Lighting and Shading

CMSC 161: Interactive Computer Graphics

2nd Semester 2014-2015

Institute of Computer Science

University of the Philippines – Los Baños

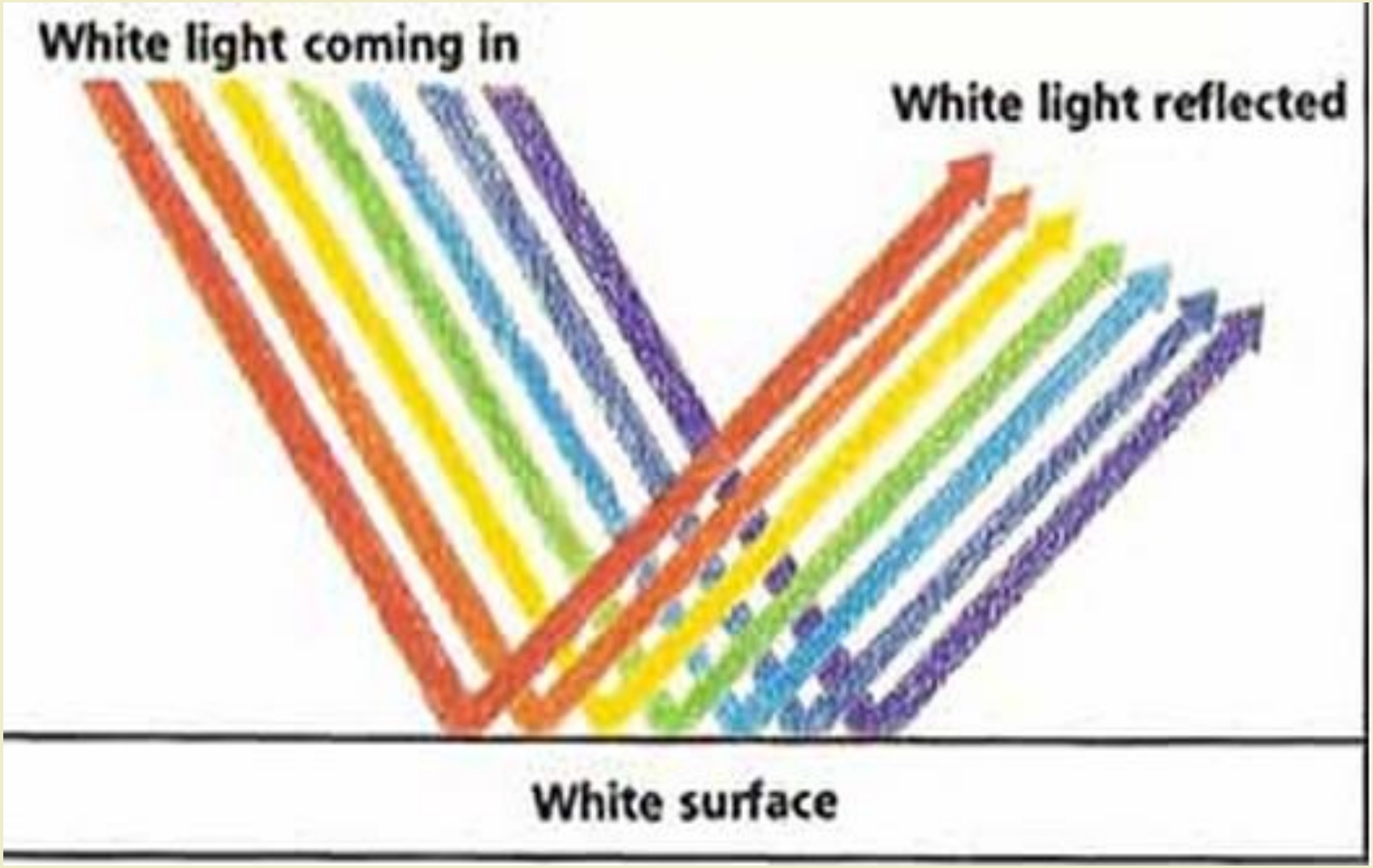
Lecture by James Carlo Plaras

Light

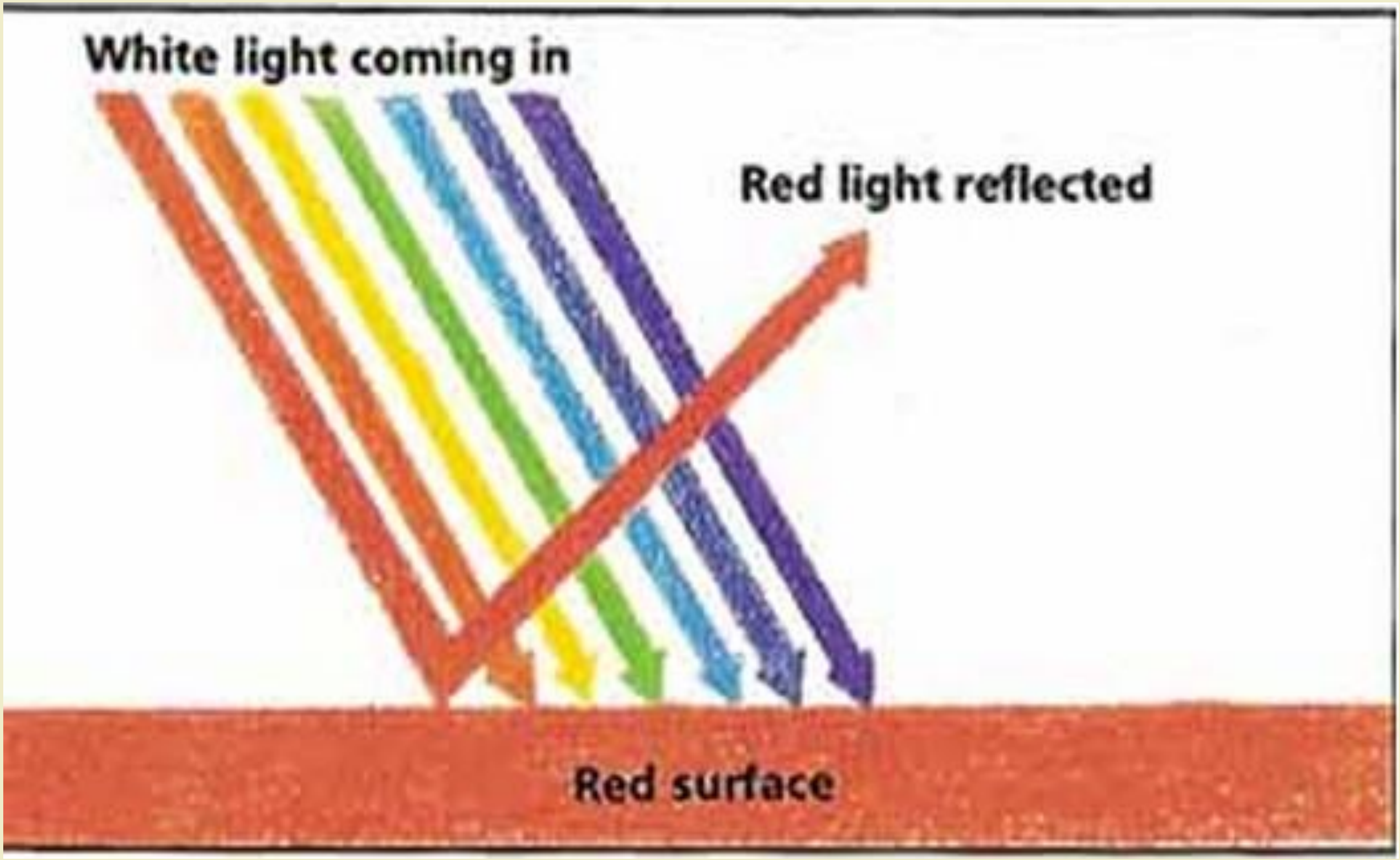
the natural agent that stimulates sight and makes things visible

a source of illumination

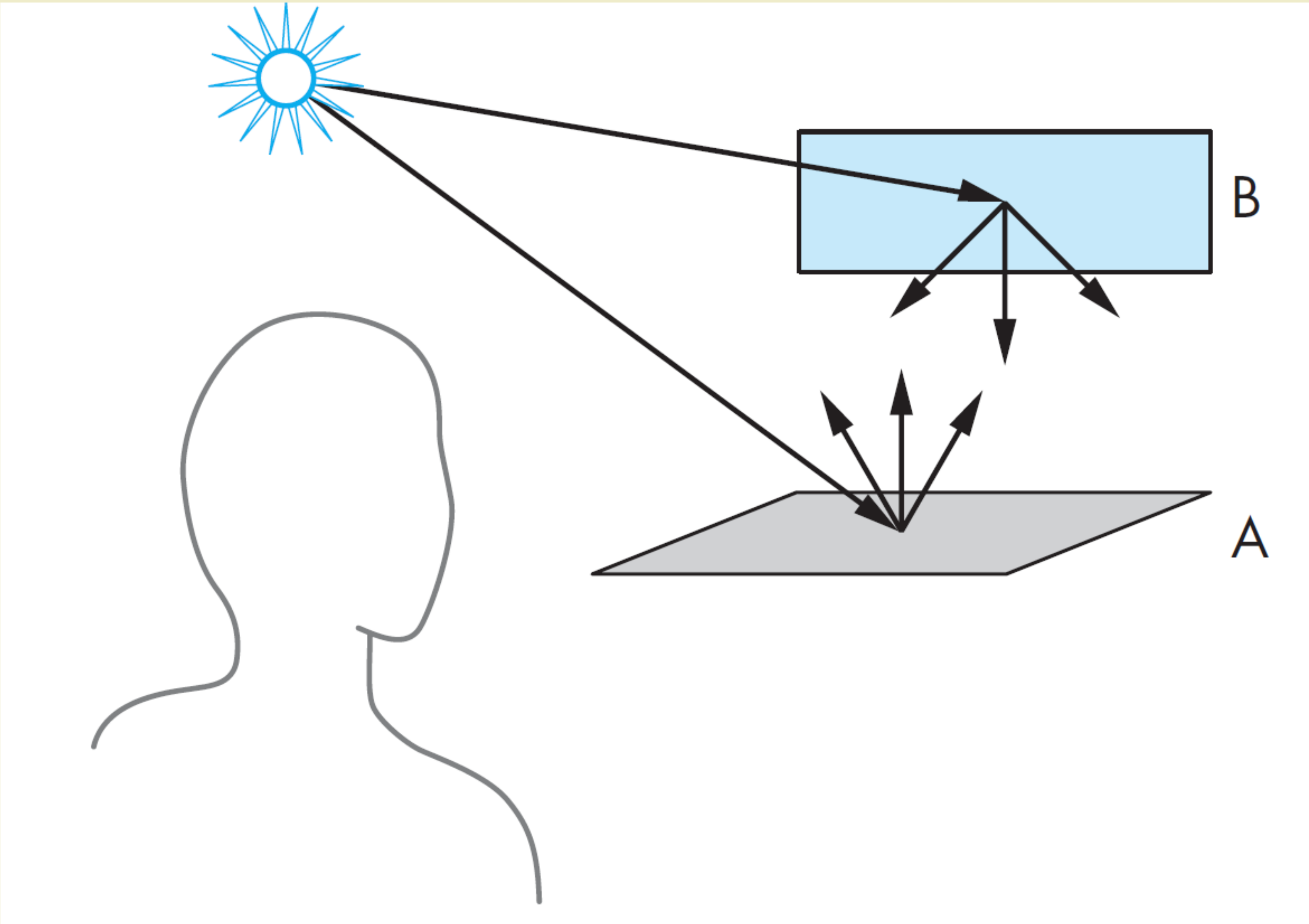
Lighting in the Real World



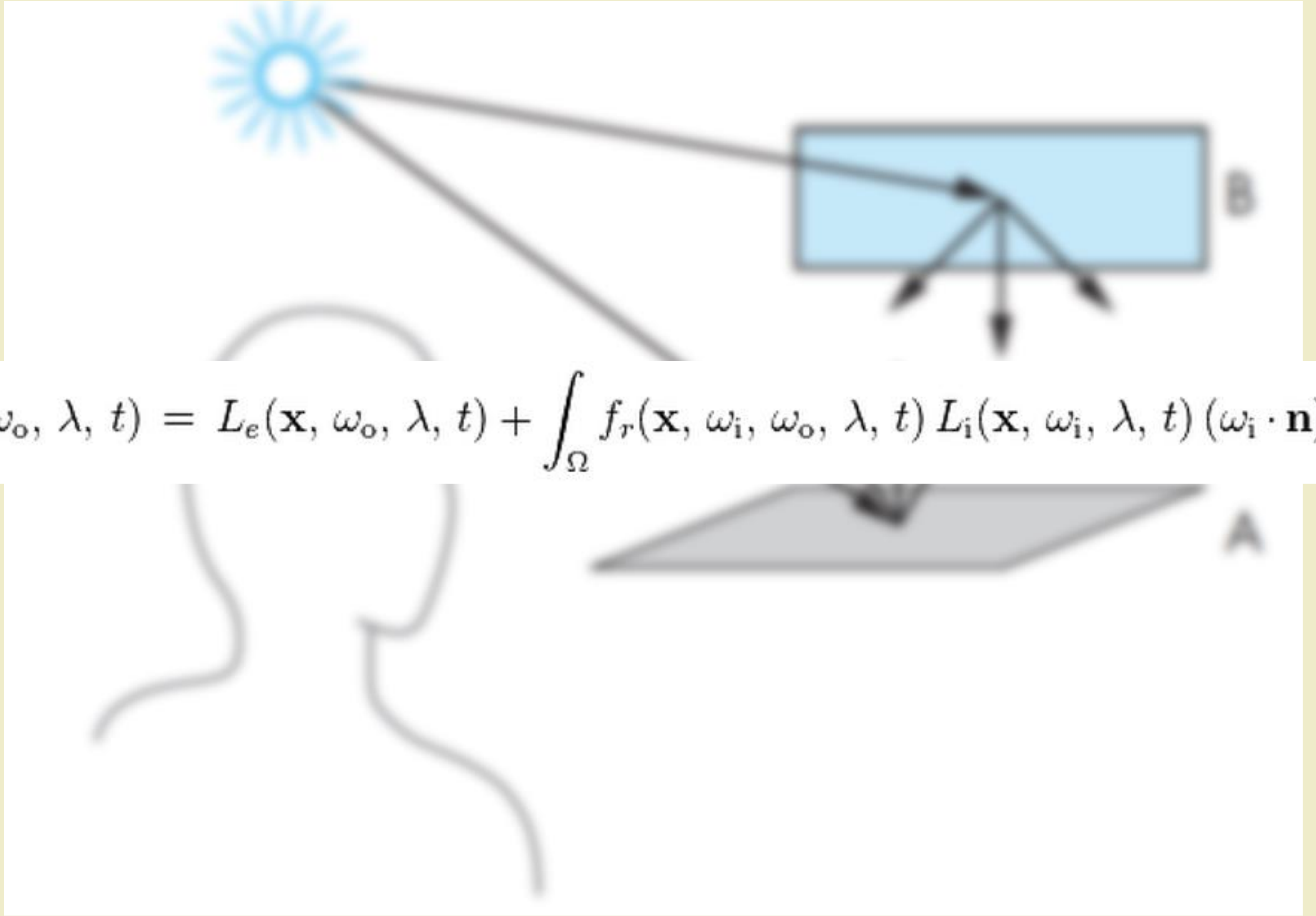
Lighting in the Real World



Lighting in the Real World



Rendering Equation



$$L_{\mathrm{o}}(\mathbf{x},\,\omega_{\mathrm{o}},\,\lambda,\,t) \,=\, L_e(\mathbf{x},\,\omega_{\mathrm{o}},\,\lambda,\,t) + \int_{\Omega} f_r(\mathbf{x},\,\omega_{\mathrm{i}},\,\omega_{\mathrm{o}},\,\lambda,\,t)\, L_{\mathrm{i}}(\mathbf{x},\,\omega_{\mathrm{i}},\,\lambda,\,t)\,(\omega_{\mathrm{i}}\cdot\mathbf{n})\,\mathrm{d}\,\omega_{\mathrm{i}}$$

Approximating Lighting

Numerical methods for computing the rendering equation are not fast enough for real-time graphics

Approximating Lighting

Solved by using

Local approximation lighting models

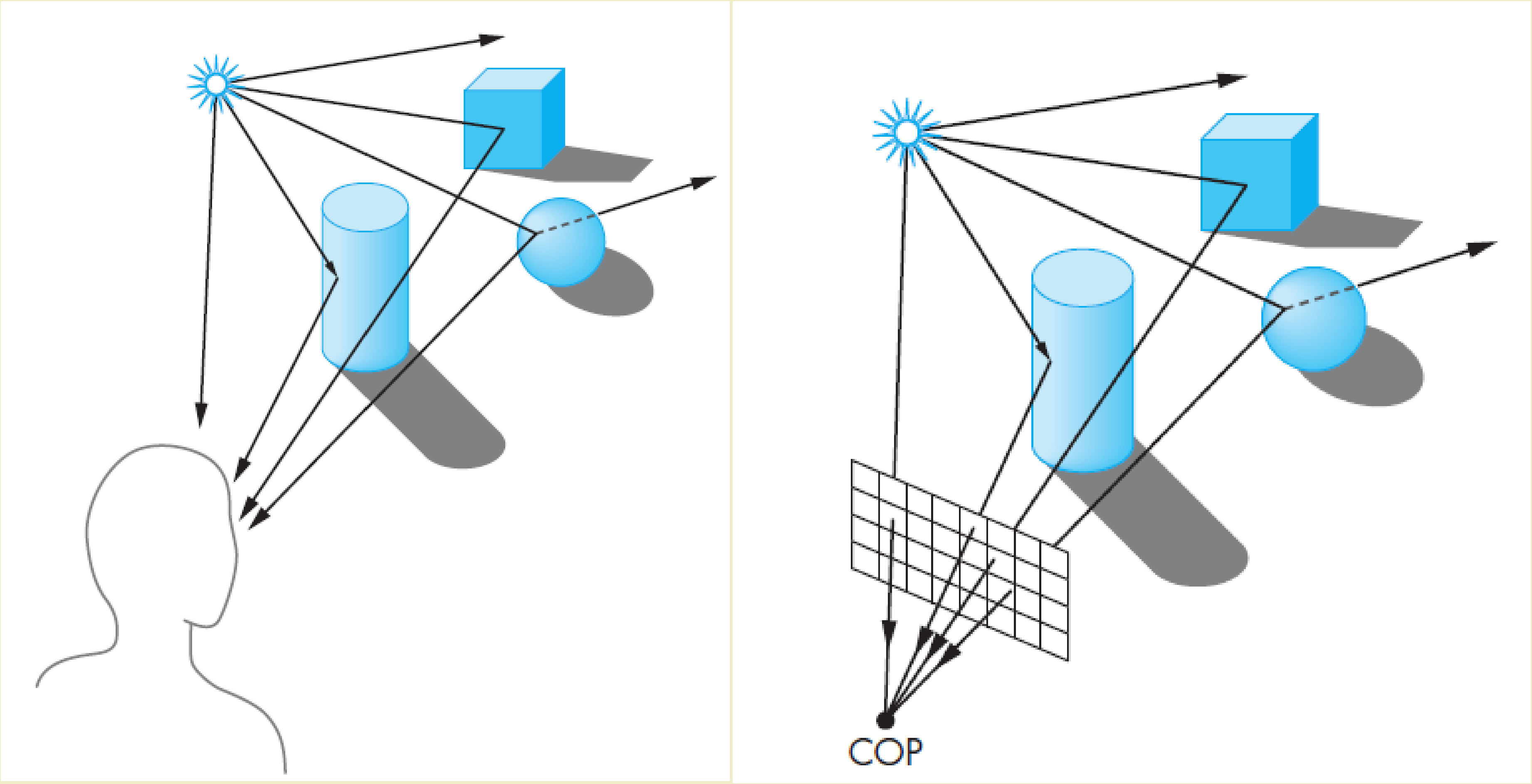
Local approximation lighting models

Direct interactions between light sources and surface material

Local approximation lighting models

Multiple object interaction is not considered

Approximating Lighting



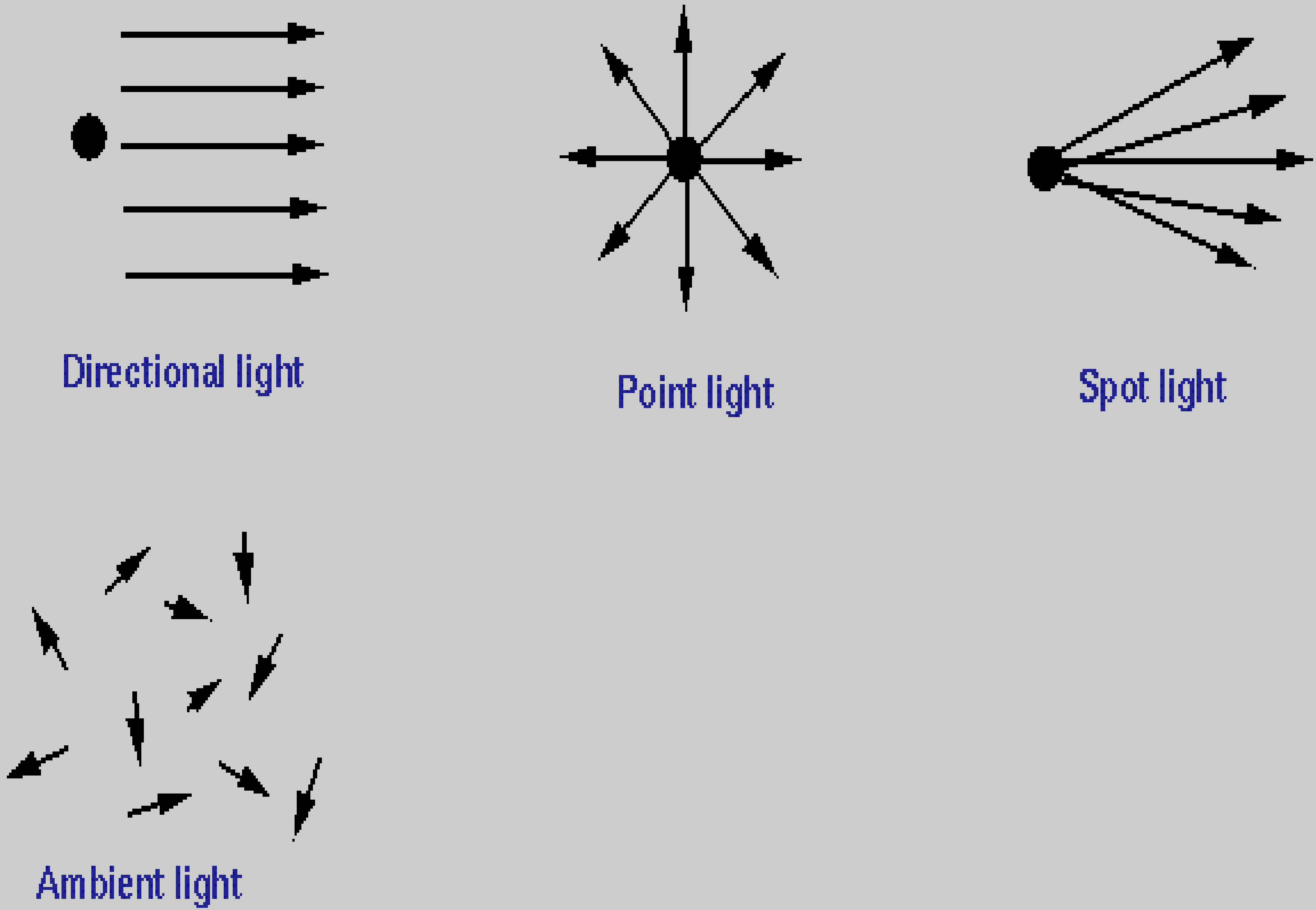
LIGHT SOURCES

Light Sources

Ambient Light

Point Light (and Spot Light)

Directional Light


Directional light
Point light
Spot light
Ambient light

Ambient Light (Real World)

Source of light that is not explicitly supplied by the photographer for the purpose of taking photos

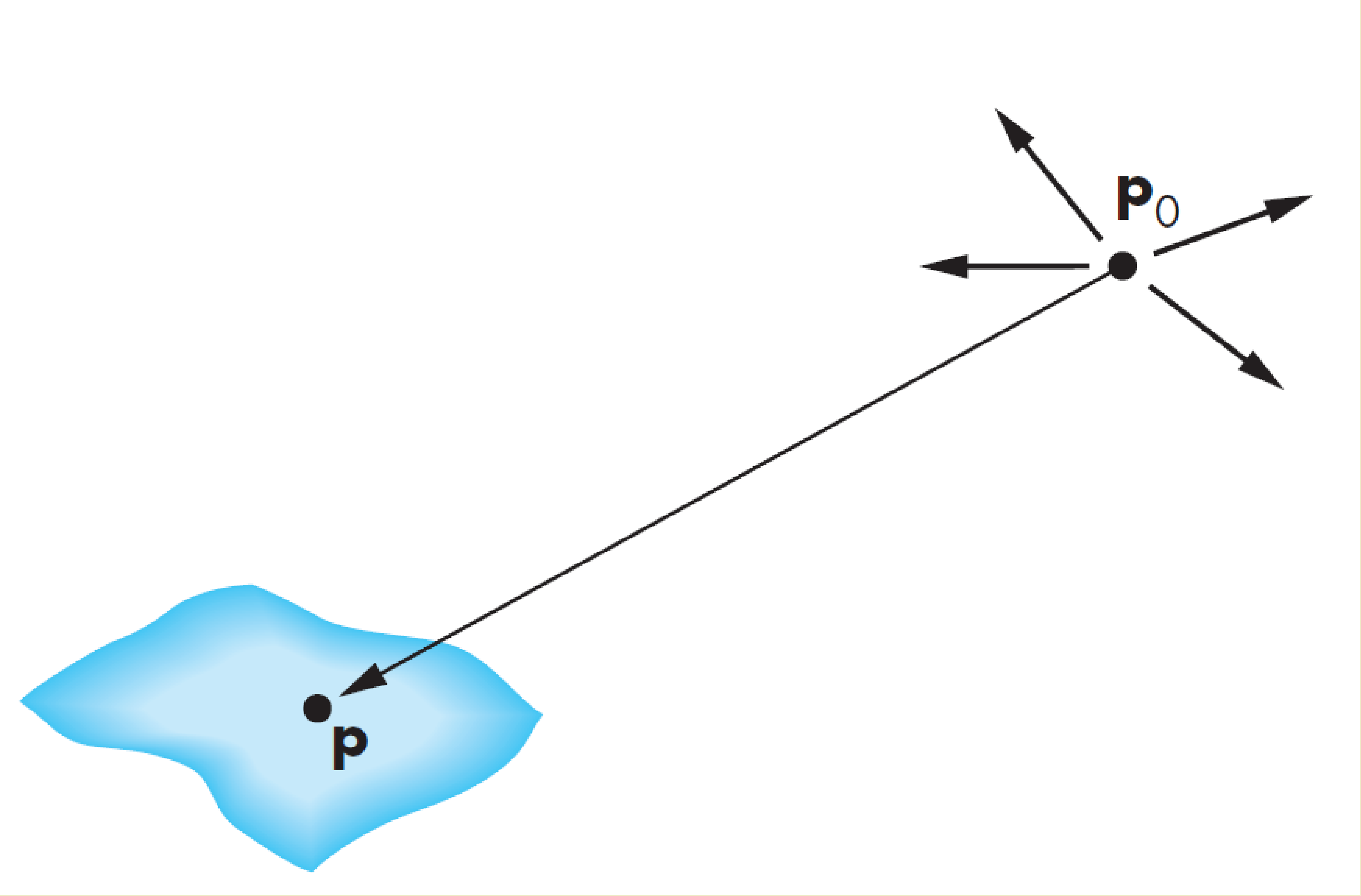
Ambient Light (Computer Graphics)

Uniform illumination on all objects

Crude approximation to global light scattering effects

Ambient Light

Point Light



Point Light

A light located at some point P

Point Light

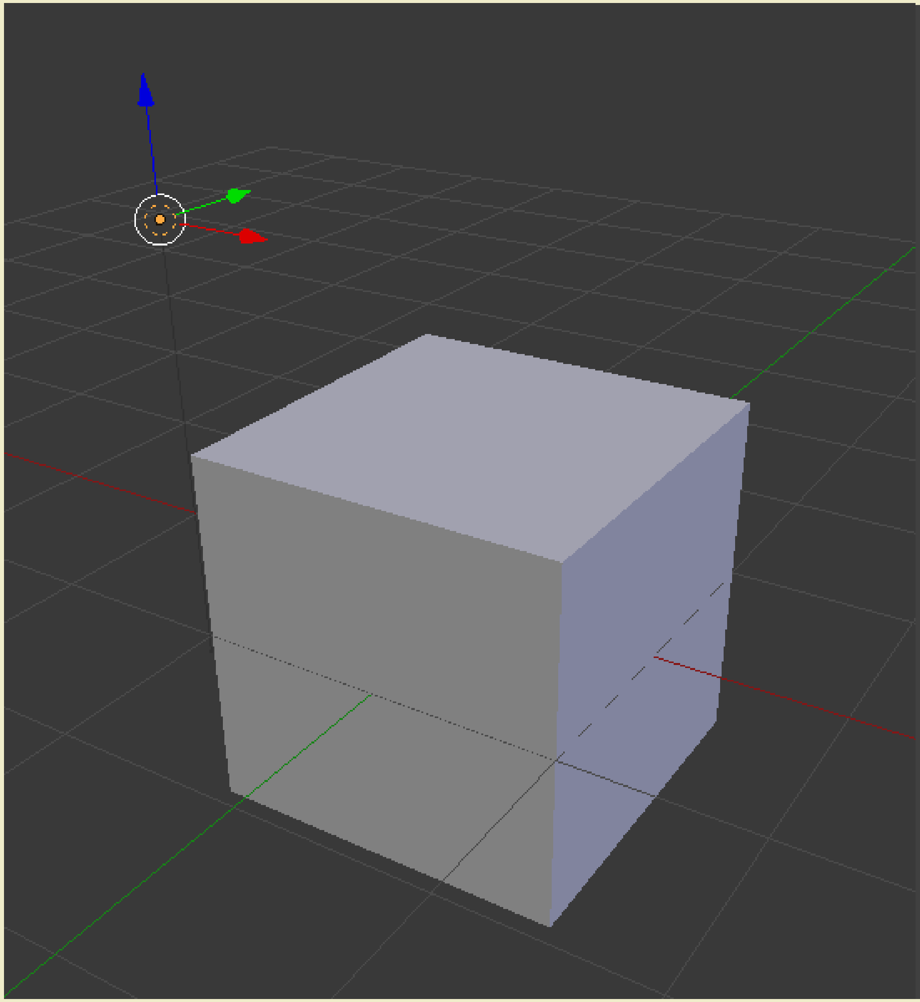
Emit light equally in all directions

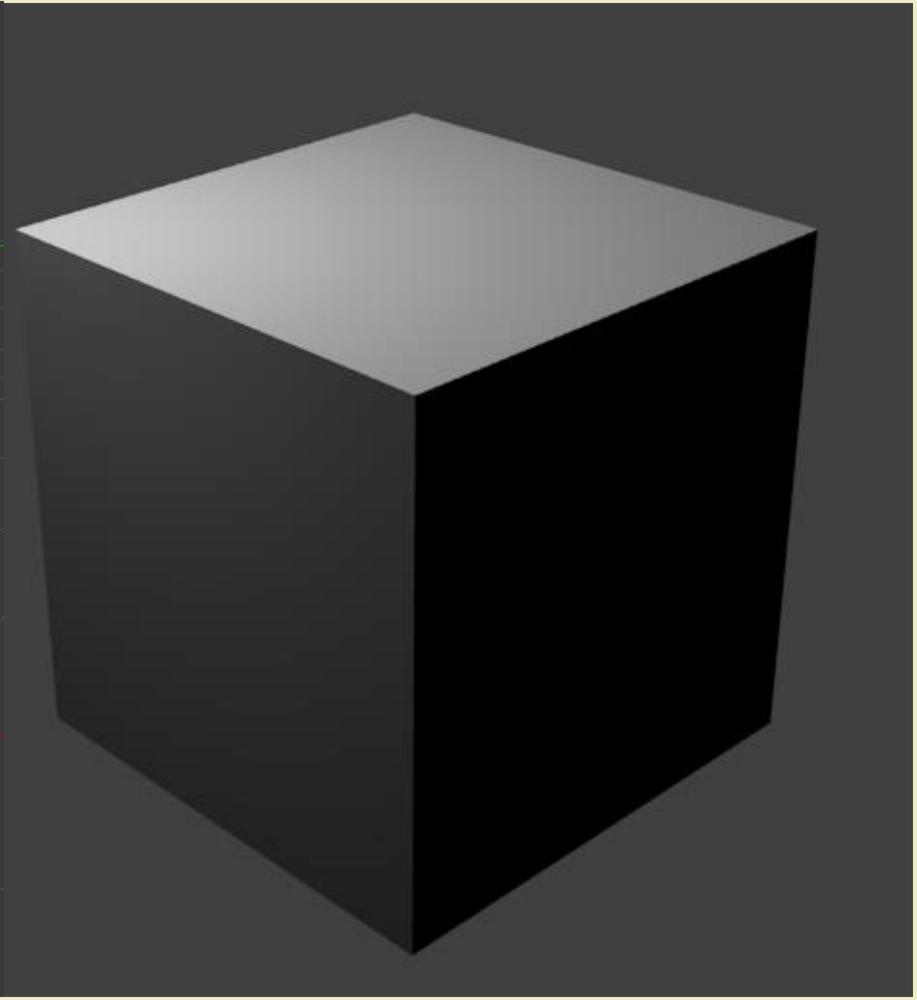
Point Light

Exhibits attenuation of light wave

Illumination from point source is inversely proportional to distance of point light to surface

Point Light





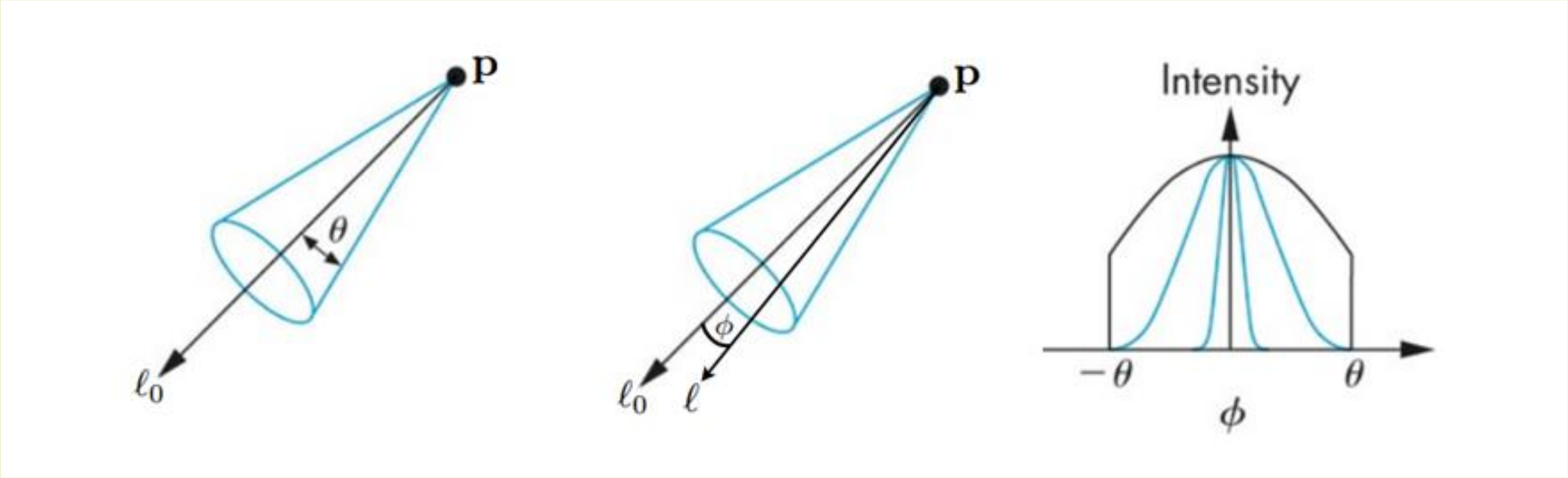
Spotlights

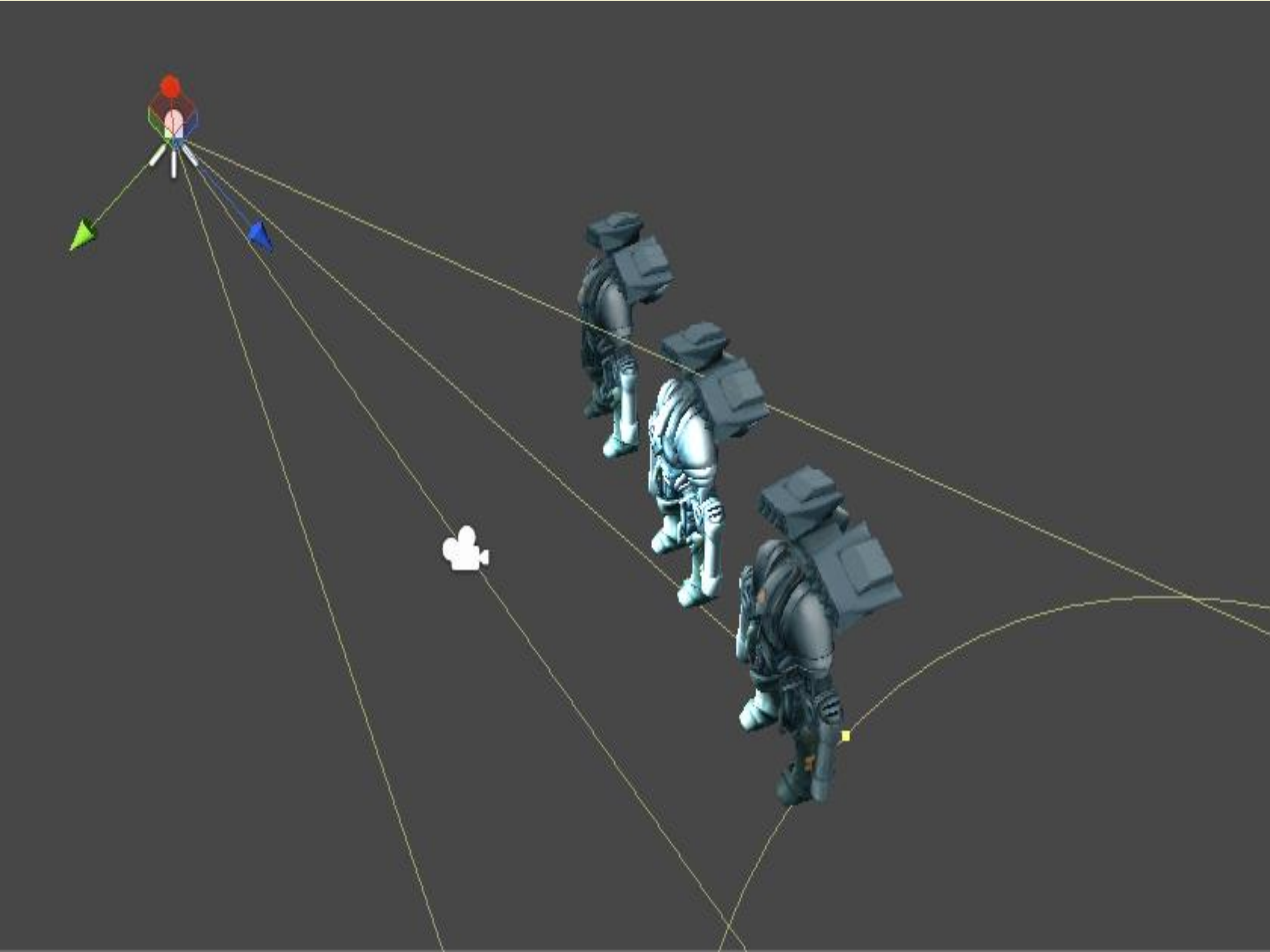
Special Point Light

Spotlights

Similar to point light but limiting the light emission to narrow range of angles

Spotlights



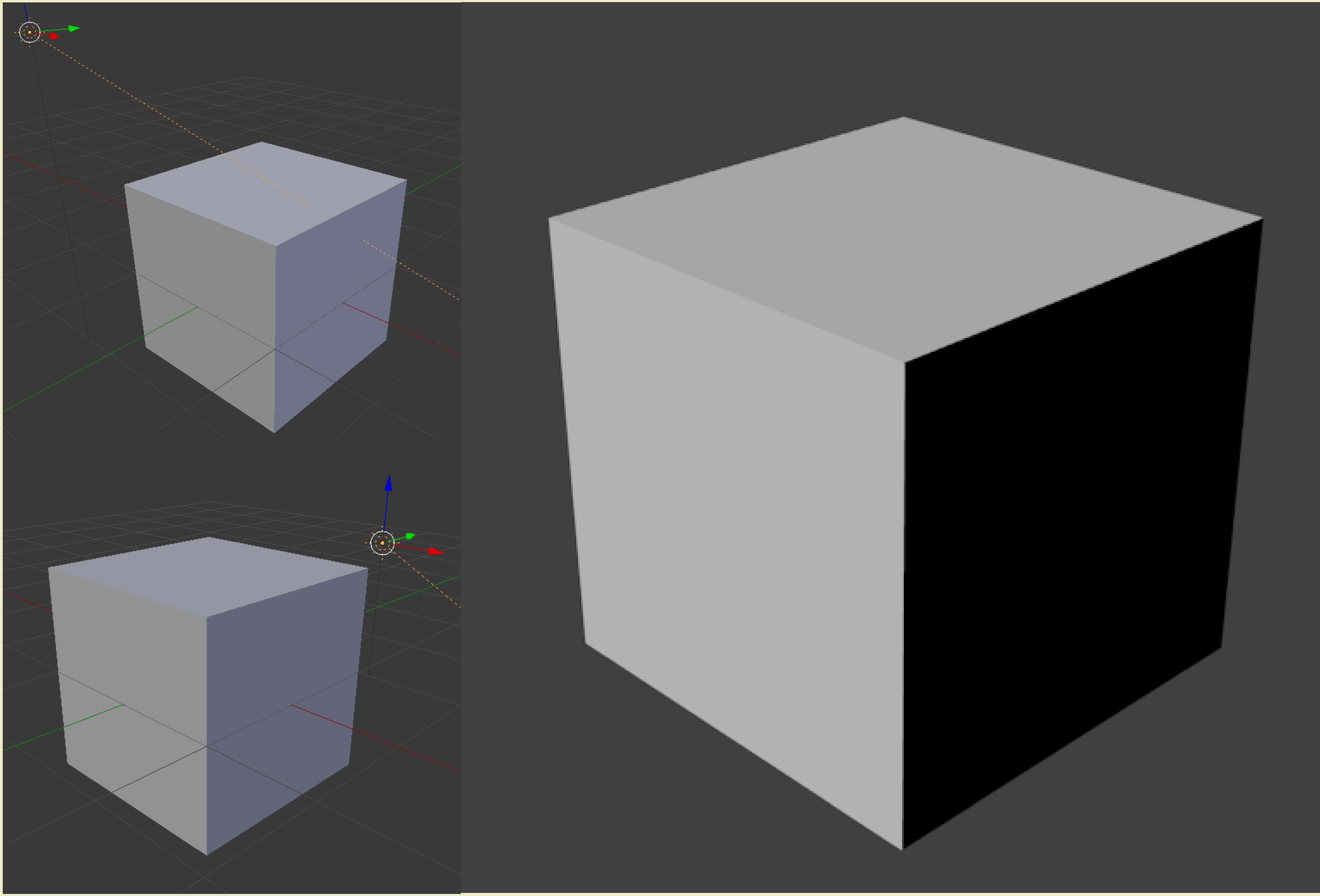


Directional Light

Approximates distant light sources where only the surface angles matter and not the distance

Directional Light

Sun is a best real world example of directional light



LIGHT-MATERIAL INTERACTION

Light-Material Interaction

Specular surface

Diffuse surface

Translucent surface

Specular Surface

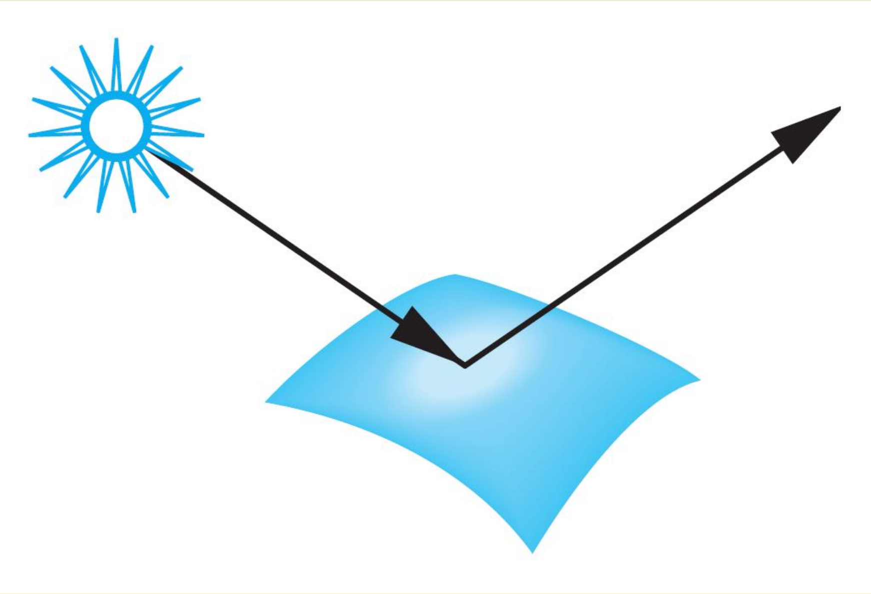
Light reflected is scattered in narrow range of angles close to the angle of reflection

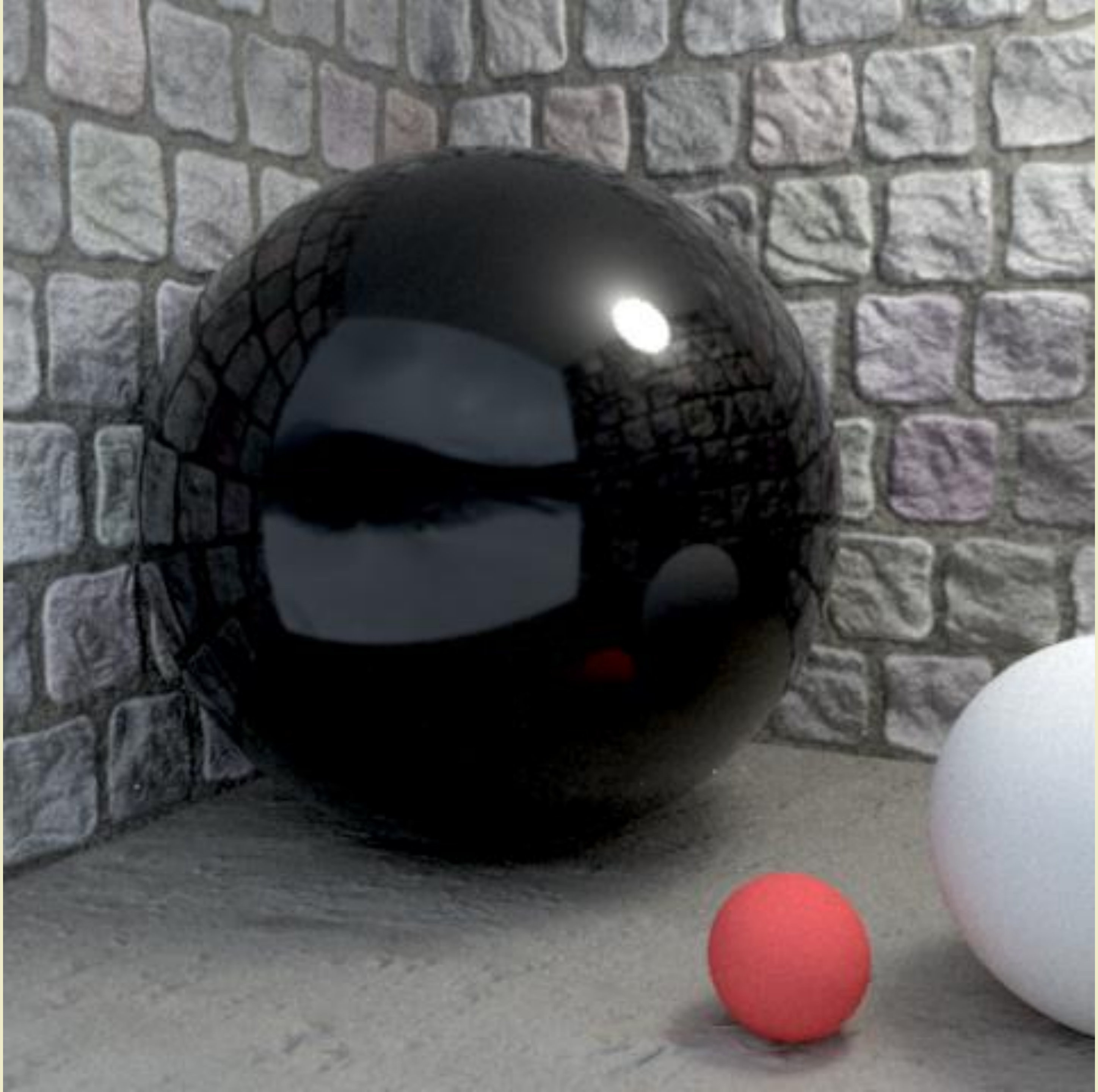
Specular Surface

Perfectly specular surface

All reflected light emerges at a single angle (Mirrors)

Specular Surface





Diffuse Surface

Light reflected is scattered in all directions

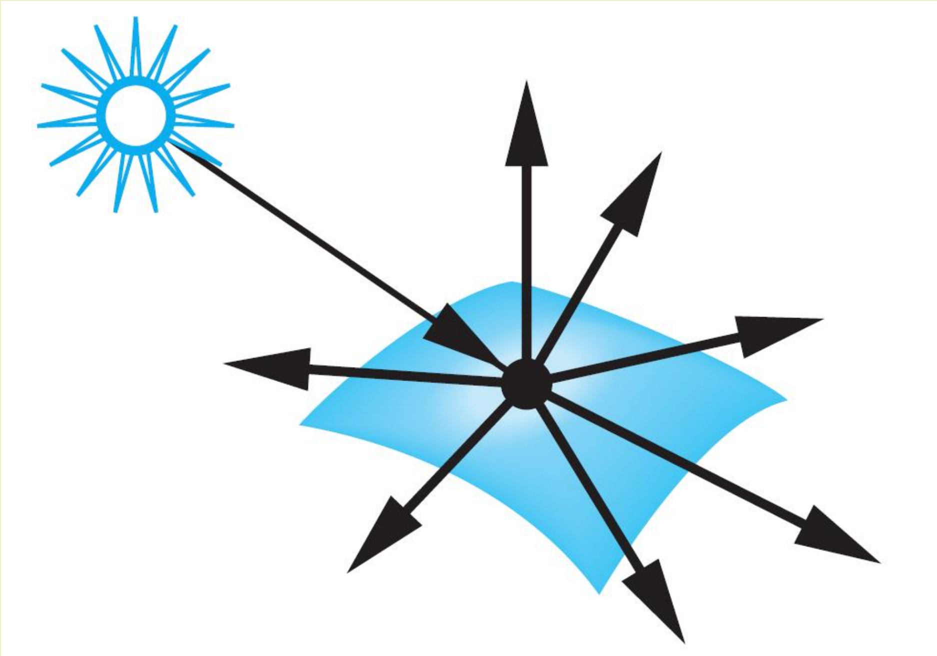
Matte (car paint), Flat paint, Fabric

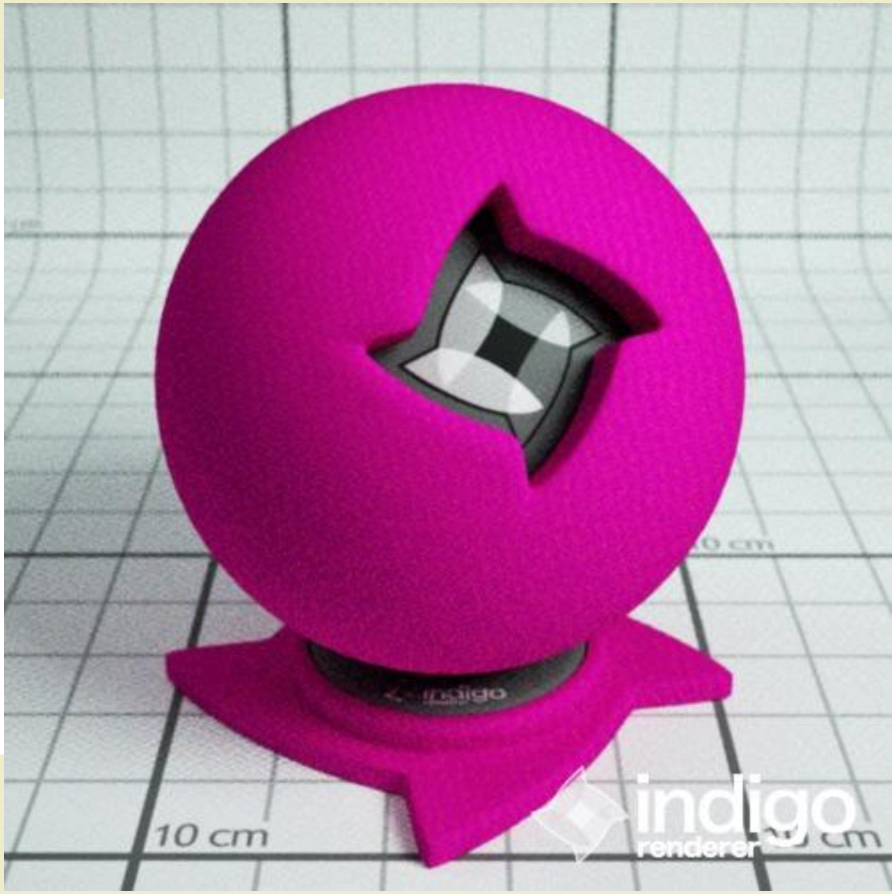
Diffuse Surface

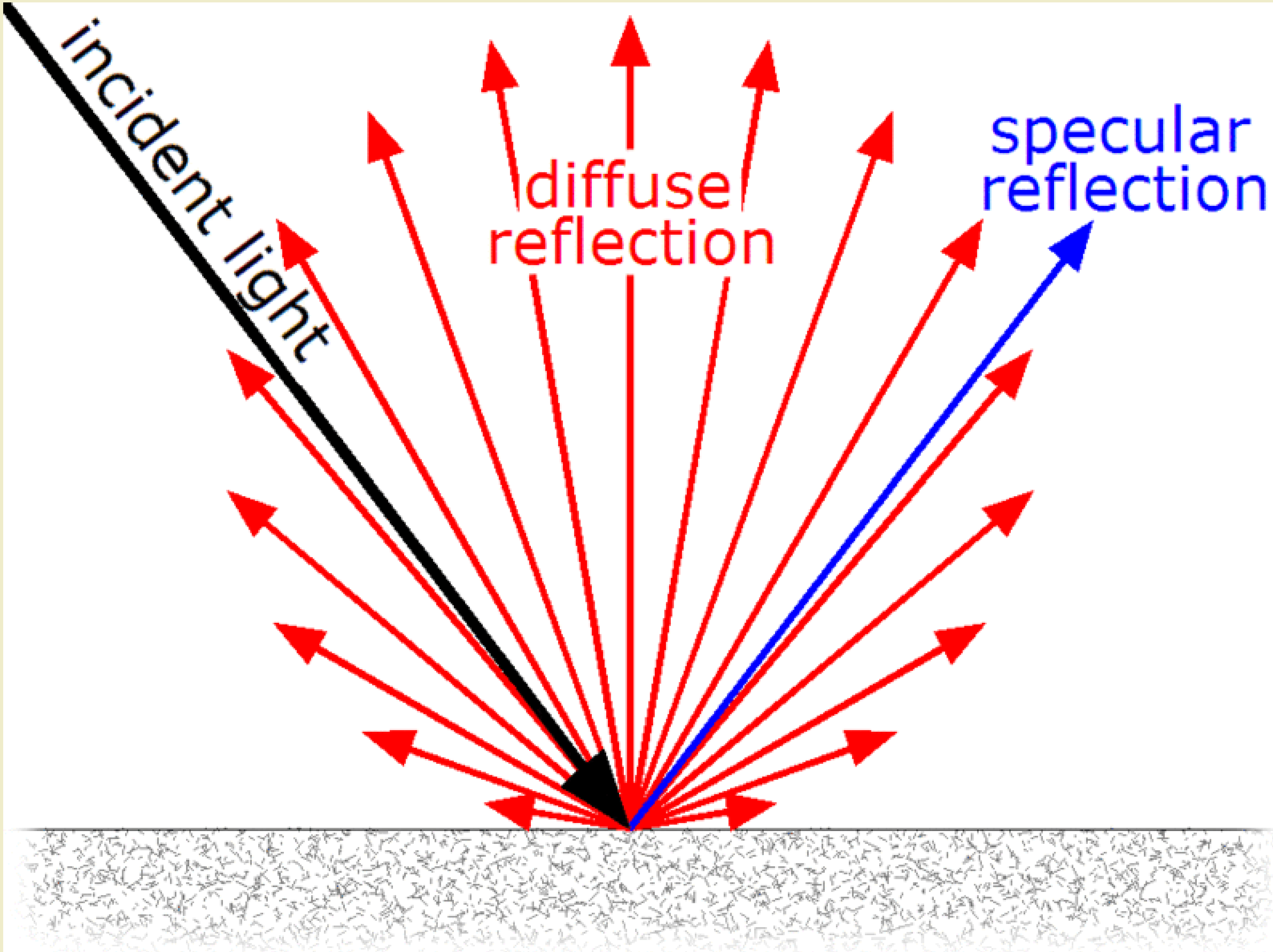
Perfectly Diffuse Surface

Materials that scatter light equally in all directions

Diffuse Surface





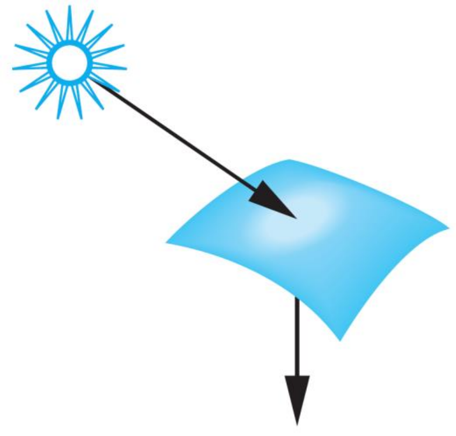

incident light
diffuse
reflection
specular
reflection

Translucent Surface

Fraction of the light penetrates the surface to emerge from another location

Refraction of light

Translucent Surface

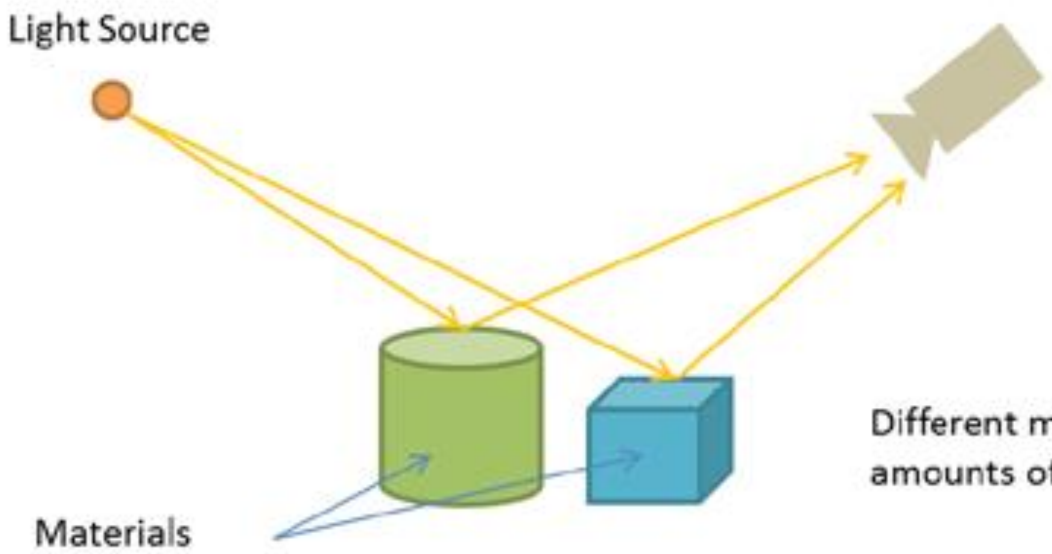


LIGHTING MODEL

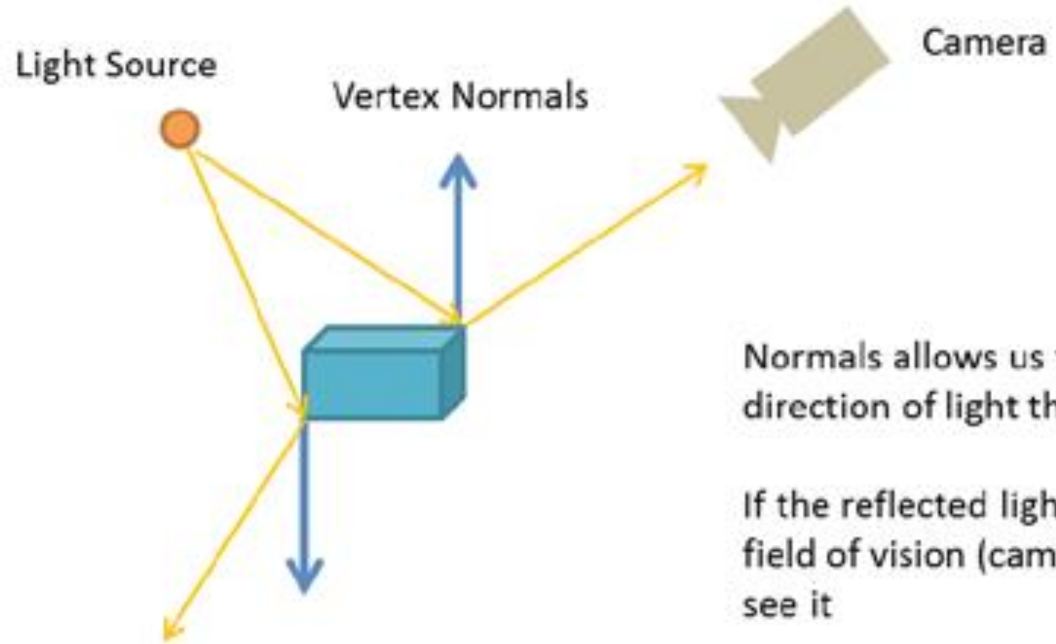
Lighting Model

A mathematical representation of how the normals, materials, and lights are combined to produce the color of the fragment

Scene Lighting



Different materials reflect different amounts of light.



Normals allows us to calculate the direction of light that is reflected.

If the reflected light lies outside of our field of vision (camera) then we will not see it

LIGHITNG MODEL: LIGHTS

Lighting Model Components: **Lights**

Ambient Light

Represented by a scalar value

Lighting Model Components: **Lights**

Point Light (Positional Light)

Represented by a point in space

Lighting Model Components: **Lights**

Directional Light

Represented by a normalized vector

Lighting Model Components: **Lights**

Color properties:

Light ambient color

Light diffuse color

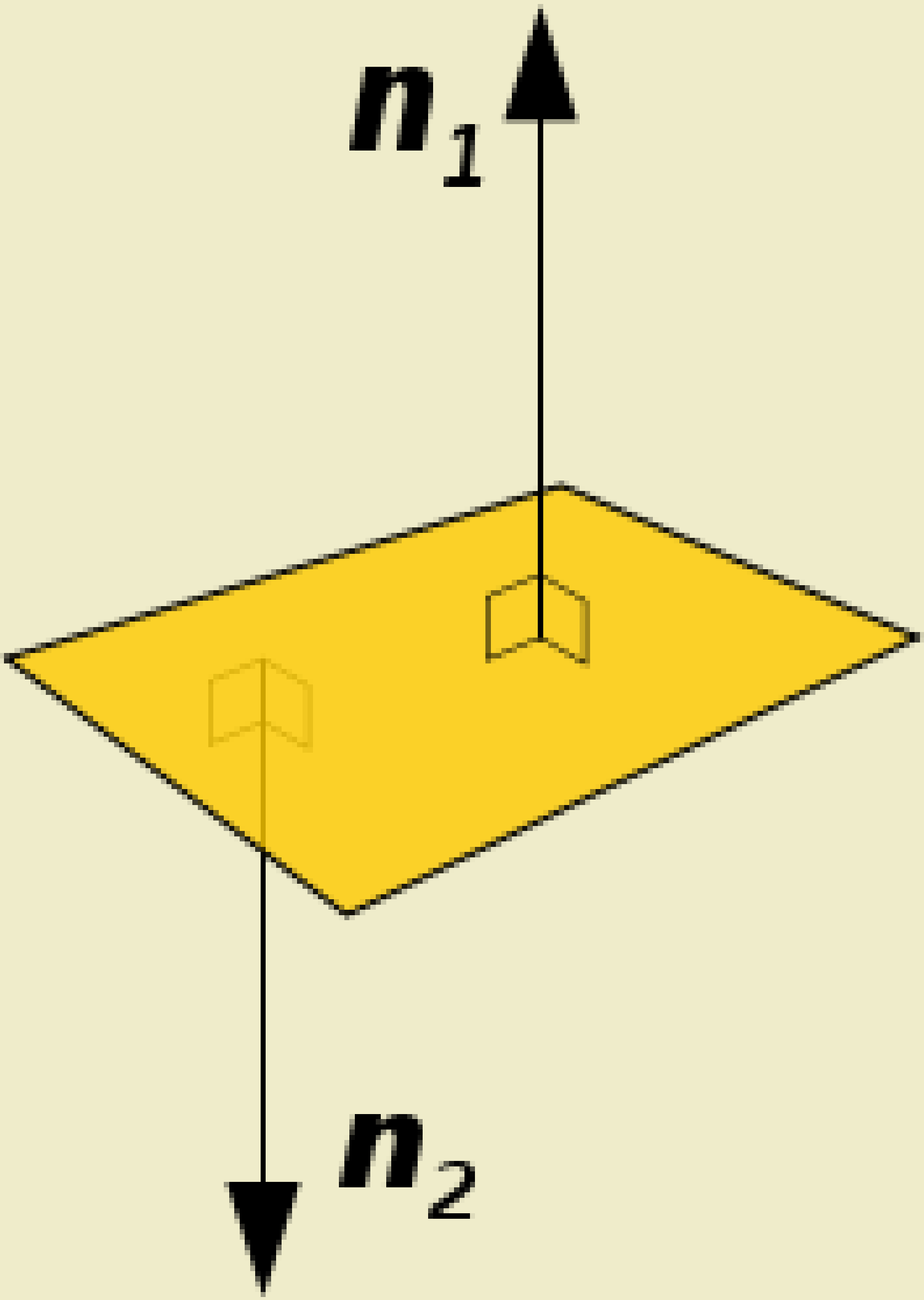
Light specular color

LIGHITNG MODEL: NORMALS

Lighting Model Components: **Normals**

Vectors perpendicular to the surface

Represent orientation of the surface

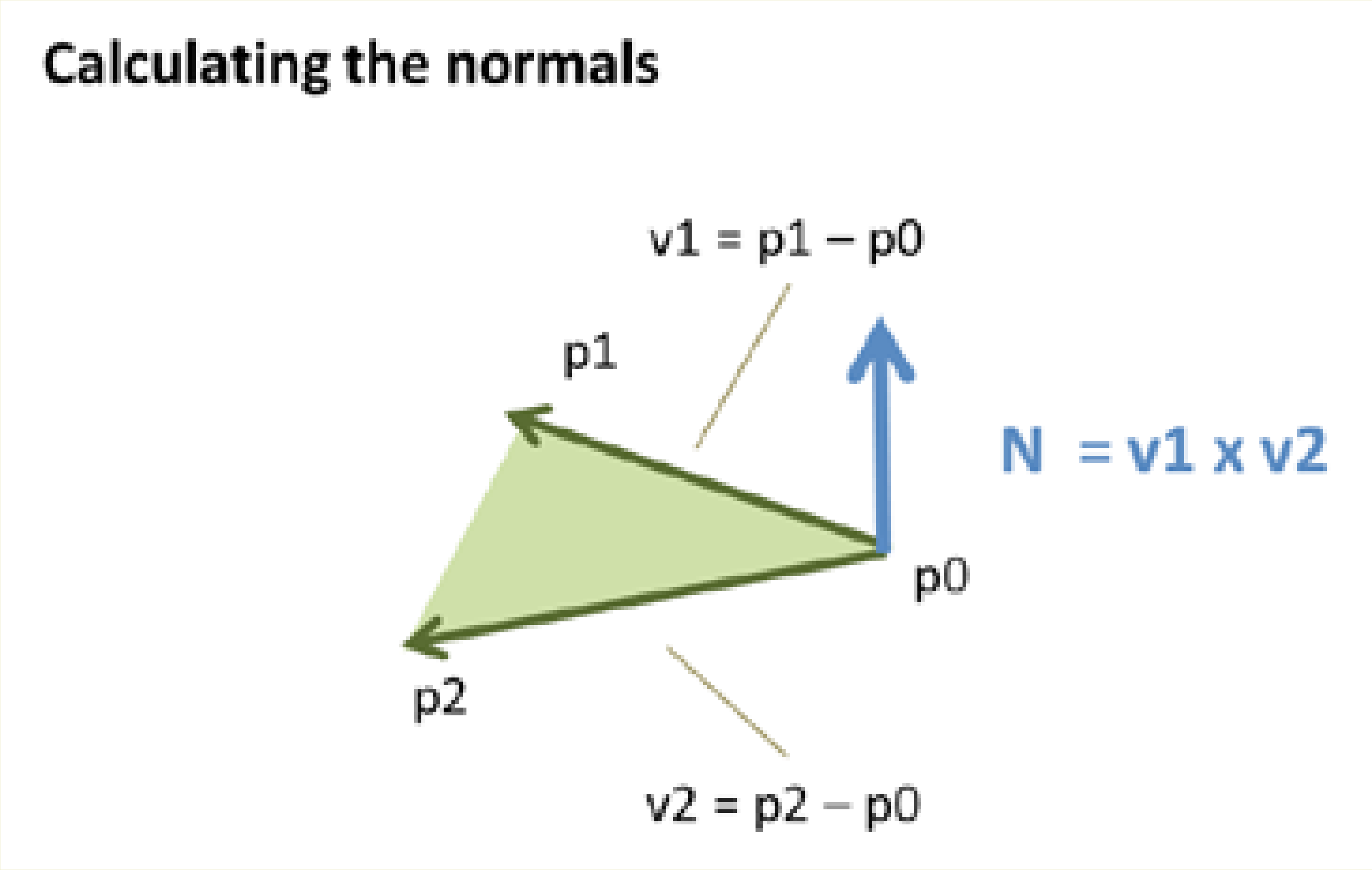

$\boldsymbol{n}_1$
$\boldsymbol{n}_2$

Lighting Model Components: **Normals**

In computer graphics

Each vertex is associated with its own normal vector

Calculating Normal Vectors



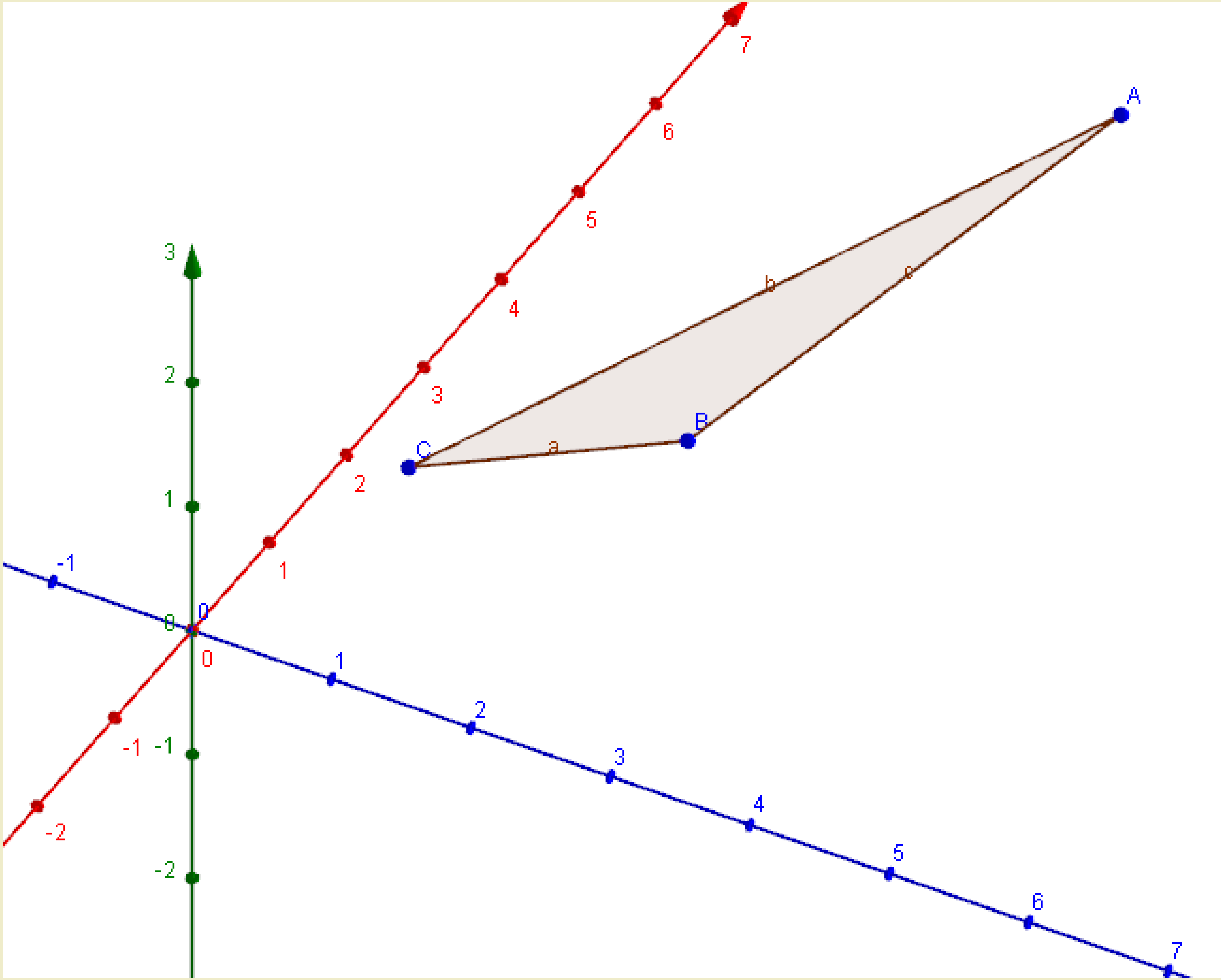
Calculating Normal Vectors

Triangle

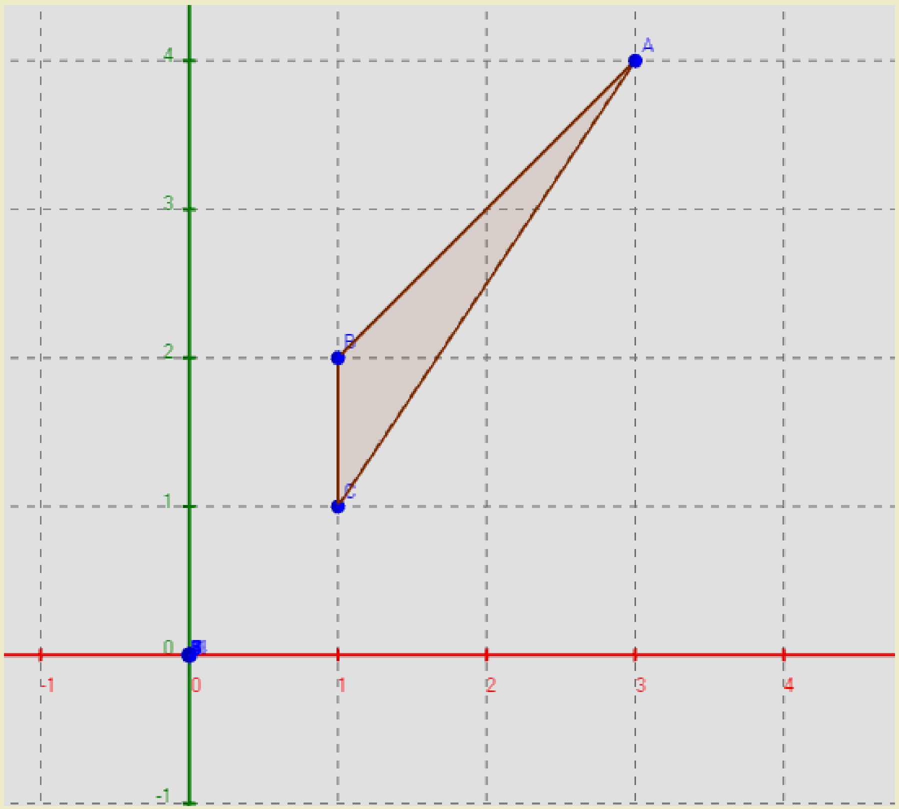
$$A = (3, 4, 5)$$

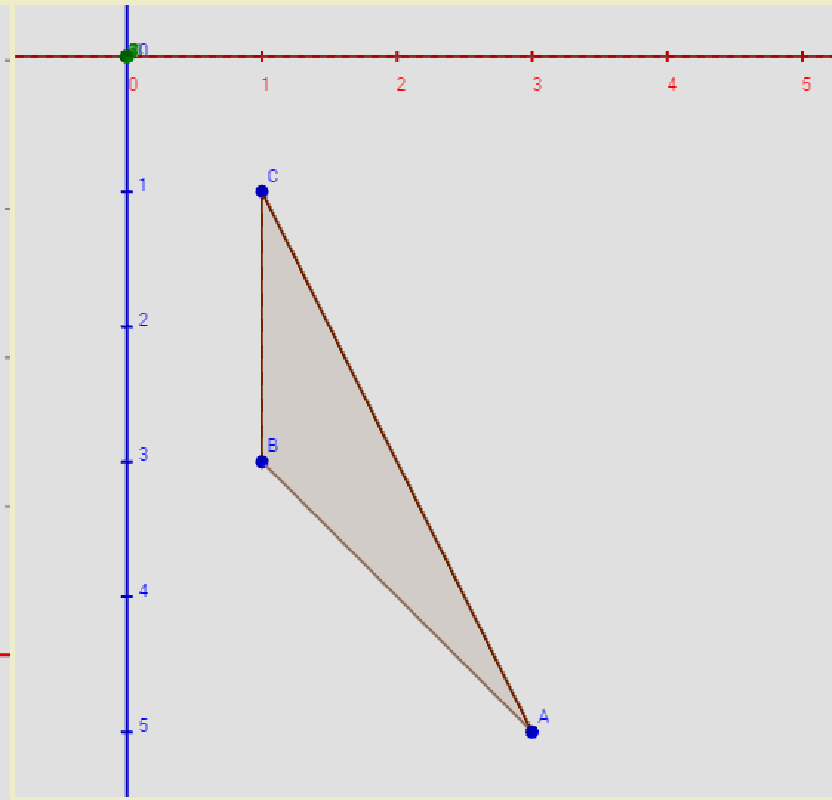
$$B = (1, 2, 3)$$

$$C = (1, 1, 1)$$



A
B
C
a
b
c


A
B
C
-1
0
1
2
3
4
-1
0
1
2
3
4


C
B
A
0
1
2
3
4
5
1
2
3
4
5

Calculating Normals

$$v_1 = B - A$$
$$v_2 = C - A$$

Calculating Normals

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 4 \\ 5 \\ 1 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 4 \\ 5 \\ 1 \end{bmatrix}$$

Calculating Normals

$$v_1 = \begin{bmatrix} -2 \\ -2 \\ -2 \\ 0 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} -2 \\ -3 \\ -4 \\ 0 \end{bmatrix}$$

Calculating Normals

$$n_1 = v_1 \times v_2$$

$$n_2 = v_2 \times v_1$$

Calculating Normals

$$n_1 = \begin{bmatrix} -2 \\ -2 \\ -2 \\ 0 \end{bmatrix} \times \begin{bmatrix} -2 \\ -3 \\ -4 \\ 0 \end{bmatrix}$$

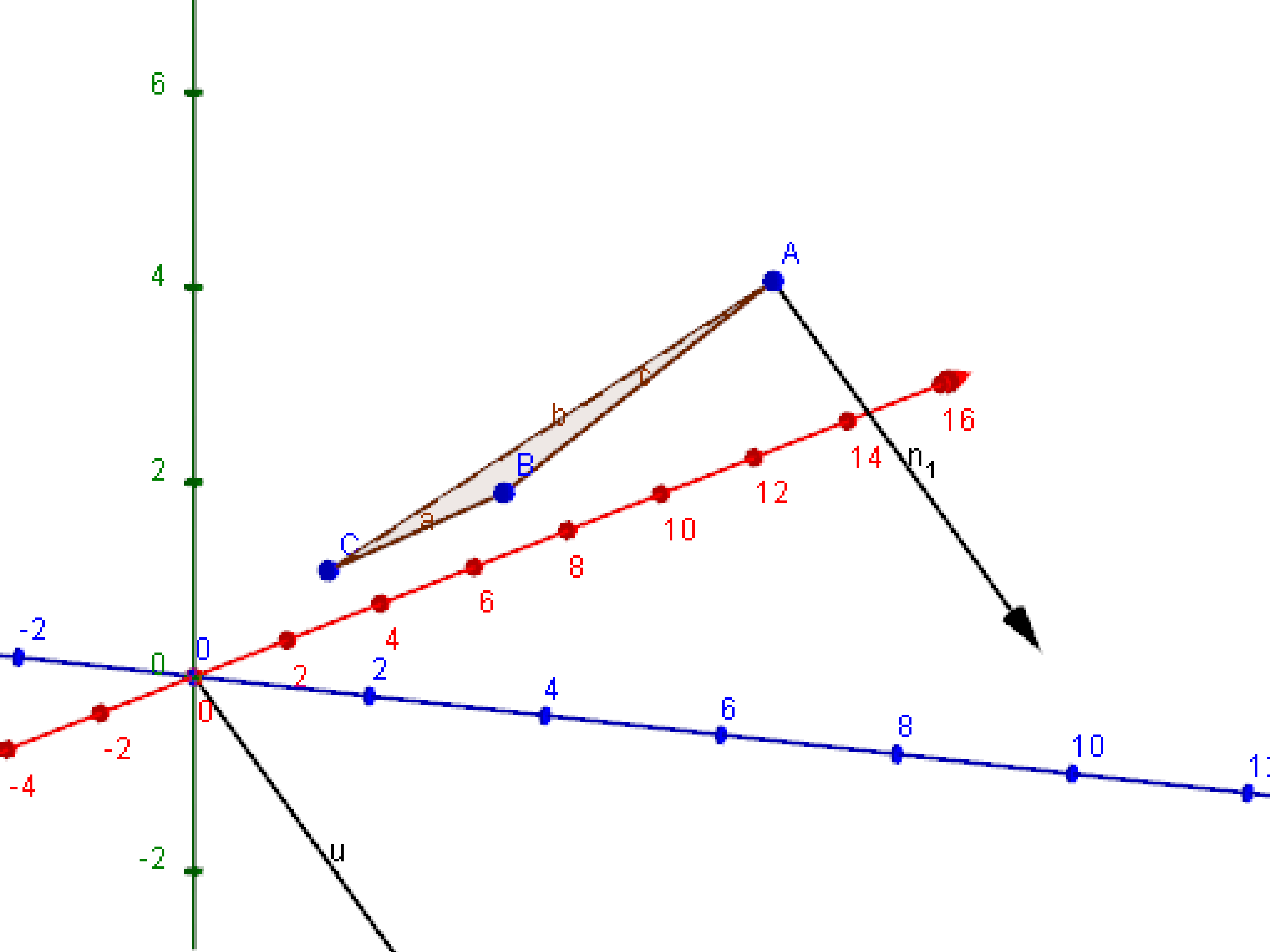
$$n_2 = \begin{bmatrix} -2 \\ -3 \\ -4 \\ 0 \end{bmatrix} \times \begin{bmatrix} -2 \\ -2 \\ -2 \\ 0 \end{bmatrix}$$

Calculating Normals

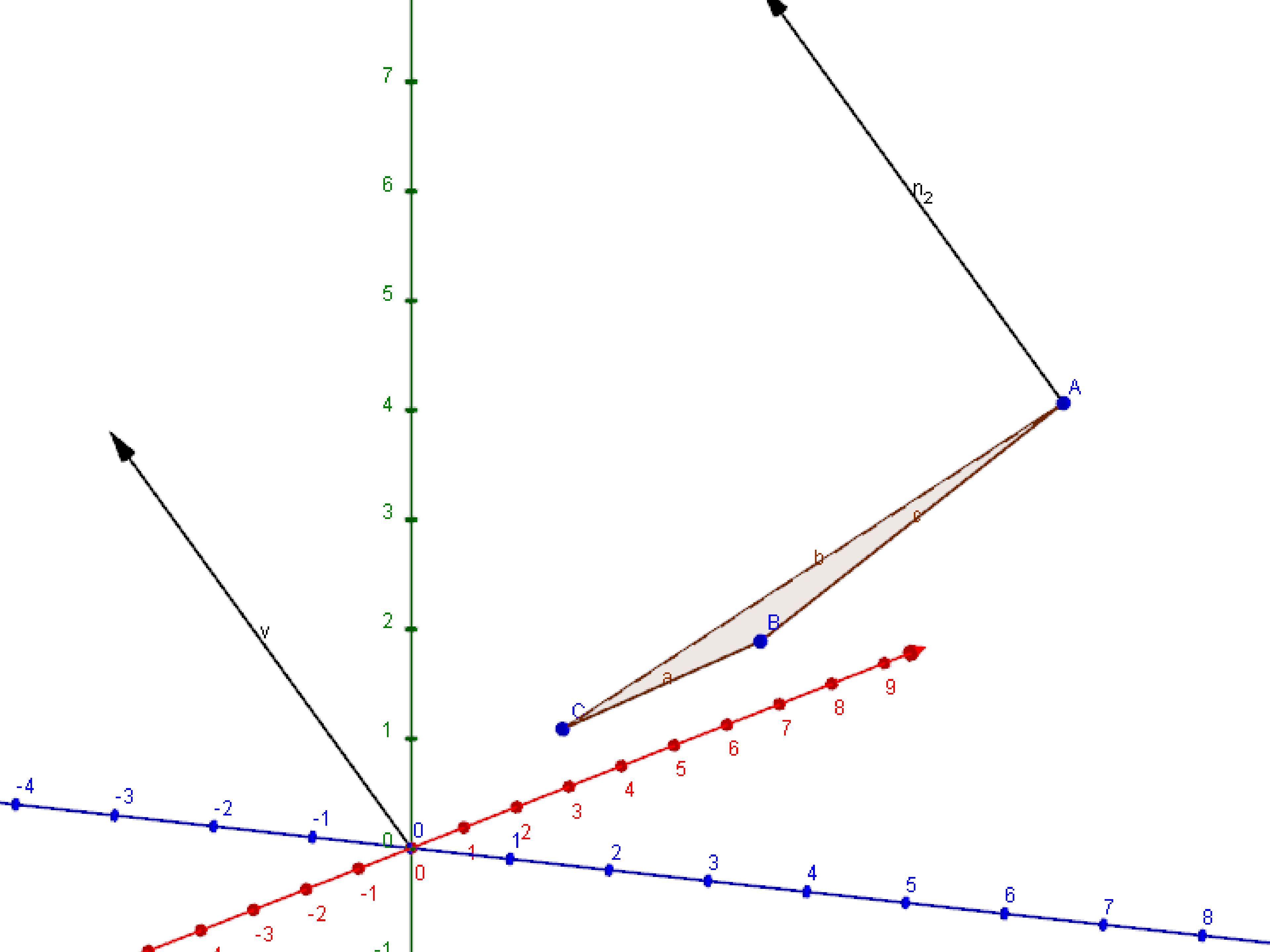
Normal vector at point A of triangle ABC

$$n_1 = < 2, -4, 2 >$$

$$n_2 = < -2, 4, -2 >$$

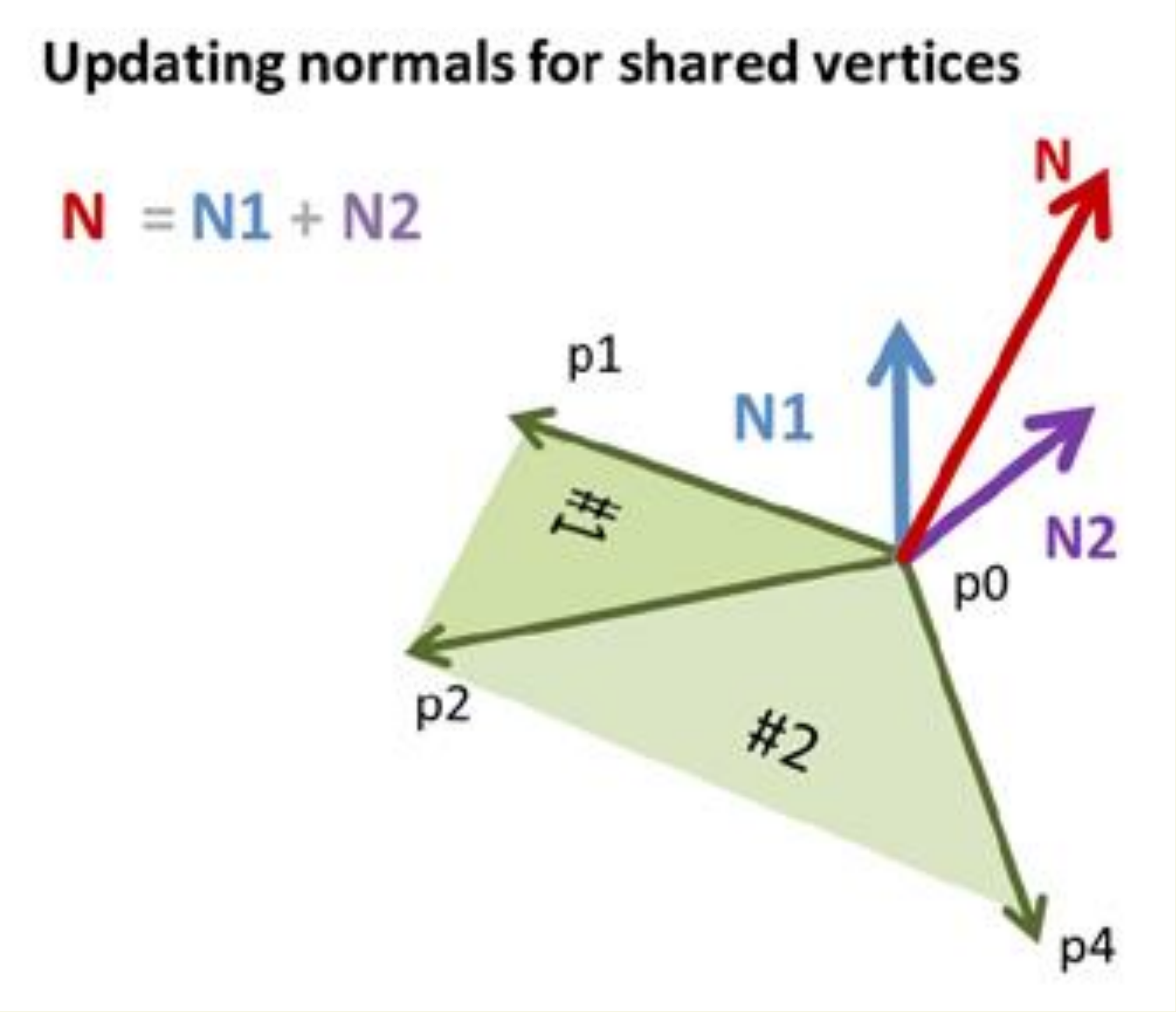


6
4
2
0
-2
A
B
C
a
b
c
n_1
u
0
-2
2
4
6
8
10
-4
-2
2
4
6
8
10
12
14
16



n_2
A
B
C
a
b
c
v

Calculating Normal Vectors



LIGHITNG MODEL: MATERIALS

Lighting Model Components: **Materials**

Colors

3-tuple: (rgb) or 4-tuple: (rgba)

Lighting Model Components: **Materials**

Textures

Images that are mapped to the surface of the object

Lighting Model Components: **Materials**

Example Brass Material

Ambient color: (0.33, 0.22, 0.03, 1.0)

Diffuse color: (0.78, 0.57, 0.11, 1.0)

Specular color:(0.99, 0.91, 0.81, 1.0)

Emission color: (0.0, 0.0, 0.0, 1.0)

Shininess factor: 27.8

LAMBERTIAN LIGHT REFLECTION MODEL

Lambertian Light Reflection Model

Lambert’s emission law

Johann Heinrich Lambert, Photometria, 1760

Lambertian Light Reflection Model

A model for diffuse reflection

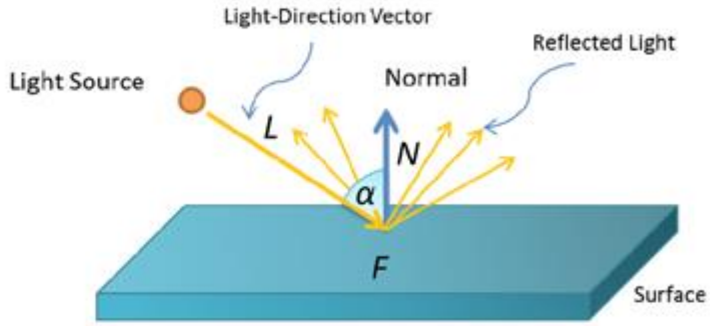
Lambertian Light Reflection Model

$$FinalDiffuseColor = LightDiffuseProperty \times MaterialDiffuseProperty \times LambertCoefficient$$

$$= L_d \times k_d \times LambertCoefficient$$

Lambertian Light Reflection Model

Lambertian Reflectance

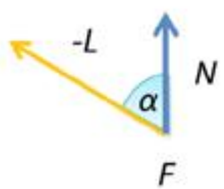


Final Diffuse Color

$$F = C_l C_m(-L \cdot N)$$

Light Diffuse Color Material Diffuse Color

Final diffuse color calculation for fragment F



$$-L \cdot N = |-L||N| \cos\alpha$$

If L and N are normalized then:

$$-L \cdot N = \cos\alpha$$

$$F = C_l C_m \cos\alpha$$

A Lambertian surface reflects light in many directions

Lambertian Light Reflection Model

$$FinalDiffuseColor = LightDiffuseProperty \times MaterialDiffuseProperty \times LambertCoefficient$$

$$= L_d \times k_d \times LambertCoefficient$$

$$= L_d \times k_d \times (-\hat{l} \cdot \hat{n})$$

Lambertian Light Reflection Model

Sample 1

Diffuse Light Color: (1.0,1.0,1.0)

Diffuse Material Color: (0.0,1.0,0.0)

Direction of light to plane: <3.0,-3.0,0.0>

Normal Vector of plane: <0.0, 2.0, 0.0>

Final Diffuse Color = ?

Lambertian Light Reflection Model

Sample 1

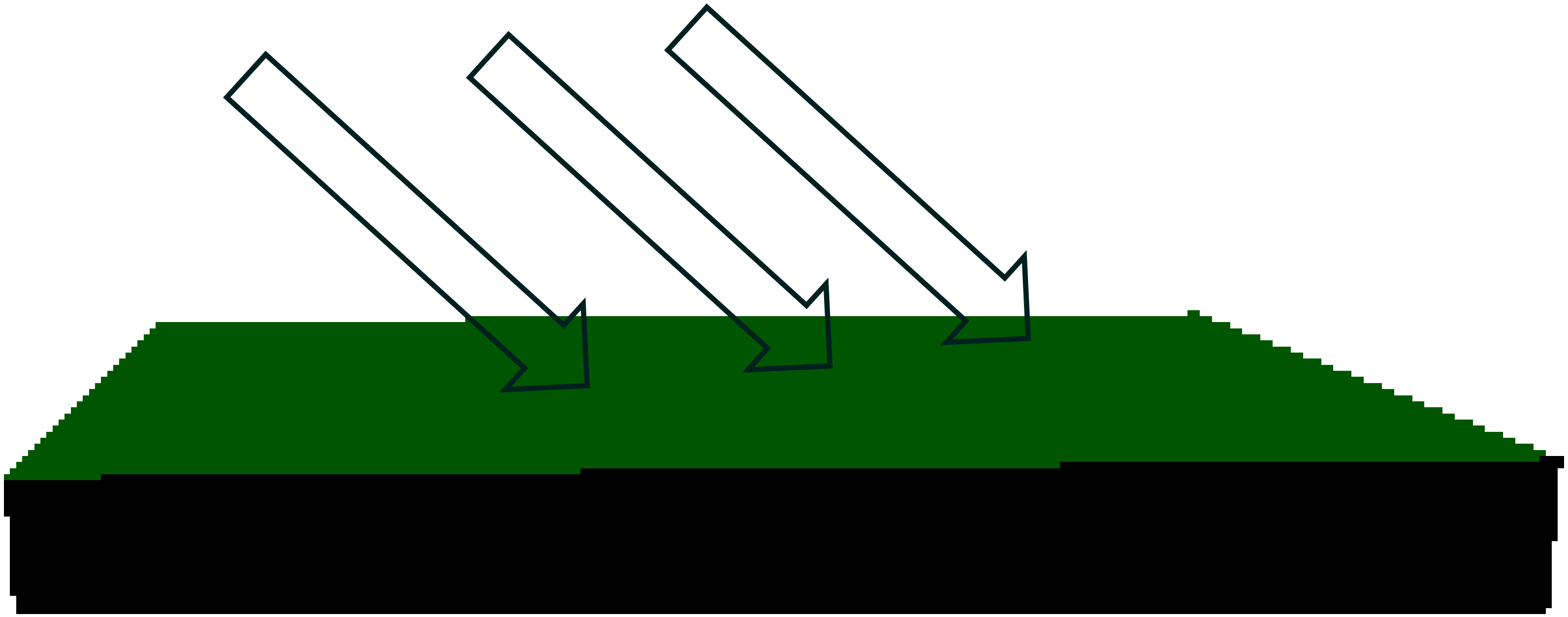
Diffuse Light Color: (1.0,1.0,1.0)

Diffuse Material Color: (0.0,1.0,0.0)

Direction of light to plane: <3.0,-3.0,0.0>

Normal Vector of plane: <0.0, 2.0, 0.0>

Final Diffuse Color = (0.0,0.71,0.0)



Lambertian Light Reflection Model

Sample 2

Diffuse Light Color: (1.0,1.0,1.0)

Diffuse Material Color: (0.0,1.0,0.0)

Direction of light to plane: <0.0,-4.0,0.0>

Normal Vector of plane: <0.0, 2.0, 0.0>

Final Diffuse Color = ?

Lambertian Light Reflection Model

Sample 2

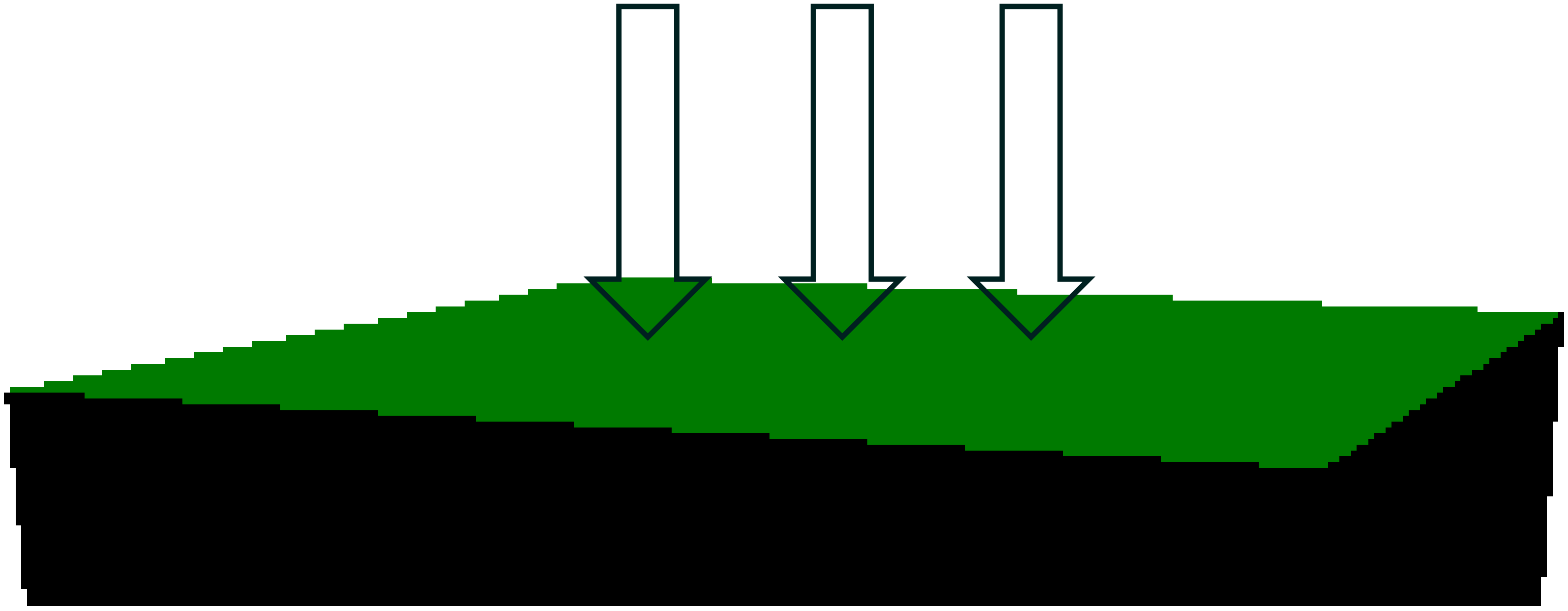
Diffuse Light Color: (1.0,1.0,1.0)

Diffuse Material Color: (0.0,1.0,0.0)

Direction of light to plane: <0.0,-4.0,0.0>

Normal Vector of plane: <0.0, 2.0, 0.0>

Final Diffuse Color = (0.0,1.0,0.0)



Lambertian Light Reflection Model

Sample 3

Diffuse Light Color: (1.0,1.0,1.0)

Diffuse Material Color: (0.0,1.0,0.0)

Direction of light to plane: <-2.0,4.0,0.0>

Normal Vector of plane: <0.0, 2.0, 0.0>

Final Diffuse Color = ?

Lambertian Light Reflection Model

Sample 3

Diffuse Light Color: (1.0,1.0,1.0)

Diffuse Material Color: (0.0,1.0,0.0)

Direction of light to plane: <2.0,4.0,0.0>

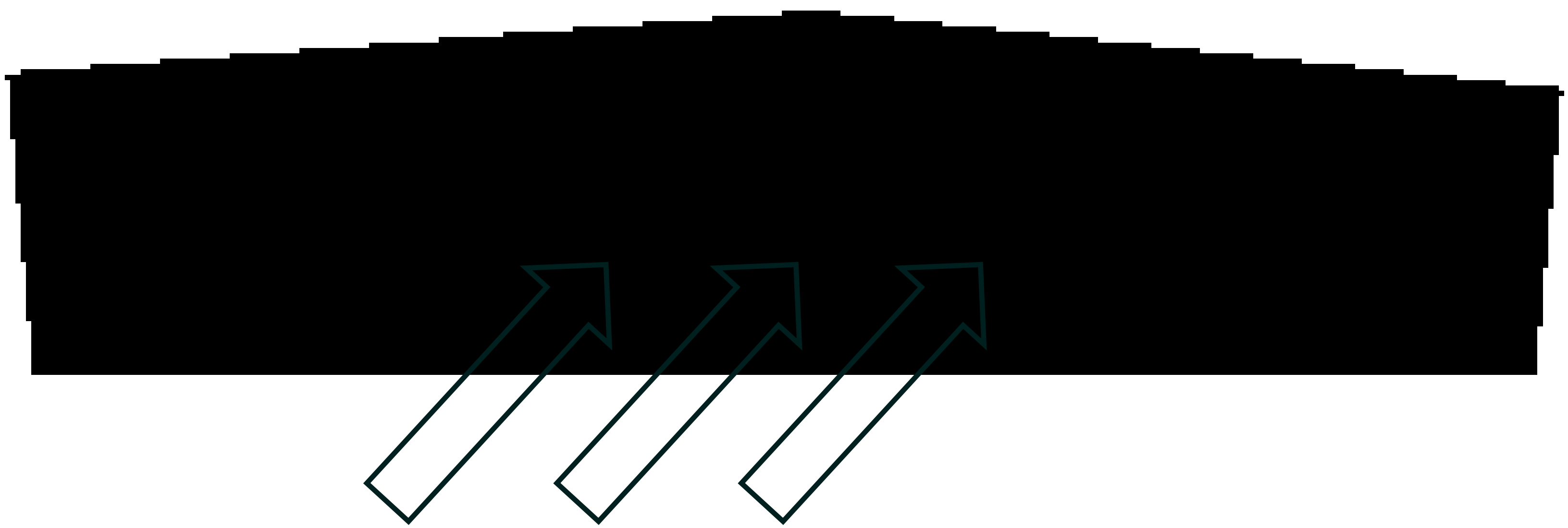
Normal Vector of plane: <0.0, 2.0, 0.0>

Final Diffuse Color = (0.0, 0.0, 0.0)

Lambertian Light Reflection Model

Negative Lambert Coefficient is automatically converted to 0

Color components (r,g,b) cannot be negative



PHONG LIGHT REFLECTION MODEL

Phong Light Reflection Model

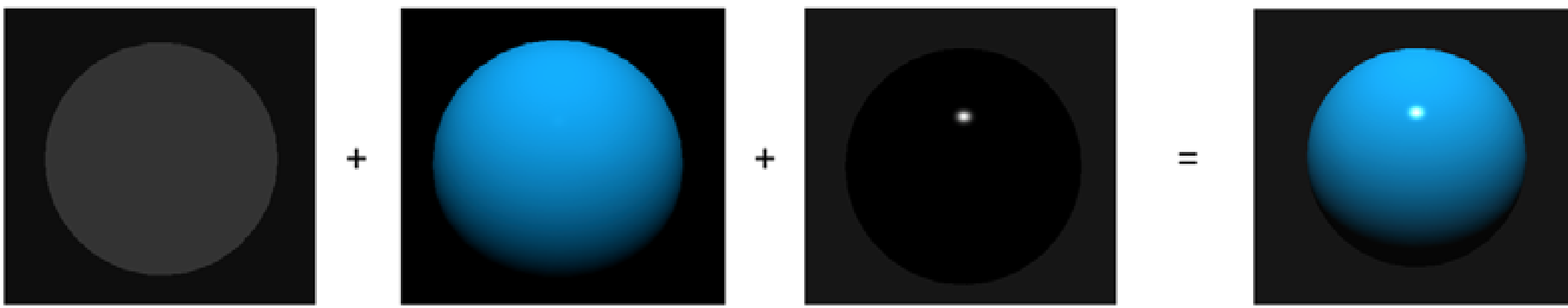
Bui Tuong Phong, Ph.D. dissertation, 1973

Describes reflection of light as three components

Phong Reflection Model

Reflected color is the result of combining three types of light-object interactions:

Ambient + Diffuse + Specular = Phong Reflection



Ambient: Amount of light present *everywhere* in the scene. Independent from any light source

Diffuse: The incident light is reflected in *many directions*. It can be modelled by a Lambertian surface.

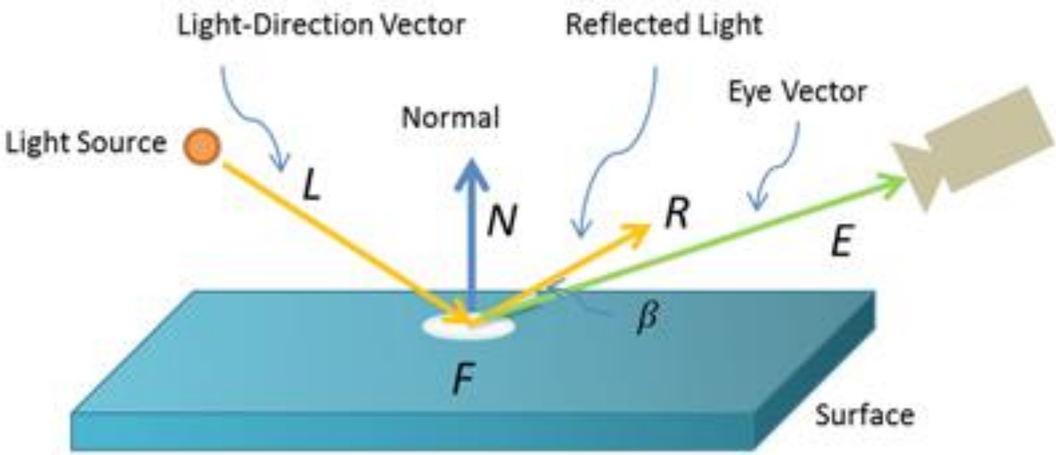
Specular: Mirror-like reflection. The direction of the incoming light and the direction of the reflected outgoing light make *the same angle with respect to the surface normal*.

Phong Light Reflection Model

$$Intensity = AmbientIntesity + DiffuseIntesity + SpecularIntesity$$

$$I = I_a + I_d + I_s$$

Specular Reflection



Final Specular Color

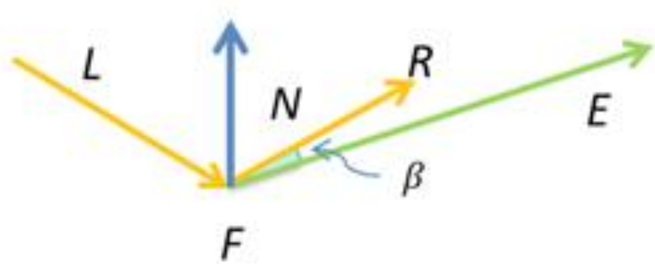
Material Shininess

$$F_s = C_l C_m (R \cdot E)^n$$

Light Specular Color

Material Specular Color

Final specular color calculation for fragment F



$$R \cdot E = |R||E| \cos\beta$$

If R and E are normalized then:

$$R \cdot E = \cos\beta$$

$$F = C_l C_m \cos^n \beta$$

The specular reflection reaches its maximum when *R* and E have the same direction.

Phong Light Reflection Model: Ambient Intensity

$$Ia = L_a \times k_a$$

Phong Light Reflection Model: Diffuse Intensity

(using lambert model)

$$I_d = L_d \times k_d \times (n \cdot -l)$$

Phong Light Reflection Model: Specular Intensity

(using phong model)

$$I_s = L_s \times k_s \times specularCoefficient$$

Specular Coefficient Computation

$$I_s = L_s \times k_s \times (\hat{r} \cdot \hat{e})^{\alpha}$$

r is the reflection of light vector

e is the surface to eye vector

α is the material shininess coefficient

Specular Coefficient Computation

$$I_s = L_s \times k_s \times (\hat{r} \cdot \hat{e})^{\alpha}$$

$$\hat{r} = 2\left(-\hat{l} \cdot \hat{n}\right)\hat{n} - (-\hat{l})$$

α is the material shininess coefficient

Phong Light Reflection Model

$$I = I_a + I_d + I_s$$

$$I = [(L_a \times k_a)] + [L_d \times k_d \times \left(\hat{n} \cdot -\hat{l}\right)] + [L_s \times k_s \times (\hat{r} \cdot \hat{e})^{\alpha}]$$

PHONG LIGHT REFLECTION MODEL

SAMPLE 1

Sample 1 – Phong LRM

Light Color Specifications:

– Ambient Light Color (L_a): **(0.1, 0.1, 0.1)**

– Diffuse Light Color (L_d): (1.0, 1.0, 1.0)

– Specular Light Color (L_s): (1.0, 1.0, 1.0)

Sample 1 – Phong LRM

Material Color Specifications :

– Ambient Material Color (K_a): (0.0, 1.0, 0.0)

– Diffuse Material Color (K_d): (0.0, 1.0, 0.0)

– Specular Material Color (K_s): (0.82,1.0, 0.82)

Sample 1 – Phong LRM

Vector and Other Specifications :

– Direction of light to plane (l):	<3.0,-3.0, 0.0>
– Normal Vector of plane (n):	<0.0, 2.0, 0.0>
– **Location of Eye/Camera ($\boldsymbol{E}$):**	(4.0, 5.0, 3.0)
– Location of Vertex (F):	(1.0, 2.0, 3.0)
– Material Shininess (α):	27.0

Sample 1 – Phong LRM

Final Colors:

– Intensity of Ambient Color (I_a) = ?

– Intensity of Diffuse Color (I_d) = ?

– Intensity of Specular Color (I_s) = ?

– Intensity of Final Color (I) = ?

Sample 1 – Phong LRM

Final Colors:

- Intensity of Ambient Color (I_a) = (0.0, 0.1, 0.0)
- Intensity of Diffuse Color (I_d) = (0.0, 0.71, 0.0)
- Intensity of Specular Color (I_s) = (0.82, 1.0, 0.82)

- Intensity of Final Color (I) = (0.82, 1.81, 0.82)

PHONG LIGHT REFLECTION MODEL

SAMPLE 2

Sample 2 – Phong LRM

Light Color Specifications:

– Ambient Light Color (L_a): **(0.1, 0.1, 0.1)**

– Diffuse Light Color (L_d): (1.0, 1.0, 1.0)

– Specular Light Color (L_s): (1.0, 1.0, 1.0)

Sample 2 – Phong LRM

Material Color Specifications :

– Ambient Material Color (K_a): (0.0, 1.0, 0.0)

– Diffuse Material Color (K_d): (0.0, 1.0, 0.0)

– Specular Material Color (K_s): (0.82,1.0, 0.82)

Sample 2 – Phong LRM

Vector and Other Specifications :

– Direction of light to plane (l):	<3.0,-3.0, 0.0>
– Normal Vector of plane (n):	<0.0, 2.0, 0.0>
– **Location of Eye/Camera** ($\boldsymbol{E}$):	**(3.0, 5.0, 2.0)**
– Location of Vertex (F):	(1.0, 2.0, 3.0)
– Material Shininess (α):	27.0

Sample 2 – Phong LRM

Final Colors:

– Intensity of Ambient Color (I_a) = ?

– Intensity of Diffuse Color (I_d) = ?

– Intensity of Specular Color (I_s) = ?

– Intensity of Final Color (I) = ?

Sample 2 – Phong LRM

Final Colors:

- Intensity of Ambient Color (I_a) = (0.0, 0.1, 0.0)
- Intensity of Diffuse Color (I_d) = (0.0, 0.71, 0.0)
- *Intensity of Specular Color* (I_s) = *(0.18, 0.22, 0.18)*

- Intensity of Final Color (I) = (0.18, 0.93, 0.18)

SHADING

Shading

Shading refers to the type of interpolation that is performed to obtain the final color for every pixel in our object

Shading

Commonly mistaken as synonymous to lighting

Shading Methods

Flat Shading

Shading Methods

Smooth Shading

Gouraud Shading OR Phong Shading

Shading Methods

Flat Shading (per-polygon)

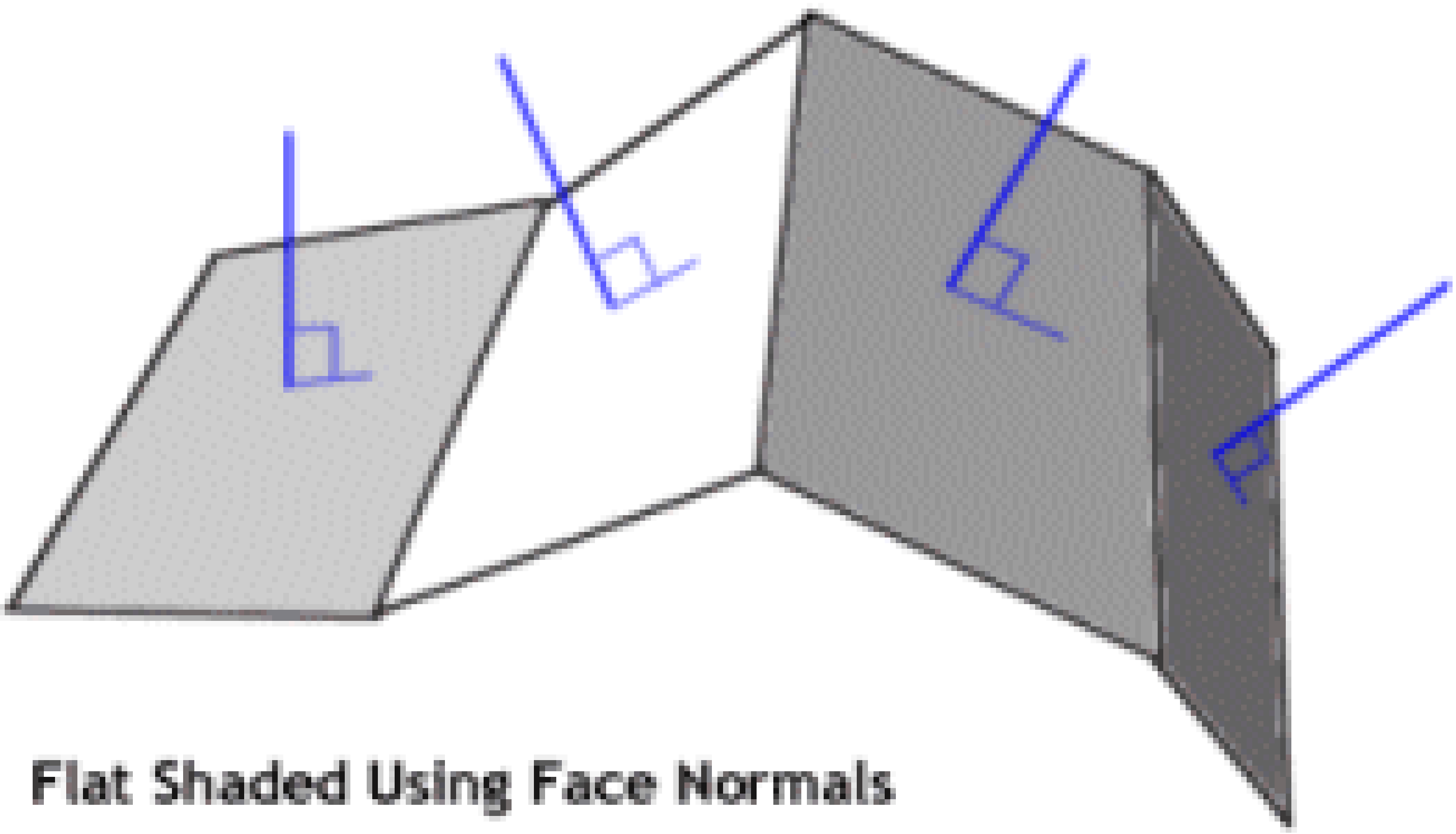
Shading Methods

Smooth Shading

Gouraud Shading (per-vertex)

Phong Shading (per-fragment)

FLAT SHADING



Flat Shaded Using Face Normals

Flat Shading

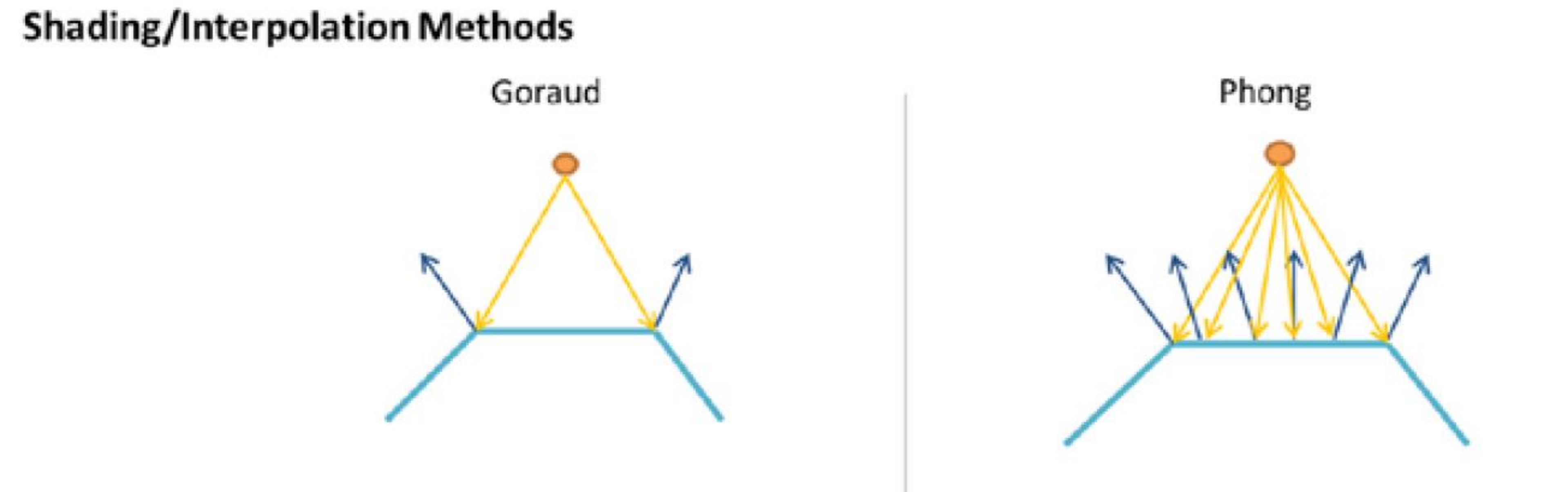
The normal vector does not vary at any point in the surface

Compute the color of a pixel in the surface and apply that on every pixel of the surface

“Per-polygon”

SMOOTH SHADING

Smooth Shading

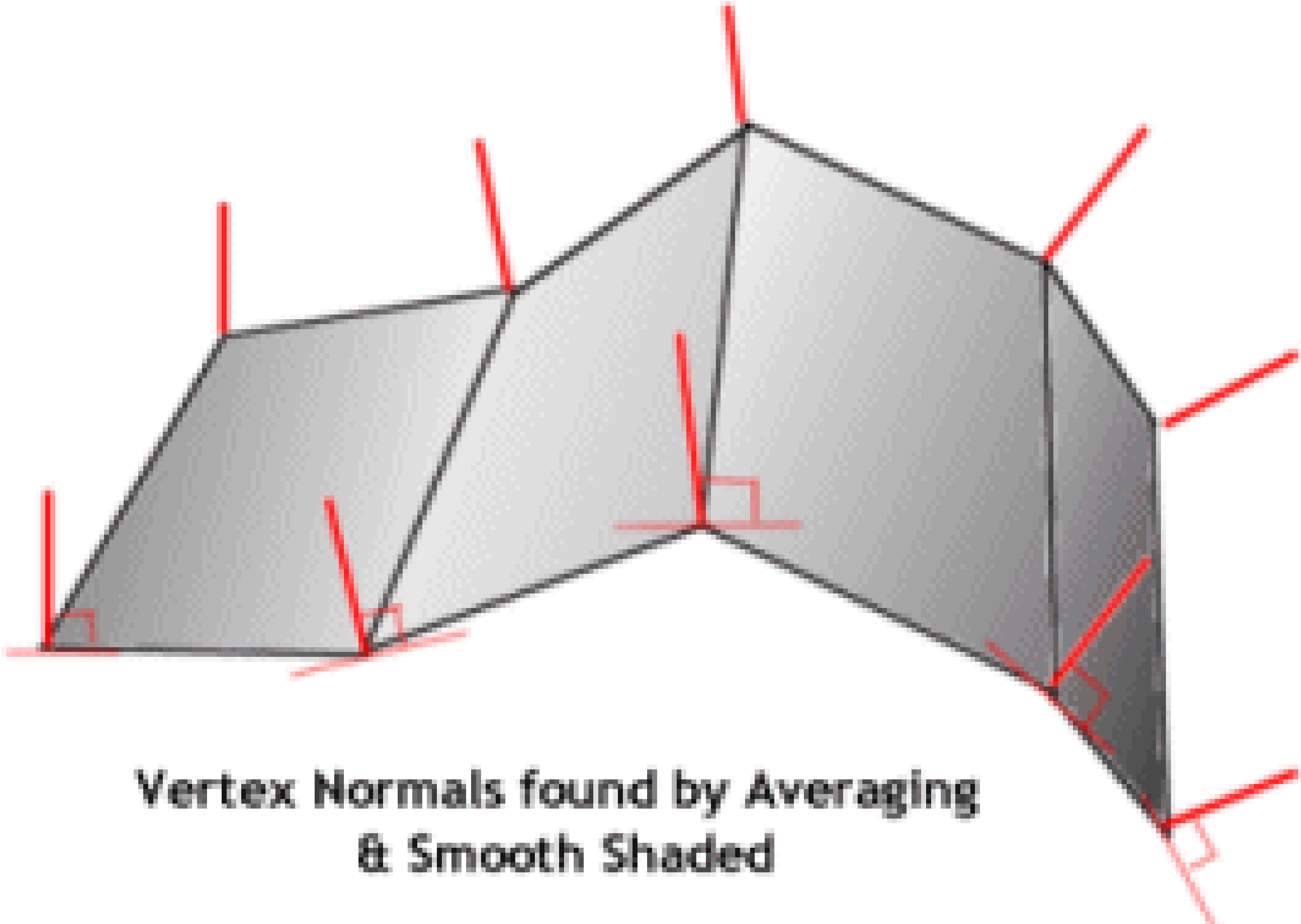


Gouraud Shading

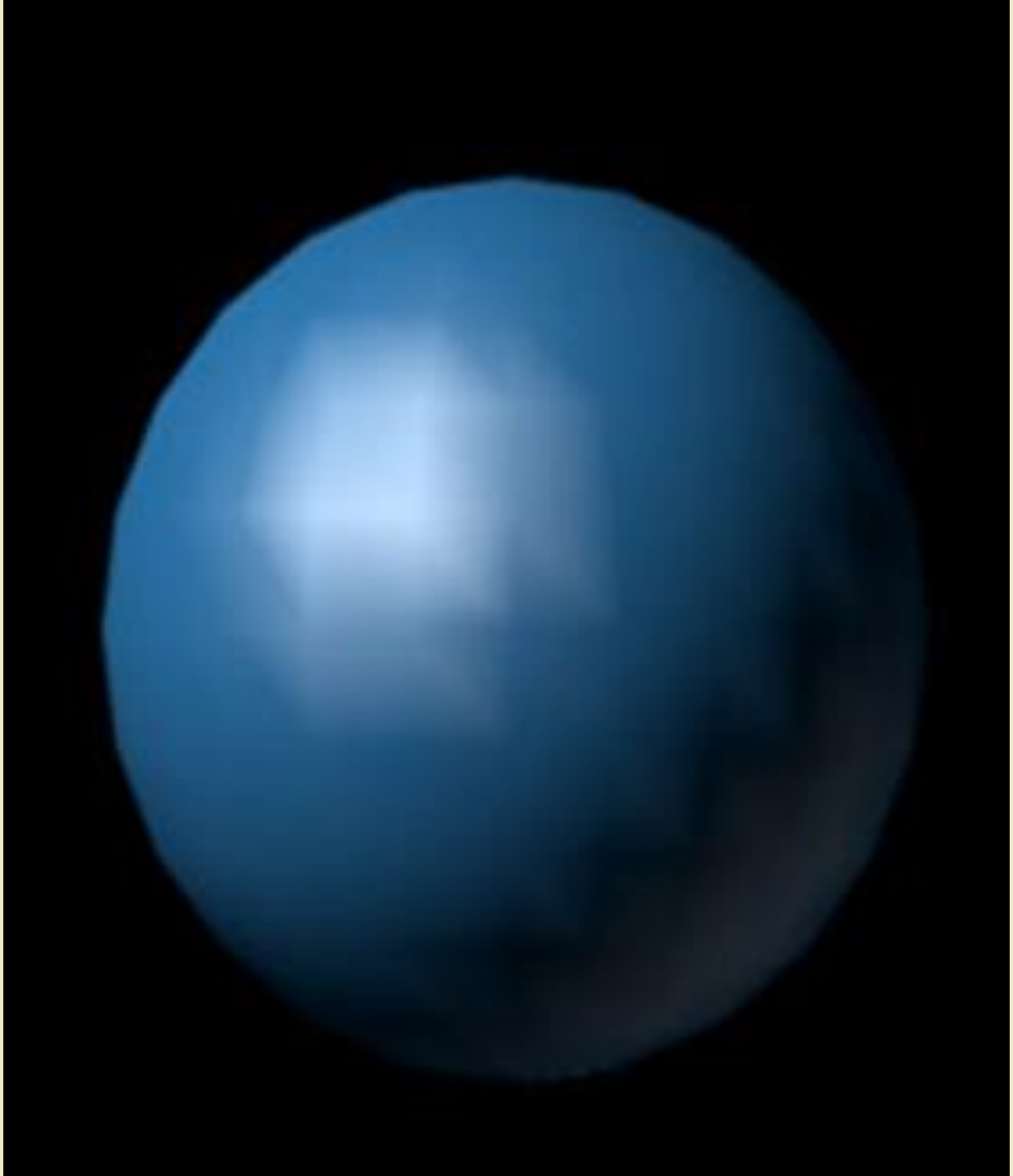
Henri Gouraud,1971

Computes color at each vertex then
Bi-linearly **interpolate** color for each interior pixel

“Per-vertex”



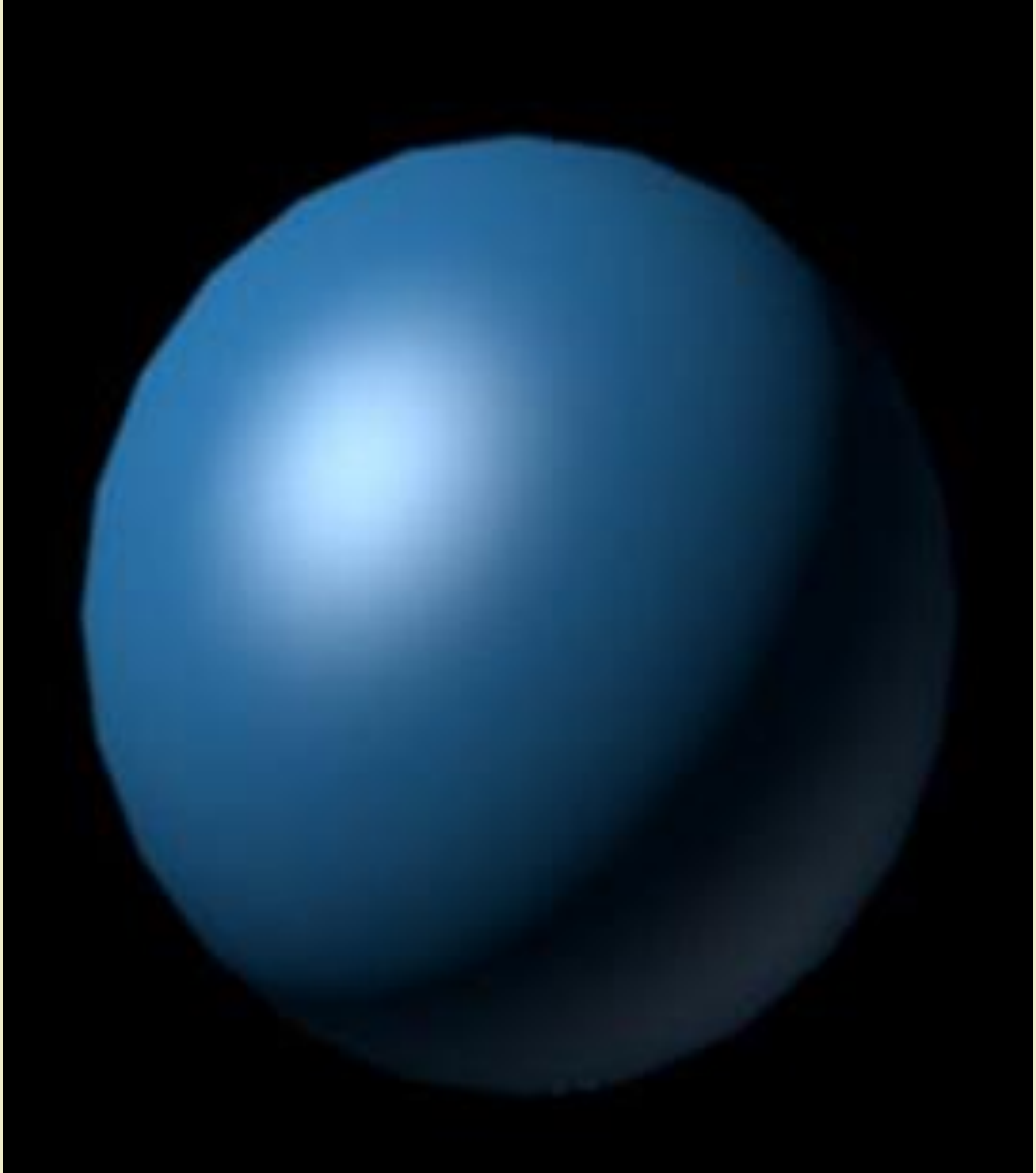
Vertex Normals found by Averaging
& Smooth Shaded



Phong Shading

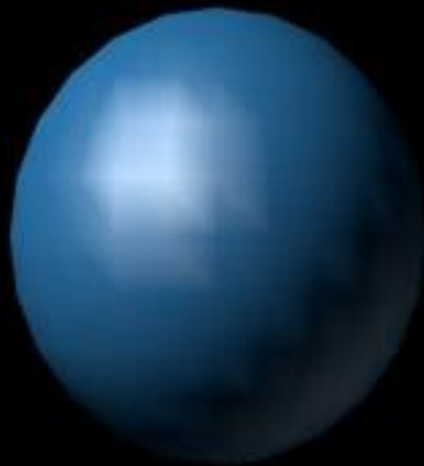
Applies lighting computation per-pixel.

Interpolation of normal vectors, rather than colors

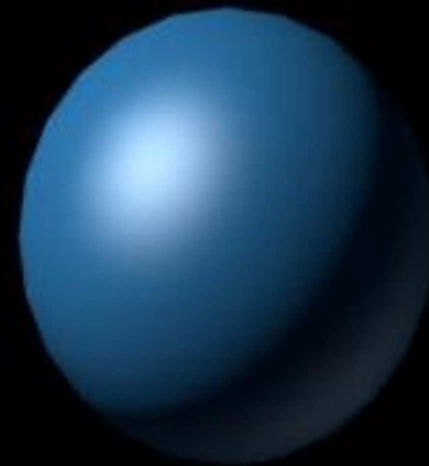


Flat

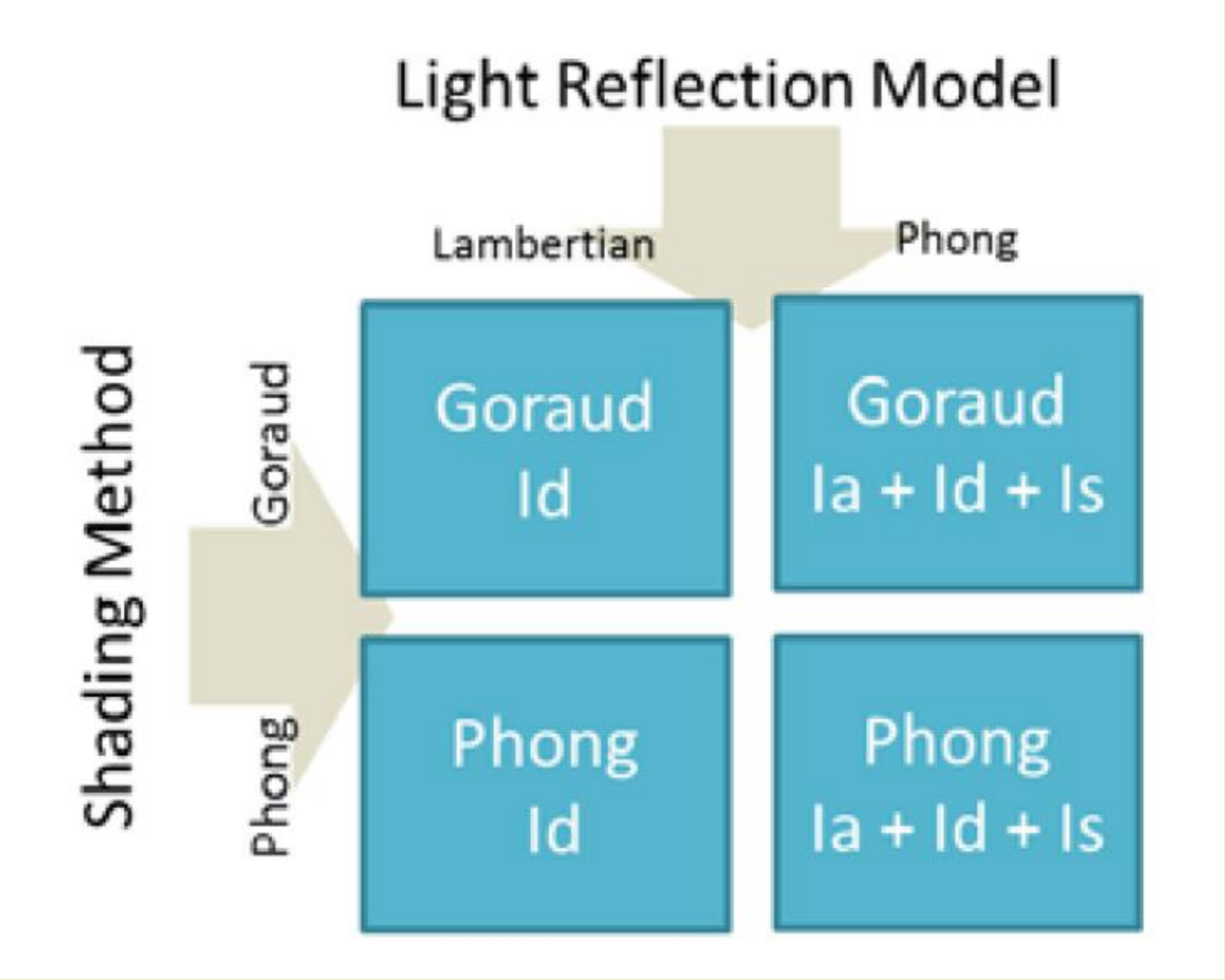
Gouraud



Phong



In Practice


Light Reflection Model
Lambertian
Phong
Shading Method
Goraud
Phong
Goraud
Id
Goraud
Ia + Id + Is
Phong
Id
Phong
Ia + Id + Is

ADVANCED LIGHTING CONCEPTS

Blinn-Phong Light Reflection Model

Variant of Phong Lighting Model

Original Phong Specular Coefficient

$$I_s = L_s \times k_s \times (\boldsymbol{r} \cdot \boldsymbol{e})^{\boldsymbol{\alpha}},$$

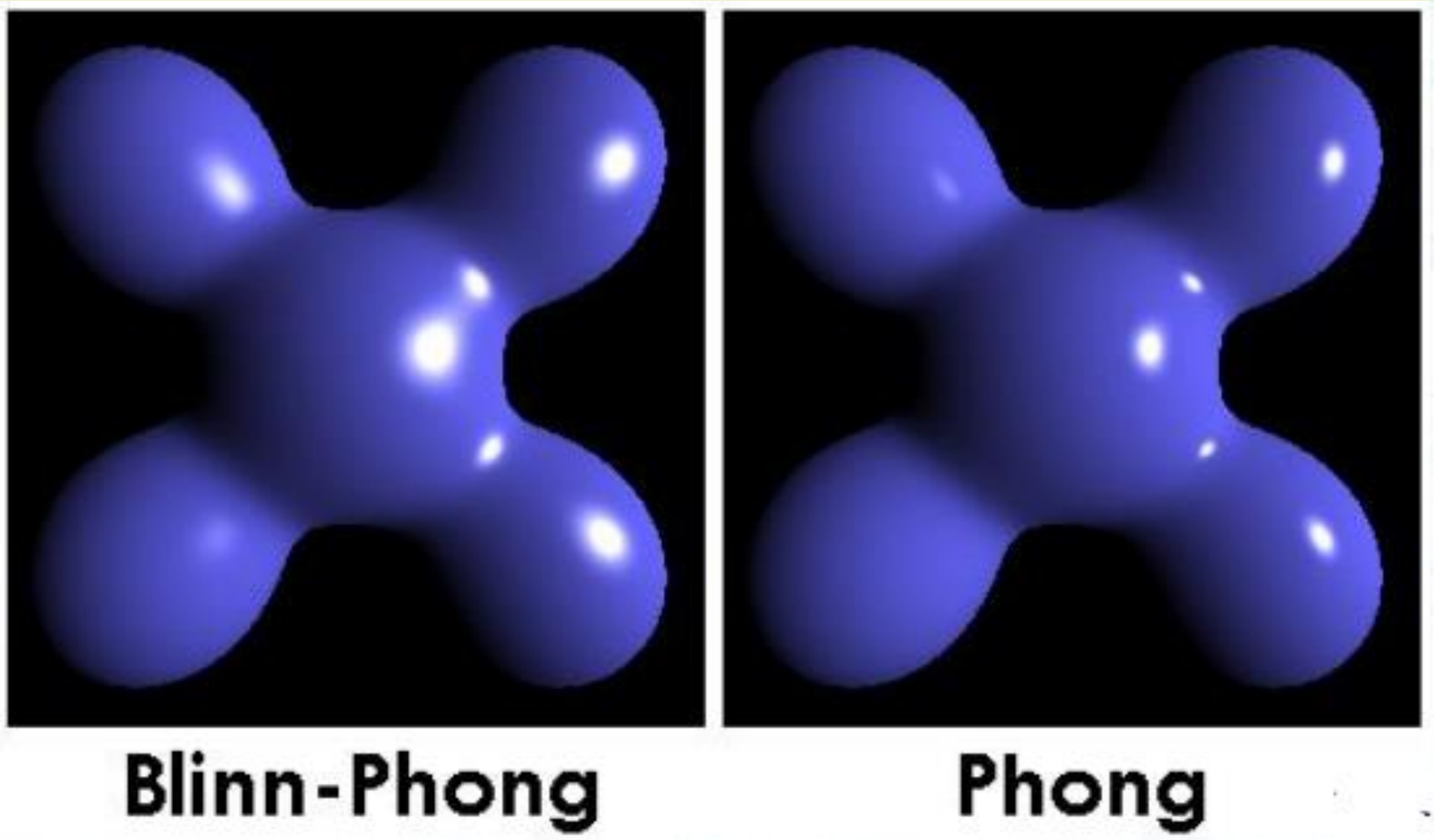
$$where\ r = 2(-l \cdot n)n - (-l)$$

Blinn-Phong Light Reflection Model

Blinn-Phong Specular Coefficient

$$I_s = L_s \times k_s \times (\boldsymbol{n} \cdot \boldsymbol{h})^{\alpha},$$

$$where\ \boldsymbol{h} = (\boldsymbol{l} + \boldsymbol{e})/|\boldsymbol{l} + \boldsymbol{e}|$$


Blinn-Phong
Phong

Multiple Lights

Sum all computed intensity from n light sources

More light sources means more computation

$$FinalColor = \sum_{i=0}^{n} I_i$$

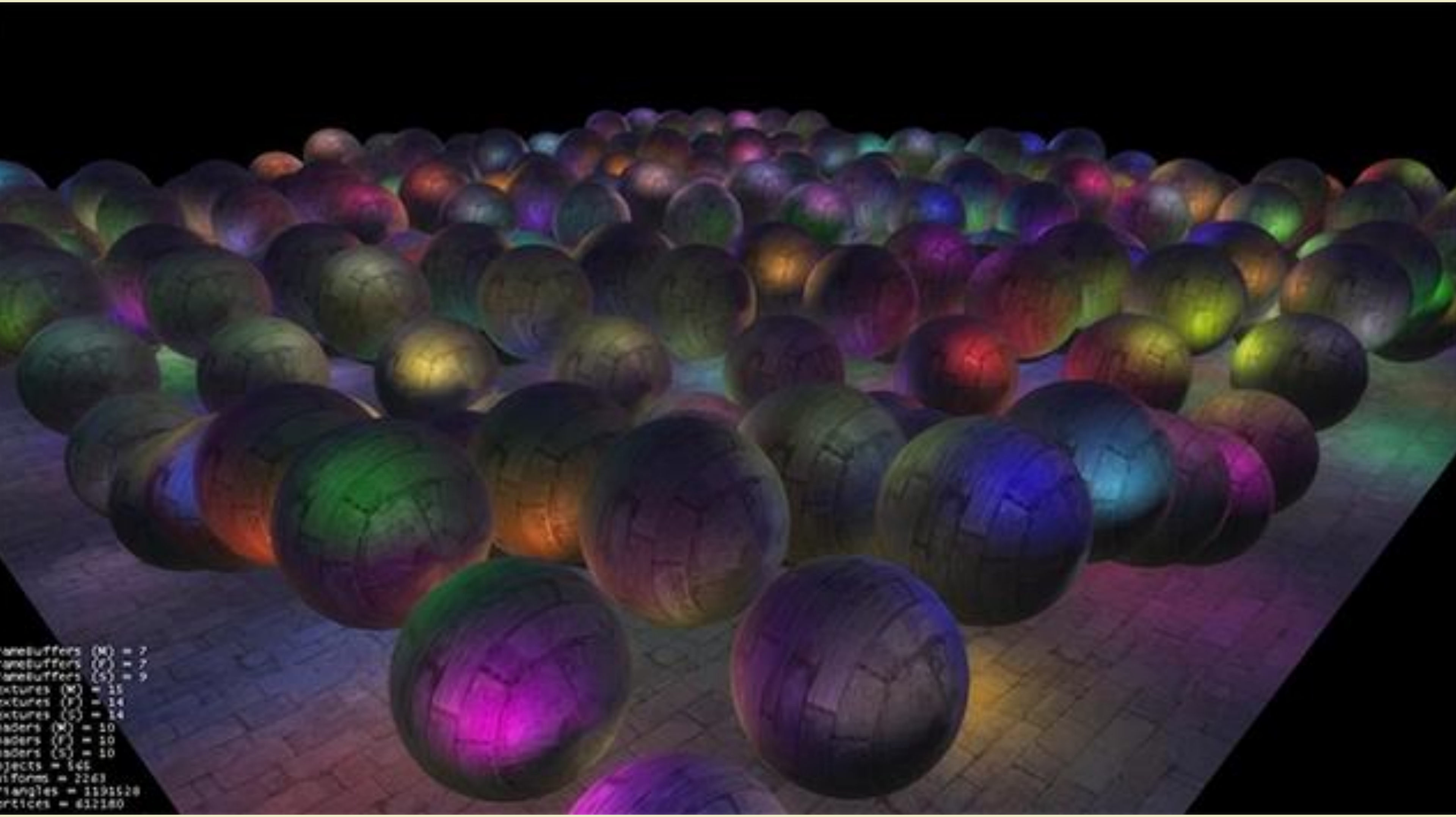
Multiple Lights Example: Game

Deferred Rendering

A method improving multiple light computation

Light computations are removed from the vertex shader and fragment shader at first shading pass *and is deferred until the second pass*

Deferred Rendering



Fog

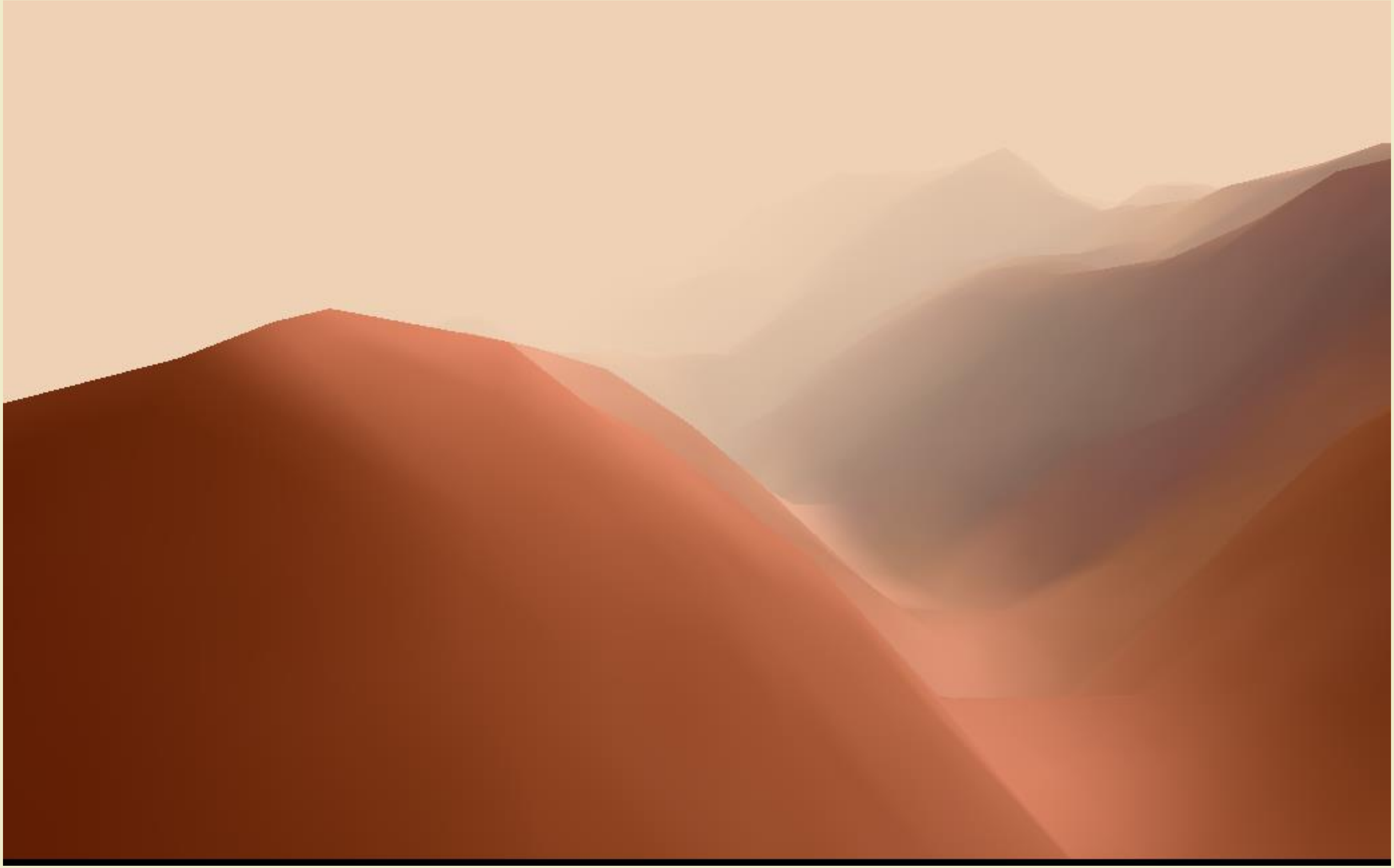
Lighting effect that is based on the distance of the pixel from the camera

$$\text{Near} = \text{OriginalColor}$$

$$\text{Far} = \text{FogColor}$$

$$\text{Between} = \text{Mix}(\text{FogColor}, \text{OriginalColor})$$

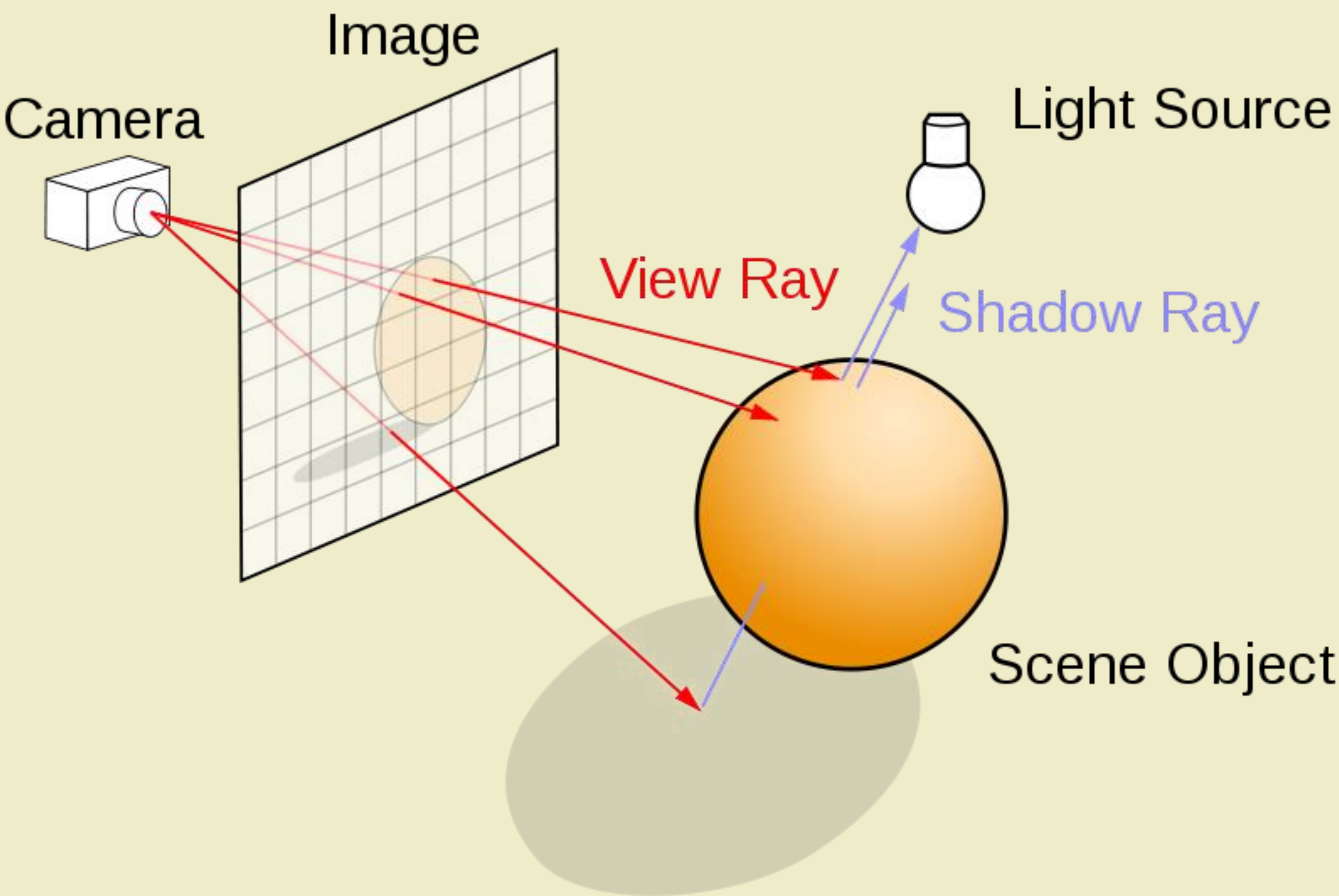
Fog



Ray Tracing

Achieves better realistic lighting by computing the view ray’s recursive reflections and refractions

Ray Tracing

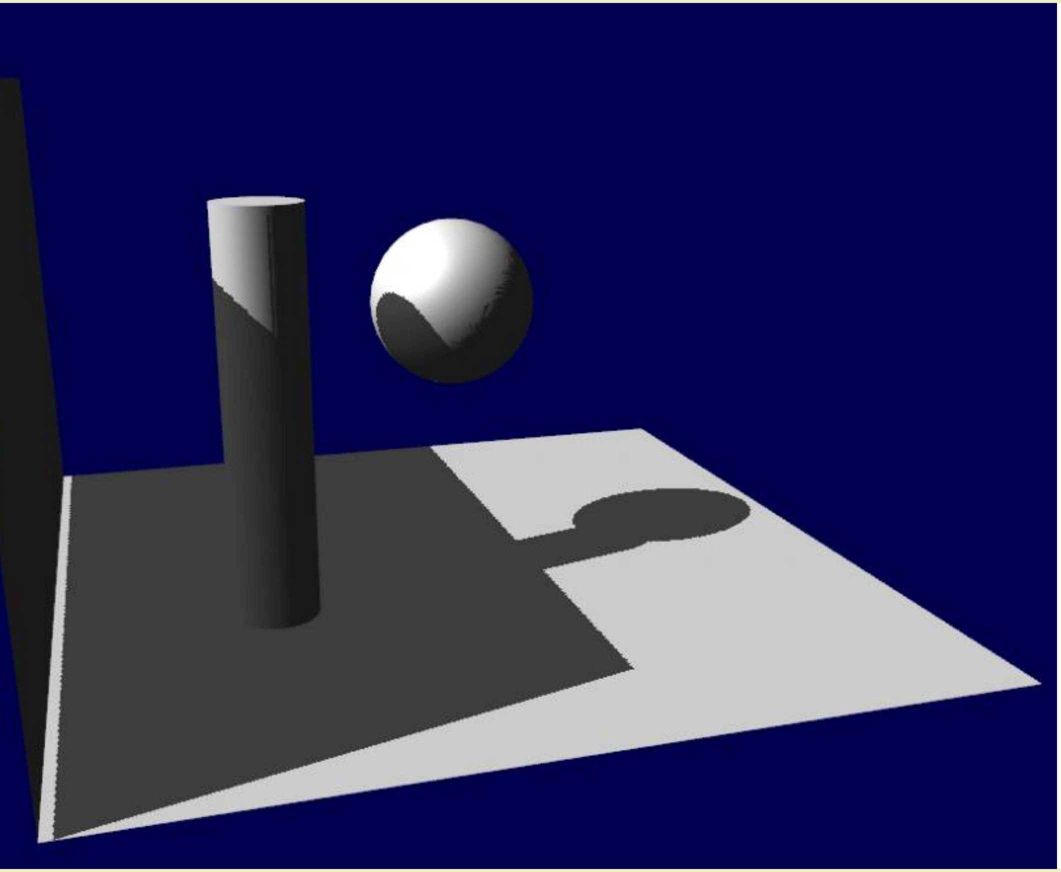
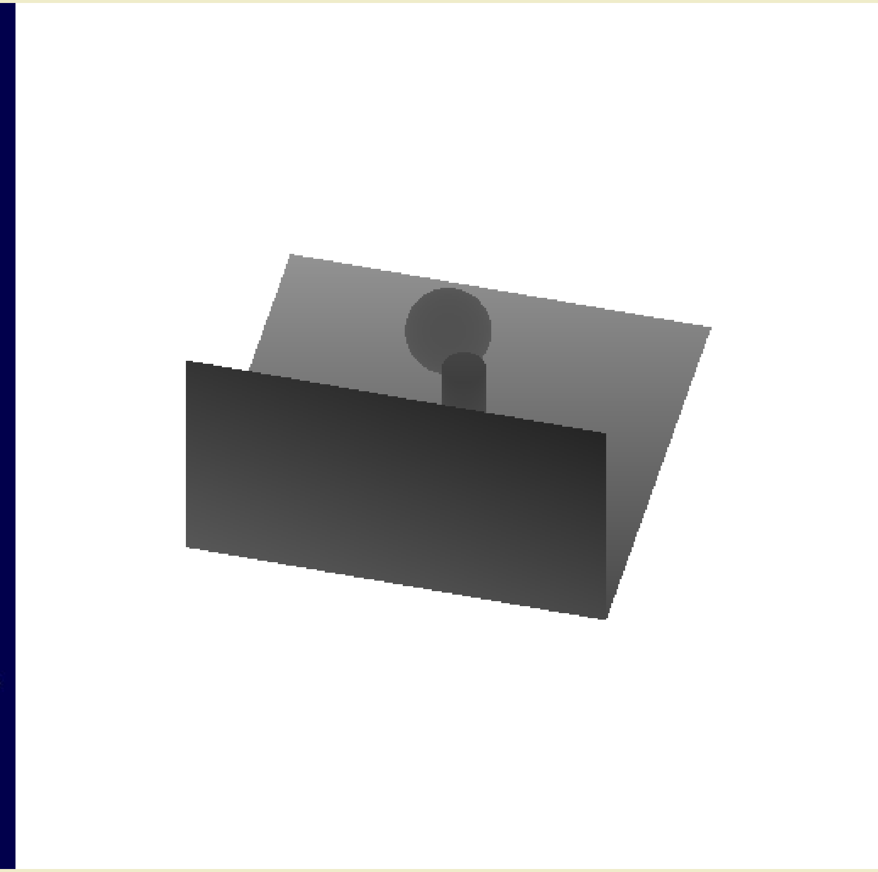


Shadow Mapping

A method for simulating shadows in the scene

Create the depth map using the light position as the camera

Use the depth map as the basis of pixels that will be lit

References

Books

- ANGEL, E. AND SHREINER, D. 2012. Interactive computer graphics : a top-down approach with shader-based OpenGL. Addison-Wesley. 6 ed. Boston, MA.
- CANTOR, D. AND JONES, B. 2012. WebGL Beginner's Guide. Packt Publishing. Birmingham, UK.
- MATSUDA, K. AND LEA, R. 2013. WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL.. Addison-Wesley. Upper Saddle River, NJ

Images

- http://fc00.deviantart.net/fs38/i/2008/350/0/d/UPLB_Christmas_Tree_by_yourex_loverisJa.jpg
- http://en.wikipedia.org/wiki/File:Tso_Kiagar_Lake_Ladakh.jpg
- http://www.indigorenderer.com/media/docs/729/techniquesmanual_html_m53bf5938.jpg
- http://en.wikipedia.org/wiki/File:Lambert2.gif
- http://www.cs.berkeley.edu/~ravir/refraction.jpg
- http://upload.wikimedia.org/wikipedia/commons/thumb/a/a5/Normal_vectors2.svg/195px-Normal_vectors2.svg.png
- http://cdn.tutsplus.com/gamedev/uploads/2013/10/scene1.png
- http://threejs.org/examples/#webgl_geometry_terrain_fog
- http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/
- http://4.bp.blogspot.com/-OLGB9JeddzY/T4c14EhZ7FI/AAAAAAAAAYY/3GXq3E_nk_s/s1600/SpecularMicrofacetBSDF.png
- http://www.codeproject.com/KB/GDI/3DSoftwareRenderingEngine/flatshaded.png