Computer Graphics

Illumination and Shading

Teacher: A.prof. Chengying Gao(高成英)

E-mail: mcsgcy@mail.sysu.edu.cn

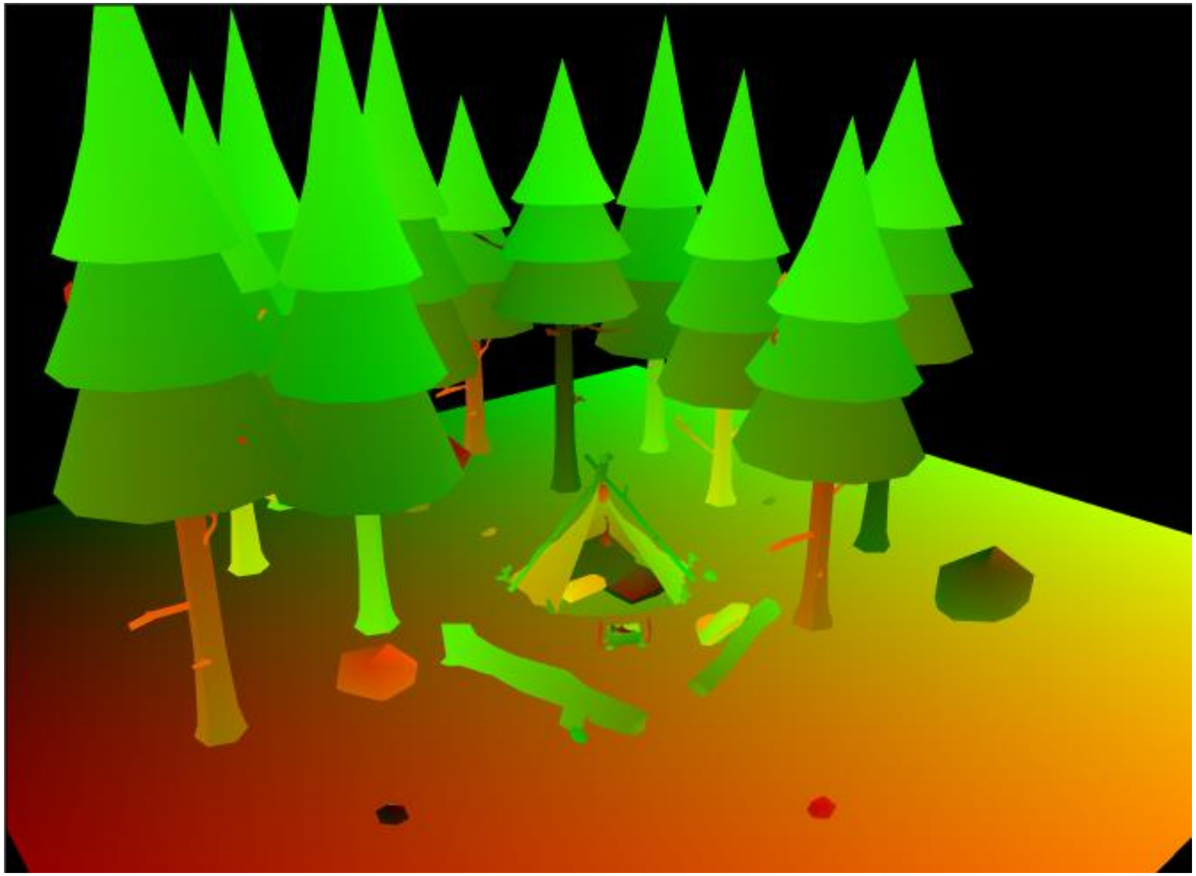
School of Computer Science and Engineering

Lighting & shading

- Without light… we do not see much of our scene!

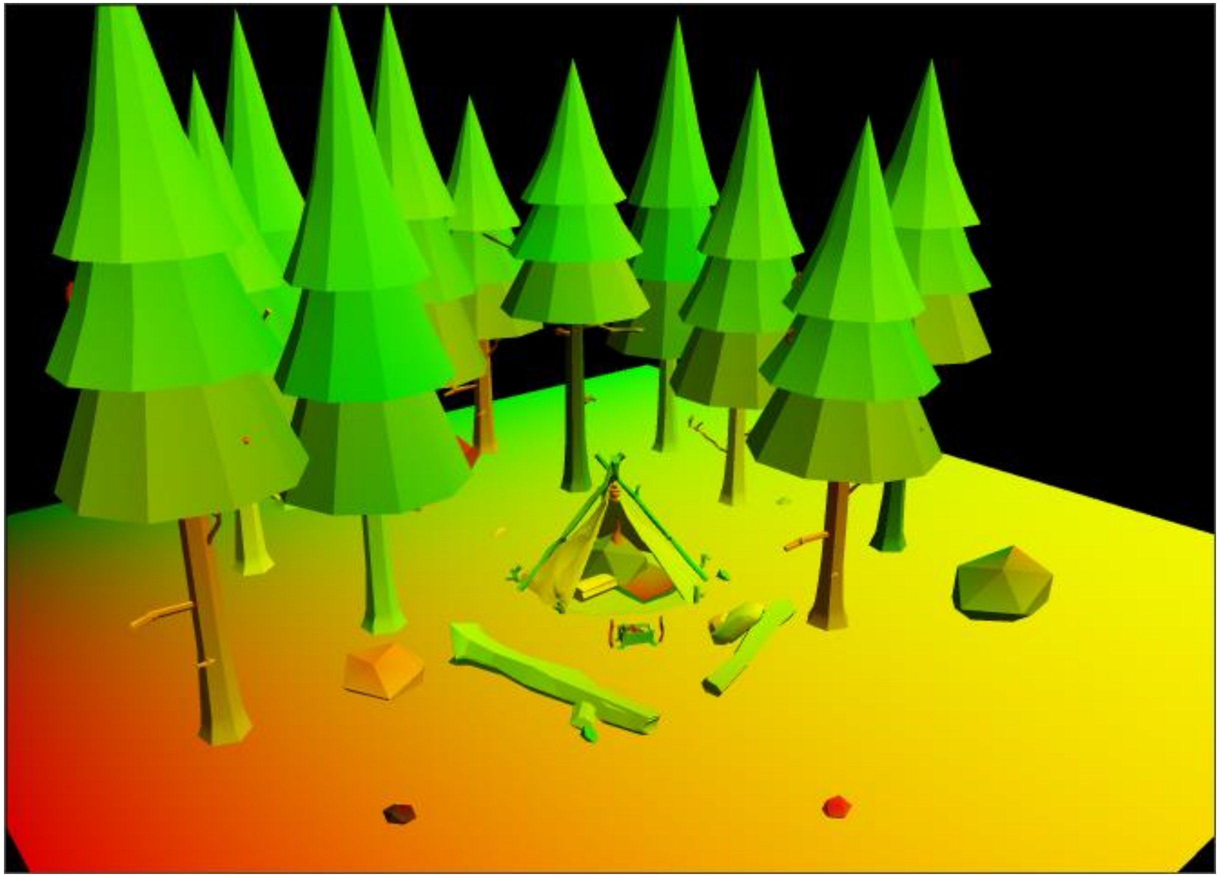
Lighting & Shading

- Without shading… Objects do not look three dimensional



Lighting & Shading

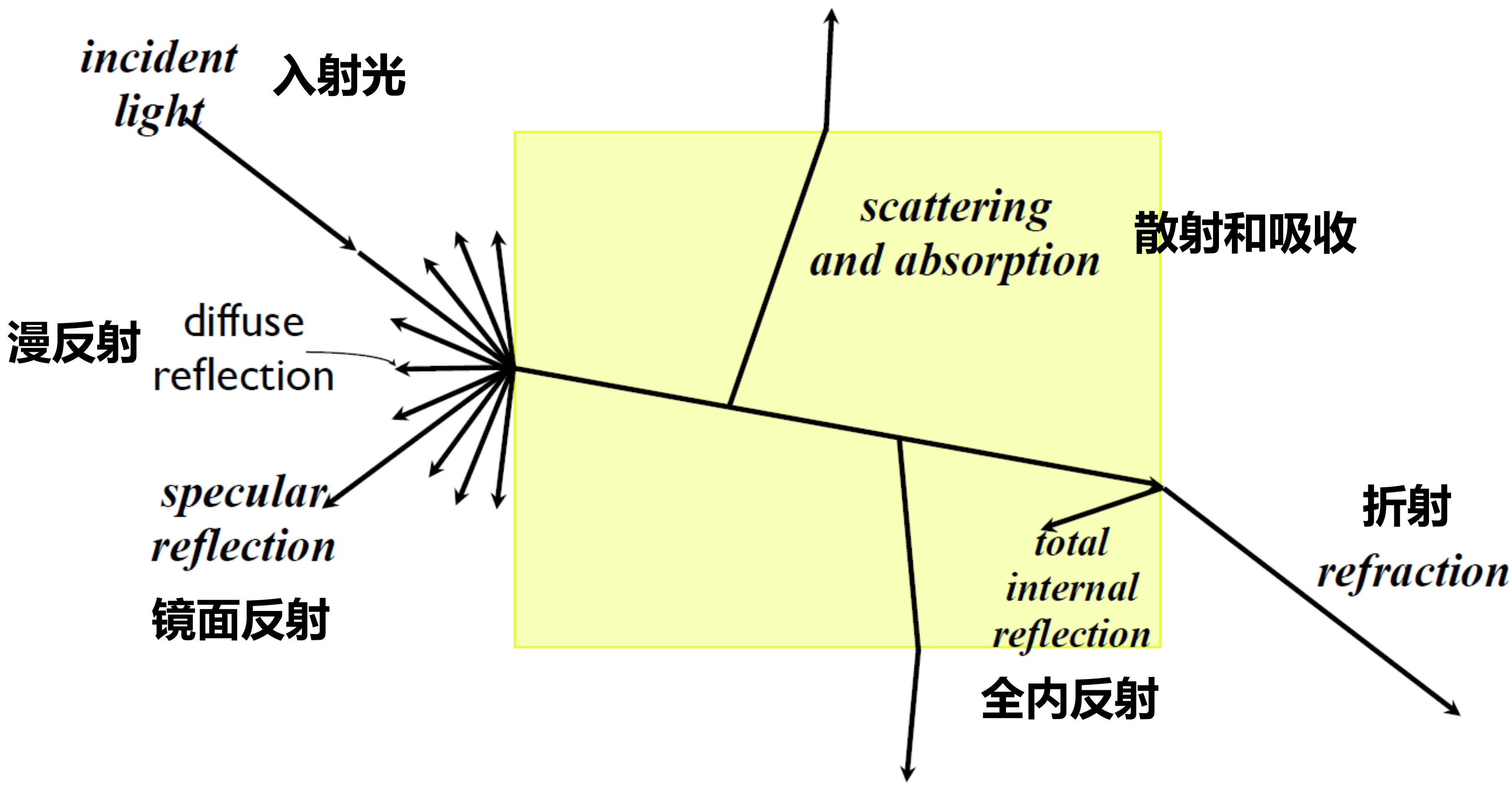
- Objects look three dimensional with shading



Illumination

- ***Illumination*** is the complete description of all the light striking a particular point on a particular surface
- ***Color*** at a point on an object is decided by the properties of the light leaving that point
- Knowing the ***illumination*** and the ***surface physics*** at a point on a surface, we can determine the properties of the light leaving that point
- In order to generate realistic images we need to understand how light interacts with the surface of objects

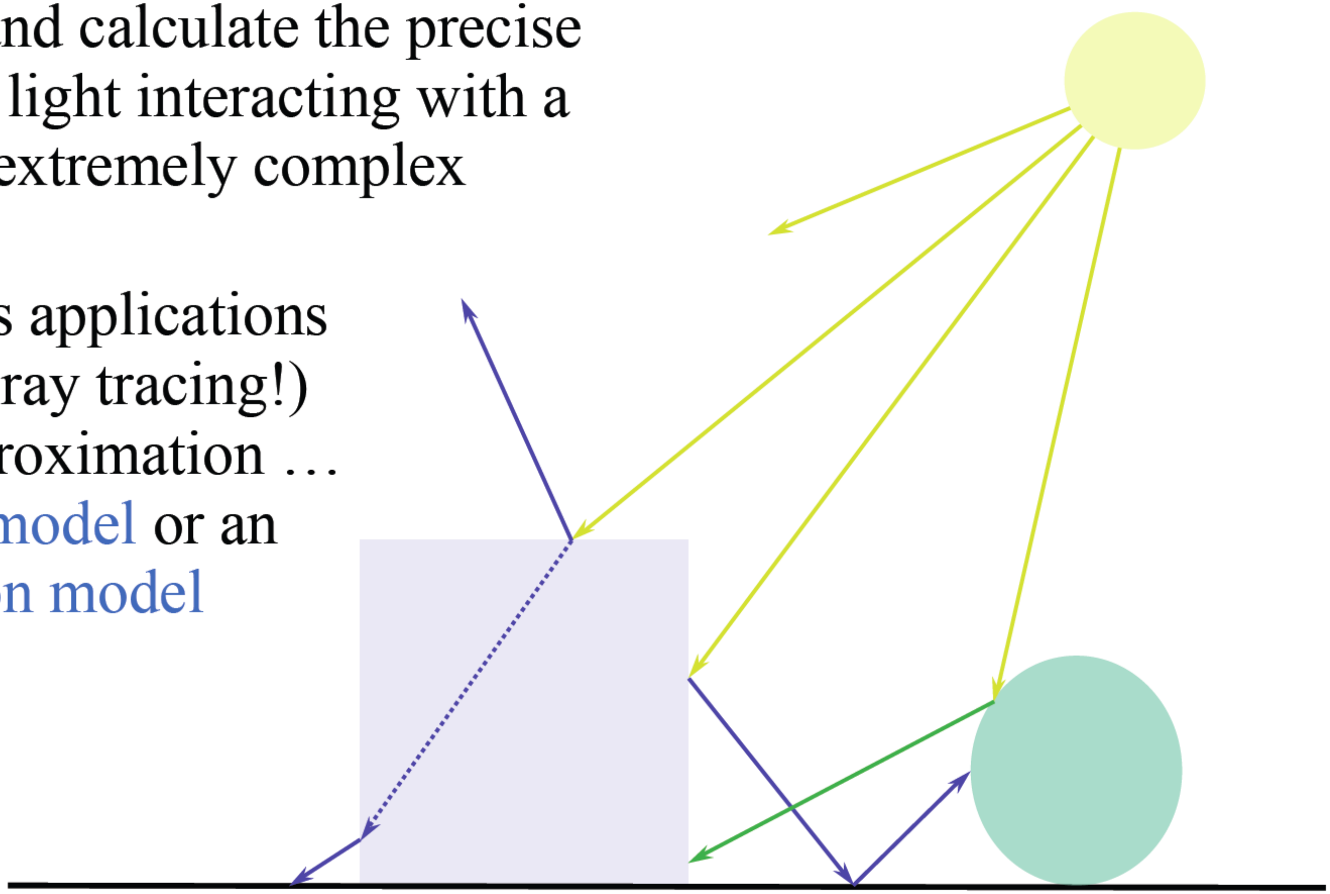
Interaction of light with a Solid



Light interaction in a Scene

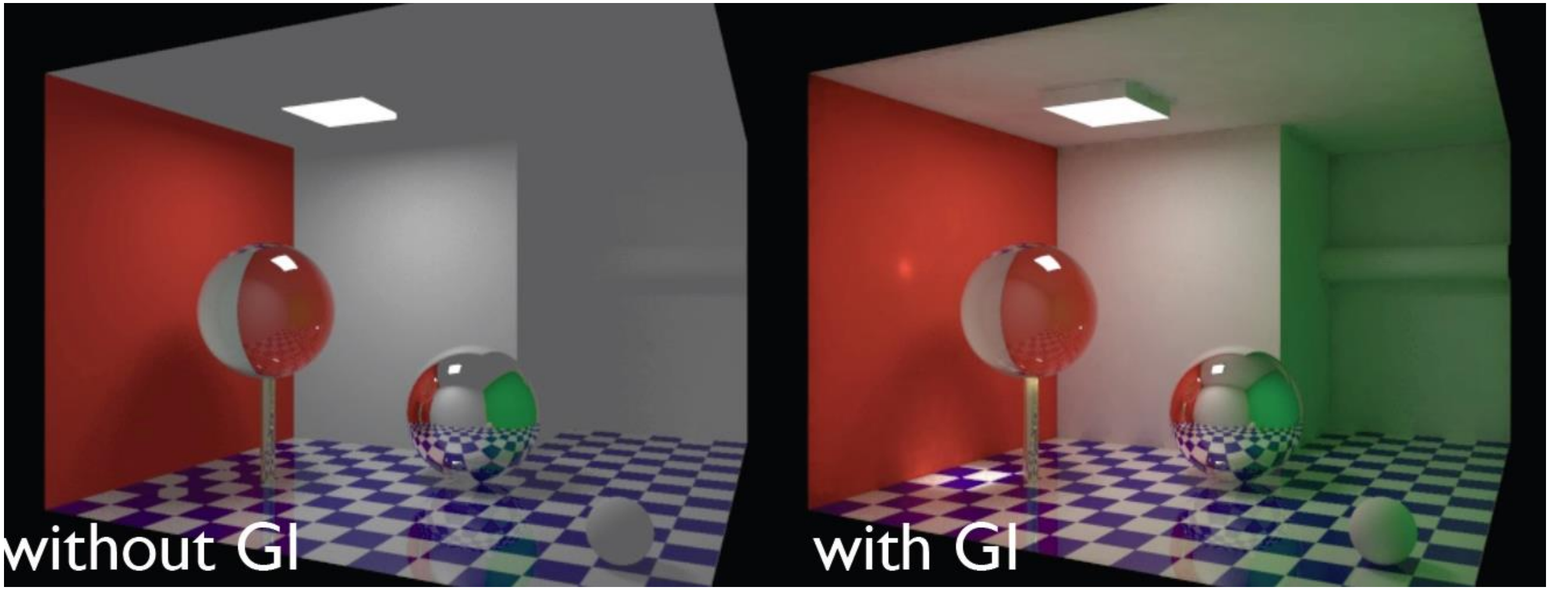
To simulate and calculate the precise physics of light interacting with a surface is extremely complex

Most graphics applications (including ray tracing!) use an approximation … a lighting model or an illumination model



Illumination Models

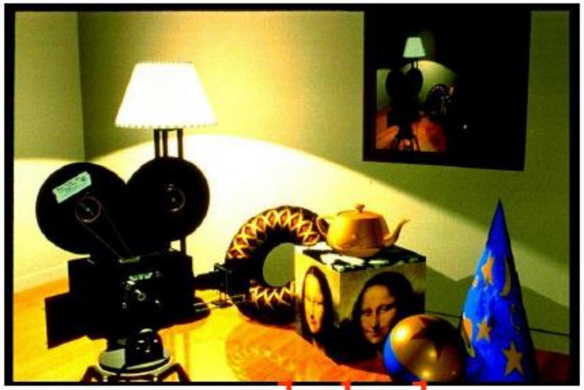
- ***Illumination models***
  - express the factors which determine the surface color at a given point on a surface
  - compute the color at a point in terms of both local and global illumination



Illumination models

- A surface point could be illuminated by
  - Local illumination – Fast
    - “Fake (假伪)” –Ignore real physics, approximate the look
    - Compute at material, from light to viewer
    - Only direct illumination from emitters to surfaces
  - Global illumination – Slow
    - It is illuminated by all the emitters and reflectors in the global scene
    - Physically based

local



global

Appearance depends on Factors

- Light sources
  - Location, type & color
- Surface materials
  - How surfaces reflect light
- Transport of light
  - How light moves in a scene
- Viewer position

The big picture (basic)

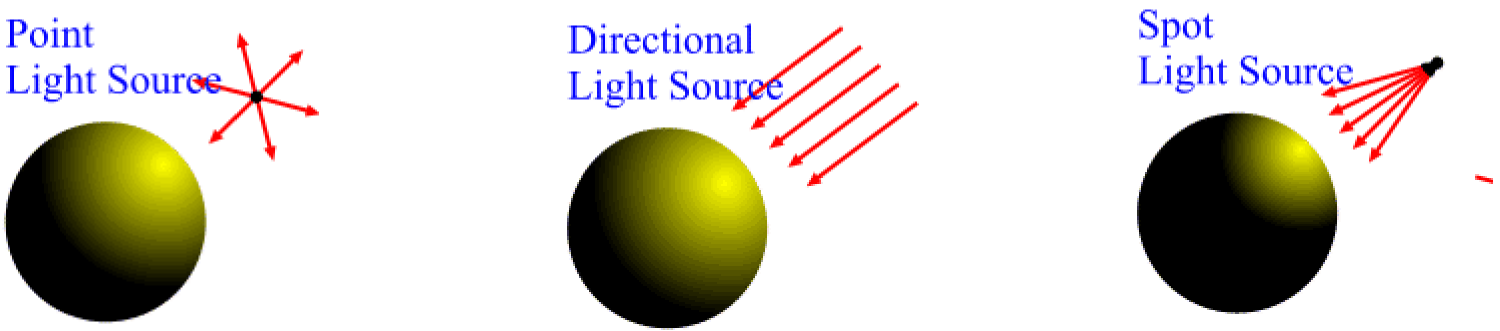
- Light: energy in a range of wavelengths
  - White light – all wavelengths
  - Colored (e.g. red) – subset of wavelengths
- Surface “color” – reflected wavelength
  - White – reflects all lengths
  - Black – absorbs everything
  - Colored (e.g. red) – absorbs all but the reflected color
- Multiple light sources add (energy sums)

Color of Light

- Light not only emits a different amount of different frequencies of light, and their direction property with frequency can also be different
  - The real physical model will be very complicated
- Human visual system is based on the theory of the three primary colors
- In most applications, light could be represented as the intensities of Red, Green and Blue.
  - luminance function is : $\mathbf{I} = [I_r, I_g, I_b]$

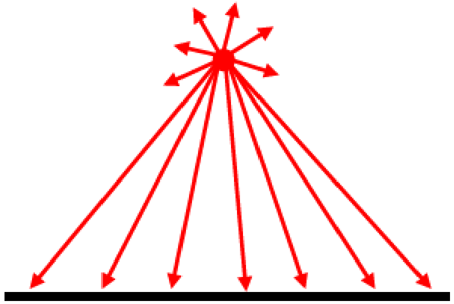
Types of Light Sources

- Point Light source
- Parallel/ Directional light source
- Spot light source



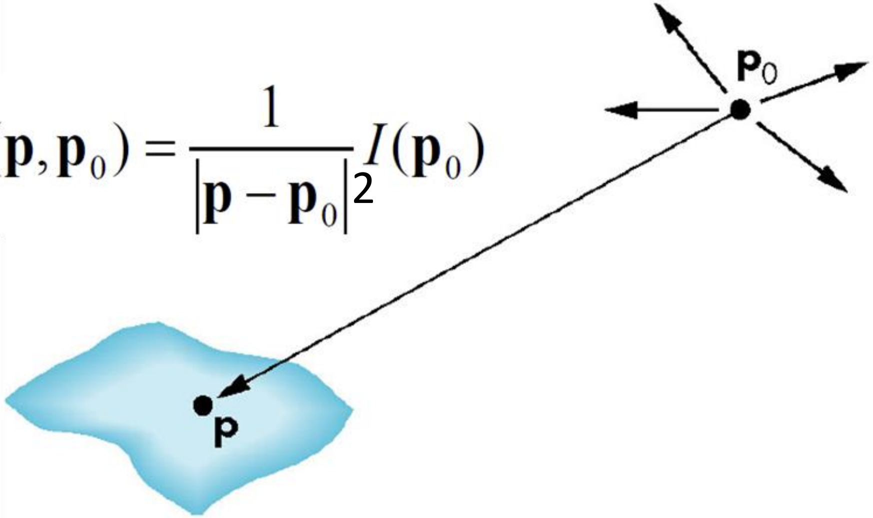
Point source

- A point light source emits light equally in all directions from a single point
- The intensity function of point source: $\mathrm{I}(\mathbf{p}_0) = [I_r(\mathbf{p}_0), I_g(\mathbf{p}_0), I_b(\mathbf{p}_0)]$
- The intensity of P is inversely proportional to the distance between P_0 and P.
- defined by **location** only

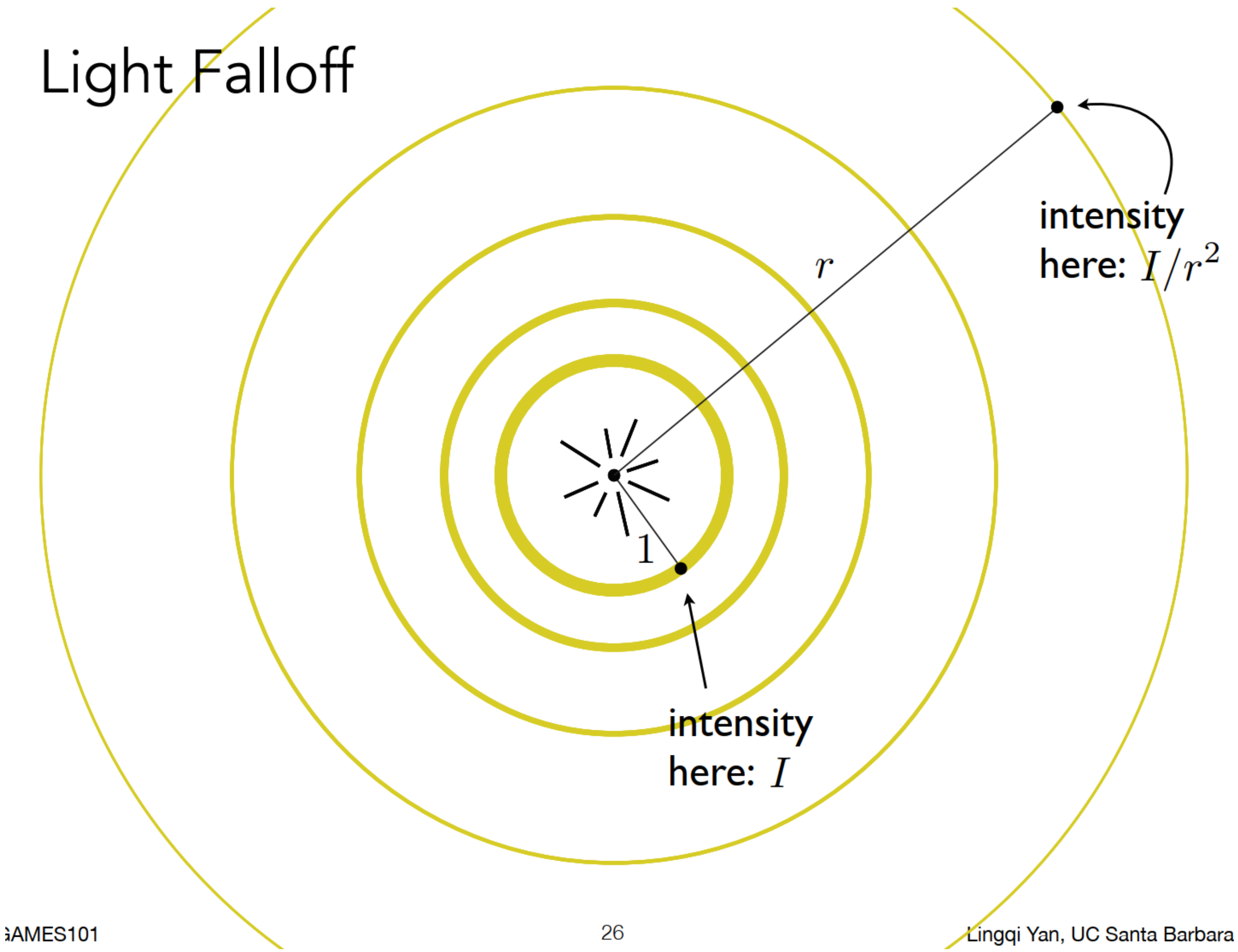


$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$i(\mathbf{p}, \mathbf{p}_0) = \frac{1}{|\mathbf{p} - \mathbf{p}_0|^2} I(\mathbf{p}_0)$$

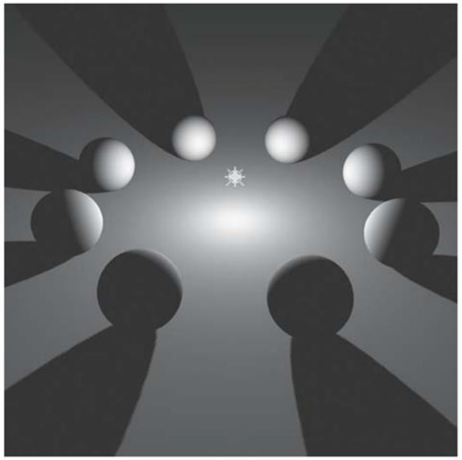


Light Falloff



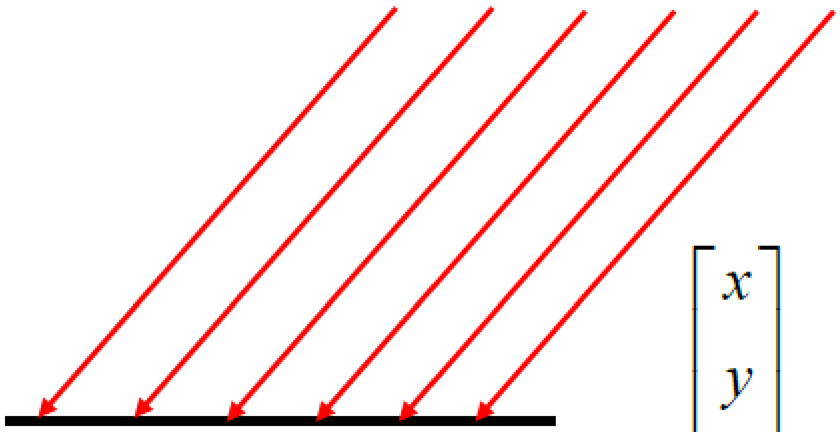
Application of Point Light

- Point sources are widely used in computer graphics for it is easy to use.
- Point sources can't simulate well the physical reality.
  - The contrast in the image is high when only has point sources in a scene: Objects appear too bright, or very dark.



Parallel source

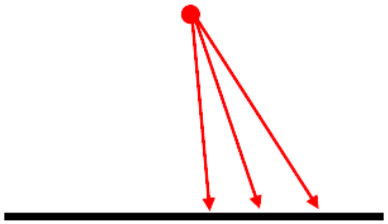
- light rays are parallel
- Rays hit a planar surface at identical angles
- May be modeled as point source at infinity
- Directional light
- defined by **direction** only



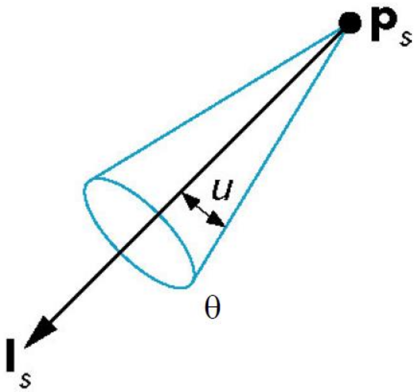
$$\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

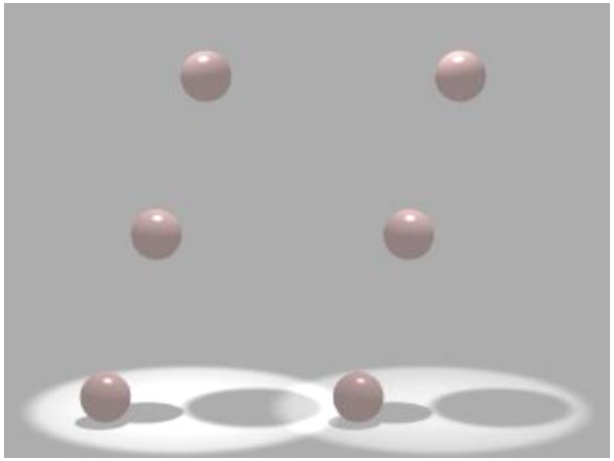
Spotlights

- Spotlights are point sources whose intensity falls off directionally. Point + direction + cutoff angle
  - Requires color, point direction, falloff parameters
  - Supported by OpenGL



- 可以给点光源加上一定的限制得到
- 锥的顶点在P_s，而中心轴方向为l_s。
- 如果$\theta = 180°$，聚光灯成为点光源





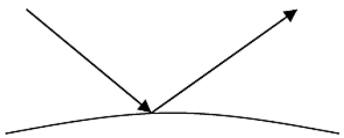
Materials

- Surface reflectance:
  - Illuminate surface point with a ray of light from different directions
  - How much light is reflected in each direction?

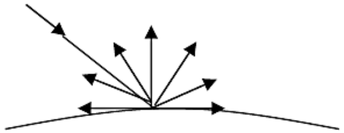


Types of Reflection

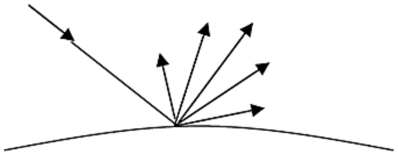
- **specular reflection** (a.k.a. *mirror* or *regular*) causes light to propagate without scattering.



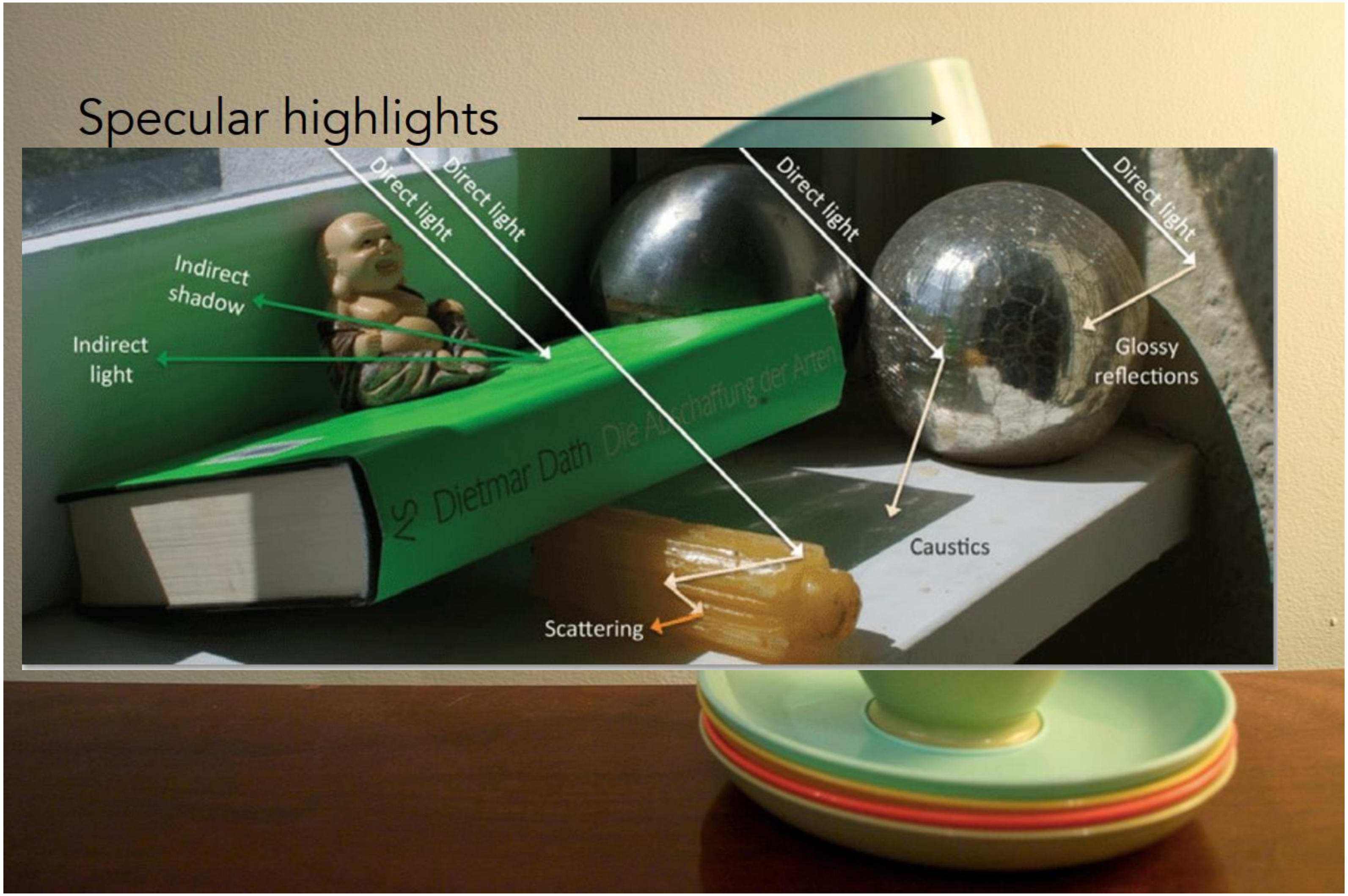
- **diffuse reflection** sends light in all directions with equal energy.



- **glossy/mixed reflection** is a weighted combination of specular and diffuse.

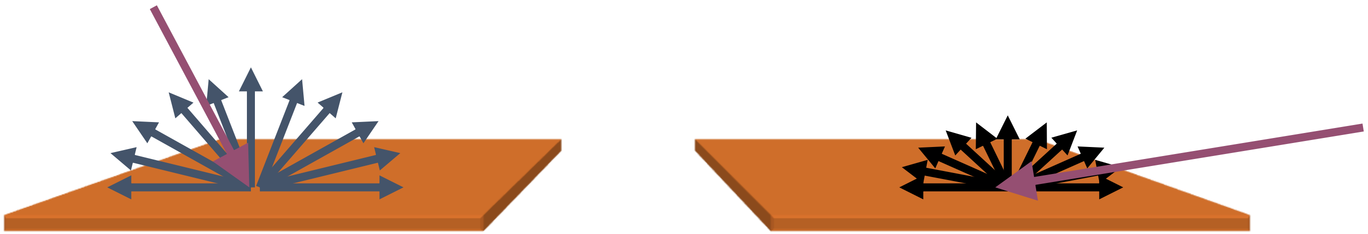


Perceptual Observations



Physics of Diffuse Reflection

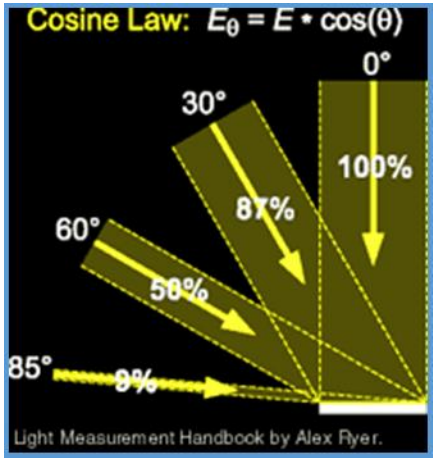
- Very rough surface at the microscopic level
  - real-world example: chalk
- Microscopic variations mean incoming ray of light equally likely to be reflected in any direction over the hemisphere



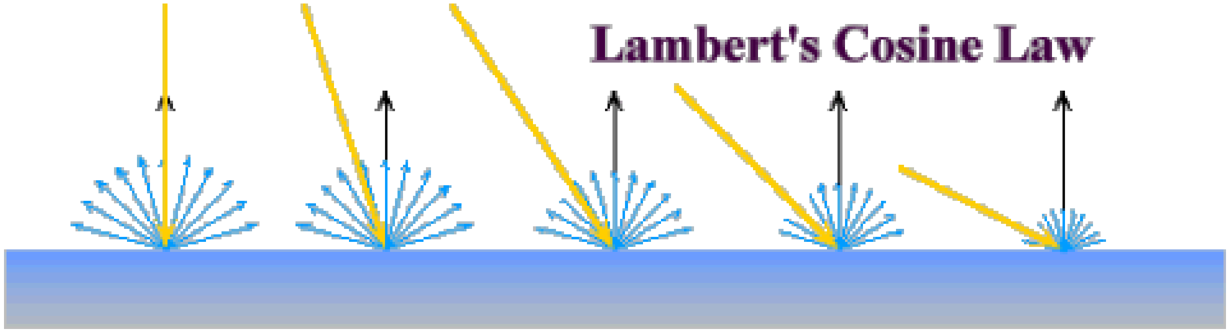
- Light is scattered uniformly in all directions
  - Surface color is the same for all viewing directions

Lambert’s Cosine Law

- Ideal diffuse surfaces reflect according to Lambert’s cosine law:
  - the energy reflected by a small portion of a surface from a light source in a given direction is proportional to the cosine of the angle between that direction and the surface normal



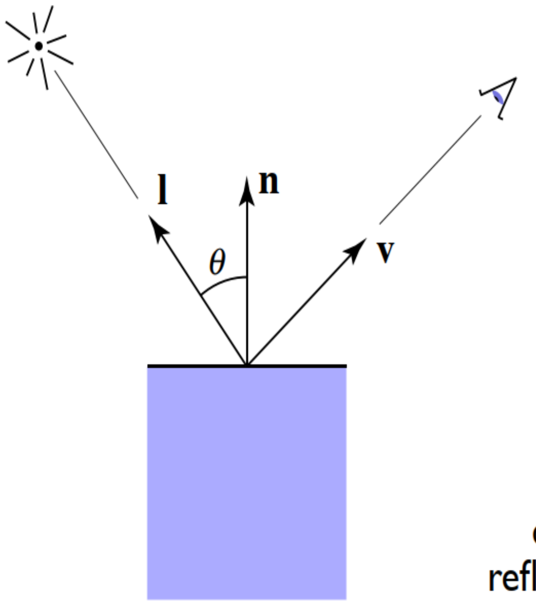
- Reflected intensity
  - independent of viewing direction
  - depends on surface orientation w.r.t. light



Computing Diffuse Reflection

Inputs:

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



energy arrived at the shading point

$$L_d = k_d \, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{l})$$

diffusely reflected light

diffuse coefficient (color)

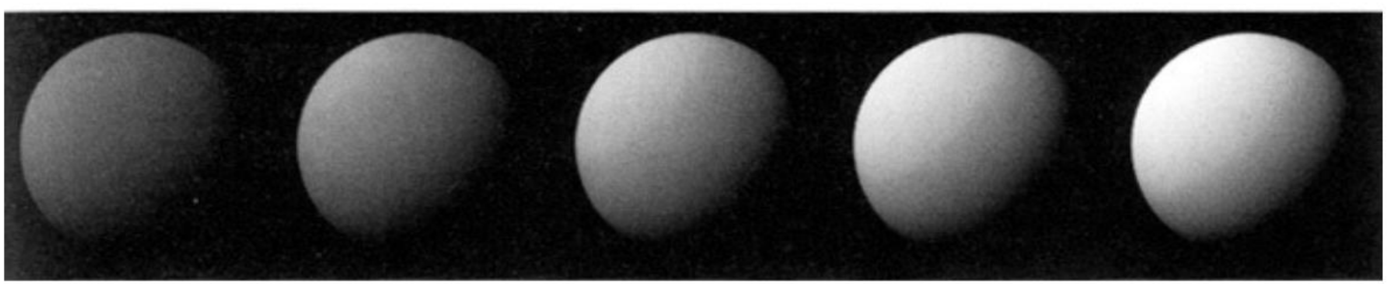
energy received by the shading point

- **always normalize vectors used in lighting!!!**
  - $\mathbf{n}$, $\mathbf{l}$ should be unit vectors

Diffuse Lighting Examples

- Lambertian sphere from several lighting angles （from 0° to 90°）：

- Different Diffuse coefficients:



[Foley et al.]

$k_d \longrightarrow$

Produces diffuse appearance

Specular Highlights

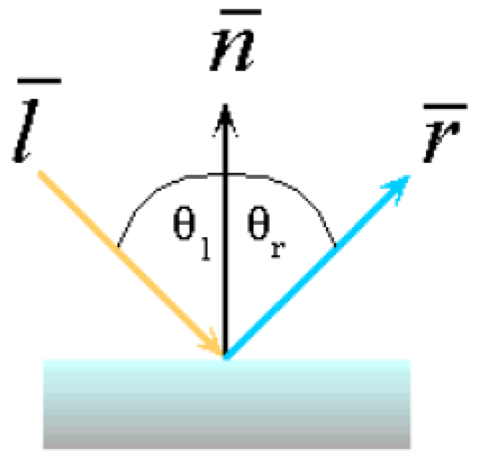
Michiel van de Panne

Specular Reflection

- shiny surfaces (光泽曲面) exhibit specular reflection
  - polished metal
  - glossy car
- specular highlight
  - bright spot from light shining on a specular surface (镜面)
- view dependent
  - highlight position is function of the viewer’s position

Optics of Reflection

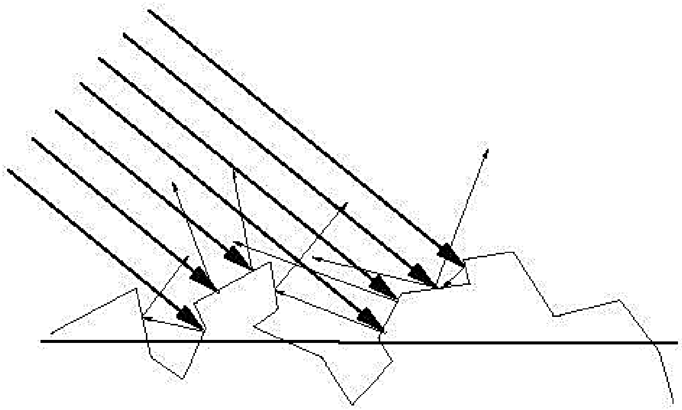
- Reflection follows **Snell's Law**:
  - incoming ray and reflected ray lie in a plane with the surface normal
  - angle the reflected ray forms with surface normal equals angle formed by incoming ray and surface normal



$$\theta_{(l)ight} = \theta_{(r)eflection}$$

Non-Ideal Specular Reflectance (Glassy Reflectance)

- Snell's law applies to perfect mirror-like surfaces, but aside from mirrors few surfaces exhibit perfect specularity
- How can we capture the "softer" reflections of surface that are glossy, not mirror-like?
- One option: model the microgeometry of the surface and explicitly bounce rays off of it

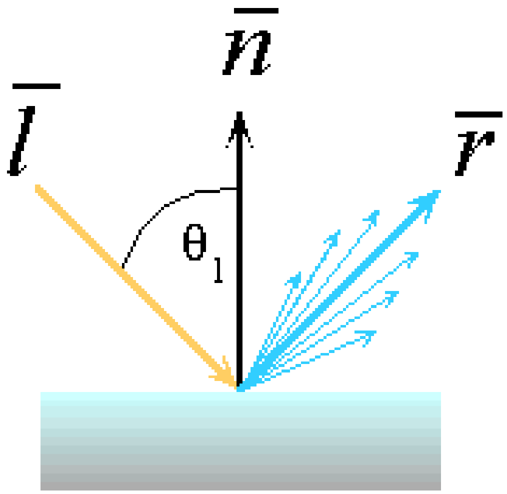


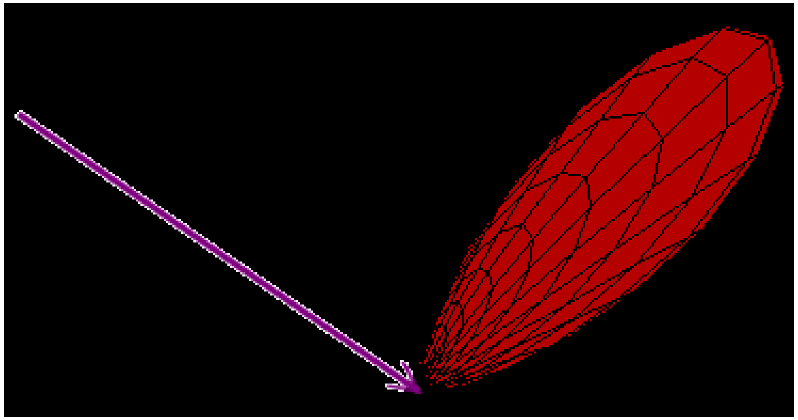
Empirical Approximation

- We expect most reflected light to travel in direction predicted by Snell's Law
- but because of microscopic surface variations, some light may be reflected in a direction slightly off the ideal reflected ray
- as angle from ideal reflected ray increases, we expect less light to be reflected

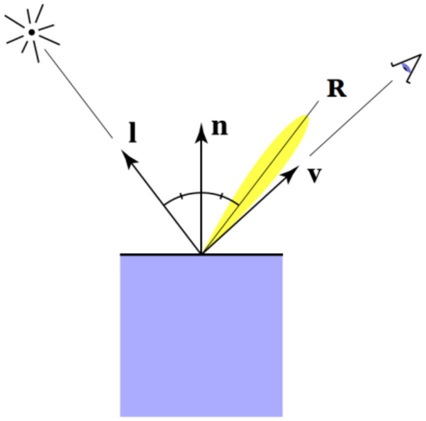
Empirical Approximation

- Angular falloff (角度下降)，No physical basis, works ok in practice





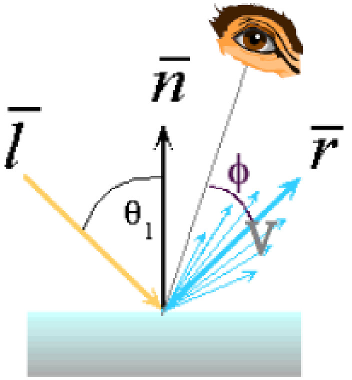
- Bright near mirror reflection direction



Empirical Approximation（Phong）

- **The cos term of lighting can be computed using vector arithmetic:**

$$I_{specular} = k_s I_{light}\left(\bar{v} \cdot \bar{r}\right)^{n_{shiny}}$$

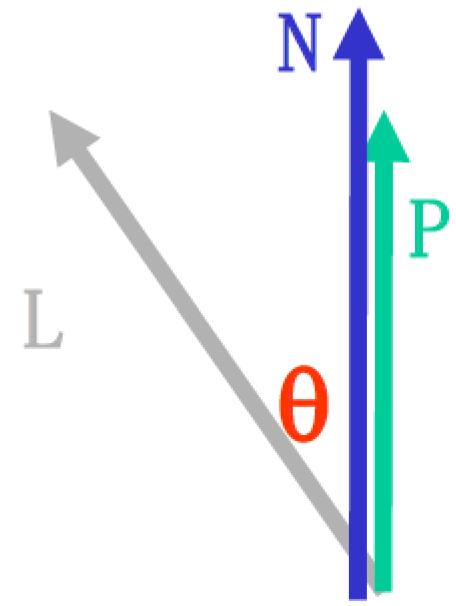


  - V is the unit vector towards the viewer
  - r is the ideal reflectance direction
  - K_s: specular component
  - I_{light}: incoming light intensity
  - n_{shiny}: purely empirical constant, varies rate of falloff(材质发光常数，值越大，表面越接近镜面，高光面积越小。)

- **How to efficiently calculate r ?**

Calculating R Vector

- $P = (N \cdot L) N$:
  - projection of L onto N, L, N are unit length



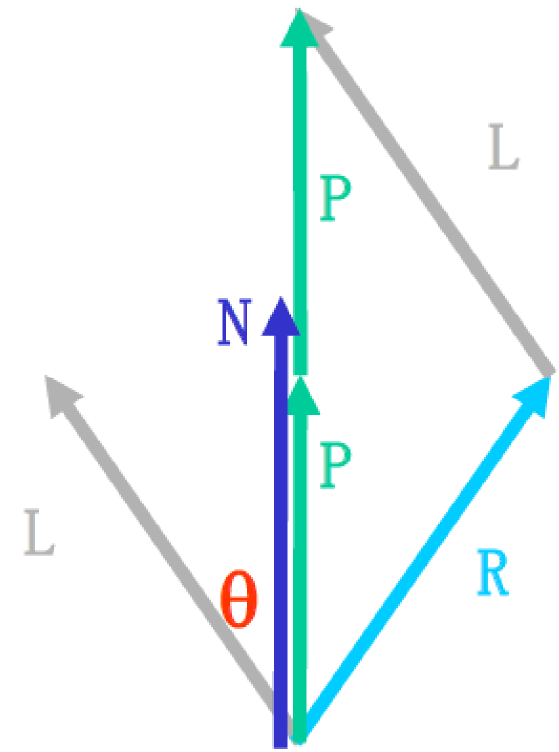
Calculating R Vector

- P = (N · L) N :
  - projection of L onto N, L, N are unit length

2 P = R + L

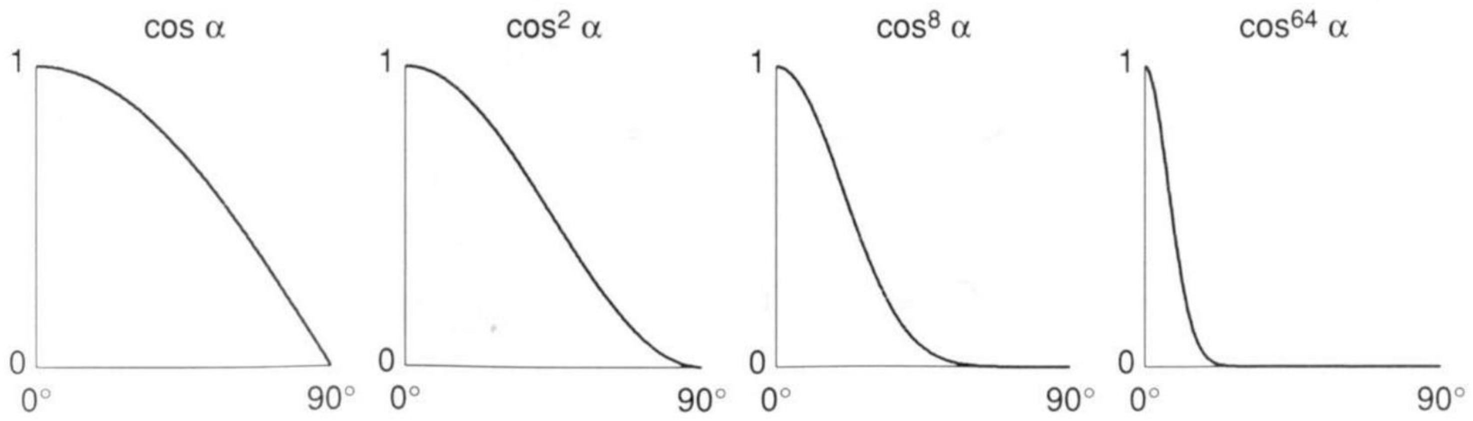
2 P – L = R

2 (N(N · L)) - L = R



Specular Term

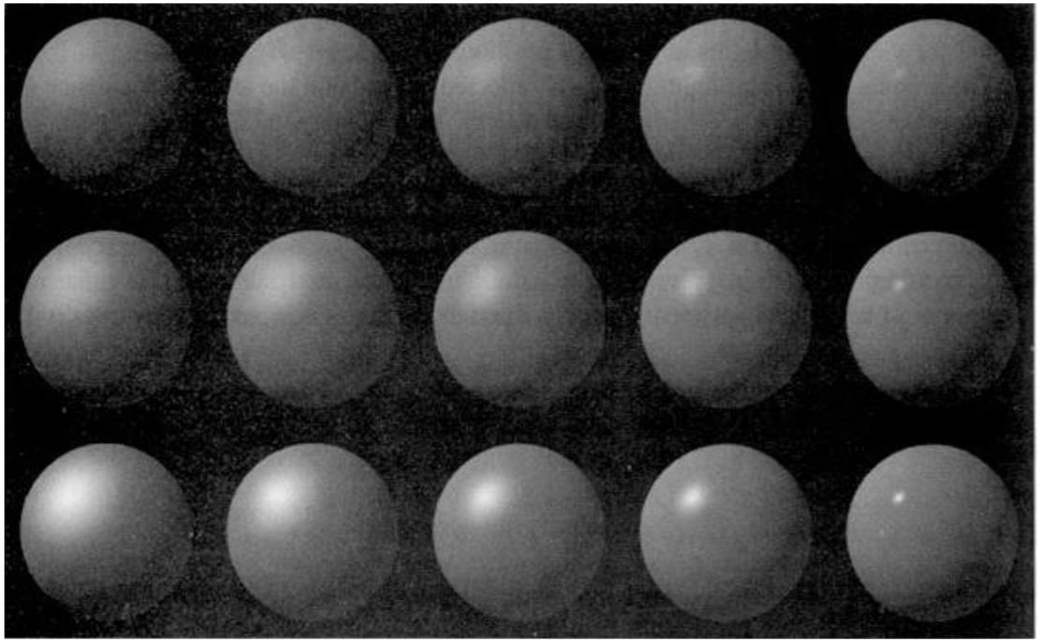
Increasing $\overset{n_{shiny}}{p}$ narrows the reflection lobe



[Foley et al.]

Specular Term

k_s

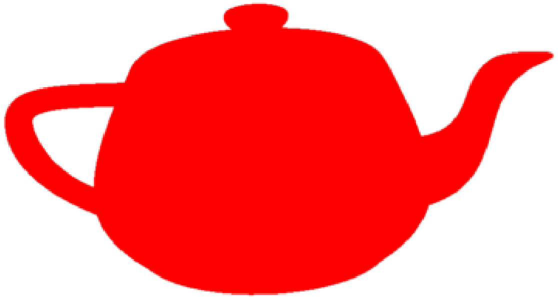


Note: showing
Ld + Ls together

$\mathbf{n}_{\text{shiny}}$

Ambient Term

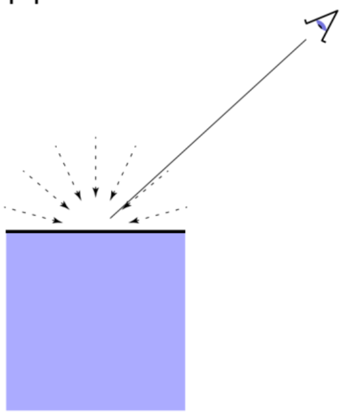
- In reality, parts of the objects not directly illuminated by the light source are not completely dark.
  - e.g., the ceiling in this room, undersides of desks
- These parts are lit by global illumination i.e. light reflected by the surrounding environment; light that is reflected so many times that doesn't seem to come from any where
- Too expensive to calculate (in real time), so we add a constant light called an ambient light
  - No spatial or directional characteristics; illuminates all surfaces equally
  - Looks like silhouette
  - Amount reflected depends on surface properties



Ambient Lights

- For each sampled wavelength (R,G,B), the ambient light reflected from a surface depends on
  - The surface properties, K_a
  - The intensity, I_a, of the ambient light source (constant for all points on all surfaces)

- This is approximate / fake!



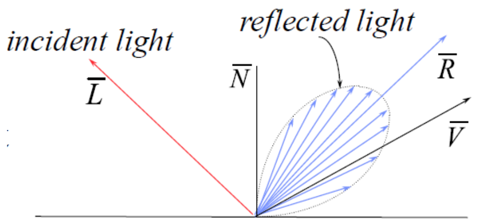
$$L_a = k_a \, I_a$$

ambient coefficient

reflected ambient light

Phong Illumination Model（经验模型）

- Developed by Phong Bui-Tuong (1975) is a popular model for non-perfect reflectors
- Local lighting model
- Reflected intensity is modeled in 3 parts of
  - Diffuse reflection component
  - Specular reflection component
  - Ambient component(to approximate the global Illumination effects)

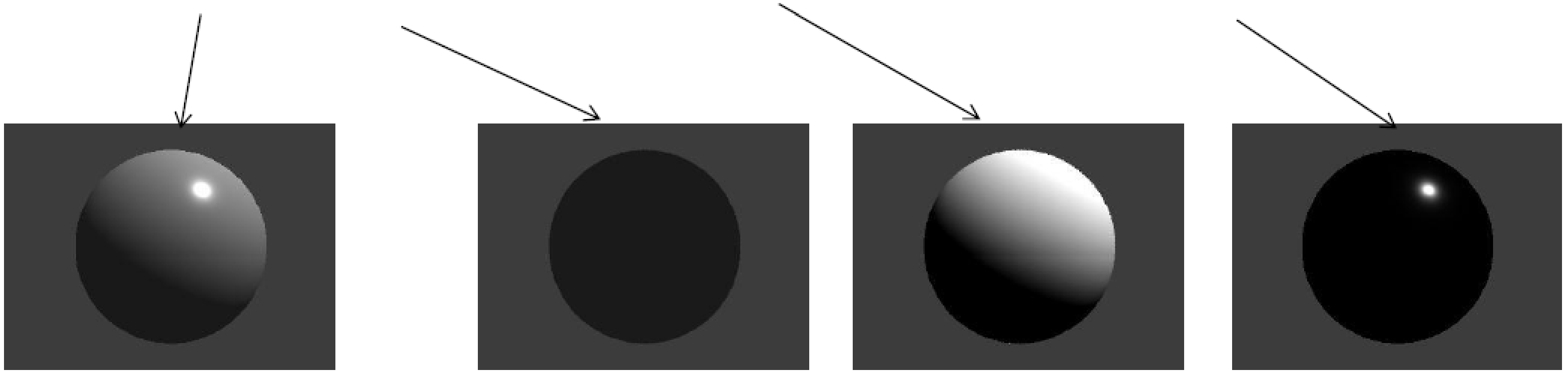


- Based on ambient, diffuse, and specular lighting and material properties
- 经验模型并不是基于物理原理，而是提出经验公式，通过调整参数来模拟光照。

Phong Illumination Model

- The illumination equation in its simplest form is given as(综合了环境光、漫反射及镜面反射)

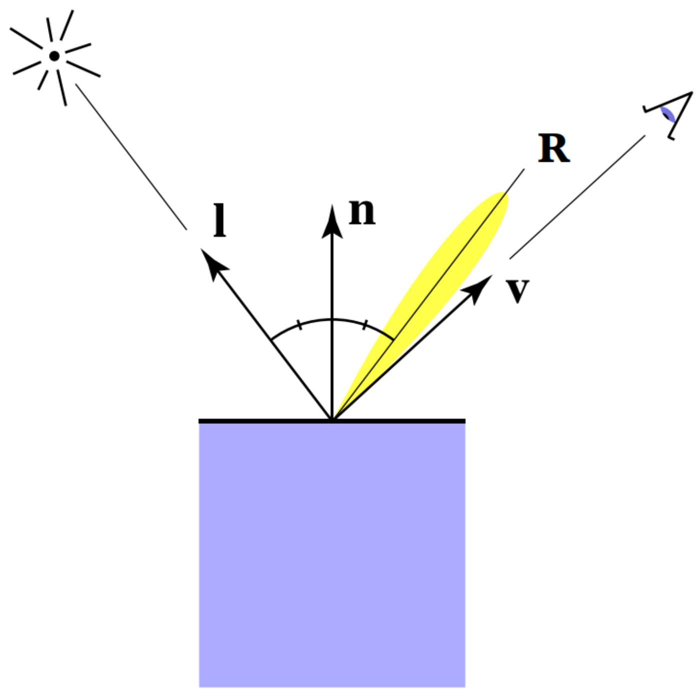
$$I = K_a I_a + K_d I_e \cos\alpha + k_s I_e \cos^n \gamma$$



Specular Term

Intensity depends on view direction

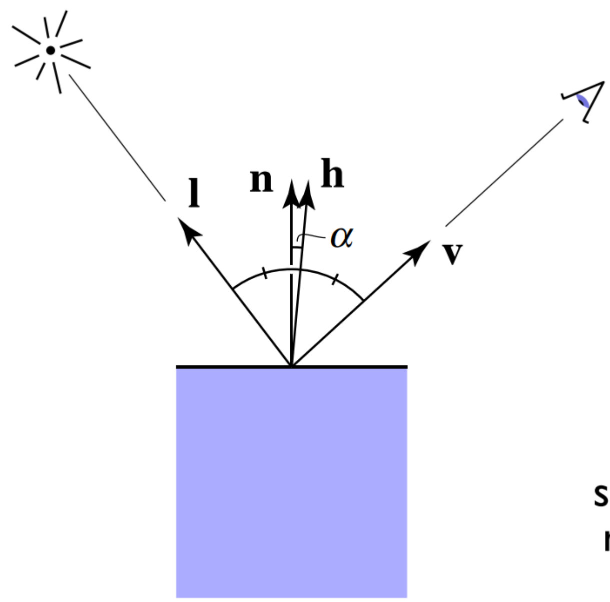
- Bright near mirror reflection direction



Specular Term (Blinn-Phong)

V close to mirror direction ⇔ **half vector** near normal

- Measure "near" by dot product of unit vectors



$$\mathbf{h} = \operatorname{bisector}(\mathbf{v}, \mathbf{l})$$

(半程向量)

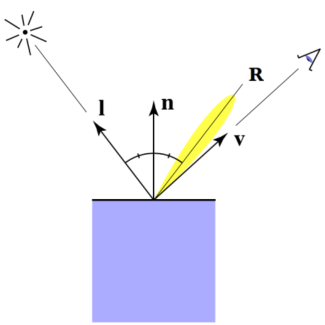
$$= \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

$$L_s = k_s\,(I/r^2)\max(0, \cos\alpha)^p$$
$$= k_s\,(I/r^2)\max(0, \mathbf{n} \cdot \mathbf{h})^p$$

L_s: specularly reflected light

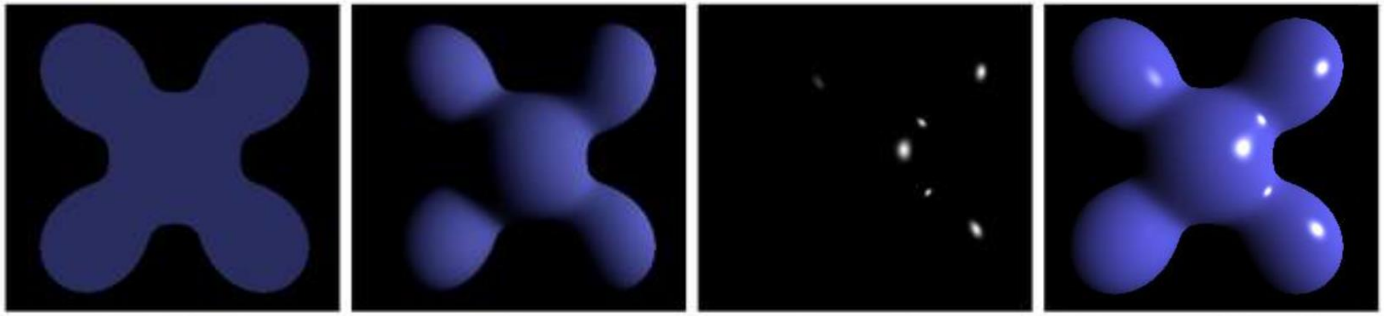
k_s: specular coefficient

- Bright near mirror reflection direction



Blinn-Phong Illumination Model

- The illumination equation in its simplest form is given as(综合了环境光、漫反射及镜面反射)



Ambient + Diffuse + Specular = Blinn-Phong Reflection

$$
\begin{aligned}
L &= L_a + L_d + L_s \\
&= k_a\, I_a + k_d\, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s\, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{h})^p
\end{aligned}
$$

Multiple Lights

- Light is **linear**
  - If multiple rays illuminate the surface point the result is just the sum of the individual reflections for each ray

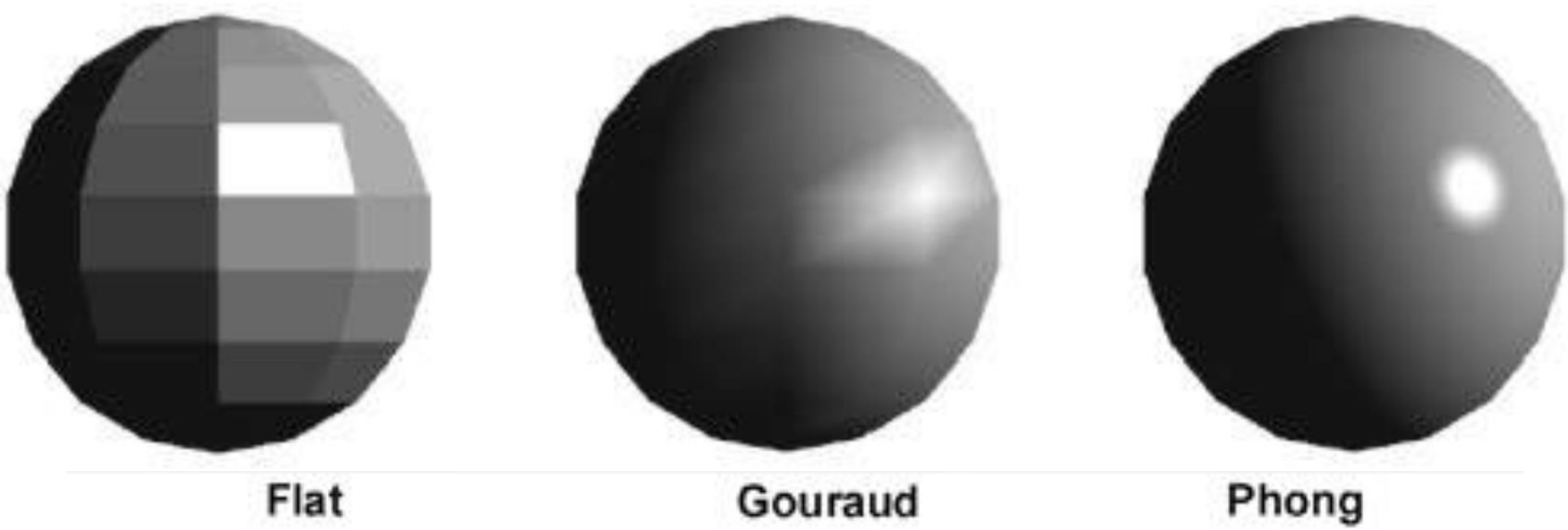
$$\boxed{\sum_p I_p (k_d (n \cdot l_p) + k_s (r_p \cdot v)^n)}$$

- Multiple lights of Phong Illumination Model

$$I = K_a I_a + \sum_{i=1}^{m} I_i (K_d (\vec{n}, \vec{l}) + k_s (\vec{v}, \vec{r})^n)$$

Shading Frequencies

- What caused the shading difference?



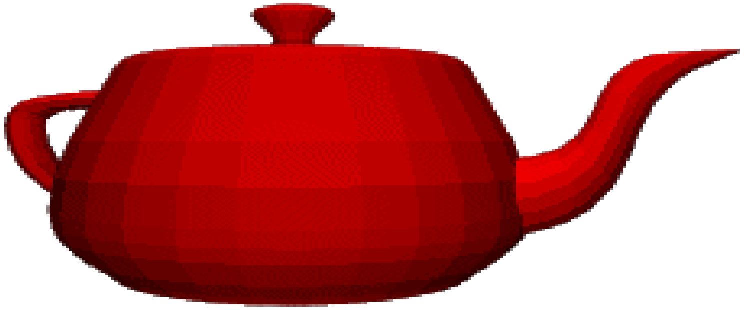
per polygon
"flat shading"

per vertex
"Gouraud shading"

per pixel
"per pixel lighting"
"Phong shading"

Flat Shading （Shade each Polygon）

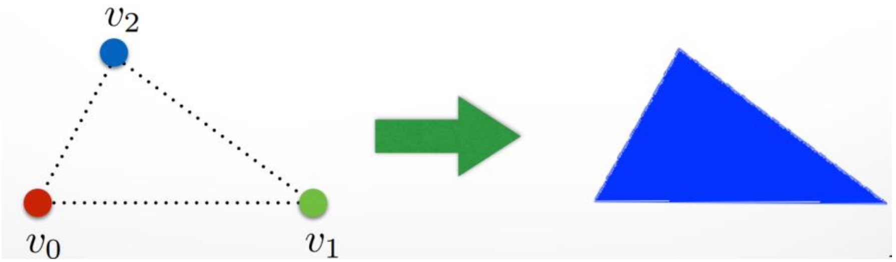
- Simplest approach calculates illumination at a single point for each polygon
- constant color for each polygon, also called constant shading



- obviously inaccurate for smooth surfaces

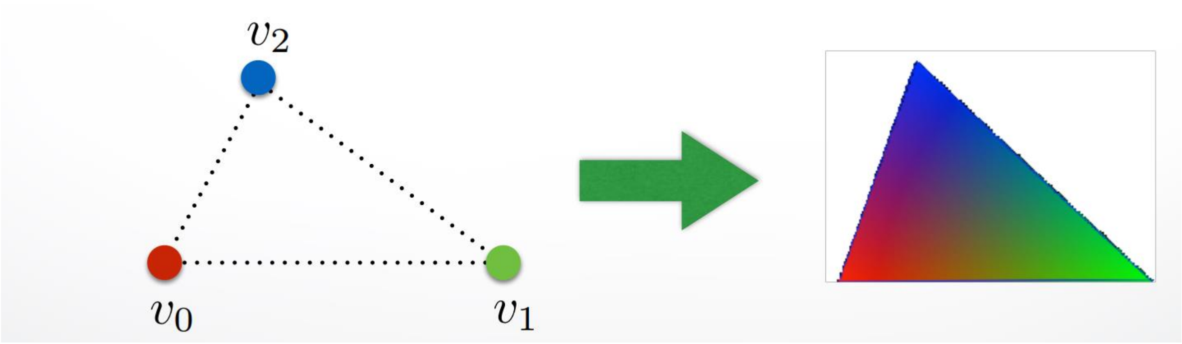
Flat Shading

- Each Triangle face is flat --- a normal vector
- Shading constant across polygon
- Color of last vertex determines interior color
- Only suitable for *very* small polygons



Gouraud Shading (Shade each vertex)

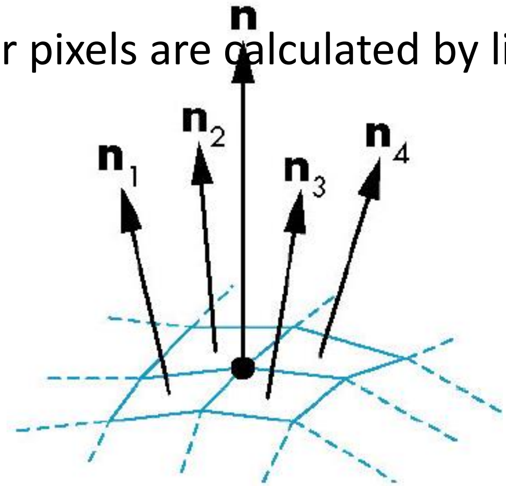
- Each vertex has a normal vector
- Interpolate color in interior
- Much better than flat shading
- More expensive to calculate (but not a problem for modern graphics cards)



Gouraud Shading

- It is an interpolative shading method, also called **intensity interpolation shading** or **color interpolation shading**. Proposed by Gouraud in 1971.
- Involves the following steps
  - Normals are computed at the vertex as the average of the normals of all the faces meeting at that vertex
  - Intensity at each vertex is calculated using the normal and an illumination model
  - For each polygon the intensity values for the interior pixels are calculated by linear interpolation of the intensities at the vertices

$$n = \frac{n_1 + n_2 + n_3 + n_4}{|n_1 + n_2 + n_3 + n_4|}$$



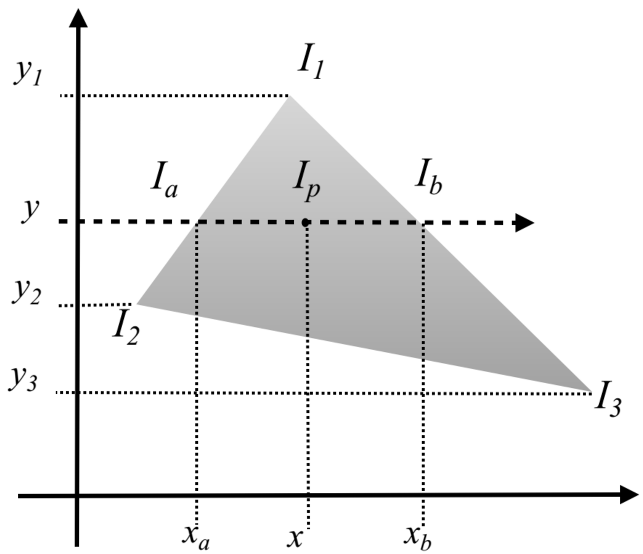
Gouraud Shading

- This is the most common approach
  - Perform Phong Illumination model at the vertices
  - Linearly interpolate the resulting colors over faces
    - Along edgges
    - Along scanlines
  - Intensity I_p at a point is calculated as

$$I_a = I_1 + (I_2 - I_1)\frac{y - y_1}{y_2 - y_1}$$

$$I_b = I_1 + (I_3 - I_1)\frac{y - y_1}{y_3 - y_1}$$

$$I_p = I_a + (I_b - I_a)\frac{x - x_a}{x_b - x_a}$$



Interpolation Across Triangles

Why do we want to interpolate?

- Specify values **at vertices**
- Obtain smoothly varying values **across triangles**

What do we want to interpolate?

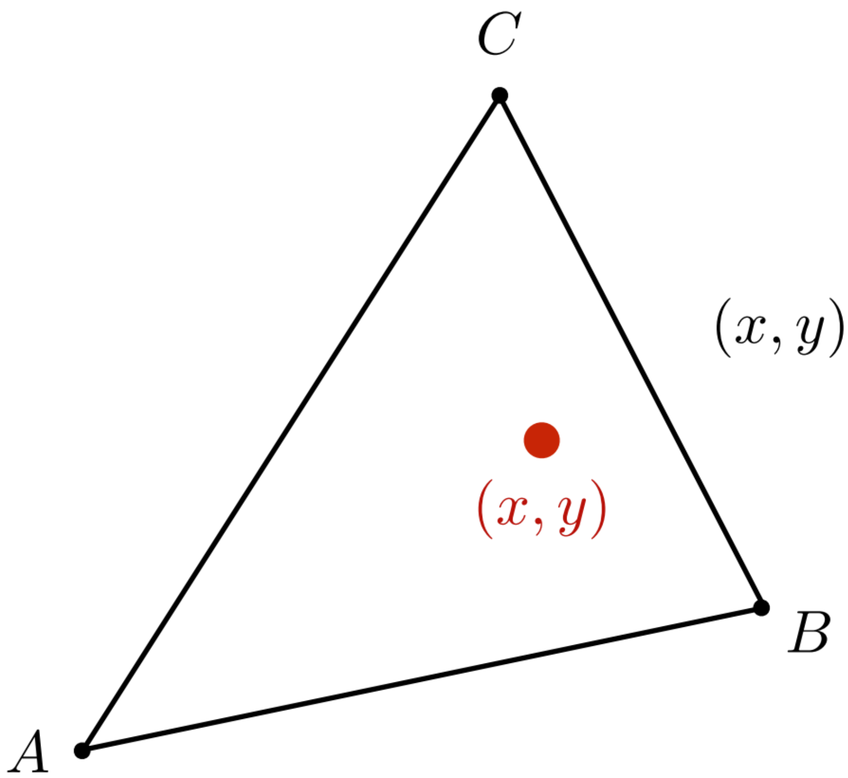
- Texture coordinates, colors, normal vectors, …

How do we interpolate?

- **Barycentric coordinates**

Barycentric Coordinates

A coordinate system for triangles (α, β, γ)



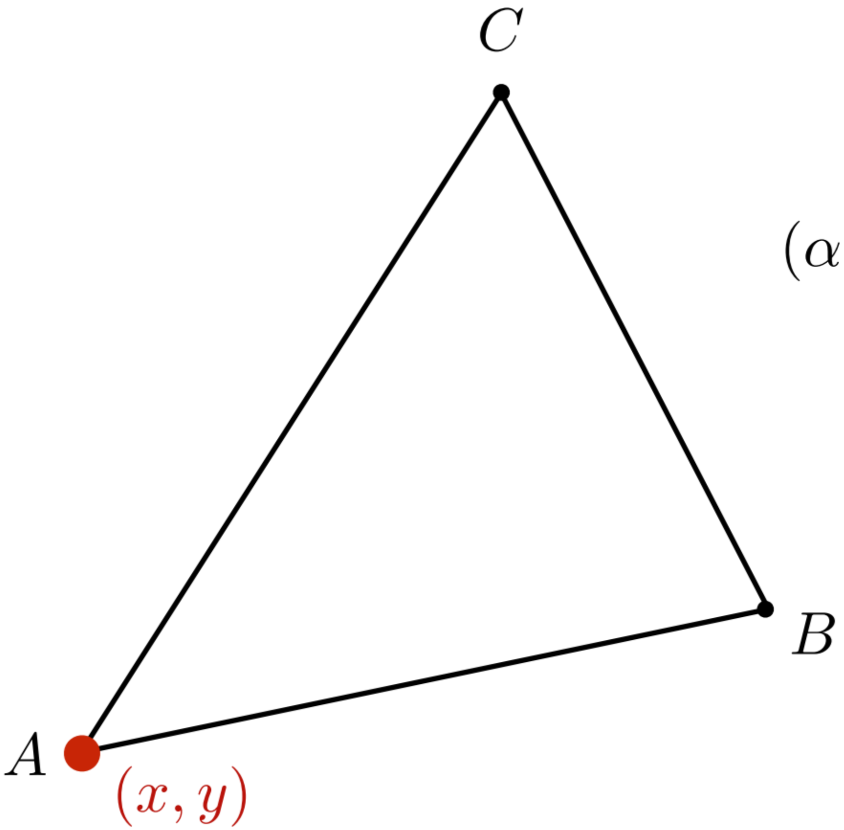
$$(x, y) = \alpha A + \beta B + \gamma C$$

$$\alpha + \beta + \gamma = 1$$

Inside the triangle if all three coordinates are non-negative

Barycentric Coordinates

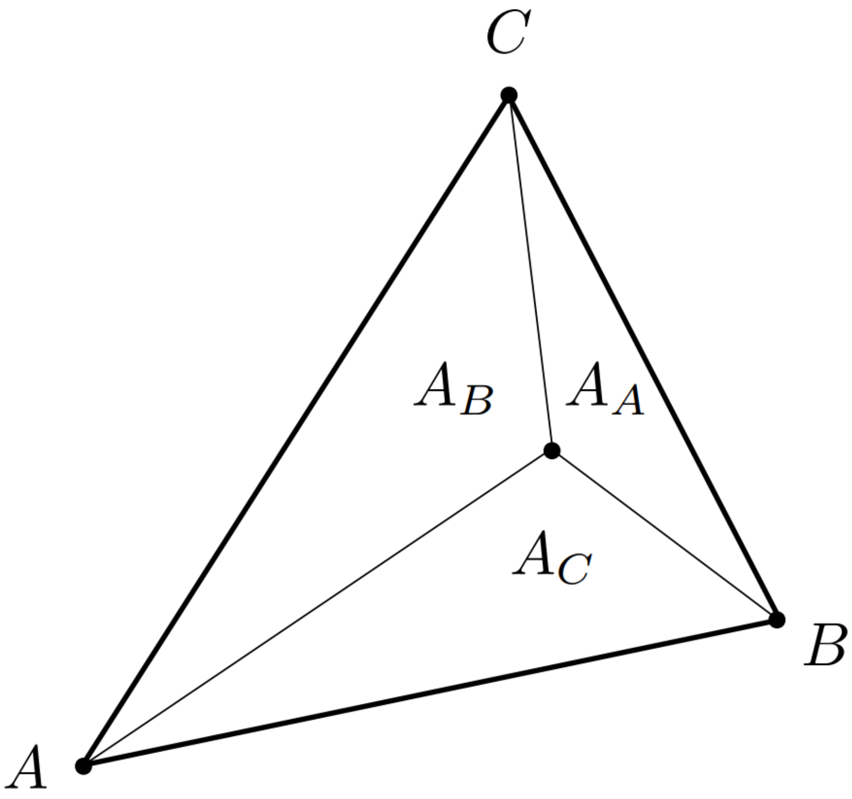
What's the barycentric coordinate of A?



$$
\begin{aligned}
(\alpha, \beta, \gamma) &= (1, 0, 0) \\
(x, y) &= \alpha\, A + \beta\, B + \gamma\, C \\
&= A
\end{aligned}
$$

Barycentric Coordinates

Geometric viewpoint — proportional areas



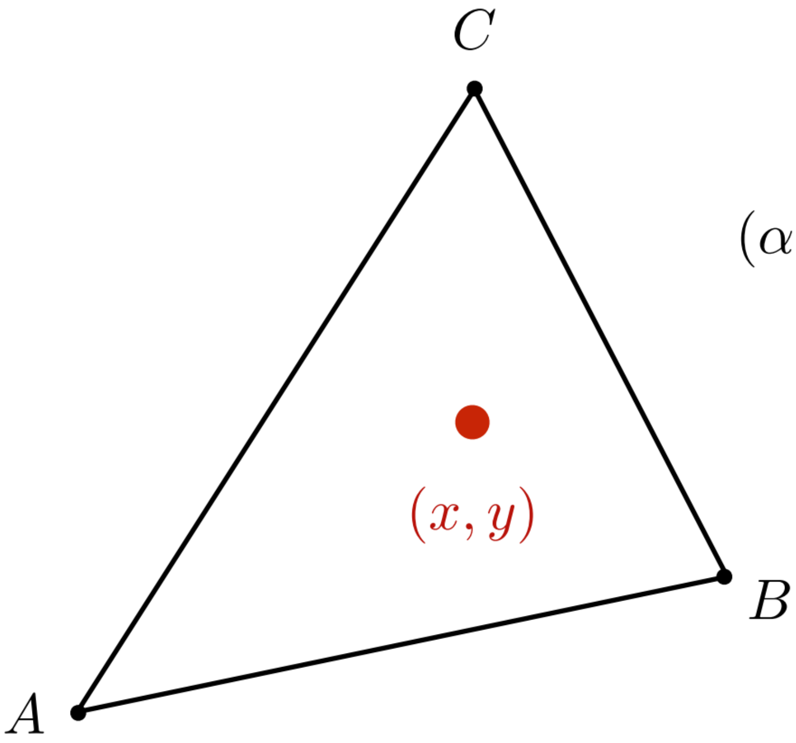
$$\alpha = \frac{A_A}{A_A + A_B + A_C}$$

$$\beta = \frac{A_B}{A_A + A_B + A_C}$$

$$\gamma = \frac{A_C}{A_A + A_B + A_C}$$

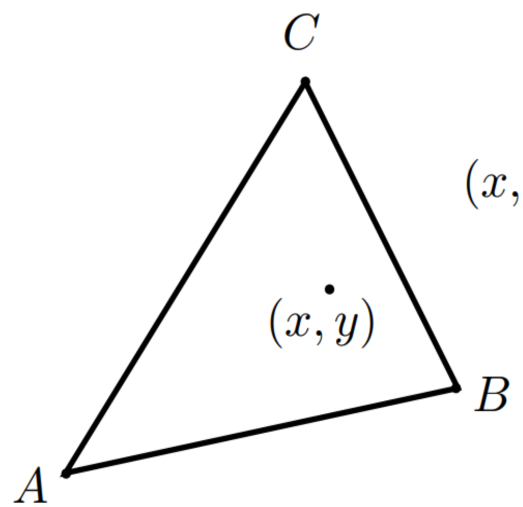
Barycentric Coordinates

What's the barycentric coordinate of the centroid?



$$(\alpha, \beta, \gamma) = \left(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right)$$
$$(x, y) = \tfrac{1}{3}A + \tfrac{1}{3}B + \tfrac{1}{3}C$$

Barycentric Coordinates： Formulas



$$(x,y) = \alpha A + \beta B + \gamma C$$
$$\alpha + \beta + \gamma = 1$$

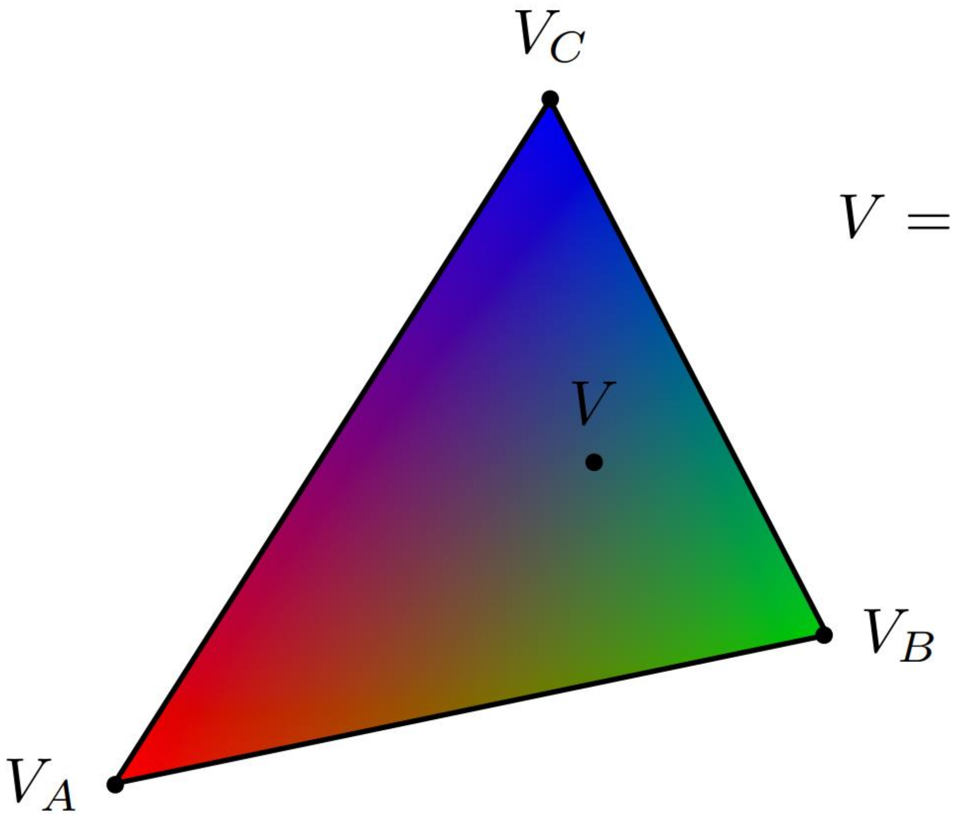
$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$

$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}$$

$$\gamma = 1 - \alpha - \beta$$

Using Barycentric Coordinates

Linearly interpolate values at vertices



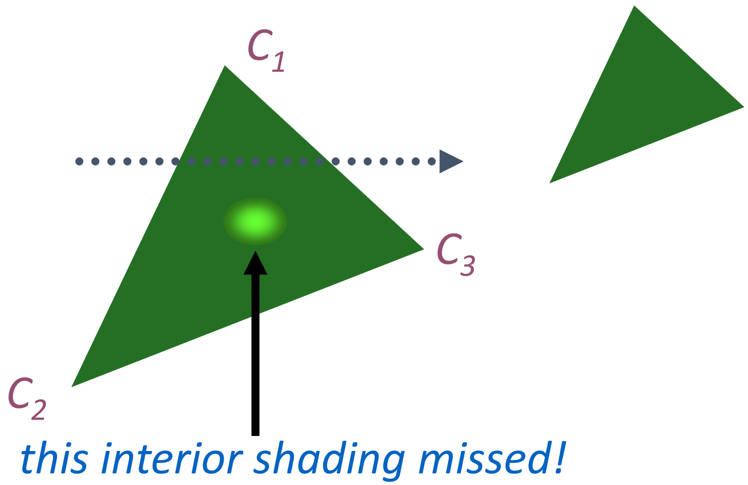
$$V = \alpha V_A + \beta V_B + \gamma V_C$$

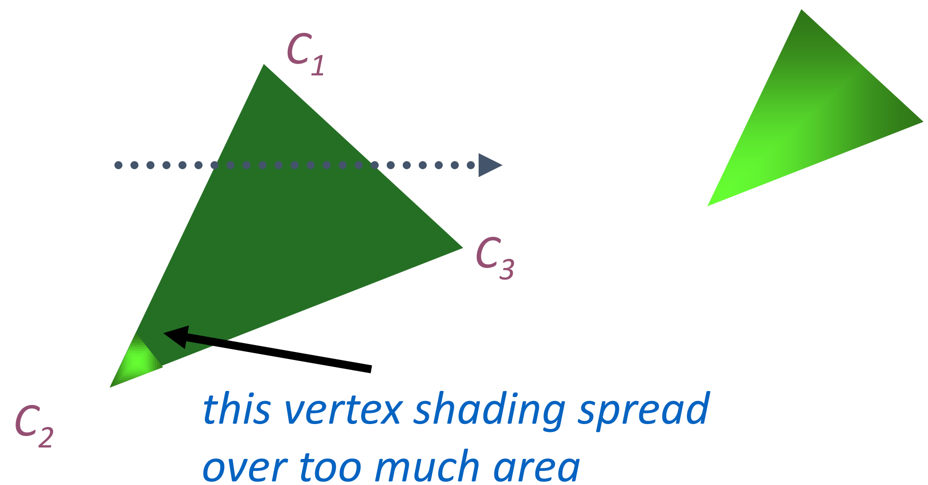
V_A, V_B, V_C can be positions, texture coordinates, color, normal, depth, material attributes…

However, barycentric coordinates are not invariant under projection!

Gouraud Shading Artifacts

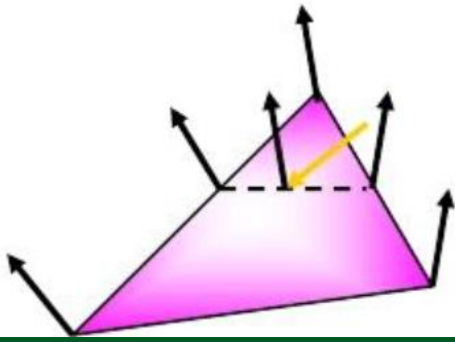
- Might miss specular highlights, if the highlight doesn't fall at the vertex
- Lacks accurate specular component
  - if included, will be averaged over entire polygon





Phong Shading (Shade each vertex)

- Phong shading is not the same as Phong Illumination Model
  - Phong Illumination(Phong Reflectance Model): the empirical model we’ve been discussing to calculate illumination at a point on a surface
  - Phong shading: linearly interpolating the surface normal across the facet, applying the Phong Illumination model at every pixel(also called normal-vector interpolation shading)
    - Same input as Gouraud shading
    - Usually more smooth-looking results:
    - But, considerably more expensive



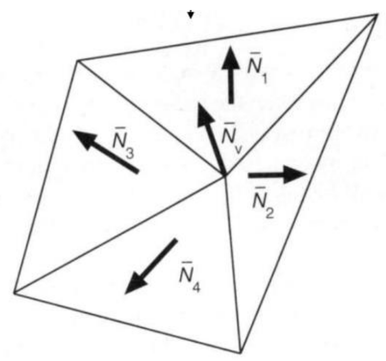
Phong Shading

- Involves the following steps
  1. Normals are computed at the vertex as the average of the normals of all the faces meeting at that vertex
  2. For each polygon
     the value of the normal for the surface occupied by each interior pixel is calculated by linear interpolation of the normals at the vertices

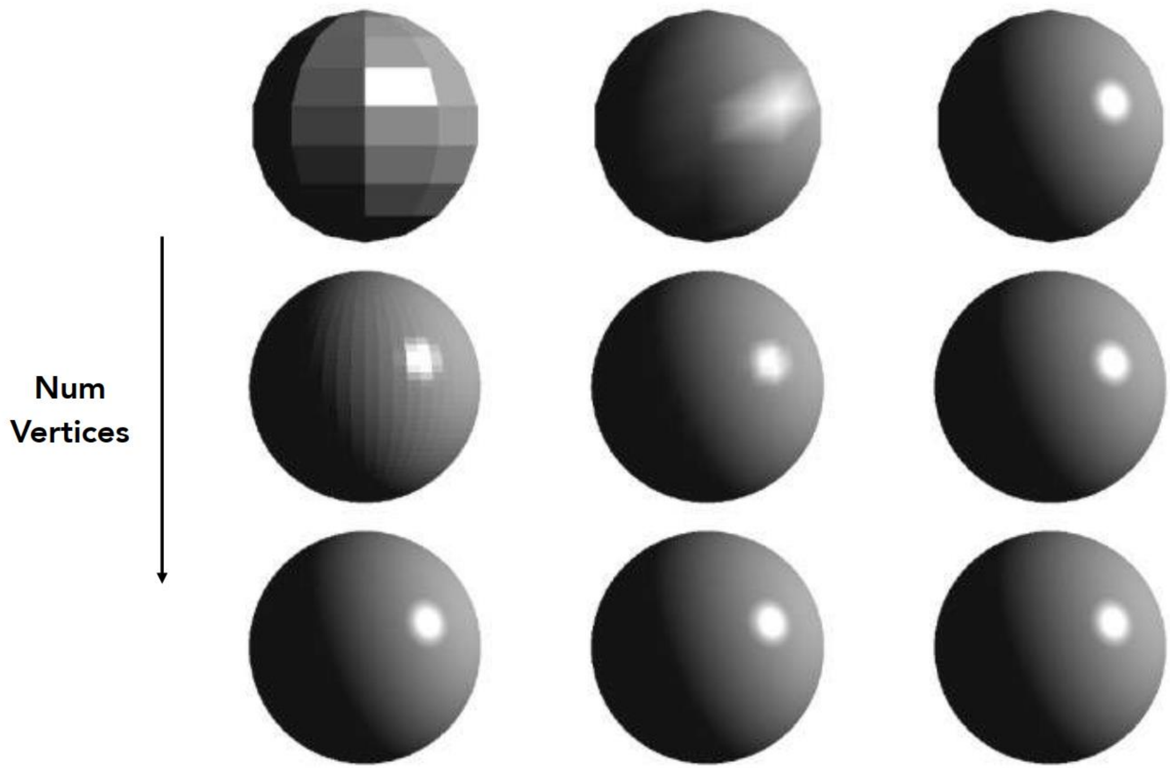
- Interpolation of normals is done exactly like intensity interpolation in Gouraud shading

• Simple scheme: **average surrounding face normals**

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



Shading Frequency: Face, Vertex or Pixel



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

Homework3