Introduction to Computer Graphics

4. Shading 著色、顯像

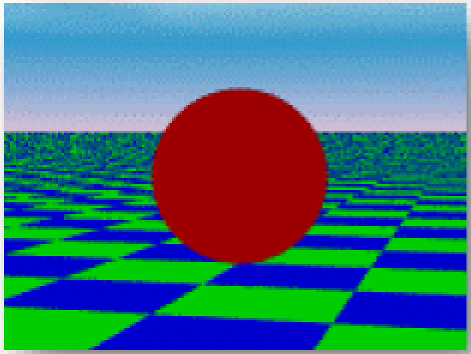
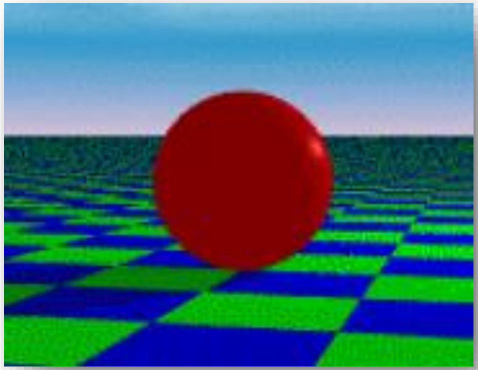
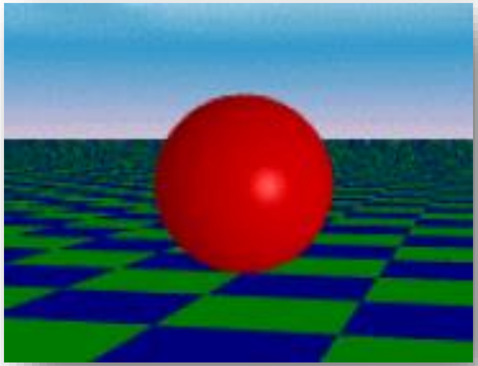
I-Chen Lin
National Chiao Tung University

Textbook: E.Angel, D. Shreiner Interactive Computer Graphics, 6th Ed., Pearson
Ref: D.D. Hearn, M. P. Baker, W. Carithers, Computer Graphics with OpenGL, 4th Ed., Pearson
J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, R. L. Phillips. Introduction to Computer Graphics, Addison-Wesley

Illumination and Shading

- Is it a ball or a plate?

- What color should I set for each pixel?

Why Do We Need Shading?

- Suppose we color a sphere model. We get something like

- But we want

Shading

- Why does the image of a real sphere look like ?

光和表面材質

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

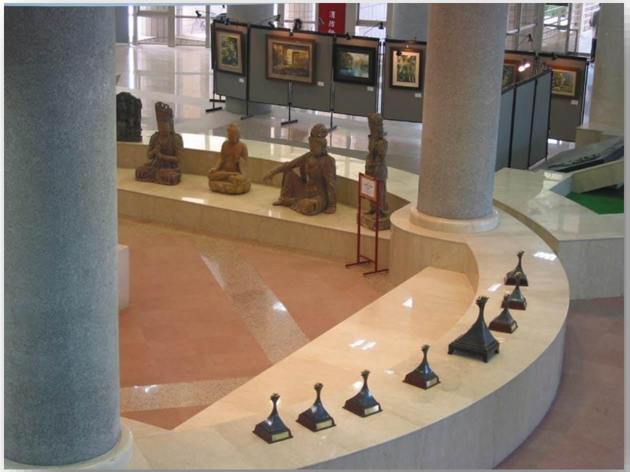
- Need to consider
  - Light sources
  - Material properties
  - Location of the viewer
  - Surface orientation

Illumination and Shading

- Factors that affect the “color” of a pixel.
  - Light sources
    - 發射光譜 Emittance spectrum (color)不同距離能量不同
    - EX : 平行光、點光源 Geometry (position and direction)
    - 定向衰減 Directional attenuation

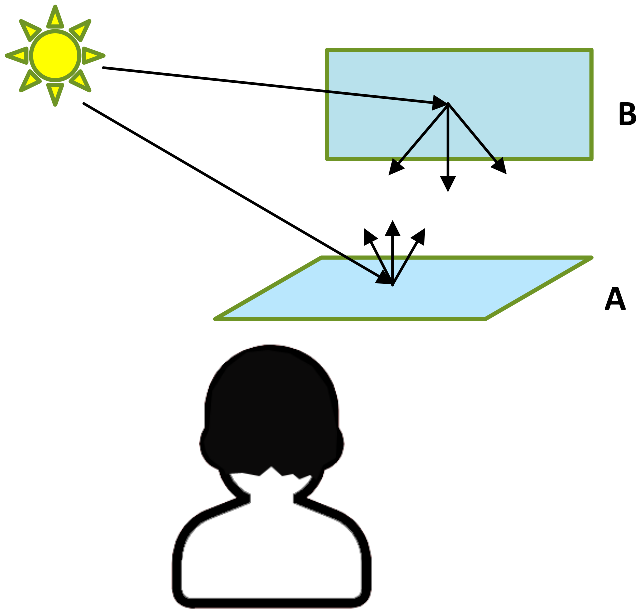
  - Objects’ surface properties
    - EX : 曲面反射角度 Reflectance spectrum (color)
    - Geometry (position, orientation, and micro-structure)
    - Absorption

*物體遮蔽性

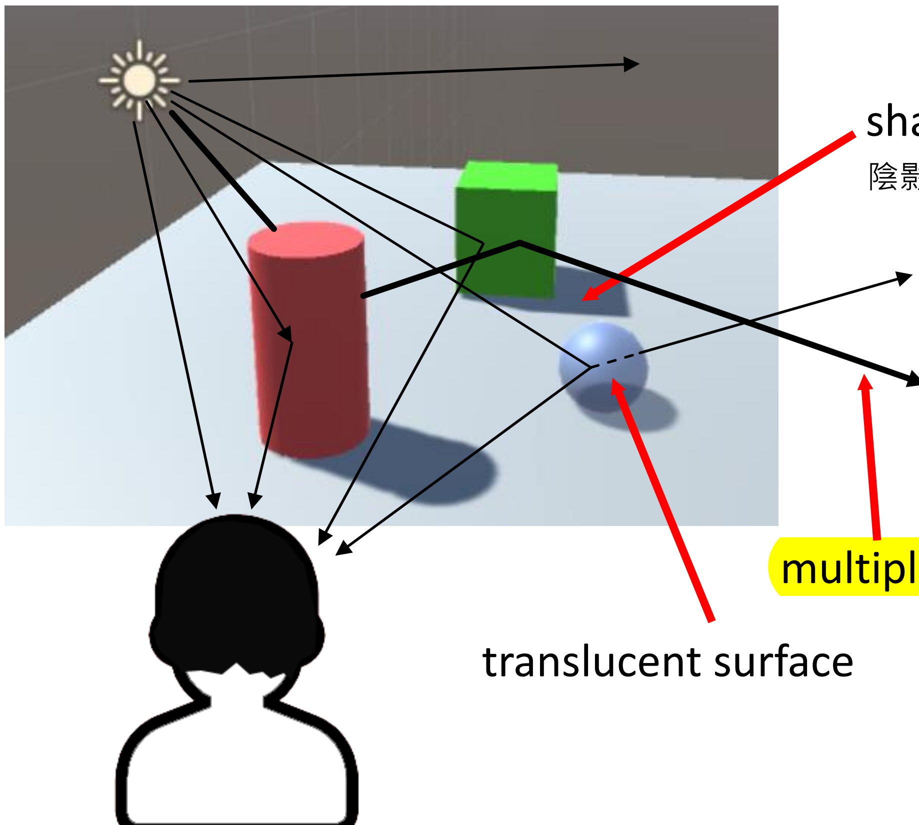


Scattering

- 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 Effects



shadow

陰影不會全黑，因為也有其他東西反射過來的光

multiple reflection

translucent surface

Light-Material Interaction

- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The amount reflected determines the color and brightness of the object
  - A surface appears red under white light because the red component of the light is reflected and the rest is absorbed

  *紅色物體容易反彈紅光、吸收藍光綠光

- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface 反彈結果與物體表面特性有關，如材質、反光特性等

Rendering Equation

- The infinite scattering and absorption of light can be described by the *rendering equation*
  - **[*outgoing*]-[*incoming*] = [*emitted*]-[*absorbed*]**

    物體自發的光　反射光 = incoming - absorb
  - **[*outgoing*] =[*emitted*]+[*reflected*](+[transmitted])**
  - Cannot be solved in general
  - Ray tracing is a special case for perfectly reflecting surfaces

全域視角，考慮整個場景

- Rendering equation is global and includes
  - Shadows
  - Multiple scattering from object to object

Local vs Global Rendering 顯像

- Correct shading requires a global calculation
  - 不相容的
    Incompatible with a 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

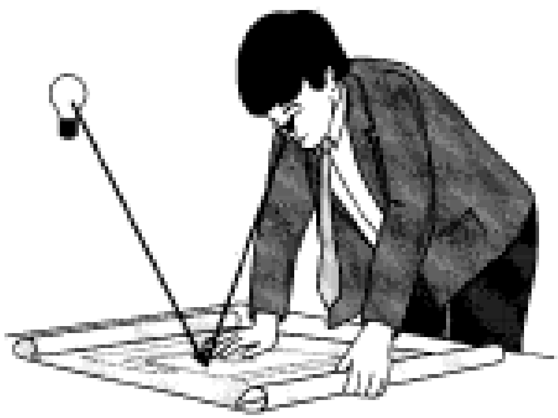
*僅考慮這兩種情況(一次反射) :
light->material->eye
light->eye

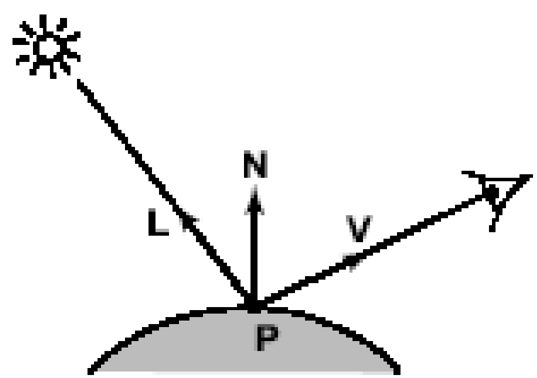
- Adequate for real-time graphics.
- No inter-reflection, no refraction, no realistic shadow ….

不考慮物體間的交互反射　　折射

0.7 absorb
0.7 absorb
0.3 reflect
0.3 reflect ……
0.7 absorb
0.3 reflect

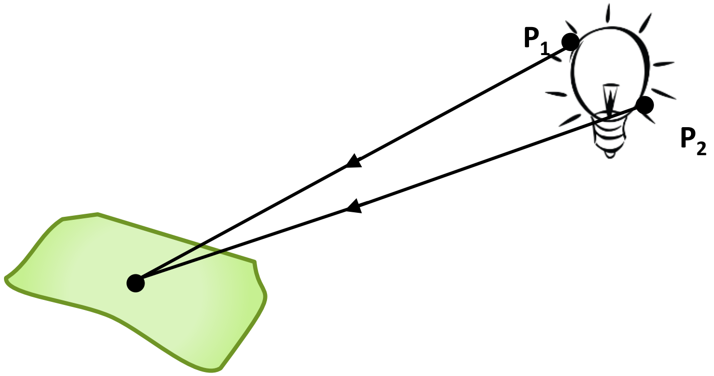
等比係數
直線下降





Light Sources

- General light sources are difficult to simulated
  - because we must integrate light coming from all points on the source.

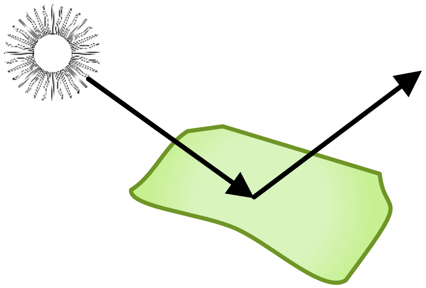


Simple Light Sources

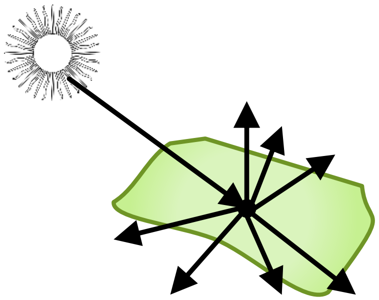
- Point source **點光源** : 光往四面八方送or光源放在無窮遠變成平行光(directional light)
  - Model with position and color
  - Distant source = infinite distance away (parallel)
- Spotlight **探照燈** : 縮小照光角度，一定角度內才照得到光
  - Restrict light from ideal point source
- Ambient light **環境光** : 為了補償光彈了很多次的結果(無方向性的光)
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces

Surface Types

- The smoother a surface, the more reflected light is concentrated in the direction 越平滑的表面(完美的表面)，反射能力越好
- A very rough surface scatters light in all directions



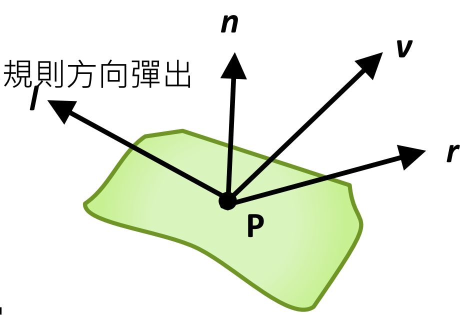
smooth surface



rough surface

Phong Reflection Model **光照模型**：用簡單數學式描述光路徑

- A simple model that can be computed rapidly
- Has three components 主要計算diffuse + specular
  - Ambient 環境光
  - Diffuse **散射**：光打到粗糙表面會往不規則方向彈出
  - Specular 鏡面反射光



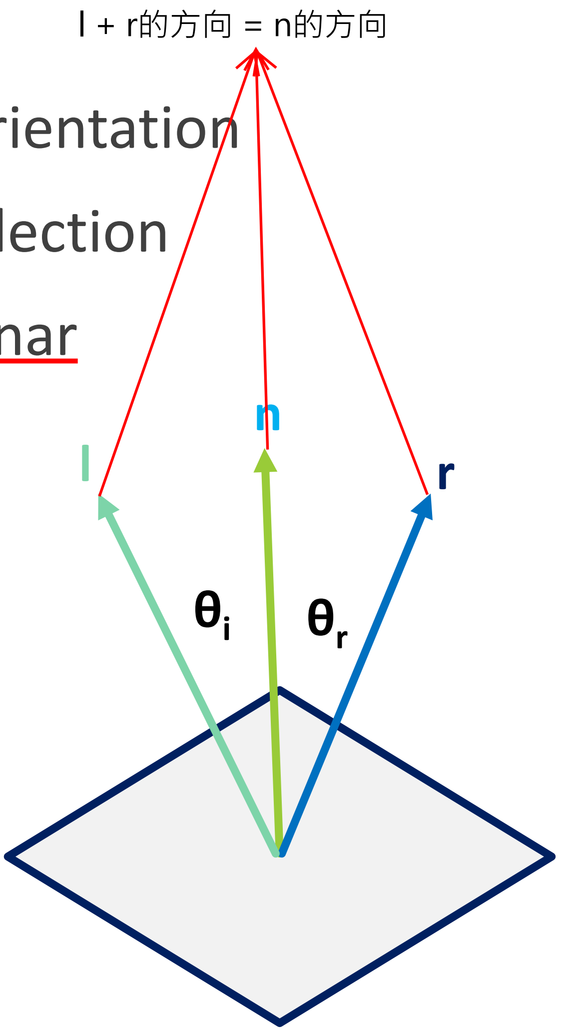
- Uses four vectors 都是單位向量
  - To source ***l***
  - To viewer ***v***
  - Normal ***n***
  - Perfect reflector ***r***

r：完美反射方向

Ideal Reflector

- Normal is determined by local orientation
- Angle of incidence = angle of reflection
- The three vectors must be coplanar

共面



已知n(法向量，通常帶在obj裡), l(光線向量)，
l + r = 2(l ˙ n) n

$$\mathbf{r} = 2\,\underbrace{(\mathbf{l}\cdot\mathbf{n})}_{\text{scalar : cos}}\,\overset{\text{vector}}{\underline{\mathbf{n}}} - \mathbf{l}$$

Ambient Light 環境光

- The result of multiple interactions between (large) light sources and the objects in the environment.

反射係數 : 同一個物體會有不同的反射係數，反射光的比例

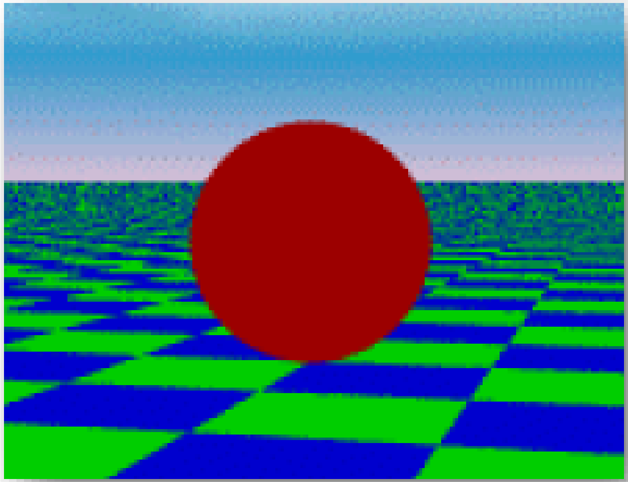
- $I_{ambient} = \boxed{K_a} \cdot \boxed{I_a}$

人為設定

EX :

0.07	0.7	0.1
0	0	0.1
0	0	0.1

某一個點的結果

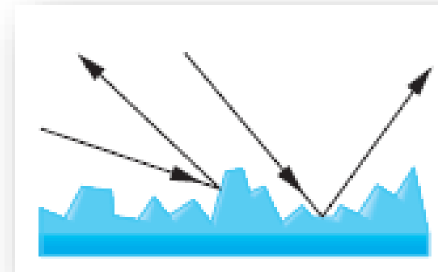


RGB : 這個例子三個色一樣就是白光
(三種顏色的光強度相同)
其中一個數值比較大的話就會偏RGB某一種顏色

Diffuse Reflection

散射光

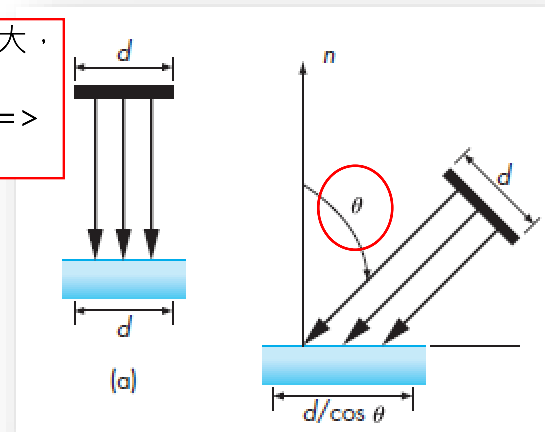
表面不光滑



theta : 與法線的交角

- Light scattered equally in all directions
- Reflected intensities vary with the direction of the light.
- Lambertian Surface
  - Perfect diffuse reflector
  - reflected light ~cos θ_i
  - $\cos \theta_i = \mathbf{l} \cdot \mathbf{n}$ if vectors normalized

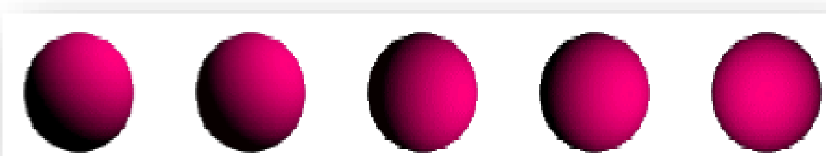
實際收到的光變大 : d/cos，面積變大，
單位表面積的光通量變小
(光斜斜打下去被分散在較大的面積=>
單位面積接收到的能量變少)



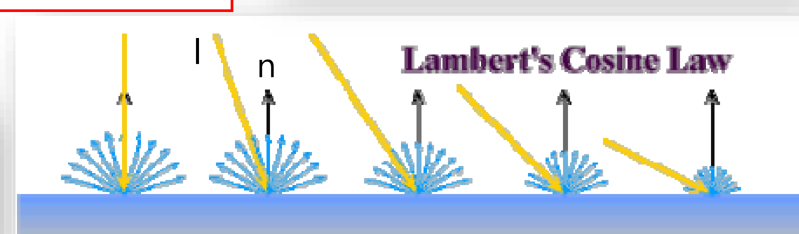
- $I_{diffuse} = \boxed{K_d} \cdot I_d\,(n \cdot l)$

最後看到的光

Kd : scaler反射比例，可以隨意假設，
Id : 光強度，可以隨意假設
n dot l : projection，光和平面的關係



可觀察物體表面情況

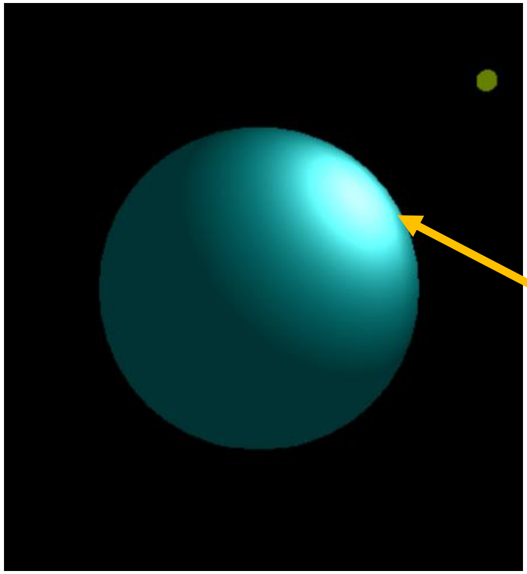


傾斜角越小越亮

傾斜角越大越暗

Specular Surfaces 鏡面反射

- Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors) **完美鏡反射**：入射角 = 反射角
- Incoming light being reflected in directions concentrated close to the direction of a perfect reflection 鏡反射的光集中在一個角度



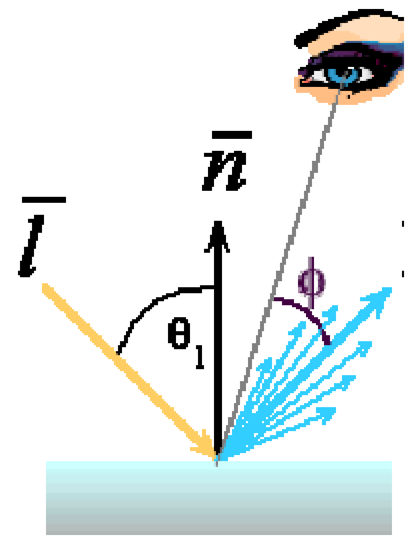
過曝：白白那圈

specular highlight

Modeling Specular Reflections

view invariant : 光會跟著視角改變
vs. diffuse : 光不跟著視角改變

- Phong proposed



但是邊邊會有能量分散，
之後快速遞減

v dot r

$$I_r \sim k_s\, I\, \boxed{\cos^a \phi}$$

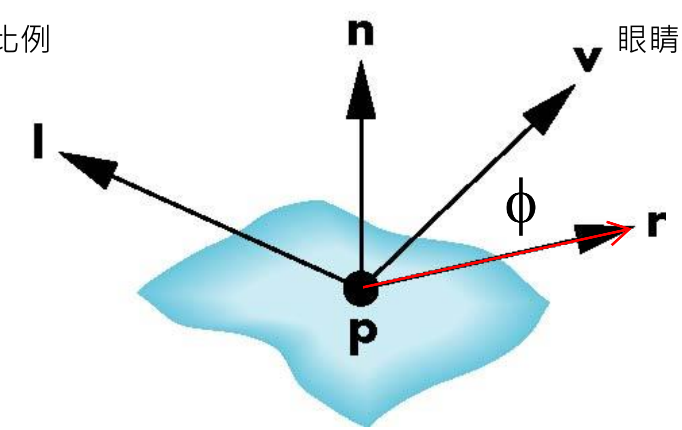
描述不同光的衰減比例

shininess coef

reflected
intensity

incoming intensity

absorption coef



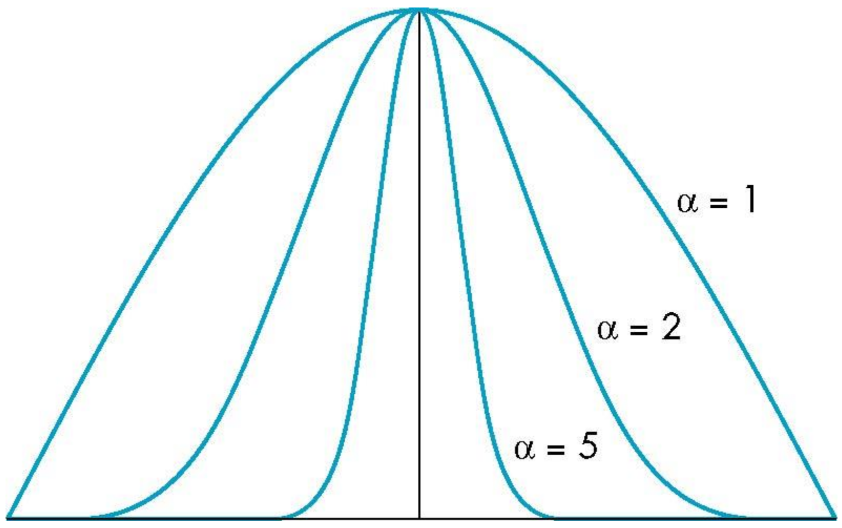
眼睛到底收到多少光餒

主要反射方向

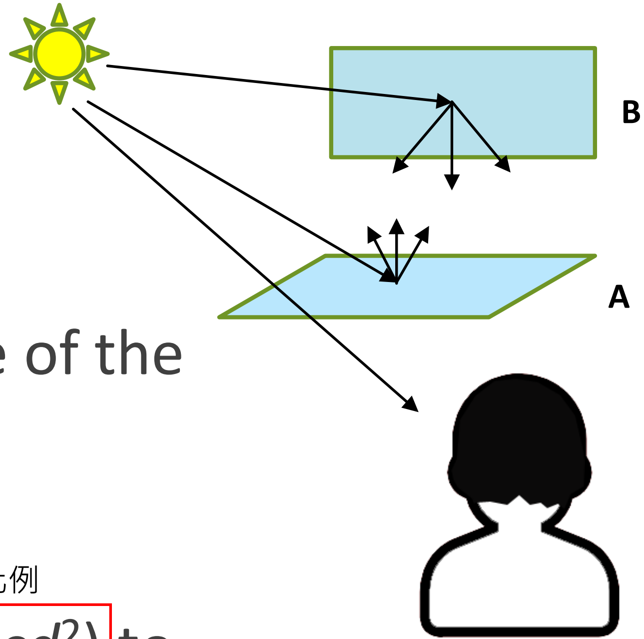
The Shininess Coefficient

霧面specular低 vs. 光面

- Values of a between 100 and 200 correspond to metals. 光反射得越早越聚焦
- Values between 5 and 10 give surface that look like plastic. 光反射得越慢的越平順



Distance Terms



- Inversely proportional to the square of the distance between them

調整光衰減的比例

- Add a factor of the form $\boxed{1/(a + bd + cd^2)}$ to the diffuse and specular terms

這兩種情況時再開distance term就好惹

- The constant and linear terms soften the effect of the point source

Coefficients

每一盞燈都要賦予光係數，不一定加起來要是1，
diffuse要大，specular次之，ambient最小

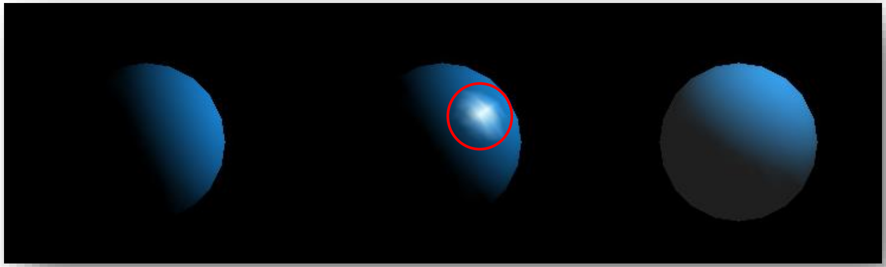
- 9 coefficients for each point light source
  - $\boxed{I_{dr}, I_{dg}, I_{db},}$ $\boxed{I_{sr}, I_{sg}, I_{sb},}$ $\boxed{I_{ar}, I_{ag}, I_{ab}}$

  EX : 0.7(大) 0.2(中) 0.1(小)
- Material properties
  - Nine absorption/reflection coefficients
    - $\boxed{k_{dr}, k_{dg}, k_{db},}$ $\boxed{k_{sr}, k_{sg}, k_{sb},}$ $\boxed{k_{ar}, k_{ag}, k_{ab}}$

    EX : 0.7(大) 0.2(中) 0.1(小)
  - Shininess coefficient α

Adding up the Components

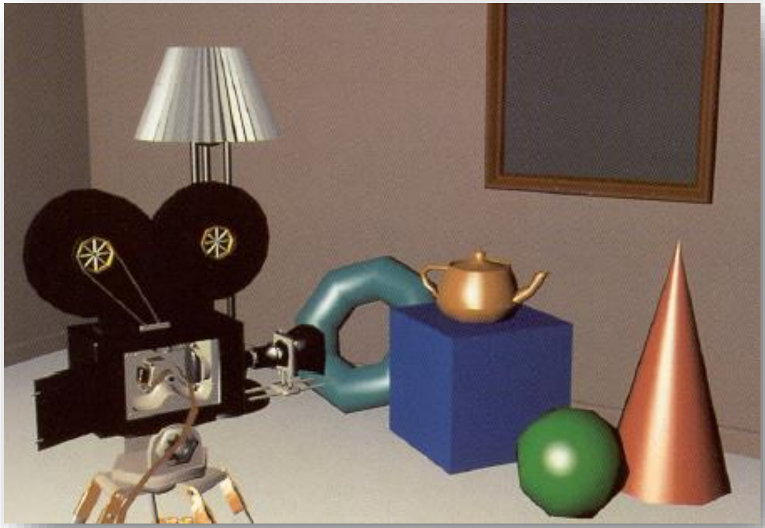
- A primitive virtual world with lighting can be shaded by combining the three light components .
- $$I = I_{ambient} + I_{diffuse} + I_{specular}$$
  $$= k_a I_a + k_d I_d(\mathbf{l} \cdot \mathbf{n}) + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha}$$



diffuse

diffuse + specular 亮點

diffuse + ambient



Modified Phong Model

- Problem: In the specular component of Phong model, it requires the calculation of a new reflection vector and view vector for each vertex

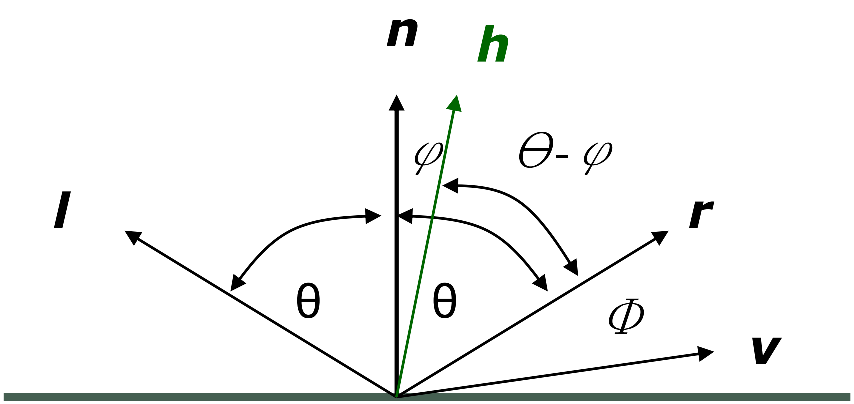
$$\boldsymbol{r} = 2\,(\boldsymbol{l} \cdot \boldsymbol{n}\,)\,\boldsymbol{n} - \boldsymbol{l}$$

- Blinn suggested an approximation using the halfway vector that is more efficient

Using the Halfway Angle

- Replace $(\mathbf{v} \cdot \mathbf{r})^a$ by $\boxed{(\mathbf{n} \cdot \mathbf{h})^b}$ n和h間的夾角：小phi
- b is chosen to match shineness
- Note that halway angle is half of angle between **r** and **v** if vectors are coplanar

平均向量 Halfway vector : $\mathbf{h} = (\mathbf{l} + \mathbf{v}) / |\mathbf{l} + \mathbf{v}|$



h的左　　h的右

$$\theta + \varphi = \theta - \varphi + \phi$$

$$\boxed{2\varphi = \phi}$$

Using the Halfway Angle

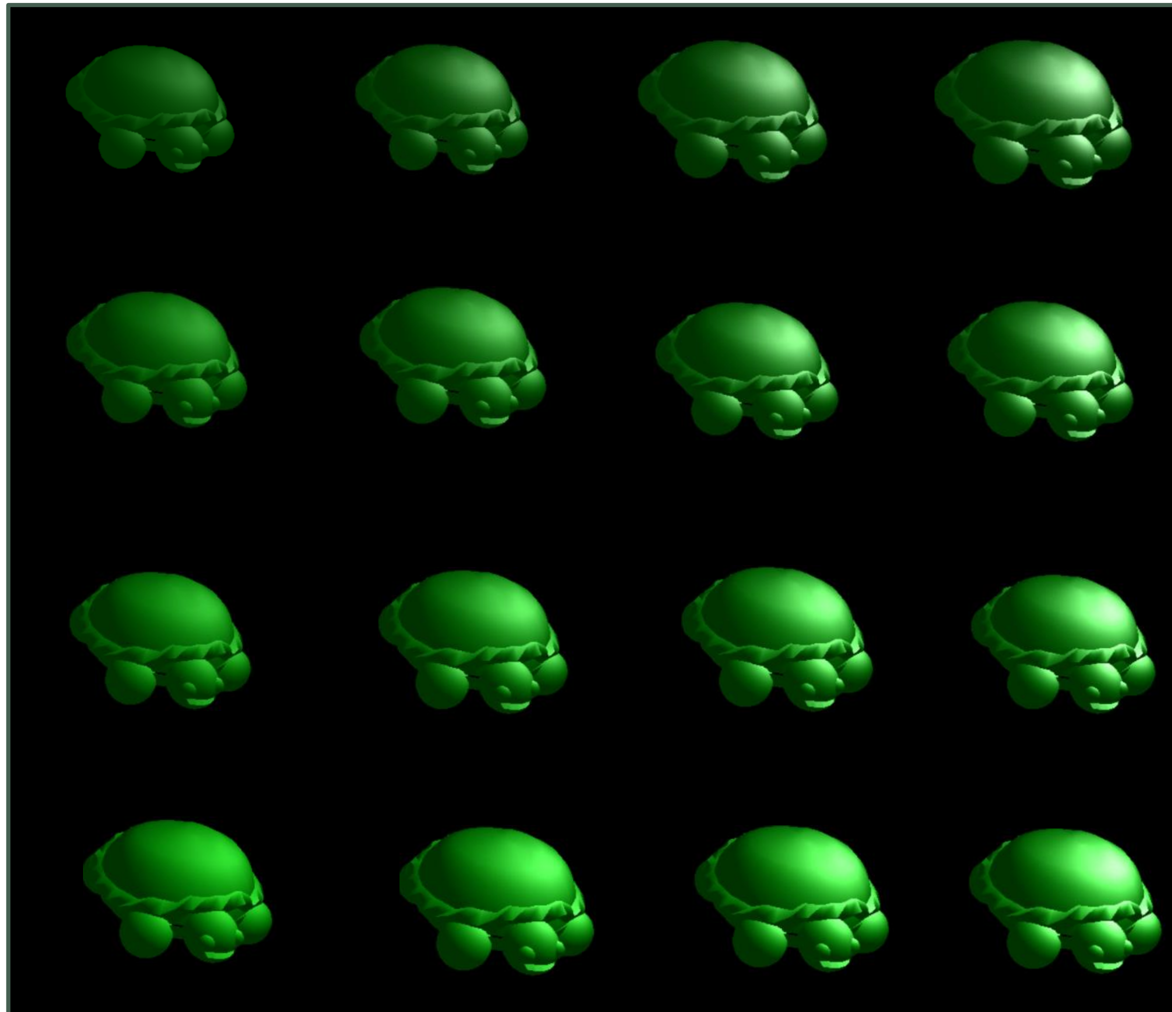
- Resulting model is known as the *modified Phong* or *Blinn* lighting model

- Specified in OpenGL standard and most real-time applications

Example

Turtles with different parameters in the modified Phong model.

Beta = 3.6



Computation of Vectors

- **l** and **v :** specified by the application
- **r:** computed from **l** and **n**
- Determine **n**
  - Depending on underlying representation of surface
  - OpenGL leaves determination of normal to application
    - Exception for GLU quadrics and Bezier surfaces

Plane Normals

- Equation of plane: $ax+by+cz+d = 0$ (a b c) x = 0

  法向量垂直

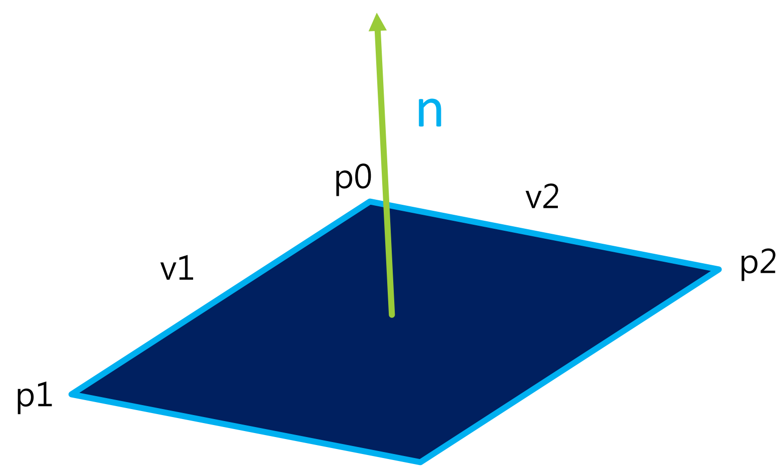
  y

  z

- Normal can be obtained by **小心法向量正反面**：法向量要朝外不能朝內

v2 v1

$$\boxed{\mathbf{n} = (p_2 - p_0) \times (p_1 - p_0)}$$

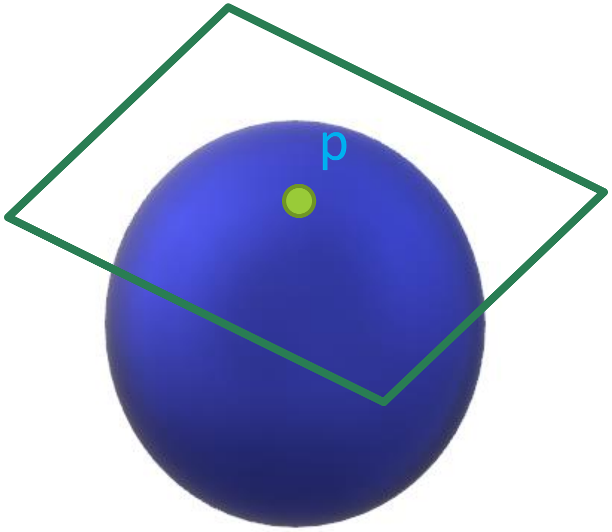


Normal to Sphere

- Implicit function $f(x,y.z)=0$

- Normal given by gradient
- Sphere

比較不實用 ▶ $\boxed{n = [\partial f/\partial x, \partial f/\partial y, \partial f/\partial z]^T}$



Parametric Form

- For sphere

  u : 仰角
  v : 水平角

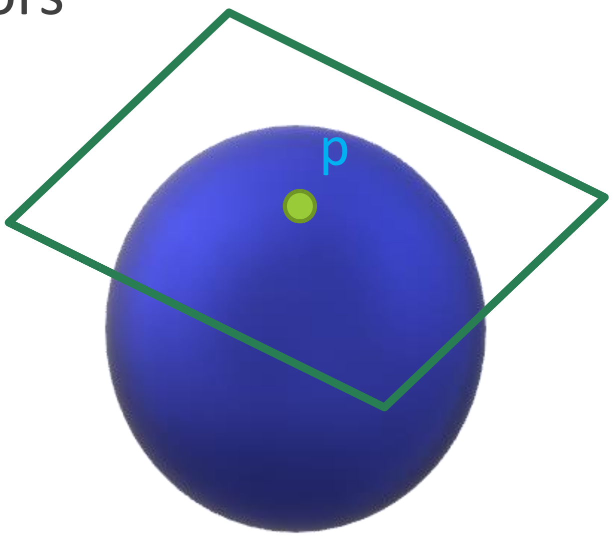
$$x = x(u,v) = \cos u \sin v$$
$$y = y(u,v) = \cos u \cos v$$
$$z = z(u,v) = \sin u$$

- Tangent plane determined by vectors

$$\partial \mathbf{p}/\partial u = [\partial x/\partial u, \partial y/\partial u, \partial z/\partial u]^T$$
$$\partial \mathbf{p}/\partial v = [\partial x/\partial v, \partial y/\partial v, \partial z/\partial v]^T$$

- Normal given by cross product

$$\mathbf{n} = \partial \mathbf{p}/\partial u \times \partial \mathbf{p}/\partial v$$



著色

Polygonal Shading

- Practical implementation to fill color within a polygon.
  - Flat shading
  - Gouraud shading (smooth shading)

    conventional graphics : OpenGL
    =>define pipeline
  - Phong shading

    可以塞小程式

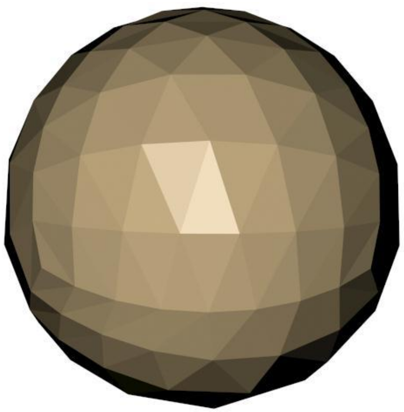
Flat Shading

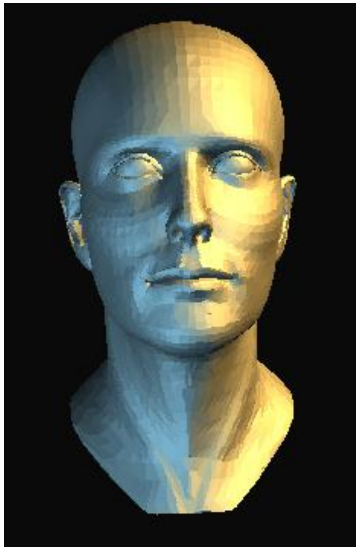
GL_FLAT()：多拿來 debug

- Flat or constant shading. 整片塗同樣的顏色，三角形內部constant

- Assume l, n, v are constant for a polygon.
  - Shading calculation: once for each polygon.

I = KaIa + KdId(n dot l) + KsIs(v dot r)^d





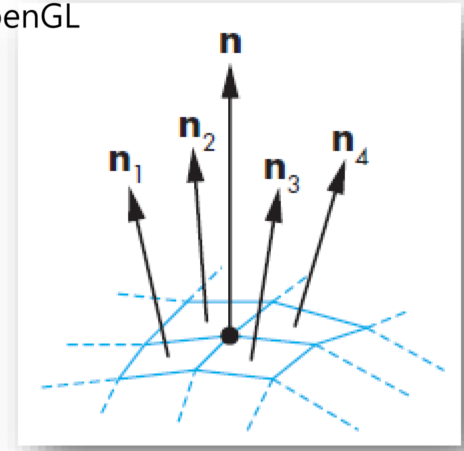
Gouraud Shading

smooth shading(平滑著色法) in OpenGL

從平面法向量找到一個頂點法向量(n : 可以代表周圍的向量)

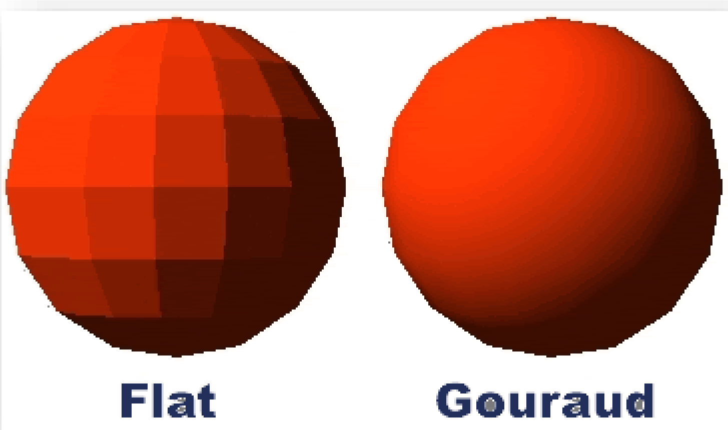
- Find average normal at each vertex
- Apply Phong lighting model at each vertex
- Interpolate vertex shades across each polygon

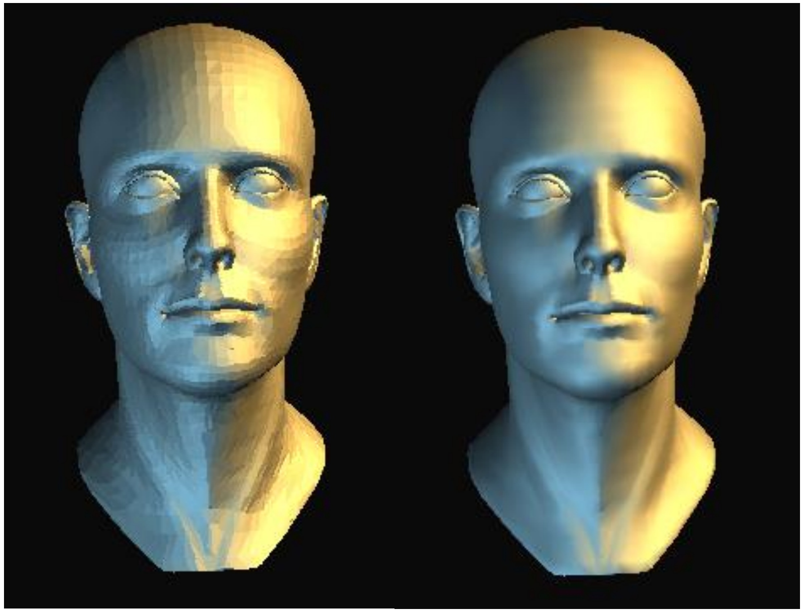
取頂點向量來著色，再使用內插將中間平面著色，
可以按照面積用比例計算(weight : 權重)



$$\mathbf{n} = (\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4)/\,|\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4|$$

邊界仍有多邊形





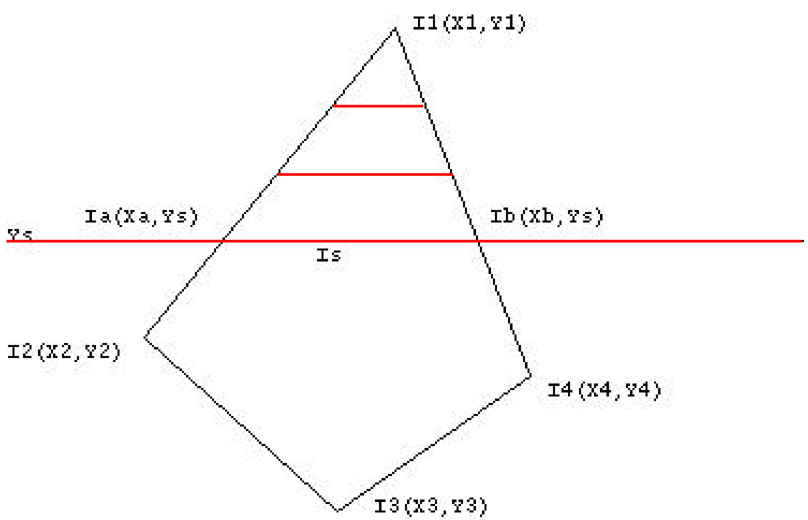
Gouraud Shading (cont.)

把多邊形打斷成三角形

$$I_a = \frac{1}{y_1 - y_2}\left[I_1(y_s - y_2) + I_2(y_1 - y_s)\right]$$

$$I_b = \frac{1}{y_1 - y_4}\left[I_1(y_s - y_4) + I_4(y_1 - y_s)\right]$$

$$I_s = \frac{1}{x_b - x_a}\left[I_a(x_b - x_s) + I_b(x_s - x_a)\right]$$



scanned line : 先算左右再算中間

vs. phong reflection : diffuse, ambient...光的問題

Phong Shading

因Gouraud Shading無法正確反應光的顏色 :
今天一個平面上有三道光反射，但是如果外圍的光強度小於中間的光，
Gouraud Shading無法做到中間特別亮的現實情況，
因為中間的顏色怎麼樣內插都不會超過三角形的三頂點(除非三角形切夠小)

假設normal延續 vs. Gouraud Shading : 假設三角形的邊連續

- Find vertex normals
- Interpolate vertex normals across edges
- Find shades along edges
- Interpolate edge shades across polygons

用三角形頂點normal做內插，
再利用phong reflection
去算三角形中心的顏色

內插的部分夾在兩個for loop裡
=>運算量大、速度慢
vs. Gouraud Shading : 內插在最外圈

phong

flat smooth

Problems about Interpolated Shading on Polygonal Models

- Polygonal silhouette? 多邊形邊角明顯

- Perspective distortion?

影響較少

同一個點旋轉完的顏色可能會變(演算法的問題)
用透視投影法算會扭曲(在投影幕上的2D比例和實際的3D比例不一樣)

- Orientation dependence?

打斷T-junction : 不要讓一個點出現在一個邊界上(避免共點)

屋頂每個點的normal會被重算然後都朝上，變成平面
解決法是每個面在同一個點也用獨立的normal

同個點給不同normal=>斷面感(O
同個點一個normal=>屋頂變成平面(X

- Problems at shared vertices?

多邊形頂點共用、使平面連續不斷面、每個頂點都要有一個normal；
而非平面要有normal，否則會造成normal不連續、斷面產生

- Unrepresentative vertex normals? 底點內插有時候不適合