Open CASCADE Technology
7.5.0.beta

PBR math (rasterization)

October 9, 2020

Contents

0.1 Preface

Empirical illumination models like **Phong reflection model** have been used in real-time graphics for a long time due to their simplicity, convincing look and affordable performance. Before programmable pipeline has been introduced, graphics cards implemented Gouraud shading as part of fixed-function Transformation & Lighting (T&L) hardware blocks. Nowadays, however, numerous trade-offs of this simplicity (like lighting partially baked into object material properties and others) pushed developers to **Physically-Based Rendering** (**PBR**) illumination models.

PBR models try to fit surface shading formulas into constrains of physical laws of light propagation / absorption / reflection - hence, called "physically-based". There are two main categories of PBR illumination:

1. Non-real-time renderer (cinematic).
2. Real-time renderer.

The main objective of cinematic renderer is uncompromised quality, so that it relies on ray-tracing (path-tracing) rendering pipeline. Although performance of current graphics hardware does not make it possible using computationally-intensive path-tracing renderer in real-time graphics, it can be used in interactive fashion.

"Physically-based" does not necessarily mean physically-correct/precise. The main objective of real-time PBR renderer is to be fast enough even on low-end graphics hardware. So that in contrast, it hardly relies on rasterization rendering pipeline, various approximations and tricks making it applicable in real-time, while looking good enough and preserving some physical properties.

OCCT 3D Viewer provides both kinds of PBR renderers, and although they share some details in common, this article is devoted to real-time PBR metallic-roughness illumination model. This article describes the math underneath PBR shading in OCCT 3D Viewer and its GLSL programs. However, this article does not clarifies related high-level APIs nor PBR material creation pipelines, as this is another topic.

0.2 Notation

n	normal (on surface)	$\|n\| = 1$
v	view direction	$\|v\| = 1$
l	light	$\|l\| = 1$
$h = \frac{v+l}{\|v+l\|}$	half vector	
m	metallic factor	$[0,1]$
r	roughness factor	$[0,1]$
IOR	index of refraction	$[1,3]$
c	albedo color	(R,G,B)

$\cos\theta_l = (n \cdot l)$

$\cos\theta_v = (n \cdot v)$

$\cos\theta_h = (n \cdot h)$

$\cos\theta_{vh} = (v \cdot h)$

0.3 Illumination model

The main goal of illumination model is to calculate outgoing light radiance L_o along the certain direction. The starting point of calculation might be the view direction v aimed from point on surface (or in more general case just in space) to viewer position. Considering the point on opaque surface with normal n the main equation of illumination can be defined as:

$$L_o = \int_H f(v,l) L_i(l) \cos\theta_l \, \mathrm{d}l$$

Where $L_i(l)$ is light radiance coming from l direction, $f(v,l)$ is **Bidirectional Reflectance Distribution Function** (**BRDF**) and H is hemisphere which is oriented regarding to the surface normal n. Opaqueness of the surface mentioned earlier is important because in that case hemisphere is enough. More general model will require to consider directions all around a whole sphere and is not observed in this paper. $\cos\theta_l$ factor appearing is caused by affection of surface area and light direction mutual orientation to the amount of radiance coming to this area. This is mainly due to geometric laws. The rest part of integral is the key of the whole illumination model. BRDF defines it's complexity and optical properties of material. It has to model all light and material interactions and also has to satisfy some following criteria in order to be physical correct [Duvenhage13]:

- Positivity: $f(v,l) \geq 0$
- Helmholtz reciprocity: $f(v,l) = f(l,v)$ (follows from 2$^{\text{nd}}$ Law of Thermodynamics)
- Energy conservation: $\forall v \int_H f(v,l) \cos\theta_l \, \mathrm{d}l = 1$ (in order not to reflect more light than came)

It is worth to be mentioned that $f(v,l)$ depends on n also but it is omitted to simplify notation. BRDF is usually split into two parts:

$$f(v,l) = f_d(v,l) + f_s(v,l)$$

Where $f_s(v,l)$ (specular BRDF) models reflection light interaction on surface and $f_d(v,l)$ (diffuse BRDF) models other processes happening depth in material (subsurface scattering for example). So that illumination equation might be rewritten as:

$$L_o = \int_H (f_d(v,l) + f_s(v,l)) L_i(l) \cos\theta_l \, \mathrm{d}l$$

PBR theory is based on **Cook-Torrance specular BRDF** [Cook81]. It imagines surface as set of perfectly reflected micro faces distributed on area in different ways which is pretty good model approximation of real world materials. If this area is small enough not to be able to recognize separate micro surfaces the results becomes a sort of averaging or mixing of every micro plane illumination contribution. In that level it allows to work with micro faces in statistical manner manipulating only probabilities distributions of micro surfaces parameters such as normals, height, pattern, orientation etc. In computer graphics pixels are units of images and it usually covers a relatively large areas of surfaces so that micro planes can be considered to be unrecognizable. Going back to the BRDF the Cook-Torrance approach has the following expression:

$$f_s(v,l) = \frac{DGF}{4\cos\theta_l \cos\theta_v}$$

Three parts presented in nominator have its own meaning but can have different implementation with various levels of complexity and physical accuracy. In that paper only one certain implementation is used. The D component is responsible for **micro faces normals distribution**. It is the main instrument that controls reflection's shape and strength according to **roughness** r parameter. The implementation with good visual results is **Trowbridge-Reitz GGX** approach used in Disney's RenderMan and Unreal Engine [Karis13]:

$$D = \frac{\alpha^2}{\pi(\cos^2\theta_h(\alpha^2-1)+1)^2}$$

Where $\alpha = r^2$. This square in needed only for smoother roughness parameter control. Without it the visual appearance of surface becomes rough too quickly during the parameter's increasing.

The second G component is called **geometric shadowing** or attenuation factor. The point is that micro surfaces form kind of terrain and can cast shadows over each other especially on extreme viewing angles [Heitz14]. **Smith-Schlick model** [Heitz14], [Schlick94] has been chosen as implementation:

$$G = \frac{\cos\theta_l \cos\theta_v}{(\cos\theta_l(1-k)+k)(\cos\theta_v(1-k)+k)}$$

Where $k = \frac{\alpha}{2}$, which means $k = \frac{r^2}{2}$ in terms of this paper. But G depends on many factors so that it's approximations has float nature and can be modified a little bit in some cases in order to get more pleasant visual results. One of this modification will be described later in following chapters.

The last component F shows **how much light is reflected from surface** and is called **Fresnel's factor**. The rest amount of radiance might be absorbed or refracted by material. The most accurate expression of it is pretty complicate for calculation so that there is a variety of approximations. The good one with less computation efforts is **Schlick's implementation** [Schlick94]:

$$F = F_0 + (1-F_0)(1-\cos\theta_{vh})^5$$

Here F_0 is material's response coefficient at normal incidence (zero angle). Fresnel's factor has to be calculated differently for metals and dielectric/non-metals, but PBR theory tries to come up with universal formula for all types of material. In order to do that it is needed to be noticed that Schlick's approximation is applicable only to non-conductors and in that case $F_0 = F_{dielectric} = \left(\frac{1-IOR}{1+IOR}\right)^2$. **Index of Refraction** IOR shows the proportion between light speed in vacuum (or even in air) and in material. The reference value of IOR for plastic is **1.5**, and this value can be considered as default for all unknown dielectrics. In practice this parameter controls reflectance ability of material. Also it should be remembered that this approximation produces poor results with large IOR values so that it is recommended to be kept in range of $[1,3]$ in order to get plausible Fresnel's factor [Lazanyi05], [Lagarde13]. This formula might be further propagated onto metals by using F_0 measured specifically for certain metal. It can be considered as some kind of a 'color' of metal and can be stored as albedo parameter c. And the final step of defining Fresnel's factor formula is mixing all this F_0 using metallic parameter m (**metalness**):

$$F_0 = F_{dielectric}(1-m) + cm$$

For pure dielectrics with $m = 0$ exactly Schlick's approximation will be used. For pure metals with $m = 1$ it will be a little inaccurate but the same formula with measured F_0 values. Everything else for $m \in (0,1)$ is not physically correct and it is recommended to keep m exactly 1 or 0. Intermediate values may represent mixed areas for smooth transition between materials - like partially rusted metal (rust is mostly dielectric). Also it might be useful when parameters are read from textures with filtering and smoothing.

BRDF described above has one important trait making computations easier called **isotropy**. Isotropy in this case means independence from rotation about normal resulting from supposition of uniform micro faces distribution at any direction along a surface. It allows to simplify random samples generation during Monte-Carlo integrals calculation and reduce dimensions of some lookup tables, which will be discussed in following chapters. Of course, isotropic materials form only subset of all real world's materials, but this subset covers majority of cases. There are special models considering special anisotropic traits of surfaces like a grinding of metal or other with dependency on rotation about normal; these models require special calculation tricks and additional parameters and are out of scope of this paper.

The only thing left to do is to define $f_d(v,l)$. This part is responsible for processes happening in depth of material. First of all the amount of input light radiance participating in these processes is needed to be calculated. And it exactly can be realized from already known Fresnel's factor F showing amount of reflected light but in negative term in this case in order to get the radiance left after reflection:

$$1 - F$$

This part of ingoing light is assumed to be refracted in depth of surface and variety of events may happen there. A sequence of absorptions, reflections and reemissions more or less leads to light's subsurface scattering. Some part of this scattered light can go back outside but in modified form and in pretty unpredictable directions and positions. For opaque materials this part is noticeable and forms it's own color. If subsurface's paths of light are small enough and points of output are distributed locally around the input point it's possible to work in statistical way similar to the micro faces. This assumption covers a big amount of real world opaque materials. Other materials like skin, milk etc. with noticeable effect of subsurface scattering usually presented in form of partial translucency and some kind of self emission have more widely distributed output points and require more accurate and complicate ways of modeling with maybe some theory and techniques from volumetric rendering. The simple but visually enough assuming for statistically driven type of materials is just the same radiance for any direction. It results to **Lambertian's BRDF**:

$$\frac{c}{\pi}$$

Where π is normalization coefficient in order to meet BRDF's criteria and c is material's own color formed by adventures of light under surface. There is one detail about light interaction bringing some physicality to the model, and that is an absence of this diffuse component in metals. Metals reflect main part of light and the rest of it is absorbed being transformed into other form (mostly heat). That is the main visual difference between metallic and non-metallic materials realizing of which brings model to higher level of quality in compare to older non-physical models.

So that all parts described above can be combined into united diffuse BRDF:

$$f_d(v,l) = (1-F)(1-m)\frac{c}{\pi}$$

m is recommended to be exactly 1 or 0 but all values between can represent transition areas, as mentioned before.

In this chapter one possible implementation of illumination model reflecting main PBR principles has been defined. The next step is using of it in practice.

0.4 Practical application

It's time to apply deduced illumination model in practice. And the first step of it is separation of **direction based light sources** from illumination integral. Directional nature of such light sources means possibility to calculate it's influence to point of surface using only one direction and its intensity. Usually sources of this type do not have physical size and are represented only by position in space (for point or spot lights) or by direction itself (direction light imagined to be too far point sources like sun). This is just a kind of abstraction, while real world light emitters have noticeably sizes. But sources with realistic form and size cannot be presented in discrete term and require continuous integrals calculations or special approximations in order to be accurately injected to the model. In most cases direct based light sources in form of emitting points in space or just certain directions are good approximations and are enough for beginning. Having finite discrete amount of it in scene and considering only single direction from every of these lights, the integral is transformed just to the sum:

$$L_{direct} = \sum_{j=1}^{M} f(v,l_j) L_i^{direct}(l_j) \cos\theta_{l_j}$$

Where M is a number of sources, l_j is a direction and L_i^{direct} is an intensity related to this direction. $direct$ label means that illumination has been computed directly from sources. The BRDF can be used directly without any calculation problems. The only exception might be k in G factor - it is recommended to use $k = (r+1)^2/8$ in order to get more pleasant results [Karis13] (that is modification mentioned in previous chapter). And actually it is enough

to finally see something. There will be correct visualization with assumption of complete dark environment and absence of other points influence. It is called **local illumination**. Based on this name there is also a global or **indirect illumination** and that is the rest of integral:

$$L_{indirect} = \int_H f(v,l) L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l$$

It includes influence of light reflected or scattered from other points and environment's contribution. It's impossible to achieve photorealistic results without this component, but is is also very difficult to compute. While the cross point light interaction cannot be calculated in a simple way (especially in real time rendering), the environment illumination has some options to be realized via precomputational work before visualization. But right now lets summarize the practical application of illumination model. At this moment the output radiance is represented as:

$$L_o = L_{direct} + L_{indirect}$$

Where L_{direct} is direction based light sources contribution which can be directly computed just applying bare formulas. It is enough to get some results in terms of local illumination but without $L_{indirect}$ component image will not be looked lifelike. $L_{indirect}$ is not trivial thing for calculation and that is stumbling block for real time rendering applications. But it can be relatively easy implemented in case of environment illumination via some precomputational work about which will be told in details in following chapters.

0.5 Image based lighting

The next goal after L_{direct} calculation is to find $L_{indirect}$. And it would be easier if $L_i^{indirect}(l)$ was known for every l. That is the main assumption of **image based lightning** (**IBL**). In practice, it can be achieved using environment image map, which is a special image representing illumination from all possible directions. This image might be a photo capturing a real world environment (spherical 360 degrees panoramas) or generated image baking the 3D scene itself, including in that case reflections of other objects. Environment image might be packed in different ways - **cube maps** and equirectangular maps are the most commonly used. Anyway, it allows $L_i^{indirect}(l)$ to be defined for every l and its practical implementation in form of images gives name for this approach. Lets back to indirect illumination integral:

$$L_{indirect} = \int_H f(v,l) L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l$$

Substituting the BRDF by its expression allows to split indirect illumination into diffuse and specular components:

$$L_{indirect} = \int_H f_d(v,l) L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l + \int_H f_s(v,l) L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l =$$

$$= (1-m)\frac{c}{\pi} \int_H (1-F) L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l + \frac{1}{4} \int_H \frac{DFG}{\cos\theta_l \cos\theta_v} L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l$$

This splitting seems not to lead to simplicity of calculation but these two parts will be computed in slightly different ways in future. Lets write down this separately:

$$L_{indirect}^d = (1-m)\frac{c}{\pi} \int_H (1-F) L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l$$

$$L_{indirect}^s = \frac{1}{4} \int_H \frac{DFG}{\cos\theta_v \cos\theta_l} L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l$$

Next transformations of these expressions require understanding of numerical way to find hemisphere integral and also its performance optimization techniques. And that the topic of the next chapter.

0.6 Monte-Carlo numeric integration

Monte-Carlo is one of numeric methods to **find integral**. It is based on idea of mathematical expectation calculation. In one dimensional case if $f(x)$ is a function with parameter distributed according to probability density $p(x)$ the mathematical expectation of it can be found using following expression:

$$E = \int_{-\infty}^{\infty} f(x)p(x)\,\mathrm{d}x$$

Also this expectation can be approximated in statistical term using certain sequence of random variable x:

$$E \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

It can be used in general definite integrals calculations. Just valid $p(x)$ defined on $[a,b]$ range and sequence x_i generated according to it are needed for that:

$$\int_a^b f(x)\,\mathrm{d}x = \int_a^b \frac{f(x)}{p(x)}p(x)\,\mathrm{d}x = \int_{-\infty}^{\infty} \frac{f(x)}{p(x)}p(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

Where $f(x)$ is considered to be equal to zero outside of $[a,b]$ range. This is also true for functions on sphere or hemisphere:

$$\int_{H|S} f(l)\,\mathrm{d}l \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(l_i)}{p(l_i)}$$

The main questions are choosing $p(l)$ and generating samples l_i. The one of the simple ways is uniform distribution along sphere or hemisphere. Lets realize that on sphere for example. There are 4π possible directions in terms of sphere's areas and steradians (direction can be presented as dot on a unit sphere):

$$\int_S 1\,\mathrm{d}l = 4\pi$$

Where S is the unit sphere. In order to be uniform $p(l)$ must be constant and satisfy normalization criteria:

$$\int_S p(l)\,\mathrm{d}l = 1$$

So that $p(l) = \frac{1}{4\pi}$. Usually direction l is parametrized by spherical coordinates $\phi \in [0,2\pi]$ and $\theta \in [0,\pi]$ boiling down to the 2D samples generation. But in these terms joint $p(\theta,\phi)$ will be looked slightly different due to variables transition. l is defined in regular Cartesian coordinates $l = (x,y,z)$ with $\|l\| = 1$. The spherical coordinates transform looks like:

$$x = r\sin\theta\cos\phi,\ y = r\sin\theta\sin\phi,\ z = r\cos\theta$$

Where $r = 1$. In order to express probability density using new variables it is needed to multiply this density by Jacobian of transform:

$$p(\theta,\phi) = p(l)|J_T|$$

Where:

$$|J_T| = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial \phi} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \phi} \\ \frac{\partial z}{\partial r} & \frac{\partial z}{\partial \theta} & \frac{\partial z}{\partial \phi} \end{vmatrix} = r^2 \sin\theta$$

So that joint probability density in new variables looks like:

$$p(\theta,\phi) = \frac{\sin\theta}{4\pi}$$

This variable transfer rule of **Probability Density Function** (**PDF**) will be useful in following chapters, when integral calculation optimization techniques will be being told. Having $p(\theta,\phi)$ the partial single dimensional probability densities are able to be found:

$$p(\phi) = \int_0^\pi p(\theta,\phi)\,\mathrm{d}\theta = \frac{1}{4\pi}\int_0^\pi \sin\theta\,\mathrm{d}\theta = \frac{1}{2\pi}$$

$$p(\theta) = \int_0^{2\pi} p(\theta,\phi)\,\mathrm{d}\phi = \frac{\sin\theta}{4\pi}\int_0^{2\pi} 1\,\mathrm{d}\phi = \frac{\sin\theta}{2}$$

The final step is sequence generation itself. In order to be able to generate values with arbitrary distributions it is helpful to start from uniform numbers in range of $[0,1]$. And that can be done via any known true- and pseudo-random generators. Even simple $\frac{1}{i}$ sequence is appropriate for beginning but it can be not so efficient in terms of computations convergence. There are specially designed series for the last reason and it will be tackled in chapter about optimizations. The ϕ variable is noticed to be uniformly distributed so that it can be directly generated without any additional manipulations. Just range $[0,1]$ is needed to be mapped to range $[0,2\pi]$. For any other variables including θ the inverse transform sampling approach can be applied. First of all **cumulative distribution function** (**CDF**) is needed to be found. It is probability of random value to be less than argument of this functions by definition. For continues distributions it can be expressed in following form:

$$F(x) = \int_{-\infty}^{x} p(x')\,\mathrm{d}x'$$

Lets find CDF for θ:

$$F(\theta) = \int_{-\infty}^{\theta} p(\theta')\,\mathrm{d}\theta' = \int_0^{\theta} \frac{\sin\theta'}{2}\,\mathrm{d}\theta' = \frac{1-\cos\theta}{2}$$

The CDF maps θ values from range of $[0,\pi]$ to probability in range of $[0,1]$. The next step is to find inverse cumulative function which can be not so trivial sometimes but pretty obvious in current case:

$$F^{-1}(u) = \arccos(1-2u)$$

If substitute uniform distributed in range $[0,1]$ values u as argument of this function the values with origin probability density will appear. In other words:

$$\theta = \arccos(1-2u),\ u \in [0,1],\ p(u) = 1 \Rightarrow p(\theta) = \frac{\sin\theta}{2}$$

That is the key of this random values generation technique. All steps described above can be also done for hemisphere:

$$p(l) = \frac{1}{2\pi}$$

$$p(\theta,\phi) = \frac{\sin\theta}{2\pi}$$

$$p(\phi) = \int_0^{\frac{\pi}{2}} p(\theta,\phi)\,\mathrm{d}\theta = \frac{1}{2\pi}\int_0^{\frac{\pi}{2}} \sin\theta\,\mathrm{d}\theta = \frac{1}{2\pi}$$

$$p(\theta) = \int_0^{2\pi} p(\theta,\phi)\,\mathrm{d}\phi = \frac{\sin\theta}{2\pi}\int_0^{2\pi} 1\,\mathrm{d}\phi = \sin\theta$$

$$\theta = \arccos(1-u)$$

Mote-Carlo integration cannot guarantee exact estimation of convergence speed with using random generated samples. There is only probability estimation of it. But this algorithm is pretty universal and relatively simple to be applied to almost any function using at least uniform distributed points. Moreover special $p(l)$ can be chosen and special pseudo-random sequences can be designed in order to speed up convergence for some functions (following chapter talk about that in details). That is why this method is widely used in computer graphics and demonstrates good results. Also another one advantage is worth to be mentioned - possibility to iteratively accumulate computations and present intermediate results during rendering which is used in some ray tracing applications.

0.7 Split sum

Lets go back to the image based lighting and the figure of specular component. As was defined before that is hemisphere integral with following expression:

$$L^s_{indirect} = \frac{1}{4}\int_H \frac{DFG}{\cos\theta_v\cos\theta_l} L_i^{indirect}(l)\cos\theta_l\,\mathrm{d}l$$

The Monte-Carlo integration algorithm can be directly applied:

$$L^s_{indirect} = \int_H f_s(v,l)L_i^{indirect}(l)\cos\theta_l\,\mathrm{d}l \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f_s(v,l_i)L_i^{indirect}(l_i)\cos\theta_{l_i}}{p(v,l_i)}$$

$p(v,l_i)$ depends on v and implicitly on r in order to be completely general. Optimization strategies use different samples distributions for different view direction orientations and roughness values. Anyway even with all optimization techniques this algorithm continues to require too much calculations. Good visual results require noticeable number of samples and using this approach for every point in real time rendering becomes unrealistic. The way to avoid these enormous calculations is doing them beforehand somehow. The first trick on the way to this is split the sum separating environment light component [Karis13]:

$$L^s_{indirect} \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f_s(v,l_i)L_i^{indirect}(l_i)\cos\theta_{l_i}}{p(v,l_i)} \approx \left(\frac{1}{N}\sum_{i=1}^{N}L_i^{indirect}(l_i)\right)\left(\frac{1}{N}\sum_{i=1}^{N}\frac{f_s(v,l_i)\cos\theta_{l_i}}{p(v,l_i)}\right)$$

Where the second brackets represent approximation of integral so that the expression can be rewritten as:

$$L^s_{indirect} \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f_s(v,l_i)L_i^{indirect}(l_i)\cos\theta_{l_i}}{p(v,l_i)} \approx \left(\frac{1}{N}\sum_{i=1}^{N}L_i^{indirect}(l_i)\right)\int_H f_s(v,l)\cos\theta_l\,\mathrm{d}l$$

This integral is exact $L^s_{indirect}$ in condition when $L_i^{indirect}(l) = 1$ what just means white uniform environment. The sum before it is kind of averaged environment illumination. The main accomplishment after all this manipulations is possibility to calculate light and BRDF components separately. The sum with $L_i^{indirect}(l_i)$ can be computed beforehand for every normal direction and stored to image called specular map but with some valuable details.

The problem is that l_i samples must be generated according to $p(v,l_i)$ distribution depended on v and r as was mentioned earlier. Variation of normal is not enough in that case and these variables are needed to be considered too. The ways to resolve it are topic of one of the following chapters and now understanding the fact that at least this part can be precomputed before rendering is enough for now. And it is important not to miss out that there is no more BRDF influence in this sum and only $p(v,l)$ can affect in this case. That is why it is so important to strict to PDF during samples generation and that is why $p(v,l)$ must be correlated with BRDF somehow in this approximation approach with splitting. For example completely mirroring materials with $r = 0$ will not be looked as expected if just uniform distribution is used because such surfaces have only one possible direction from which light can be reflected along view direction in compare with N absolutely scattered in case of uniform or many other distributions.

The rest part also can be saved to image. Lets unroll its expression:

$$\int\limits_H f_s(v,l)\cos\theta_l \,\mathrm{d}l = \int\limits_H \frac{DGF}{4\cos\theta_v\cos\theta_l}\cos\theta_l \,\mathrm{d}l$$

This integral is not actually a scalar. That is RGB value due to only F factor and even more only to F_0. In order to simplify future computations F_0 is needed to be moved out of integral. Substitution of Schlick's approximation helps to achieve it:

$$F = F_0 + (1-F_0)(1-\cos\theta_{vh})^5 = F_0(1-(1-\cos\theta_{vh})^5) + (1-\cos\theta_{vh})^5$$

$$\int\limits_H \frac{DGF}{\cos\theta_v\cos\theta_l}\cos\theta_l \,\mathrm{d}l = F_0\int\limits_H \frac{DG}{4\cos\theta_v\cos\theta_l}(1-(1-\cos\theta_{vh})^5)\cos\theta_l \,\mathrm{d}l + \int\limits_H \frac{DG}{4\cos\theta_v\cos\theta_l}(1-\cos\theta_{vh})^5\cos\theta_l \,\mathrm{d}l$$

This form may not look easier, but it has several advantages. The first one is independence from globally defined $L_i^{indirect}(l)$, so that normal orientation does not matter and can be set in any handful way for calculations (Z axis for example). The second one results from isotropic illumination system allowing ϕ component of view vector to be set arbitrarily (0 for example) and $\cos\theta_v$ will be enough to define view direction. And the third one is scalar nature of integrals so that only two precomputed numbers are needed to find BRDF part of $L_{indirect}^s$. Considering dependency of these integrals from $\cos\theta_v$ and r both of it can be precomputed and stored to 2D look-up image variating these two parameters in range $[0,1]$ with two channels consisting of scale and bias for F_0.

Current result for $L_{indirect}^s$ is computing it using 2D image for BRDF part and omnidirectional image for environment illumination. There were a lot of words about Monte-Carlo optimizations techniques and about PDF choice which is important not only in terms of numeric integration but in terms of visual results correctness. It's time to talk about that.

0.8 Importance sampling

Current goal is to speed up Monte-Carlo integration of Cook-Torrance like integrals with following expression:

$$\int\limits_H \frac{DG}{4\cos\theta_v\cos\theta_l} g(v,l)\cos\theta_l \,\mathrm{d}l$$

Where $g(v,l)$ is just arbitrary function representing Fresnel's factor itself or its components. In order to increase convergence the samples with larger contribution (or in other words with larger function's values) have to appear more frequently than others proportionally to its contribution. So that less significant summand with less influence to result will be considered rarely and in opposite way parts brining noticeable changes to the sum will be taken often. That is the main idea of **importance sampling technique**. $p(l)$ has to represent significance of sample in terms of integrated function via probability somehow. And it will be like that if PDF is already part of original function because in that case probability density directly affects to contribution forming. Separating this distribution component is one possible and effective way to realize importance sampling strategy. In integral presented above PDF part already exists and that is D component. But it is distribution of micro faces normals or ideally reflection direction or h in other word and not light directions distribution which is needed in fact. Anyway that is good starting point and lets generate h vectors first. D has the following expression:

$$D = \frac{\alpha^2}{\pi(\cos^2\theta_h(\alpha^2-1)+1)^2}$$

Frankly speaking $D(h)$ is called normal distribution but cannot be directly used as hemisphere distribution. Originally it is statistical factor used to define total area of micro faces $\mathrm{d}A_h$ whose normals lie withing given infinitesimal solid angle $\mathrm{d}h$ centered on h direction using the original small enough area of macro surface $\mathrm{d}A$ [Walter07]:

$$dA_h = D(h)\,\mathrm{d}h\,\mathrm{d}A$$

First of all this factor must be positive:

$$D(h) \geq 0$$

But the total area of micro faces landscape is at least equal to origin surface but even bigger in general:

$$1 \leq \int_H D(h)\,\mathrm{d}h$$

This trait does not allow to use D as hemisphere distribution. But it can be fixed with following feature:

$$\forall v \int_H D(h)(v\cdot h)\,\mathrm{d}h = (v\cdot n)$$

Which means that total area of micro faces projected to any direction must be the same as projected area of origin macro surface. It is pretty tricky trait in D definition but it leads to interesting results in condition when $v = n$:

$$\int_H D(h)\cos\theta_h\,\mathrm{d}h = 1$$

So that $\cos\theta_h$ coefficient normalizes normal distribution in terms of hemisphere and allows to use it as distribution. Finally PDF of half vectors can be wrote:

$$p(\theta_h,\phi_h) = D\cos\theta_h\sin\theta_h = \frac{\alpha^2\cos\theta_h\sin\theta_h}{\pi(\cos^2\theta_h(\alpha^2-1)+1)^2}$$

$\sin\theta_h$ results from spherical coordinate system transfer which was described in Monte-Carlo integration chapter. Lets strict to samples generation procedure and find partial probability densities:

$$p(\phi_h) = \int_0^{\frac{\pi}{2}} p(\theta_h,\phi_h)\,\mathrm{d}\theta_h = \int_0^{\frac{\pi}{2}} \frac{\alpha^2\cos\theta_h\sin\theta_h}{\pi(\cos^2\theta_h(\alpha^2-1)+1)^2}\,\mathrm{d}\theta = \frac{1}{2\pi}$$

$$p(\theta_h) = \int_0^{2\pi} p(\theta_h,\phi_h)\,\mathrm{d}\phi_h = \int_0^{2\pi} \frac{\alpha^2\cos\theta_h\sin\theta_h}{\pi(\cos^2\theta_h(\alpha^2-1)+1)^2}\,\mathrm{d}\phi = \frac{2\alpha^2\cos\theta_h\sin\theta_h}{(\cos^2\theta_h(\alpha^2-1)+1)^2}$$

$p(\phi_h)$ is unnecessary to be calculated analytically. The fact of independency from ϕ is enough to figure out that this coordinate is uniformly distributed. Anyway the $F(\theta_h)$ is next step [Cao15]:

$$F(\theta_h) = \int_0^{\theta_h} \frac{2\alpha^2\cos\theta'_h\sin\theta'_h}{(\cos^2\theta'_h(\alpha^2-1)+1)^2}\,\mathrm{d}\theta'_h = \int_{\theta_h}^{0} \frac{2\alpha^2}{(\cos^2\theta'_h(\alpha^2-1)+1)^2}\,\mathrm{d}(\cos^2\theta'_h) = \frac{\alpha^2}{\alpha^2-1}\int_0^{\theta_h} \mathrm{d}\frac{1}{\cos^2\theta'_h(\alpha^2-1)+1} =$$

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

In order to apply inverse transform sampling the $F^{-1}(u)$ is needed to be found:

$$F^{-1}(u) = \theta_h = \arccos\sqrt{\frac{1-u}{u(\alpha^2-1)+1}}$$

So that there is no more obstacles to generate h. But the main goal was l direction. In order to get it the view vector v has to be reflected related to already found h:

$$l = 2(v\cdot h)h - v$$

That is practical side of light direction generation. But the theoretical one is needed to be resolved to calculate sum. It is time to find $p(l)$ using known $p(h)$. First of all the fact that l is just transformed h is needed to be understood. In that way the light direction's PDF has following expression:

$$p(l) = p(h)|J_T|$$

Where $|J_T|$ is Jacobian of reflection transformation. Lets find it. Right now n is axis from which θ spherical coordinate is encountered. The first step is setting v as starting point of θ instead of n. This is linear transform so that $|J_T| = 1$. Next step is transfer to spherical coordinate with $|J_T| = \sin\theta_{vh}$. Due to previous step θ_{vh} is used instead of θ_h. In this coordinate system reflecting of v relative to h is just doubling θ_{vh} and Jacobian of it is equal to $\frac{1}{2}$. In series of transform the Jacobians are multiplied so that currently $|J_T| = \frac{1}{2}\sin\theta_{vh}$. And the final step is inverse transform to Cartesian coordinate system with $|J_T| = (\sin\theta_{vl})^{-1} = (\sin 2\theta_{vh})^{-1}$. Combining this all together the following expression is obtained for reflection transform Jacobian [Schutte18]:

$$|J_T| = \frac{\sin\theta_{vh}}{2\sin 2\theta_{vh}} = \frac{\sin\theta_{vh}}{4\sin\theta_{vh}\cos\theta_{vh}} = \frac{1}{4\cos\theta_{vh}}$$

And finally $p(l)$ looks like:

$$p(l) = p(h)|J_T| = \frac{D\cos\theta_h}{4\cos\theta_{vh}}$$

Lets go back to the Monte-Carlo sum and insert found result to it:

$$\int_H \frac{DG}{4\cos\theta_v\cos\theta_l}g(v,l)\cos\theta_l\,\mathrm{d}l \approx \frac{1}{N}\sum_{i=1}^{N}\frac{DG\,g(v,l_i)\cos\theta_{l_i}}{4\cos\theta_v\cos\theta_{l_i}\,p(l_i)} = \frac{1}{N}\sum_{i=1}^{N}\frac{G\,g(v,l_i)\cos\theta_{l_i}\cos\theta_{vh_i}}{\cos\theta_v\cos\theta_{l_i}\cos\theta_{h_i}}$$

Here G component is recommended to be calculated with original $k = \frac{\alpha}{2} = \frac{r^2}{2}$ in order to get more plausible result. Of course, all $\cos$ must be clamped to range $[0,1]$ because integral is calculated on a hemisphere and all expressions are defined on it. $\cos\theta_v\cos\theta_{l_i}$ in denominator can be reduced with exactly the same part in geometric attenuation factor in order to avoid additional zero division cases.

Summarizing importance sampling strategy described above the convergence of Monte-Carlo integration can be improved using special PDF correlated with integrated function. In case of BRDF with normal distribution functions D the PDF producing procedure is defined. Practically half vector h is generated first and l is obtained from it by view vector v reflecting. Due to this transformation final form of probability density used in sum is quite different but also has defined algorithm of calculation. For isotropic Cook-Torrance BRDF the $\cos\theta_v$ and roughness r are enough to start generation so that all integrals of that kind can be precalculated in 2D look-up tables variating these two parameters. The same samples generation procedure must be used in specular map baking described in next chapter.

0.9 Specular map

The situation with BRDF part of $L^s_{indirect}$ is clear now and $L_i^{indirect}(l)$ sum is left to be discussed. That was called **specular map** and has following form:

$$\frac{1}{N}\sum_{i=1}^{N} L_i^{indirect}(l_i)$$

As was mentioned this sum must be calculated for every normal direction using the same samples generation principles as in numeric integration computation. This principles require two scalar parameters $\cos\theta_v$ and r but now ϕ really matters. So that in fact the specular map has to be saved in 3D table consisting omnidirectional textures. That is a big expense of computational and memory resources. A couple of tricks helps to reduce dimensions. First of all the $\cos\theta_v$ and ϕ can be just excluded. In that way v is considered to be equal to n. Of course this approach produces an error and affects the final result. It can be fixed more or less by $\cos\theta_{l_i}$ weighting [Karis13]:

$$\frac{1}{N}\sum_{i=1}^{N} L_i^{indirect}(l_i)\cos\theta_{l_i}$$

It is not a complete solution but practice shows that it is enough to get plausible illumination with sacrificing of lengthy reflections at grazing angles which exist in fact if everything is honestly computed. The problem is that for $v \neq n$ considering this sum to be defined related to n became incorrect. For example, for complete mirroring materials with $r = 0$ this sum must boil down to $L_i^{indirect}(v_r)$ but not to $L_i^{indirect}(n)$ where v_r is just reflected v or in other words $v_r = 2(v \cdot n)n - v$. That it just mirroring reflection principle. Assumption of $n = v$ also means that $n = v = v_r$. In that way radiance map better to be considered as averaging of illumination coming from v_r. So that it has become to be defined related to reflection direction which has to be calculated before map's fetching.

Anyway, there are just two dimensions in radiance look-up table remain. The rest one with r parameter cannot be reduced. There is no other ways except just roughness variation but in order to simplify that computations can be done for several values and the rest ones lying between can be obtained from linear interpolation. This is another source of visual artifacts but it also works good in practice and that is pretty common approach. But it still requires noticeably amount of samples and that is for every pixel related to each r value. It can be appropriate for precomputations but still limits using dynamic environments in real time rendering or just even static environments but on weak devices such as mobile ones. And there are several possible ways to improve radiance map baking performance.

The first one is using textures with smaller resolutions for larger roughnesses. The point is that smaller r values produce map saving more details from origin environment in opposite to larger ones representing lower frequency components and working as low pass filters in fact. So less pixels in combination with linear interpolation is enough to store less detailed convolutions. Moreover, this approach naturally works with textures levels of details in graphics API so that every certain radiance map related to certain r can be stored on its own mip level and be directly fetched with linear interpolation not only over one texture but over levels too. As practice shows 6 levels are enough.

After reducing pixels count it is turn for samples number. And again correlation with roughness can be noticed. For example map for completely mirroring materials with $r = 0$ the same sample $l_i = v_r$ will be produced. So that only one sample is enough in this case. In opposite way directions for $r = 1$ will be scattered over almost whole hemisphere what requires as much samples as available. The 'locality' of distribution is decreased during increasing roughness and it is possible to assume that samples number might to be proportional to this 'locality' keeping accuracy at the same level. But how can 'locality' be interpreted in terms of probability distribution? One possible way is CDF meaning. $F(\theta_h)$ has been already defined and by definition it shows the probability of random value θ_h to be less than argument of CDF. In other words $F(\theta'_h) = u$ means that probability of θ_h to be in range of $[0, \theta'_h]$ is u. The inverse task of range searching using given u can be solved with help of $F^{-1}(u) = \theta'_h$. If u is close to 1 (exact 1 has no sense because in that case $\theta'_h = \max\theta_h = \frac{\pi}{2}$) then θ'_h represents the range of the most probable or most frequently generated values and that can be interpreted as 'locality' of distribution. After that if samples number of the worst case with $r = 1$ is set ($N(1) = \max N$) the other ones can be estimated using following formula:

$$N(r) = N(1)\frac{\theta'_h(r)}{\frac{\pi}{2}} = N(1)\frac{2\theta'_h(r)}{\pi} = N(1)\frac{2F^{-1}(u)}{\pi} = N(1)\frac{2}{\pi}\arccos\sqrt{\frac{1-u}{u(\alpha^2-1)+1}}$$

It is approximate expression representing only estimated general proportionality so that cases of $r = 0$ and $r = 1$ must be processed separately with $N(0) = 1$ and $N(1) = \max N$. u can be parameter of this optimization strategy controlling speed of samples reducing in order to balance performance and quality ($u = 1$ disables this optimization at all). This pretty tricky technique allows reducing calculations for every pixels without sacrificing the quality.

In addition to optimizations mentioned before another one can be applied in order to help to reduce samples numbers as previous one. Using less samples produces image noise due to discrete nature of Monte-Carlo approximation. But it can be slightly smoothed using some prefiltration. The idea is that for the directions with small PDF or in other words for rare directions the samples near of it is unlikely to be generated. So that this direction represents the averaged illumination from relatively big area on hemisphere but approximate it by just a constant. It wold be better to get from such direction already averaged over bigger area environment. It can be achieved using mip levels of origin $L_i^{indirect}$ whose pixels of one level is exact 4 averaged pixels from previous one. Also mip levels generation is build in most common graphic API so there are no problems with it. But first of all the area covered by one sample is needed to be found. And that can be done as [Colbert07]:

$$\Omega_s = \frac{1}{N\,p(l)} = \frac{4\cos\theta_{vh}}{ND\cos\theta_h}$$

Circumstance of $v = v_r = n$ leads to $\cos\theta_{vh}$ and $\cos\theta_h$ reducing so expression becomes even simpler:

$$\Omega_s = \frac{4}{ND}$$

Of course all zero divisions must be avoided by clamping, for example. After that the area covered by one pixel of environment map is calculated. In case of a cube map it looks like:

$$\Omega_p = \frac{4\pi}{6k^2}$$

Where k is size of one cube map side in pixels (sides are assumed to be quads). Finally the mip level of origin environment map which is needed to be fetched for this certain sample is defined by following expression:

$$lod = \frac{1}{2}\log_2\left(\frac{\Omega_s}{\Omega_p}\right)$$

The mathematics connected with mip levels sizes lie behind it but this is out of scope of this paper. In combination with previous optimization technique this approach allows $N(1)$ to be smaller keeping visual results good.

That is not all possible optimization tricks but at least these three significantly reduces compute efforts and brings radiance map calculation to weak devices or even to dynamic environments in real time but in reasonable limits.

In that way $L^s_{indirect}$ can be completely computed without any integral approximations. Only 2D look-up table of BRDF part and mip mapped omnidirectional texture of irradiance map are needed. The first one can be got even without any environment. It was achieved using some rough approximations and assumptions but despite of that the visual result are still plausible and can be compared even with ray traced images. In order to complete whole image based lighting the $L^d_{indirect}$ component is left to be discussed.

0.10 Spherical harmonics

Lets go back to diffuse indirect illumination component represented by following formula:

$$L^d_{indirect} = (1-m)\frac{c}{\pi}\int_H (1-F)L_i^{indirect}(l)\cos\theta_l\,\mathrm{d}l$$

Of course, Monte-Carlo algorithm can be applied directly and hemisphere integral can be precalculated for every normal direction but dependence from v in Fresnel's factor does not allow to do it efficiently (every v direction is needed to be considered again). In order to resolve it modified version of Schlick's approximation has been created [[?](TODO)]:

$$F \approx F_{ss} = F_0 + (\max(1-r, F_0))(1-\cos\theta_v)^5$$

It differs from origin one and loses accuracy a little bit but now there is no light direction inside so that it can be considered as kind of screen space defined Fresnel's factor (ss means exactly 'screen space') and can be removed from integral:

$$L^d_{indirect} = (1-m)(1-F_{ss})\frac{c}{\pi}\int_H L_i^{indirect}(l)\cos\theta_l \,\mathrm{d}l$$

The resulted expression without $(1-m)$ and $(1-F_{ss})$ parts is pretty known entity called **irradiance**. It can be precalculated using $\cos\theta_l$ as PDF for importance sampling (actually it is only option in this case excluding uniform distribution). But even with that samples will be scattered almost over whole hemisphere. As was discussed in previous chapter this case requires significant amount of samples in order to average illumination with appropriate quality. Poor accuracy resulted from lack of summand can be noticed especially on high frequency environments having a lot of contrasting details. It worth to be mentioned that irradiance is used not only in BRDF. Omnidirectional diffuse illumination captured for certain point or even for several points uniformly or hierarchically distributed is base of some baking global illumination techniques. There it is called a **light probe**. So that other way to compute and store irradiance maps was found resolving many mentioned problems. The Fourier's decomposition analogue for spherical function allows to achieve this. That would be easy to explain concept directly on example. So lets start from $L_i^{indirect}(l)$. It is spherical function because directions are just points on sphere. The decomposition looks like:

$$L_i^{indirect}(l) = \sum_{i=0}^{\infty}\sum_{j=-i}^{i} L_i^j y_i^j(l)$$

Where $y_i^j(l)$ are spherical functions forming orthonormalized basis called spherical harmonics and L_i^j is decompositions coefficients. Orthonormality means that dot product of two basis elements is equal to 1 if this is the same functions and is equal to zero otherwise. Dot product on a sphere is defined as integral of functions multiplication. In other words:

$$\int_S y_i^j(l)\, y_{i'}^{j'}(l)\,\mathrm{d}l = \begin{cases} 1 & i,j = i',j' \\ 0 & \text{otherwise} \end{cases}$$

Function basis with such traits is known and is described by following formulas [Guy18]:

$$y_i^{j>0}(\theta,\phi) = \sqrt{2}K_i^j \cos(j\phi)P_i^j(\cos\theta)$$
$$y_i^{j<0}(\theta,\phi) = \sqrt{2}K_i^j \sin(j\phi)P_i^{|j|}(\cos\theta)$$
$$y_i^0(\theta,\phi) = K_i^0 P_i^0(\cos\theta)$$

$$K_i^j = \sqrt{\frac{(2i+1)(i-|j|)!}{4\pi(i+|j|)!}}$$

$$P_0^0(x) = 1$$
$$P_1^0(x) = x$$
$$P_i^i(x) = (-1)^i(2i-1)!!(1-x^2)^{\frac{i}{2}}$$
$$P_i^j(x) = \frac{(2i-1)xP_{i-1}^j(x) - (i+j-1)P_{i-2}^j}{i-j}$$

Here K_i^j are normalization factors and P_i^j are **Legendre's polynomials**. Decomposition coefficients L_i^j are dot product of origin function ($L_i^{indirect}(l)$ in current case) and corresponding basis element. It can be written down as:

$$L_i^j = \int_S L_i^{indirect}(l)\, y_i^j(l)\, \mathrm{d}l$$

Fact that all calculation happen over a sphere but not over hemisphere right now is important not to be missed. That was example of spherical function decomposition but not a solution for original task which looks like:

$$\int_H L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l$$

First of all, lets transform this integral to be defined not over hemisphere but sphere:

$$\int_H L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l = \int_S L_i^{indirect}(l) \overline{\cos}\theta_l \, \mathrm{d}l$$

Where $\overline{\cos}$ is cosine clamped to zero which can be expressed as:

$$\overline{\cos}\theta_l = \max(\cos\theta_l, 0)$$

Resulted expression can be considered as convolution in terms of spherical functions where $L_i^{indirect}(l)$ is target and $\overline{\cos}\theta_l$ is core. This integral may seem independent but in fact hemisphere is oriented related to n therefore $\overline{\cos}\theta_l$ depends on it too and became a kind of 'oriented' version of cosine. That is pretty tricky and explanation about meaning of convolution on sphere is out of scope of this paper. Fact that this is convolution analogue related to n is enough for now [Aguilar17], [Ramamoorthi01]. The goal of looking at integral from this angle is using of convolution's trait allowing to compute decomposition using just only coefficients of function and core. $\overline{\cos}\theta_l$ is independent from ϕ_l and in case of such radial symmetric cores the resulting coefficients boil down to following formula:

$$(L_i^{indirect}(l) * \overline{\cos}\theta_l)_i^j = \frac{1}{K_i^0} L_i^j \, c_i^0 = \sqrt{\frac{4\pi}{2i+1}} L_i^j \, c_i^0$$

Where c_i^0 are spherical harmonics factors corresponding to $\overline{\cos}\theta$. Therefore the final decomposition looks like:

$$\int_{H(n)} L_i^{indirect}(l) \cos\theta_l \, \mathrm{d}l = \int_S L_i^{indirect}(l) \overline{\cos}\theta_l \, \mathrm{d}l = \sum_{i=0}^{\infty} \sum_{j=-i}^{i} \sqrt{\frac{4\pi}{2i+1}} L_i^j \, c_i^0 \, y_i^j(n)$$

c_i^0 is left to be found. Due to independence from ϕ all $c_i^{j\neq 0} = 0$. The rest ones are calculated by regular dot product with basis functions:

$$c_i^0 = c_i = \int_S y_i^0(l) \overline{\cos}\theta_l \, \mathrm{d}l = \int_0^{2\pi} \mathrm{d}\phi \int_0^{\pi} y_i^0(\theta,\phi) \overline{\cos}\theta \sin\theta \, \mathrm{d}\theta = \int_0^{2\pi} \mathrm{d}\phi \int_0^{\frac{\pi}{2}} y_i^0(\theta,\phi) \cos\theta \sin\theta \, \mathrm{d}\theta =$$

$$= 2\pi \int_0^{\frac{\pi}{2}} y_i^0(\theta,\phi) \cos\theta \sin\theta \, \mathrm{d}\theta = 2\pi K_i^0 \int_0^{\frac{\pi}{2}} P_i^0(\cos\theta) \cos\theta \sin\theta \, \mathrm{d}\theta$$

$\sin\theta$ appears due to transfer from integral over sphere to double integral where $\mathrm{d}l = \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi$. There is an analytical solution for this expressions:

$$c_1 = \sqrt{\frac{\pi}{3}}$$

$$c_{odd} = 0 \quad c_{even} = 2\pi \sqrt{\frac{2i+1}{4\pi}} \frac{(-1)^{\frac{i}{2}-1}}{(i+2)(i-1)} \frac{i!}{2^i \left(\frac{i!}{2}\right)^2}$$

Starting from about the third c_i the coefficients become less and less valuable so that only couple of them is enough in order to approximate $\overline{\cos}\theta$ with appropriate accuracy. The same principle is true for convolution too because its coefficients are multiplied by c_i. So there is no need to use more than even three bands ($i = 0,1,2$) of basis functions. Lets write down them all in Cartesian coordinate [Ramamoorthi01]:

$$y_0^0 = \frac{1}{2}\sqrt{\frac{1}{\pi}} = Y_0^0$$

$$y_1^{-1} = -\frac{1}{2}\sqrt{\frac{3}{\pi}}y = Y_1^{-1}y$$

$$y_1^0 = \frac{1}{2}\sqrt{\frac{3}{\pi}}z = Y_1^0 z$$

$$y_1^1 = -\frac{1}{2}\sqrt{\frac{3}{\pi}}x = Y_1^1 x$$

$$y_2^{-2} = \frac{1}{2}\sqrt{\frac{15}{\pi}}xy = Y_2^{-2}xy$$

$$y_2^{-1} = -\frac{1}{2}\sqrt{\frac{15}{\pi}}yz = Y_2^{-1}yz$$

$$y_2^0 = \frac{1}{4}\sqrt{\frac{5}{\pi}}(3z^2-1) = Y_2^0(3z^2-1)$$

$$y_2^1 = -\frac{1}{2}\sqrt{\frac{15}{\pi}}xz = Y_2^1 xz$$

$$y_2^2 = \frac{1}{4}\sqrt{\frac{15}{\pi}}(x^2-y^2) = Y_2^2(x^2-y^2)$$

All Y_i^j are just constants so that it can be moved from integral during calculations and can be taken from precomputed table. Other constants related to c_i can be united and also be calculated beforehand:

$$\hat{c}_i = \frac{1}{K_i^0}c_i = \sqrt{\frac{4\pi}{2i+1}}c_i$$

Finally expression of irradiance map approximation can be defined:

$$\int\limits_{H(n)} L_i^{indirect}(l)\cos\theta_l \,\mathrm{d}l \approx \sum_{i=0}^{2}\sum_{j=-i}^{i} L_i^j \hat{c}_i y_i^j(n)$$

Where $\hat{c}_i$ is precalculated constants $y_i^j(n)$ are pretty easy functions and only L_i^j are needed to be precomputed. Of course L_i^j are integrals over even whole sphere but now there is only nine of it instead of one for every pixel of omnidirectional image. Moreover, texture is not needed at all in that case and only 9 colors representing L_i^j have to be saved. The Monte-Carlo algorithm can be applied with just uniform samples distribution without importance sampling at all. Y_i^j are used twice: in L_i^j calculations and in sum after that. So there is sense to store only squares of it. All tables with constants presented below [Ramamoorthi01]:

$(Y_0^0)^2 \approx (0.282095)^2$
$(Y_1^{-1})^2 = (Y_1^0)^2 = (Y_1^1)^2 \approx (0.488603)^2$
$(Y_2^{-2})^2 = (Y_2^{-1})^2 = (Y_2^1)^2 \approx (1.092548)^2$
$(Y_2^0)^2 \approx (0.315392)^2$
$(Y_2^2)^2 \approx (0.546274)^2$

$\hat{c}_0$	3.141593
$\hat{c}_1$	2.094395
$\hat{c}_2$	0.785398

Summarizing all mathematics above spherical harmonics decomposition boils down irradiance map to only 9 values which is needed to be precalculated as integrals. As practice shows this is very good approximation of diffuse indirect illumination component.

0.11 Transparent materials

TODO

0.12 Low discrepancy sequence

TODO

0.13 References

[Duvenhage13]	Bernardt Duvenhage, Kadi Bouatouch, D.G. Kourie, "Numerical Verification of Bidirectional Reflectance Distribution Functions for Physical Plausibility", *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, October 2013.
[Cook81]	Robert Cook, Kenneth Torrance, "A Refectance Model for Computer Graphics", *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, August 1981, pp. 307-316.
[Karis13]	Brian Karis, "Real Shading in Unreal Engine 4", *SIGGRAPH 2013 Presentation Notes*.
[Heitz14]	Eric Heitz, "Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs", *Journal of Computer Graphics Techniques*, Vol. 3, No. 2, 2014.
[Schlick94]	Christophe Schlick, "An inexpensive brdf model for physically-based rendering", *Computer Graphics Forum 13*, 1994, pp. 233-246.
[Lazanyi05]	Istvan Lazanyi, Lazslo Szirmay-Kalos, "Fresnel term approximations for Metals", January 2005.
[Lagarde13]	Sebastien Lagarde, "Memo on Fresnel equations", *Blog post*: https://seblagarde.wordpress.com/2013/04/29/memo-on-fresnel-equations/.
[Walter07]	Bruce Walter, Stephen Marschner, Hongsong Li, Kenneth Torrance, "Microfacet Models for Refraction through Rough Surfaces", *Proceedings of Eurographics Symposium on Rendering*, 2007.
[Cao15]	Jiayin Cao, "Sampling microfacet BRDF", November 1, 2015, *Blog post*: https://agraphicsguy.wordpress.com/2015/11/01/sampling-microfacet-brdf/.
[Schutte18]	Joe Schutte, "Sampling techniques for GGX with Smith Masking-Shadowing: Part 1", March 7, 2018, *Blog post*: https://schuttejoe.github.io/post/ggximportancesamplingpart1/.
[Colbert07]	Mark Colbert, Jaroslav Krivanek, "GPU-Based Importance Sampling", *NVIDIA GPU Gems 3*, Chapter 20, 2007.

[Guy18]	Romain Guy, Mathias Agopian, "Physically Based Rendering in Filament", *Part of Google's Filament project documentation*: https://google.github.io/filament/
[Aguilar17]	Orlando Aguilar, "Spherical Harmonics", *Blog post*: http://orlandoaguilar.github.io/sh/spherical/harmonics/irradiance/map/2017/02/12/SphericalHarmonics.html
[Ramamoorthi01]	Ravi Ramamoorthi, Pat Hanrahan, "An Efficient Representation for Irradiance Environment Maps", *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, August 2001, pp. 497-500