

# Rendu PBR

Sébastien Beugnon

1 décembre 2022

**Sébastien Beugnon**

*R&D Researcher*

mail : sebastien.beugnon@emersya.com

Github : @sbeugnon



# Sommaire

Rappels

Théorie de la lumière

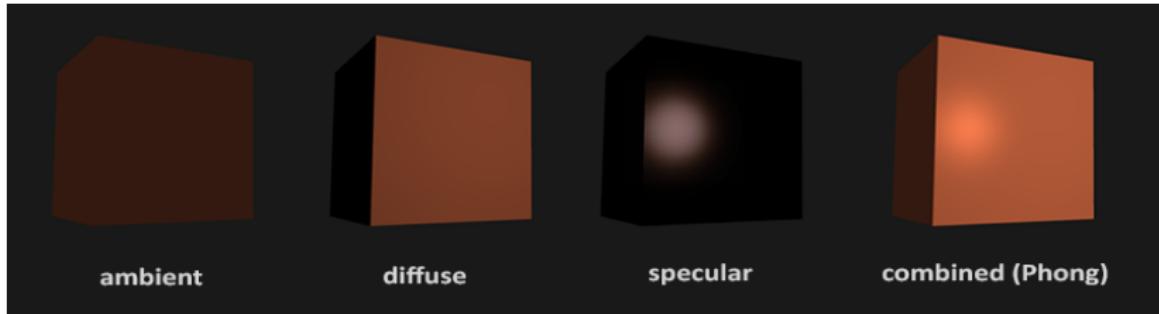
PBR

TP

## Rappels

## Modèle Blinn-Phong

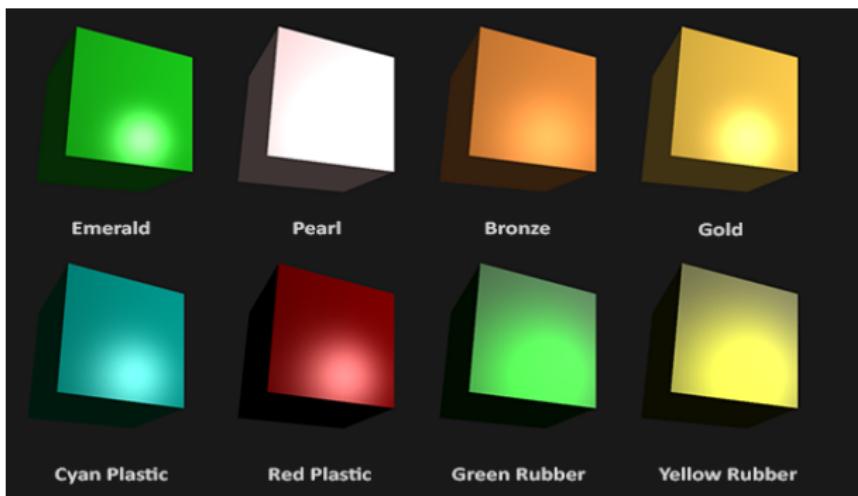
- ▶ Ambiant (*Ambient*)
  - ▶ Diffuse (*Diffuse*)
  - ▶ Spéculaire (*Specular + Shininess*)



## Limites du modèle Phong

## Limitations

- ▶ Matériaux simples (plastiques, métaux)
  - ▶ pas de réelle réflexion
  - ▶ Seulement 3 paramètres (plus si on utilise des textures)



## Shaders : Phong

## ► Vertex Shader

```
#version 330 core
layout(location = 0) in vec3 aPosition;
layout(location = 1) in vec3 aNormal;
layout(location = 2) in vec2 aUv0;

uniform mat4 projection;
uniform mat4 view;
uniform mat4 model;
uniform vec3 lightPos;

out vec3 oFragPos;
out vec3 oFragNormal;
out vec2 oFragUV0;
out vec3 oLightPos;

void main() {
    oFragPos = vec3(model * vec4(aPosition, 1.0f));
    gl_Position = projection * view * vec4(oFragPos, 1.0);

    mat4 normalMatrix = mat3(transpose(inverse(model)));
    oFragUV0 = aUv0;
    oFragNormal = normalMatrix * aNormal;
    oLightPos = vec3(view * vec4(lightPos, 1.0));
}
```

## Shaders : Phong

## ► Fragment Shader

```
#version 330 core
in vec3 oFragPos;
in vec3 oFragNormal;
in vec2 oFragUV0;
in vec3 oLightPos;

out vec4 FragColor;
// Punctual Light
uniform vec3 lightColor;
// Material
struct Material {
    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
    float shininess;
};
uniform Material material;
// ...
```

## Shaders : Phong

## ► Fragment Shader

```
// ...
void main()
{
    // ambient
    vec3 ambient = light.color * material.ambient;
    // diffuse
    vec3 norm = normalize(oFragNormal);
    vec3 lightDir = normalize(oLightPos - oFragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = lightColor * (diff * material.diffuse);

    // specular
    vec3 viewDir = normalize(-FragPos);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
    vec3 specular = lightColor * (spec * material.specular);

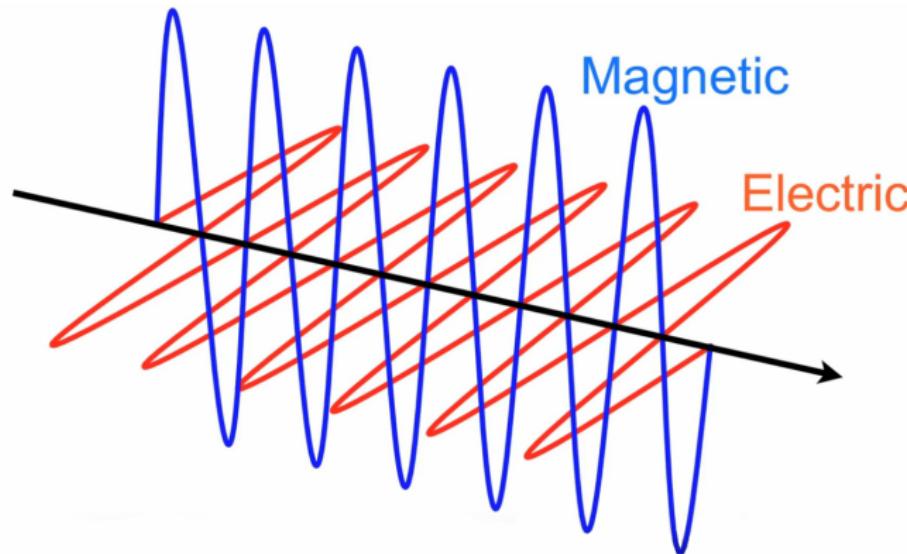
    vec3 result = ambient + diffuse + specular;
    FragColor = vec4(result, 1.0);
}
```

## Physique de la lumière

- ### ► C'est quoi la lumière ?

# Physique de la lumière

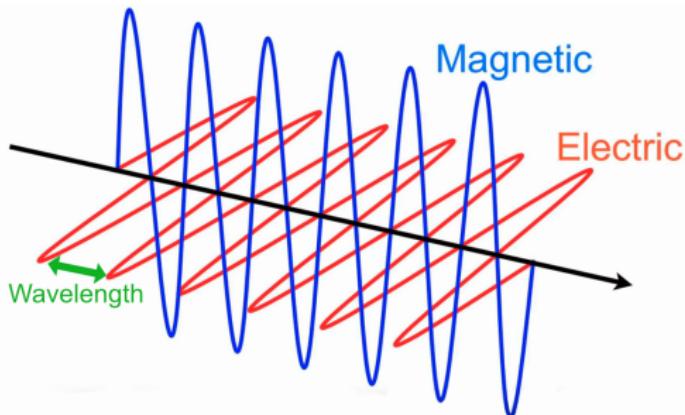
- ▶ C'est quoi la lumière ?
  - ▶ Une onde électro-magnétique



T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, S. Hillaire  
Real-Time Rendering, Fourth Edition.  
CRC Press, 2018

# Physique de la lumière

- ▶ C'est quoi la lumière ?
  - ▶ Une onde électro-magnétique



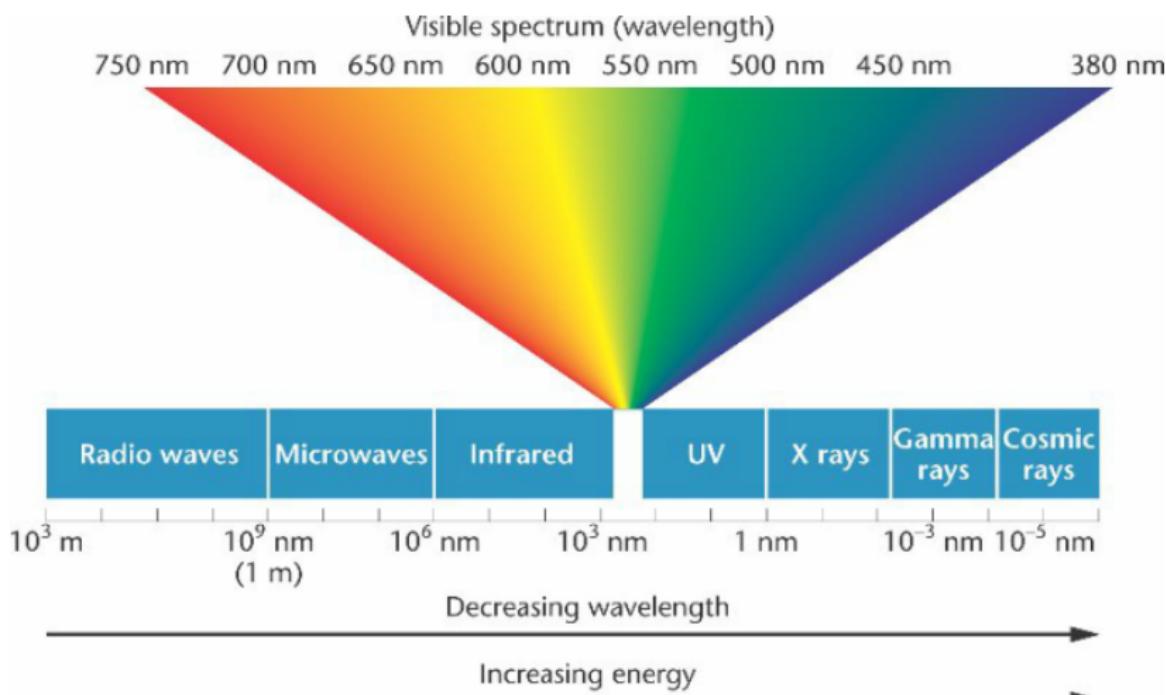
## Définition

- ▶ Onde transversale
- ▶ Fréquence
- ▶ Longueur d'onde  $\lambda$  (nm)
- ▶ Irradiance (Energie)



T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, S. Hillaire  
Real-Time Rendering, Fourth Edition.  
CRC Press, 2018

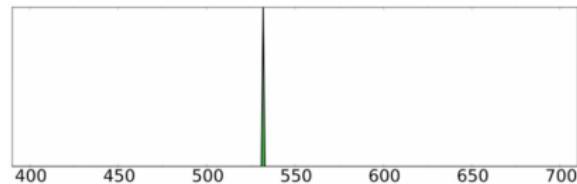
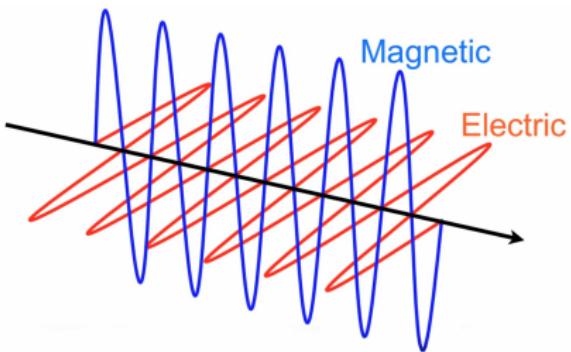
# Physique de la lumière



# Physique de la lumière

## Laser

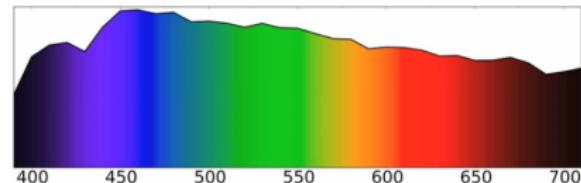
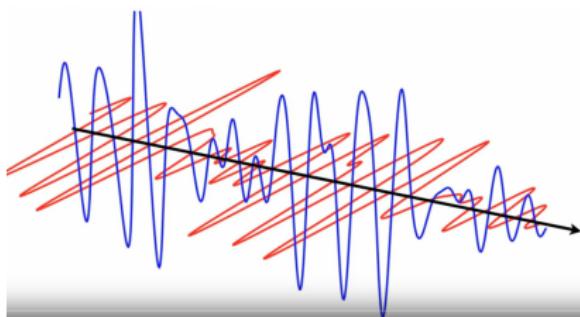
- ▶ Onde monochromatique = Belle sinusoïde
- ▶ Profil spectrale discret (Spectrum profile)



# Physique de la lumière

## Lumière blanche

- ▶ Onde bruitée
- ▶ Profil spectral continu (Spectrum profile)



# Physique de la lumière

## ► Comment la lumière est émise ?

### Évènement physique

Une onde lumineuse (*lightwave*) est émise lorsque les charges électriques d'un objet oscillent.

## ► Comment les charges oscillent ?

### Évènement physique : Émission

Conversion d'une énergie (chaleur, électrique, chimique)

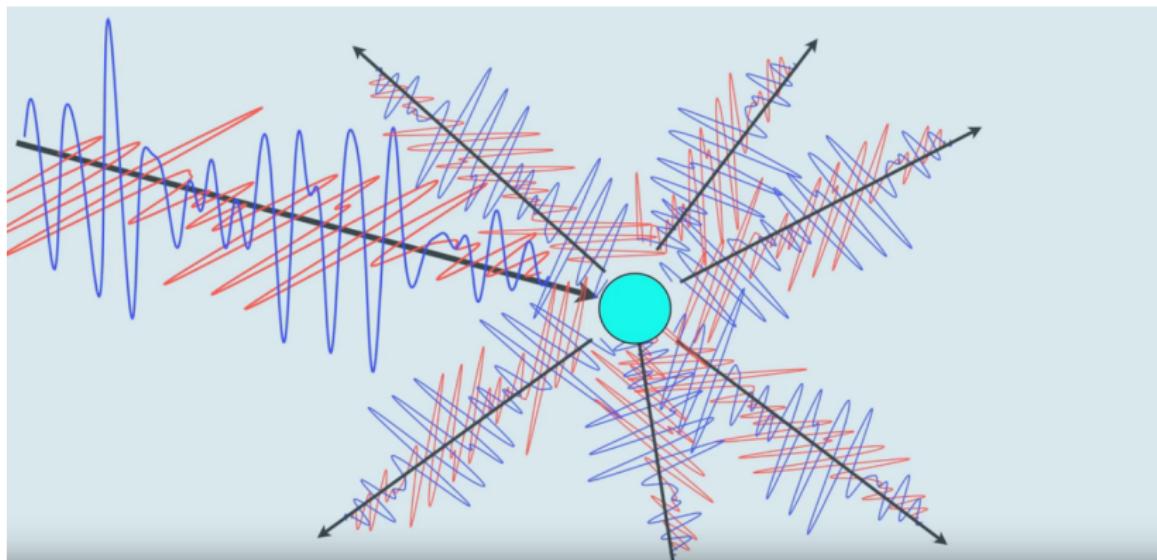
Une partie est convertie en énergie lumineuse et rayonne de l'objet

## ► Un objet *émissif* est considéré comme une source lumineuse (*light source*) dans le domaine du rendu.

# Particule

## Scattering (Diffusion, Dispersion)

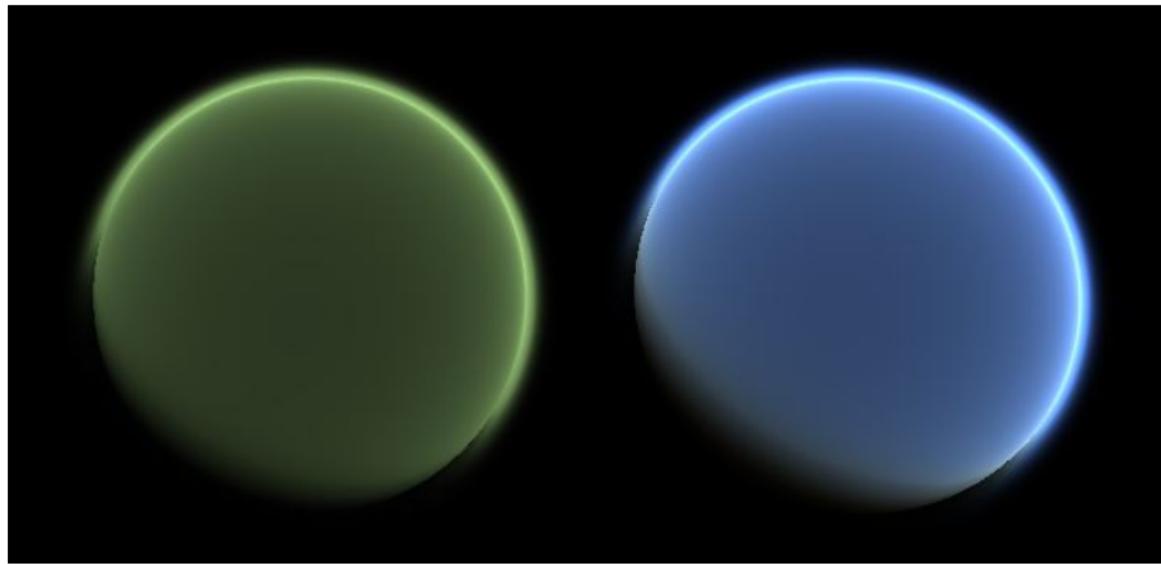
### ► Interaction matière-lumière



# Particule

## Diffusion de Rayleigh (*Rayleigh Scattering*)

- ▶ Ciel (*Atmospheric Scattering*)



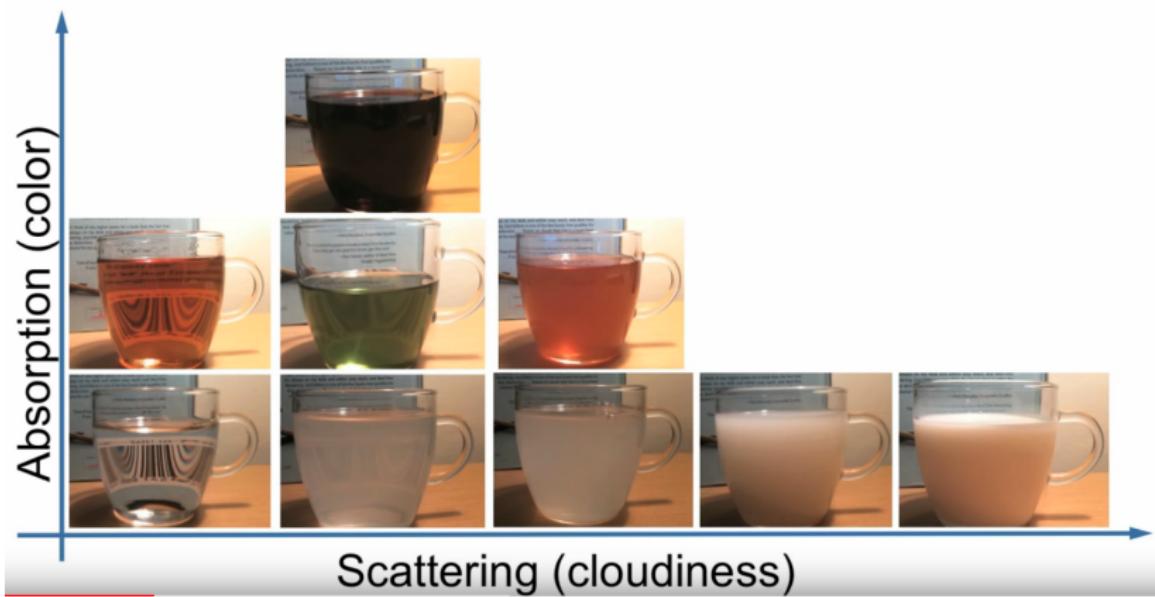
# Particule

## Diffusion de Mie (*Mie Scattering*)

► *Nuages*



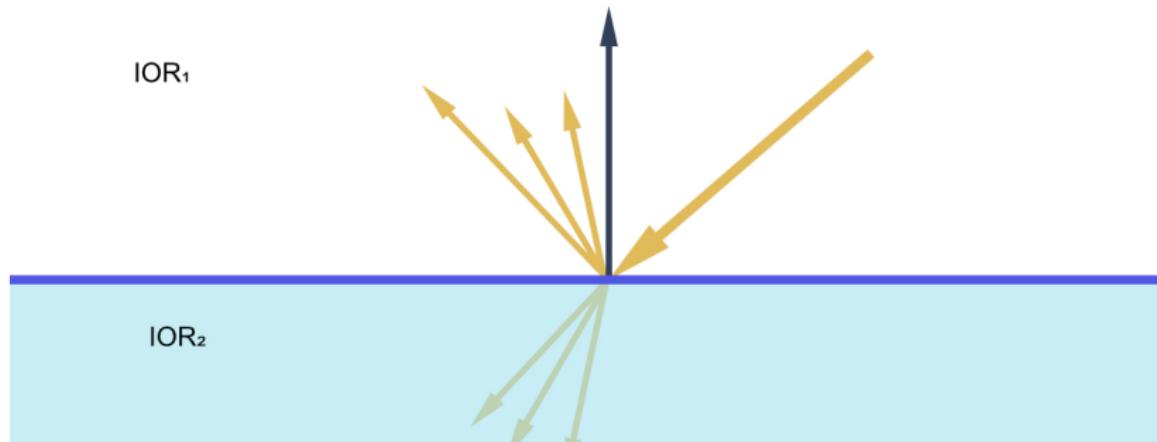
## Support



# Surface

## 2 composantes de lumière

- ▶ Réflexion (*Reflection*) : le changement de direction de la lumière retournant dans le milieu initial
- ▶ Réfraction (*Refraction*) : le changement de direction de la lumière se déplaçant au sein du nouveau milieu



# Rugosité de la surface

La surface n'est pas forcément un miroir parfait.

## 3 niveaux

- ▶ Macro-géométrie (Visible à l'oeil nu)
- ▶ Micro-géométrie (Au niveau microscopique)
- ▶ Nano-géométrie (Au niveau de la longueur d'onde)

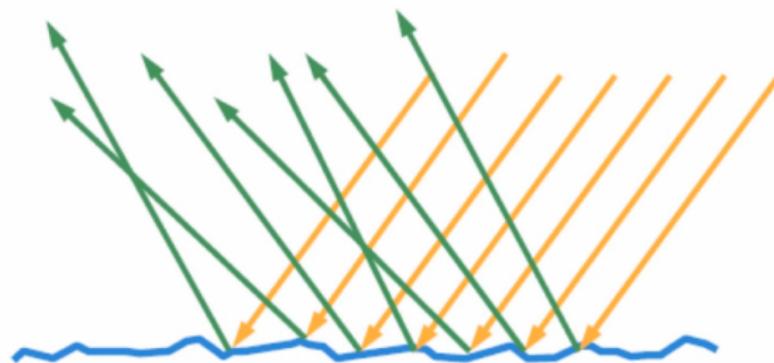


# Rugosité de la surface

La surface n'est pas forcément un miroir parfait.

## 3 niveaux

- ▶ Macro-géométrie (Visible à l'oeil nu)
- ▶ Micro-géométrie (Au niveau microscopique)
- ▶ Nano-géométrie (Au niveau de la longueur d'onde)

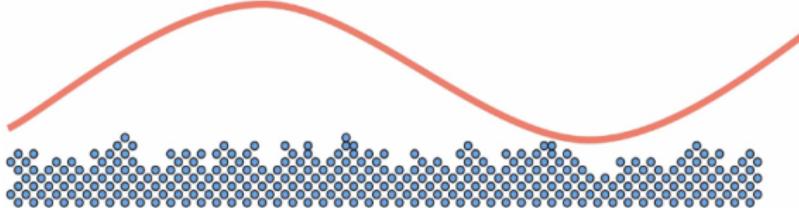


# Rugosité de la surface

La surface n'est pas forcément un miroir parfait.

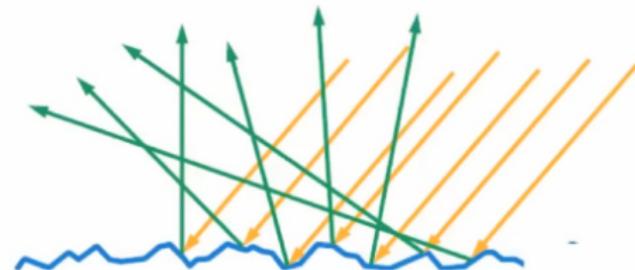
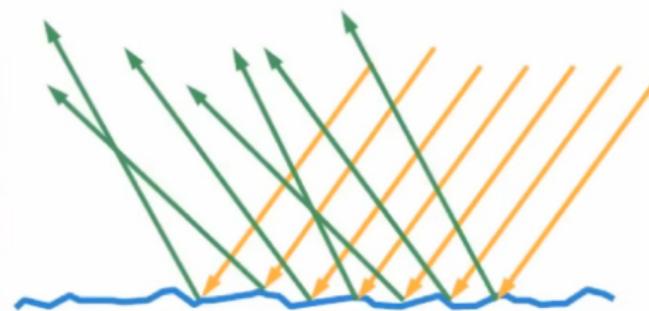
## 3 niveaux

- ▶ Macro-géométrie (Visible à l'oeil nu)
- ▶ Micro-géométrie (Au niveau microscopique)
- ▶ Nano-géométrie (Au niveau de la longueur d'onde)



# Rugosité de la surface

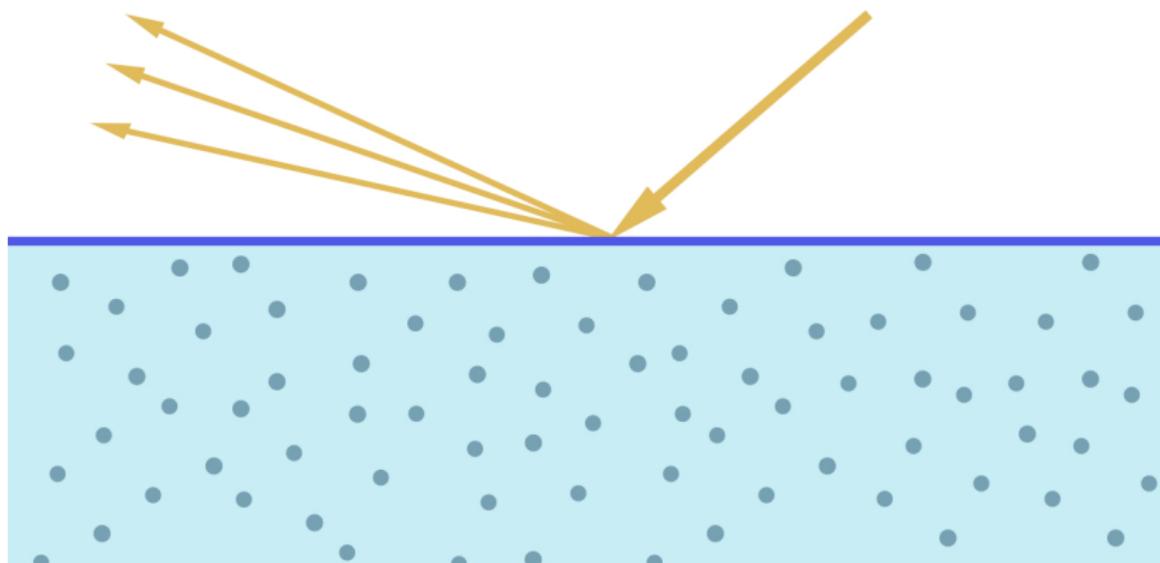
+ Rugueux = + Réflexions flous



# Diffusion sous la surface

## *Subsurface Scattering*

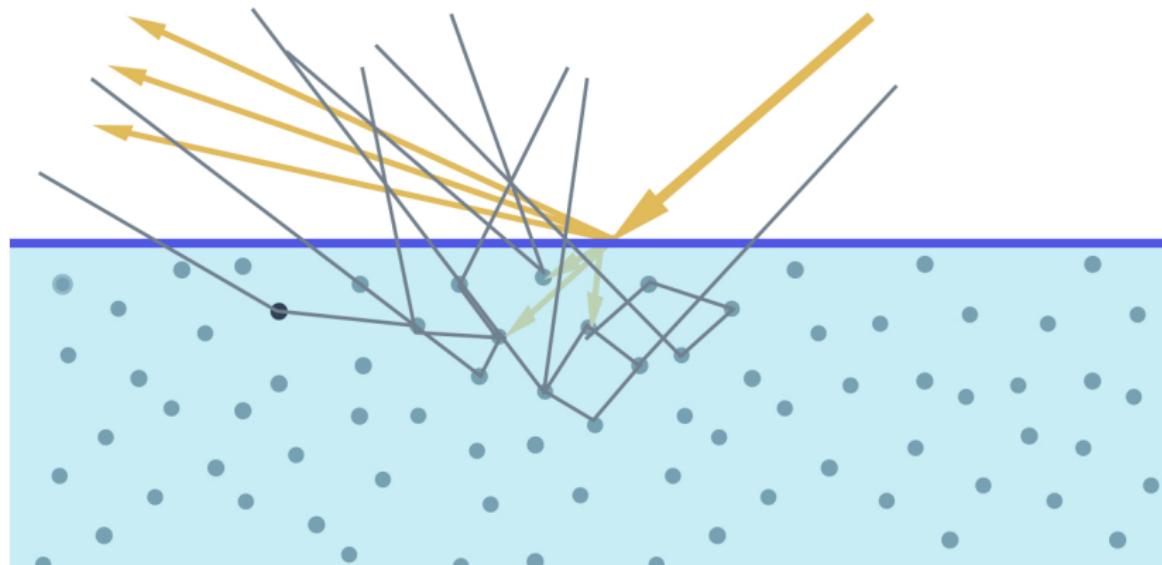
- ▶ Diffusion de particules puis changement ( $\neq$  Transmission)



# Diffusion sous la surface

## *Subsurface Scattering*

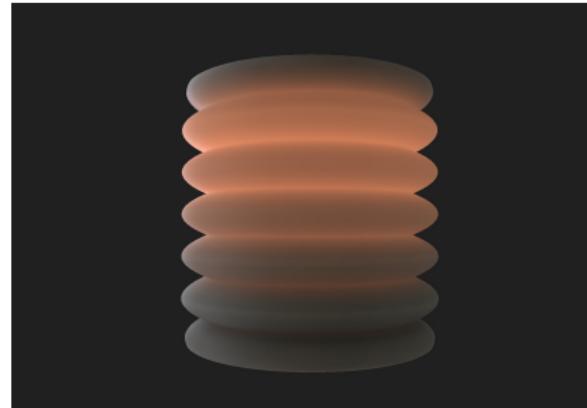
- ▶ Diffusion de particules puis changement ( $\neq$  Transmission)



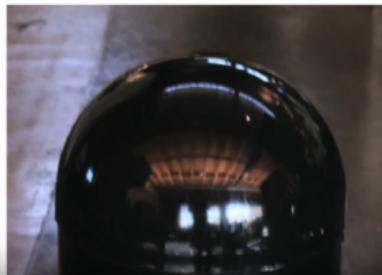
# Diffusion sous la surface

## *Subsurface Scattering*

- ▶ Diffusion de particules puis changement ( $\neq$  Transmission)



# Distinctions des matières



SIGGRAPH 2015 

# Distinctions des matières

D'un point de vue optique :

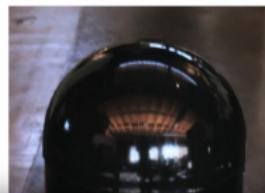
- ▶ Métaux (Conducteurs)
- ▶ Diélectriques (Isolateurs)
- ▶ Semi-conducteurs



# Distinctions des matières

D'un point de vue de rendu :

- ▶ Métaux
- ▶ Non-métaux

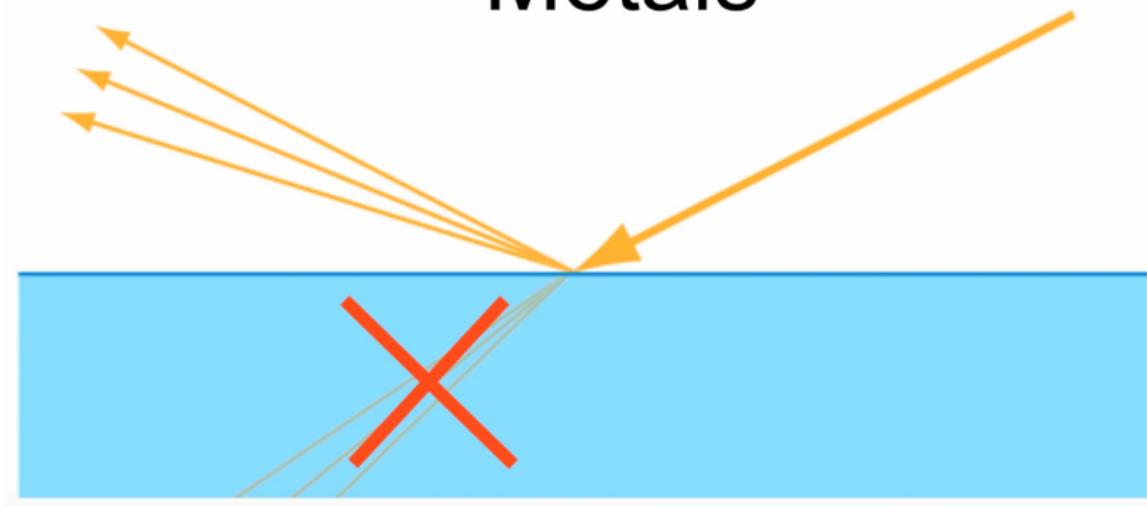


SIGGRAPH 2015

# Métaux

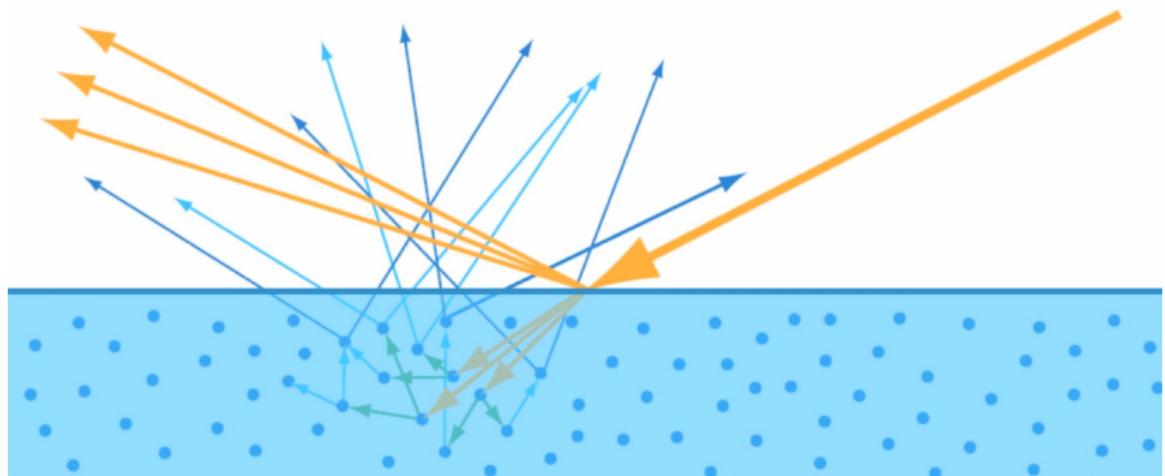
- ▶ Absorbe ou Réfléchit la lumière

## Metals

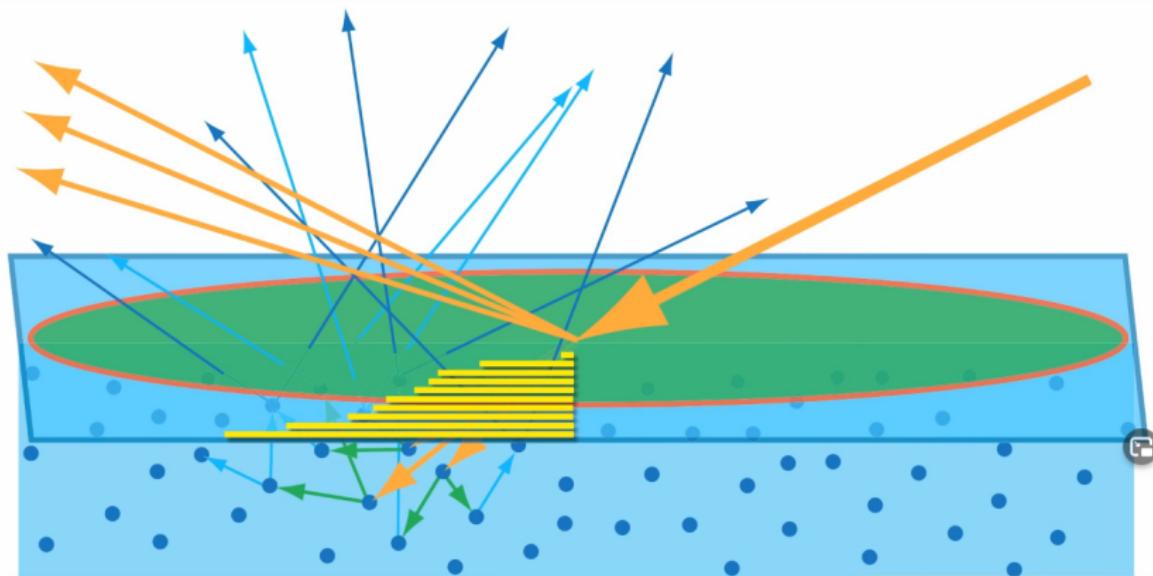


Non-métaux

# Non-Metals



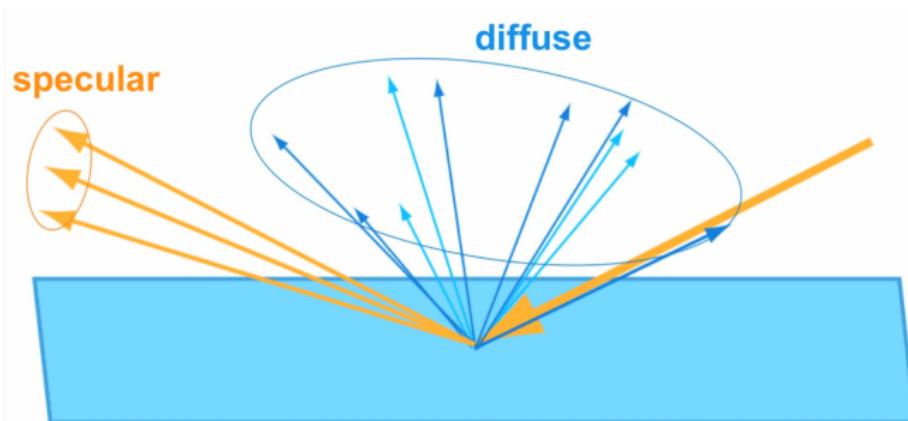
# Difficulté de modélisation



# Simplification du modèle

## 2 composantes

- ▶ Diffusion (Diffuse)
- ▶ Spéculaire (Specular)



# Physically-based rendering

## Usages

- ▶ Infographie (architecture, commerce)
- ▶ Films d'animation
- ▶ Jeux vidéo



# Physically-based rendering

## Définition

- ▶ Ensemble de techniques de rendu permettant de réaliser un rendu s'approchant des propriétés physiques de la lumière et de la matière.
- ▶ Une baseline (un modèle) pour développer le rendu d'une image selon des propriétés physiques

## 3 éléments principaux

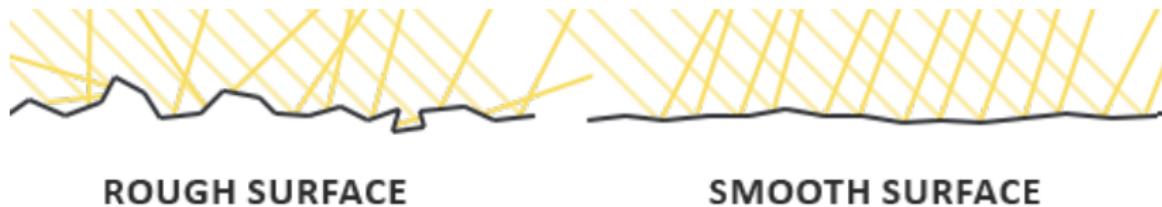
- ▶ Modèle de micro-facettes (*Microfacet model*)
- ▶ Conservation de l'énergie (*Energy conservation*)
- ▶ Lumière utilisant une fonction de réflectivité bidirectionnelle (BRDF)

# Modèle de micro-facettes

## Définition

Certaines surfaces ne sont pas lisses d'un point de vue optique.

- ▶ Rugosité (*Roughness*)



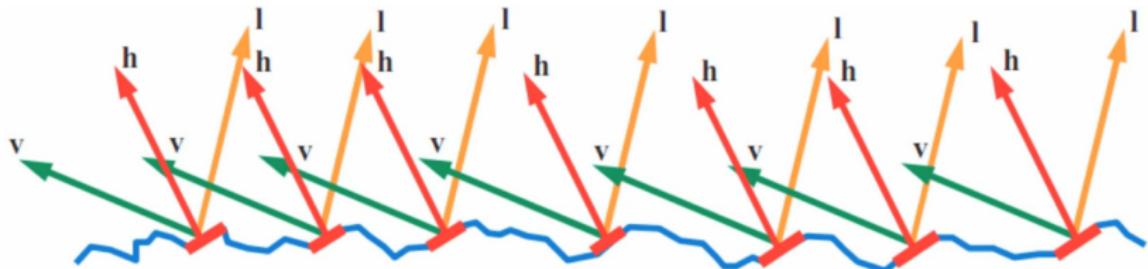
- ▶ Voir n'importe quelle surface comme de petits miroirs réfléctifs (*microfacets*)

# Modèle de micro-facettes

## Approximation de la rugosité

A partir de la direction de la lumière  $l$  et de la direction vers la vue  $v$ , on calcule la normale de la micro-facette  $h$  ou couramment appelée *half-vector*.

$$h = \frac{v + l}{\|v + l\|} \quad (1)$$



# Rappel de l'équation de rendu

## ► Équation de rendu

$$\mathbb{L}_{\text{out}}(p, v) = L_{\text{emission}}(p, n, v) + \int_{\Omega} L_{\text{in}}(p, l) f_{\text{reflect}}(p, n, l, v) \text{sat}(n.l) d\omega$$

## ► Approximation temps réel

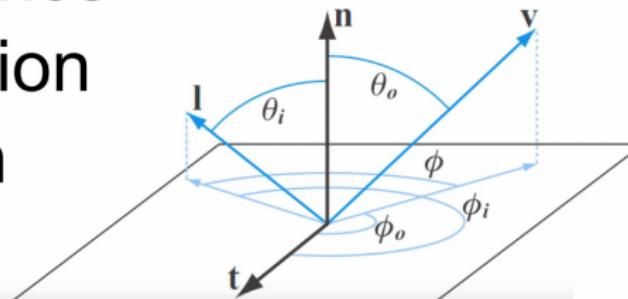
$$C_{\text{shaded}} = f_{\text{ambient}}(C_{\text{ambient}}, v) + \sum_{k=1}^n f_{\text{direct}}(C_{\text{light}}^k, n, l_k, v)$$

## BRDF

► C'est quoi une BRDF?

Bidirectional  
Reflectance  
Distribution  
Function

$$f(l, v)$$



$f(l, v)$  de l'équation de rendu

$p$ , position du point  $l$ , direction de la lumière vers le point  $n$ ,  
normal au sommet  $v$ , direction vers la vue

# Exemple de BRDF : Diffusion de Lambert

## Illumination directe

$$f_{\text{direct}}(C_{\text{light}}^k, n, l_k, v) = C_{\text{light}}^k f_{\text{BRDF}}(n, l_k, v) \text{sat}(n.l_k)$$

## Lambertian diffuse

En fonction de l'angle d'incidence de la lumière à la surface =>  
 $\text{sat}(n.l)$

$$f_{\text{BRDF}}(n, l, v) = \frac{\rho}{\pi} C_{\text{diffuse}} \approx \frac{C_{\text{diffuse}}}{\pi}$$

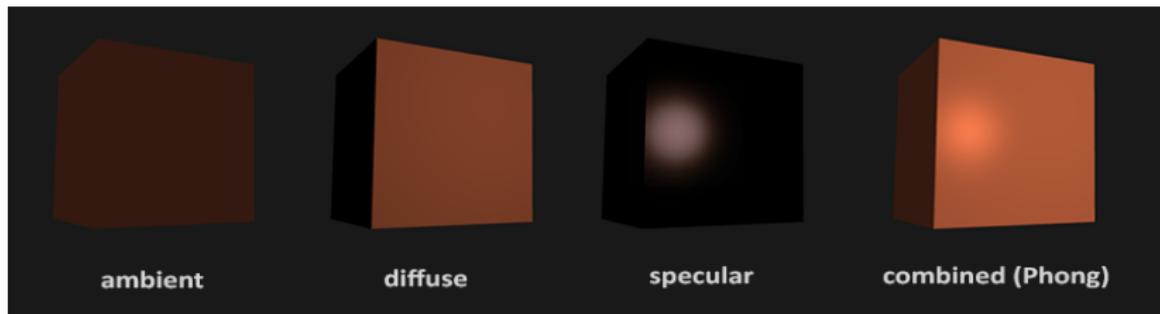
# Exemple de BRDF : Spéculaire de Blinn-Phong

## Specular Reflection

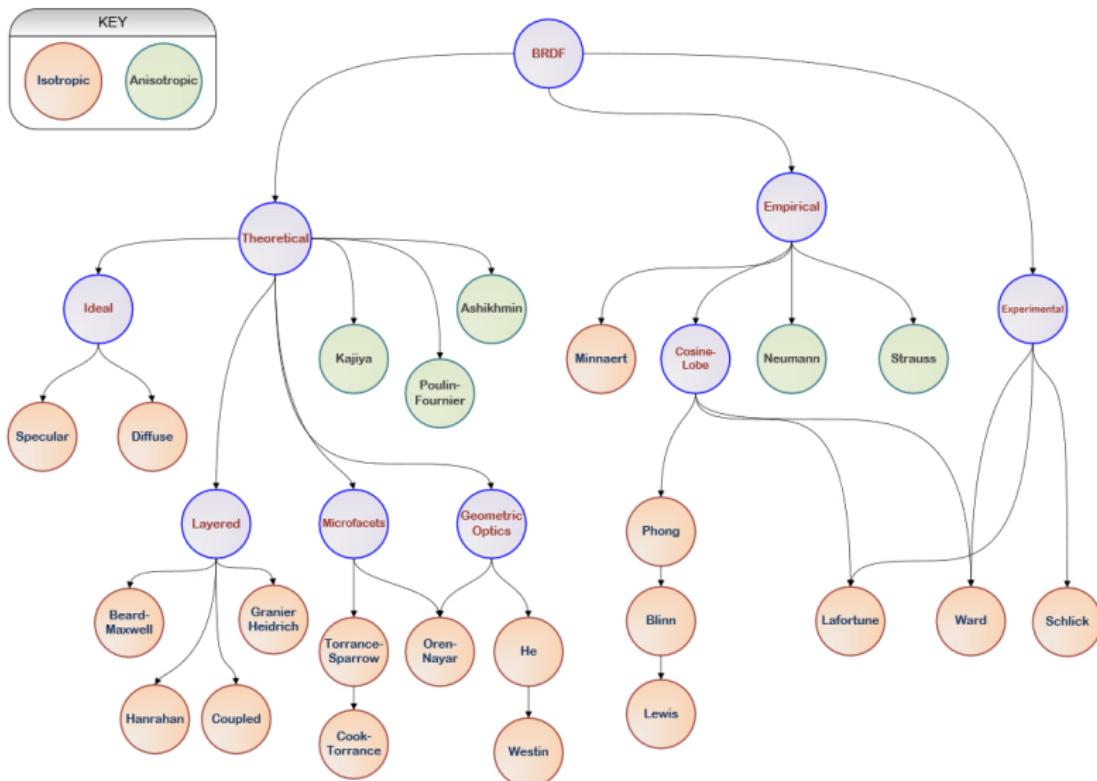
$$f_{\text{direct}}(C_{\text{light}}, n, l, v) = C_{\text{light}} \left\{ \frac{\rho}{\pi} C_{\text{diffuse}} \text{sat}(n.l) + C_{\text{specular}} [\text{sat}(n.h)]^{\alpha} \right\}$$

$h$  : le vecteur *half-vector*

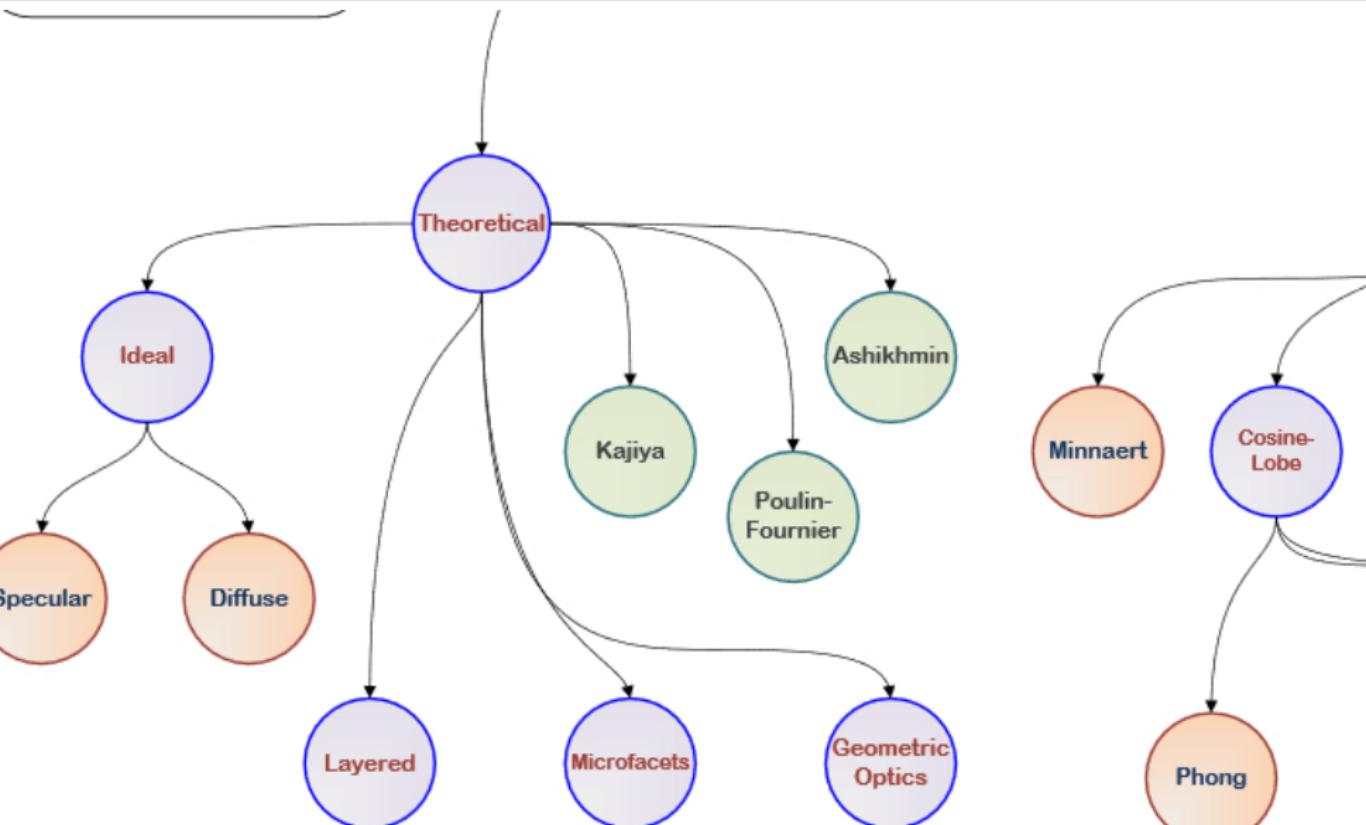
$\alpha$  : la puissance de l'effet spéculaire



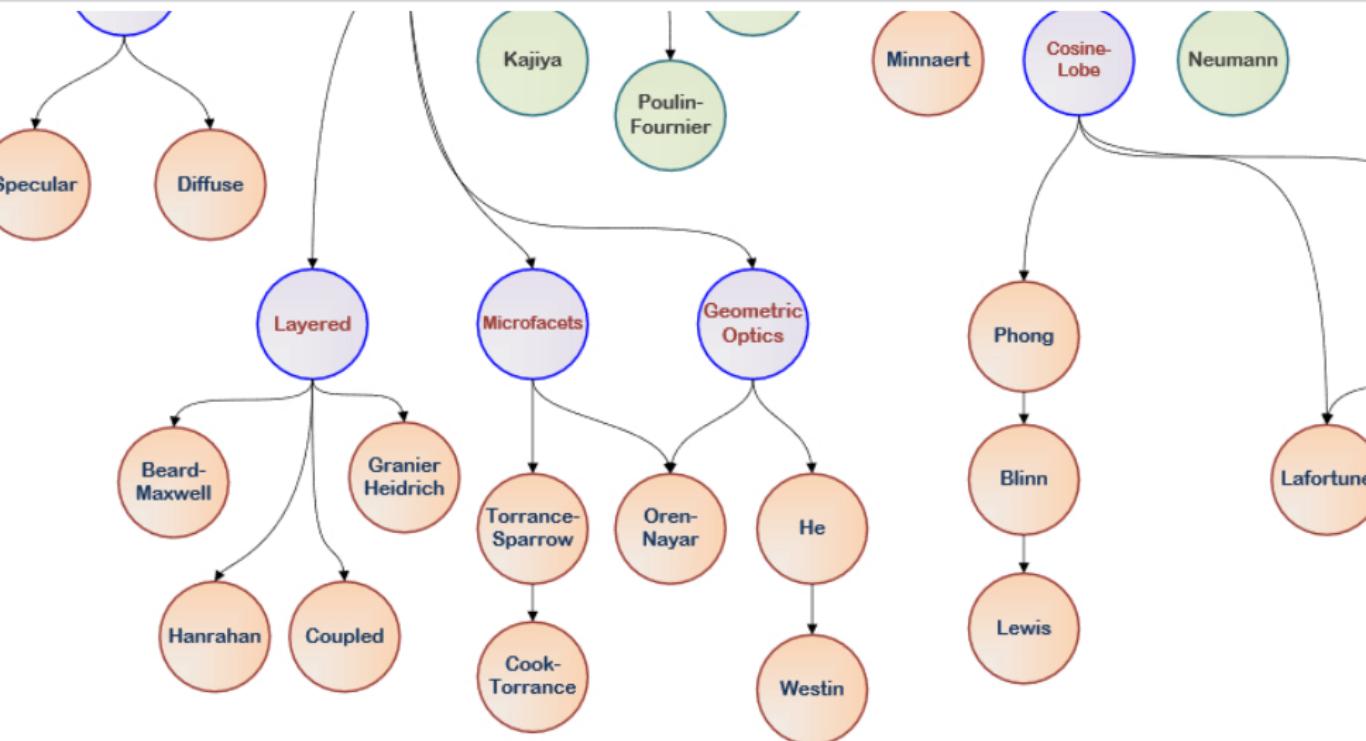
## BRDF Big Family



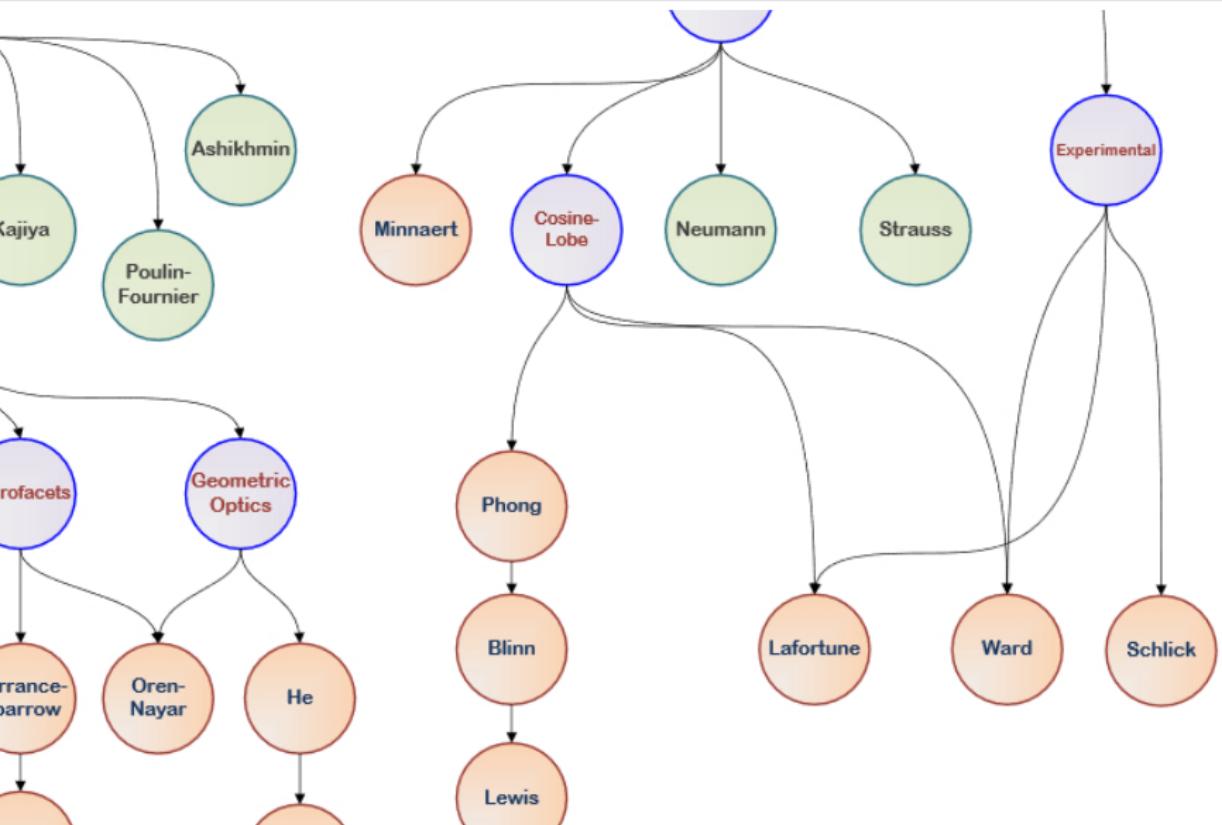
## BRDF Big Family



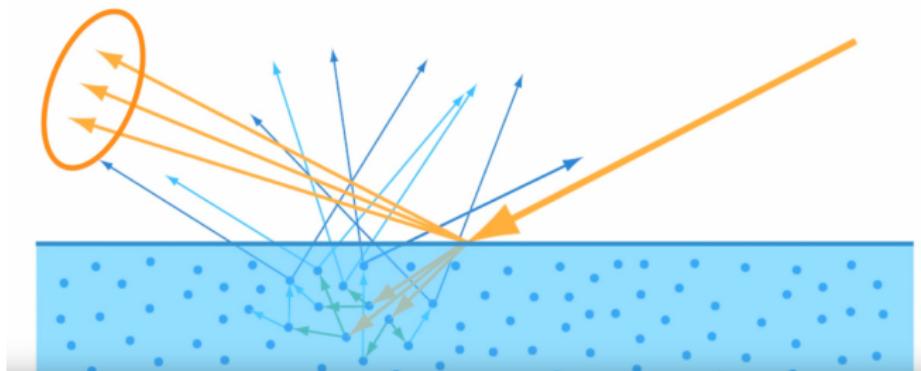
# BRDF Big Family



# BRDF Big Family



# Cook Torrance Specular BRDF



## Microfacet Specular BRDF

$$f(h, l, v) = \frac{F(l, h)G(l, v, h)D(h)}{4(n.l)(n.v)} \quad (2)$$

où  $F$  est la fonction de Fresnel,  $G$  est la fonction de géométrie et  $D$  la fonction de distribution.

# Fonction de Fresnel

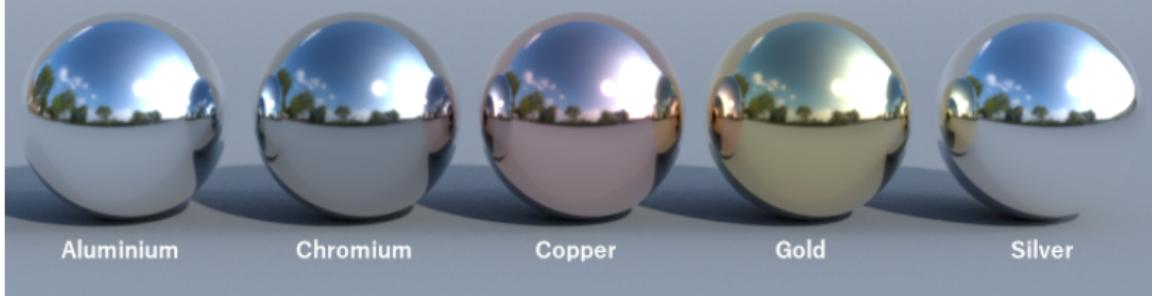
## Définition

Le ratio physique de la réflexion de surface selon différents angles de la surface.

- ▶ Varie selon la matière
- ▶ Plus élevé sur les métaux

### Fresnel Presets (%) :

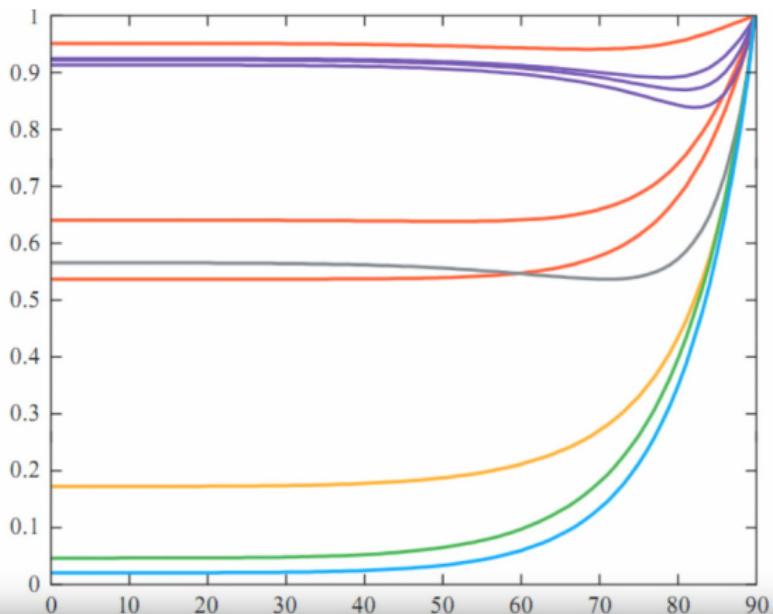
Roughness 10%



# Fonctions de Fresnel

## Fresnel Reflectance

- copper
- aluminum
- iron
- diamond
- glass
- water



$$F_0 = F(0)$$

La couleur spéculaire lorsque l'angle d'indice est de zéro.

# Fonctions de Fresnel

## Approximation de Schlick

- ▶ Paramétrisation par  $F_0$

$$F_{Schlick}(F_0, l, n) = F_0 + (1 - F_0)(1 - (l \cdot n))^5$$

- ▶ Pour les micro-facettes : ( $n = h$ )

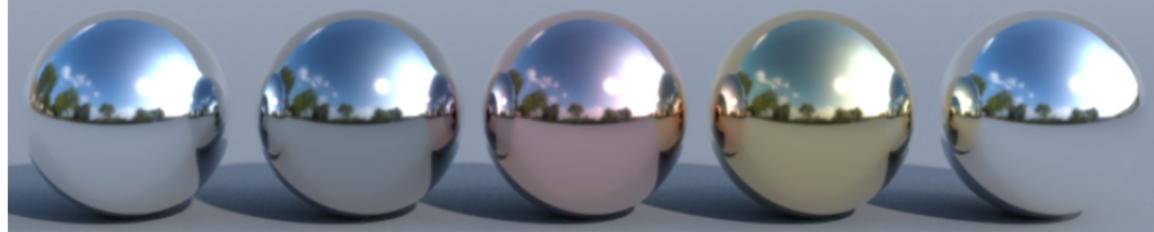
Dielectric	$F_0$ (Linear, Float)	$F_0$ (sRGB, U8)	Color
Water	0.020	39	
Plastic, Glass	0.040 – 0.045	56 – 60	
Crystalware, Gems	0.050 – 0.080	63 – 80	
Diamond-like	0.100 – 0.200	90 – 124	

# Fonctions de Fresnel

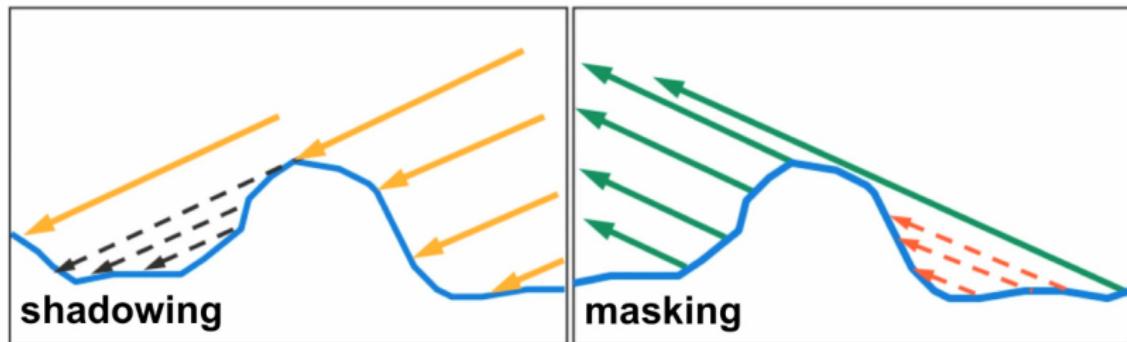
Metal	$F_0$ (Linear, Float)	$F_0$ (sRGB, U8)	Color
Titanium	0.542,0.497,0.449	194,187,179	
Chromium	0.549,0.556,0.554	196,197,196	
Iron	0.562,0.565,0.578	198,198,200	
Nickel	0.660,0.609,0.526	212,205,192	
Platinum	0.673,0.637,0.585	214,209,201	
Copper	0.955,0.638,0.538	250,209,194	
Palladium	0.733,0.697,0.652	222,217,211	
Zinc	0.664,0.824,0.850	213,234,237	

Fresnel Presets (%) :

Roughness 10%



# Fonction de Géométrie $G$



## Définition

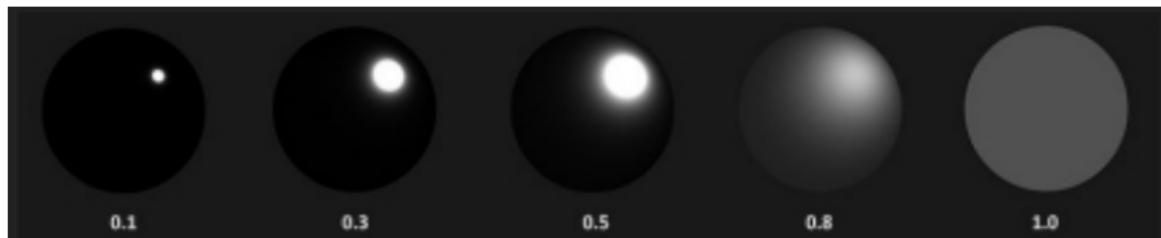
Aussi appelée *shadowing-masking function*. Elle définit la propriété auto-ombrage (*self-shadowing*) des microfacettes. Quand une surface est relative rugueuse, les microfacettes peuvent générer la lumière et réduire ainsi la lumière que la surface reflète.

# Fonction de Géométrie $G$

## Approximation de SchlickGGX

$$G_{Schlick-GGX}(n, v, r) = \frac{n.v}{(n.v)(1 - k) + k}$$

où  $k = \frac{(1+r)^2}{8}$  et  $r$  est le facteur de rugosité.

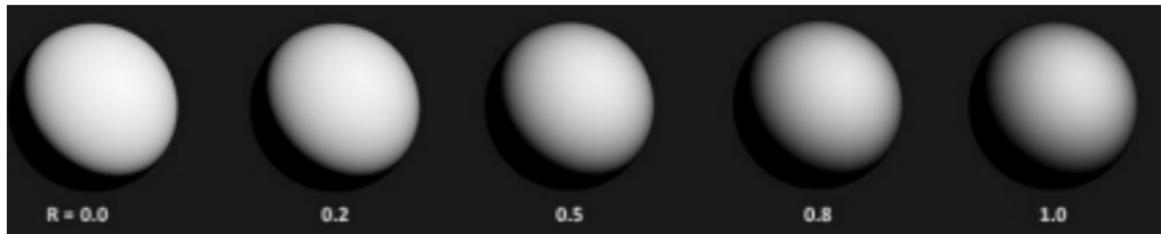


# Fonction de Géométrie $G$

## Approximation de Smith

$$G_{Smiths}(n, v, l, r) = G_{Schlick-GGX}(n, v, r) * G_{Schlick-GGX}(n, l, r)$$

Il existe d'autres fonctions de *shadowing-masking* plus efficace comme celle de Smith.



# Fonction de Distribution $N$

## Définition (*Normal Distribution Function* : NDF)

Il s'agit de l'approximation de la quantité de micro-facettes alignées avec le vecteur  $h$  selon la rugosité de la surface.

$$D_p(\mathbf{m}) = \frac{\alpha_p + 2}{2\pi} (\mathbf{n} \cdot \mathbf{m})^{\alpha_p}$$

$$D_{uabc}(\mathbf{m}) = \frac{1}{(1 + \alpha_{abc1} (1 - (\mathbf{n} \cdot \mathbf{m})))^{\alpha_{abc2}}}$$

$$D_{tr}(\mathbf{m}) = \frac{\alpha_{tr}^2}{\pi ((\mathbf{n} \cdot \mathbf{m})^2 (\alpha_{tr}^2 - 1) + 1)^2}$$

$$D_b(\mathbf{m}) = \frac{1}{\pi \alpha_b^2 (\mathbf{n} \cdot \mathbf{m})^4} e^{-\left(\frac{1 - (\mathbf{n} \cdot \mathbf{m})^2}{\alpha_b^2 (\mathbf{n} \cdot \mathbf{m})^2}\right)}$$

# Fonction de distribution

## Distribution GGX

$$D(n, h, r) = \frac{r^2}{\pi * ((n.h^+)^2(r^2 - 1) + 1)^2}$$

# Cook-Torrance BRDF

## Equation

$$f_r = k_d \cdot f_{\text{Lambert}} + k_s \cdot f_{\text{CookTorrance}}$$

## Lambert Diffuse

Une constante de couleur

$$f_{\text{lambert}} = \frac{c}{\pi}$$

Où  $c$  est l'*albedo* (la couleur de base de la matière)

## Comment calculer $k_d$ et $k_s$

$k_s = F$ , la valeur de *Fresnel* de la composante spéculaire

$k_d = 1.0 - k_s$ , la partie d'énergie non reflétée par le spéculaire

# Lumières

## La luminance de la lumière

- ▶ Pondérée par la distance entre la lumière

$$L_k = C_{\text{light}}^k \cdot a(p, p_k)$$

où  $C_{\text{light}}^k$  est la couleur de la lumière et  $a$  la fonction d'atténuation de la luminance selon la distance entre le point  $p$  sur la surface et l'origine de la lumière.

## Fonction d'atténuation $a$

$$a(p, p_i) = \frac{1}{(||p - p_i||)^2}$$

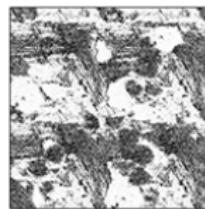
## PBR Shader



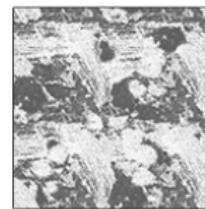
ALBEDO



NORMAL



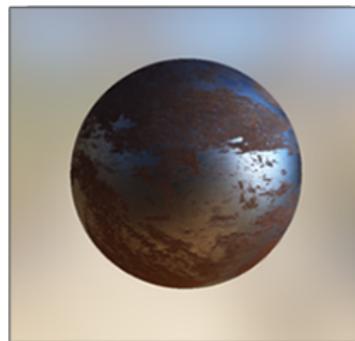
METALLIC



ROUGHNESS



AO



# Astuces

- ▶ Conversion Linear RGB en sRGB (pour éviter les couleurs ternes car le rendu est une image sRGB)
- ▶ A réaliser juste avant d'assigner la couleur au *shader*

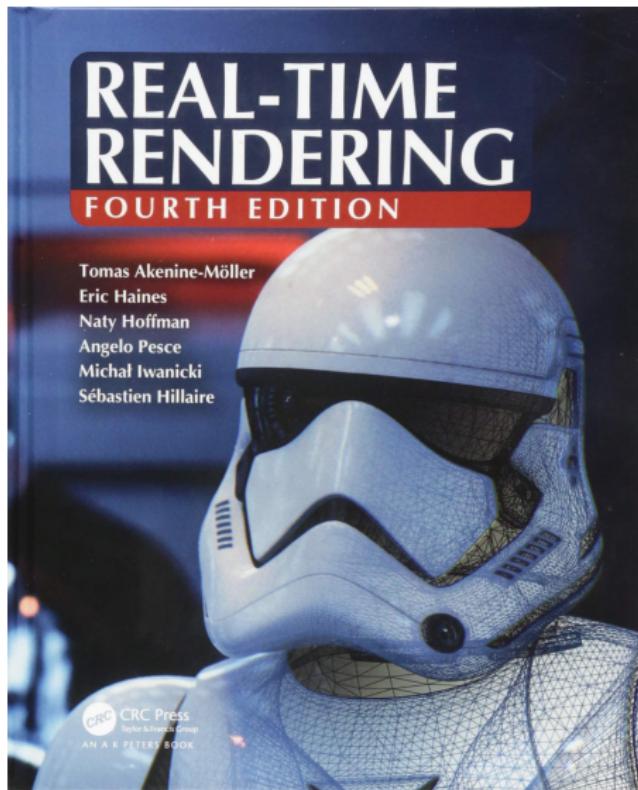
## HDR tonemapping

$$\text{FragColor} = \text{color}/(\text{color} + \text{vec3}(1.0));$$

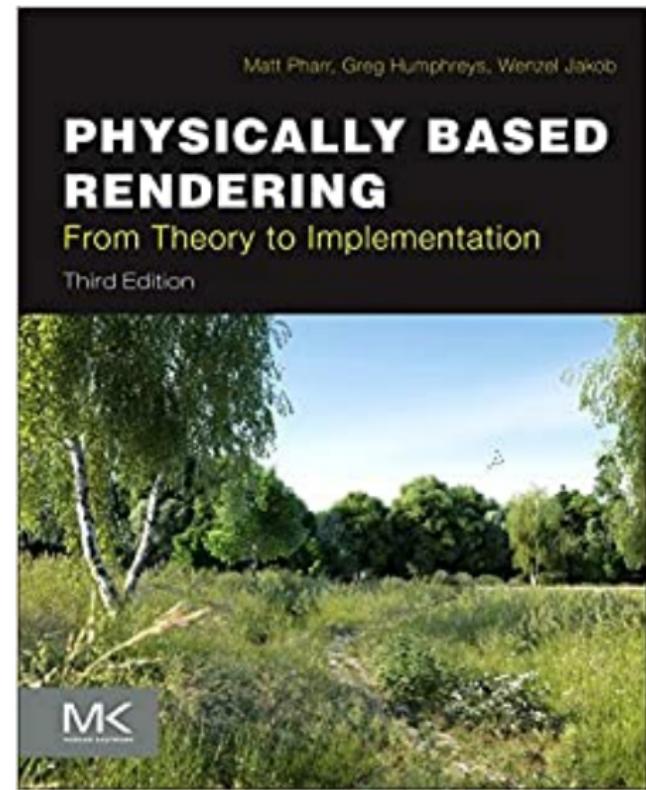
## Gamma correction

$$\text{FragColor} = \text{pow}(\text{FragColor}, \text{vec3}(1.0/2.2));$$

## Références



Sébastien Beugnon



PBR

45/46

# Références

-  T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, S. Hillaire  
Real-Time Rendering, Fourth Edition.  
CRC Press, 2018
-  Matt Pharr, Wenzel Jakob, and Greg Humphreys  
Physically Based Rendering : From Theory to Implementation.  
(Third Edition)  
Morgan Kaufmann Publishers, 2016

# Travaux pratiques

\* Vous pouvez utiliser votre propre moteur si vous le souhaitez

## Niveau 0

Développer votre propre modèle (*shader*) PBR avec les paramètres suivants :

- ▶ Albedo (Couleur + Texture)
- ▶ Metalness (Valeur + Texture)
- ▶ Roughness (Valeur + Texture)
- ▶ AO (Valeur + Texture)

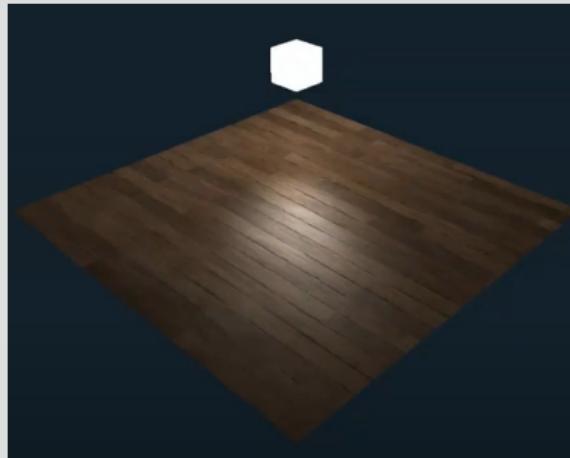
Créer votre classe de matière associant un shader program et des paramètres à injecter dedans

# Travaux pratiques

\* Vous pouvez utiliser votre propre moteur si vous le souhaitez

## Niveau 1

Rajouter l'effet d'émission et d'ambiant occlusion :



# Travaux pratiques

\* Vous pouvez utiliser votre propre moteur si vous le souhaitez

## Niveau 2

Niveau 2 : Choisir un modèle PBR connu parmi les suivants et essayer de développer certaines fonctionnalités de ces derniers :

- ▶ glTF PBR
- ▶ Filament
- ▶ Dassaut (2022)