Enterprise PBR Shading Model

Enterprise PBR Shading Model

Dassault Systèmes

Version 2025x

The Enterprise PBR Shading Model (DSPBR) is a high-quality, easy-to-use material optimized for performance in real-time and interactive scenarios, suitable for rasterization and ray tracing. Inspired by the Disney Principled BRDF [Bur12, Bur15] and the Unreal Shading Model [Kar13], it combines a metallic and a dielectric BSDF, including transparency for thin-walled and volumetric objects. In addition, it provides effects like emission, clear coat, metallic flakes and sheen to cover a wide range of appearances.

In this document we provide a mathematical description of the Enterprise PBR Shading Model in terms of bidirectional scattering distribution functions (BSDF) and related concepts. Following the description makes it possible to derive a visually consistent implementation in a physically-based renderer.

Versions: 2019x 2021x 2022x 2025x

Components

At its core, the Enterprise PBR Shading Model is a linear blend of a metallic BSDF and a dielectric BSDF, see Figure 1. The dielectric BSDF can be either opaque or transparent. The former is used for materials like plastics, wood, or stone, and translucent materials like wax or skin. The latter makes the surface transparent, allowing light rays to refract into the object, resulting in attenuation and subsurface scattering. This is needed, for example, for glass, water, or ice. Section 1.1 explains the BSDFs in more detail.

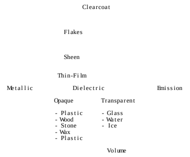

Figure 1: Structure of the Enterprise PBR Shading Model.

The weights m (metallic) and t (transparency) control the blending between the BSDFs, as shown in Figure 2 and described in Section 1.1.4. The values range from 0 to 1 and may vary based on the position on the surface.

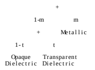

Figure 2: Blending of metallic and dielectric BSDFs based on parameters m (metallic) and t (transparency).

On top of the core, a Fresnel-weighted specular BSDF adds coating effects, see Section 1.5. Moreover, the material can be configured to emit light using the emission component (Section 1.8).

The material can operate in two modes: it can either simulate an infinitesimally thin, two-sided object or a volume boundary. Volume boundaries have to be attached to a closed mesh, i.e., the mesh encloses a volume and does not have any holes. The parameters listed in (Section 1.9) describe what happens inside the volume.

Core

The three BSDFs at the core of the Enterprise PBR Shading Model are built from a small number of building blocks, exploiting as many similarities as possible to reduce the mathematical complexity and, thus, computational effort. In the following, we first describe the individual BSDFs, then we show how to combine them efficiently, and finally we provide detailed equations for the distribution functions.

Metallic BSDF

The metallic BSDF (Figure 3) is the combination of a microfacet BSDF M_r and a configurable Fresnel term F, corrected for energy loss using the terms $M_{r,ms}$ and F_{ms}. Users can change the anisotropic roughness r_u, r_v and the color at normal incidence ρ. The color at grazing angles is fixed to 1. Section 1.1.5 describes the building blocks in more detail, for now we just show how they are combined and weighted.

$$\text{Metallic}(v,l;r_u,r_v,\rho) = \underbrace{M_r(v,l,h_r;r_u,r_v)}_{\text{Microfacet BRDF}} \underbrace{F(\cos\theta_r;\rho,1)}_{\text{Fresnel}} + \underbrace{M_{r,ms}(v,l;r_u,r_v)}_{\text{Multiple-scattering compensation for } M_r} \underbrace{F_{ms}(\rho,1)}_{\text{Multiple-scattering Fresnel}} \tag{1}$$

We parameterize the Fresnel $F(\cos\theta;f_0,f_{90})$ (Schlick approximation) by the angle θ_r, the color at normal incidence f_0, and the color at grazing incidence f_{90}. By computing the angle θ_r from the half-vector h_r, we take the orientation of the microfacets into account.

$$\cos\theta_r = (h_r \cdot v) \tag{2}$$

$$h_r = \frac{v+l}{\|v+l\|} \tag{3}$$

See Section 1.1.5 for more details about the Fresnel term.

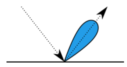

Figure 3: The metallic BSDF is a single specular lobe.

Dielectric BSDF for Transparent Surfaces

The Fresnel-weighted combination of a microfacet BRDF M_r and a microfacet BTDF M_t forms the BSDF for transparent dielectric surfaces like glass, oil, water, or air. As before, r_u and r_v change the anisotropic roughness of the BSDF. In addition, we now provide a parameter to change the index of refraction η, which determines how the light ray is refracted when entering the volume beneath the surface, and how the amount of incoming light is distributed to reflection and transmission. s and ρ_s allow fine-tuning the amount of reflection and its color. ρ changes the color of the transmissive component. We again compensate for energy-loss of the reflection component using $M_{r,ms}$ and F_{ms}. Energy compensation for transmission may be added in a future revision.

The BSDF differs for thin-walled and volumetric materials. The volumetric case has to be enabled via a flag, see Section 1.9. Figure 4 shows the two versions side-by-side. In the thin-walled case (left) the microfacet BTDF is identical to the microfacet BRDF, except that the lobe is mirrored onto the lower hemisphere. In the volumetric case (right) the transmissive lobe is bent according to the refraction angle computed from the index of refraction.

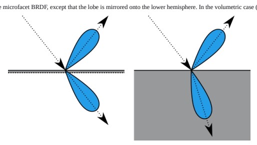

Figure 4: The dielectric BSDF for transparent surfaces is composed of a reflective lobe and a transmissive lobe. The material is configured (left) thin-walled or (right) volumetric. Note the different refraction angles.

The thin-walled dielectric BSDF:

$$\text{ThinTransparentDielectric}(v,l;r_u,r_v,\rho,s,\eta,\rho_s) = s\underbrace{M_r(v,l,h_r;r_u,r_v)}_{\text{Microfacet BRDF}} \underbrace{F(\cos\theta_r;F_0\rho_s,1)}_{\text{Fresnel}} + \rho \underbrace{M_r(v,l',h'_r;r_u,r_v)}_{\text{Microfacet BTDF (flipped BRDF)}} (1 - s\underbrace{F(\cos\theta'_r;F_0\rho_s,1)}_{\text{Fresnel}}) + s\underbrace{M_{r,ms}(v,l;r_u,r_v)}_{\text{Multiple-scattering compensation for } M_r} \underbrace{F_{ms}(F_0\rho_s,1)}_{\text{Multiple-scattering Fresnel}} \tag{4}$$

with

$$l' = l + 2n(-l \cdot n) \tag{5}$$

to flip the light direction l from lower to upper hemisphere in case of thin transmission and

$$\cos\theta_r = (v \cdot h_r) \qquad \cos\theta'_r = (v \cdot h'_r) \tag{6}$$

$$h_r = \frac{v+l}{\|v+l\|} \qquad h'_r = \frac{v+l'}{\|v+l'\|} \tag{7}$$

to compute half vector and Fresnel angle based on the light direction. The Fresnel term at normal incidence depends on the index of refraction:

$$F_0 = \left(\frac{\eta_i - \eta_o}{\eta_i + \eta_o}\right)^2 \tag{8}$$

η_i and η_o determine the index of refraction of the current medium and the medium on the other side of the surface, respectively. They have to be setup properly by the renderer, taking into account the material's η.

The volumetric dielectric BSDF:

$$\text{VolumetricTransparentDielectric}(v,l;r_u,r_v,\rho,s,\eta,\rho_s) = s\underbrace{M_r(v,l,h_r;r_u,r_v)}_{\text{Microfacet BRDF}} \underbrace{F_d(\cos\theta_r;F_0\rho_s,1,\eta_i,\eta_o)}_{\text{Fresnel (refractive version)}} + \rho \underbrace{M_t(v,l,h_t;r_u,r_v,\eta_i,\eta_o)}_{\text{Microfacet BTDF}} (1 - s\underbrace{F_d(\cos\theta_t;F_0\rho_s,1,\eta_i,\eta_o)}_{\text{Fresnel (refractive version)}}) + s\underbrace{M_{r,ms}(v,l;r_u,r_v)}_{\text{Multiple-scattering compensation for } M_r} \underbrace{F_{ms}(F_0\rho_s,1)}_{\text{Multiple-scattering Fresnel}}$$

with same reflection half vector and Fresnel angle as in the thin-walled case, but different calculation for transmission half vector and Fresnel angle:

$$\cos\theta_t = (v \cdot h_t) \qquad h_t = \frac{-v\eta_i - l\eta_o}{\|-v\eta_i - l\eta_o\|} \tag{10}$$

Note that we now also use a different Fresnel term than in the thin-walled version. The Fresnel term F_d takes total internal reflection into account.

Dielectric BSDF for Opaque Surfaces

A lot of dielectric materials, like the majority of plastics, appear semi-transparent or opaque. This results from the strong subsurface scattering due to particles in the volume below the surface interacting with the transmitted light. A diffuse BRDF is a simple approximation for this effect. Consequently, we replace the BTDF introduced before by a diffuse-like BRDF (Figure 5), resulting in a combination of a diffuse-like base BRDF B and a Fresnel-weighted microfacet BRDF M_r. Albedo parameter ρ controls the color of B, all other parameters behave consistent to the transparent BSDF: The amount of specular contribution from the microfacet BRDF is determined by the parameters for specular tint ρ_s and the index of refraction η, which both influence the Fresnel term. In addition, there is a directional-independent scaling factor s. r_u and r_v change the anisotropic roughness of the microfacet BRDF.

Paper, leaves of plants and other thin, dielectric objects transmit some of the incoming light into a random direction to the backside of the surface, the material looks translucent. Translucency is the physical property of a material which allows light to pass through, but not necessarily follow Snell's law on a macroscopic scale. A translucent material is made up of components with different indices of refraction. To achieve this effect, we provide the translucency parameter l that controls the blending between the diffuse BRDF and a diffuse BTDF. The translucency color parameter ρ_l controls the color of the diffusely transmitted part of the light. Besides thin objects, this can be useful for volumetric objects with subsurface scattering and short scattering distances. By scattering the light not only inside the volume, but also already when passing the interface, objects look more blurry and may converge much faster in a ray tracer.

As in the metallic BSDF, the energy loss of the microfacet term is compensated using $M_{r,ms}F_{ms}$.

The BSDF looks as follows:

$$\text{OpaqueDielectric}(v,l;r_u,r_v,\rho,\rho_l,s,\eta,\rho_s,l) = \underbrace{B(v,l;\rho,\rho_l,r_u,r_v,l,s,F_0\rho_s)}_{\text{Diffuse-like base BRDF}} + s\underbrace{M_r(v,l,h_r;r_u,r_v)}_{\text{Microfacet BRDF}} \underbrace{F(\cos\theta_r;F_0\rho_s,1)}_{\text{Fresnel}} + s\underbrace{M_{r,ms}(v,l;r_u,r_v)}_{\text{Multiple-scattering compensation for } M_r} \underbrace{F_{ms}(F_0\rho_s,1)}_{\text{Multiple-scattering Fresnel}} \tag{11}$$

with h_r from Equation 7.

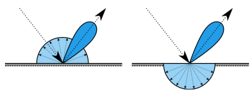

Figure 5: The dielectric BSDF for opaque ($l=0$, left) and translucent ($l=1$, right) surfaces is composed of a diffuse lobe and a specular lobe.

Typically, the base BSDF B is diffuse-like, reflecting the energy that is left after evaluating the microfacet layer equally in all directions. We follow the approach of [KS01] and [KC17] to measure the remaining energy and derive an energy-conserving and energy-preserving diffuse BRDF/BTDF out of it. Controlled by the specular parameter s (i.e., the amount of specular contribution), we gradually blend between this new diffuse BSDF B_c and the classic Lambertian BSDF B_u. The derivation of B_c is explained in more detail in Section 1.1.5 and Section 6.2.

$$B(v,l;\rho,\rho_l,r_u,r_v,l,s,f_0) = B_u(v,l;\rho,\rho_l,l)(1-s) + sB_c(v,l;\rho,\rho_l,r_u,r_v,l,f_0) \tag{12}$$

Putting It All Together

The core BSDF of the Enterprise PBR material is a linear combination of the metallic BSDF, the transparent dielectric BSDF and the opaque dielectric BSDF. The individual BSDFs are weighted by the material parameters m (metallic) and t (transparency).

$$\text{Core}(\rho,\rho_l,s,\eta,\rho_s) = m\,\text{Metallic}(\rho) + (1-m)(1-t)\,\text{OpaqueDielectric}(\rho,\rho_l,s,\eta,\rho_s) + (1-m)t\,\text{TransparentDielectric}(\rho,s,\eta,\rho_s) \tag{13}$$

TransparentDielectric is either the thin-walled BSDF ThinTransparentDielectric or the volumetric BSDF VolumetricTransparentDielectric, depending on the corresponding thin-walled parameter μ (see Table 3).

Table 1 lists all the terms shared in the three BSDFs. In this and the following sections, we will define these terms. With the shared terms and parameters, we significantly reduce the amount of parameters of the complete material.

Term	Description
$M_r(v,l,h_r;r_u,r_v)$	Microfacet BRDF for glossy and perfectly-specular reflection
$M_t(v,l,h_t;r_u,r_v,\eta_i,\eta_o)$	Microfacet BTDF for glossy and perfectly-specular transmission
$M_{r,ms}(v,l;r_u,r_v)$	Multiple-scattering approximation for microfacet BRDF
$F(\cos\theta;f_0,f_{90})$	Fresnel (reflection only)
$F_d(\cos\theta;f_0,f_{90},\eta_i,\eta_o)$	Fresnel (reflection and refractive transmission)
$F_{ms}(f_0,f_{90})$	Multiple-scattering Fresnel (reflection only)
$B_u(v,l;\rho,\rho_l,l)$	Lambertian reflection/translucency BSDF
$B_c(v,l;\rho,\rho_l,r_u,r_v,l,f_0)$	Diffuse-like reflection/translucency BSDF simulating internal scattering layered beneath the microfacet BRDF

Table 1: List of terms that are composed to form the core BSDF.

Evaluating the complete material as sum of all BSDFs is expensive, as each BSDF contains a microfacet model and, therefore, we would have to compute the microfacet distribution multiple times. In order to avoid this overhead, we merge the microfacet models and put the weights m and t into the Fresnel term. This optimization is possible because we use Schlick's Fresnel approximation, see Section 1.1.5.

In case the material is thin-walled, metallic and dielectric BSDFs use the same Fresnel term. The BSDF looks as follows, omitting some parameters for brevity. The complete derivation can be found in Section 6.1.

$$\text{ThinCore}(\rho,s,\rho_s) = M_r F(\rho_{s,r,0},\rho_{s,r,90}) + M_{r,ms}F_{ms}(\rho_{s,r,0},\rho_{s,r,90}) + \rho_t M'_t(1 - sF(F_0\rho_s,1)) + \rho_d[B_u(1-s) + sB_c(F_0\rho_s)] \tag{14}$$

with

$$\rho_{s,r,0} = (1-m)F_0 s\rho_s + m\rho \tag{15}$$

$$\rho_{s,r,90} = (1-m)s + m \tag{16}$$

$$\rho_t = \rho(1-m)t \tag{17}$$

$$\rho_d = (\rho(1-l) + \rho_l l)(1-m)(1-t) \tag{18}$$

If the material is volumetric, the dielectric Fresnel term handles refraction and total internal reflection. Therefore, we cannot simplify as much as before:

$$\text{VolumetricCore}(\rho,s,\eta,\rho_s) = M_r[mF(\rho,1) + (1-m)(1-t)sF(F_0\rho_s,1) + (1-m)tsF_d(F_0\rho_s,1)] + M_{r,ms}F_{ms}(\rho_{s,r,0},\rho_{s,r,90}) + \rho_t M_t(1 - sF_d(F_0\rho_s,1)) + \rho_d[B_u(1-s) + sB_c(F_0\rho_s)] \tag{19}$$

In-Depth Look at the BSDFs

Tangent Space

We first need to define the tangent plane in which the BSDF calculations take place. The mesh data provides us with normal n and tangent t' in world space. The tangent must be aligned with the first axis (u axis) of the texture coordinates on the mesh.

The anisotropic microfacet BSDFs are parameterized with roughness in tangent (r_u) and bitangent (r_v) directions. The two stretch the glossy BSDF lobe in the corresponding direction on the tangent plane. In addition the material provides the parameter anisotropy rotation β that rotates the tangent plane and thus the glossy BSDF lobe. Given n and t', the rotated tangent vector t and bitangent vector b are calculated as follows:

$$t = t'\cos\beta - (n \times t')\sin\beta \tag{20}$$

$$b = n \times t \tag{21}$$

The bitangent b is aligned with the second axis (v axis) of the texture coordinates on the mesh.

The parameterization of anisotropic roughness with r_u and r_v is inconvenient for users. In most cases users want to create isotropic materials. For this reason, we offer the material parameters isotropic roughness r and anisotropy strength o instead. As long as o is kept at its default value of 0, the material is isotropic. Only in specific cases users may optionally add anisotropy by increasing o. r_u and r_v are calculated from r and o as follows:

$$\begin{pmatrix} r_u \\ r_v \end{pmatrix} = r\begin{pmatrix} 1 \\ 1-o \end{pmatrix}$$

(rurv)=r(11−o)(22)

Moreover, being an angle in range 0° to 360° makes ββ difficult to use in textures. Therefore, we scale the range to [0,1][0,1], with 0 corresponding to 0° and 1 corresponding to 360°. The scaled aniostropy rotation parameter is called bb. Please refer to Table 2 for an overview of all parameters and valid ranges offered by the core BSDF of the Enterprise PBR Shading Model.

Microfacet BSDF

Reflection and transmission are based on microfacet BSDFs [Cook82, WMLT07]. As in [Bur12] we square the roughness (ru,rv)(ru,rv) to make it perceptually linear.

Mr(v,l,hr;ru,rv)=D(hr;r2u,r2v)G(v,l,hr;r2u,r2v)4|v·n||l·n|Mt(v,l,ht;ru,rv,ηi,ηo)=|l·ht||v·ht||l·n||v·n|η2oD(ht;r2u,r2v)G(v,l,ht;r2u,r2v)(ηi(v·ht)+ηo(l·ht))2

Mr(v,l,hr;ru,rv)Mt(v,l,ht;ru,rv,ηi,ηo)==D(hr;r2u,r2v)G(v,l,hr;r2u,r2v)4|v·n||l·n||l·ht||v·ht||l·n||v·n|η2oD(ht;r2u,r2v)G(v,l,ht;r2u,r2v)(ηi(v·ht)+ηo(l·ht))2(23)(24)

The BSDFs operate in the rotated local coordinate system (right-handed) defined by normal nn, tangent tt and bitangent bb, see Figure 6. We can convert between polar angle θθ, azimuthal angle ϕϕ, and vectors as follows:

h·t=sinθhcosϕhh·b=sinθhsinϕhh·n=cosθh

h·th·bh·n===sinθhcosϕhsinθhsinϕhcosθh(25)

The half vector hh can be either the reflection half vector hrhr or the transmission half vector htht.

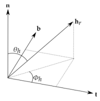

Figure 6: Local coordinate system (t,b,n)(t,b,n) in which the BSDFs are evaluated and exemplary reflection half vector hrhr with corresponding polar angle θhθh and azimuthal angle ϕhϕh.

For the microfacet distribution, the material uses the Generalized Trowbridge-Reitz GTR model (generalized anisotropic GGX) [Bur12] with γ=2γ=2. Using Equation 2525 we have

D(h;αu,αv)=1παuαvχ+(h·n)((h·t)2α2u+(h·b)2α2v+(h·n)2)2

D(h;αu,αv)=1παuαvχ+(h·n)((h·t)2α2u+(h·b)2α2v+(h·n)2)2(26)

with the step function

χ+(x)={1if x>00otherwise

χ+(x)={1if x>00otherwise(27)

The appropriate Smith shadowing-masking term (height-correlated) [Hei14] for GGX:

G(v,l,h;αu,αv)=χ+(l·h)χ+(v·h)1+Λ(v;αu,αv)+Λ(l;αu,αv)Λ(x;αu,αv)=−1+√1+1a221a2=1(x·n)2((x·t)2α2u+(x·b)2α2v)

In the isotropic case (α=αu=αv), the distribution simplifies to:

D(h;α)=1πα2((h·n)2(α2−1)+1)2

Correspondingly, 1a2 in the shadowing-masking term simplifies to:

1a2=α2(1(x·n)2−1)

Fresnel

We use Schlick's Fresnel approximation for performance reasons (Section 1.1.4). For thin-walled materials and the metallic BSDF we ignore refraction and total internal reflection, leading to a very simple form of the Fresnel term F. Taking both into account results in Fd, which is used for volumetric materials.

F(cosθ;f0,f90)=f0+(f90−f0)(1−|cosθ|)5Fd(cosθ;f0,f90,ηi,ηo)={f0+(f90−f0)(1−|cosθ|)5if ηo≥ηif0+(f90−f0)(1−|cosθo|)5if ηo<ηi∧sin2θo<11if ηo<ηi∧sin2θo≥1

where θo is the angle of transmission, computed from the angle of incidence using sin2θo=(ηiηo)2(1−cos2θ) [Lag13]. Note that in case of total internal reflection, the Fresnel does not depend on f90 anymore, so it is impossible to disable the reflection in this situation. This allows users to change the specular component of a refractive object without introducing artifacts, i.e., dark appearance in areas where total internal reflection occurs.

Energy Preservation

The standard microfacet model does not account for scattering occuring between microfacets on the microsurface. As the surface becomes rougher, the multiple scattering becomes stronger, effectively resulting in high energy loss if this is not accounted for (see Figure 7). [KC17] describes a method based on [KS01] to approximate multiple scattering for microfacet models that can be adapted for real-time rendering [Fde19].

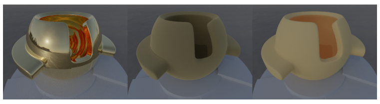

Figure 7: Left: Metallic surface with roughness 0. Middle: Non-energy preserving metallic surface with roughness 1. Right: Energy-preserving metallic surface with roughness 1 by accounting for multiple scattering. Note that the surface in the middle looks significantly darker than the others.

Following the approach of [KS01] and [KC17], we measure the directional albedo Em(cosθ;ruv) and average albedo Em,avg(ruv) by integrating the cosine-weighted single-scattering GGX microfacet model over all light directions. We precompute the integrals and store the result in lookup tables. In order to reduce the number of parameters, we replaced the anisotropic roughness ru and rv with an isotropic approximation ruv.

Based on the measured albedo, we build an energy compensation term Mr,ms(v,l;ru,rv)Fms(f0,f90). The term is specifically tailored to Mr and as such added in all places where Mr is present.

Mr,ms(v,l;ru,rv)Fms(f0,f90)=(1−Em(v·n;ruv))(1−Em(l·n;ruv))π(1−Em,avg(ruv))_Multiple-scattering GGXFm,avg(f0,f90)2Em,avg(ruv)1−Fm,avg(f0,f90)(1−Em,avg(ruv))_Multiple-scattering Fresnel

with ruv=√rurv. In the isotropic case, ruv=ru=rv.

Diffuse BRDF

The Lambertian diffuse BRDF/BTDF Bu(v,l;l) is described by the following equation:

Bu(v,l;l)=1πT(v,l;l)

with

T(v,l;l)={(1−l) if (v·n)(l·n)>0l otherwise

The term (v·n)(l·n) is positive if v and l point into the same hemisphere wrt. n. This enables the diffuse reflection (BRDF). The term is negative if v and l point into opposite hemispheres wrt. n. This enables the diffuse transmission/translucency (BTDF). The parameter l (translucency) determines the ratio between reflection and transmission.

As shown in Section 1.1.3, Bu is only used for the non-specular fraction of the dielectric BSDF. As soon as the microfacet BRDF Mr blends in via s, we have to deduct its energy from the diffuse term.

Following the approach of [KS01] and [KC17], we measure the directional albedo E(cosθ;ruv,f0) of the Fresnel-weighted microfacet GGX BRDF including the multi-scatter approximation. From this we build a complementary diffuse BRDF Bc that, when added to Mr and Mr,ms, results in an energy-conserving and energy-preserving total BRDF. Section 6.2.2 describes the process in more detail. As in Section 1.1.5.4, we use isotropic roughness ruv=√rurv.

Bc(v,l;ru,rv,l,f0)=(1−E(v·n;ruv,f0,max))(1−E(l·n;ruv,f0,max))π(1−Eavg(ruv,f0,max))T(v,l;l)

We again use T(v,l;l) to blend between diffuse reflection and transmission based on the translucency parameter l. f0,max denotes the maximum value of the wavelength-dependent color f0 in linear sRGB,E color space. The maximum is used because the color is subtracted, leading to unexpected behavior if this is done per channel.

Directional albedo E(cosθ;ruv,f0) and average albedo Eavg(ruv,f0) are expensive to compute, as it involves integrating the Fresnel-weighted GGX microfacet BRDF over all light directions on the hemisphere. Therefore, we provide precomputed lookup tables for download in Section 6.2.2.

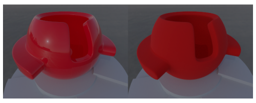

Figure 8: Combination of diffuse and specular component with energy compensation for roughness 0 (left) and 1 (right). Note that the combination is energy-conserving (no energy is produced, even at multiple bounces inside the object) and energy-preserving (especially at grazing angles).

Parameters

Table 2 and Figure 9 describe the user parameters of the Enterprise PBR Shading Model. Most parameters can be textured, i.e., the value may vary across the surface. Some of them cannot be textured. Parameters that cannot be textured have the *uniform* modifier attached to their type.

Name	Type	Default	Range	Description
μ	uniform bool	true		Material describes volume boundary (false) or thin, two-sided object (true).
ρ	color	1	[0,1]	Albedo of the material.
ρs	color	1	[0,1]	Specular tint to adjust the color of the specular lobe near θ=0∘.
m	float	0	[0,1]	Metallic-ness (0 = dielectric, 1 = metallic).
s	float	1	[0,1]	Directional-independent amount of specular contribution to fine-tune the effect of the index of refraction.
l	float	0	[0,1]	Translucency (0 = opaque, 1 = translucent).
ρl	color	1	[0,1]	The color of the diffusely transmitted light.
t	float	0	[0,1]	Transparency (0 = opaque, 1 = transparent).
η	uniform float	1.5	[1,inf)	Index of refraction of the volume (t=1) or the dielectric coating (m=0).
r	float	0	[0,1]	Roughness of microfacet distribution.
o	float	0	[0,1]	Anisotropy strength.
b	float	0	[0,1]	Anisotropy rotation angle (counter-clockwise rotation of the tangent space around local normal, where 0.25=90∘, 0.5=180∘, 1.0=360∘.
n	float3	(0,0,1)	‖n‖=1	Normal for normal mapping.

Table 2: List of core parameters.

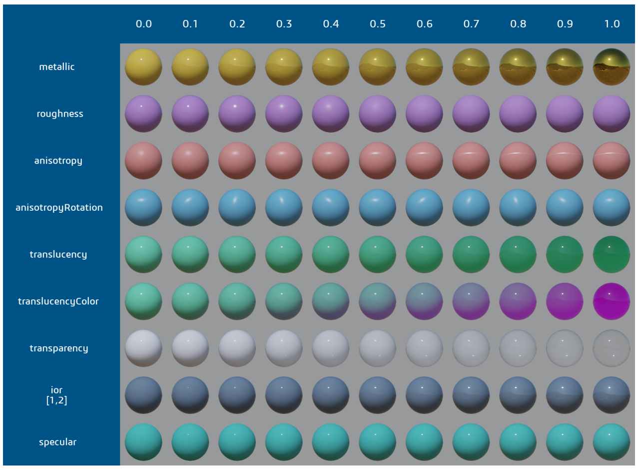

Figure 9: Examples for various values of core parameters.

As the list of parameters might be overwhelming at first, Table 3 shows how to use the main parameters *metallic*, *specular* and *transparency* for different kinds of materials.

m	s	t	Active Components	Use this for	Description
0	0	0	Diffuse		Pure diffuse material.
0	0	1	Refraction		Pure refractive material.
0	1	0	Diffuse, Reflection	Plastic, wood, stone	Specular coating on top of diffuse base layer. Adjust ηo to change the index of refraction of the coating. For artistic effects, adjust s and ρs to fine-tune the coating layer.
0	1	1	Reflection, Refraction	Glass, water, ice	Transparent material. Use ηo to adjust the index of refraction. For artistic effects, adjust s and ρs to fine-tune the reflection.
1	N/A	N/A	Reflection	Metal	Specular material with Fresnel effect that fades to white at grazing angles. Metallic takes precedence, s and t have no effect.

Table 3: Metallic, Specular and Transparency.

The basic workflow to configure a physically-plausible material involves three steps:

Choose the color ρ of the material.

Choose the material type with m and t.

Choose an IOR for the coating/refraction.

For more fine-grained artistic control, the material can now be tweaked even further:

Fine-tune amount of specular contribution with s.

Fine-tune specular color near θ=0∘ with ρs.

Thin Film

Thin-film interference is an optical phenomenon in which a material's surface changes color when viewed or illuminated from different angles (*goniochromatic*). The effect is responsible for the vibrant colors observed in soap bubbles or oil slicks. Practical industry applications include the design of anti-reflective coatings, interference filters for cameras, and the enhancement of colors in paints. In a more subtle manner, these color shifts can be observed in everyday scenarios such as finger traces or surface finishes on furniture.

Thin-film interference manifests as an iridescent effect resulting from the interference between light waves that scatter in a wavelength-dependent manner when interacting with films of thickness close to visible wavelengths. This interference occurs at both the upper and lower interfaces of the thin layer, influencing whether reflected light is suppressed or enhanced.

The appearance of the thin-film effect is controlled by the film's layer weight i, the thin-film thickness ti and its IOR ηi.

Name	Type	Default	Range	Description
i	float	0.0	[0,1]	Weight of the thin-film layer.
di	float	0.4	[0,1]	Thickness of the thin-film layer in microns.
ηi	float	1.3	[1,2.5]	Index of refraction of the thin-film layer.

Table 4: List of thin-film parameters.

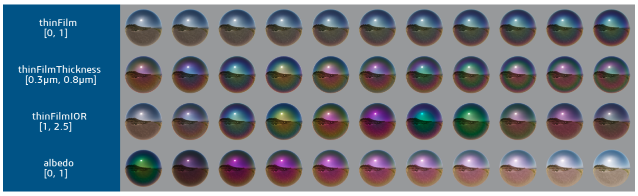

Figure 10: Metal spheres with thin-film coating.

We model the thin-film effect by introducing a thin layer on top of our core material, as proposed by [BB17]. This layer approximates iridescence by a physically plausible simulation, leveraging the principles of wave optics. Evaluating it effectively replaces our Fresnel term F(cosθ,f0,f90) (Section 1.1.5.3) with a more intricate term R(cosθ,η,di,ηi), the thin-film reflectance. The new term comprehensively considers all inter-reflections inside the thin-film layer, encompassing constructive and destructive interference effects.

In the following, we reformulate the core components as given in Section 1.1 taking into account the thin-film reflectance.

ThinFilmOverMetallic(v,l;ru,rv,ρ,i,di,ηi)=(1−i)Metallic(v,l;ru,rv,ρ)+ i [Mr(v,l,hr;ru,rv)R(cosθr,ηρ,di,ηi)+Mr,ms(v,l;ru,rv)Fms(ρ,1)]

where

ηρ=1.0+√max(ρ)1.0−√max(ρ)

ThinFilmOverThinTransparentDielectric(v,l;ru,rv,ρ,s,η,ρs,i,di,ηi)= (1−i)ThinTransparentDielectric(v,l;ru,rv,ρ,s,η,ρs)+ i [Mr(v,l,hr;ru,rv)R(cosθr,η,di,ηi)+ρMr(v,l′,h′r;ru,rv)(1−R(cosθ′r,η,di,ηi))+sMr,ms(v,l;ru,rv)Fms(F0ρs,1)]

ThinFilmOverVolumetricTransparentDielectric(v,l;ru,rv,ρ,s,η,ρs,i,di,ηi)= (1−i)VolumetricTransparentDielectric(v,l;ru,rv,ρ,s,η,ρs)+ i [Mr(v,l,hr;ru,rv)R(cosθr,η,di,ηi) +ρMt(v,l,ht;ru,rv)(1−R(cosθt,η,di,ηi)) +sMr,ms(v,l;ru,rv)Fms(F0ρs,1)]

ThinFilmOverOpaqueDielectric(v,l;ru,rv,ρ,ρl,s,η,ρs,l,i,di,ηi)= (1−i)OpaqueDielectric(v,l;ru,rv,ρ,ρl,s,η,ρs,l)+ i [Bu(v,l;ρ,ρl,l)(1−max(R(v·n;η,di,ηi),R(l·n;η,di,ηi))) +Mr(v,l,hr;ru,rv)R(cosθr,η,di,ηi) +sMr,ms(v,l;ru,rv)Fms(F0ρs,1)]

With these reformulated components we can also adjust the definitions of the core BSDFs that were defined in Section 1.1.4

ThinFilmOverCore(ρ,ρl,s,η,ρs,i,di,ηi)=mThinFilmOverMetallic(ρ,i,di,ηi)+(1−m)(1−t)ThinFilmOverOpaqueDielectric(ρ,ρl,s,η,ρs,i,di,ηi)+(1−m)tThinFilmOverTransparentDielectric(ρ,s,η,ρs,i,di,ηi)

ThinFilmOverThinCore(ρ,s,ρs,i,di,ηi)=(1−i)MrF(ρs,r,0,ρs,r,90)+(1−i)ρd[Bu(1−s)+sBc(F0ρs)]+(1−i)ρtMt′(1−sF(F0ρs,1))+Mr,msFms(ρs,r,0,ρs,r,90)+ iρtM′t(1−R(cosθt,η,di,ηi))+ iρdBu(1−max(R(v·n;η,di,ηi),R(l·n;η,di,ηi)))+i(1−t)Mr[(1−m)R(cosθr,η,di,ηi)+mR(cosθr,ηρ,di,ηi)]

ThinFilmOverVolumetricCore(ρ,s,η,ρs,i,di,ηi)=Mr[(1−i)mF(ρ,1)+ imR(cosθr,ηρ,di,ηi)+ (1−i)(1−m)(1−t)sF(F0ρs,1)+ i(1−m)(1−t)R(cosθr,η,di,ηi)+ (1−m)tsFd(F0ρs,1)]+Mr,msFms(ρs,r,0,ρs,r,90)+(1−i)ρtMt(1−sFd(F0ρs,1))+iρtMt(1−R(cosθt,η,di,ηi))+ρd[Bu(1−s)+sBc(F0ρs)]+iρdBu(1−max(R(v·n;η,di,ηi),R(l·n;η,di,ηi)))

Sheen

For rendering cloth-like materials diffuse and specular BRDFs are not sufficient. Fibers standing perpendicular to the surface lead to backscattering, brightening the rims of objects. We simulate this effect with a microfacet-based sheen BRDF Mb, layered on top of the core described in Section 1.1. Its distribution with roughness rb generates micro-fiber normals, centered around grazing directions. The color is controlled using sheen color parameter ρb. The BRDF and layering technique is taken from [CK17].

Name	Type	Default	Range	Description
ρb	color	0	[0,1]	Sheen color.
rb	float	0.5	[0,1]	Roughness of sheen microfacet distribution.

Table 5: List of sheen parameters.

Sheen is layered on top of the core like a coating [CK17]. Similar to the diffuse BRDF (Section 6.2.2), we scale the bottom layer using the directional albedo of the top layer, resulting in an energy-conserving and (almost) energy-preserving mix of the BSDFs. The directional albedo of the sheen BRDF is fetched from a lookup table.

ShinyCore(v,l;ρb,rb,…)=ThinFilmOverCore(v,l;…)min(1−^ρbEb(v·n;rb),1−^ρbEb(l·n);rb)+Mb(v,l,hr;rb)ρb

As in Section 6.2.2, we subtract the maximum value ^ρb=maxρb of the sheen color to avoid unexpected color shifts.

The sheen BRDF follows the usual microfacet equation. Note that the distribution roughness αb is the squared sheen roughness: αb=r2b.

Mb(v,l,hr;rb)=Db(hr;r2b)Gb(v,l,hr;r2b)4|v·n||l·n|

The distribution of normals as given in [CK17]:

Db(h;αb)=(2+1/αb)sin1/αbθb2πsinθb=√1−(h·n)2

Since there is no easy analytical solution for the shadowing-masking term, [CK17] compute the term numerically and use curve fitting. Although their fit is good, the error becomes quite noticable at small roughness values. The final sheen BRDF shows significant brightening at grazing angles in a white furnace test. For this reason we refined the fit to reduce the error, in particular near αb=0. The key improvement is the new interpolation function for intermediate roughness values, which in our case is a bit more costly to compute.

Gb(v,l,h;αb)=χ+(l·h)χ+(v·h)1+Λ(v·n;αb)+Λ(l·n;αb)Λ(cosθ;αb)={eL(|cosθ|;αb)if |cosθ|<0.5e2L(0.5)−L(1−|cosθ|;αb)otherwiseL(x;αb)=a01+a1xa2+a3x+a4

	a0	a1	a2	a3	a4
p0	11.9095	4.68753	0.33467	−2.22664	−1.76591
p1	13.7000	2.92754	0.28670	−0.81757	−1.22466

Table 6: Values for L(x;αb) used in Λ curve fitting. Parameters are interpolated according to (1.0−w(αb))p1+w(αb)p0.

w(αb)=c01+c1αbc2+c3

c0	c1	c2	c3
−1.59612	0.20375	−0.55825	1.32805

Table 7: Constants for w(αb).

If performance is a concern, due to the computation cost of the term above, we propose an alternative visibility term, given in [NP13]:

Mb(v,l,hr;rb)=Db(hr;r2b)4((v·n)+(l·n)−(v·n)(l·n)) The visual differences between the main model and the alternative model can be interactively observed on the following ShaderToy: https://www.shadertoy.com/view/wl3SWs.

Figure 11: Main sheen model for growing roughness.

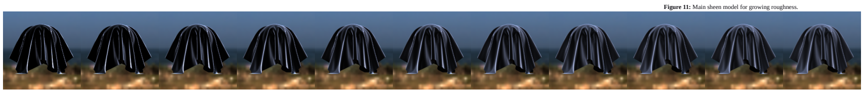

Figure 12: Alternative sheen model for growing roughness.

We can see through Figure 11 and Figure 12 that consistency gets better as roughness grows. The alternative model is noticeably brighter for very low roughness values. The directional albedo changes can be found in Section 6.2.4

Flakes

Some materials, such as car paint or grainy metals, exhibit shifting random patterns of bright sparkles or glints. These glints may suddenly appear or disappear if light or view direction changes. In car paints, for example, this phenomenon comes from small mirror-like flakes distributed in the material below its coating. This can be modeled with the flakes layer of the Enterprise PBR Shading Model.

Parameters

Name	Type	Default	Range	Description
f	float	0	[0,1]	Flake coverage (0 = no flakes, 1 = only flakes).
ρf	color	1	[0,1]	Flake color.
sf	float	0	[0,inf)	Flake size (diameter) in mm in world space.
rf	float	0	[0,1]	Roughness of distribution that controls flake orientation.
ff	float	0	[0,1]	Flip flop effect (0 = none, 1 = full).
ρff	color	1	[0,1]	Flake color at normal incidence if flip flop effect is enabled.

Table 8: List of flakes parameters

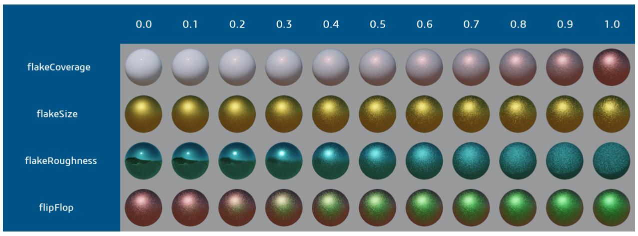

Figure 13: Renderings with various values for flake parameters.

Flake coverage f determines the probability for a flake to be opaque. Flakes are layered on top of the sheen as seen on the Figure 1. It's a boolean layering operation which shows either the flakes or the underlying material, depending on the transparency of individual flakes.

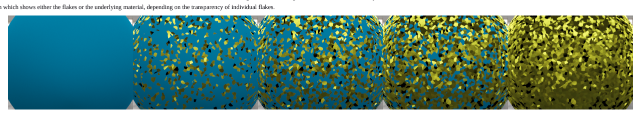

Figure 14: Close-up look at effect of increasing flake coverage f: 0.0, 0.25, 0.5, 0.75, 1.0.

Flake size sf chooses the average flake diameter in millimeters. Thus the conversion factor from scene unit to physical length unit must be known by the renderer in order to guarantee the correct appearance.

The total roughness rf is distributed mostly to the orientation roughness of the normals and to a smaller part to the cone angle of the cone BSDF used for the individual flakes (Equation 64). To determine the distribution of flake orientations we sample from a GGX normal distribution function.

The flake color ρf determines the color of the individual flakes. This is sufficient for a large variety of car paints. There are, however, some paints that exhibit strong directional-dependent color effects. These are often referred to as “flip flop effects” or “pearlescent paints”. To mimic these effects, we provide an additional light-facing color ρff and an intensity parameter ff.

Layering

With flake opacity texture of(p;f,sf), flake normal texture nf(p;f,sf,rf) and flake flip-flop color cf(v,l;ρf,ff,ρff) the layers are combined as follows:

GlitteryCore(v,l;f,sf,rf,ρf,…)=ShinyCore(v,l;…)(1−of(p;f,sf))+cf(v,l;ρf,ff,ρff)Bf(v,l;rf,nf)of(p;f,sf)

Note that p is a shading point coordinate defined in object space.

Flake Texture

Flake texture is fully procedural and is evaluated on the fly during shading. There are in fact 2 textures generated simultaneously: flake opacity texture of(p;f,sf) and flake normal texture nf(p;f,sf,rf). We procedurally generate the total number of flakes N to be distributed over 1m2 texture space from the user-given flake size sf. Flake texture is represented as a Voronoi diagram where individual flakes are represented as Voronoi cells.

rflake=11000(sf2)flake radius in mAavgcell=3√32r2flakeaverage hexagonal cell area in m2N=1/Aavgcellnumber of Voronoi cells per m2≈1.5396·106s2f

Now with the number of flakes N in place we determine the resolution of a regular grid where each grid cell will be responsible for generating a fixed number k of flakes.

res=max(4,⌈√Nk⌉)

We chose k=4 but this can be changed to optimize performance and/or for getting more randomness due to less stratified flake positions at higher k (Section 6.3).

Now that we have the resolution to house all k·res2≈N flakes, each grid cell needs to be able to be probed for its flakes. In order to do that each grid cell creates an initial seed value by inputting its integer coordinates (ij)=(⌊res·mod(u,1)⌋⌊res·mod(v,1)⌋)∈[0,res−1]×[0,res−1] into a hash function. The hash is then used to seed an PRNG which is then used to produce all random numbers needed to produce the flake position, flake direction and opacity for each of the k flakes in the grid cell. Procedural normal map is mapped onto each object via triplanar mapping (Equation 62). Triplanar mapping does surface parameterization based on a object space shading point p and bumped surface normal n simply by projecting it along the dominant orthogonal direction of n and returns parametric coordinates (u, v) on a surface. Tmap(p,n)={(frac(+sgn(px)py+12),frac(pz+12))if |nx|>|ny|∧|nx|>|nz|(frac(−sgn(py)px+12),frac(pz+12))if |ny|>|nx|∧|ny|>|nz|(frac(+sgn(pz)px+12),frac(py+12))otherwise Where sgn(x) and frac(x) are simple functions that return sign and fractional part of the floating point number. In order to find a flake under the shading point we go through following steps:

Determine (u, v) texture coordinates of the shading point using triplanar mapping (Equation 62).

Determine which flake grid cell it belongs (Equation 61).

Iterate over grid cells which can potentially contain closest flake.

Hash current grid cell coordinate and initialize PRNG seed with it.

Generate k number of flakes per cell, check and find the closest to (u, v).

Those steps are also shown in Listing 1.

```
(flake_position, flake_normal, flake_opaqueness) generate_flake(float r_f, float f, uint32 s) {
  (xi_1, xi_2) = prng(s);   //random numbers for flake position within the grid cell
  (xi_3, xi_4) = prng(s);   //random numbers for flake orientation
  xi_5 = prng(s);           //random number for flake transparency
  return ((xi_1, xi_2), K(r_f, xi_3, xi_4), O_f(f, xi_5));
}
```

Flake generation for a (i, j) grid cell uses random number generator for generating 5 sequential random numbers for a single flake. As you can see we use first two random numbers from PRNG as a flake coordinate within the grid cell. For more information about PRNG please see section 6.3.

In order to generate flake normal we sample GGX normal distribution:

K(rf,ξ1,ξ2)=(√1−wcos(2πξ2),√1.0−wsin(2πξ2),√w)

where w=1−ξ11−(1−α2ggx)ξ1. αggx which controls GGX distribution depends on a rf paramenter as well as BSDF cone angle αcone (Equation 69).

αggx=√r2f−α2cone

During flake generation we also call Of to get flake opaqueness. Of(f,ξ)={1if ξ<f0otherwise

Based on a flake coverage and uniform random number unique per flake Of(f,ξ) returns 1 for opaque flakes and 0 for transparent ones.

We can now derive the textures of(p;f,sf) and nf(p;f,sf,rf) as follows:

```
(o_f, n_f) generate_textures(float3 p, float f, float s_f, float r_f) {
  //Determine (u, v) texture coordinates of the shading point using triplanar mapping
  uv = T_map(p, ...)

  //Determine which flake grid cell it belongs
  i, j = grid_cell(uv)

  //Iterate over grid cells which can potentially contain closest flake.
```

```
  for(ii=i-1; ii<=i+1; ii++) {
    for(jj=j-1; jj<=j+1; jj++) {
      //Hash current grid cell coordinate and initialize PRNG seed with it
      s = initPRNG(hash(ii, jj))
      nearest_d = inf
      //Generate k number of flakes per cell, check and find the closest to (u, v)
      for(fl=0; fl < k; fl++) {
        flake_pos, flake_normal, flake_opacity = generate_flake(f, r_f, s)
        voronoi_cell_d = length(uv-flake_pos);
        if(voronoi_cell_d < nearest_d) {
          nearest_o = flake_opacity
          nearest_n = flake_normal
        }
      }
    }
  }

  return nearest_o, nearest_n
}
```

Listing 1: Sampling the procedural flake textures $o_f(p;f,s_f)$ and $n_f(p;f,s_f,r_f)$.

Flake BSDF

Generated flakes have a very simple BSDF. They behave like blurred mirrors, so they reflect light uniformly within a small cone aligned to a perfect reflection direction. Solid angle of the cone is defined as $\Omega_{cone}(\cos\theta)=2\pi(1-\cos\theta)$

Taking it into account the flake BSDF looks like this : $B_f(v,l;r_f,n_f)=\begin{cases}\frac{1}{\Omega_{cone}(\cos\theta_{cone})} & \text{if } (l\cdot r)>\cos\theta_{cone}\\ 0 & \text{otherwise}\end{cases}$ with $r=2n_f(v\cdot n_f)-v$, which is the reflection of the view vector v on the generated flake with a normal n_f. θ_{cone} mentioned above is small to guarantee glittery appearence of flakes and depends only on r_f parameter.

$$\cos\theta_{cone}=\frac{1-2\alpha^2_{cone}}{1+\alpha^2_{cone}} \quad \alpha^2_{cone}=\min\left(\frac{\Omega_{cone}(\cos(\theta'_{cone}))}{4\pi-\Omega_{cone}(\cos(\theta'_{cone}))},\frac{r_f}{2}\right)$$

where target cone angle $\theta'_{cone}=7^\circ$.

Flip-Flop Effect

The flake BSDF is tinted with flake color ρ_f and flip flop color ρ_{ff}. The two colors are mixed depending on the angle between view direction and flake normal. As the flakes itself are not perfectly specular, we take the reflection half-vector instead of the flake normal. This ensures that the BRDF is reciprocal.

The flip flop strength f_f determines how much of the flip flop color is visible. A strength of 0 disables the effect.

The function c_f that computes the angle-dependent flake tint is given as follows:

$$c_f(v,l;\rho_f,f_f,\rho_{ff})=\rho_f+(\rho_{ff}-\rho_f)(v\cdot h_r)^{\frac{1}{f_f}} \quad h_r=\frac{v+l}{\|v+l\|}$$

Instead of $(v\cdot h_r)$ renderers may also choose the dot product of view vector and surface normal $(v\cdot n)$. This may give more accurate results with certain approximations like preconvolved IBL.

Efficient Implementation

Rendering flakes is a challenging task for both offline and real-time renderers. While offline renderers solve pixel aliasing via Monte Carlo method, real-time renderers don't have computational budget high enough to afford that. Calculating reflected light on a flaky surface is very hard when single pixel contains hundreds or thousands of flakes. In this case raw point based Monte Carlo method can cause extremely high variance and therefore suffer from an aliasing artifacts, on the other hand none of the existing denoisers work well with flakes because it's hard to distinguish it from actual noise. So it becomes necessary to do some type of filtering of the reflection, which will efficiently find most of the flakes under the pixel, which reflect light towards the viewer [Jak14, AK16]. Additional optimization can be done for a flaky surface seen from a large distance. When the number of flakes per pixel gets too high the reflection becomes smoother and therefore could be well approximated with Cook-Torrance reflection model with GGX distribution [Cook82].

Clearcoat

The clearcoat component adds a dielectric coating as a specular layer on top of the core described in Section 1.1. $GlitteryCore(v,l)$ and coating are weighted with a Fresnel term. The clearcoat reflectivity parameter c determines the strength of the effect.

$$Coated(v,l;c,r_c,n_c,\dots)=GlitteryCore(v,l;\dots)(1-c\max(F_c(v\cdot n_c),F_c(l\cdot n_c)))+M_c(v,l,h_r;r_c,n_c)\,c\,F_c(h\cdot v)$$

The coating is based on a microfacet BRDF with an isotropic GGX distribution and squared roughness. Distribution and shadowing-masking function are identical to the reflection component in the core material (Section 1.1.5.2).

$$M_c(v,l,h_r;r_c)=\frac{D(h_r;r^2_c,r^2_c)G(v,l,h_r;r^2_c,r^2_c)}{4|v\cdot n_c||l\cdot n_c|}$$

The index of refraction in Schlick's Fresnel term is fixed to 1.5 to resemble a glass-like dielectric coating.

$$F_c(\theta)=0.04+0.96\,(1-\cos\theta)^5$$

As shown in Table 9, the clearcoat is parameterized using two scalar values and a normal.

Name	Type	Default	Range	Description
c	float	0	[0,1]	Clearcoat reflectivity (0 = no clearcoat, 1 = full clearcoat).
r_c	float	0	[0,1]	Roughness of clearcoat's microfacet distribution.
n_c	float3	(0,0,1)	$\|n_c\|=1$	Clearcoat normal for normal mapping.

Table 9: List of clearcoat parameters.

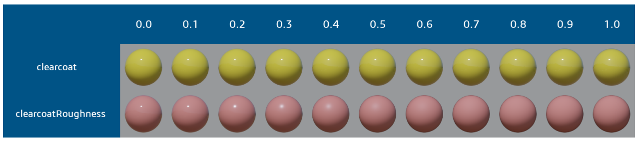

Figure 15: Renderings with various values for the clearcoat parameters.

Cut-out

The cut-out acts like an alpha channel for the surface. It makes parts of the surface partially or fully invisible, independent of any BSDF effects.

Name	Type	Default	Range	Description
α	float	1	[0,1]	Cut-out opacity value (0 = transparent, 1 = opaque).

Table 10: List of cut-out parameters.

Figure 16: Renderings with various values for the cut-out parameters.

The cut-out parameter is primarily used in conjunction with textures to fake details in thin-walled geometry, see Figure 17. The behavior of using cut-out on transparent and translucent volume boundaries in combination with attenuation and subsurface scattering is implementation-defined.

Figure 17: Cut-out texture applied to a sphere.

Displacement

Displacement mapping modifies the position p of surface points to add details to geometry. The displacement parameter h is a vector, specifying the direction and length of the displacement in the local surface coordinate system defined by normal n, tangent t, and bitangent b (see Section 1.1.5.1). The new position p' is given as

$$p'=p+th_x+bh_y+nh_z$$

Name	Type	Default	Range	Description
h	float3	(0,0,0)	(-inf,inf)	Displacement in the direction of the surface normal.

Table 11: List of displacement parameters.

Emission

In addition to the reflective and transmissive components, the material contains an emissive component that turns surfaces into diffuse area lights. Light is emitted into the upper hemisphere with regard to the normal. The emission component is below the clearcoat in the layering stack, and thus it is scaled by clearcoat reflectivity c and clearcoat Fresnel F_c, see Section 1.5.

$$L_e(v)=\begin{cases}\rho_e(1-cF_c(v\cdot n_c)) & \text{if } (v\cdot n)>0\\ 0 & \text{otherwise}\end{cases}$$

ρ_e is luminance given in $\frac{lm}{m^2\,sr}$. It is computed from the parameters given in Table 12.

Name	Type	Default	Range	Description
l_c	color	1	[0,inf)	Emission color.
l_v	uniform float	0	[0,inf)	Emission value in lx ($=\frac{lm}{m^2}$) (luminous emittance, the relative power leaving the object per surface area) or lm (luminous power, the absolute power leaving the object). The parameter for unit selection may be called “emission mode”.
l_{norm}	uniform bool	false		Normalize emission color.

Table 12: List of emission parameters.

Figure 18: Renderings with various values for the emission parameters.

Lights are defined in photometric units. This means that the energy is spectrally weighted with the response curve of the human eye.

The emission is provided as color l_c and energy value l_v. We expect the color to be in linear sRGB color space with a D65 whitepoint. If the energy normalization flag l_{norm} is enabled, the color will be normalized by its luminance. This behavior is needed for physical workflows and it guarantees that the light exactly emits the specified value in its respective physical units.

The following equations show how to compute ρ_e from these parameters. $area$ denotes the area of the entire surface of an object in square meters.

$$l_{vc}=\begin{cases}l_v l_c & \text{if } l_{norm} \text{ is false}\\ \frac{l_v l_c}{l_c\cdot(0.2126729\ 0.7151522\ 0.0721750)} & \text{otherwise}\end{cases} \quad \rho_e=\begin{cases}\frac{1}{\pi\,area}l_{vc} & \text{if } l_v \text{ is given in lm}\\ \frac{1}{\pi}l_{vc} & \text{if } l_v \text{ is given in } \frac{lm}{m^2}\end{cases}$$

Volume

If the thin-walled material parameter is set to false, the surface to which the material is applied is treated as the volume boundary. Light rays falling to such a boundary may enter the volume, depending on the transparency or translucency of the surface's BSDF. Inside the volume light might be absorbed or scattered by particles, and eventually hit some surface from inside the volume. It can be a different surface and will most probably be a different point in space compared to the entrance point. At this exit point, if the material properties of the surface allow, light may leave the volume.

Absorption is controlled by the absorption coefficient σ_a; subsurface scattering by the scattering coefficient σ_s and the phase function $p(v,l)$. The coefficients σ_a and σ_s are wavelength-dependent values in range [0,inf). Such an infinite range makes them hard to be controlled by the users.

To provide convenient parameterization for users we introduce three new user-friendly parameters: attenuation color a, attenuation distance d, and subsurface color ρ_{ms}. Attenuation color and distance are mapped to the attenuation coefficient σ_t (or its inverse, the mean free path length mfp). Subsurface color is mapped to the single-scattering albedo ρ_{ss} [KC17].

$$\sigma_t=\frac{1}{mfp}=-\frac{\log(a)}{d} \quad \rho_{ss}=1-(4.09712+4.20863\rho_{ms}-\sqrt{9.59217+41.6808\rho_{ms}+17.7126\rho^2_{ms}})^2$$

From this we derive the absorption and scattering coefficient.

$$\sigma_a=\frac{1-\rho_{ss}}{mfp}=\sigma_t(1-\rho_{ss}) \quad \sigma_s=\frac{\rho_{ss}}{mfp}=\sigma_t\rho_{ss}=\sigma_t-\sigma_a$$

Intuitively, the mean free path length describes the reciprocal of the density of the medium. The denser the medium, the shorter the distance until a particle is hit and, therefore, the more light is absorbed or scattered out. In our case, after the light traveled d units into the medium, all that remains is the color a.

When a particle is hit, the light color gets multiplied by the single-scattering color ρ_{ss}. On the surface, the color ρ_{ms} can be observed from the multiple scattering events occurring in the medium.

In addition the phase function $p(v,l)$ determines the probability density of outgoing directions l given an incident direction v at a scattering event. We use the phase function developed by Henyey-Greenstein [HG41]. It is controlled by a single parameter g in the range $[-1,1]$, the *anisotropy parameter*.

$$p(v,l;g)=\frac{1}{4\pi}\frac{1-g^2}{(1+g^2+2g(v\cdot l))^{3/2}}$$

Negative values of g correspond to backward scattering and positive values to forward scattering. $g=0$ results in isotropic scattering.

Name	Type	Default	Range	Description
a	uniform color	1	[0,1]	Attenuation color.
d	uniform float	inf	[0,inf)	Attenuation distance. A value of 0 will make to surface opaque.
ρ_{ms}	uniform color	0	[0,1]	Subsurface color. Subsurface scattering is disabled if $\rho_{ms}=0$.
g	uniform float	0	[−1,1]	Subsurface anisotropy (<0: backward, =0: isotropic, >0: forward scattering).

Table 13: List of volume parameters.

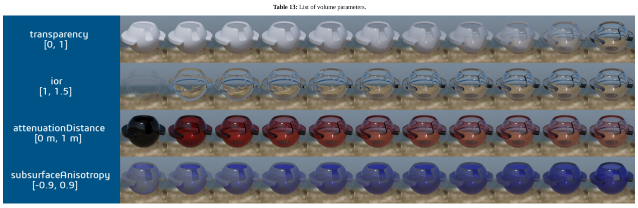

Figure 19: Renderings with various values for the volume parameters.

Dispersion

In optics, dispersion is the phenomenon in which the index of refraction depends on the wavelength of the light. It is usually quantified by the Abbe number v_d. Typical values for the Abbe number range from 20 (flint glass) to 60 (crown glass). The larger the Abbe number, the weaker the effect.

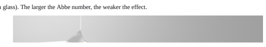

Figure 20: A dispersive prism breaks up white light into its spectral components (colors of the rainbow). This is the consequence of refraction angles varying with the wavelength of light rays.

The Abbe number ν_d is defined as

$$\nu_d = \frac{\eta_d - 1}{\eta_F - \eta_C},$$

where η_d, η_F and η_C are the indices of refraction at the wavelengths corresponding to the Fraunhofer d, F and C lines:

$$d = 587.56\ \mathrm{nm} \quad F = 486.13\ \mathrm{nm} \quad C = 656.27\ \mathrm{nm}$$

For rendering, we need a function that maps from any wavelength to an index of refraction, given the material's index of refraction η_d and Abbe number ν_d. One such mapping is Cauchy's equation. In its most general form, the equation is

$$\eta(\lambda) = a + \frac{b}{\lambda^2} + \frac{c}{\lambda^4} + \ldots,$$

where λ is the wavelength and a, b, c, ... are coefficients obtained by fitting measured data. Usually, its sufficient to use the first two coefficients. We can compute those from the two given values η_d and ν_d as follows:

$$b = \frac{\eta_d - 1}{\nu_d} \frac{1}{\frac{1}{F^2} - \frac{1}{C^2}} \quad a = \eta_d - \frac{b}{d^2}$$

Thus Cauchy's equation based on the Abbe number is (with symbolic constants replaced by their actual value):

$$\eta(\lambda) = \max\left(\eta_d + \frac{\eta_d - 1}{\nu_d}\left(\frac{523655}{\lambda^2} - 1.5168\right), 1\right)$$

η_d in this equation has to be set to the original value of the user-facing parameter for index of refraction η. The result $\eta(\lambda)$ then replaces the parameter's value. As a special case we disable dispersion if Abbe number $\nu_d = 0$.

Name	Type	Default	Range	Description
vd	uniform float	0	[0,inf)	Abbe number. Dispersion effect is disabled if vd=0.

Material Parameters

All information about parameter ordering and grouping, names, types, ranges, default values etc. required for a UI can be found in params-2025x.json. Application developers should follow these recommendations so that the UI is consistent across applications. The file format is documented in the appendix, Section 6.4. You can use the Javascript-based viewer application to inspect the JSON file.

Material Subtypes

The Enterprise PBR Shading Model offers a limited but still substantial amount of parameters. For easier usability, especially for novice users, we recommend to use the following subtypes, which expose only subsets of parameters:

Car Paint

Metal

Emissive

Textile

Leather

Wood

Glass

Plastic

For compatibility with the widely spread Unreal model [Kar13], additionally a general purpose material called "Basic" is recommended.

In addition to the parameters of the Enterprise PBR Shading Model (Section 2), the JSON file params-2025x.json documents parameters and default values for the subtypes. The file format is documented in the appendix, Section 6.4. You can use the Javascript-based viewer application to inspect the JSON file.

Design Rationale

The Enterprise PBR Shading Model is the basis for a large variety of use-cases, ranging from high-performance VR applications to high-quality offline GI renderings. However, it is not expected that it will cover all use-cases one can imagine, but it should be sufficient for day-to-day use. This requires some tradeoffs, which are based on the following guidelines and requirements.

Most important, the material should be *easy to use* for inexperienced users, but still powerful enough for trained artists in real-time rendering and global illumination. For this reason, the material has only a small amount parameters, following industry-standard workflows known from real-time engines (Unreal Engine 4, Unity Engine) and movie production (Disney BRDF). Using the *metallic/roughness workflow*, it is possible to create a wide variety of materials using only a subset of the complete parameter set, namely albedo, metallic, roughness and normal map. Another aspect is the fact that it allows users to parameterize materials in a way that they are *physically-based by default*, but can be tweaked when needed for more artistic freedom using parameters like specular tint and specular. Most parameters are linear and texturable, so that *layering via parameter blending* is possible. Notable exceptions are parameters affecting the volume, because there are no texture coordinates available inside an object.

Especially for GI renderers, there are certain physical and mathematical properties that materials or BSDFs must fullfill. A BSDF must obey the *Helmholtz reciprocity* (for bidirectional rendering methods), be *positive* and *energy conserving* (for fast convergence).

Performance in real-time rendering is a critical aspect. Most parameters do not influence each other or have a clearly defined state in which an effect is disabled, making it possible to specialize shader code for certain use-cases. Clearly, offline GI rendering also profits from this, but besides that it is also important to choose BSDFs that can be *sampled efficiently*, like the GGX microfacet model.

As many software packages offer a unified interface to different renderers, *consistency across renderers* and rendering-algorithms is another key requirement for the material. A material prepared in one renderer, for example in GI, should look the same in other renderers, for example in a rasterizer. As this is not always possible, because some effects like refraction or brute-force subsurface scattering are impossible in a real-time rasterizer, the *rasterizers will approximate* the *ground-truth GI solution*. The material is defined in a way that it encourages approximations wherever possible.

Not all real-world effects can be rendered with the Enterprise PBR Shading Model. Especially for high-quality renderings or light simulation, artists and engineers need more flexbility. Other material systems like Open Shading Language (OSL) from Imageworks or Material Definition Language (MDL) from NVIDIA allow users to build their own materials from basic building blocks structured as a node graph. The Appearance Exchange Format (AxF) from X-Rite stores results from a capturing process using their material scanners. Such material systems and models are projected onto the parameters of the Enterprise PBR Shading Model via *fitting* (or *distilling*).

Changelog

Version 2025x

Added thin-film interference (Section 1.2)

Added translucency color base material parameter (Section 1.1.4 and Section 1.1.3)

Version 2022x

Added material subtype for Leather and changed defaults of Textile, Basic (anisotropy is now hidden) and CarPaint (flake color).

Specified displacement, translucency and subsurface anisotropy parameters for material subtypes.

Removed sheen parameter.

Sheen roughness is now squared.

Improved sheen shadowing-masking term.

Added flip-flop effect to flakes.

Added dispersion.

Version 2021x

Added displacement mapping.

Added translucency.

Changed sheen effect. The new effect is based on the "Charlie" Sheen BRDF, layered on top of core material via albedo-scaling [CK17].

Added an alternative visibility term for the "Charlie" Sheen BRDF [NP13].

Emission is now below clearcoat in the layering stack.

Added Henyey-Greenstein phase function for volume scattering [HG41].

Added flakes layer.

Version 2019x

Initial version.

Appendix

Combining Metallic and Dielectric BSDFs

The core of the Enterprise PBR Material is a weighted sum of three BSDFs, the metallic BSDF, the opaque dielectric and the transparent dielectric BSDF. The similarity between the BSDFs allows us to simplify the material significantly. We start from the sum of the three BSDFs:

$$\mathrm{ThinCore}(\rho,\rho_l,s,\eta,\rho_s) = m\,\mathrm{Metallic}(\rho) + (1-m)(1-t)\,\mathrm{OpaqueDielectric}(\rho,\rho_l,s,\eta,\rho_s) + (1-m)t\,\mathrm{ThinTransparentDielectric}(\rho,s,\eta,\rho_s)$$

Note that we omitted some parameters here to make the equations shorter. These do not influence the weighting.

$$\mathrm{ThinCore}(\rho,\rho_l,s,\eta,\rho_s) = \mathrm{ThinCore}(v,l;r_u,r_v,\rho,\rho_l,s,\eta,\rho_s,l) \quad \mathrm{Metallic}(\rho) = \mathrm{Metallic}(v,l;r_u,r_v,\rho) \quad \mathrm{OpaqueDielectric}(\rho,\rho_l,s,\eta,\rho_s) = \mathrm{OpaqueDielectric}(v,l;r_u,r_v,\rho,\rho_l,s,\eta,\rho_s,l) \quad \mathrm{ThinTransparentDielectric}(\rho,s,\eta,\rho_s) = \mathrm{ThinTransparentDielectric}(v,l;r_u,r_v,\rho,s,\eta,\rho_s)$$

Substituting all terms leads to

$$\mathrm{ThinCore}(\rho,\rho_l,s,\eta,\rho_s) = mM_rF(\cos\theta_r;\rho,1) + mM_{r,ms}F_{ms}(\rho,1) + (1-m)(1-t)B(v,l;\rho,\rho_l,s,F_0,\rho_s) + (1-m)(1-t)sM_rF(\cos\theta_r;F_0\rho_s,1) + (1-m)(1-t)sM_{r,ms}F_{ms}(F_0\rho_s,1) + (1-m)tsM_rF(\cos\theta_r;F_0\rho_s,1) + (1-m)t\rho M_r'(1-sF(\cos\theta'_r;F_0\rho_s,1)) + (1-m)tsM_{r,ms}F_{ms}(F_0\rho_s,1)$$

and after rearranging we get

$$= M_r[mF(\cos\theta_r;\rho,1) + (1-m)(1-t)sF(\cos\theta_r;F_0\rho_s,1) + (1-m)tsF(\cos\theta_r;F_0\rho_s,1)] + M_{r,ms}[mF_{ms}(\rho,1) + (1-m)(1-t)sF_{ms}(F_0\rho_s,1) + (1-m)tsF_{ms}(F_0\rho_s,1)] + (1-m)t\rho M_r'(1-sF(\cos\theta'_r;F_0\rho_s,1)) + (1-m)(1-t)B(v,l;\rho,\rho_l,s,F_0,\rho_s)$$

With the Schlick approximation for the Fresnel term

$$F(\cos\theta,f_0,f_{90}) = f_0 + (f_{90} - f_0)(1-\cos\theta)^5$$

the following properties hold

$$aF(\cos\theta,f_0,f_{90}) = F(\cos\theta,af_0,af_{90}) \quad F(\cos\theta,a,b) + F(\cos\theta,c,d) = F(\cos\theta,a+c,b+d)$$

Approximately, that is also the case for the multi-scattering Fresnel F_{ms}, so we assume

$$aF_{ms}(f_0,f_{90}) \approx F_{ms}(af_0,af_{90}) \quad F_{ms}(a,b) + F_{ms}(c,d) \approx F_{ms}(a+c,b+d)$$

This allows us to simplify the combined BSDFs and we arrive at the final form:

$$\mathrm{ThinCore}(\rho,s,\eta,\rho_s) = M_rF(\cos\theta_r;(1-m)sF_0\rho_s + m\rho,(1-m)s+m) + M_{r,ms}F_{ms}((1-m)sF_0\rho_s + m\rho,(1-m)s+m) + (1-m)t\rho M_r'(1-sF(\cos\theta'_r;F_0\rho_s,1)) + (1-m)(1-t)B(v,l;\rho,\rho_l,s,F_0,\rho_s)$$

Energy Compensation

Multiple-Scattering GGX BRDFs

The GGX microfacet BRDF $M_r(v,l;r_u,r_v)$ only models a single scattering event, which is the first reflection that occurs at a facet on the microsurface. Multiple bounces between the facets are ignored, effectively leading to too much absorption in the model. [KC17] describes a method to approximate the missing contribution from multiple scattering. It requires directional and average albedo of the single-scattering model. We precompute those values and store them in a lookup table. In this section we describe how the precomputation step works.

In [KC17], the missing energy due to the missing multiple-scattering in the GGX microfacet model M_r is approximated by adding an additional compensation term $M_{r,ms}$. We will call the sum $F_{r,1}$:

$$F_{r,1}(v,l;r_u,r_v) = M_r(v,l;r_u,r_v) + M_{r,ms}(v,l;r_u,r_v) = M_r(v,l;r_u,r_v) + \frac{(1-E_m(v\cdot n;r_{uv}))(1-E_m(l\cdot n;r_{uv}))}{\pi(1-E_{m,avg}(r_{uv}))}$$

We use this sum in all core BSDFs of our model. For now, however, we have omitted the Fresnel term, because at first we are only interested in a surface that is 100% reflective. E_m and $E_{m,avg}$ denote the directional albedo and average albedo respectively. To keep it simple we replaced the anisotropic roughness r_u,r_v with an isotropic approximation $r_{uv} = \sqrt{r_u r_v}$. This means that in the end our multiple-scattering term slightly underestimates the real contribution, but this is not a huge issue in typical scenes.

First we compute the directional albedo E_m.

$$E_m(v;r_{uv}) = \int_{\Omega_l} M_r(v,l,h;r_{uv})(l\cdot n)\,dl \implies E_m(\theta_v,\phi_v;r_{uv}) = \int_0^{2\pi}\int_0^{\pi/2} M_r(\theta_l,\phi_l,\theta_v,\phi_v;r_{uv})\cos\theta_l\sin\theta_l\,d\theta_l\,d\phi_l$$

We are only interested in the difference between ϕ_l and ϕ_v, because rotating the BRDF does not change the result. Therefore, with $\phi = \phi_l - \phi_v$ and $\phi_v = 0$ ($\implies d\phi = d\phi_l$) we get

$$E_m(\theta_v;r_{uv}) = \int_0^{2\pi}\int_0^{\pi/2} M_r(\theta_l,\phi,\theta_v;r_{uv})\cos\theta_l\sin\theta_l\,d\theta_l\,d\phi$$

By reparameterization with $\mu_{\{l,v\}} = \cos\theta_{\{l,v\}}$ ($\implies d\mu_{\{l,v\}} = -\sin\theta_{\{l,v\}}d\theta_{\{l,v\}}$), we arrive at the simpler form

$$E_m(\mu_v;r_{uv}) = \int_0^{2\pi}\int_0^1 M_r(\mu_l,\phi,\mu_v;r_{uv})\mu_l\,d\mu_l\,d\phi$$

Evaluating this integral for different values of μ_v and isotropic roughness r results in the image shown in Figure 21.

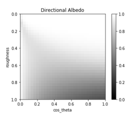

Figure 21: Directional albedo of single-scattering microfacet GGX BRDF.

Integrating over the hemisphere once more gives us the average albedo (Figure 22):

$$E_{m,avg}(r_{uv}) = 2\int_0^1 E_m(\mu_v;r_{uv})\mu_v\,d\mu_v$$

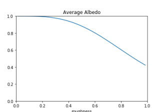

Figure 22: Average albedo of single-scattering microfacet GGX BRDF.

We are now able to store these results in a lookup table (e.g., with a size of 32×32 pixels) and fetch them during the evaluation of the BSDF to compute the energy compensation term $M_{r,ms}$.

So far, we ignored the Fresnel term. But similar to scaling the single-scattering microfacet term with a single-scattering Fresnel term, we need to scale the multiple-scattering compensation term with the average Fresnel. This is the average Schlick Fresnel:

$$F_{m,avg}(f_0,f_{90}) = \frac{1}{21}f_{90} + \frac{20}{21}f_0$$

Results from the white furnace test are shown in Figure 23, Figure ? and Figure 24. While there is still a bit of energy loss when using the fitted function, the sphere becomes almost invisible when using the more accurate data from the 32×32 table.

Figure 23: White furnace test without energy compensation for $r = \{0.1, 0.6, 0.9\}$.

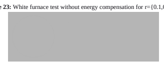

Figure 24: White furnace test with energy compensation.

We provide the lookup tables for download in our Github repository:

res/GGX_E.exr: $E_m(\cos\theta;r_{uv})$, x: $\cos\theta$, y: r_{uv}.

res/GGX_E_avg.exr: $E_{m,avg}(r_{uv})$, x: $\cos\theta$.

Diffuse BRDF

The incident illumination that hits an object will either be reflected at the surface or enter it and be scattered equally in all directions. The latter results in a diffuse look that is modeled with the diffuse component B. The ratio between specular and diffuse reflection is determined by the Fresnel term.

For an energy-preserving and energy-conserving combination of the components we apply the same ideas as for the GGX energy compensation described in Section 6.2.1. We measure how much energy is left after evaluating the Fresnel-weighted multiple-scattering GGX BRDF and scale the diffuse BRDF accordingly.

First we compute the directional albedo of the multiple-scattering GGX including the Fresnel term:

$E(\mu_r r_{ms}, f_0) = \int_0^{2\pi}\int_0^1 M(\mu_l, \phi_l, \mu_r r_{ms}, \phi_r) \mu_l \mu_r \mathrm{Fms}(f_0, 1) \, d\mu_l \, d\phi_l$

This results in a 3D lookup table, see Figure 25. As the function is smooth, a resolution of 16 pixels in each dimension is sufficient.

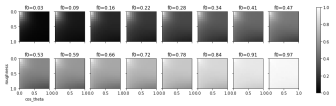

Figure 25: Directional albedo of Fresnel-weighted multi-scattering microfacet GGX BRDF for different values of f0.

As before, we integrate over the hemisphere once more to get average albedo

$E_{avg}(\mu_{r,f0}) = 2 \int_0^1 E(\mu_r r_{ms}, f_0) \mu_r \mu_r \, d\mu_r$

and store the result of the integration in a table with 16×16 pixels (Figure 26).

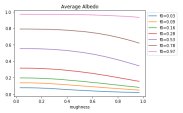

Figure 26: Average albedo of Fresnel-weighted multi-scattering microfacet GGX BRDF for different values of f0.

Figure 27: White furnace test for combination of diffuse and specular component.

We provide the lookup tables for download in our Github repository:

tex/Bstar_E.exr: $E(\cos\theta_r r_{ms}, f_0)$, x: cosθ, y: rms, filename: f0

tex/Bstar_Eavg.exr: $E_{avg}(r_{ms}, f_0)$, x: rms, y: f0

Sheen BRDF

We use the albedo-scaling technique described in [KS01] and [CK17] to layer the sheen BRDF on top of the core material. As the technique supports arbitrary BSDFs as base layer, it does not depend on the parameters of the core material.

As before, we precompute the directional albedo of the sheen BRDF (Equation 115). This results in a 16×16 2D lookup table (Figure 28).

$E(\mu_r r_s, f_0) = \int_0^{2\pi}\int_0^1 M(\mu_l, \phi_l, \mu_r r_s, \phi_l) \mu_l \mu_r \, d\mu_l \, d\phi_l$

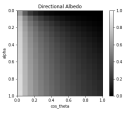

Figure 28: Directional albedo of the sheen BRDF. The y-axis is the squared sheen roughness $\alpha_s = r_s^2$.

We provide the lookup table for download in our Github repository:

tex/Sheen_E.exr: $E(\cos\theta_r, \alpha_s)$, x: cosθ, y: $\alpha_s = r_s^2$

Alternative Sheen BRDF

We precompute the directional albedo of the alternative sheen BRDF using Equation 115 (Figure 29).

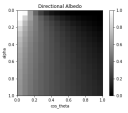

Figure 29: Directional albedo of the alternative sheen BRDF. The y-axis is the squared sheen roughness $\alpha_s = r_s^2$.

Figure 30: White furnace test of alternative sheen BRDF with roughness from 0 to 1, left to right. Red pixels indicate energy gain.

The lookup table is also provided for download in our Github repository:

tex/Sheen_ssh_E.exr: $E_{ssh}(\cos\theta_r, \alpha_s)$, x: cosθ, y: $\alpha_s = r_s^2$

Flake Consistency

You might have noticed that we do not define the hashing function or the PRNG used for generating flakes. On the one hand this limits consistency of flakes between different renderers because different random number generators will produce different flake position and orientation values for grid cells. On the other hand this gives you an opportunity to try out different hashing functions and PRNGs depending on your needs. From our experience, using different hashing functions and PRNGs can significantly affect shading performance.

Another thing you can experiment with is the the number of flakes per grid cell. For cheaper hashing functions it's beneficial to have low number of flakes per grid cell like 1 or 2.

Parameter JSON file format

We provide a JSON file containing a machine-readable list of all user-facing parameters of the Enterprise PBR Shading Model and its subtypes: params-2025x.json.

In addition, there is a Javascript-based viewer application to inspect the file.

Format

The JSON file contains multiple materials under the root node called `materials`. A material has a unique identifier and a set of properties describing its display name, preview image, and parameters.

```
{
  "materials": {
    "generic": {
      "name": "Enterprise PBR Shading Model",
      "preview": "previewImage.png",
      "parameters": {
        ...
      }
    }
  }
}
```

Parameters have a unique identifier, a display name, a type, type-specific defaults and ranges, and properties to describe optimal placement in the UI.

```
{
  "materials": {
    "generic": {
      ...
      "parameters": {
        "albedo": {
          "name": "Albedo",
          "type": "color",
          "max": [1, 1, 1],
          "default": [1, 1, 1],
          "uiOrder": 0,
          "group": "Base"
        },
        "metallic": {
          "name": "Metallic",
          "type": "float",
          "min": 0,
          "max": 1,
          "default": 0,
          "uiOrder": 1,
          "group": "Base"
        },
        "ior": {
          "name": "Index of Refraction",
          "type": float,
          "uniform": true,
          "uiOrder": 20,
          "group": "Volume",
          ...
        }
        ...
      }
    }
  }
}
```

The properties `uiOrder` and `group` specify the order and group in which the parameters should be displayed in the user interface. The empty group `''` must be displayed before any other named group. In addition, the Boolean property `hidden` (not shown in the example) controls the visibility of a parameter in the UI. This is useful in conjunction with inheritance and will be explained later in Section 6.4.5.

Types

Table 14 lists the supported parameter types.

Type	Default	Min	Max	Description
`float`	0	-inf	+inf	Floating-point number.
`color`	[0, 0, 0]	[0, 0, 0]	[inf, inf, inf]	Color in linear sRGB, given as an array of 3 numbers.
`normal`	[0, 0, 1]	n/a	n/a	Unit-length 3D normal vector, given as an array of 3 numbers.
`vector`	[0, 0, 0]	[-inf, -inf, -inf]	[+inf, +inf, +inf]	3D Vector, given as an array of 3 numbers.
`boolean`	false	n/a	n/a	Boolean value, `true` or `false`.
`enum`	n/a	n/a	n/a	Enumeration, valid values are given in an array called `enum`. Example: `{ "type" : "enum", "enum": ["apples", "oranges"], "default": "apples" }`.

Table 14: List of parameter types supported in the JSON file.

The properties `min`, `max`, and `default` are optional. If not set, the values are derived from the type.

Since JSON does not define the value `inf`, the following object can be used instead of a floating-point number to indicate +inf: `{ "infinity": true }`.

A parameter type can be marked as uniform with the `uniform` property. If unset the parameter is not uniform.

Inheritance

Materials can derive from other materials via the `inherits` property. Inheriting from a material means to recursively merge the properties from the derived material into the base material. Derived properties overwrite base properties. Example:

```
{
  "materials": {
    "dielectric": {
      "name": "Dielectric",
      "parameters": {
        "ior": {
          "type": "float",
          "default": 1.5
        },
        "transparency": {
          "type": "float",
          "default": 0
        }
      },
    },
```

```
  "glass": {
    "inherits": "dielectric",
    "name": "Glass",
    "parameters": {
      "ior": {
        "default": 1.52
      },
      "transparency": {
        "default": 1,
        "hidden": true
      }
    }
  }
}
```

We derive a new material `glass` from the base material `dielectric`. The `glass` material changes the default `ior` from 1.5 to 1.52 and the default `transparency` from 0 to 1. In addition, we hide the `transparency` parameter from the user by specifying `{ "hidden": true }`. Note that by default all parameters of the base material are visible in the derived material. Use the `hidden` property to make them invisible in derived materials.

Bibliography

[**AK16**] Asen Atanasov and Vladimir Koylazov. A Practical Stochastic Algorithm for Rendering Mirror-Like Flakes. 2016.

[**Bur12**] Brent Burley. Physically-Based Shading at Disney, SIGGRAPH 2012.

[**Bur15**] Brent Burley. Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering., SIGGRAPH 2015.

[**CK17**] Alejandro Conty Estevez, Christophe Kulla. Production Friendly Microfacet Sheen BRDF, SIGGRAPH 2017.

[**Cook82**] Robert L. Cook, Kenneth E. Torrance. 1982. A Reflectance Model for Computer Graphics.

[**Fde19**] Carmelo J. Fdez-Agüera. A Multiple-Scattering Microfacet Model for Real-Time Image-based Lighting., Journal of Computer Graphics Techniques Vol. 8, No. 1, 2019.

[**Hei14**] Eric Heitz. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs., Journal of Computer Graphics Techniques Vol. 3, No. 2, 2014.

[**HG41**] Louis G. Henyey and Jesse L. Greenstein. Diffuse radiation in the galaxy. Astrophysical Journal 93, 70–83, 1941.

[**Jak14**] Wenzel Jakob et al. Discrete Stochastic Microfacet Models, ACM Transactions on Graphics (ACM SIGGRAPH 2014).

[**Kar13**] Brian Karis. Real Shading in Unreal Engine 4., SIGGRAPH 2013.

[**KC17**] Christopher Kulla, Alejandro Conty. Revisiting Physically Based Shading at Imageworks. 2017.

[**KS01**] Csaba Kelemen, Laszlo Szirmay-Kalos. A Microfacet Based Coupled Specular-Matte BRDF Model with Importance Sampling, Eurographics 2001.

[**NPI3**] David Neubelt and Matt Pettineo. Crafting a Next-Gen Material Pipeline for The Order: 1886. 2013.

[**WML T07**] Bruce Walter, Stephen R. Marschner, Hongsong Li, Kenneth E. Torrance. Microfacet models for refraction through rough surfaces., EGSR 2007.

License