Physically Based Shading at DreamWorks Animation

Feng Xie and Jon Lanz

1 BRIEF HISTORICAL OVERVIEW

Before we talk about shading, which models the physical interaction of light and material surfaces, let us give a brief overview of rendering and shading history at PDI and DreamWorks Animation.

Figure 1: *Antz*: Pacific Data Images' first animated film.

Antz was the first film made by PDI/DreamWorks. The production of this film lead to the development of: an early animation production pipeline with a new scene graph format developed by Lawence Kesteloot; a scanline-based polygon renderer `D_RENDER` that supported deferred shading using deep frame buffer developed by Dan Wexler; and an interactive lighting system called `Light`, developed by Drew Olbrich. `Light` was designed to give interactive redraw and reshade feedback to users on entry-level Unix workstations, such as SGI O2 systems[1]. It had a spreadsheet UI for asset definitions; a 3D viewer for interactive geometry, camera and light placement, and view examination; and a render viewer that was tightly integrated with `D_RENDER` to allow users to seamlessly update shadow maps, rasterize the deep framebuffer and reshade subviewports at will when lights or materials were changed. `D_RENDER` supported a C++ shading API for lights, materials and procedural nodes. For shading *Antz* mostly used a simple surface shader with lambertian diffuse and Phong specular model with a few heuristic Fresnel components.

Shrek and *Shrek2* brought about a lot of progress in shading. Jonathan Gibbs developed an uber surface shader with Ward specular model and a Kajiya Kay based hair shader, while Rick Glumac developed an eye shader that used pseudo raytracing of procedurally generated geometry. A visit and presentation of diffusion-based sub-surface scattering by Henrik Wann Jensen from Stanford lead to a collaboration with PDI FX artist Juan Buhler. Together, they developed an octree-based solution for rapid evaluation of diffusion-based subsurface scattering, which made the approach viable for feature animation [JB02]. Fiona, in *Shrek*, was the first PDI/DreamWorks character to have translucency in her skin.

[1] 200 MHz MIPS CPUs with a 1 MB L2 cache.

Figure 2: *Shrek* added base, hair and skin shaders. Fiona was the first animated CG character to use subsurface scattering on her skin.

Figure 3: *Shrek 2* was the first PDI/DreamWorks film to use single *bounce* indirect lighting, allieviating the need for artists to manually place many fill lights to approximate indirect lighting effects.

The shading and lighting paradigm in `D_RENDER` was similar to Pixar's Renderman, in that materials performed light integration by looping through all applicable lights. This paradigm was sufficient for supporting traditional CG point and directional lights, which were the primary lighting models we used in those early films. Indirect lighting was emulated by artists manually placing a large number of fill lights, which was obviously quite tedious and often incorrect. Around the early 2000s, Eric Tabellion and Arnold Lamorlette [TL04] developed an irradiance- cache-based global illumination appoximation system for `D_RENDER` and PDI's lighting system. Their solution was integrated into the rendering system as a "bounce" light shader and it was used by many films at PDI and DreamWorks, from *Shrek 2* onward.

Around 2007 to 2008, the term *physically based shading* started to emerge in CG film shading, primarily in visual effects productions where there was a high degree of motivation for CG rendered assets to match physical plates/footage in their lighting and reflectance appearance. Within DreamWorks, the motivation for physically based shading did not come from matching physical realism, but from two interrelated motivations: the first being the artistic desire to create a richer and more nuanced look in our characters and environments, the second being that variation in material and lighting setups resulted in inconsistencies between surfacing (look development) and production lighting. We looked to physically based shading models that were consistent or plausible under various lighting conditions, as a way to increase the richness of our looks without incurring an

explosion in lighting and material setup complexity.

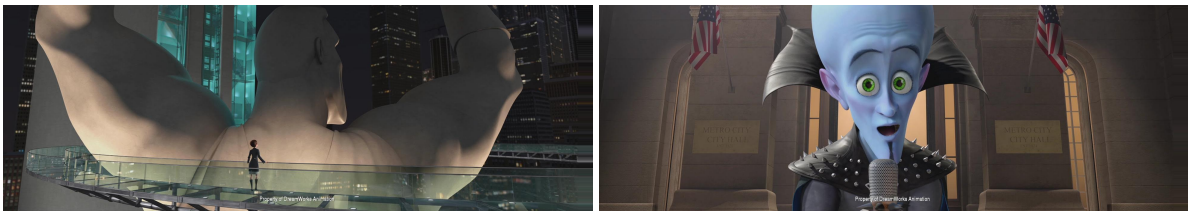

Figure 4: *Megamind*: For this superhero movie, set in a modern cityscape with lots of metal and glass, we revamped our base material to use physically based Fresnel with accurate metal support.

The films that defined this "look and shading"-based transition from both the artistic and technical perspectives are *Megamind*, *Rise of the Guardians* and *The Croods*.

Megamind (2007–2009) was an animated superhero film that was set in a modern/futuristic city landscape. The look of the film called for more physical realism than any of our previous productions. The environment had a lot of glass and metals, with volumetric lighting effects. With support from VFX Supervisor Philippe Denis and Look Dev Supervisor David Doepp, we explored the use of physically based Fresnel models, with accurate metal support. This approach did not have wide support across other shows because the controls for physically based Fresnel were quite complex and did not deliver a compelling enough benefit on predominatly dielectric opaque assets in most of our films at the time. Moreover, at the time, the lighting models and light material interactions we were able to model in `D_RENDER` for glossy reflections were still in early development. With so many approximations being made to solve the rendering equation, making the BRDF Fresnel term more accurate alone would not have delivered a comprehensive improvement on the look of assets or characters. However, the process taught us much about the importance of using more physically based principles in modeling material appearance, as well as the importance of working with artists to understand the level of controls they need.

With *Rise of the Guardians* (2009–2011), the VFX supervisor David Prescott and Look Dev Supervisor Andy Harbeck were very motivated to work with R&D on the development of character shaders that were grounded in physical principles, in order to achieve the "magical realism" look of the film. Inspired by Donner and Jensen's work on multi-layered translucent materials [DJ05], we developed a three-layer skin shading model that allowed artists to have separate controls for epidermis and dermis. We also developed a physically based eye shading model, which allowed our animated characters to have eyes with more depth and realism. For hair, we implemented the Marschner hair shading model [Mar+03] with a more intuitive transmission and glint term.

However, one significant factor to remember is that, for all of these shaders, we focused on the accurate modeling of the interaction of *direct* lighting with the surface material. There was no consideration yet to accurately model the interaction of these materials with *indirect* lighting.

Concurrent to the progress on character shaders by the shading team, Eric Tabellion lead the rendering development of Point-Based Global Illumination (PBGI) solutions that he presented in the *Global Illumination Across Industries* SIGGRAPH 2010 Course [Kři+10]. This work leveraged an importance-sampled Ward BRDF and directional importance mapping techniques for PBGI octree traversal to enable efficient and accurate modeling of hard surface reflections under IBL-based area and environment light sources.

For *The Croods* (2009–2013), Head of Lighting Mark Edwards lead his lighting supervisors to develop and deploy

Figure 5: *Rise of Guardians*: This fantasy film marked a milestone in shading at DreamWorks, by establishing a set of physically based character shaders for skin, hair and eyes.

an IBL-based lighting rig using point-based occlusion and point-based indirect lighting computation with glossy reflection. The simplified lighting setup worked consistently on both the characters' skin and environment elements such as leaves, which was very important to the look of the film set in a natural environment. However, we could not use the IBL-based lighting rigs on hair or fur assets because there was no known technique for importance sampling of hair at the time.

In the summer of 2011, Jiawei Ou and Feng Xie started research on importance sampling of the Marschner hair model. Eric and the render team had developed a clean library interface for building BSDF models required for physically based materials. In 2012, leveraging this API, we integrated and deployed importance-sampled hair glossy reflection under environment lighting [Ou+12]. It was first used for Astrid in *How to Train Your Dragon 2*.

Figure 6: Left: The natural outdoor lighting style of *The Croods* lends itself well to physically based lighting using IBL. Right: Astrid in *How to Train Your Dragon 2* was the first DreamWorks animation character to use IBL based lighting on glossy hair.

After *The Croods*, IBL-based light rigs were used across all of our shows. However, in terms of reflection models used for indirect lighting computation, we still had a lot of challenges: energy conservation, consistency, etc. One part of these challenges was PBGI as a solution to lighting transport, which was inherently still an approximation. The other part was the architecture constraint of the built-in indirect diffuse lighting integration in our PBGI system, which was based on the core assumption that each shading microfacet had a hemispherical diffuse shading model. The limitation of our PBGI rendering implementation in its ability to emulate indirect lighting transport accurately, and the software complexity of a feature-rich scanline renderer that was now 15 years old, made it more and more difficult to create and extend physically plausible material shaders and lighting models. Hair scattering in particular became quite very expensive with PBGI.

Outside of DreamWorks, a revolution was happening in the world of production rendering [Kel+15]. Path tracing started with Arnold, then spread to Disney's Hyperion, Renderman's RIS, and Weta's Manuka. Accordingly, in 2014, Eric Tabellion and Brian Green lead a small team of developers to build a completely new path tracing renderer, Moonray, at DreamWorks. This renderer uses unidirectional path tracing with next event prediction, as described in [PJH16]. It fully supports physically based light transport, with the exception of caustic light paths. The Moonray renderer has a built-in set of importance-sampled lighting models; a library of (and a C++ API for extending) importance-sampled BSDFs, including all glossy reflection and refraction models described in [Wal+07], as well as our importance-sampled Marschner hair model [Ou+12]. Like `D_RENDER`, Moonray has a C++ shading API for materials and procedural shaders; unlike `D_RENDER`, Moonray's material shaders return a BSDF closure to the renderer instead of performing the lighting integration within the material shader. The actual lighting and material interaction and integration is computed by the path integrator of the renderer.

2 DWA PHYSICALLY BASED SURFACE MATERIAL

Starting in the fall of 2015, we had the opportunity to design and develop a new set of physically based production shaders for the Moonray renderer. We will introduce our design philosophy for the material shaders and briefly discuss their features, as well as our design and implementation of our material layering system.

2.1 Design Motivation

In designing our material shaders the primary goals included building a small set of materials that are as simple as possible to use individually, and supporting the ability to layer these materials into networks to enable the creation of more complex and varied looks. In this section we examine some of the benefits and drawbacks of several common approaches.

So-called "uber materials" are traditionally full-featured, multi-model shaders that are capable of representing several different common material types (metal, glass, plastic, etc.). The artist defines the type of material being represented by specifying a particular combination of parameters. With texture mapping the artist can vary the surface characteristics (color, roughness, shading normal, etc.), but importantly can also vary the type of material being represented over the surface. This native capability to spatially vary the type of material represented without requiring a material layering system makes the uber material an attractive option.

There are several design and workflow challenges associated with uber materials. An uber material can in some ways be thought of as a grab-bag of shading features, and the burden is on the artist to choose which features

are appropriate for the targeted material type. The number of attributes required to offer this flexibility can be cumbersome if not overwhelming to the artist. It may be difficult for the developer to choose reasonable default values for each of these attributes or to limit their values to an appropriate range, as it is not known at development time what type of material the artist will attempt to represent.

Another class of material shaders stands in contrast with the uber material, with each shader presenting a single shading model that is designed to represent a single real-world material type. With these specialized materials, while the surface characteristics can also be spatially varied using texture maps, the material type itself cannot. To overcome this limitation, these shaders often rely on an accompanying material layering system in order to represent compound materials or complex surfaces.

This class of limited-purpose, specialized materials offers an advantage in that the shaders can expose a streamlined and intuitive set of attributes, and because the type of material being represented is known in advance, reasonable defaults can be chosen. Such materials encourage proper usage due to their simplicity, and the valid range of values for any physical parameter is better understood. For creating simple looks, these single-model materials are in many ways preferable to the uber material.

One common functional similarity between the uber material and a material layering system is that both allow the artist to vary or blend the type of material being represented over different areas of the same surface. While the uber material allows for this implicitly, it may require potentially complex texture map networks, which can be difficult to paint and maintain. On the other hand, a material layering system allows for a simplified and non-destructive workflow, and each material being layered can easily be viewed and adjusted independently.

An interesting quality of the uber material is that once it has been designed and developed, its structure and behavior are then well known, with a fixed set of potential closures that will be produced. These closures will be defined in such a way that the succession of light interactions between lobes is fixed, and the Fresnel behavior is well understood. This advanced knowledge of the foundation makes ensuring energy conservation a manageable task for the developer, regardless of how the particular instance of the material is parameterized by the artist. Additionally, complex and compound looks can be authored without significant rendering performance impact as it is all handled within a single material. A well-written uber material shader can implicitly and efficiently blend between the supported shading models while behaving plausibly and maintaining energy conservation.

While designing and implementing a robust and efficient uber material is non-trivial, the challenge of building a robust layering system that can efficiently handle the layering of arbitrary material types with differing closure configurations and Fresnel behaviors while maintaining energy conservation is even more daunting. In a Monte Carlo ray tracing framework, BSDF sampling efficiency is very important, and implementing a naive layering system could result in long render times for production. If each material being layered produces its own set of weighted closures, the total number of closures can quickly add up and cause performance issues during integration.

2.2 Design and Implementation of Material Layering

Our solution builds on the parameter blending approach presented in Disney's "principled" layering system [Bur12] and attempts to leverage the strengths of each of the aforementioned approaches. At the core is a library with a base class uber material, which is capable of modeling and mixing several common material types. We derive from this base class several specialized materials, each of which exposes only the relevant features and

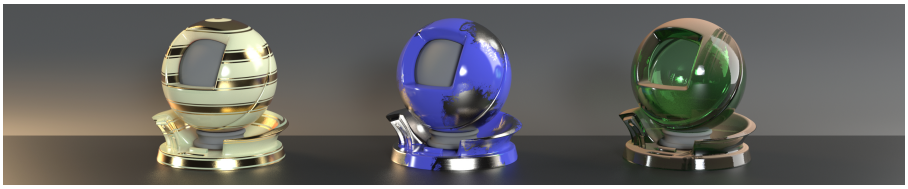

Figure 7: Three different looks created using our layering system: wax over gold, paint over steel and dust covered glass.

attributes for representing that particular material type. Layering is facilitated by a simple `Layer` material, which provides "A over B" functionality with a single alpha mask input. Our initial implementation includes the following specialized materials for several common material types: `Metal`, `Refractive`, and `Solid Dielectric`. These three materials can be used independently or layered together into arbitrarily deep networks. Layering is achieved by splitting the shading process into two distinct stages: parameter resolution and closure creation.

During the first stage, starting at the top layer material, each material in the network potentially evaluates all of its attributes and texture maps in top-down depth-first order. The `Layer` material's mask is evaluated first, and each of the two sub-materials are then potentially evaluated as needed. The resulting parameters are then blended together using linear interpolation into one set of parameters. This process continues at each node in the hierarchy until one final set of blended parameters has been resolved for the underlying base class uber material. This lazy evaluation of the material and map network means that only attributes and texture maps that are needed are evaluated.

In the second stage, one set of closures is created and configured using the final set of parameters. In this way, the entire material network is effectively collapsed at shading time into a single instance of the underlying uber material.

2.3 Shading Features of DWA Physical Materials

Figure 8: DWA Physical Material examples: metal (copper), opaque dielectric (plastic), clearcoated opaque dielectric (wood), refractive dielectric.

At DreamWorks Animation, we create movies with a wide variety of looks, ranging from fairly realistic to strongly stylized. In any animated feature, the style is heavily driven by character and environment design, color choices, animation, textural complexity, and creative lighting, and not strictly by material response, where

consistent behavior is perhaps most important. In designing the features of our new material shaders, we have generally strived for simplicity and out-of-the-box physical behavior over full artistic freedom, and have been very selective when it comes to exposing any feature or control that might cause unpredictable behavior during lighting. Our production teams fully support this strategy, as we attempt to strike a balance that allows us to reap the full benefits of physically based shading, while not being forced to compromise artistic control by its constraints. We anticipate that our materials will continue to evolve to fit the needs of each new production, and that we will likely discover cases where bending the rules is necessary to achieve stylistic goals.

Our three layerable material shaders each present a streamlined set of features that is appropriate for the associated material type. Each is designed to require minimal handwork by the artist to achieve a plausible look:

- `Metal` is the simplest of our specialized materials. The anisotropic Cook-Torrance microfacet model with a Beckmann distribution [Wal+07], is paired with conductor Fresnel behavior, which is parameterized using the artist-friendly metallic reflectivity and edge tint colors presented in [Gul14].
- `Refractive` also uses an anisotropic Cook-Torrance microfacet model with a Beckmann distribution for the primary specular lobe. The dielectric Fresnel is parameterized by a non-mappable `refractive index` control that has a default value of 1.5. Mirror reflection, glossy reflection and transmission are supported with a single `roughness` attribute that controls both effects. We also provide a separate, optional `transmission refractive index` control to decouple the bending of light from the reflectivity, which allows for non-physical artistic control for certain objects, such as eyeglasses. Additionally, a `transmission color` allows for tinting the transmitted light, while support for true volumetric absorption is provided via a separate volume shader.
- `SolidDielectric` shares the same specular reflection model and attributes as the `Refractive` material. For diffuse reflection, we provide `albedo`, `scattering radius` and `scattering color` attributes. When `scattering radius` is set to 0, a Lambertian BRDF is used. When `scattering radius` is perceptible, a BSSRDF is used, with support for the dipole model [Jen+01] and the empirical diffusion profile described in [Chr15]. Additionally, a diffuse transmission feature is available, with a single color attribute (fittingly named `diffuse transmission`) to control it.

Several additional features are supported across all three models. First is an optional zero thickness clearcoat layer, with physical refraction and dielectric Fresnel behavior. Attributes are exposed to specify the refractive index, thickness, and color attenuation due to absorption within the clearcoat layer. Next, an artist-friendly iridescence feature is controlled by an additional set of attributes. Finally, an `emission` attribute allows the artist to specify additional energy that is to be emitted by the material.

Each material also exposes a special `specular control` attribute, in $[0, 1]$, which defaults to 1. This attribute can be used to "kill" specular reflection in areas where certain otherwise unrepresented micro-scale geometric surface details (such as a deep pits or gaps) should result in no visible specular reflection. It is intended that this attribute be mapped with binary 0|1 values in order to maintain a plausible look.

With our system, our artists enjoy a workflow that involves the layering of multiple instances of our discrete, specialized and streamlined materials to form branchable, compound material networks. Each material in the network hierarchy is valid as a stand-alone material, with full support for spatially varying surface properties. At shading time, each network is reduced to an efficient-to-render instance of our underlying advanced uber material. Future work includes extending our layering system and uber material to include support for our fabric, velvet and skin shading models.

Figure 9: Characters and final logo rendered with glitter shader.

3 DWA GLITTER

The production designer for *Trolls* had a unique vision for the world in which the happy and music-loving trolls lived: the troll village is lush, fuzzy and translucent. Into this soft and translucent world, our designer wanted to add even more magic and happiness, and so that is where we needed glitter — lots of glitter!

In [Jak+14] and [AK16], the authors described algorithms to generate and render stochastic flakes with density N, in the order of 10^6 to 10^8 flakes per unit area; these solutions couldn't be directly applied to *Trolls*' glitter because the density of our glitter flakes was orders of magnitude smaller, ranging from 10^2 to 10^3. At this density scale, the glitter flakes have visible structure and our art direction wanted control over the color and size of the individual flakes on the main characters. To meet the artistic needs for glitter on *Trolls*, we developed a glitter flake material that gave our artists the desired controls, while adhering to the principles of a physically plausible material and also remaining efficient to render.

3.1 Algorithm Overview

A fundamental requirement to any glitter flake appearance model is the distribution of the flakes. The stochastic distribution algorithms described in [Jak+14] and [AK16] create a uniform distribution of high density flakes in texture space. The dependency on texture space can lead to stretching artifacts on deformed characters. Additionally, those algorithms assume there is no visible structure to the glitter: each stochastically distributed flake is either entirely inside or outside of the pixel footprint, which is a reasonable assumption for flakes with sub-pixel footprint.

For *Trolls*, we wanted glitter on characters undergoing extreme animation. We also wanted flakes that were roughly the size of a pixel, where the area coverage of an individual flake within a pixel was important to capture the flake size and appearance. For these reasons, we chose a well-known 3D noise function with easy to control density parameters as the foundation of our glitter flake distribution.

Our `Noise Worley` implementation populates a 3D grid with a Poisson distribution of points, without the need for heavy precomputation or storage. The original algorithm in [Wor96] provides an efficient method to query the nearest four feature points for any given location in 3D space. Given the closest four feature spheres — each corresponding to a glitter flake — our first priority is to determine the coverage of each flake within the pixel or shading footprint.

3.2 Coverage of each Glitter Flake Within a Shading Disk

Figure 10 shows how we compute the coverage of a glitter flake within a shading disk, where the glitter flake is represented as a disk cut from a feature sphere by the shading plane.

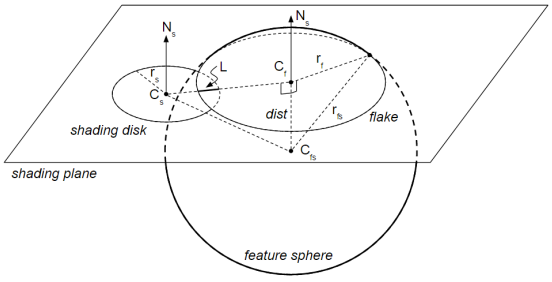

Figure 10: Computing sphere to disk intersection.

Shading point query setup: we transform the shading polygon to a disk on a noise-space plane, with center C_s, normal N_s and radius r_s. For each feature sphere with center C_{fs} and radius r_{fs}, compute the desired coverage as follows:

1. Compute the flake disk (radius r_f, center C_f) from the feature sphere and the shading plane:

$$r_f = \sqrt{{r_{fs}}^2 - dist^2}, \tag{1}$$

$$C_f = C_{fs} + dist\, N_s, \tag{2}$$

   where $dist = (C_s - C_{fs}) \cdot N_s$.

2. Determine the coverage between the shading disk and the flake, based on the overlap length L:

$$\text{coverage} = \frac{L}{2\, r_s}, \tag{3}$$

   where $L = r_s + r_f - \|C_f - C_s\|$.

Note that $|dist| < r_{fs}$ is required for the feature sphere to intersect the shading plane and create a valid glitter disk, and L needs to be clamped to a maximum of $2 \min(r_s, r_f)$.

3.3 Glitter Flake BRDF

After resolving the coverage of the flakes that overlap each shading polygon, we are ready to create our glitter flake BRDF. Each visible flake is instantiated as a GGX microfacet lobe [Wal+07], with a Schlick Fresnel term [Sch94], using the computed flake color as incident specular color, and flake normal as the microfacet normal. The glitter flake BRDF can have up to four flakes, each one a GGX lobe, with the weight (scale) set to the flake coverage.

Energy conservation and reciprocity of the glitter flake BRDF is ensured by the fact it is a linear combination of GGX lobes, which are both energy conserving and reciprocal. To efficiently sample the glitter flake BRDF, we first choose a flake with probability proportional to its coverage, then sample the selected flake using the importance-sampling strategy for the GGX microfacet lobe [Wal+07].

As a result, we have created a physically plausible material model for glitter flakes that is both energy conserving and efficient to sample and evaluate. We have integrated this BRDF model into our scanline-based production renderer, with extensions for supporting physically based lighting and shading. More recently we have also integrated the glitter flake BRDF into our new path tracing renderer.

3.4 Glitter Flake Controls

Density and size control: the size of each Worley sphere is proportional to the density, as the default point cloud packs the space without overlap. The default size of each feature sphere in noise space is 1; this size scale allows artists to adjust the flake size once the density control is set to the desired level.

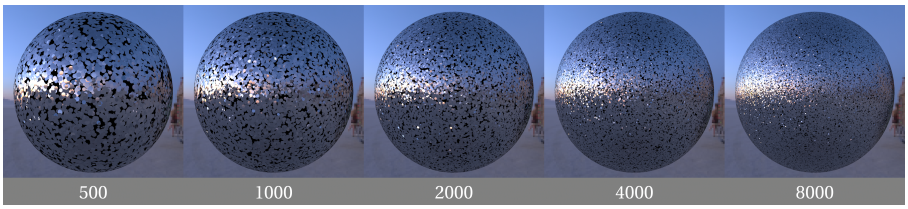

Figure 11: Glitter material with increasing density from 500 to 8000, roughness = 0.3.

Flake color: The color of each flake is determined either through a user-specified ramp or a randomized color. Both can be tinted by a user-specified color.

Flake roughness: The normal of each visible flake is determined by sampling the GGX distribution [Wal+07] centered around the shading normal, with a user-specified flake roughness.

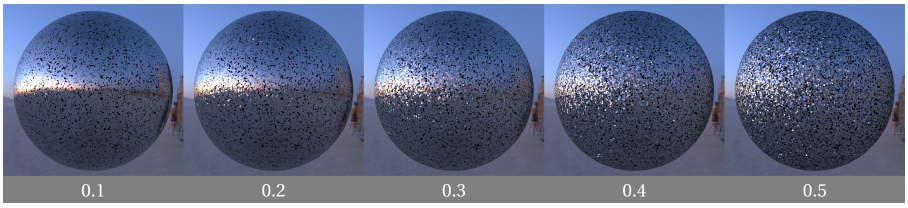

Figure 12: Glitter material with increasing roughness from 0.1 to 0.5, density = 4000.

All of the controls (density, size, color and roughness) can be made spatially varying using input textures, if desired.

3.5 Consistent Glitter Flakes for Deformed Characters

The glitter flakes looked great on character spins, but as the characters danced, the glitter flakes started to swim on their skin. We solved this problem by computing the flake position in reference space, while the flake size is still computed using the deformed shading micropolygon size so that the flakes don't look stretched as the underlying skin deforms.

3.6 LOD Generalization for Glitter Flakes

The glitter shader is efficient to render and easy to control, but when the camera moves away from the glitter-shaded assets and characters, we need to solve the "many more than four flakes per pixel" problem.

We can extend our algorithm to support glittery looks with density N, ranging from 10^2 to 10^8 by modifying our `Noise Worley` algorithm to return all of the spheres within the noise cloud that overlap the shading micropolygon. The regular grid structure of the noise cloud allows us to quickly identify all of the grid cells that overlap the bounding sphere of the shading polygon. Then, within each grid cell, we perform our efficient coverage test to determine if each feature sphere overlaps with the shading polygon disk and return the coverage and normal for each visible flake. We traverse the grids in near to far wavefront order, so we can keep track of the nearest four visible flakes easily.

To construct the generalized glitter BRDF, we first create a GGX microfacet lobe for each one of the nearest four visible flakes, $f_{[1,4]}$. Then, we create a separate mirror-flake BRDF [AK16] using the remaining visible flakes, $f_{[5,K]}$ (where K is the total number of flakes covering the shading micropolygon), with the weight of the mirror-flake BRDF set to $1 - \sum_{i=1}^{4} \text{coverage}(f_i)$. When the estimated flake count within a shading polygon falls within a user-configurable range (our default is $[500, 2000]$), we fade the mirror-flake BRDF to a GGX microfacet lobe of the same roughness, as described in [AK16]. As a result, we have a glitter BRDF solution that supports glittery flakes with density N ranging from 10^2 to 10^8, as shown in Figure 13.

The generalized glitter flake BRDF, with support for the full density range, remains energy conserving and reciprocal, because it is still a linear combination of energy conserving and reciprocal BRDF models, scaled by normalized weights. To efficiently sample and evaluate the generalized glitter BRDF, we rely on the efficient sampling and evaluation strategy of each component: the glitter flake BRDF from Section 3.3, the mirror-flake BRDF from [AK16], and the GGX microfacet lobe from [Wal+07].

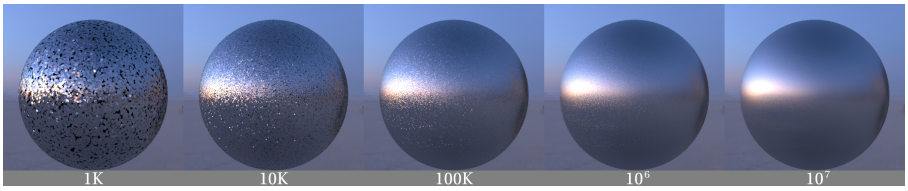

Figure 13: Glitter material with increasing density from 10^3 to 10^7, roughness = 0.3.

3.7 Production Results for Glitter

We developed a glitter material shader using our glitter BRDF and used it extensively on the movie *Trolls*. The main character, Poppy, had glitter on her cheeks and a key character, Guy Diamond, was covered in glitter, as were many secondary and crowd characters. To allow for maximum flexibility, look development artists layered the glitter material shader on top of the baseline surface or hair material. Where applicable, the lighting team sometimes created separate glitter AOVs to accentuate them during compositing, but in general glitter was lit using the same lighting rig as the underlying surface. This is directly due to our glitter material model being physically plausible. As such, it exhibits consistent behavior under different lighting conditions, and its lighting response is also consistent with other physically plausible materials at DreamWorks.

In the future, we would like to evaluate Gulbrandsen's *Artist Friendly Metallic Fresnel* [Gul14] with our glitter BRDF, for metallic flakes.

Figure 14: Generic glittery Trolls rendered with our glitter shader.

4 DWA FABRIC MODELING

Since *Shrek 2*, DreamWorks artists have used the fabric model developed by [GD04] extensively on cloth material shading. Even after we developed the physically based micro-cylinderical cloth model by [Sad+13], they continued to prefer the intuitive control of the DreamWorks fabric shading model (also a cyindrical shading model), with its easy to use artistic controls for highlights and highlight directions.

In this section, we will present our approach to making this much-loved, production-oriented fabric shading model physically plausible, so we can use it within the framework of a physically based renderer that requires its BRDF models to be energy conserving and, ideally, reciprocal as well. We also cover an accurate importance sampling algorithm for this fabric shading model that makes it efficient to render within a path tracer.

4.1 DWA Fabric BRDF

As described in [GD04], DreamWorks' fabric shading model considers the class of fabrics composed of cylindrical, dielectric fibers that are woven together. We assume that individual fibers are not discernible and instead choose to model the light response as the aggregate light reflected from a small patch of fibers. Marschner et al. [Mar+03] showed that light incident on a dielectric cylinder is reflected in a specular *cone* along the mirror vector with respect to the tangent. Our BRDF models this specular cone with the following equation:

$$S(\theta_i, \phi_i, \theta_o, \phi_o) = (1 - |\sin\theta_h|)^n, \tag{4}$$

where $\theta_h = (\theta_o + \theta_i)/2$, and θ is the angle with respect to the normal plane.

4.2 Energy Conservation for Fabric BRDF

In order for our fabric model to be physically plausible, we need to ensure that it is energy conserving. We compute the normalization factor for our BRDF by integrating it over the outgoing hemisphere for a maximally reflective incident vector. For a cylindrical fiber, the maximally reflective incident vector will always be located in the normal plane, thus $\theta_i = 0$ and therefore $\theta_h = \theta_o/2$. The normalization integral over the entire outgoing hemisphere is

$$I_o = \int_\Omega (1 - |\sin\theta_h|)^n \, \mathrm{d}\omega_o. \tag{5}$$

This integral can be evaluated using a combination of binomial expansion and a recursive formulation for the integral of an exponential sinusoid. The final expression, which can be evaluated analytically for any given exponent, comes out to be

$$I_o = 4\pi \sum_{k=0}^{n} \binom{n}{k} (-1)^k \left(\int_0^{\frac{\pi}{4}} \sin^k \theta_h \, \mathrm{d}\theta_h - 2 \int_0^{\frac{\pi}{4}} \sin^{k+2} \theta_h \, \mathrm{d}\theta_h \right). \tag{6}$$

The reciprocal, $1/I_o$, is the normalization factor for the BRDF. In practice, we precompute the normalization factors for a predetermined range of exponents. In our tests, a range of 0 to 30 for the exponent was more than sufficient to express the spectrum of fabrics required in our workflow. To allow for intuitive user control, we expose a fabric roughness α parameter that is internally converted to exponent n using the following equation:

$$n = \mathrm{ceil}(1 + 29\,(1 - \alpha)^2). \tag{7}$$

Figure 15 shows the results of white furnace tests, where we see that total reflected energy increases with roughness for the unnormalized fabric BRDF, while the normalized fabric BRDF test results show consistent reflected energy across all roughness values.

4.3 Importance Sampling of Fabric BRDF

To sample the fabric BRDF efficiently, we first select θ_h based on the following PDF:

$$p(\theta_h) = \frac{n+1}{\pi} (1 - \sin\theta_h)^n. \tag{8}$$

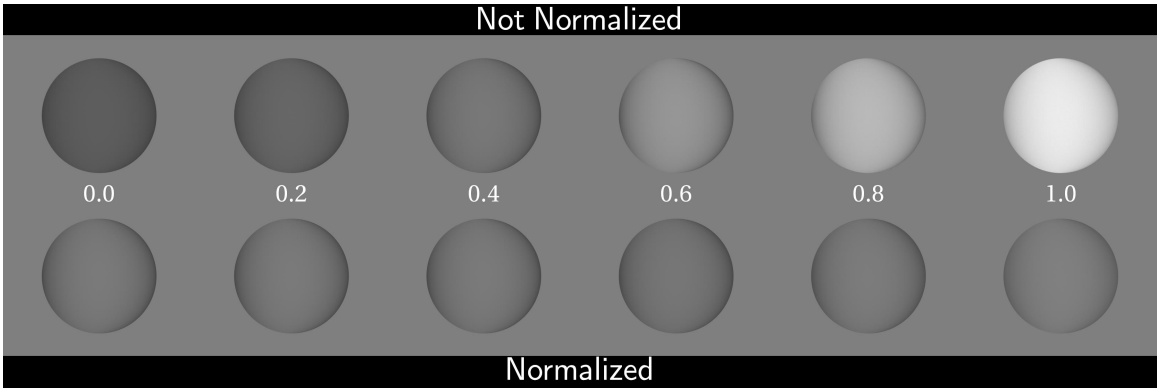

Figure 15: Furnace tests of the fabric model with roughness α ranging from 0 to 1, before and after normalization.

This PDF can be accurately sampled using

$$\theta_h = \sin^{-1}(1 - \xi^{n+1}). \tag{9}$$

We then convert θ_h to θ_i, using $\theta_i = 2\theta_h - \theta_o$ to compute the sample direction. Additionally, we divide the PDF of θ_h by the Jacobian of the inversion, in order to determine the PDF of θ_i:

$$p(\theta_i) = p(\theta_h) \frac{\mathrm{d}\omega_h}{\mathrm{d}\omega_i}. \tag{10}$$

Finally, in [Wal+07], the Jacobian to convert the sampling from the half vector to the incident vector was shown to be $\frac{\mathrm{d}\omega_h}{\mathrm{d}\omega_i} = \frac{1}{4(\omega_i \cdot \omega_h)}$, so the PDF of θ_i is

$$p(\theta_i) = \frac{n+1}{4\,\pi\,(\omega_i \cdot \omega_h)} (1 - \sin\theta_h)^n. \tag{11}$$

Our sampling algorithm is highly efficient because it is based on the exact inversion of the analytical integral of the PDF. Figure 16 shows a simple cylinder rendered with our fabric material model with low roughness. In this case, using uniform sampling requires 6x to 8x the number of samples to converge to a clean render compared to using our importance sampling algorithm. So, using our importance sampling algorithm translates to nearly a 8x speedup in a path tracer.

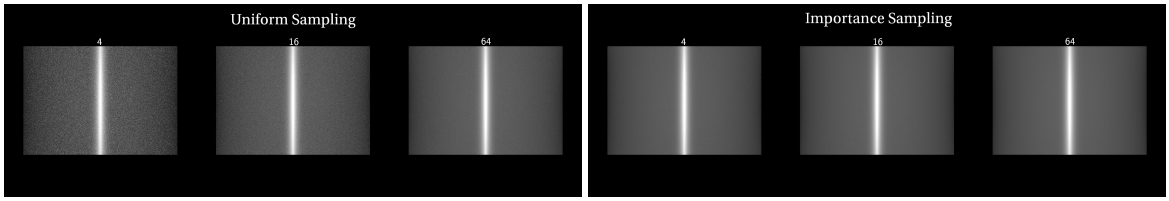

Figure 16: Uniform versus importance sampling of our fabric BRDF.

4.4 Results for Fabric BRDF

By examining the mathematical characteristics of an empirical fabric shading model, we developed an artist-friendly and physically plausible fabric BRDF, with efficient closed-form solutions for computing its normalization factor for energy conservation and for importance sampling the BRDF. Figure 17 shows a variety of cloth appearances — ranging from silk to velvet — created with our fabric BRDF (under the same lighting, with two raytrace bounces).

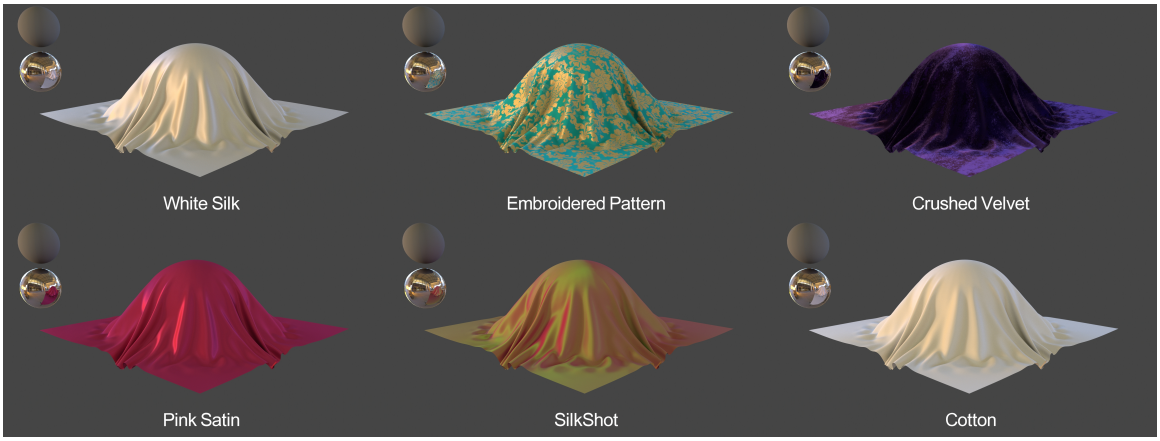

Figure 17: A variety of fabric appearances achieved using artist-specified inputs for fabric thread direction, orientation, roughness and color.

5 SUMMARY

From *Antz* in 1998 to *Boss* in 2017, PDI and DreamWorks Animation Studios have used a multi-pass scanline renderer `D_RENDER` to create over 25 feature films. Along the way, motivated by the desire for lighting consistency and efficiency, we attempted to achieve increasing levels of physical accuracy in our rendering and shading models; only to reach a ceiling in both computation and workflow complexity as we tried to expand the capability beyond what was possible in the various lighting and shading approximations we adopted. We finally transitioned to a path-traced renderer in 2016, and developed a suite of physically plausible material shaders ranging from basic hard surfaces to more specialized shading models for fabric, glitter and hair. Our users have fully embraced the new physically based renderer and shaders. They like the interactive workflow, the intuitive material controls, and the improved consistency for looks under different lighting conditions. We continue to work with them in refining the controls to achieve artistic flexibility while retaining the fundamental principles of physically plausible shading.

REFERENCES

[AK16] A. Atanasov and V. Koylazov. "A Practical Stochastic Algorithm for Rendering Mirror-like Flakes". In: *ACM SIGGRAPH 2016 Talks*. SIGGRAPH '16. Anaheim, California: ACM, 2016. URL: https://

docs.chaosgroup.com/display/RESEARCH/A+Practical+Stochastic+Algorithm+for+Rendering+Mirror-Like+Flakes.

[Bur12] B. Burley. "Physically Based Shading at Disney". In: *Practical Physically Based Shading in Film and Game Production, ACM SIGGRAPH 2012 Courses*. SIGGRAPH '12. Los Angeles, California: ACM, 2012. URL: http://selfshadow.com/publications/s2012-shading-course/.

[Chr15] P. H. Christensen. "An Approximate Reflectance Profile for Efficient Subsurface Scattering". In: *ACM SIGGRAPH 2015 Talks*. SIGGRAPH '15. Los Angeles, California: ACM, 2015.

[DJ05] C. Donner and H. W. Jensen. "Light Diffusion in Multi-layered Translucent Materials". In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. Los Angeles, California: ACM, 2005.

[GD04] R. Glumac and D. Doepp. "Generalized Approach to Rendering Fabric". In: *ACM SIGGRAPH 2004 Sketches*. SIGGRAPH '04. Los Angeles, California: ACM, 2004.

[Gul14] O. Gulbrandsen. "Artist Friendly Metallic Fresnel". In: *Journal of Computer Graphics Techniques (JCGT)* 3.4 (Dec. 2014). URL: http://jcgt.org/published/0003/04/03/.

[Jak+14] W. Jakob, M. Hašan, L.-Q. Yan, J. Lawrence, R. Ramamoorthi, and S. Marschner. "Discrete Stochastic Microfacet Models". In: *ACM Trans. Graph.* 33.4 (July 2014).

[JB02] H. W. Jensen and J. Buhler. "A Rapid Hierarchical Rendering Technique for Translucent Materials". In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. San Antonio, Texas: ACM, 2002.

[Jen+01] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. "A Practical Model for Subsurface Light Transport". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM, 2001.

[Kel+15] A. Keller, L. Fascione, M. Fajardo, I. Georgiev, P. Christensen, J. Hanika, C. Eisenacher, and G. Nichols. "The Path Tracing Revolution in the Movie Industry". In: *ACM SIGGRAPH 2015 Courses*. SIGGRAPH '15. Los Angeles, California: ACM, 2015.

[Kři+10] J. Křivánek, M. Fajardo, P. H. Christensen, E. Tabellion, M. Bunnell, D. Larsson, and A. Kaplanyan. "Global Illumination Across Industries". In: *ACM SIGGRAPH 2010 Courses*. SIGGRAPH '10. Los Angeles, California: ACM, 2010.

[Mar+03] S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. "Light Scattering from Human Hair Fibers". In: *ACM SIGGRAPH 2003 Papers*. SIGGRAPH '03. San Diego, California: ACM, 2003.

[Ou+12] J. Ou, F. Xie, P. Krishnamachari, and F. Pellacini. "ISHair: Importance Sampling for Hair Scattering". In: *ACM SIGGRAPH 2012 Talks*. SIGGRAPH '12. Los Angeles, California: ACM, 2012.

[PJH16] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.

[Sad+13] I. Sadeghi, O. Bisker, J. de Deken, and H. W. Jensen. "A Practical Microcylinder Appearance Model for Cloth Rendering". In: *ACM Trans. Graph.* 32.2 (Apr. 2013).

[Sch94] C. Schlick. "An Inexpensive BRDF Model for Physically-based Rendering". In: *Computer Graphics Forum* 13.3 (1994).

[TL04] E. Tabellion and A. Lamorlette. "An Approximate Global Illumination System for Computer Generated Films". In: *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. Los Angeles, California: ACM, 2004.

[Wal+07] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. "Microfacet Models for Refraction Through Rough Surfaces". In: *Proceedings of the 18th Eurographics Conference on Rendering Techniques*. EGSR'07. Grenoble, France: Eurographics Association, 2007.

[Wor96] S. Worley. "A Cellular Texture Basis Function". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: ACM, 1996.

APPENDIX A - NORMALIZING THE DREAMWORKS FABRIC BRDF

When normalizing a BRDF, we are interested in the maximum reflected energy for any given incident light direction. The normalization integral for our cloth BRDF can be written as

$$I_o = \int_\Omega (1 - |\sin\theta_h|)^n \, \mathrm{d}\omega_o. \tag{12}$$

This integral can be expressed in spherical coordinates as

$$I_o = \int_0^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} (1 - |\sin(\theta_h)|)^n \cos\theta_o \, \mathrm{d}\theta_o \mathrm{d}\phi_o. \tag{13}$$

Since our ϕ plane is perpendicular to the tangent vector, this conversion includes a cosine term instead of the sine term. We can integrate the independent ϕ term and remove the absolute operator using the symmetry of the cloth BRDF. With these changes, the integral becomes

$$I_o = 2\pi \int_0^{\frac{\pi}{2}} (1 - \sin\theta_h)^n \cos\theta_o \, \mathrm{d}\theta_o. \tag{14}$$

For a cylindrical BRDF, the maximum reflected light energy occurs when the incoming light vector is in the normal plane. This makes $\theta_i = 0$, and $\theta_o = 2\theta_h$. Using this relation, we can express the integral in terms of θ_h:

$$\begin{aligned} I_o &= 2\pi \int_0^{\frac{\pi}{4}} (1 - \sin\theta_h)^n \cos(2\theta_h) \, \mathrm{d}(2\theta_h), \\ &= 4\pi \int_0^{\frac{\pi}{4}} (1 - \sin\theta_h)^n (1 - 2\sin^2\theta_h) \, \mathrm{d}\theta_h. \end{aligned} \tag{15}$$

We can use binomial expansion to expand the $(1 - \sin\theta_h)^n$ term as

$$(1 - \sin\theta_h)^n = \sum_{k=0}^{n} \binom{n}{k} (-1)^k \sin^k\theta_h. \tag{16}$$

The integral can now be written as

$$I_o = 4\pi \int_0^{\frac{\pi}{4}} \left(\sum_{k=0}^{n} \binom{n}{k} (-1)^k \sin^k\theta_h \right) (1 - 2\sin^2\theta_h) \, \mathrm{d}\theta_h. \tag{17}$$

Moving the integral inside and factoring out the two terms, we get an expression than can be evaluated as a summation over the difference of integrals of two sine functions:

$$I_o = 4\pi \sum_{k=0}^{n} \binom{n}{k} (-1)^k \left(\int_0^{\frac{\pi}{4}} \sin^k\theta_h \, \mathrm{d}\theta_h - 2\int_0^{\frac{\pi}{4}} \sin^{k+2}\theta_h \, \mathrm{d}\theta_h \right). \tag{18}$$

The integral of $\sin^k x$ has a well-studied recursive form:

$$\int \sin^k x \, \mathrm{d}x = -\left(\frac{1}{k} \sin^{k-1} x \, \cos x + \frac{k-1}{k} \int \sin^{k-2} x \, \mathrm{d}x \right). \tag{19}$$

The binomial expansion, along with the above recursive evaluation, results in an integral that can be evaluated analytically for any given exponent, giving us the maximum reflected energy I_o a given exponent. The reciprocal of I_o is our normalization factor.

APPENDIX B - IMPORTANCE SAMPLING THE DREAMWORKS FABRIC BSDF

The BRDF can be efficiently sampled by choosing a half vector with PDF $p(\theta_h)$ and using change of variables to convert this PDF to $p(\theta_i)$. We select a half-angle vector based on the equation

$$p(\theta_h) = K\,(1 - |\sin\theta_h|)^n, \tag{20}$$

where K is a normalization factor that we need to determine. For it to be a valid PDF, we need to ensure that

$$\int_{\Omega_h} p(\theta_h)\,\mathrm{d}\omega_h = 1, \tag{21}$$

which expands to

$$K\int_0^{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}(1 - |\sin\theta_h|)^n\cos\theta_h\mathrm{d}\theta_h\mathrm{d}\phi_h = 1. \tag{22}$$

Using the the symmetric property of the function around $\theta_h = 0$, we obtain

$$2\pi\,K\int_0^{\frac{\pi}{2}}(1 - \sin\theta_h)^n\cos\theta_h\mathrm{d}\theta_h = 1. \tag{23}$$

Substituting $t = \sin\theta_h$ yields

$$\begin{aligned} 2\pi\,K\int_0^1(1-t)^n\mathrm{d}t &= 1,\\ 2\pi\,K\frac{1}{n+1} &= 1. \end{aligned} \tag{24}$$

Thus $K = \frac{n+1}{2\pi}$, so our PDF is

$$p(\theta_h) = \frac{n+1}{2\pi}\,(1 - |\sin\theta_h|)^n. \tag{25}$$

Using CDF inversion, we can generate sampling vectors accurately using

$$\begin{aligned} \theta_h &= \pm\sin^{-1}(1 - \xi_1^{\frac{1}{n+1}}),\\ \phi_h &= \pi\xi_2. \end{aligned} \tag{26}$$

To be a valid PDF with respect to the scattering vector ω_i, we need to include the Jacobian of the half direction transform, as described in [Wal+07]:

$$p(\theta_i) = p(\theta_h)\left\|\frac{\partial\omega_h}{\partial\omega_i}\right\|. \tag{27}$$