Lighting and Shading

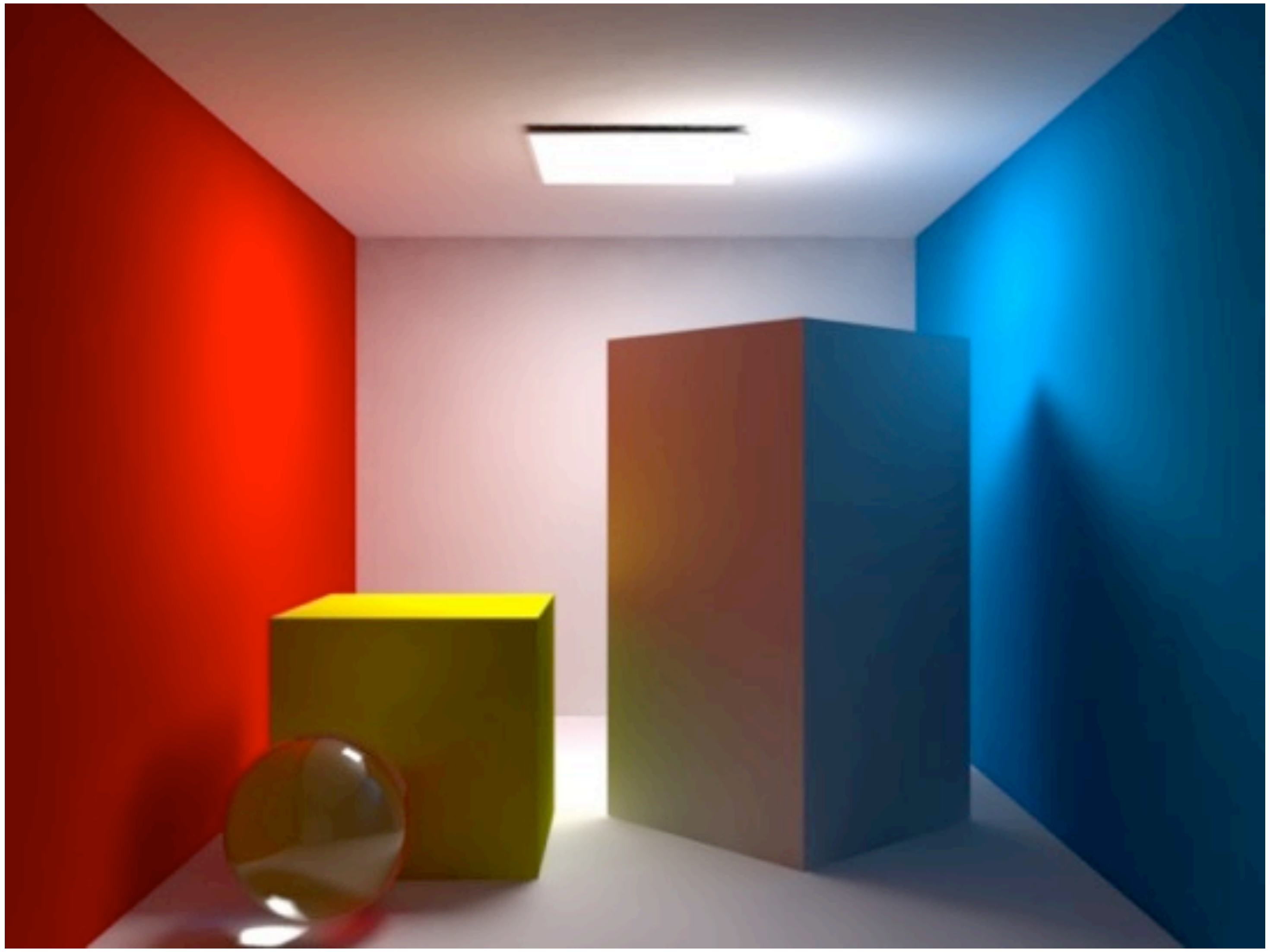
Prof. Vladlen Koltun
Computer Science Department
Stanford University

Lighting

- The appearance of objects arises out of their interaction with light. Without light, nothing is visible. The simulation of materials interacting with light is at the core of rendering.

Source: Mental Images

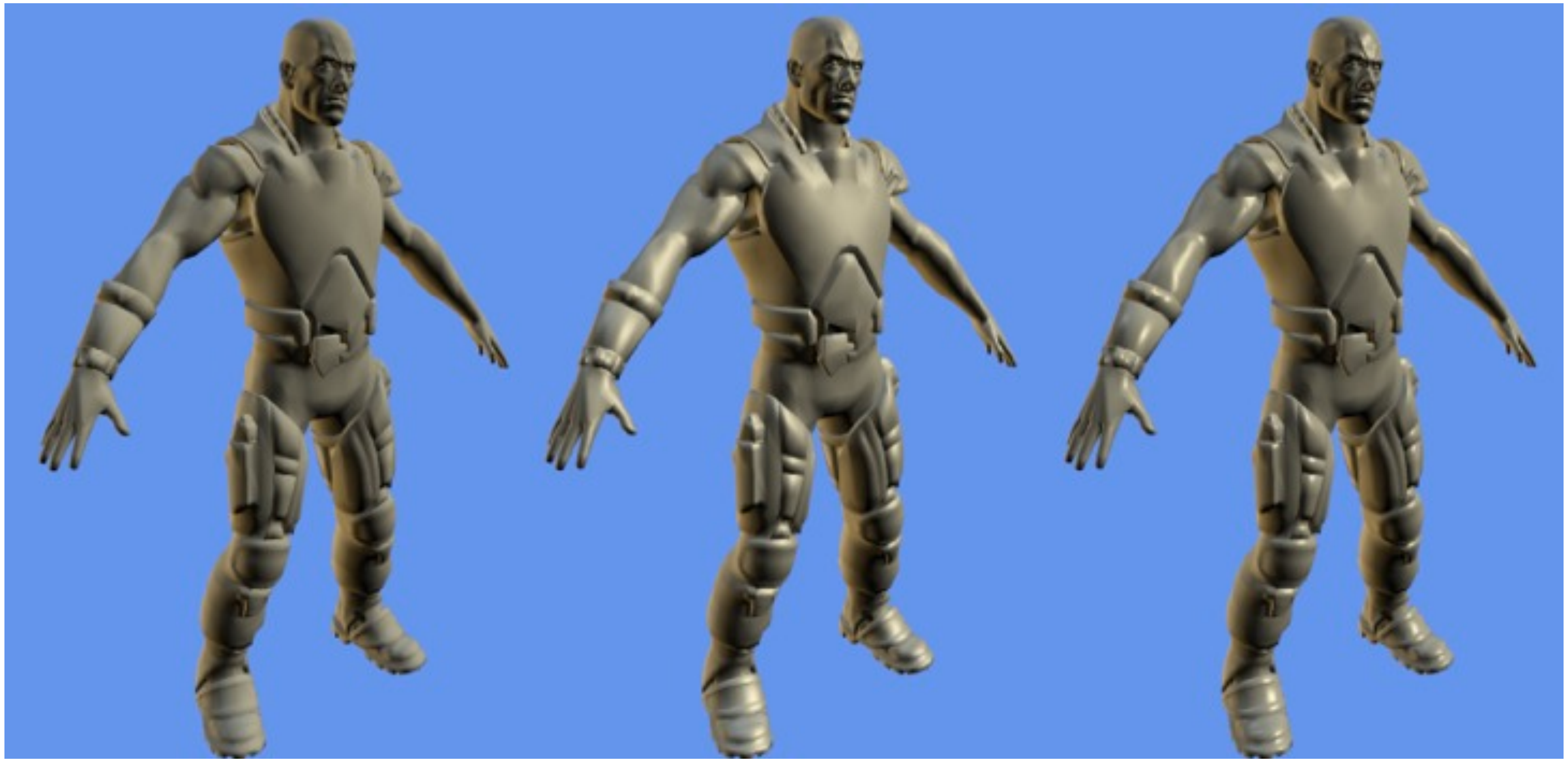
Lighting



- Accurate lighting simulation must account for inter-reflections between objects, shadows that arise due to occlusion, soft shadows due to area light sources, translucency, subsurface scattering, and other phenomena.

Source: CGTutorials

Lighting



- To integrate extremely fast shading into the graphics pipeline, we use a local approximation that only considers direct interactions between individual light sources and surfaces. Interactions between multiple objects are not taken into account.

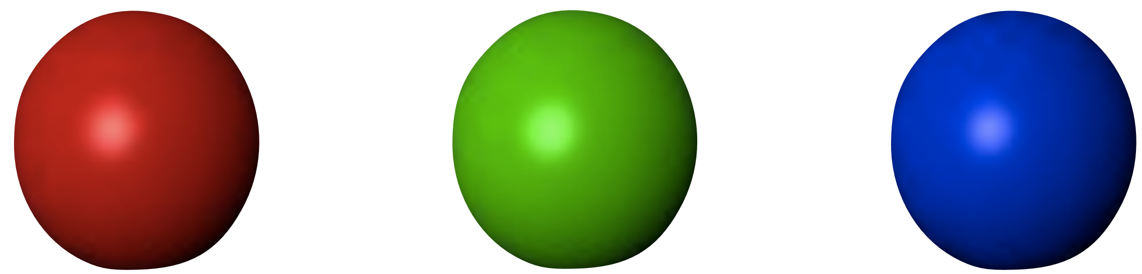
Source: Shawn Hargreaves

Lighting Approximation

Three key components:

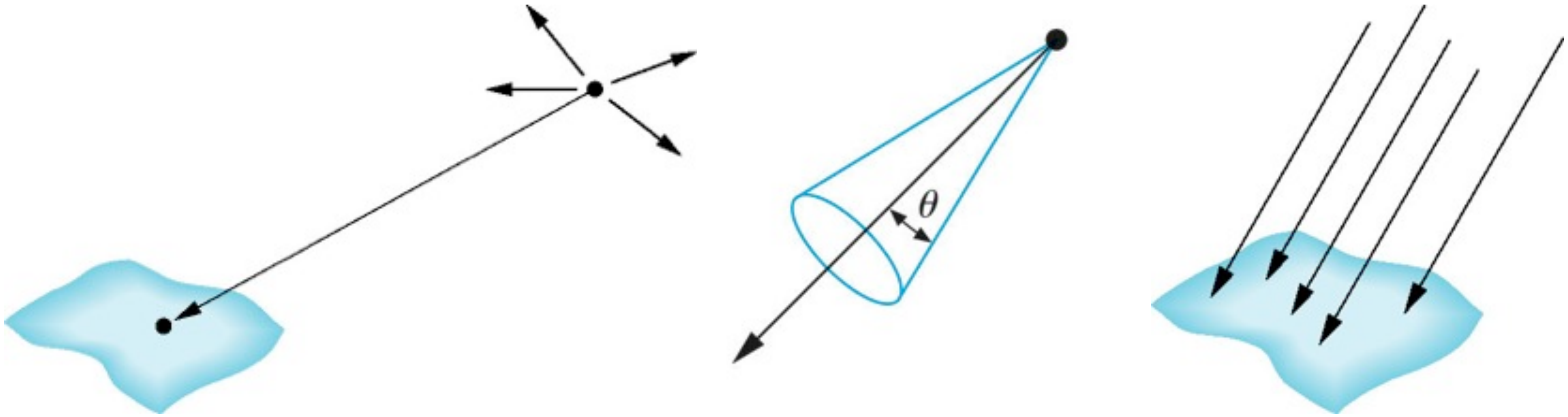
- Different wavelengths
- Idealized light sources
- Approximation of possible material structures by combinations of two components: specular and diffuse

Different Wavelengths



- Instead of considering the full visible spectrum, we perform lighting calculations *independently* for the three primary colors: red, green, and blue
- Each light and each material will have separate parameters for each color channel
- From now on, we will discuss single-channel lighting calculations, assuming that all the parameters and computations are performed in parallel for all three channels

Idealized light sources



- In nature, photons are emitted by objects with varying geometry.
- Our approximation considers only four types of idealized light sources: ambient light, point lights, spotlights, and directional lights.

Ambient Light

- A crude approximation to global scattering effects
- Softens the lighting and makes sure that even back-facing surfaces are not completely black.
- Allows us to avoid a large number of lights to cover a wide area
- Has a scalar intensity parameter L_a

Point Lights

- Approximates the effect of streetlights, candles, light bulbs, and other small omnidirectional light sources
- Emits light with equal intensity, L_p, in all directions from a single point
- In principle, light intensity falls off quadratically with distance. Thus the received intensity is

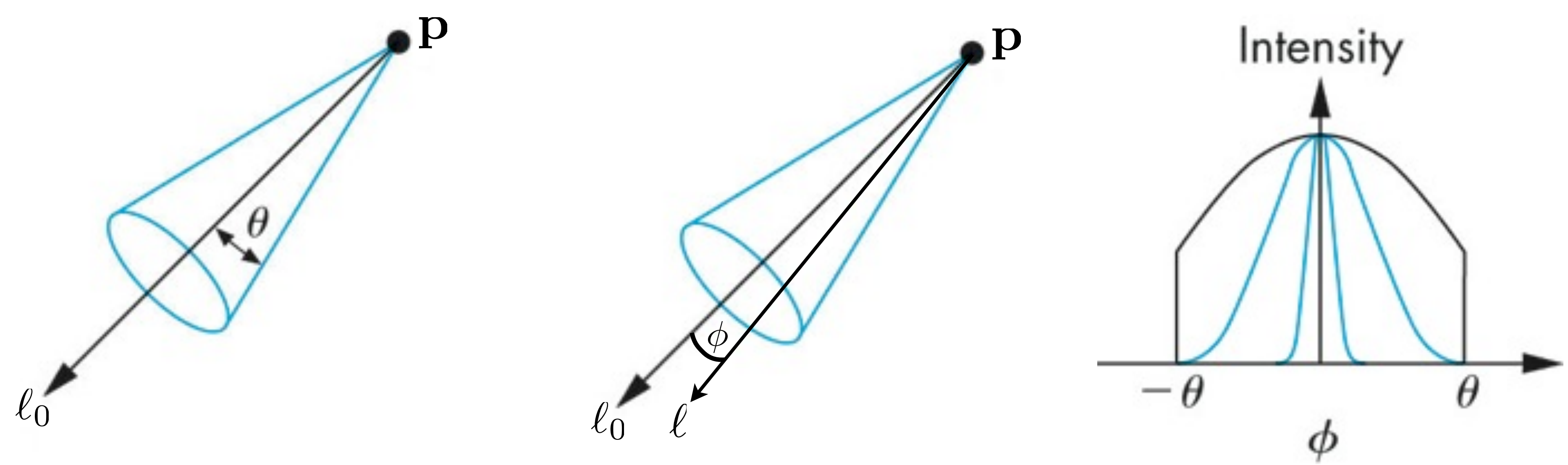
$$\frac{1}{d^2} L_p$$

- In practice this results in a sharp falloff. To better control the light distribution and avoid placing a large number of lights, we work with a more general falloff

$$\frac{1}{ad^2 + bd + c} L_p$$

- This leads to smoother and softer falloff and longer-range influence

Spotlights

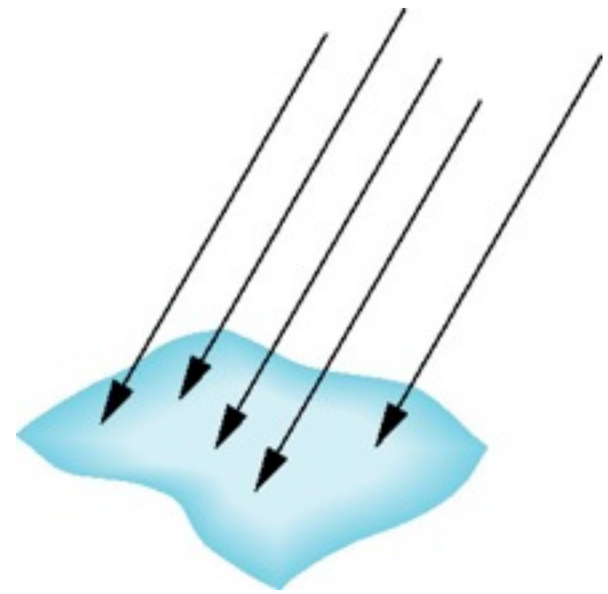


- Akin to a theatrical light that emits light within a cone of directions of radius θ, centered around the spotlight direction ℓ_0
- Light falls off around ℓ_0 as a function of the cosine of direction ϕ with ℓ_0. Thus the intensity of light in direction ℓ is modeled as

$$\cos^a(\phi) = (\ell_0 \cdot \ell)^a$$

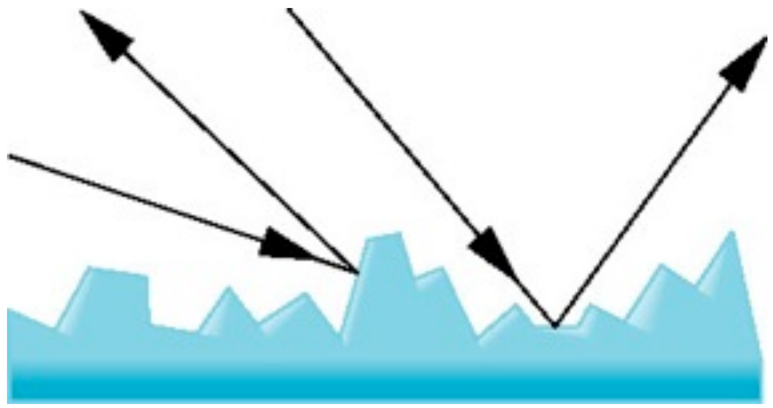
- The exponent a controls the speed of the falloff, and thus how focused the spotlight is around its direction

Directional Lights

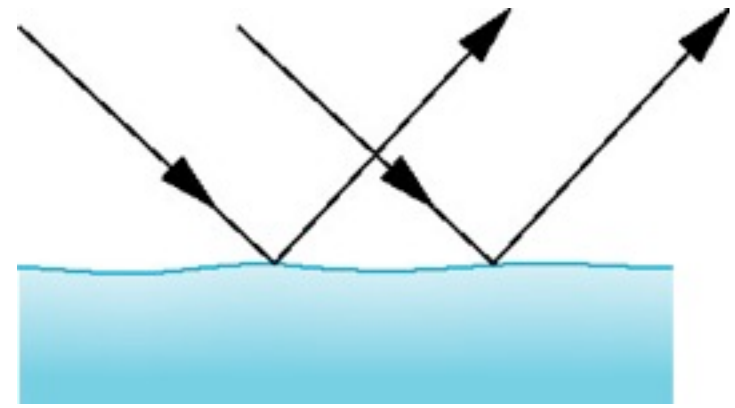


- Model distant light sources such as the sun, for which the amount of illumination depends only on the surface's angle to the light, not on distance.
- Shading computation for directional lights depends only on the orientation of the surface, not its position in space

Material Approximation



diffuse



specular

Materials are approximated by a combination of two components:

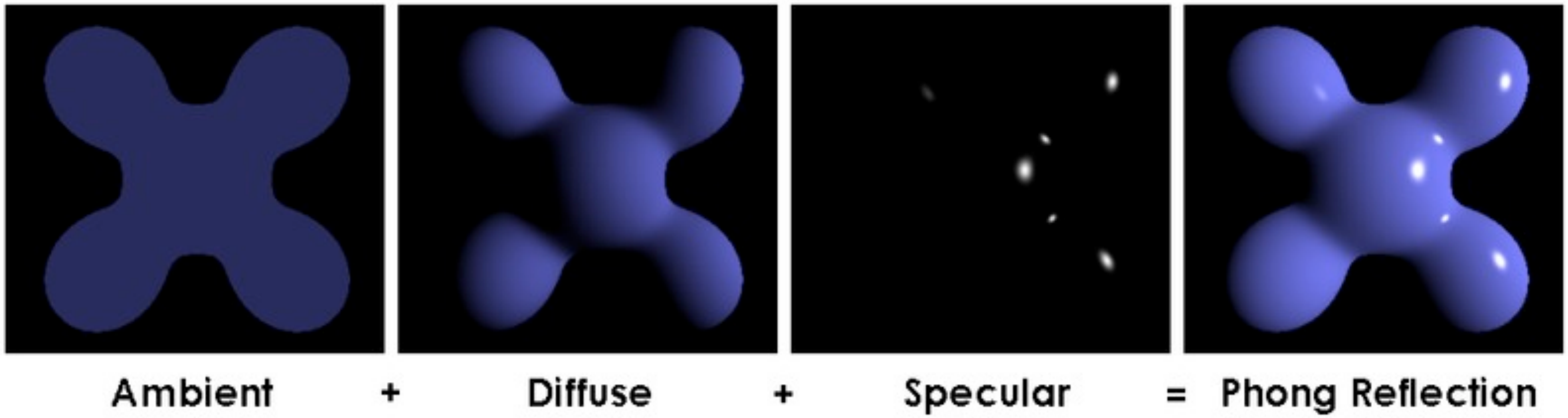
- Diffuse: rough surfaces that diffuse light omni-directionally
- Specular: smooth surfaces that reflect light in a narrow range around the angle of reflection

Source: Shawn Hargreaves

Phong Lighting Model

- Computed separately for each light. Contributions of individual lights are added together
- Each light has a separate ambient, diffuse, and specular component
- The surface also has independent ambient, diffuse, and specular components

Phong Lighting Model



- For each light, the reflected intensity is

$$I = I_a + I_d + I_s = R_a L_a + R_d L_d + R_s L_s$$

where the L terms represent the intensity of incoming ambient, diffuse, and specular illumination from the considered light source, and the R terms represent how much of the illumination is reflected by the surface.

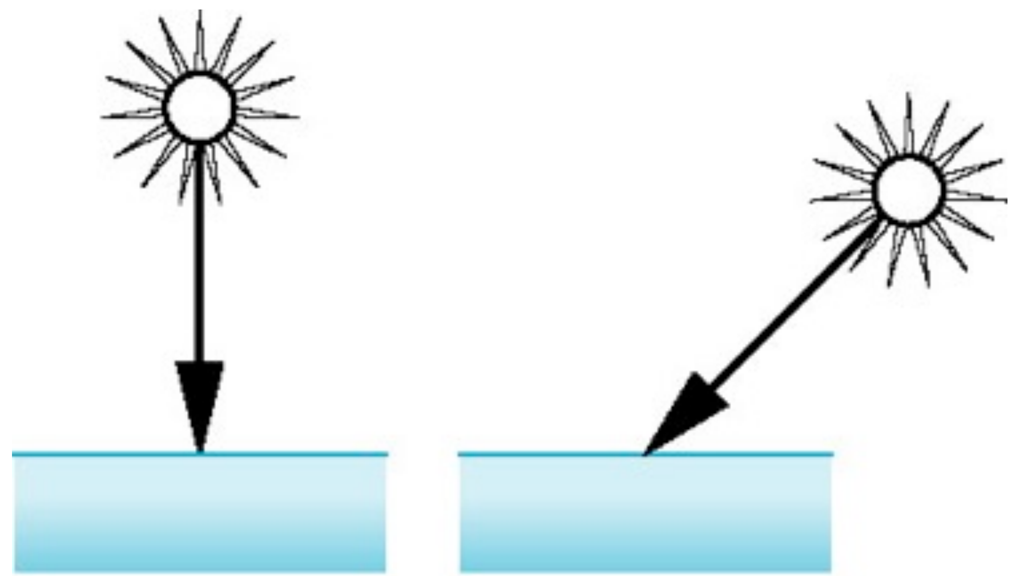
Source: Brad Smith, Wikimedia Commons

Ambient Reflection

- The ambient reflection term is just a constant:

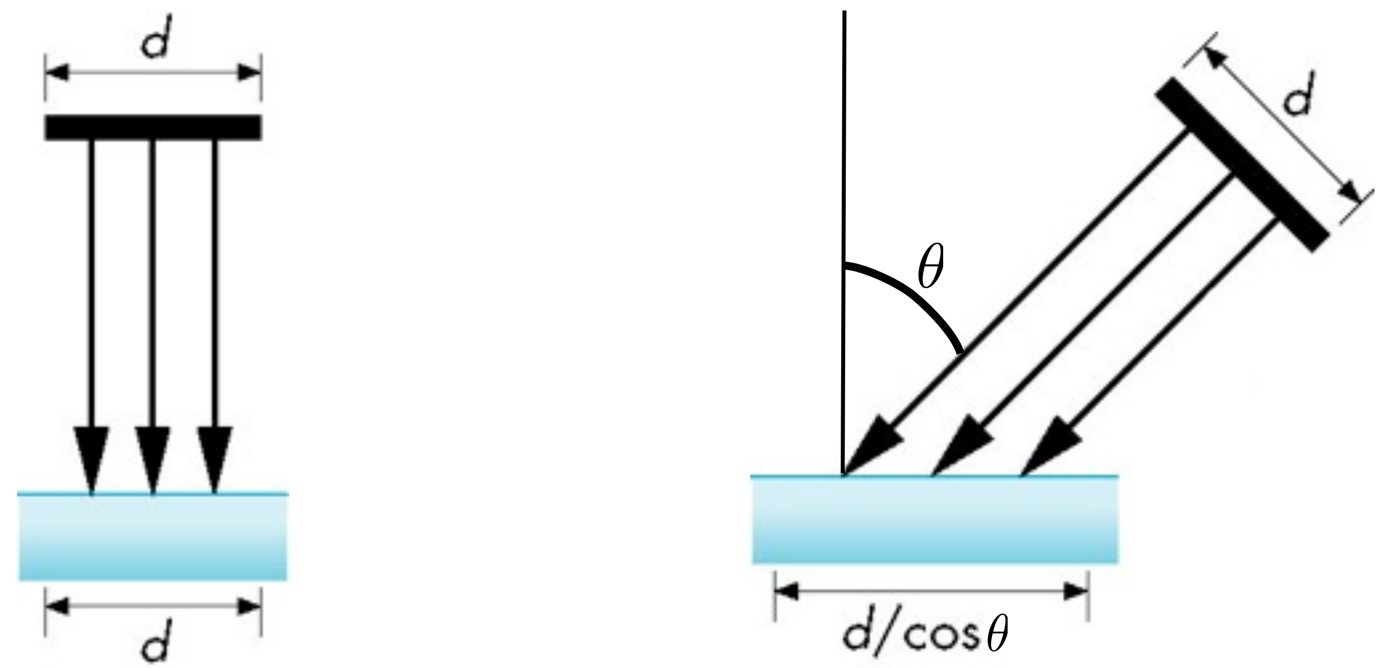
$$R_a = k_a, \quad 0 \leq k_a \leq 1$$

Diffuse Reflection



- Independent of viewing direction: perfectly diffuse lighting is reflected equally in all directions.
- Does depend on angle of incoming light.

Diffuse Reflection



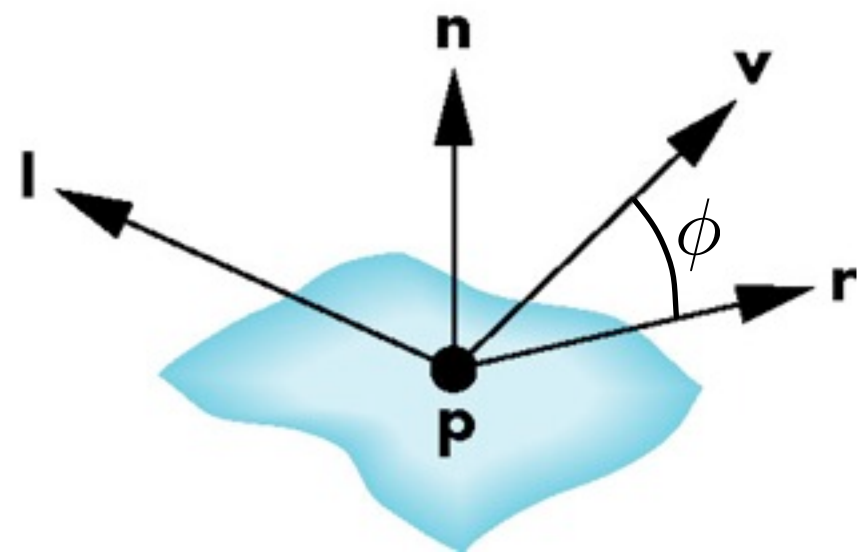
- Lambert's law: reflected diffuse intensity is proportional to the cosine of the angle of the light direction with the normal direction:

$$R_d = k_d \cos\theta = k_d(\mathbf{l} \cdot \mathbf{n})$$

where n is the normal vector, l is the unit vector from the surface point in the direction of the light source, and k_d is a diffuse reflection coefficient.

- To avoid negative values, we use $\max(0, \mathbf{l} \cdot \mathbf{n})$

Specular Reflection



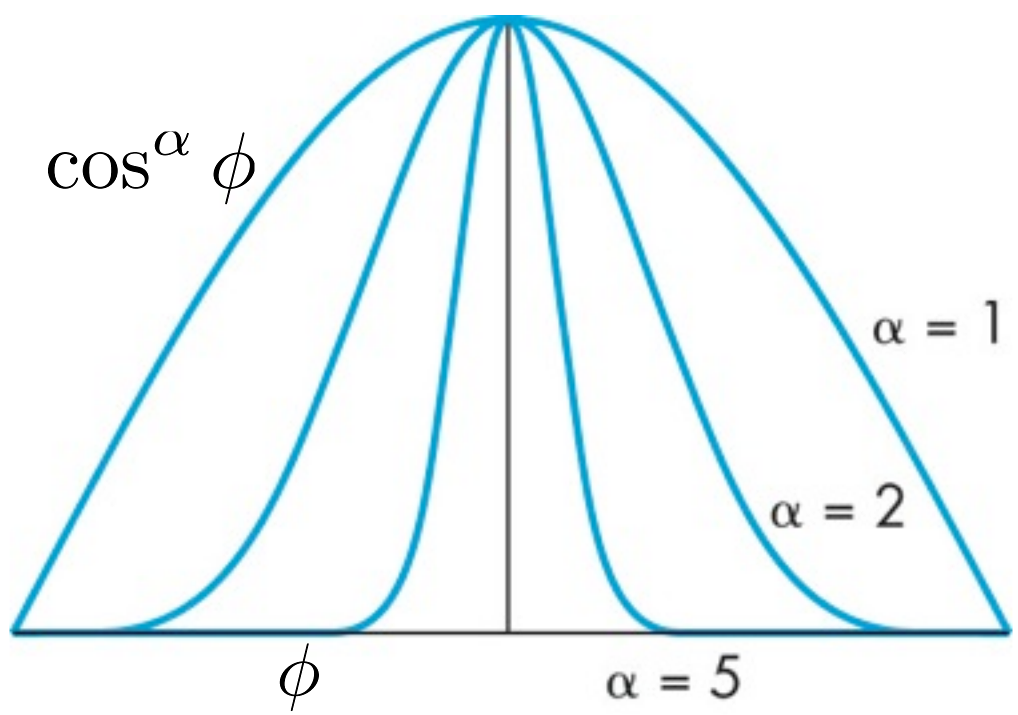
- Depends on the viewing angle:

$$R_s = k_s \cos^\alpha \phi = k_s(\mathbf{r} \cdot \mathbf{v})^\alpha$$

where r is the reflected light direction, v is the unit vector from the surface point to the viewer, and k_s is a specular reflection coefficient.

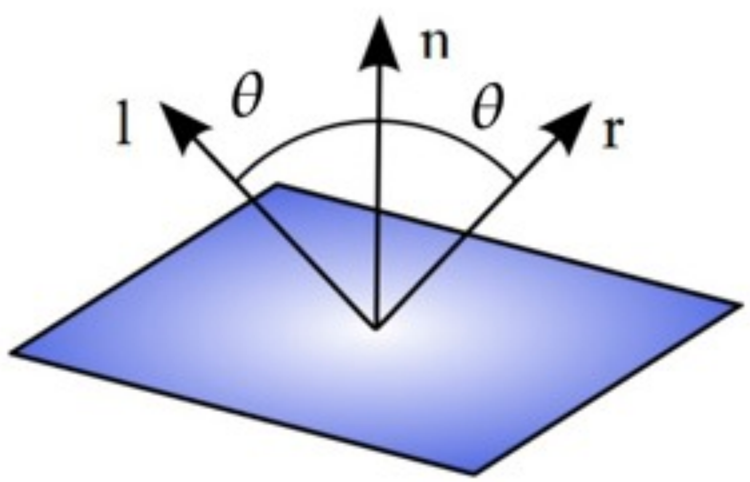
- To avoid negative values, we again use $\max(0, \mathbf{r} \cdot \mathbf{v})$

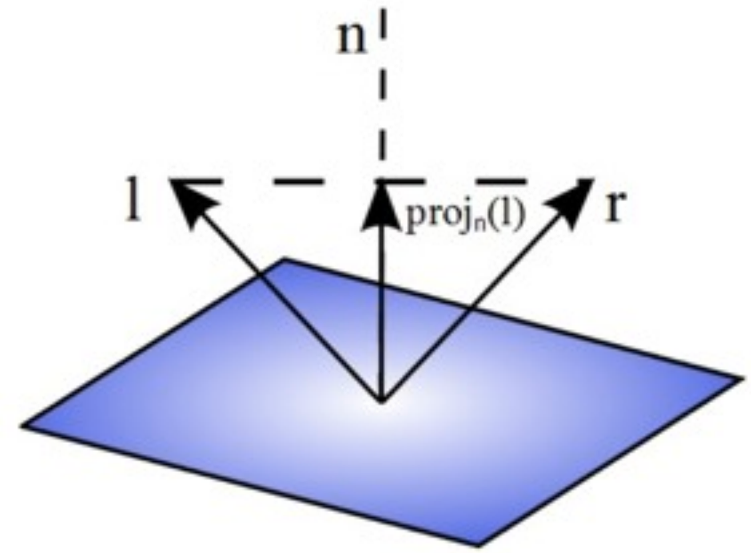
Specular Reflection



- α is the shininess coefficient of the material. The higher it is, the more concentrated the specular highlights are about the reflection direction, and the smoother the surface appears.

Computing Reflection Vectors



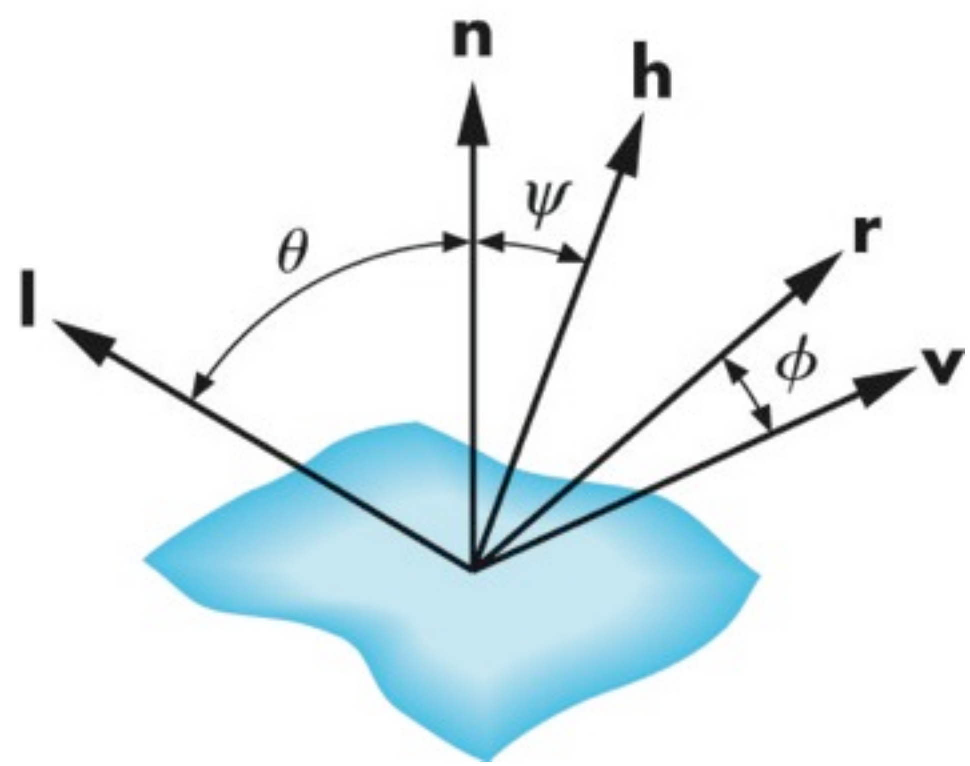


$$\frac{\mathbf{l}+\mathbf{r}}{|\mathbf{l}+\mathbf{r}|}=\mathbf{n} \quad \Rightarrow \quad \mathbf{l}+\mathbf{r}=|\mathbf{l}+\mathbf{r}|\mathbf{n}$$

$$|\mathbf{l}+\mathbf{r}| = 2\ \mathrm{proj}_{\mathbf{n}}(\mathbf{l}) = 2(\mathbf{l}\cdot\mathbf{n})$$

$$\mathbf{r} = 2(\mathbf{l}\cdot\mathbf{n})\mathbf{n}-\mathbf{l}$$

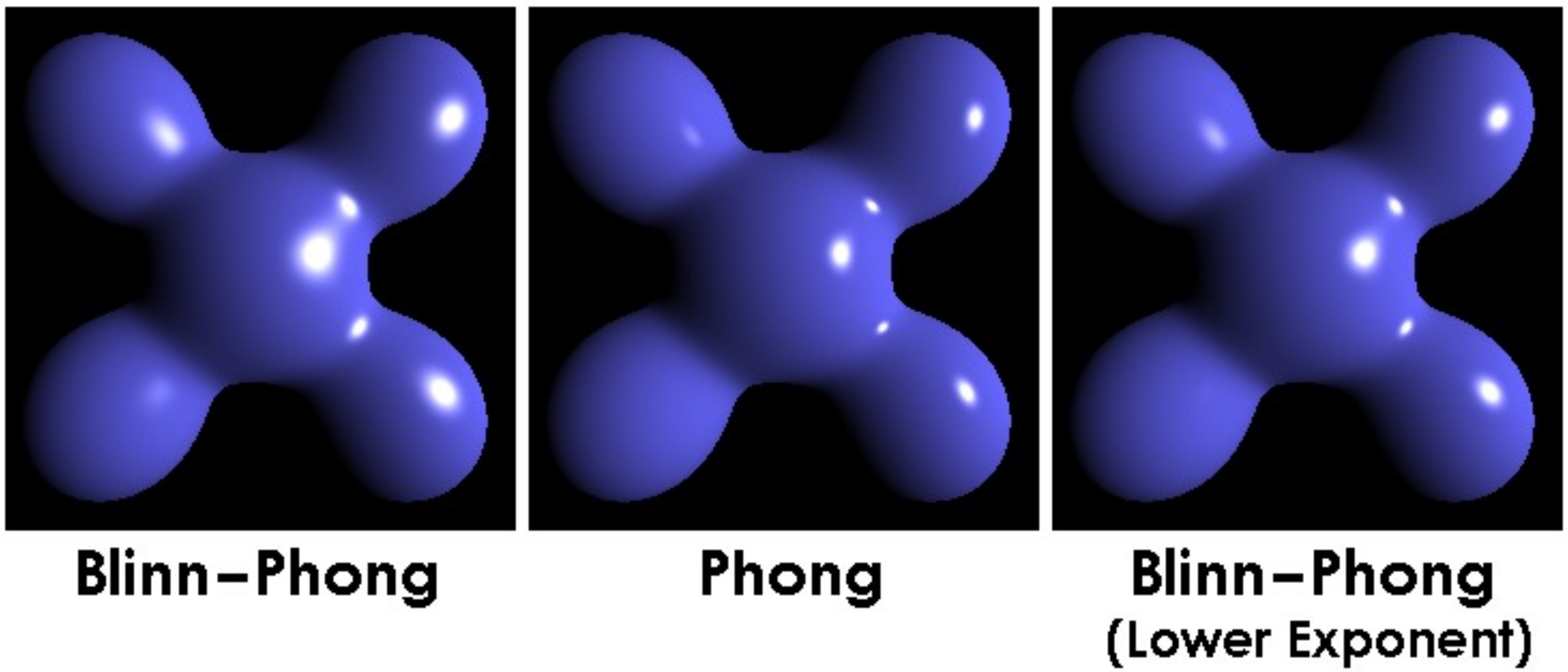
Blinn-Phong Model



- Avoids computation of reflection vector
- Replaces angle ϕ with the halfway angle ψ
- Requires computing the halfway vector

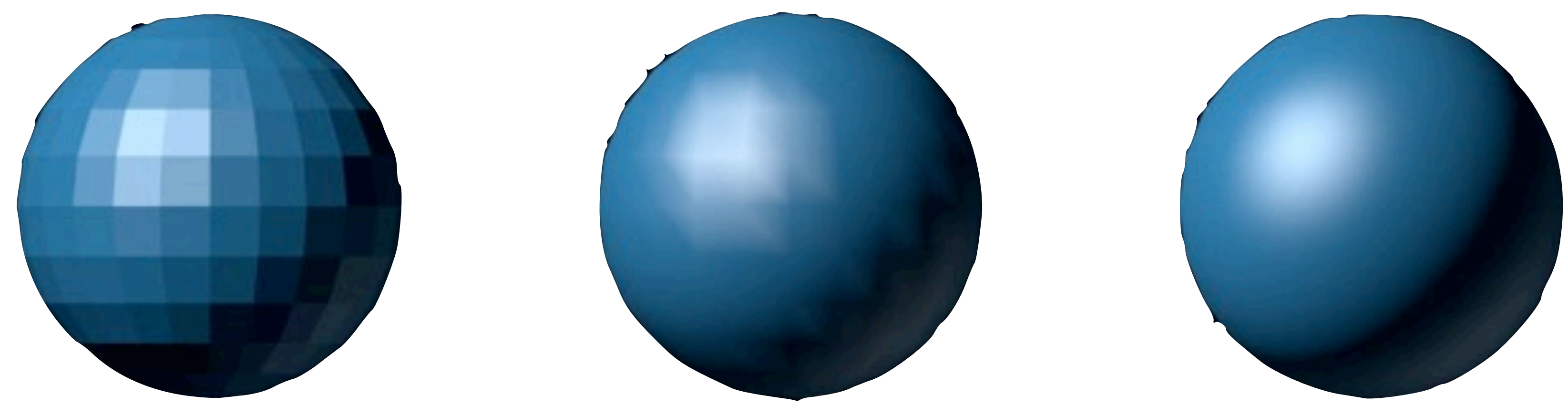
$$\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{|\mathbf{l} + \mathbf{v}|}$$

Phong vs. Blinn-Phong



Source: Brad Smith, Wikimedia Commons

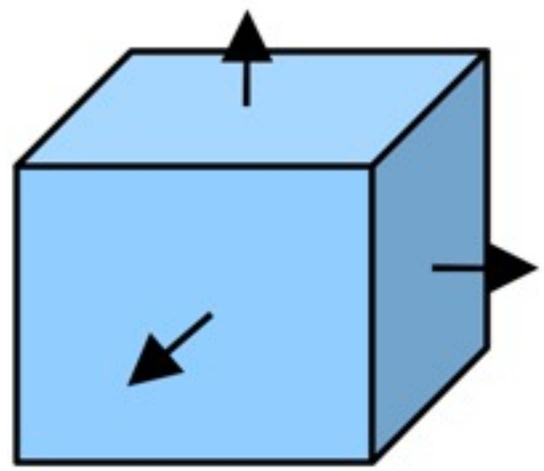
Shading Models

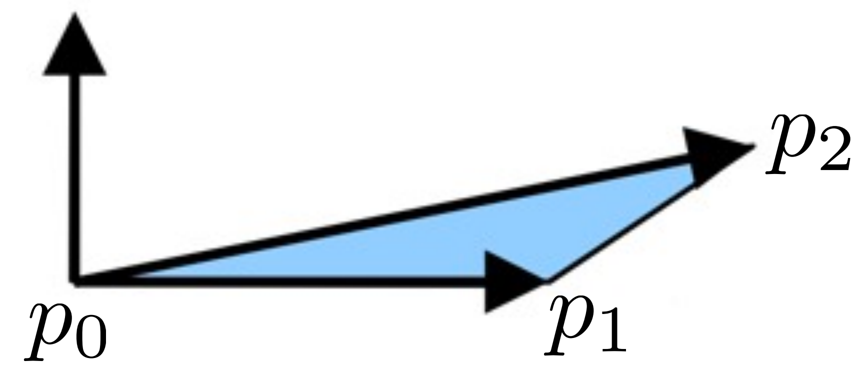


flat shading Gouraud shading Phong shading

- Flat shading: Perform lighting calculations per face using normal vector to the face.
- Gouraud shading: Perform lighting calculations per vertex using vertex normals.
- Phong shading: Perform lighting calculations per fragment using interpolated normals.

Normal Vectors





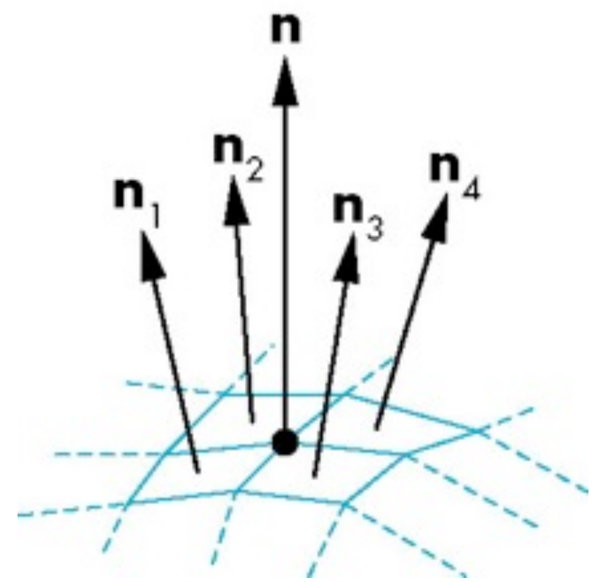
- The unnormalized normal vector is the cross-product

$$n = (p_1 - p_0) \times (p_2 - p_0)$$

- The vertices p_0, p_1, and p_2 should be in counterclockwise order around the face, when viewed from outside the object. (Right-hand rule.)

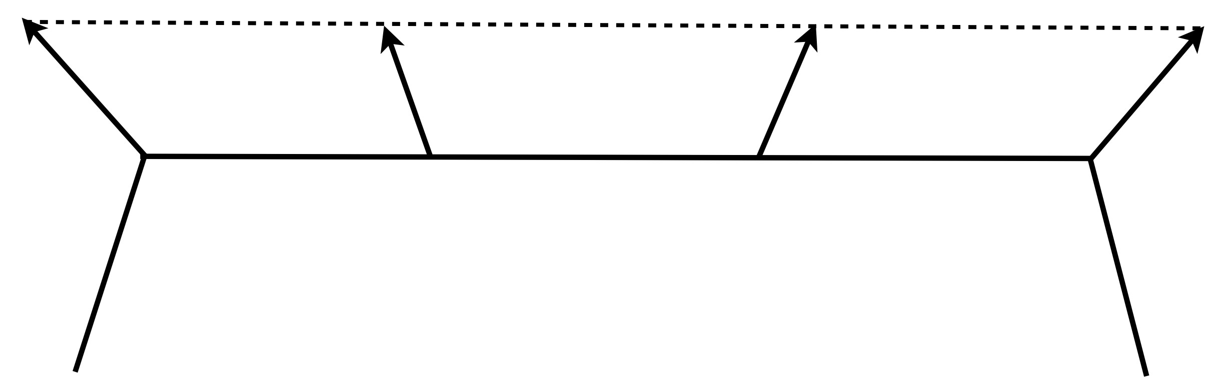
- Important since the cross-product is not commutative.

Vertex Normals



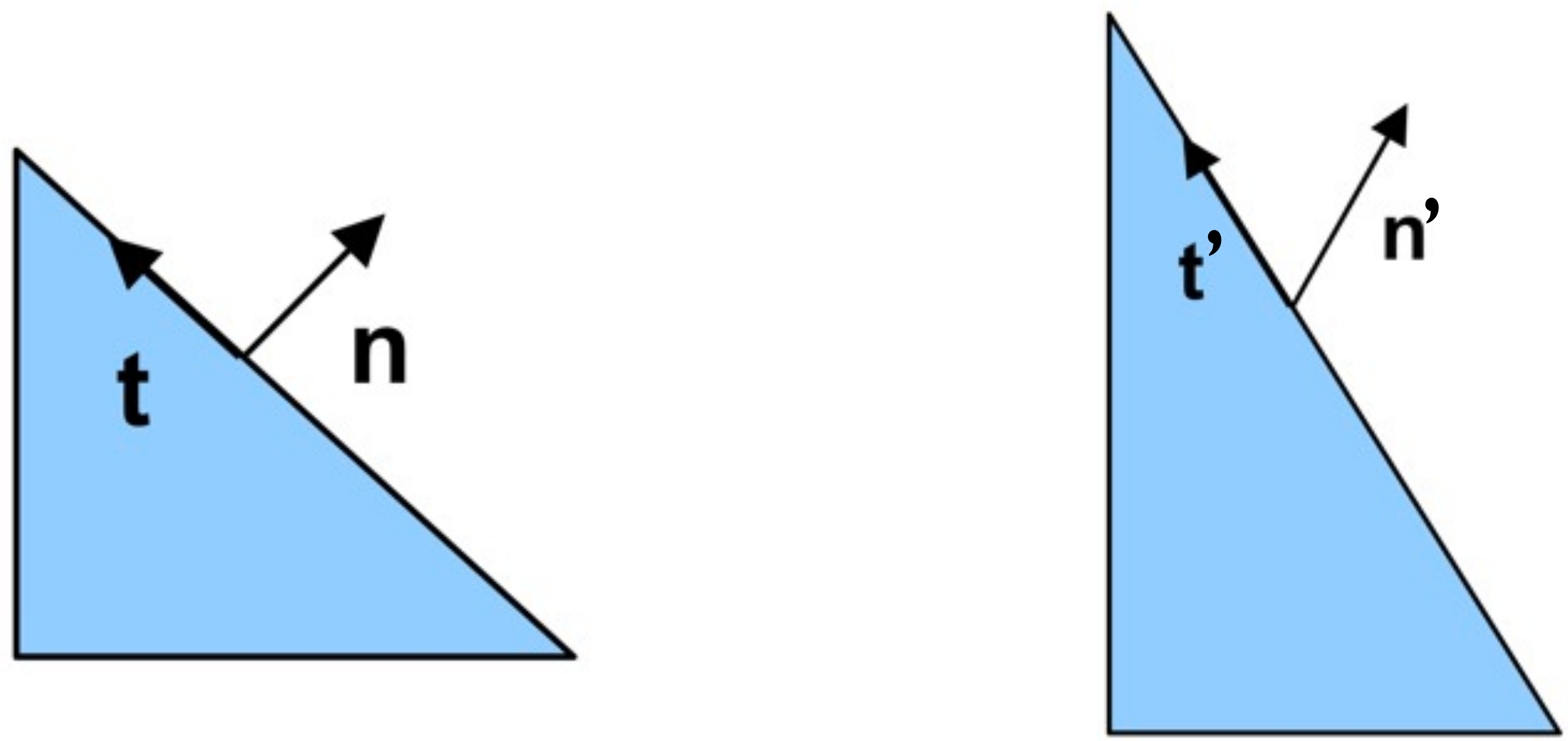
- Ill-defined: we don't know what the right normal is
- Can be obtained by averaging the normals of the incident faces
- Can be weighted by the area of each face or (better) by the incident angle of the face at the vertex

Interpolating Normals



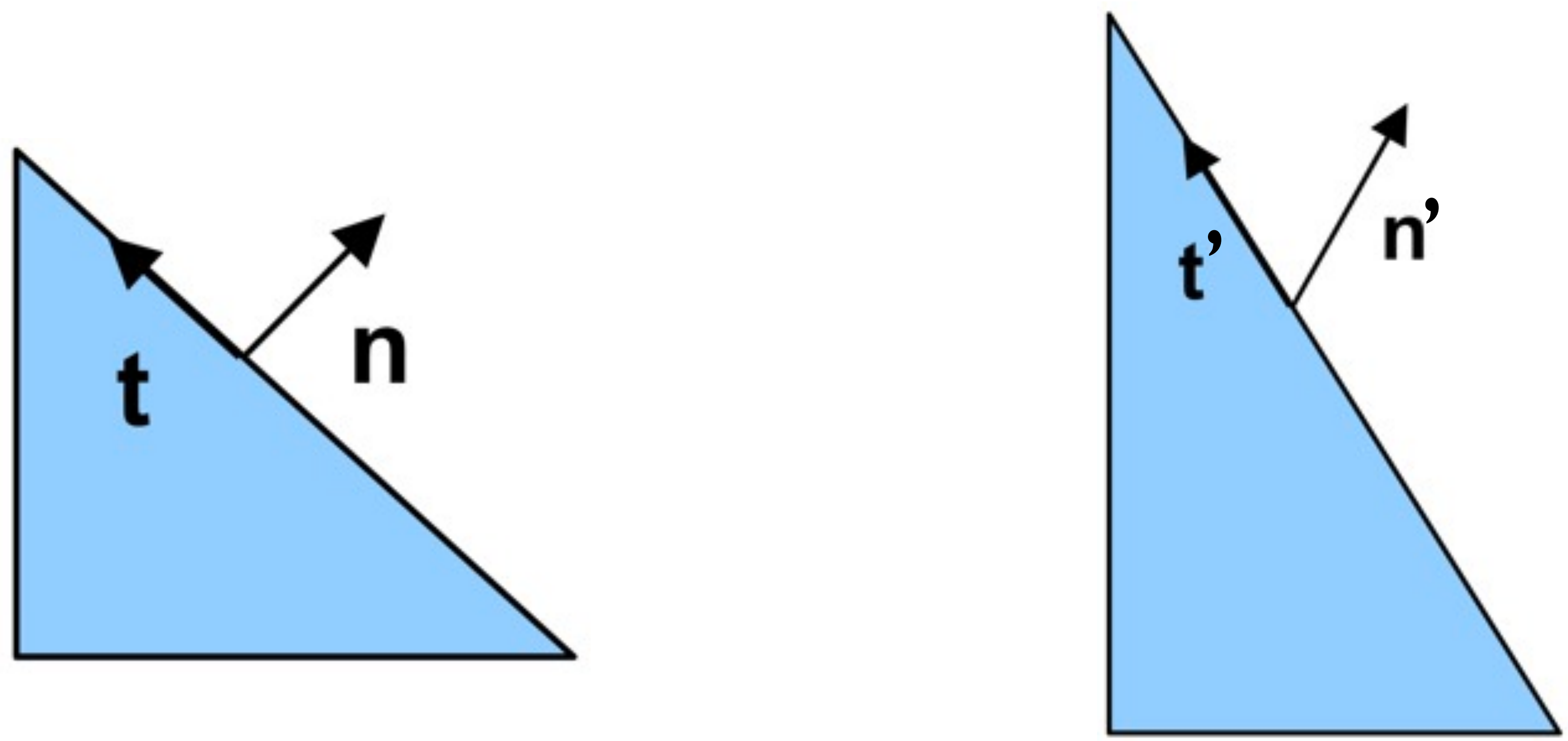
- Again ill-defined: we don't know the curvature of the surface inside the polygon
- Not seeking correctness, merely consistency
- Linear interpolation in world space, or perspective-correct interpolation in screen space

Transforming Normals



- Applying the same modelview transform to normals can break orthogonality.

Transforming Normals



- We store normals in Cartesian coordinates (x,y,z), since translation is irrelevant.
- Let M be the upper 3x3 of the modelview transform. We wish to find a transform N that keeps the normal orthogonal to surface.

Transforming Normals

Consider a tangent vector t and the normal n. We know that

$$n \cdot t = 0$$

The vector t is transformed by the matrix M into $t' = Mt$

Our goal is to find a matrix N that transforms n into $n' = Nn$

such that $n' \cdot t' = 0$

Equivalently,

$$(n')^T t' = 0$$

$$(Nn)^T M t = 0$$

$$n^T N^T M t = 0$$

Transforming Normals

$$n^T N^T M t = 0$$

Since $n^T t = 0$, this condition is satisfied when

$$N^T M = I$$

$$N = (M^{-1})^T$$

This is the correct transform for the normal.

Note that we must normalize the transformed normal n'.