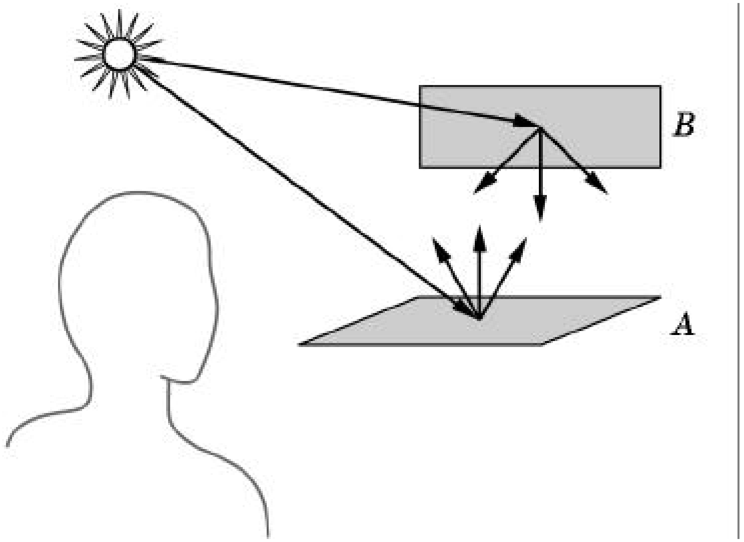
Shading

Shading

- realistic computer graphics 의 첫 걸음
  - gradation of colors
    - 색상이 부드럽게 변해 가야
  - what is needed ?

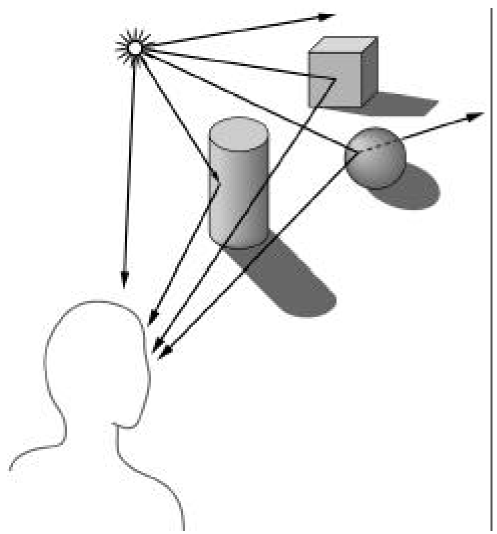


    - light : 광원
    - matter (= material) : 물체 표면의 특성
    - optics (광학) or physics

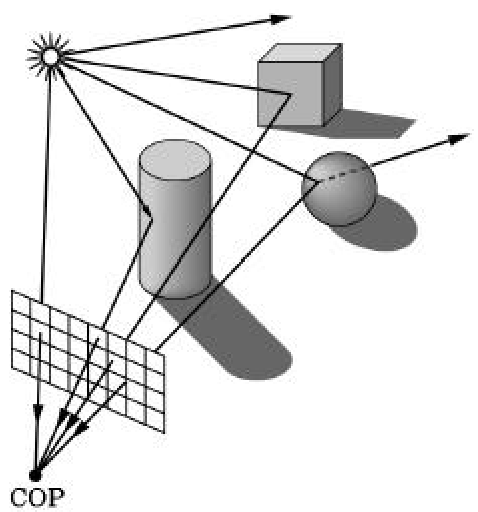
6.1 Light and Matter

Light and Matter

- Light representation in computer graphics
  - based on three color theory : Red, Green, Blue
- Matter
  - material properties determine which wavelengths are absorbed or reflected to some extent



in real world



in computer graphics

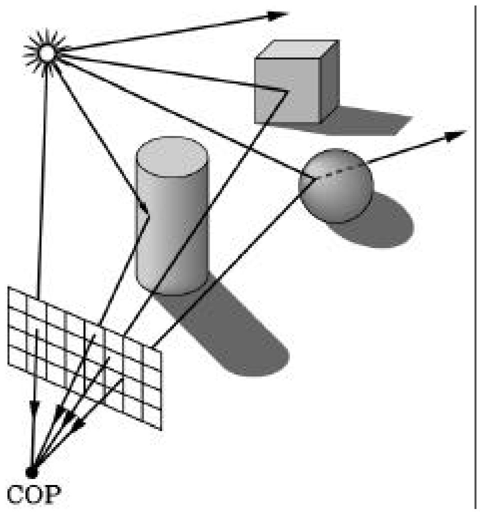
Rendering Equation

- J. T. Kajiya, “The Rendering Equation”, *SIGGRAPH ’86*, pp.143–150, (1986).
  - computer graphics 에서의 종합적인 rendering 방법
  - optics 이론의 approximation
- rendering equation의 다양한 해석 가능
  - 다양한 illumination model

light source, matter, 기타 정보

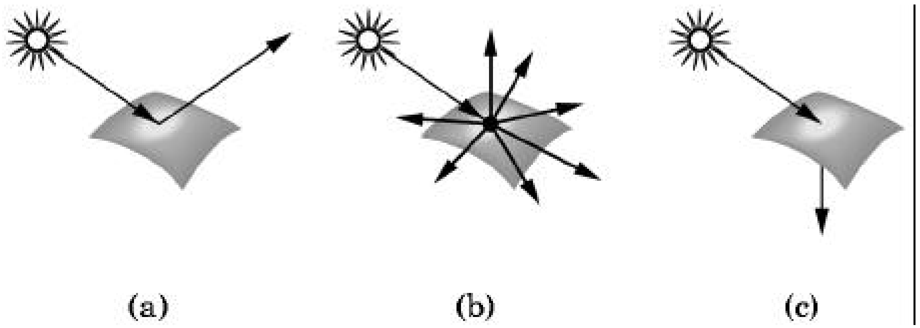
rendering equation

pixel 값의 결정



Illumination Model

- = shading model, lighting model
  - a method to calculate
    the intensity of light
    at a given point on the surface of an object
- Light-Material interactions
  - material **특성에 따라**…



specular surface **정반사**　　diffuse surface **난반사**　　translucent surface **굴절**

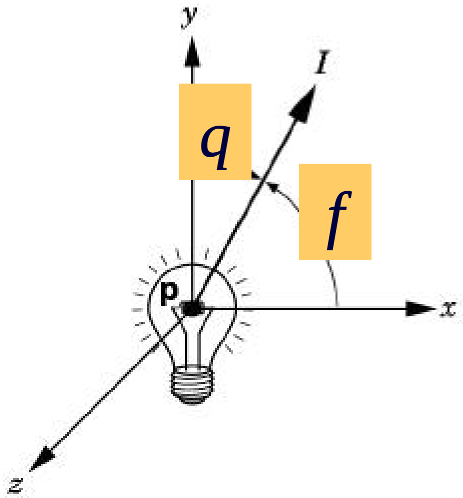
6.2 Light Sources

Light Sources

- light 관점에서의 물체 표면에서는
  - self-emission : 실제로 빛을 발생 (광원)
  - reflection : 빛을 반사
- light source (광원 光原)
  - 광의 : 모든 물체는 광원
  - 협의 : 빛을 발생하는 물체만
- illumination function

  $I(x, y, z, q, f, l)$

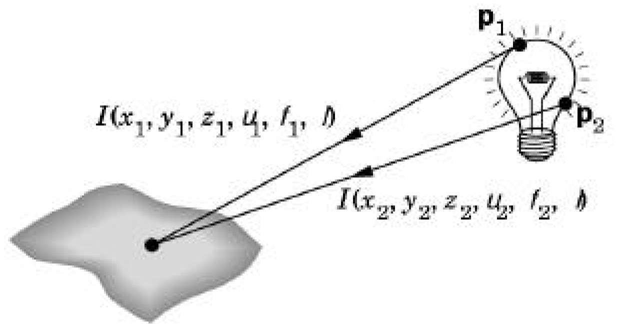
  - point (x, y, z) 에서
    (q, f) 방향으로 가는
    파장이 l 인 빛의 세기



Light Sources

- additive behavior of light
  - **물체의 표면 색** : integral of all lights
  - color : R, G, B **각각 따로 계산**

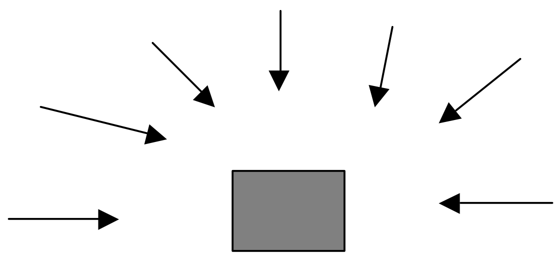
$$I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_r(x, y, z, q, f, l) \\ I_g(x, y, z, q, f, l) \\ I_b(x, y, z, q, f, l) \end{bmatrix}$$



- four types of light sources
  - ambient lighting
  - point light source
  - spotlight
  - distant light

Ambient Light

- uniform lighting (typically in the room)
  - the combination of light reflections from various surfaces
  - 여러 물체들의 난반사에 의한 균일한 밝기의 light
    - 예 : 실내 전체가 균일하게 밝은 경우
  - ambient light 가 가해지면, 모든 물체는 대응되는 밝기를 가짐
- ambient illumination : a scalar value I_a

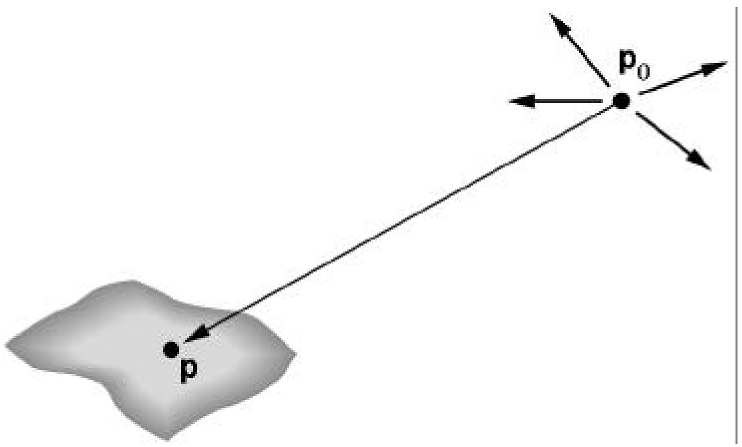


모든 방향에서
동일한 밝기로 들어오는 빛

Point Light Sources

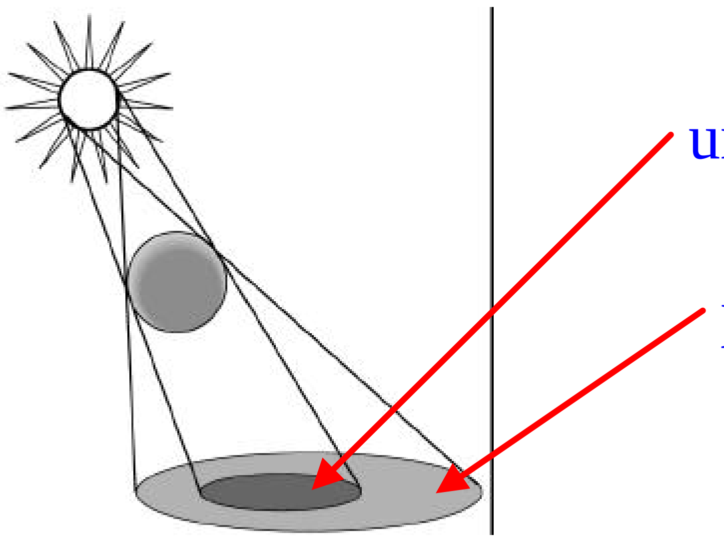
- ideal point light source
  - **점 $\mathbf{p}_0$ 에서 모든 방향으로 균일한 양을 방사**
  - emits light equally in all directions
- light received from the point light source
  - **거리에 반비례**

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



Point Light Sources

- point light source in real world
  - 대표적인 예 : 백열전구
  - ideal point light source 와의 차이점
    - not point, but area light source
    - 그림자에서 차이

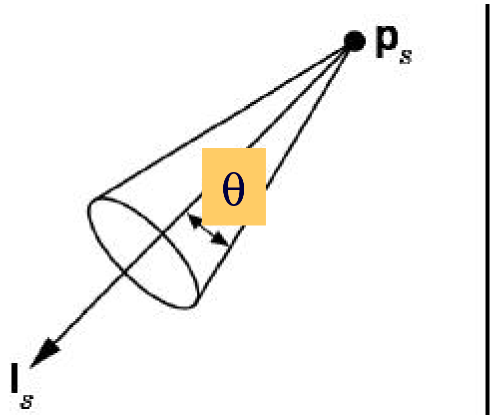


umbra : (진짜) 그림자

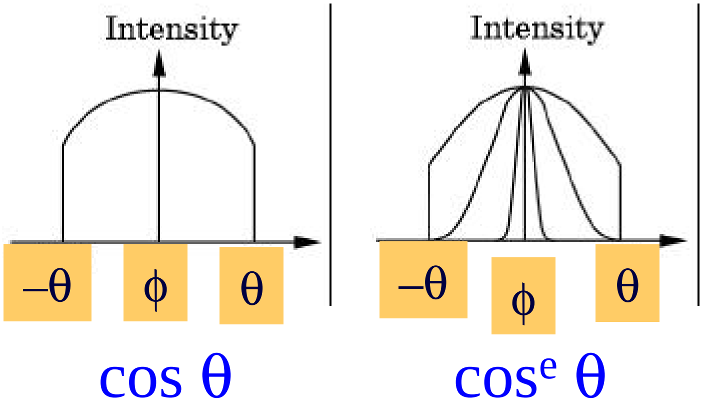
penumbra : 희미한 그림자

Spotlights

- cone 형태로 emit 되는 빛
  - $\mathbf{p}_s$: apex (정점)
  - $\mathbf{I}_s$: light direction
  - θ : angle
    - point light source : θ =180°

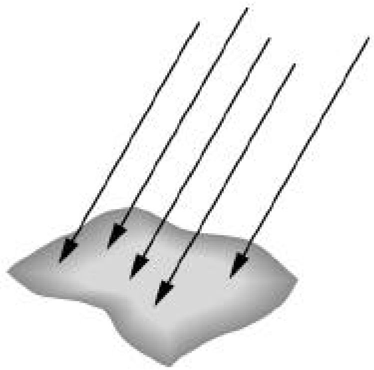


- more realistic case
  - 중심축과의 각도 f
  - 빛의 세기 $\propto \cos^e f$



Distant Light Sources

- = parallel light source
  - 빛이 일정한 방향으로 평행하게 들어오는 경우
  - 예: 태양 광선
  - 장점 : 계산이 훨씬 간단하다
  - in homogeneous coordinate,



$$\mathbf{p}_0 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \longrightarrow \mathbf{p}_0 = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

point light source　　　　parallel light source

6.3 The Phong Reflection Model

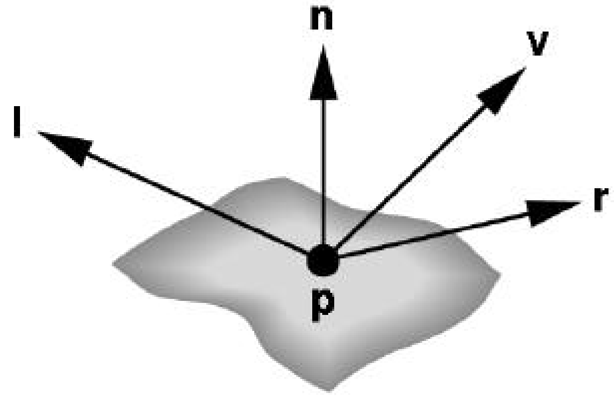
Phong Reflection Model

- Bui-T. Phong, "Illumination for Computer Generated Pictures", *Comm. ACM*, 18(6):311–317, (1975).
- local illumination model
  - 계산을 빠르게 하기 위해서,
    물체 표면에서의 방향 관계만 따진다.
  - 그림자는 계산 불가능 !

normal vector
물체 표면에 수직

light direction
광원 방향

view direction
눈의 방향

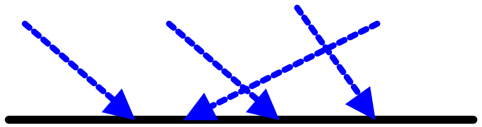


reflection
정반사 방향

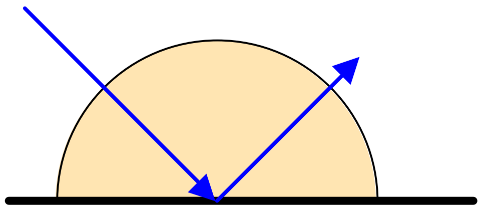
  - ambient term
  - diffuse term
  - specular term

Phong Reflection Model

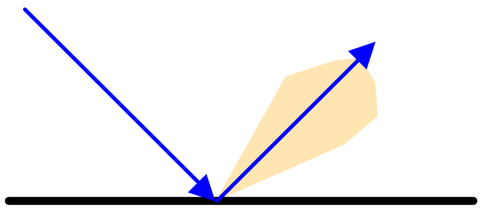
- reflection on the surface
  - ambient illumination
    - indirect reflection을 approximate



  - diffuse illumination
    - 난반사



  - specular illumination
    - 정반사

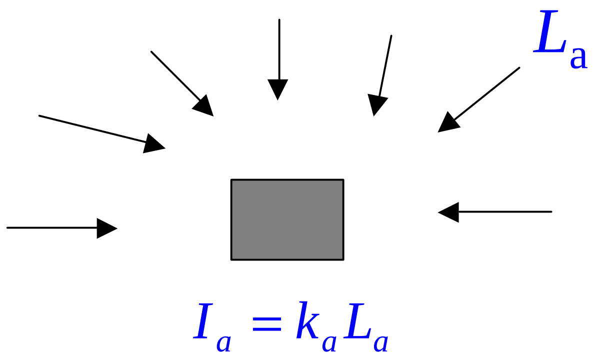


Ambient Reflection

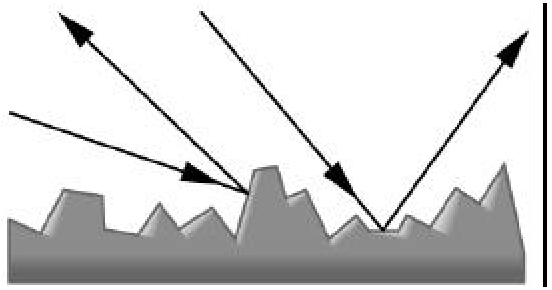
- ambient reflection I_a
  - **왜 필요한가**?
    - a surface that is not exposed directly to a light source still will be visible if nearly objects are illuminated
  - L_a : **현재 광원에서 나오는** ambient light **의 세기**
  - k_a : **물체 표면의** ambient reflection coefficient

$$0 \le k_a \le 1$$

$$I_a = k_a L_a$$

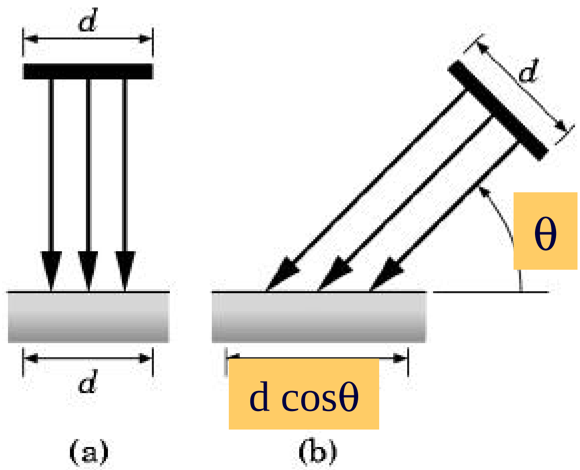


Diffuse Reflection



- 난반사 (亂반사)
  - 물체 표면이 거칠면, 모든 방향으로 빛을 반사
  - Lambertian surface : 빛을 모든 방향으로 고루 반사
- $I_d = k_d L_d \cos\theta = k_d(\mathbf{l} \cdot \mathbf{n})L_d$
  - k_d : diffuse reflection coefficient
  - L_d : intensity of light source
  - **n** : unit normal vector of surface
  - **l** : unit light direction vector
- attenuation form : 거리(d)를 감안
  - a, b, c : constants

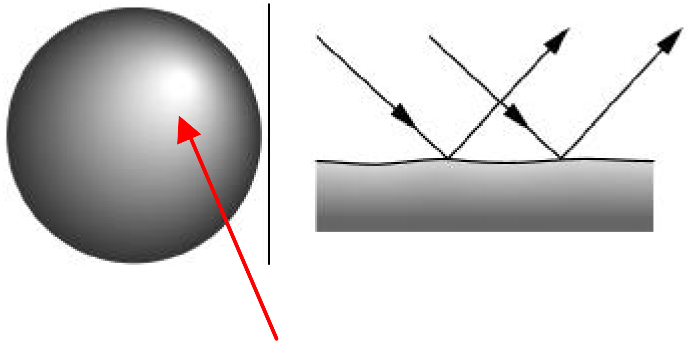
$$I_d = \frac{k_d}{a + bd + cd^2}(\mathbf{l} \cdot \mathbf{n})L_d$$



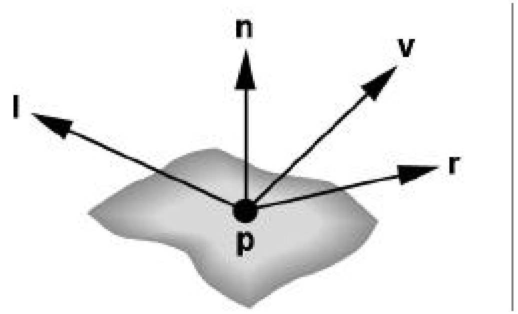
Specular Reflection

- 정반사에 의한 highlight 생성
  - 거울 : perfect specular reflection
- $I_s = k_s \cos^{\alpha}\phi \, L_s = k_s(\mathbf{r} \cdot \mathbf{v})^{\alpha} L_s$
  - k_s : specular reflection coefficient
  - L_s : intensity of light source
  - r : unit reflection vector
  - v : unit view direction vector
  - α : shineness coefficient
- attenuation form : 거리(d)를 감안
  - a, b, c : constants

$$I_s = \frac{k_s}{a + bd + cd^2} (\mathbf{r} \cdot \mathbf{v})^{a} L_s$$



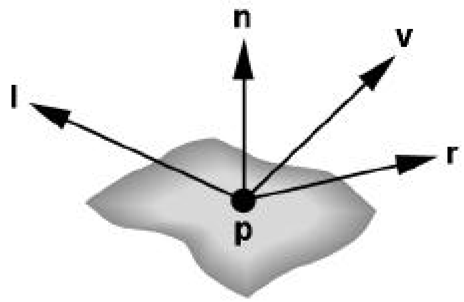
highlight



Phong Reflection Model

- Phong 의 계산식
  - ambient, diffuse, specular term 모두를 합성

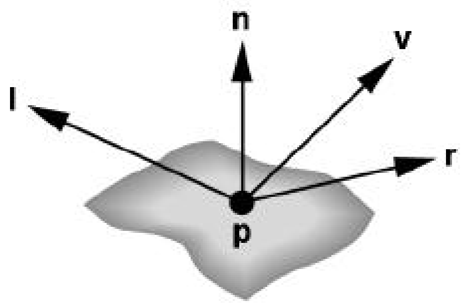
$$
\begin{aligned}
I &= I_a + I_d + I_s \\
&= k_a L_a + \frac{k_d}{a+bd+cd^2}(\mathbf{l}\cdot\mathbf{n})L_d + \frac{k_s}{a+bd+cd^2}(\mathbf{r}\cdot\mathbf{v})^a L_s \\
&= \frac{1}{a+bd+cd^2}\left(k_d L_d(\mathbf{l}\cdot\mathbf{n}) + k_s L_s(\mathbf{r}\cdot\mathbf{v})^a\right) + k_a L_a
\end{aligned}
$$



6.4 Computation of Vectors

Normal Vector

- Phong shading model



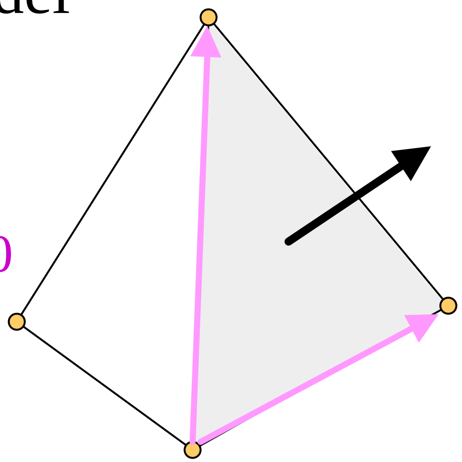
- **n** : surface normal vector
  - polygonal model

$$\mathbf{p}_2 = (x_2, y_2, z_2)$$

$$\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_0) = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

surface equation

$$ax + by + cz + d = 0$$



$$\mathbf{p}_1 = (x_1, y_1, z_1)$$

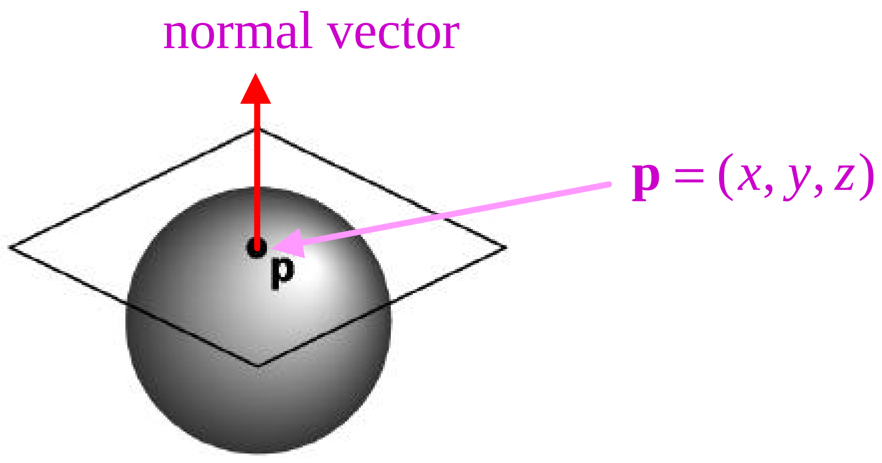
$$\mathbf{p}_0 = (x_0, y_0, z_0)$$

Normal Vector

- implicit equation : 미분으로 계산
  - 예 : unit sphere

$$f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$$

$$\mathbf{n} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \\ \dfrac{\partial f}{\partial z} \end{bmatrix}$$



Normal Vector

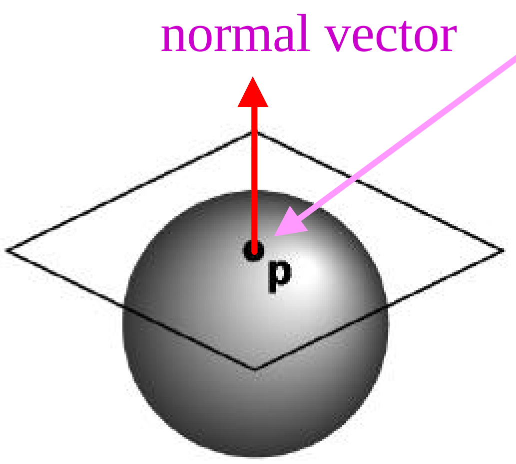
- parametric equation : **미분으로 계산**

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

$$\mathbf{p}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$$

$$\frac{\partial \mathbf{p}}{\partial u} = \begin{bmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \\ \frac{\partial z}{\partial u} \end{bmatrix}, \quad \frac{\partial \mathbf{p}}{\partial v} = \begin{bmatrix} \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial v} \end{bmatrix}$$

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

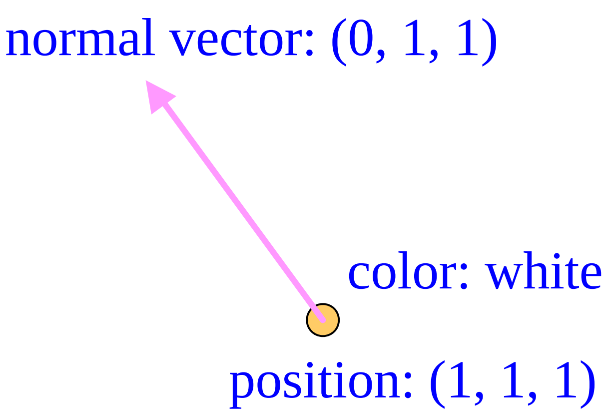


Normal Vector

- in OpenGL
  - void glNormal3f(float x, float y, float z);
  - void glNormal3fv(float v[3]);
    - use the normal vector for a vertex

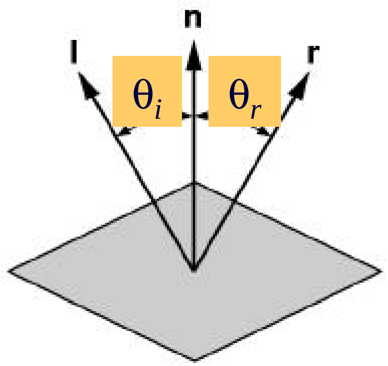
- example

```
glBegin(GL_QUADS);
    glColor3f(1.0, 1.0, 1.0);
    glNormal3f(0.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);
    …
glEnd(GL_QUADS);
```



Reflection Vector

- light direction **l**: 광원 방향
- reflection vector **r**: 정반사 각도
- **l**, **n**, **r** 모두 normalize



$$|\mathbf{l}| = |\mathbf{n}| = |\mathbf{r}| = 1$$

$$q_i = q_r \quad \Rightarrow \quad \cos q_i = \cos q_r$$

$$\cos q_i = \mathbf{l}\cdot\mathbf{n} = \cos q_r = \mathbf{n}\cdot\mathbf{r}$$

$$\text{let } \mathbf{r} = a\,\mathbf{l} + b\,\mathbf{n}$$

$$\mathbf{n}\cdot\mathbf{r} = a\,\mathbf{l}\cdot\mathbf{n} + b = \mathbf{l}\cdot\mathbf{n}$$

$$\mathbf{r}\cdot\mathbf{r} = a^2 + 2ab\,\mathbf{l}\cdot\mathbf{n} + b^2$$

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

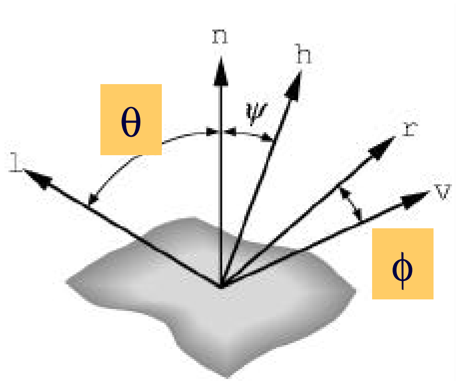
Halfway Vector

- J. Blinn, "Models of Light Reflection For Computer Synthesized Pictures", *SIGGRAPH'77*, 192–198, (1977).
- specualr reflection : $I_s = k_s \cos^{\alpha}\phi\, L_s = k_s(\mathbf{r}\cdot\mathbf{v})^{\alpha} L_s$
- halfway vector **h** : 빠른 계산을 위해서…

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

$$\mathbf{r}\cdot\mathbf{v} = \cos f = \cos 2y$$

$$\mathbf{n}\cdot\mathbf{h} = \cos y < \cos f$$



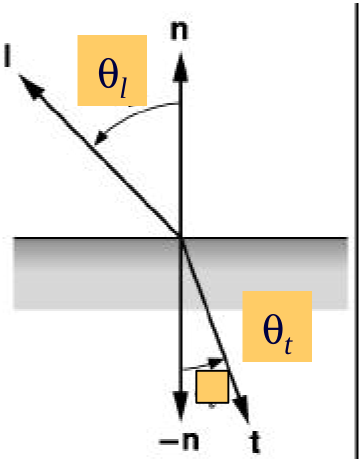
- $(\mathbf{r}\cdot\mathbf{v})^{\alpha} \approx (\mathbf{n}\cdot\mathbf{h})^{\alpha'}$ 이 되도록, α' 값을 조절
  - approximation 이지만,
  - 오차가 별로 크기 않음
  - 계산 속도는 빨라짐

Transmitted Light

- 굴절(refraction) 시에 사용
  - 유리, 물의 표현에 필수적
- Snell's law : 빛의 굴절 법칙

$$\frac{\sin\theta_I}{\sin\theta_t} = \frac{\eta_t}{\eta_I} = \eta$$

  - η_I, η_t : index of refraction (굴절률)



$$\cos\theta_t = \sqrt{1 - \frac{1}{\eta^2}(1 - \cos^2\theta_I)}$$

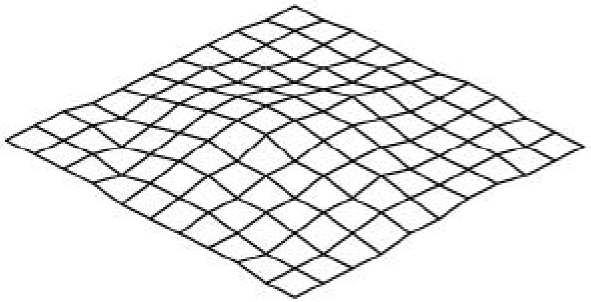
$$\mathbf{t} = \alpha\,\mathbf{n} + \beta\,\mathbf{I}$$

$$\mathbf{t} = -\frac{1}{\eta}\mathbf{I} - \left(\cos\theta_t - \frac{1}{\eta}\cos\theta_I\right)\mathbf{n}$$

6.5 Polygonal Shading

Polygon Shading

- polygonal object 를 shading 하는 방법
  - 대부분의 물체가 polygonal object 임

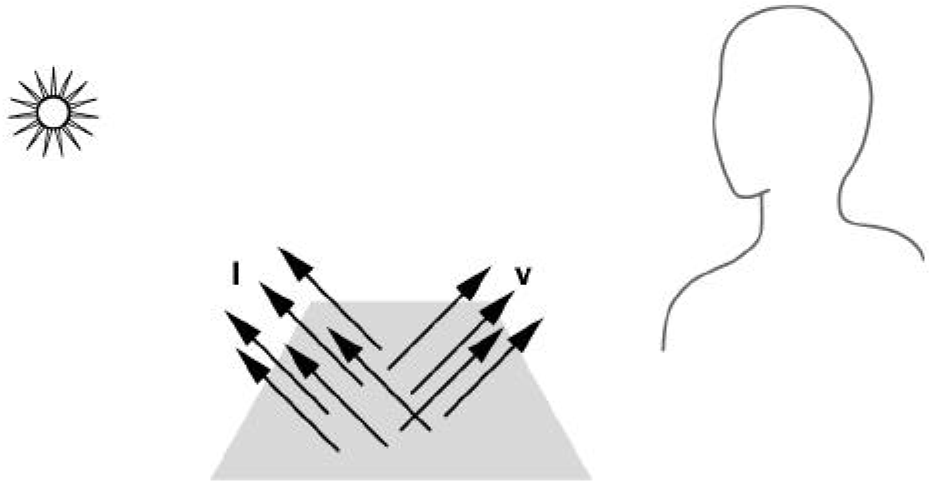


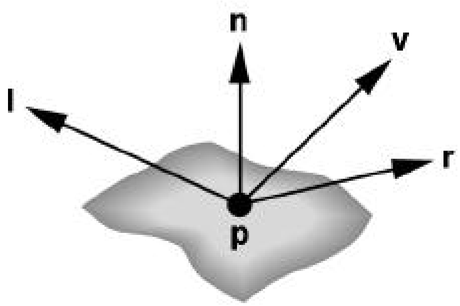
부드러운 물체도
polygonal mesh 로 표현

- 계산량의 증가에 따라,
  - flat shading
  - Gouraud shading
  - Phong shading

Flat Shading

- = constant shading
  - 가정 1: flat polygon   **n** = constant
  - 가정 2: distant viewer **v** = constant
  - 가정 3: distant light   **l** = constant, **r** = constant
  - 결론적으로, 같은 polygon 내에서는 같은 shading 결과 사용 가능





Flat Shading

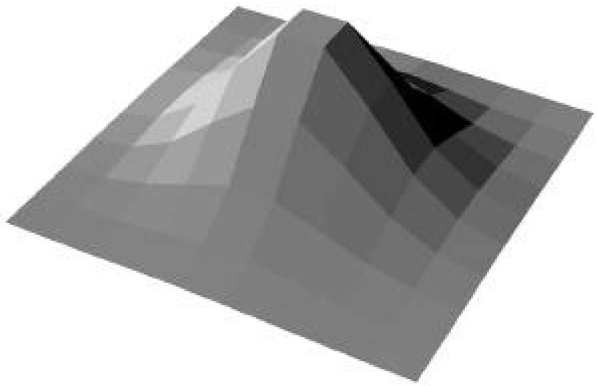
- shading 값 계산 시는 Phong model 사용

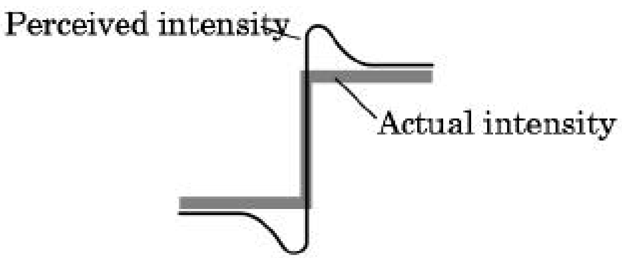
$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a$$

- in OpenGL

  glShadeModel(GL_FLAT);

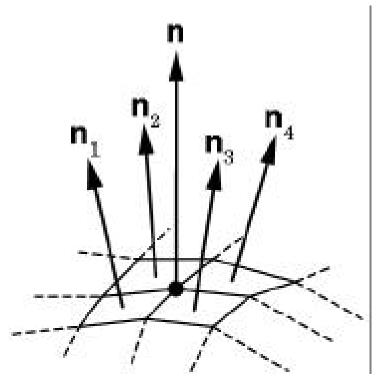
- 문제점: Mach band effect
  - polygon 경계에서 계단 효과

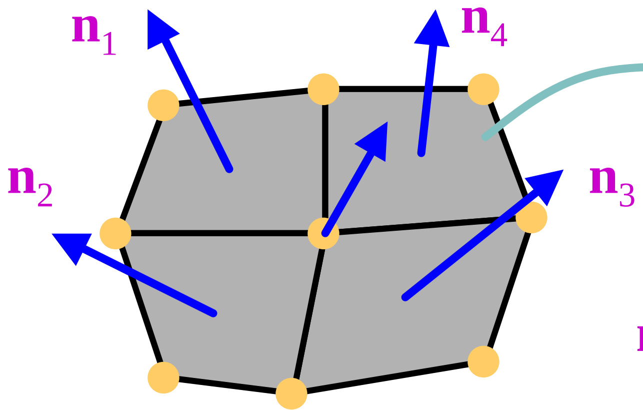


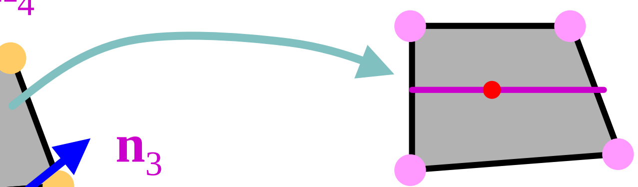


Gouraud Shading

- = interpolative shading
- Mach band effect 를 줄이는 방법?
  - vertex normal은 인접한 face들의 normal들을 평균해서 계산
  - face 내는 vertex 에서의 shading 결과를 bi-linear interpolation







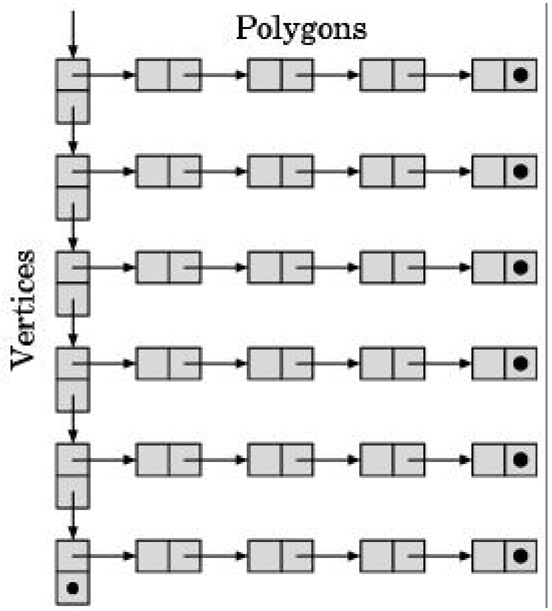
bi-linear interpolation

$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$

Gouraud Shading

- in OpenGL

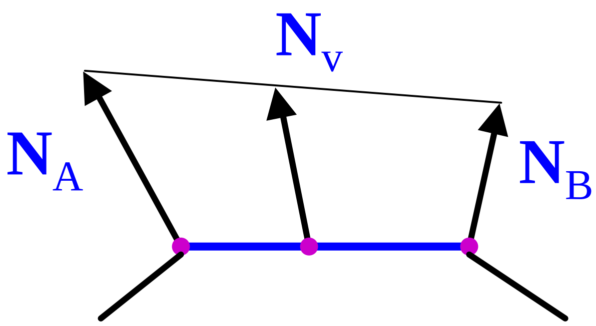
  glShadeModel(GL_SMOOTH);
- 문제점
  - vertex normal 을 계산하기 위한 data structure 필요
  - 주로, linked list 구조



Phong Shading

- Gouraud shading 의 문제점
  - vertex 에서의 shading 결과를 interpolate
  - highlight 계산 불가능
- Phong shading : normal vector를 interpolate
  - shading 결과는 각 pixel에서 다시 계산
    - 계산 속도는 당연히 느려짐

bi-linear interpolation

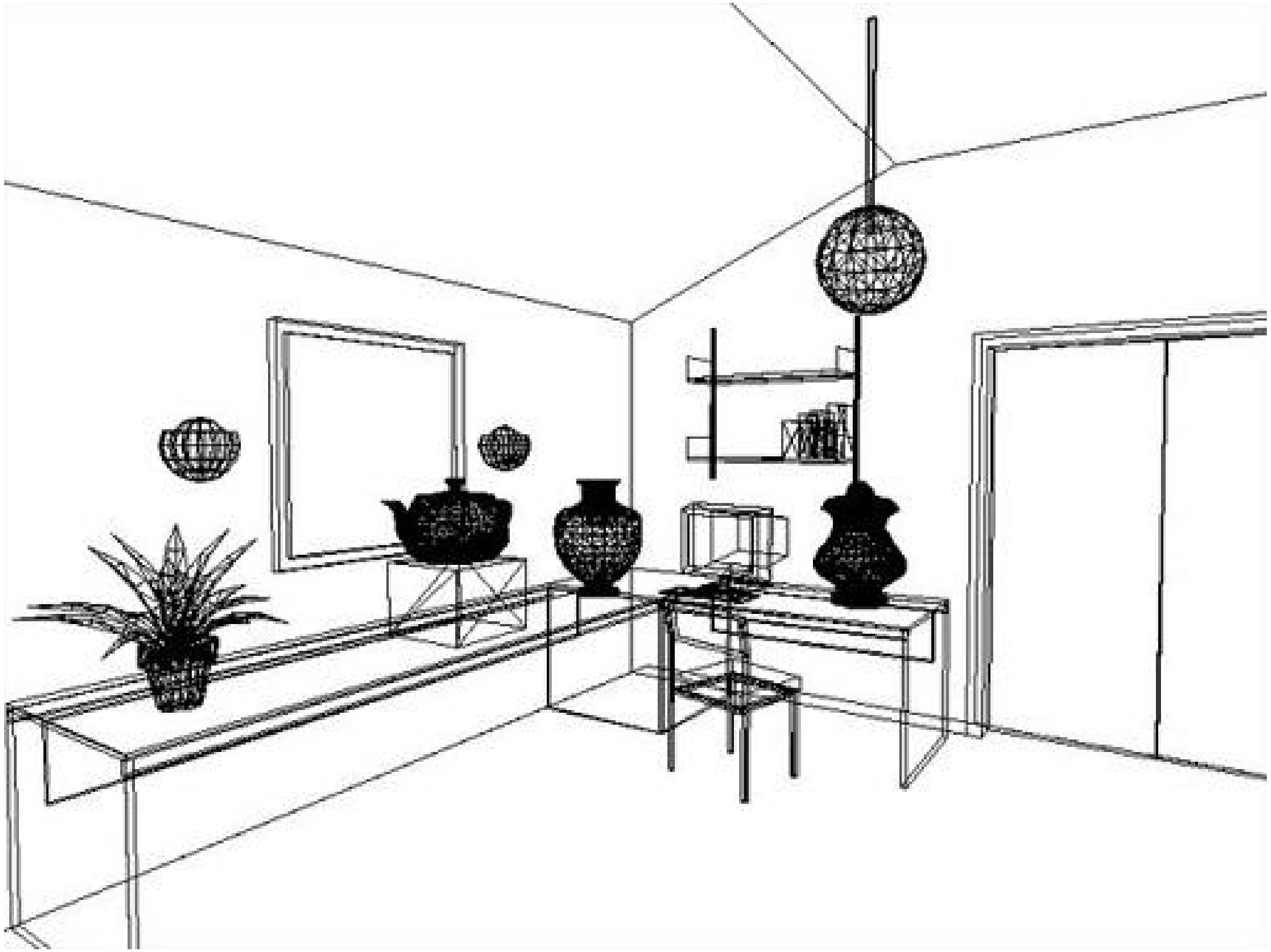


normal vector만 있으면,
Phong model 로
pixel 값 계산 가능

Rendering Options

- 일반적인 graphics package 에서는 선택 사항
  - quality vs. speed 의 trade-off
- wireframe edge 만 표시
- flat-shading face 전체를 같은 색으로
- Gouraud shading
- Phong shading
- mixed shading
  - non-specular object : Gouraud shading
  - specular object : Phong shading

Wire-frame



Ambient Illumination Only

Flat-Shading

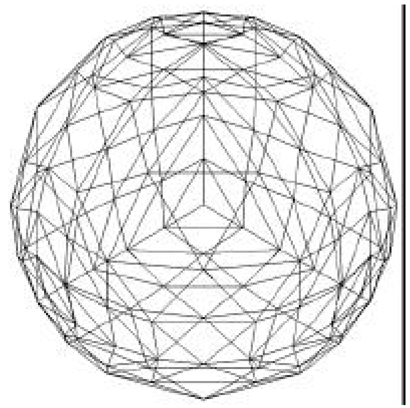
Gouraud Shading

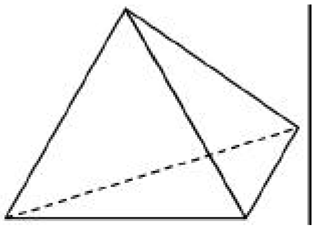
Phong Shading

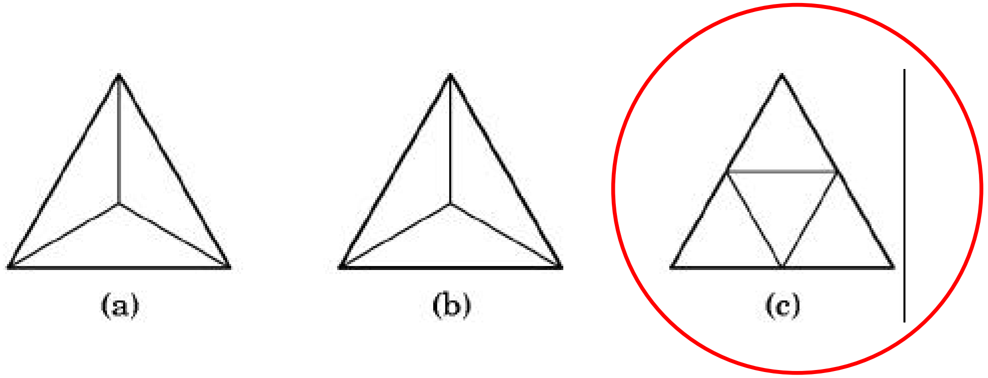
6.6 Approximation of a Sphere

Unit Sphere

- implicit equation ⇒ polygonal approximation
  $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$
- recursion 으로 구현
  - 시작: tetrahedron
  - 각 face를 subdivision



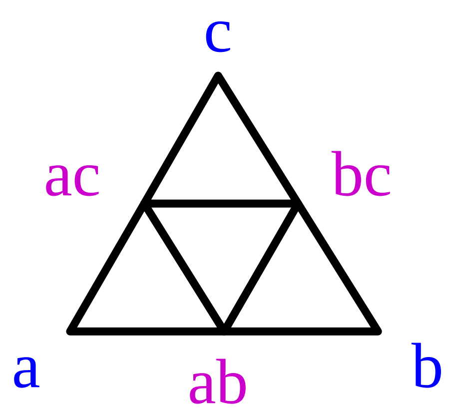




Unit Sphere

- pseudo code

```
void triangle(vertex a, vertex b, vertex c, int level) {
    if (level >= limit) {
        draw_triangle(a, b, c);
        return;
    }
    vertex ab = normalize( (a + b) / 2 );
    vertex bc = normalize( (b + c) / 2 );
    vertex ac = normalize( (a + c) / 2 );
    triangle(a, ab, ac);
    triangle(ab, b, bc);
    triangle(c, ac, bc);
    triangle(ab, bc, ac);
}
```



6.7 Light Sources in OpenGL

Light Sources in OpenGL

- Phong shading model

$$
\begin{aligned}
I &= I_a + I_d + I_s \\
&= k_a L_a + \frac{k_d}{a + bd + cd^2}(\mathbf{l}\cdot\mathbf{n})L_d + \frac{k_s}{a + bd + cd^2}(\mathbf{r}\cdot\mathbf{v})^a L_s \\
&= \frac{1}{a + bd + cd^2}\left(k_d L_d(\mathbf{l}\cdot\mathbf{n}) + k_s L_s(\mathbf{r}\cdot\mathbf{v})^a\right) + k_a L_a
\end{aligned}
$$

- at least 8 light sources
  - GL_LIGHT0, …, GL_LIGHT7

```
glEnable(GL_LIGHTING);        // light source 사용
glEnable(GL_LIGHT0);          // light source #0 on
```

Light Sources in OpenGL

- light source 설정 함수들

  void glLightf(GLenum light, GLenum name, GLfloat value);

  void glLightfv(GLenum light, GLenum name, GLfloat* values);

- light source position (or direction)

  glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);

  – point light source, spotlight : position

    GLfloat light0_pos[ ] = { 1.0, 2.0, 3.0, 1.0 };

  – distant light source : direction vector

    GLfloat light0_pos[ ] = { 1.0, 2.0, 3.0, 0.0 };

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a$$

Light Sources in OpenGL

- the amount of ambient, diffuse, specular light

```
GLfloat ambient_0[ ] = {1.0, 0.0, 0.0, 1.0};   // RGBA rep.
glLightfv(GL_LIGHT0, GL_AMBIENT, ambient_0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse_0);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular_0);
```

- global ambient light (if necessary)

```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
```

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a$$

Light Sources in OpenGL

- distance attenuation model

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c);
```

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a$$

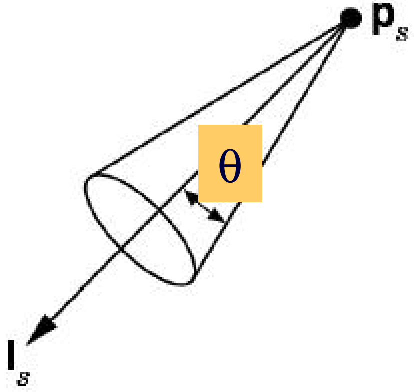
Light Sources in OpenGL

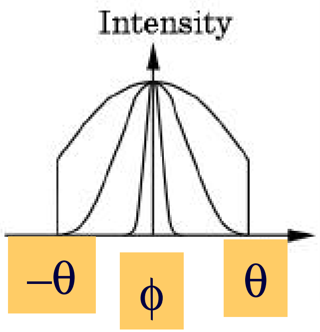
- spot light

  glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 30.0F);

  glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, e);

  glLightfv(GL_LIGHT0, GL_SPOT_DIRECTOIN, vector);





$\cos^e \theta$

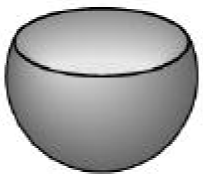
Light Sources in OpenGL

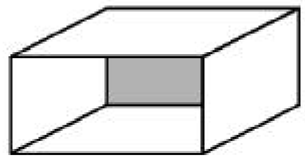
- viewer location
  - default: infinite position (방향은 COP 방향)
  - COP 에서 보는 것으로 설정 가능
    - 큰 차이는 없지만, shading 계산시의 v 값 변화

glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a$$

- two-side shading
  - back face도 shading 시킬 때





glLightModeli(GL_LIGHT_MODEL_TWO_SIDED, GL_TRUE);

6.8 Specification of Materials in OpenGL

Materials in OpenGL

- material functions

  void glMaterialf(GLenum face, GLenum name, GLfloat value);

  – **이후의 모든** face **가 정해진 값을 사용**

  – face = GL_FRONT, GL_BACK, GL_FRONT_AND_BACK

```
GLfloat ambient[ ] = {0.2, 0.2, 0.2, 1.0}
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialfv(GL_BACK,   GL_SPECULAR, specular);
```

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a$$

Materials in OpenGL

- shininess

  glMaterialf(GL_BACK, GL_SHINESS, 100.0);

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a$$

- self-emission
  - surface 자체가 빛을 내는 경우

  GLfloat emission[ ] = {0.0, 0.3, 0.3, 1.0};

  glMaterialf(GL_FRONT_AND_BACK, GL_EMISSION, emission);

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^a\right) + k_a L_a + emission$$

Materials in OpenGL

- color tracking
  - set the materials parameters by tracking the colors set by glColor*( )

```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
```

```
glColor4fv(color);
```

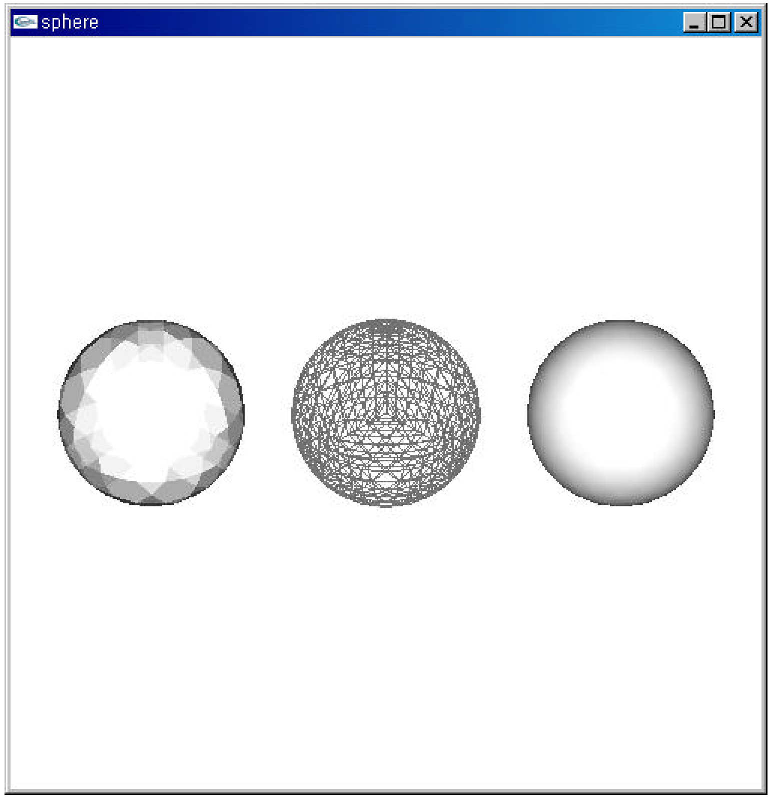
자동 설정

```
glMaterialfv(GL_FRONT, GL_AMBIENT, color);
glMaterialfv(GL_FRONT, GL_DIFFUSE, color);
```

6.9 Shading of the Sphere Model

Unit Sphere Shading

- 2가지 shading model 선택 가능
  – flat shading
    glShadeModel(GL_FLAT);
  – Gouraud shading
    glShadeModel(GL_SMOOTH);

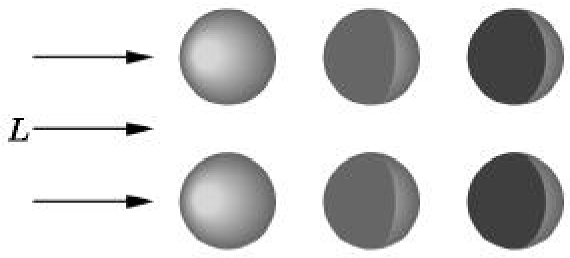


- 자세한 코드는 교재 참고
  – sphere.c

6.10 Global Rendering

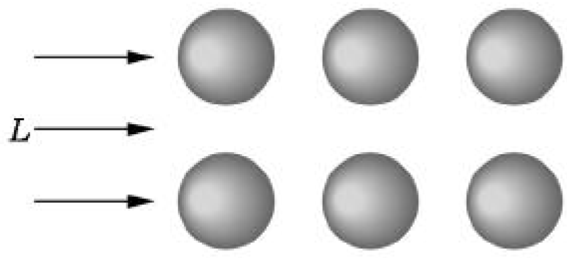
Local and Global Illumination

- local illumination : Phong shading model
  - 그림자, 반사광 등의 처리가 힘들다



(a)

global illumination



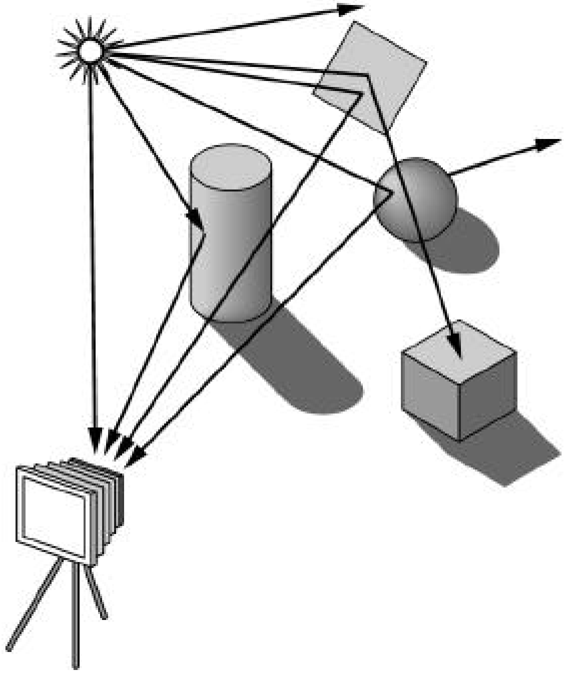
(b)

local illumination

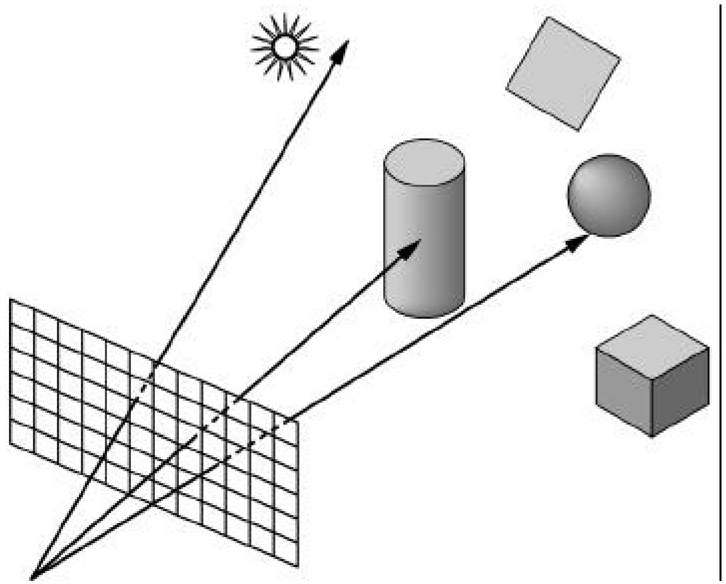
- global illumination
  - Ray Tracing : specular, 굴절, 그림자 처리 능력
  - Radiosity : diffuse 처리가 완벽

Ray Tracing

- forward ray tracing : **실세계**
  - **광원** ⇒ **물체** ⇒ **카메라**

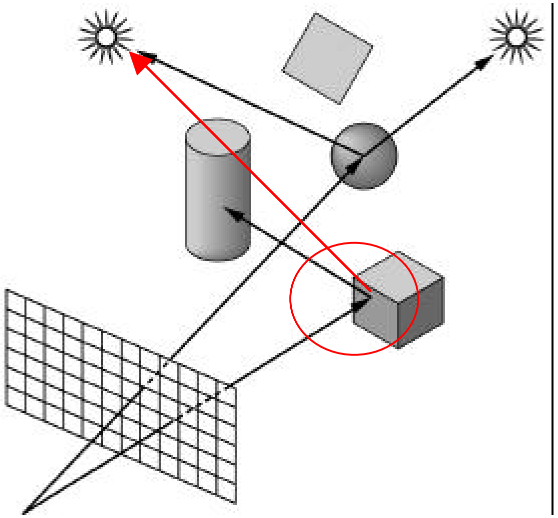


- backward ray tracing
  - **계산량을 줄이려면, 역추적**
  - **카메라** ⇒ **픽셀** ⇒ **물체** ⇒ **광원**

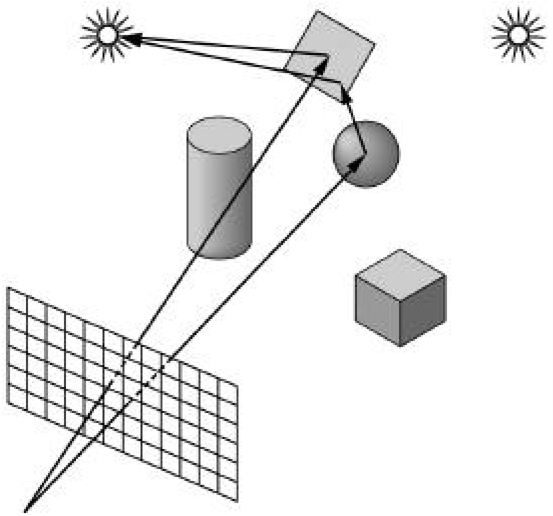


Ray Tracing

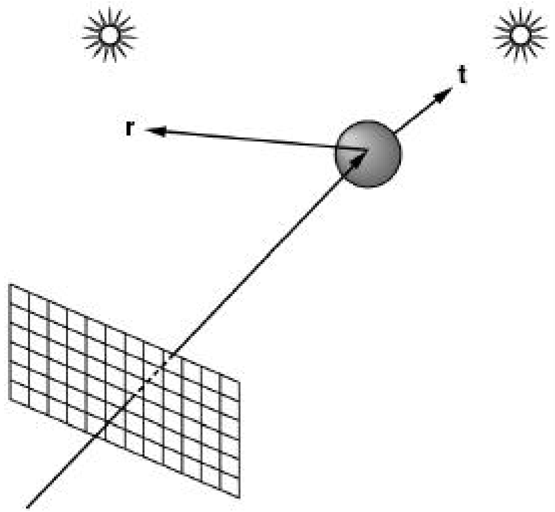
- A. Glassner, *An Introduction to Ray Tracing*, Academic Press, (1989).
- 실제 ray를 trace 하는 효과가 있으므로, 다양한 기능



shadow rays
그림자 처리 가능



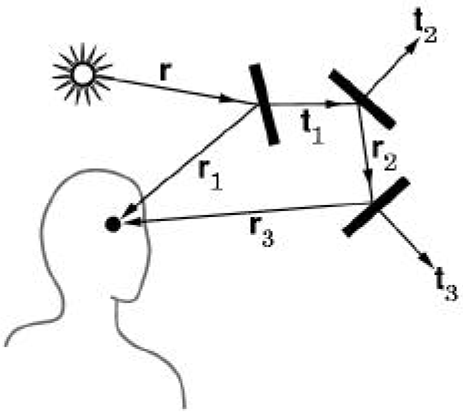
mirror effect
거울 처리 가능

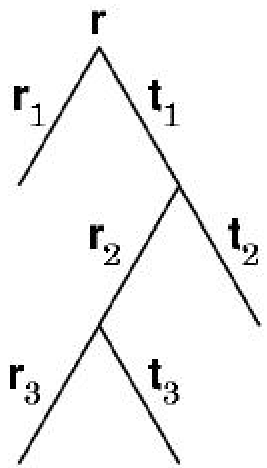


refraction & reflection
투명한 물체 처리 가능

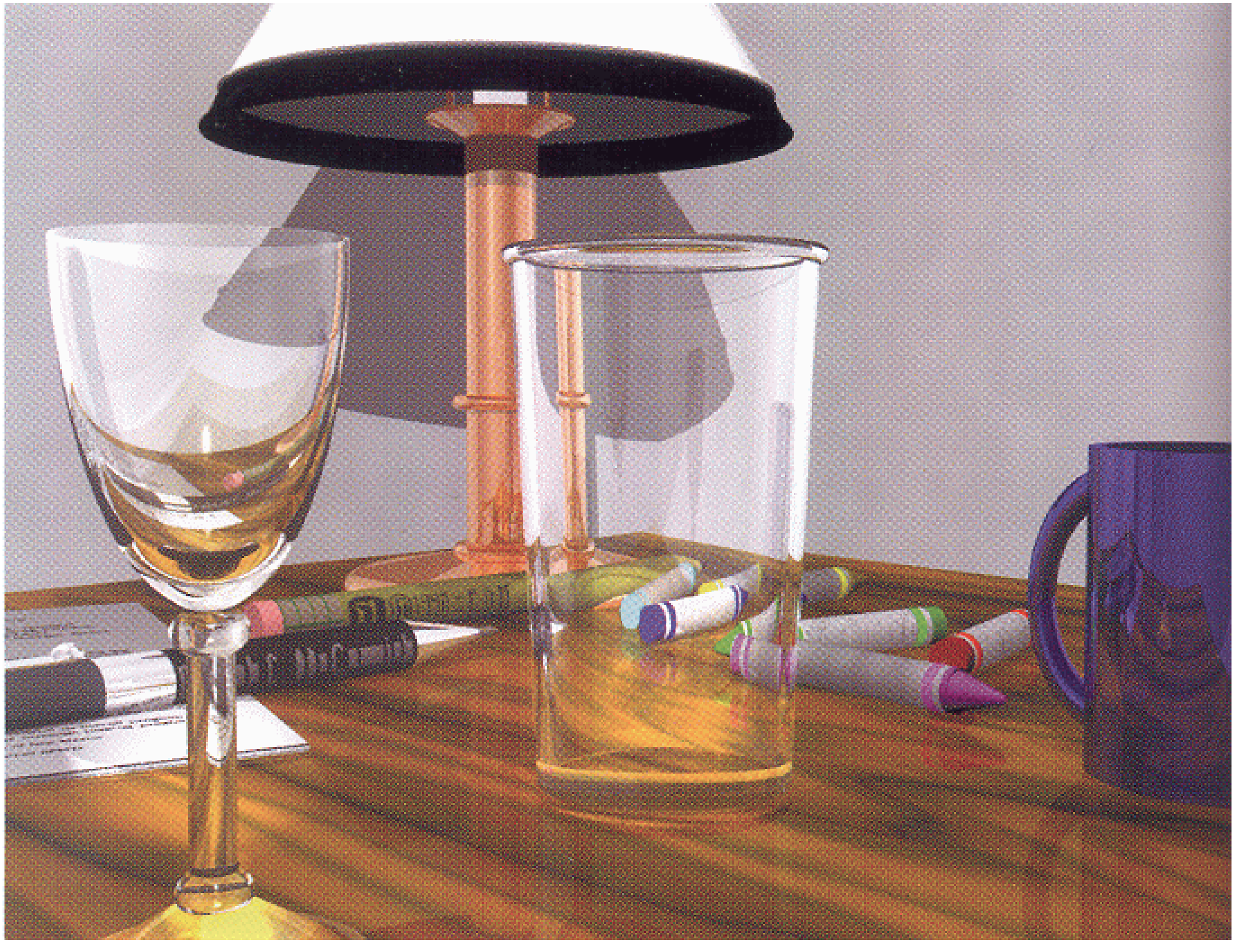
Ray Tracing

- screen 상의 pixel 을 지나가는 ray 를 하나씩 발사
  - construct a binary ray-tracing tree
    - left branch : reflected ray path
    - right branch : transmitted ray path
    - limit depth 까지 계속 추적
  - tree를 역추적하면서, 최종 pixel 값 계산



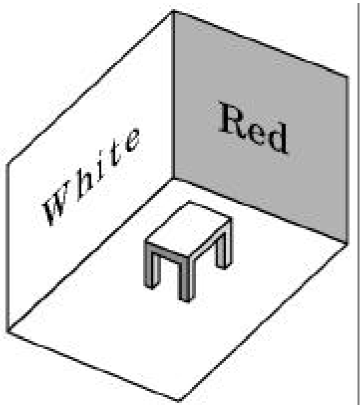


Ray Tracing 결과



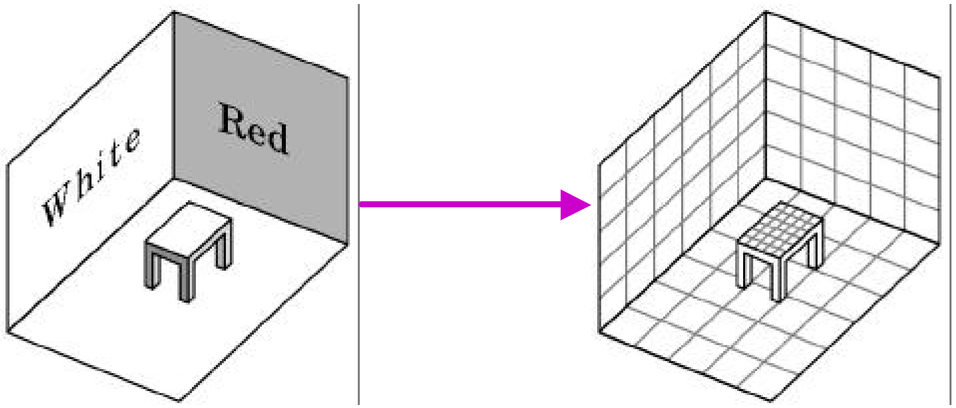
Radiosity

- C. Goral, et al., "Modelling the Interaction of Light Between Diffuse Surfaces", *SIGGRAPH'84*, 212–222, (1984).
- Basic idea
  - thermal-engineering models for the emission and reflection of radiation
  - conservation of light energy in a closed environment
    - **주로 실내 환경에서만 적용**
  - energy equilibrium



Radiosity

- real diffuse effect ?
  - 모든 surface 들 간의 light interaction으로 modeling
  - 모든 surface 들 끼리 서로 반사광을 주고 받음
- patch : 계산의 기본 단위
  - 모든 surface들을 작은 patch 로 분할
  - 각 patch 마다 diffuse term을 따로 계산
  - form factor : pacth 1개에서 다른 patch 로 가는 energy

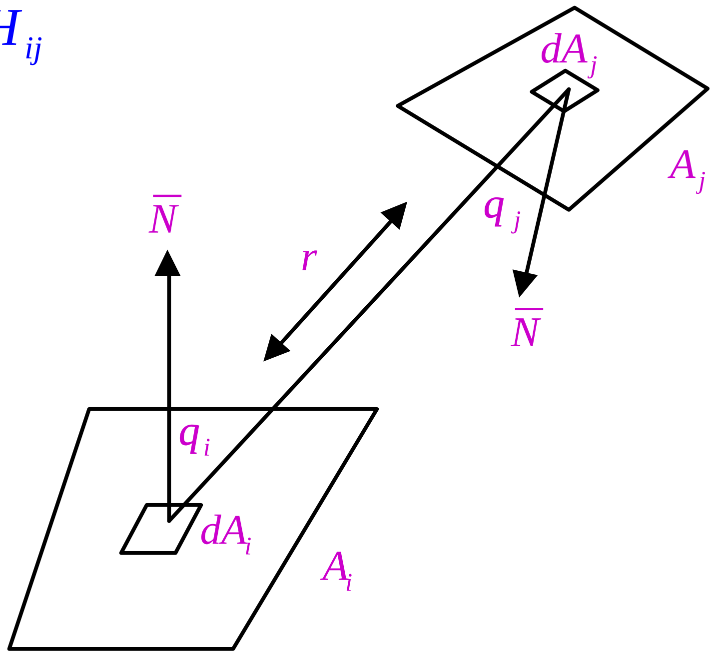


Radiosity

- form factors

$$F_{ij} = 1/A_i \int_{A_i}\int_{A_j} (\cos q_i \cos q_j /(pr^2)) H_{ij}\, dA_j\, dA_i$$

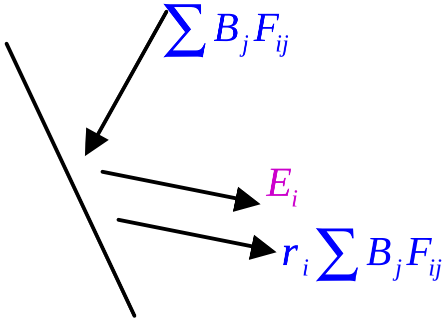
$$\approx \int_{A_j} \frac{\cos q_i \cos q}{pr^2}\, dA_j H_{ij}$$



Radiosity

- energy equilibrium equation

$$B_i A_i = E_i A_i + r_i \sum_{j=1}^{n} B_j F_{ji} A_j$$



$$F_{ij} A_i = F_{ji} A_j$$

$$\therefore \quad B_i = E_i + r_i \sum_{j} B_j F_{ij}$$

  - radiosity B_j : the rate of energy leaving the surface j
  - emission E_j : the rate of energy emitted from surface j
  - reflectivity r_j : the fraction of the light incident on a surface j which is reflected back into the environment
  - form-fraction F_{ij} : the fraction of energy leaving the surface i and landing on surface j

Radiosity 결과

Global Illumination

- ray tracing : specular effect 에 최적
- radiosity : diffuse effect 에 최적
- hybrid approach : 2개 모두 적용

- 문제점 ?
  – 수행 시간!
  – video card level에서 구현 불가능
  – 보통 몇 시간 정도의 처리 시간 필요
  – 구현 자체도 상당히 까다로움