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

by Wadii Bellamine

Presentation Outline:

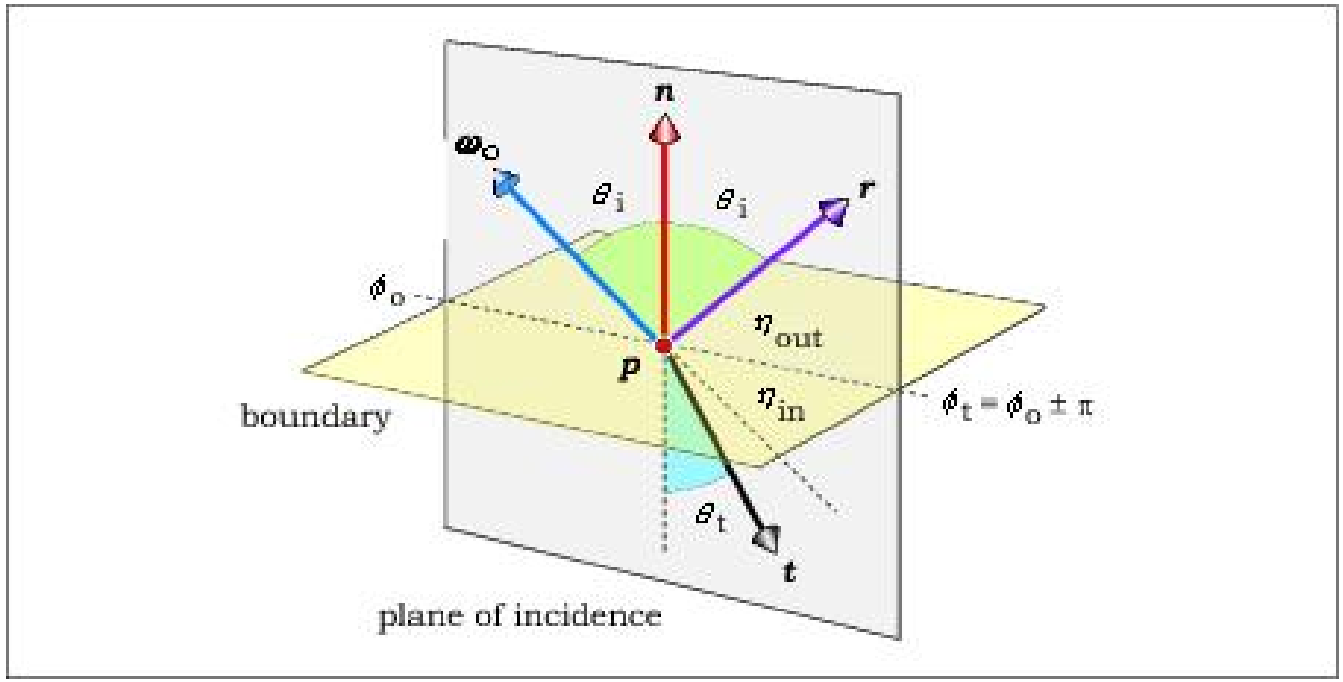
- Definition and physics of refraction
- Total Internal Reflection
- Illumination model
- Implementation
- Analysis of transparency using transparent spheres

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

Transparent objects

- Allow light to pass through them.
- Referred to as dieletrics:
  - Gases (air), glass, clear plastics, etc.

- When a ray hits the boundary between two dialetrics, 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:

$$c = 2.99 \times 10^7 \text{ m/s.}$$

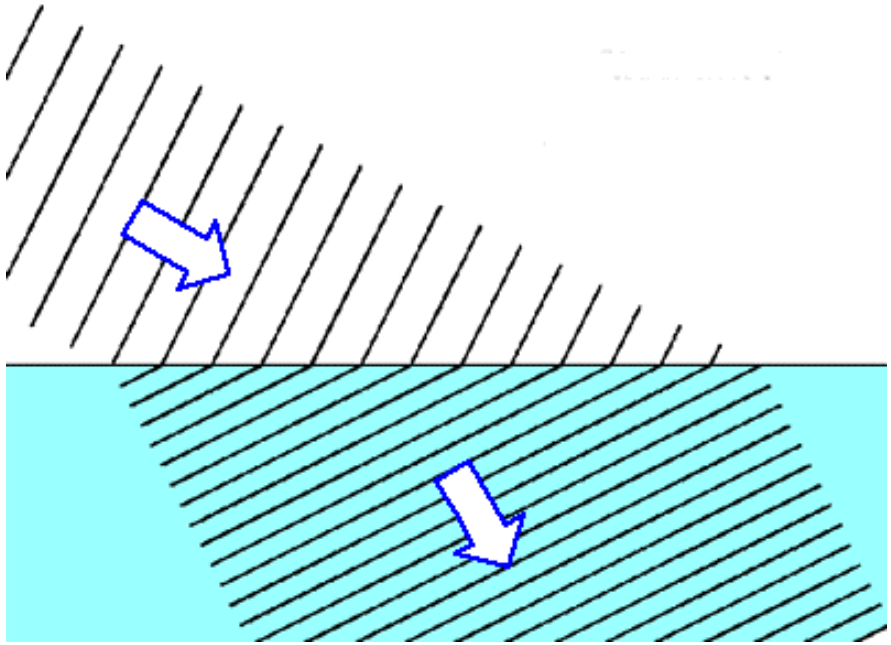
- Speed of light is slower when traveling through other media because of interaction with the medium's molecules.

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

- Absolute index of refraction:

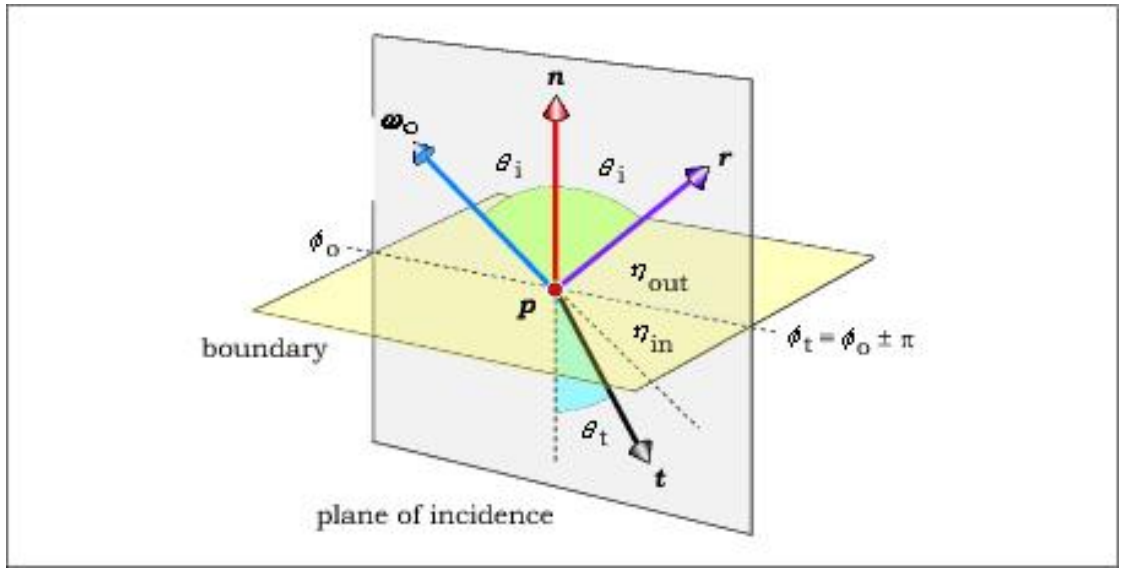
$$\square = c/v$$

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



http://www.school-for-champions.com/Science/images/waveobstacle-refraction.gif

Refraction



ω_o: incident ray
r: reflected ray
t: transmitted ray
θ_i: angle of incidence
θ_t: angle of refraction

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

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

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

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

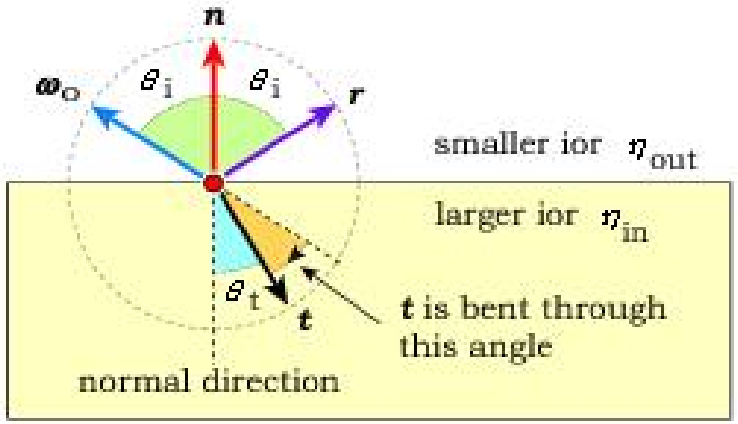
$$\cos\theta_i = \boldsymbol{n}\cdot\boldsymbol{\omega_o}$$

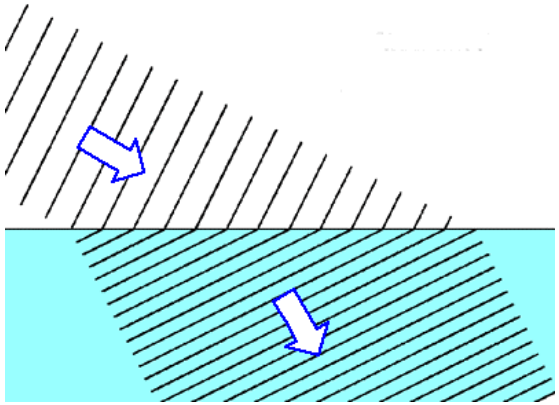
Refraction

- ***t*** is bent differently for □>1 and □<1.

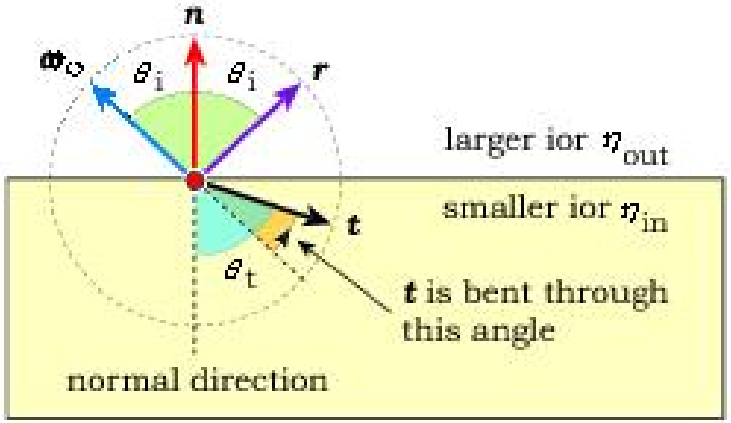
$$\eta = \frac{\eta_{in}}{\eta_{out}}$$

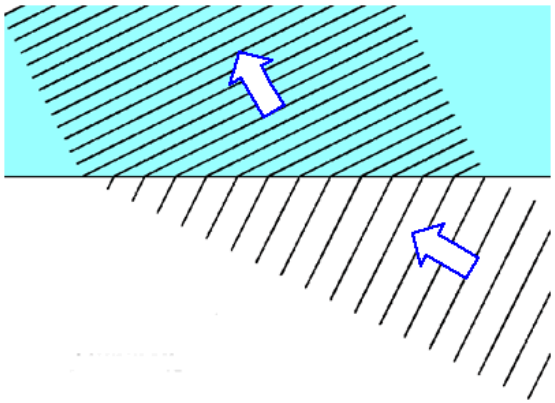
t bends towards the normal for □>1





t bends away from the normal for □<1

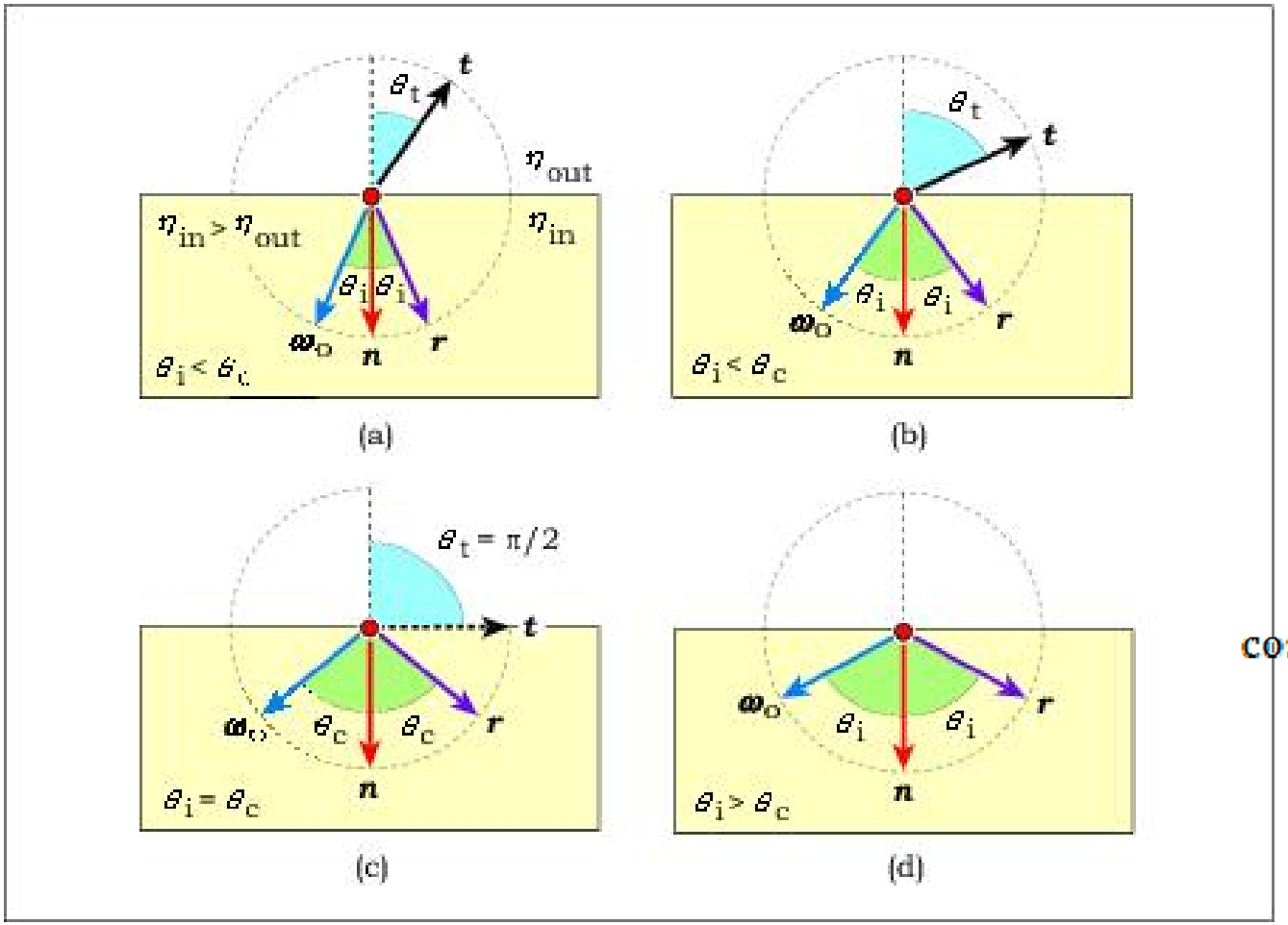




(Mentally flip arrows)

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 □<1).



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

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

▪When □$_c$ (critical angle) is reached, ***t*** has no energy, and ***r*** has all the energy.

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

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

Illumination Model

- Exitant radiance at a point **p**:

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

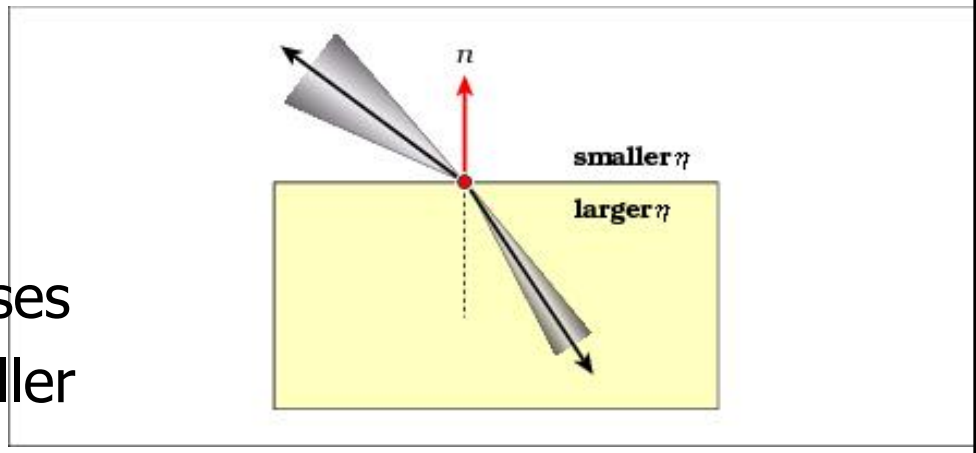
$$L_{indirect}(\boldsymbol{p}, \boldsymbol{\omega_o}) = L_r(\boldsymbol{p}, \boldsymbol{\omega_o}) + L_t(\boldsymbol{p}, \boldsymbol{\omega_o})$$

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

$f_{t,s}(\boldsymbol{p}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ 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 □ to larger □, cone angle decreases ∴ radiance increases
  - Going from larger □ to smaller □, 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:

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

- Using:

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

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

We get:

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

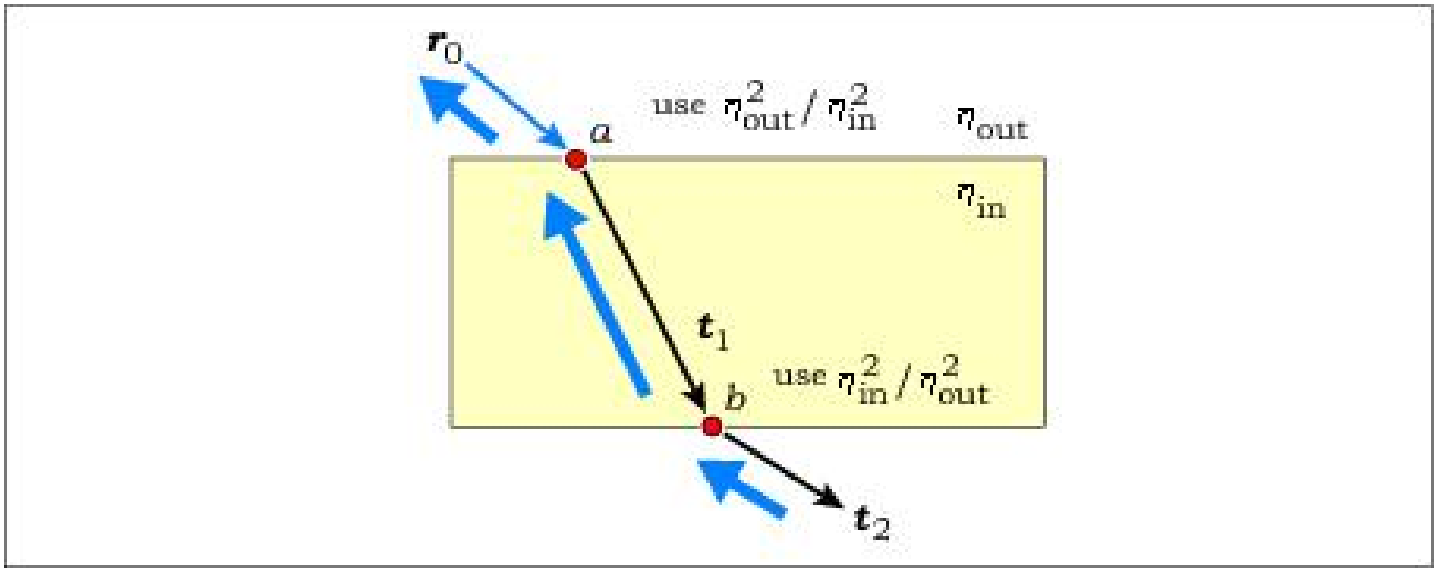
Where:

$$L_i(\boldsymbol{p}, \boldsymbol{\omega_i}) = L_o(\boldsymbol{r}_c(\boldsymbol{p}, \boldsymbol{\omega_i}), -\boldsymbol{\omega_i})$$

Illumination Model

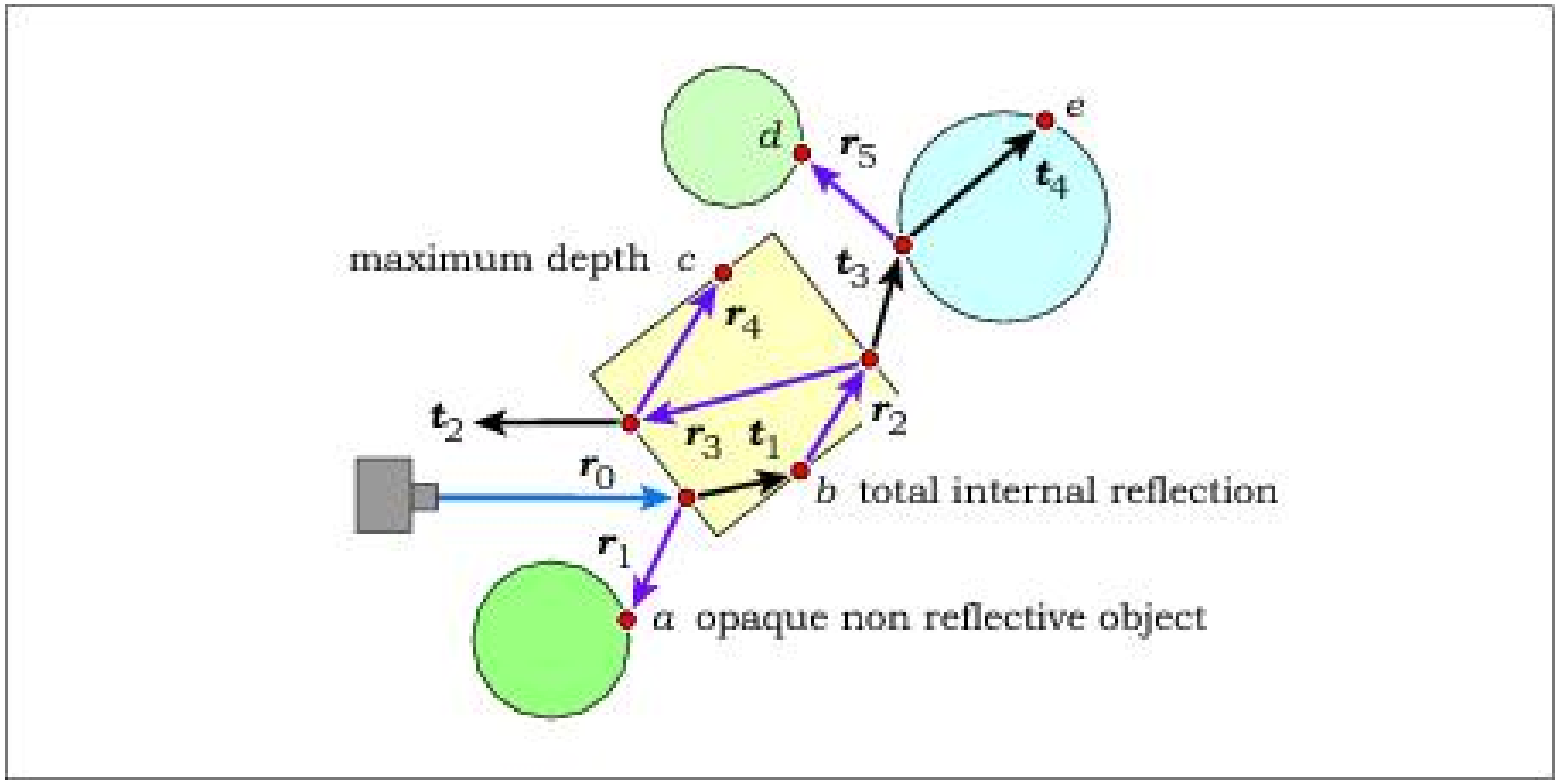
- Remember:
  - $\square_t$: ior of transmitted radiance side
  - $\square_i$: ior of incident radiance side

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

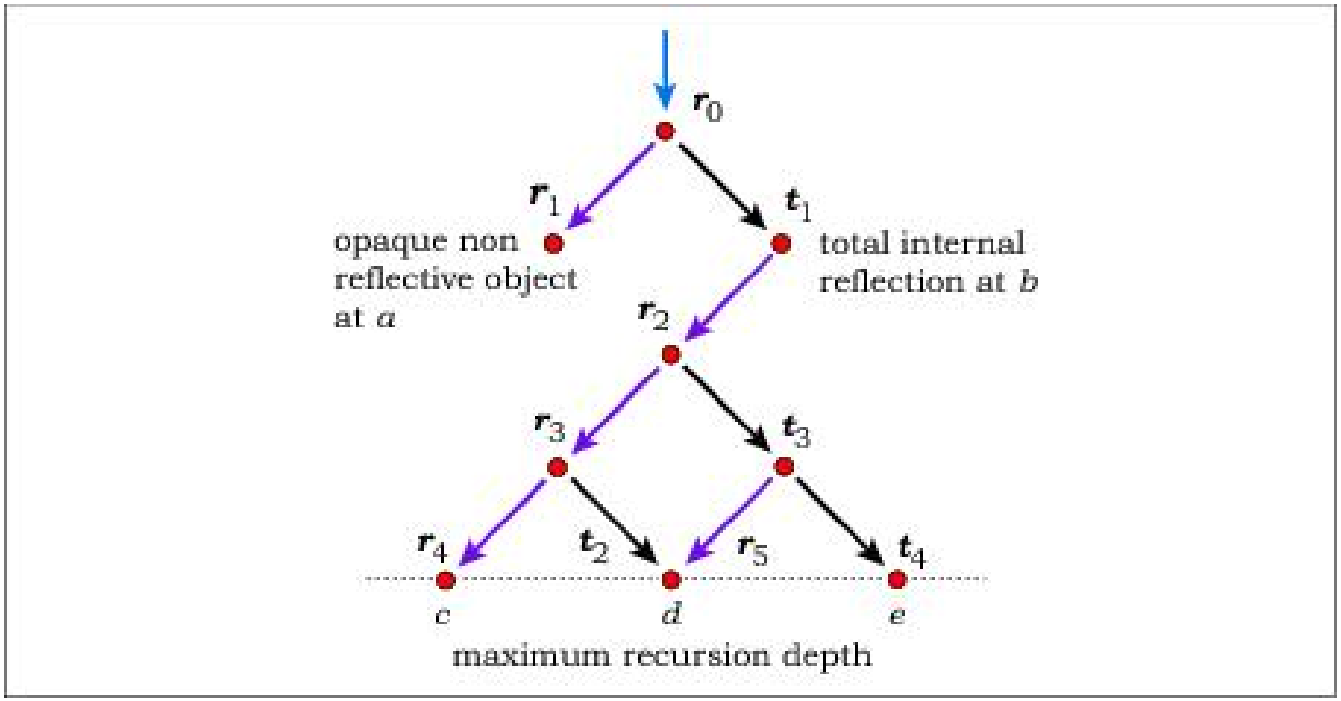
- Now that we have refraction, how do the rays propagate in the scene?



- What is the lower bound for MAX_DEPTH?

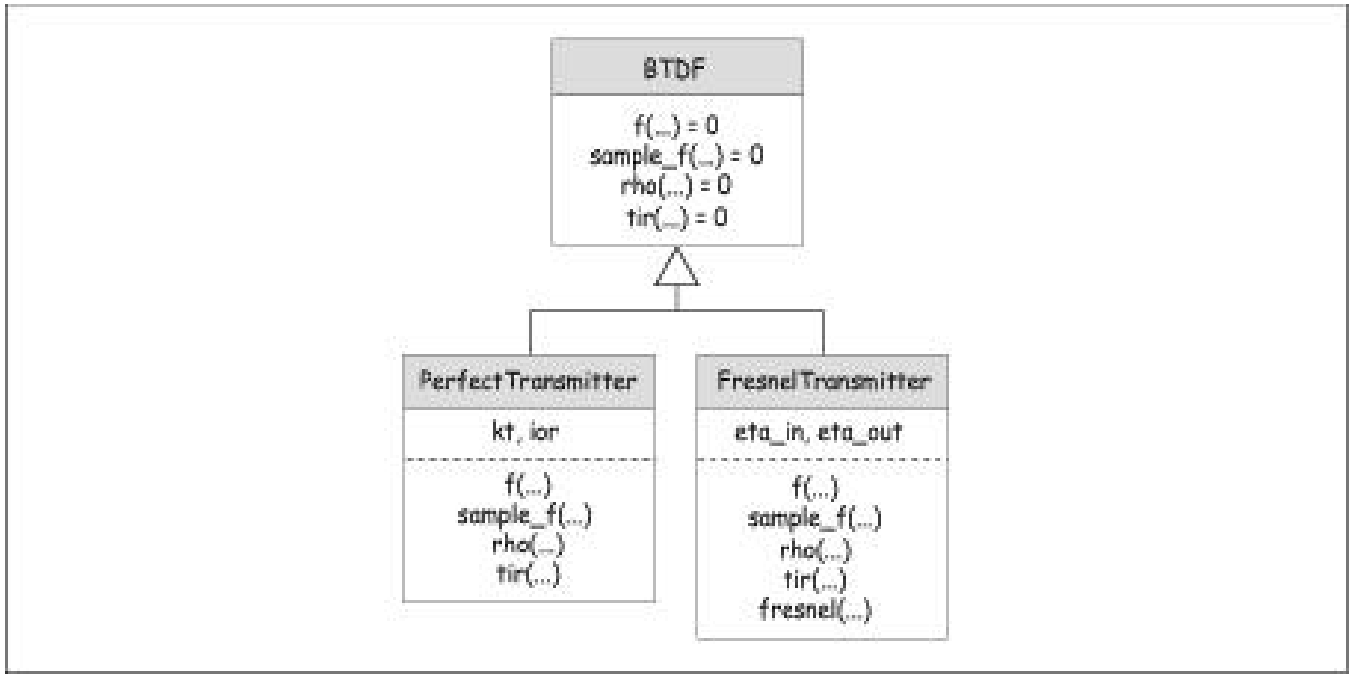
Practical Issues

- 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 1xRadiance of reflected ray (because refracted ray is gone, so kr=1.0 and kt=0.0).
  - If no tir: gets wi from BRDF sample_f, gets wo from BTDF sample_f, and uses radiance equation to calculate radiance.

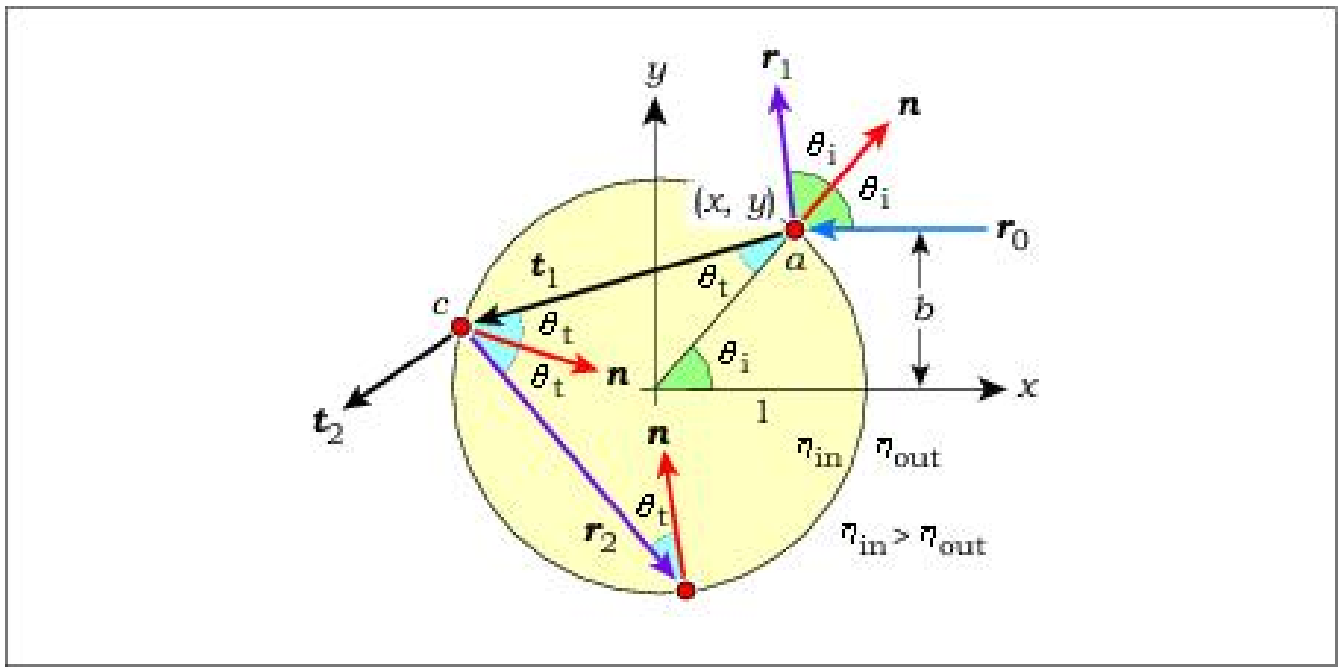
Implementation

```
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_btdf->tir(sr))
          L += sr.w.tracer_ptr->trace_ray(reflected_ray, sr.depth +1);
    else {
          Vector3D wt;
          RGBColor ft = specular_btdf->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. □>=1) , tir can't occur because of the condition:

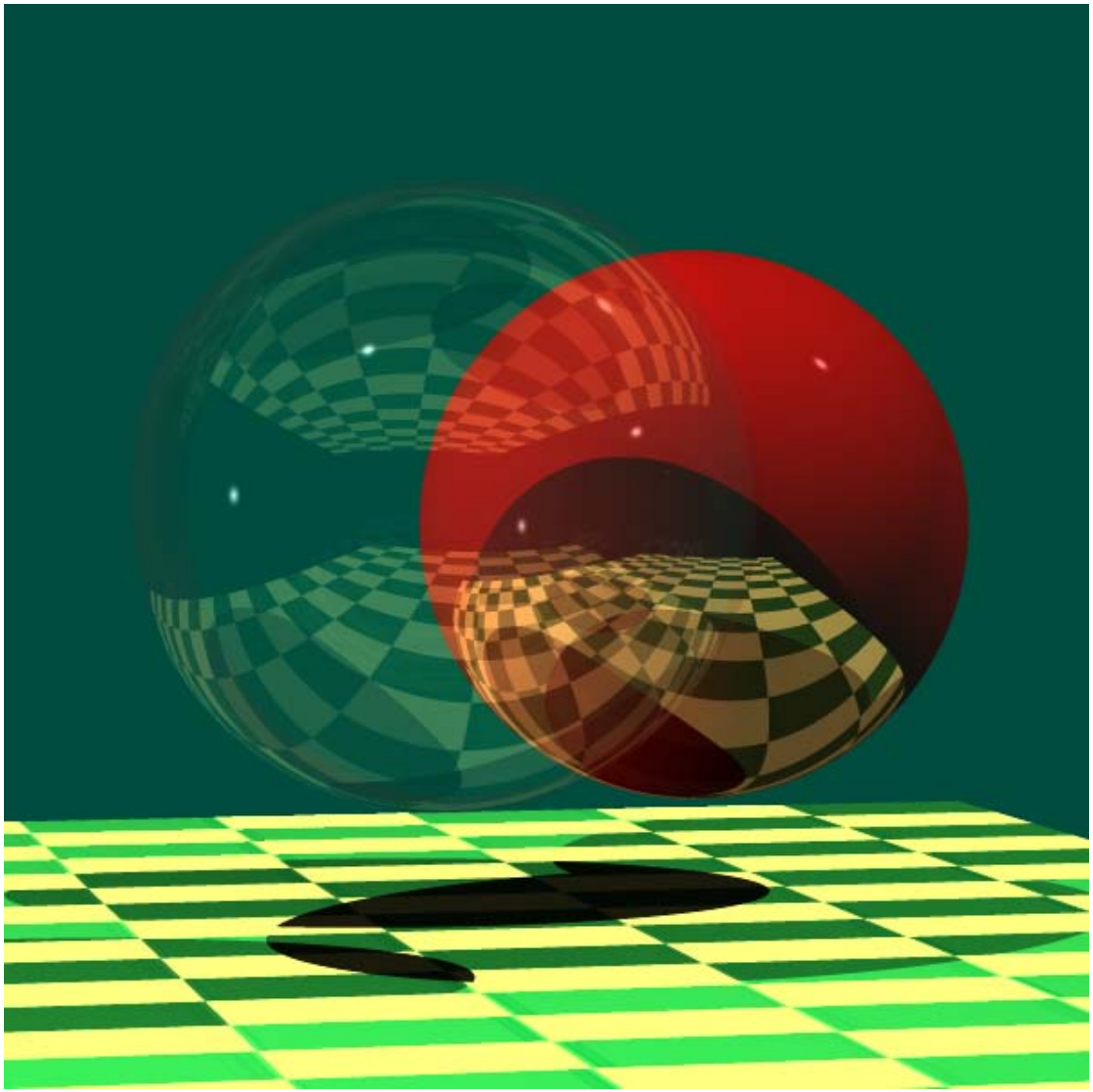
$$b > \square,\ b \in [0,1]$$



Analysis Using Spheres

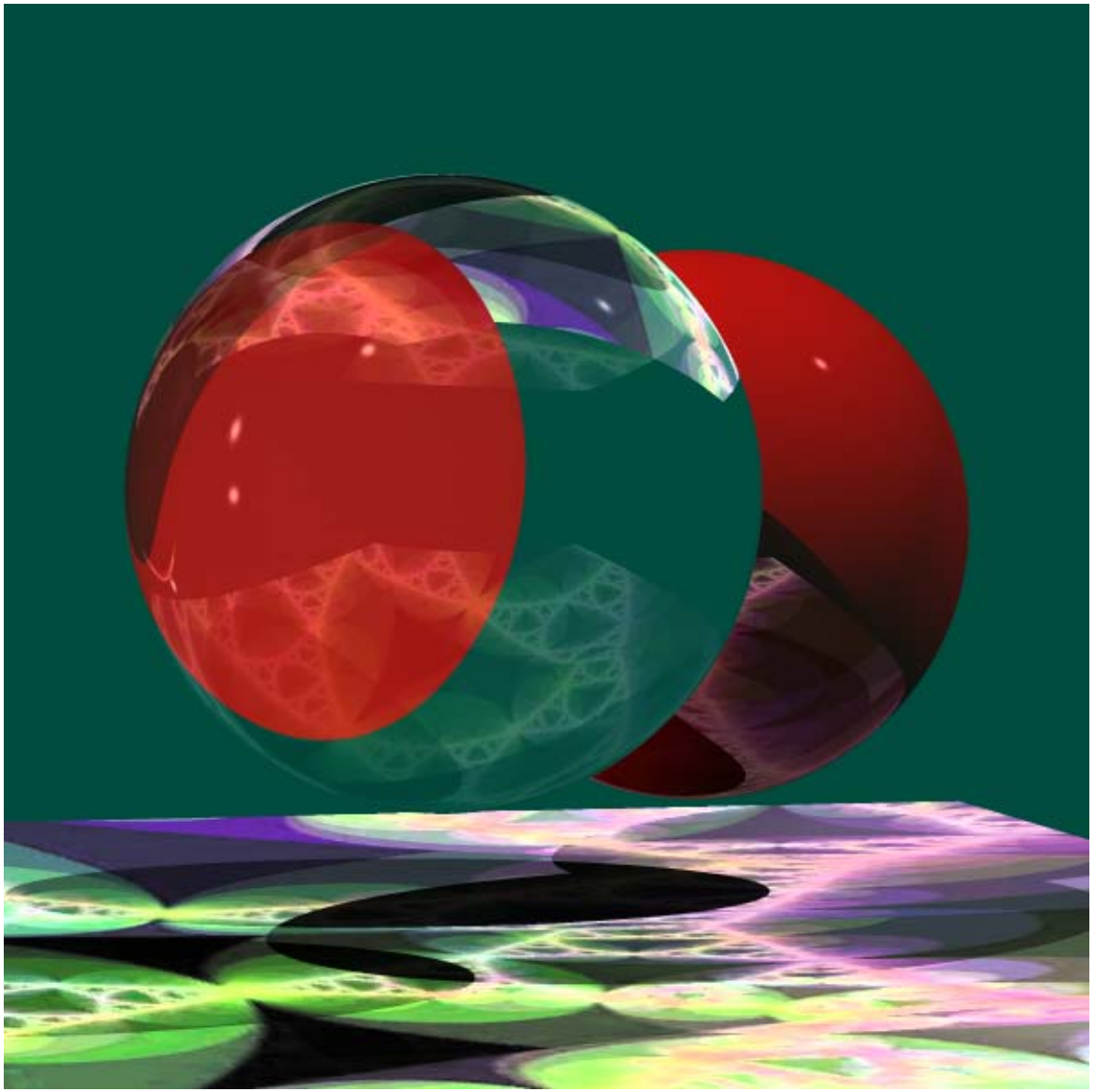
- To test if refraction is working properly, let □ =1:

```
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);
```



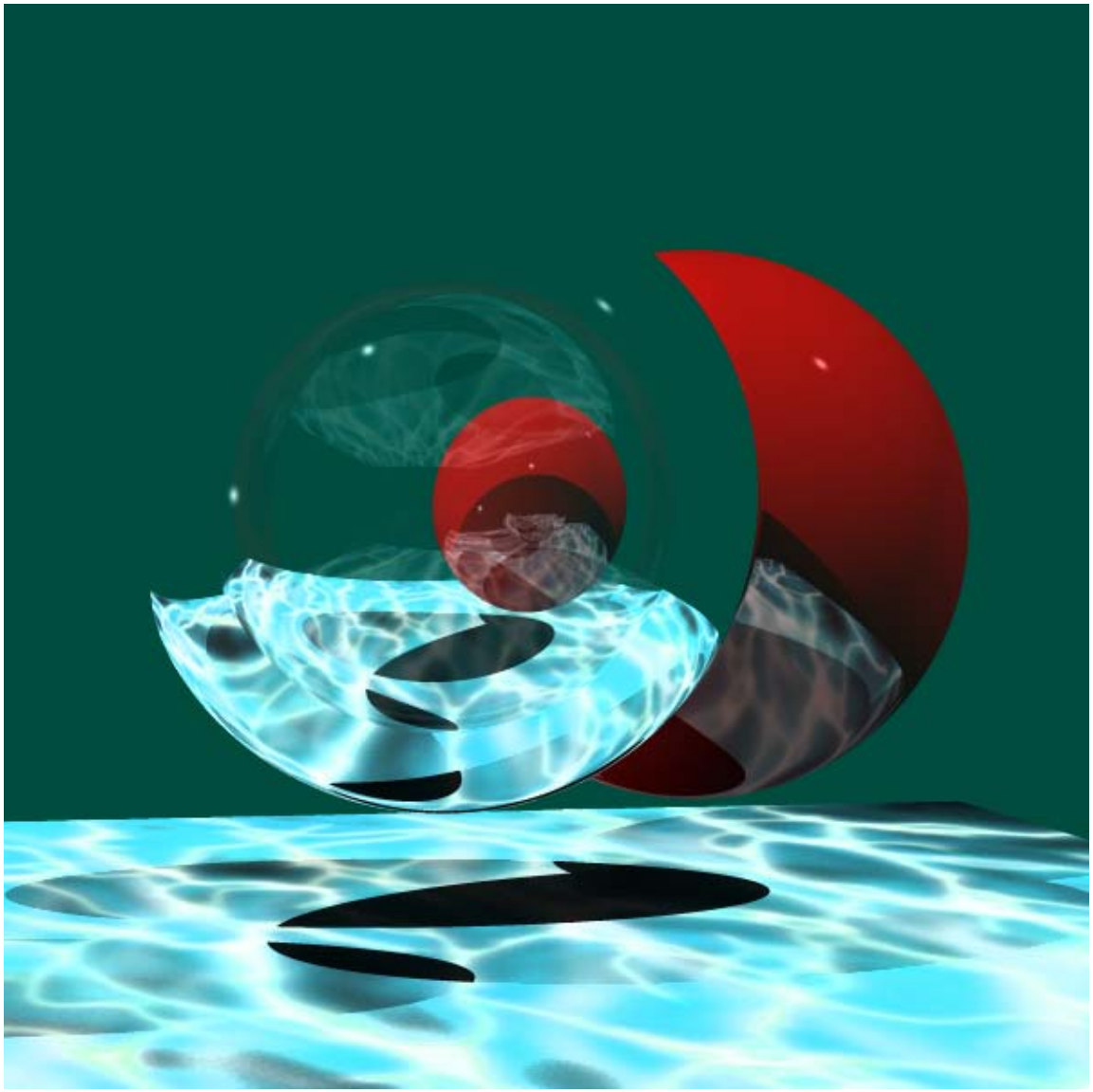
Analysis Using Spheres

- To generate a glass material, use n=1.5

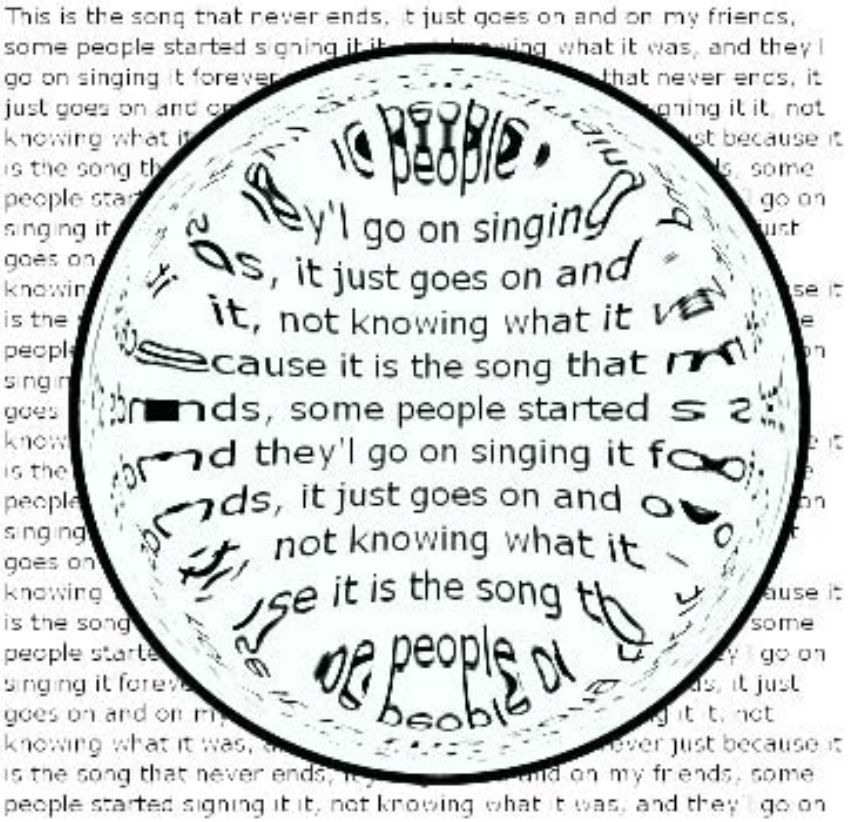


Analysis Using Spheres

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



Magnifying Glass:



Be Careful we compound objects

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