BRDF

Bidiractional Reflactance Diffusion Function

Computer Graphics CS 488

Francesco Paduano
Matthias Untergassmair
Gianluca Venturini
Muxuan Wang

November 14, 2014

Contents

1 Brief Introduction to BRDF

The *Bidirectional Reflectance Distribution Function* is a function which defines how an opaque surface reflects the light. A BRDF function takes as inputs the direction of the incoming light and the direction of the outgoing light and returns the ration between the reflected radiance over the incoming irradiance.

The aim of BRDFs is to find accurate approximations of the physical phenomena involved in the interaction of light and matter. It allows us to model material properties in more detail, thereby making objects look more realistically.

The BRDF function depends on the wavelength of the light and on a large amount of different material properties. BRDF models differs from each other in terms of which properties of the material they are intended to represent.
The necessity of Computer Graphics to have lightweight models for lighting has given birth in the past to some non-realisic shaders as the *phong shader*. Real time rendering has found major applications in video games where photorealistic results are difficult to obtain due to hardware limitations
The surging improvements of the hardware technologies and the wide applications of *non real time rendering* has raised new intereset in photorealistic results and has brought the necessity to build lighting models coherent to the physic of the light. Thus, BRDFs has played an important role in introducing photorealistic lighting effects in Computer Graphics, even if the needs of an efficient representation has often overcame the phyisical formulation.
This paper will presents in a first place the principal components of BRDF, such as

- Fresnel Reflectance
- Surface Scattering
- Roughness Modeling
- Anisotropic Modeling

Then it will follows an overview of different BRDF models, such as

- The Lambertian BRDF
- The Phong BRDF
- The Blinn-Phong BRDF
- The Ward BRDF
- The Oren-Nayar BRDF

The principles presented in this short paper heavily rely on energy conservation laws and in many cases statistical assumptions are made to model much more complex and involved phenomena.

2 Reflectance and Refraction

When the light comes from a material with a given refractive index *n1* to a second material with a different refractive index *n2* it is both reflected and refracted.
Under the assumption of a flat surface [1] the direction of the light refracted follow the *Snell's Law* wheras the light reflected follow the *Laws of Reflection*
The amount of light reflected and refracted is a property of the material considered. *Fresnel Equations* are a matematical formulation to compute the light reflected given the angle of incidence and the material properties.

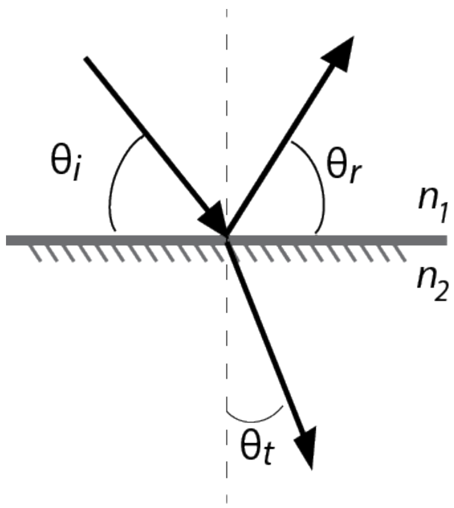

Figure 1: Diagram showing the incidence light, reflection light and reflaction light

2.1 Snell's Law

The *Snell's Law* is a formula which describe the relationship between the angle of incidence an the angle of refraction. Using Figure 1 as reference, the *Sneel's Law* can be formulated as follows

$$\frac{sin(\theta_t)}{sin(\theta_i)} = \frac{n_2}{n_1}$$

Where n_1 and n_2 are respectively the refraction index of the material where the light is coming and the refraction index in which the light in entering.

[1] A flat surface is a surface which does not have any irregularities smaller than the light wavelenght

2.2 Reflection Laws

The well-know *Reflection Laws* define the angle of reflection θ_r given the incidence angle θ_i and the plane where the the incidence vector lies.
The laws are:

1. The incident ray, the reflected ray and the normal to the reflection surface at the point of the incidence lie in the same plane.

2. The angle which the incident ray makes with the normal is equal to the angle which the reflected ray makes to the same normal.

3. The reflected ray and the incident ray are on the opposite sides of the normal.

2.3 Fresnel Reflectance

Fresnel Reflectance is a phenomenon which occurs on the surface of an object illuminated by the light.
The Fresnel Reflectance R_f can be expressed as a function of only the incoming angle θ_i. Obviously, the function R_f is different for each material. Figure 2 plots R_f according to different incidence angles and different materials.
Is worth noticing that every materials has a *Fresnel Reflectance* equals to 1 when the incidence angle is parallel to the surface normal, but might have different values for the angle of incidence equals to 0.

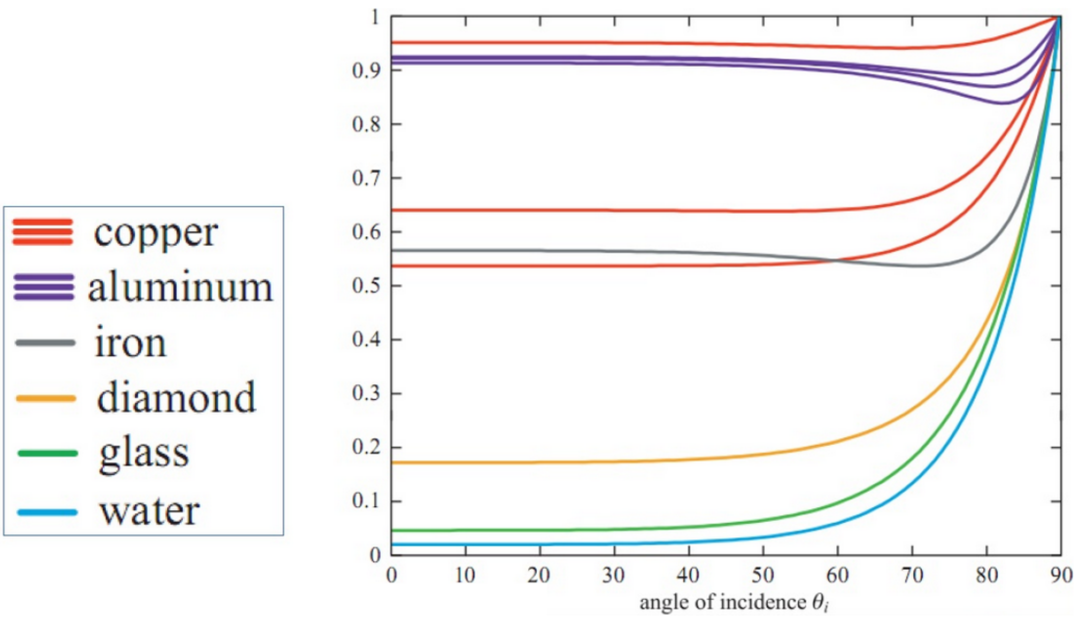

Figure 2: Fresnel Equations for some material samples

3 Computer Graphics implementation

For illustrative purpose we have developed a toy application in OPENGL and GLSL which implements Fresnel Equations.
The scene is composed by a pool modeled with the Blender software[2], a plane surface composed by 10.000 faces and one light.
The water surface is distorted by a noise function developed ad-hoc to simulate the water waves.
The purposes of the demo are to implement the Fresnel Equations and other reflection and refraction effects for a realistic water surface.

3.1 Approximation of Fresnel Equation

Various tecnique are available for a fast computation of the Fresnel Equations.
A first category of tecniques precalculate the equations values and store them in a texture binded to the shader.
A second category of tecniques uses a matematical approximation of the Fresnel Equations. In the current demo we have used the following approximation [1]

$$R_f(\theta_i) \approx R_f(0) + (1 - R_f(0))(1 - cos(\theta_i))^5$$

And the value of $R_f(0)$ for the water has been estimated given the *refraction index* n of the water and the following formula

$$R_f(0) = (\frac{n-1}{n+1})$$

Figure 3 shows the results of the Fresnell Reflection on the water surface. The effects of two different values of $R_f(0)$ are showed in the picture: on the left it has been used a value close to the Aluminum whereas on the right it has been used the correct value of $R_f(0)$ for the water.

3.2 Skybox reflection

Fresnell Reflection is not enough to achieve a realistic water material. By using a 3D texture we have added to the surface of the water the reflection of the surrounding environment. In order to compute the reflaction direction it must be taken in account that the normal direction of the faces has been changed due to the waves.
Given $y = noise(x, z, t)$ the noise function used for simulating the waves, the normal direction of the point $p = x, y, z$ can be computed by considering the vectors v_1, v_2 defined as the difference between two points close Δx and Δy from p and affected by

[2]www.blender.org

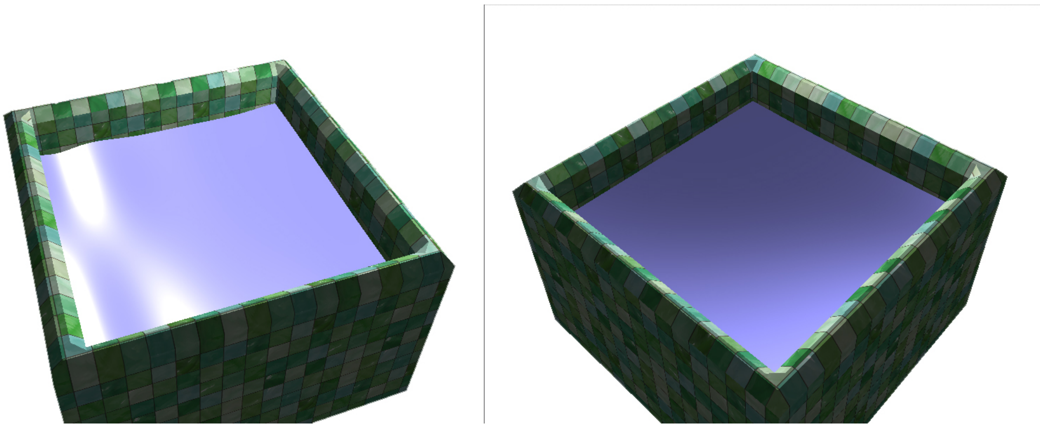

Figure 3: On the left the water renderd with RF(0) = 0.9 (Alluminum), on the right the water rendered with RF(0) = 0.02 (Water)

the same noise function. Then, the normal direction can be computed as follows:

$$normal = v_1 \times v_2$$

Please use Figure 4 as reference.
Figure **??** makes a comparison between a flat reflection and a reflection which takes in account of the deformed normals of the surface.

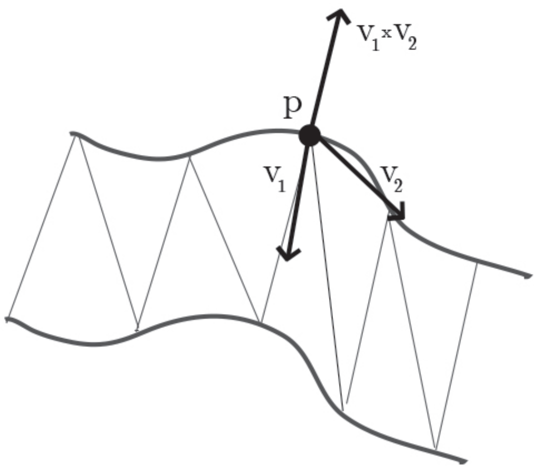

Figure 4: computation of the normal of a deformed surface

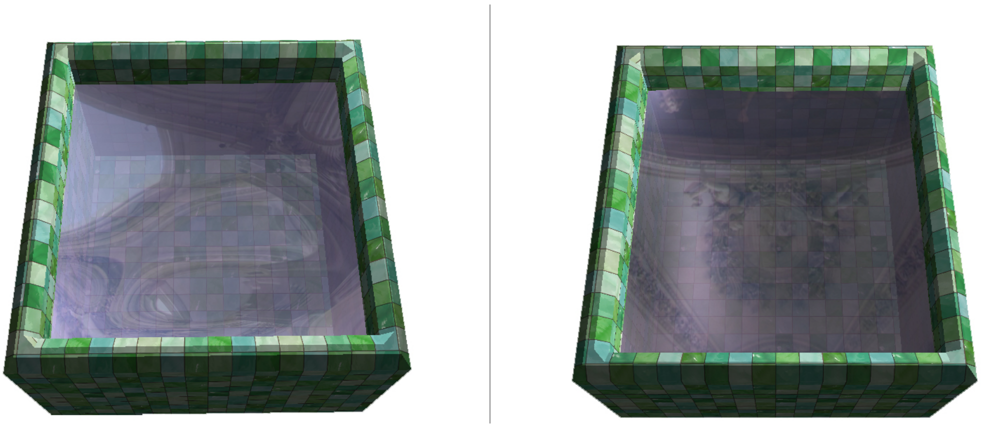

Figure 5: On the left the reflection are subject to the movements of the surface, on the right the flat reflection

3.3 An easy approach for light refraction

When the light bumps on the water surface is reflected and refracted. The *Fresnell Equations* can be employed for computing the amount of light refracted, indeed under the assumption of conservation of energy the following equation holds

$$R_f = 1 - R_r$$

where R_r is the amount of light refracted.

But the light refracted change direction according to the *Snell's Law*. That means that the objects visible behind the water surface must be deformed accordingly.

This could seem challengy to achieve, but the problem has been solved easly by using a *Frame Buffer Object* and *multi-pass rendering*.

The scene is rendered on a a first time without the water surface on a *Frame Buffer Object*. Then, the scene is rendered again and the *Frame Buffer Object* is passed as a texture to the water shader. The shader performs the clipping, projections and deformations of the area of the texture according to the refraction angle. Is important to notice that this tecnique is not 100% realistic. The viewer has only the impression to see a refracted image, it is actually just deformed in a semi-realistic way.

Figure 6 illustrates the difference between the water with and without the refraction effect.

Finally, Figure 7 shows the final result of the combination of Fresnell Reflection, skybox reflection and refraction.

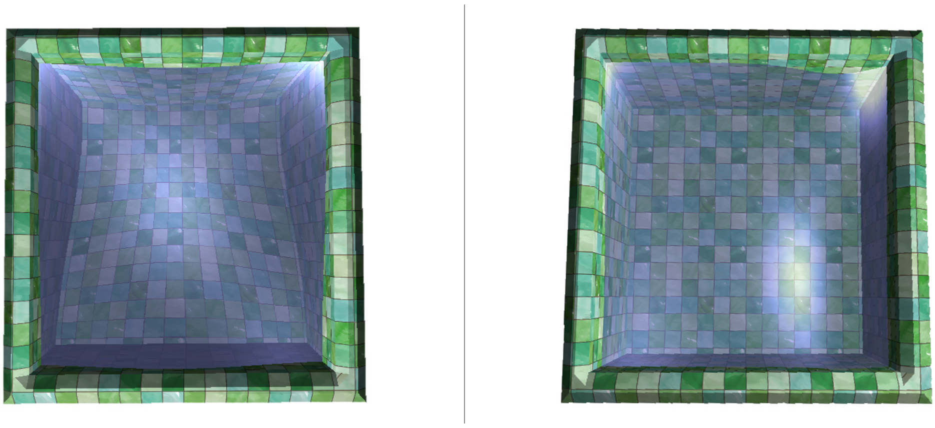

Figure 6: On the left the transparent water with the refraction effect (exaggerated), on the right the transparent water without refraction

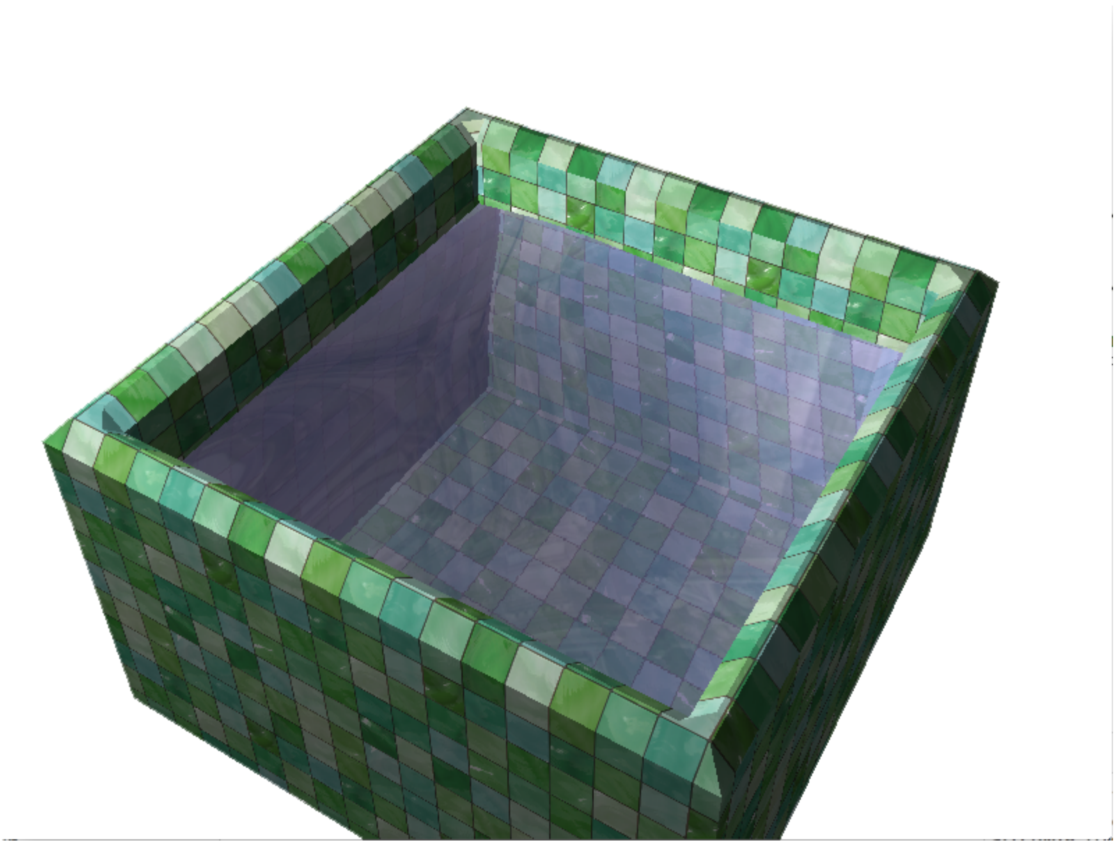

Figure 7: The final result of the pool, combination of fresnel reflection and refraction, skybox reflection and multi-pass refraction

4 Subsurface Scattering

Subsurface Scattering (SSS for short) is a technique used in Computer Graphics to represent materials more realistically. This is achieved by finding models that approximate the physical processes taking place when light penetrates a translucent or semi-translucent non-metallic objects surface and scatters beneath the surface.

SSS is one of the most important techniques for realistically rendering non-metals.

In general, a distinction needs to be made between optical phenomena that occur at the objects (infinitely thin) surface and those occurring under the surface, i.e. inside the body. We speak of *surface reflectance* and *volume* or *body reflectance.* [3]

Surface Reflectance The surface itself is an infinitely thin *optical discontinuity*: it scatters light, but it does not absorb any light. In fact, all incoming light is either reflected or transmitted.
FAs we have seen in previous sections, Fresnel reflectance is a good model for this effect. It assumes that the angle of the outgoing light ray is equal to the angle of incidence of the light. It further takes account of the portion of the light that is absorbed by the object and the associated “loss” of energy [4].
Subsurface Scattering goes one step further in the sense that it addresses the question: *What happens to light, that is not undergoing surface reflectance?*

Body Reflectance The light being transmitted in the interior of the object may be absorbed, further scattered and reflected, and in some cases even exit the material again. This mostly depends on the composition of the objects interior, i.e. the optical properties associated with the material.

If the object does not have a perfectly uniform density, there will be further *optical discontinuities* in the layer under the surface that can change the path of the light. For this reason, surface transmission is very important also in highly absorptive materials [**?**].

A real world example showing the effects of body reflectance is the following: The foam on a liquid always appears brighter than the liquid itself. The optical properties of the liquid itelf are still the same, but the numerous additional air-liquid interfaces greatly increase the amount of scattering, therefore light is more likely to be reflected and finally exit the liquid again rather than to just be absorbed in the liquid. As there is a greater amount of light reflected than in the liquid with no air bubbles, it appears to be brighter.

It is important to note that the SSS technique does not aim to reconstruct the exact path of the light inside the material. Although this would theoretically be possible -

[3] Note that specular BRDF terms are mostly associated with surface reflectance, while diffuse terms are mainly modeling body reflectance.

[4] of course, energy is not lost, but converted into thermal energy of the object

all interactions of light and matter can be described by reflection, refraction and (total) reflectance - it is not desireable to do so:
One reason is that we mostly don't know about the exact alignment of optical discontinuities within the material itself.
However, and more importantly, any such *ray tracing* technique would be far too expensive for real time rendering and equally good results can be achieved by using mathematical models that try to approximate this behavior.
This is where energy conservation and statistical assumptions come into place.

4.1 History of Subsurface Scattering

Early representations in computer graphics were often looked lifeless and plastic because they were lacking practical models for simulating translucence or subsurface scattering. The BRDF approach to subsurface scattering is the earliest and simplest approach to representing subsurface scattering. Modern research focuses more on techniques for subsurface scattering that do not rely on the BRDF model.
This model was used up until the late 90s when Henrik Jensen first introduced the BSSRDF model into computer graphics. Jensen's BSSRDF research was used to render Gollum's skin in The Lord of the Rings films, for which he was awarded the first ever Academy Award for technical achievement.

The modern developments after the early-to-mid 2000s rarely relies on BRDF for representing subsurface scattering, unless the model represents the material properties sufficiently well [**?**]

4.2 Local Subsurface Scattering

The term *local* subsurface scattering encodes a very important assumption: It assumes that - after undergoing subsurface scattering and being partially absorbed - the light exits the material in the same point where it entered the material.
It can be seen easily that this requirement is not met in reality. However, the difference between point of incidence and point of exit are in most cases so small, that the assumption is absolutely valid on a macroscopic scale.

Having stated this elementary assumption, it also shows the limitations of subsurface scattering. SSS can only be applied in cases where the path travelled inside the material is sufficiently small to still meet the above requirements.
If this is not the case (e.g. for highly transparent materials like water, oil, jelly), then we have to rely on a different type of SSS, namely *global* subsurface scattering.

Global SSS is another important technique for reallistically representing materials with a higher level of transparency. However, global subsurface scattering cannot be modeled by BRDF and is therefore beyond the scope of this paper.

A quick overview and references to other sources relating to the topic is given in section 4.3.

4.2.1 Physical Background

We know that light is an *electromagnetic* wave, therefore metals[5] (*conductors*) and *semi-conductors* quickly absorb any transmitted light.
Without going into the physical details, this can be explained intuitively by recalling that the electromagnetic wave consists of mutually inducing magnetic and electric fields. In the presence of charge carriers, these electric fields are instantaneously dissipated and the light is "absorbed".
What happens from an energetic point of view is that the energy contained in the light (i.e. in its electric field) is converted into kinetic energy of the charge carriers (i.e. thermal energy), causing the material to heat up.
For this reason, body reflectance does not need to be considered for conductors.

The situation is different for *insulators*: the electromagnetic wave can still live inside the object, therefore the light ray can undergo further interactions once inside the body. Since insulators transmit most incoming light rather than reflecting it, the subsurface scattering effects are usually more visually important than the Fresnel reflectance ($R_F(\theta_i)$).
It should be noted here that many of the objects we want to render are in fact insulators, such as all organic materials like plants, human skin, etc. as well as anorganic materials like stone or plastic.
Therefore, the realistic rendering of such objects is of big interest.

If an insulator is homogeneous or nearly homogeneous[6], then it is percieved as transparent (e.g. crystal, water, etc.).

However, most insulators are *heterogeneous*, they have numerous microscopic discontinuities caused by structural changes, density variations, foreign particles, enclosed air bubbles, and many more.

With each material, we can associate a characteristic *scattering albedo* ρ:

$$\rho := \frac{\text{energy of the light exiting the material}}{\text{energy of the light entering the material}}$$

As noted in [**?**] and many other scientific papers, there exist simple and efficient methods to determine the albedo of a given material in experiments.

If we consider objects that only interact with incoming light, but do not create their own light, then the scattering albedo must clearly satisfy $0 \leq \rho \leq 1$ where $\rho = 1$ corresponds to a material that does not absorb any energy and $\rho = 0$ for a material that absorbs all the energy of the incoming light. The interesting values of ρ however are in between these two extreme values.
In general, the scattering albedo can also depend on the frequency of the light, i.e. $\rho = \rho(\nu)$.

[5] An exception must be made for Aluminum

[6] the word homogeneous here is intended in the relevant length scale, i.e. the orders of magnitude of visible light $\sim 400nm$ and larger

Material	ρ
snow	0.8
white paint	0.7
stone	$0.15 - 0.4$
coal	≈ 0.0

Table 1: Characteristic scattering albedo values some example materials

Some characteristic scattering albedo values are shown in table 1. Although the Fresnel reflectance and the scattering albedo have similar mathematical representations, they rely on different physical phenomena: Fresnel reflectance takes place solely on the surface of materials, while subsurface scattering is caused by body reflectance. In particular, ρ and $R_F(\theta_i)$ can have very different spectral distributions.

An interesting real world example showing the difference between these two coexisting effects is colored plastic. Colored plastic is composed of a clear, transpartent substrate with embedded pigment particles for the color. For this reason, light reflecting specularly will be uncolored, while diffuse light will be colored when it is scattered from the colored pigments under the surface.

4.2.2 Representation of local subsurface scattering

In most cases, local subsurface scattering is modeled as a Lambertian diffuse term in the BRDF.

$$f_{\text{diff}}(\boldsymbol{l}, \boldsymbol{v}) = \frac{\boldsymbol{c}_{\text{diff}}}{\pi}$$

When chosing $\boldsymbol{c}_{\text{diff}}$, several considerations should be made.

The first approach would be to include the scattering albedo ρ and chose

$$f_{\text{diff}}(\boldsymbol{l}, \boldsymbol{v}) = \frac{\rho}{\pi} \tag{1}$$

However, this approach is physically incorrect since it does still not account for the fact that only the light that is not reflected by fresnel reflectance is available for subsurface scattering. However, this can easily be corrected by including an additional term

$$f_{\text{diff}}(\boldsymbol{l}, \boldsymbol{v}) = \frac{\boldsymbol{c}_{\text{diff}}}{\pi} = (1 - R_F(\theta_i))\,\frac{\rho}{\pi} \tag{2}$$

Note that the BRDF value depends on the incident angle θ_i, but not on the outgoing direction.
Intuitively, this can be explained by the assumption that the direction of subsurface scattering is randomized, so the intensity is the same for each outgoing direction.
However, even this assumption is not fully correct yet: the light rays must still undergo

Fresnel reflectance on their way out of the material, where transmission and internal reflection will may take place. This will impose some directional preference of the outgoing light.

To resemble more realistic conditions, more complicated terms can used, like

$$f_{\text{diff}}(\boldsymbol{l}, \boldsymbol{v}) = k_{\text{norm}} \left(1 - R_{\text{spec}}(\boldsymbol{l})\right) \left(1 - R_{\text{spec}}(\boldsymbol{v})\right) \rho$$

However, for most applications, the simple approximation 2 is sufficiently accurate.

Figure 8: Surface reflectance (left), body reflectance (center) and combined effect (right) **Source:** "ShellOpticalDescattering" by Meekohi - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:ShellOpticalDescattering.png#mediaviewer/File:ShellOpticalDescattering.png`

4.3 Global Subsurface Scattering

In contrast to local subsurface scattering (section 4.2), *global subsurface scattering* considers materials where the point of incidence of light in the material and ... cannot be assumed to be the same anymore.
This model needs to be applied for materials that have higher degrees of transparency.

More general models for subsurface scattering go far beyond the BRDF model and therefore also beyond the scope of this paper. A very good reference can be found in [**?**] that describes the bidirectional surface scattering distribution function (BSSRDF) model that can describe light transport between *any* two points of a surface.
Such techniques allow to further visualize effects like color bleeding within materials and diffusion of light across shadow boundaries.
The representation of multilayered materials is discussed in [**?**] and shows very impressive rendering of plants and human skin.
A more practical introduction to implementing real time subsurface scattering can be found in [**?**].

In conclusion, it should be noted that using BRDF for subsurface scattering is a very good model for making a large range of materials look more realistic, but it is still a

very raw approximation of the physical truth.
In fact, most of the current research does not rely on the BRDF for modelling subsurface scattering.
Nevertheless, the BRDF model is very inexpensive compared to other methods and is therefore still the method of choice in many real time visualizations.
[?] talks about recent alternative methods that allow real time rendering of translucent materials. The key idea in this paper is to substitute the 3D integration by a 2D texture integration, which makes computations significantly more efficient on the GPU side. This model explicitly includes both local and global scattering components.

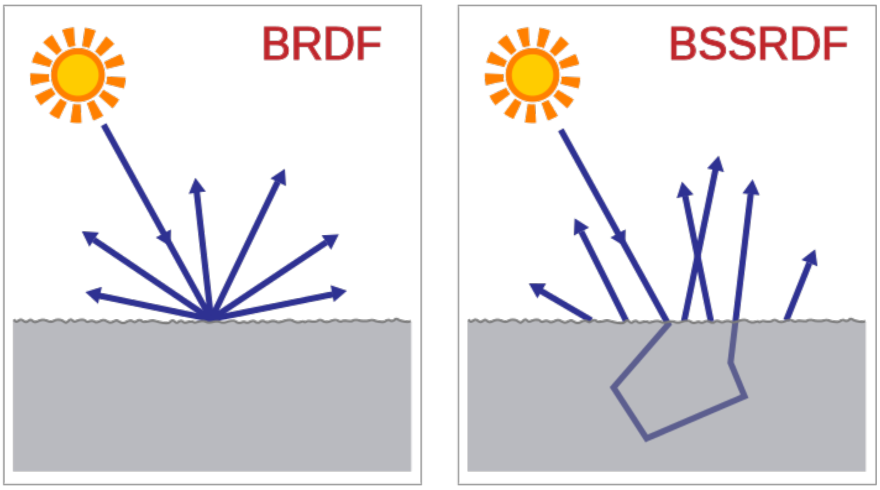

Figure 9: BRDF vs. BSSRDF: the BRDF model (left) assumes local subsurface scattering, i.e. the light exits the object in the same point where it entered it. This approximation is valid for materials with very low transparency.
Source: "BSSDF01 400" by Jurohi (original); Pbroks13 (redraw) Original uploader was Pbroks13 at en.wikipedia - Transferred from en.wikipedia; transfered to Commons by User:Pbroks13 using CommonsHelper.(Original text : `http://en.wikipedia.org/wiki/Image:BSSDF01_400.png`).Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:BSSDF01_400.svg#mediaviewer/File:BSSDF01_400.svg`

5 Roughness modeling

In order to model the reflection of the light on a surface that is not completely flat we must take into account different effects that occurs at microgeometry scale. **Microgeometry** is defined as geometry that can't be directly seen in a rendered scenes because is smaller than one pixel (irregularities smaller than one pixel and larger than 800nm can be classified microgeometry, everything smaller can be ignored because it can't interact with visible light). The microgeometry can't be modeled using vertex coordinates [14]. There are three effects that occur:

1. Shadow: the light doesn't reach all the surface because it is blocked by some irregularities. Different areas on the surface have different amount of light that reach them. See figure 10.

2. Mask: during the process of reflection part of the light is blocked by the irregularities and absorbed by the material. See figure 11.

3. Interreflection: the light is reflected more than one time on the microfacets, it can be partially absorbed and can change color. See figure reffig:interreflection.

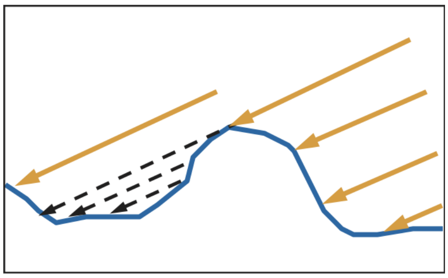

Figure 10: Shadow effect [3]

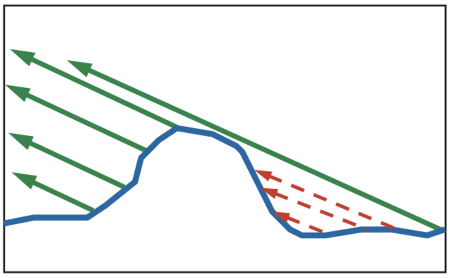

Figure 11: Mask effect [3]

5.1 Microfacets theory

In order to represent this effects and the amount of light reflected by the material Torrance and Sparrow developed the **microfacets theory** [16] that is based on modeling

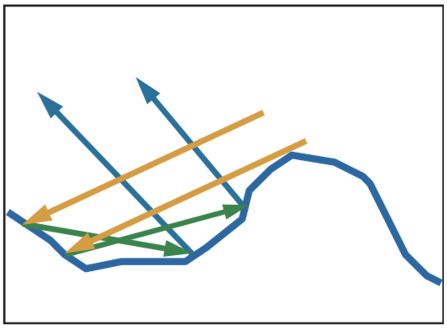

Figure 12: Interreflection effect [3]

the **microgeometry** as a collection of **microfacets**, each microfacet is a tiny flat **Fresnel mirror** and has its own normal vector. Every macro face (triangle that compose a mesh) is composed by a large number of microfacets, their normals are described using a statistical function called **normal distribution function** or NDF (usually the term $p(\theta_h)$ is used). The NDF given a certain gap angle (angle between the normal of the macroface and the microface) return the probability that a microface normal has that orientation. Since the function takes into account only the angle and not the direction it is possible to model only **isotropic** surfaces. A more general form of the NDF can be used for **anisotropic** surfaces: $p(\mathbf{h})$

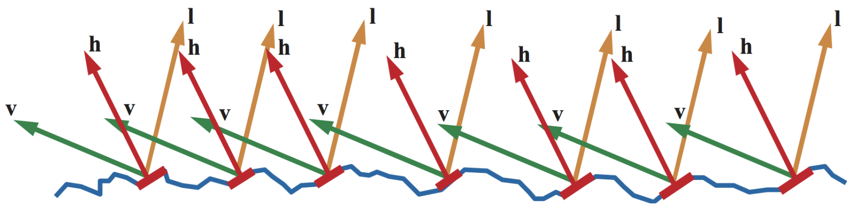

Figure 13: Active microfacets [3]

Using the **microfacets theory** it is possible to derive the following BRDF, (created the first time by Ashikhmin [14]):

$$f(\mathbf{l}, \mathbf{v}) = \frac{p(\mathbf{h})G(\mathbf{l},\mathbf{v})R_F(\alpha_h)}{4K_p\overline{cos}\theta_i\overline{cos}\theta_i}$$

1. $p(\mathbf{h})$: the normal distribution function, we can use a gaussian distribution. This term can be simplified using only the angle θ_h (the angle between the half vector and the normal of the face, see figure 14).

2. $G(\mathbf{l}, \mathbf{v})$: geometrical attenuation factor, it takes into account shadowing and masking effects.

3. $R_F(\alpha_h)$: Fresnel reflectance

4. K_p: normalization factor calculated as:

$$K_p = \int_\Omega p(\mathbf{h}) cos(\theta_h)\, d\omega$$

Where $d\omega$ is the infinitesimal solid angle and Ω is the surface of a sphere of row 1.

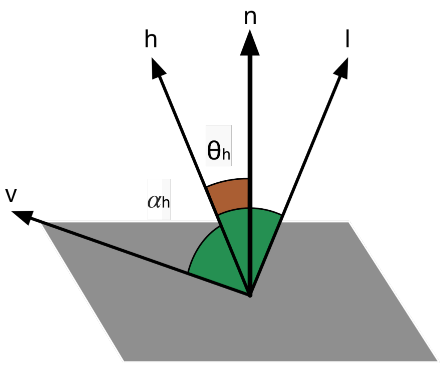

Figure 14: Half vector

5.2 Half vector and Reflection vector

In order to calculate the highlight there are two different methods that BRDFs can use:

1. **Half vector**: based on the angle between the half vector **h** and the normal angle. The half vector **h** is calculated as the average between **v** and **l** (as seen from figure 14)

2. **Reflectance vector**: based on the angle between the reflection vector **r** and the normal vector **n** (as seen from figure 15).

In the Ashikhmin [14] equation $p(\theta_h)$ it is based on the half vector. The half vector gives better results on flat surfaces. As see in figure 16 it is clear that renderings are more realistic using this technique [15].

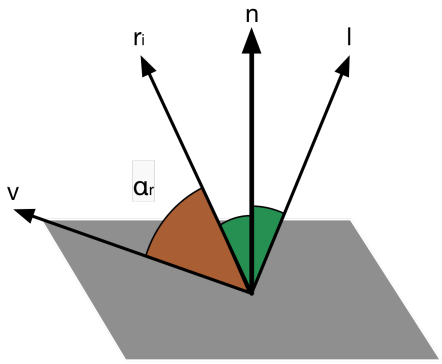

Figure 15: Reflection vector

Figure 16: Sunset photography

6 Isotropic and anisotropic surfaces

A surface is **isotropic** when it looks uniform in all the orientations, a clear example of that can be seen in figure 17. If a surface is not isotropic it is called **anisotropic** and it means that the light reflection depends on the orientation of the object (an example of anisotropic surface can be seen in figure 18)

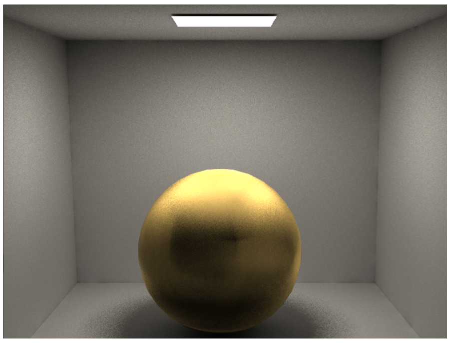

Figure 17: Isotropic sphere

Figure 18: Anisotropic surface

Not all BRDFs models can deal with anisotropic surfaces. The fastest method for rendering anisotropic objects is using **anisotropic texturing**, it is a 2D texture that encode the direction of irregularities on the material for every point (for example micro crack created during manifacturing process).
With the anisotropic texture it is available a tangent vector **t** that points in the direction of the crack for every point in the surface and it is possible to modify the BRDFs as Kajiya-Kay [17] proposed:

$$\overline{cos}(\alpha'_r) = max(\sqrt{1-(1\cdot\mathbf{t})^2}\sqrt{1-(\mathbf{v}\cdot\mathbf{t})^2}-(l\cdot\mathbf{t})(\mathbf{v}\cdot\mathbf{t}),0)$$

An example of BRDF that permits to create realistic renderings is the **Ward BRDF**.

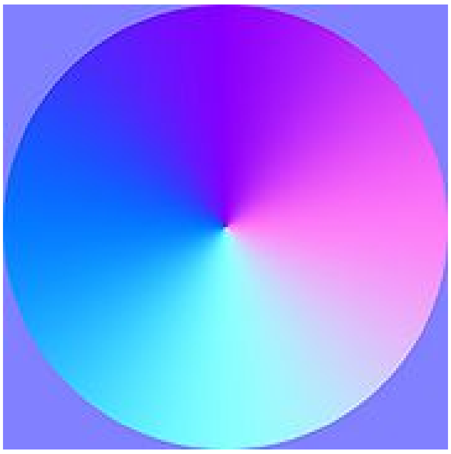

Figure 19: Anisotropic texture

7 BRDF implementation

In this section four different implementation will be discussed. Usually during the implementation of BRDF functions in realtime rendering the developer must takes into account two factor: the final result and the time that the rendering takes in order to be completed. Every material that can be created is the result of a trade off between this two parameters, since the hardware is getting faster every year and the software API (like openGL) are improving at every release the engineers can write batter looking BRDFs exploiting the higher power of the new systems and the results are materials that seems realistic.
Our group coded six simple examples in order to underline the differences between different BRDFs models and make an example of what happens changing some parameters in the models. In order to see part of the shader code for the implementation refer to Appendix A.
In the figure 20 four different BRDFs models are represented.

- Top left: Oren-Nayar
- Top right: Oren-Nayar BRDF with $\sigma = 0$ and three lights
- Middle left: Phong BRDF with three lights
- Middle right: Normalized Phong with three lights
- Bottom left: Custom Ashikhmin implementation with Fresnel Reflactance $R_F(0°) = (1.00, 0.71, 0.29)$, that is typical of the gold and $p(\theta_h)$ normal distribution with $\mu = 0$ and $\sigma = 0.6$. It also uses the Torrance and Sparrow geometrical factor (version simplified by Blinn).
- Bottom right: Custom Ashikhmin implementation with Fresnel Reflactance $R_F(0°) = (0.95, 0.93, 0.88)$, that is typical of the silver and $p(\theta_h)$ normal distribution with

$\mu = 0$ and $\sigma = 0.6$. It also uses the Torrance and Sparrow geometrical factor (version simplified by Blinn) [13].

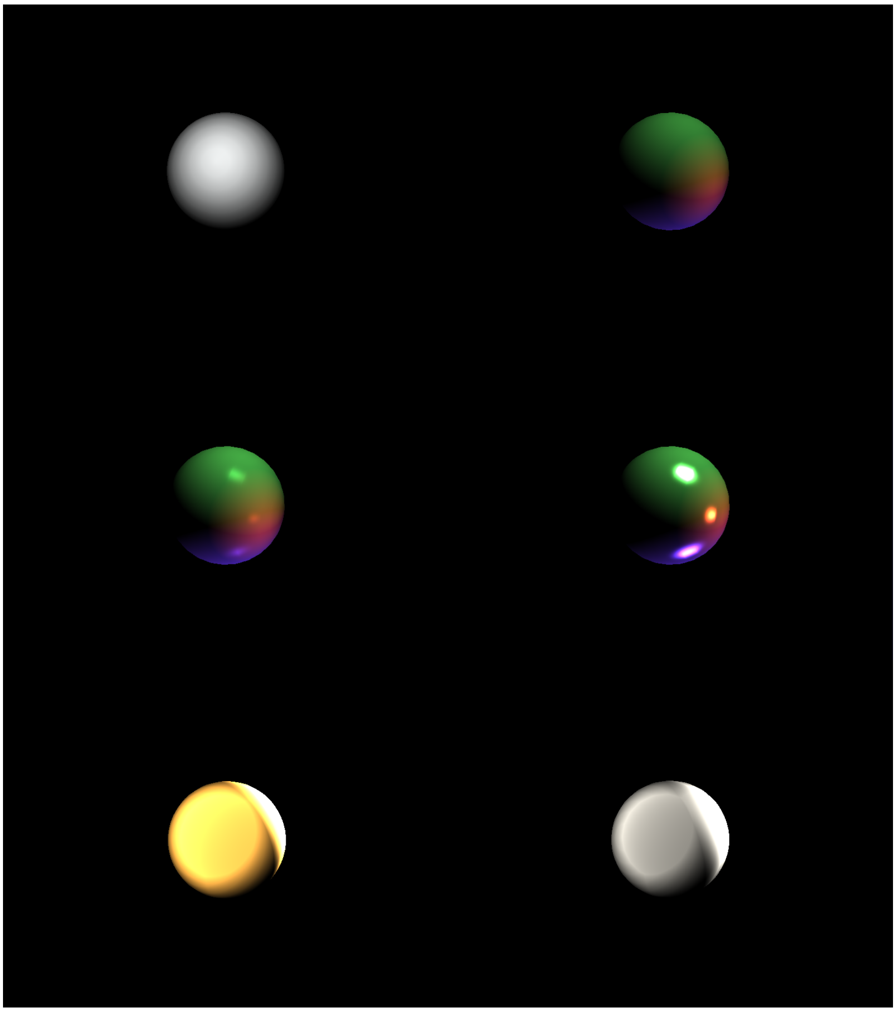

Figure 20: Different BRDFs models

8 BRDF Models

8.1 General classification

BRDF falls into two board groups: Theoretical and Empirical. Theoretical models are trying to simulate light scattering by physical laws. Empirical modes provide a simple formulation sepcifically designed to fit a class of surface type. Since the empirical BRDF modes are simpler, they are more commonly used in real-time rendering.

8.2 A review of BRDF models

Many representations for BRDFs have been proposed in the computer graphics literature. In this section, we introduce four models that are most popular surface reflectance models currently used in real-time rendering.

The Lambertian BRDF

The simplest reflectance to model is Lambertian, with equal scattering in all direction. In other words, the brightness of such a surface to an observer is the same regardless of the observer's angle of view.
In this case, the directional-hemispherical reflectance of the diffuse term is set to a constant value, yielding the following diffuse BRDF term:

$$f_{diff}(l,v) = \frac{\mathbf{c}_{diff}}{\pi}.$$

The Phong BRDF

Phong introduced the first sepcular model to computer graphics in 1975 [2]. Basically, it is an empirical model which obeys neither energy conservation nor reciprocity, but its simplicty has made it very popular use in real-time rendering. Phong BRDF is generally for glossy reflection. The standard representation is

$$f(l,v) = \begin{cases} \frac{\mathbf{c}_{diff}}{\pi} + \frac{\mathbf{c}_{spec}\overline{\cos}^m \alpha_r}{\pi \overline{\cos}^m \theta_i}, & \text{where } \theta_i > 0, \\ 0, & \text{where } \theta_i \leq 0. \end{cases}$$

$\mathbf{c}_{diff}$ is equal to the directional-hemisepherical reflectance of the diffuss term. As a reflectance value, $\mathbf{c}_{diff}$ is restricted to values between 0 and 1, so it can be selected with color-picked interfaces. $\mathbf{c}_{spec}$ is the directional-hemisepherical reflectance of the specular term . θ_i is the angel between the light direction vector and the surface normal. From the equation, it turns out the specular term is unbounded which will cause a very bright area at glancing angle, as it may goes to infinity when θ_i increses to 90°. In order to solve the problem, we can remove the division by $\overline{\cos}\theta_i$. Which is good, because it also removes the condition. Thus, a simpler BRDF is generated:

$$f(l,v) = \frac{\mathbf{c}_{diff}}{\pi} + \frac{\mathbf{c}_{spec}\overline{\cos}^m \alpha_r}{\pi}.$$

This simpilified version of the Phong BRDF is more physically plausible in serveral ways-its reflectance does not go to infinity, it is reciprocal, and it lacks the abrupt cutoff when $\theta_i = 90°$[3]. But it's still not maintain the engergy conservation.

The Blinn BRDF

The Blinn BRDF is also called Blinn-Phong reflection model. It is a modification of the Phong BRDF. Instead of using reflection vector, what Blinn proposed is to compute a vector which is half way in between incomming light l and view vector v. A normalized form of Blinn-Phong BRDF is as follows:

$$f(l, v) = \frac{\mathbf{c}_{diff}}{\pi} + \frac{m+8}{8\pi}\mathbf{c}_{spec}\overline{\cos}^m\theta_h.$$

One advantage of using the half vector h is that the computation of it requires fewer operations than the reflection vector $\mathbf{r}_i$. Therefore, although the resulting exponent value of specular term is different, the overall look of Blinn is very similar to Phong and it's much cheaper to compute.

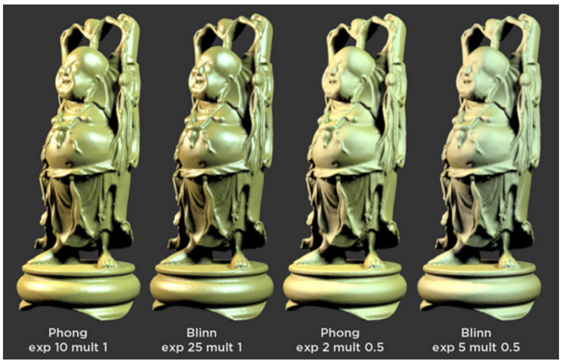

Figure 21: Blinn and Phong models with different exponent value (*Images courtesy of Kevin George*)[4].

Although Blinn BRDF can be seen as an approximation of the phong model, the differences lie between the half vector and reflectance vector. Phong refelction are always round for a flat surface, the Blinn-Phong reflection become elliptical when the surface is viewed from a step angle [6]. The Blinn-Phong BRDF will always tend to produce a elongated specular hightlights. Good examples of using this model would be light reflected in oceans or wet streets.

The Ward BRDF

Althoug Phong and Blinn-Phong BRDFs are simple to compute, some surfaces are not modeled well with this style BRDF. For example, anisotropic surfaces like brushed metal and hair. So Ward developed a mathematical description of the refectance of anisotropic materials. The orignial Ward BRDF consist of two components [7]. The first is the diffuss term $\frac{\mathbf{c}_{diff}}{\pi}$ and the second component is a gaussian anisotropic gloss lobe defined by three paramerters, $\mathbf{c}_{spec}$, α_x and α_y. The mathmatical expression is:

$$f(l, v) = \frac{\mathbf{c}_{diff}}{\pi} + \frac{\mathbf{c}_{spec}}{4\pi\alpha_x\alpha_y\sqrt{\cos\theta_i\cos\theta_0}} e^{-\tan^2\theta_h(\frac{\cos^2\phi_h}{\alpha_x^2}+\frac{\sin^2\phi_h}{\alpha_y^2})}.$$

where $\mathbf{c}_{spec}$ controls the magnitude of the lobe, and α_x and α_y are the standard deviation of the surface slope in the x and y direction, which determine the width of the lobe in the two pricipal directions of anisotropy. ϕ is the azimuth angle of the half vector h projected into the surface plane (definitions of parameters are defined by Ward)[7]. For the case of perfect specular refelection (or called perfect mirror refelection), the exponent need to be 0. The normalization factor $\frac{1}{4\pi\alpha_x\alpha_y}$, guarantees the correct integration of this function on the hemisphere of directions [8]. If $\alpha_x = \alpha_y$ then the model is for isotropic. Figure 22 gives a good example of comparing Ward BRDF with Phong and Blinn-Phong BRDF on rendering anisotropic surface. It is one of the most versatile reflectance funtions as is cheap to evaluate, has direct sampling methods and fits well to measured reflectance data [8].

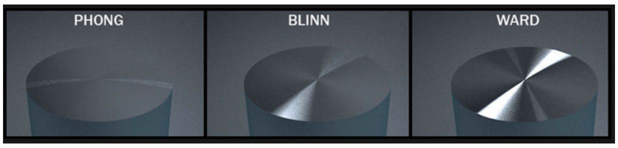

Figure 22: Images of rendered brushed metal with Phong, Blinn-Phong and Ward BRDFS.[9]

The Oren-Nayar BRDF

A physical phenomenon can not modeled by the modifided Phong BRDF is the retro-reflection seen in certain rough surface. Due to the microscale surface roughness, an retro-reflector can reflect light back to its source with a minimum of scattering and gives the rough surface a flat appearance.
The Lamertain model is able to render a smooth matte surface, but not good for a rough matte surface. The Oren-Nayar BRDF is an improvement on the Lambertian for this type of surface. It is also a diffuse-only model and does not focus on rendering specular highlights. A simplified version of this BRDF is given as:

$$f(l, v) = \frac{\mathbf{c}_{diff}}{\pi}(A + B\overline{\cos}\phi\sin(min(\theta_i, \theta_0))\tan(max(\theta_i, \theta_0)))$$

where ϕ is the azimuth angle between the projections of the incoming light direction l and the view direction v. A and B are defined as:

$$A = 1 - 0.5\frac{\sigma^2}{\sigma^2 + 0.33}$$

$$B = 0.45\frac{\sigma^2}{\sigma^2 + 0.09}$$

The roughness σ, is defined as the standard deviation of the angel between the microfacet surface normals and the macroscopic surface normal [3]. σ is ranging from 0 to 1. The dirty material tends to be more retroreflection. This characteristic is controlled by the roughness σ. The larger σ, the material is more retro-reflective. In the case of $\sigma = 0$, surface becomes smooth and the model is equivalent to Lamertain (as $A = 1, B = 0$). Figure 23 shows a rendered images with Oren-Nayar BRDF, corresponding to different roughness of the surface.

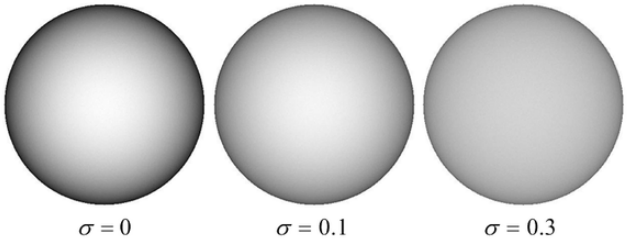

Figure 23: Rendered images of a sphere with different surface roughness by Oren-Nayar BRDF. (images courtesy of Wikipedia)

The Oren-Nayar BRDF is able to explain the view dependence appearance of the matte surface with geometric optics. For this reason, it is physically based model or theoretical model which is good for rendering diffuse reflection. Application of trigonometry transformations can substantially improve the implementation of this BRDF [8] & [10].

8.3 BRDF Normalization

BRDF normalization means, in simple term, that the shading model scales the intensity of the specular hightlight in proportion to it's regular size, such that the total reflected energy remains constant with vary surface smoothness[5]. The value that used for scaling is called a normalization factor, and the resulting BRDF is referred to as a normalized BRDF.

Why we need to normalize a BRDF model? There are many advantages of using normalized shading model over non-normalized ones. One of the most important reason is that we want to maintain the energy conservation in order to simulate a more realistic physical light scattering. We'll use Phong model as examples.

As we mentioned above, a simpler Phong BRDF is:

$$f(l,v) = \frac{\mathbf{c}_{diff}}{\pi} + \frac{\mathbf{c}_{spec}\overline{\cos}^m \alpha_r}{\pi}.$$

With this version, we can calculate the directional-hemispherical reflectance of the specular term. If $\theta_i = 0$, it reaches a maximum value of specluar hightlight, which is $\frac{2\mathbf{c}_{spec}}{m+2}$. To make the $\mathbf{c}_{spec}$ be equal to the maximum directional-hemispherical reflectance, we can scale the specular term by $\frac{m+2}{2}$, which yeild the nomalized Phong BRDF model:

$$f(l,v) = \frac{\mathbf{c}_{diff}}{\pi} + \frac{m+2}{2\pi}\mathbf{c}_{spec}\overline{cos}^m \alpha_r.$$

α_r is a certain angle between the view direction and the perfect specular reflection. The reflected intensity is the cosine at that angle with some exponent m. Phong decided a large m, $m \geq 200$, for a shiny surface and small m for a dull surface [2]. In the Figure below we can see how the function $\overline{\cos}^m \alpha_r$ behaves for different value of m.

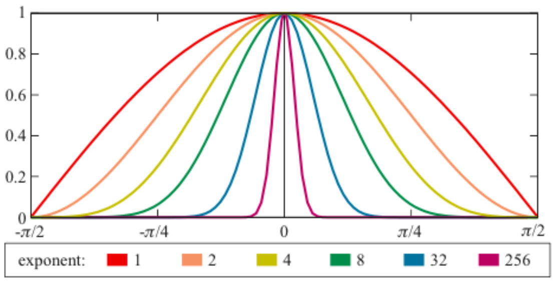

Figure 24: Graph of unnormalized $\overline{\cos}^m \alpha_r$ for various exponent m.[3]

With higher exponent, the width of the curve decreases; however the hight of the curve stays the same. Which means that brightness of the highlight remains the same as the highlight area decrease. There is a loss of energy in the highlight as the integral under the curve is decreased.

Compare this figure to the Figure 25 , which shows normalized curves. By scaling the specular highlight with the normalized factor $\frac{m+2}{2\pi}$, the reflectance value can excessd 1. Therefore, increasing the exponent not only decrease the size of the curve, but also make it higher, which makes the total reflectance constant.

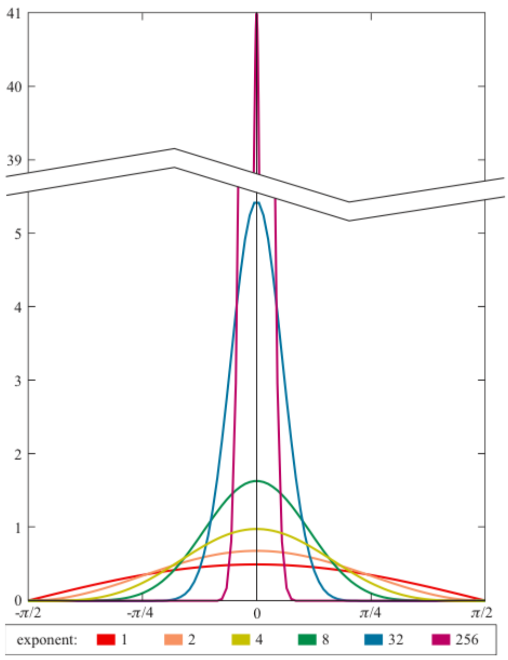

Figure 25: Graph of normalized $\overline{\cos}^m \alpha_r$ for various exponent m.[3]

So with the non-normalized model, changing the exponenent m will both change the amount and distribution of the reflected light. But with the normalized model, the exponent m parameter only controls the surface roughness. Figure 26 shows rendered images of a red plastic sphere with both original and normalied Phong BRDFs.

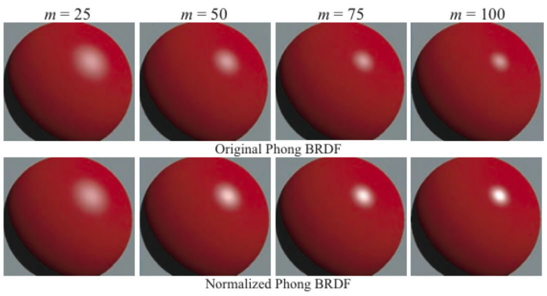

Figure 26: Rendered images of a red plastic sphere with both original and normalied Phong BRDFs.[3]

From the images rendered with normalized Phong BRDF, we can see that by increasing the smoothness of the surface (the exponent m), the highlight grows much bigger as it gets narrower, which is correct behavior - the outgoing light is concentrated in a narrow spot, so it is brighter. But with the non-normalized Phong BRDF, it can be seen that the highlight remains equally bright as it gets narrower, surface appears to be getting less reflective. Although the original Phong BRDF could render the same images as the normalized one, it requires a more complex and carefully adjustment with $\mathbf{c}_{spec}$.

8.4 Conclusion

Many representations for BRDFs have been proposed in the computer graphics literature. Different representations determine different types of the highlights and glossy reflections for a material. Ward BRDF is idear for anisotropic surface such as brushed metal. Blinn and Phong are good for plastics and none metals. Oren-Nayar is for rough surface. And these four types of BRDFs are the most popular surface reflectance model currently used in computer graphics, especially the Phong model. BRDF models can give results that are often "good enough", without significantly slowing down the computation speed.

Montes and Urena[8] gave a detailed review of different BRDF models. They think these reflectance models should exhibit a set of desired properties [12] to make them realistic and reliable at the same time. Some are physically plausible, some are realistic, some are efficient to implement and some are accurate. They also gave a table of brief summary of the properties exhibited by their reviewed BRDFs. But most importantly, models can be "mix and match" together to produce a new analytic BRDF.

Rusinkiewicz[11] stated that the problem with current BRDF representations is that they introduce visible and objectionable artifacts (e.g. the ringing associated with spherical harmonics) when the BRDFs are compressed too much. Therefore, it would be useful to find methods that are free of such visually jarring artifacts, even when relatively few coefficients are kept. He also stated that for future research, there is a need both for more efficient and computationally inexpensive BRDF representations, and for extended (especially spatially-varying) BRDFs.

9 High Dynamic Range Imaging

HDRI is a post processing technique applied to images in order to improve the contrast and the details that can be seen inside the scene.
The problem solved with this technique is that in a scene the dynamic range of the light captured is often limited to eight bit per color channel. Some light sources (for example direct light sources) are usually hundreds to thousands of times brighter than the indirect illumination and eight bit are not enough.
There are different standards for representing images at higher color precision (every format store the information for the color of a pixel in 32 bit):

1. **RGBE**: invented by Ward, it uses floating point values composed by three 8-bit mantissas (one per color channel) and one 8-bit exponent (shared among all the three color channels).

2. **R9G9B9E5**: introduced in Directx10, based on 9-bit mantissas and 5-bit exponent

3. **R11G11B10FLOAT**: introduced in Directx10, it uses floating point values and every channel has 5 or 6 bit mantissas (6-bit for **red** red and **green** and 5-bit for **blue**) and 5-bit exponent.

In photography in order to produce an HDR image different images taken at different exposure levels are needed. The final image will be the result of the blending operation executed using different weights for different pixels. For every pixel of the final photo an algorithm has to recognize which input photo has the correct exposition and assign a higher weight. After that all images are mixed together using the weights. The result of this operation is a photo with the correct exposition in every region.
In computer graphic this technique is used in order to produce realistic effects, for example it is possible to simulate the effect of a brightness light in the scene (this is called bloom effect).
The **bloom** effect is achieved merging two different images:

1. the original image

2. an image created using a bright pass filter and a two-pass blur filter (a two pass filter is more efficient than executing a 2D convolution). In order to improve the performance of the pipeline for using it in realtime rendering, it is possible to use a downsampled image (since it is blurred). Usually the suggested values for downsampling goes from 1/2 to 1/8 of originals width and height.

This technique is illustrated in figure 27 and it is widely used in every scene where are rendered ambient with different light brightness (an example takes from the Unreal Engine 4 is reported in figure 28)

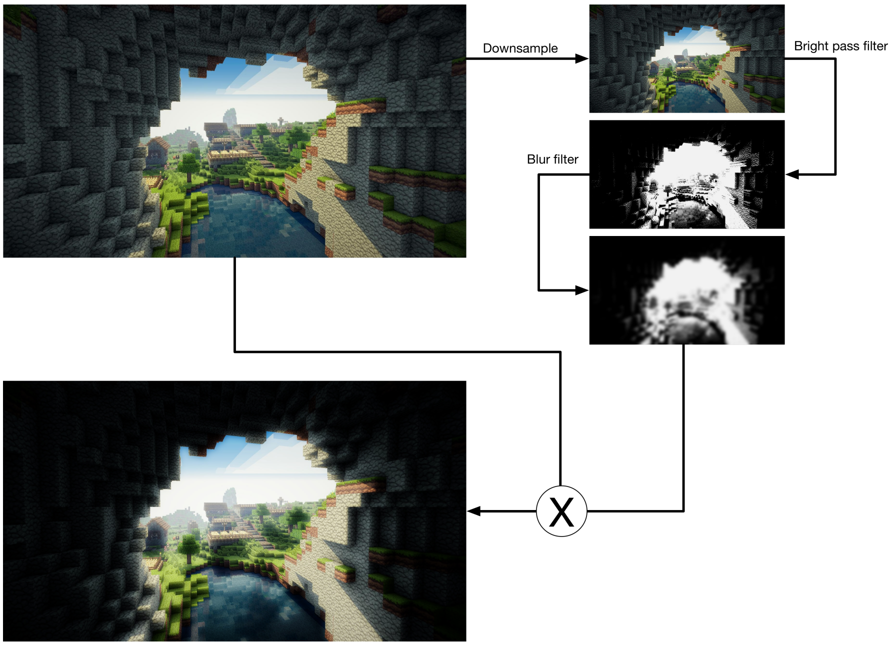

Figure 27: HDR process

Figure 28: HDR bloom effect in Unreal Engine 4 [18]

Appendix A

This is the code:

```
#define MAX_LIGHT 16    // Number of light is limited to 16 in the scene

in vec3 V, N, L[MAX_LIGHT];
uniform vec3 l_diffuse[MAX_LIGHT];
uniform vec3 l_specular[MAX_LIGHT];

uniform int lightNumber;

uniform float s;        // Standard deviation for Oren-Nayar BRDF

uniform int brdfModel;  // The index of the model that you want to use

in vec3 color;          // The color of the objects

out vec4 frag;          // Output color of the pixel

float specFact = 32;    // The m parameter for Phong BRDF

vec3 OrenNayar(vec3 L, vec3 V, vec3 l_diffuse) {            // Oren-Nayar
     BRDF
    float A = 1.0 - 0.5 * pow(s, 2) / (pow(s, 2) + 0.33);
    float B = 0.45 * pow(s, 2) / (pow(s, 2) + 0.09);

    vec3 vproj = V - N * (N * V);
    vec3 lproj = L - N * (N * L);
    float fi = acos(dot(vproj, lproj));

    float vi = acos(dot(N, L));
    float vo = acos(dot(N, V));
    return l_diffuse * 0.8 * (A + B * clamp(cos(fi), 0.0, 1.0) * sin(min(
        vi, vo)) * tan(max(vi,vo)));
}

vec3 Phong(vec3 L, vec3 V, vec3 l_diffuse, vec3 l_specular) {       //
    Phong BRDF
    vec3 R = -reflect(L, N);
    return l_diffuse + l_specular * pow(clamp(dot(R, V), 0.0, 1.0),
        specFact);
}

vec3 PhongNorm(vec3 L, vec3 V, vec3 l_diffuse, vec3 l_specular) {   //
    Normalized Phong BRDF
    vec3 R = -reflect(L, N);
    return l_diffuse + l_specular * (specFact + 2) / 2 * pow(clamp(dot(R,
         V), 0.0, 1.0), specFact);
}

void main(){

    vec3 outColor = vec3(0.0);
```

```
    for(int k=0; k<lightNumber; k++) {  // For every light

        float vi = acos(dot(N, L[k]));

        vec3 c;
        switch(brdfModel) {              // Choose the model based on the
            brdfModel
            case 0: // Phong BRDF
                c = Phong(L[k], V, l_diffuse[k], l_specular[k])/1.5 *
                    clamp(cos(vi), 0.0, 1.0);
                break;
            case 1: // Normalized Phong BRDF
                c = PhongNorm(L[k], V, l_diffuse[k], l_specular[k])/1.5 *
                     clamp(cos(vi), 0.0, 1.0);
                break;
            case 2: // Oren Nayar BRDF
                c = OrenNayar(L[k], V, l_diffuse[k])/1.5 * clamp(cos(vi),
                     0.0, 1.0);
                break;
        }

        // Sum the reflected light of the current light source
        if(c.x >= 0)
            outColor.x += c.x;
        if(c.y>=0)
            outColor.y += c.y;
        if(c.z >= 0)
            outColor.z += c.z;
    }


    frag = vec4( clamp(outColor, 0.0, 1.0), 1.0);
}
```

References

[1] An Inexpensive BRDF Model for Physical Based Rendering *Christoper Schlick*,p. 251

[2] Phong, Bui Tuong, "Illumination for Computer Generated Pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, June 1975. Cited on p. 112, 115, 201, 249, 252, 266.

[3] Tomas A., Eric H., & Naty H.(2002), *Real-Time Rendering.* A.K. Peters Ltd. Natick, Massachusetts.

[4] Kevin George, *RenderMan Portfolio*, 2014. Available at: `http://www.sfdm.scad.edu/faculty/mkesson/vsfx755/wip/best/spring2012/kevin_george/final/index.html`.

[5] Christian, *The Blinn-Phong Normalization Zoo*, 2011, Available at: `http://www.thetenthplanet.de/archives/255`.

[6] WikiPedia, *Blinn-Phong Shading Model*, `http://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_shading_model`.

[7] WARD, G. J. 1992. *Measuring and modeling anisotropic reflection.* In Computer Graphics (Proceedings of SIGGRAPH 92), vol. 26, 265–272.

[8] Montes R., & Urena C., *An Overview of BRDF Models*, LSI-2012-001, 2012. University of Granada, Spain

[9] *Anisotropic shader tutorial* `http://www.evermotion.org/tutorials/show/7875/anisotropic-shader-tutorial-using-vray-1-5-final-sp-1`.

[10] Michael Oren and Shree K. Nayar., *"Generalization of Lambert's reflectance model"*, ACM SIGGRAPH 1994 Proceedings, ISBN 0-89791-667-0

[11] Szymon Rusinkiewicz *A Survey of BRDF Representation for Computer Graphics*, 1997, Preston `http://www.cs.princeton.edu/~smr/cs348c-97/surveypaper.html`

[12] Shirley P *Monte carlo methods for rendering.*, In ACM siggraph '96 Course Notes - Global Illumination in Architecture and Entertainment. 1996, pp. 1-26.5

[13] James F. Blinn. 1977. Models of light reflection for computer synthesized pictures. SIGGRAPH Comput. Graph. 11, 2 (July 1977)

[14] Michael Ashikmin, Simon Premože, and Peter Shirley. 2000. A microfacet-based BRDF generator. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques

[15] Andreas Schilling. Towards Real-Time Photorealistic Rendering: Challenges and Solutions (1997)

[16] K. E. TORRANCE and E. M. SPARROW. Theory for Off-Specular Reflection From Roughened Surfaces.

[17] Jan Kautz, Hans-Peter Seidel.Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs.

[18] Unreal Engine 4: `http://www.unrealengine.com`