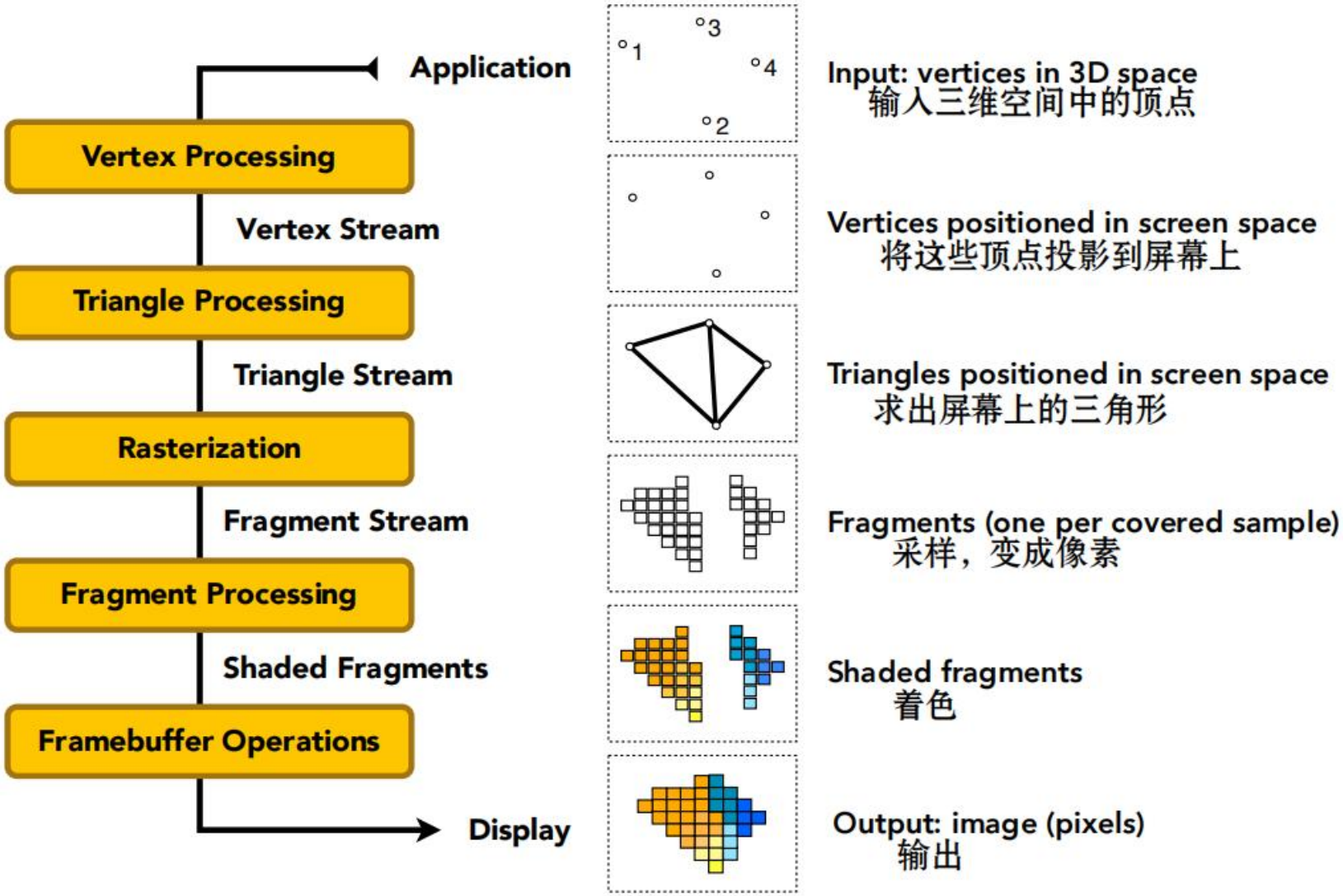
第五章 光照模型和纹理映射

真实感图形绘制

如何在显示器上绘制真实感图形?

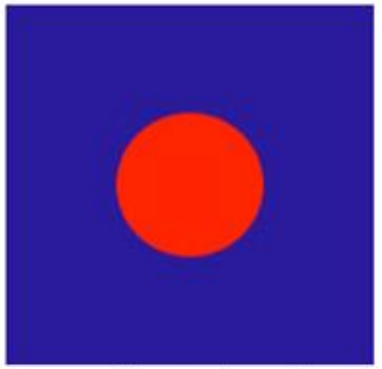
- **三维造型**：用数学方法建立是三维场景的几何描述
- **透视变换**：将三维几何描述转换为二维透视图
- **光栅化**：确定场景中所有可见面
- **着色**：计算场景中可见面的颜色

Graphics Pipeline



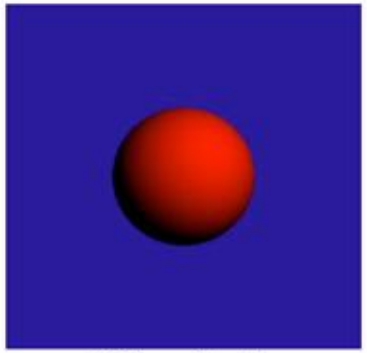
Shading（着色）

- Suppose we build a model of a sphere using many polygons and color it with glColor. We get something like



object color only

- But we want

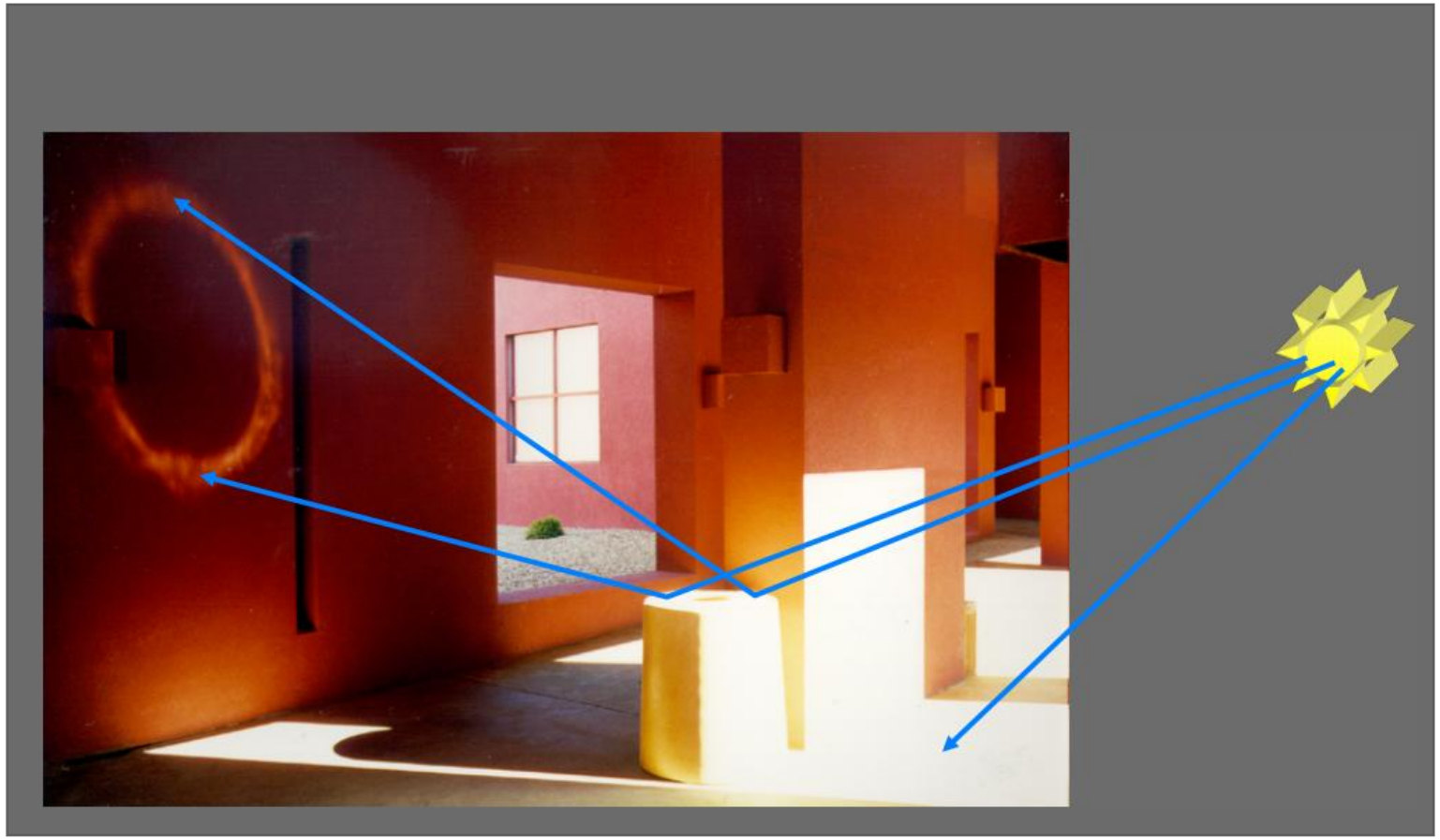


Diffuse Shading

Why we need shading?

- Input for realistic rendering
  - Geometry, Lighting and Materials
- Material appearance
  - Intensity and shape of highlights
  - Glossiness（光泽）
  - Color
  - Spatial variation, i.e., texture

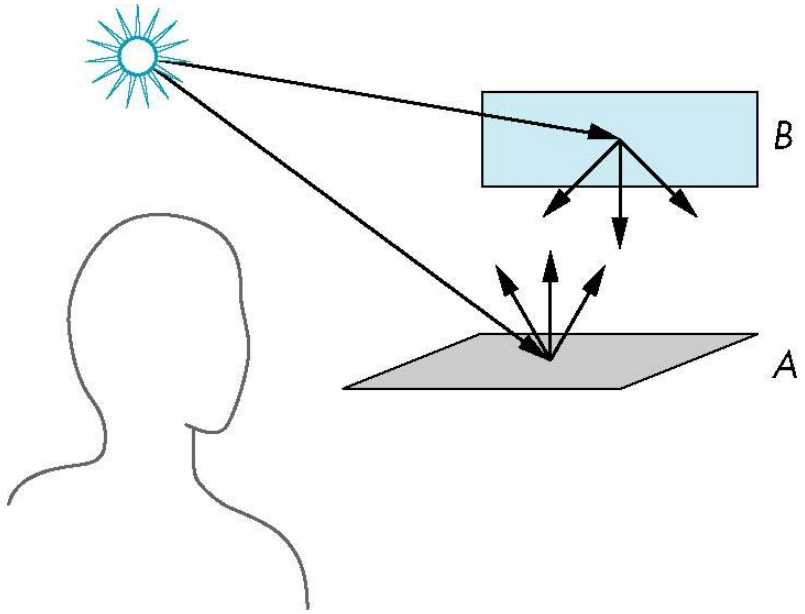
Why we need shading?



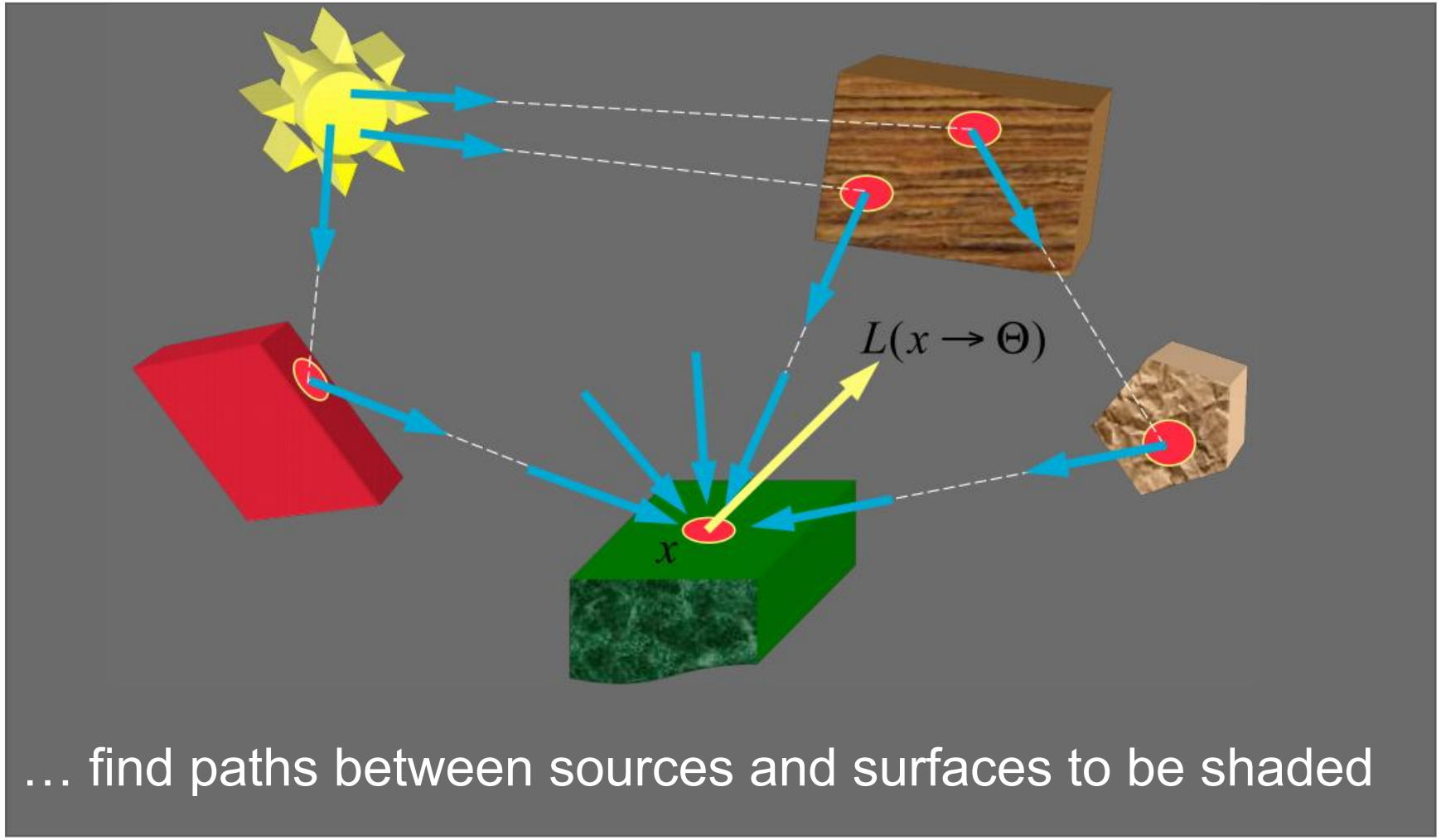
Light-material interactions cause each point to have a different color or shade

Scattering

- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered light strikes A
  - and so on



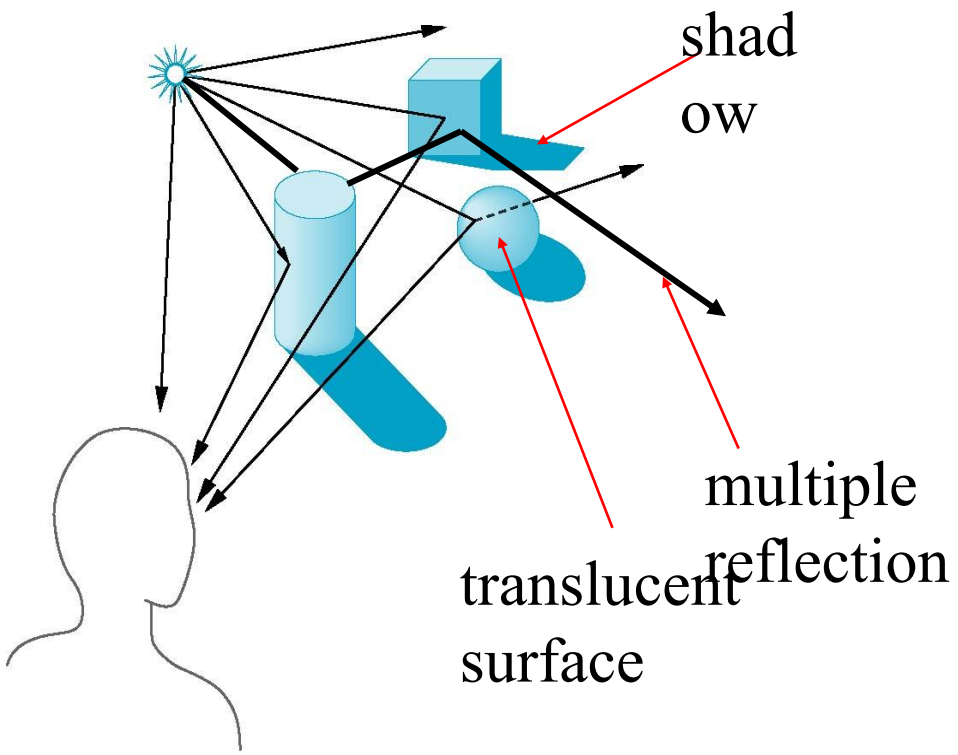
Global Illumination



Global Illumination

- The infinite scattering and absorption of light can be described by the rendering equation
  - Cannot be solved in general
  - **Ray tracing** is a special case

  for perfectly reflecting surfaces.

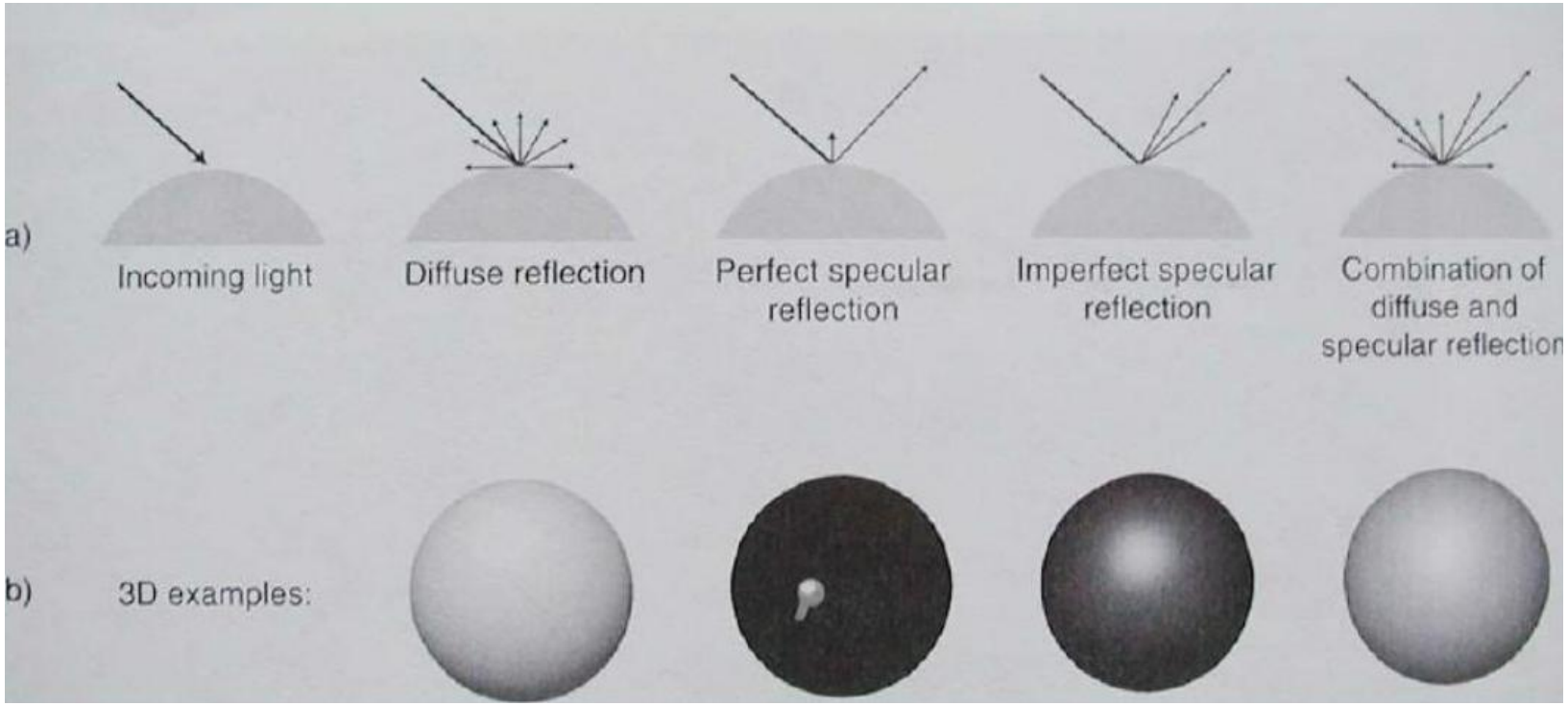


Let’s first use something simple

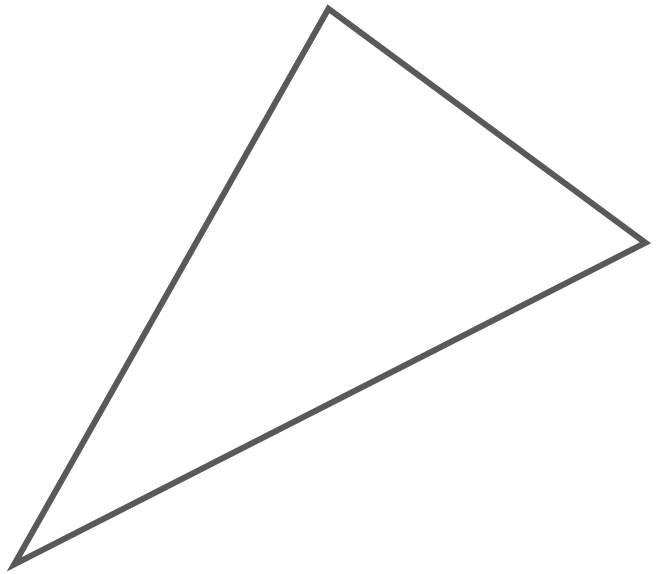
- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with pipeline model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things “**look right**”
  - Exist many techniques for approximating global effects

Local Illumination

- Lighting
  - Only consider the lighting from different “light sources”
  - Ignore the influence of other objects
- Material
  - BRDF function



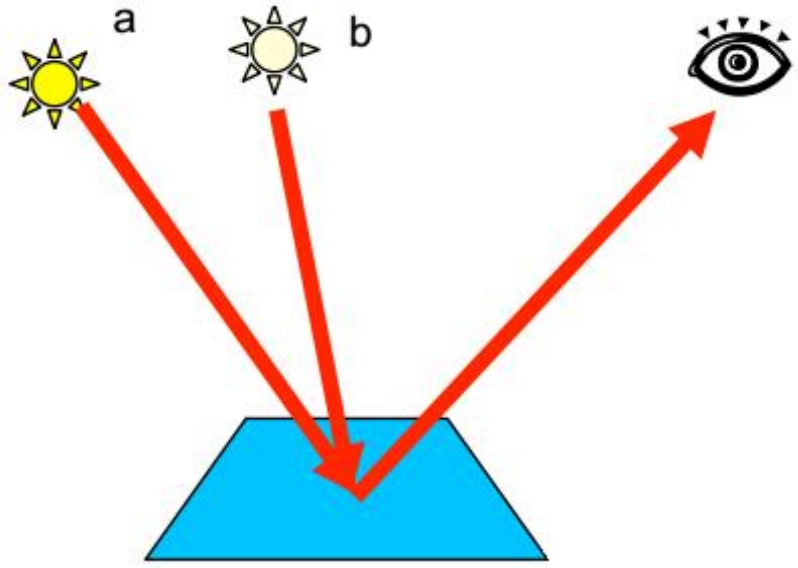
5.1 光照模型



Light Sources

- Today, we only consider point light sources
  - Thus we don’t need to care about solid angles
- For multiple light sources, use linearity
  - We can add the solutions for two light sources
  - I(a+b) = I(a) + I(b)

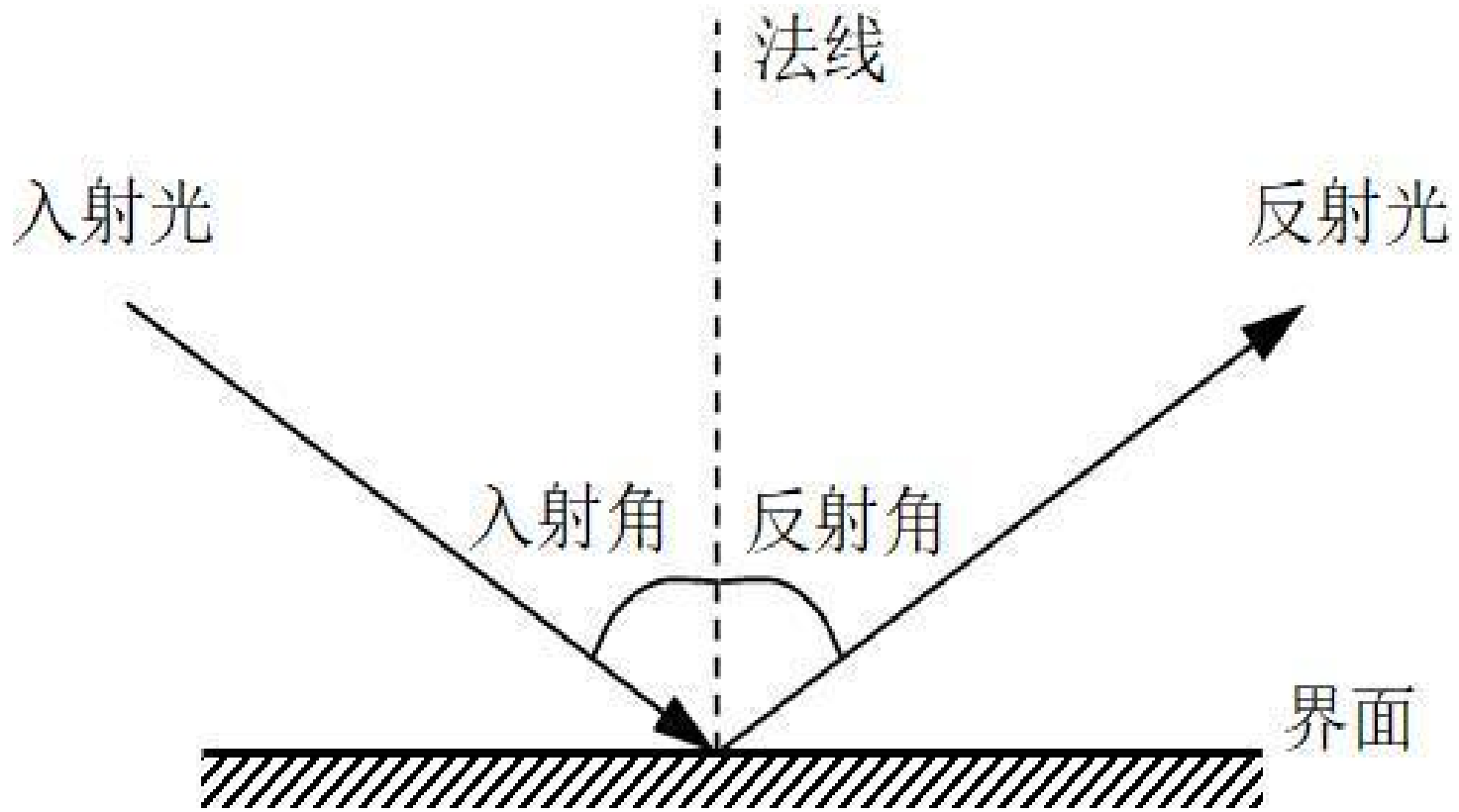
Yet again, linearity is our friend!



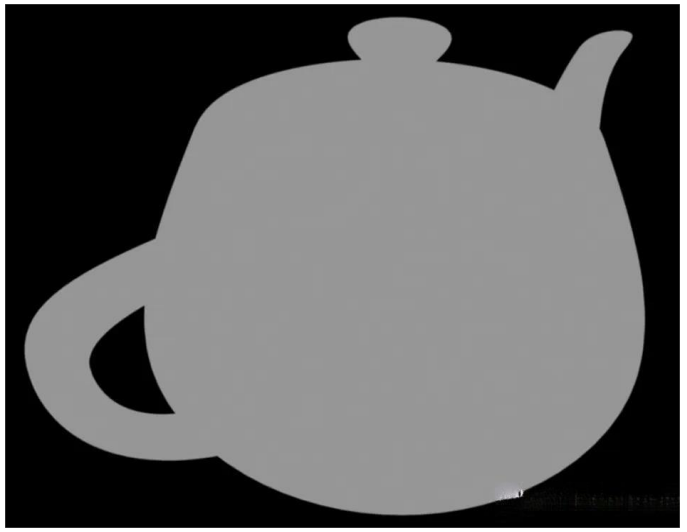
Light Sources

- **Point Light（点光源）**：resembles a light bulb or a small spotlight, radiates evenly in all directions from a single point.
- **Directional Light（平行光）**：The parallel light rays that emanate from a directional light source go in that direction.
- **Spot Light（聚光灯）：**A spot light emits light from a particular position, with a clearly defined cone-shaped beam that may be broadened or constri

Light Sources



Ambient Lighting



环境光

- 由环境光在临近物体上经过多次反射所产生
- 通常简化为在各个方向都有均匀的光强度
- 环境反射光亮度表示为

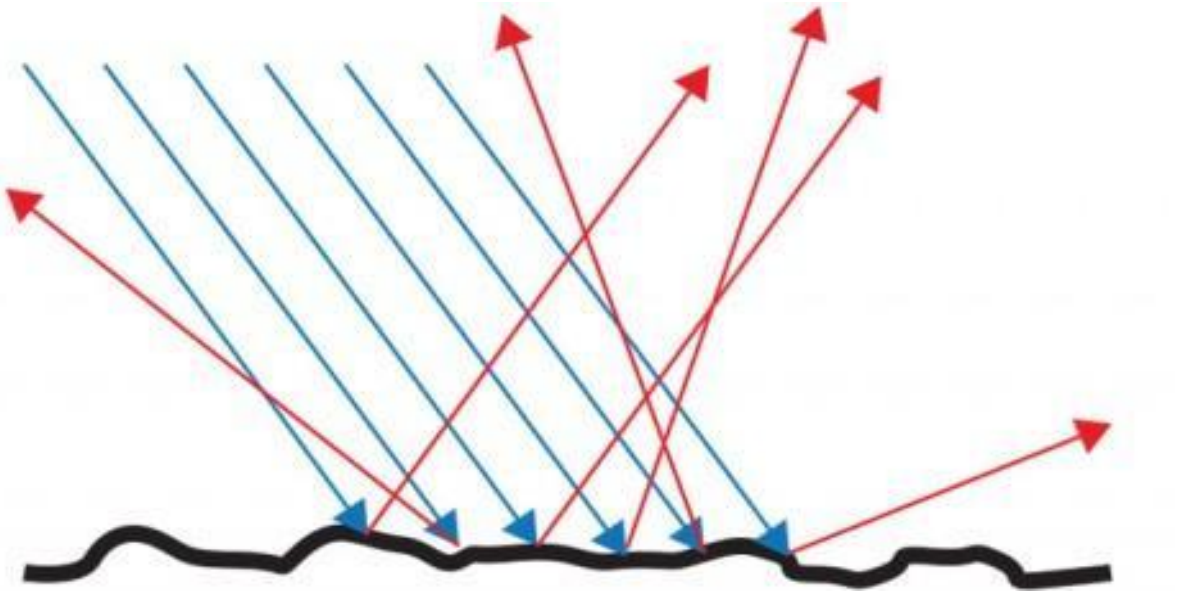
$$I_a = k_a I_{pa}$$

I_a为物体的环境光反射亮度，I_{pa}为环境光亮度，k_a为物体表面的环境光反射系数$(0 \leq k_a \leq 1)$

Diffuse Lighting

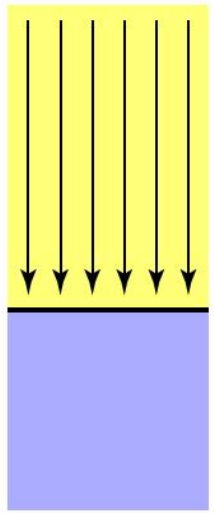
漫反射光

- 表面某一点向空间**各方向**均匀反射
- 在每个方向上都有**相同强度**的反射
- 一个比较粗糙的、无光泽的物体对光的反射，如粉笔

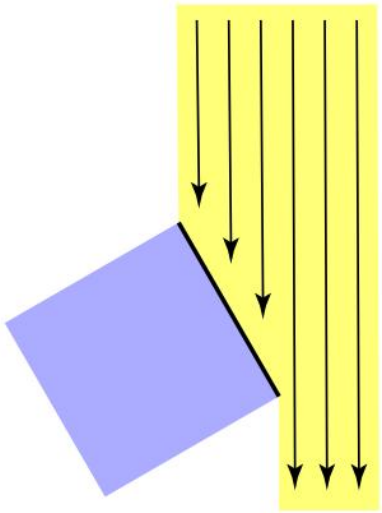


Lambert's Cosine Law

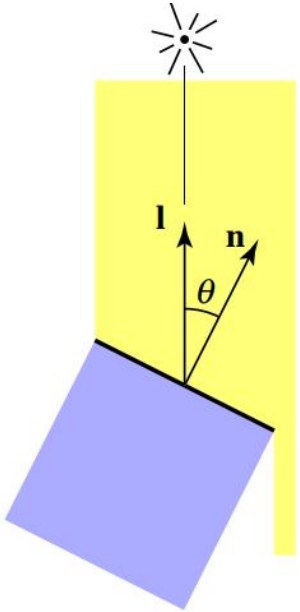
- The illumination of a surface is directly proportional to the cosine of the angle between the normal to the surface and the direction of incident light.



Top face of cube receives a certain amount of light



Top face of 60° rotated cube intercepts half the light



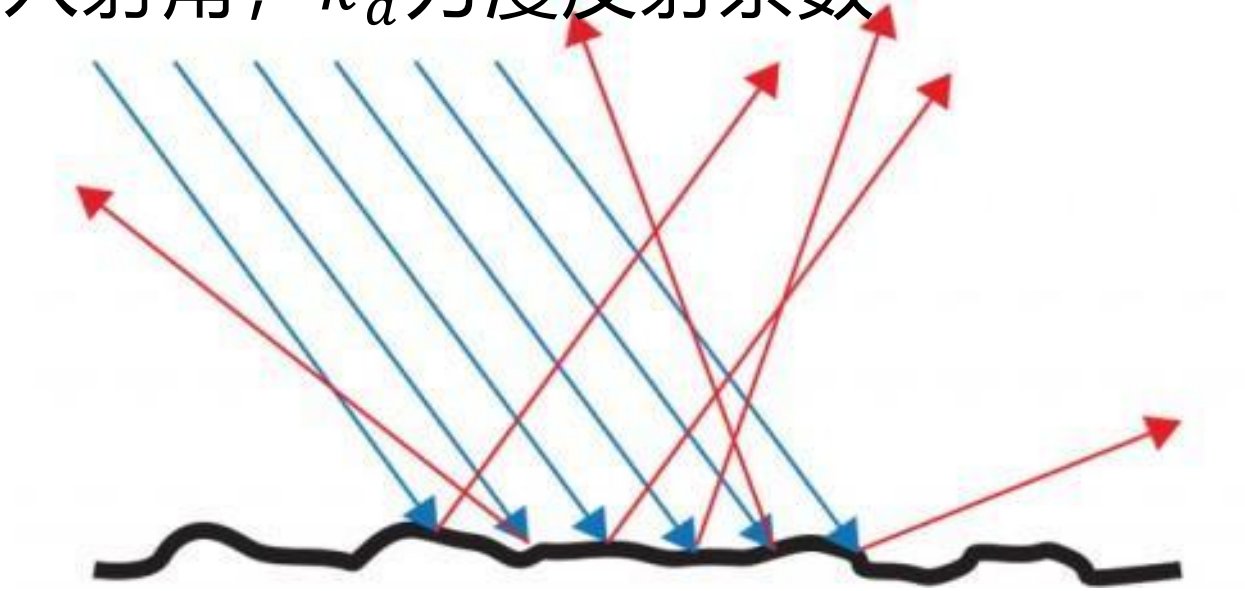
In general, light per unit area is proportional to $\cos \theta = l \cdot n$

Lambert's Cosine Law

- 只与入射光的光亮度和入射方向有关
- 与漫反射光的反射方向无关

$$I_d = k_d I_{pd} \cos i$$

I_d为物体表面漫反射光的光亮度，I_{pd}为光源垂直入射时反射光的光亮度，i为光源入射角，k_d为漫反射系数



Lambert Model

- Therefore, the lighting computation is the sum:

$$I_d = k_d I_{pd} \cos i$$

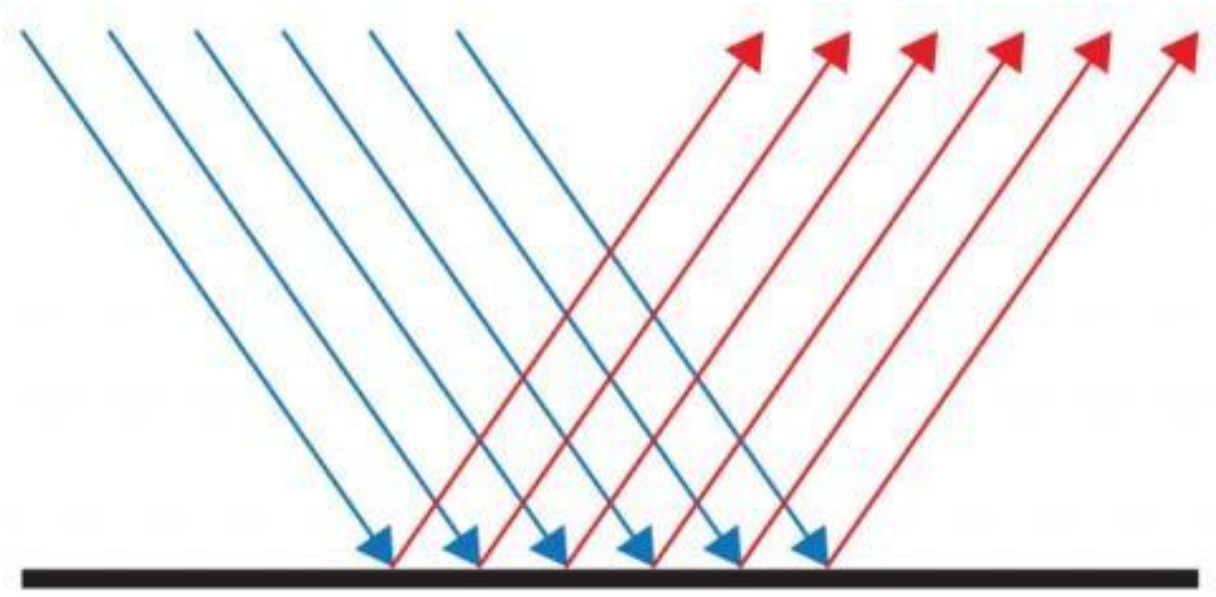
$$I_a = k_a I_{pa}$$

$$I = I_a + I_d$$

Specular Lighting

镜面反射光

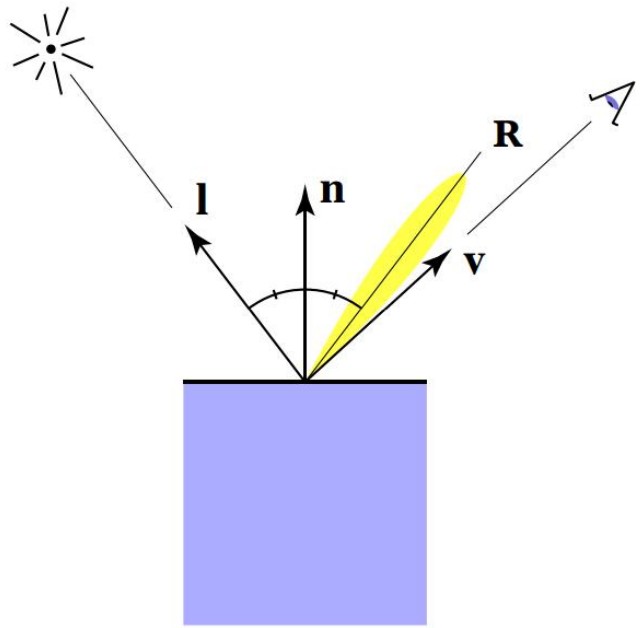
- 镜面反射光为朝一定方向的反射光。反射光和入射光对称地分布于表面法向的两侧。



Specular Lighting

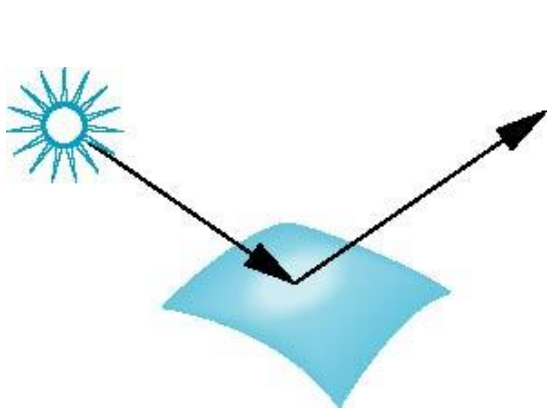
镜面反射光

- 镜面反射光为朝一定方向的反射光。反射光和入射光对称地分布于表面法向的两侧。
- 比如，金属球在阳光下会形成高光，且高光位置随视点的位置而变化。

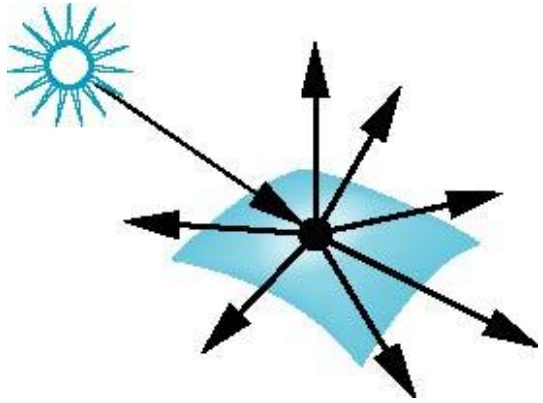


Surface Types

- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflected the light
- A very rough surface scatters light in all directions



smooth surface



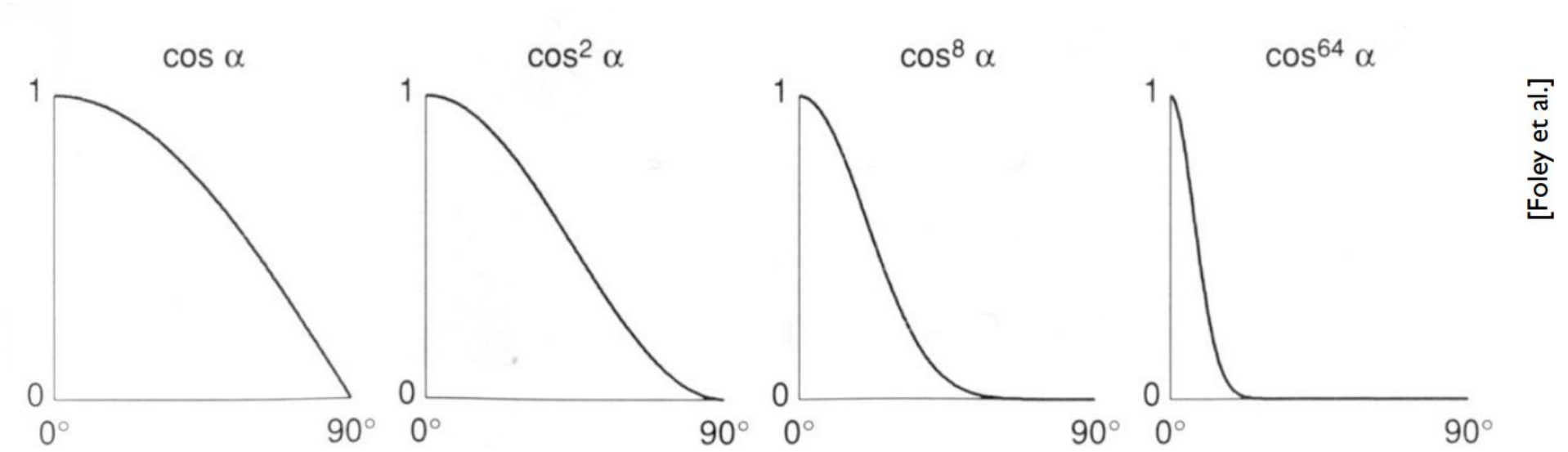
rough surface

Phong Model

- 用余弦函数的幂次来模拟一般光滑表面的镜面反射光的空间分布

$$I_s = k_s I_{ps} cos^p \theta$$

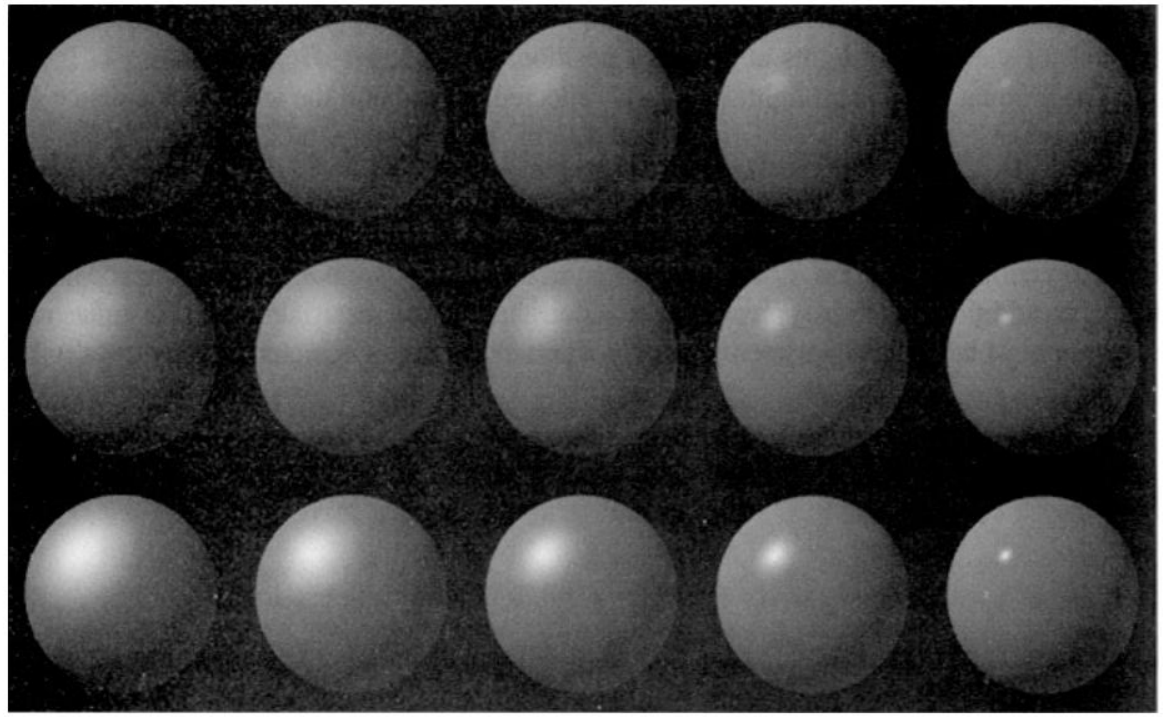
I_s为观察者接受到的镜面反射光亮度，I_{ps}为入射光的光亮度，θ为镜面反射方向和视线方向的夹角，p为镜面反射光的会聚指数（与物体表面光滑程度有关）



Phong Model

- 用余弦函数的幂次来模拟一般光滑表面的镜面反射光的空间分布

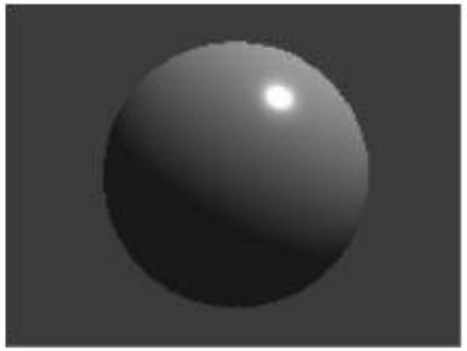
$$I_s = k_s I_{ps} cos^p \theta$$



Phong Model

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

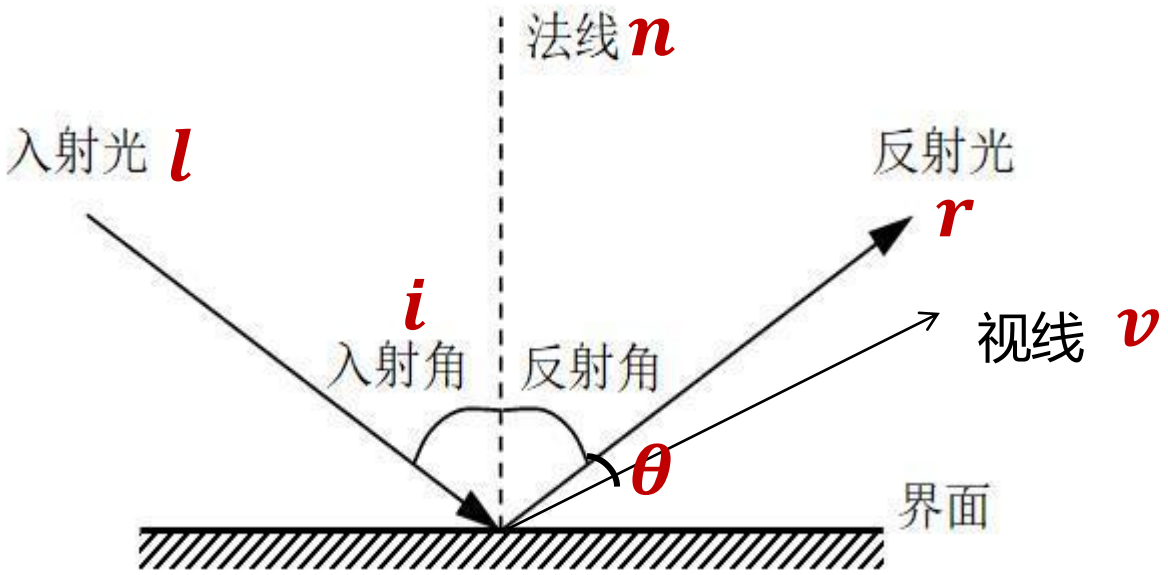
$$= k_a I_{pa} + k_d I_{pd} \cos i + k_s I_{ps} cos^p \theta$$




How to compute the cosine?

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

$$= k_a I_{pa} + k_d I_{pd} \cos i + k_s I_{ps} cos^p \theta$$



$$\cos i = l \cdot n$$

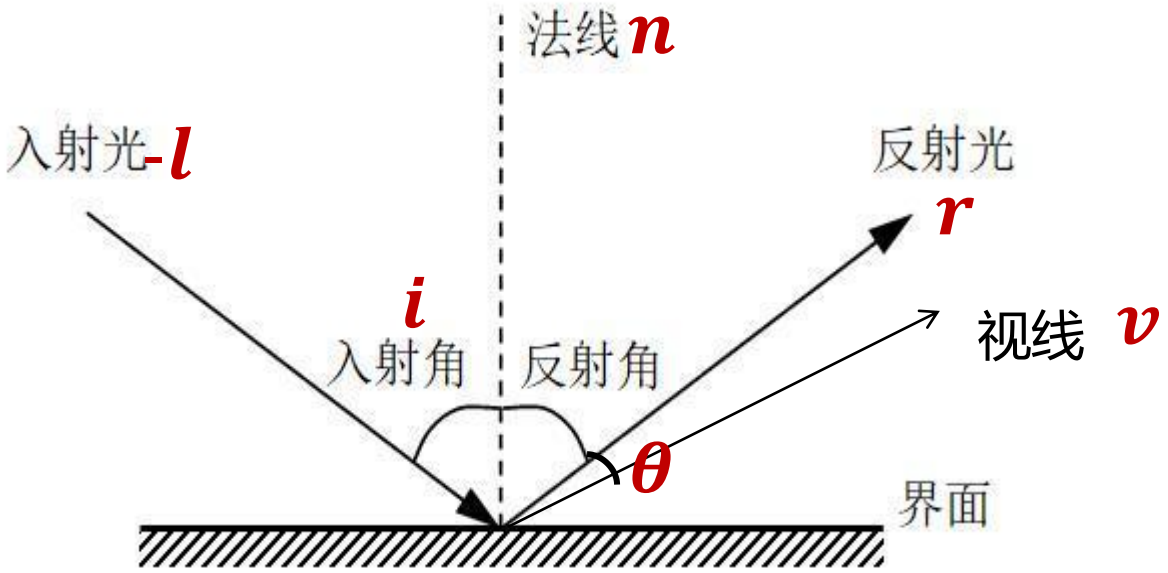
$$\cos \theta = r \cdot v$$

Phone Model

How to compute the cosine?

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

$$= k_a I_{pa} + k_d I_{pd} \cos i + k_s I_{ps} cos^p \theta$$

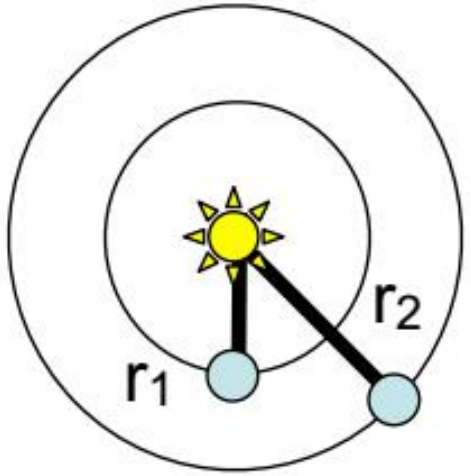


$$\cos i = l \cdot n$$

$$\cos \theta = \frac{v + l}{\|v + l\|}$$

Blinn-Phone Model

Intensity Fall-Off



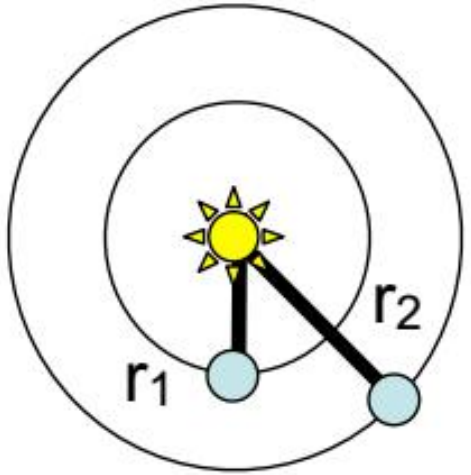
$1/r^2$ fall-off for isotropic point lights

- Why? An isotropic point light
- outputs constant power per solid
- angle (“into all directions”)
- Must have same power in all
- concentric spheres
    - Sphere’s surface area grows with r2 => energy obeys 1/r2

Intensity Fall-Off

$1/r^2$ fall-off for isotropic point lights

- Why? An isotropic point light outputs constant power per solid angle (“into all directions”)
- Must have same power in all concentric spheres
- In particular, 1/(ar2+br+c) is popular
  - I’ll write 1/r2, but take that with a grain of salt

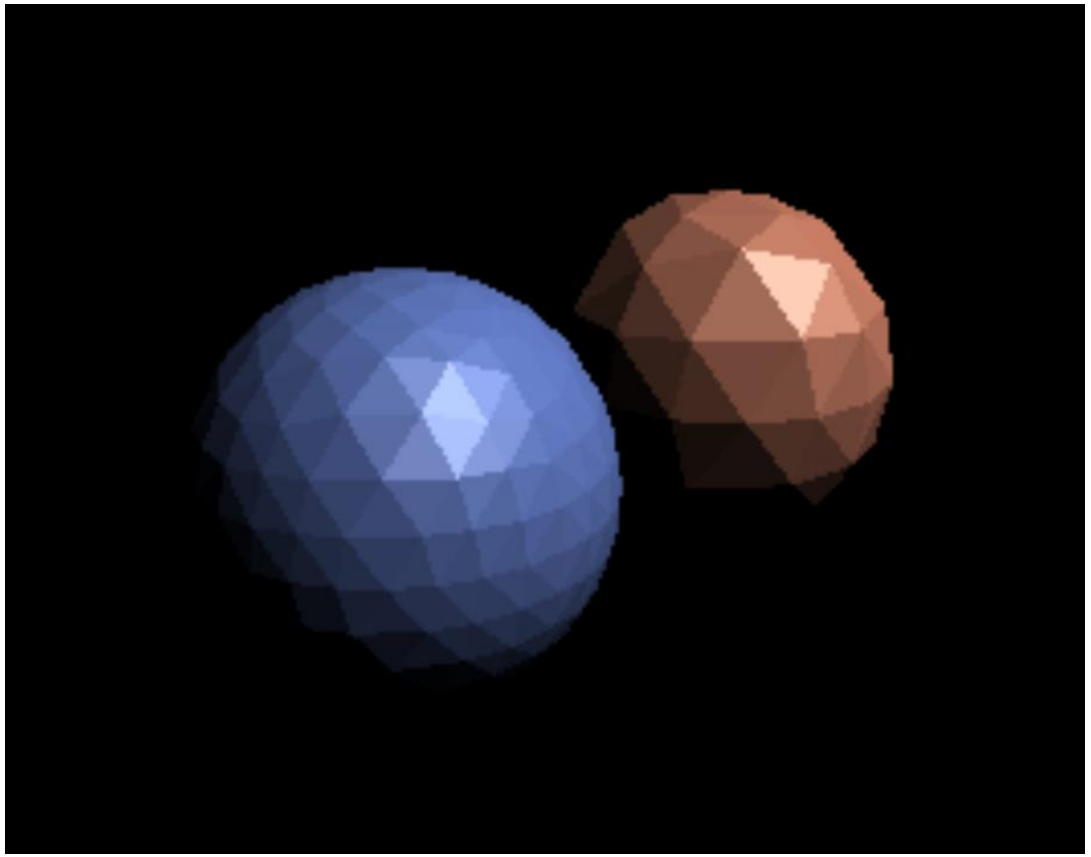


Now we get the shaded result!

Why?

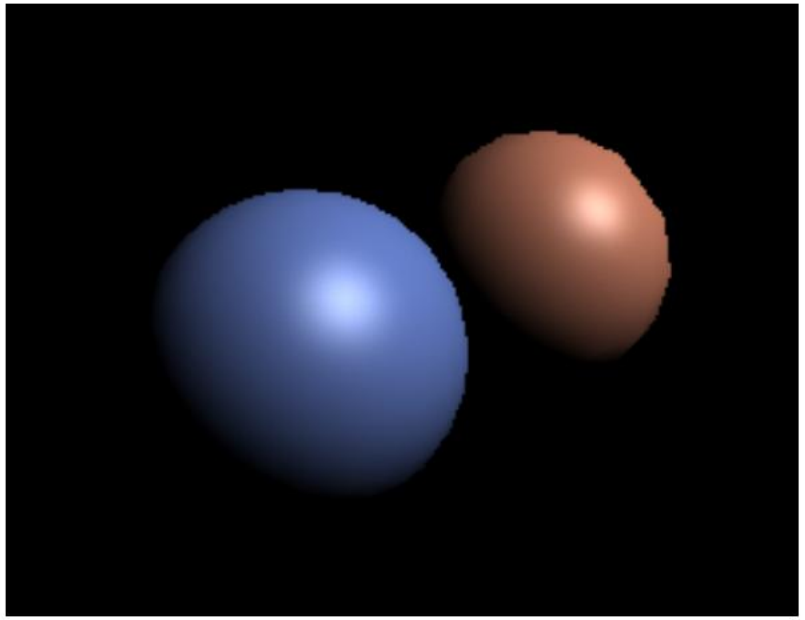
Flat Shading

- Shade each triangle
- Triangle face is flag - one normal vector



Gouraud Shading

- Shade each vertex
- **Interpolate intensity** from vertices across triangles



Step 1. Compute the average normal at each vertex

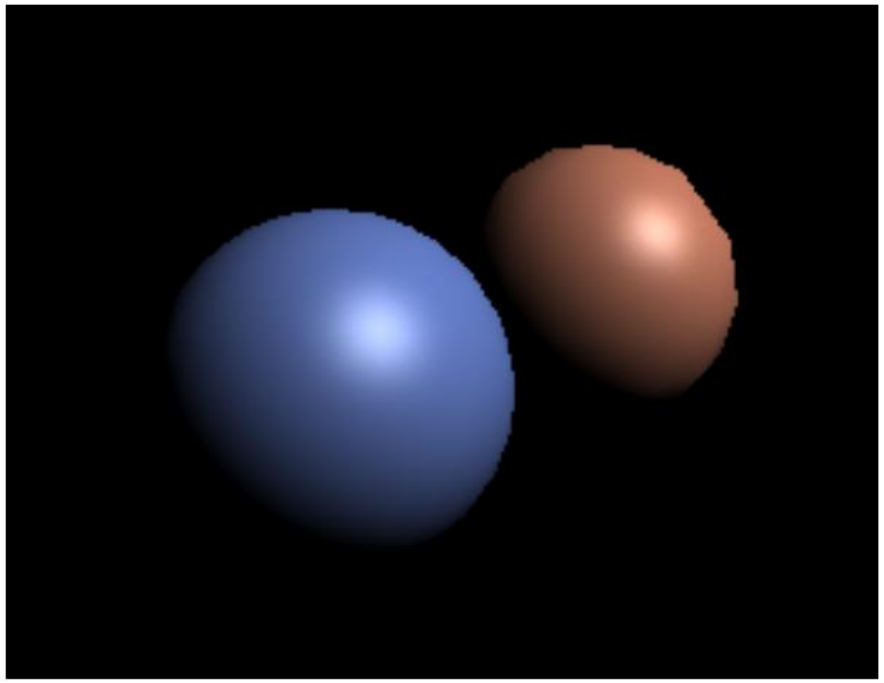
Step 2. Compute the intensity at each veretex with Phong model

Step 3. Interpolate the intensity of each edge

Step 4. Interpolate each point in the triangles

Phong Shading

- Shade each pixel
- **Interpolate normal vectors** across each triangle

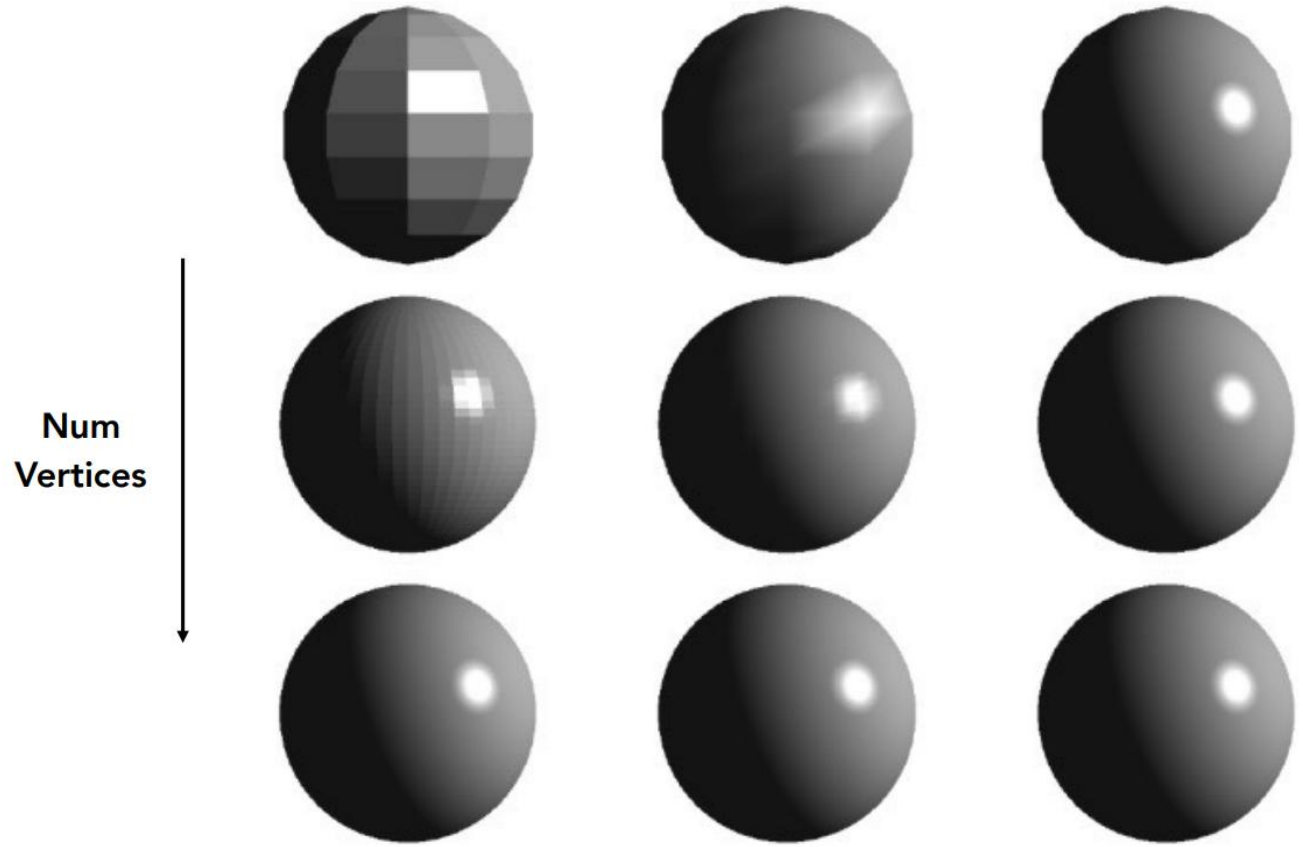


Step 1. Compute the average normal at each vertex

Step 2. Interpolate the normal vector at each point

Step 3. Compute the intensity at this point

Shading Result



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