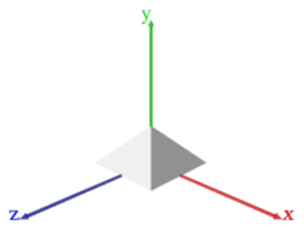
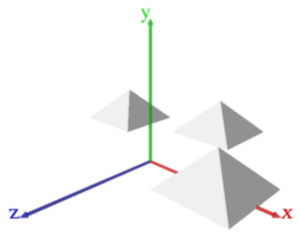
GRK 5

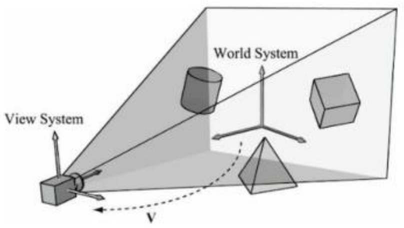
dr Wojciech Palubicki

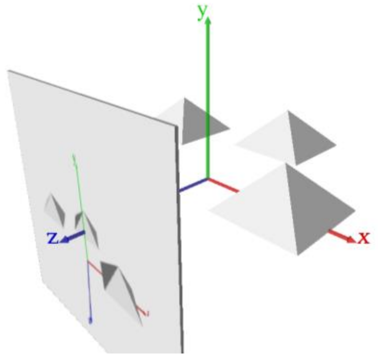
Simplified Rendering Pipeline

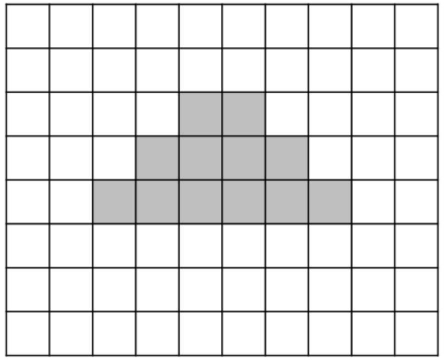
Model Matrix → View Matrix → Projection Matrix → Viewport Transform











Object Space

World Space

View Space

Clip Space

Screen Space

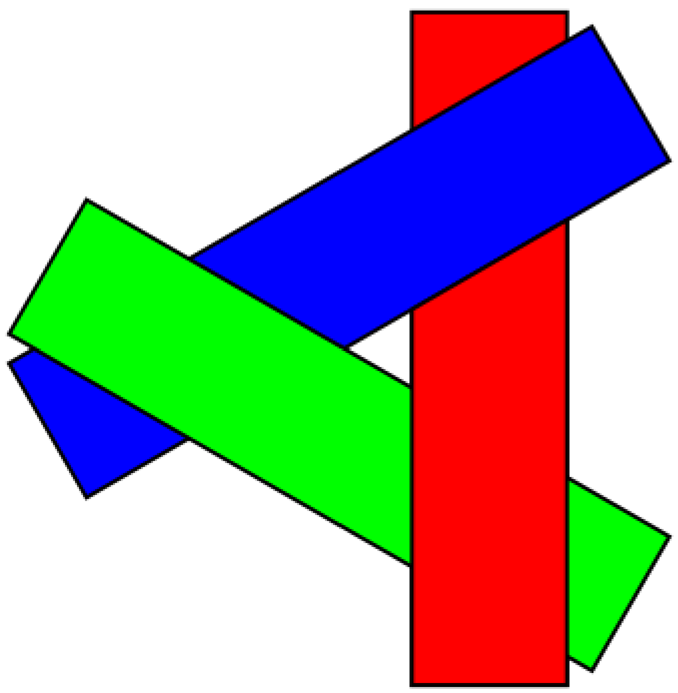
Drawing on the Display

- The following pseudo-code illustrates how to draw a line between two points a and b

compute M_{vp}
compute M_{per}
compute M_{cam}
$M = M_{vp} M_{per} M_{cam}$

for each line segment (a, b) do
    p = M a
    q = M b
    drawline(p_x/p_w, p_y/p_w, q_x/q_w, q_y/q_w)

Partial polygon overlay



z-buffer / depth buffer

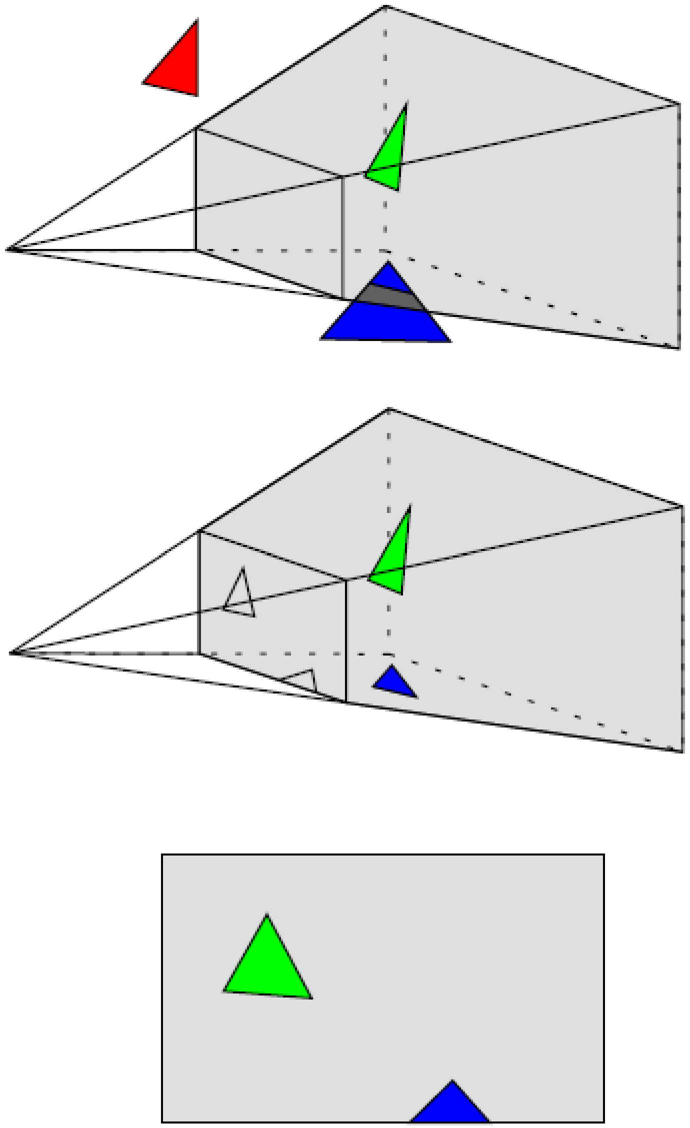
Algorithm 1 Z buffer

Require: a set of polygons P, a depth buffer array Z and a frame buffer array F

initialise Z to $z_{\max}$
for all polygons in P **do**
  for all pixels in the current polygon **do**
    calculate the z co-ordinate of the point corresponding to the current pixel
    if $z < Z(x, y)$ **then**
      Replace $Z(x, y)$ with z
      Replace $F(x, y)$ with the colour of the current polygon
    end if
  end for
end for
Display F on screen

Clipping/Culling

- Triangles that lie (partly) outside of the view frustum don’t have to be projected and are culled (**clipping/culling**)
- The remaining triangles are projected on the view plane



Lighting

Without lighting

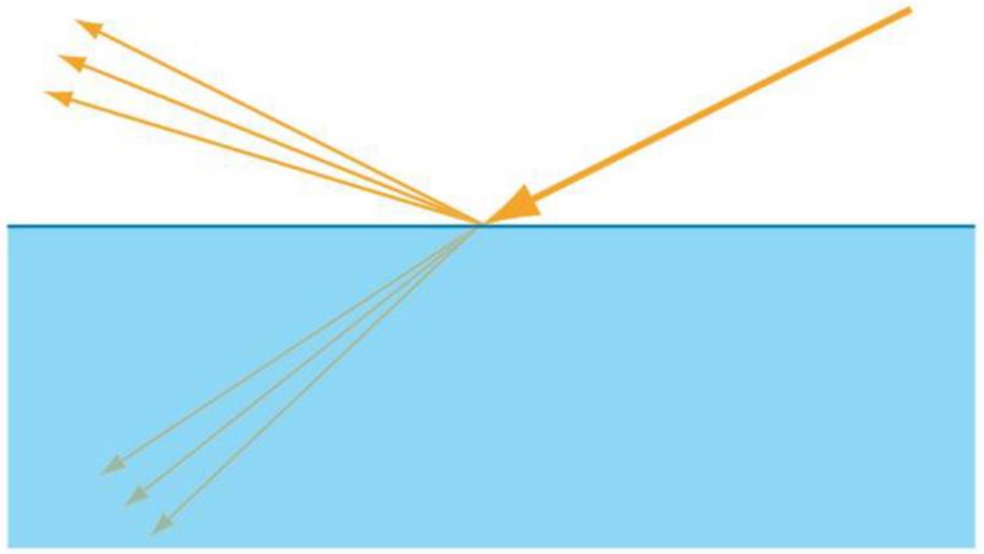
Lighting

Without lighting

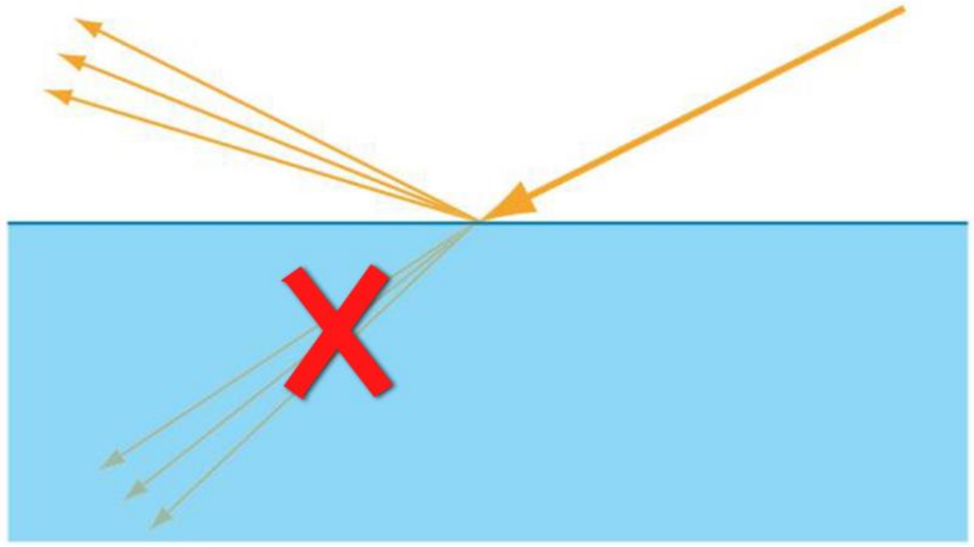
With lighting

Light

- Non-metal material
  - A portion of light is reflected
  - Another portion enters the material

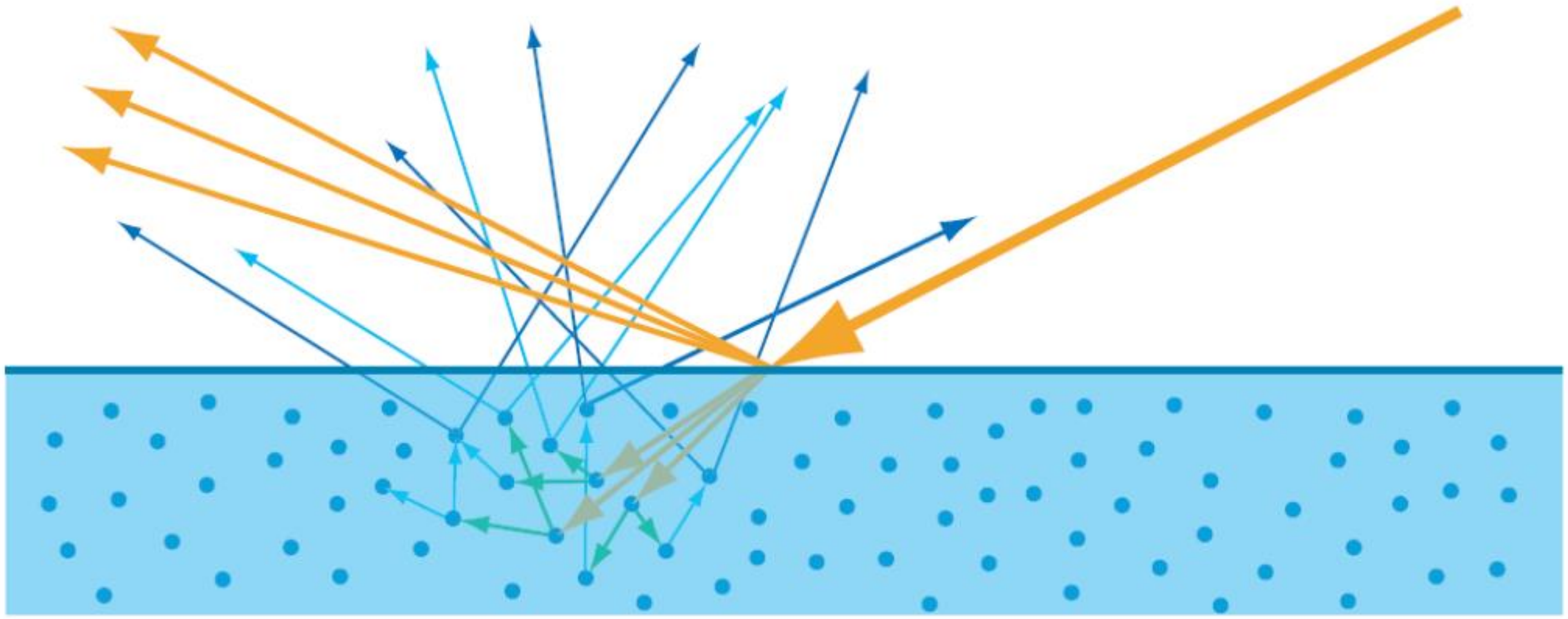


- Metal
  - A portion of light is reflected
  - Another portion is absorbed



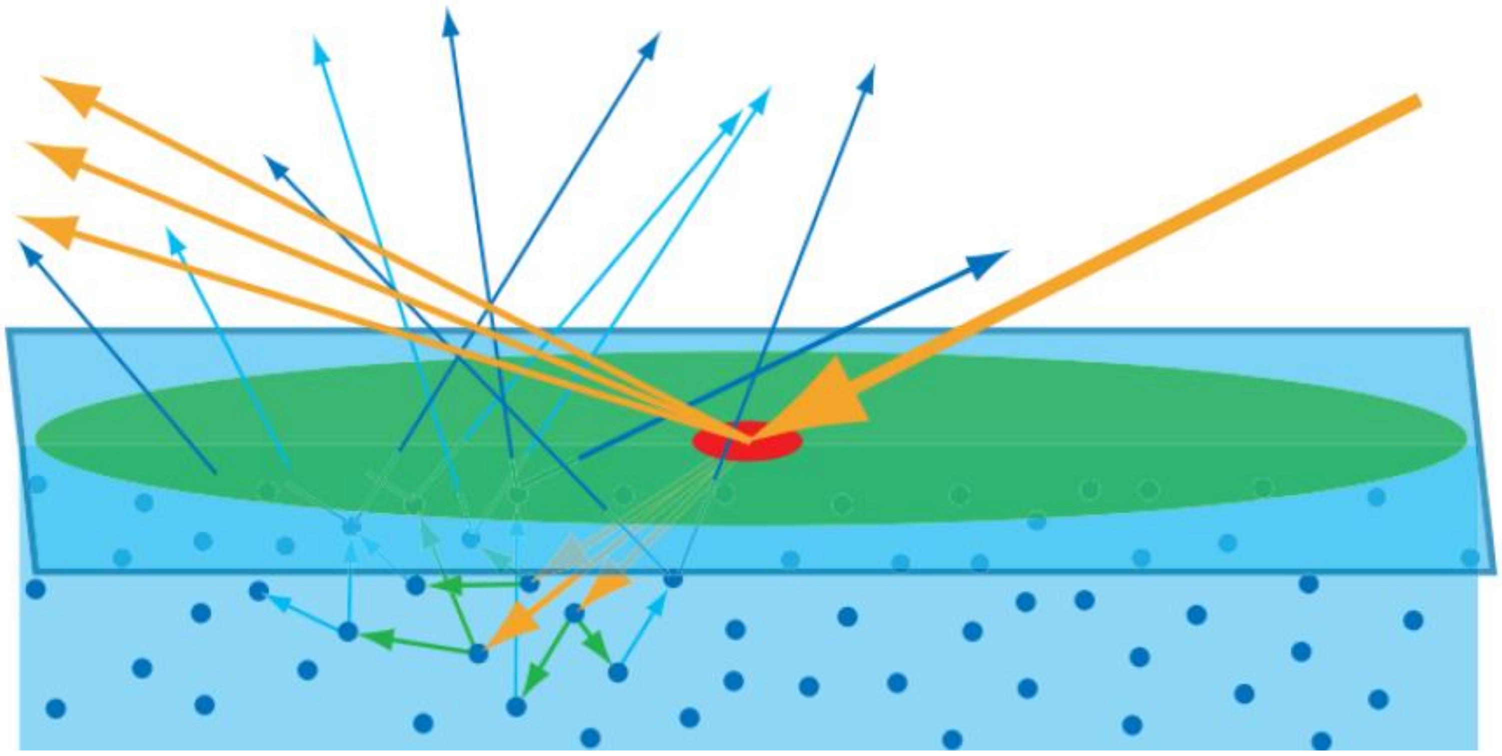
Non-metal materials

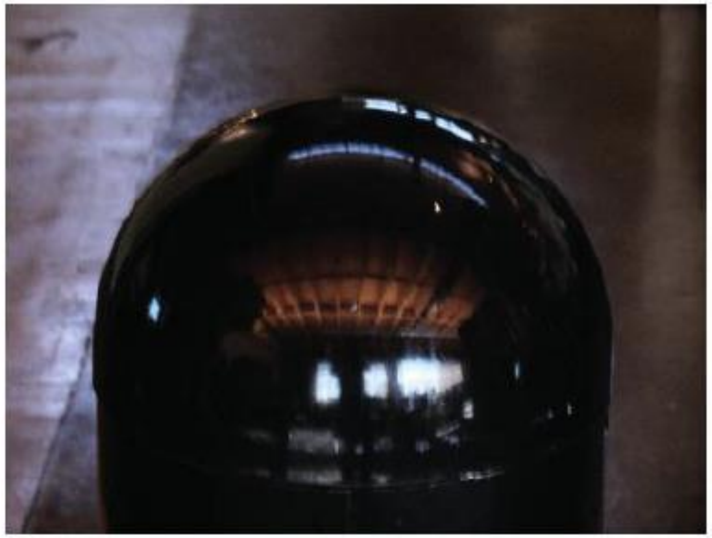
- Light that is entering the material
  - Is absorbed
  - Or re-emitted after a while

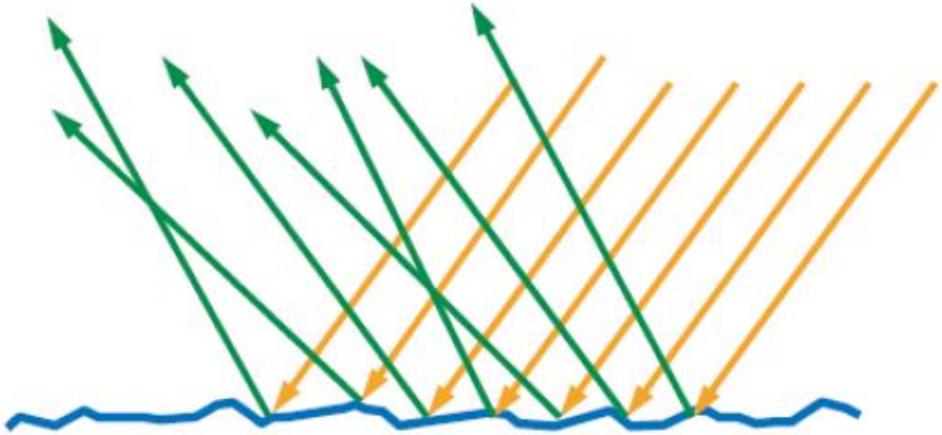


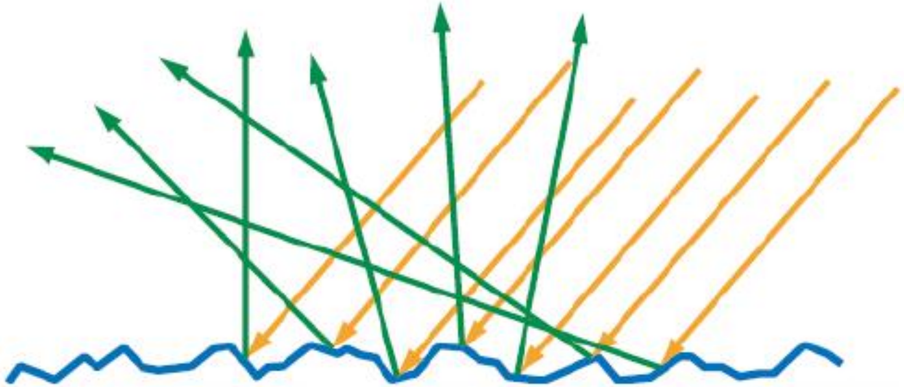
Subsurface scattering

- The distance between entry and exit points of light rays is specified by the material



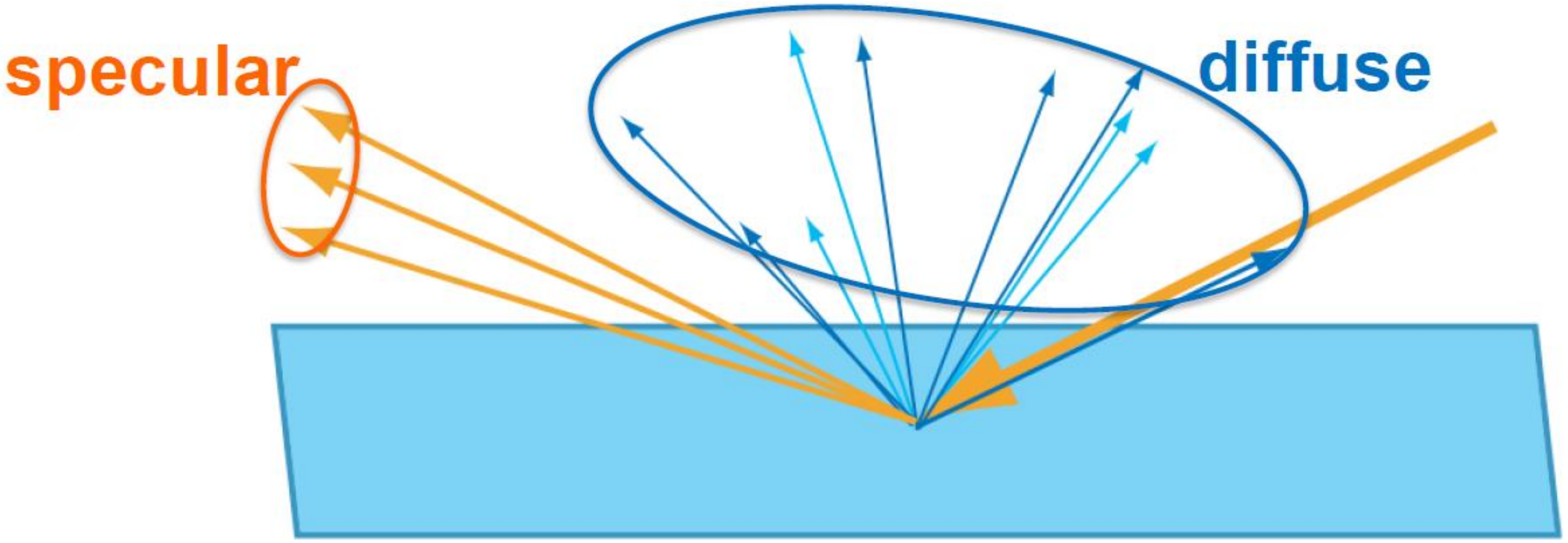


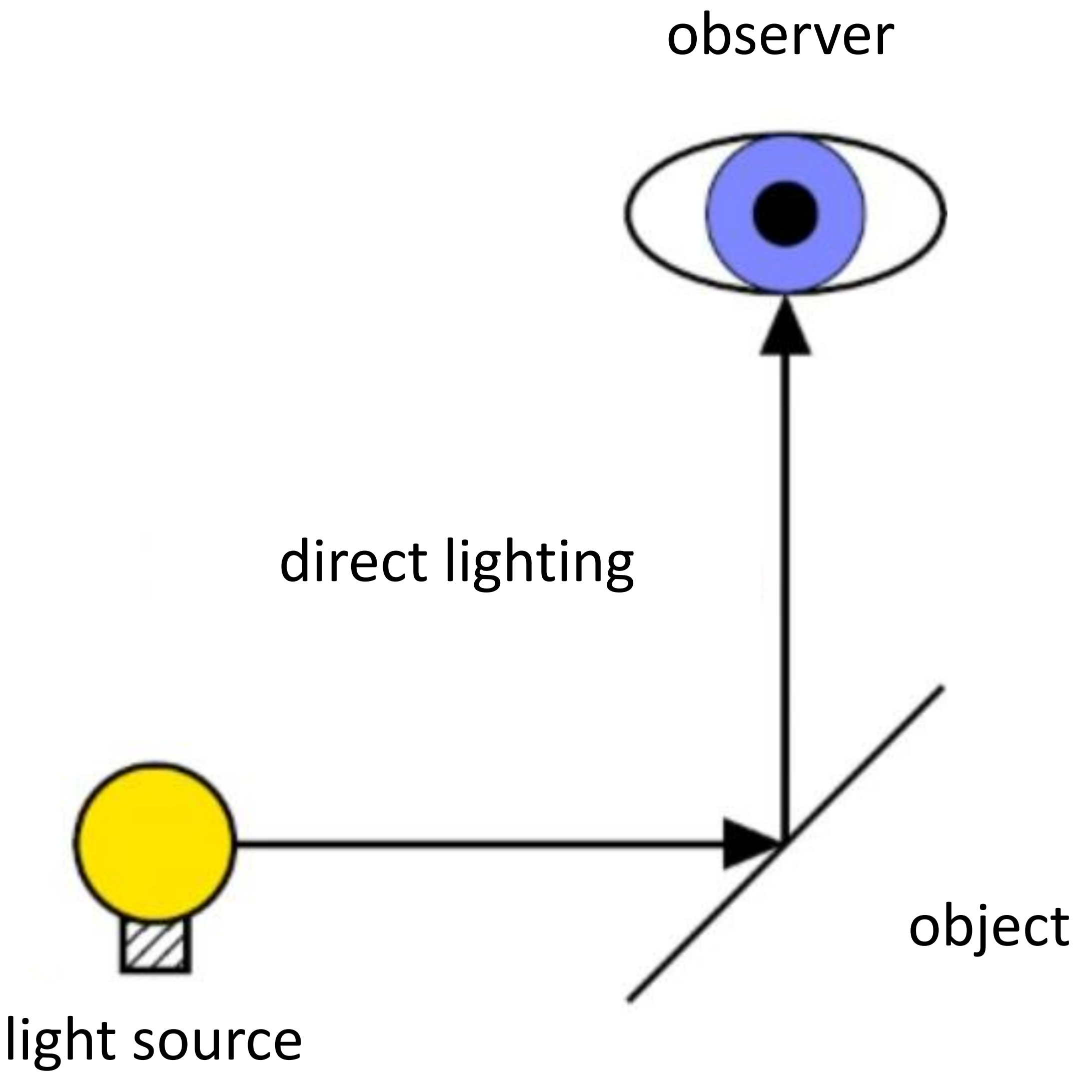


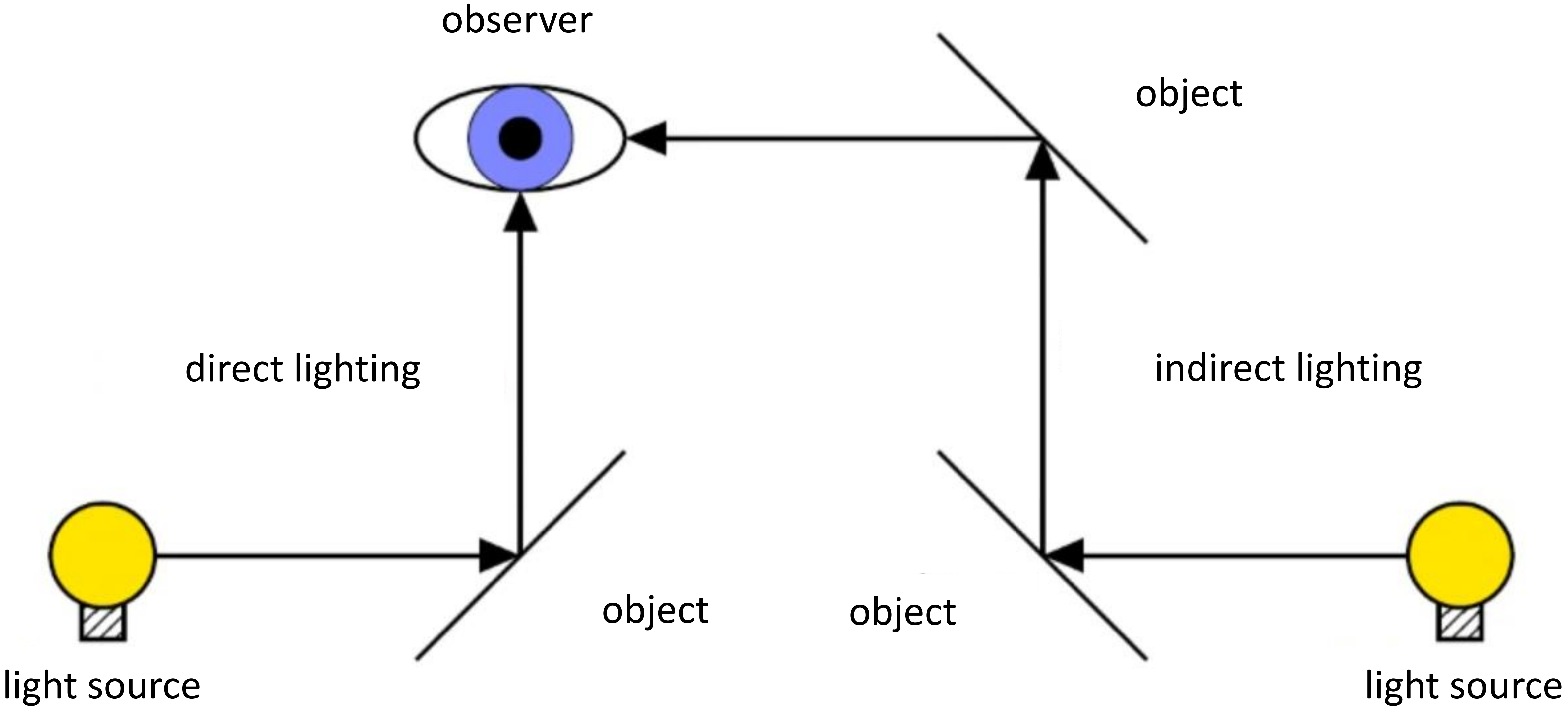


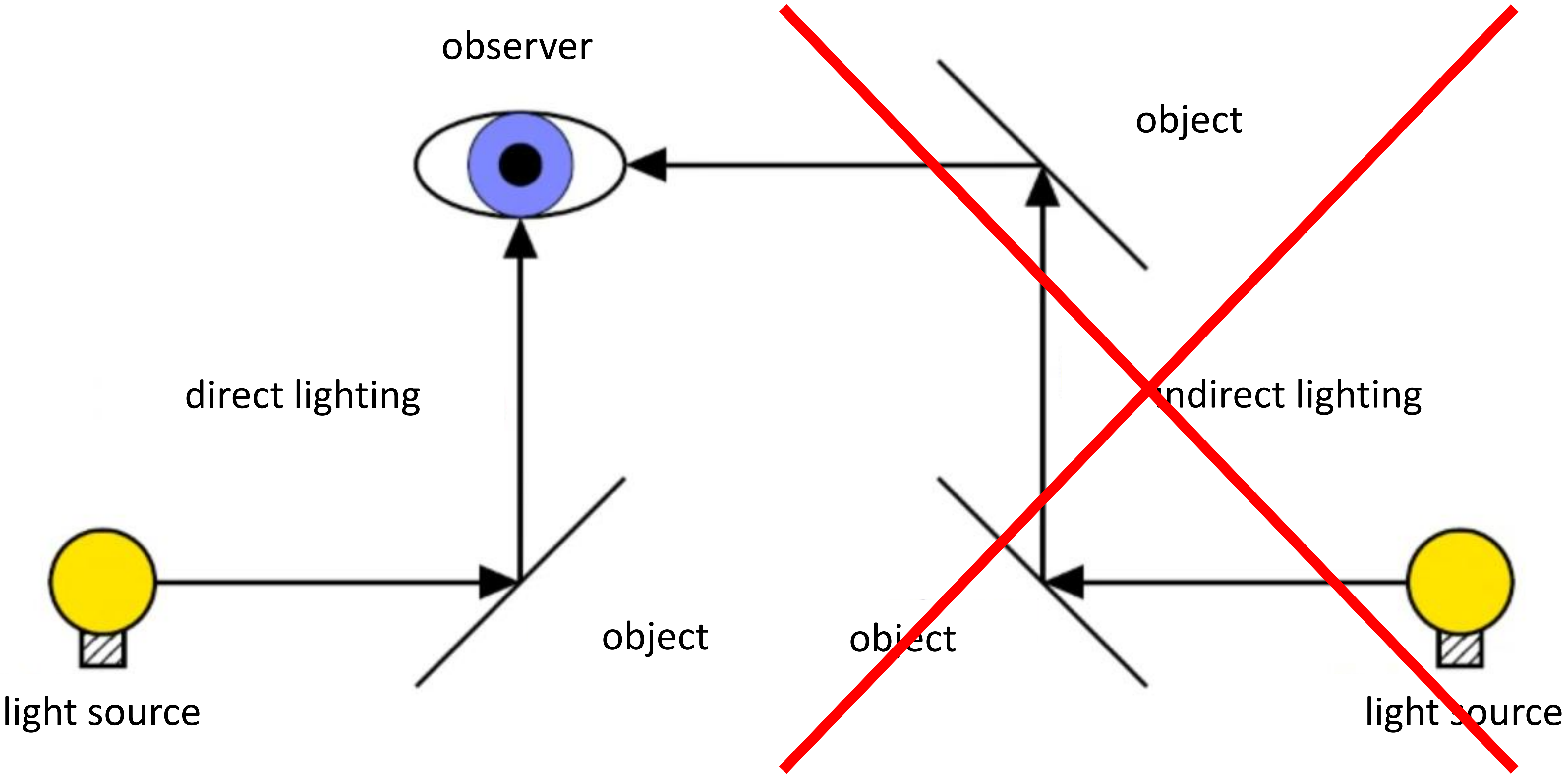
Modeling Light with BRDF’s

- BRDF stands for *Bidirectional Reflectance Distribution Function*
- BRDF’s are modeling light by ignoring the difference between entry and exit points of a material

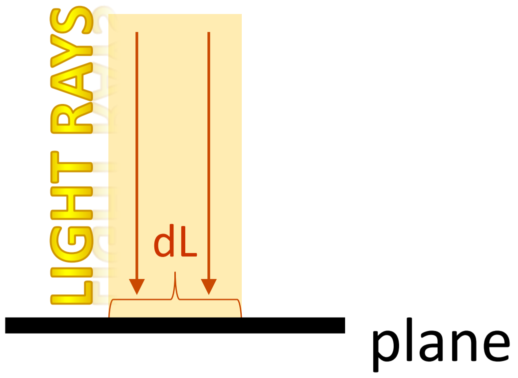



observer
direct lighting
object
light source

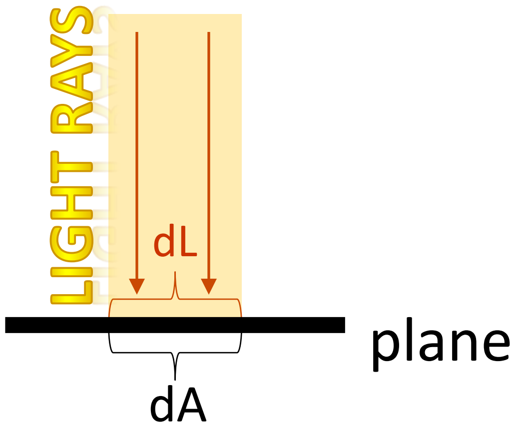

observer
object
direct lighting
indirect lighting
object
object
light source
light source


observer
object
direct lighting
indirect lighting
object
object
light source
light source

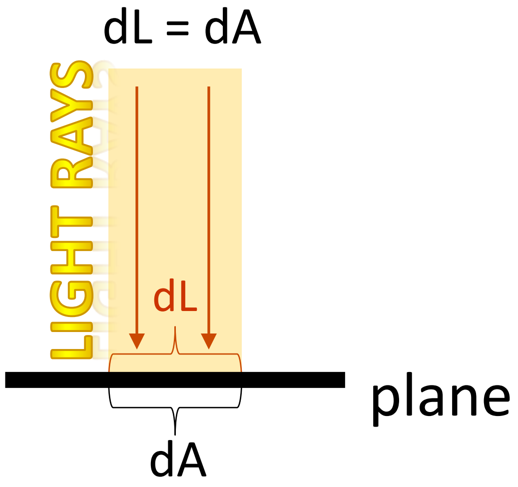
Observation



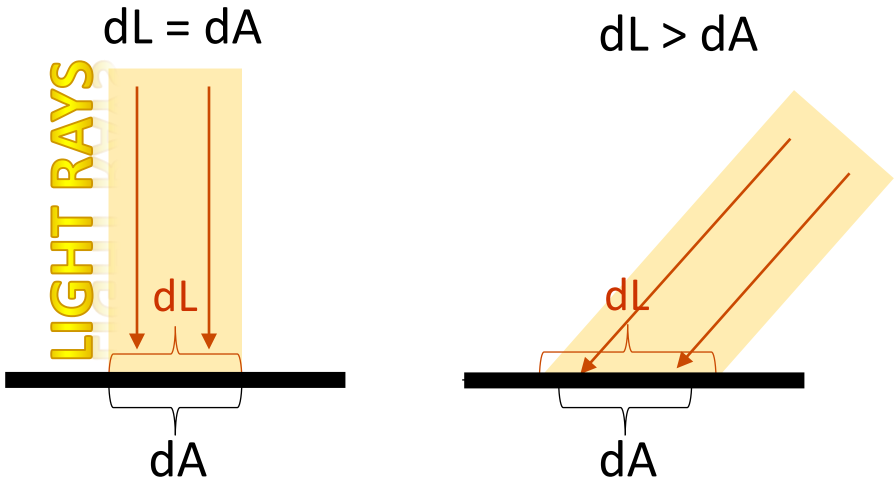
Observation



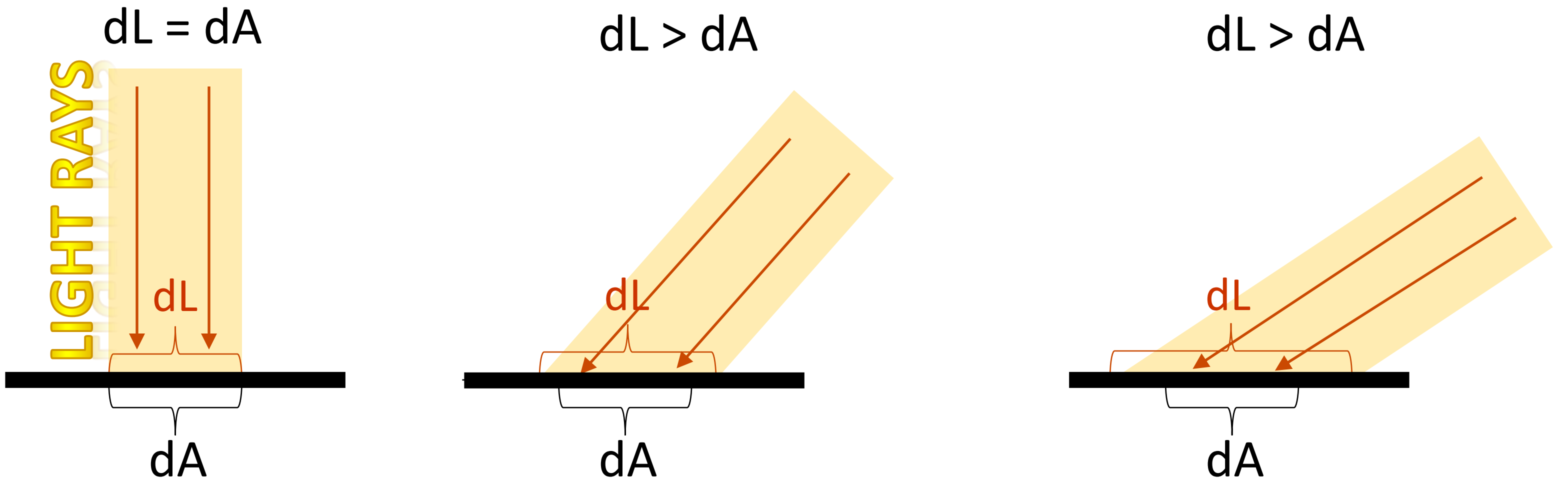
Observation



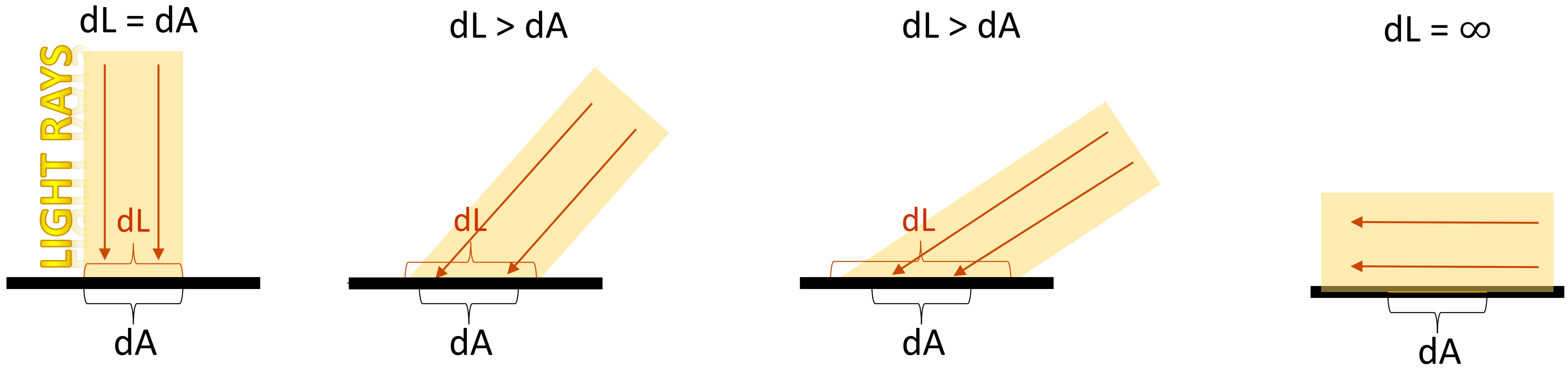
Observation



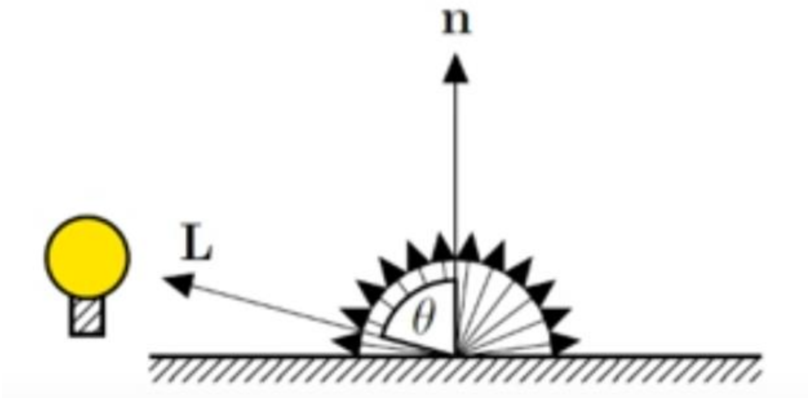
Observation



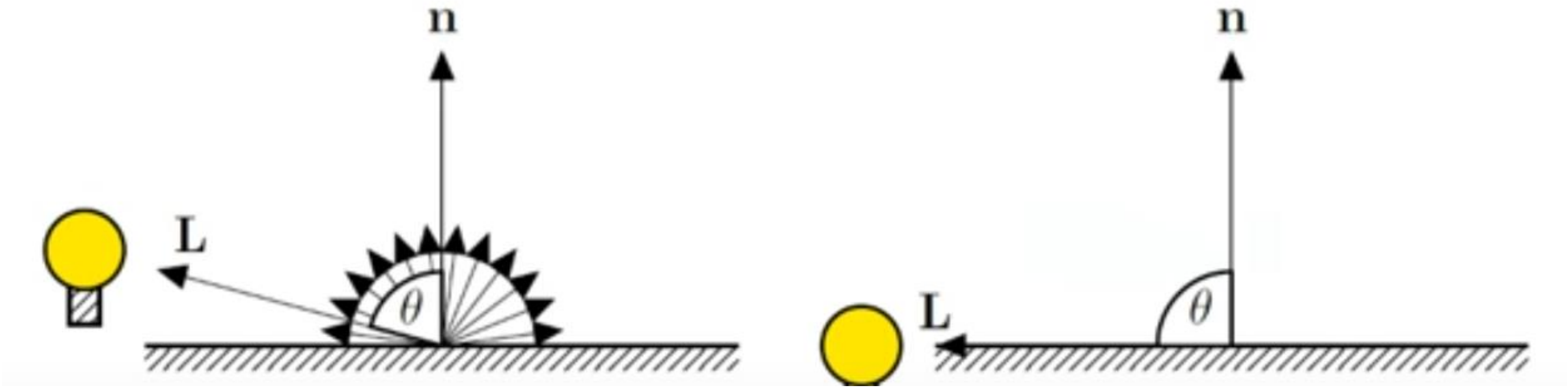
Observation



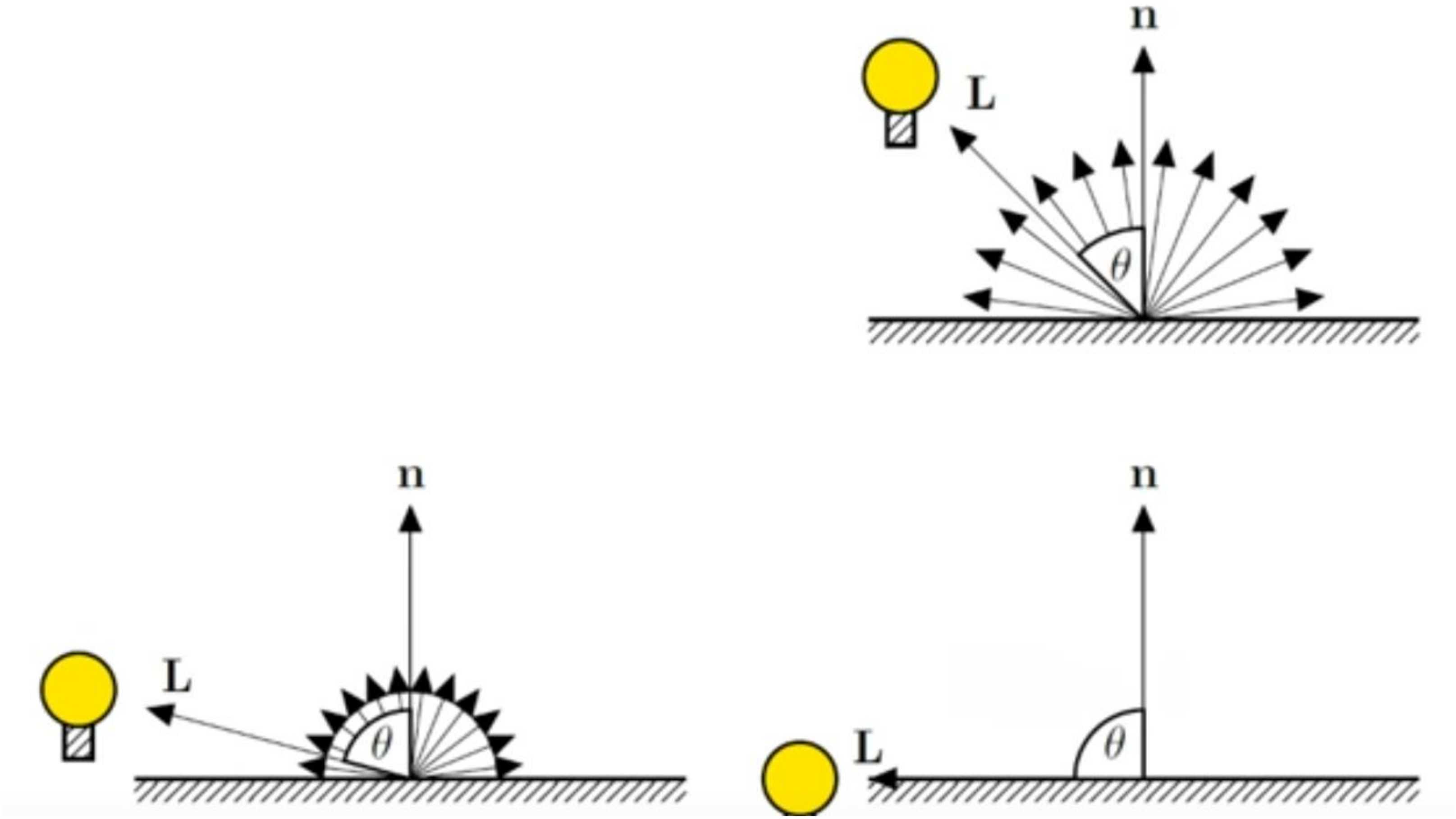
Phong model of diffuse lighting



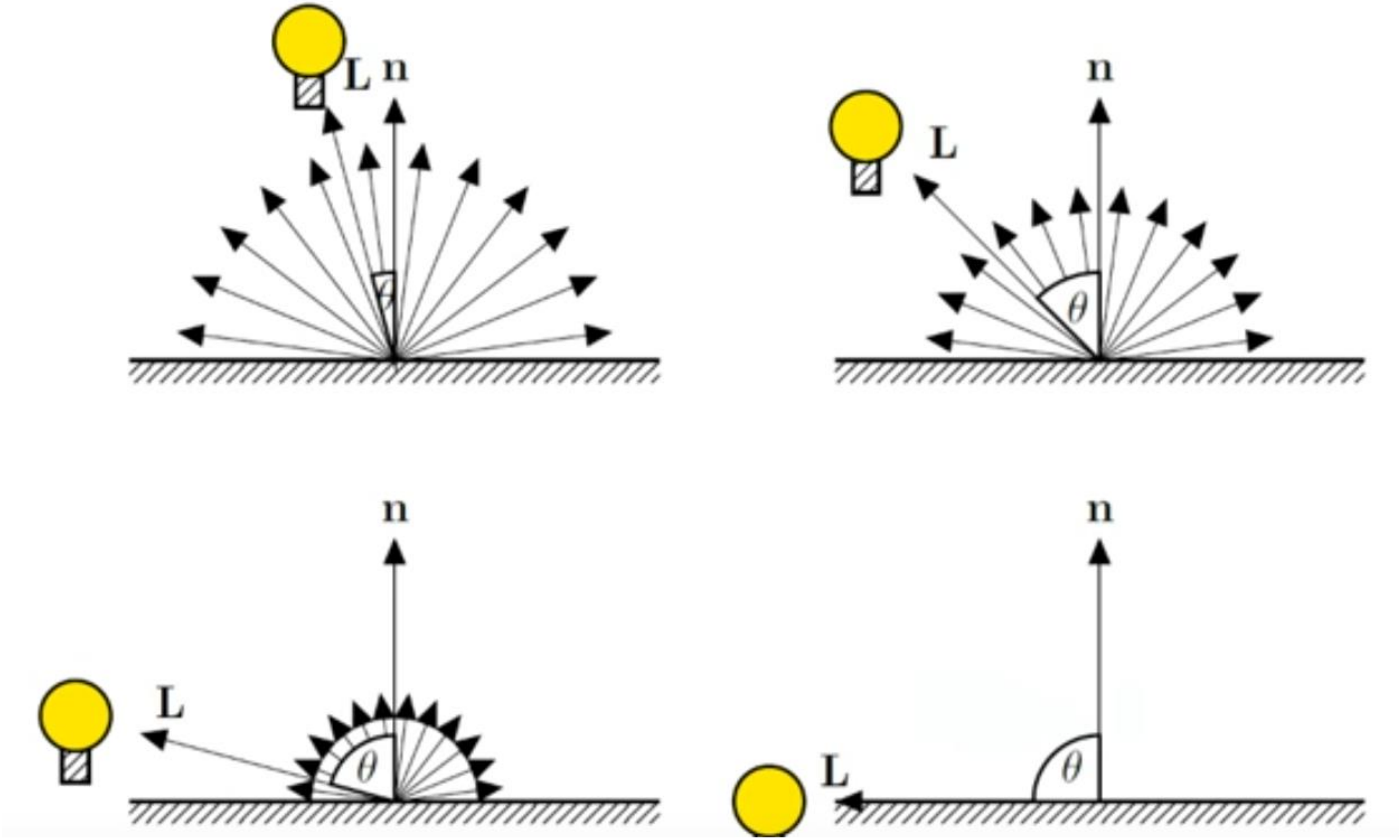
Phong model of diffuse lighting



Phong model of diffuse lighting



Phong model of diffuse lighting



Phong model of diffuse lighting

- Phong model of diffuse lighting depends on the position of the light source relative to the object surface
- $\boldsymbol{D = I_p k_d \max[\cos(\theta), 0]}$

Phong model of diffuse lighting

- Phong model of diffuse lighting depends on the position of the light source relative to the object surface
- $\boldsymbol{D = I_p k_d} \, \mathbf{max}[\mathbf{cos}(\boldsymbol{\theta}), \mathbf{0}]$
- Where
  - I_p is the light intensity of the light source
  - $k_d \in [0, 1]$ is a diffuse lighting coefficient
  - θ is the angle between L and the normal of the object surface n

Phong model of diffuse lighting

- Phong model of diffuse lighting depends on the position of the light source relative to the object surface
- $\boldsymbol{D = I_p k_d} \ \mathbf{max}[\mathbf{cos}(\boldsymbol{\theta}), \mathbf{0}]$
- Where
  - I_p is the light intensity of the light source
  - $k_d \in [0, 1]$ is a diffuse lighting coefficient
  - θ is the angle between L and the normal of the object surface n
- Function $\mathbf{max}[\mathbf{cos}(\boldsymbol{\theta}), \mathbf{0}]$ is used so that light is not reflected when the light source is behind the object surface

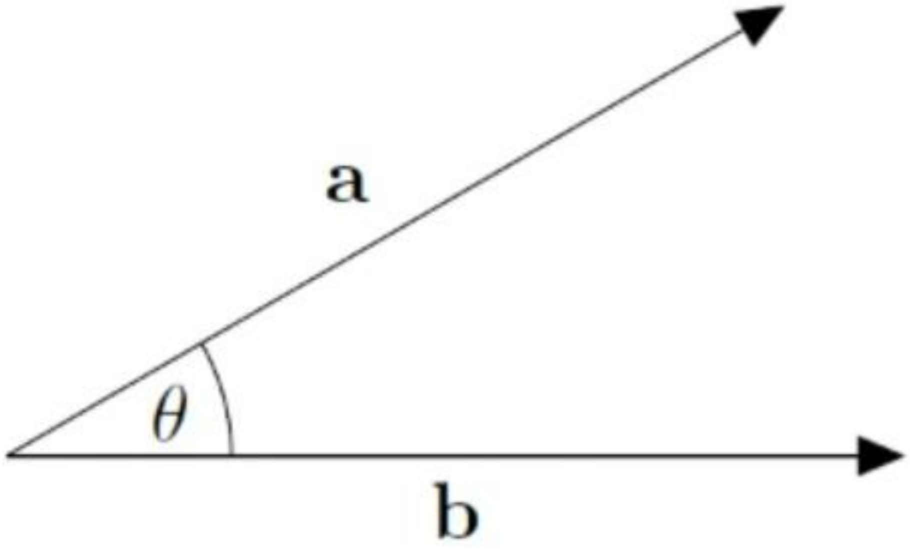
$k_d = 0.25$

$k_d = 0.5$

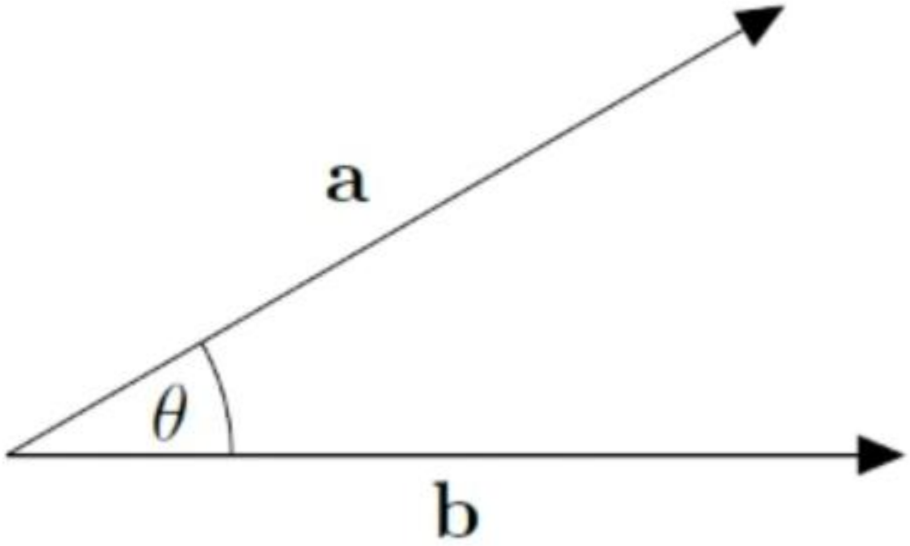
$k_d = 0.75$

$k_d = 1$

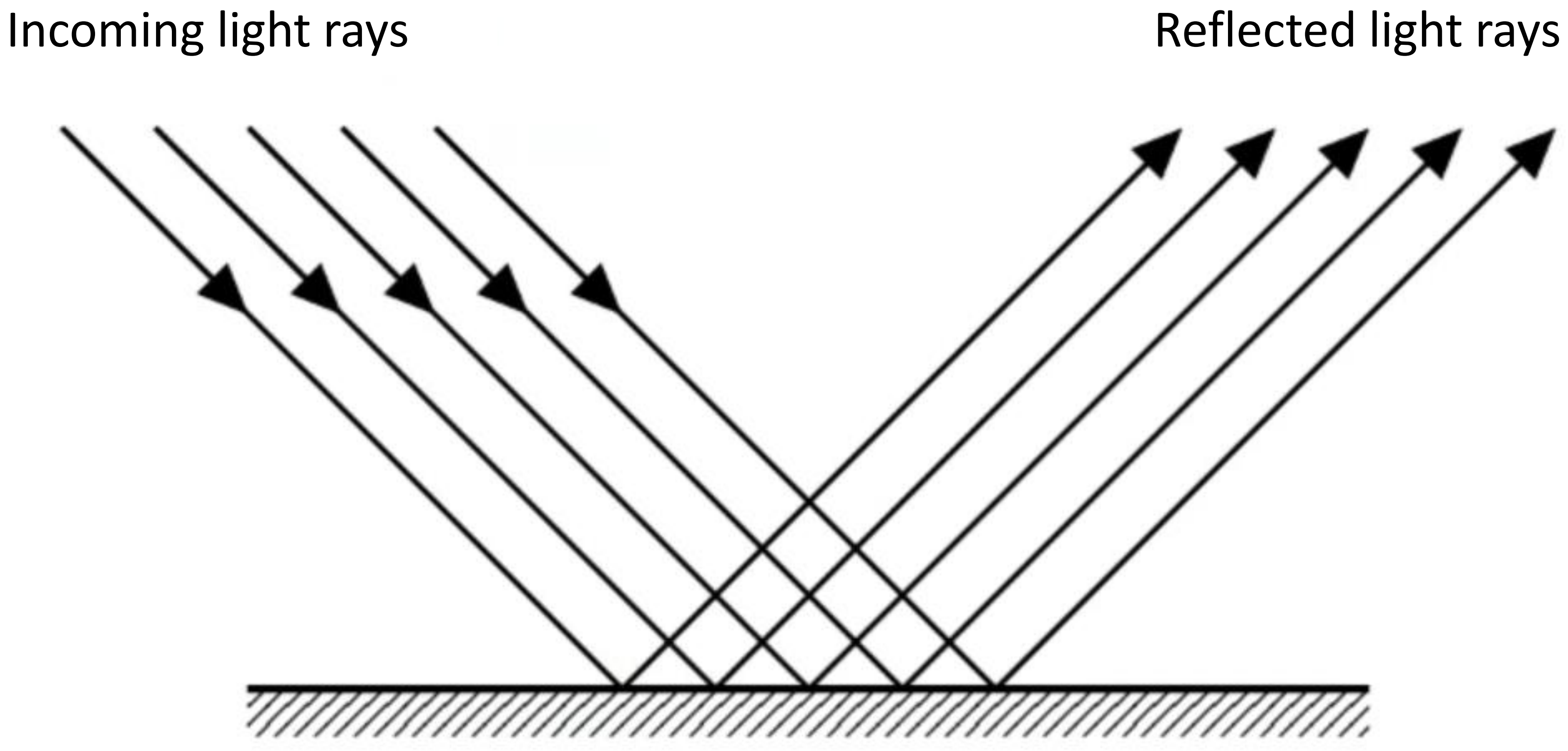
- Dot product between two vectors:
  - $a \cdot b = |a|\,|b| \cos(\theta)$
- However if the vectors are of unit length:
  - $L \cdot n = \cos(\theta)$



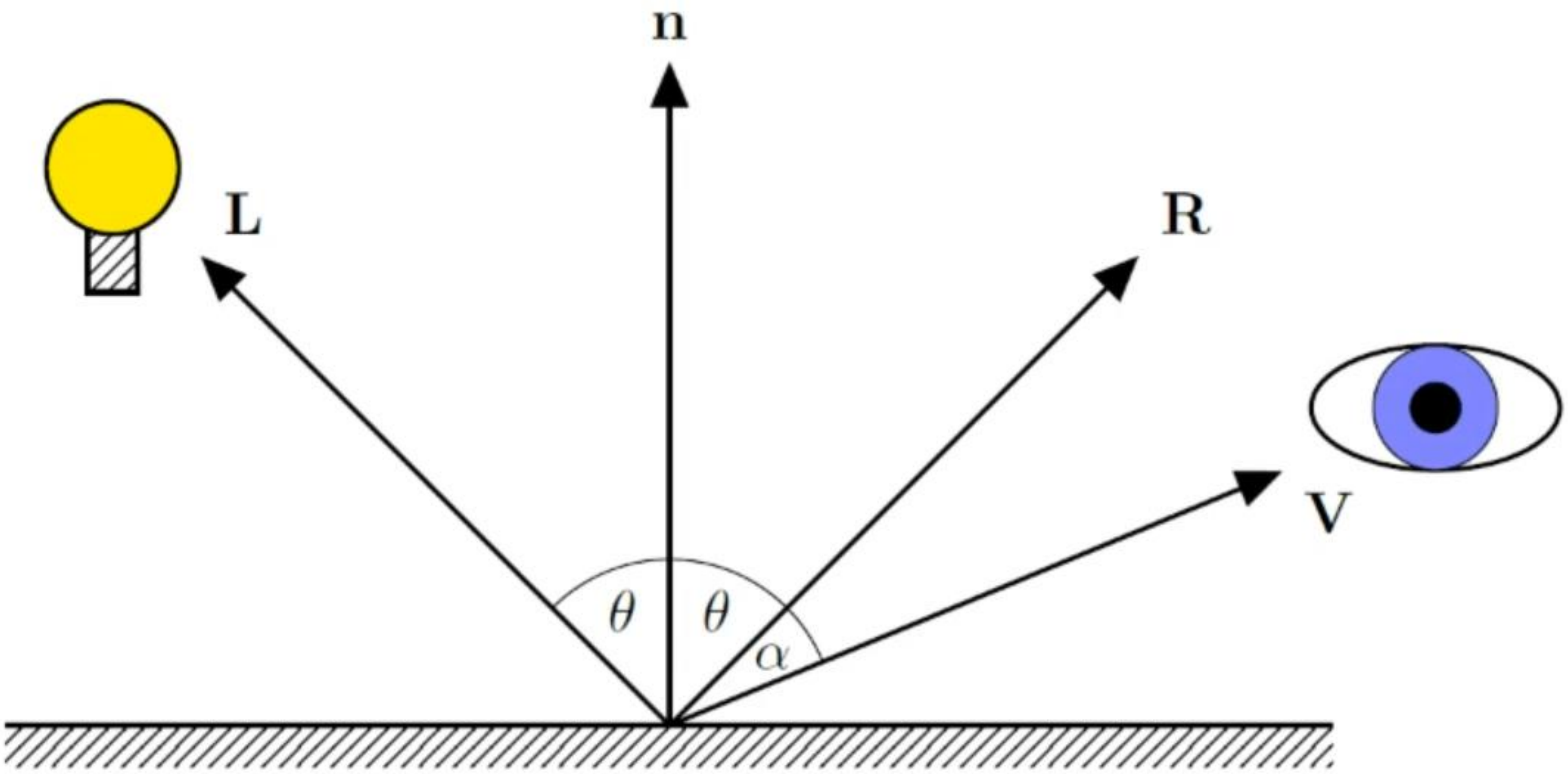
- Dot product between two vectors:
  - $a \cdot b = |a|\,|b|\cos(\theta)$
- However if the vectors are of unit length:
  - $L \cdot n = \cos(\theta)$
- We can substitute the costly cosine calculation with the dot product
  - $\boldsymbol{D = I_p k_d \max[L \cdot n, 0]}$



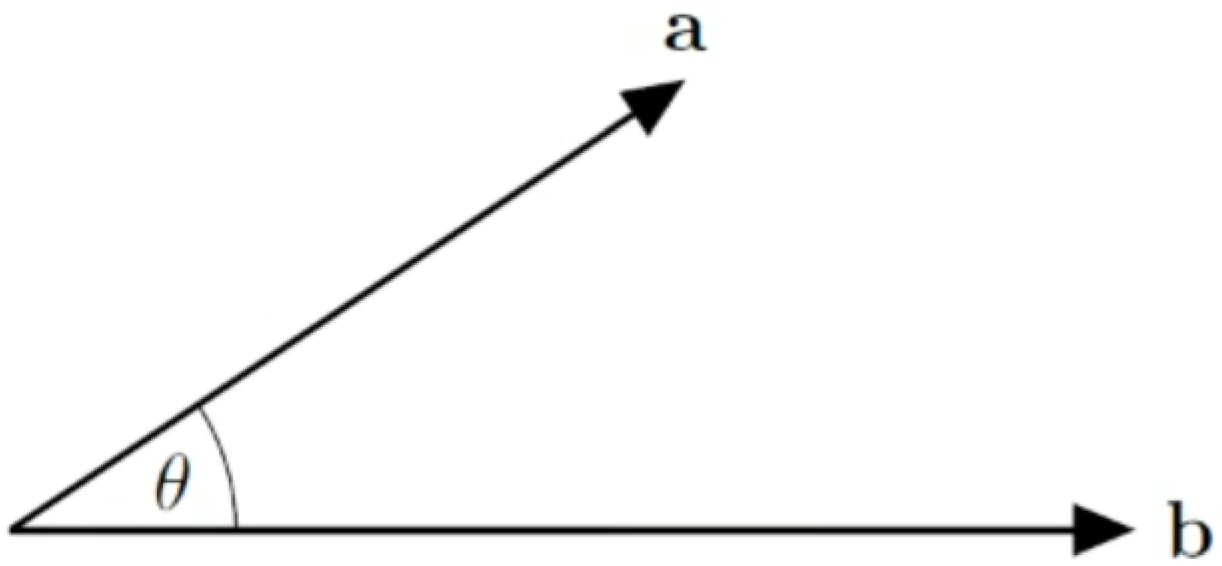
Light reflection



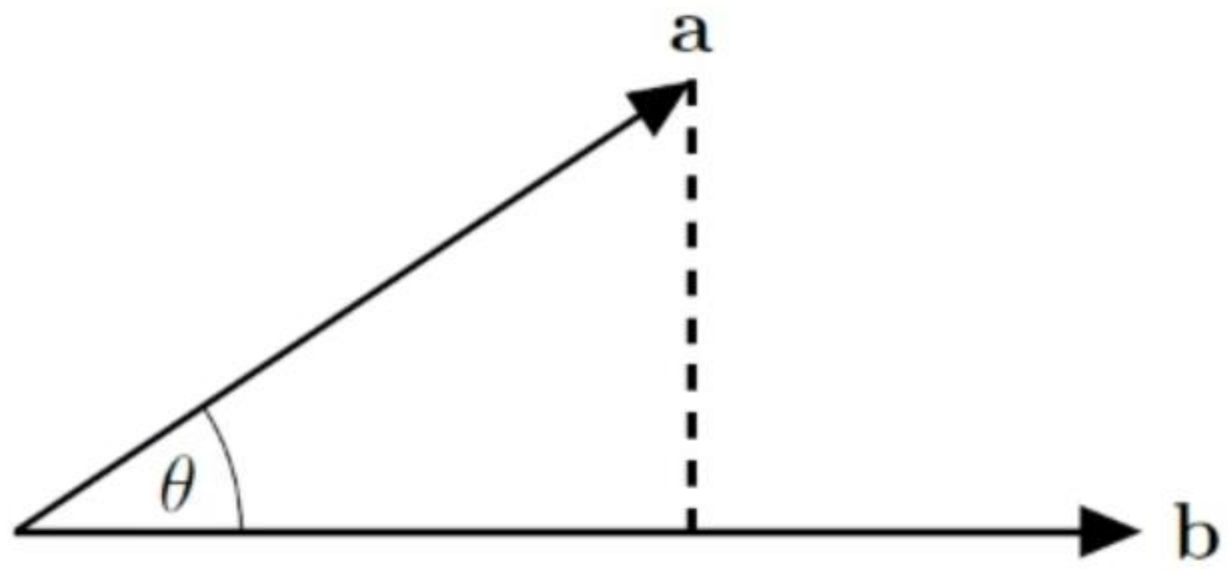
Specular lighting - vectors



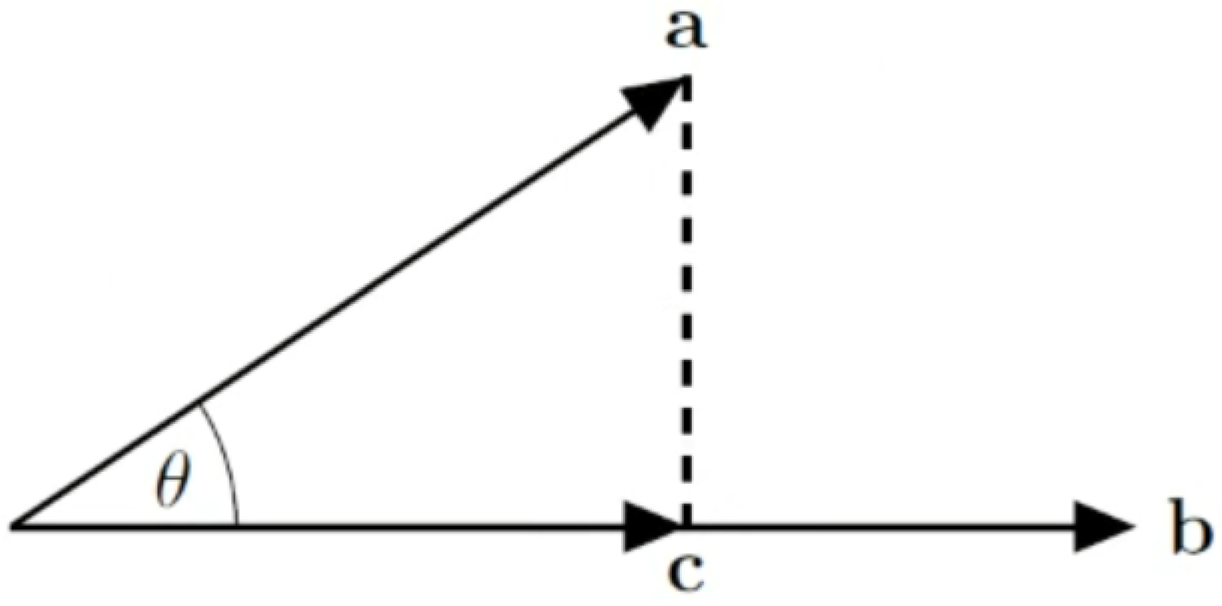
Vector projection

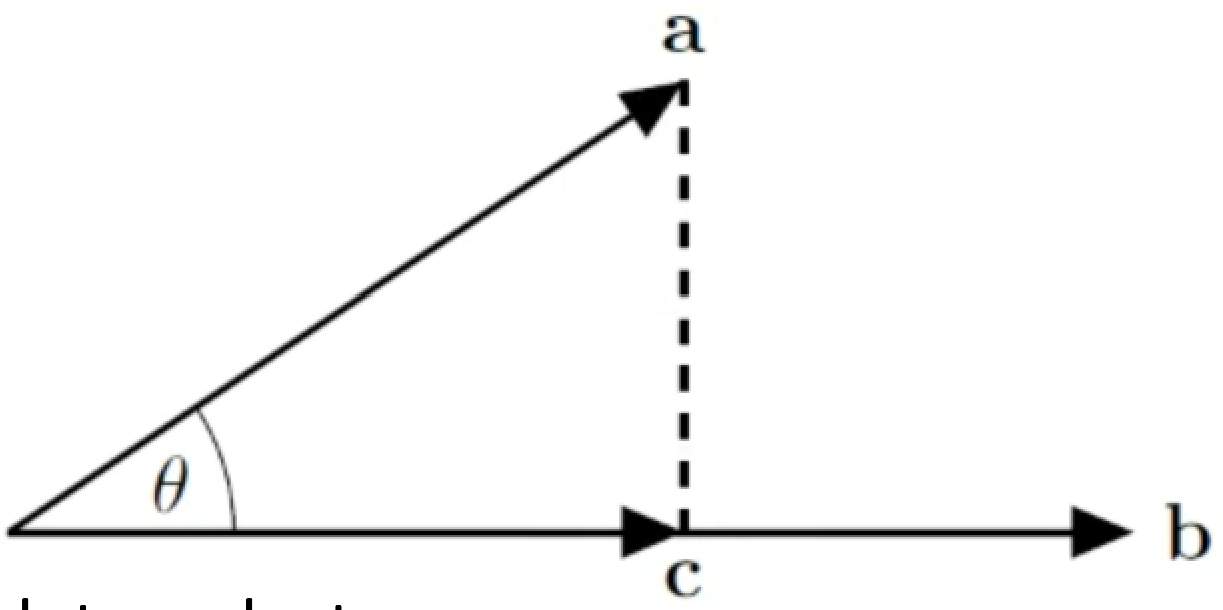


Vector projection

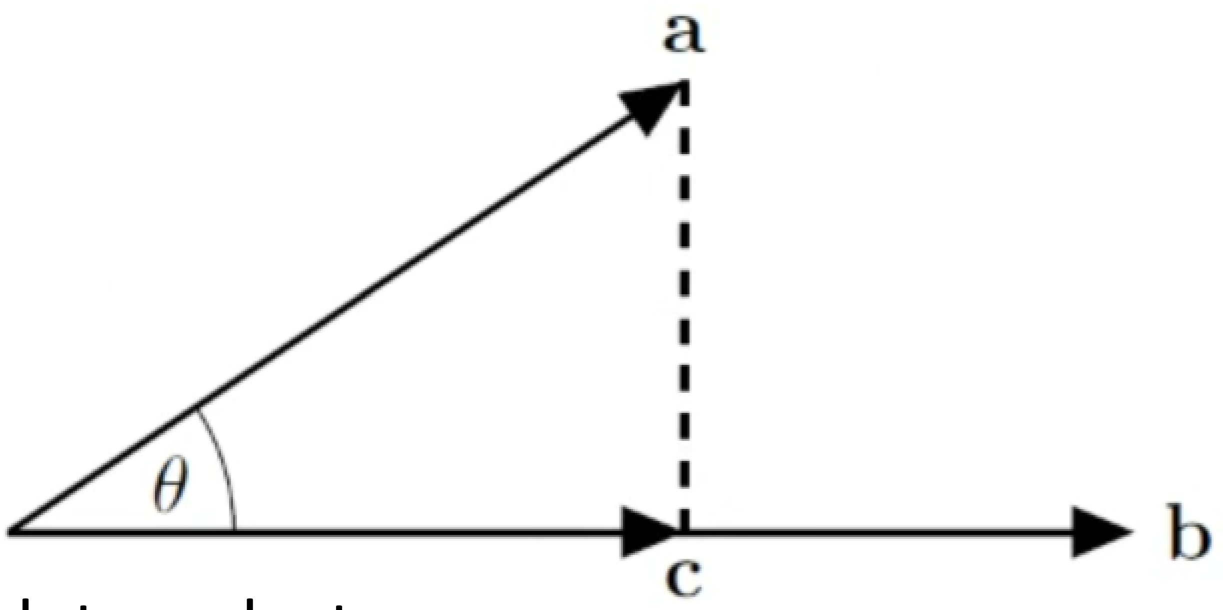


Vector projection

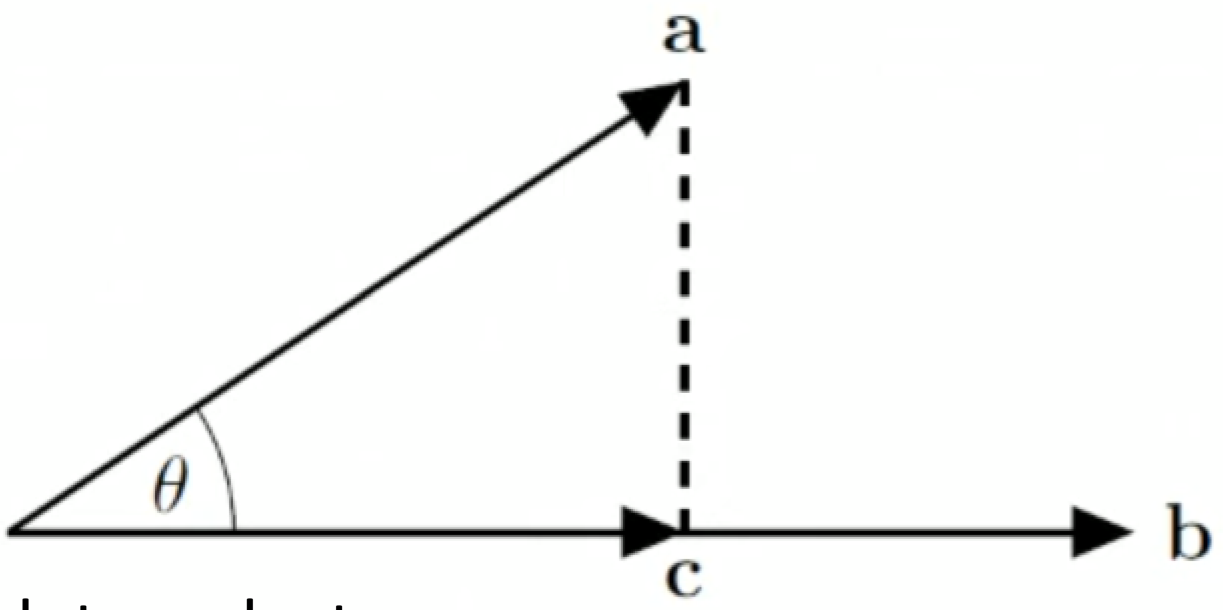




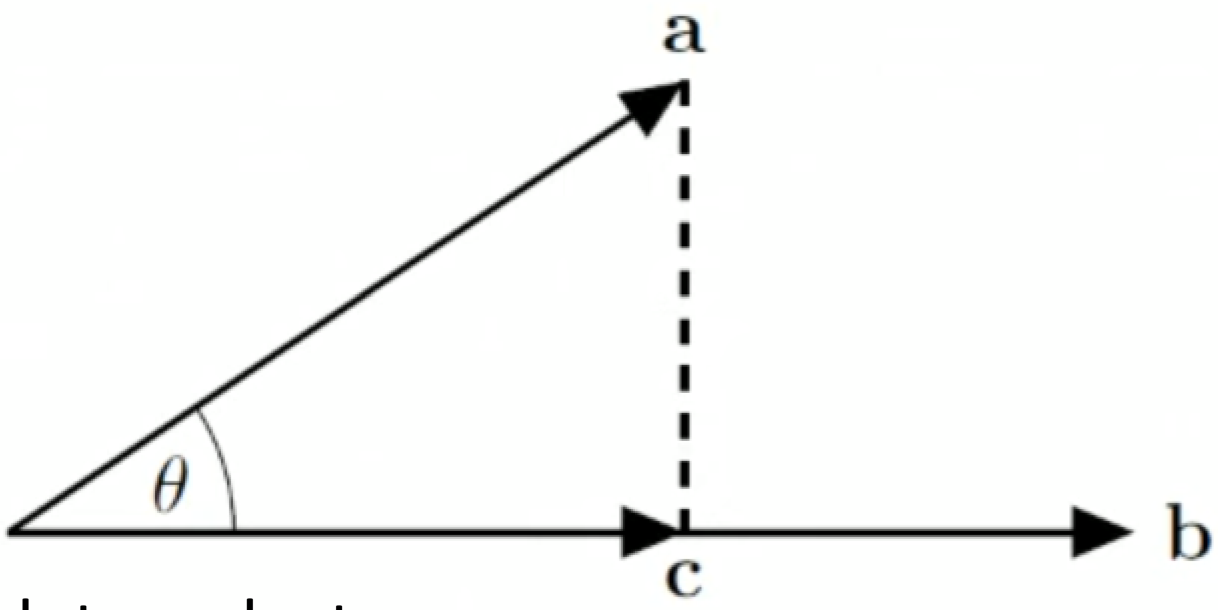
- From the definition of the dot product:
- $a \cdot b = |a||b| \cos(\alpha)$



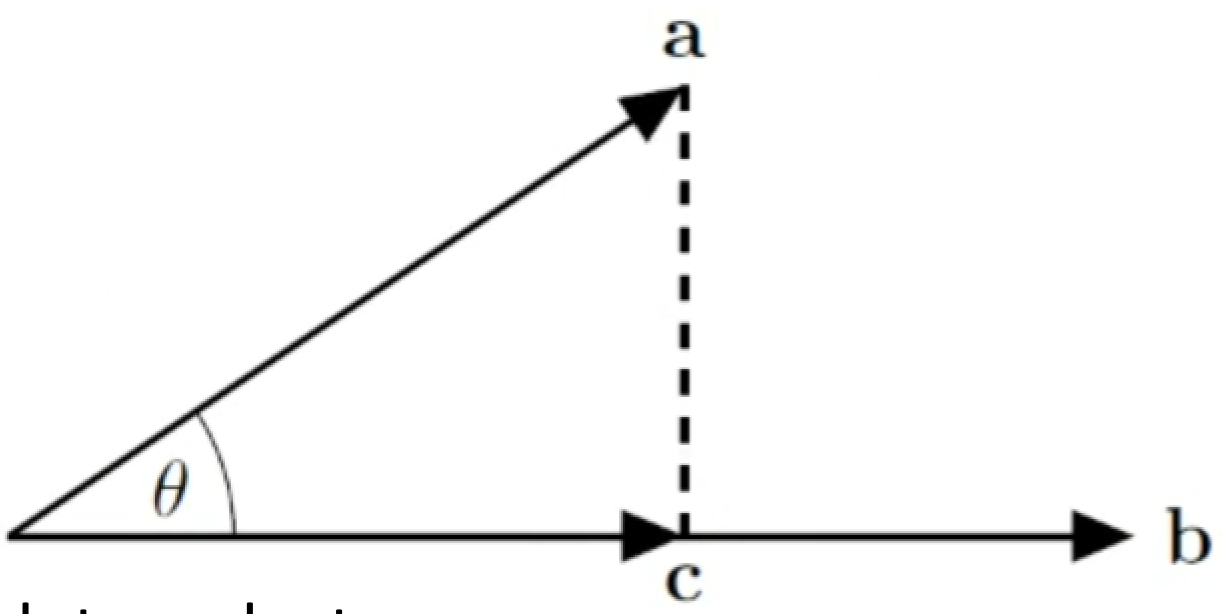
- From the definition of the dot product:
- $a \cdot b = |a||b|\cos(\alpha) = |a||b|\frac{|c|}{|a|}$



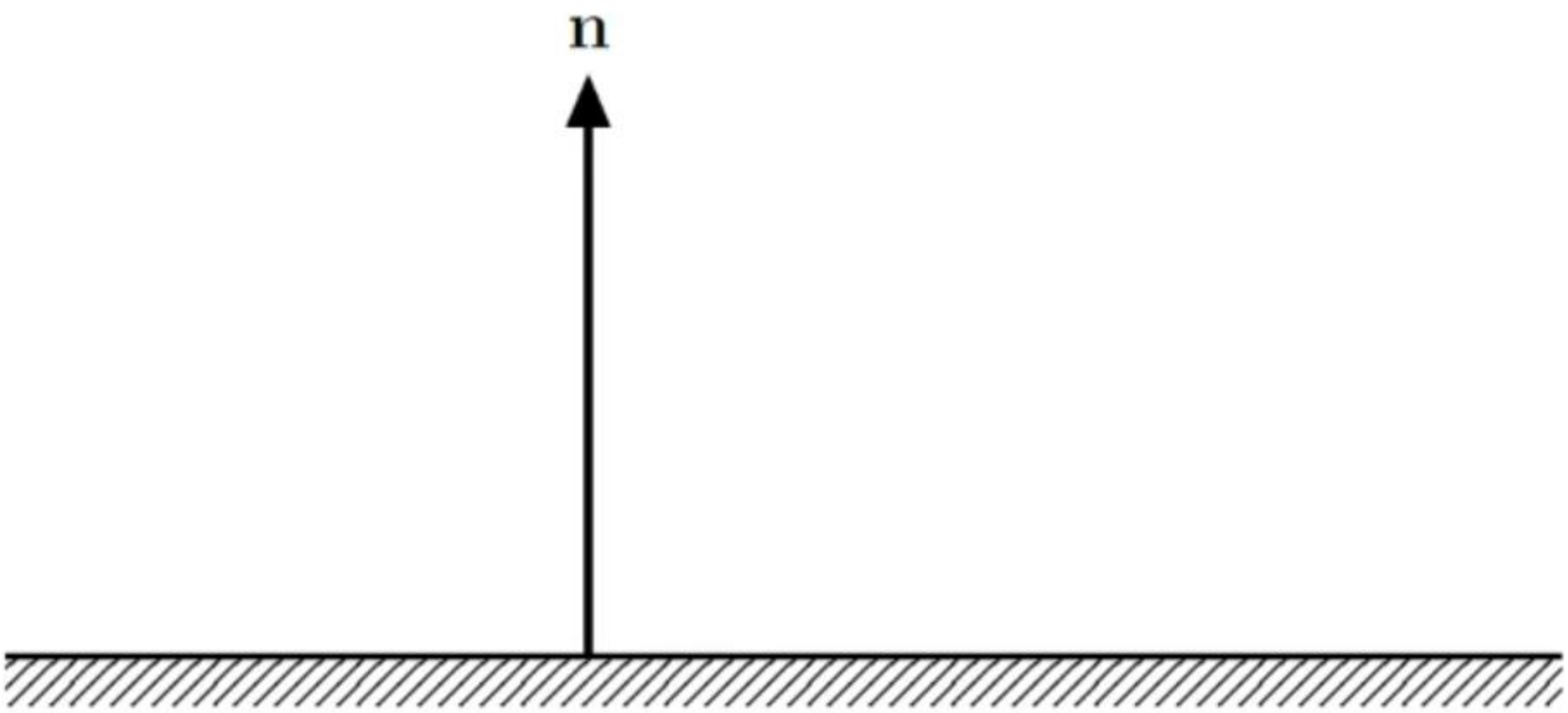
- From the definition of the dot product:
- $a \cdot b = |a||b| \cos(\alpha) = |a||b| \frac{|c|}{|a|} = |b||c|$

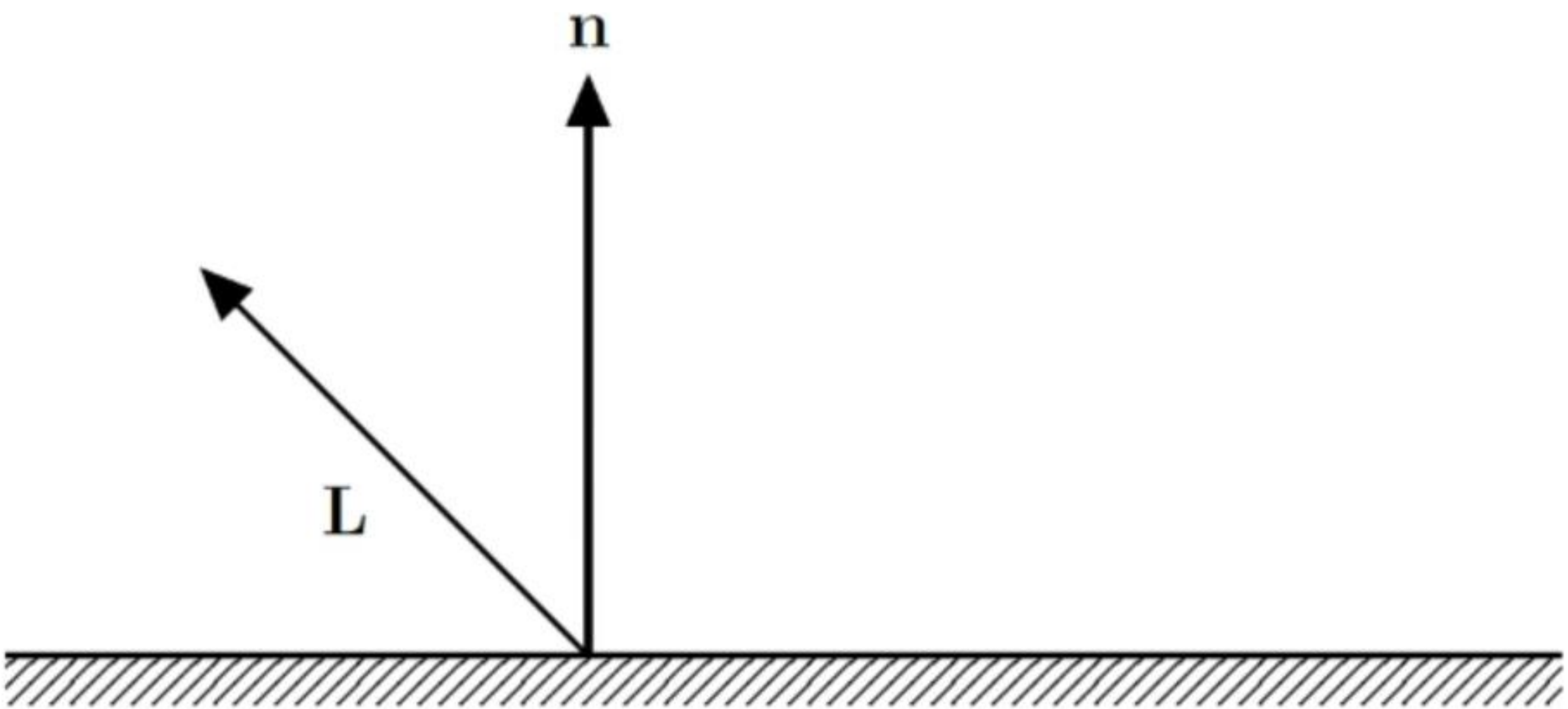


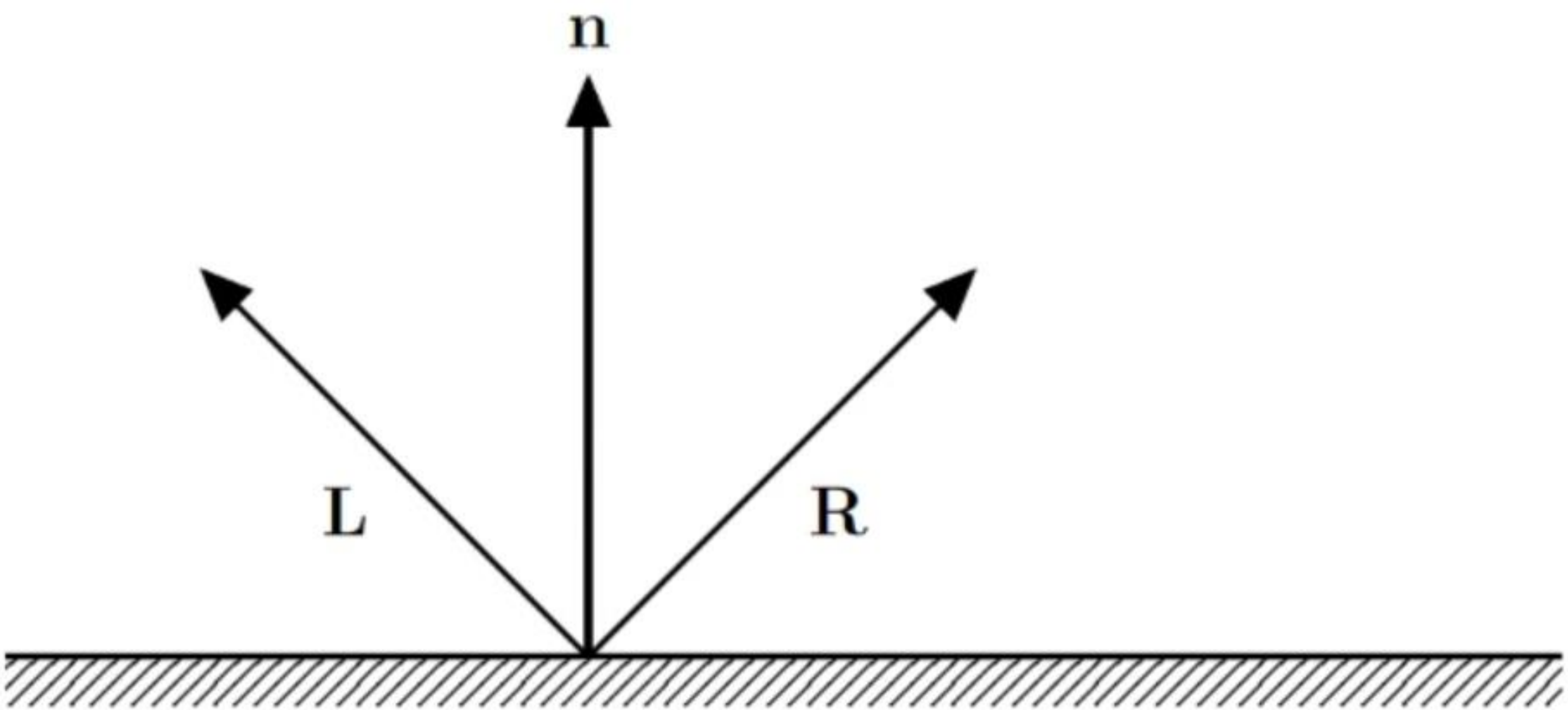
- From the definition of the dot product:
- $a \cdot b = |a||b|\cos(\alpha) = |a||b|\frac{|c|}{|a|} = |b||c|$
- $|c| = \frac{a \cdot b}{|b|}$

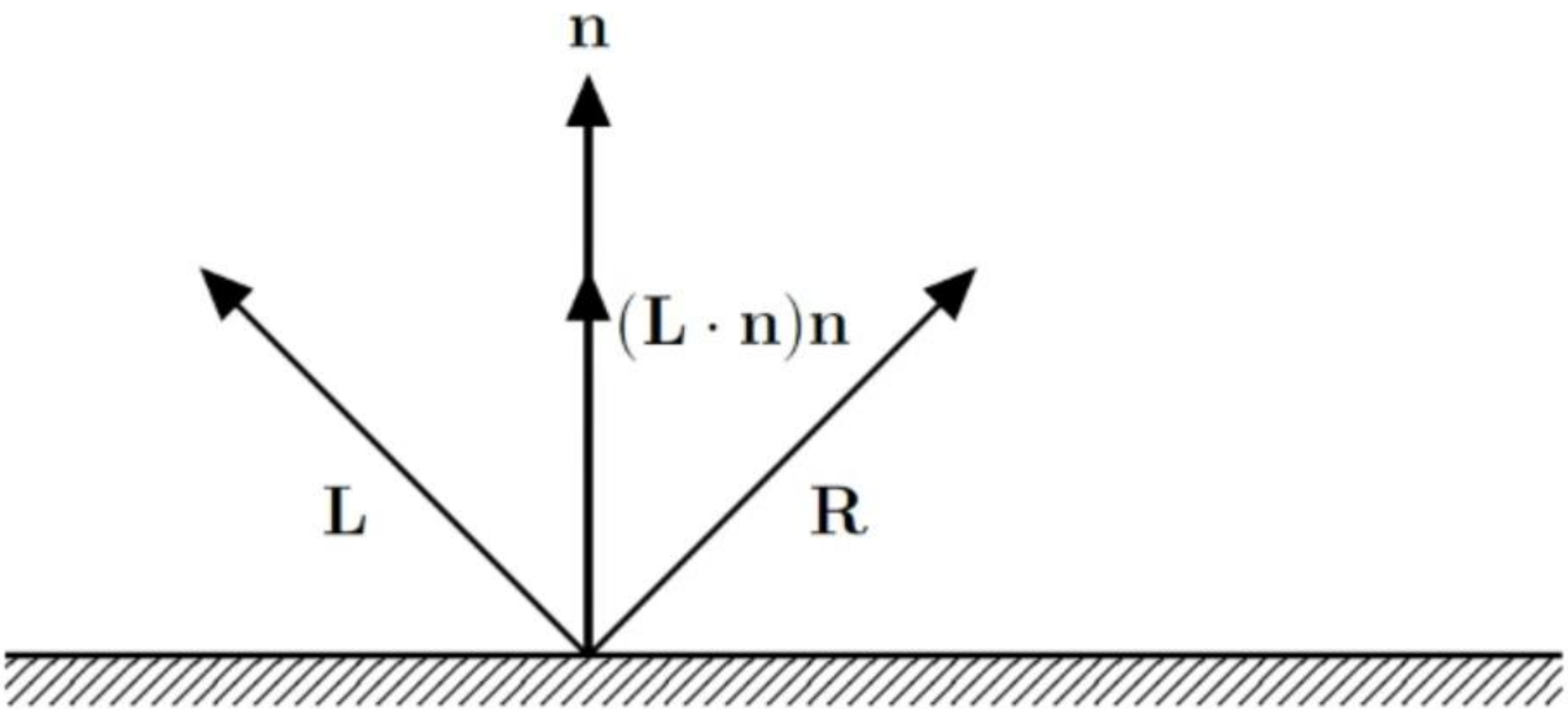


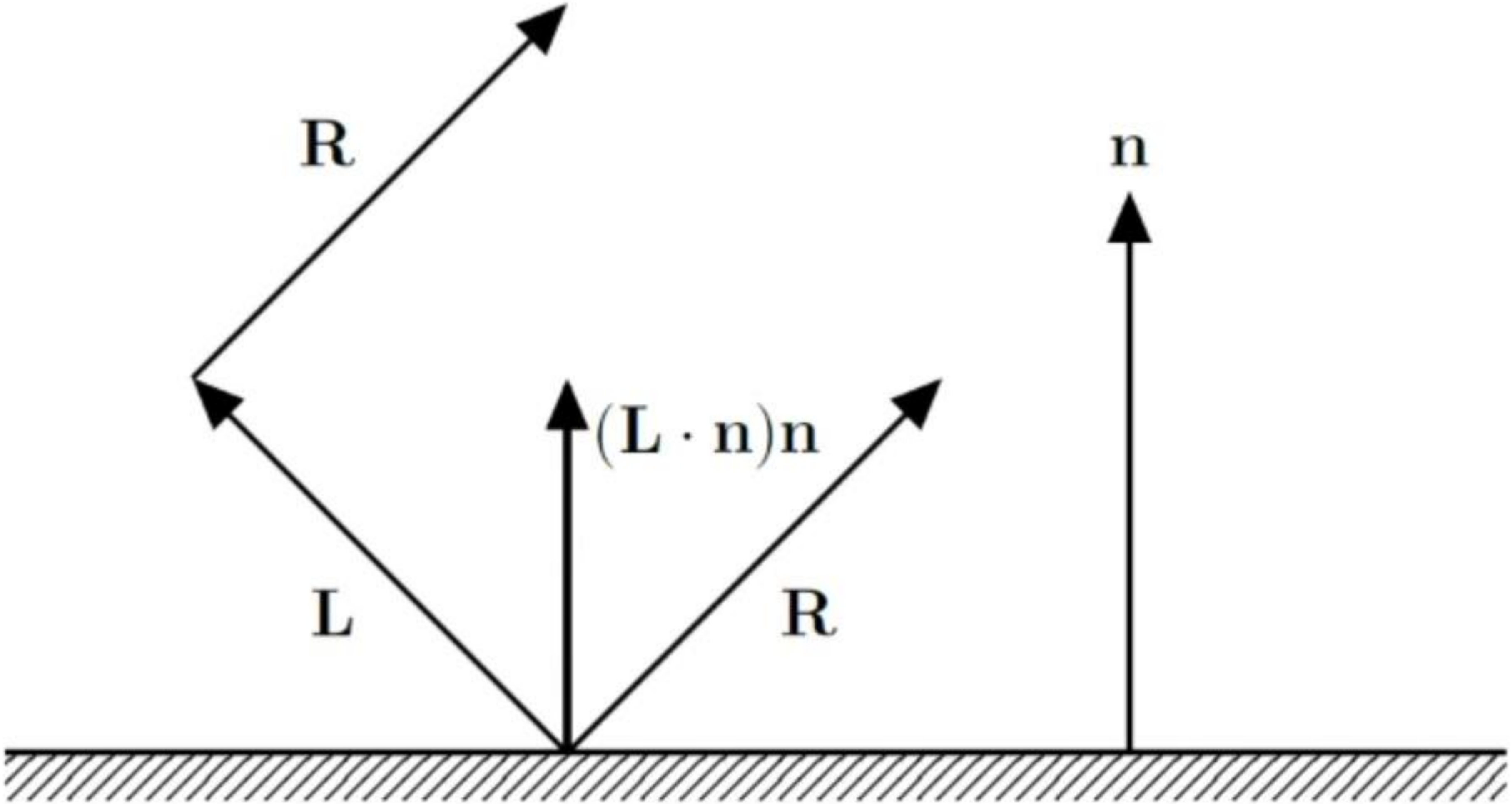
- From the definition of the dot product:
- $a \cdot b = |a||b|\cos(\alpha) = |a||b|\frac{|c|}{|a|} = |b||c|$
- $|c| = \frac{a \cdot b}{|b|}$
- If b is a unit vector $|c| = a \cdot b$ then
- $c = (a \cdot b)b$

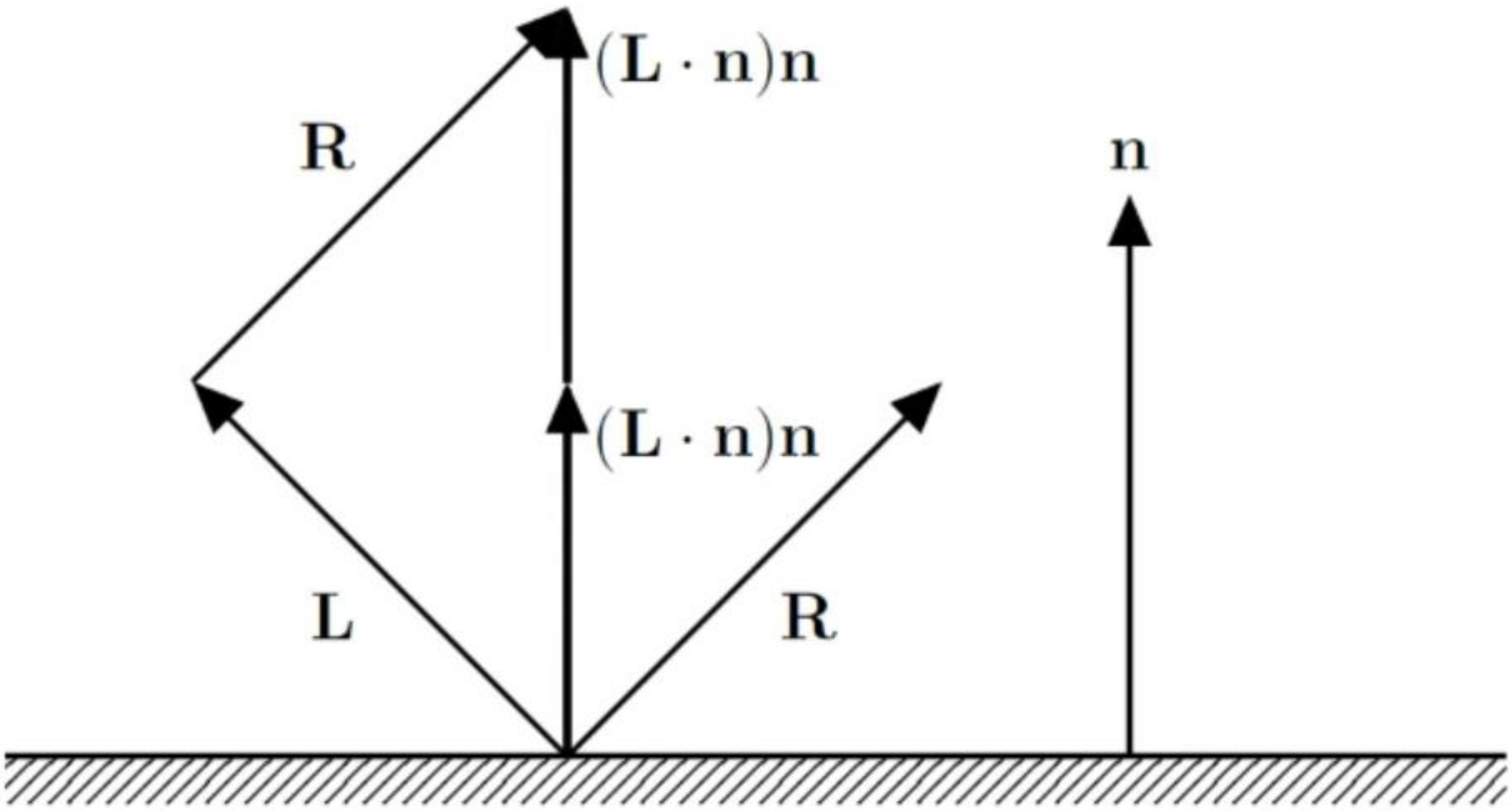

n

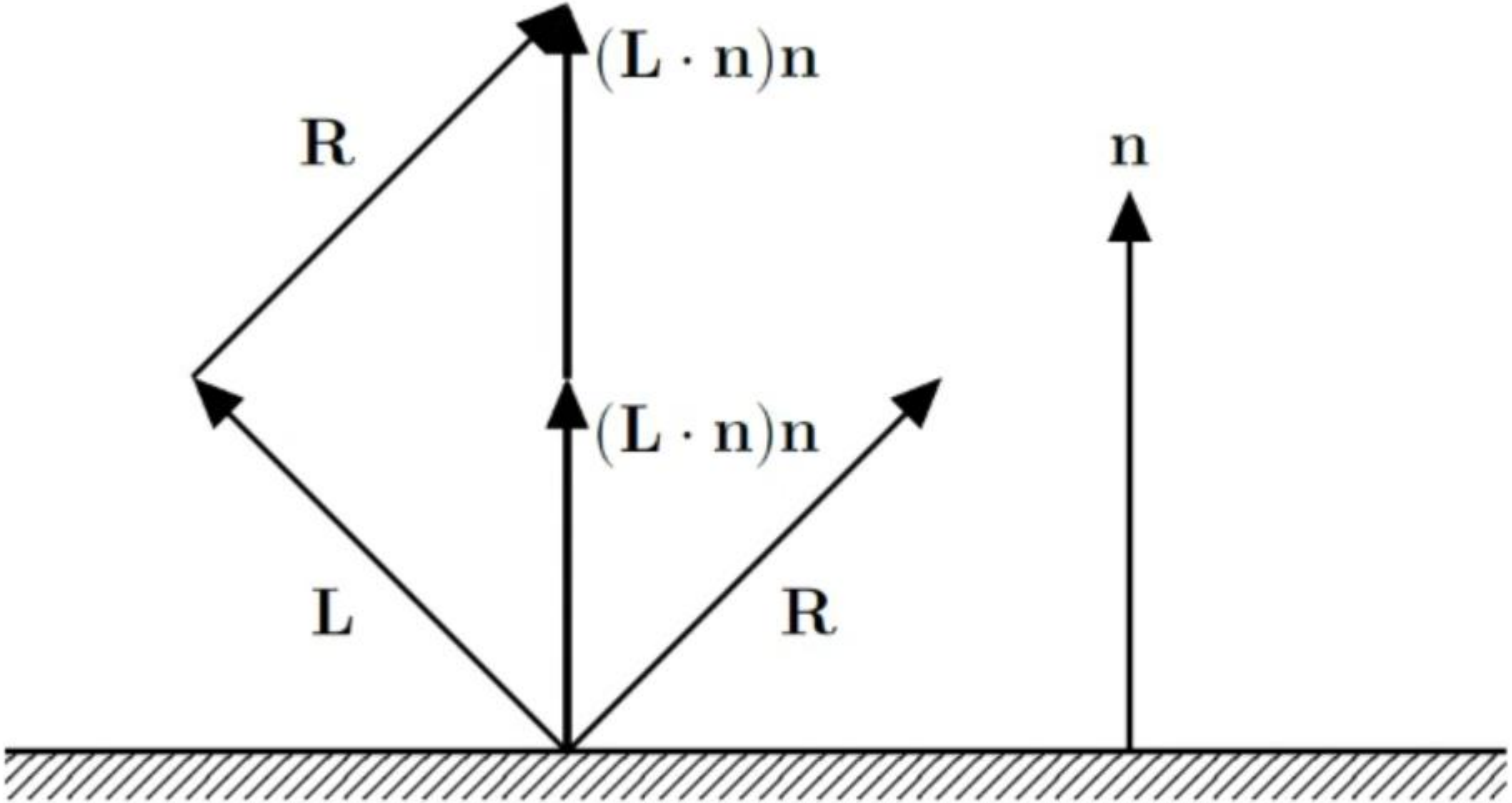

n
L


n
L
R

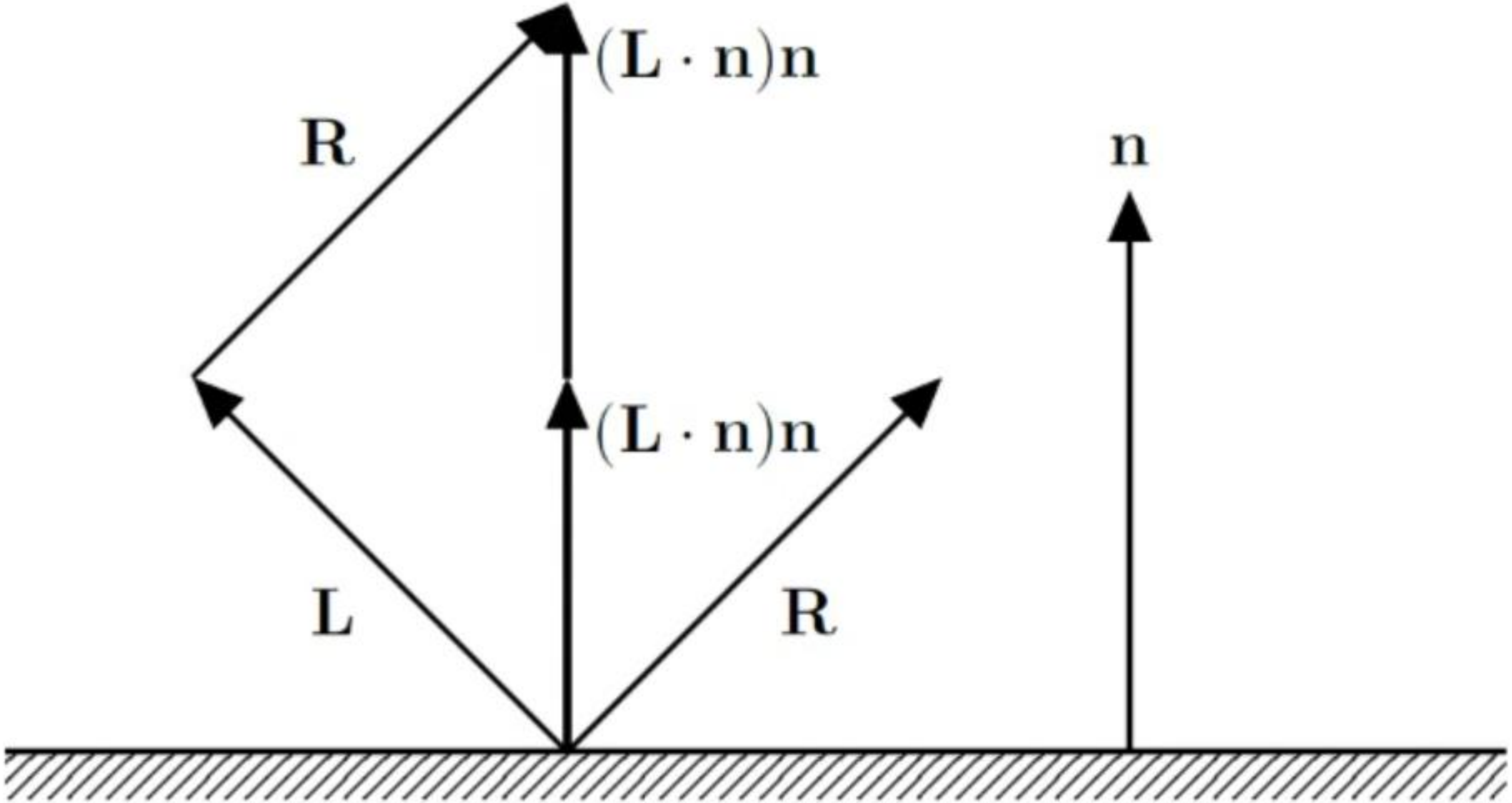

n
(L · n)n
L
R


R
n
$(\mathbf{L} \cdot \mathbf{n})\mathbf{n}$
L
R


(L · n)n
R
n
(L · n)n
L
R



$$\mathbf{L} + \mathbf{R} = 2(\mathbf{L} \cdot \mathbf{n})\mathbf{n}$$



$$\mathbf{L} + \mathbf{R} = 2(\mathbf{L} \cdot \mathbf{n})\mathbf{n}$$

$$\therefore \mathbf{R} = 2(\mathbf{L} \cdot \mathbf{n})\mathbf{n} - \mathbf{L}$$

GLSL

r = reflect(I, n)

Name

reflect — calculate the reflection direction for an incident vector

Declaration

```
genType reflect( genType I,
                 genType N);

genDType reflect( genDType I,
                  genDType N);
```

Parameters

I

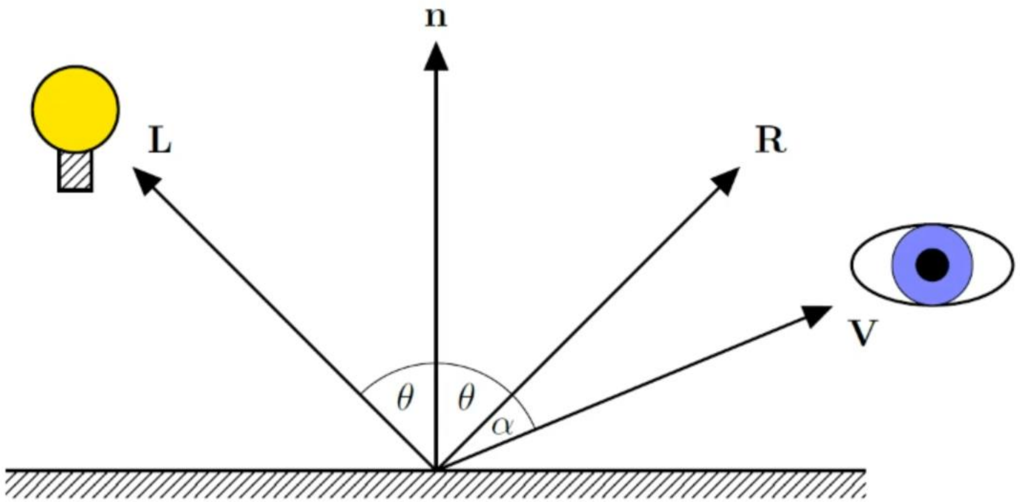
Specifies the incident vector.

N

Specifies the normal vector.

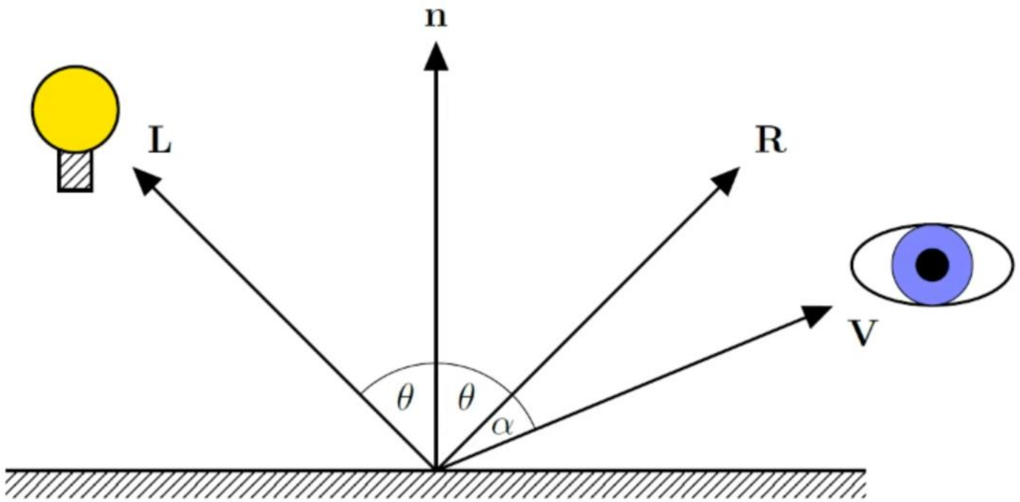
Specular lighting - vectors

- The Phong model of specular lighting is:
- $S = I_p k_s cos^n(\alpha) = I_p k_s (V \cdot R)^n$



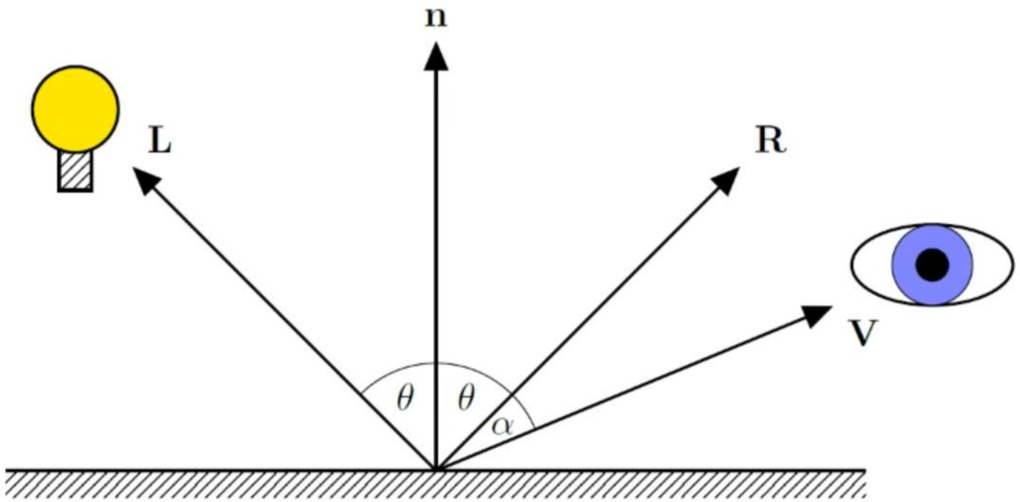
Specular lighting - vectors

- The Phong model of specular lighting is:
- $\boldsymbol{S = I_p k_s cos^n(\alpha) = I_p k_s (V \cdot R)^n}$
- Where
  - $k_s \in [0, 1]$ is the specular lighting coefficient
  - n is the specular lighting exponent
  - α is the angle between R and V

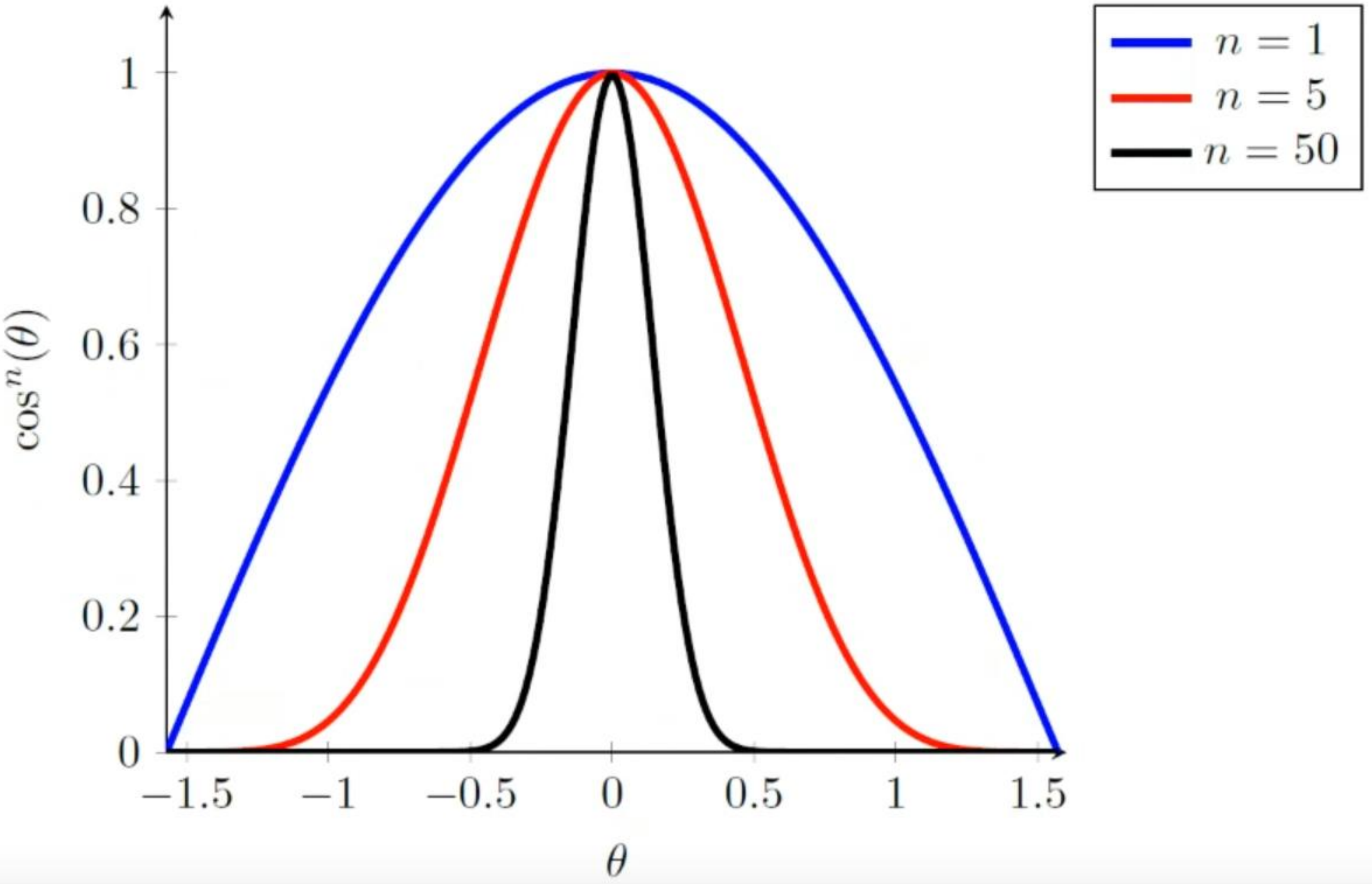


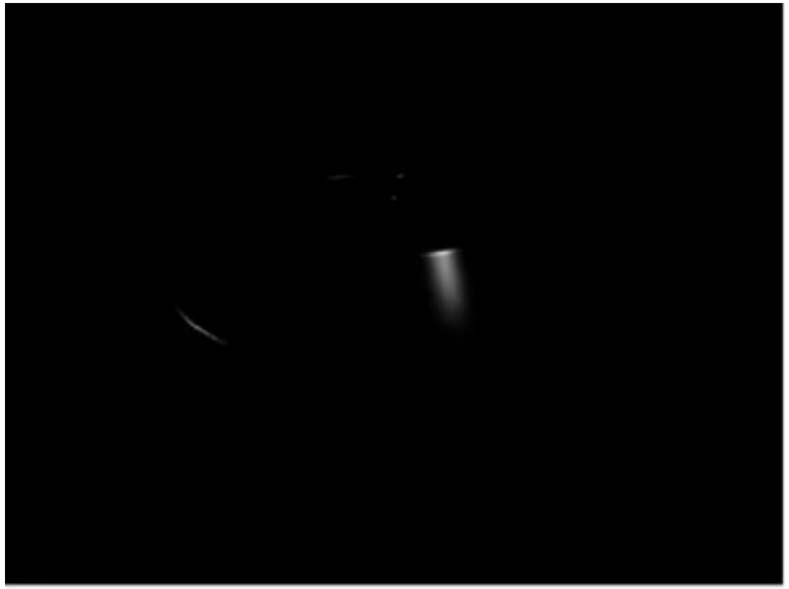
Specular lighting - vectors

- The Phong model of specular lighting is:
- $\boldsymbol{S} = \boldsymbol{I_p k_s cos^n(\alpha)} = \boldsymbol{I_p k_s (V \cdot R)^n}$
- Where
  - $k_s \in [0, 1]$ is the specular lighting coefficient
  - n is the specular lighting exponent
  - α is the angle between R and V
- $cos^n(\alpha)$ determines how much light is reflected by a given material



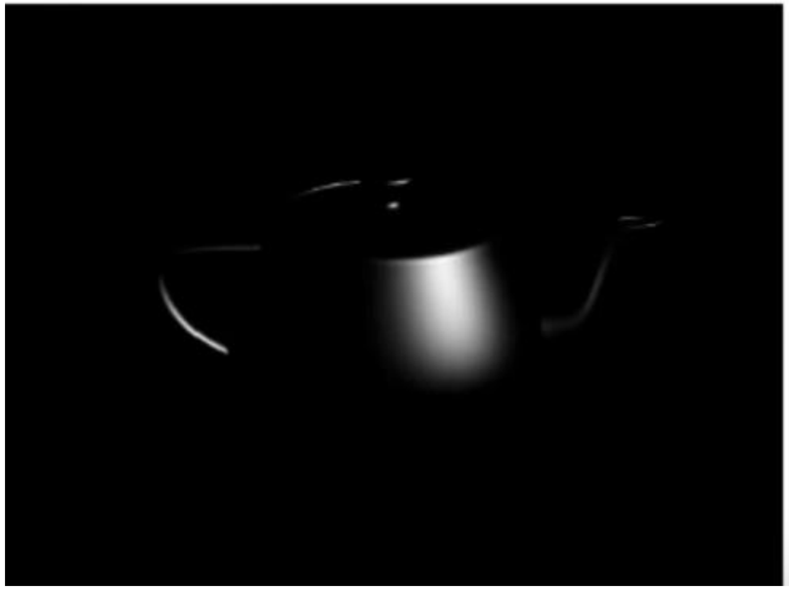
Specular lighting coefficient





n = 50

n = 20

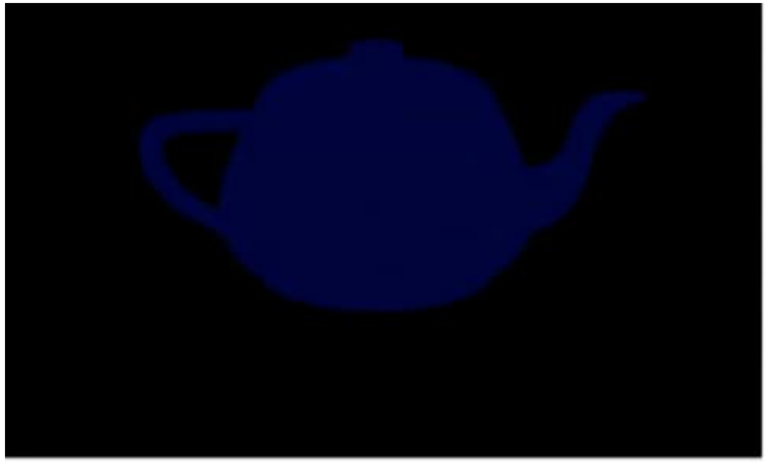


n = 5

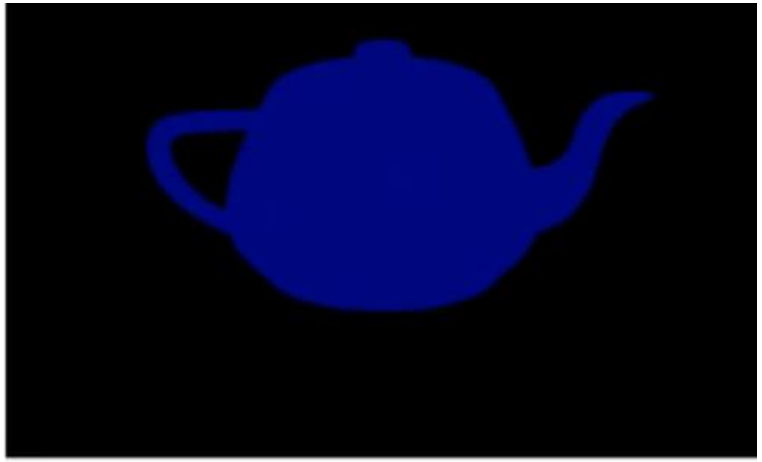
n = 1

Ambient lighting

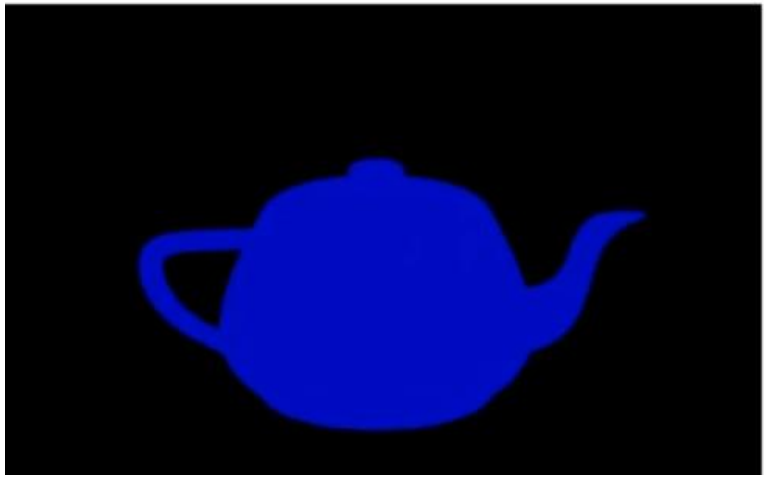
- Ambient lighting, is the reflected light not coming directly from the light source
- $\boldsymbol{A = \ I_a k_a}$
- Where I_a denotes light intensity $k_a \in [0, 1]$ is the coefficient of the environment
- k_a allows to specify a minimum amount of light available for all objects in a scene, n.p. $k_a \to 1$ for bright $k_a \to 0$ for dark scenes



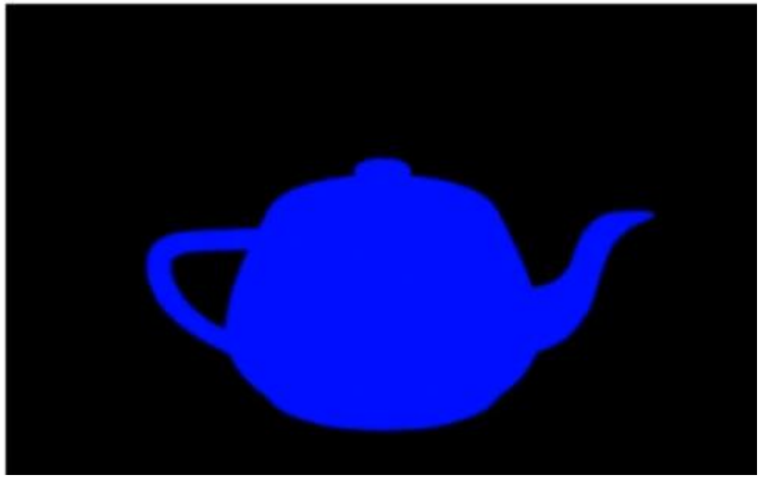
$k_a = 0.25$



$k_a = 0.5$



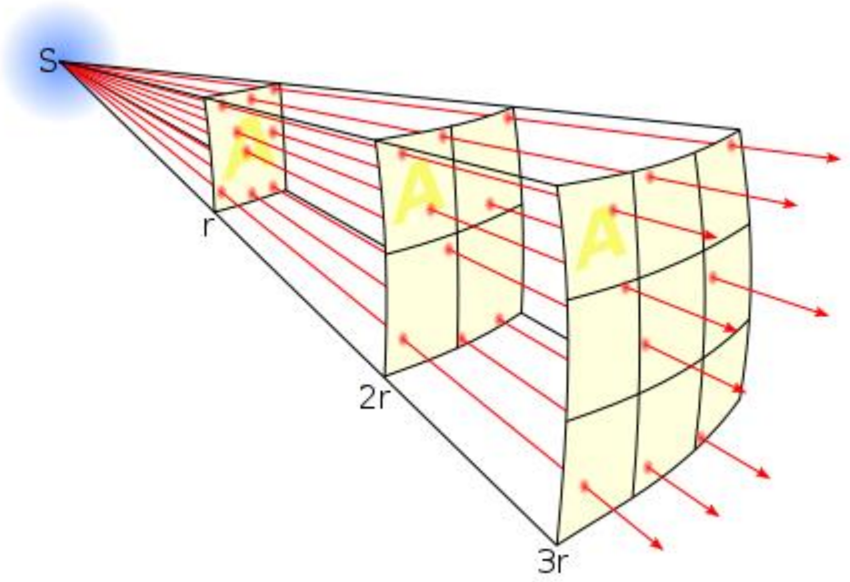
$k_a = 0.75$



$k_a = 1.0$

Attenuation

- Attenuation describes the diminishing of light energy in space
- In the Phong model the coefficient f_{att} models this phenomenon
- Physically-based models propose the following relation
- $f_{att} \sim \frac{1}{d^2}$



Attenuation

- Because the Phong model is a direct lighting model not accounting for reflected light rays usually too much light would be removed using the physically-based formulation. Instead, the following relation is used:
- $f_{att} = 1 - (\frac{d}{r})^2$
- Where r is the radius of the sphere surrounding the light source that represents the furthest possible light

Attenuation

- Adding all the individual parts of the Phong model gives the full Phong light reflectance model
- $I = I_a k_a + \boldsymbol{f_{att}} I_p k_d \max[L \cdot n, 0] + \boldsymbol{f_{att}} I_p k_s (V \cdot R)^n$

Attenuation

- Adding all the individual parts of the Phong model gives the full Phong light reflectance model
- $I = I_a k_a + \boldsymbol{f_{att}} I_p k_d \max[L \cdot n, 0] + \boldsymbol{f_{att}} I_p k_s (V \cdot R)^n$

- For scenes with more than one light source, the individual intensities are simply added together
- $I = I_a k_a + \sum_{i=1}^{m} \boldsymbol{f_{att}} I_{p,i} [k_d \max(L_i \cdot n, 0) + k_s (V \cdot R_i)^n]$

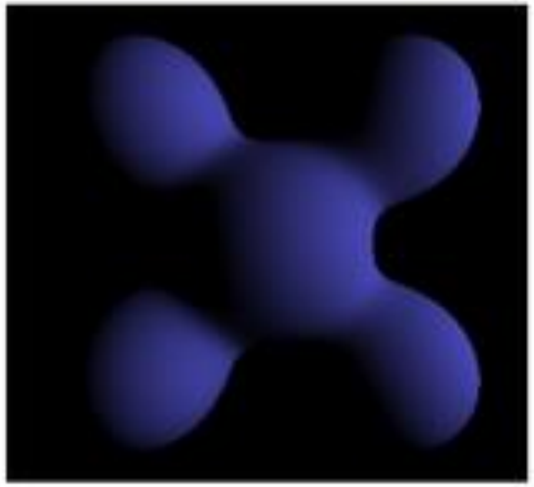
Light intensity and RGB colors

- We can simply scale the **color vectors** in the fragment shader: $v_{new} = v \cdot I$

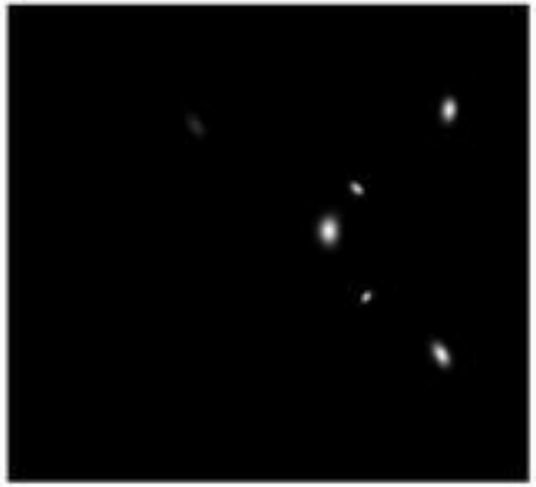
- For example: the red color $v =$(1, 0, 0) with 50% intensity becomes $v_{new} =$ (0.5, 0, 0)

Result

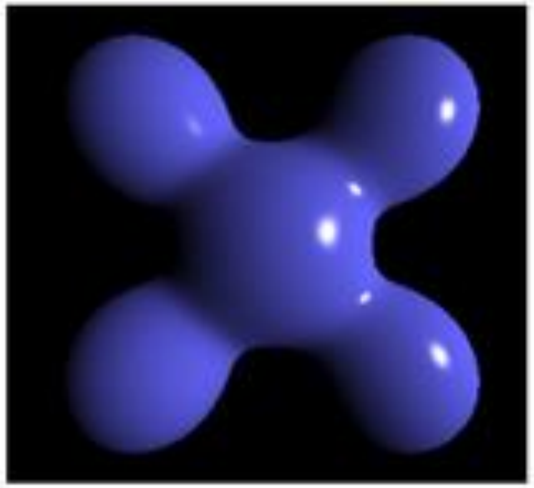
Ambient light



Diffuse light



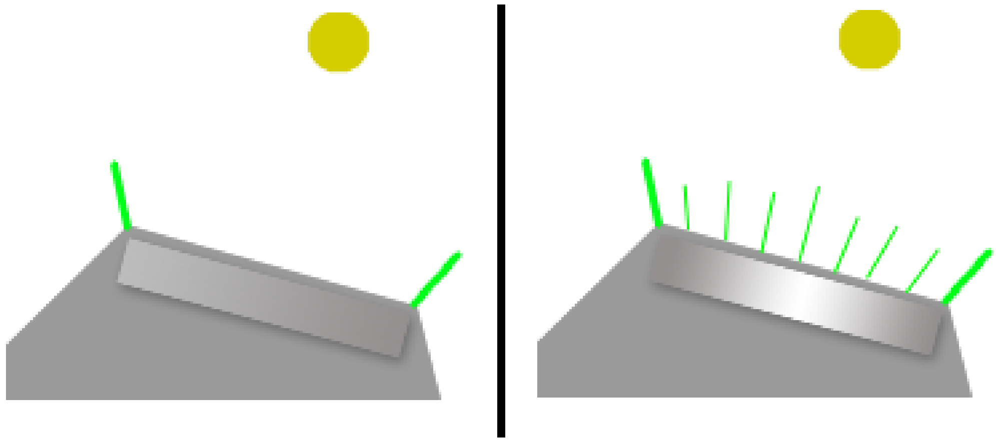
Specular light



Phong lighting model

Shading

- The Phong lighting model calculates light intensity using the surface normal of an object



Shading

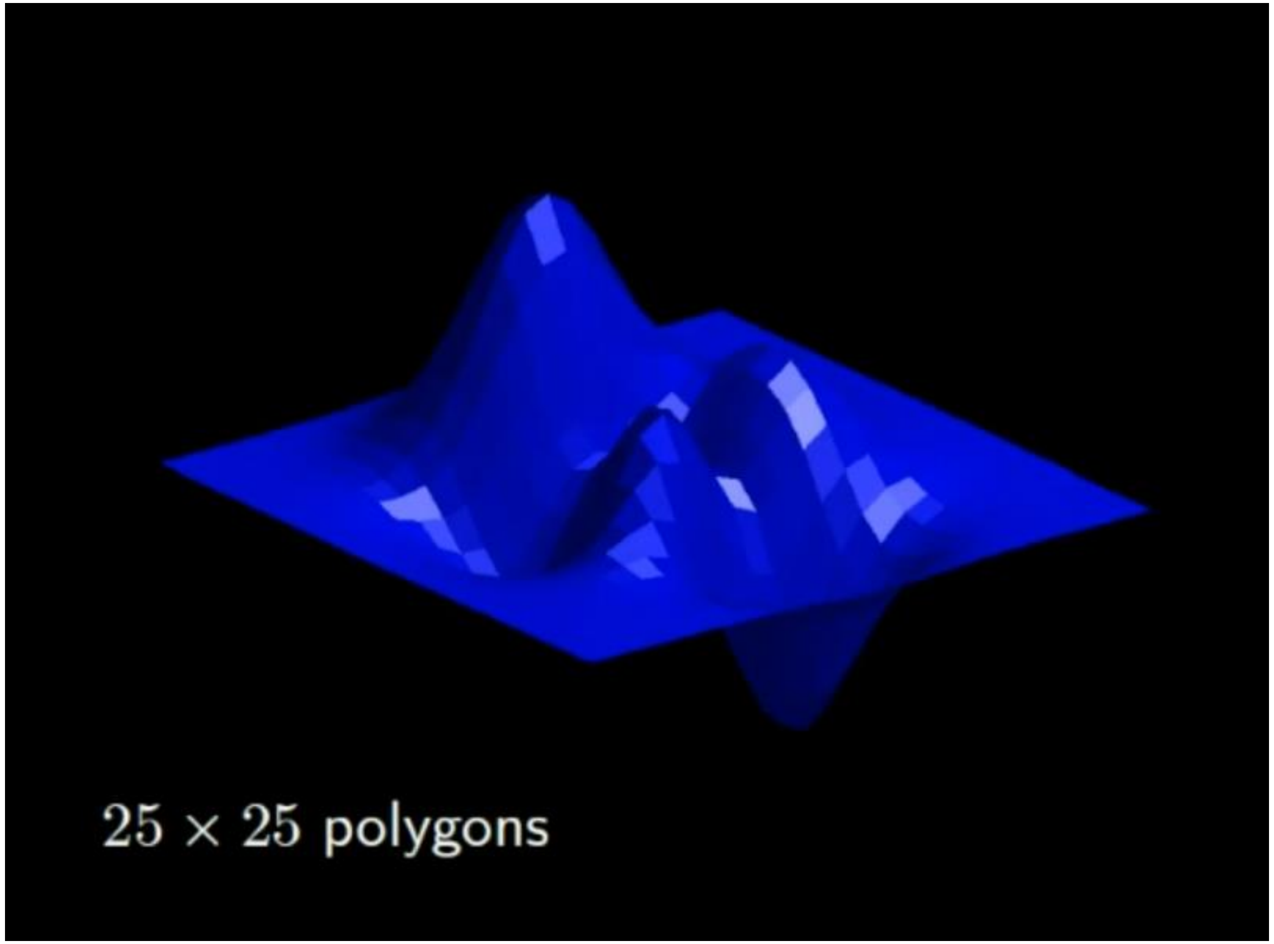
- The Phong lighting model calculates light intensity using the surface normal of an object
- The calculation of lighting for a whole triangle is called shading

Shading

- The Phong lighting model calculates light intensity using the surface normal of an object
- The calculation of lighting for a whole triangle is called shading
- Basic shading methods:
  - Lambert shading
  - Gouraud shading
  - Phong shading

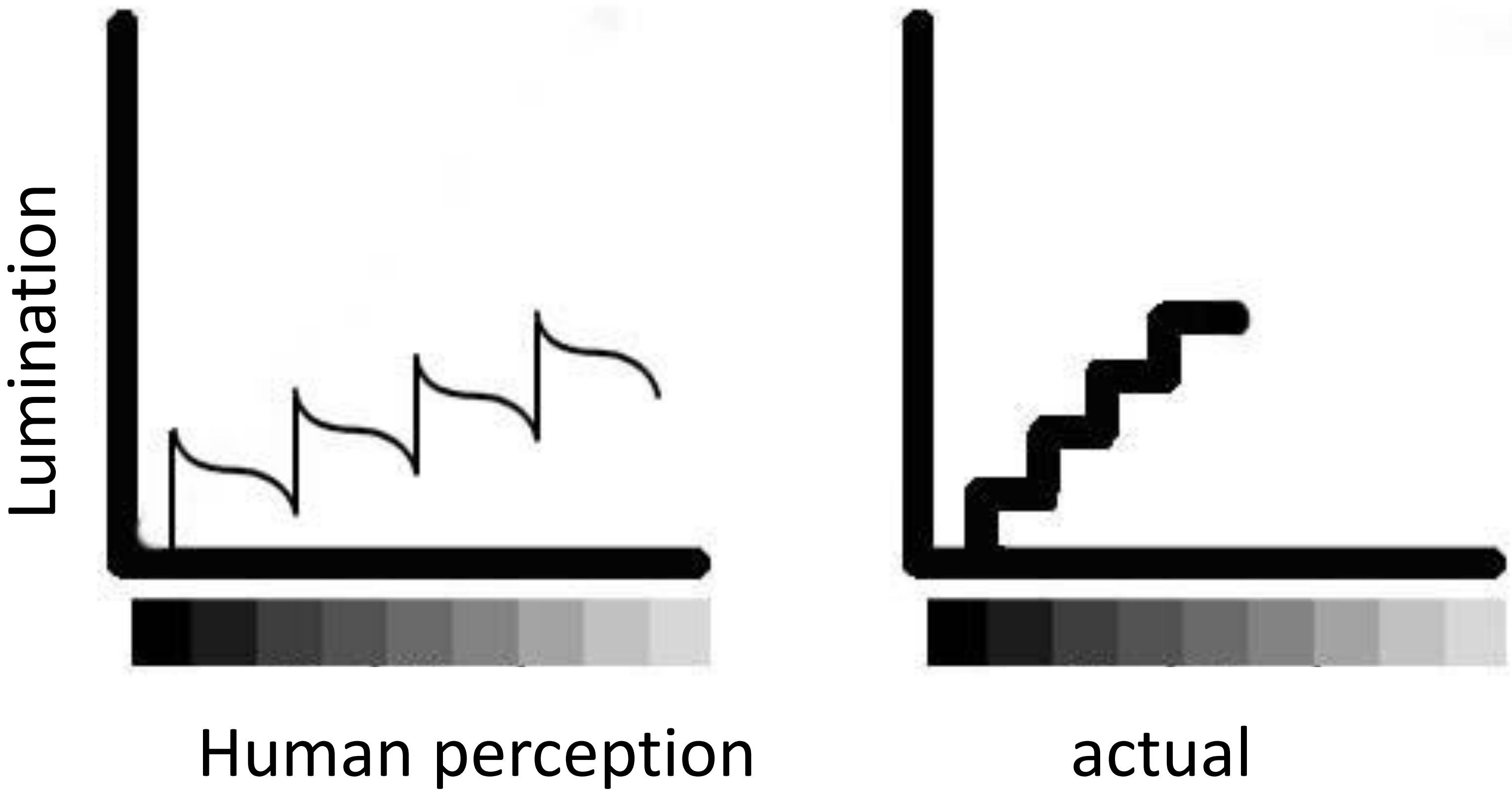
Lambert shading (flat shading)

- Flat shading assumes that all triangle pixels are shaded with the same color

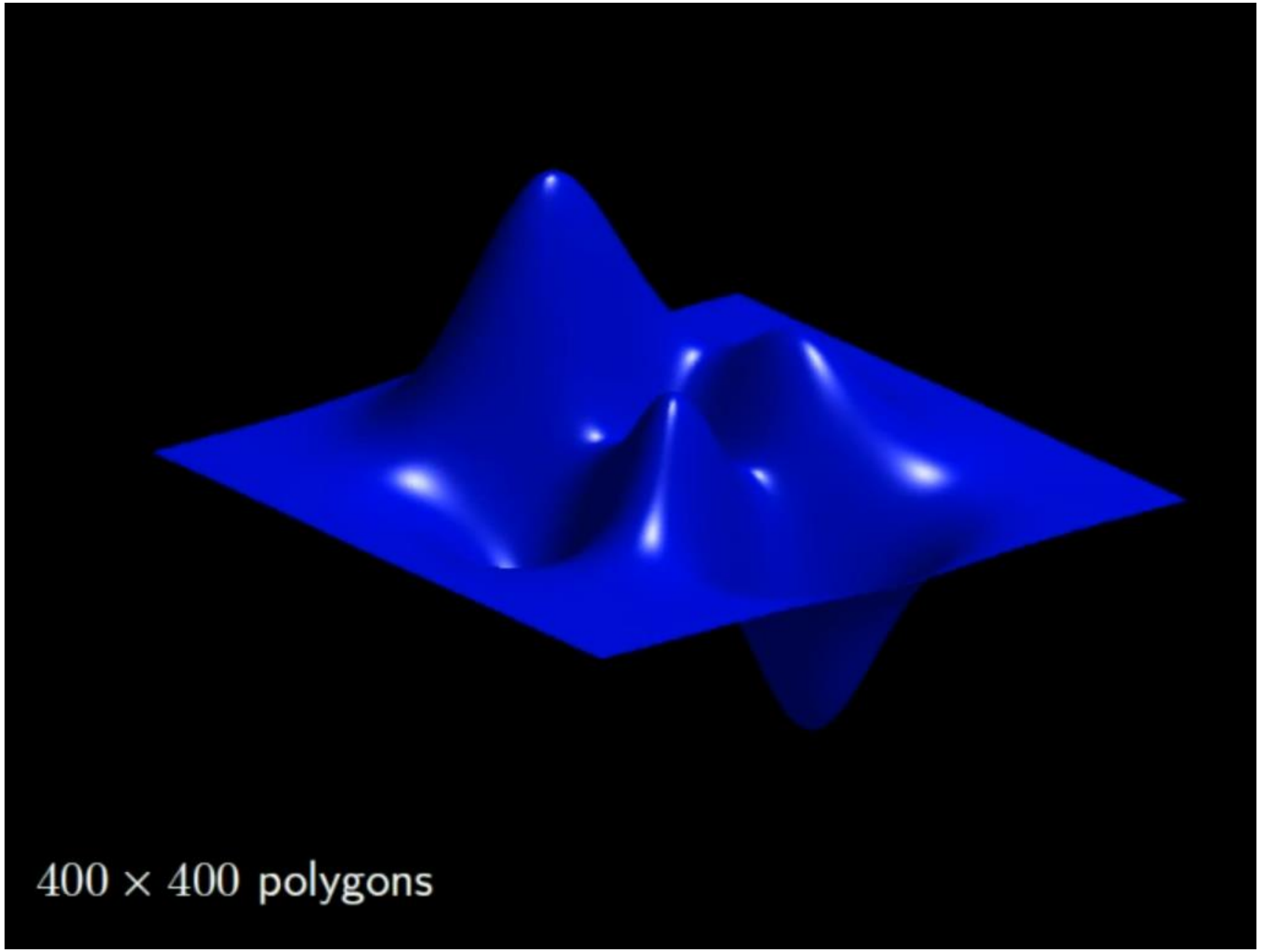


Mach bands optical illusion

Mach bands optical illusion

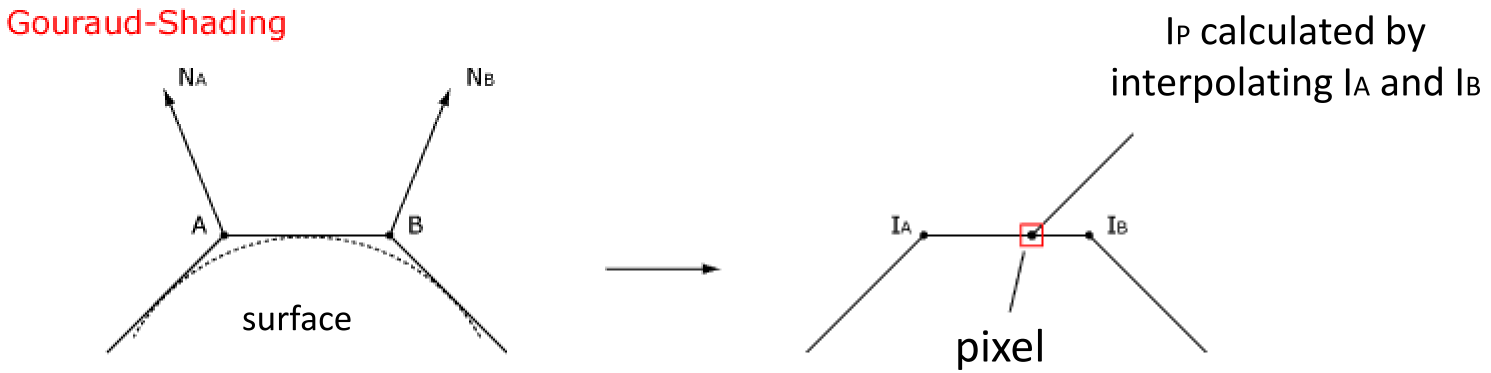


Solution: more polygons



Gouraud shading

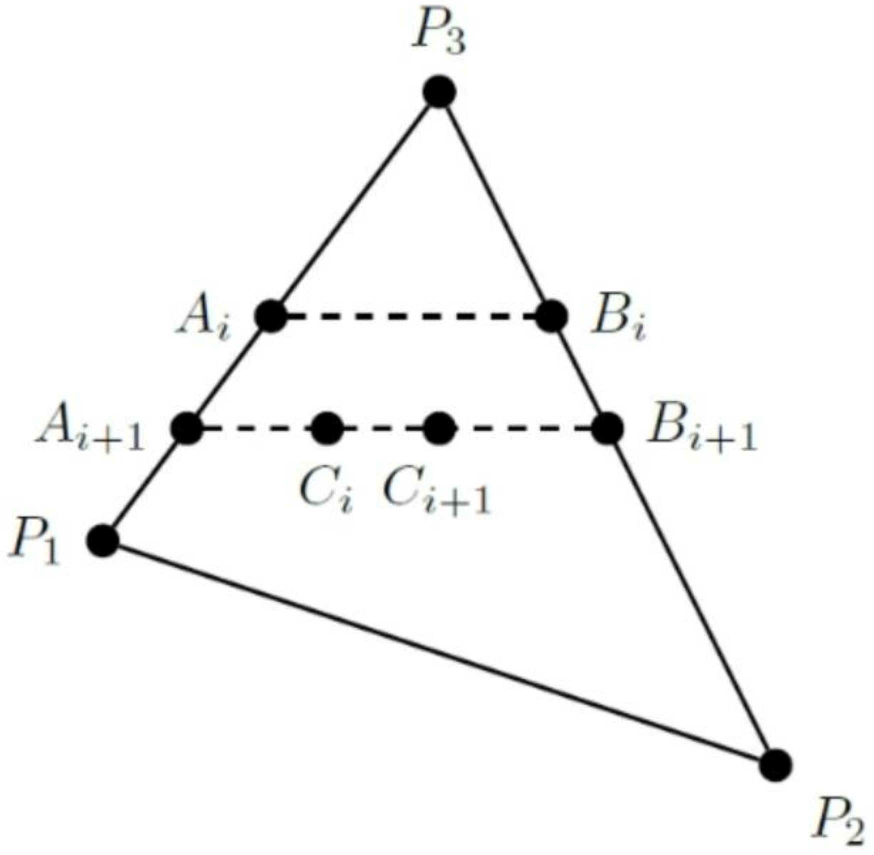
- Gouraud shading calculates lighting for **vertices** of a polygon and linearly interpolate light intensity values of all polygon pixels



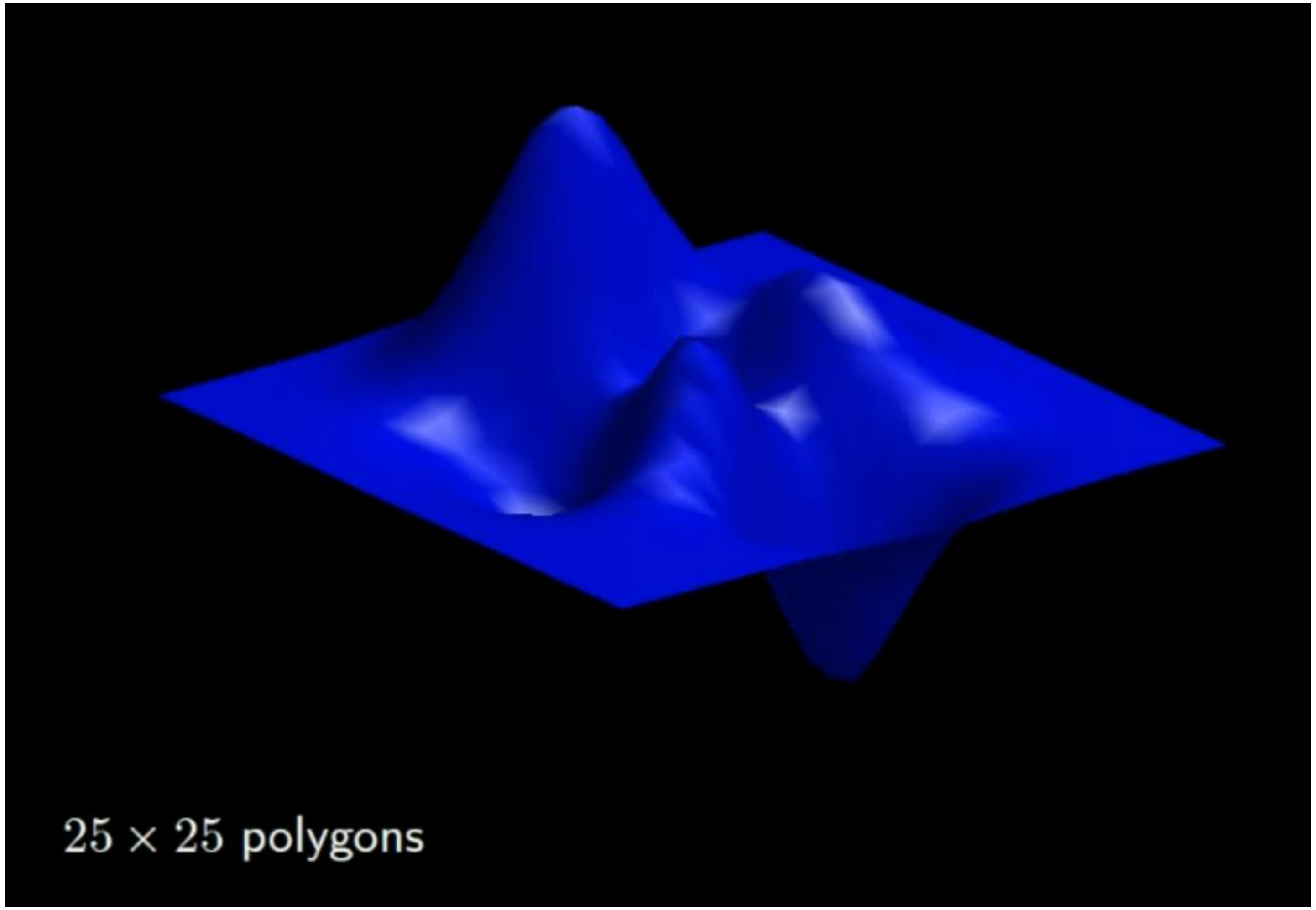
Interpolating intensities

- If *I* is the **light intensity** in a given point P:
- $I_{A,i+1} = I_{A,i} - \Delta I_A$
- $I_{B,i+1} = I_{B,i} - \Delta I_B$
- $I_{C,i+1} = I_{C,i} - \Delta I_C$

- Where
- $I_A = \frac{I_3 - I_1}{y_3 - y_1}, I_B = \frac{I_3 - I_2}{y_3 - y_2}, I_C = \frac{x_B - x_A}{I_B - I_A}$

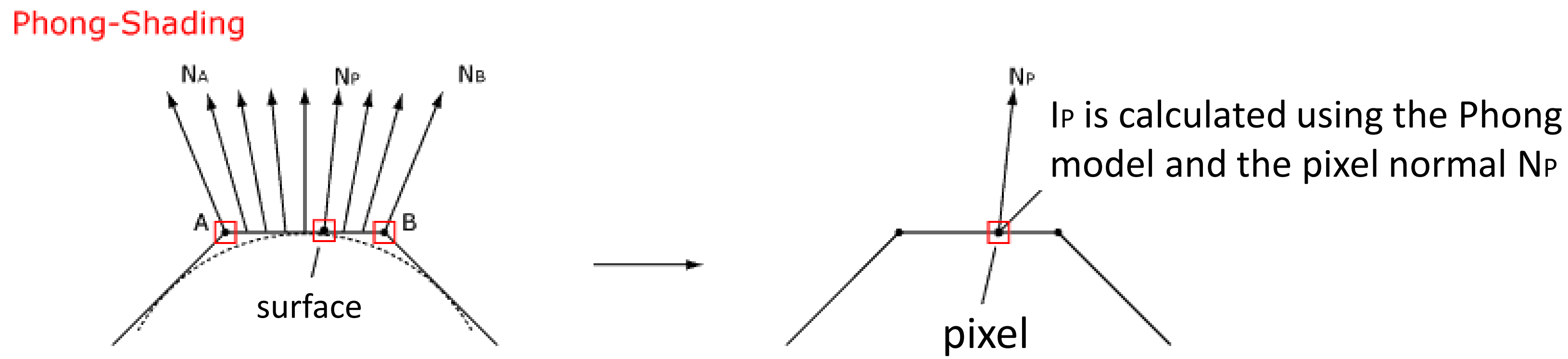


Gouraud shading



Phong shading

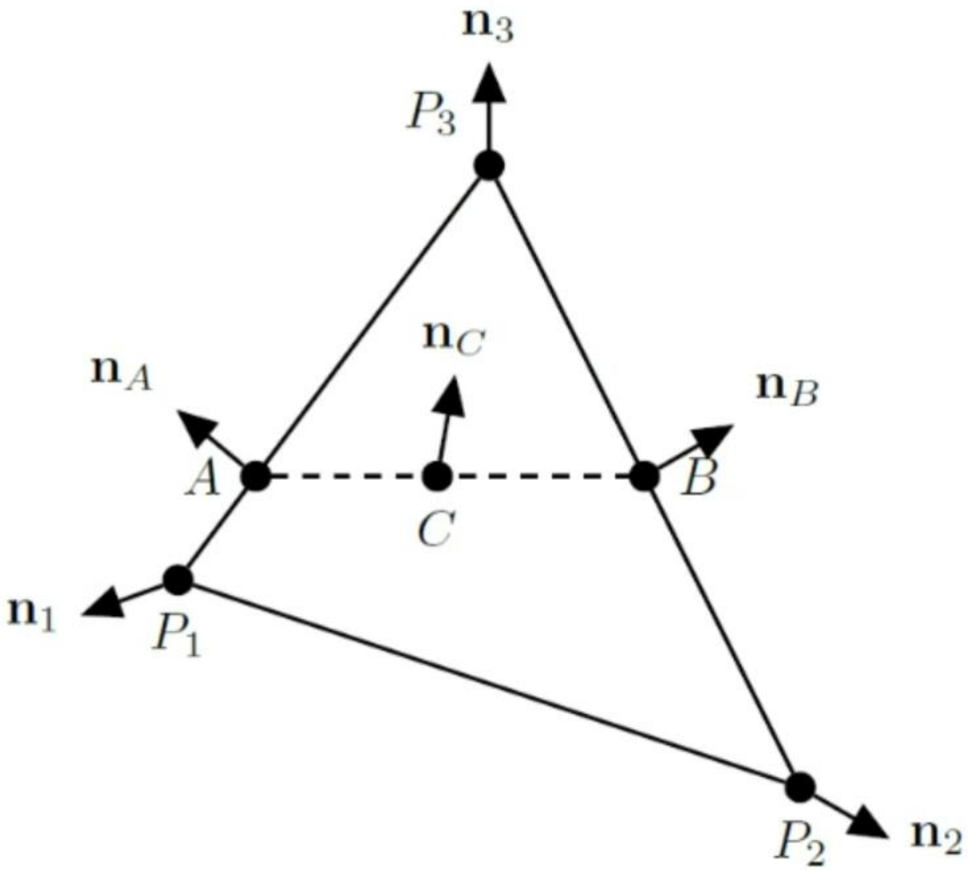
- Phong shading interpolates normal vectors of all pixels of a polygon



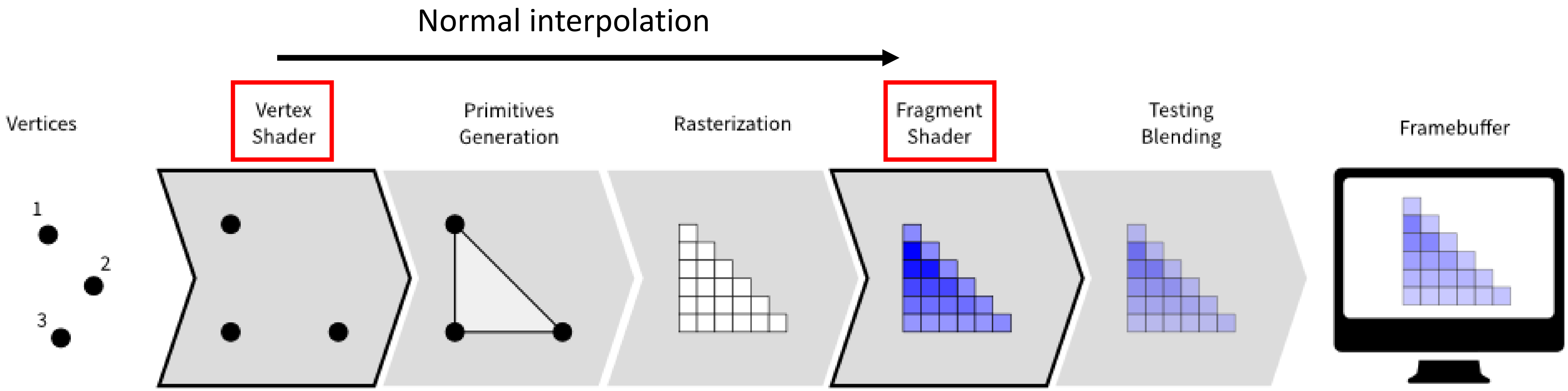
Normal interpolation

- If **n** is a normal vector of a vertex P:
- $\mathbf{n}_{A,i+1} = \mathbf{n}_{A,i} - \Delta\mathbf{n}_A$
- $\mathbf{n}_{B,i+1} = \mathbf{n}_{B,i} - \Delta\mathbf{n}_B$
- $\mathbf{n}_{C,i+1} = \mathbf{n}_{C,i} - \Delta\mathbf{n}_C$

- Where
- $\mathbf{n}_A = \frac{\mathbf{n}_3 - \mathbf{n}_1}{y_3 - y_1}, \mathbf{n}_B = \frac{\mathbf{n}_3 - \mathbf{n}_2}{y_3 - y_2}, \mathbf{n}_C = \frac{\mathbf{n}_B - \mathbf{n}_A}{x_B - x_A}$

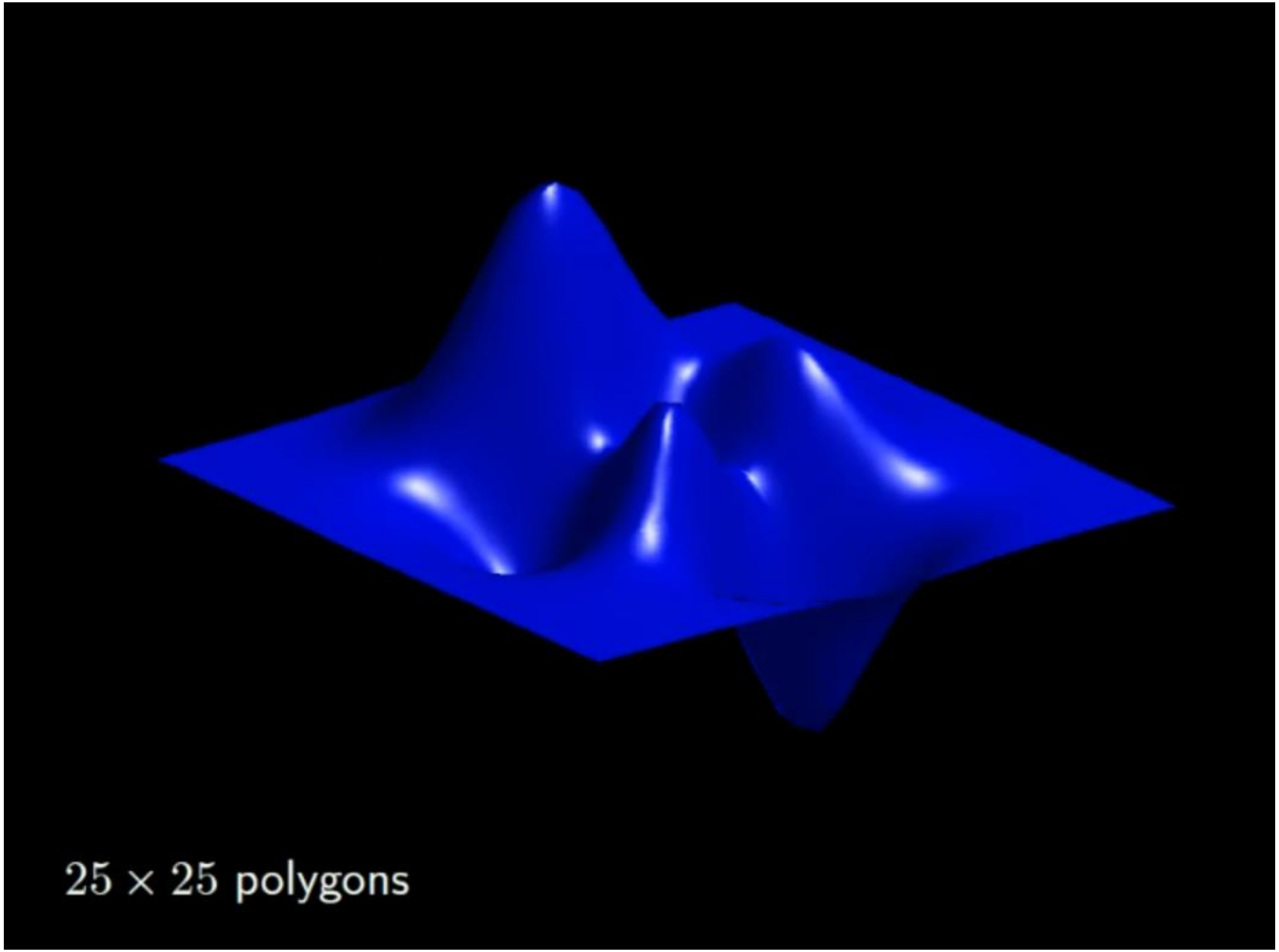


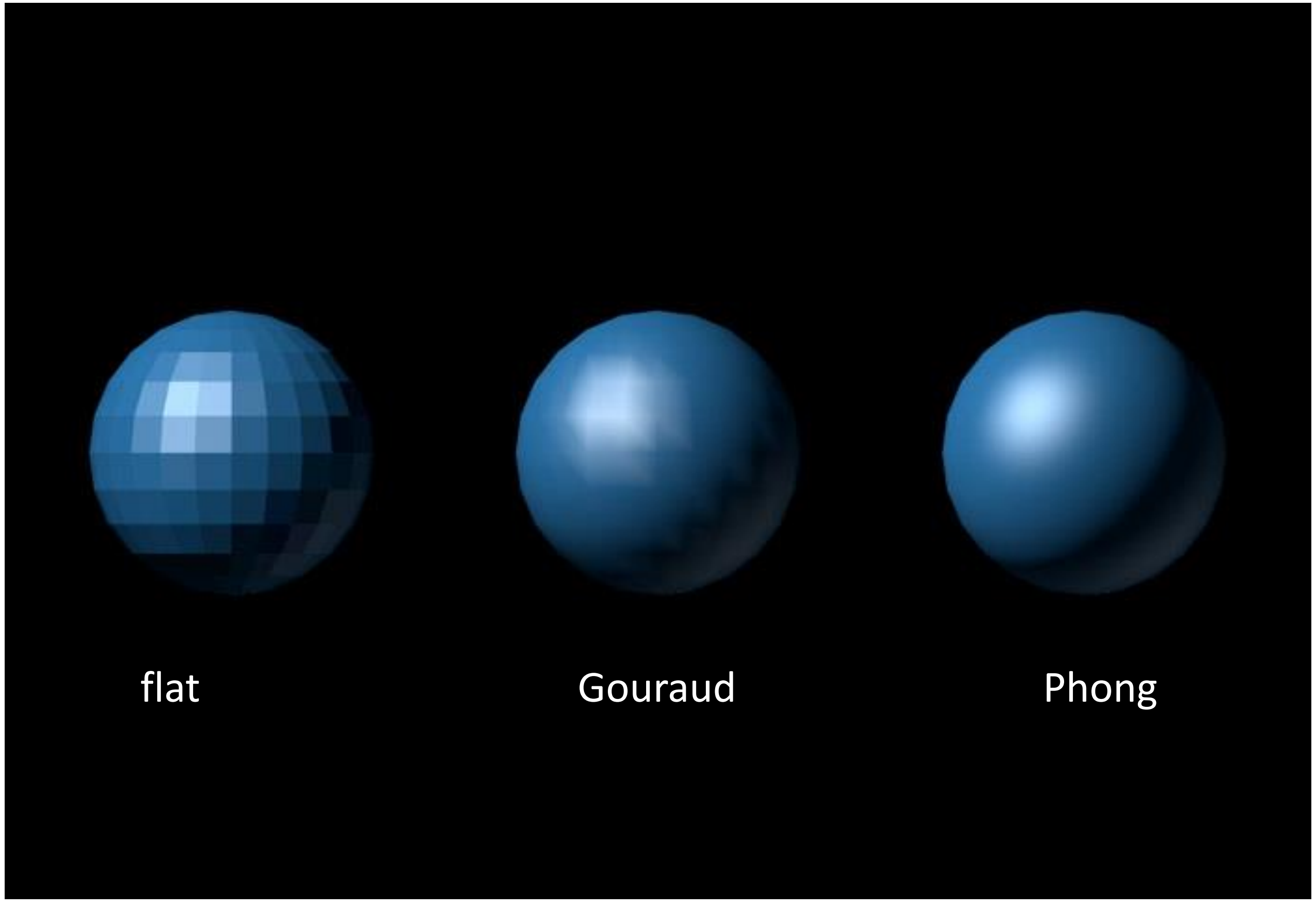
OpenGL



Normal interpolation
Vertices
1
2
3
Vertex Shader
Primitives Generation
Rasterization
Fragment Shader
Testing Blending
Framebuffer

Normal Map Visualized with RGB Model


25 × 25 polygons

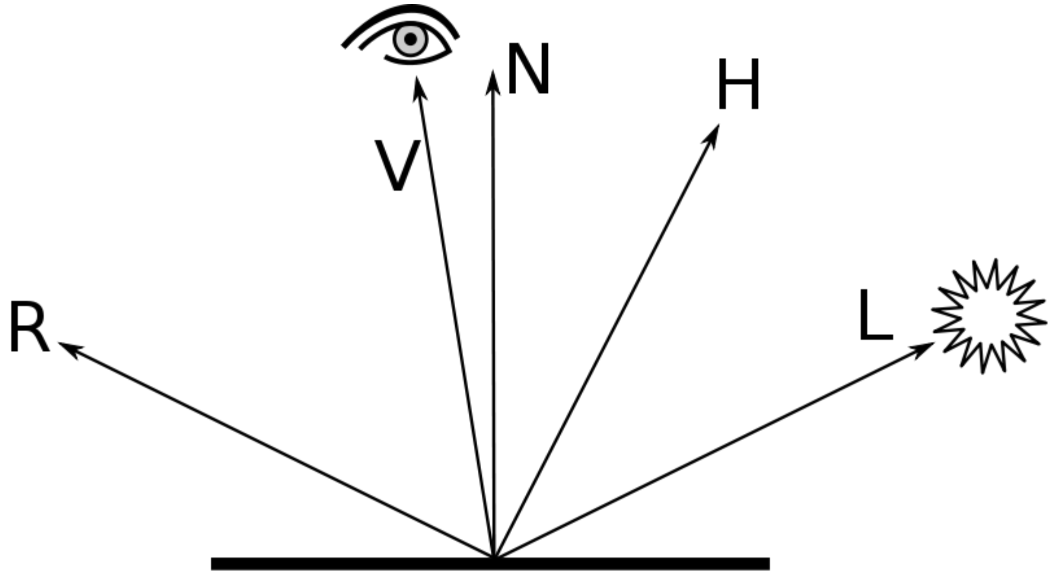

flat
Gouraud
Phong

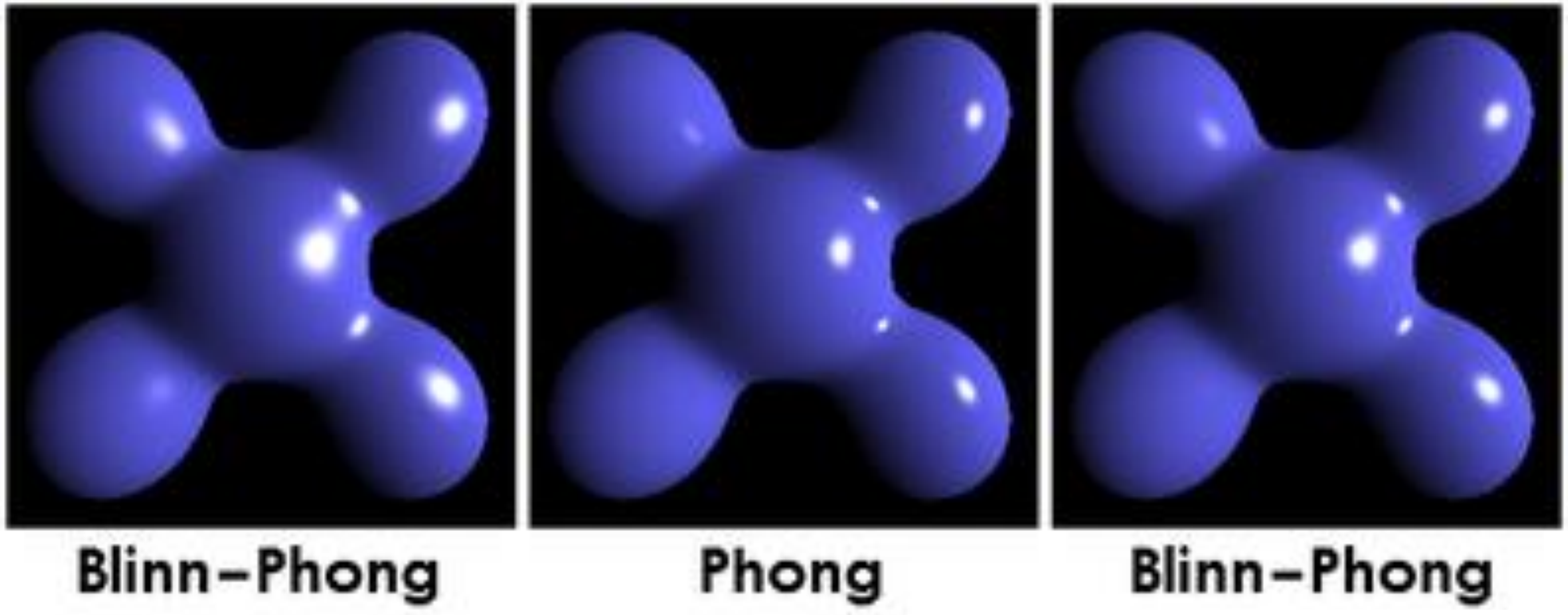
Blinn-Phong

- A popular variation of the Phong model implemented in OpenGL and Direct3D
- Instead of calculating vector R we define a unit vector exactly half-way between light vector L and the view vector V

- $\boldsymbol{I_p k_s (N \cdot H)^n}$, where $\boldsymbol{H} = \frac{\boldsymbol{L+V}}{|\boldsymbol{L+V}|}$

- Calculation of H is more efficient than R





Higher exponent

https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_shading_model