

# 後半パート

---

- はじめに
- 基礎知識
  - マイクロファセット (Microfacet)
  - 法線分布関数 (Normal Distribution Function)
  - マスキング関数 (Masking-Shading Function)
  - 可視法線分布関数 (Visible Normal Distribution Function)
- 最近の話題
  - Microsurface Transformationsの解説  
ヤコビアンさえあれば良い (Jacobian Is All You Need)
- 最後に

CEDEC2023

私(大垣)のパートでお話しうる内容はこのようになっていて、まず、下準備として基本的なことについて軽く説明し、それから最近の研究であるMicrosurface Transformationsを私の解釈を交えてできるだけわかりやすく解説したいと思います。元の論文はなかなか複雑ですが、特にヤコビアンだけわかっていていればGGXを実装する際に必要となるサンプリングや確率密度関数の計算が簡単になる、ということを伝えたいと思っています。

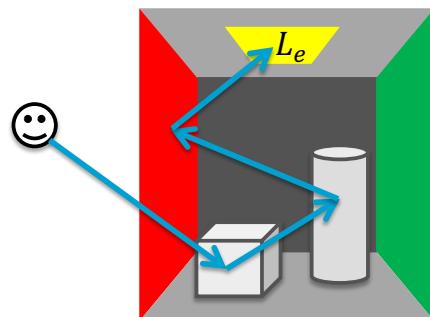
## 後半パート

$$L(\omega_o) = L_e + \int_{\Omega^\pm} f(\omega_i, \omega_o) L_{in}(\omega_i) |\omega_n \cdot \omega_i| d\omega_i$$

レンダリング方程式をパストレーシングで解く

GGXを使う時に必要となるのは

- サンプリング
- 確率密度関数



CEDEC2023

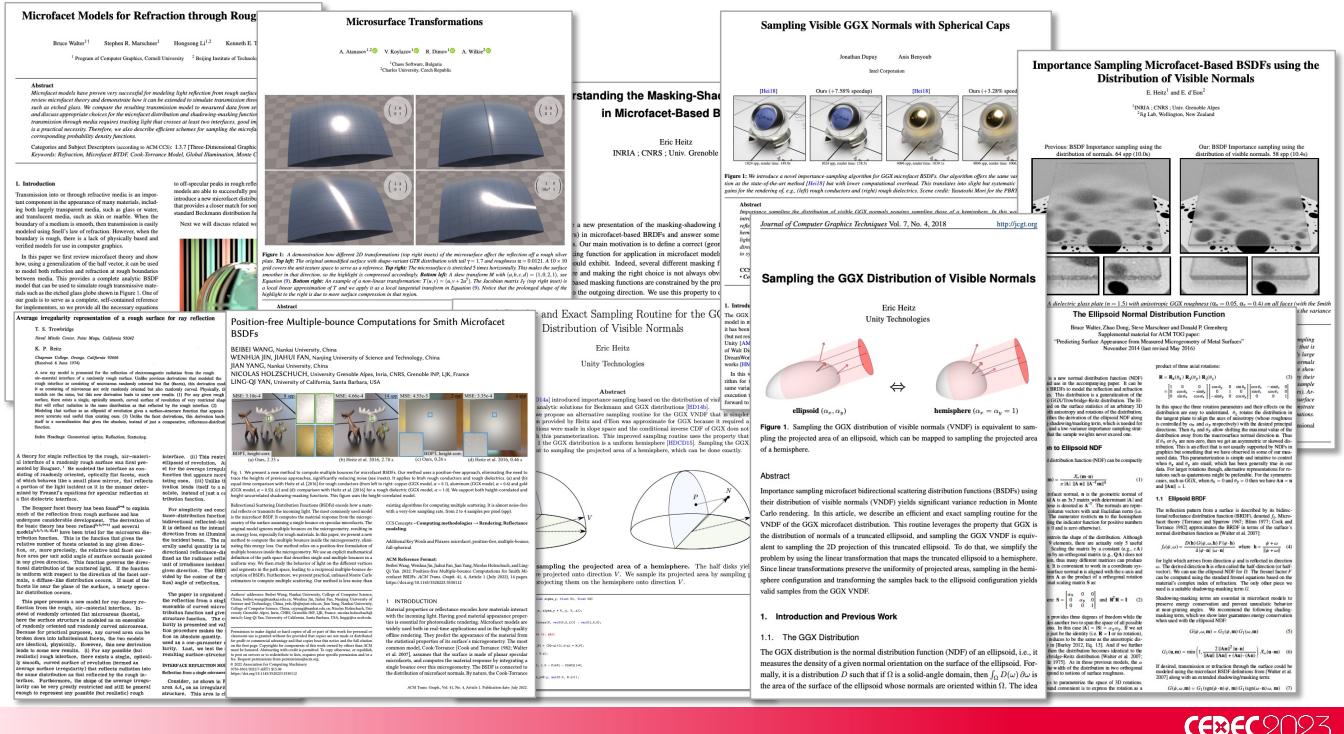
前半のパートはBRDFの成り立ちや評価についてでしたが、後半では、レンダリングの支配的な手法となりつつあるパストレーシングでGGXを用いるという状況を想定していて、サンプリングや確率密度関数の計算が主なテーマとなります。これらをどうやって簡単に行うかをお伝えするのが目的となっています。

---

# はじめに

CEDEC2023

では早速始めましょう。



# はじめに

---

一つの単純な疑問から

CEDEC2023

今日はどうすれば少しでも楽に実装できるのかを理解してもらうため、  
一つの極めて単純な質問から始めたいと思います。

# はじめに

---

以下の関数はどんな形か？

$$D(\omega_m) = \frac{1}{\pi \alpha_x \alpha_y \left( \frac{\omega_m^2}{\alpha_x^2} + \frac{\omega_m^2}{\alpha_y^2} + \omega_m^2 \right)^2}$$

( $\alpha_x$ と $\alpha_y$ は縦方向と横方向での粗さ)

CEDEC2023

その質問というのは、ここに書かれたGGXの法線分布関数と呼ばれるものですが、これを見て一体どのような形か想像がつくでしょうか？というものです。私は全くイメージできませんでした。

## はじめに

---

試しに  $\alpha_x = \alpha_y = 1$  を代入してみる

$\|\omega_m\| = 1$  なので

$$D(\omega_m) = \frac{1}{\pi (\omega_m_x^2 + \omega_m_y^2 + \omega_m_z^2)^2} = \frac{1}{\pi}$$

CEDEC2023

こういった場合私はどうするかというと、具体的な数字を代入してみます。2つのパラメータ  $\alpha_x, \alpha_y$  があるので、試しに両方に 1を入れてみましょう。するとなんと  $\frac{1}{\pi}$  という形になります。これは偶然ではありません。

# はじめに

---

この単純化された $D(\omega_m)$ を

$$D_{std}(\omega_m) = \frac{1}{\pi}$$

と書くことにする

CEDEC2023

この単純化された法線分布関数を $D_{std}$ と書くことにします。

# はじめに

---

$D_{std}(\omega_m)$ を使うと全てが簡単になる！

CEDEC2023

後ほど詳しくみていきますが、実は $D_{std}$ だけ知っておくとサンプリングやPDFの計算、その理解が非常に楽になります。

# はじめに

---

- 法線分布関数が質感を決める重要な役割を果たす
- GGXを使いこなすにはJacobianさえ分かれば良い

CEDEC2023

数式が出てきてすぐに分からなくても全く問題ないですが、この2点だけ記憶に留めておいていただくと、マイクロファセットやそれにまつわる新しい研究を理解する助けになるのではないかと思います。

---

# Microfacet

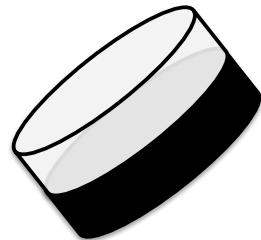
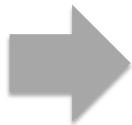
## マイクロファセット

CEDEC2023

最近の話題を紹介するにあたって予備知識が必要となるので、いくつか見ていきましょう。まずはマイクロファセットからです。

# マイクロファセット

---



一つ一つの微小な面は  
「鏡」  
とよく書かれている

上側のみ鏡、下側は完全な黒  
(Mono-sided microflakes)

CEDEC2023

マイクロファセット理論では、物体の表面を細かいマイクロファセットと呼ばれる面で表現し、その向きを統計的な分布で表すことで、さまざまな質感を再現することを容易にします。一般的な文献ではマイクロファセットは、完全鏡面であるとしています。今回話す内容では、マイクロファセットはdouble-sided materialで、上側のみ完全鏡面反射面で、下側、つまり、マクロな面を向いている側は完全な黒であると仮定します。これはmono-sided microflakesとも呼ばれます。

# マイクロファセット

---



Heightfieldのみ扱える



珊瑚のような形状は扱えない

CEDEC2023

また古典的なマイクロファセットモデルはハイトイールドは扱えますが、珊瑚のような入り組んだ形状は扱えません。

# マイクロファセット

---



CEDEC2023

粗さを1としたときのGGXで表されるマイクロサーフェスの形状は、マイクロファセットを並べ替えると、

# マイクロファセット

---



CEDEC2023

このように綺麗な半球へと変換できます。

---

---

# Normal Distribution Function

## 法線分布関数

CEDEC2023

続いて法線分布関数についてみていきましょう。マイクロファセットの向きの分布を表すのが法線分布関数と呼ばれる関数です。英語では Normal Distribution Function、略称NDFです。

# 法線分布関数

---

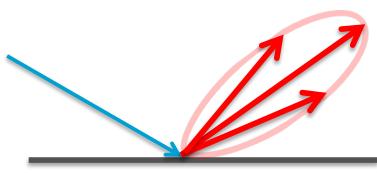
$$f_{\text{reflection}}(\omega_i, \omega_o) = \frac{F(\omega_i, \omega_m) G_2(\omega_i, \omega_o, \omega_m) D(\omega_m)}{4|\omega_o \cdot \omega_n| |\omega_i \cdot \omega_n|}$$

CEDEC2023

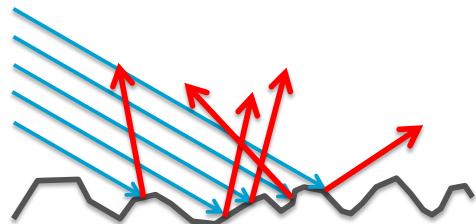
前半部分で出てきたBRDFの式のこの赤で強調された部分で通常 $D$ を用います。

# 法線分布関数

$$f_{\text{reflection}}(\omega_i, \omega_o) = \frac{F(\omega_i, \omega_m)G_2(\omega_i, \omega_o, \omega_m)}{4|\omega_o \cdot \omega_n| |\omega_i \cdot \omega_n|} D(\omega_m)$$



$f_{\text{reflection}}(\omega_i, \omega_o)$ は  
反射した光線の相対的な  
強さの分布を表す



反射する光線の向きは  
法線分布関数  $D(\omega_m)$  によって決まる

CEDEC2023

BRDFは反射する光線の相対的な強さの分布を表しますが、微小面の向きを決める法線分布関数は、その分布形状の大部分を決定づける大切な役割を果たします。

# 法線分布関数

---

$$f_{\text{reflection}}(\omega_i, \omega_o) = \frac{F(\omega_i, \omega_m) G_2(\omega_i, \omega_o, \omega_m) D(\omega_m)}{4|\omega_o \cdot \omega_n| |\omega_i \cdot \omega_n|}$$

CEDEC2023

また、マスキング関数は法線分布関数によって決まりますし、

## 法線分布関数

---

$$f_{\text{reflection}}(\omega_i, \omega_o) = \frac{F(\omega_i, \omega_m) G_2(\omega_i, \omega_o, \omega_m) D(\omega_m)}{4 |\omega_o \cdot \omega_n| |\omega_i \cdot \omega_n|}$$

CEDEC2023

フレネルもマイクロファセットの面の向きを用いて計算するので、ここにも影響を及ぼします。

# 法線分布関数

---

⚠ 球面上の分布と思いがち

CEDEC2023

気をつけて欲しいのは、この法線分布関数は球面上で定義された確率密度関数と誤解されがちなんですが、実はそうではないということです。

# 法線分布関数

---

⚠ 球面 $\Omega$ 上で積分しても1にならない

$$\int_{\Omega^\pm} D(\omega_m) d\omega_m \neq 1$$

CEDEC2023

残念ながら、これは通常正規化されておらず、球上で積分しても1になりません。

# 法線分布関数

---

$\omega_m$ の方向を向いている  
マイクロファセットの面積

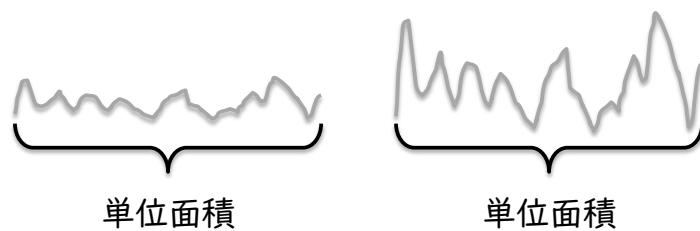
(単位は[m<sup>2</sup>/sr])

CEDEC2023

なぜかというと、 $D$ は方向 $\omega_m$ を向いている面の面積を表すものだからです。

# 法線分布関数

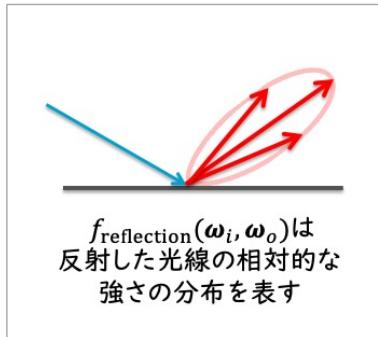
凹凸が激しい場合には大きな値



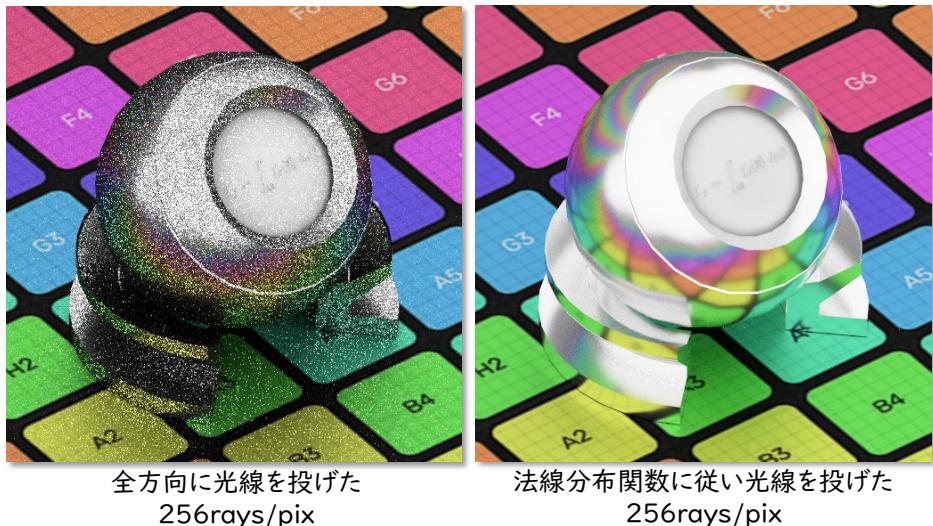
CEDEC2023

従って、似たような形状であったとしても、凹凸が激しいほど値が大きくなり、このままでは扱いづらいです。

# 法線分布関数



※ 光線の方向を決める  
ことをサンプリングという



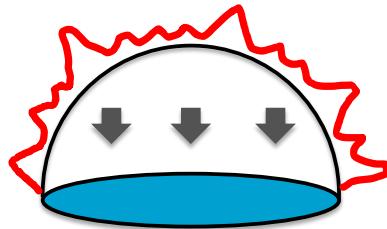
CEDEC2023

先ほどのBRDFの図を少し思い返してみましょう。BRDFの値が小さくなる方向に光線を飛ばしても、反射の計算にあまり役立ちません。出鱈目な方向に光線を投げて反射の計算をした場合には効率が悪く真ん中のようにノイズだらけの画像となってしまい、法線分布関数に従って光線を投げた場合には、右のように綺麗な反射の像を得ることができます。このように、法線分布関数に従って光線を飛ばすには、法線の方向を決めるための、言い換えれば、サンプリングをするための確率密度関数が必要になります。

# 法線分布関数

確率密度関数にするにはペタンとつぶす

$$\int_{\Omega^\pm} D(\omega_m) d\omega_m \neq 1$$



半球面で定義されたヒストグラム

CEDEC2023

実は、反球面上で定義された法線の向きのヒストグラムをマクロサーエスの法線方向にペタンと潰すことで、どんなハイトフィールドの法線分布でも確率密度関数へと変換することができます。

# 法線分布関数

確率密度関数にするにはペタンとつぶす

$$\int_{\Omega^\pm} D(\omega_m) \langle \omega_m, \omega_n \rangle d\omega_m = 1$$



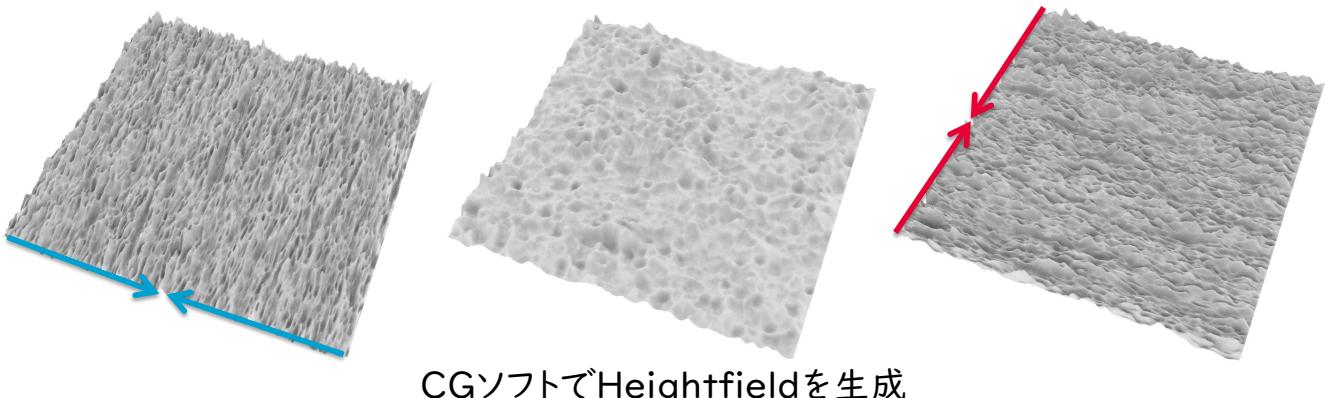
※ ヒストグラムの値にコサインをかけている

CEDEC2023

ハイトフィールドは潰してもそれを構成する面同士が重なることはありません。ですので、凹凸が激しい場合も、それほどではない場合も、潰してしまえば、ヒストグラムの総和はマクロサーフェスの単位面積と一致することになります。このつぶす操作というのは内積を一つかけるだけなので、 $D(\omega_m) \langle \omega_m, \omega_n \rangle$ が確率密度関数として機能するようになります。粗さが1の場合の標準法線分布関数の場合はヒストグラムが平らなので、文字通り半球をペタンと平面に潰すことになります。

# 法線分布関数

$$D(\omega_m) = \frac{1}{\pi \alpha_x \alpha_y \left( \frac{\omega_m^2}{\alpha_x^2} + \frac{\omega_m^2}{\alpha_y^2} + \omega_m^2 \right)^2}$$

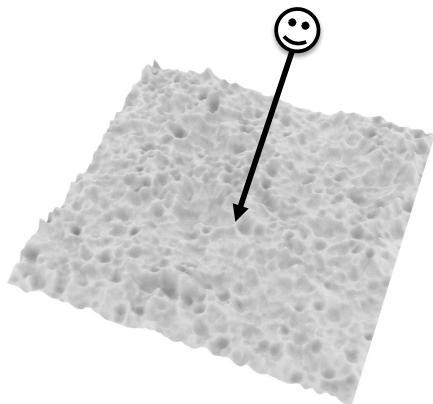


CGソフトでHeightfieldを生成

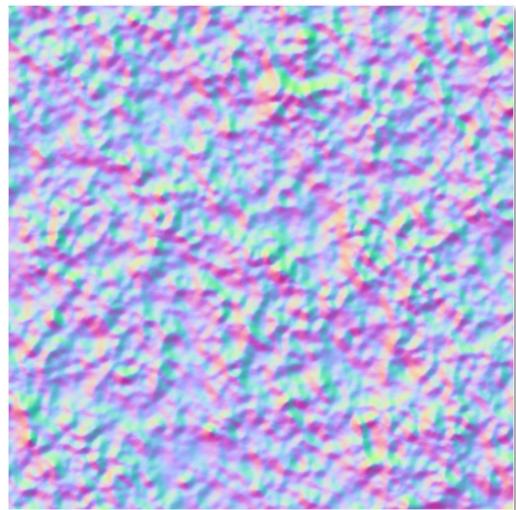
CEDEC2023

式ばっかりでは面白くありませんので、ハイトフィールドから確率密度関数をどう作るかみてみましょう。まずCGソフトでハイトフィールドを生成します。この凹凸は物体表面を顕微鏡で拡大したときの様子であると思ってください。この凹凸の程度が異なれば当然ながら違った材質に見えます。GGXの法線分布関数で現れる $\alpha_x$ と $\alpha_y$ はいわば縦横それぞれの潰れ度合いを表していて、潰れるほど大きな値となり、引き延ばされるほど小さくなります。潰す操作や引き延ばす操作というのは、線形変換でできます。

# 法線分布関数



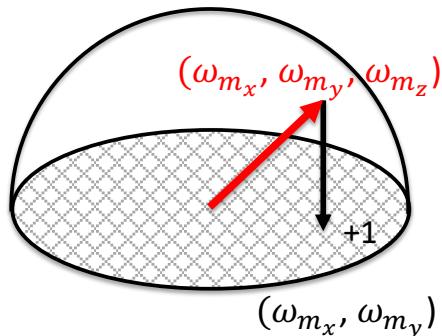
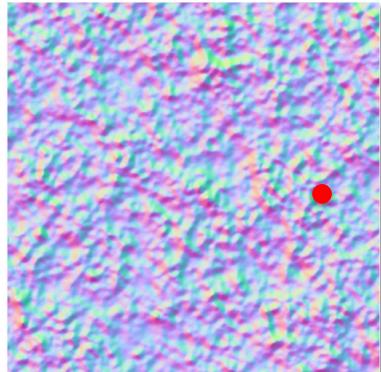
凹凸を真上から見てノーマルマップを生成



CEDEC2023

先ほど作った凹凸の面の向きに関する分布が欲しいので、ジオメトリを真上から見てノーマルマップを生成してみます。このノーマルマップからどの向きに向いている面が多いかというのを知るためのヒストグラムを作ります。実は真上から見るというのがマクロサーフェスの方向に沿って「潰す」という操作に相当します。

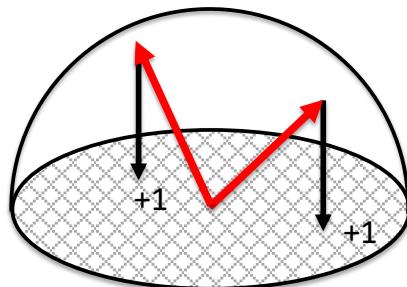
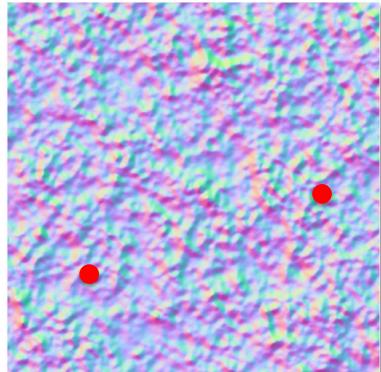
# 法線分布関数



CEDEC2023

やり方はいろいろありますが、ここでは過去の論文と似たような図を作るために、ディスク形状のヒストグラム、濃淡画像を作ります。ノーマルマップ内の1つのピクセルを選び、その方向をディスクに投影して、その場所の場所のカウンタを1つ増やします。単位ベクトルを扱っているので、向きをエンコードするにはx座標とy座標だけがわかれば構いません。

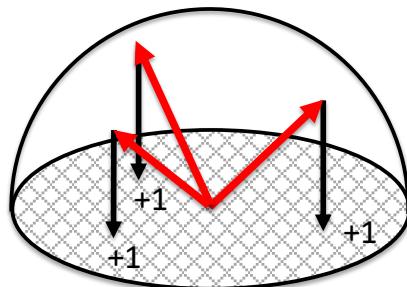
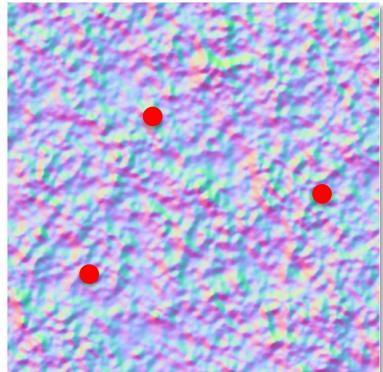
# 法線分布関数



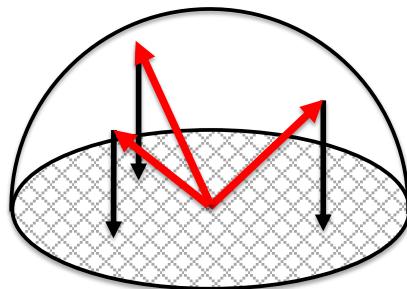
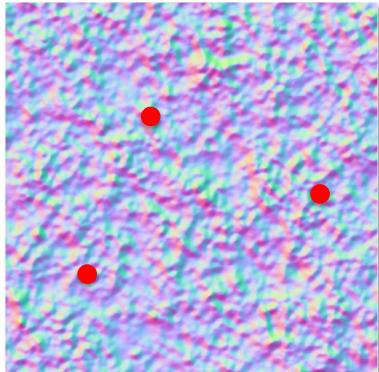
CEDEC2023

同様の操作をノーマルマップの全ての画素に対して行います。

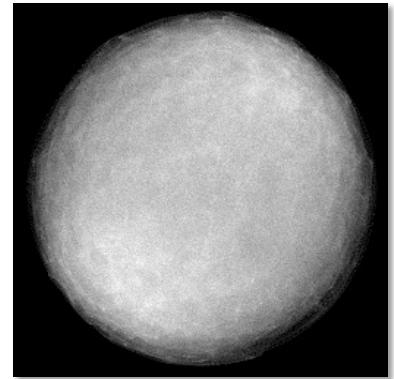
# 法線分布関数



# 法線分布関数



$$D(\omega_m) \langle \omega_m, \omega_n \rangle$$



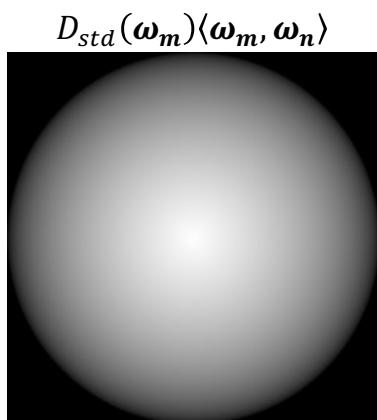
ヒストグラムの総和を画素数  
で割ると**1.0**になっている

CEDEC2023

続けていくと、最終的に右の画像のようなヒストグラムが完成し、これを確率密度関数としてレンダリングに用いることができます。

# 法線分布関数

---



標準法線分布関数の場合

CEDEC2023

粗さ 1 のGGXの場合はこのような綺麗な分布となります。以上が簡単な法線分布関数の話でした。

---

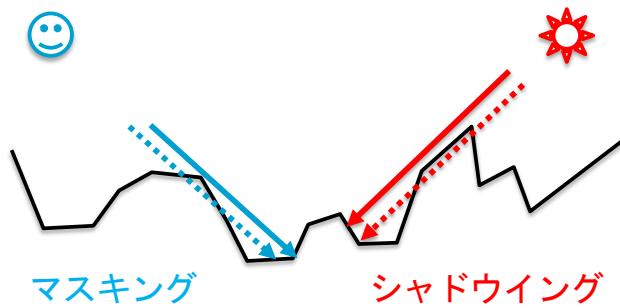
# Masking-Shadowing Function

## マスキング関数

CEDEC2023

続いてマスキング関数について復習していきたいと思います。

# マスキング関数



CEDEC2023

前半でもありました、マスキング関数(masking-shadowing function)というのは、どのくらい視線が遮れずに届くか、また、どのくらい光源からの光が遮られずに届くか、を表します。

# マスキング関数

---

可視法線分布関数を  
使う時に楽なので  
分離可能の形式で考える

$$G_2 = G_1 G_1$$

CEDEC2023

MaskingとShadowingの2つを加味した関数を $G_2$ としそれを使っても良いのですが、実際の計算においては、その2つを $G_1 \times G_1$ といった2つの積の形で表現した方が便利なことが多いです。なので、ここでも $G_1$ の積の形で表せる、という仮定を採用します。

# マスキング関数

$$G_{1, std}(\omega_m, \omega_o) = \chi^+(\omega_m \cdot \omega_o) \frac{2|\omega_{o_z}|}{1 + \omega_{o_z}}$$

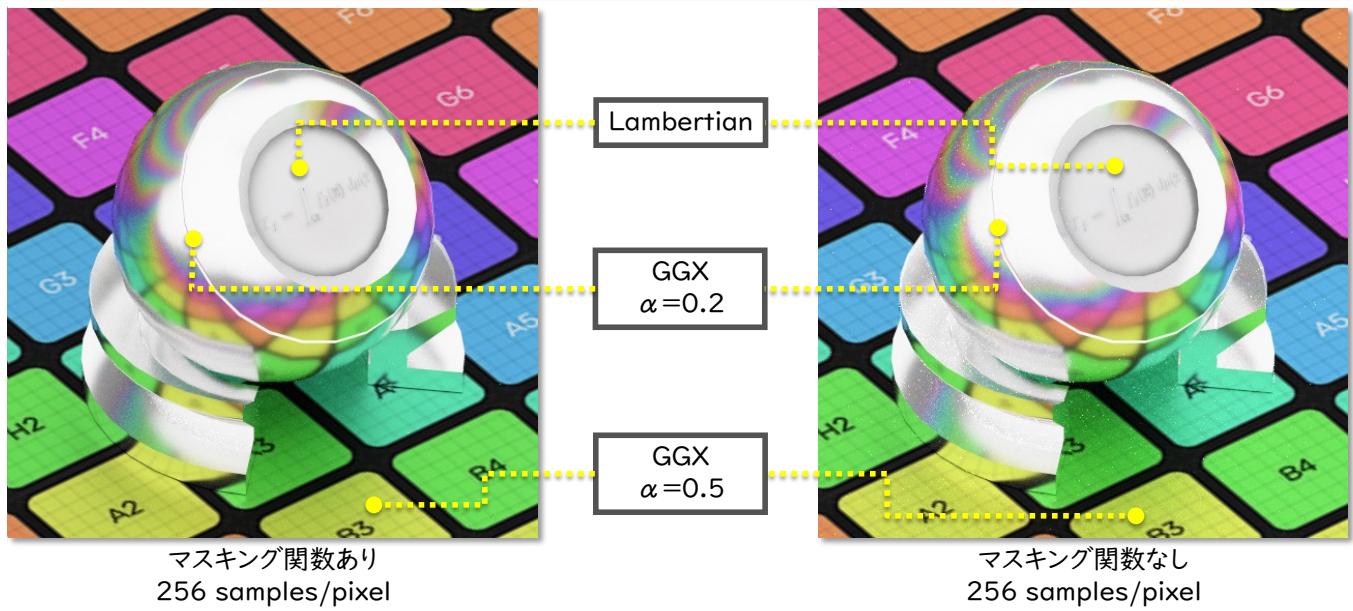
$\chi^+(\omega_m \cdot \omega_o)$

- $\omega_m \cdot \omega_o$ が常に正となるため、BRDFの場合は不要
- 多くの場合クランプトコサイン $\langle \omega_m, \omega_o \rangle$ と併用され不要

CEDEC2023

標準法線分布関数を用いている場合の標準マスキング関数はこの式で表されます。導出や最近の研究について配布資料で記述していますので、興味ある方は後ほど見てみてください。この式にはいくつか注意があって、まず $\chi^+(\omega_m \cdot \omega_o)$ はBRDFの場合は不要です。それから、 $\chi^+(\omega_m \cdot \omega_o)$ は、クランプトコサイン $\langle \omega_m, \omega_o \rangle$ と併用されるので多くの場合不要です。

# マスキング関数



CEDEC2023

前半でもありました  
が、マスキング関数がない場合は不自然に明るくなり、ノイズが発生してしまいます。

---

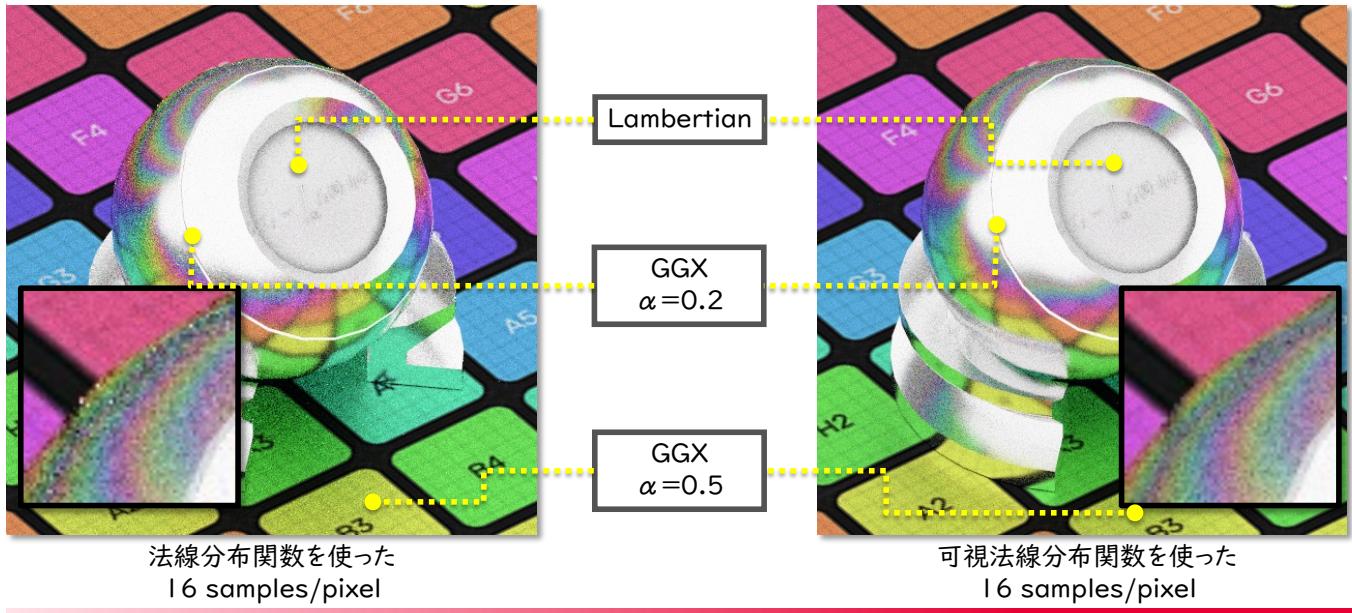
# Visible Normal Distribution Function

## 可視法線分布関数

CEDEC2023

続いて可視法線分布関数VNDFについて少しだけお話しします。これまで視点の位置などを一切考慮しない法線分布関数の話でした。法線分布関数を用いてももちろんさまざまな凹凸を持つ面を表現できますが、実際サンプリングするときにマイクロファセットの面の向きを求めて、それが遮蔽されていたり、視点の反対を向いていたりすれば、せっかくの計算が無駄になります。なので、視点の方へ向いていて、かつ、遮蔽もされていないマイクロファセットのみからサンプリングを行いたいという要求が生じます。可視法線分布関数というのは(マスキングもされず)見えているマイクロファセットの向きの分布を表すため、それを用いることで無駄な計算をしなくて済むようになります。

# 可視法線分布関数



CExEC2023

ここではどのような効果があるかのみお見せし、詳細は配布資料に含めるので興味のある方は後で読んでみてください。左は法線分布関数を用いた計算結果で、右は可視法線分布関数を用いて計算した結果です。私のPCではどちらも0.5sec程度ですが、可視法線分布関数で計算した方が綺麗な絵になっています。ただ、他のサンプリング方法と組み合わせて使用した場合はこのメリットが薄れてしまうこともあります。

---

---

# Microsurface Transformations

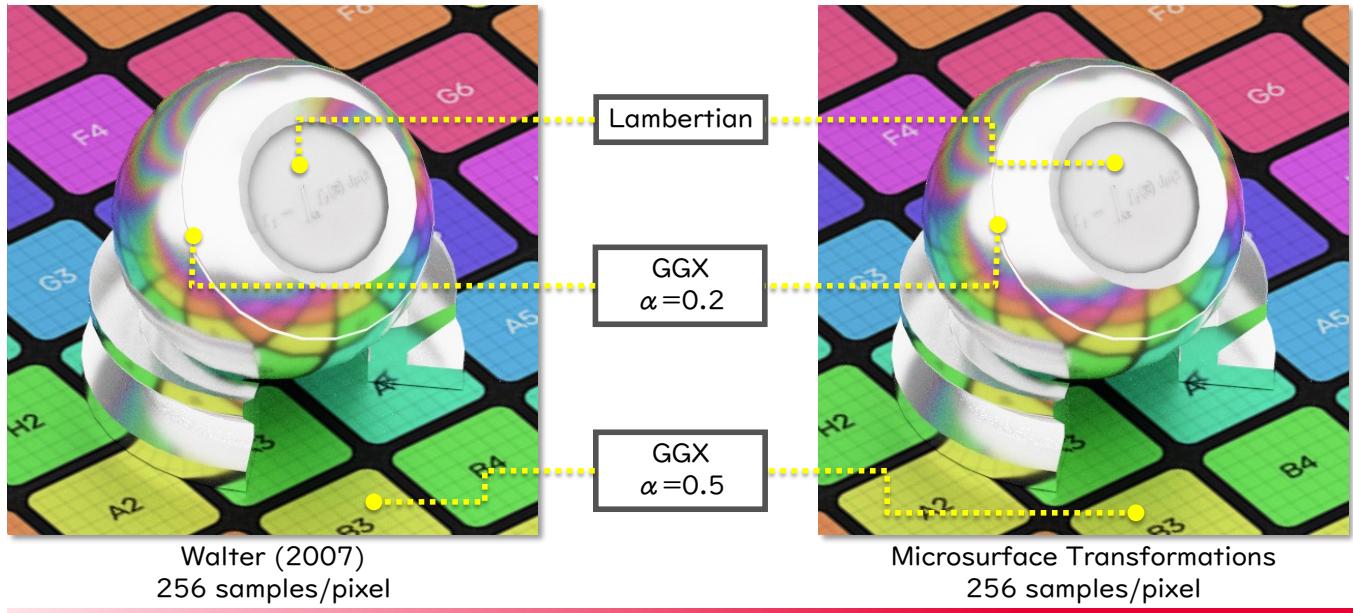
## Jacobian Is All You Need

### ヤコビアンさえあれば良い

CEDEC2023

さて、本題に入ります。ここからはMicrosurface Transformationsの内容をなるべく簡素化してお話ししたいと思います。少し込み入った話が続きますが、最終的に得られる計算は実はとっても簡単で、機械学習の有名な論文になぞらえれば、Jacobianさえ知っていれば良いですよ、といった内容になっています。

# Microsurface Transformations



CEDEC2023

先にレンダリングした絵をお見せします。左はWalterの2007年の論文の方法で計算したもの、右はこれから紹介するMicrosurface Transformationsを使って計算したもので、全く同じ結果が得られます。ただし、ずっと理解でしやすい方法で計算することができます。

## 標準法線分布関数

---

$$\int_{\Omega^\pm} D(\omega_m) \langle \omega_m, \omega_n \rangle d\omega_m = 1$$

確率密度関数にするには $\langle \omega_m, \omega_n \rangle$ が必要だった

CEDEC2023

先ほどあった確率密度関数の話を思い出してみましょう。 $D(\omega_m)$ を確率密度関数にするには $\langle \omega_m, \omega_n \rangle$ が必要でした。

# 標準法線分布関数

---

$$\int_{\Omega^\pm} D_{std}(\omega_m) \langle \omega_m, \omega_n \rangle d\omega_m = \int_{\Omega^\pm} \frac{1}{\pi} \langle \omega_m, \omega_n \rangle d\omega_m = 1$$

(これはLambertianを思い起こさせる)

CEDEC2023

余談になりますが、標準法線分布関数の確率密度関数は完全拡散反射面のLambertianのBRDFと酷似しています。LambertianのBRDFは反射ベクトルについての式ですが、ここに示した式はマイクロファセットの面の向きに関する式となっています。

# 標準の場合の計算手順

	マイクロファセット $\omega_m$	反射/屈折したレイ $\omega_i$
サンプリング	単位円から一点を選ぶ (concentric disc map)	<ul style="list-style-type: none"><li>点を半球面に投影</li><li>反射/屈折の計算</li></ul>
PDF	$\frac{1}{\pi}$	Jacobianを2つかける

CEDEC2023

標準の場合についてサンプリングとPDFの計算をまとめます。サンプリングは次のように行います。まず、単位円からランダムに1点を選び、それを半球面に投影して向きを求めます。それを用いて反射あるいは屈折したベクトルを求めて完了です。必要となるJacobianは、単位円から半球面に変換するのに必要なJacobianと、求めた向きで反射/屈折を計算するときに必要となるJacobianの2つです。

# Jacobian ①投影

---

⚠ 標準法線分布関数  $D_{std}$  で  
生成した方向を  $\omega_{std}$  と呼ぶ

CEDEC2023

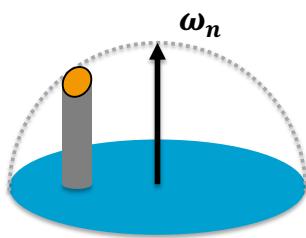
ここからは説明の上で明確に区別するため、標準法線分布関数  $D_{std}$  で  
生成した向きを  $\omega_{std}$  と呼び、一般の場合の  $D$  で生成された向きに相当する  
ものを引き続き  $\omega_m$  と呼ぶことにします。

# Jacobian ①投影

単位円上のサンプルから球面上のサンプルへ

$$\int_{Disc} D_{std}(\omega_{std}) dA = \int_{\Omega^\pm} D_{std}(\omega_{std}) \left\| \frac{dA}{d\omega_{std}} \right\| d\omega_{std}$$

Jacobian

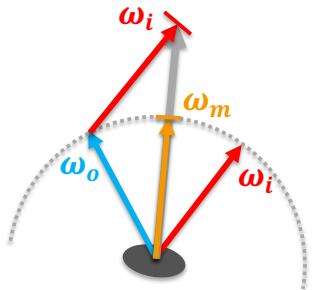


$$dA = d\omega_{std} \langle \omega_{std}, \omega_n \rangle$$

CEDEC2023

先ほども話したように  $\omega_{std}$  は、単位円を一様にサンプリングして、生成した点を半球面に投影することで、方向に変換することで得られます。非常に簡単ですね。今、半球面に投影するという処理が入ったので、変数変換を考慮するための Jacobian が必要になりますが、これは面積と立体角の関係から簡単に求まります。

## Jacobian ②反射



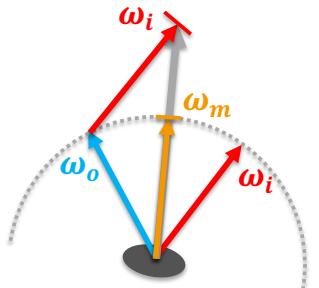
$\omega_m$ から反射/屈折したベクトルへ

$$\omega_i = 2(\omega_m \cdot \omega_o)\omega_m - \omega_o$$

CEDEC2023

ここで得られた $\omega_m$ はマイクロファセットの向きですので、そのままレンダリングには使えません。入射したレイを、マイクロファセット面で反射あるいは屈折させてあげる必要があります。まずは反射させる場合から見ていきましょう。反射した向きはこのように単純な計算で求まります。

## Jacobian ②反射



反射/屈折したベクトルのPDFが欲しい

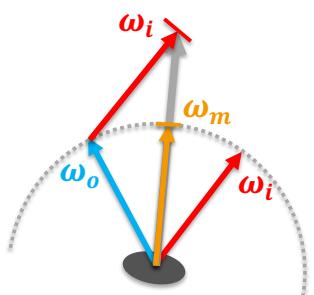
$$\int_{\Omega^\pm} D(\omega_m) \langle \omega_m, \omega_n \rangle \left\| \frac{d\omega_m}{d\omega_i} \right\| d\omega_i = 1$$

CEDEC2023

$\omega_m$ が動いた場合当然反射したベクトルも動きますが、動く量というの  
は同じではありません。それを加味するために確率密度関数の計算で  
は、色がついている部分つまりJacobianが必要となります。

## Jacobian ②反射

マイクロファセットで反射したベクトル



$$\omega_m = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}$$

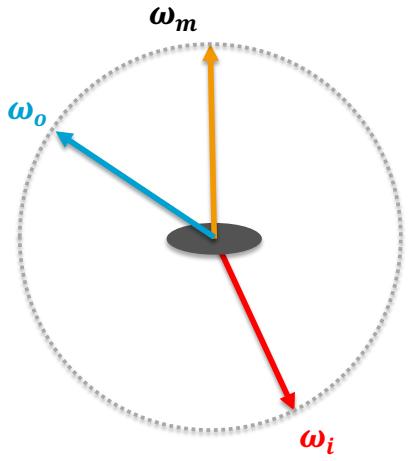
$$d\omega_m = \frac{|\omega_m \cdot \omega_o|}{\|\omega_i + \omega_o\|^2} d\omega_i = \frac{1}{4|\omega_m \cdot \omega_i|} d\omega_i$$

$$\left\| \frac{d\omega_m}{d\omega_i} \right\| = \frac{1}{4|\omega_m \cdot \omega_i|}$$

CEDEC2023

それは立体角の関係からこのように求めることができます。詳しい証明はWalterの2007年の論文にあるのでそちらを参考にしてください。

## Jacobian ②屈折



$$\omega_i = \frac{\omega_m (\omega_m \cdot \omega_o - \sqrt{\eta^2 - (1 - (\omega_m \cdot \omega_o)^2)}) - \omega_o}{\|\omega_m (\omega_m \cdot \omega_o - \sqrt{\eta^2 - (1 - (\omega_m \cdot \omega_o)^2)}) - \omega_o\|}$$

$$\left\| \frac{d\omega_m}{d\omega_i} \right\| = \frac{\eta^2 |\omega_m \cdot \omega_i|}{((\omega_m \cdot \omega_o) + \eta(\omega_m \cdot \omega_i))^2}$$

( $\eta$ は2つの媒質の相対屈折率)

CEDEC2023

屈折についても証明は省略します。反射ないしは屈折する場合の Jacobian のどちらかと、先ほどの半球面に投影する分の Jacobian を  $\frac{1}{\pi}$  にかけることで PDF を求めることができます。

## 一般の場合の計算手順

---

$D_{std}$ は等方性で  $\alpha = 1$  を仮定

$D_{std}$  でサンプリングした  $\omega_{std}$   
を任意の  $(\alpha_x, \alpha_y)$  で使いたい

( $\alpha_x$  と  $\alpha_y$  は縦方向と横方向での粗さ)

CEDEC2023

ここまで標準の場合、 $\alpha = 1$  の場合についての話だったので、一般的な場合には今お話しした方法と同じ方法で求められないであれば意味がないのでは?と思うかもしれません。しかし、実は一般的な場合もほとんど同じ手順でマイクロファセットの面の向きを得ることができます。

# 一般の場合の計算手順

---

一般の場合はどうすれば良いか

$$(1, 1) \neq (\alpha_x, \alpha_y)$$

CEDEC2023

では、一般の場合 $\alpha \neq 1$ の時はどうすれば良いでしょうか。サンプリングは向きに何か変換をかまえば良いので簡単にできそうですが、確率密度関数の計算は少し難しそうですね。

# 一般の場合の計算手順

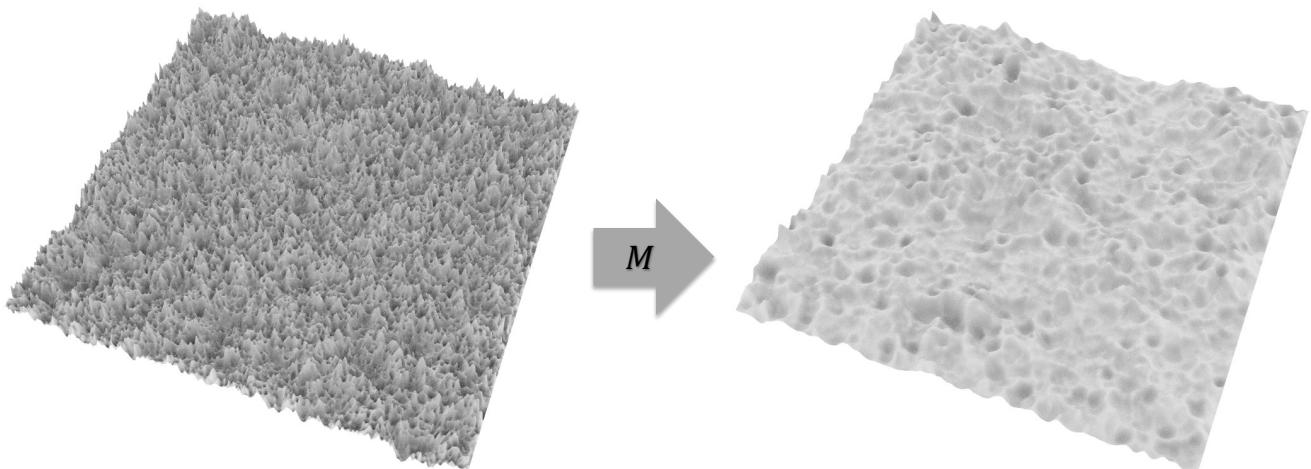
	マイクロファセット $\omega_m$	反射/屈折したレイ $\omega_i$
サンプリング	単位円から一点を選ぶ (concentric disc map)	<ul style="list-style-type: none"><li>点を半球面に投影</li><li>微小面の向きに線形変換</li><li>反射/屈折の計算</li></ul>
PDF	$\frac{1}{\pi}$	Jacobianを3つかける

CEDEC2023

先ほどと同様に、一般の場合の手順を表にまとめます。サンプリングは標準の場合とほぼ同じで、まず単位円からランダムに1点を選び、それを半球面に投影して向きを求めます。次に任意の $\alpha$ に対応させるため、生成した方向ベクトルに線形変換を施します。それを用いて反射あるいは屈折したベクトルを生成して完了です。PDFの計算も標準の場合と大きく変わらず、生成したベクトルに施した線形変換に伴って、PDFを補正するため追加のJacobianが1つ必要となるだけです。

## 一般の場合の計算手順

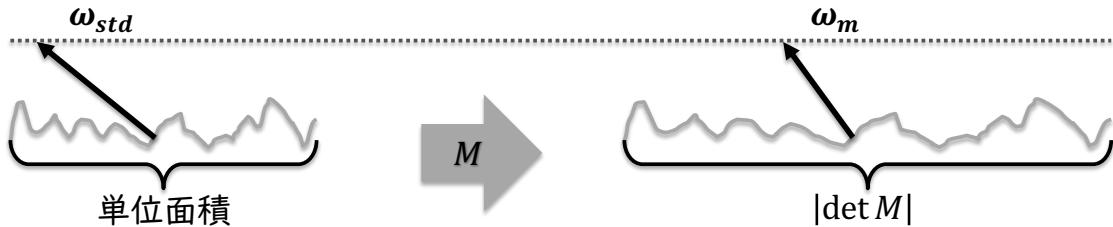
---



CEDEC2023

まず引き延ばしについて見てみましょう。左のような凹凸を右のような少し滑らかなものに変換することを考えます。

## 一般の場合の計算手順



$$M = \begin{pmatrix} \frac{1}{\alpha_x} & * & \mathbf{0} \\ * & \frac{1}{\alpha_y} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix}$$

CEDEC2023

線形変換 $M$ によって凹凸のある面を引き伸ばすと、マクロサーフェスの面積は $|\det M|$ 倍になります。引き伸ばされることで滑らかになるので光沢が増します。変換 $M$ はマクロサーフェスに沿って行われるので、Z成分には作用せず、ベクトルのZ成分も当然ながら変化しません。

## 一般の場合の計算手順

---

マイクロファセット	変換	逆変換
ジオメトリ	$M$	$M^{-1}$
$\omega_m$ (法線) covector	$M^{-T}$	$M^T$

CEDEC2023

引き延ばされるのはマイクロサーフェスのジオメトリで、行列Mはジオメトリに対する変換を表します。 $\omega_m$ はcovector、ここでは法線であるので、 $D_{std}$ から生成した $\omega_{std}$ にはMの逆転置行列をかけてあげる必要があります。

# Jacobian ③引き伸ばし

the micro-normal  $\mathbf{m} = (m_x, m_y, m_z)$  is  $M^T \mathbf{m} = (am_x + bm_y, cm_y, dm_y, m_z)$  and has length

$$\|M^T \mathbf{m}\| = \sqrt{(am_x + bm_y)^2 + (cm_y)^2 + m_z^2}. \quad (14)$$

Note that the microfacet distribution is two-dimensional: the normal  $\mathbf{m}$  is on the hemisphere, and can be represented by the first two components  $m_x$  and  $m_y$ . The  $z$  component is projected on the hemisphere  $m_z = \sqrt{1 - m_x^2 - m_y^2}$ . In order to find a new distribution  $D_M$  by transforming the argument of  $D$  with  $N$  we need to normalize by the absolute value of the Jacobian determinant of  $N$  [PHH16]

$$D_M(\mathbf{m}) = |\det J_N| D(\mathbf{u}), \quad (15)$$

where

$$\det J_N = \begin{vmatrix} \frac{\partial u_x}{\partial m_x} & \frac{\partial u_x}{\partial m_y} \\ \frac{\partial u_y}{\partial m_x} & \frac{\partial u_y}{\partial m_y} \end{vmatrix} = \frac{\partial u_x}{\partial m_x} \frac{\partial u_y}{\partial m_y} - \frac{\partial u_x}{\partial m_y} \frac{\partial u_y}{\partial m_x}. \quad (16)$$

We proceed to compute the partial derivatives

$$\frac{\partial u_x}{\partial m_x} = \frac{m_y^2(ad^2 - bcd - acd + b - bc^2) + a}{\|M^T \mathbf{m}\|^3} \quad (17)$$

$$\frac{\partial u_x}{\partial m_y} = \frac{m_x^2(bc^2 - acd - b) + m_x m_y(bcd + a - ad^2) + b}{\|M^T \mathbf{m}\|^3} \quad (18)$$

$$\frac{\partial u_y}{\partial m_x} = \frac{m_y^2(cb^2 - abd - c) + m_x m_y(abc + d - da^2) + c}{\|M^T \mathbf{m}\|^3} \quad (19)$$

$$\frac{\partial u_y}{\partial m_y} = \frac{m_x^2(da^2 - abc - d) + m_x m_y(abd + c - cb^2) + d}{\|M^T \mathbf{m}\|^3}. \quad (20)$$

After simplification of Equation (16) using Mathematica [Wol16] we arrive at a concise result

$$\det J_N = \frac{ad - bc}{\|M^T \mathbf{m}\|^4} = \frac{|\det M|}{\|M^T \mathbf{m}\|^4}. \quad (21)$$

We use this result in Equation (15) to obtain the microfacet distribution of the transformed microsurface

$$D_M(\mathbf{m}) = \frac{|\det M|}{\|M^T \mathbf{m}\|^3} D(\mathbf{u}). \quad (22)$$

画像は “Microsurface Transformations” より

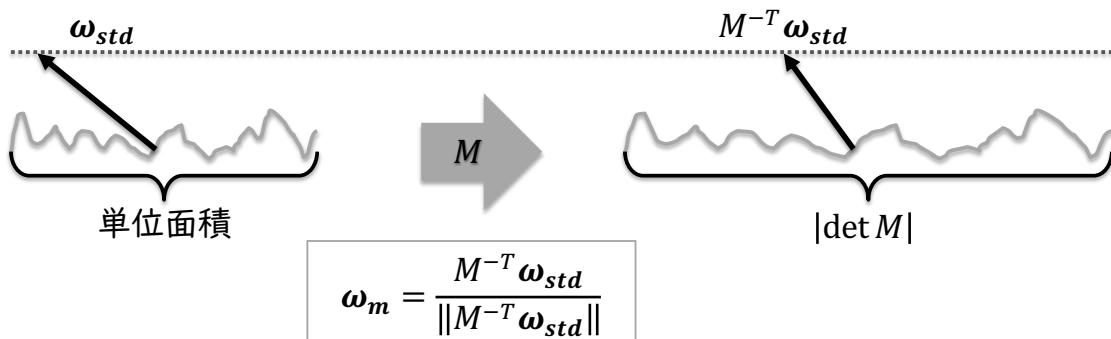
$\omega_{std}$  と  $\omega_m$  の関係は

$$D(\omega_m) = \frac{|\det M|}{\|M^T \omega_m\|^4} D_{std}(\omega_{std})$$

サンプリングの追加処理は  $M^{-T}$  をかけるだけであり、いたって単純です。一方、PDFは少し複雑です。ここからは引き延ばしによって必要となるJacobianを詳しく見てていきましょう。まず  $D_{std}$  で生成したベクトル  $\omega_{std}$  と  $\omega_m$  の関係を知る必要があるわけですが、Microsurface Transformationsの論文によると、一般の場合の  $D$  と標準の場合の  $D_{std}$  の関係はこのように表されます。論文出は左の画像のように非常にややこしい式を経由してこの結果を得ています。分子の  $|\det M|$  については、マイクロサーフェスが引き伸ばされて面積が  $|\det M|$  に変わるので出所が想像できますが、分母については一見しただけではどういうものか分かりません。

## Jacobian ③引き伸ばし

$$\int_{\Omega^\pm} D_{std}(\omega_{std}) \langle \omega_{std}, \omega_n \rangle d\omega_{std} = \int_{\Omega^\pm} D_{std}(\omega_{std}) \langle \omega_{std}, \omega_n \rangle \left\| \frac{d\omega_{std}}{d\omega_m} \right\| d\omega_m$$

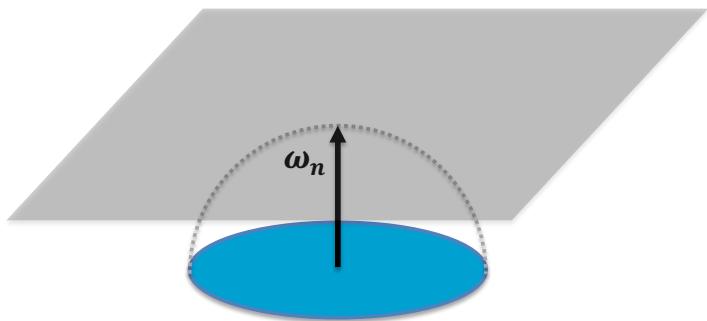


CEDEC2023

標準法線分布関数からサンプリングしたベクトルを線形変換しますが、ここでも変数変換を行なってみましょう。先ほど引き延ばしによるベクトルの変化はマクロサーフェスに沿って行われ、マクロサーフェスの法線方向の成分は変化しない述べました。ですので、引き延ばしによる立体角の変化を考えるには、スロープスペースを経由すると分かりやすくなります。 $\omega_m$ は一番下にあるように引き伸ばしてから正規化します。

## Jacobian ③引き伸ばし

---

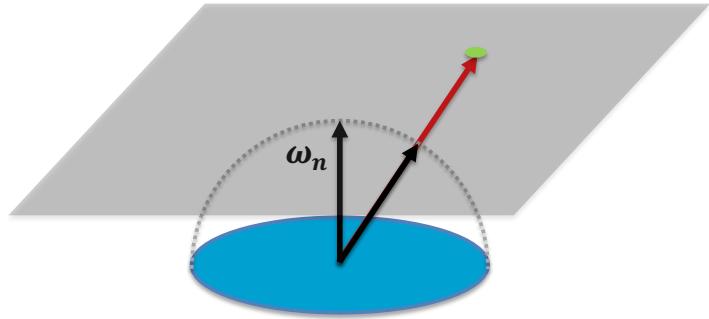


CEDEC2023

スロープスペースというと難しそうですが、そんなことはありません。  
。半径 1 の半球に無限に続く平面が乗っている状況を思い浮かべてください。

## Jacobian ③引き伸ばし

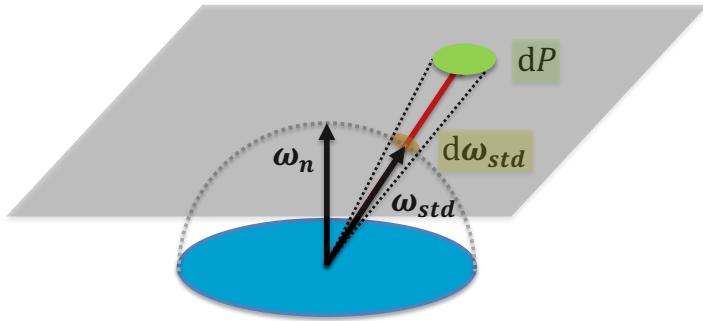
---



CEDEC2023

方向ベクトルを、この平面上の点の位置によって表現するのに使います。

## Jacobian ③引き伸ばし



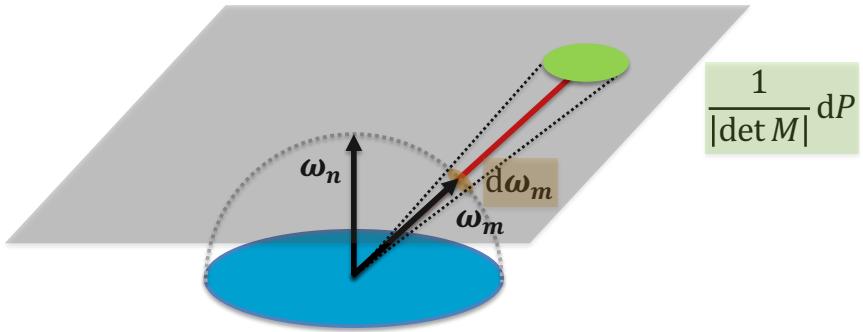
微小面積  $dP$  と微小立体角  $d\omega_{std}$  の関係から

$$dP \langle \omega_{std}, \omega_n \rangle \frac{1}{\frac{1}{\langle \omega_{std}, \omega_n \rangle^2}} = dP \langle \omega_{std}, \omega_n \rangle^3 = d\omega_{std}$$

CEDEC2023

ここで平面上の微小な面積  $dP$  と微小な立体角  $d\omega_{std}$  の関係は内積の3条を使って表すことができます。 $dP$  が表す立体角を求めていくのですが、面の傾きと距離の2条とを合わせると、内積の3条が出てきます。赤いベクトルは  $\omega_{std}$  が平面に当たるまで延長したもので、赤い項はそのベクトルの長さです。グレーの内積は微小面  $dP$  と  $\omega_{std}$  との向きにより求まります。

## Jacobian ③引き伸ばし



同様に微小立体角  $d\omega_m$  は引き延ばされた微小面積  $\frac{1}{|\det M|} dP$  を用いて  
$$\frac{1}{|\det M|} dP \langle \omega_m, \omega_n \rangle^3 = d\omega_m$$

CEDEC2023

同様に微小な立体角  $d\omega_m$  は引き延ばされた微小な面積  $\frac{1}{|\det M|} dP$  を用いて  
このように表すことができます。

## Jacobian ③引き伸ばし

$$\int_{\Omega^\pm} D_{std}(\omega_{std}) \langle \omega_{std}, \omega_n \rangle \left\| \frac{d\omega_{std}}{d\omega_m} \right\| d\omega_m = 1$$

微小面積  $dP$  と微小立体角  $d\omega_{std}$  の関係から

$$dP \langle \omega_{std}, \omega_n \rangle^3 = d\omega_{std}$$

$dP$  を引き延ばすので  $\omega_m$  に関しては

$$\frac{1}{|\det M|} dP \langle \omega_m, \omega_n \rangle^3 = d\omega_m$$

CEDEC2023

代入してまとめていきましょう。ここにある変数変換は  $D_{std}(\omega_{std})$  が  $\frac{1}{\pi}$  なので次のように書き換えることができます。

## Jacobian ③引き伸ばし

---

$$1 = \int_{\Omega^\pm} \frac{1}{\pi} |\det M| \frac{\langle \omega_{std}, \omega_n \rangle^3}{\langle \omega_m, \omega_n \rangle^3} \langle \omega_{std}, \omega_n \rangle d\omega_m$$

CEDEC2023

色付けしてみていましょう。

## Jacobian ③引き伸ばし

---

$$1 = \int_{\Omega^\pm} \frac{1}{\pi} |\det M| \frac{\langle \omega_{std}, \omega_n \rangle^3}{\langle \omega_m, \omega_n \rangle^3} \langle \omega_{std}, \omega_n \rangle d\omega_m$$

CEDEC2023

PDFは標準法線分布の $\frac{1}{\pi}$ 、引き伸ばしによる青色の項 $\frac{\langle \omega_{std}, \omega_n \rangle^3}{\langle \omega_m, \omega_n \rangle^3}$ の積、半球面に投影した内積 $\langle \omega_{std}, \omega_n \rangle$ 、となっています。色付けされた3つの項の積は $\omega_m$ のPDFなので、これに反射あるいは屈折の分のJacobianをかけるとPDFの導出は終了です。

## Jacobian ③引き伸ばし

---

$$D(\omega_m) = |\det M| \frac{\langle \omega_{std}, \omega_n \rangle^4}{\langle \omega_m, \omega_n \rangle^4} D_{std}(\omega_{std})$$

CEDEC2023

実は先ほどの結果はMicrosurface Transformationsの論文の導出とは少し異なっています。論文ではこういった変換がなされています。

## Jacobian ③引き伸ばし

---

$$D(\omega_m) = |\det M| \frac{\langle \omega_{std}, \omega_n \rangle^4}{\langle \omega_m, \omega_n \rangle^4} \frac{1}{\pi}$$

を

$$1 = \int_{\Omega^\pm} D(\omega_m) \langle \omega_m, \omega_n \rangle d\omega_m$$

に代入

これを最初に出てきた積分の式に代入してみましょう。

## Jacobian ③引き伸ばし

---

$$D(\omega_m) = |\det M| \frac{\langle \omega_{std}, \omega_n \rangle^4}{\langle \omega_m, \omega_n \rangle^4} \frac{1}{\pi}$$

$$1 = \int_{\Omega^\pm} \frac{1}{\pi} |\det M| \frac{\langle \omega_{std}, \omega_n \rangle^3}{\langle \omega_m, \omega_n \rangle^3} \langle \omega_{std}, \omega_n \rangle d\omega_m$$

CEDEC2023

一部キャンセルされるので、先ほど求めた結果と一致します。めでたしめでたしどういうわけです。

# マスキング関数

---

$$G_{1, std}(\omega_m, \omega_o) = \chi^+(\omega_m \cdot \omega_o) \frac{2|\omega_{o_z}|}{1 + \omega_{o_z}}$$

ベクトルに適切な変換を施せば  $G_{1, std}$  がそのまま使える

$$G_1(\omega_m, \omega_o) = G_{1, std}\left(\frac{\mathbf{M}^T \omega_m}{\|\mathbf{M}^T \omega_m\|}, \frac{\mathbf{M}^{-1} \omega_o}{\|\mathbf{M}^{-1} \omega_o\|}\right)$$

- 線形変換変換を受けても遮られる部分は変化しない
  - 見えていた場所は見えたまま
  - 影になっていた部分は影のまま

CEDEC2023

マスキング関数については粗さが 1 の時の標準のマスキング関数をそのまま使うことができます。これはマイクロファセットに線形変換を施したとしても、見ている方向についても適切な変換を施してあげれば、全てが単純に引き伸ばされただけであるので、見えていた部分は見えているまま、影になっていた部分は影のままであるためです。

# 一般の場合の計算手順

---

- サンプリング
  - 単位円内的一点を(Concentric mapを用いて)選ぶ
  - 点を半球面に投影して方向に変えて  $\omega_{std}$ を得る
  - $\omega_{std}$ に $M^{-T}$ をかけて  $\omega_m$ を得る
- PDF
  - $\frac{1}{\pi}$ に3つのJacobianをかける
    - 半球面に投影した分:  $\langle \omega_{std}, \omega_n \rangle$
    - $M^{-T}$ をかけた分:  $|\det M| \frac{\langle \omega_{std}, \omega_n \rangle^3}{\langle \omega_m, \omega_n \rangle^3}$
    - 反射/屈折の分:  $\frac{1}{4|\omega_m \cdot \omega_o|}$  など

CEDEC2023

サンプリングの手法と、PDFの計算をまとめてみます。とても簡単ですね。

# 一般の場合の計算手順



$$M = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

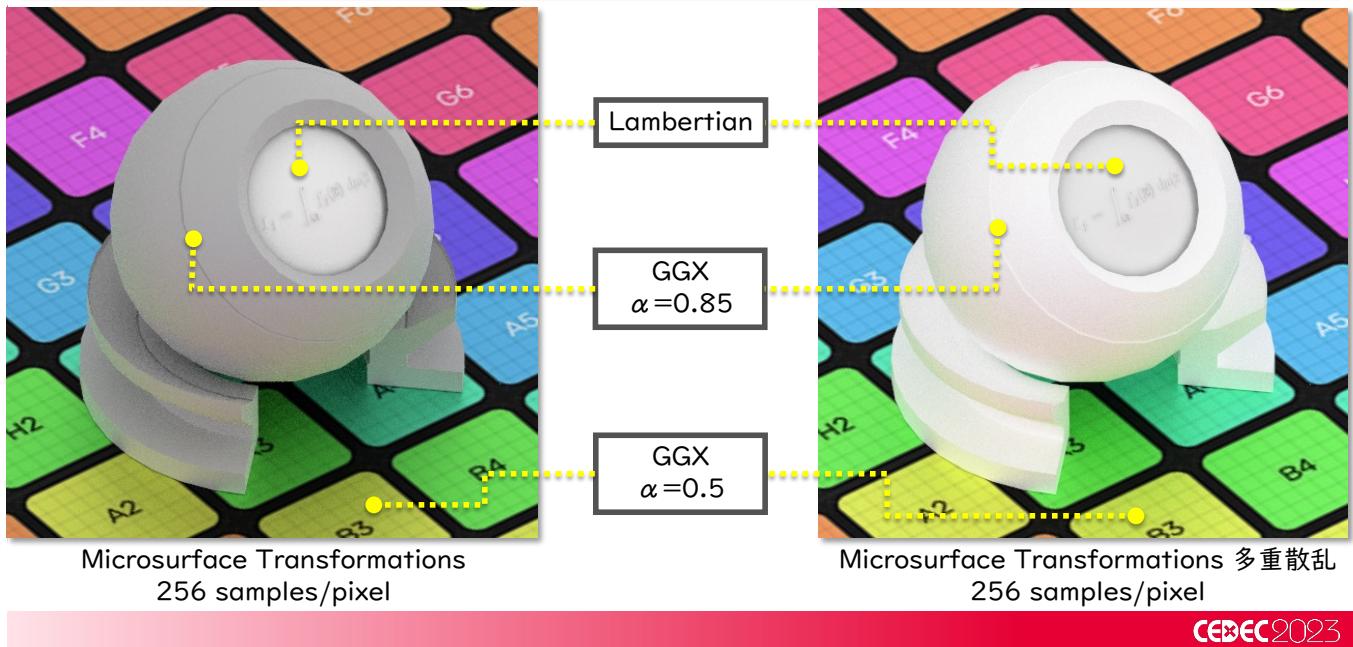
$$M = \begin{pmatrix} 10 & \mathbf{20} & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

CEDEC2023

計算は理解しやすくなるものの、実際に高速化につながるわけではありません。ただ、理解しやすい以外にも実用上のメリットはあります。線形変換には対角成分以外に値があってもよく、メッシュの変形に追従して光沢を変化させることができます。

# 一般の場合の計算手順



今回は紹介しませんでしたが、多重散乱も粗さ1の標準法線分布関数や標準マスキング関数を使って行えるので実装が簡単になります。粗さが大きい場合でも、右のように暗くなることなく自然な結果がえられます。

---

---

最後に

# 最後に

---

- マイクロファセット、NDF、VNDF、マスキング関数について復習した
- Microsurface Transformationsを用いると、異方性の場合や、Skewがかかった場合でも、GGXのNDFのサンプリングが容易になり、PDFも $\frac{1}{\pi}$ にJacobianを数個かけるだけで簡単に求められることを示した
- 論文では複雑な式が用いられていることが多いが、変数変換を施しわかりやすい空間に持ってくることで、簡略化できる

CEDEC2023

私のパートではGGXを中心にサンプリングとPDFの計算についてまとめ、論文のように複雑な計算をしなくとも同じ結果に辿り着けることをいくつかの例で示しました。論文を読んでそのまま理解するのは第一歩で、咀嚼するとその先に必ず大切な何かが見つかります。それは、本質的な疑問であったり、簡略化であったり、新しい手法であったりします。検索をするとBRDFについての解説記事は山ほど見つかりますが、できれば査読を経た論文をしっかりと読んで、上辺だけではない知識をしっかりと身につけて行ってください。