8.2 Specular Reflection and Transmission

The behavior of light at perfectly smooth surfaces is relatively easy to characterize analytically using both the physical and geometric optics models. These surfaces exhibit perfect specular reflection and transmission of incident light; for a given ω_i direction, all light is scattered in a single outgoing direction ω_o. For specular reflection, this direction is the outgoing direction that makes the same angle with the normal as the incoming direction:

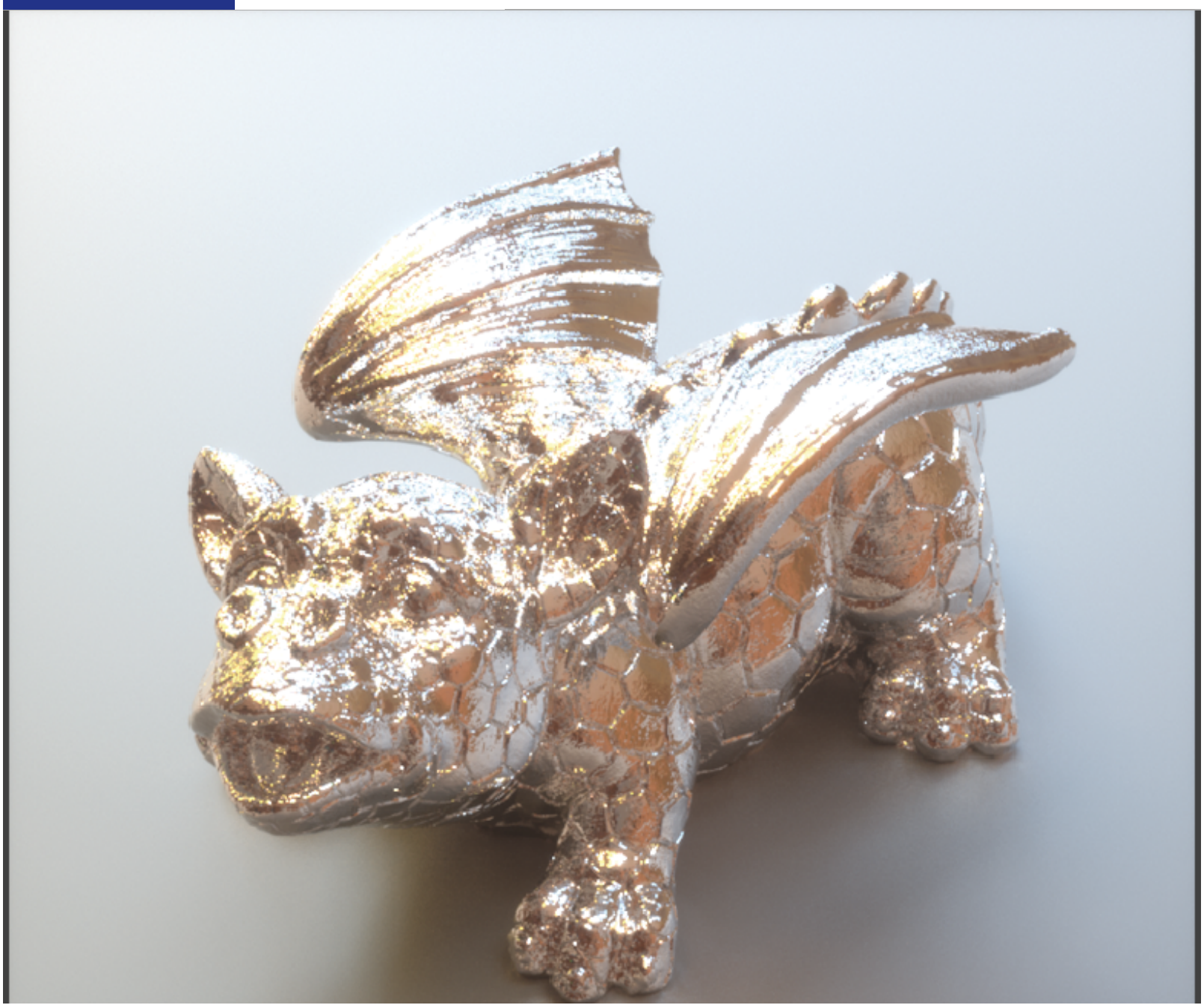
$$\theta_\mathrm{i} = \theta_\mathrm{o},$$

and where $\phi_\mathrm{o} = \phi_\mathrm{i} + \pi$. For transmission, we again have $\phi_\mathrm{o} = \phi_\mathrm{i} + \pi$, and the outgoing direction θ_t is given by *Snell's law*, which relates the angle θ_t between the transmitted direction and the surface normal $\mathbf{n}$ to the angle θ_i between the incident ray and the surface normal $\mathbf{n}$. (One of the exercises at the end of this chapter is to derive Snell's law using Fermat's principle from optics.) Snell's law is based on the *index of refraction* for the medium that the incident ray is in and the index of refraction for the medium it is entering. The index of refraction describes how much more slowly light travels in a particular medium than in a vacuum. We will use the Greek letter η, pronounced "eta," to denote the index of refraction. Snell's law is

$$\eta_\mathrm{i} \sin\theta_\mathrm{i} = \eta_\mathrm{t} \sin\theta_\mathrm{t}. \tag{8.2}$$

In general, the index of refraction varies with the wavelength of light. Thus, incident light generally scatters in multiple directions at the boundary between two different media, an effect known as *dispersion*. This effect can be seen when incident white light is split into spectral components by a prism. Common practice in graphics is to ignore this wavelength dependence, since this effect is generally not crucial for visual accuracy and ignoring it simplifies light transport calculations substantially. Alternatively, the paths of multiple beams of light (e.g., at a series of discrete wavelengths) can be tracked through the environment in which a dispersive object is found. The "Further Reading" section at the end of Chapter 14 has pointers to more information on this topic.

Specular Reflection [1] Specular Transmission



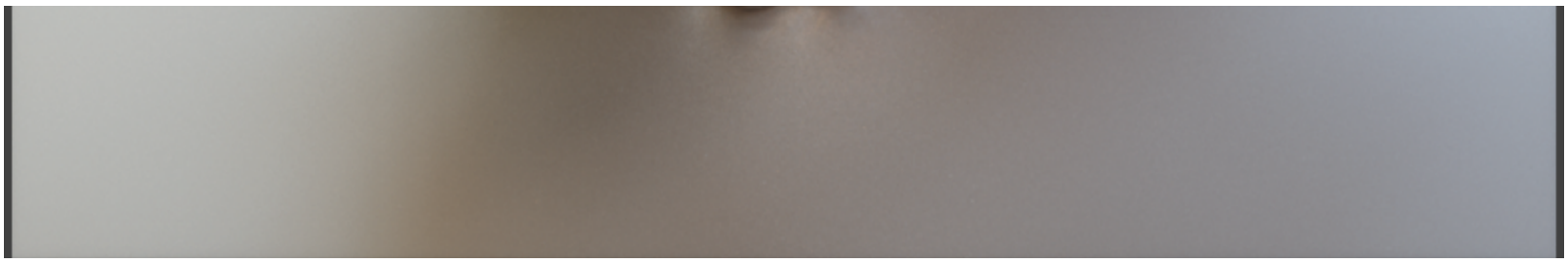


Figure 8.4: Dragon model rendered with (1) perfect specular reflection and (2) perfect specular refraction. Image (2) excludes the effects of external and internal reflection; the resulting energy loss produces conspicuous dark regions. *(Model courtesy of Christian Schüller.)*

Figure 8.4 shows the effect of perfect specular reflection and transmission.

8.2.1 Fresnel Reflectance

In addition to the reflected and transmitted directions, it is also necessary to compute the fraction of incoming light that is reflected or transmitted. For physically accurate reflection or refraction, these terms are directionally dependent and cannot be captured by constant per-surface scaling amounts. The *Fresnel equations* describe the amount of light reflected from a surface; they are the solution to Maxwell's equations at smooth surfaces.

Given the index of refraction and the angle which the incident ray makes with the surface normal, the Fresnel equations specify the material's corresponding reflectance for two different polarization states of the incident illumination. Because the visual effect of polarization is limited in most environments, in `pbrt` we will make the common assumption that light is unpolarized; that is, it is randomly oriented with respect to the light wave. With this simplifying assumption, the Fresnel reflectance is the average of the squares of the parallel and perpendicular polarization terms.

At this point, it is necessary to draw a distinction among several important classes of materials:

1. The first class is *dielectrics*, which are materials that don't conduct electricity. They have real-valued indices of refraction (usually in the range 1-3) and transmit† a portion of the incident illumination. Examples of dielectrics are glass, mineral oil, water, and air.
2. The second class consists of *conductors* such as metals. Valence electrons can freely move within the their atomic lattice, allowing electric currents to flow from one place to another. This fundamental atomic property translates into a profoundly different behavior when a conductor is subjected to electromagnetic radiation such as visible light: the material is opaque and reflects back a significant portion of the illumination. A portion of the light is also transmitted into the interior of the conductor, where it is rapidly absorbed: total absorption typically occurs within the top 0.1 μm of the material, hence only extremely thin metal films are capable of transmitting appreciable amounts of light. We ignore this effect in `pbrt` and only model the reflection component of conductors. In contrast to dielectrics, conductors have a complex-valued index of refraction $\bar{\eta} = \eta + \mathrm{i}k$.
3. Semiconductors such as silicon or germanium are the third class though we will not consider them in this book.

Both conductors and dielectrics are governed by the same set of Fresnel equations. Despite this, we prefer to create a special evaluation function for dielectrics to benefit from the particularly simple form that these equations take on when the indices of refraction are guaranteed to be real-valued.

Table 8.1: Indices of refraction for a variety of objects, giving the ratio of the speed of light in a vacuum to the speed of light in the medium. These are generally wavelength-dependent quantities; these values are averages over the visible wavelengths.

Medium	**Index of refraction η**
Vacuum	1.0
Air at sea level	1.00029
Ice	1.31
Water $(20^\circ\mathrm{C})$	1.333

Medium	Index of refraction η
Fused quartz	1.46
Glass	1.5–1.6
Sapphire	1.77
Diamond	2.42

To compute the Fresnel reflectance at the interface of two dielectric media, we need to know the indices of refraction for the two media. Table 8.1 has the indices of refraction for a number of dielectric materials. The Fresnel reflectance formulae for dielectrics are

$$r_{\parallel} = \frac{\eta_t \cos\theta_i - \eta_i \cos\theta_t}{\eta_t \cos\theta_i + \eta_i \cos\theta_t},$$
$$r_{\perp} = \frac{\eta_i \cos\theta_i - \eta_t \cos\theta_t}{\eta_i \cos\theta_i + \eta_t \cos\theta_t},$$

where $r_{\parallel}$ is the Fresnel reflectance for parallel polarized light and $r_{\perp}$ is the reflectance for perpendicular polarized light, η_i and η_t are the indices of refraction for the incident and transmitted media, and ω_i and ω_t are the incident and transmitted directions. ω_t can be computed with Snell's law (see Section 8.2.3).

The cosine terms should all be greater than or equal to zero; for the purposes of computing these values, the geometric normal should be flipped to be on the same side as ω_i and ω_t when computing $\cos\theta_i$ and $\cos\theta_t$, respectively.

For unpolarized light, the Fresnel reflectance is

$$F_r = \frac{1}{2}(r_{\parallel}^2 + r_{\perp}^2).$$

Due to conservation of energy, the energy transmitted by a dielectric is $1 - F_r$.

The function `FrDielectric()` computes the Fresnel reflection formula for dielectric materials and unpolarized light. The quantity $\cos\theta_i$ is passed in with the parameter `cosThetaI`.

<<BxDF Utility Functions>>=
```
Float FrDielectric(Float cosThetaI, Float etaI, Float etaT) {
    cosThetaI = Clamp(cosThetaI, -1, 1);
    <<Potentially swap indices of refraction>> ⊞
    <<Compute cosThetaT using Snell's law>> ⊞
    Float Rparl = ((etaT * cosThetaI) - (etaI * cosThetaT)) /
                  ((etaT * cosThetaI) + (etaI * cosThetaT));
    Float Rperp = ((etaI * cosThetaI) - (etaT * cosThetaT)) /
                  ((etaI * cosThetaI) + (etaT * cosThetaT));
    return (Rparl * Rparl + Rperp * Rperp) / 2;
}
```

To find the cosine of the transmitted angle, `cosThetaT`, it is first necessary to determine if the incident direction is on the outside of the medium or inside it, so that the two indices of refraction can be interpreted appropriately.

The sign of the cosine of the incident angle indicates on which side of the medium the incident ray lies (Figure 8.5). If the cosine is between 0 and 1, the ray is on the outside, and if the cosine is between -1 and 0, the ray is on the inside. The parameters `etaI` and `etaT` are adjusted such that `etaI` has the index of refraction of the incident medium, and thus it is ensured that `cosThetaI` is nonnegative.

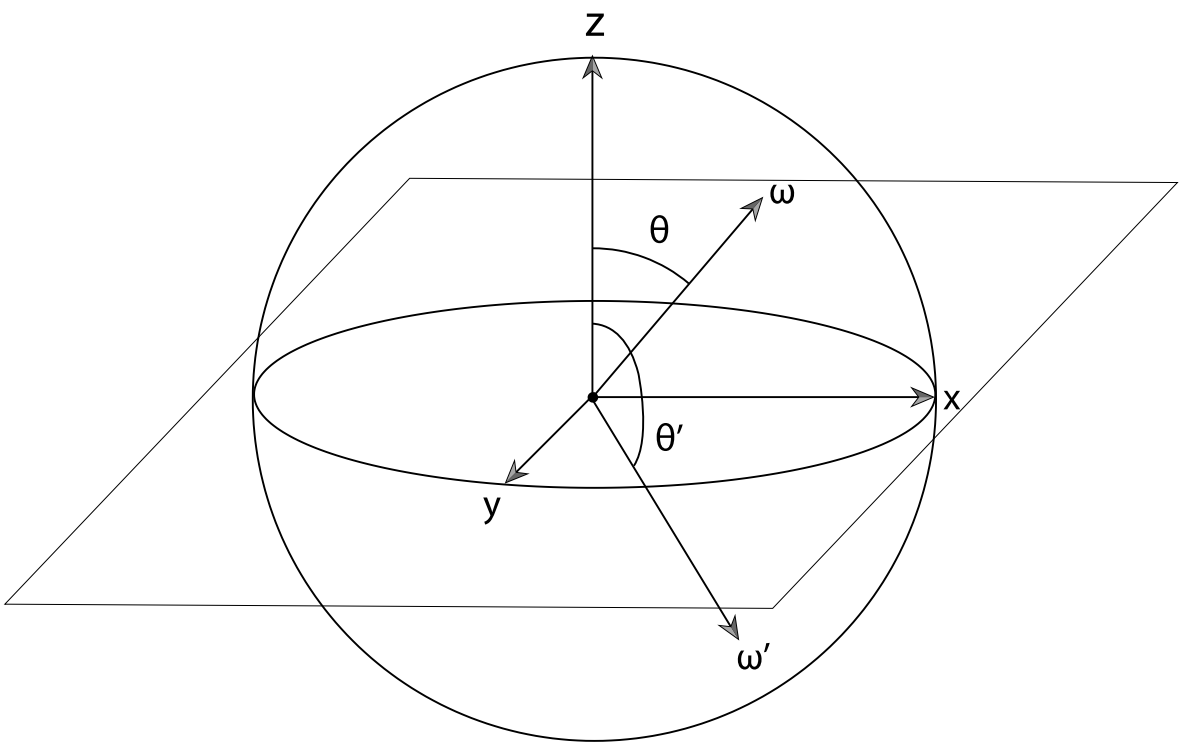


Figure 8.5: The cosine of the angle θ between a direction ω and the geometric surface normal indicates whether the direction is pointing outside the surface (in the same hemisphere as the normal) or inside the surface. In the standard reflection coordinate system, this test just requires checking the z component of the direction vector. Here, ω is in the upper hemisphere, with a positive-valued cosine, while ω' is in the lower hemisphere.

<<Potentially swap indices of refraction>>=
```
bool entering = cosThetaI > 0.f;
if (!entering) {
    std::swap(etaI, etaT);
    cosThetaI = std::abs(cosThetaI);
}
```

Once the indices of refraction are determined, we can compute the sine of the angle between the transmitted direction and the surface normal, $\sin\theta_t$, using Snell's law (Equation (8.2)). Finally, the cosine of this angle is found using the identity $\sin^2\theta + \cos^2\theta = 1$.

<<Compute cosThetaT using Snell's law>>=
```
Float sinThetaI = std::sqrt(std::max((Float)0,
                                     1 - cosThetaI * cosThetaI));
Float sinThetaT = etaI / etaT * sinThetaI;
<<Handle total internal reflection>>
Float cosThetaT = std::sqrt(std::max((Float)0,
                                     1 - sinThetaT * sinThetaT));
```

When light is traveling from one medium to another medium with a lower index of refraction, none of the light at incident angles near grazing passes into the other medium. The largest angle at which this happens is called the *critical angle*; when θ_i is greater than the critical angle, *total internal reflection* occurs, and all of the light is reflected. That case is detected here by a value of $\sin\theta_t$ greater than one; in that case, the Fresnel equations are unnecessary.

<<Handle total internal reflection>>=
```
if (sinThetaT >= 1)
    return 1;
```

We now focus on the general case of a complex index of refraction $\bar{\eta} = \eta + ik$, where some of the incident light is potentially absorbed by the material and turned into heat. In addition to the real part, the general Fresnel formula now also depends on the imaginary part k that is referred to as the *absorption coefficient*.

Figure 8.6 shows a plot of the index of refraction and absorption coefficient for gold; both of these are wavelength-dependent quantities. The directory `scenes/spds/metals` in the `pbrt` distribution has wavelength-dependent data for η and k for a variety of metals. Figure 9.4 in the next chapter shows a model rendered with a metal material.

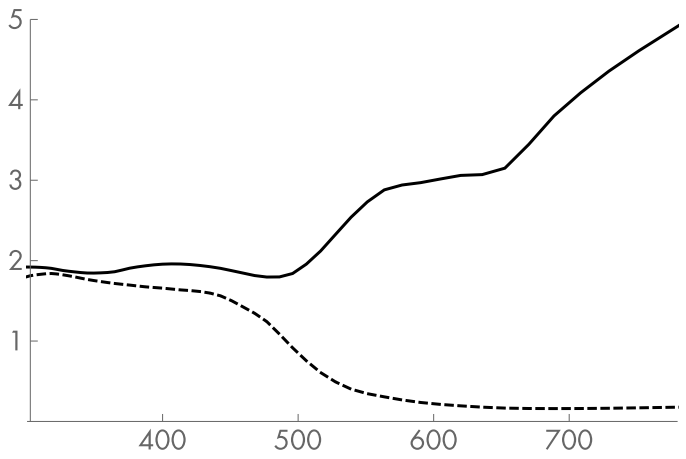


Figure 8.6: Absorption Coefficient and Index of Refraction of Gold. This plot shows the spectrally varying values of the absorption coefficient k (solid line) and the index of refraction η (dashed line) for gold, where the horizontal axis is wavelength in nm.

The Fresnel reflectance at the boundary between a conductor and a dielectric medium is given by

$$r_\perp = \frac{a^2 + b^2 - 2a\cos\theta + \cos^2\theta}{a^2 + b^2 + 2a\cos\theta + \cos^2\theta},$$

$$r_\parallel = r_\perp \frac{\cos^2\theta(a^2 + b^2) - 2a\cos\theta\sin^2\theta + \sin^4\theta}{\cos^2\theta(a^2 + b^2) + 2a\cos\theta\sin^2\theta + \sin^4\theta}, \tag{8.3}$$

where

$$a^2 + b^2 = \sqrt{(\eta^2 - k^2 - \sin^2\theta)^2 + 4\eta^2 k^2},$$

and $\eta + \mathrm{i}k = \bar{\eta}_\mathrm{t}/\bar{\eta}_\mathrm{i}$ is the relative index of refraction computed using a complex division operation. However, generally $\bar{\eta}_\mathrm{i}$ will be a dielectric so that a normal real division can be used instead.

This computation is implemented by the `FrConductor()` function[†] ; its implementation corresponds directly to Equation (8.3) and so isn't included here.

<<Reflection Declarations>>=
```
Spectrum FrConductor(Float cosThetaI, const Spectrum &etaI,
    const Spectrum &etaT, const Spectrum &k);
```

For convenience, we will define an abstract `Fresnel` class that provides an interface for computing Fresnel reflection coefficients. Using implementations of this interface helps simplify the implementation of subsequent BRDFs that may need to support both forms.

<<BxDF Declarations>>+=
```
class Fresnel {
public:
    <<Fresnel Interface>>
};
```

The only method provided by the `Fresnel` interface is `Fresnel::Evaluate()`. Given the cosine of the angle made by the incoming direction and the surface normal, it returns the amount of light reflected by the surface.

<<Fresnel Interface>>=
```
virtual Spectrum Evaluate(Float cosI) const = 0;
```

Fresnel Conductors

`FresnelConductor` implements this interface for conductors.

<<BxDF Declarations>>+=
```
class FresnelConductor : public Fresnel {
public:
    <<FresnelConductor Public Methods>>
private:
```

```
    Spectrum etaI, etaT, k;
};
```

Its constructor stores the given index of refraction η and absorption coefficient k.

<<FresnelConductor Public Methods>>=
```
FresnelConductor(const Spectrum &etaI, const Spectrum &etaT,
    const Spectrum &k) : etaI(etaI), etaT(etaT), k(k) { }
```

The evaluation routine for FresnelConductor is also simple; it just calls the FrConductor() function defined earlier. Note that it takes the absolute value of cosThetaI before calling FrConductor(), since FrConductor() expects that the cosine will be measured with respect to the normal on the same side of the surface as ω_i, or, equivalently, that the absolute value of $\cos\theta_\mathrm{i}$ should be used.

<<BxDF Method Definitions>>+=
```
Spectrum FresnelConductor::Evaluate(Float cosThetaI) const {
    return FrConductor(std::abs(cosThetaI), etaI, etaT, k);
}
```

Fresnel Dielectrics

FresnelDielectric similarly implements the Fresnel interface for dielectric materials.

<<BxDF Declarations>>+=
```
class FresnelDielectric : public Fresnel {
public:
    <<FresnelDielectric Public Methods>> ⊞
private:
    Float etaI, etaT;
};
```

Its constructor stores the indices of refraction on the exterior and interior sides of the surface.

<<FresnelDielectric Public Methods>>=
```
FresnelDielectric(Float etaI, Float etaT) : etaI(etaI), etaT(etaT) { }
```

The evaluation routine for FresnelDielectric analogously calls FrDielectric().

<<BxDF Method Definitions>>+=
```
Spectrum FresnelDielectric::Evaluate(Float cosThetaI) const {
    return FrDielectric(cosThetaI, etaI, etaT);
}
```

A Special Fresnel Interface

The FresnelNoOp implementation of the Fresnel interface returns 100% reflection for all incoming directions. Although this is physically implausible, it is a convenient capability to have available.

<<BxDF Declarations>>+=
```
class FresnelNoOp : public Fresnel {
public:
    Spectrum Evaluate(Float) const { return Spectrum(1.); }
};
```

8.2.2 Specular Reflection

We can now implement the SpecularReflection class, which describes physically plausible specular reflection, using the Fresnel interface to compute the fraction of light that is reflected. First, we will derive the BRDF that describes specular reflection. Since the Fresnel equations give the fraction of light reflected, $F_\mathrm{r}(\omega)$, then we need a BRDF such that

$$L_\mathrm{o}(\omega_\mathrm{o}) = \int f_\mathrm{r}(\omega_\mathrm{o}, \omega_\mathrm{i})\, L_\mathrm{i}(\omega_\mathrm{i})\, |\cos\theta_i|\, \mathrm{d}\omega_\mathrm{i} = F_\mathrm{r}(\omega_\mathrm{r})\, L_\mathrm{i}(\omega_\mathrm{r}),$$

where $\omega_\mathrm{r} = \mathrm{R}(\omega_\mathrm{o}, \mathbf{n})$ is the specular reflection vector for ω_o reflected about the surface normal $\mathbf{n}$. (Recall that $\theta_\mathrm{r} = \theta_\mathrm{o}$ for specular reflection, and therefore $F_\mathrm{r}(\omega_\mathrm{o}) = F_\mathrm{r}(\omega_\mathrm{r})$.)

Such a BRDF can be constructed using the Dirac delta distribution. Recall from Section 7.1 that the delta distribution has the useful property that

$$\int f(x)\,\delta(x - x_0)\,\mathrm{d}x = f(x_0). \tag{8.4}$$

The delta distribution requires special handling compared to standard functions, however. In particular, numerical integration of integrals with delta distributions must explicitly account for the delta distribution. Consider the integral in Equation (8.4): if we tried to evaluate it using the trapezoid rule or some other numerical integration technique, by definition of the delta distribution there would be zero probability that any of the evaluation points x_i would have a nonzero value of $\delta(x_i)$. Rather, we must allow the delta distribution to determine the evaluation point itself. We will encounter delta distributions in light transport integrals both from specular BxDFs as well as from some of the light sources in Chapter 12.

Intuitively, we want the specular reflection BRDF to be zero everywhere except at the perfect reflection direction, which suggests the use of the delta distribution. A first guess might be to use delta functions to restrict the incident direction to the specular reflection direction ω_r. This would yield a BRDF of

$$f_\mathrm{r}(\omega_\mathrm{o}, \omega_\mathrm{i}) = \delta(\omega_\mathrm{i} - \omega_\mathrm{r}) F_\mathrm{r}(\omega_\mathrm{i}).$$

Although this seems appealing, plugging it into the scattering equation, Equation (5.9), reveals a problem:

$$\begin{aligned} L_\mathrm{o}(\omega_\mathrm{o}) &= \int \delta(\omega_\mathrm{i} - \omega_\mathrm{r}) F_\mathrm{r}(\omega_\mathrm{i}) L_\mathrm{i}(\omega_\mathrm{i}) |\cos\theta_\mathrm{i}|\,\mathrm{d}\omega_\mathrm{i} \\ &= F_\mathrm{r}(\omega_\mathrm{r}) L_\mathrm{i}(\omega_\mathrm{r}) |\cos\theta_\mathrm{r}|. \end{aligned}$$

This is not correct because it contains an extra factor of $\cos\theta_\mathrm{r}$. However, we can divide out this factor to find the correct BRDF for perfect specular reflection:

$$f_\mathrm{r}(\mathrm{p}, \omega_\mathrm{o}, \omega_\mathrm{i}) = F_\mathrm{r}(\omega_\mathrm{r}) \frac{\delta(\omega_\mathrm{i} - \omega_\mathrm{r})}{|\cos\theta_\mathrm{r}|},$$

<<BxDF Declarations>>+=

```
class SpecularReflection : public BxDF {
public:
    <<SpecularReflection Public Methods>> ⊞
private:
    <<SpecularReflection Private Data>> ⊞
};
```

The SpecularReflection constructor takes a Spectrum that is used to scale the reflected color and a Fresnel object pointer that describes dielectric or conductor Fresnel properties.

<<SpecularReflection Public Methods>>=

```
SpecularReflection(const Spectrum &R, Fresnel *fresnel)
    : BxDF(BxDFType(BSDF_REFLECTION | BSDF_SPECULAR)), R(R),
      fresnel(fresnel) { }
```

<<SpecularReflection Private Data>>=

```
const Spectrum R;
const Fresnel *fresnel;
```

The rest of the implementation is straightforward. No scattering is returned from SpecularReflection::f(), since for an arbitrary pair of directions the delta function returns no scattering. †

<<SpecularReflection Public Methods>>+=

```
Spectrum f(const Vector3f &wo, const Vector3f &wi) const {
    return Spectrum(0.f);
}
```

However, we do implement the `Sample_f()` method, which selects an appropriate direction according to the delta distribution. It sets the output variable `wi` to be the reflection of the supplied direction `wo` about the surface normal. The `*pdf` value is set to be one; Section 14.1.3 discusses some subtleties about the mathematical quantity that this value of one represents.

<<BxDF Method Definitions>>+=

```
Spectrum SpecularReflection::Sample_f(const Vector3f &wo,
        Vector3f *wi, const Point2f &sample, Float *pdf,
        BxDFType *sampledType) const {
    <<Compute perfect specular reflection direction>> ⊞
    *pdf = 1;
    return fresnel->Evaluate(CosTheta(*wi)) * R / AbsCosTheta(*wi);
}
```

The desired incident direction is the reflection of ω_o around the surface normal, $\mathrm{R}(\omega_o, \mathbf{n})$. This direction can be computed fairly easily using vector geometry. First, note that the incoming direction, the reflection direction, and the surface normal all lie in the same plane. We can decompose vectors ω that lie in a plane into a sum of two components: one parallel to $\mathbf{n}$, which we'll denote by $\omega_\parallel$, and one perpendicular, $\omega_\perp$.

These vectors are easily computed: if $\mathbf{n}$ and ω are normalized, then $\omega_\parallel$ is $(\cos\theta)\mathbf{n} = (\mathbf{n}\cdot\omega)\mathbf{n}$ (Figure 8.7). Because $\omega_\parallel + \omega_\perp = \omega$,

$$\omega_\perp = \omega - \omega_\parallel = \omega - (\mathbf{n}\cdot\omega)\mathbf{n}.$$

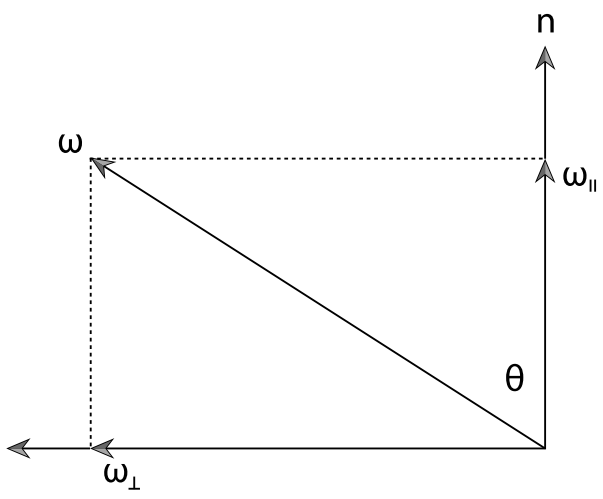


Figure 8.7: The parallel projection of a vector ω on to the normal $\mathbf{n}$ is given by $\omega_\parallel = (\cos\theta)\mathbf{n} = (\mathbf{n}\cdot\omega)\mathbf{n}$. The perpendicular component is given by $\omega_\perp = (\sin\theta)\mathbf{n}$ but is more easily computed by $\omega_\perp = \omega - \omega_\parallel$.

Figure 8.8 shows the setting for computing the reflected direction ω_r. We can see that both vectors have the same $\omega_\parallel$ component, and the value of $\omega_{\mathrm{r}\perp}$ is the negation of $\omega_{\mathrm{o}\perp}$. Therefore, we have

$$\begin{aligned}\omega_\mathrm{r} = \omega_{\mathrm{r}\perp} + \omega_{\mathrm{r}\parallel} &= -\omega_{\mathrm{o}\perp} + \omega_{\mathrm{o}\parallel} \\ &= -(\omega_\mathrm{o} - (\mathbf{n}\cdot\omega_\mathrm{o})\mathbf{n}) + (\mathbf{n}\cdot\omega_\mathrm{o})\mathbf{n} \\ &= -\omega_\mathrm{o} + 2(\mathbf{n}\cdot\omega_\mathrm{o})\mathbf{n}.\end{aligned} \tag{8.5}$$

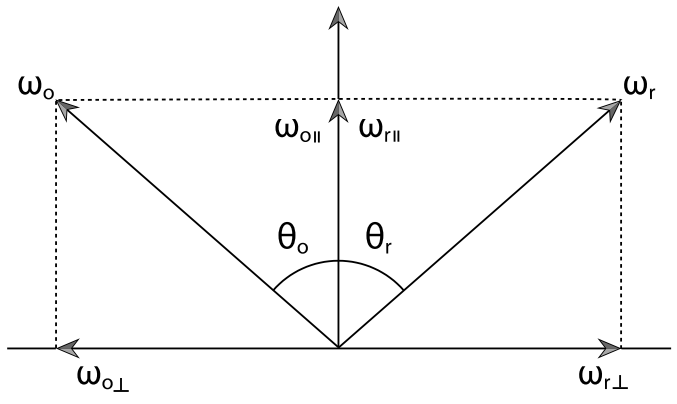


Figure 8.8: Because the angles θ_o and θ_r are equal, the parallel component of the perfect reflection direction $\omega_{\mathrm{r}\parallel}$ is the same as the incident direction's: $\omega_{\mathrm{r}\parallel} = \omega_{\mathrm{o}\parallel}$. Its perpendicular component is just the

incident direction's perpendicular component, negated.

The `Reflect()` function implements this computation.

<<BSDF Inline Functions>>+=
```
inline Vector3f Reflect(const Vector3f &wo, const Vector3f &n) {
    return -wo + 2 * Dot(wo, n) * n;
}
```

In the BRDF coordinate system, $\mathbf{n} = (0, 0, 1)$, and this expression is substantially simpler.

<<Compute perfect specular reflection direction>>=
```
*wi = Vector3f(-wo.x, -wo.y, wo.z);
```

8.2.3 Specular Transmission

We will now derive the BTDF for specular transmission. Snell's law is the basis of the derivation. Not only does Snell's law give the direction for the transmitted ray, but it can also be used to show that radiance along a ray changes as the ray goes between media with different indices of refraction.

Consider incident radiance arriving at the boundary between two media, with indices of refraction η_i and η_o for the incoming and outgoing media, respectively (Figure 8.9).

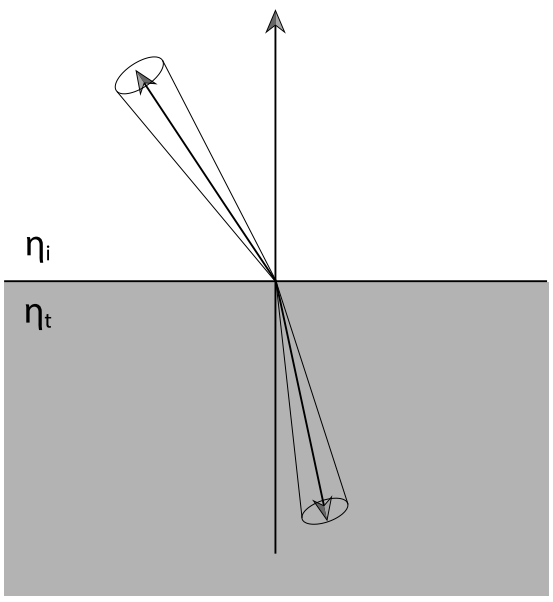


Figure 8.9: The amount of transmitted radiance at the boundary between media with different indices of refraction is scaled by the squared ratio of the two indices of refraction. Intuitively, this can be understood as the result of the radiance's differential solid angle being compressed or expanded as a result of transmission.

We use τ to denote the fraction of incident energy that is transmitted to the outgoing direction as given by the Fresnel equations, so $\tau = 1 - F_\mathrm{r}(\omega_\mathrm{i})$. The amount of transmitted differential flux, then, is

$$\mathrm{d}\Phi_\mathrm{o} = \tau \mathrm{d}\Phi_\mathrm{i}.$$

If we use the definition of radiance, Equation (5.2), we have

$$L_\mathrm{o} \cos\theta_\mathrm{o} \,\mathrm{d}A \,\mathrm{d}\omega_\mathrm{o} = \tau (L_\mathrm{i} \cos\theta_\mathrm{i} \,\mathrm{d}A \,\mathrm{d}\omega_\mathrm{i}).$$

Expanding the solid angles to spherical angles, we have

$$L_\mathrm{o} \cos\theta_\mathrm{o} \,\mathrm{d}A \sin\theta_\mathrm{o} \,\mathrm{d}\theta_\mathrm{o} \,\mathrm{d}\phi_\mathrm{o} = \tau L_\mathrm{i} \cos\theta_\mathrm{i} \,\mathrm{d}A \sin\theta_\mathrm{i} \,\mathrm{d}\theta_\mathrm{i} \,\mathrm{d}\phi_\mathrm{i}. \quad (8.6)$$

We can now differentiate Snell's law with respect to θ, which gives the relation

$$\eta_\mathrm{o} \cos\theta_\mathrm{o} \,\mathrm{d}\theta_\mathrm{o} = \eta_\mathrm{i} \cos\theta_\mathrm{i} \,\mathrm{d}\theta_\mathrm{i}.$$

Rearranging terms, we have

$$\frac{\cos\theta_o \, d\theta_o}{\cos\theta_i \, d\theta_i} = \frac{\eta_i}{\eta_o}.$$

Substituting this relationship and Snell's law into Equation (8.6) and then simplifying, we have

$$L_o \, \eta_i^2 \, d\phi_o = \tau L_i \, \eta_o^2 \, d\phi_i.$$

Because $\phi_i = \phi_o + \pi$ and therefore $d\phi_i = d\phi_o$, we have the final relationship:

$$L_o = \tau L_i \frac{\eta_o^2}{\eta_i^2}. \tag{8.7}$$

As with the BRDF for specular reflection, we need to divide out a $\cos\theta_i$ term to get the right BTDF for specular transmission:

$$f_r(\omega_o, \omega_i) = \frac{\eta_o^2}{\eta_i^2}(1 - F_r(\omega_i)) \frac{\delta(\omega_i - \mathrm{T}(\omega_o, \mathbf{n}))}{|\cos\theta_i|},$$

where $\mathrm{T}(\omega_o, \mathbf{n})$ is the specular transmission vector that results from specular transmission of ω_o through an interface with surface normal $\mathbf{n}$.

The $1 - F_r(\omega_i)$ term in this equation corresponds to an easily observed effect: transmission is stronger at near-perpendicular angles. For example, if you look straight down into a clear lake, you can see far into the water, but at grazing angles most of the light is reflected as if from a mirror.

The `SpecularTransmission` class is almost exactly the same as `SpecularReflection` except that the sampled direction is the direction for perfect specular transmission. Figure 8.10 shows an image of the dragon model using specular reflection and transmission BRDF and BTDF to model glass.

Figure 8.10: When the BRDF for specular reflection and the BTDF for specular transmission are modulated with the Fresnel formula for dielectrics, the realistic angle-dependent variation of the amount of reflection and transmission gives a visually accurate representation of the glass. *(Model courtesy of Christian Schüller.)*

<<BxDF Declarations>>+=
```
class SpecularTransmission : public BxDF {
public:
    <<SpecularTransmission Public Methods>>
private:
    <<SpecularTransmission Private Data>>
};
```

The `SpecularTransmission` constructor stores the indices of refraction on both sides of the surface, where `etaA` is the index of refraction above the surface (where the side the surface normal lies in is "above"), `etaB` is the index of refraction below the surface, and `T` gives a transmission scale factor. The `TransportMode` parameter indicates whether the incident ray that intersected the point where the `BxDF` was computed started from a light source or whether it was started from the camera. This distinction has implications for how the `BxDF`'s contribution is computed.

<<SpecularTransmission Public Methods>>=
```
SpecularTransmission(const Spectrum &T, Float etaA, Float etaB,
        TransportMode mode)
    : BxDF(BxDFType(BSDF_TRANSMISSION | BSDF_SPECULAR)), T(T), etaA(etaA),
      etaB(etaB), fresnel(etaA, etaB), mode(mode) {
}
```

Because conductors do not transmit light, a `FresnelDielectric` object is always used for the Fresnel computations.

<<SpecularTransmission Private Data>>=
```
const Spectrum T;
const Float etaA, etaB;
const FresnelDielectric fresnel;
const TransportMode mode;
```

As with `SpecularReflection`, zero is always returned from `SpecularTransmission::f()`, since the BTDF is a scaled delta distribution.

<<SpecularTransmission Public Methods>>+=
```
Spectrum f(const Vector3f &wo, const Vector3f &wi) const {
    return Spectrum(0.f);
}
```

Equation (8.7) describes how radiance changes as a ray passes from one medium to another. However, it turns out that while this scaling should be applied for rays starting at light sources, it must *not* be applied for rays starting from the camera. This issue is discussed in more detail in Section 16.1, and the fragment that applies this scaling, <<*Account for non-symmetry with transmission to different medium*>>, is defined there.

<<BxDF Method Definitions>>+=
```
Spectrum SpecularTransmission::Sample_f(const Vector3f &wo,
        Vector3f *wi, const Point2f &sample, Float *pdf,
        BxDFType *sampledType) const {
    <<Figure out which η is incident and which is transmitted>>
    bool entering = CosTheta(wo) > 0;
    Float etaI = entering ? etaA : etaB;
    Float etaT = entering ? etaB : etaA;

    <<Compute ray direction for specular transmission>>
    if (!Refract(wo, Faceforward(Normal3f(0, 0, 1), wo), etaI / etaT, wi))
        return 0;

    *pdf = 1;
```

```
    Spectrum ft = T * (Spectrum(1.) - fresnel.Evaluate(CosTheta(*wi)));
    <<Account for non-symmetry with transmission to different medium>>
       if (mode == TransportMode::Radiance)
           ft *= (etaI * etaI) / (etaT * etaT);

    return ft / AbsCosTheta(*wi);
}
```

The method first determines whether the incident ray is entering or exiting the refractive medium. `pbrt` uses the convention that the surface normal, and thus the $(0, 0, 1)$ direction in local reflection space, is oriented such that it points toward the outside of the object. Therefore, if the z component of the ω_o direction is greater than zero, the incident ray is coming from outside of the object.

<<Figure out which η is incident and which is transmitted>>=

```
bool entering = CosTheta(wo) > 0;
Float etaI = entering ? etaA : etaB;
Float etaT = entering ? etaB : etaA;
```

<<Compute ray direction for specular transmission>>=

```
if (!Refract(wo, Faceforward(Normal3f(0, 0, 1), wo), etaI / etaT, wi))
    return 0;
```

To derive the expression that gives the transmitted direction vector, we can follow a similar approach to the one we used earlier for specular reflection. Figure 8.11 shows the setting.

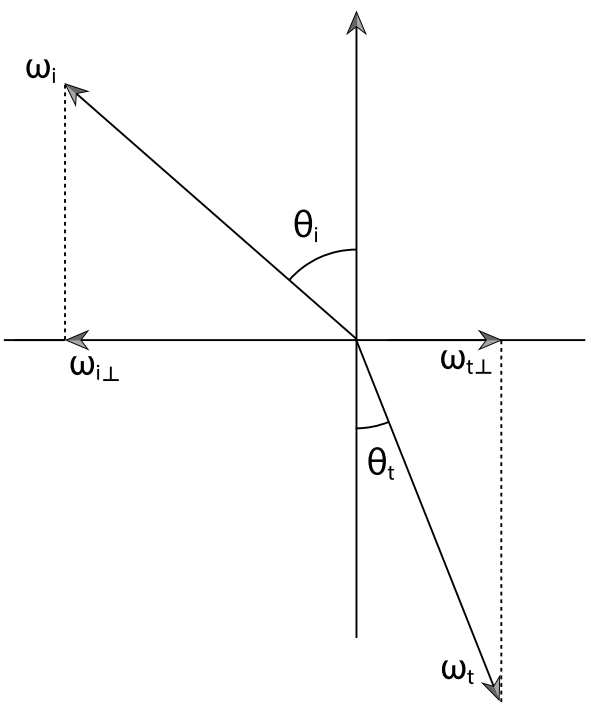


Figure 8.11: The Geometry of Specular Transmission. Given an incident direction ω_i and surface normal $\mathbf{n}$ with angle θ_i between them, the specularly transmitted direction makes an angle θ_t with the surface normal. This direction, ω_t, can be computed by using Snell's law to find its perpendicular component $\omega_{t\perp}$ and then computing the $\omega_{t\parallel}$ that gives a normalized result ω_t.

This time, however, we'll start with the perpendicular component: if the incident vector is normalized and has perpendicular component $\omega_{i\perp}$, then we know from trigonometry and the definition of $\omega_\perp$ that the length of $\omega_{i\perp}$ is equal to $\sin\theta_i$. Snell's law tells us that $\sin\theta_t = \eta_i/\eta_t \sin\theta_i$. Negating the direction of $\omega_{i\perp}$ and adjusting the length accordingly, we have

$$\omega_{t\perp} = \frac{\eta_i}{\eta_t}(-\omega_{i\perp}).$$

Equivalently, because $\omega_\perp = \omega - \omega_\parallel$,

$$\omega_{t\perp} = \frac{\eta_i}{\eta_t}\left(-\omega_i + (\omega_i \cdot \mathbf{n})\mathbf{n}\right).$$

Now for $\omega_{t\parallel}$: we know that $\omega_{t\parallel}$ is parallel to $\mathbf{n}$ but facing in the opposite direction, and we also know that ω_t should be normalized. Putting these together,

$$\omega_{t\parallel} = -\left(\sqrt{1 - \|\omega_{t\perp}\|^2}\right)\mathbf{n}.$$

The full vector ω_t, then, is

$$\omega_t = \omega_{t\perp} + \omega_{t\parallel} = \frac{\eta_i}{\eta_t}(-\omega_i) + \left[\frac{\eta_i}{\eta_t}(\omega_i \cdot \mathbf{n}) - \sqrt{1 - \|\omega_{t\perp}\|^2}\right]\mathbf{n}.$$

Because $\|\omega_{t\perp}\| = \sin\theta_t$, the term under the square root is $1 - \sin^2\theta_t = \cos^2\theta_t$, which gives the final result:

$$\omega_t = \frac{\eta_i}{\eta_t}(-\omega_i) + \left[\frac{\eta_i}{\eta_t}(\omega_i \cdot \mathbf{n}) - \cos\theta_t\right]\mathbf{n}. \tag{8.8}$$

The `Refract()` function computes the refracted direction `wt` given an incident direction `wi`, surface normal `n` in the same hemisphere as `wi`, and `eta`, the ratio of indices of refraction in the incident and transmitted media, respectively. The Boolean return value indicates whether a valid refracted ray was returned in `*wt`; it is `false` in the case of total internal reflection.

<<BSDF Inline Functions>>+=
```
inline bool Refract(const Vector3f &wi, const Normal3f &n, Float eta,
    Vector3f *wt) {
    <<Compute cos θ_t using Snell's law>>
    *wt = eta * -wi + (eta * cosThetaI - cosThetaT) * Vector3f(n);
    return true;
}
```

Squaring both sides of Snell's law lets us compute $\cos\theta_t$:

$$\eta_i^2 \sin^2\theta_i = \eta_t^2 \sin^2\theta_t \qquad \sin^2\theta_t = \frac{\eta_i^2}{\eta_t^2}\sin^2\theta_i$$

$$1 - \cos^2\theta_t = \frac{\eta_i^2}{\eta_t^2}\sin^2\theta_i \qquad \cos\theta_t = \sqrt{1 - \frac{\eta_i^2}{\eta_t^2}\sin^2\theta_i}$$

<<Compute $\cos\theta_t$ using Snell's law>>=
```
Float cosThetaI = Dot(n, wi);
Float sin2ThetaI = std::max(0.f, 1.f - cosThetaI * cosThetaI);
Float sin2ThetaT = eta * eta * sin2ThetaI;
<<Handle total internal reflection for transmission>>
Float cosThetaT = std::sqrt(1 - sin2ThetaT);
```

We need to handle the case of total internal reflection here as well. If the squared value of $\sin\theta_t$ is greater than or equal to one, total internal reflection has occurred, so `false` is returned.[†]

<<Handle total internal reflection for transmission>>=
```
if (sin2ThetaT >= 1) return false;
```

8.2.4 Fresnel-Modulated Specular Reflection and Transmission

For better efficiency in some of the Monte Carlo light transport algorithms to come in Chapters 14, 15, and 16, it's useful to have a single `BxDF` that represents both specular reflection and specular transmission together, where the relative weightings of the types of scattering are modulated by the dielectric Fresnel equations. Such a `BxDF` is provided in `FresnelSpecular`.

<<BxDF Declarations>>+=
```
class FresnelSpecular : public BxDF {
public:
    <<FresnelSpecular Public Methods>>
private:
    <<FresnelSpecular Private Data>>
};
```

<<FresnelSpecular Public Methods>>=

```
FresnelSpecular(const Spectrum &R, const Spectrum &T, Float etaA,
        Float etaB, TransportMode mode)
    : BxDF(BxDFType(BSDF_REFLECTION | BSDF_TRANSMISSION | BSDF_SPECULAR)),
      R(R), T(T), etaA(etaA), etaB(etaB), fresnel(etaA, etaB),
      mode(mode) { }
```

Since we only focus on the dielectric case, a FresnelDielectric object is always used for the Fresnel computations.

<<FresnelSpecular Private Data>>=

```
const Spectrum R, T;
const Float etaA, etaB;
const FresnelDielectric fresnel;
const TransportMode mode;
```

<<FresnelSpecular Public Methods>>+= ▲

```
Spectrum f(const Vector3f &wo, const Vector3f &wi) const {
    return Spectrum(0.f);
}
```

Because some of the implementation details depend on principles of Monte Carlo integration that are introduced in Chapter 13, the implementation of the `Sample_f()` method is in Section 14.1.3.