Physically-based rendering (PBR)

in OpenGL

Introduction

PBR

Introduction

- **Physically-based rendering** or **PBR** is an umbrella term that encompasses tools and techniques that make use of physically-based models of light and reflection.
- Generally be described as a shading/reflection model that tries to model the physics of light interacting with matter as accurately as possible.
- We are interested primarily in how it differs from the Phong and the Blinn-Phong reflection models.

Introduction

- PBR lighting model it is more detailed and accurate with regards to the physics of the interaction being represented than the Blinn-Phong model.
- The Blinn-Phong model uses a few parameters which are not physically based but produce effective results (light separation).
- However, it provides many "tuneable" parameters to the artist to work with, giving them the flexibility to achieve the desired look.

Introduction

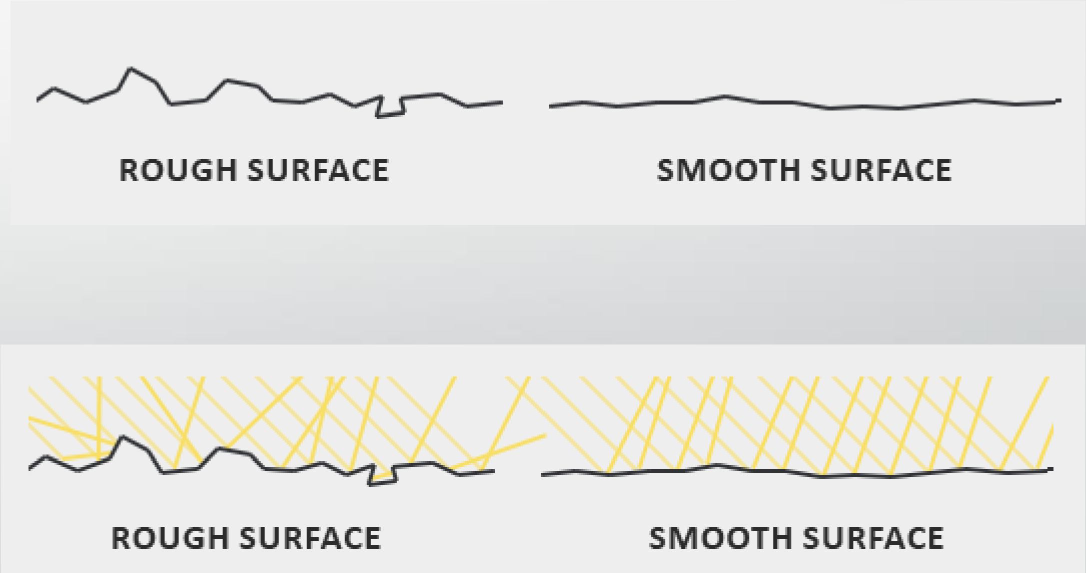
- PBR lighting model in order to be considered physically based, it has to satisfy the following 3 conditions:
  - Be based on the microfacet surface model.
  - Be energy conserving.
  - Use a physically based BRDF (bidirectional reflectance distribution function).

The microfacet model

PBR

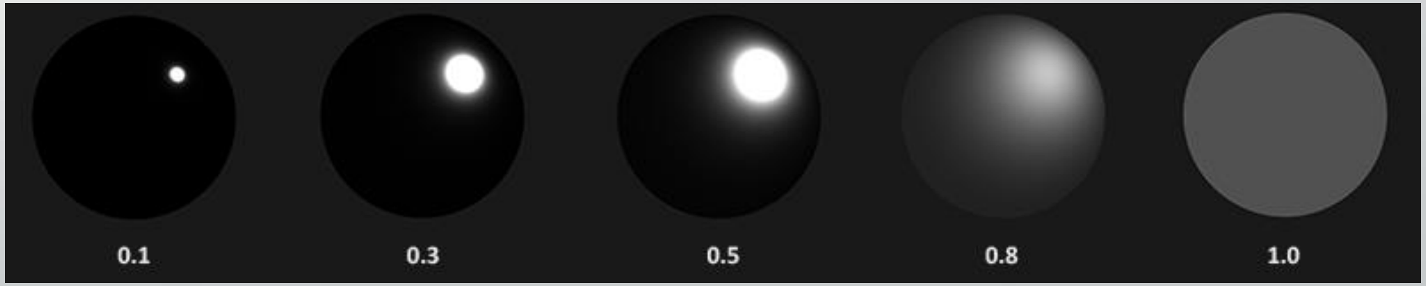
The microfacet model

- All the PBR techniques are based on the theory of microfacets. The theory behind this describes that any surface at a microscopic scale can be described by tiny little perfectly reflective mirrors called **microfacets**.



The microfacet model

- Based on the roughness of a surface, we can calculate the ratio of microfacets roughly aligned to some vector h.
- This vector h is the halfway vector that sits half way between the light and the view vector.
- Higher roughness values display a much larger specular reflection shape, in contrast with the smaller and sharper specular reflection shape of smooth surfaces.



Energy conservation

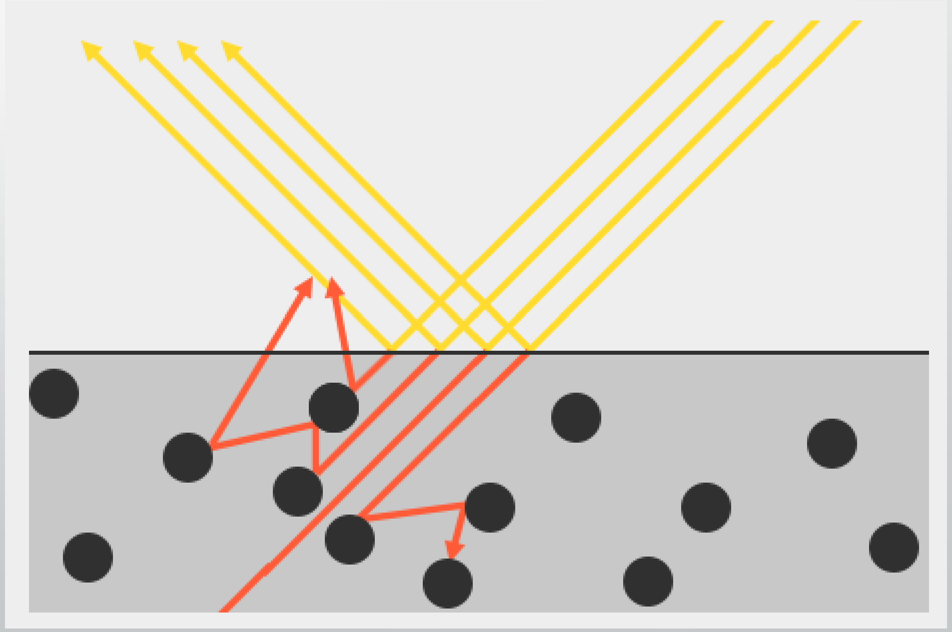
PBR

Energy conservation

- The rule is: outgoing light energy should never exceed the incoming light energy (excluding emissive surfaces).
- The **reflection** part is light that directly gets reflected and doesn't enter the surface; this is what we know as **specular lighting**.
- The **refraction** part is the remaining light that enters the surface and gets absorbed; this is what we know as **diffuse lighting**.

Energy conservation

- An additional element is the surface reaction to light:
- Metallic surfaces
  - All refracted light gets absorbed without scattering
  - Have only specular light
- Dielectrics (non-metallic)



Energy conservation

- The reflection and the refraction surfaces are mutually exclusive.
- We preserve this energy conserving relation by first calculating the **specular fraction** that amounts the percentage the incoming light's energy is reflected.

Reflection:

```
float Ks= calculateSpecular();        //between 0-1
```

Refraction:

```
float Kd = 1.0f – Ks;        //the remaining value
```

Render equation

PBR

PBR rendering

- We'll use 4 important vectors:
  - n: The surface normal
  - l: The vector representing the incoming light
  - v: The direction towards the viewer (camera)
  - h: The vector halfway between l and v (same vector as in Blinn-Phong implementation)

PBR rendering

The reflectance equation:

$$L_o(v) = \int_\Omega f(l, v) L_i(l)(n \cdot l) d\omega_i$$

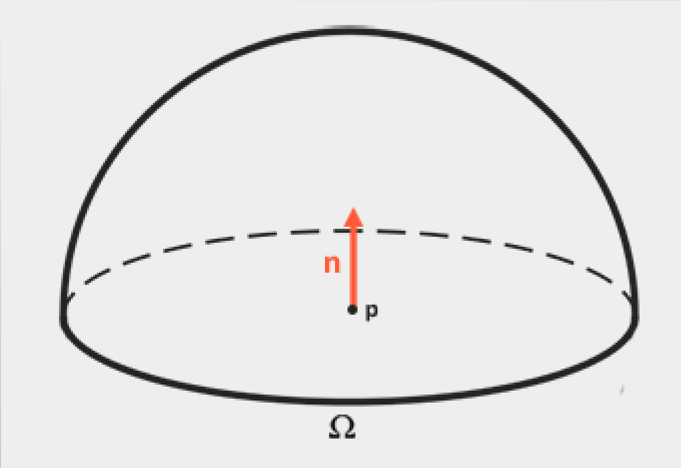
L_o: radiance of a surface

v: viewer

f: bidirectional reflectance distribution function (BRDF)

l: light direction

$d\omega_i$: all incoming directions in an hemisphere Ω



PBR rendering

BRDF:

$$L_o(v) = \int_\Omega f(l, v) L_i(l)(n \cdot l) d\omega_i$$

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

f_d: the diffuse BRDF

f_s: surface reflectance

$$f_s = \frac{F(l,h)G(l,v,h)D(h)}{4(n \cdot l)(n \cdot v)}$$

F: Fresnel reflection (we'll use Schlick approximation in our shader)

G: geometry function (we'll use geomSmith implementation in the shader)

D: the microgeometry normal distribution function (or microfacet distribution function) (we'll use the GGX/Trowbridge-Reitz implementation in our shader)

PBR rendering

The reflectance equation becomes:

$$L_o(v) = \pi \sum_{i=1}^{N} L_i f(l_i, v)(n \cdot l_i)$$

L_i: the illumination received at the surface due to i^{th} light source

N: number of point light sources

l_i : direction towards i^{th} light source

The intensity of light decreases with distance, we'll use an inverse-square implementation:

$$L_i = \frac{I_i}{d_i^2}$$

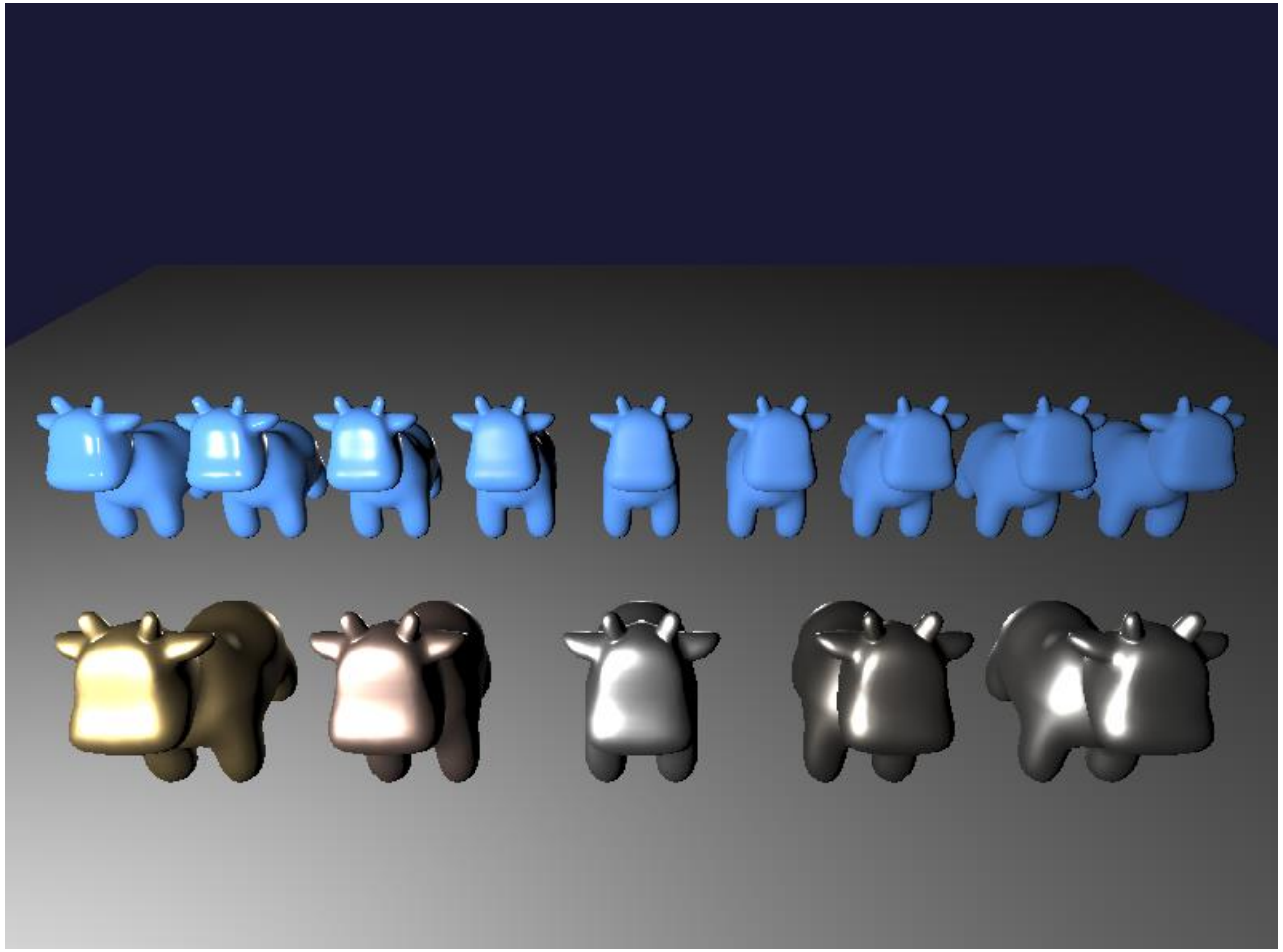
I_i: the intensity of the light source

d_i^2: the distance from the surface point to the light source

PBR rendering

The control parameters used in the shader are:

- The surface roughness (r), a value between 0 and 1 (float)
- A Boolean that represents whether or not the material is metallic
- A colour which is interpreted as the diffuse colour for dielectrics, or the characteristic specular reflectance ($\mathbf{F_0}$) for metals (vec3)



PBR rendering

Useful links

- Learn OpenGL - PBR: https://learnopengl.com/PBR/Theory
- SIGGRAPH 2013 course: Physically based shading in theory and Practice: https://blog.selfshadow.com/publications/s2013-shading-course/
- To read: Book - Physically Based Rendering by Pharr, Jakob, and Humphreys
- To read: Lighting and shading: A physically-based reflection model (OpenGL 4 Shading Language Cookbook).