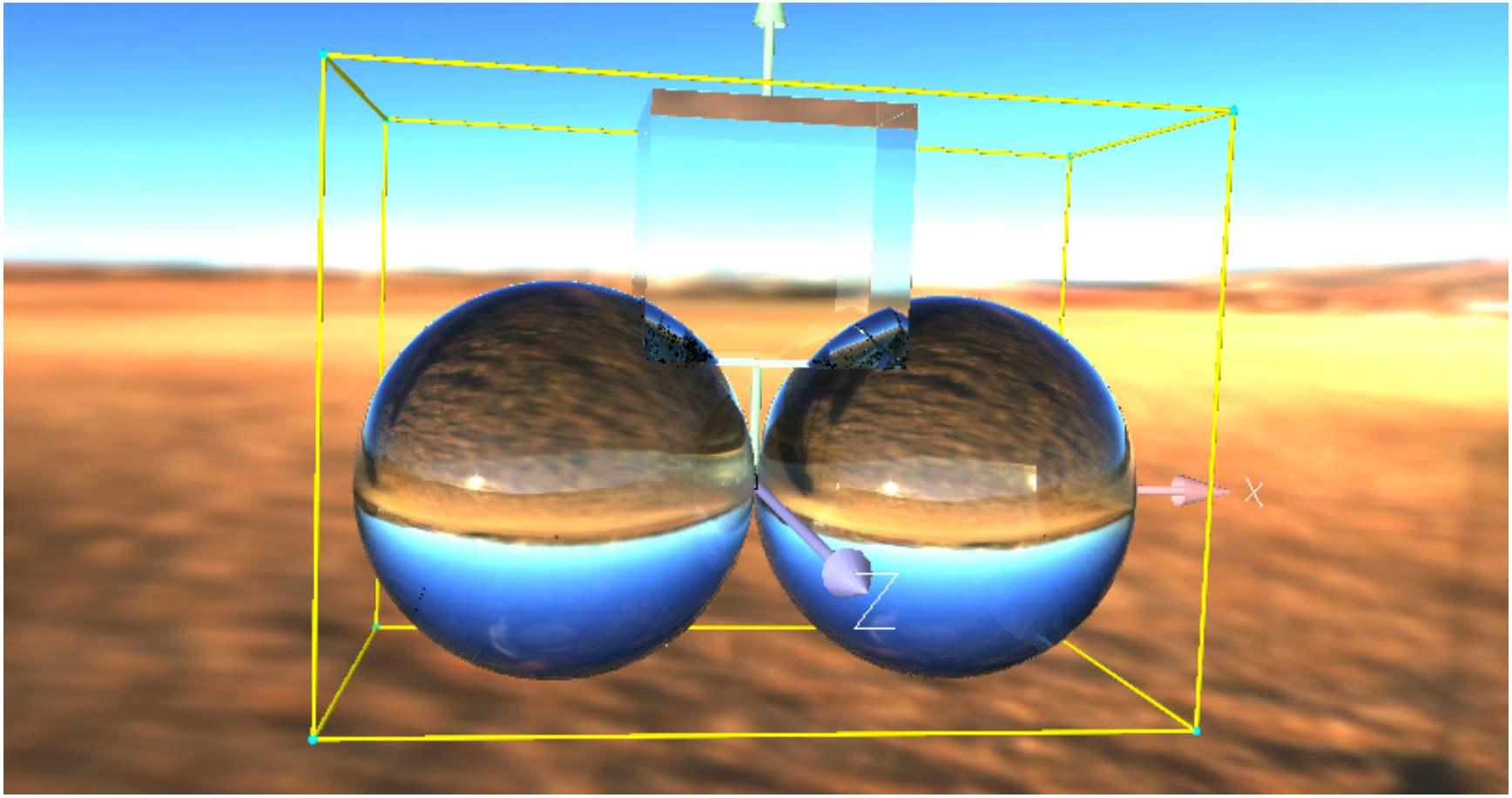
DigiPen
INSTITUTE OF TECHNOLOGY

CS300 Refraction

Translucency / Refraction

Transparency



Snell's Law

- Let α_i be the angle of incidence, which is the angle between the incident ray and the surface normal.
- α_t is the angle of refraction, which is formed by the refracted ray and the surface normal.
- The relationship between the two angles is given by Snell's law

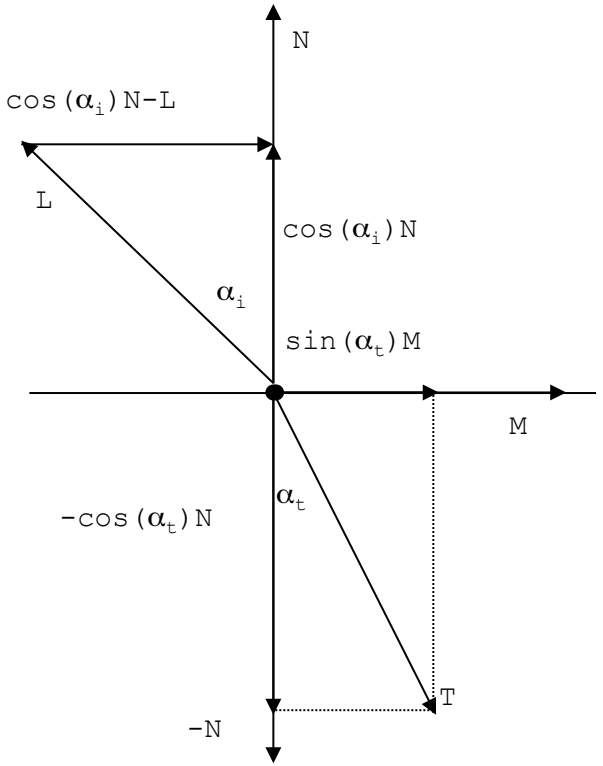
$$n_i \sin \alpha_i = n_t \sin \alpha_t$$

Schematic Diagram

L : light vector

N: surface normal

T : transmitted vector / refraction vector



Snell's Law *(cont'd)*

- Where n_i and n_t are the refraction indices of the materials through which the light passes.
- The refraction index of a material is the ratio of the speed of light in vacuum over the speed of light in the material.
- A vacuum has an index of 1, as does the atmosphere to close approximation; all other materials have higher values.

Snell's Law *(cont'd)*

- Transmitted light effect can be modeled by:

$$I_{refract} = I_s K_t (\max(V.T, 0))^{nt}$$

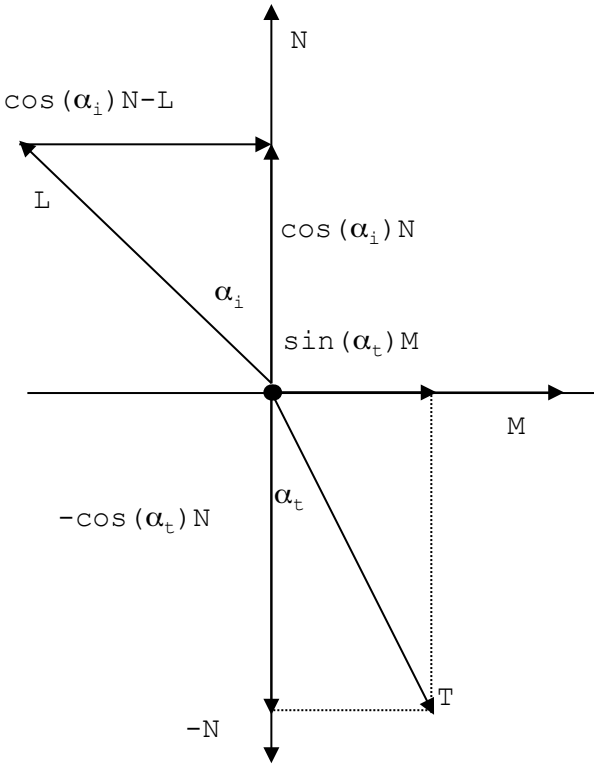
- Where:
  - I_s is the incoming light source intensity
  - K_t is the transmission coefficient
  - T is the unit transmit refraction vector
  - V is the unit view direction vector
  - nt is the index that controls the highlight tightness

Refraction Vector

- Calculation of the refracted light vector T can be illustrated by the following picture:

$$T = V_1 + V_2$$

$$T = \sin(\alpha_t)M - \cos(\alpha_t)N$$



Refraction Vector *(cont'd)*

- Where M is a unit vector perpendicular to the surface normal N.
- M belongs to the incident plane defined by N and the incident light unit vector L.
- We know that:

  M is a vector parallel to $\cos(\alpha_i)N - L$

  M length is 1 $\left|\cos(\alpha_i)N - L\right| = \sin(\alpha_i)M$

Refraction Vector *(cont'd)*

- Then:

$$\left|\cos(\alpha_i)N - L\right| = \sin(\alpha_i)M$$

$$M = (\cos(\alpha_i)N - L) / \sin(\alpha_i)$$

- Substituting M yields:

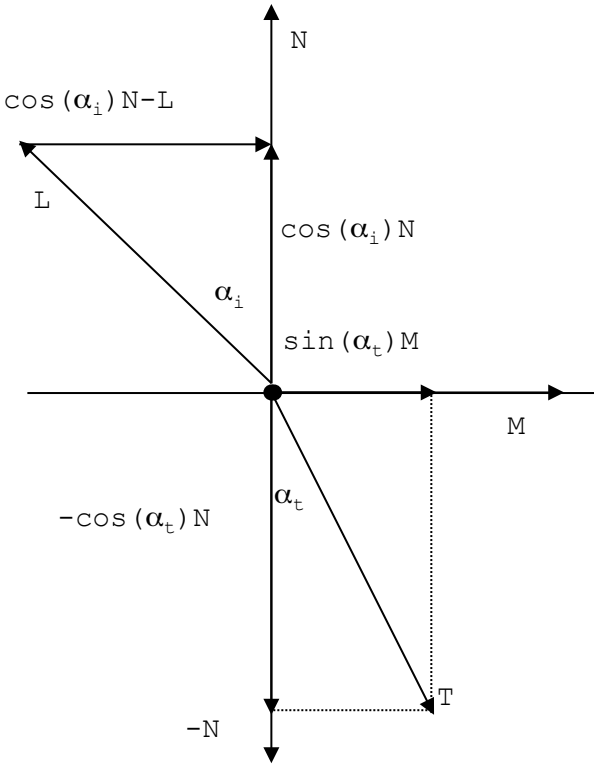
$$T = K(\cos(\alpha_i)N - L) - \cos(\alpha_t)N$$

$$T = (K\cos(\alpha_i) - \cos(\alpha_t))N - KL$$

- where

$$K = \frac{\sin(\alpha_t)}{\sin(\alpha_i)} = \frac{n_i}{n_t}$$

Refraction Vector *(cont'd)*



- Since

$$\cos(\alpha_i) = N.L$$

- and

$$\cos(\alpha_t) = \sqrt{1 - \sin^2(\alpha_t)}$$

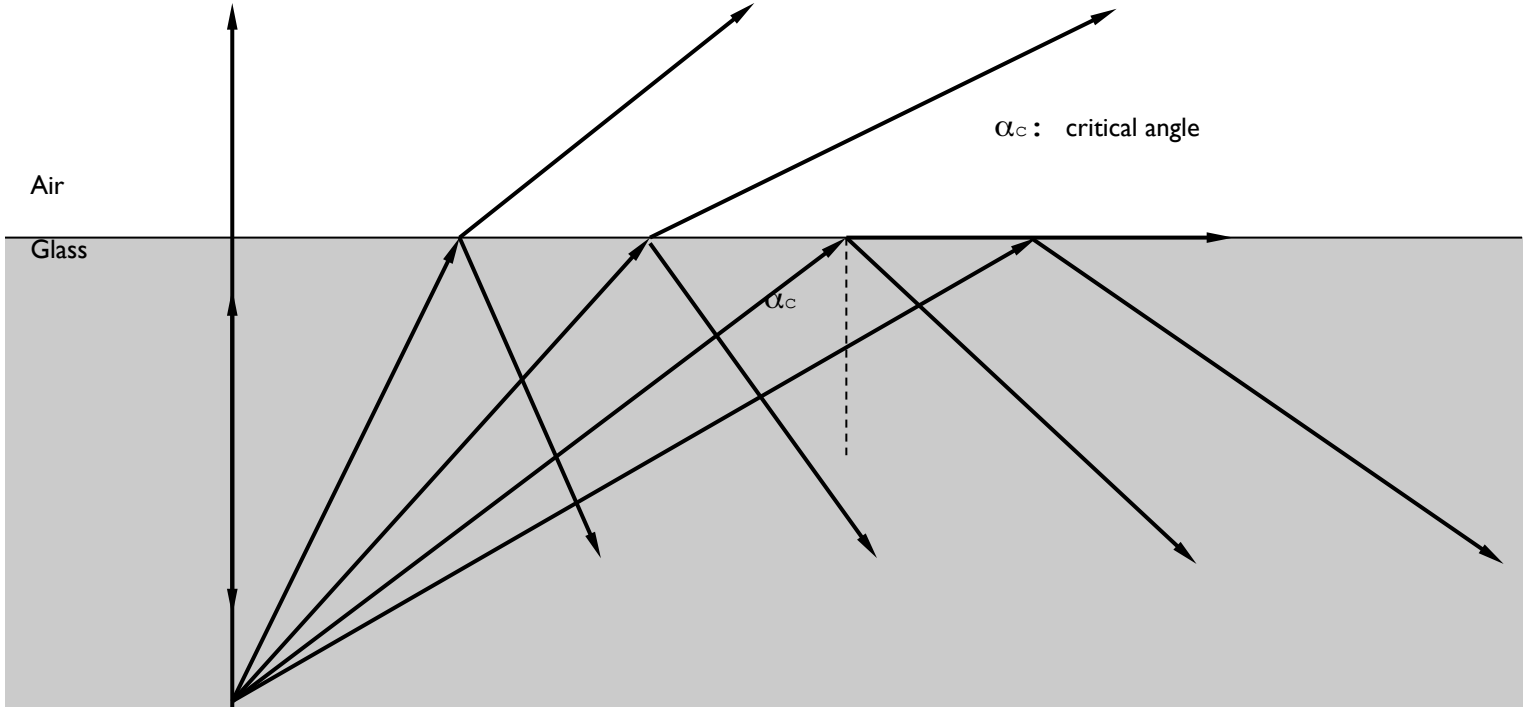
$$\cos(\alpha_t) = \sqrt{1 - K^2 \sin^2(\alpha_i)}$$

$$\cos(\alpha_t) = \sqrt{1 - K^2(1 - (N.L)^2)}$$

- Thus

$$T = \left[K(N.L) - \sqrt{1 - K^2(1 - (N.L)^2)}\right]N - K.L$$

Total Internal Reflection



Total Internal Reflection

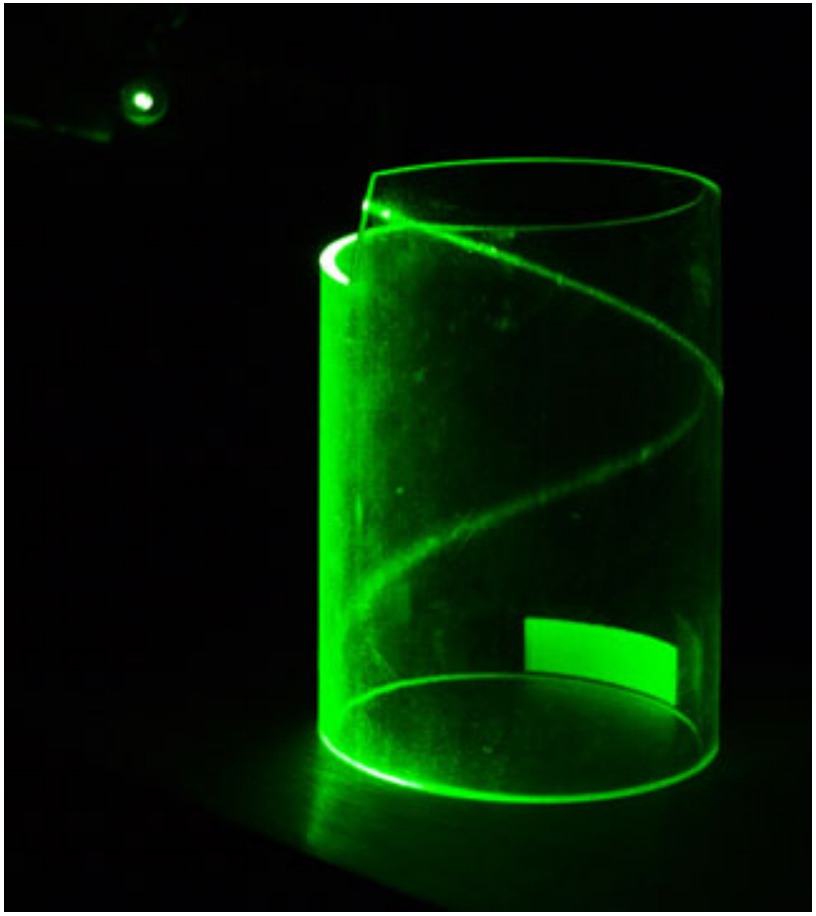
- The transmitted angle formed by the refracted ray and the surface normal is greater than the incident angle formed by the incident ray and the surface normal.
- This case occurs when the light passes from a material to another material with a lower index of refraction.
- As the incident angle α_i becomes larger, the refracted angle α_t might become greater than 90^o.

Total Internal Reflection *(cont'd)*

- This will cause the ray to be reflected from the interface between the media, rather than being transmitted.
- This phenomenon is called total internal reflection and it occurs when the incident angle exceeds the critical angle, whose value can be calculated by:

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

TIR Example in Real Life



Local Light Intensity

$$I_{local} = I_{ambient} + I_{diffuse} + I_{specular} + I_{refract}$$

$$I_{local} = E_{mat} + G_a K_a + Att \times SpotEffect \times \left(I_{ambient} + I_{diffuse} + I_{specular} + I_{refract} \right)$$

$$I_{local} = E_{mat} + G_a K_a + Att \times SpotEffect \times \left(I_{ambient} + I_d K_d (N.L) + I_s K_s (R.V)^{ns} + I_s K_t (T.V)^{nt} \right)$$

Refraction Index

Material	Index value
Vacuum	1.0
Air (20° Celsius)	1.0003
Water	1.33
Fused quartz	1.46
Crown glass	1.52
Dense flint glass	1.66

Fresnel Reflection

- When light moves from a material with refractive index n_i to another material with refractive index n_t, both reflection and refraction could occur at the material interface.
- The amount of reflection (or refraction) that occurs is determined by Fresnel reflection.
- The equations depends on the light polarization with respect to the surface.

Fresnel Reflection *(cont'd)*

- If the light is s-polarized, the reflective coefficient is:

$$R_s = \left[\frac{\sin(\theta_t - \theta_i)}{\sin(\theta_t + \theta_i)}\right]^2 = \left(\frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t}\right)^2 = \left[\frac{n_1 \cos\theta_i - n_2\sqrt{1 - \left(\frac{n_1}{n_2}\sin\theta_i\right)^2}}{n_1 \cos\theta_i + n_2\sqrt{1 - \left(\frac{n_1}{n_2}\sin\theta_i\right)^2}}\right]^2$$

- If the light is p-polarized, the reflective coefficient is:

$$R_p = \left[\frac{\tan(\theta_t - \theta_i)}{\tan(\theta_t + \theta_i)}\right]^2 = \left(\frac{n_1 \cos\theta_t - n_2 \cos\theta_i}{n_1 \cos\theta_t + n_2 \cos\theta_i}\right)^2 = \left[\frac{n_1\sqrt{1 - \left(\frac{n_1}{n_2}\sin\theta_i\right)^2} - n_2 \cos\theta_i}{n_1\sqrt{1 - \left(\frac{n_1}{n_2}\sin\theta_i\right)^2} + n_2 \cos\theta_i}\right]^2$$

- Note: n_1 and n_2 are n_i and n_t respectively.

Fresnel Reflection *(cont'd)*

- If the light is not polarized, contains equal mix of s and p polarization:

$$R = (R_s + R_p) * 0.5$$

- In all case, the refractive coefficient is:

$$T = 1 - R$$

Approximation

- Fresnel terms are complex to calculate
- Approximation suggested by Christophe Schlick

$$F = f + (1 - f)(1 - V \bullet N)^5$$

$$f = \frac{(1.0 - \frac{n_1}{n_2})^2}{(1.0 + \frac{n_1}{n_2})^2}$$

Vertex Shader

```
#version 140
…
const float Eta = 0.66; // Ratio of indices of refraction
const float FresnelPower = 5.0;
const float F = ((1.0-Eta) * (1.0-Eta)) / ((1.0+Eta) * (1.0+Eta));

out vec3 Reflect;
out vec3 Refract;
out float Ratio;

void main()
{
 …
  Ratio = F + (1.0 - F) * pow((1.0 - dot(-i, n)), FresnelPower);

  gl_Position = MVPMatrix * MCvertex;
}
```

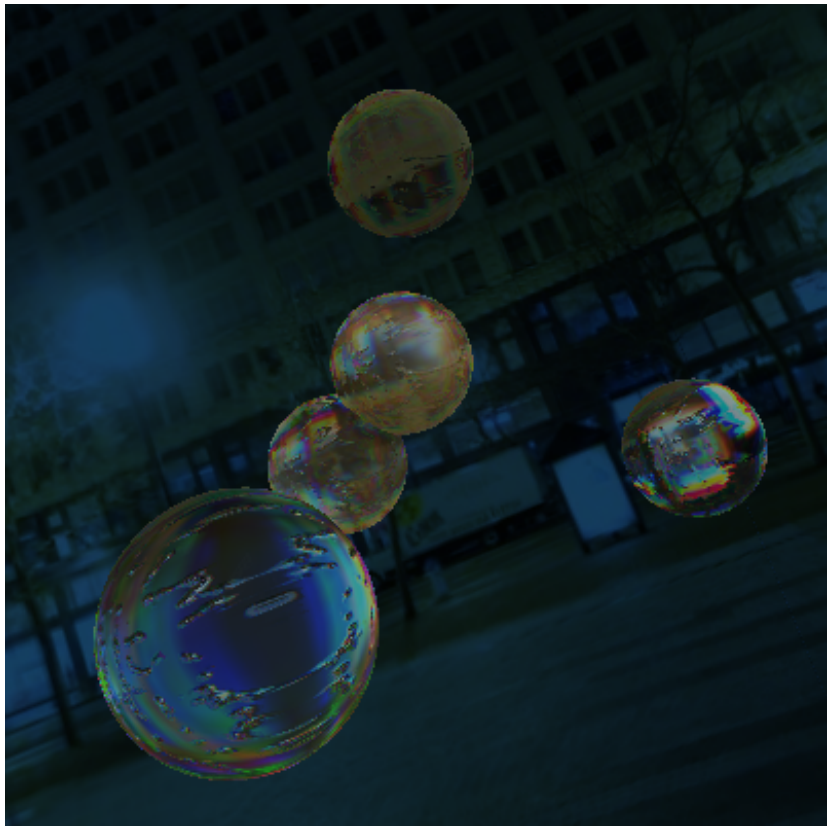
Fragment Shader

```
#version 140

uniform samplerCube Cubemap;

in vec3 Reflect;
in vec3 Refract;
in float Ratio;

out vec4 FragColor;

void main()
{
  vec3 refractColor = vec3(texture(Cubemap, Refract));

  vec3 reflectColor = vec3(texture(Cubemap, Reflect));

  vec3 color = mix(refractColor, reflectColor, Ratio);

  FragColor = vec4(color, 1.0);
}
```

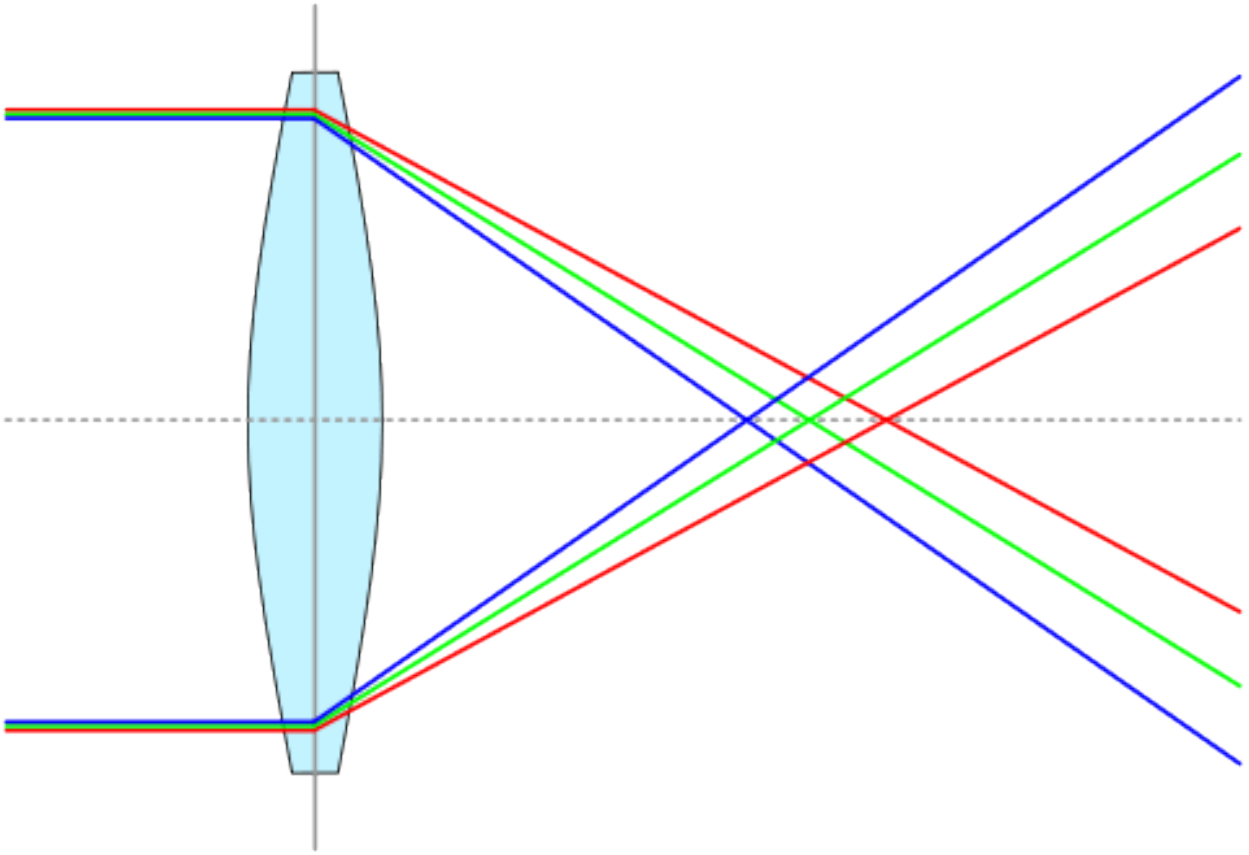
Output using Fresnel Term

Chromatic Aberration

- Differing wavelengths of light are refracted in different amounts
- Presence of rainbows, or “fringes” in glass objects



For RGB “wavelengths”



Implementation

- Choose different **coefficient of refraction** for each of the R,G and B components
  - Typical values (for glass = 0.66)
    - R = 0.65
    - G = 0.67
    - B = 0.69
- In fragment shader
  - Use three lookups into the environment map for refraction
  - One lookup for reflection
  - Use the ratio obtained by Fresnel term to “mix” the two colors together (use GLSL function mix())

Vertex Shader

```
#version 140
…
const float EtaR = 0.65;
const float EtaG = 0.67; // Ratio of indices of refraction
const float EtaB = 0.69;
const float FresnelPower = 5.0;
const float F = ((1.0-EtaG) * (1.0-EtaG)) / ((1.0+EtaG) * (1.0+EtaG));

out vec3 Reflect;
out vec3 RefractR;
out vec3 RefractG;
out vec3 RefractB;
out float Ratio;

void main()
{
…
 Ratio = F + (1.0 - F) * pow((1.0 - dot(-i, n)), FresnelPower);
 RefractR = refract(i, n, EtaR);
 RefractR = vec3(TextureMatrix * vec4(RefractR, 1.0));

 RefractG = refract(i, n, EtaG);
 RefractG = vec3(TextureMatrix * vec4(RefractG, 1.0));

 RefractB = refract(i, n, EtaB);
 RefractB = vec3(TextureMatrix * vec4(RefractB, 1.0));

 Reflect = reflect(i, n);
 Reflect = vec3(TextureMatrix * vec4(Reflect, 1.0));

 gl_Position = MVPMatrix * MCVertex;
}
```

Fragment Shader

```
#version 140

uniform samplerCube Cubemap;

in vec3 Reflect;
in vec3 RefractR;
in vec3 RefractG;
in vec3 RefractB;
in float Ratio;

out vec4 FragColor;

void main()
{
 vec3 refractColor, reflectColor;

 refractColor.r = vec3(texture(Cubemap, RefractR)).r;

 refractColor.g = vec3(texture(Cubemap, RefractG)).g;

 refractColor.b = vec3(texture(Cubemap, RefractB)).b;

 reflectColor = vec3(texture(Cubemap, Reflect));

 vec3 color = mix(refractColor, reflectColor, Ratio);
 FragColor = vec4(color, 1.0);
}
```

Output with Chromatic Abberation

Compare

References

- http://www.wikipedia.org
- OpenGL Shading Language – "Orange" Book
  - 3rd edition (Shader program code snippets)