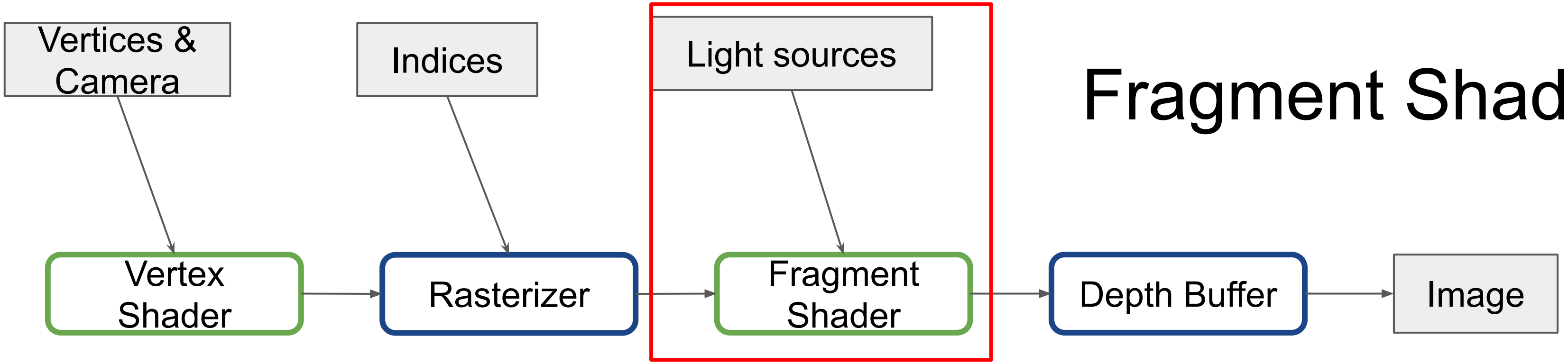
3D Graphics

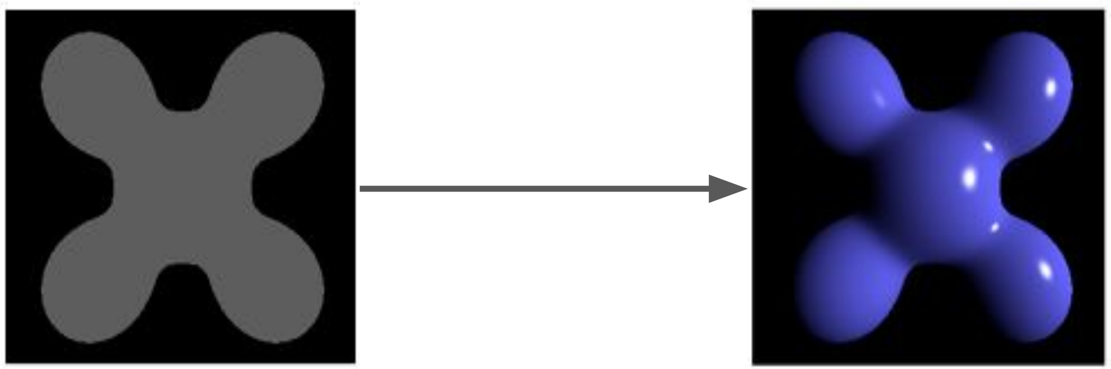
The rendering pipeline

Fragment Shader

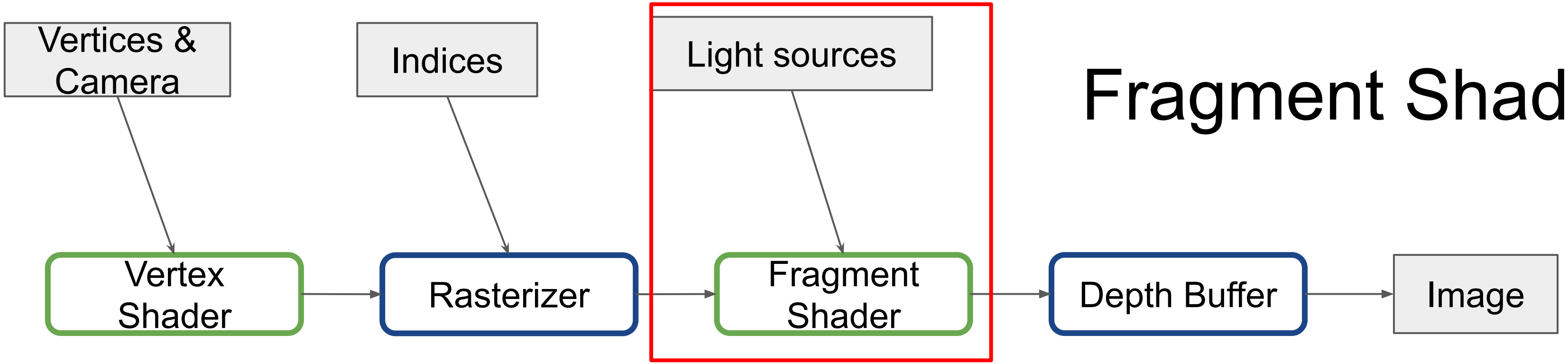
Fragment Shader



Compute the color to give each fragments



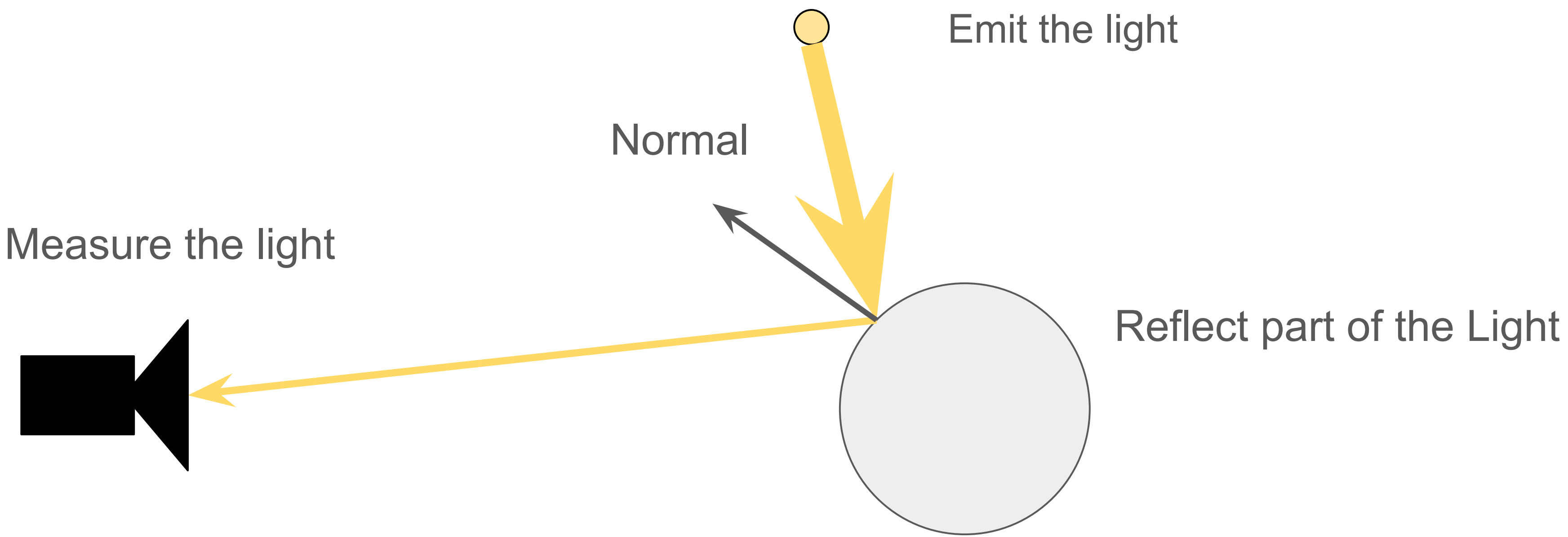
Fragment Shader



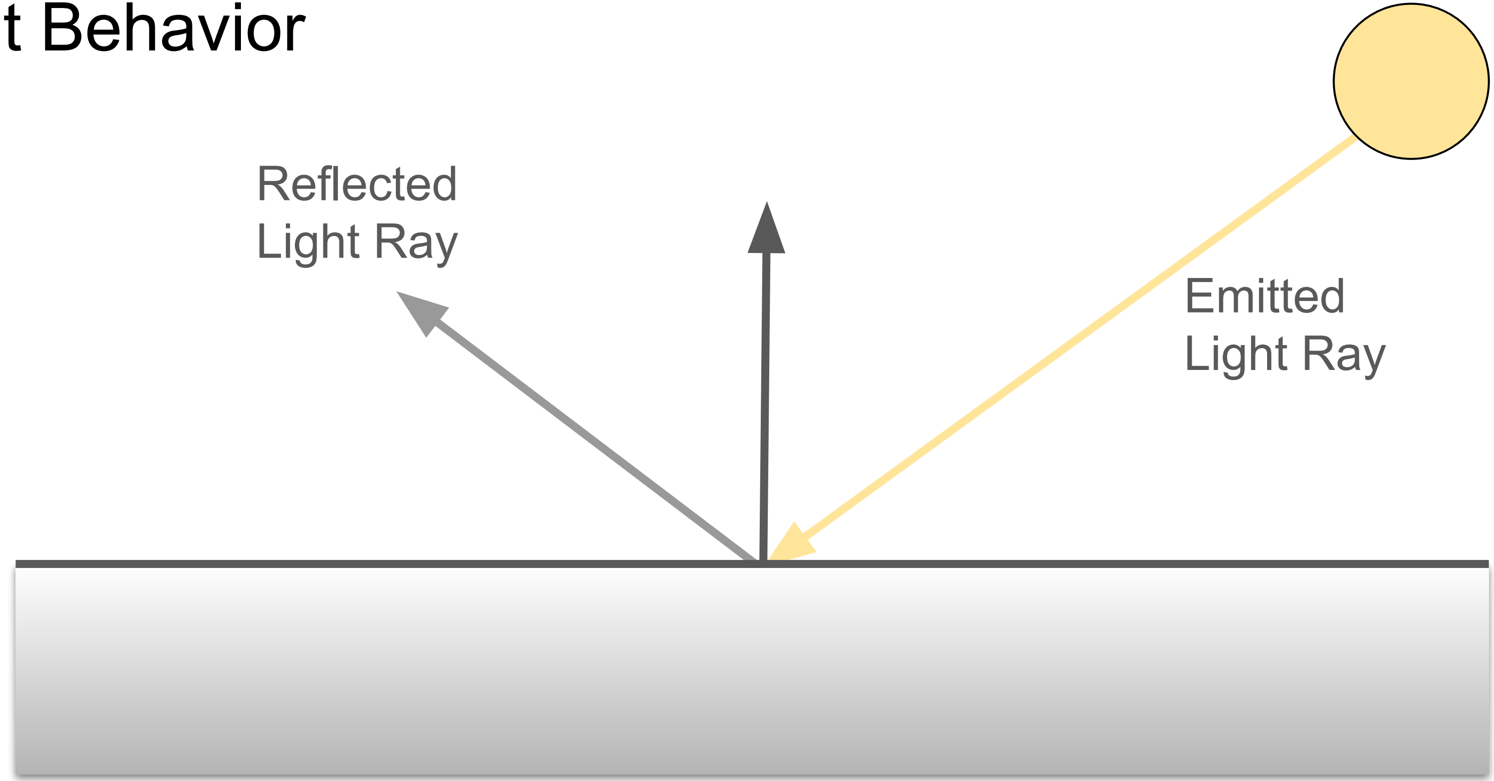
Take a fragment as an input, and output a new fragment with color information

Simulate how material interact with light

We want to measure the light arriving at the camera

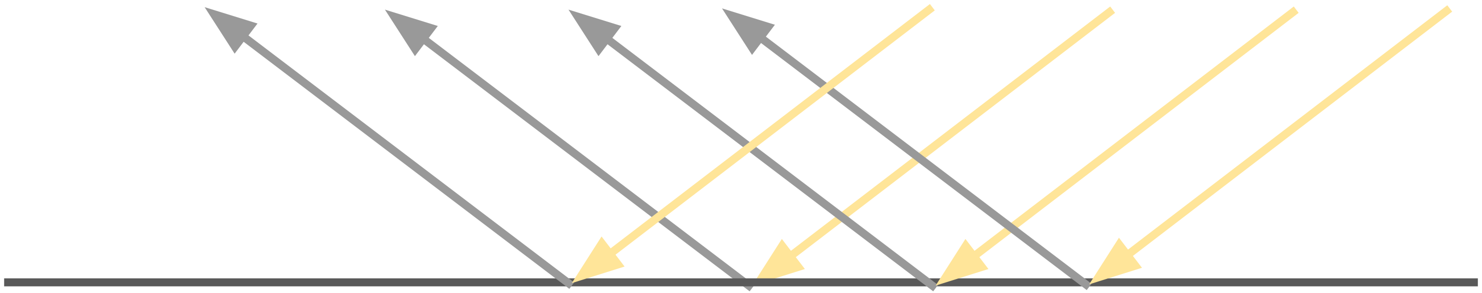


Light Behavior



Specular material

Parallel emitted light ray stay parallel once reflected

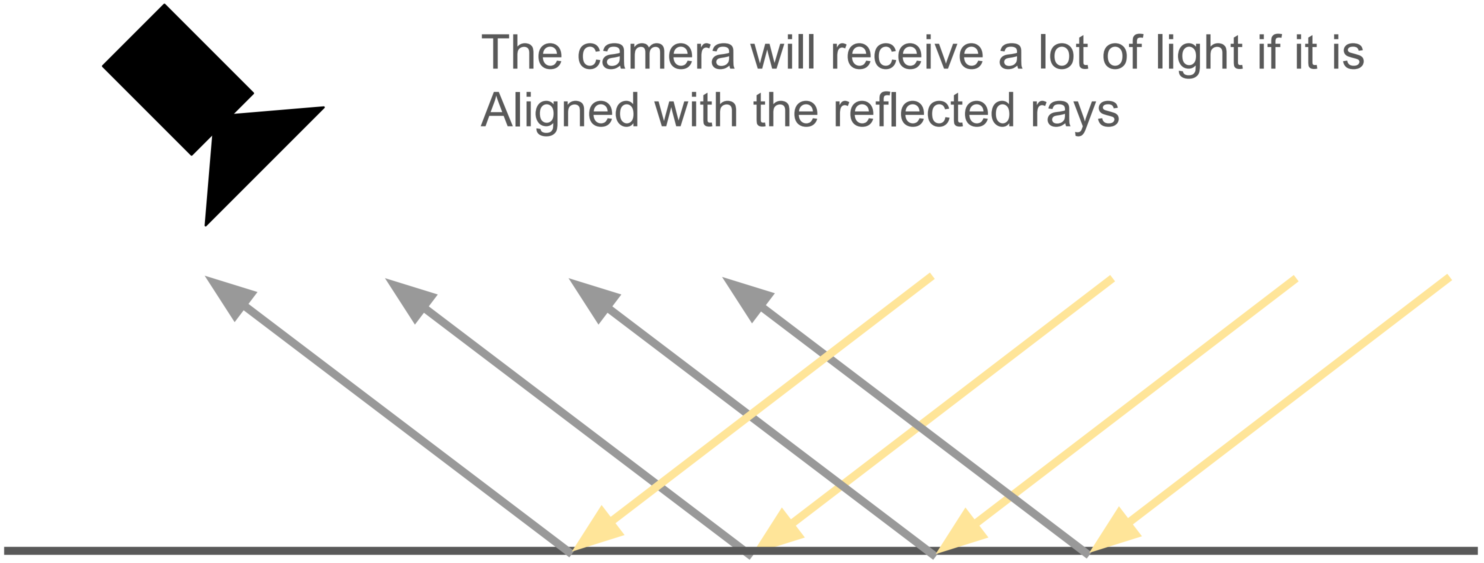


Specular material

Smooth surfaces

Parallel emitted light ray stay parallel once reflected

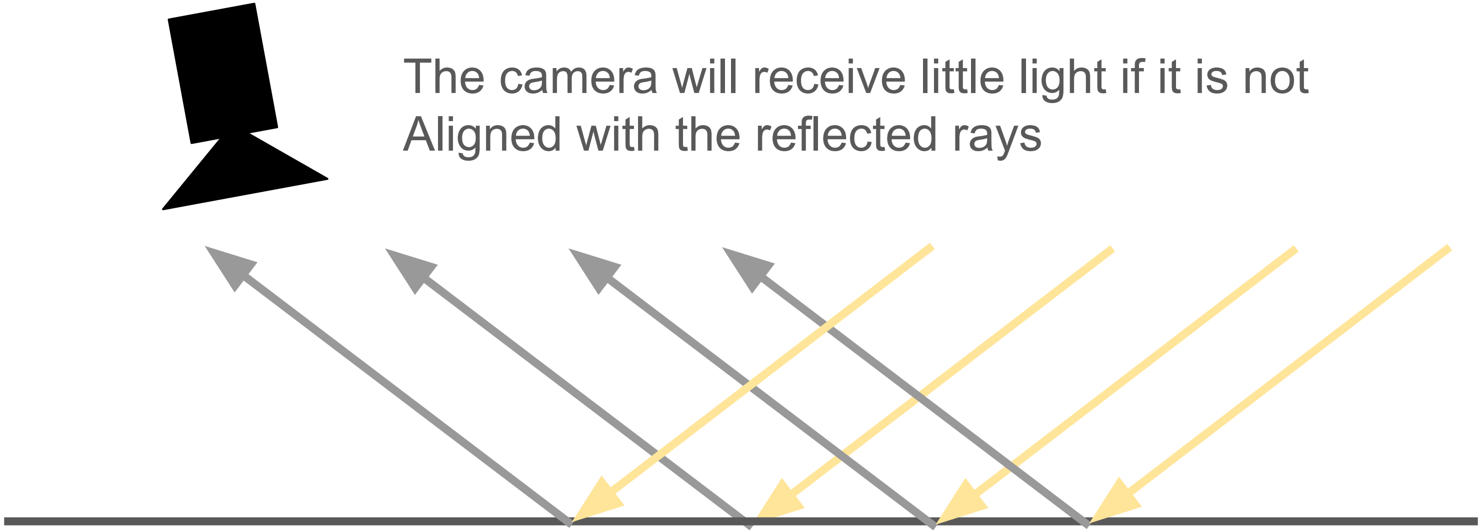
The camera will receive a lot of light if it is
Aligned with the reflected rays



Specular material

Parallel emitted light ray stay parallel once reflected

The camera will receive little light if it is not
Aligned with the reflected rays



Example

Specular

Question

What are real life specular surfaces ?

(1 minute alone)

(2 minutes with your neighbors)

(5 minutes with the whole group)

Example of specular surfaces

Polished metal

Example of specular surfaces

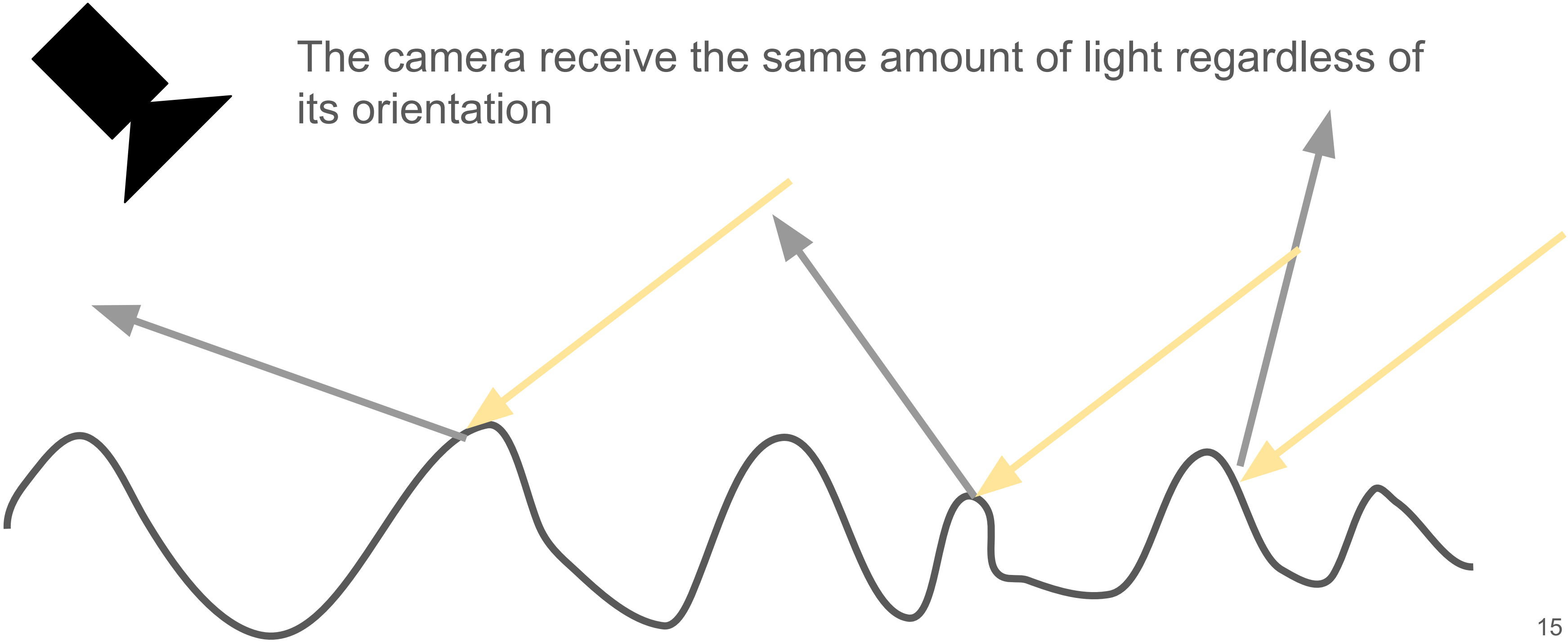
Polished surfaces

Example of specular surfaces

waxed parquet

Diffuse material

The surface is rough at microscopic level, reflected rays are radom

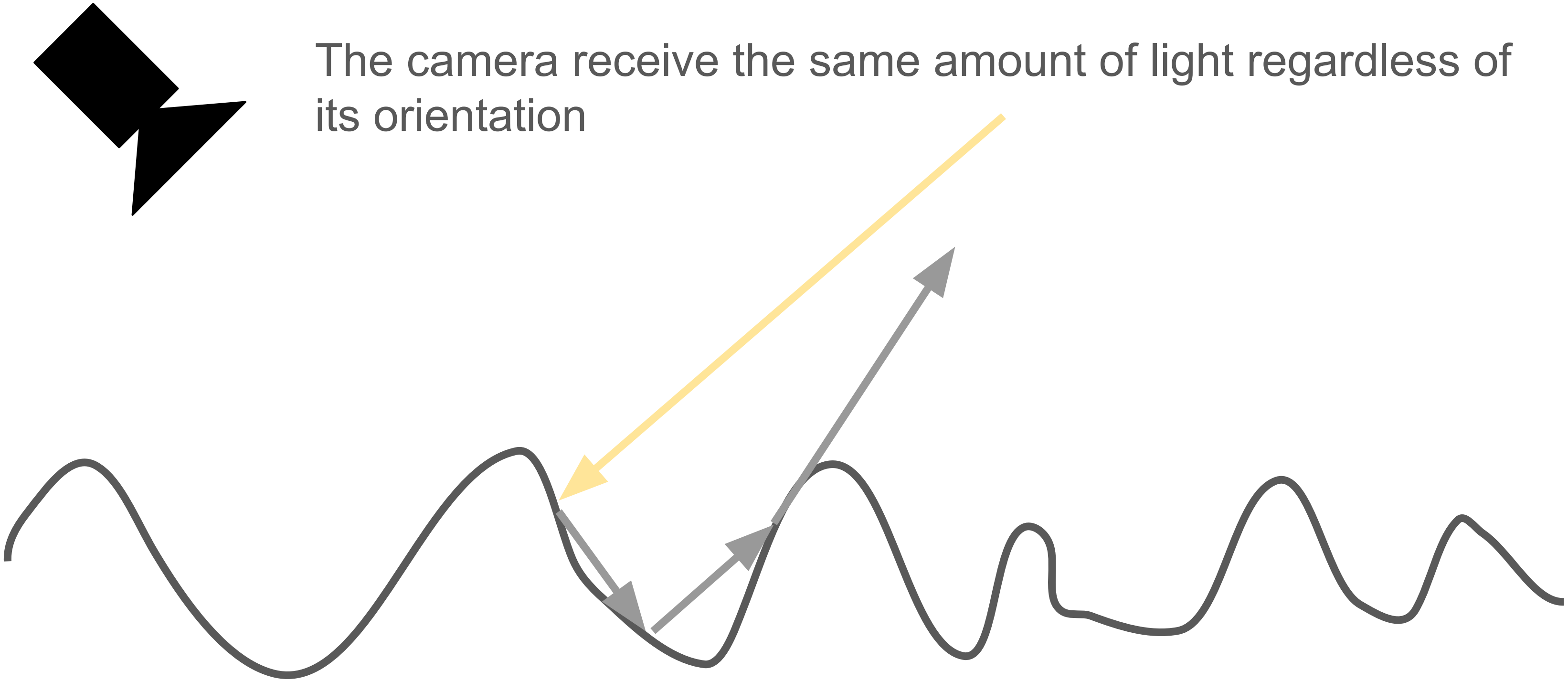


The camera receive the same amount of light regardless of its orientation

Diffuse material

The surface is rough, reflected rays are radom

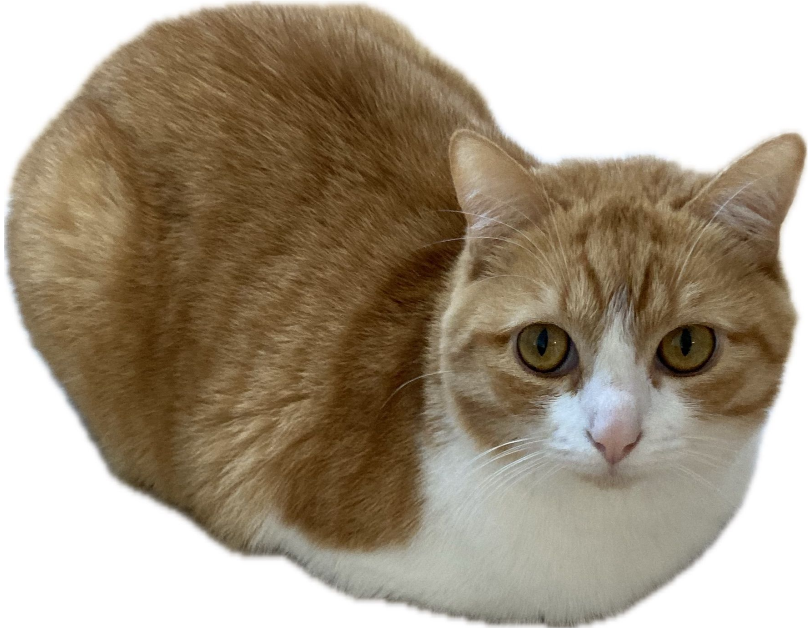
The camera receive the same amount of light regardless of its orientation



Example of diffuse surfaces

Rusted metal

Example of diffuse surfaces



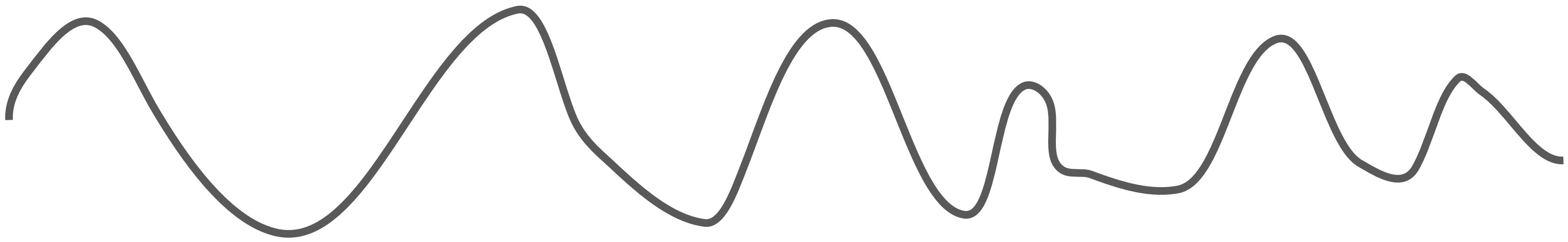
Wool, fabric, fur

dry

wet

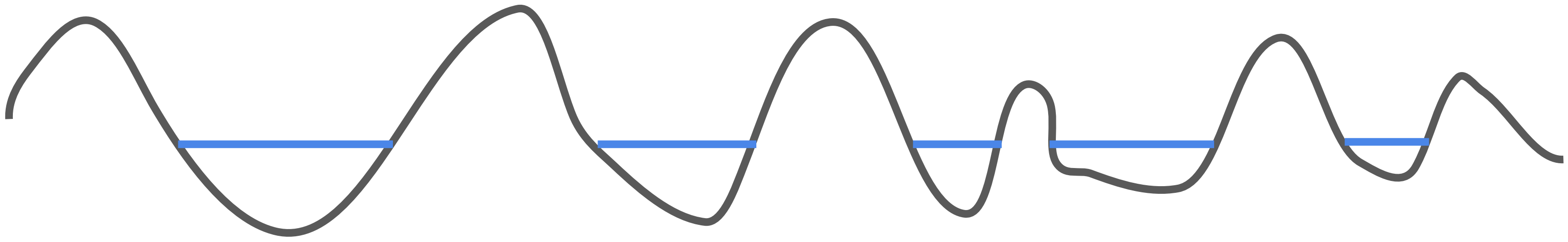
Why wet surfaces appear more specular that dry one?
(1 minute alone)
(2 minutes with your neighbors)
(5 minutes with the whole group)

Dry material

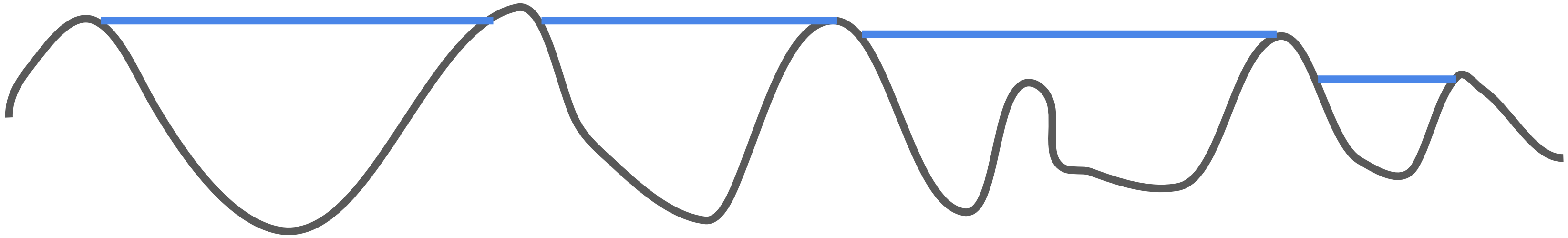


Wet material

Water fill the rugosity of the material

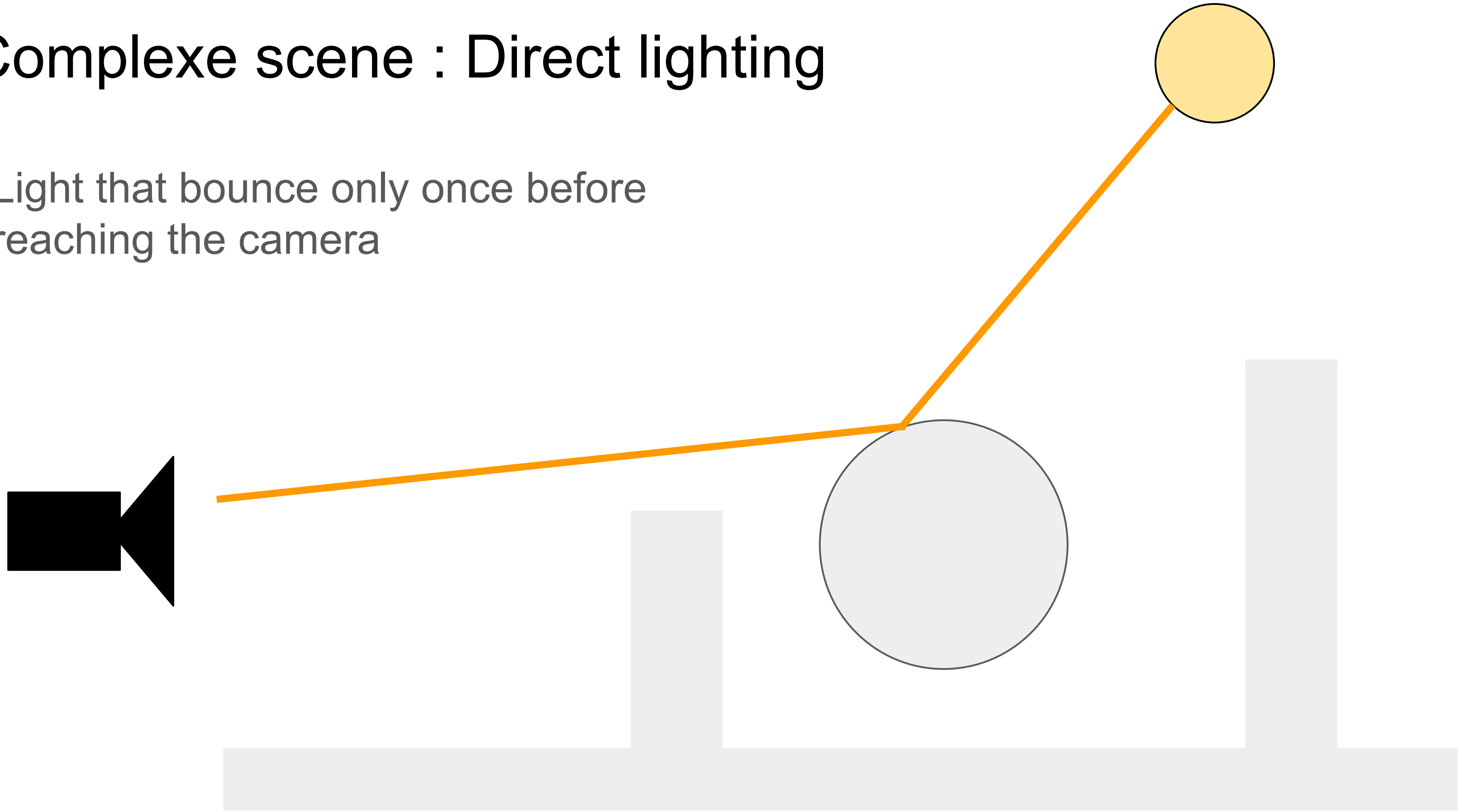


The more water present in a material the smoother the surface will appear



Complexe scene : Direct lighting

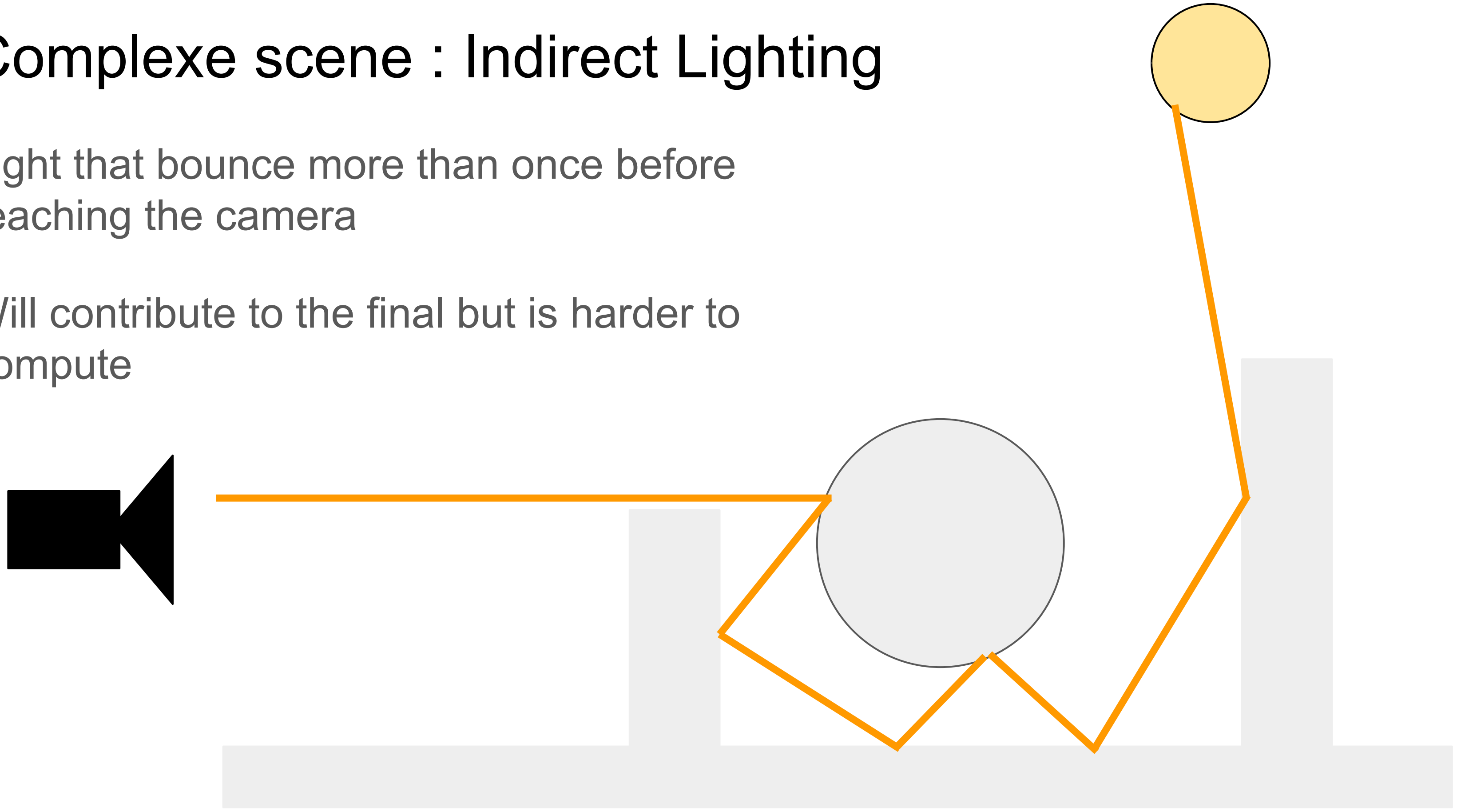
Light that bounce only once before reaching the camera



Complexe scene : Indirect Lighting

Light that bounce more than once before reaching the camera

Will contribute to the final but is harder to compute



Indirect Lighting Simplification : ambient lighting

We replace indirect lighting with a value for the constant for all the scene

Ambient Lighting ≈ average color in the scene

Question

Is it possible for a scene to have only indirect Lighting ?
If so what would this scene look like ?

(1 minute alone)

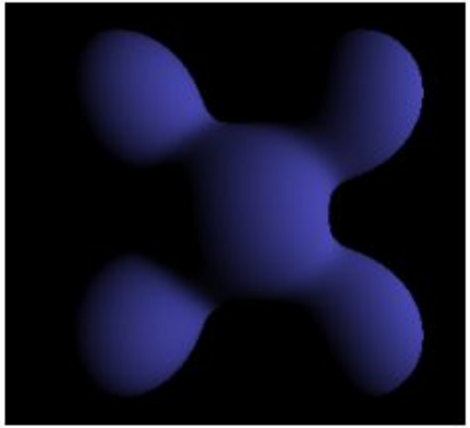
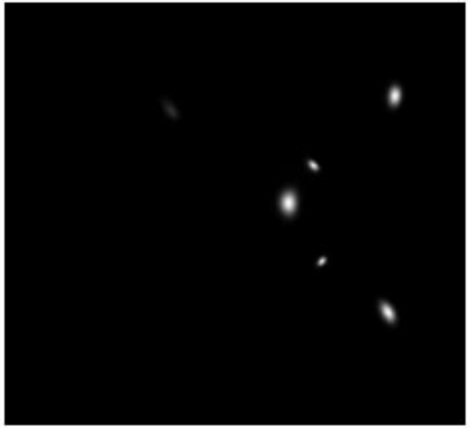
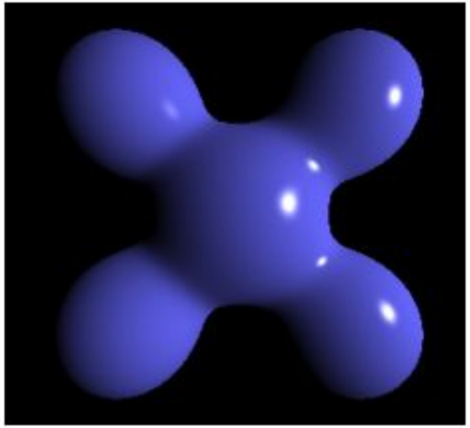
(2 minutes with your neighbors)

(5 minutes with the whole group)

Densely Clouded scene

Almost no variation of color

Phong Model

Ambient Light + Diffuse Light + Specular Light = Phong model

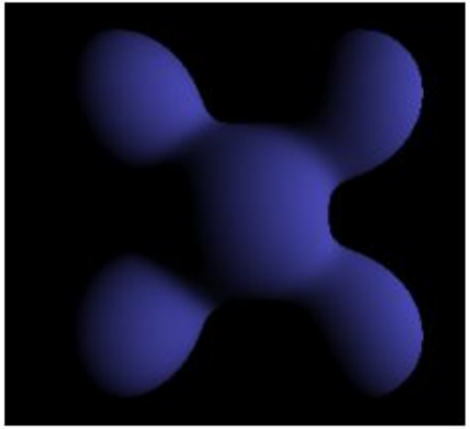
Phong Model : Ambient Light

$$Ambient = i_a$$

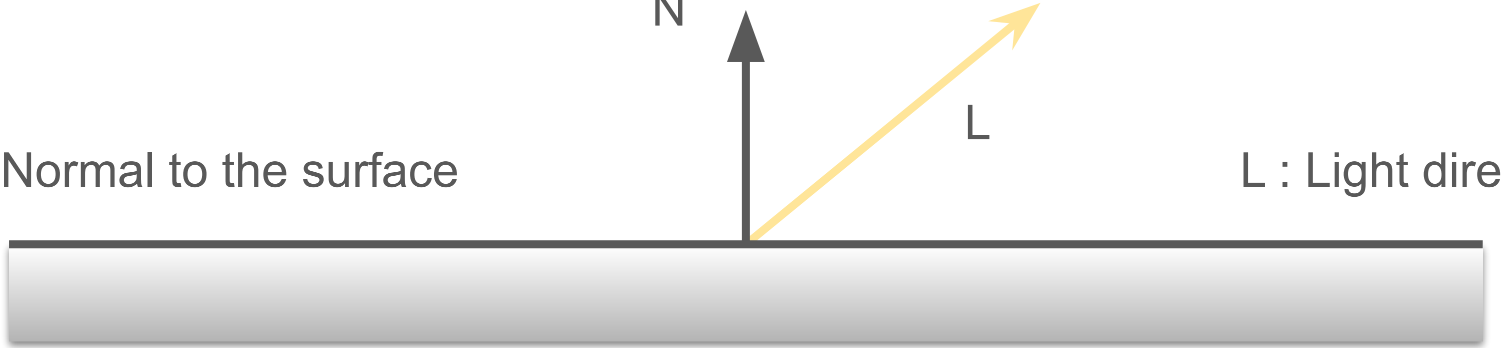
Ambient Light

Phong Model : Diffuse Light

$$Diffuse = max\left(dot\left(\overrightarrow{L},\overrightarrow{N}\right),0\right)$$



Diffuse Light



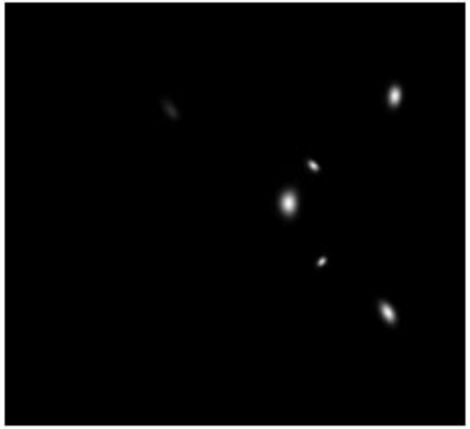
N Normal to the surface

L : Light direction

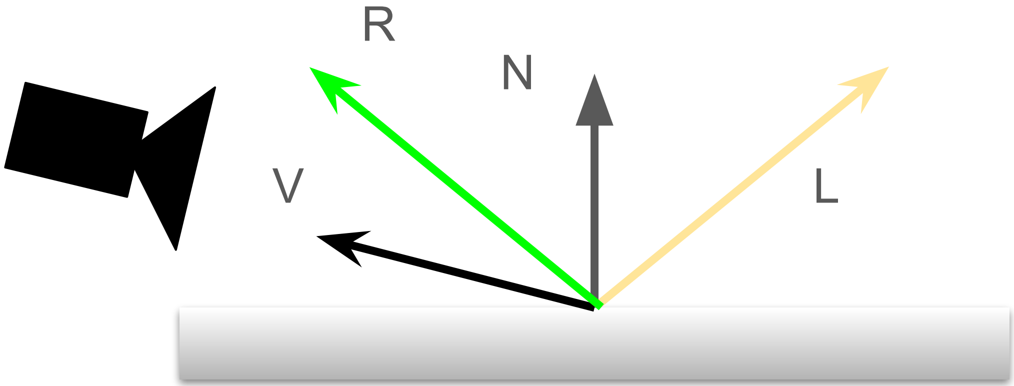
All vectors are normalized

Phong Model : Specular Light

$$Specular = max\left(dot\left(\overrightarrow{R}, \overrightarrow{V}\right), 0\right)^{\alpha}$$



Specular Light



V : View vector, the direction toward the camera

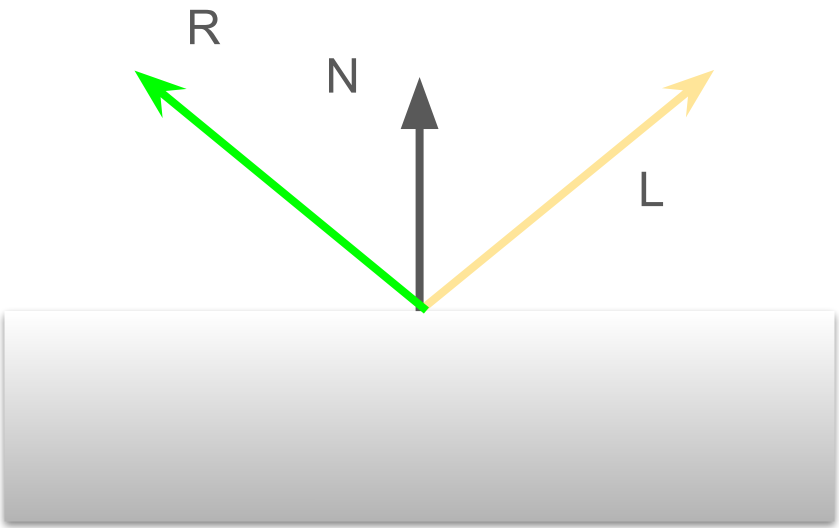
R : reflected vector direction

α : shininess of the material

All vectors are normalized

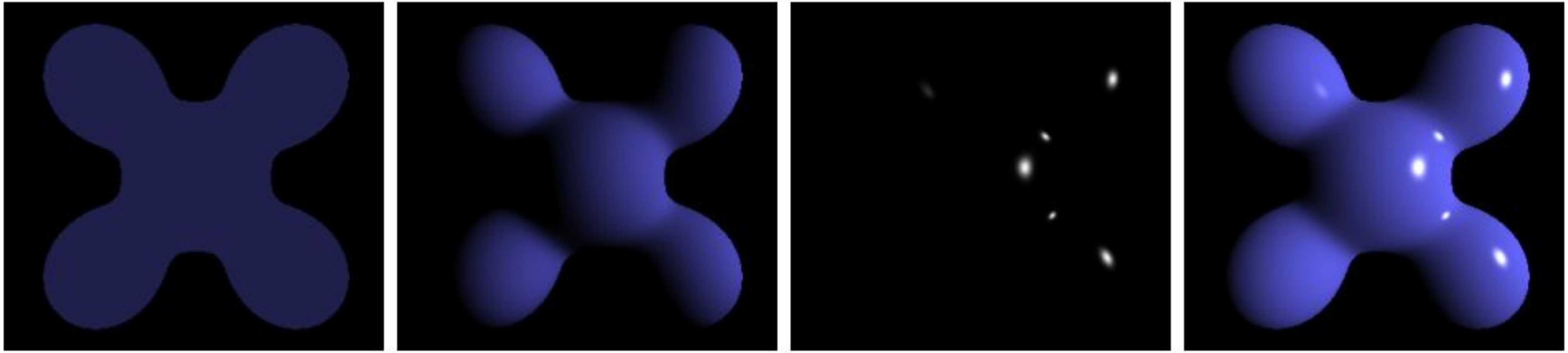
Reminder

$$\overrightarrow{R} = 2 * dot\left(\overrightarrow{L}, \overrightarrow{N}\right)\overrightarrow{N} - \overrightarrow{L}$$



All vectors are normalized

Phong Model : Mixing



$$phong = Ambient * k_a + Diffuse * k_d + Specular * k_s$$

$k_a \; k_d \; k_s$ ⟶ Mixing coefficient

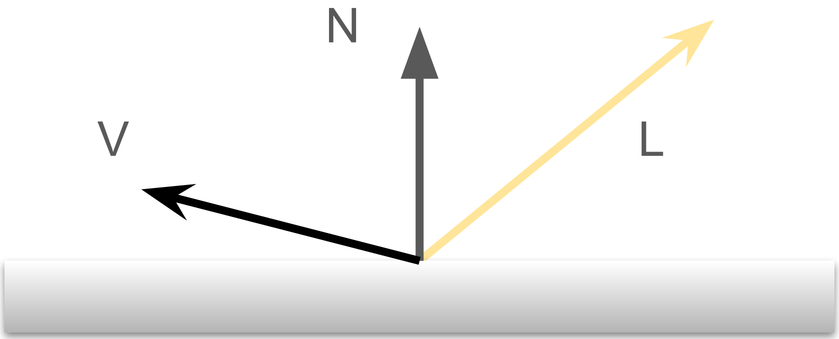
Problem :

Currently we only know how to compute :

- ✓ The fragment position in screen
- ✓ The fragment depth

We need to know :

V,N,L

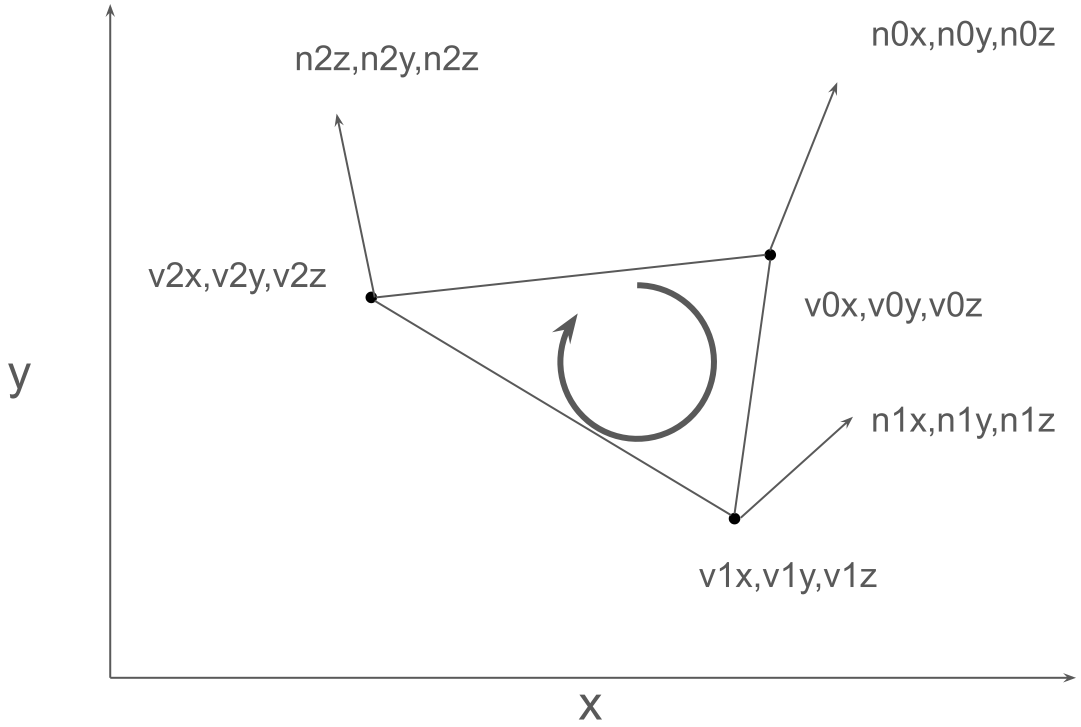


Normals : stored in vertices

Vertices list
{
v0x,v0y,v0z, n0x,n0y,n0z
v1x,v1y,v1z, n1x,n1y,n1z
v2x,v2y,v2z, n2z,n2y,n2z
}

Indices list
{0,1,2}

Unchanged



Vertex shader

Vertex shader

$$v_{tmp} = [proj]\,[view]\,v$$

$$v' = \frac{v_{tmp}}{v_{tmp}.w}$$

$$\overrightarrow{N} = n$$

$$\overrightarrow{V} = cameraPosition - v$$

$$\overrightarrow{L} = lightPosition - v$$

Vertex shader

Vertex shader

$$v_{tmp} = [proj]\,[view]\,v$$

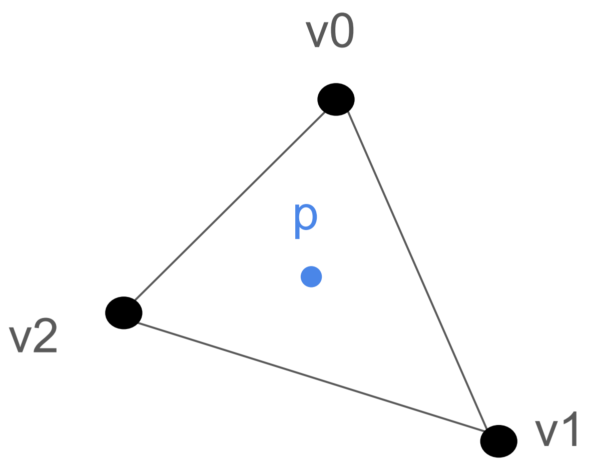
$$v' = \frac{v_{tmp}}{v_{tmp}.w}$$

Vertex Attributes

$$\boxed{\begin{aligned}\overrightarrow{N} &= n \\ \overrightarrow{V} &= cameraPosition - v \\ \overrightarrow{L} &= lightPosition - v\end{aligned}}$$

Rasterizer

$$p = \lambda_0 v0 + \lambda_1 v1 + \lambda_2 v2$$



$$f.interpollated = \lambda_0 v0.attributes + \lambda_1 v1.attributes + \lambda_2 v2.attributes$$

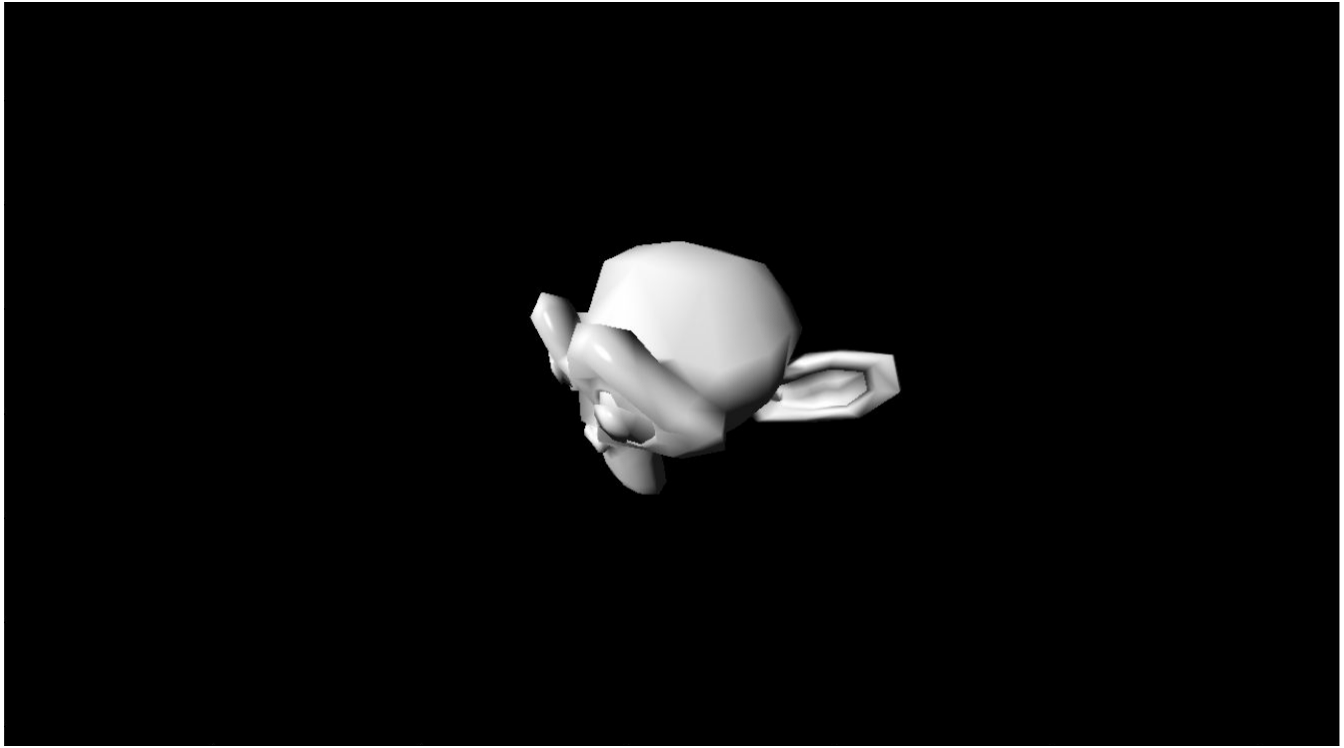
Warning : for this operation we need to use 3D barycentric coordinates

Fragment Shader

$$phong = Ambient * k_a + Diffuse * k_d + Specular * k_s$$

$$fragmentColor = objectColor * phong$$

Results



Next lecture : flipped classroom

In this lecture is dedicated to question answering

To prepare :

Read all previous lectures

Look at all the previous practical

Come with questions