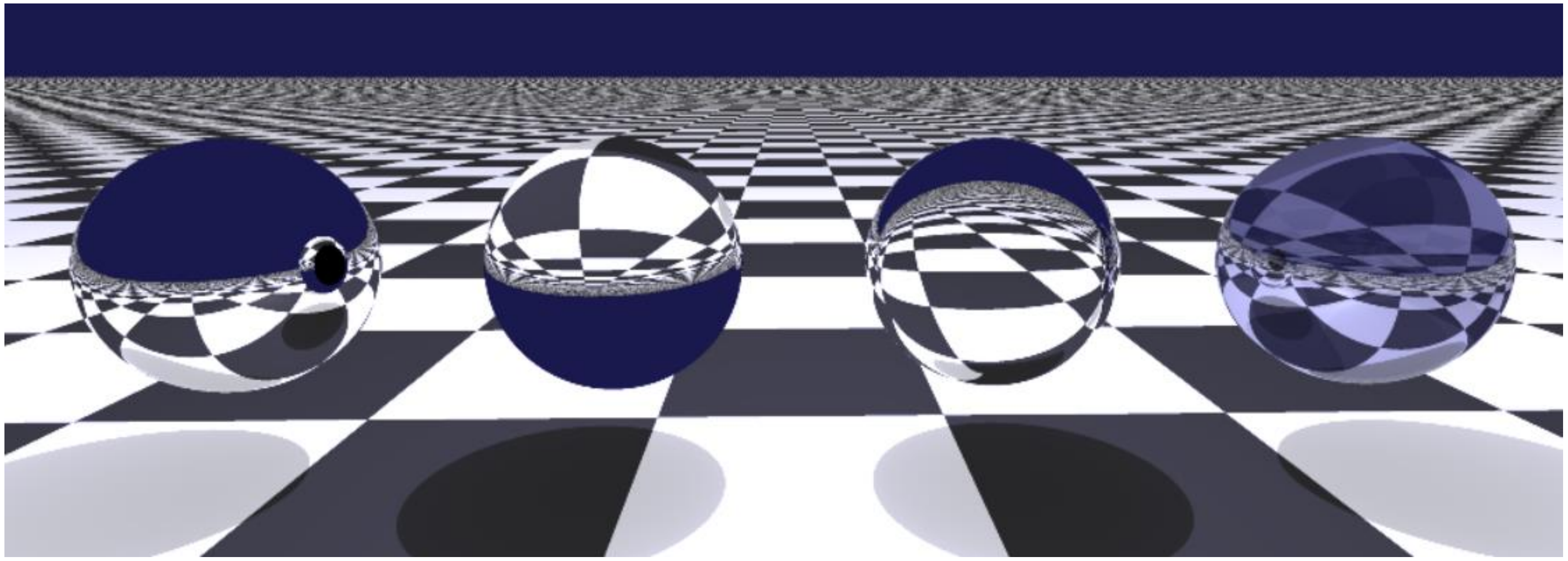


Your work

- Find a way to texture the ground as a checkerboard (so that we can better assess the light effects)
- Implement reflective and transmissive materials.
- The user can control the recursion depth.
- Raytrace the following spheres
  1) 100% reflective
  2) 100% refractive (solid)
  3) 100% refractive (hollow)
  4) 50% reflective and 50% refractive

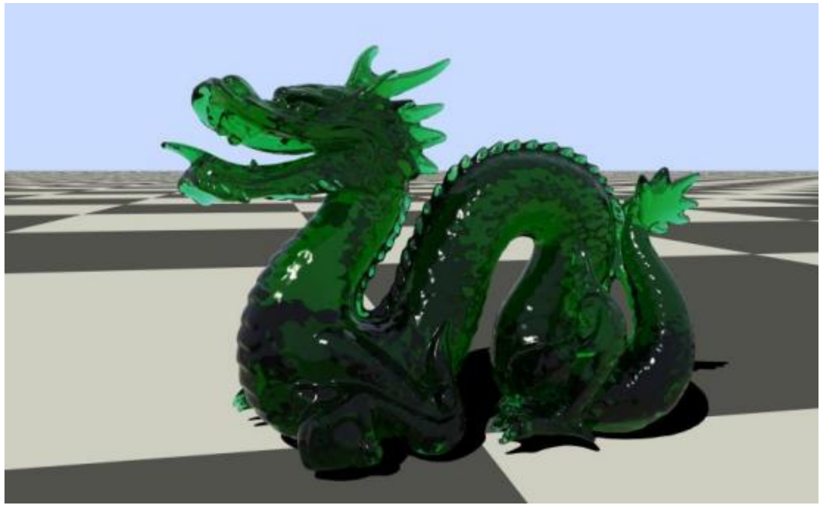
Material

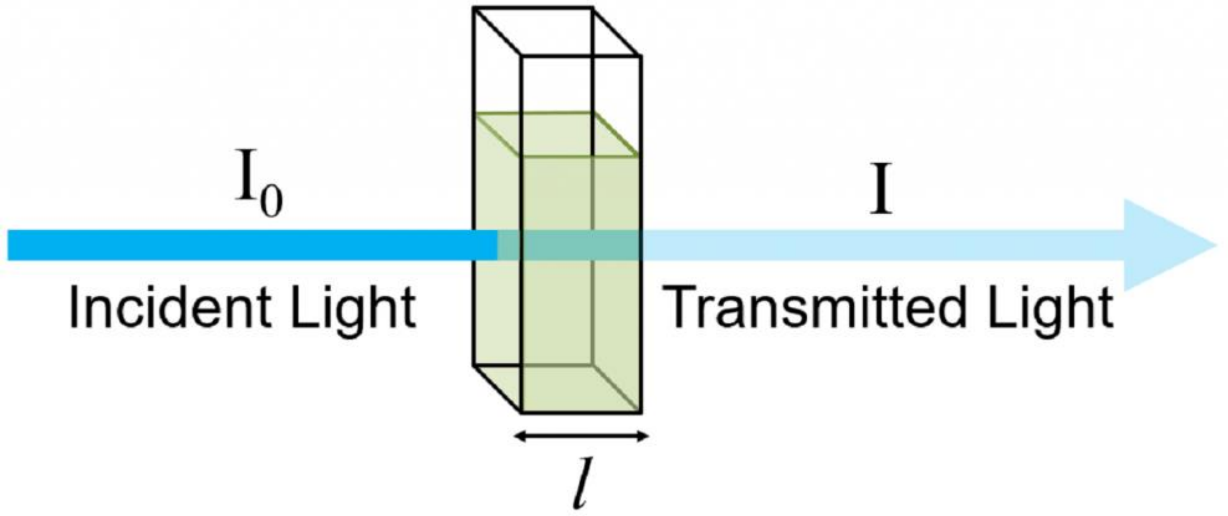
```
Ambient color ka
Diffuse color kd
Specular color ks
Shininess h
Reflection coef cReflect
Refraction coef cRefract
Refraction index eta
```



Absorption

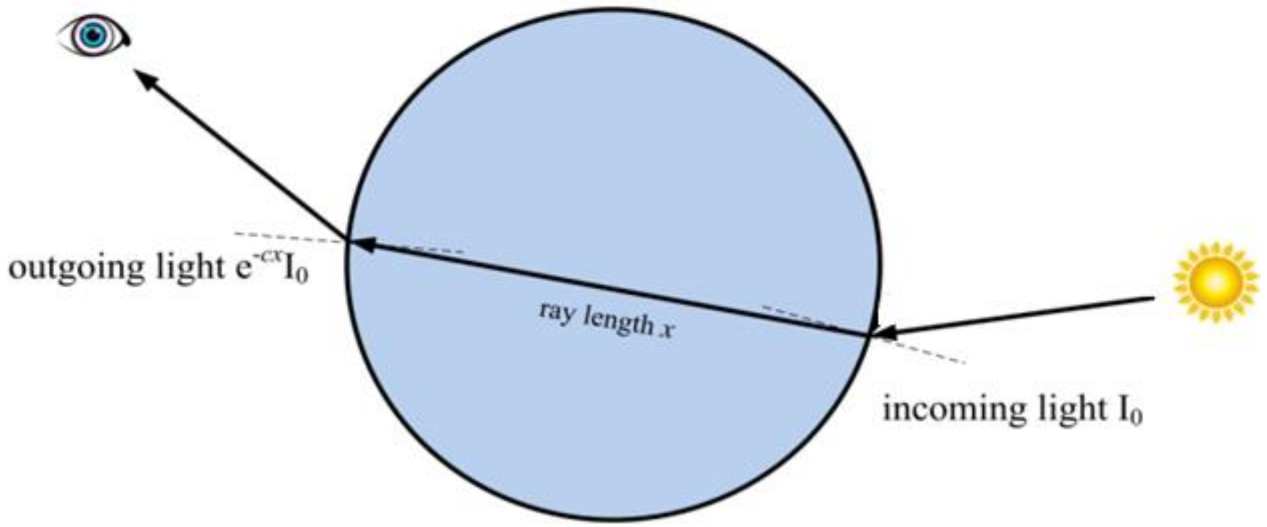
- Light is absorbed and scattered as it travels through material
- This attenuates the amount of light traveling along a straight line
- The amount of attenuation depends on the distance traveled through the material
- Different colors (actually, different wavelengths) are attenuated at different rates





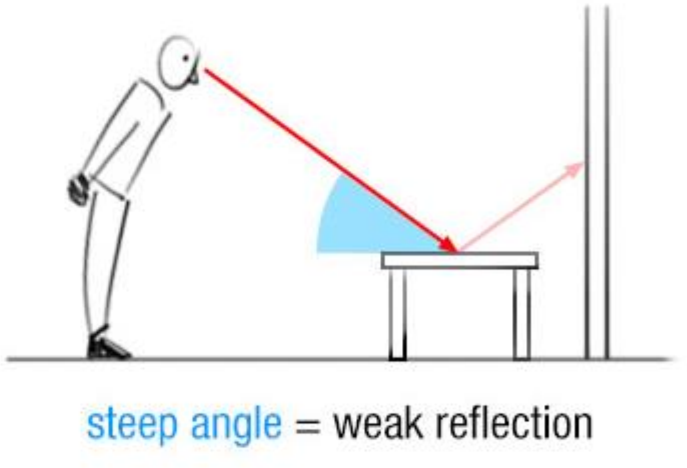
Beer’s Law

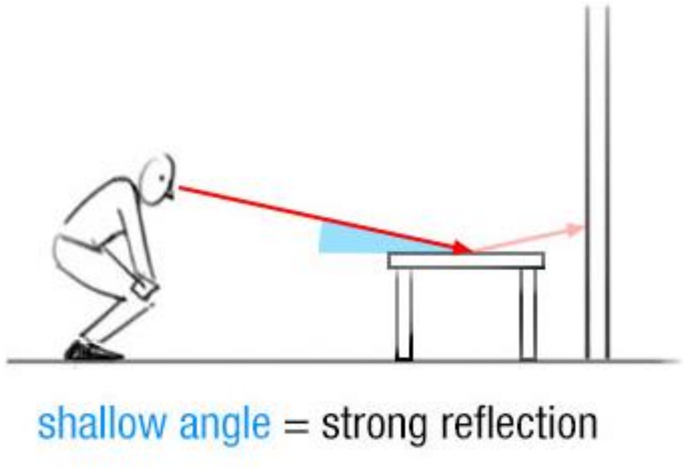
- For homogeneous media, absorption can be approximated by Beer’s Law
- Note that this law handles transparent materials with **absorbing** particles and not materials that contain **scattering** particles
- The light intensity I that remains after crossing an absorbing material is: $I = I_0\, e^{(-cx)}$ where
  - I_0 is the original amount of light
  - c is the absorption coefficient (cR, cG, cB)
  - x the traveling distance



Fresnel Effect

- Real transparent materials such as glass or water are both refractive and reflective.
- The reflectiveness is not constant, but it increases as the viewing angle goes from perpendicular (coincident with the normal) to parallel (orthogonal to the normal).





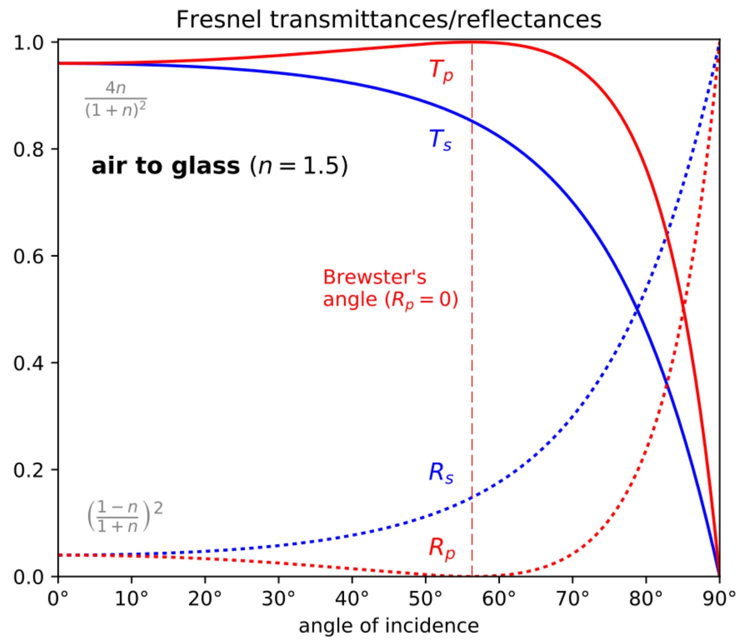
Fresnel Equations

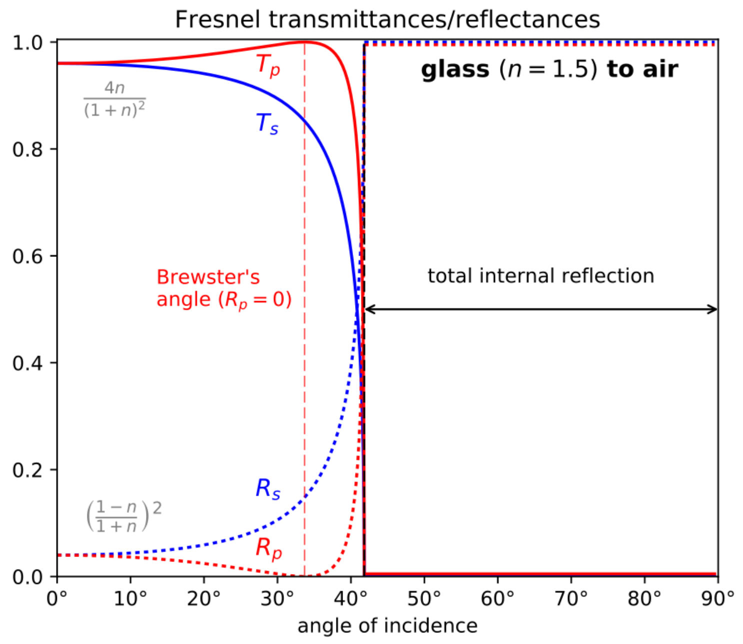
$$R_s = \left| \frac{\eta_1 \cos\theta_i - \eta_2 \sqrt{1 - \left(\frac{\eta_1}{\eta_2}\sin\theta_i\right)^2}}{\eta_1 \cos\theta_i + \eta_2 \sqrt{1 - \left(\frac{\eta_1}{\eta_2}\sin\theta_i\right)^2}} \right|^2$$

$$R_p = \left| \frac{\eta_1 \sqrt{1 - \left(\frac{\eta_1}{\eta_2}\sin\theta_i\right)^2} - \eta_2 \cos\theta_i}{\eta_1 \sqrt{1 - \left(\frac{\eta_1}{\eta_2}\sin\theta_i\right)^2} + \eta_2 \cos\theta_i} \right|^2$$

The s and p subscripts denote the polarization of light: s is perpendicular to the propagation direction and p is parallel.

Most ray tracers ignore light polarization by simply averaging the two equations to arrive at a final reflection magnitude R = (R_s + R_p) / 2.





Schlick Approximation

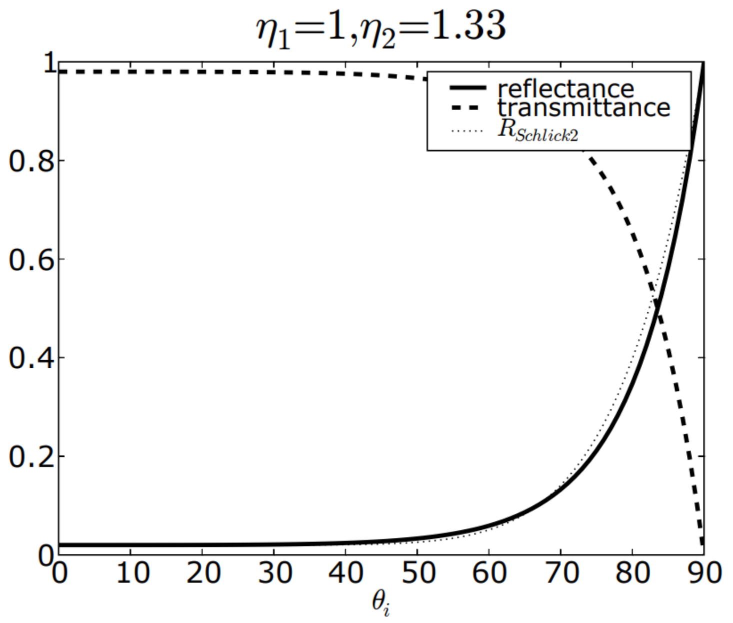
- The Fresnel equations are exact, but complicated to evaluate.
- In computer graphics, Schlick's approximation (1994) is often used instead:

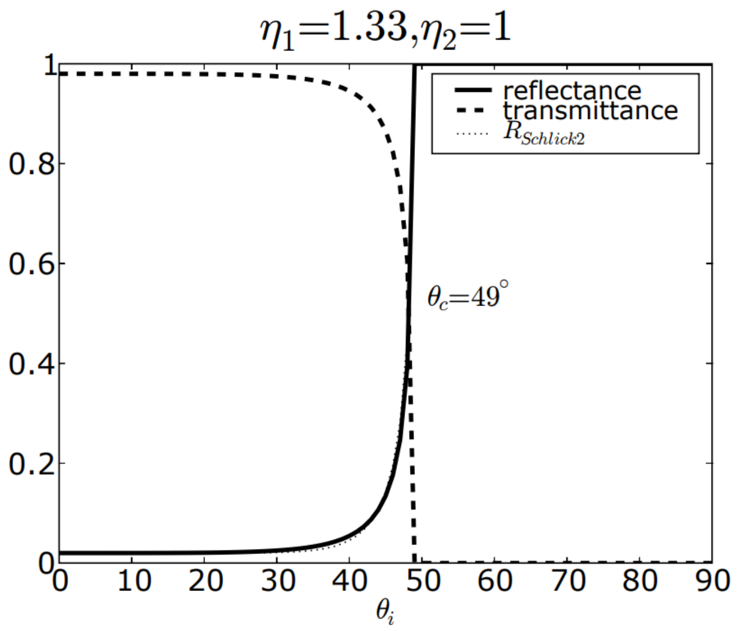
$$R_0 = \left(\frac{\eta_1 - \eta_2}{\eta_1 + \eta_2}\right)^2$$

$$R_{Schlick}(\theta_i) = R_0 + (1 - R_0)(1 - \cos\theta_i)^5$$

However, this approximation fails to model the reflectance when $\eta_1 > \eta_2$. In that case, the problem is fixed by using $\cos\theta_t$ instead of $\cos\theta_i$. In case of TIR, the function returns 1:

$$R_{Schlick2}(\theta_i) = \begin{cases} R_0 + (1 - R_0)(1 - \cos\theta_i)^5 & \Leftrightarrow \eta_1 \leq \eta_2 \\ R_0 + (1 - R_0)(1 - \cos\theta_t)^5 & \Leftrightarrow \eta_1 > \eta_2 \wedge \neg\mathrm{TIR} \\ 1 & \Leftrightarrow \eta_1 > \eta_2 \wedge \mathrm{TIR} \end{cases}$$



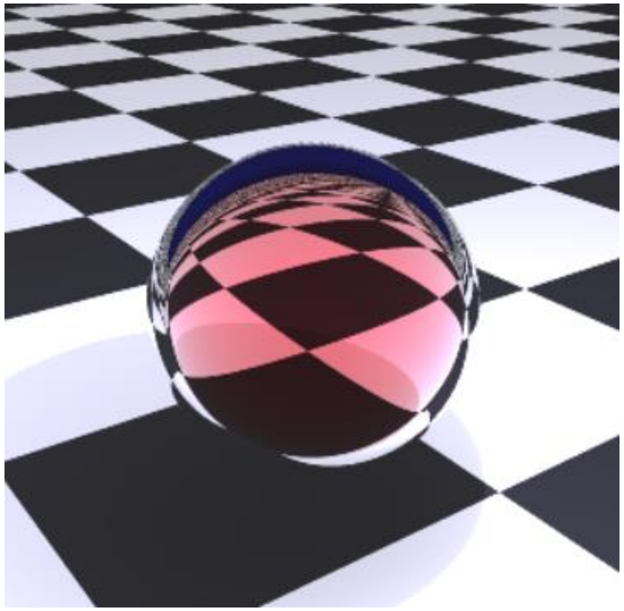


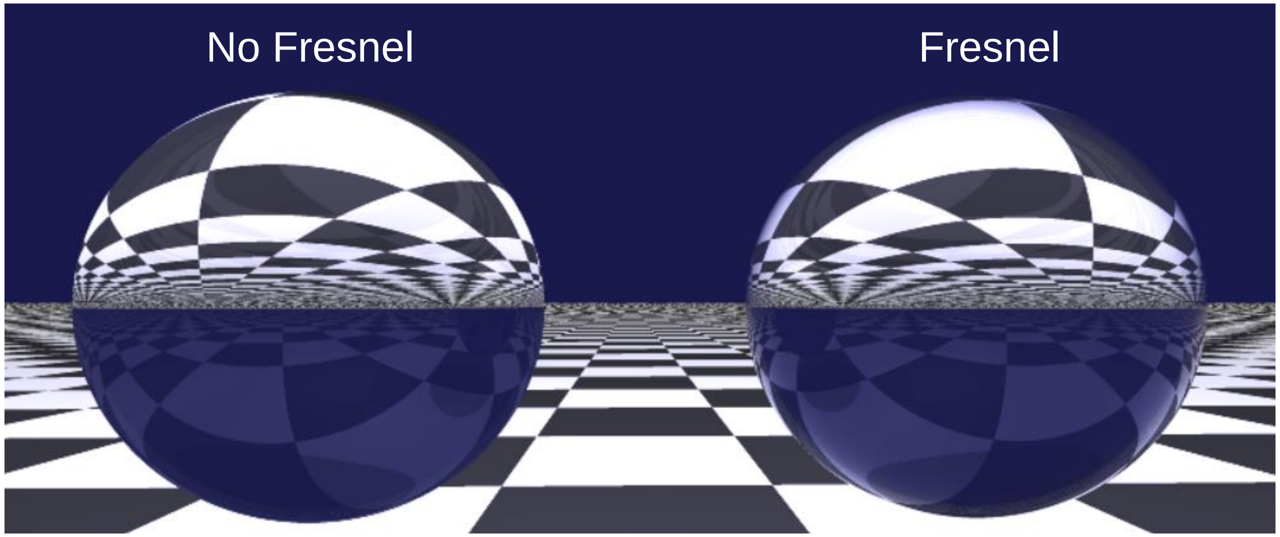
Raytracer with Fresnel and Beer

```
trace (ray)
  compute nearest intersection
  compute shadow ray
  get cReflect and cRefract of the material
  color = (1-cReflect-cRefract) * shading
  if (recursion limit not reached)
    if (applyFresnel) // adjust cReflect and cRefract
      R = Fresnel_factor
      cReflect += R * cRefract
      cRefract -= R * cRefract
    if (cReflect>0)
      color += cReflect * trace (reflected ray)
    if (cRefract>0)
      color += cRefract * trace (refracted ray)
  if (applyBeer and (ray is leaving the material))
    color *= Beer_absorption
```

Your work

- Fill an absorbing substance into the hollow sphere
- Add the Fresnel effect to your transmissive materials using Schlick's approximation
- Compare Fresnel material to transmissive material having an equivalent constant reflectance R_0.




No Fresnel
Fresnel