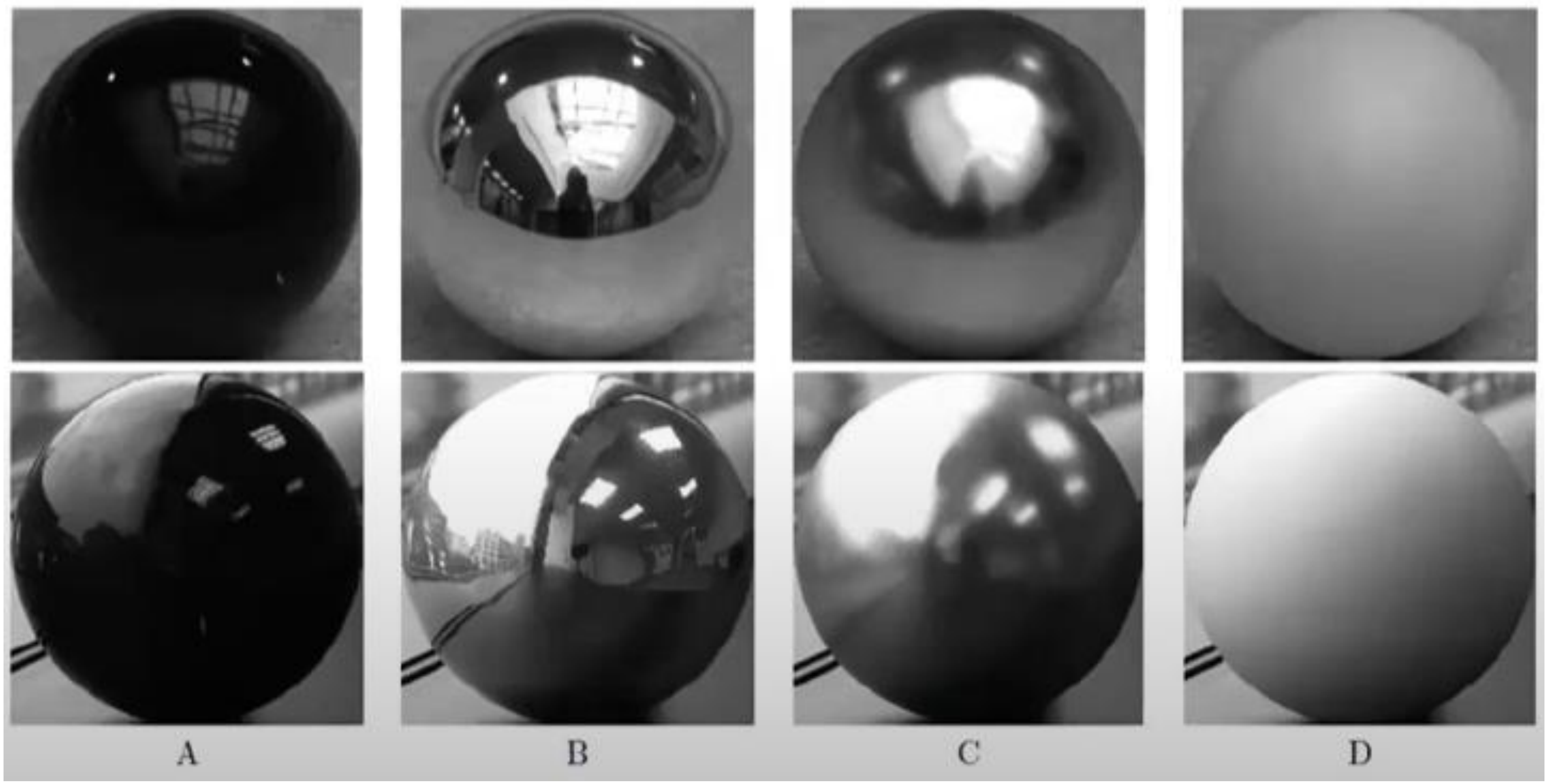
Light and Shading

Light Source

- The total reflected light is the sum of the contributions from the light source and other reflecting surface
- Light source are referred as ***Light emitting source***
- Reflecting surface is referred as ***light reflecting surface***.
- The amount of light reflected by a surface depends on the type of the material.

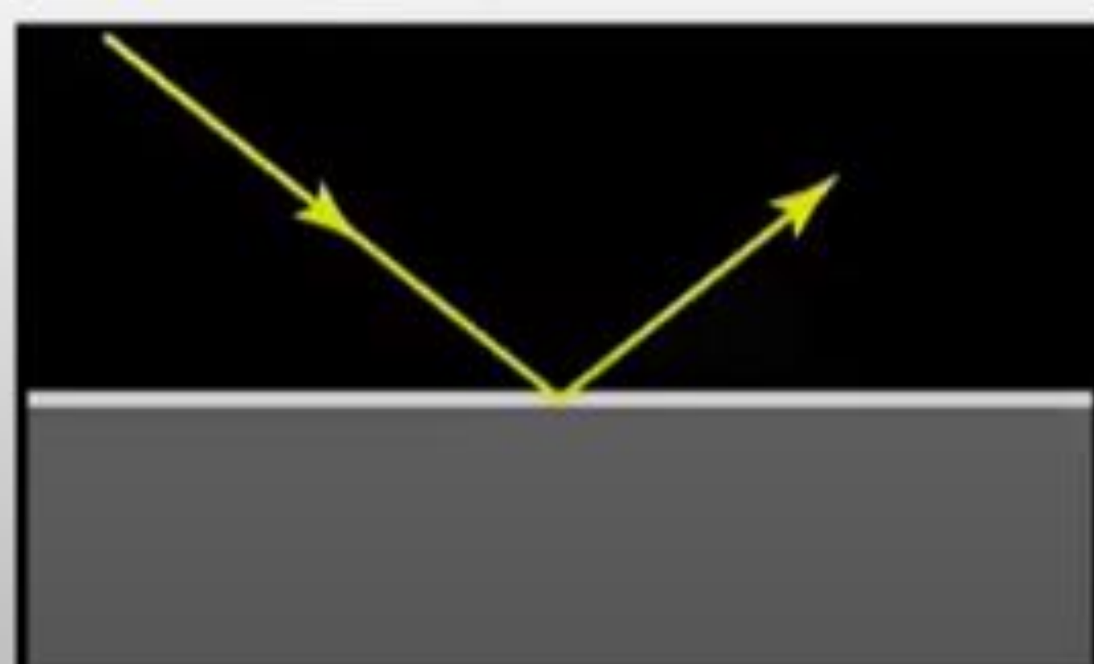
Recognizing Material

- Human visual are quite good at understanding shading

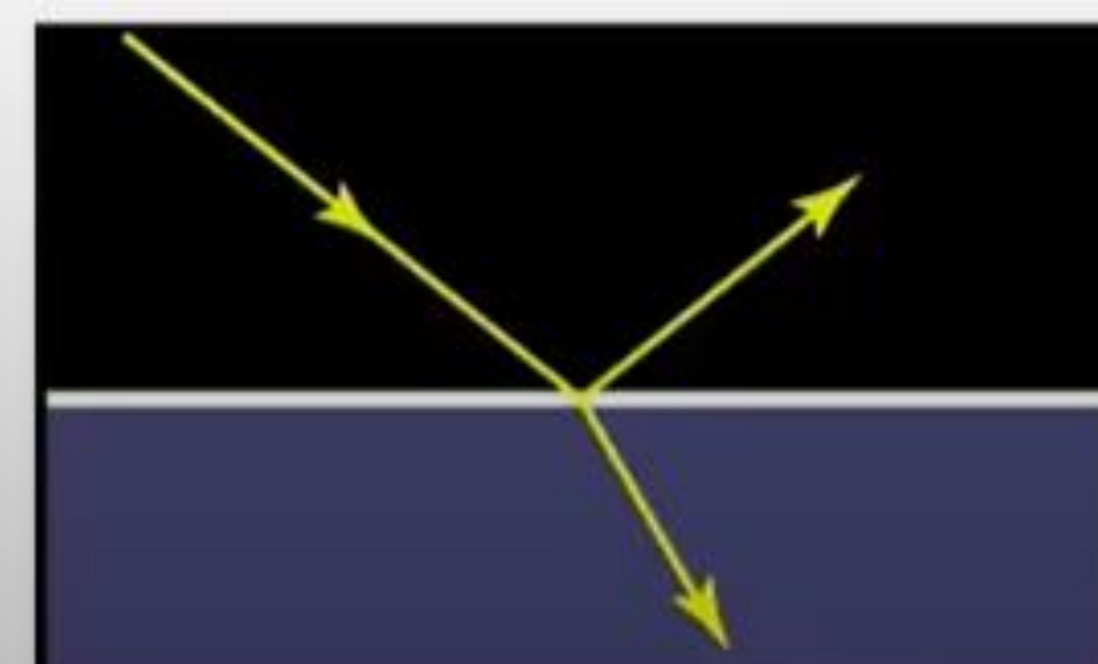


Simple materials

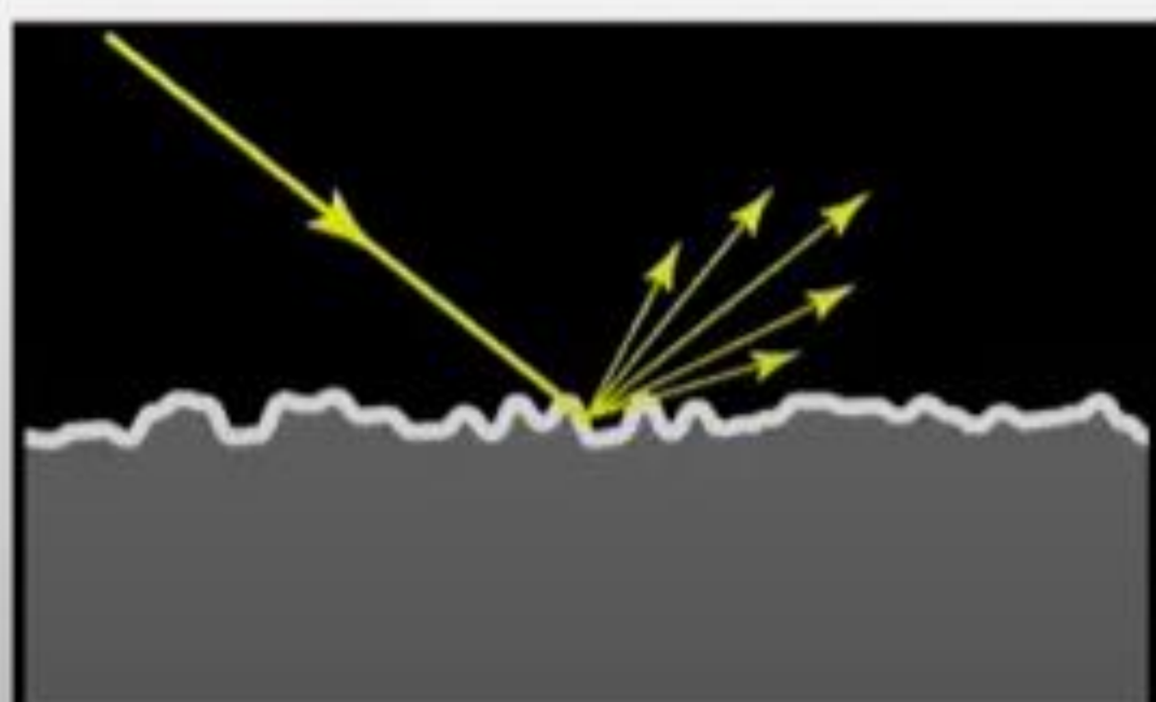
metal

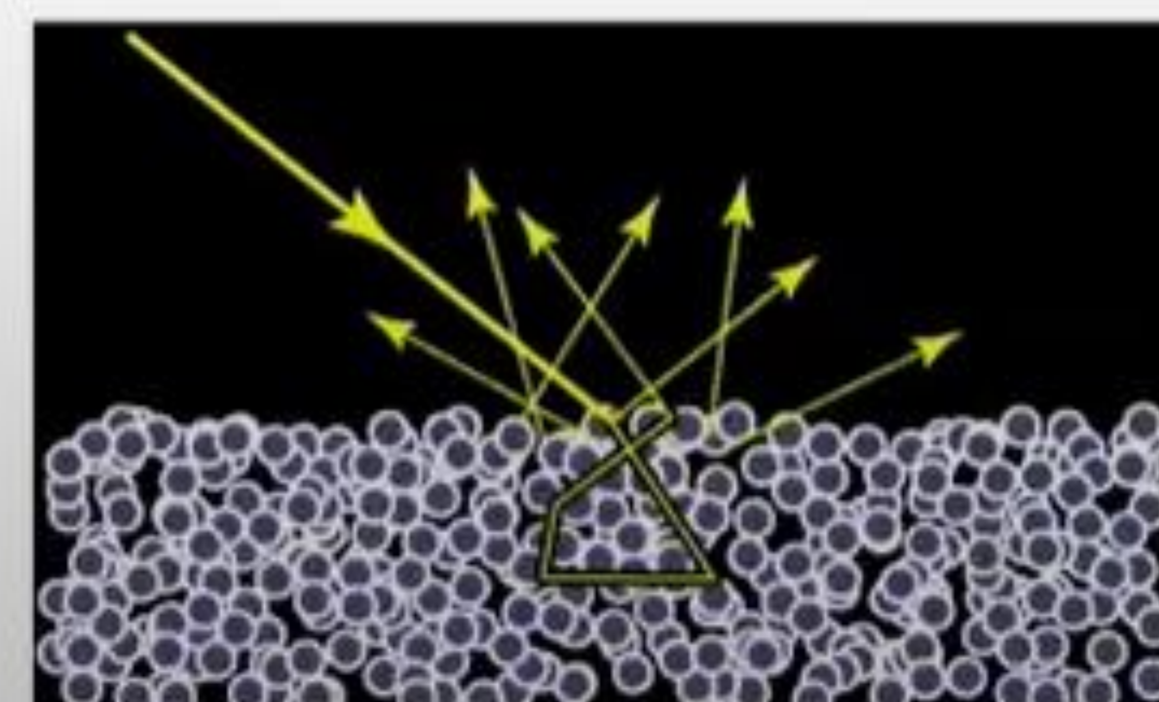


dielectric

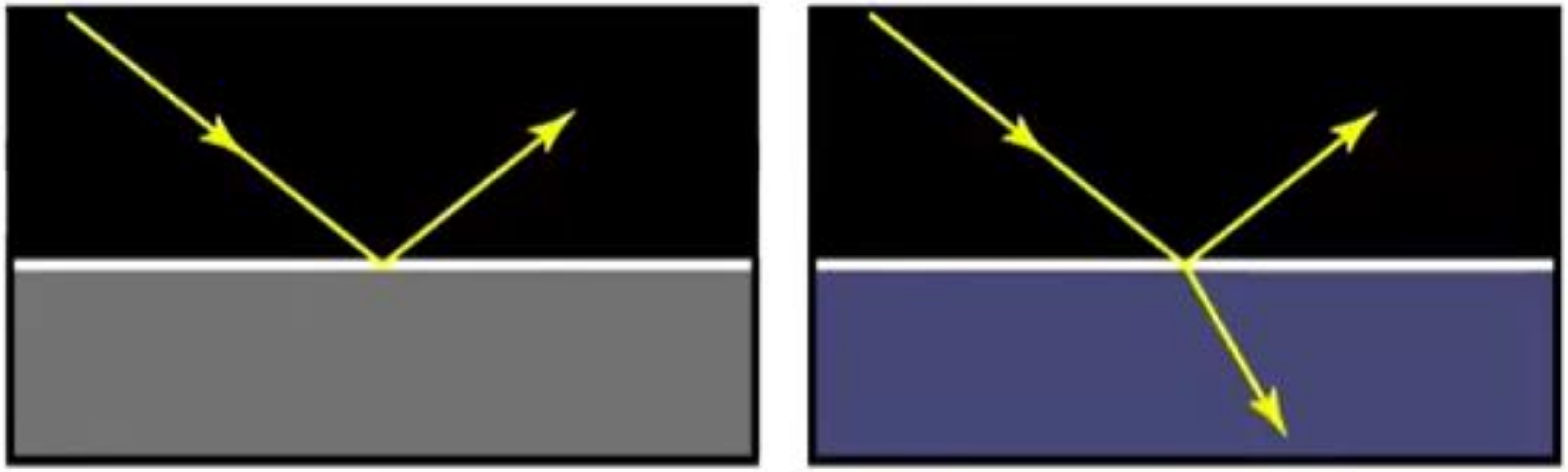


Adding microgeometry

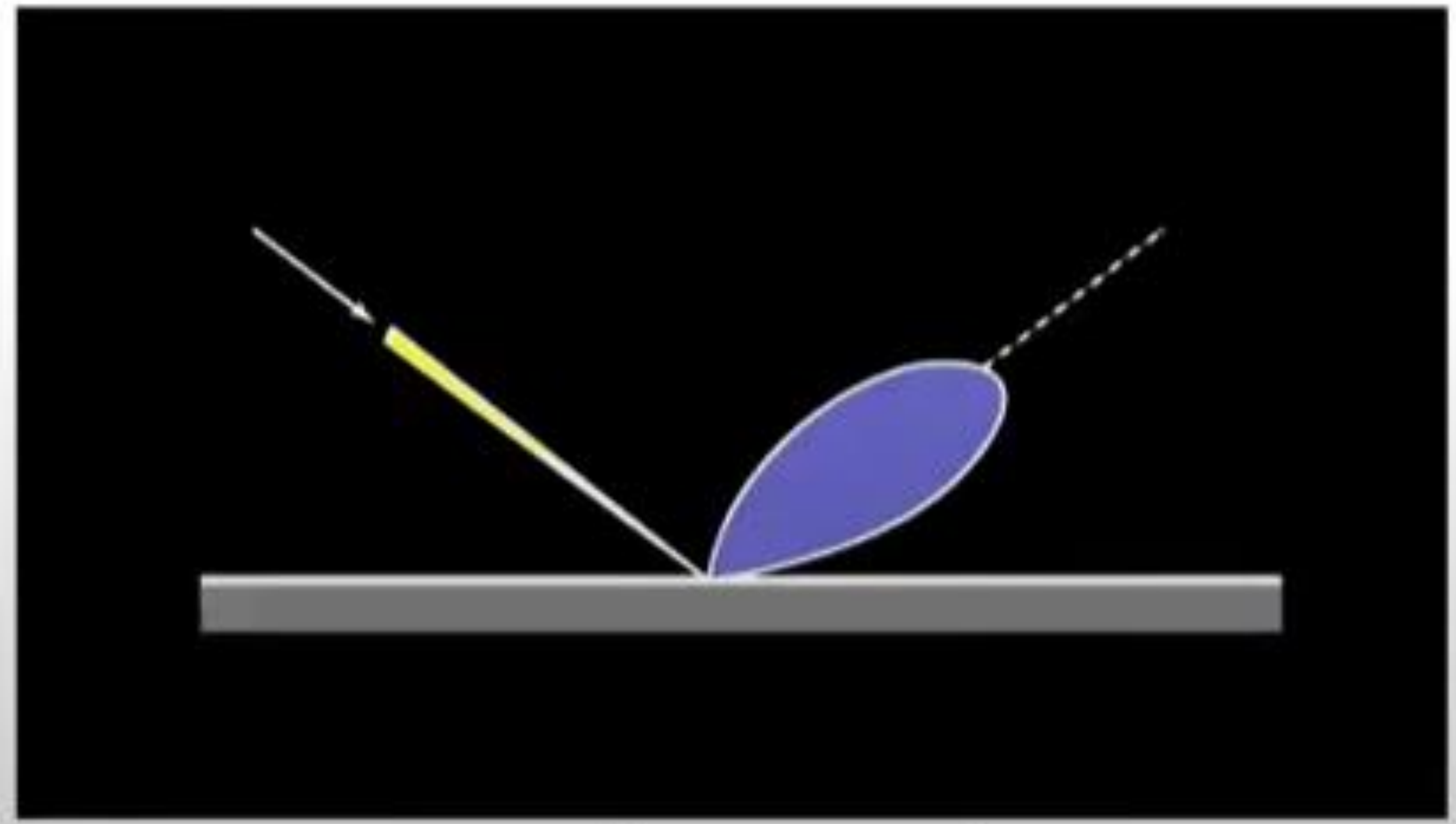




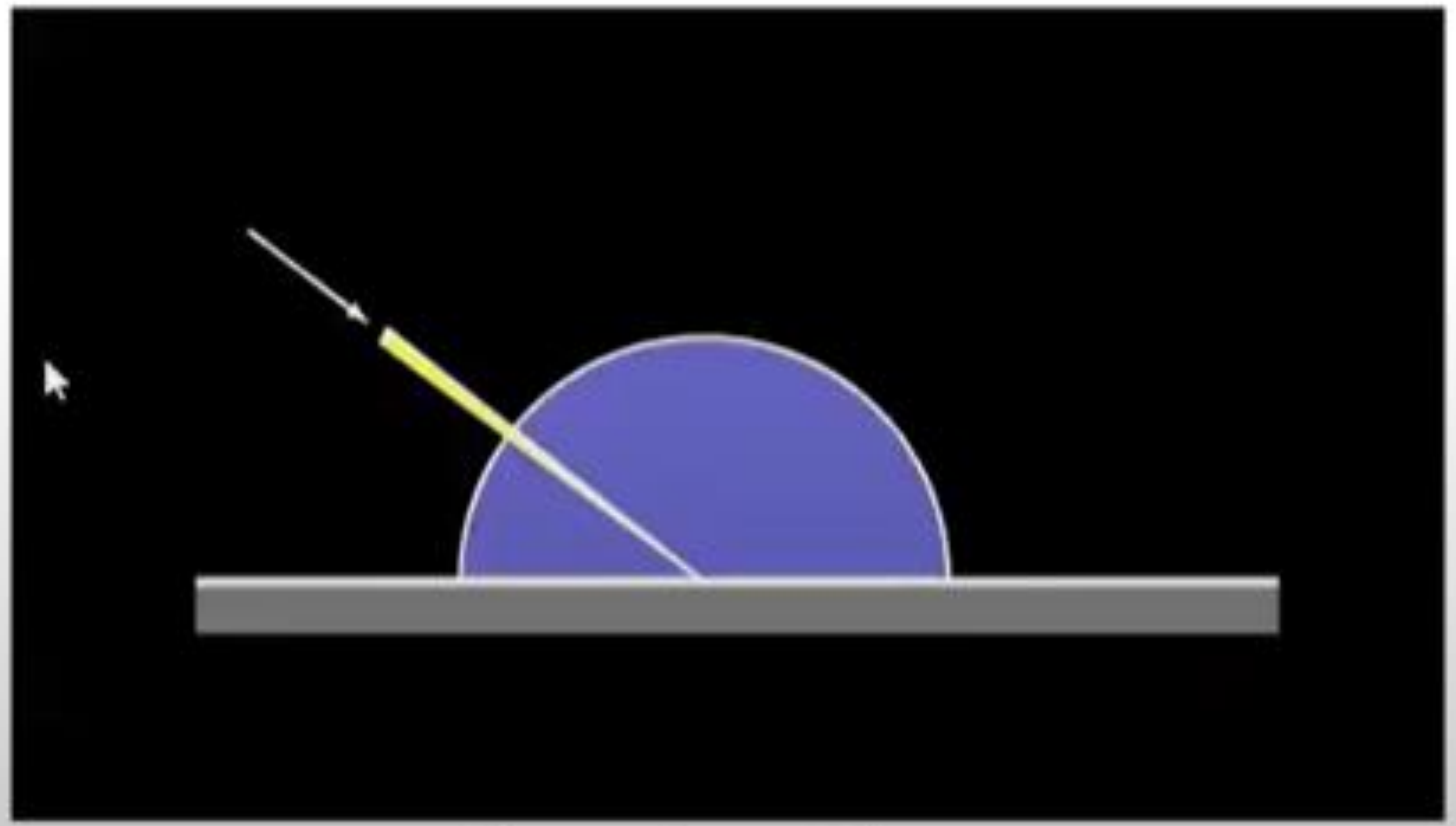
Classic reflection behavior



ideal specular (Fresnel)

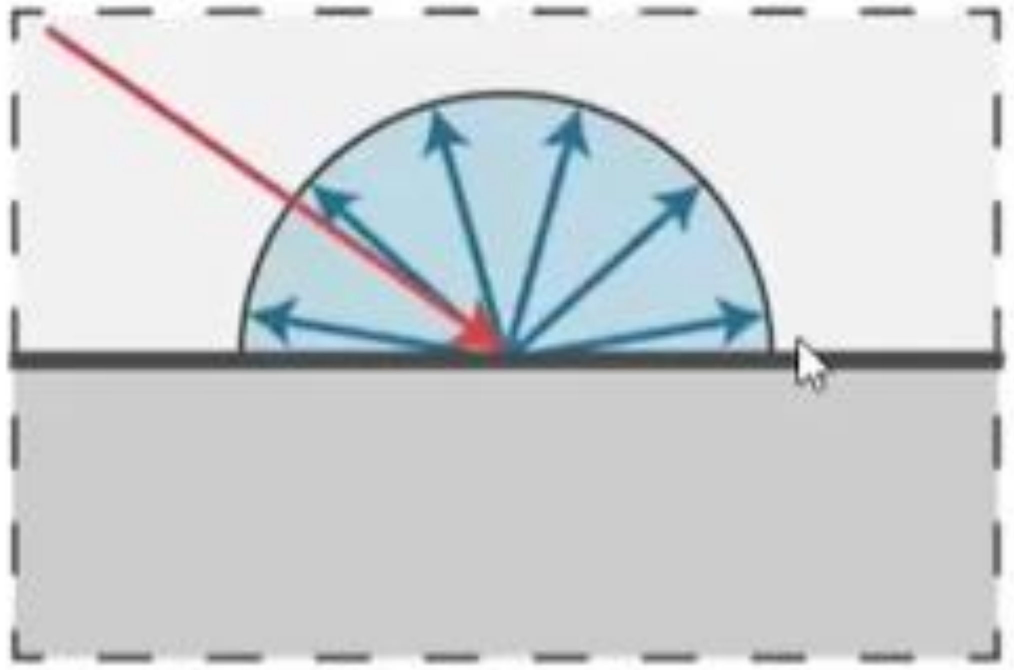


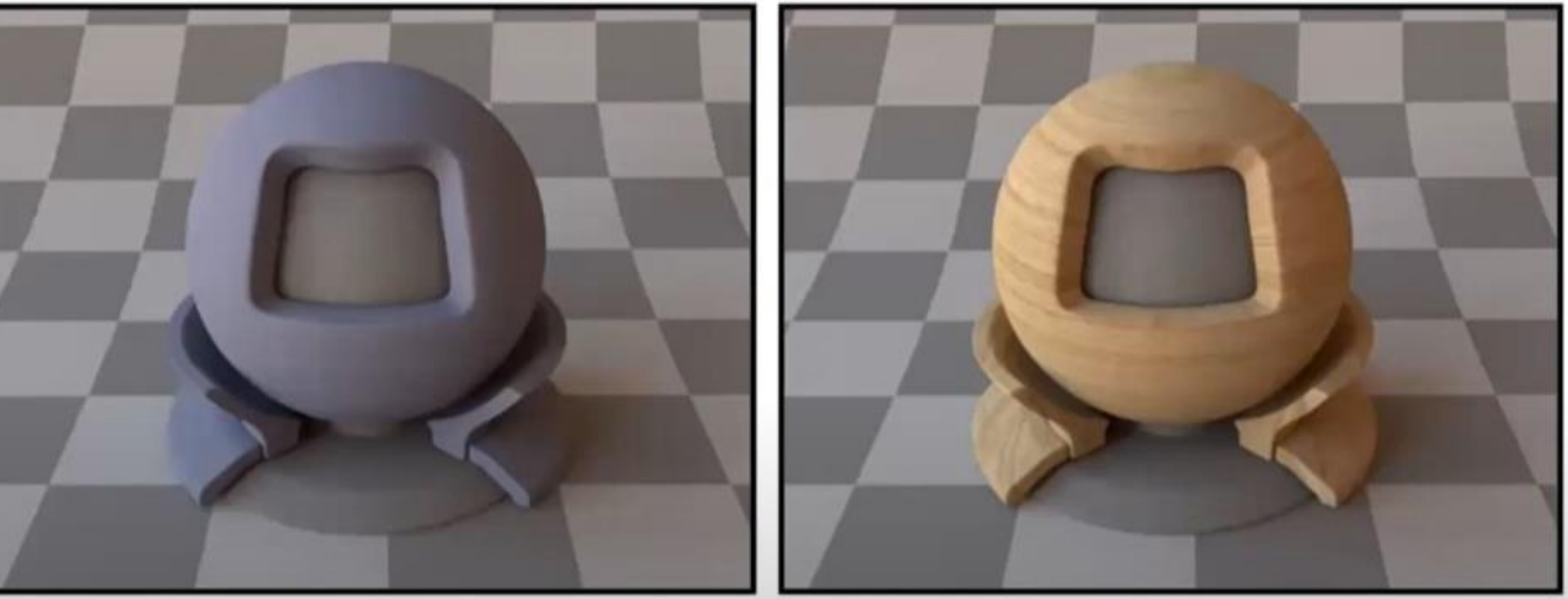
rough specular



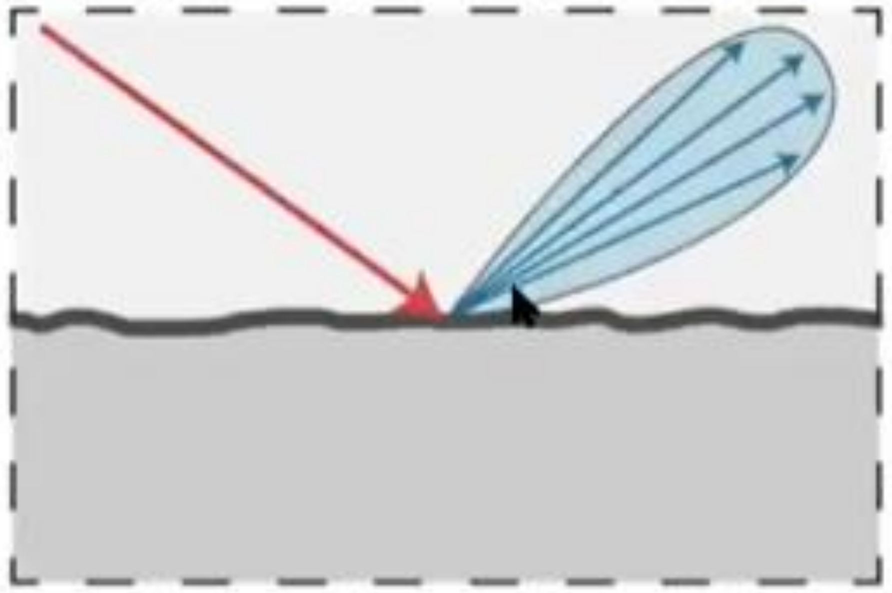
Lambertian

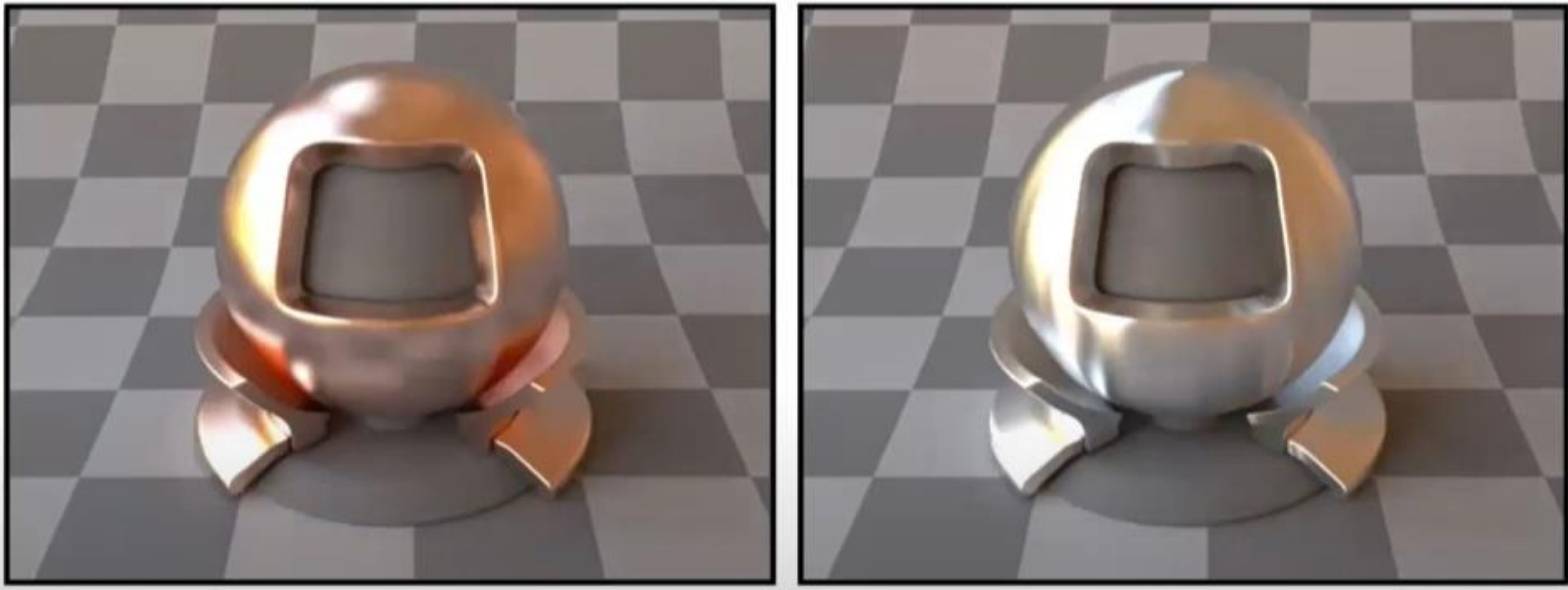
Diffuse Reflection





Glossy Material





Perceptual Observations

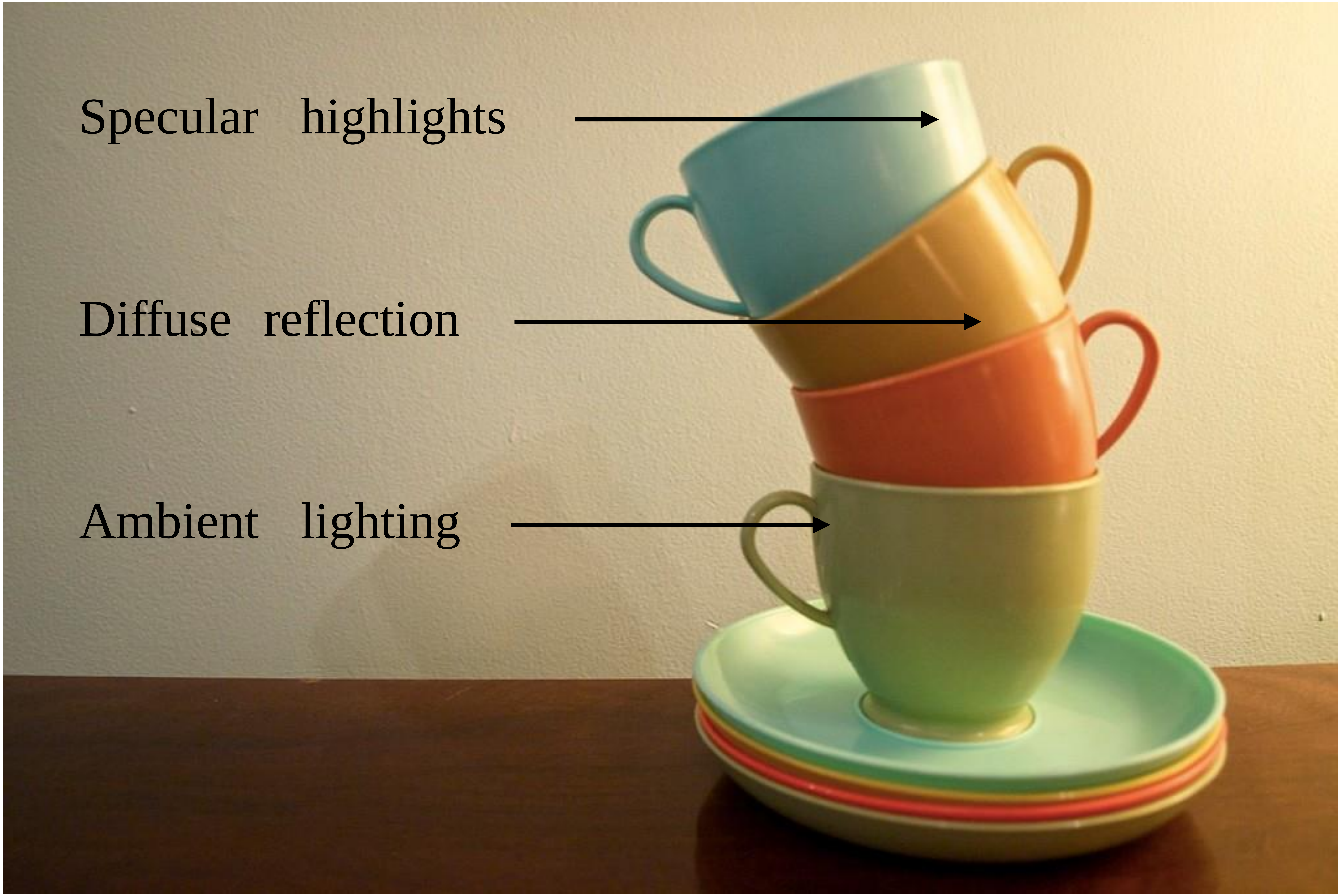


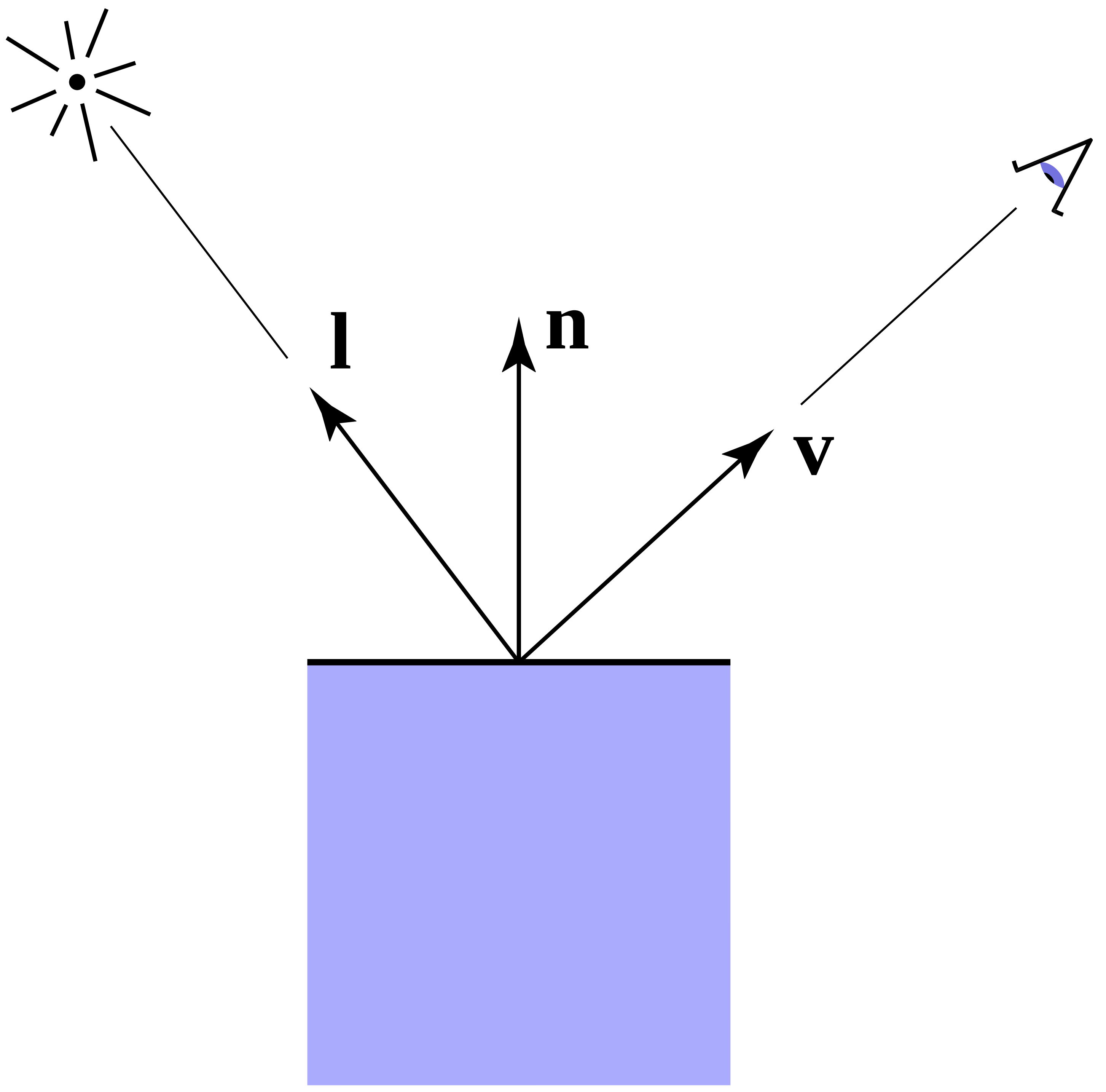
Photo credit: Jessica Andrews, flickr

Local Shading

Compute light reflected toward camera

Inputs:

- Viewer direction, v
- Surface normal, n
- Light direction, l
  (for each of many lights)
- Surface parameters
  (color, shininess, …)



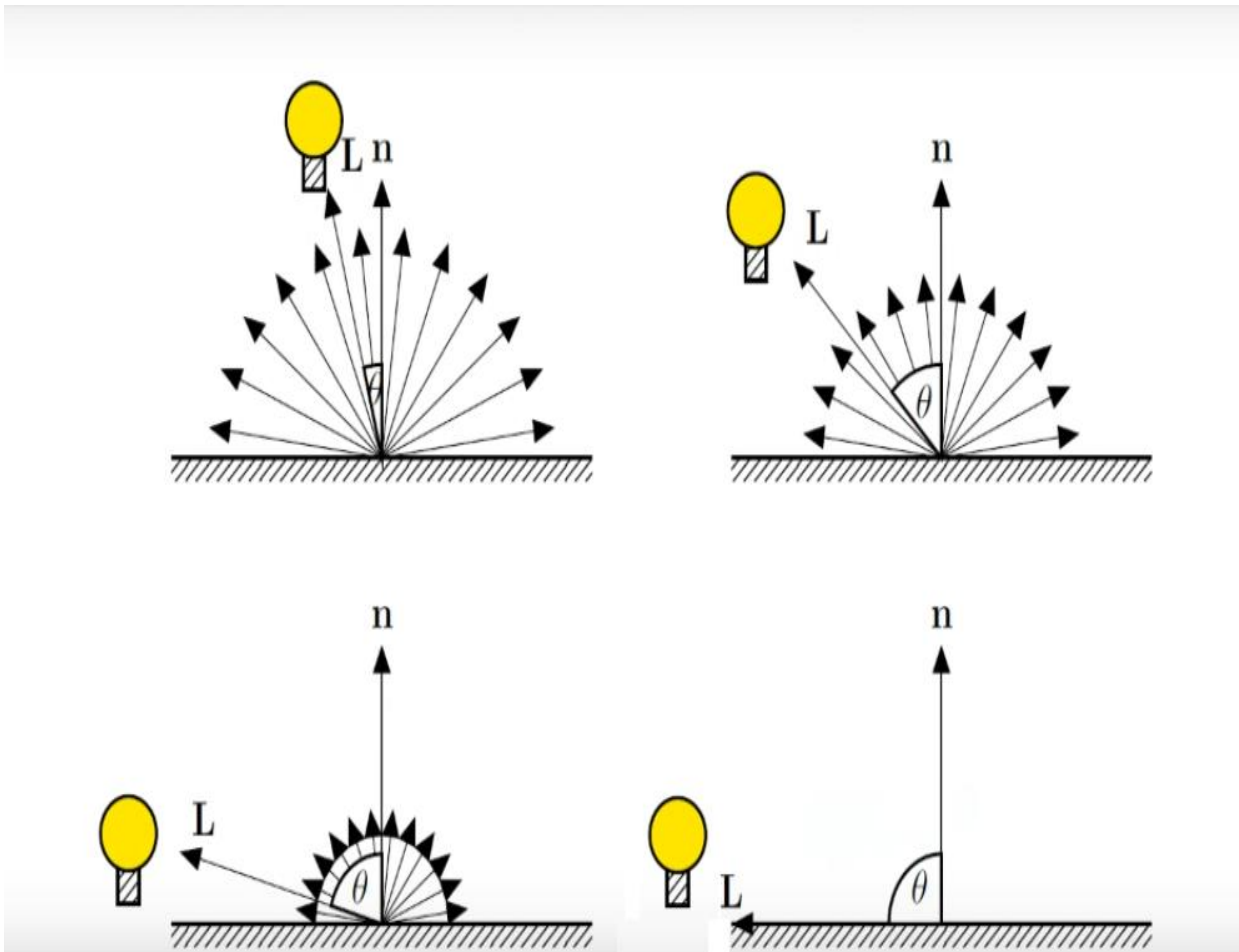
Shading Coefficient

- k_{abs} (Coeff. of absorption): Specifies how much of the incident light is absorbed. If all of the incident light is absorbed, the object appears black 1.0.
- k_{spec} (Coeff. of specular reflection) Specifies how much of the incident light is specularly reflected in one direction. For a highly reflective surface such as a mirror, the value is close to 1.0
- k_{diff} (Coeff. of diffuse reflection): Diffuse scattering occurs when the incident light is re-radiated uniformly in all directions. For a rough non-reflective surface, the value is close to 1.0.
- k_{amb} (Coeff. of ambient reflection): Specifies how much of the ambient light is reflected by the surface. Often this is same as the diffuse reflection coefficient k_{diff}.

Lambertian (Diffuse) Shading

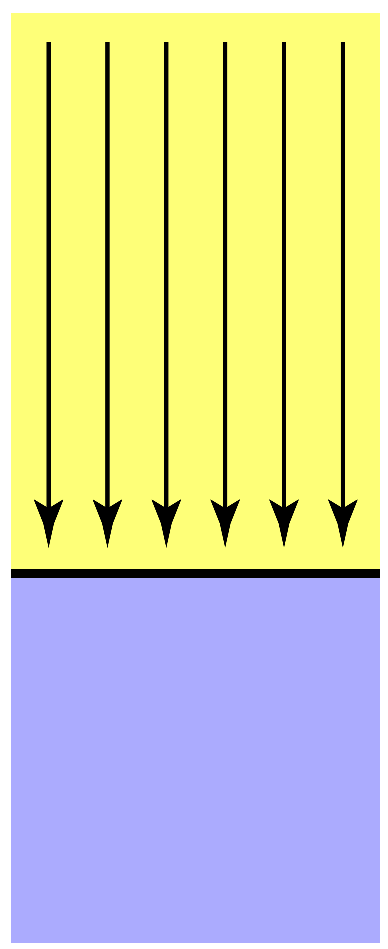
Light is scattered uniformly in all directions

- Surface color is the same for all viewing directions

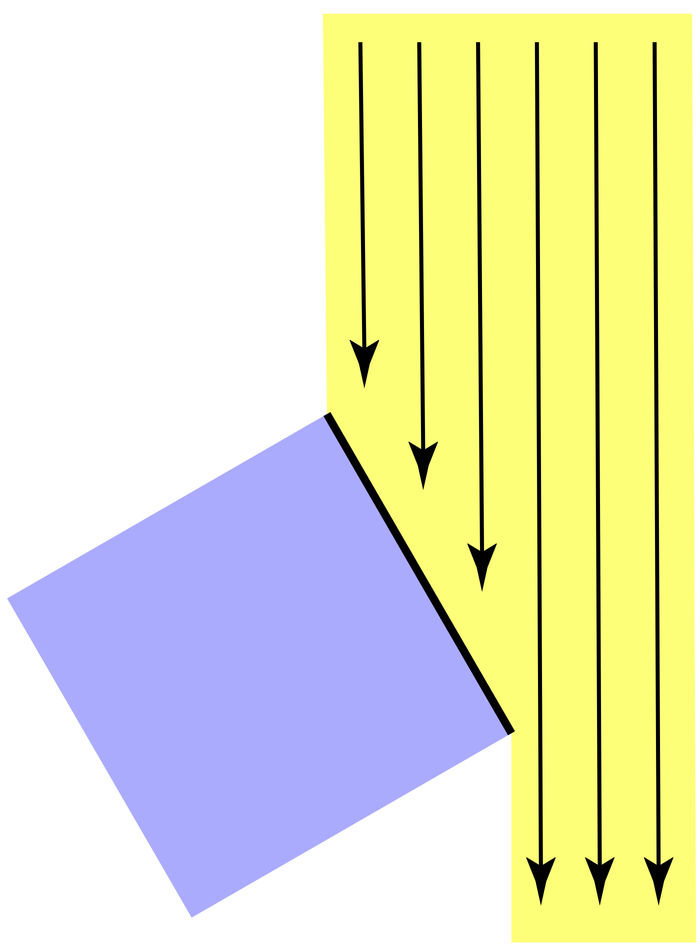


Diffuse Reflection

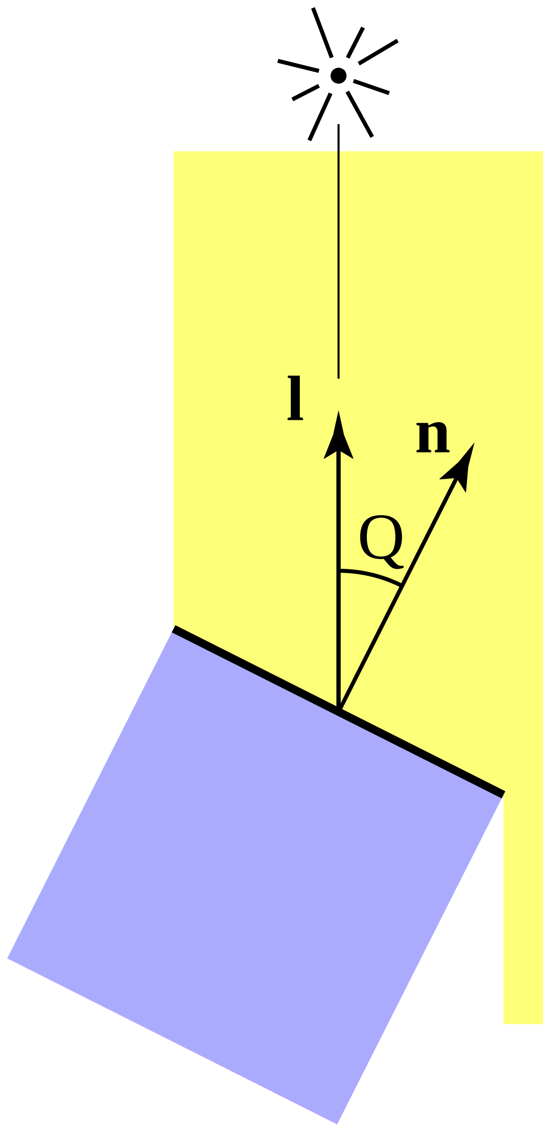
Lambert's cosine law



Top face of cube receives a certain amount of light



Top face of 60° rotated cube intercepts half the light



In general, light per unit area is proportional to cos θ = **l • n**

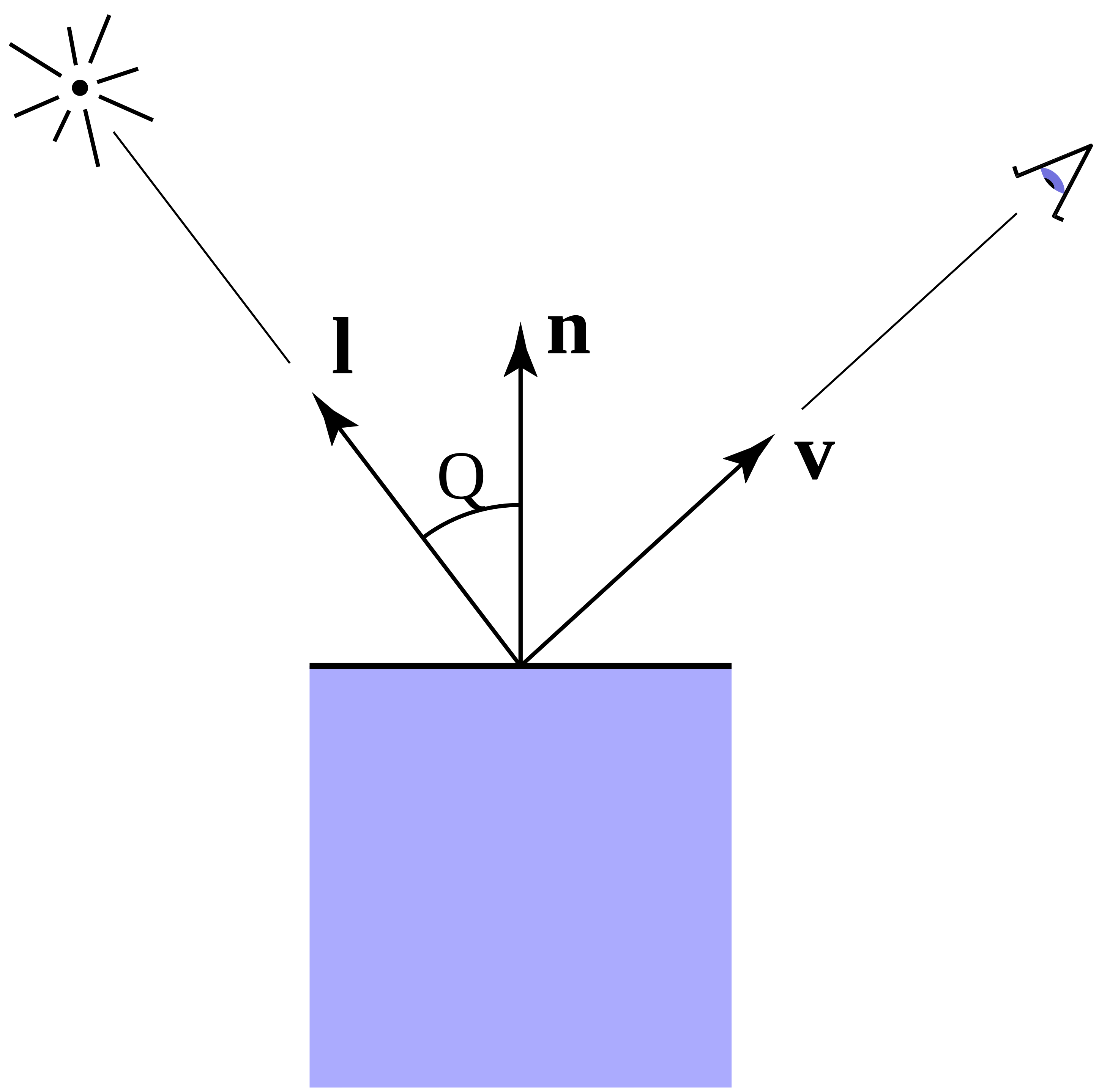
Lambertian (Diffuse) Shading

Shading independent of view direction

$$L_d = k_d \, I_l \max(\text{Cos}\Theta, 0)$$

diffuse coefficient

diffusely reflected light



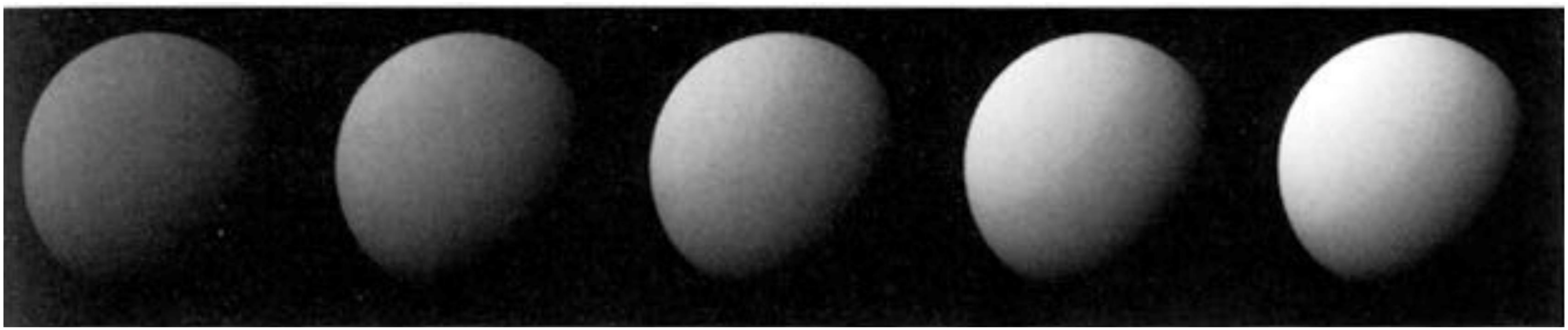
I_l : Intensity of point light source, how strong is your light

Kd: Diffuse Coefficient [0,1]

Θ: is the angle between normal vector and the light source.

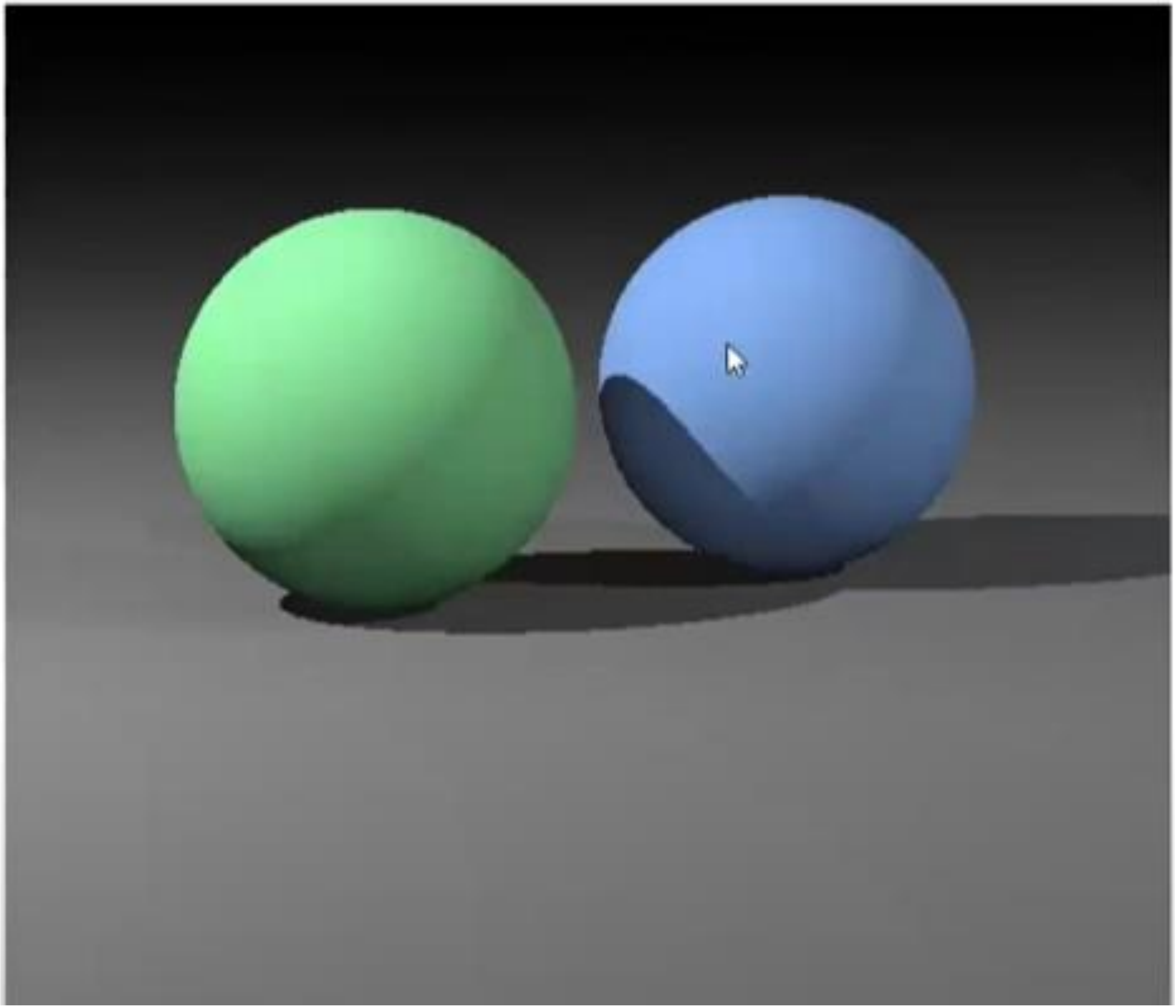
Lambertian (Diffuse) Shading

Produces matte appearance



$k_d \longrightarrow$

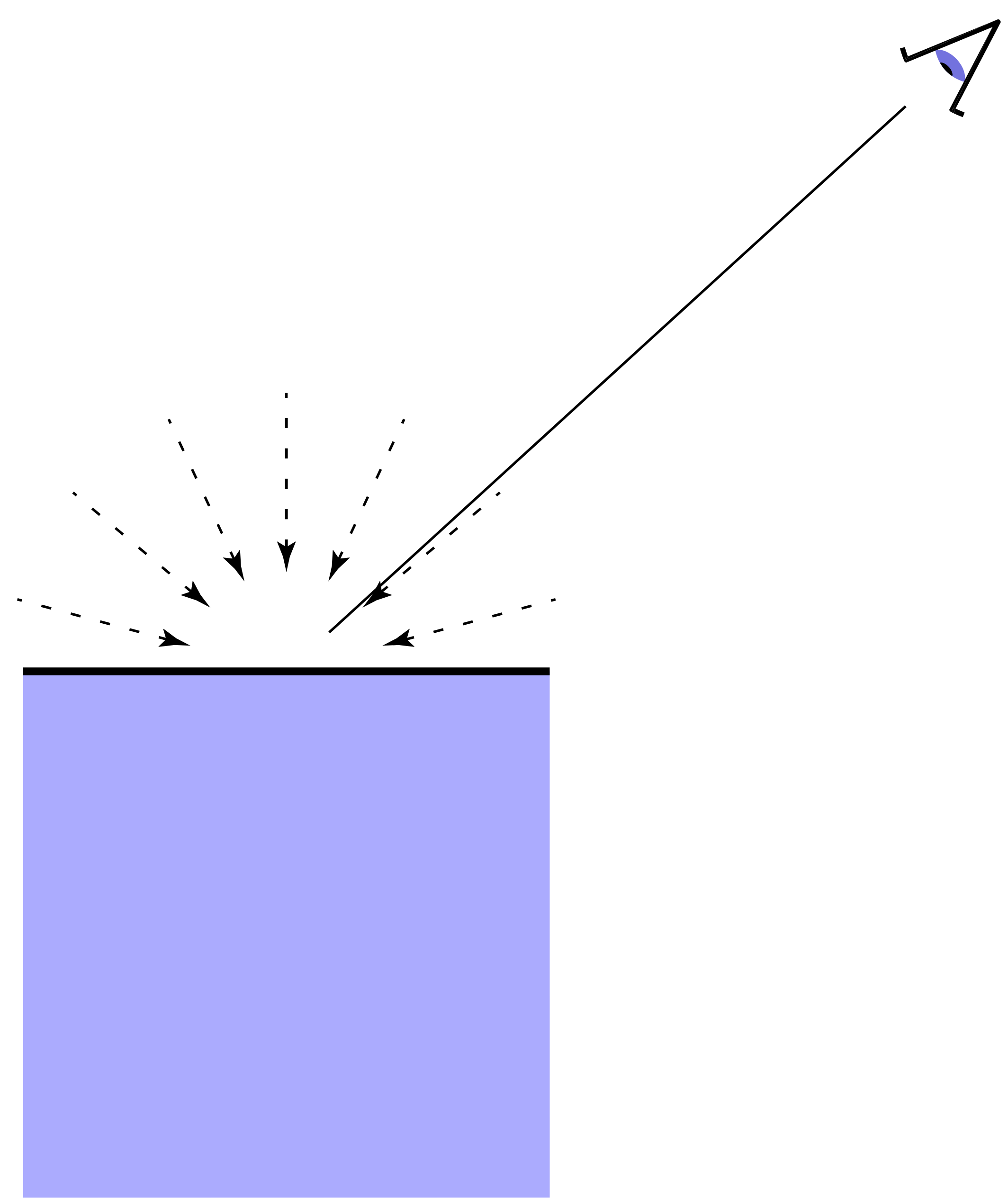
Lambertian (Diffuse) Shading



Ambient Shading

Shading that does not depend on anything

- Add constant color to account for disregarded illumination and fill in black shadows



$$L_a = k_a I_a$$

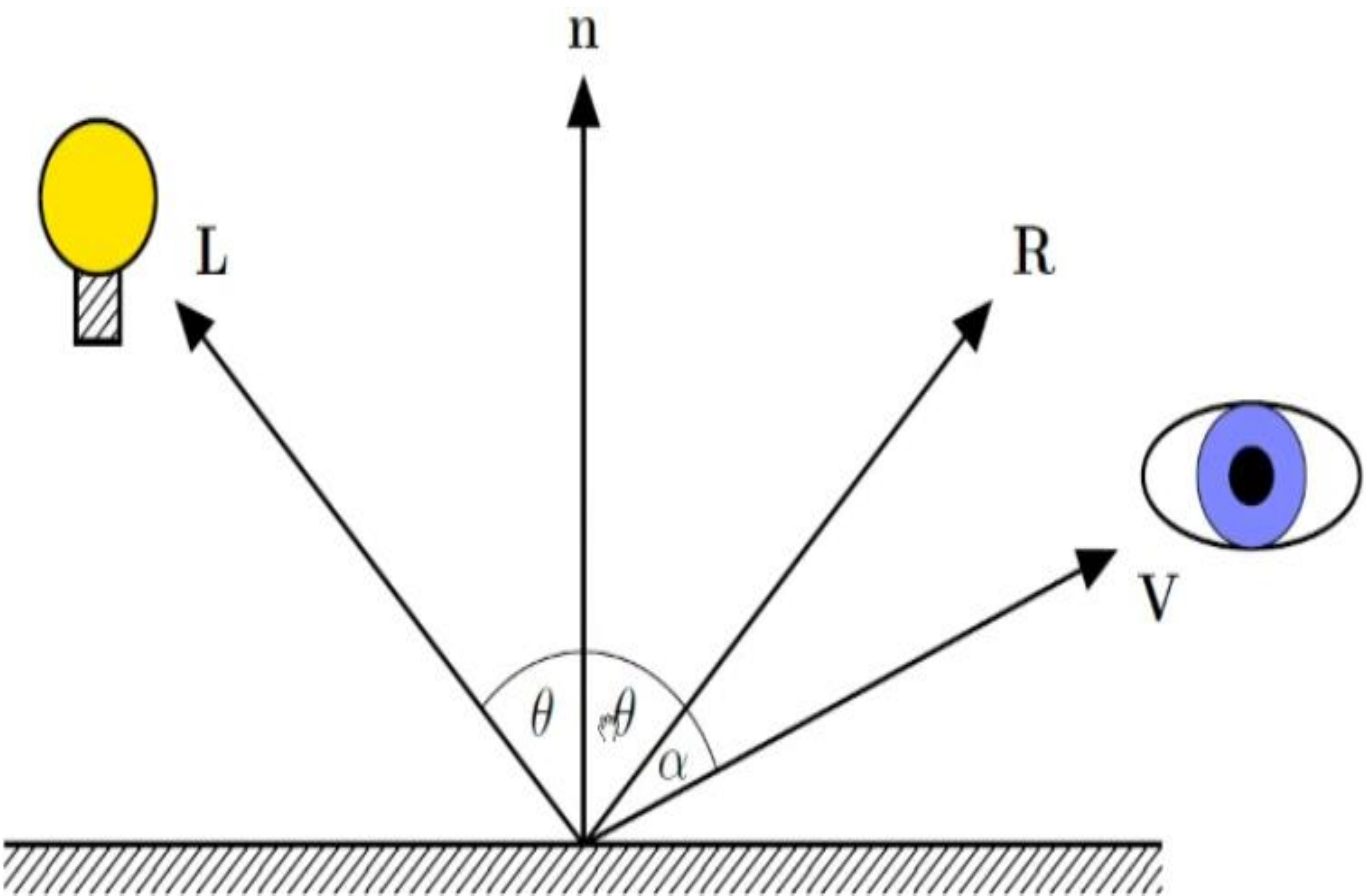
ambient coefficient

reflected ambient light

I_a : Intensity of light, how bright is your light

K_a: How much ambient light are we using

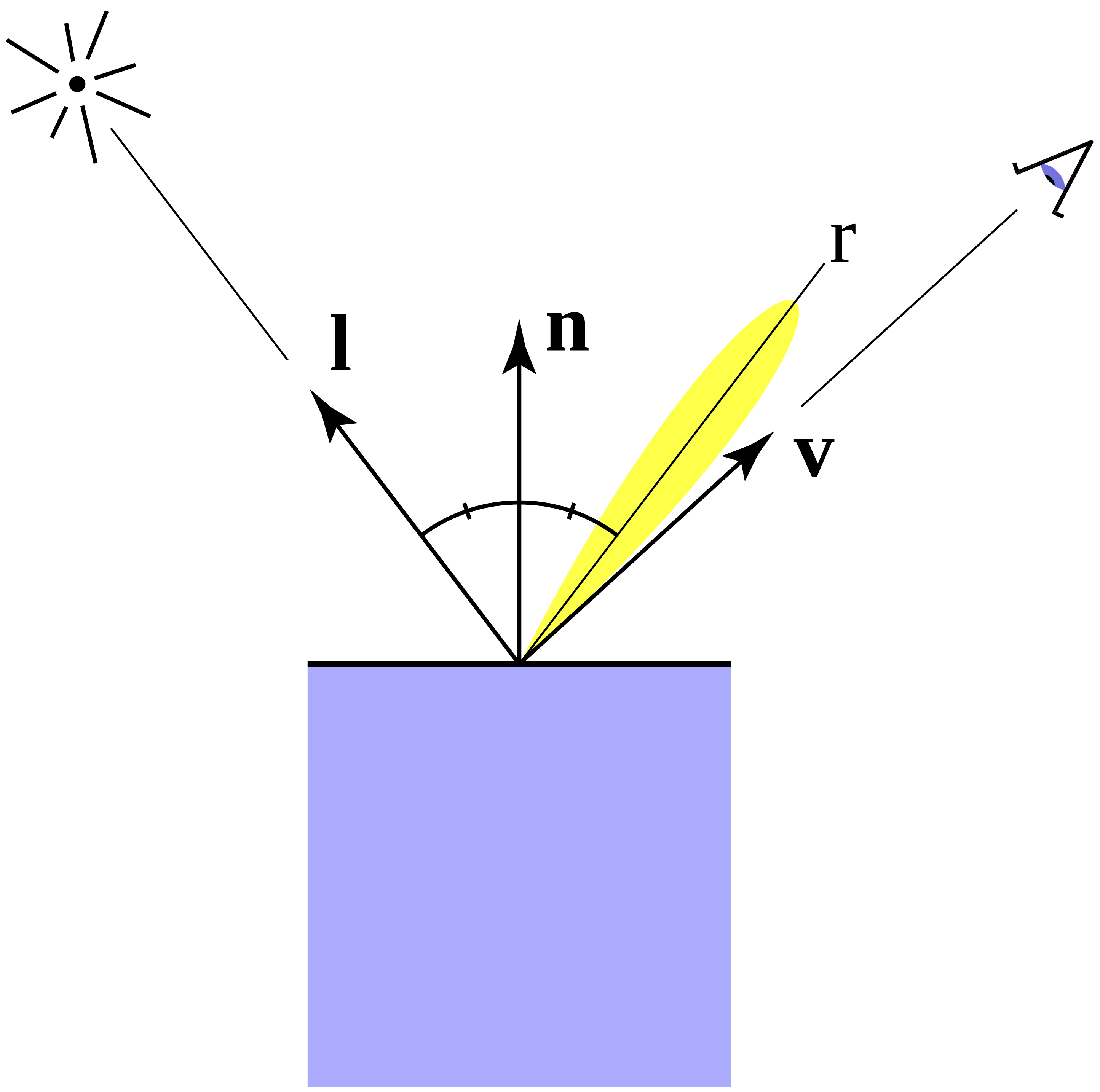
Specular Shading (Blinn-Phong)



Specular Shading (Blinn-Phong)

Intensity depends on view direction

- Bright near mirror reflection direction



$$\boldsymbol{L_s} = \mathrm{I}_s\, k_s \cos^n(\alpha)$$

– K_{spec} €[0,1] is the specular coefficient.

– n is the specular coefficient.

– α is the angle between R and V.

– The $\cos^n(\alpha)$ term determines the amount of light that is reflected.

Specular Shading (Blinn-Phong)

Increasing *n* narrows down the reflection blob

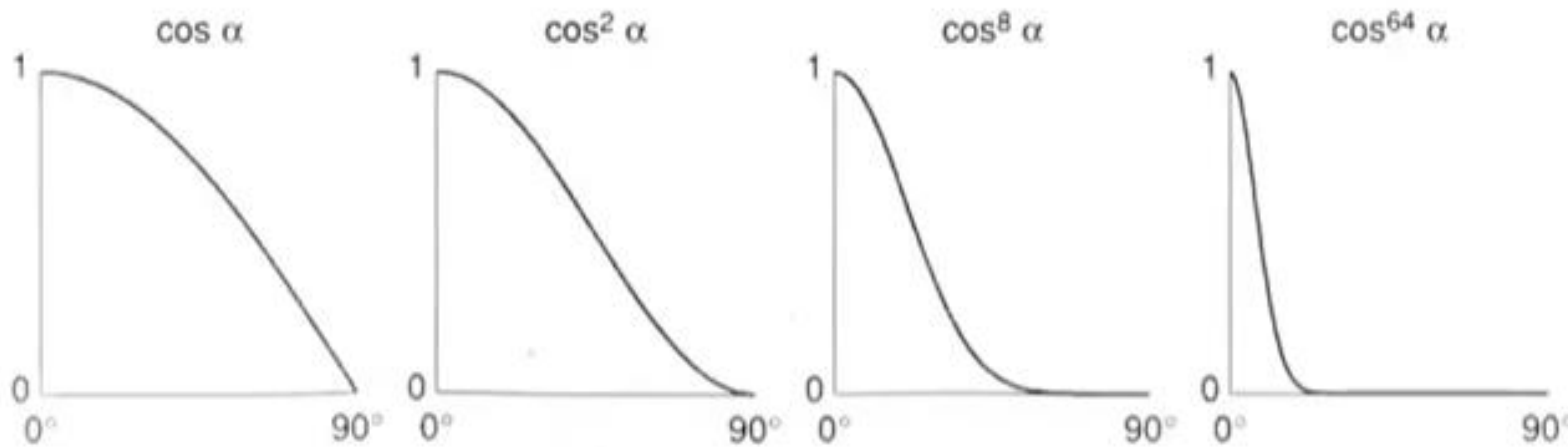
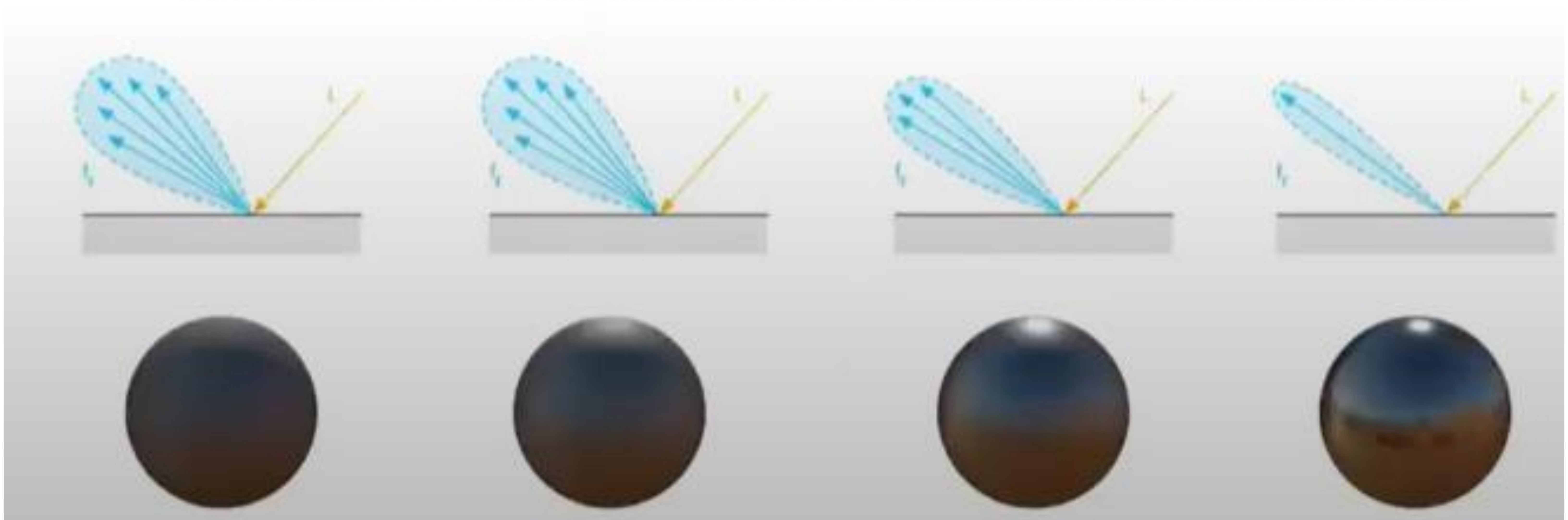
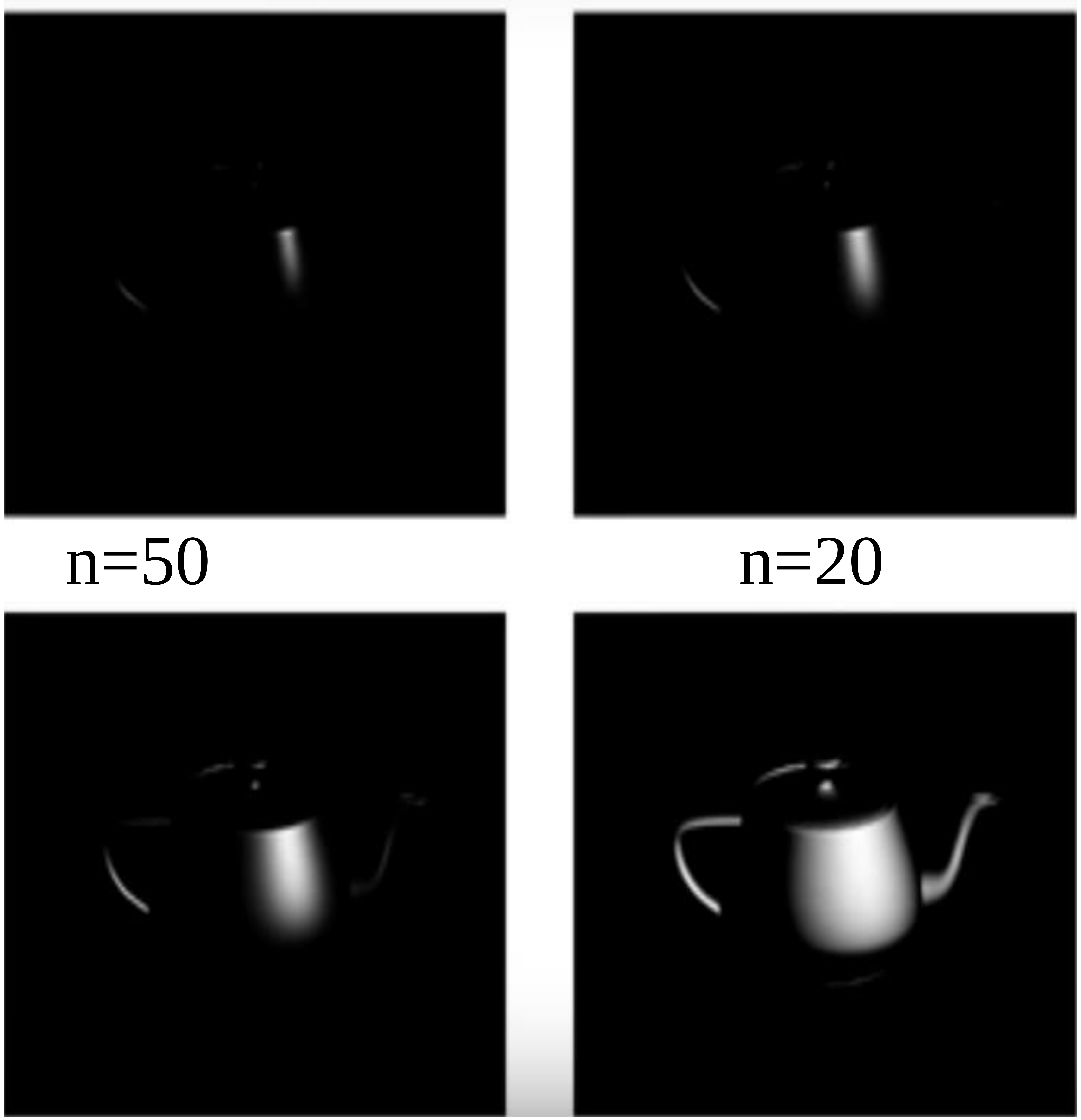


Fig. 16.9 Different values of $\cos^n \alpha$ used in the Phong illumination model.





n=50 n=20

n=5 n=1

Flat Shading

- A fast and simple method for rendering an object with polygon surfaces is ***constant intensity shading*** also called ***flat shading***
- Flat shading assumes that all pixels in a polygon are illuminated equally.
- The Phong reflection model is calculated using the normal vector for the polygon and all pixels assume same illumination intensity.

Flat Shading

- Flat shading of polygon provides an accurate rendering for an object if all the following assumptions are valid.
  - The object is a polyhedron and is not an approximation of an object with a curved surface.
  - All light sources illuminating the object are sufficiently far from the surface so that N.L and the attenuation are constant over the surface.
  - The viewing position is sufficiently far from the surface so that V.R is constant over the surface.

Solution- Use more polygons

Gouraud Shading

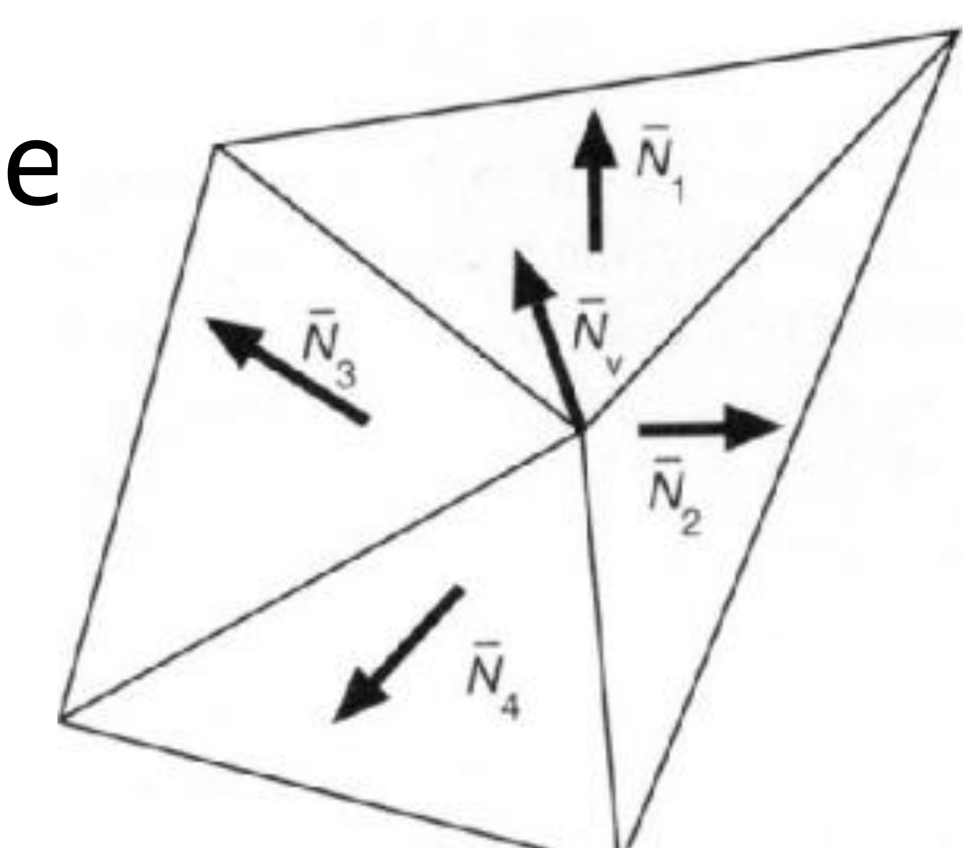
- Gouraud Shading calculates the Phong reflection model at the polygon vertices and then linearly interpolates the intensities of the vertex pixels across the pixels of the polygon.
- The normal vector at the vertices is calculated by averaging the normals for all polygons that shades the vertex.
- Intensity value for each polygon are matched with the values of adjacent polygons along the common edges.
- Eliminates the intensity discontinuities that can occur in flat shading.

Gouraud Shading

- Each polygon surface is rendered with Gouraud shading by performing the following calculations:
  - Determine the average unit normal vector at each polygon vertex.

$$N_v = \frac{\sum_i N_i}{\| \sum_i N_i \|}$$

  - Apply an illumination model to each vertex to calculate the vertex intensity.
  - Linearly interpolate vertex intensities over the the polygon.



Problem of Gouraud Shading

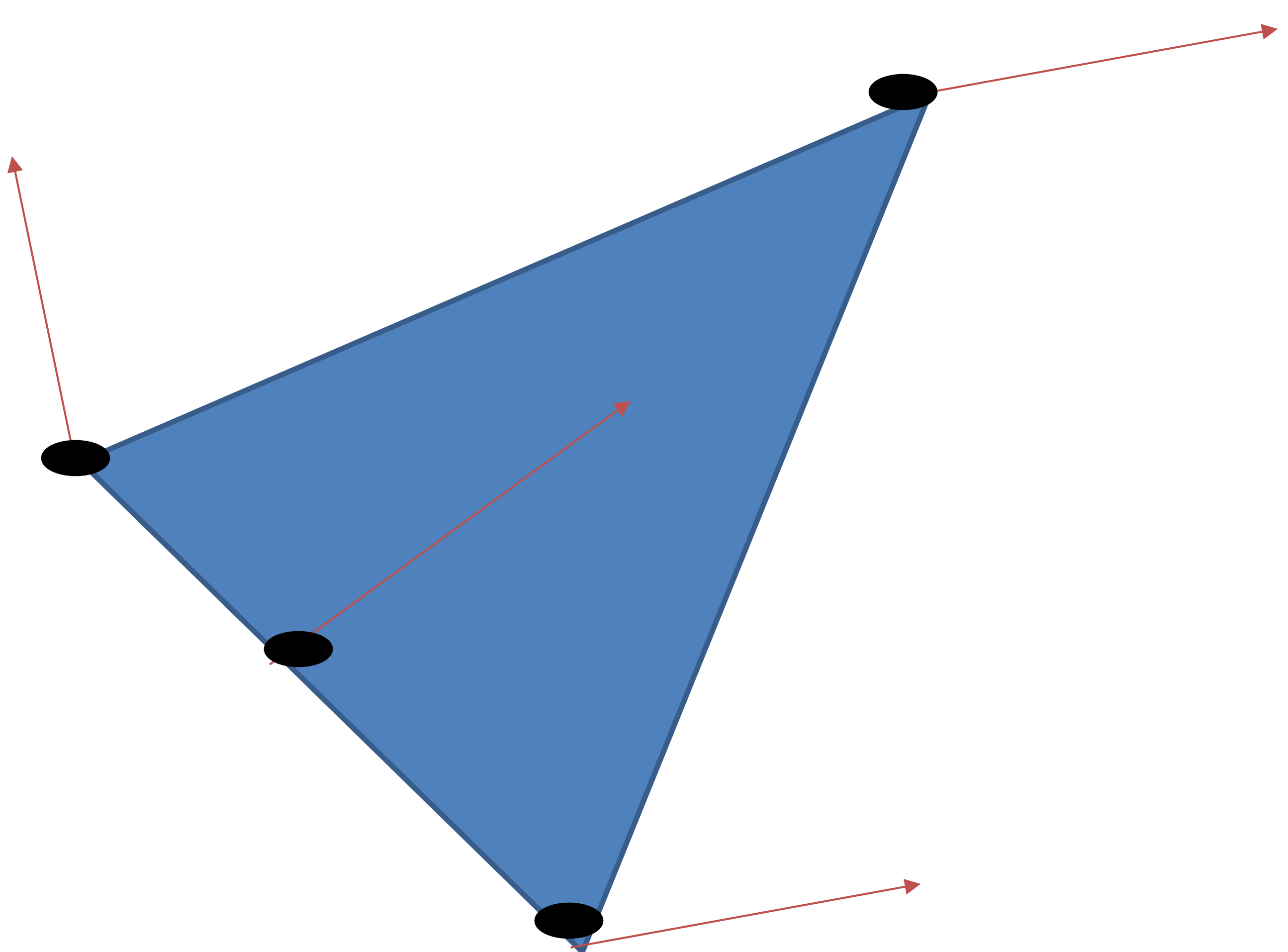
- Highlights on the surface are sometimes displayed with anomalous shapes.
- Linear intensity interpolation can cause bright or dark intensity streaks on the surface.

Phong Shading

- Interpolates the normal at normal vectors of the vertex pixels across the polygon.
- The normal at the vertices are calculated by averaging the normal for all polygons that shared a vertex.
- Since each pixel will have a normal associated with it. Phong’s reflection model can be applied for each pixel.
- It displays more realistic highlights on a surface and greatly reduces Mach band effect.

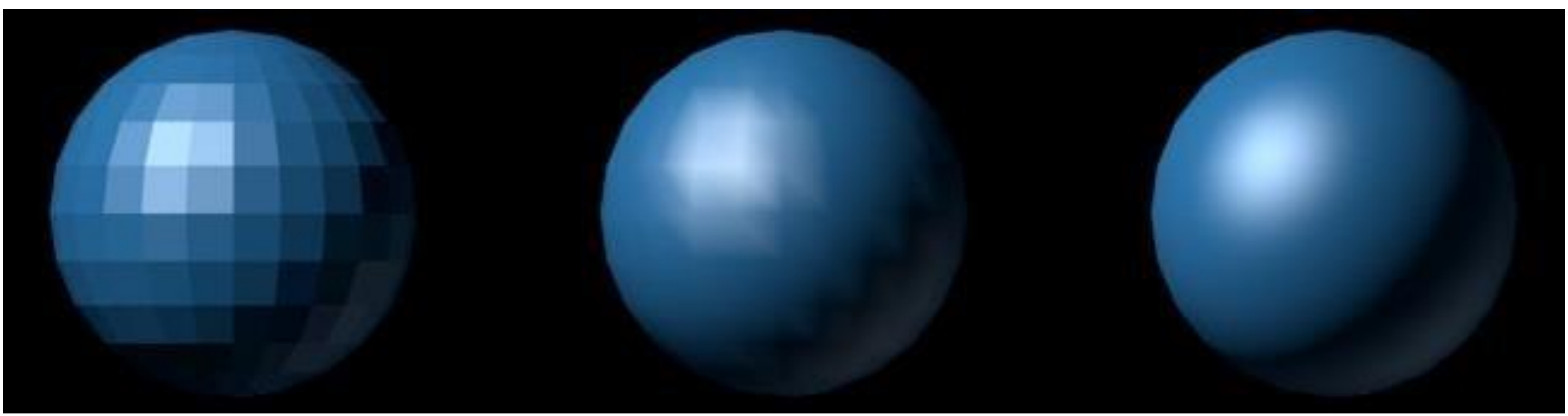
Phong Shading

- A polygon is rendered using Phong shading by carrying out the following steps:
  - –Determine the average unit normal vector at each vertex.
  - –Linearly interpolate the vertex normal over the surface of the polygon.
  - –Apply an illumination model along each scan line to calculate the projected pixel intensities for the surface points.



Flat vs. Gouraud vs. Phong (Shading)

- Flat: use the actual normal, i.e. the real geometry (you can see the triangles)
- Gouraud: use averaged vertex normals to compute a color at each vertex; then, interpolate vertex colors to the triangle interior
- Phong: use averaged vertex normals and interpolate those normals to the triangle interior for shading (smooth shading)

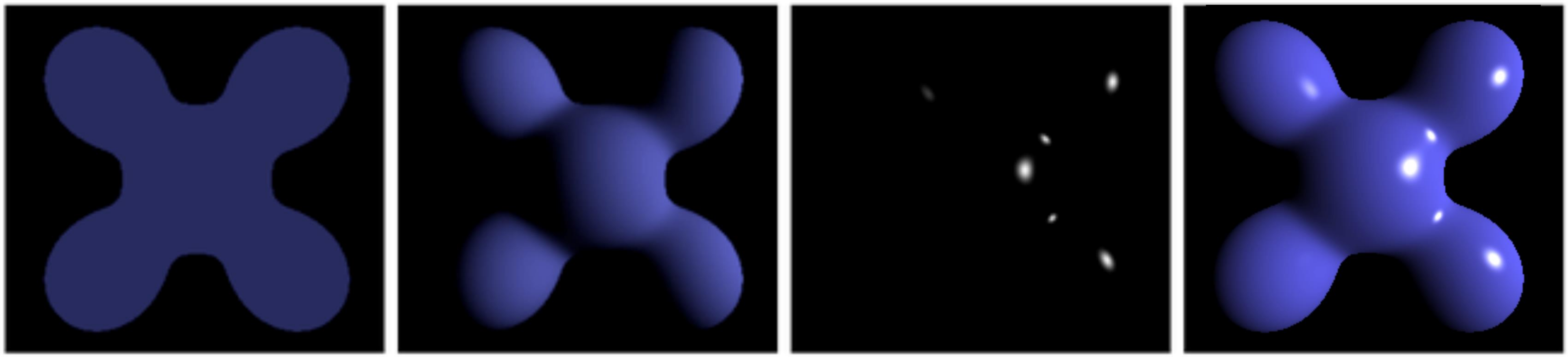


flat Gouraud

Flat vs. Gouraud vs. Phong (Shading)

Feature	**Flat Shading**	**Gouraud Shading**	**Phong Shading**
Lighting Calculation	Once per **polygon (face)**	At each **vertex**	At each **pixel (fragment)**
Interpolation	No interpolation — same color across the face	Interpolates **vertex colors** across the face	Interpolates **normals** across the face, then computes lighting per pixel
Visual Quality	Faceted, low realism	Smoother than flat, but may miss specular highlights	Very smooth and realistic, good specular highlights
Performance	Fastest (least computation)	Medium performance	Slowest (most computation)
Used For	Low-poly styles, preview rendering	Early 3D games, medium-quality renders	High-quality rendering, modern graphics

Blinn-Phong Reflection Model



Ambient + Diffuse + Specular = Phong Reflection

$$L = L_a + L_d + L_s$$

$$= k_a I_a + k_d I_l \max(n.l, 0) + I_s k_s (V.R)^n$$

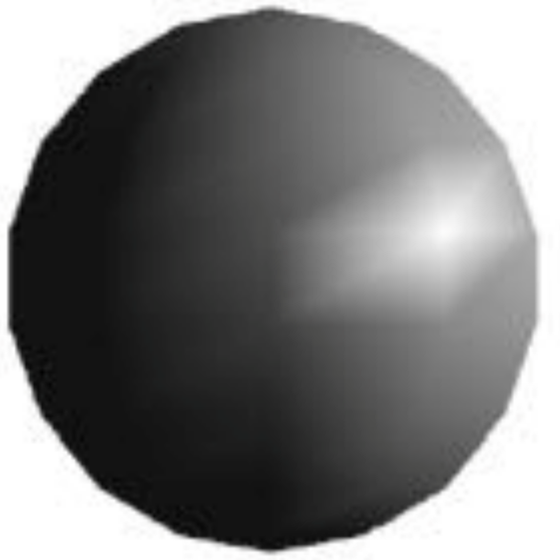
Shading Triangle Meshes

Shading Frequency: Triangle, Vertex or Pixel

Shade each triangle (flat shading)

- Triangle face is flat — one normal vector
- Not good for smooth surfaces

Shade each vertex ("Gouraud" shading)

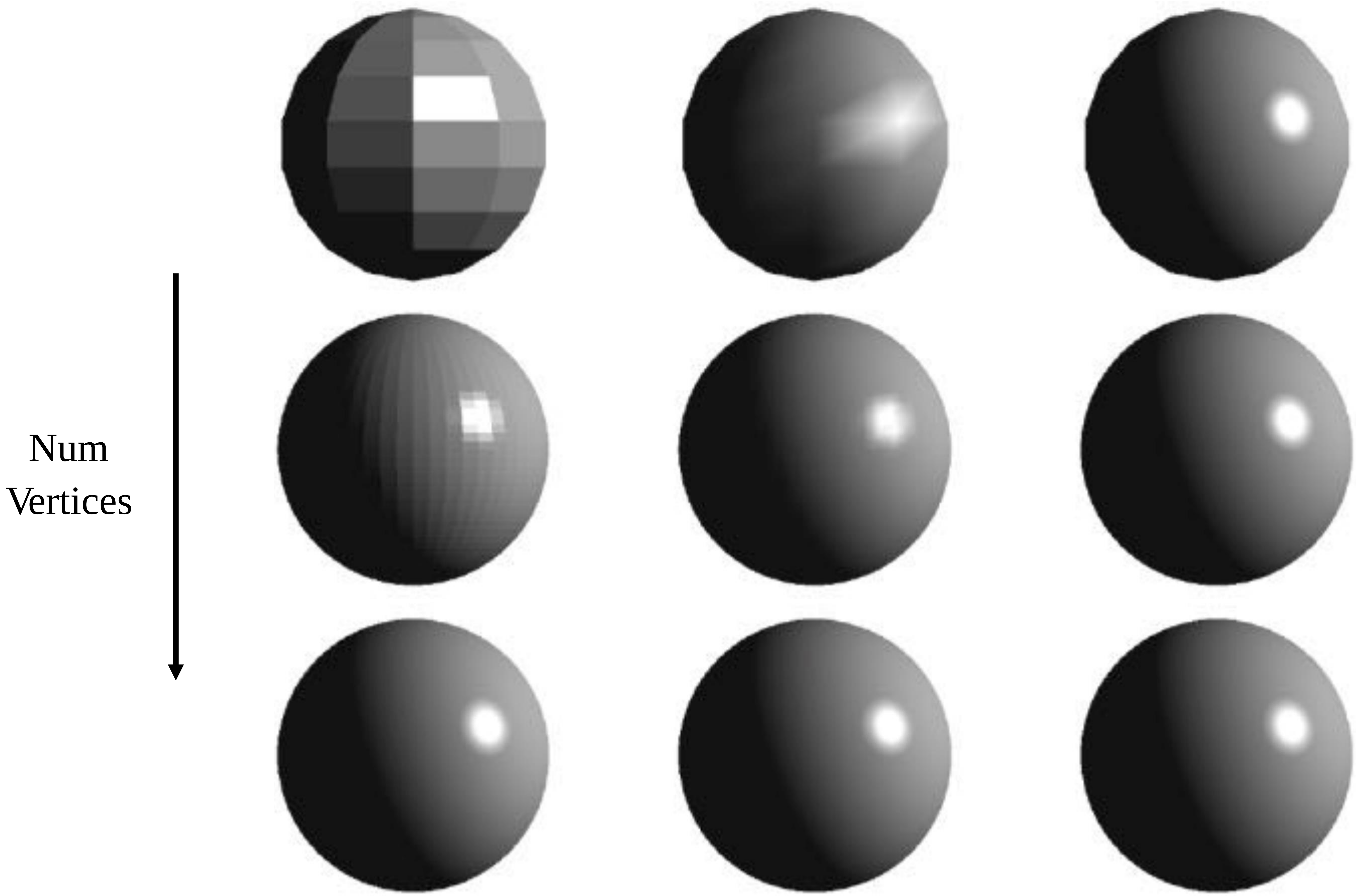


- Interpolate colors from vertices across triangle
- Each vertex has a normal vector

Shade each pixel ("Phong" shading)

- Interpolate normal vectors across each triangle
- Compute full shading model at each pixel

Shading Frequency: Face, Vertex or Pixel



Shading freq. :	Face	Vertex	Pixel
Shading type :	Flat	Gouraud	Phong

Image credit: Happyman, http://cg2010studio.com/