Crash Course in BRDF Implementation

by Jakub Boksansky

Figure 1 The rendering of BEEPLE Zero-Day scene [1] using BRDFs described in this blog. Rendered with 2 million rays per pixel using the Falcor path tracer [2].

1 Introduction

Creating photorealistic material models has been a topic of computer graphics research for decades. Many models have been created over the years – empirical, physically based and models based on data measured of physical surfaces. Early empirical models, such as Phong or Lambertian managed to provide plausible results, while being efficient to evaluate with limited computational resources. One of the keys to their longevity was their ability to reasonably reproduce a range of real-world materials, while also providing intuitive parameters for tuning by artists (as opposed to using, e.g., the physical units).

Advances in algorithms for efficient evaluation of light transport in the scene, such as path tracing, photon mapping, and other methods enabled us to evaluate complex materials based on multiple interactions of light with many surfaces in the scene (while implicitly rendering the effects such as caustics, indirect (global) illumination, soft shadows etc.). The light transport algorithm is responsible for establishing the paths that light travels in the scene, while the material model evaluates how light interacts with surfaces along the path. Formal model for this approach is described by the famous rendering equation [3], where the contribution of material on the surface is given by the bidirectional reflectance distribution function (BRDF).

In this article we look at implementation of some basic BRDFs commonly used in games and the theory behind them. A lot of existing BRDF code is highly optimized and uses many approximations and tricks, making it hard to understand and modify. To make it easier to study existing implementations, we provide code with explanations of inner workings and derivation of all discussed BRDFs. We will also discuss the usage of BRDFs for indirect lighting, e.g., in Monte Carlo path tracing,

where importance sampling and ensuring energy conservation is needed. These are not usually implemented in game engines as they were not needed for rasterization but are necessary for applications making use of emerging real-time ray tracing.

2 So, what is the BRDF?

The BRDF describes reflectance of the surface for given combination of incoming and outgoing light direction. In other words, it determines how much light is reflected in given direction when certain amount of light is incident from another direction, depending on properties of the surface. For example, for diffuse surfaces, the BRDF specifies small amount of reflection in all directions, while for mirrors, all the light is reflected in one direction. This formulation is very flexible as it enables us to encapsulate the response of surface material into a BRDF implementation and setting of its parameters, which is independent of underlying algorithm for light transport. Note that BRDF does not distinguish between direct and indirect incoming light, meaning it can be used to calculate contribution of both virtual lights placed in the scene (local illumination), and indirect light reflected one or more times from other surfaces (global illumination). This also means that BRDF is independent of the implementation of lights which can be developed and authored separately (BRDF only needs to know direction of incident light and its intensity at shaded point).

$$L(P, \omega_o) = L_e + \int_{\Omega} \boldsymbol{f}(\boldsymbol{P}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) L_i(P, \omega_i)(\omega_i \cdot \hat{n}_P) d\omega_i$$

Figure 2 The rendering equation with BRDF highlighted in bold.

Independence of the rendering algorithm and lighting means we can implement such abstract BRDFs as "material plugins", accessed through the simple API, and the visual results produced by different renderers will be the same. This has led to the emergence of material libraries (e.g., the MDL [4] and universal material authoring tools (e.g., Substance Designer) favored by technical artists. Materials can also be studied and developed independently of the rendering algorithms.

For this abstraction to work, and to make BRDF physically based (photorealistic), the BRDF should obey to the following principles:

- **Helmholtz reciprocity** – the incoming and outgoing directions can be swapped (hence the name bidirectional), and the resulting reflectance will be the same. Note that some algorithms trace light rays originating from the camera (path tracing), other from the light sources (photon mapping) or both (bidirectional path tracing). Helmholtz reciprocity ensures consistent results for all types of algorithms.
- **Energy conservation** – the energy reflected from the surface should be equal to the energy received (for the surface with perfectly white albedo without absorption).

Early empirical models like Phong generally do not satisfy these requirements and therefore are not suitable for photorealistic rendering. They also cannot reproduce behavior of many materials correctly, e.g., rough surfaces or metals, as we will discuss later in more detail. This has driven the shift from empirical to physically based models even for real-time rendering once programmable and sufficiently powerful GPUs were available.

It must be noted that some of the successful models do not satisfy these requirements. Disney model [5] is not energy conserving (but note that sequel to their article [6] discusses energy conservation for later projects) and Autodesk model [7] does not necessarily satisfy Helmholtz reciprocity. The Disney model was used in combination with unidirectional path tracing where lack of reciprocity did not introduce significant issues. Perfect energy conservation may not be a strict

requirement either, because unless the BRDF reflects more energy than it receives it is unlikely to cause problems, except for the darkening of surfaces where energy is lost. In the end, high performance of BRDF implementation and ease of use by artists can be preferable to satisfying all theoretical requirements for BRDFs, especially when rendered content is fully under control.

2.1 Construction of BRDF

Most BRDFs used in games are composed of two BRDFs – diffuse and specular lobes. This decomposition comes from a fact that when light interacts with a surface, part of it is reflected away (contributing to specular BRDF), while the remaining part scatters into the surface. Inside of the material, more subsurface scattering occurs, and part of light eventually hits the surface again and exits the material (contributing to the diffuse BRDF) in randomized direction. Due to scattering inside of a surface, this light has been "filtered" and causes the object to appear with certain color. This also means that light can travel some distance under the surface, carrying the light away from the entry point, creating a "subsurface scattering" effect. A common simplification is to disregard this phenomenon and set the exit point to be equal to the entry point. We will not go into details of subsurface scattering in this article and recommend a recent blog post by Pettineo [8] which provides a nice summary on the topic.

Figure 3 Decomposition into diffuse (left) and specular (right) lobes. BRDF values shown in blue, while the red part shows cosine weighted BRDF contribution (scattered energy / exiting radiance contribution).

This decomposition into specular and diffuse reflections has been discussed in optics literature as early as 1924 by Pokrowski [9]. Note an important distinction between BRDF value (blue) and scattered energy (red) which is cosine weighted. Cosine term comes from rendering equation and will be discussed further in Section 3.1, but for the detailed explanation we recommend chapter about Materials in Graphics Codex [10].

How much light reflects away (or scatters into) the material is described by Fresnel equations. Light incident under grazing angles is more likely to be reflected, which creates an effect sometimes called the "Fresnel reflections" (see Figure 4). Fresnel term is also responsible for modeling distinction between metals and dielectrics. Metals have much higher absorption coefficient than dielectrics, meaning part of the light that would otherwise be reflected is absorbed. Because this absorption is dependent on wavelength, reflections of some metals (e.g., gold) are colored depending on how individual wavelengths are absorbed, while reflections of dielectrics generally take over an unchanged color of the light source. More details about Fresnel term will be discussed in Section 4.3.

Figure 4 "Fresnel reflections" on still water surface.

Also note that because light contributing to diffuse BRDF has crossed the material boundary twice, technically we should also apply the Fresnel term twice, but diffuse BRDFs are typically created to account for this.

We have already mentioned that light travels some distance under the surface before exiting, creating a subsurface scattering effect. That assumed the travelled distance was relatively short, and the light exited via the same (or very similar) surface it had entered. However, for fully- or semi-transparent materials, the light can pass through the object and exit on the other side. To complicate things further, scattering inside of the material and/or rough surface can cause the light to be diffused (transmitted in randomized directions, e.g., via the frosted glass, ice…). Both transmittance and subsurface scattering can be rendered using the "volumetric light transport" methods which simulate how light travels inside the volume. These methods are generally expensive, and its effects can be approximated using:

- **BTDF (Bidirectional Transmittance Distribution Function)** – a function describing how the light is transmitted through the surface to the other side (on the object boundaries), introduced by Jos Stam [11]**.**
- **Participating media** – a simple extinction of light due to travelling through the object.
- **Subsurface scattering algorithm** – to calculate the approximate exit point for incident light.

Note that BTDF only describes transmittance on surface boundaries, but not inside of the object. Light only interacts with material at points where it enters and exits the volume, as if the mesh was a hollow shell (actually, that's how it's mesh is modeled), while it ignores scattering that might happen inside of the media. In this case, light extinction which would occur has to be applied explicitly.

So far, we have discussed BRDF and BTDF to describe reflection and transmission of light – but there is also an umbrella term "bidirectional scattering distribution function" (BSDF or BxDF) commonly seen in the literature meaning either BRDF, BTDF, BSSRDF (for subsurface scattering), etc.

Finally, all our BSDFs (specular BRDF, diffuse BRDF and BTDF) must be combined in a meaningful way, greatest concern being the energy conservation, which we will discuss in Section 5.1.

3 Implementing simple BRDFs

In this section we describe well-known simple BRDFs – Lambertian, Phong, and few others. We will also define an interface which will enable us to implement more complex BRDFs as drop-in replacements. Before we start, let us define some commonly used vectors with the help of Figure 5.

Vector N is a shading normal for given surface (possibly coming from a normal map). Vector V, the view vector, specifies a direction from shaded point towards the viewer (camera). More generally, it specifies outgoing light direction, and in a typical path tracer, this is the opposite of ray direction. Vector L points toward a light source, or, depending on context, specifies a direction in which the incident ray reflects. Vector H, a half (half-way) vector lies half-way between V and L. It is also equal to the microsurface normal of a microfacet model discussed in Section 4. Vector R is a direction of perfect (mirror) reflection of vector $-L$ along the normal N, and vector T is a tangent perpendicular to N on the triangle plane. All these vectors are normalized. Vectors v and l are projections of V and L onto triangle plane.

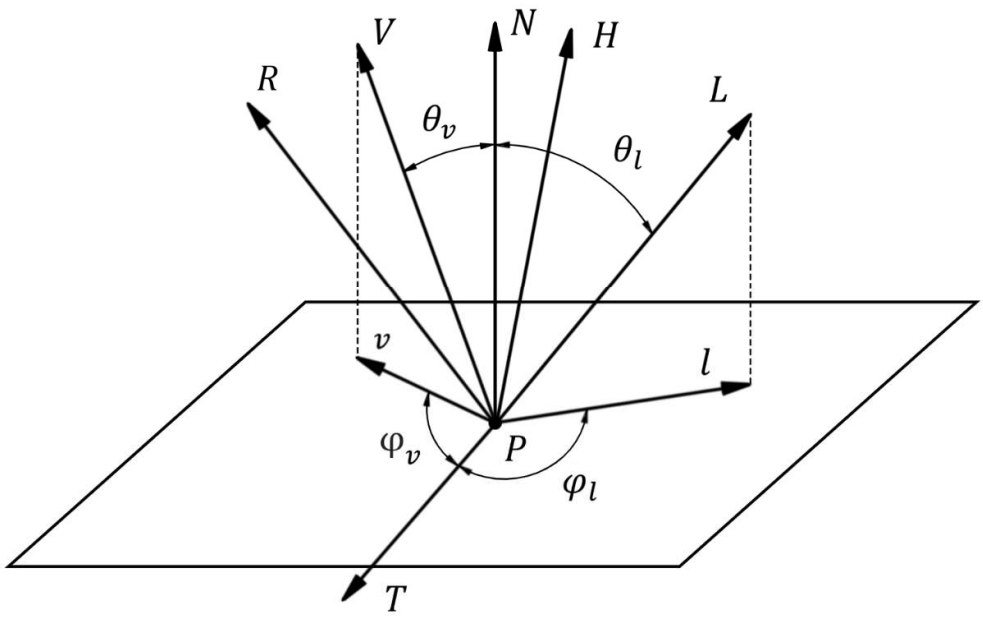

Figure 5 Vectors commonly used for BRDF evaluation.

3.1 Lambertian diffuse BRDF

Lambertian is one of the simplest BRDF functions used for diffuse term which assumes that incident light is scattered in all possible directions equally (within the hemisphere around the surface normal), but it is still a good approximation for behavior of many real-world materials and is very fast to evaluate. Therefore, it can be a good choice, e.g., for secondary light bounces (in combination with suitable specular BRDF). It is also a default diffuse BRDF used in UE4 engine [12]. It is not view-dependent, so it is friendly to algorithms involving temporal accumulation with reprojection (denoising, TAA, etc.). For a full derivation from first principles there is a nice article written by Saikia [13], here we will start with definition of Lambertian BRDF which is:

$$f_{Lambertian}(L,V) = \frac{diffuse\_reflectance}{\pi}$$

Evaluating this for light coming from a known direction L (e.g., a virtual light source) and reflecting towards a viewer is straightforward:

```
brdfLambertian = (diffuseReflectance / PI) * dot(N,L);
```

By this we implement the function of our BRDF interface we will call *eval*. Note that we have moved the cosine term from rendering equation (Figure 2) to evaluation function. This is a common practice as in some cases, it can cancel out with another cosine term used in the BRDF. Such implementations are sometimes referred to as "BRDFs with cosine term built in" and care must be taken not to apply it again. Diffuse reflectance (often denoted with the symbol ρ) specifies how much of the incident light is reflected in the diffuse lobe and is typically calculated from a "base color" parameter which we will discuss in Section 4.3.14.3. A term albedo is sometimes also used, which can be ambiguous since albedo (defined as ratio of incident and reflected light) can be dependent on direction, although in practice it is typically specified for direction of normal incidence.

Another common optimization is to move dividing by a constant π into preprocessing step, pre-dividing diffuse reflectance textures offline. However, all functions using diffuse reflectance (base color) must then be adjusted accordingly, some of which may not contain division by π. Another option is to pre-divide light intensities, but for path tracing where light can come from multiple bounces, this is not applicable. To make things less confusing, we include division by π inside of BRDF in our code sample. Thorough discussion on this topic can be found in the article "to PI or not to PI" by Lagarde [14].

So far, we can calculate BRDF contribution for light coming from known direction using our *eval* function. For most light transport algorithms (e.g., the Monte Carlo path tracing), another function is needed to figure out in which direction should we trace the next ray. We will call this function *sample*. It will take a random number as a parameter and return a new ray direction, along with a value of probability density function (PDF), which can be understood as a likelihood of choosing that particular direction over other possible directions. Looking at how PDF is used in Monte Carlo estimator:

$$F = \frac{f(x)}{pdf(x)}$$

and substituting $f(L, V)$ for our Lambertian BRDF function we get:

$$I = \frac{f_{Lambertian}(L, V)}{pdf(L)}$$

$$I = \frac{\frac{diffuse\_reflectance}{\pi}(N \cdot L)}{pdf(L)}$$

We call the value I a *weight of the sample* and we will introduce it in the next paragraph. The value of *pdf* depends on the sampling method, and its choice can be arbitrary, as long as its PDF is positive and nonzero whenever BRDF is nonzero. Looking at the last equation we can see that by clever choice of sampling function, its PDF could cancel out with some BRDF terms and simplify the calculation. Most basic sampling method, which is actually a great choice for Lambertian and other diffuse BRDFs is sampling in a cosine-weighted hemisphere, with PDF = $\frac{(N \cdot L)}{\pi}$. The code for this *sample* method can be found, e.g., in Sampling Transformations Zoo in Ray Tracing Gems [15], and is used in our code sample. These methods generally work by generating a random point in disk (by warping two random numbers which form a square to a disk), and projecting that point up onto the hemisphere (just by calculating the Z component).

To see how PDF and BRDF terms cancel out we substitute our cosine-weighted PDF, and we get:

$$I = \frac{\frac{diffuse\_reflectance}{\pi}(N \cdot L)}{\frac{(N \cdot L)}{\pi}}$$

$$I = diffuse\_reflectance$$

Now we can see how moving cosine term from rendering equation into BRDF evaluation made it possible to cancel it out, along with π, so that only the reflectance remains in the final equation. This value is useful when Monte Carlo integration is used, e.g., in path tracing, and we will call it the *weight of sample*. When ray hits a surface and we sample a direction of the reflected ray according to BRDF, the amount of light that this ray can "carry" has to be adjusted by this weight, to account for material properties of the surface.

This approach of constructing the sampling function with PDF closely matching the BRDF not only simplifies the code, but also reduces variance (noise) of Monte Carlo estimator and is therefore an important aspect of BRDF implementation. A good example of such method is the sampling with VNDF (distribution of visible normals) which will be discussed in Section 4.4.

Using this, we have now created an optimized function we will call *evalIndirect,* which combines *eval* and *sample* into one function which returns the sampled direction and its weight. But notice that it also eliminates calculation of true PDF, which can still be useful for some algorithms, e.g., multiple importance sampling. To fix this, we introduce new method called *pdf* which simply returns PDF of sampling given direction. To verify whether our optimized *evalIndirect* returns results consistent with *sample* and *pdf*, we can always implement *evalIndirect* by calling *eval* and *pdf* explicitly and check if value $\frac{eval}{pdf}$ is equal to the weight of the sample of optimized *evalIndirect* function.

These 4 functions define our interface for any BRDF, so we can implement different material types easily by implementing these functions and reuse code for different light transport algorithms. To summarize:

- ***eval*** evaluates the BRDF function for known incoming and outgoing directions
- ***sample*** samples BRDF to return a new (outgoing) ray direction
- ***pdf*** returns PDF of sampling given outgoing direction
- ***evalIndirect*** combines these functions in an optimized version

3.2 Phong Reflection Model

Another popular reflection model commonly used with Lambertian in the past was introduced by Phong [16] to provide a computationally inexpensive specular highlight for real time rendering (which he defined as rendering with >30 FPS). Phong reflection model is not to be confused with Phong shading – a method for normal interpolation to achieve smooth specular highlights. While the original Phong's paper states that his reflection model is physically based (derived by observation of real materials), it is not physically based by today's standards as it lacks phenomena such as off-specular peak, Fresnel reflections and others. Its formula is simple, leading to a straightforward implementation (notice the $(N \cdot L)$ was term added to the code the same way as for Lambertian):

$$f_{Phong}(L, V) = specular\_reflectance\ (R \cdot V)^{shininess}$$

```
brdfPhong = specular_reflectance * pow(dot(R, V), shininess) * dot(N, L);
```

The width of the highlight is controlled by the exponent (often called shininess, or shine) which is an unbounded parameter, but it is common to limit its range to about 10000. To evaluate Phong model, we must calculate the vector R (vector $-L$ reflected along the normal), but as shown by Blinn

in his optimized version [17], the half vector H can be used instead. To implement *sample* function in our code we use a method found in Ray Tracing Gems [15].

Since it is neither energy conserving nor reciprocal, it is technically not correct to call it a "Phong BRDF" in this form, however, there have been attempts to fix these deficiencies and make it suitable for path tracing. Articles by Lafortune [18] and Lawrence [19] show such solutions, including importance sampling. Various normalization terms have been created to make Phong model energy conserving (see, for example, the article by Giesen [20]). Blinn introduced an optimized version [17] and combined Phong's specular highlight with Lambertian diffuse and constant ambient terms, creating a Blinn-Phong reflection model which became a standard in real time applications until it was replaced by more advanced physically based models. Now mostly obsolete ambient term was intended to account for indirect illumination which has since been replaced by image-based lighting, real time ray tracing, or similar methods.

One of the most comprehensive models based on Phong reflection was introduced by Ashikhmin and Shirley [21] and featured energy conserving reciprocal BRDF, anisotropic reflections, was coupled with a suitable diffuse BRDF and an importance sampling method. More recently, another model based on Phong was introduced by Gotanda [22] which was highly optimized for low computational cost. Gotanda also later introduced an improved layered version of his model [23].

3.3 Oren-Nayar Diffuse Reflectance Model

More advanced reflectance model for diffuse reflection which accounts for surface roughness was introduced by Oren and Nayar [24]. They based their model on observation that rough surfaces reflect more light when view direction approaches incident light direction than Lambertian model predicts (an effect called backscattering, an example of such non-Lambertian surface is the moon). Oren-Nayar model accounts for this phenomenon by generalizing Lambertian model and making it sensitive to surface roughness and viewing direction. It was derived using the microfacet theory, just like the physically based specular reflectance model which will be discussed in following section.

Note that when this model is coupled with a specular BRDF, it is possible that underlying microfacet model and distribution functions are different (Oren-Nayar uses V-cavity model with Gaussian distribution of slopes), and the roughness parameter may need to be converted from one to another or remapped to plausible range. Because roughness in Oren-Nayar model (denoted σ) is defined as standard deviation of microfacet orientation angle, it is specified in angular units. Appendix B in Lagarde's article on Frostbite [25] suggests a conversion between Beckmann roughness and Oren-Nayar roughness as:

$$roughness_{OrenNayar} = \frac{1}{\sqrt{2}} \arctan\left(roughness_{Beckmann}\right)$$

Oren and Nayar presented several versions of their model with various degrees of simplifications. Code accompanying this article uses their simplest "qualitative" model:

$$f_{OrenNayar}(L,V) = \frac{diffuse\_reflectance}{\pi}(N \cdot L)(A + B \max\left(0, \cos\left(\varphi_v - \varphi_l\right)\right) sin\alpha\, tan\beta)$$

$$A = 1.0 - 0.5\frac{\sigma^2}{\sigma^2 + 0.33} \;\; ; \;\; B = 0.45\frac{\sigma^2}{\sigma^2 + 0.09} \;\; ; \;\; \alpha = \max(\theta_l, \theta_v) \;\; ; \;\; \beta = \min\left(\theta_l, \theta_v\right)$$

where θ_l (and θ_v) are angles between normal and light (and view) vector but note that φ_v and φ_l are azimuth angles between these vectors (angle between their projections onto plane defined by the normal). Our code sample calculates projection of these vectors onto normal plane to obtain correct

angles, but optimized version which also eliminates use of some trigonometry functions (for the expense of square roots) can be found in Physically Based Rendering [26].

3.4 Disney (Burley) Diffuse Model

Another popular and widely used model is the Disney diffuse [5], sometimes also called the Burley diffuse. It is an empirical model derived by observation of measured data which includes the phenomenon of grazing retroreflection dependent on roughness. In this regard, it is like the Oren-Nayar model, but is simpler to evaluate. It is based on the Schlick's Fresnel approximation formulas (see Section 4.3) and extends basic Lambertian model to either increase or decrease reflectance on grazing angles within specified bounds (0.5 to 2.5) depending on roughness:

$$F_{D90} = 0.5 + 2\,roughness\,cos^2\theta_d$$

$$f_{Disney}(L,V) = \frac{diffuse\_reflectanc}{\pi}(1 + (F_{D9} - 1)(1 - \cos\theta_l)^5)(1 + (F_{D9} - 1)(1 - \cos\theta_v)^5)$$

Because this model was not derived using the microfacet theory (unlike the Oren-Nayar model), the roughness parameter does not have the same meaning with regards to statistical distribution of microfacet normals and may not be compatible with roughness used for specular BRDF. Burley's paper suggests a remapping of roughness to specular BRDF roughness (α) as:

$$\alpha = roughness^2$$

Furthermore, this model is not energy conserving, and it can be problematic to use in renderers where energy conservation is important (e.g., in path tracing). Lagarde describes a version of this model in his excellent article on rendering in Frostbite engine [25], which significantly improves energy conservation, and ensures that no more energy is reflected than is received when coupled with a GGX-based specular BRDF.

3.5 More diffuse BRDFs

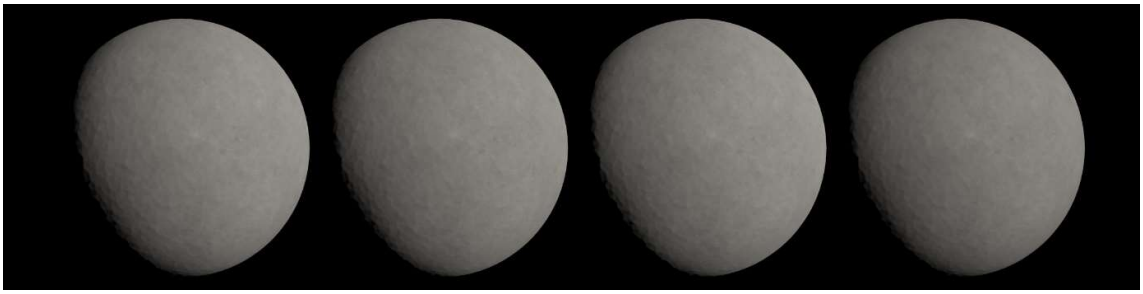

Figure 6 Subtle differences. Comparison of diffuse BRDFs on a rough object (roughness = 0.5). From left to right: Lambertian, Oren-Nayar, Disney diffuse, and Frostbite diffuse.

Another diffuse BRDFs worth studying not covered in this article are, e.g., the ones by Hanrahan-Krueger [27] and Heitz-Dupuy [28]. An interesting research direction is to find diffuse models that combine well with widely used GGX-based microfacet BRDF with regards to energy conservation and plausible visual results, such as the one found in work by Gotanda [29].

4 Microfacet Model

Many recent BRDFs are based on the microfacet theory, originally developed in the optics literature, and introduced to the graphics community in articles by Blinn [17] and Cook and Torrance [30]. The motivation for microfacet model was to better understand and model reflection of light from rough surfaces, which as shown by measurements, do not match results predicted by simple Lambertian and Fresnel reflections. For example, the microfacet theory accounts for the phenomenon

called *off-specular peak*, occurring when the maximum reflectance is achieved for direction shifted away from the direction of perfect reflection, as discussed by Torrance and Sparrow [31].

BRDFs using the microfacet model are generally called *physically based* to indicate they're designed with laws of physics (optics) in mind rather than empirically by observation, though in computer graphics we often use simplifications and approximations to make computations feasible, and to make BRDFs more practical by eliminating physical units and parameters.

A good example of such simplification is omitting complex index of refraction used in Fresnel equations. Sometimes the term *physically plausible* is also used to describe models that behave as observed real-world materials but are not necessarily derived using the laws of optics. For real time rendering we also typically use geometrical optics and RGB triplets to represent light instead of wave optics and spectral distributions. Trade-off here is that such renderers cannot account for effects like diffraction due to thin film interference, and polarization of light.

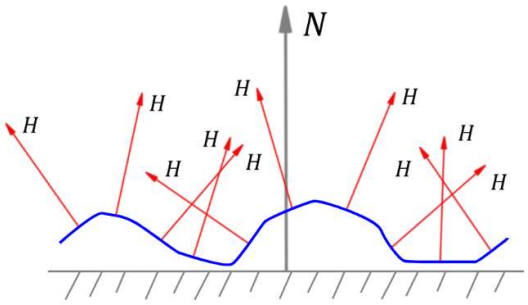

Figure 7 A microsurface (blue) consisting of many microfacets. Microsurface normals H (red) are shown for a few selected microfacets. Geometric surface and its normal N is highlighted with gray. Note that depending on microsurface model, the microsurface itself does not need to be continuous as the example on this image.

A microfacet theory models the surface as a collection of tiny surfaces – microfacets – with varying slope and height (see Figure 7). How much and in what way do microfacets vary is described by the microfacet distribution function D (which typically depends on the roughness parameter to control the surface appearance). This gives the BRDF its shape which affects the final appearance of the material.

Typically, individual microfacets are assumed to act as perfect mirrors, but any BSDF could be used for microfacets as well. Note that only the geometric surface is modeled, and the effects of microfacets are evaluated using the microfacet model, so we only observe their aggregate effect[1].

A microfacet model reflection term is defined as [31] [30] [32]:

$$R = \frac{F(L,H)\,G(L,V,H)\,D(H)}{4\,(N\cdot L)(N\cdot V)}$$

where N is a shading normal and H is a half vector pointing in a direction half-way between L and V ($H = \|L+V\|$). It is also called *microsurface normal*, *highlight vector* or *half-way vector* and was previously used by Blinn [17] to optimize the Phong shading calculation. Formulation using the half vector is handy as it can also be used for refraction as described by Walter [32], and for importance sampling by generating half vectors corresponding to given normal distribution function D as we will

[1] But there is an interesting paper by Heitz which procedurally generates meshes of Beckmann distribution surfaces for research purposes [71]

discuss later. All these terms will be discussed in sections below in more detail, here is just a short summary:

- **D Term** – A microfacet distribution function – tells us what fraction of microfacets are oriented in direction H so that light incoming from direction L will be reflected in direction V.
- **F Term** – Fresnel term, evaluates how much light is reflected off the surface under given angle of incidence.
- **G Term** – Geometric attenuation term (also *masking and shadowing term* denoted G_2 later in this text), accounts for mutual shadowing (and masking) of microfacets, sometimes also used for normalization of BRDF.
- **Denominator** – comes from derivation of the microfacet model using perfect mirrors as microfacets. Note that original paper by Cook and Torrance [30] uses the constant π instead of 4 in denominator. Walter points out [32] this is due to different normalization of D term, and most recent literature agrees on using the constant 4.

4.1 Distribution Term

Distribution term D uses a microfacet normal distribution function (NDF) which evaluates what fraction of microfacets is oriented in direction H so that light incoming from direction L will be reflected in direction V. In other words – how much light can be reflected between given L and V directions assuming no occlusion occurs between individual microfacets (discussed below). NDF here should not be confused with normal (meaning Gaussian) distribution, although it is one of possible implementations of NDF (used, e.g., by Torrance and Sparrow [31]).

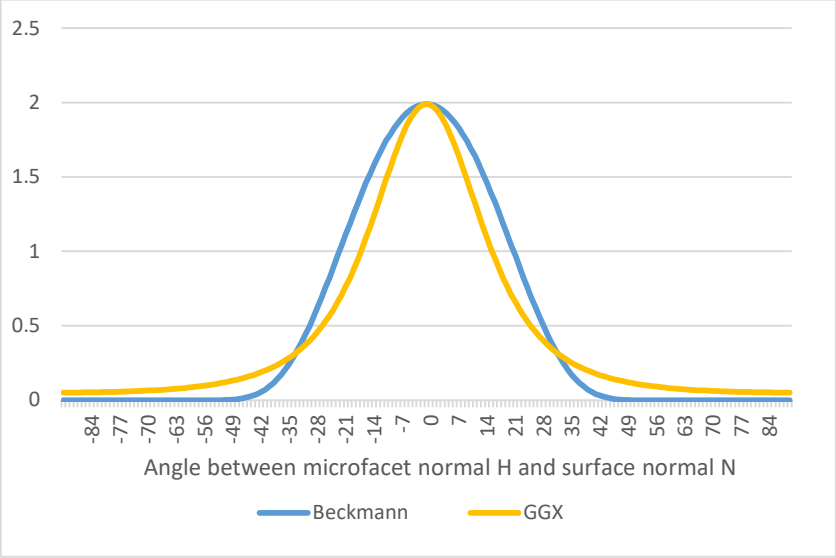

Figure 8 "Longer tail" of GG-X distribution, shown for alpha value of 0.4.

Various distribution functions can be found in the literature. An example is a Beckmann NDF widely used in optics, but also computer graphics. Beckmann distribution uses an intuitive roughness parameter specified as a root mean square (RMS) slope of microfacets to control its shape. Another important NDF was proposed by Trowbridge and Reitz [33] and recommended by Blinn in 1977 [17]. It was later re-derived under the name GG-X[2] by Walter [32]. While Trowbridge-Reitz distribution is identical to GG-X and predates it, we will use simpler GG-X formulation in this article. Compared to Beckmann NDF, the shape of specular lobe it provides has a "longer tail", meaning the falloff of

[2] GG-X is likely an abbreviation of "Ground Glass - roughness unknown" [72], named after panes of ground glass polished to certain roughness used for measurements in optics.

specular peak is slower, as shown on Figure 8. This matches many measured real-world materials more closely, and GG-X is widely used in games, although Beckmann distribution is still used, e.g., as mentioned in paper describing shading at Pixar [34]. It should be noted that evaluation of GG-X is generally less expensive thanks to many optimizations and approximations contributed by the graphics community.

The Beckmann distribution is defined as:

$$D_{Beckmann} = \frac{e^{\frac{-\tan^2(\theta_h)}{\alpha^2}}}{\pi\,\alpha^2 cos^4\theta_h}$$

where θ_h is the angle between normal and half vector, and α specifies the roughness. Relation between α and RMS slope of microfacets σ is $\alpha = \sqrt{2\sigma}$ [26]. As an optimization, we can use the equivalency $\tan^2(\theta_h) = \frac{1-cos^2\theta_h}{cos^2\theta_h}$ as done in the article by Hoffman [35] to eliminate the tangent function, which we replace by handily available cosine of θ_h which is equal to easy to calculate $N \cdot H$:

$$D_{Beckmann} = \frac{e^{\frac{cos^2\theta_h - 1}{\alpha^2 cos^2\theta_h}}}{\pi\,\alpha^2 cos^4\theta_h}$$

The GG-X distribution is defined as:

$$D_{GGX} = \frac{\alpha^2}{\pi\, cos^4\theta_h(\alpha^2 + \tan^2(\theta_h))^2}$$

This formula can also be further simplified [35]. We replace the tangent function in a same way as for the Beckmann distribution, and expand denominator, which can then be expressed as $\left((\alpha^2 - 1)cos^2\theta_h + 1\right)$ squared. Resulting optimized formula is:

$$D_{GGX} = \frac{\alpha^2}{\pi\left((\alpha^2 - 1)cos^2\theta_h + 1\right)^2}$$

There's also a NDF derived by Blinn using the Phong reflection model (dubbed Blinn-Phong NDF) [17], and also discussed in Walter's paper [32], where he concludes that for certain roughness values, Beckmann and Blinn-Phong distributions are very similar. This can explain a longevity of Phong shading model, as it is able to accurately represent certain materials (most notably the plastics). Formula for approximate conversion between Beckmann roughness and Phong exponent is also provided in Walter's paper [32] (which also shows that for roughness values around 0.2, Phong distribution is nearly identical to Beckmann). It might seem outdated today but may come handy when loading older models with materials specified using the Phong exponent (e.g., in the popular OBJ format).

$$shininess = \frac{2}{\alpha^2} - 2 \qquad \alpha = \sqrt{\frac{2}{shininess + 2}}$$

As we can see, the units of roughness can be very different for each model. Torrance and Sparrow and Oren-Nayar models use standard deviation of normal distribution directly, Beckmann and Trowbridge-Reitz use statistical root mean square of slopes, while Burley uses empirically chosen values in plausible range. When multiple BRDFs are coupled together (typically specular and diffuse BRDFs), it is important to carefully convert roughness specified by artists to roughness units used by underlying BRDFs. A good example of such conversion can be found in Disney's Principled BRDF [5]. As

already mentioned, they remap roughness parameter to α used for specular BRDF as a square of that value. This makes changes to roughness perceptively linear and compatible with underlying diffuse BRDF, but care must be taken to square the α value again in calculations of D and G terms (see formulas above). Alternatively, one can specify roughness for each BRDF separately for more artistic control.

4.2 Geometric Attenuation Term

Geometric attenuation term G accounts for attenuation of reflected light due to the geometry of the microsurface which occurs when some microfacets block each other. It is sometimes also used as a normalization term of the BRDF. As pointed out by Torrance and Sparrow [31], the G term counteracts the Fresnel term and is responsible for the "off-specular peak" that occurs for materials of certain roughness when high reflectance predicted by Fresnel is attenuated by significant shadowing due to G term at grazing angles, with peak reflectance being achieved at lower angle.

The geometry of the microsurface is given by the profile used to model its shape. There have been two significant microsurface profiles in use: V-Cavity model which assumes the microsurface is composed of V shaped grooves of certain width and height, and the Smith model based on randomized distribution of slopes [36]. V-cavity model was used by Cook-Torrance and Oren-Nayar in their BRDFs, but as shown in an extensive article by Heitz [37], Smith function is the correct one to use of these two, therefore, we will only discuss Smith's G function in this article, which is defined as:

$$G_1(H,S) = \frac{1}{1 + \lambda(a)} \quad ; \quad a = \frac{(H \cdot S)}{\alpha \sqrt{1 - (H \cdot S)^2}}$$

where S is either L or V vector, H is a microfacet normal and λ is a function specific for the selected distribution function (NDF). The process to derive λ for given NDF has been described in paper by Brown [38] and papers by Walter [32] and Heitz [37] show λ functions (and optimized G_1terms) for various NDFs (Beckmann, GG-X, Blinn-Phong):

$$\lambda_{GGX}(a) = \frac{-1 + \sqrt{1 + \frac{1}{a^2}}}{2} \; ; \; \lambda_{Beckmann}(a) = \begin{cases} \dfrac{1 - 1.259a + 0.396a^2}{3.535a + 2.181a^2}, & where\ a < 1.6 \\ 0, & where\ a \geq 1.6 \end{cases}$$

Here, we show a rational approximation to $\lambda_{Beckmann}$ derived by Walter [32], rather than original function which requires evaluation of an expensive error function. Evaluation of G_1 can be optimized for specific NDF by substituting its λ function into general G_1formula and simplifying the expression, as done in our code sample.

Another common rational approximation of G_1 for Beckmann distributions was introduced by Schlick [39] in the same paper where the well-known Schlick's Fresnel approximation appears. However, he approximated different version of Smith G function [32] [35] [37], which is unsuitable for microfacet BRDF.

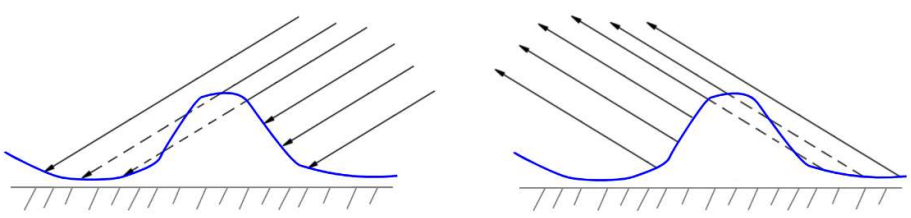

Figure 9 Shadowing (left) occurs when incident light is blocked by other microfacet on the microsurface (blue). Masking (right) occurring when reflected light is blocked by another microfacet.

Geometric attenuation occurs because many microfacets can be oriented in a way to reflect light incident under given direction, but ultimately only the one closest to the light source reflects the light (others are shadowed, as shown on Figure 9). Same applies for the reflected light which can be blocked (masked) by other microfacets on its way out of the microsurface. The Smith's G_1 function can be used for both shadowing $G_1(H, L)$ and masking $G_1(H, V)$, and their combination gives the masking-shadowing function:

$$G_2 = G_1(H, L) * G_1(H, V)$$

This combination formula assumes that masking and shadowing is uncorrelated, however, this is not accurate as facets deeper in the microsurface have higher probability of being both shadowed and masked. This can be solved by using the height-correlated form of Smith's masking-shadowing function:

$$G_2 = \frac{1}{1 + G_1(H, L) + G_1(H, V)}$$

Note that either form of G_2 can be used for the G term in the microfacet model reflection formula introduced earlier, but correlated version is preferable since it is only slightly more expensive and is more accurate. A significantly optimized implementation of height-correlated G_2 for GG-X distribution can be found in work by Lagarde [25]. By substituting G_1 terms for GG-X distribution into formula for G_2 and simplifying, we get the expression which contains terms that conveniently cancel out with denominator of microfacet model, making its evaluation even less costly. Another interesting approximation to height-correlated $G_2$for GG-X was introduced by Hammon [40], which is very fast to evaluate, but introduces some amount of error. For the Beckmann distribution, our code sample does not provide such optimizations and evaluates all the terms directly.

Since the G term effectively specifies the fraction of microfacets that are visible, the combined NDF and G_1 ($D * G_1$) term gives the *distribution of visible normals* [41] which is a base of the efficient sampling routine which we will discuss later.

Note that so far, we assumed that light being shadowed or masked is lost, which is not true even for our simplified model with perfectly smooth microfacets. This is where the energy loss of microfacet-based BRDFs occurs and unfortunately, modeling multiple scattering between microfacets is complex and computationally expensive and is usually ignored in real-time rendering. Papers by Heitz [42], Hitchhiker's Guide by d'Eon [43], and technique used by Imageworks [44] (also discussed in Real-Time Rendering [45]) go into more details about this topic.

4.2.1 Anisotropic materials

Materials exhibiting anisotropic reflections (such as brushed metals) can be modelled by using anisotropic versions of NDFs and their corresponding geometric terms. Thorough discussion on this topic can be found in papers by Heitz [37] and Ward [46]. These functions are generally more complex to evaluate and are controlled by two-component roughness values (different roughness for two perpendicular directions). Evaluation must be performed in tangent space to ensure correct and consistent orientation of anisotropic specular highlight across the whole mesh. Article on Disney Principled BRDF [5] shows an intuitive parameterization using additional *anisotropic* parameter (in the addendum of latest version of the article). This is used in combination with standard roughness parameter to calculate 2D roughness within plausible range.

A simpler solution suggested in Physically Based Rendering (p. 543) [26] still uses isotropic versions of NDF and G terms but adjusts the roughness value based on the direction where anisotropic highlight should occur.

4.3 Fresnel Term

Fresnel term F determines how much light will be reflected off the surface, effectively telling us how much light will contribute to evaluated BRDF. The remaining part $(1 - F)$ will be passed to underlying material layer (e.g., the diffuse BRDF, or transmission BTDF). Our implementation so far only discusses two layers (specular and diffuse), but it is possible to create complex materials with many layers. The Fresnel term should be evaluated on each interface where light passes from one layer to another. For simplicity, the thickness of individual layers is typically zero, but some thickness could be modeled in combination with spectral rendering to create effects such as diffraction (iridescence) occurring when layer interface is thinner than wavelengths of light passing through it. For further discussion about layered materials, we recommend articles by Autodesk [7] and works by Jakob et al. [47] and Weidlich and Wilkie [48]. For implementing iridescence, an interesting read is the recent paper by Belcour and Parla [49].

Fresnel term is dependent on viewing direction with regards to the surface normal, material properties (index of refraction and extinction coefficient), and is expressed by the relatively complex Fresnel equations which, however, also account for the polarization of light. Fresnel term always approaches unity for grazing angles, meaning all light is reflected, although for in-between angles the "color shift" occurs as discussed by Cook and Torrance [30].

Christophe Schlick introduced a widely used approximation to the Fresnel term for use in computer graphics [39]. As he points out, full Fresnel Equations are not only computationally expensive, but depend on unintuitive index of refraction n and extinction k that do not fit our need for predictable and easy to use parameters. To complicate things further, full index of refraction is a complex number consisting of real part (the refraction index) and imaginary part (extinction coefficient) and is specified per wavelength. This is unsuitable for rendering with RGB triplets. Article by Naty Hoffman titled "Fresnel Equations Considered Harmful" [50] provides further discussion on the topic and concludes that full Fresnel equations aren't even more precise than Schlick's approximation, unless spectral rendering is used.

Schlick's approximation uses the observation that when viewed under 90 degrees, all materials exhibit perfect reflectance, and we can use only one parameter – reflectance of the surface under normal incidence (at 0 degrees - F_0) to approximate full Fresnel equations:

$$F = F_0 + (F_{90} - F_0) * (1 - u^5)$$

where u is the cosine of angle between normal and the viewing direction ($u = N \cdot V$) and F_{90} is equal to 1, except for the case discussed in the section below. Note that index of refraction and extinction coefficients have been eliminated, but we now need to specify reflectance at normal incidence F_0. Another consequence of using Schlick's formula is that shading cannot account for polarization of light. An interesting work in this area is done by Mojzik et al. [51].

4.3.1 Specifying reflectance at normal incidence

Elimination of n and k simplified the calculation, but also caused the loss of control whether material behaves as a dielectric (low absorption coefficient – plastic, wood, etc.) or conductor (high absorption coefficient – copper, gold, etc.). Difference between these is best observed as a color tint on reflections for metals (because part of the light spectrum is absorbed), while reflections off dielectrics take over unchanged color of the light source. As noted by Cook and Torrance [30], metallic reflections are tinted based on the base color of the material for normal incidence (denoted F_0, sometimes also R_{F0}). Using this knowledge, we can "fix" the color of the Fresnel reflection for metals, by introducing a parameter called *metalness* [52], which calculates F_0 as a blend between default reflectance for dielectrics and base color for metals [53]:

$$F_0 = lerp(F_{0Dielectrics}, base\_color, metalness)$$

A common choice for $F_{0Dielectric}$ is 0.04 (4% reflectivity) as used by, e.g., UE4 [12] and Frostbite [25]. Note that reflectance of some real world materials is even lower, 2% for water, but can also be higher for some dielectric materials, e.g., 0.18 for diamonds [35]. This value doesn't change with material but is fixed in the renderer and its choice has significant impact on final appearance. Metalness also attenuates diffuse reflectance for metals [30] [52], which is calculated from base color:

$$diffuse\_reflectance = base\_color * (1 - metalness)$$

This combination of parameters (base color and metalness) is sometimes called the *metalness workflow,* and is restrictive in a way that it doesn't enable setting different color hues for diffuse and specular reflectance. Another approach is „specular workflow" which enables to specify diffuse and specular reflectance directly [54], rather than calculating it from the base color. It requires more storage, but allows for greater artistic freedom by making it possible to create wider variety of materials, including unrealistic ones with significantly different diffuse and specular reflectance.

For some materials, even more control over the transition from F_0 to 1 may be desirable to reduce error of Schlick's approximation. RGB values for F_0 have been measured or calculated for normal incidence, but for some materials the color hue can differ slightly for in-between angles (e.g., certain metals and coated surfaces), or when we want to create unrealistic materials – *unobtainiums*). Gulbrandsen [55] addressed this problem by introducing two parameters to control reflectance and parameter he called *edge-tint*. More recently, Hoffman provided an improved solution [50] by introducing additional parameter denoted *h* to control *edge-falloff*, which includes Lazanyi's error term [56] to reduce Schlick's approximation error. The *h* parameter can be understood as F_{82} [57] – a reflectance at 82 degrees and can be either calculated using ground truth solution or measured for given material.

Note that F_0 originally depends on refraction indices on both sides of the surface and it is often assumed that surface is surrounded by air. If this is not the case (e.g., under water), F_0 should be adjusted accordingly (see, e.g., Real-Time Rendering p.324 [45]).

The above mentioned Lazanyi's fix re-introduces *n* and *k* parameters (but not needed in Hoffman's solution). If needed, these can be found for various materials in the excellent database created by Polyanskiy [58], however, still specified per wavelength.

An interesting optimization to Schlick's Fresnel term was proposed by Lagarde which uses the Spherical Gaussian approximation [59] and has been used in UE4 [12]:

$$F = F_0 + (F_{90} - F_0) * 2^{(-5.55473*u - 6.983146)*u}$$

Important implementation detail is that the Fresnel term must be evaluated for the normal of the sampled microfacet (half vector), rather than the surface (or shading) normal, otherwise the roughness of the material would be essentially ignored for Fresnel term and object would appear more reflective than it is.

A related issue comes from the fact that normal mapping is often used to create cavities, scratches, or patina, where artists darken the base color to make these features less reflective. Because Fresnel term always approaches 1 at 90 degrees, the effect is often the opposite of artist's intention, and cavities become more reflective, because their normals are approaching 90 degrees angle to the view direction. One of possible solutions is a fix introduced by Schüler [60], which prevents Fresnel term from approaching 1 under certain conditions. Instead of setting $F_{90} = 1$, we calculate F_{90} as $F_{90} = \min(1, 60\ luminance(F_0))$. This creates a smooth, but very fast falloff from 1 for F_0 values

with reflectance less than $1/60$. Number $1/50$ or reciprocal of the value $F_{0Dielectrics}$ is also often used in this place. We know that no real world material reflects less than 2%, so these low values can be considered as reserved for limiting of the Fresnel term. Finally, notice that this approach will not have any effect when F_0 is calculated using metalness as it is by default in our code sample, because no F_0 will be less than $F_{0Dielectrics}$. We provide this fix in the code for completeness, since it will be useful for cases when F_0 is calculated using different way or set directly by artist. Some engines provide the occlusion value that can be applied on specular component or used for Schüler's fix instead for similar effect. Note that there's only a handful of F_0 values that will trigger this fix when it comes from an 8-bit texture. More discussion on this topic can be found at the end of Hoffman's Crafting Physically Motivated Shading Models for Game Development [61].

4.4 Sampling the microfacet BRDF

With all this knowledge, we can now implement the *eval* function for microfacet specular BRDF, see the code sample accompanying this article for full listing.

For efficient implementation of *sample* method, we need a routine that performs importance sampling of microfacet BRDF constructed using selected D and G functions. In this article we use sampling based on VNDF introduced by Heitz [41], who also provided an improved and optimized version of his sampling for GG-X distribution in the separate article [62].

This sampling routine works with vectors specified in *local space*, where the positive Z axis is aligned with the shading normal (like most sampling functions). We must therefore transform our view vector into this local space before sampling, and transform the resulting light vector back using the inverse of such transform. Note that many renderers perform shading in tangent space, e.g., due to normal mapping, which is similar but not the same as local space of VNDF sampling. While tangent space may be constructed around geometry normal, our local space surrounds the shading normal. Therefore, additional transformation will be necessary also for the vectors already specified in tangent space.

Unit quaternions are an elegant way of representing rotations, and because we aim to find rotation between shading normal and standard basis vector (0, 0, 1), construction of such quaternion can be significantly simplified. Therefore, we use quaternion rotations for transforming vectors to VNDF local space and back (inverse transformation is found by simply inverting the axis of the quaternion).

To obtain the weight of the sampled direction, we must calculate BRDF divided by PDF of this sampling method as discussed in Section 3.1. Thanks to the construction of VNDF sampling, many terms of the BRDF cancel out with the terms of the PDF, and resulting weight is just $F * (\frac{G_2}{G_1(H,V)})$. Fraction of G_2 and G_1 can be rewritten using G_{1L} and G_{1V} terms only, as shown in the appendix of Heitz's paper on diffuse BRDF [28]. Because vector H here is constructed to be exactly halfway between L and V, the G_{1L} term is equal to the G_{1V}, and we can further optimize its evaluation as done in our code sample.

Original implementation of VNDF sampling (provided in supplemental material with VNDF paper [41]) relies on precomputing data during initialization, and as pointed out by Wenzel Jakob [63], a version for Beckmann distribution contains a discontinuity, which may cause problems when using low discrepancy sequences (e.g., blue noise), or for light transport algorithms such as Metropolis light transport. Jakob's paper contains an improved method which fixes these deficiencies, including the code. Previously widely used method that should be mentioned was introduced by Walter [32], but is less efficient than VNDF and has an undesirable property of generating samples with potentially very

large sample weights which lead to fireflies (although this can be partially mitigated for Beckmann distribution using the "Walter's trick" by adjusting roughness as $(\alpha' = (1.2 - 0.2\sqrt{|N \cdot L|})\alpha)$.

5 Combining BRDFs

To couple specular BRDF with diffuse BRDFs in this article, we use a simple method of blending specular and diffuse BRDFs together based on the Fresnel term. Note that microfacet model already weighs specular BRDF by F, so we additionally weigh diffuse BRDF by $(1 - F)$. This is inspired by an idea of layered materials where light interacts with each layer and Fresnel term is used to evaluate how much light reflects off the surface (contributing to specular lobe) and how much scatters into the surface (contributing to the lower layer, in our case, the diffuse lobe). Using this method, arbitrary BRDFs can be used together and combined into multiple layers [48], for example, a widely used combination of microfacet-based GG-X specular BRDF and either Lambertian or Disney diffuse BRDF. More complex multi-layered materials typically use one or two specular lobes (where additional specular lobe simulates a transparent coating on top of base surface), diffuse BRDF with optional subsurface scattering and BTDF for transmission. Note that each BRDF in the layered material should be energy conserving by itself.

Notice that to evaluate the diffuse BRDF layer, we need to know the Fresnel term of specular BRDF layer. In our code we sample a half vector for the specular BRDF and use it for Fresnel term evaluation, even when only diffuse term is evaluated. This can be expensive, so a better method with using an approximate normalization terms or tabulated data can be used instead.

More correct, but also more rigid solution is to use models which contain both diffuse and specular models that are designed to work well together, e.g., models by Ashikhmin-Shirley [21] and Kelemen-Szirmay-Kalos [64]. Another option is to select the preferred BRDF and find a suitable diffuse BRDF which can then be simply summed together. An example of this approach is the modification to Disney diffuse done by Lagarde [25]. More information on multi-layered materials used in practice can be found, e.g., in the talk on Call of Duty materials [65].

When combining the multiple BRDFs, the resulting BRDF should still obey to the basic principles of assuring energy conservation and Helmholtz reciprocity, however that is in practice often difficult to achieve.

5.1 Energy Conservation

One of the basic principles that BRDFs must respect stated in the introduction section is the requirement of energy conservation – the surface should not reflect more light than it receives. Furthermore, if the surface is a perfect reflector (white albedo), no energy loss should occur as well. In practice, it is often the case that some energy loss occurs even for physically based BRDFs derived from the microfacet model. The main source of energy loss being the lack of multiple reflections between microfacets.

When constructing material models for real-time rendering, the problem of energy conservation was often overlooked in the past, especially when only single bounce lighting was used. For path ray tracing and global illumination algorithms, it is important to ensure that no energy is created when light bounces off the surface, otherwise the convergence of path tracers could not be guaranteed.

Figure 10 White furnace test showing BRDF with perfect energy conservation (left) and BRDF with an energy loss (right).

An easy to implement and practical test of energy conservation is the *white furnace test* [37], which is a rendering of a white sphere illuminated by white light from all directions. If the material is energy conserving, the sphere will disappear against the white background. Any differences indicate either energy loss or gain (see Figure 10). Because rendering of the sphere contains every possible configuration of view and light vectors, we get a good sense about angles where the energy gain or loss occurs.

6 Parametrizing the BRDF

In this section we will summarize how do parameters of described BRDFs relate to material properties set by artists. So far are we have used the following parameters:

- Specular BRDF
  - Specular Reflectance at normal incidence F_0
  - Specular Reflectance at 82 degrees F_{82} (optional)
  - Roughness
- Diffuse BRDF
  - Diffuse Reflectance
  - Roughness

All these parameters could be set directly for greater artistic freedom (as single values or loaded from textures), or to model materials with known values (either measured or precalculated), but it is common to limit values that can be set for reflectance and roughness to make it easier to specify physically plausible materials and to make material properties more compact for storage.

Instead of using diffuse and specular reflectance directly, we use base color and metalness parameters to calculate them as described in the Section 4.3. This ensures that physically impossible materials are not easily created, and it also enables us to model distinction between dielectrics and metals. Roughness value is specified directly and remapped for specular BRDF the same way as in Disney Principled BRDF (by squaring). F_{82} is only used with Hofmann's improved Fresnel approximation and is especially useful to improve appearance of metals with known F_{82} values such as chrome or gold.

6.1 Transmission

To introduce an effect of transmission needed for rendering of semitransparent surfaces, we can reuse the concepts already discussed in this article – namely the Fresnel term, microfacet model and layered-materials. Fresnel equations tell us how much light is reflected away and scattered into

the surface, so we only have to decide what part of light that scatters inside is going to contribute to the diffuse BRDF, while remaining part is going to contribute to BTDF. We introduce a new parameter to our material model – the diffuse probability, or *transmittance,* which directly specifies ratio of how much scattered light is going to contribute to diffuse BRDF and to the BTDF. Note that this parameter is not the same as opacity which also affects specular BRDF, because we will still get Fresnel reflection under grazing angles for materials with high transmittance, while objects with zero opacity disappear completely.

The transmitted part of light can be either refracted perfectly according to Snell's Law, or we can use a microfacet BTDF to create rough refractions, the same way we created rough reflections. Walter's paper [32] provides an extensive summary of this method and resulting model for refractions is similar to microfacet reflection term.

7 Code Sample

This article comes with the code sample (https://github.com/boksajak/brdf), where all discussed BRDFs are implemented. File is written in HLSL, but also compiles in C++ environment (with the addition of a library to support HLSL types and functions, such as GLM), and can be easily integrated into rasterizer or path tracer. Note that code can be further optimized for the selected combination of BRDFs that will be used.

The default setting is combination of GG-X based microfacet specular BRDF with Lambertian diffuse BRDF, which are highly optimized and work well together. A code sample can be integrated by calling functions `evalCombinedBRDF` to evaluate contribution of a given light source to given point, and `evalIndirectCombinedBRDF` which samples new ray direction and its weight based on the selected BRDF (diffuse or specular), to determine a direction of the next ray in the path tracer.

8 Conclusion and Further Reading

In the production of movies and games, it can be more important to create material models that are physically based and plausible, rather than fully realistic. This means having models which do not break laws of physics (too much), but also enable enough artistic control by exposing intuitive and easy to understand parameters, even at the cost of omitting physical quantities such as the index of refraction.

It is interesting to see what parameters (and their ranges) individual models expose. An article on Disney Principled BRDF by Brent Burley [5] is a great starting point which discusses choice of parameters, ranges, and their mappings, along with technical decisions of choosing underlying BRDFs to construct successful material model. There's also interesting discussion on comparison with real world materials measured in the MERL database. It also provides chronologically sorted list of notable works worth studying (but only until 2012 when his paper was published) and there is a Github repository with their code freely available [66]. A follow-up article was published by Burley in 2015 [6] with more improvements based on practical use of their model.

For implementations used in games there are excellent works by Lagarde (Frostbite) [25], Karis (Unreal) [12] and Lazarov (Call of Duty) [67]. It is interesting to see how different developers came to different conclusions to same questions (e.g., use of Lambertian versus Disney diffuse, choice of GG-X versus Beckmann NDF distribution, etc.) indicating there is no single universal answer to these questions.

Deeper theoretical background of physically based BRDFs can be obtained by studying chapter 9 in Real-Time Rendering [45], which provides more pointers to relevant works, chapters 8 and 9 in Physically Based Rendering [26], from Naty Hoffman's Background: Physics and Math of Shading [35],

Graphics Codex [10] and from Eric Heitz's works on BRDFs. Physically based shading course webpages contain huge number of resources for further studies. It is also worth familiarizing with Substance PBR guide [54] to understand how technical artists work with PBR materials.

BRDFs used in movie production and offline rendering can be studied in articles by Burley [5] [6], descriptions of Arnold renderer [68], rendering at Pixar [34] and the material used by Autodesk [7]. These generally expose more parameters to artists than BRDFs used in games (for example the clearcoat and sheen parameters).

Because BRDF and materials terminology used across research and computer graphics industry is often confusing, and many terms are used interchangeably or incorrectly, we recommend reading A Taxonomy of Bidirectional Scattering Distribution Function Lobes for Rendering Engineers by McGuire et al. [69].

For an inspiration of what specific values to use to model different kinds of metals, there's a beautiful material study by Jarrod Hasenjager [70].

Another interesting topic to study is development of specialized BRDFs to solve a specific problem, e.g., rendering of skin, hair, textile, scratched surfaces, time-varying materials, water, and many other special materials.

9 References

[1] M. Winkelmann, "Zero-Day, Open Research Content Archive (ORCA)," 2019. [Online]. Available: https://developer.nvidia.com/orca/beeple-zero-day.

[2] N. Benty, K.-H. Yao, P. Clarberg, L. Chen, S. Kallweit, T. Foley, M. Oakes, C. Lavelle and C. Wyman, "The Falcor Rendering Framework," 2020. [Online]. Available: https://github.com/NVIDIAGameWorks/Falcor.

[3] J. T. Kajiya, "The rendering equation," *Proceedings of the 13th annual conference on Computer graphics and interactive techniques,* pp. pp. 143-150, 1986.

[4] L. Kettner, M. Raab, D. Seibert, J. Jordan and A. Keller, "The Material Definition Language," 2015.

[5] B. Burley, "Physically Based Shading at Disney," 2012.

[6] B. Burley, "Extending the Disney BRDF to a BSDF with," 2015.

[7] I. Georgiev, J. Portsmouth, Z. Andersson, A. Herubel, A. King, S. Ogaki and F. Servant, "Autodesk Standard Surface," 2019. [Online]. Available: https://autodesk.github.io/standard-surface/.

[8] M. Pettineo, "An Introduction To Real-Time Subsurface Scattering," 2019. [Online]. Available: https://therealmjp.github.io/posts/sss-intro/.

[9] G. I. Pokrowski, "Zur Theorie der diffusen Lichtreflexion," pp. 66-72, 1924.

[10] M. McGuire, "Graphics Codex," [Online]. Available: http://graphicscodex.com/.

[11] J. Stam, "An illumination model for a skin layer bounded by rough surfaces," in *Rendering Techniques*, 2001.

[12] B. Karis, "Real shading in unreal engine 4," in *Proc. Physically Based Shading Theory Practice 4*, 2013.

[13] S. Saikia, "Deriving Lambertian BRDF from first principles," 2019.

[14] S. Lagarde, "PI or not to PI in game lighting equation," 2012.

[15] P. Shirley, S. Laine, D. Hart, M. Pharr, P. Clarberg, E. Haines, M. Raab and D. Cline, "Sampling Transformations Zoo," in *Ray Tracing Gems*, 2019.

[16] B. Phong, "Illumination for computer generated pictures," *Communications of the ACM,* 1975.

[17] J. F. Blinn, "Models of light reflection for computer synthesized pictures," in *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, 1977.

[18] E. P. Lafortune and Y. D. Willems, "Using the modified phong reflectance model for physically based rendering," 1994.

[19] J. Lawrence, "Importance Sampling of the Phong Reflectance Model," 2008.

[20] F. Giesen, "Phong Normalization Factor derivation," 2009.

[21] M. Ashikhmin and P. Shirley, "An anisotropic phong BRDF model," *Journal of graphics tools,* 2000.

[22] Y. Gotanda, "Physically Based Shading Models in Film and Game Production: Practical Implementation at tri-Ace," 2010.

[23] Y. Gotanda, "Practical Physically Based Shading in Film and Game Production: Beyond a Simple Physically Based Blinn-Phong Model in Real-Time," 2012.

[24] M. Oren and S. K. Nayar, "Generalization of Lambert's reflectance model," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994.

[25] S. Lagarde and C. de Rousiers, "Moving Frostbite to Physically Based Rendering," 2014.

[26] M. Pharr, W. Jakob and G. Humphreys, Physically based rendering: From theory to implementation, Morgan Kaufmann, 2016.

[27] P. Hanrahan and W. Krueger, "Reflection from layered surfaces due to subsurface scattering," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 1993.

[28] E. Heitz and J. Dupuy, "Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation," 2015.

[29] Y. Gotanda, "Designing Reflectance Models for New Consoles," 2014.

[30] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," *ACM Transactions on Graphics,* 1982.

[31] K. E. Torrance and E. M. Sparrow, "Theory for off-specular reflection from roughened surfaces," *Josa,* 1967.

[32] B. Walter, S. R. Marschner, H. Li and K. E. Torrance, "Microfacet Models for Refraction through Rough Surfaces," *Rendering techniques,* 2007.

[33] T. S. Trowbridge and K. P. Reitz, "Average irregularity representation of a rough surface for ray reflection," *JOSA,* 1975.

[34] C. Hery and R. Villemin, "Physically Based Lighting at Pixar," 2013.

[35] N. Hoffman, "Background: Physics and Math of Shading," 2012.

[36] B. Smith, "Geometrical shadowing of a random rough surface," *IEEE Trans. on Antennas and Propagation,* 1967.

[37] E. Heitz, "Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs," 2014.

[38] G. Brown, "Shadowing by non-Gaussian random surfaces," *IEEE Transactions on Antennas and Propagation,* 1980.

[39] C. Schlick, "An inexpensive BRDF model for physically-based rendering," *Computer graphics forum,* 1994.

[40] E. J. Hammon, "PBR Diffuse Lighting for GGX+Smith Microsurfaces," GDC, 2005.

[41] E. Heitz and E. d'Eon, "Importance sampling microfacet-based BSDFs using the distribution of visible normals," *Computer Graphics Forum,* 2014.

[42] E. Heitz, J. Hanika, E. d'Eon and C. Dachsbacher, "Multiple-scattering microfacet BSDFs with the Smith model," *ACM Transactions on Graphics (TOG),* 2016.

[43] E. d'Eon, A Hitchhiker's Guide to Multiple Scattering, 2016.

[44] C. Kulla and A. Conty, "Revisiting Physically Based Shading at Imageworks," 2017.

[45] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki and S. Hillaire, Real-time Rendering, CRC Press, 2019.

[46] G. J. Ward, "Measuring and modeling anisotropic reflection," 1992.

[47] W. Jakob, E. d'Eon, O. Jakob and S. Marschner, "A comprehensive framework for rendering layered materials," 2014.

[48] A. Weidlich and A. Wilkie, "Arbitrarily layered micro-facet surfaces," in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, 2007.

[49] L. Belcour and P. Barla, "A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence," 2017.

[50] N. Hoffman, "Fresnel Equations Considered Harmful," 2019.

[51] M. Mojzík, T. Skřivan, A. Wilkie and J. Křivánek, "Bi-directional Polarised Light Transport," *EGSR,* 2016.

[52] P. S. Strauss, "A realistic lighting model for computer animators," *IEEE Computer Graphics and Applications,* 1990.

[53] B. Smith, "Reflection Model Design for WALL-E and Up," 2012.

[54] Allegorithmic, "The PBR guide by Allegorithmic," 2018. [Online]. Available: https://academy.substance3d.com/courses/the-pbr-guide-part-2.

[55] O. Gulbrandsen, "Artist friendly metallic fresnel," *Journal of Computer Graphics Techniques,* 2014.

[56] I. Lazániy and L. Szirmay-Kalos, "Fresnel term approximations for metals," 2005.

[57] N. Hoffman, "Fresnel Equations Considered Harmful (slides)," 2019. [Online]. Available: http://renderwonk.com/publications/mam2019/naty_mam2019.pdf.

[58] M. Polyanskiy, "Refractive index database," [Online]. Available: refractiveindex.info.

[59] S. Lagarde, "Spherical Gaussian approximation for Blinn-Phong, Phong and Fresnel," 2012. [Online]. Available: https://seblagarde.wordpress.com/2012/06/03/spherical-gaussien-approximation-for-blinn-phong-phong-and-fresnel/.

[60] C. Schüler, "An efficient and Physically Plausible Real Time Shading Model," in *ShaderX7*, 2009.

[61] N. Hoffman, "Crafting Physically Motivated Shading Models for Game Development," 2010.

[62] E. Heitz, "Sampling the GGX distribution of visible normals," *Journal of Computer Graphics Techniques,* 2018.

[63] W. Jakob, "An Improved Visible Normal Sampling," 2014.

[64] C. Kelemen and L. Szirmay-Kalos, "A microfacet based coupled specular-matte BRDF model with importance sampling," 2001.

[65] M. Drobot and A. Micciulla, "Practical Multilayered Materials - Call of Duty Infinite Warfare," in *SIGGRAPH*, 2017.

[66] B. Burley, "Disney BRDF Explorer Github repository," 2012. [Online]. Available: https://github.com/wdas/brdf.

[67] D. Lazarov, "Physically Based Lighting in Call of Duty: Black Ops," 2011.

[68] A. Langlands, "Physically Based Shader Design in Arnold," 2014.

[69] M. McGuire, J. Dorsey, E. Haines, J. F. Hughes, S. Marschner, M. Pharr and P. Shirley, "A Taxonomy of Bidirectional Scattering Distribution Function Lobes for Rendering Engineers," in *Workshop on Material Appearance Modeling*, 2020.

[70] J. Hasenjager, "Material Studies: Metals," 2016. [Online]. Available: https://www.behance.net/gallery/35636521/Material-Studies-Metals.

[71] E. Heitz, "Generating Procedural Beckmann Surfaces," 2015.

[72] M. McGuire, "Tweet #1174191724694032387," 2019. [Online]. Available: https://twitter.com/CasualEffects/status/1174191724694032387.

[73] P. Shirley, Ray Tracing in One Weekend, 2018.