



**CS 563 Advanced Topics in
Computer Graphics**
Chapter 27 – Simple Transparency

by Wadii Bellamine



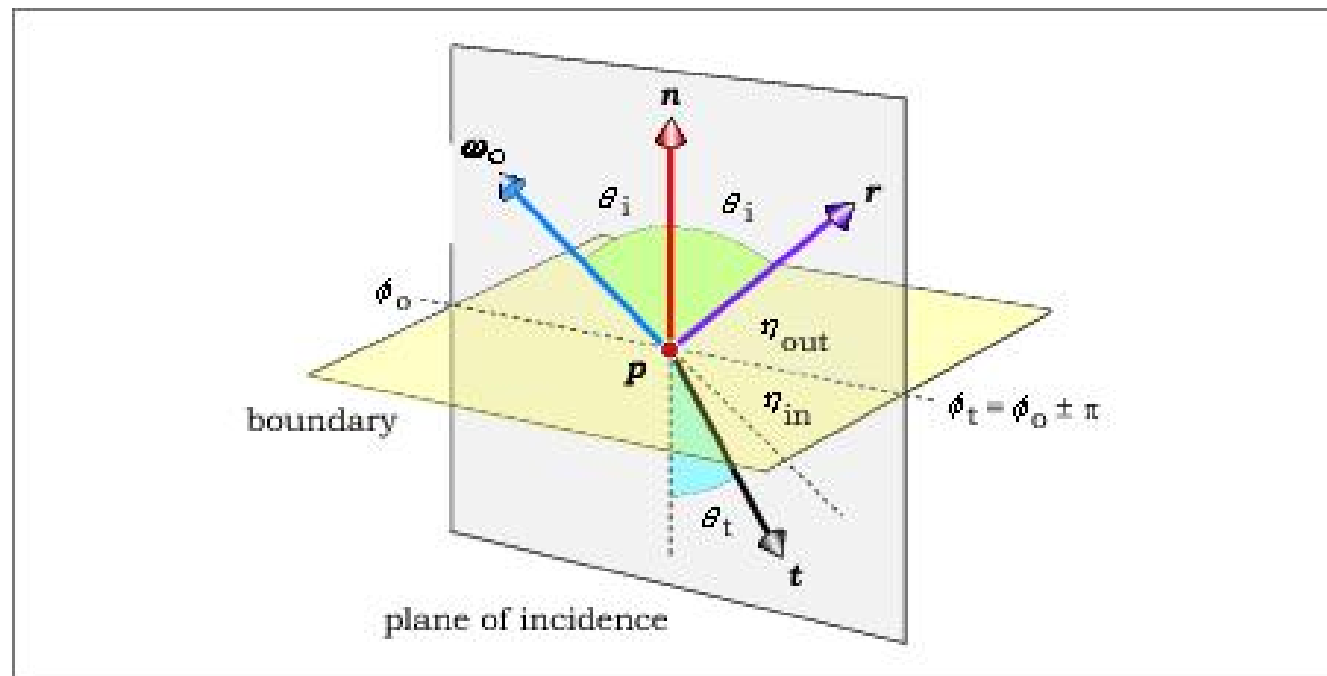
Presentation Outline:

- Definition and physics of refraction
- Total Internal Reflection
- Illumination model
- Implementation
- Analysis of transparency using transparent spheres

(Disclaimer: all unreferenced figures have been shamelessly stolen from the textbook).

Transparent objects

- Allow light to pass through them.
- Referred to as dielectrics:
 - Gases (air), glass, clear plastics, etc.
- When a ray hits the boundary between two dielectrics, the outcome is a reflected ray, and a transmitted ray*.
- The transmitted ray traverses the object, and undergoes refraction.



Refraction

- Speed of light in a perfect vacuum:

$$c = 2.99 \times 10^8 \text{ m/s.}$$

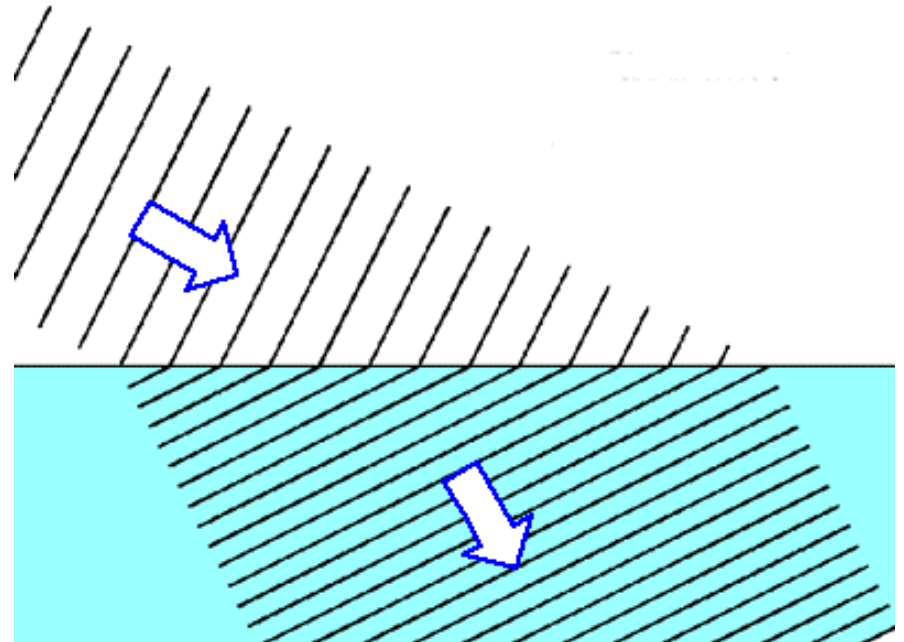
- Speed of light is slower when traveling through other media because of interaction with the medium's molecules.

- The denser the medium, the slower the speed. Speed of light in non-vacuum = v .

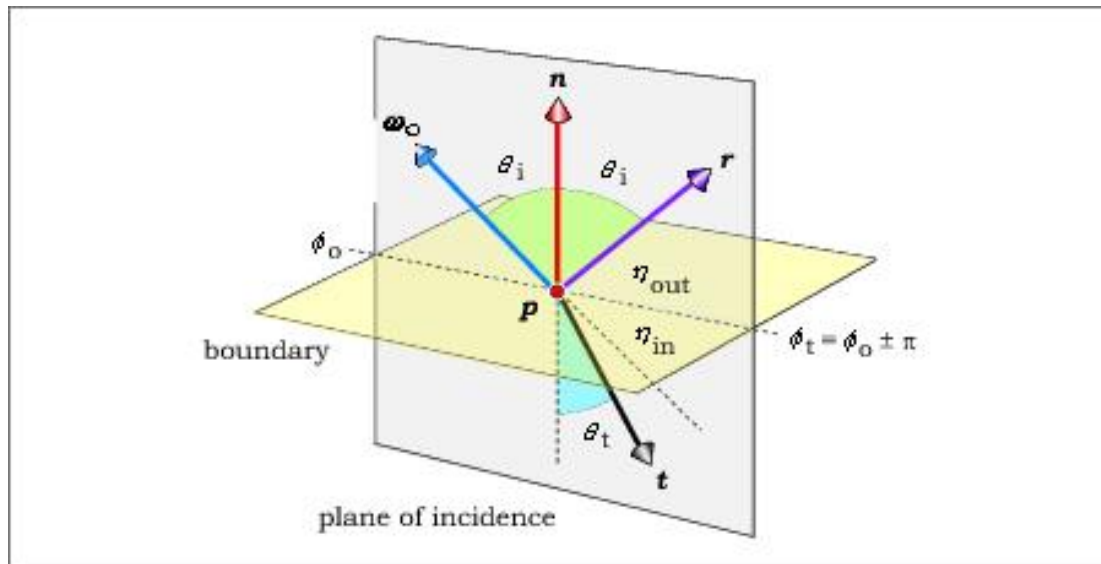
- Absolute index of refraction:

$$n = c/v$$

- Vacuum: $n = 1.0$
- Air: $n = 1.0003$
- Water: $n = 1.33$
- Diamond: $n = 2.42$



Refraction



ω_o : incident ray

r : reflected ray

t : transmitted ray

\angle_i : angle of incidence

\angle_t : angle of refraction

- Objective: Calculate the transmitted ray t
- Snell's law (aka law of refraction):

$$\frac{\sin \theta_i}{\sin \theta_t} = \frac{\eta_{in}}{\eta_{out}} = \eta$$

$$t = \frac{1}{\eta} \omega_o - \left(\cos \theta_t - \frac{1}{\eta} \cos \theta_i \right) n$$

$$\cos \theta_t = \left[1 - \frac{1}{\eta^2} (1 - \cos^2 \theta_i) \right]^{1/2}$$

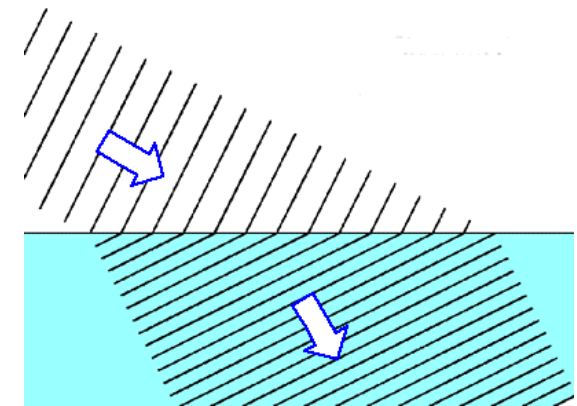
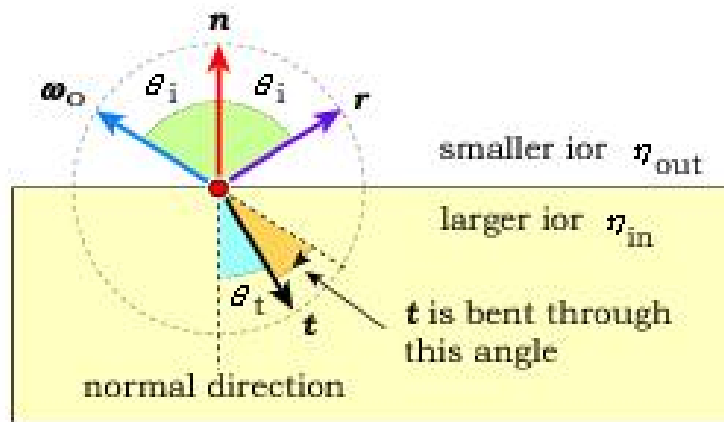
$$\cos \theta_i = n \cdot \omega_o$$

Refraction

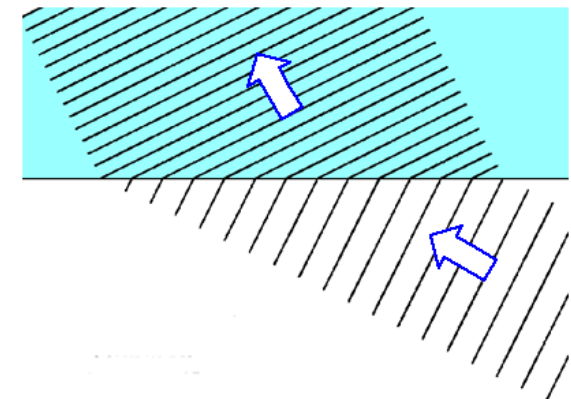
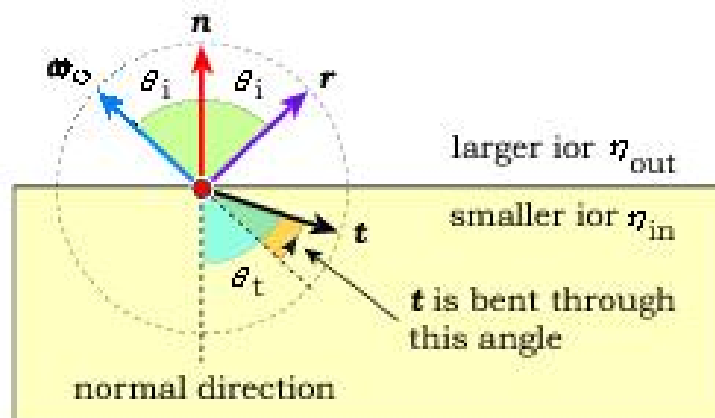
- t is bent differently for $n > 1$ and $n < 1$.

$$n = \frac{n_{in}}{n_{out}}$$

t bends towards the normal for $n > 1$



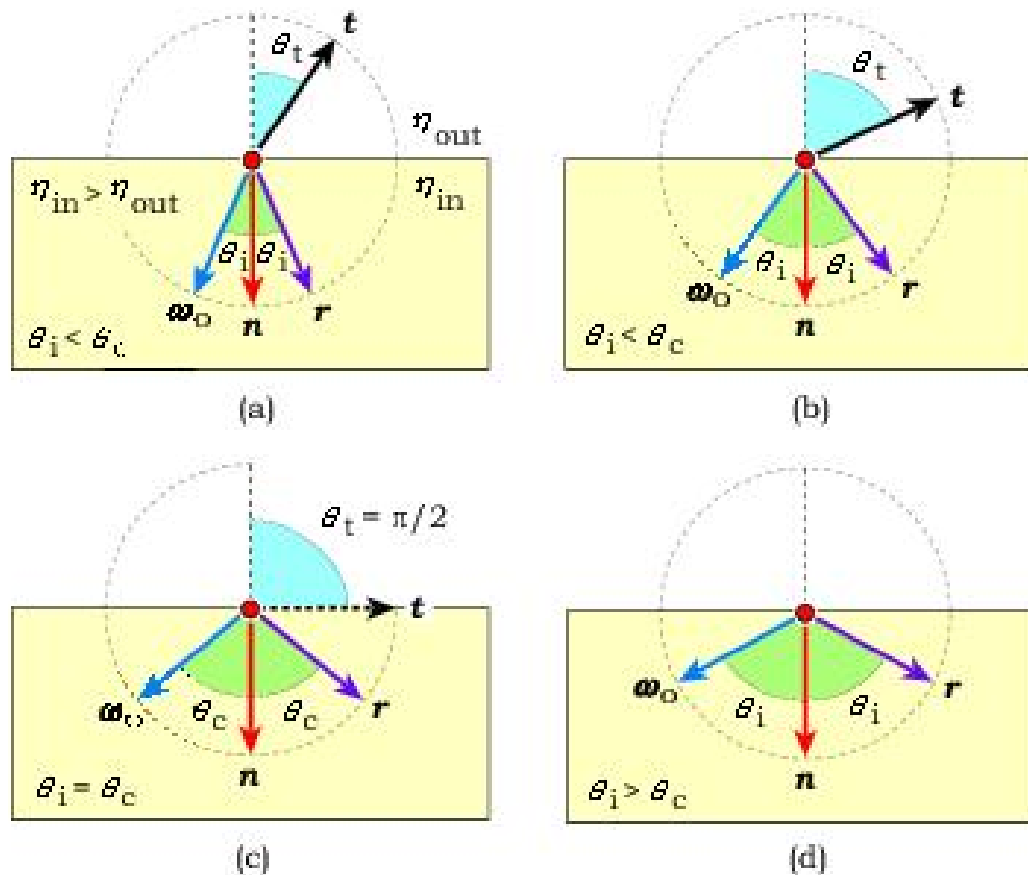
t bends away from the normal for $n < 1$



(Mentally flip arrows)

Total Internal Reflection

- Total internal reflection may occur when the incident ray is being cast from inside the medium with the larger ior.
- This is because the transmitted ray bends away from the normal (since $n < 1$).



■ **Note:** normal is flipped when casting from inside, therefore:

$$\eta = \frac{n_{out}}{n_{in}}$$

■ When θ_c (critical angle) is reached, t has no energy, and r has all the energy.

$$\cos \theta_t = \left[1 - \frac{1}{n^2} (1 - \cos^2 \theta_i) \right]^{1/2}$$

■ **Condition:** if term inside sqrt is negative, we have total internal reflection.

Illumination Model

- Exitant radiance at a point \mathbf{p} :

$$L_r(\mathbf{p}, \omega_o) = L_{\text{direct}}(\mathbf{p}, \omega_o) + L_{\text{indirect}}(\mathbf{p}, \omega_o)$$

$$L_{\text{indirect}}(\mathbf{p}, \omega_o) = L_r(\mathbf{p}, \omega_o) + L_t(\mathbf{p}, \omega_o)$$

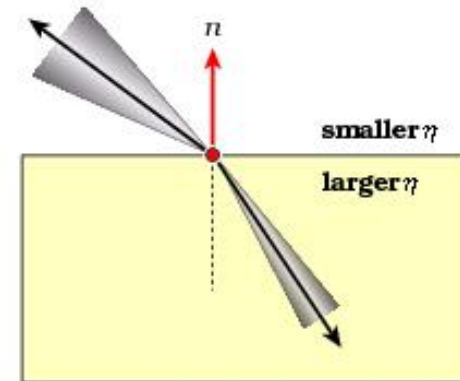
$$L_t(\mathbf{p}, \omega_o) = \int_{2\pi^-} f_{t,s}(\mathbf{p}, \omega_i, \omega_o) L_o(\mathbf{r}_c(\mathbf{p}, \omega_i), -\omega_i) |\cos \theta_i| d\omega_i$$

$f_{t,s}(\mathbf{p}, \omega_i, \omega_o)$ is the *bidirectional transmission distribution function* (BTDF)

Illumination Model

- As a ray crosses a boundary, its radiance changes due to refraction:

- When going from smaller η to larger η , cone angle decreases \therefore radiance increases
- Going from larger η to smaller η , radiance decreases.



$$L_t = k_t \left(\frac{\eta_t^2}{\eta_i^2} \right) L_i$$

- L_t : transmitted radiance
- L_i : incident radiance
- k_t : transmission coefficient
 - $k_t \in [0,1]$
 - $k_r + k_t = 1$
- \square_t : ior of transmitted radiance side
- \square_i : ior of incident radiance side

Illumination Model

- The BTDF can be written as follows:

$$f_{t,s}(\mathbf{p}, \omega_i, \omega_o) = k_t \left(\frac{\eta_t^2}{\eta_i^2} \right) \frac{\delta(\omega_i - \mathbf{t}(\mathbf{n}, \omega_o))}{|\cos \theta_i|}$$

- Using:

$$L_t(\mathbf{p}, \omega_o) = \int_{2\pi^-} f_{t,s}(\mathbf{p}, \omega_i, \omega_o) L_o(r_c(\mathbf{p}, \omega_i), -\omega_i) |\cos \theta_i| d\omega_i$$

$$L_t = k_t \left(\frac{\eta_t^2}{\eta_i^2} \right) L_i$$

We get:

$$L_t(\mathbf{p}, \omega_o) = k_t \left(\frac{\eta_t^2}{\eta_i^2} \right) L_i(\mathbf{p}, \omega_i)$$

Where:

$$L_i(\mathbf{p}, \omega_i) = L_o(r_c(\mathbf{p}, \omega_i), -\omega_i)$$

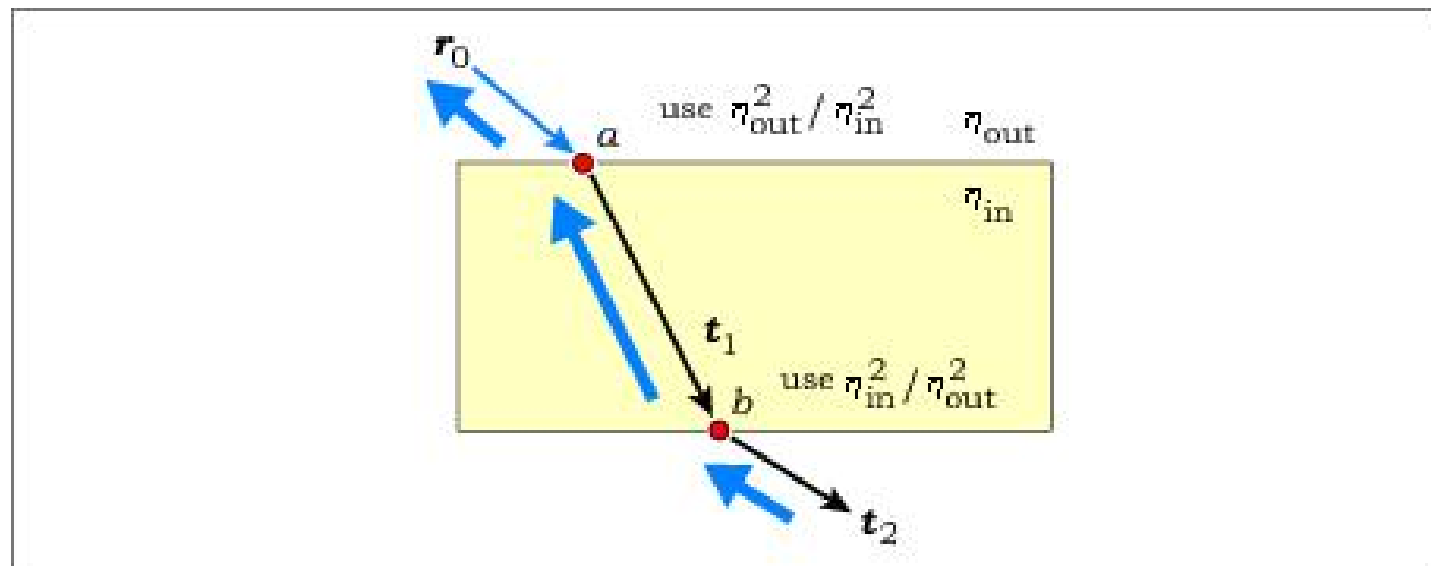
Illumination Model

- Remember:

\square_t : ior of transmitted radiance side

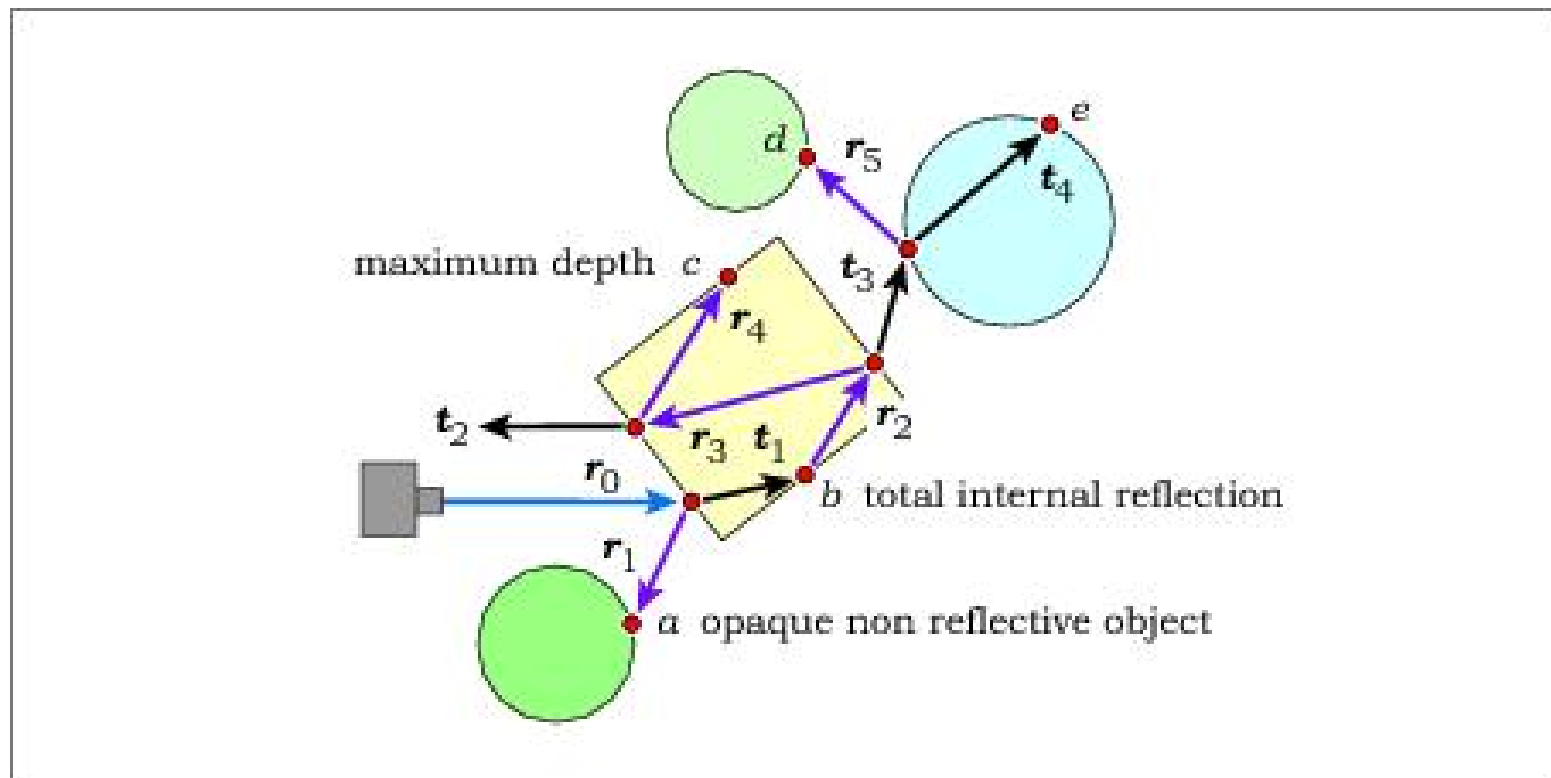
\square_i : ior of incident radiance side

In the expression $\left(\frac{\eta_t^2}{\eta_i^2}\right)$, it is easy to get these terms flipped because radiance direction is opposite ray direction:



Practical Issues

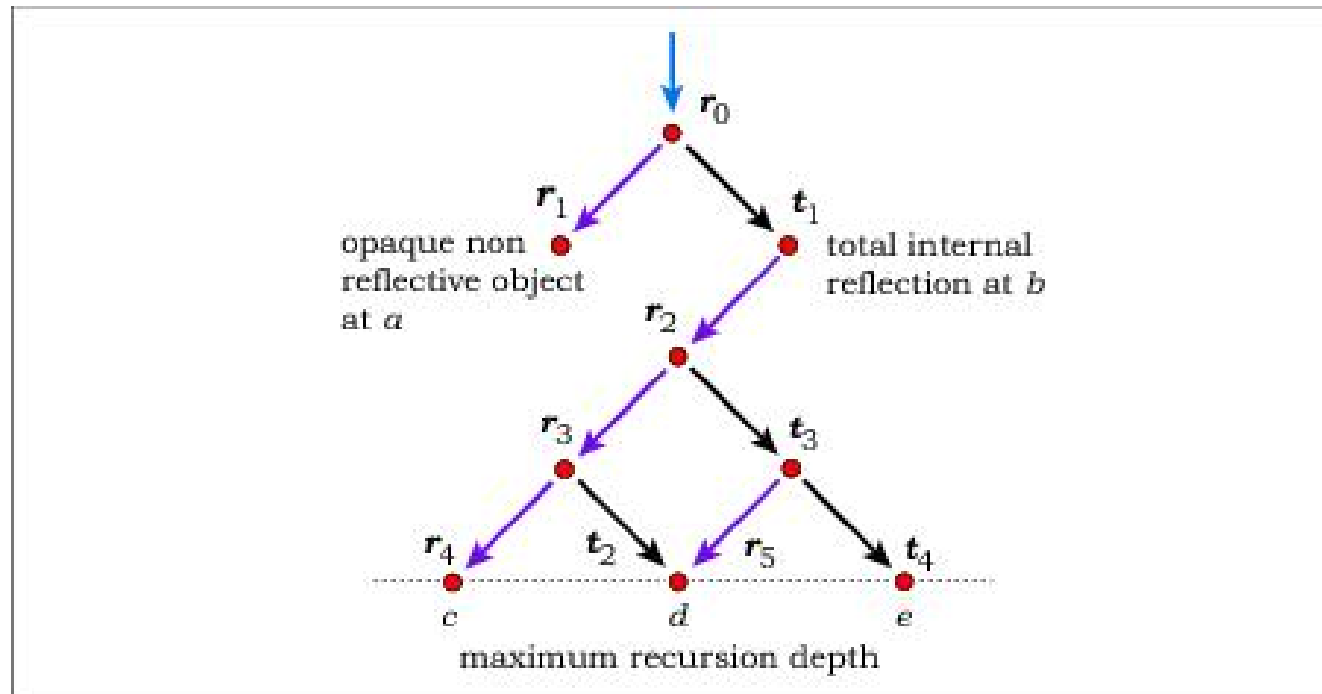
- Now that we have refraction, how do the rays propagate in the scene?



- What is the lower bound for MAX_DEPTH?

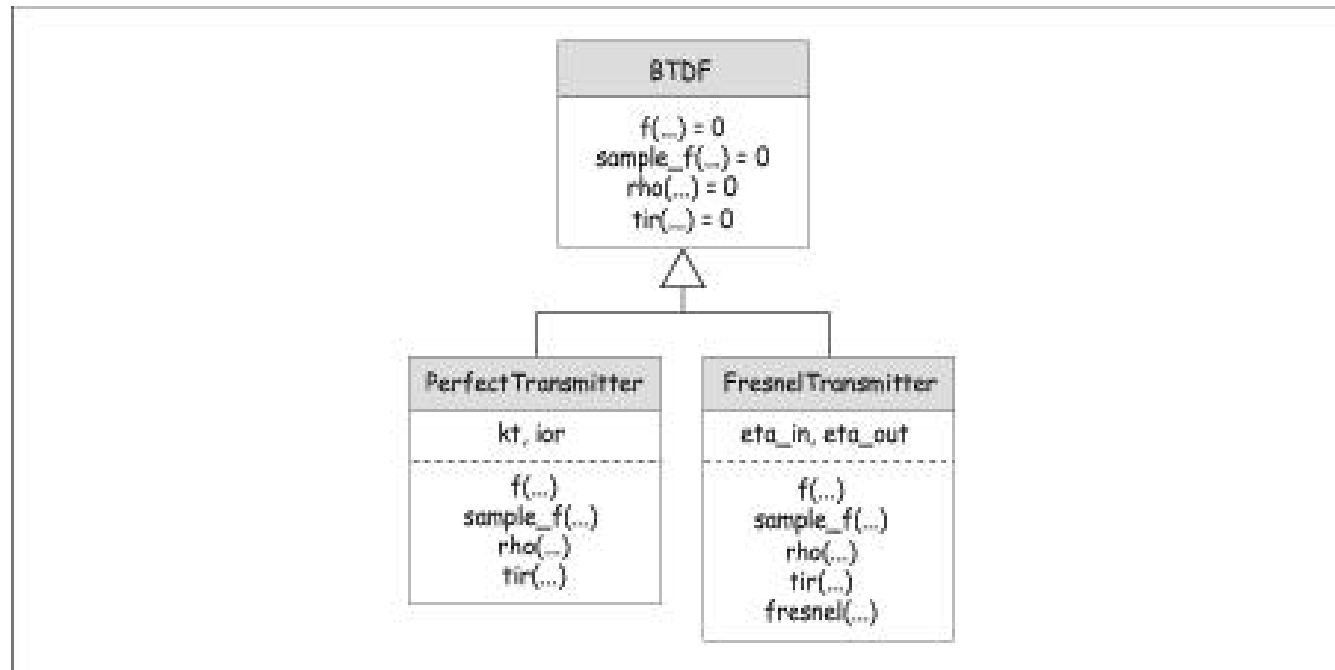
Practical Issues

- This can be represented as a binary tree:



- Do you see a potential problem?

Implementation



- Note: Since `ior` of air is 1 (or very close to 1), and the relative index of refraction is $\frac{\eta_{in}}{\eta_{out}}$, $\square = \square_{in}$ is used in the code.
- PerfectTransmitter:: `tir` determines if `tir` exists.
- PerfectTransmitter:: `sample_f` calculates the transmitted ray `wt`, and returns the RGB color value scaling factors (since color is not part of simple transparency)

Implementation

```
bool
PerfectTransmitter::tir(const ShadeRec& sr) const {
    Vector3D wo(-sr.ray.d);
    float cos_thetai = sr.normal * wo;
    float eta = ior;

    if (cos_thetai < 0.0)
        eta = 1.0 / eta;

    return (1.0 - (1.0 - cos_thetai * cos_thetai) / (eta * eta) < 0.0);
}
```

Implementation

```
// this computes the direction wt for perfect transmission
// and returns the transmission coefficient
// this is only called when there is no total internal reflection

RGBColor
PerfectTransmitter::sample_f(const ShadeRec& sr, const Vector3D& wo, Vector3D&
    wt) const {

    Normal n(sr.normal);
    float cos_thetai = n * wo;
    float eta = ior;

    if (cos_thetai < 0.0) { // transmitted ray is outside
        cos_thetai = -cos_thetai;
        n = -n; // reverse direction
of normal
        eta = 1.0 / eta; // invert ior
    }

    float temp = 1.0 - (1.0 - cos_thetai * cos_thetai) / (eta * eta);
    float cos_theta2 = sqrt(temp);
    wt = -wo / eta - (cos_theta2 - cos_thetai / eta) * n;

    return (kt / (eta * eta) * white / fabs(sr.normal * wt));
}
```

Implementation

- The material Transparent inherits from Phong, to give a reflective highlight to the transparent object (allows us to see that it exists in certain cases).
- Uses 1 BRDF for reflection, 1 BTDF for refraction, and 3 BRDF's inherited from Phong.
- `Transparent::shade` calculates the radiance by using the radiance equations discussed previously.
 - It first checks if `tir` exists: increments radiance by $1 \times \text{Radiance}$ of reflected ray (because refracted ray is gone, so $k_r=1.0$ and $k_t=0.0$).
 - If no `tir`: gets w_i from BRDF `sample_f`, gets w_o from BTDF `sample_f`, and uses radiance equation to calculate radiance.

Implementation

```
RGBColor
Transparent::shade(ShadeRec& sr) {
    RGBColor L(Phong::shade(sr));

    Vector3D wo = -sr.ray.d;
    Vector3D wi;
    RGBColor fr = reflective_brdf->sample_f(sr, wo, wi);
    Ray reflected_ray(sr.hit_point, wi);

    if(specular_btddf->tir(sr))
        L += sr.w.tracer_ptr->trace_ray(reflected_ray, sr.depth + 1);
    else {
        Vector3D wt;
        RGBColor ft = specular_btddf->sample_f(sr, wo, wt);
        Ray transmitted_ray(sr.hit_point, wt);

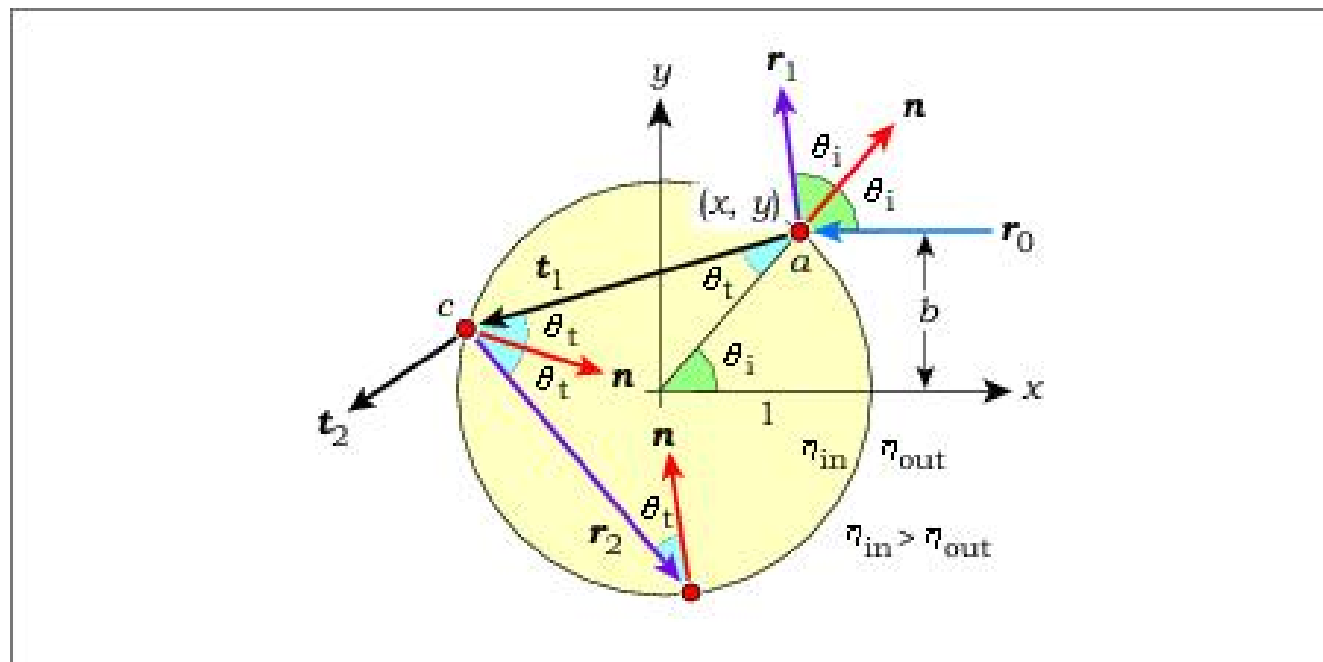
        L += fr * sr.w.tracer_ptr->trace_ray(reflected_ray, sr.depth + 1) *
            fabs(sr.normal * wi);
        L += ft * sr.w.tracer_ptr->trace_ray(transmitted_ray, sr.depth + 1) *
            fabs(sr.normal * wt);
    }

    return (L);
}
```

Analysis Using Spheres

- Because of their simplicity, spheres allow us to test our transparency rather intuitively.
- When a sphere's ior is larger than that of its surrounding medium (i.e. $\eta > 1$), t_{ir} can't occur because of the condition:

$$b > \eta, b \in [0,1]$$



Analysis Using Spheres

- To test if refraction is working properly, let $\eta = 1$:

```
Transparent* glass_ptr = new Transparent;
```

```
glass_ptr->set_ks(0.2);
```

```
glass_ptr->set_exp(2000.0);
```

```
glass_ptr->set_ior(1.0);
```

```
glass_ptr->set_kr(0.1);
```

```
glass_ptr->set_kt(0.9);
```

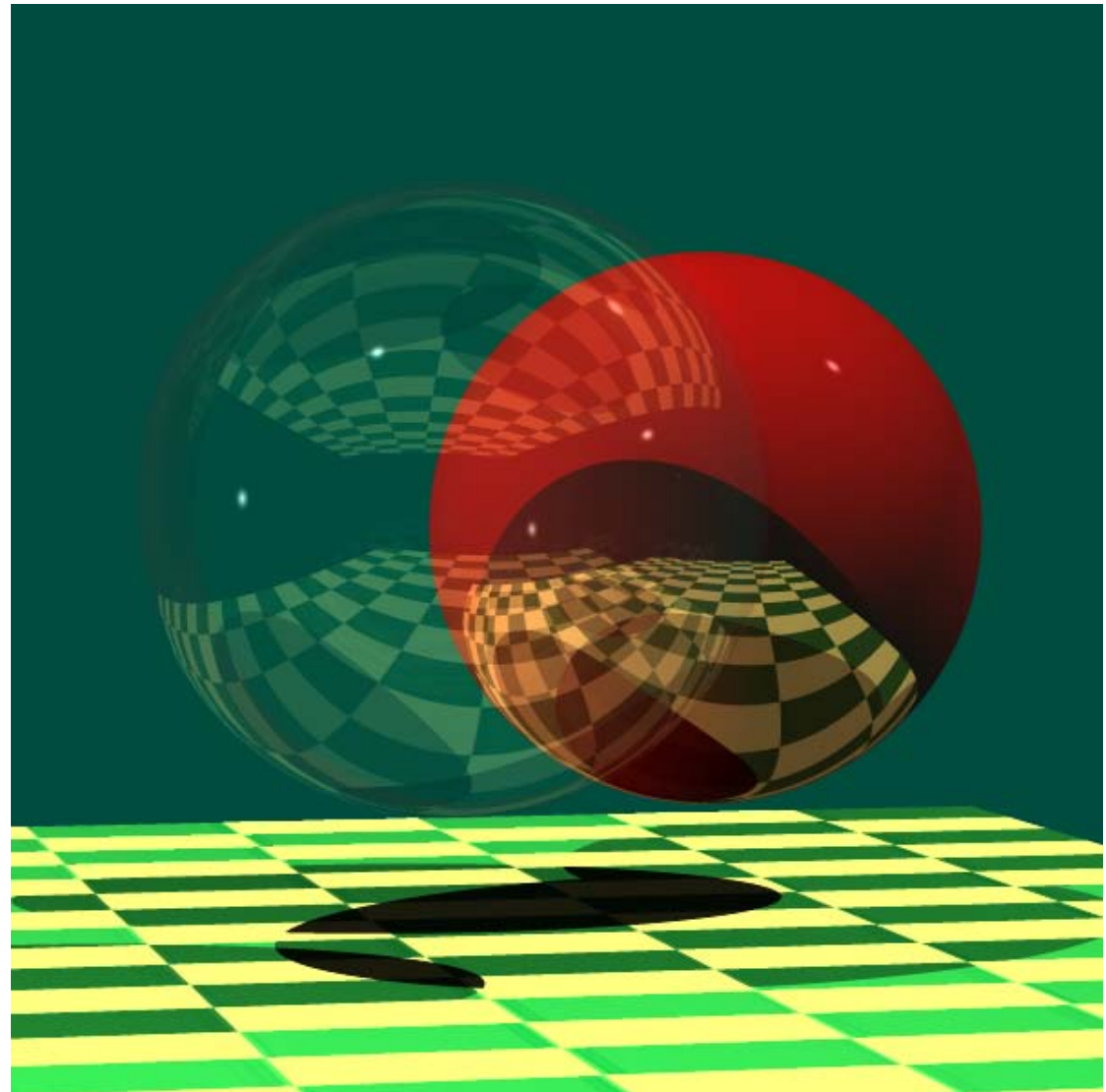
```
int num_samples = 16;
```

```
vp.set_hres(600);
```

```
vp.set_vres(600);
```

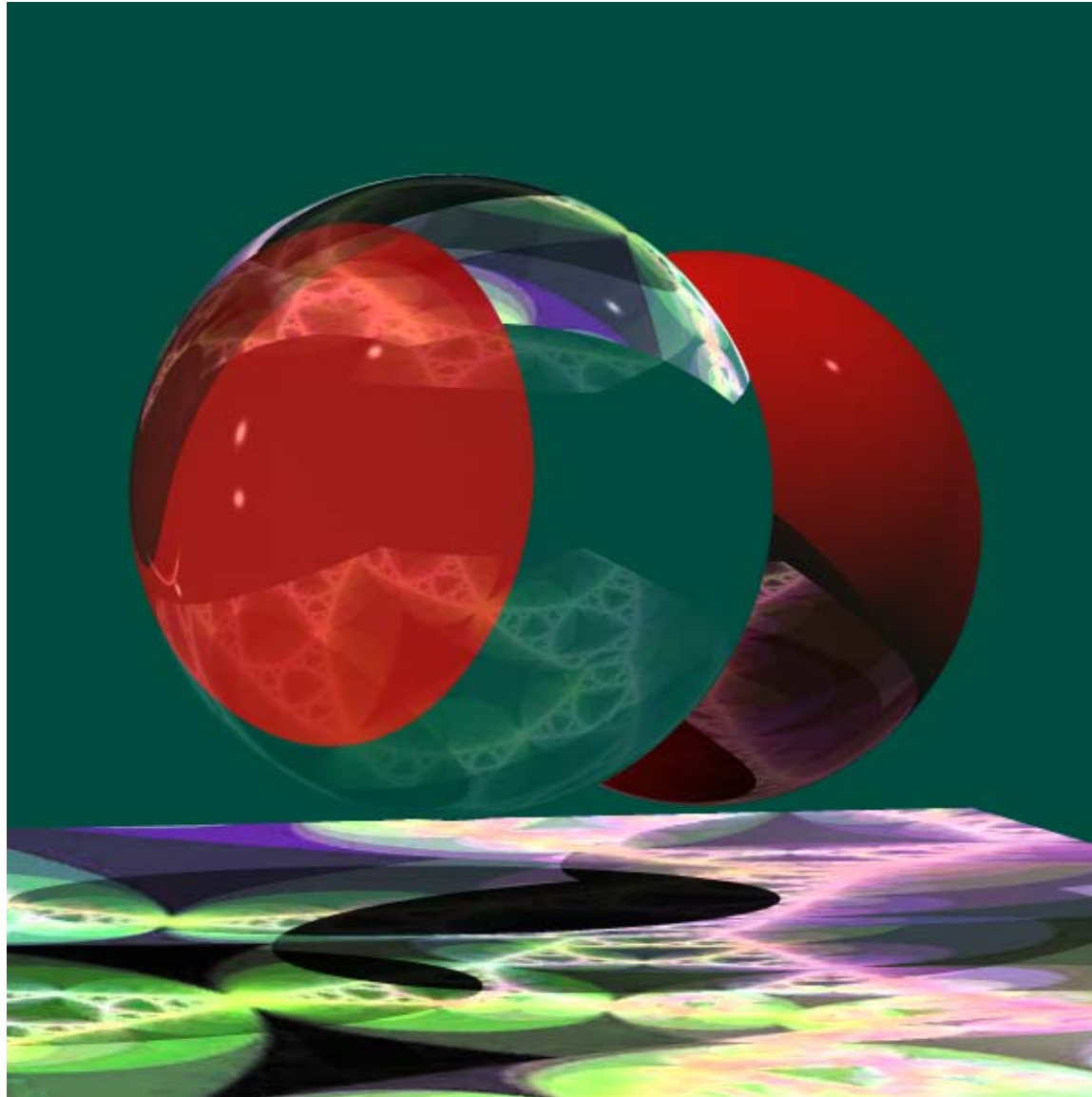
```
vp.set_samples(num_samples);
```

```
vp.set_max_depth(5);
```



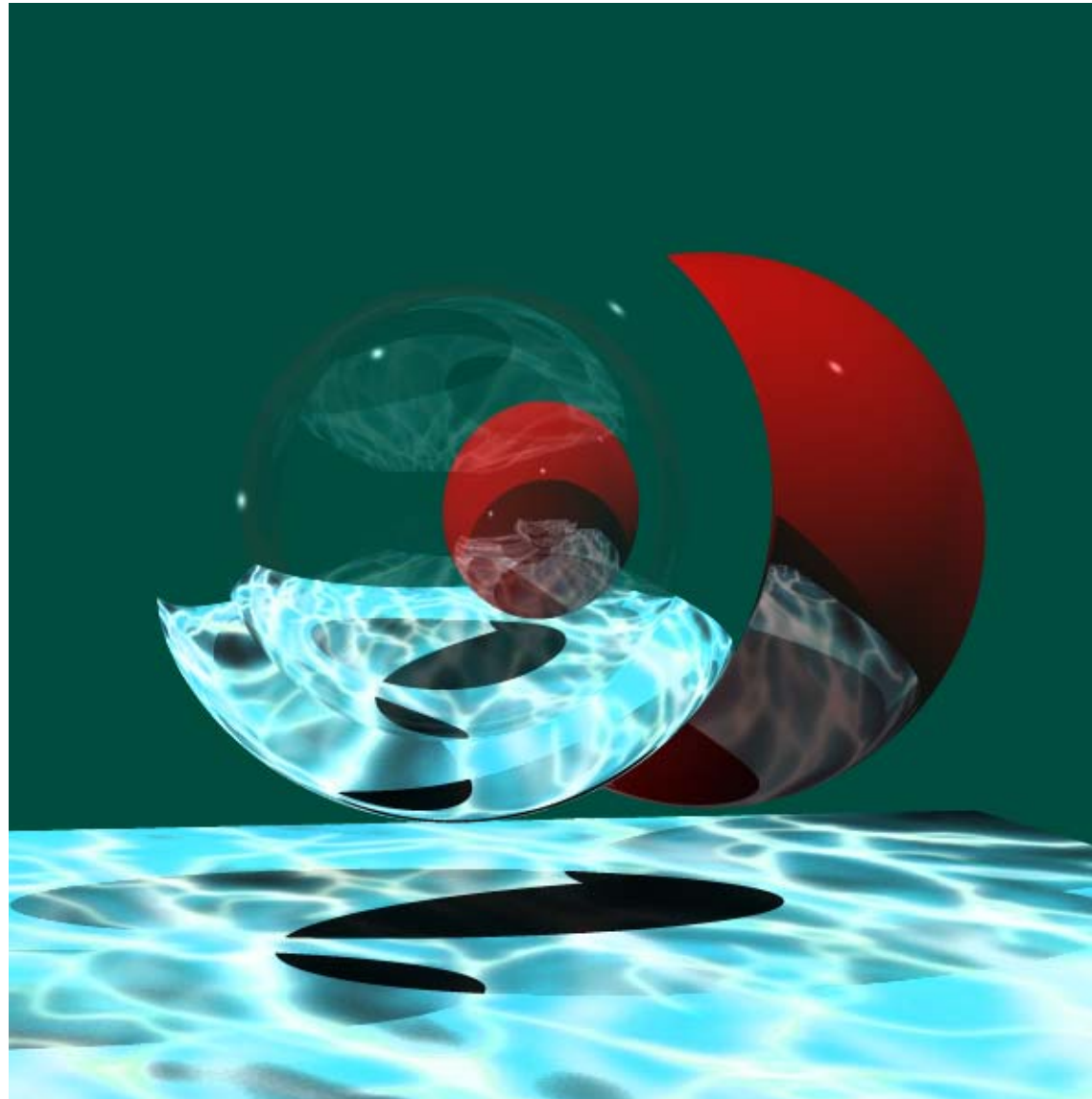
Analysis Using Spheres

- To generate a glass material, use $n=1.5$



Analysis Using Spheres

- To generate a bubble in water, use $n=1/1.33 = 0.75$



Be Careful we compound objects

 C:\Users\liquid\Desktop\Ray-Traced-Images\Figure27.20.jpg