Image-Based Lighting

David Schedl

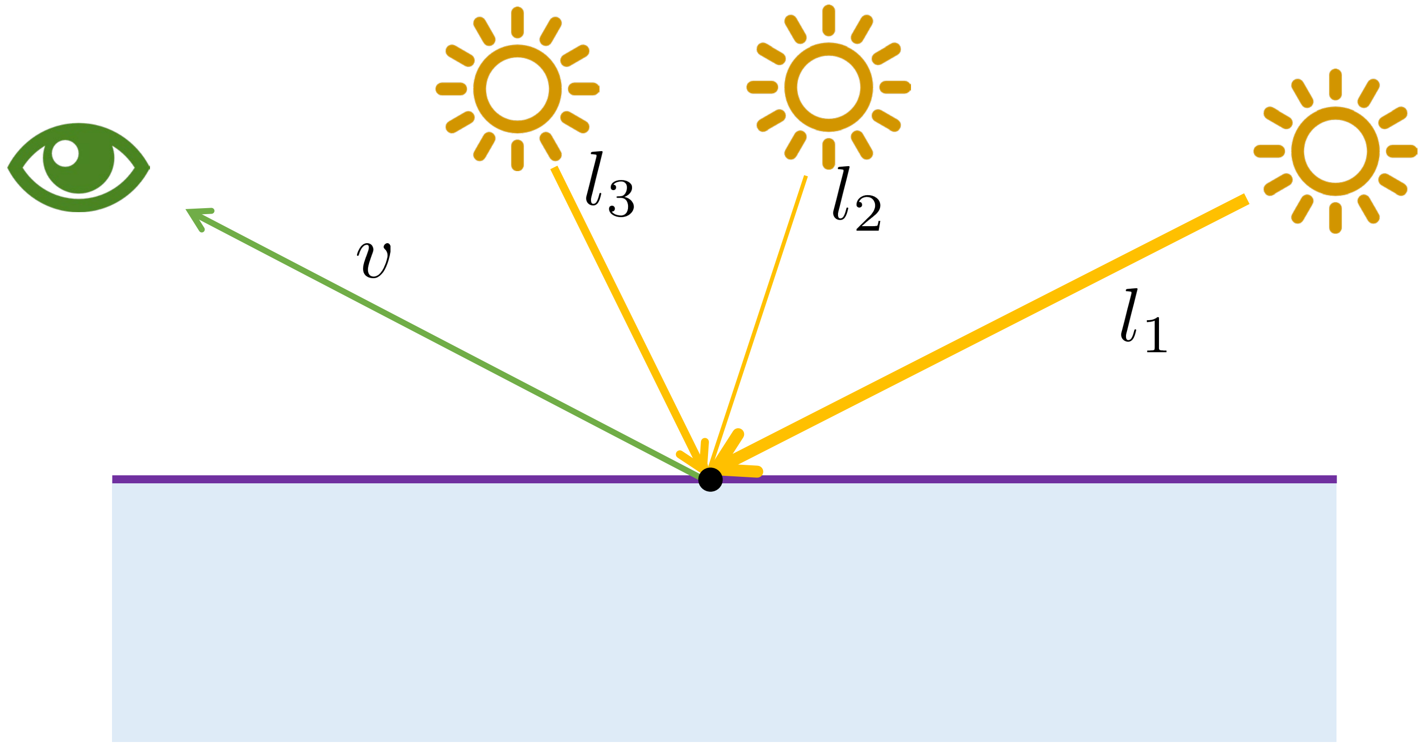
david.schedl@fh-hagenberg.at

Last week …

… today!

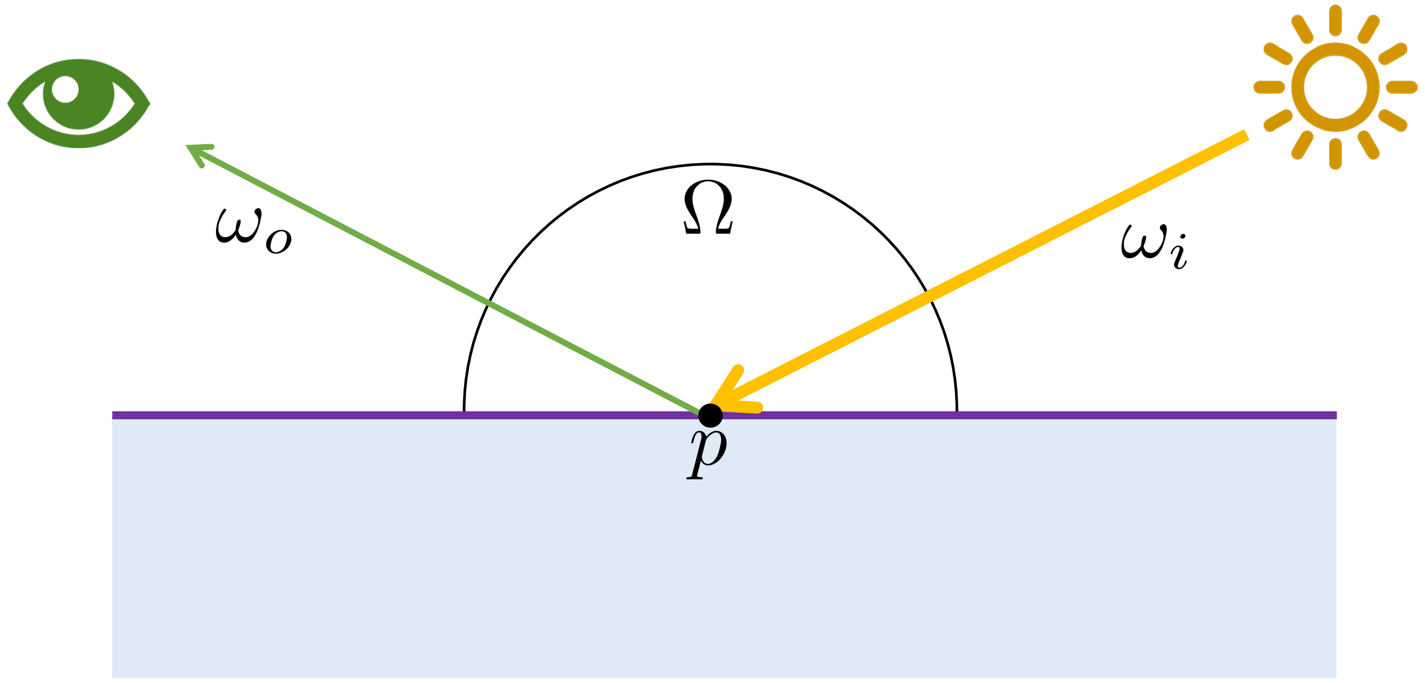
Recap PBR

$$L_v = \sum_{i=1}^{3} f(l_i, v) L_i n \cdot l_i$$



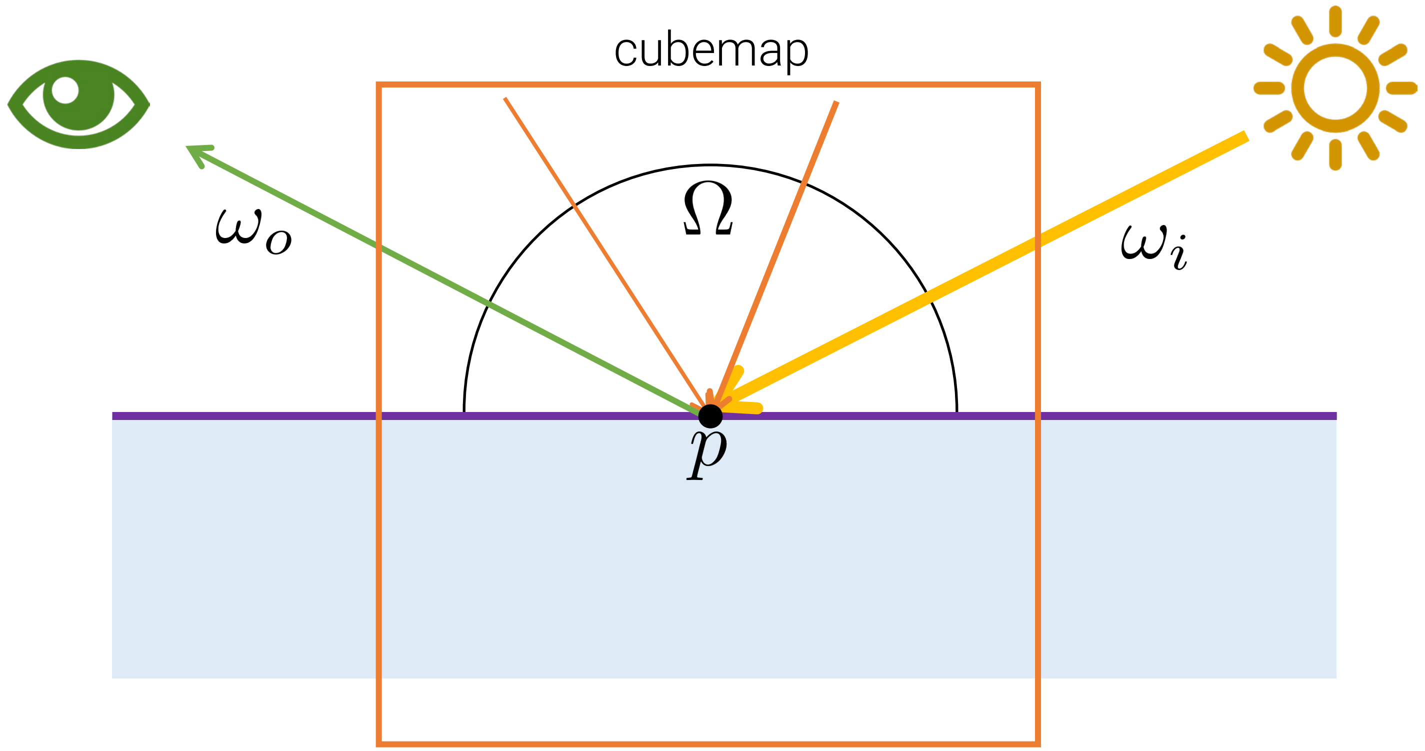
Reflectance Equation

$$L_o(p,\omega_o) = \int_{\Omega} f(p,\omega_i,\omega_o) L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$



Reflectance Equation

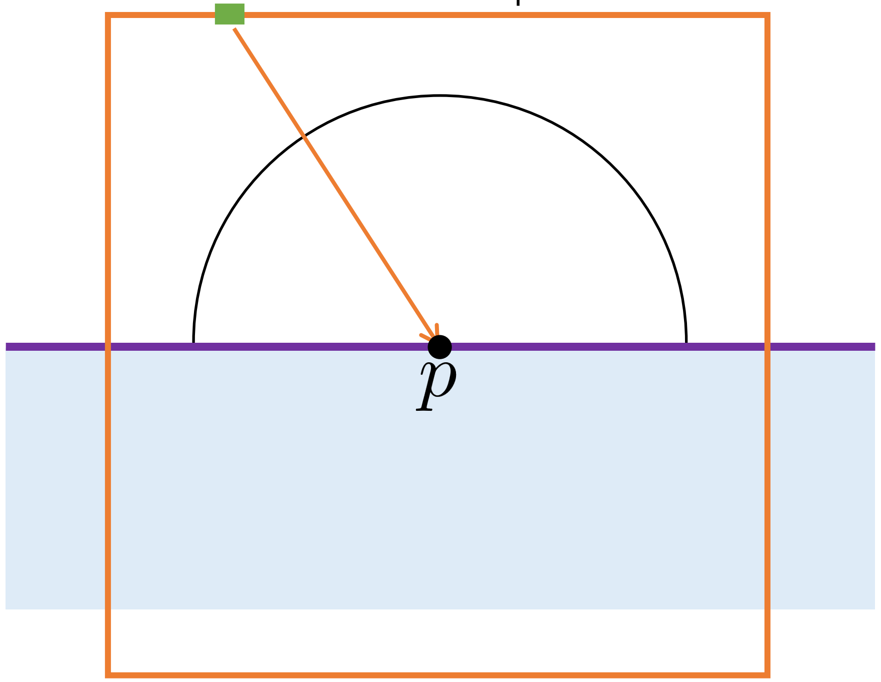
$$L_o(p, \omega_o) = \int_{\Omega} f(p, \omega_i, \omega_o) L_i(p, \omega_i) n \cdot \omega_i d\omega_i$$



Reflectance Equation

- ω_i corresponds to a texel in the cubemap texture.

cubemap



HDR texture

Real-Time Approximations

$$L_o(p,\omega_o) = \int\limits_{\Omega} (k_d \frac{c}{\pi} + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$

Real-Time Approximations

$$L_o(p,\omega_o) = \int\limits_\Omega (k_d \frac{c}{\pi}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i + \int\limits_\Omega (\frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$

- Split into diffuse and specular sum.

Diffuse Term

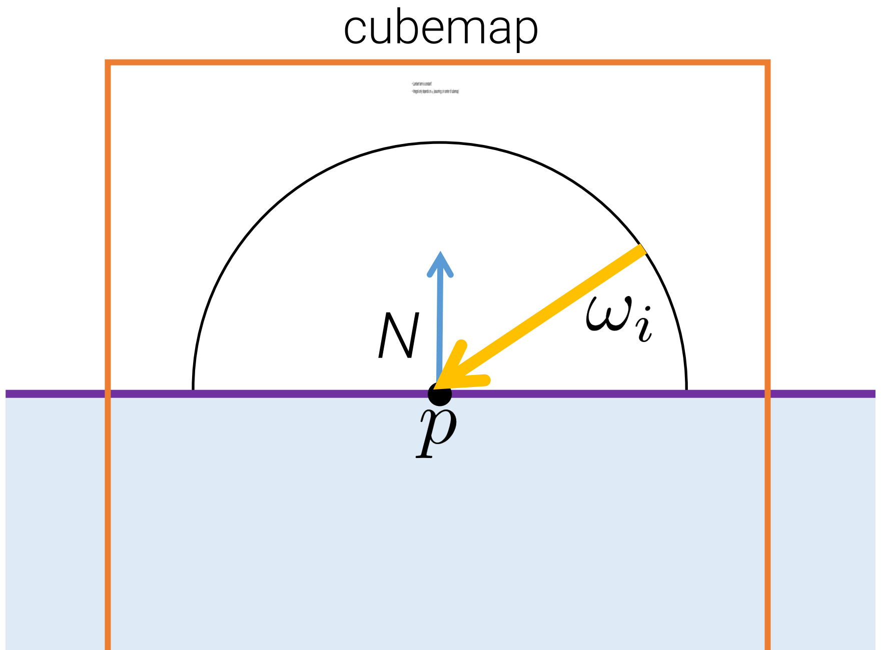
$$L_o(p,\omega_o) = \int\limits_{\Omega} (k_d \frac{c}{\pi}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$

- Let’s look at diffuse first (easier too).

Diffuse Term

$$L_o(p, \omega_o) = k_d \frac{c}{\pi} \int_{\Omega} L_i(p, \omega_i) n \cdot \omega_i d\omega_i$$

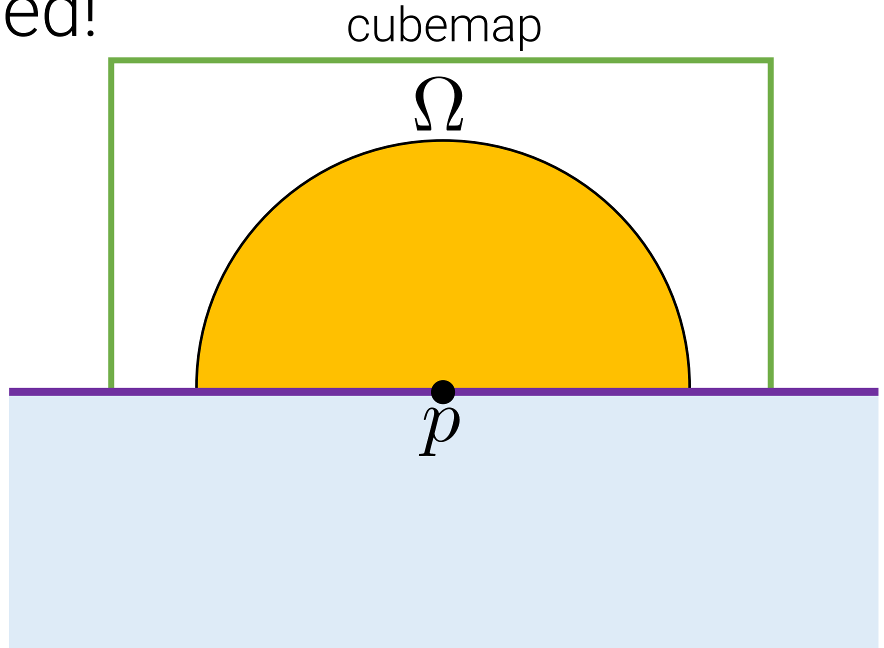
- Lambert term is constant!
- Integral only depends on ω_i (assuming p in center of cubemap)



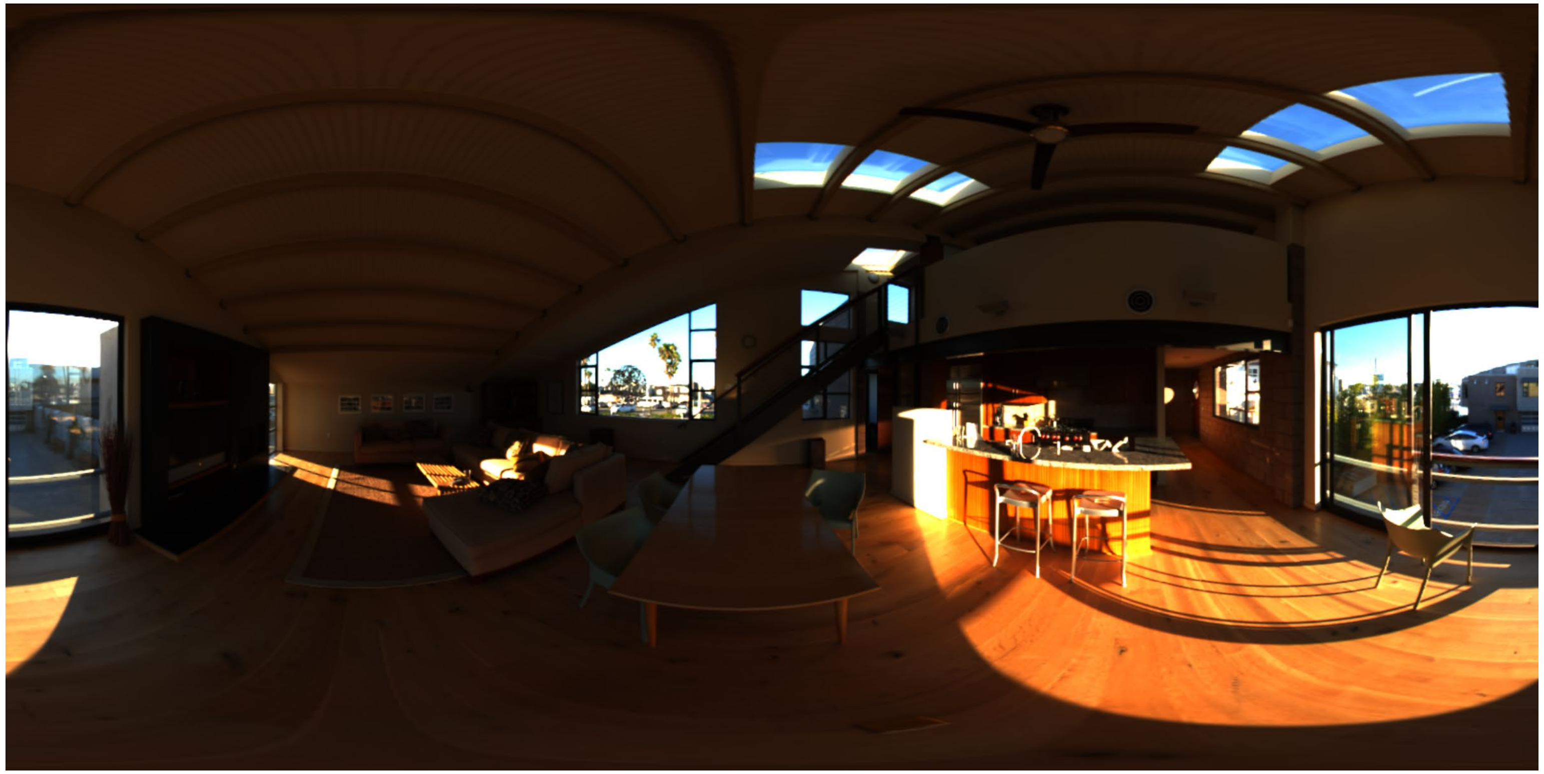
Diffuse Term

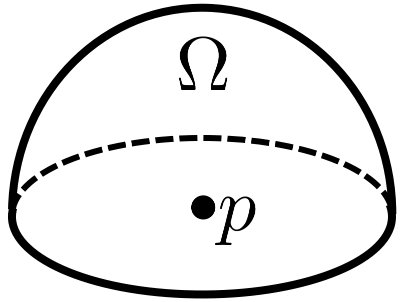
$$L_o(p,\omega_o) = k_d \frac{c}{\pi} \int_{\Omega} L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$

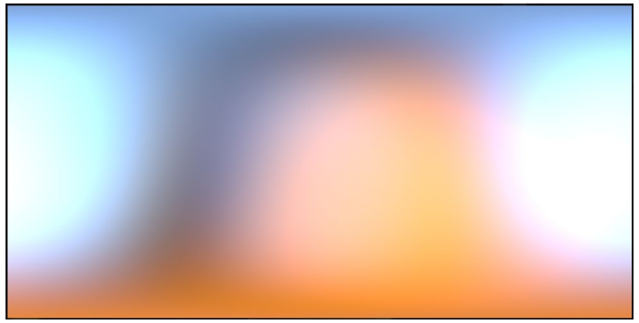
- Lambert term is constant.
- Integral only depends on ω_i (assuming p in center of cubemap).
- Can be precomputed!



Irradiance


HDR environment texture (not filtered)




irradiance map (filtered)

TASK #01: Irradiance

- Download the code.
- Let’s get diffuse IBL working
- Hints:
  - Only edit FS!
  - You need to sample the irradiance map.
  - Use IBL for ambient part.

Solution #01: GLSL Code

```
…
    // ambient lighting (we now use IBL as the ambient term)
    vec3 kS = vec3(0.0);
    vec3 kD = 1.0 - kS;
    kD *= 1.0 - metallic;

    vec3 irradiance = texture(irradianceMap, N).rgb;
    vec3 diffuse      = irradiance * albedo;

    vec3 ambient = (kD * diffuse) * ao;
```

Diffuse & Specular Sum

$$L_o(p,\omega_o) = \boxed{\int\limits_{\Omega} (k_d \frac{c}{\pi}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i} + \int\limits_{\Omega} (\frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$

Specular Part

$$L_o(p,\omega_o) = \int\limits_{\Omega} (k_d \frac{c}{\pi}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i + \boxed{\int\limits_{\Omega} (\frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i}$$

Specular Part

$$L_o(p,\omega_o) = \int\limits_{\Omega} (\frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$

- Depends on view vector (not that easy this time).
- We’ll do a few approximations!

Specular Part

$$L_o(p,\omega_o) = \int\limits_{\Omega} (\frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i$$

- Split in two sums (this is already an approximation):

$$L_o(p,\omega_o) = \int\limits_{\Omega} L_i(p,\omega_i) d\omega_i * \int\limits_{\Omega} f_r(p,\omega_i,\omega_o) n \cdot \omega_i d\omega_i$$

$$f_r(p,w_i,w_o) = \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}$$

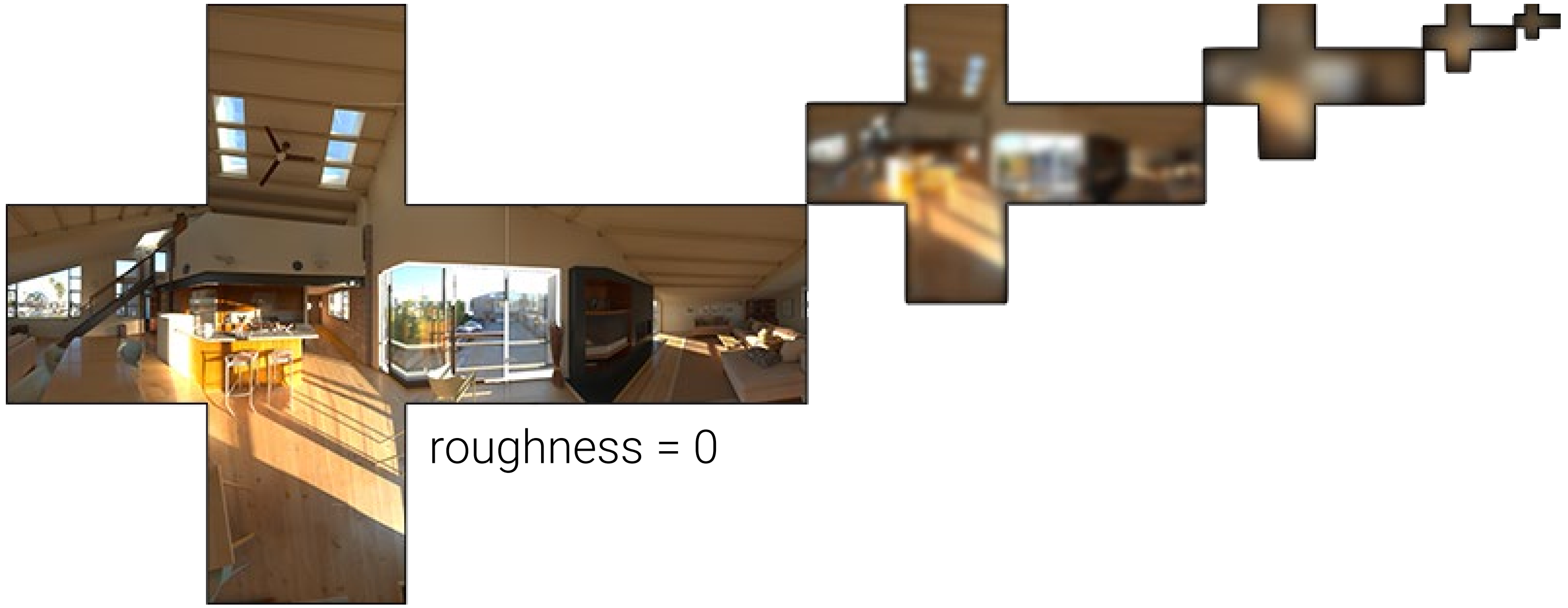
Specular Part

$$L_o(p,\omega_o) = \boxed{\int\limits_\Omega L_i(p,\omega_i)d\omega_i} * \boxed{\int\limits_\Omega f_r(p,\omega_i,\omega_o)n\cdot\omega_i d\omega_i}$$

- Strongly depends on view vector (not that easy this time).
- Split in two sums:
  - First term: incoming light as pre-filtered environment map (like the irradiance map, but taking roughness into account)
  - Second term: BRDF part approximated with a 2D lookup texture (called LUT or BRDF integration map)

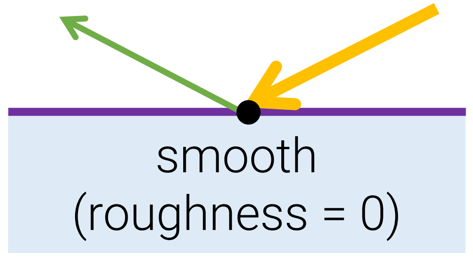
Pre-Filtered Environment Map

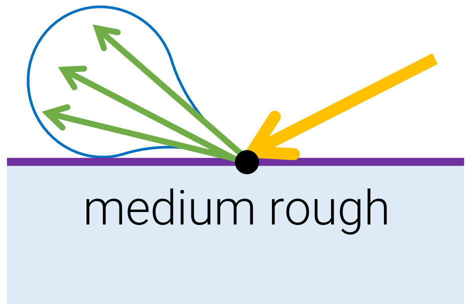
- Prefilter environment with NDF (D) for varying roughness and store in environment mipmap (varying resolution):

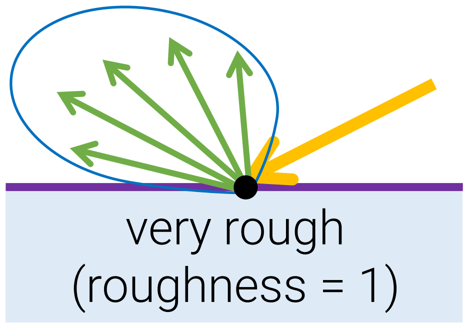


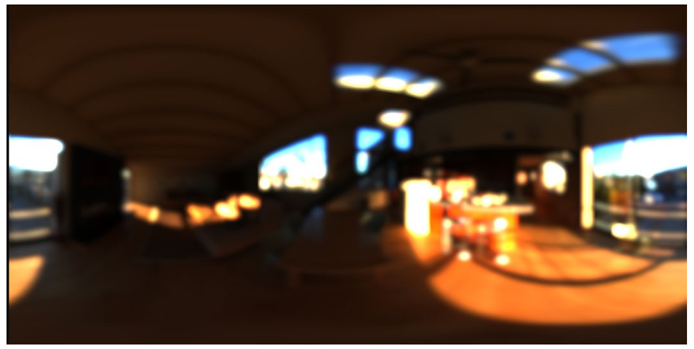
Pre-Filtered Environment Map

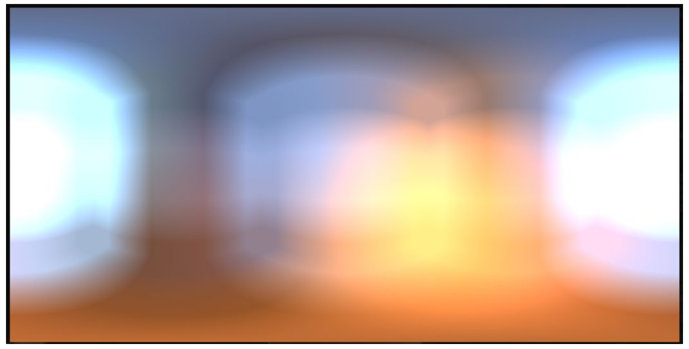
- Normal distribution function





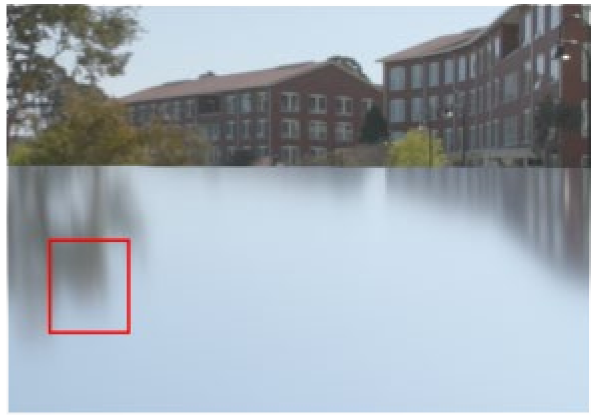


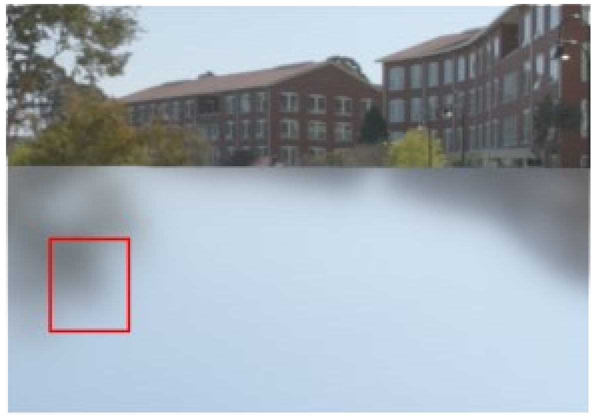




Pre-Filtered Environment Map

- Assuming V,N, and R are the same (causes artifacts):

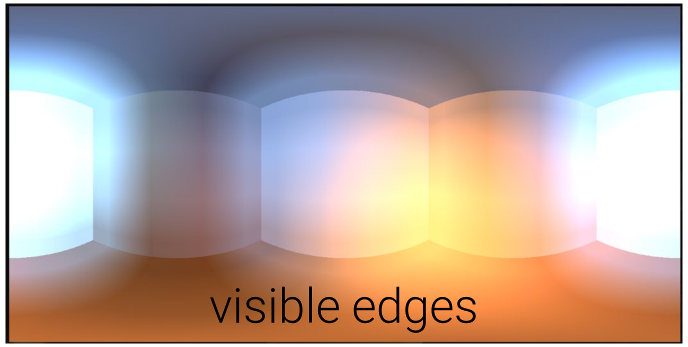

GRAZING SPECULAR REFLECTIONS

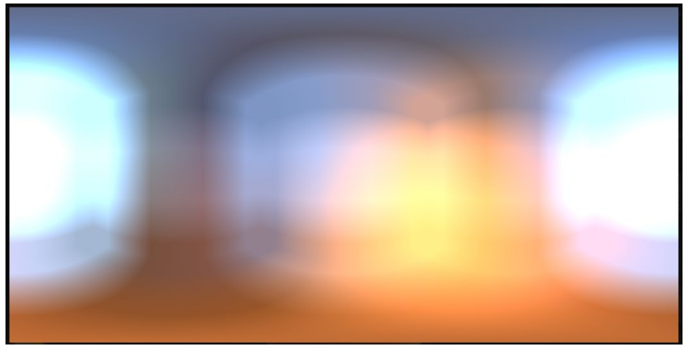

V = R = N

- Filtering with Importance Sampling (we'll not go into detail now).

Pre-Filtered Environment Map

- Make sure to filter over cubemap boundaries:
  `glEnable(GL_TEXTURE_CUBE_MAP_SEAMLESS);`





Access in GLSL Code:

```
vec3 prefilteredColor = textureLod(prefilterMap, R, roughness*MIPMAP_LEVELS).rgb;
```

BRDF Integration Map

$$L_o(p,\omega_o) = \int\limits_{\Omega} L_i(p,\omega_i)d\omega_i * \boxed{\int\limits_{\Omega} f_r(p,\omega_i,\omega_o)n\cdot\omega_i d\omega_i}$$

- Equation depending on roughness and $n \cdot \omega_o$
- Assuming constant radiance $L_i = 1$
- 2D LUT for varying roughness and $n \cdot \omega_o$

BRDF Integration Map

- Add Fresnel term to equation:

$$\int\limits_{\Omega} f_r(p, \omega_i, \omega_o) n \cdot \omega_i d\omega_i = \int\limits_{\Omega} f_r(p, \omega_i, \omega_o) \frac{F(\omega_o, h)}{F(\omega_o, h)} n \cdot \omega_i d\omega_i$$

- Reformulate to use F0:

$$\int\limits_{\Omega} \frac{f_r(p, \omega_i, \omega_o)}{F(\omega_o, h)} F(\omega_o, h) n \cdot \omega_i d\omega_i = \int\limits_{\Omega} \frac{f_r(p, \omega_i, \omega_o)}{F(\omega_o, h)} (F_0 + (1 - F_0)\alpha) n \cdot \omega_i d\omega_i$$

- Split into two sums (again):

$$\int\limits_{\Omega} \frac{f_r(p, \omega_i, \omega_o)}{F(\omega_o, h)} (F_0 * (1 - \alpha)) n \cdot \omega_i d\omega_i + \int\limits_{\Omega} \frac{f_r(p, \omega_i, \omega_o)}{F(\omega_o, h)} (\alpha) n \cdot \omega_i d\omega_i$$

BRDF Integration Map

- Pull out F0 of integral:

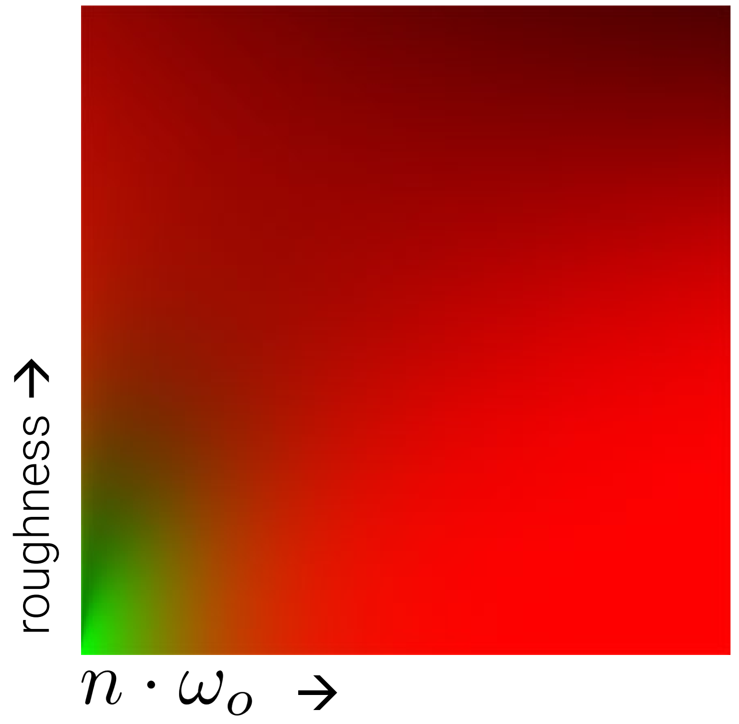
$$F_0 * \boxed{\int_\Omega \frac{f_r(p,\omega_i,\omega_o)}{F(\omega_o,h)}(1-\alpha)n\cdot\omega_i d\omega_i} + \boxed{\int_\Omega \frac{f_r(p,\omega_i,\omega_o)}{F(\omega_o,h)}(\alpha)n\cdot\omega_i d\omega_i}$$

- So it has the general form:

$$F_0 * a + b$$

2D BRDF LUT

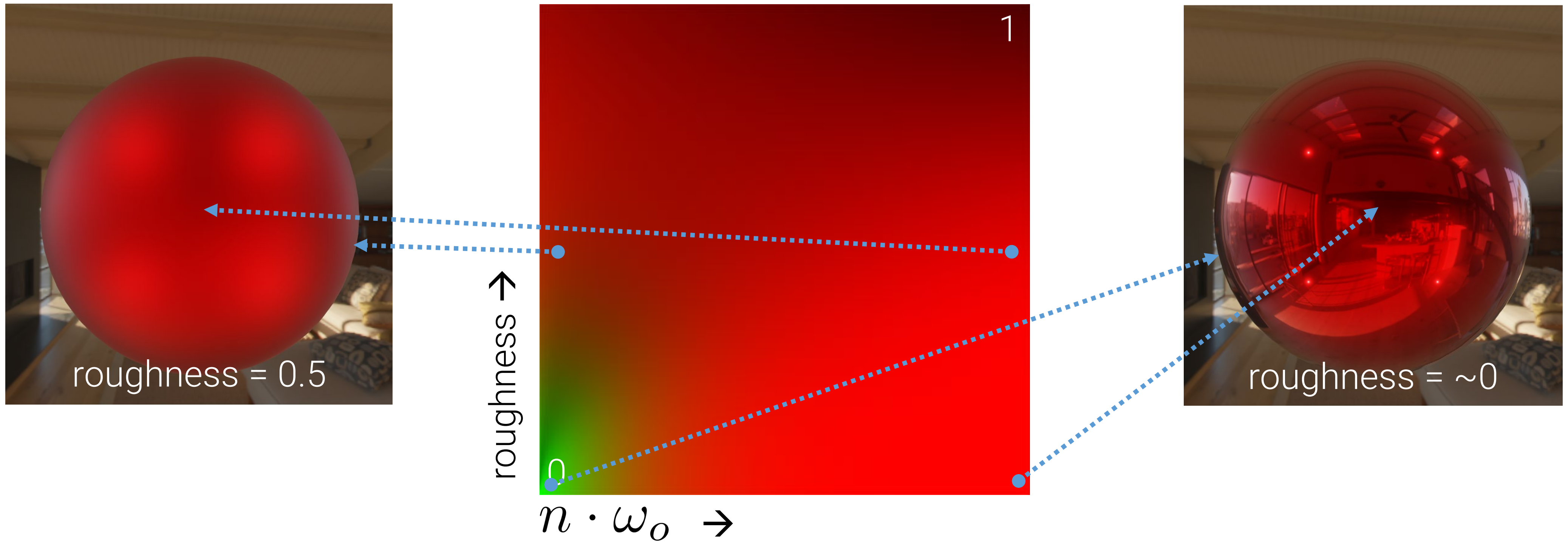
- Precompute with importance sampling and store $\boldsymbol{a}$(red) and $\boldsymbol{b}$(green) in texture:



$$F_0 * a + b$$

2D BRDF LUT

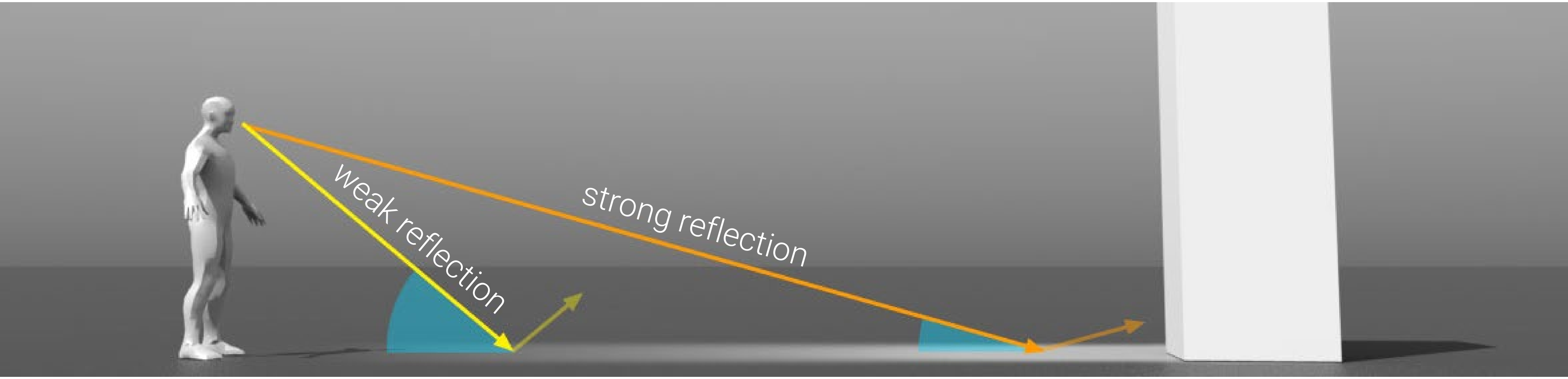
- Precompute with importance sampling and store a(red) and b(green) in texture:



Fresnel?

- There is no half vector (h)!
- Use normal and view vector instead.

$$F(v, h)$$



Fresnel?

$F(v, h)$

- There is no half vector (h)!
- Use normal and view vector instead.
- Roughness correction term
  (fewer reflections with high roughness).

GLSL Code:

```
vec3 fresnelSchlickRoughness(float cosTheta, vec3 F0, float roughness){
    return F0 + (max(vec3(1.0 - roughness), F0) - F0) * pow(1.0 - cosTheta, 5.0);
}
```

TASK #02: Specular part

- Let’s get specular IBL working
- Hints:
  - Only edit FS!
  - You need to sample the prefiltered environment map and the BRDF integration map, and integrate it in the ambient term.

Solution #02: GLSL Code

```glsl
vec3 F = FresnelSchlickRoughness(max(dot(N, V), 0.0), F0, roughness);
vec3 kS = F;
vec3 kD = 1.0 - kS;
kD *= 1.0 - metallic;
vec3 irradiance = texture(irradianceMap, N).rgb;
vec3 diffuse      = irradiance * albedo;

// IBL specular part.
const float MAX_REFLECTION_LOD = 4.0;
vec3 prefilteredColor = textureLod(prefilterMap, R, roughness *
       MAX_REFLECTION_LOD).rgb;
vec2 brdf  = texture(brdfLUT, vec2(max(dot(N, V), 0.0), roughness)).rg;
vec3 specular = prefilteredColor * (F * brdf.x + brdf.y);

vec3 ambient = (kD * diffuse + specular) * ao;
```

Useful Resources

- Importance Sampling for IBL: https://learnopengl.com/PBR/IBL/Specular-IBL
- Unreal Engine: https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf
- Trend Reed’s Blog: https://www.trentreed.net/blog/physically-based-shading-and-image-based-lighting/
- HDR Environment Maps: https://hdrihaven.com/hdris/

Exercise 04 – Your PBR material

Find an object with an interesting appearance (surface, texture, roughness, …) and take a picture of it. Take multiple images in different lighting situations. Ideal are objects with different and interesting materials (e.g., plastic, wood, metal, rust, …).

Then try to recreate the object's appearance as a physically-based-rendering material and use it in our example application. Define the textures required: roughness, metallness, albedo, [if needed: AO, normal map].

You can use a model (new or reuse Exercise#03 or #04), or simply use a flat plane (quad).

Work in teams and upload a PDF for the progress report.

Questions?

david.schedl@ fh-hagenberg.at