Computer Graphics

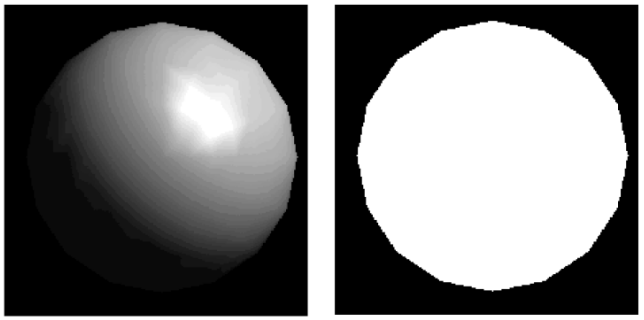
Prof. Jibum Kim

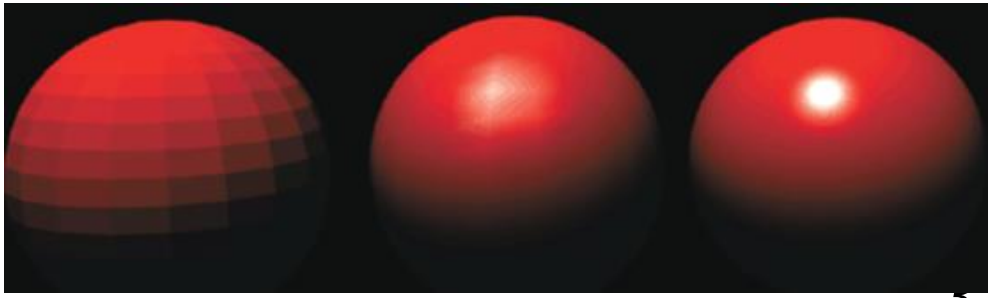
Department of Computer Science & Engineering

Incheon National University

lighting and shading

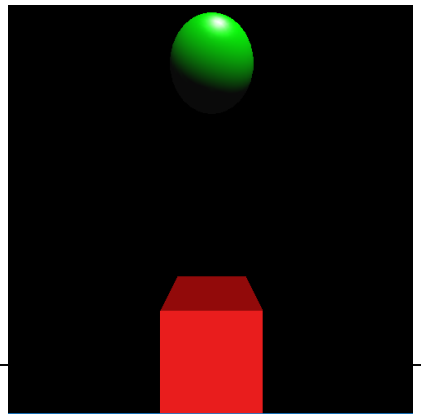
- 조명 (lighting): 물체 정점의 색을 부여하는 작업
- 광원 (light source)과 물체 (material) 특성을 감안하여 정점 (vertex)에서의 빛 세기를 계산하는 작업
- 음영 (shading): 이렇게 부여된 정점 색을 기준으로 해당 물체 면의 내부에 색을 칠하는 작업
- (left) 조명과 음영 처리 후 3차원 구로 보임
- (center) 조명과 음영 처리가 없으면 2차원의 구로 보임
- (right) 다양한 shading 방법들

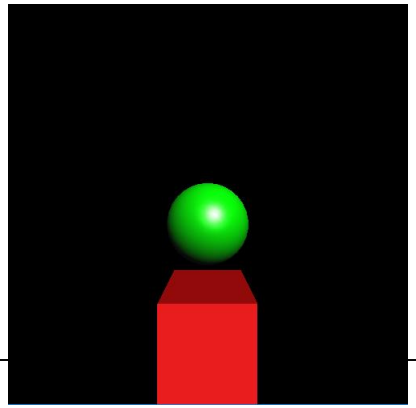




- **Phong reflection model과 shading을 이용한 애니메이션 예**

- **https://www.dropbox.com/s/qsmuxv1kj7rcsjk/ball_bounce_light_animation.txt?dl=0**





- 교재 소스코드 예제
- Chapter11/SphereinBox1/
- Press the up-down arrow

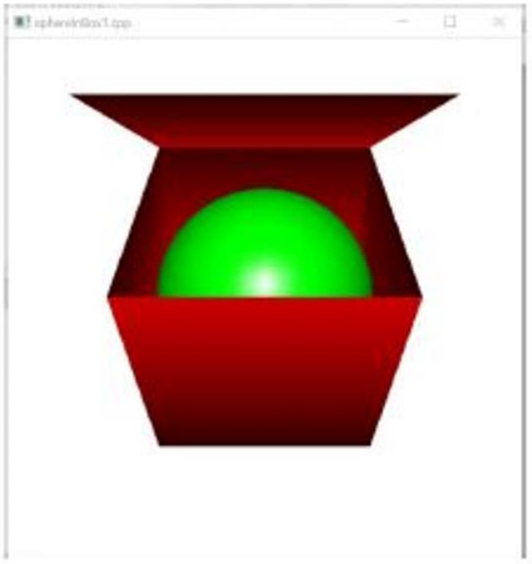


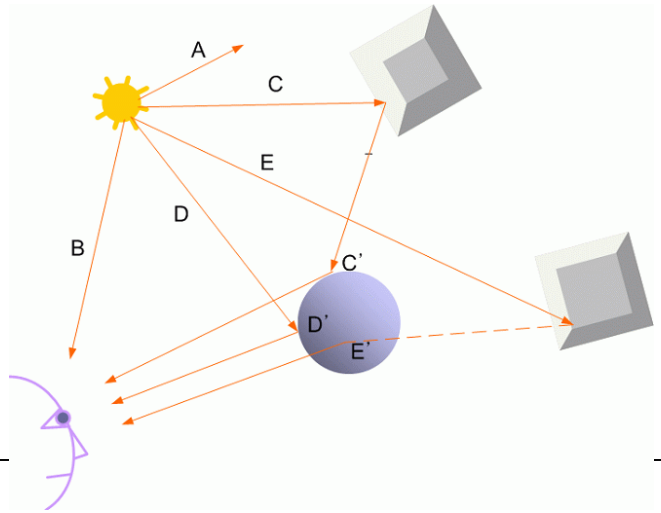
Figure 11.19:
Screenshot of
sphereInBox1.cpp.

- **Local illumination model**

- **Light source (광원): light-emitting surface**

- **A model of interaction between light sources (광원) and objects is called a lighting model (reflection model, illumination model)**

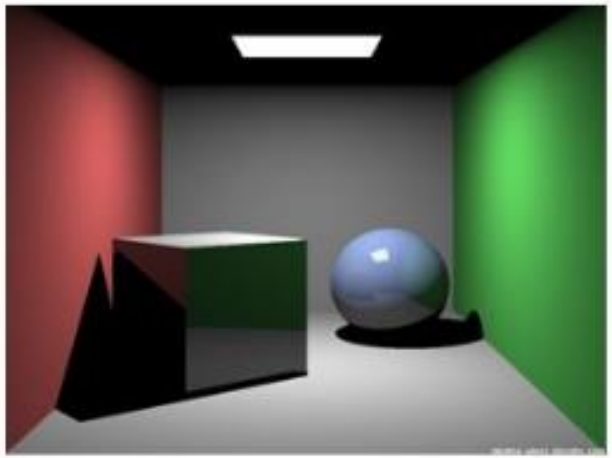
- **Light source와 반사, viewer**
- **광원에서 A방향으로 진행하는 빛 (다른 물체에 반사되지 않는 경우) : 고려할 필요 없음**
- **광원에서 B방향으로 진행하는 빛: 광원 자체를 볼 수 있음**
- **광원에서 D방향으로 진행하는 빛: 물체면 D’에 부딪쳐서 우리눈에 옴**
- **다른 물체에 반사되어 입사되는 빛도 있는 경우: C’, E’**

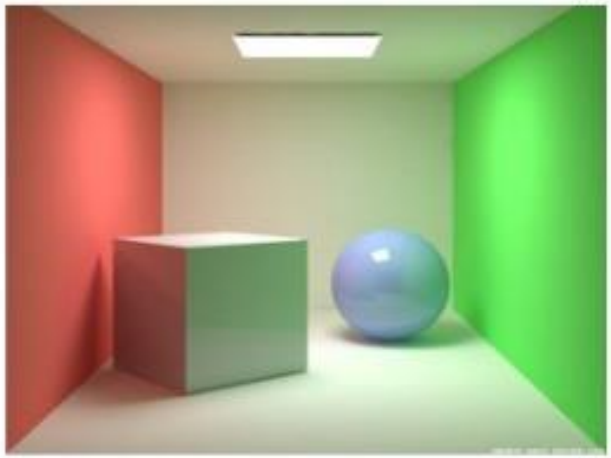


- **Global illumination model (전역 조명 모델)**
- 다른 물체에서 반사되어 입사되는 빛까지 고려한다
- 물체 상호간의 반사까지 고려한 모델
- Ray tracing

- **Local (direct) illumination model (지역 조명 모델)**
- 다른 물체에서 반사된 빛은 일체 고려하지 않는다
- 광원으로부터 직접 물체에 부딪쳐 눈에 들어오는 빛 만 고려 한다
- OpenGL에서 주로 사용 (처리 속도가 빠르다)

- **Left (local illumination), right (global illumination)**





- **Light sources**
- **https://docs.unity3d.com/kr/current/Manual/Lighting.html**

- **1. Color sources**
- **We model light sources as having three components – red, green, blue (R, G, B)**
- **We descbie a source through a three-component intensity (강도) function, each of whose component is the intensity of the independent red, green, and blue compoenents**
- $I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$

- **2. Ambient light (주변광)**
- **In many rooms, such as classrooms or kitchens, the light have been designed and positioned to provide uniform illumination throughout the room**
- **This uniform lighting is called ambient light, I_a**
- **It is identical at every point in the scene**
- $I_a = \begin{bmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{bmatrix}$
- **We will use the scalar I_a to denote any one of the red, green, or blue componenets of I_a**

Unity manual

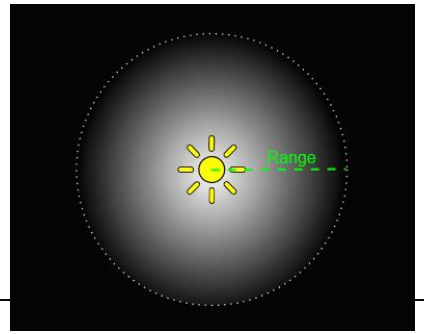
주변광

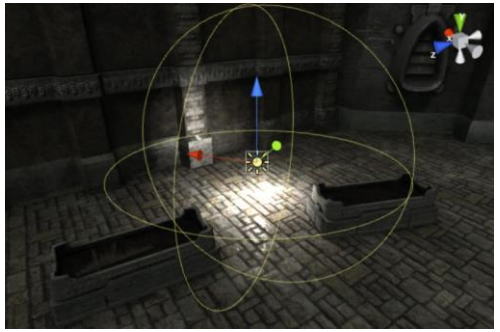
주변광(Ambient light)은 씬 전체에 있고 특정 광원 오브젝트에서 나오지 않는 광원입니다. 이 광원은 씬의 전체적인 외양과 밝기에 중요하게 기여할 수 있습니다.

주변광은 선택된 아트 스타일에 따라 여러 경우에 유용할 수 있습니다. 예로는 어두운 그림자가 바람직하지 않을 수 있는 밝은 카툰 스타일 렌더링이나 조명을 텍스처에 수동으로 칠하는 경우를 들 수 있습니다. 주변광은 개별 광원을 조정하지 않고 씬의 전체적인 밝기를 높여야 하는 경우에도 유용할 수 있습니다.

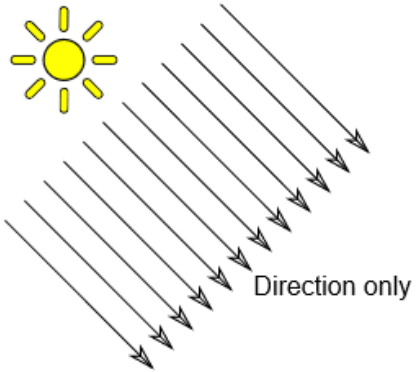
주변광 설정은 조명 창을 참고하십시오.

- **3. Point light source (점광원)**
- **An ideal point light emits light equally in all directions**
- **A Point Light is located at a point in space and sends light out in all directions equally.**
- **The intensity diminishes with distance from the light, reaching zero at a specified range.**
- **In homogeneous coordinates, a point light source at p is represented as a 4-D column matrix, $P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$**

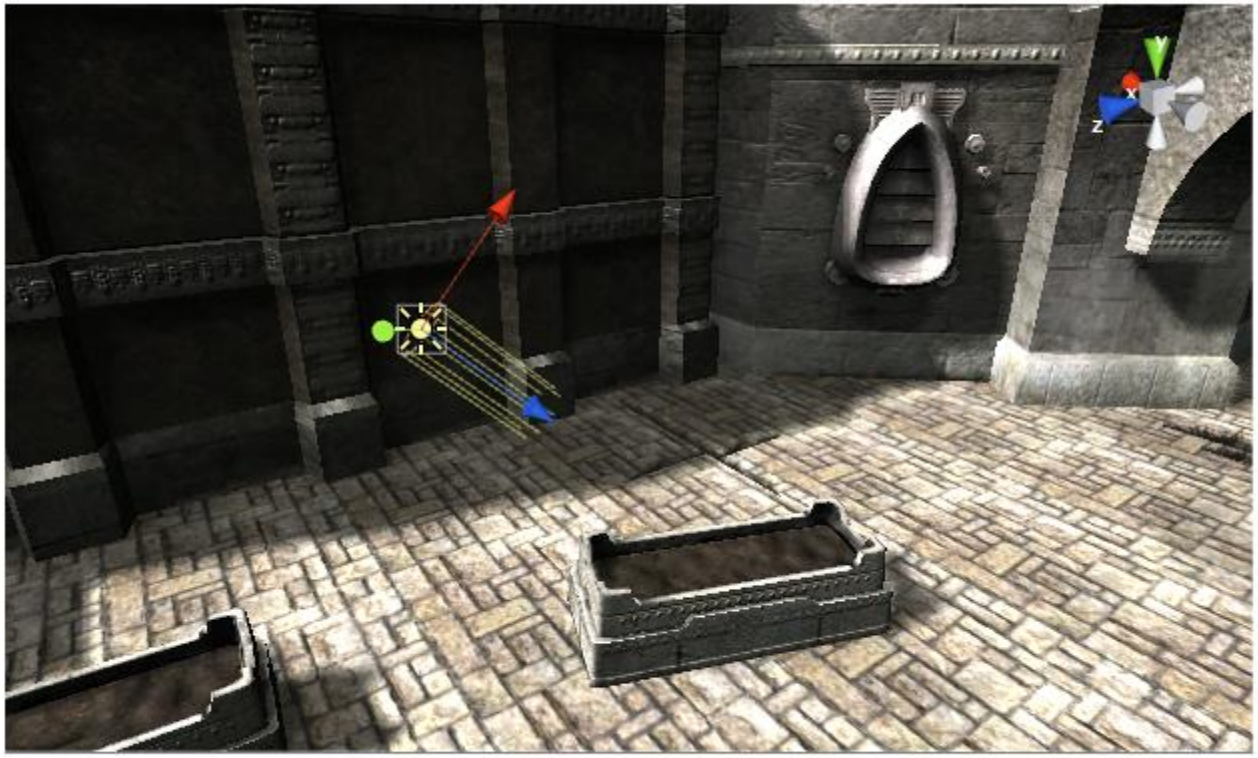




- **4. Distant light sources (directional light)**
- **A Directional Light does not have any identifiable source position and so the light object can generally be placed anywhere in the scene**
- **광원이 무한대 거리에 위치 (e.g., 태양) All objects in the scene are illuminated as if the light is always from the same direction.**
- **The distance of the light from the target object is not defined and so the light does not diminish.**
- **The distance light source is described by a direction vector whose representation in homogeneous coordinates is the matrix,** $P = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$

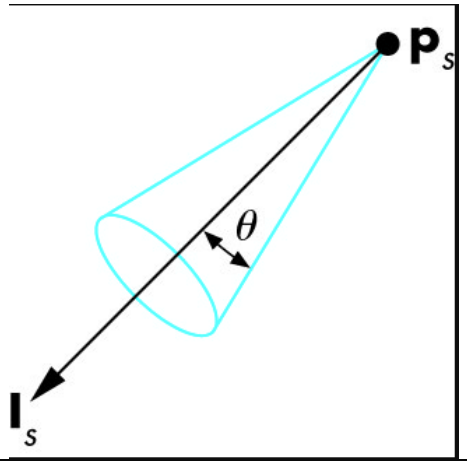


Directional light

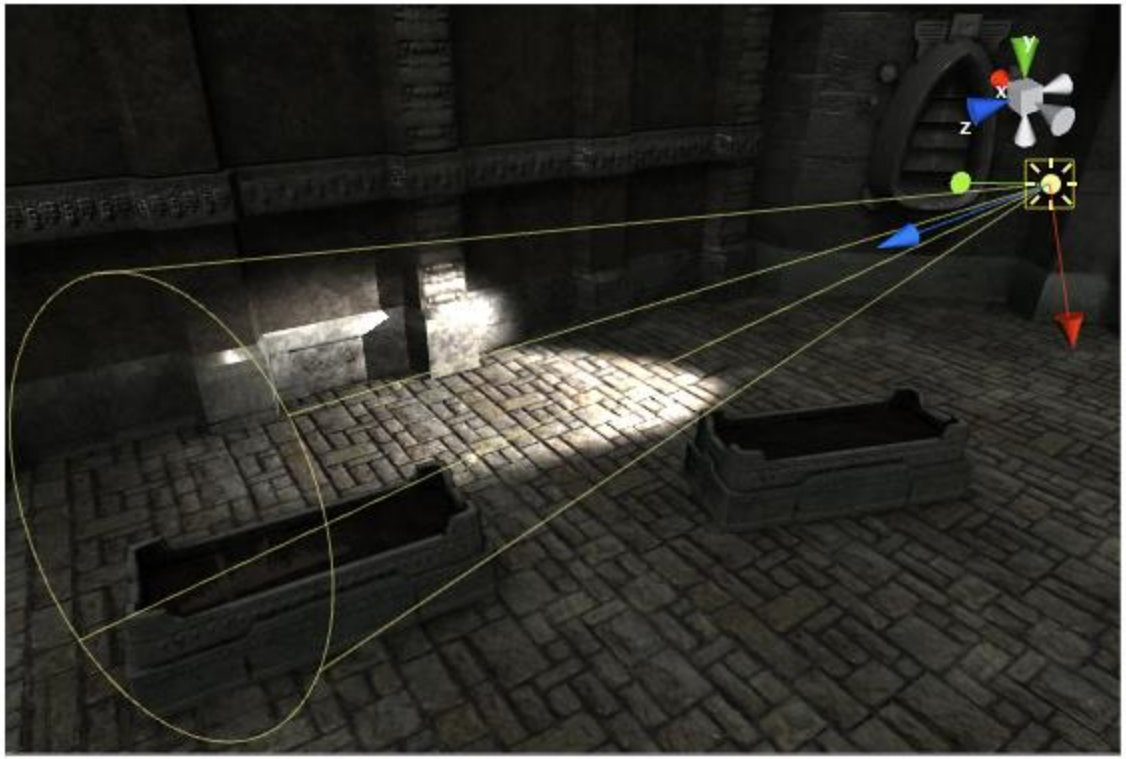


Effect of a Directional Light in the scene

- **5. Spotlights**
- **Spotlights are characterized by a narrow range of angles through which light is emitted**
- **We can construct a simple spotlight from a point source by limiting the angles at which light from the source can be seen**
- **We can use a cone whose apex is at p_s, which points in the direction l_s, and whose width θ is determined by an angle**

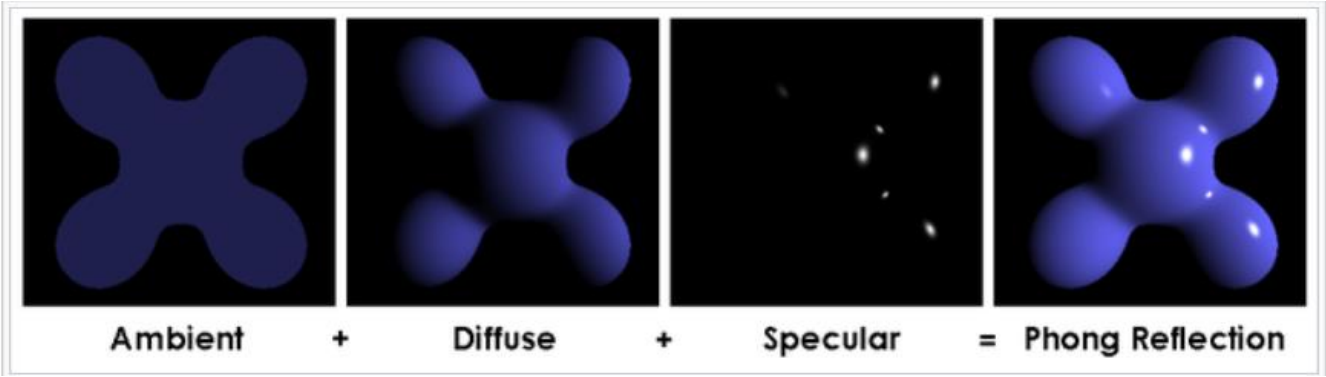


- **Spot light**



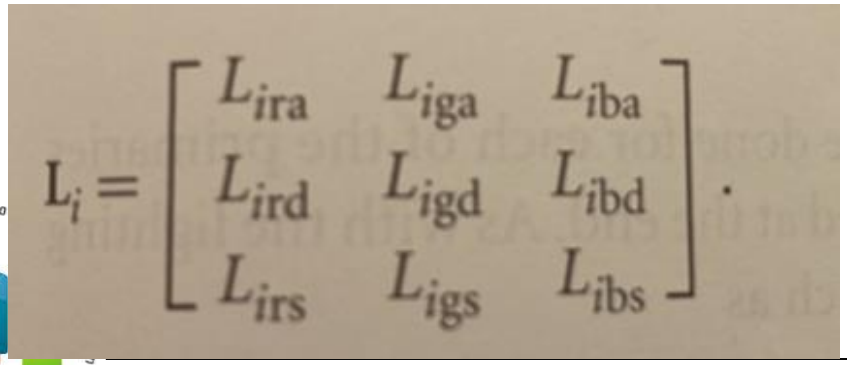
- **Phong lighting model**

- **Phong lighting model**
- **1975 Phong Bui Tuong이 제안**
- **A close enough approximation to physical reality to produce good rendering under a variety of lighting conditions and material properties**



- **Phong model supports three types of material-light interations – ambient, diffuse, and specular**
- **We assume that each source can have separate ambient, diffuse, and specular components for each of the three primary colors**
- **We need nine coefficients to characterize these terms at any point p on the surface**
- **Orthogonal splitting of light**

- 아래 행렬: 3x3 illumination matrix for the ith light source
- Row 1:ambient intensities for the R, G, B terms from source i
- Row 2: diffuse intensities for the R, G, B terms from source i
- Row 3: specular intentisites for the R, G, B terms from source i



Red	Green	Blue	Ambient
Red	Green	Blue	Diffuse
Red	Green	Blue	Specular

- **Similarly, for each point, we have nine coefficients that we can place in a matrix of reflection**

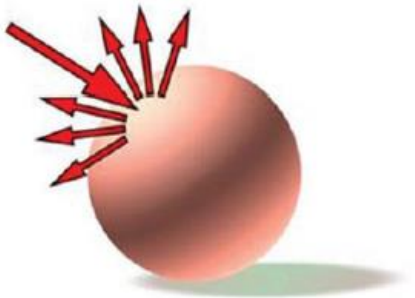
$$\mathbf{R}_i = \begin{bmatrix} R_{ira} & R_{iga} & R_{iba} \\ R_{ird} & R_{igd} & R_{ibd} \\ R_{irs} & R_{igs} & R_{ibs} \end{bmatrix}.$$

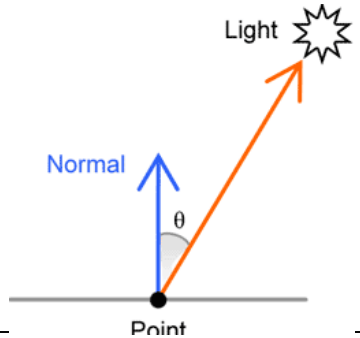
- **We can then compute the contribution for each color source by adding the ambient, diffuse, and specular component.**

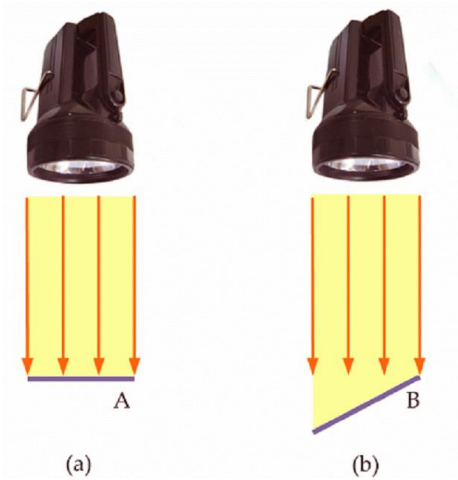
- We can omit the subscripts i, r, g, b
- $I = I_a + I_d + I_s = L_a R_a + L_d R_d + L_s R_s$
- a: ambient, d: diffuse, s: specular
- L: light source
- R: reflection
- I_a : ambient intensity
- I_d : diffuse intensity
- I_s : specular intensity

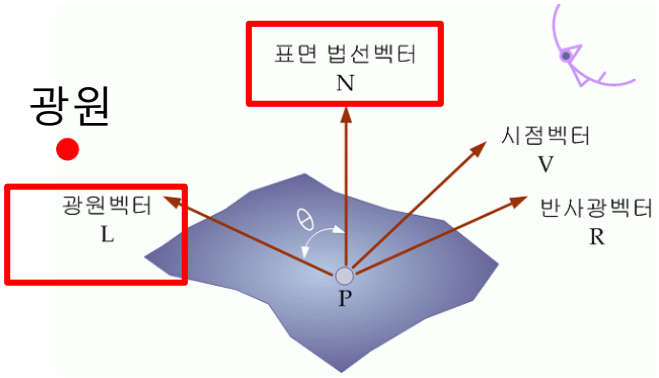
- **1. Ambient reflection**
- **The intensity of ambient light I_a is the same at every point on the surface**
- **The amount reflected is given by the ambient reflection coefficient k_a $(0 \leq k_a \leq 1)$**
- $I_a = k_a L_a$
- **L_a can be any of the individual light soruces or it can be a global ambient term**

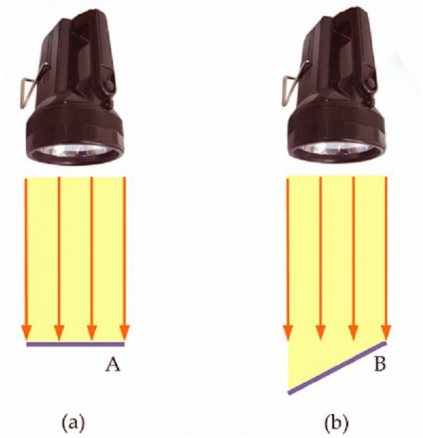
- **2. Diffuse reflection**
- **A perfectly diffuse reflector scatters the light that it reflects equally in all directions**
- **diffuse reflection 세기는 물체 면 (surface)의 법선 벡터(N)와 광원 벡터 (L)가 이루는 각도 (θ)와 관계가 있다**
- **광원 벡터 (L): 물체 면 (surface) P에서 광원을 향하는 벡터**
- **(a) 광원에 정면으로 노출된 면 (θ=0)**
- **(b) 광원에 비스듬히 놓인 면 (θ>0)**

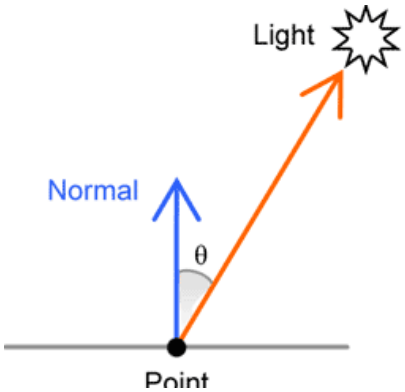










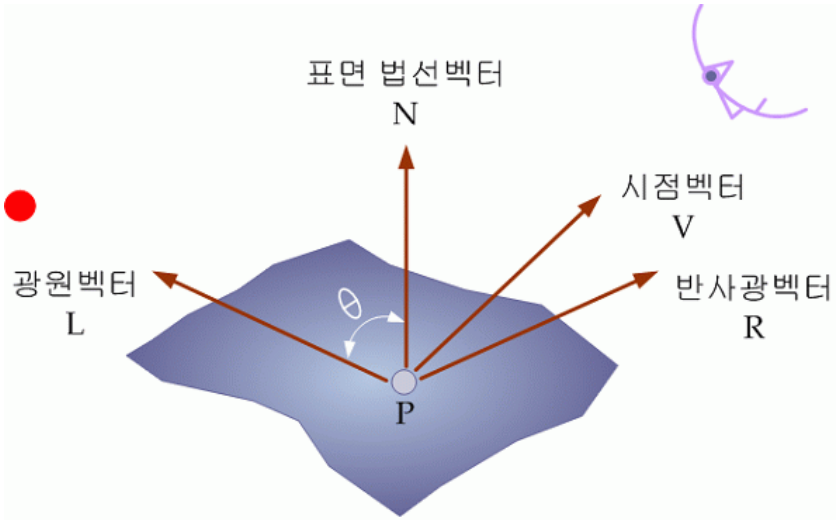


- 광원 벡터 (L): 물체 면 (surface) P에서 광원을 향하는 벡터
- θ : Surface P에서의 법선 벡터 (N)와 광원 벡터 (L)이 이루는 각
- (a)처럼 θ가 0도에 가까우면 광원에 정면으로 노출되는 면은 광원벡터와 법선벡터의 방향이 일치하는 면으로 가장 강한 diffuse reflection이 된다
- (b)처럼 θ가 점점 커질수록 diffuse reflection의 세기는 줄어든다
- θ가 90도이면 diffuse reflection의 세기는 0이 된다

즉, diffuse reflection의 세기는 $cos(\theta)$ 에 비례한다

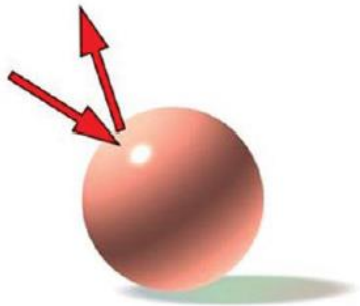
- **람베르트 법칙 (Lambert's law):** diffuse reflection의 세기는 광원 벡터 (L)와 법선 벡터(N)가 이루는 각 (입사각, θ)의 코사인 값, 즉 $cos(\theta)$,에 비례한다

- $$cos(\theta) = \frac{N \cdot L}{|N||L|}$$

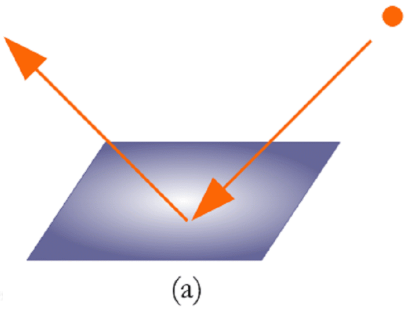


- **Diffuse reflection**
- **Diffuse reflection에서는 viewer의 위치는 중요치 않음 (why?)**
- **If we add in a reflection coefficient k_d representing the fraction of incoming diffuse light that is reflected, we have the diffuse reflection term:**
- $I_d = k_d L_d cos(\theta) = k_d L_d \max(\frac{N \cdot L}{|N||L|}, 0)$
- **이유: use zero rather than a negative value**

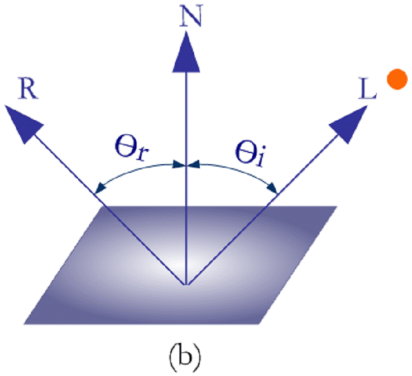
- **3. specular reflection**
- **What we are missing are the highlights that we see reflected from shiny objects**
- **A specular surface is smooth**
- **Diiffuse reflection과 다르게 모든 방향으로 균일하게 반사되지 않기 때문에 이 경우 viewer의 위치가 중요**



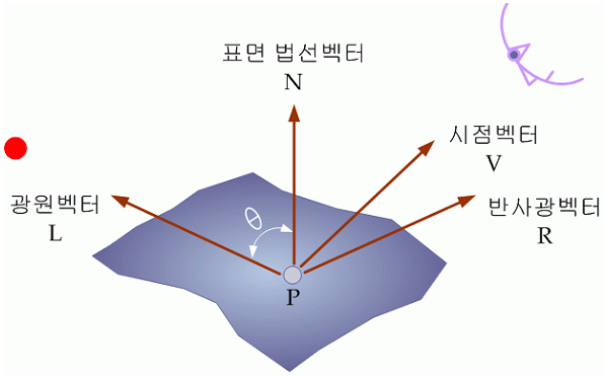
- 광원 벡터 (L): 물체 표면 (surface)에서 광원을 향하는 벡터
- 입사각 (θ_i): 광원 벡터 (L)와 surface의 법선벡터 (N)가 이루는 각
- 반사각 (θ_r): specular reflection (R)과 법선벡터 (N)가 이루는 각
- 거울처럼 완벽하게 매끄러운 면의 경우
- 입사각 θ_i 과 반사각 θ_r이 완전히 동일. $\theta_i = \theta_r$



(a)

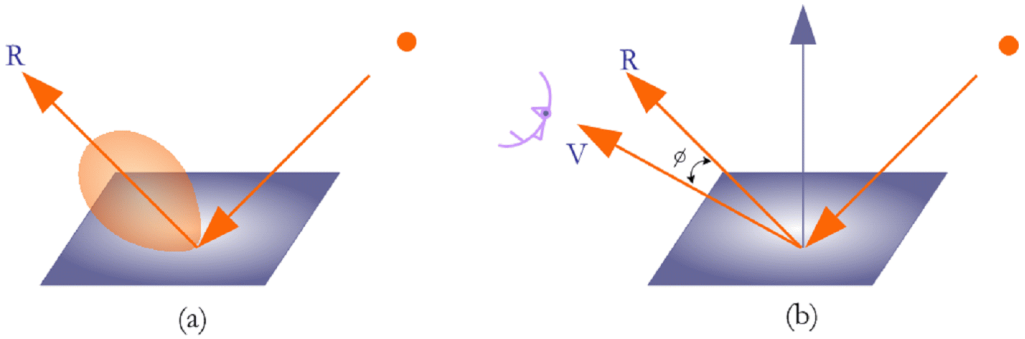


(b)

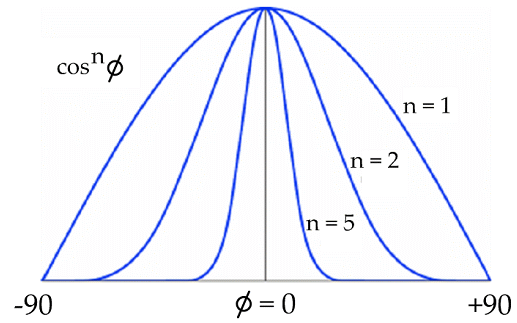


인천대학교

- 하지만, 실제는 어떤 물체 면이 완벽하게 매끄러울 수 없으므로 specular reflection도 약간은 다른 방향으로 흩어진다
- 시점 벡터 (V): surface로부터 viewer (카메라)위치로의 벡터
- ϕ: specular reflection (R)과 시점 벡터 V가 이루는 각
- ϕ가 0일때 viewer에게 가장 센 specular reflection
- ϕ가 커지면 커질수록 viewer가 받는 specular refleciton의 세기는 감소
- Viewer에게 오는 specular reflection의 세기는: $cos(\phi)$에 비례한다

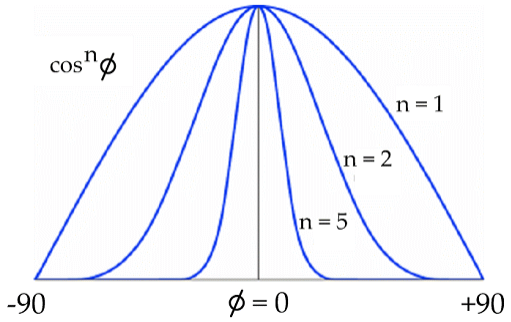


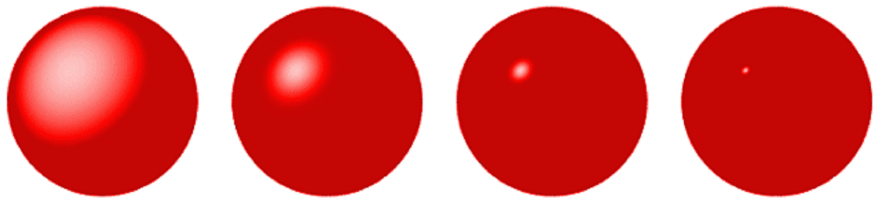
- Phong 반사모델에서는 $cos(\phi)$에 가해지는 승수 n에 의해 물체면의 매끄러운 정도를 반영한다
- 이 승수를 광택 계수 (shininess coefficient)라 한다
- Viewer에게 오는 경변 반사의 양은 광택 계수 고려시: $cos(\phi)^n$
- n값을 키우면 키울수록 $cos(\phi)^n$ 이 아래 그림과 같이 급속도로 좁아지므로 viewer가 정반사 되는 위치에서 조금만 벗어나도 viewer가 받는 specular reflection은 급속히 줄게 된다



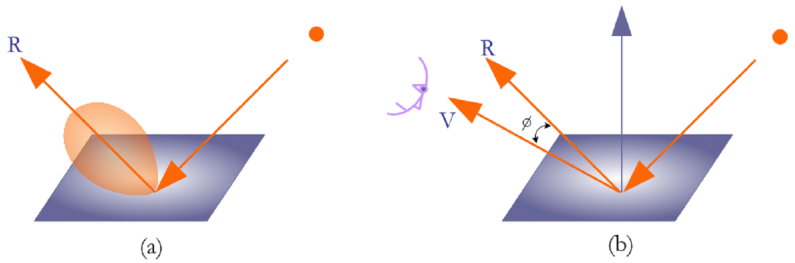
- Phong 반사모델에서는 $cos(\phi)$에 가해지는 승수 n에 의해 물체면의 매끄러운 정도를 반영한다
- 이 승수를 광택 계수 (shininess coefficient)라 한다
- Viewer에게 오는 경변 반사의 양은 광택 계수 고려시: $cos(\phi)^n$
- n값을 키우면 키울수록 $cos(\phi)^n$ 이 아래 그림과 같이 급속도로 좁아지므로 viewer가 정반사 되는 위치에서 조금만 벗어나도 viewer가 받는 specular reflection은 급속히 줄게 된다

- 하이라이트: specular reflection에 의해 물체면에 형성된 반짝이는 이미지
- Viewer에게 오는 경변 반사의 양은 광택 계수 고려시: $cos(\phi)^{n}$
- 예: 오른쪽으로 갈수록 n값을 크게한 것인데 하이라이트가 아주 좁은 시야에서 관찰 되게 된다





- **Specular reflection**
- **The coefficient k_s is the fraction of the incoming specular light that is reflected**
- **n: shiness coefficient**
- $I_s = k_s L_s cos(\phi)^{\,n} = k_s L_s \max(0, (\frac{R \cdot V}{|R||V|})^n)$

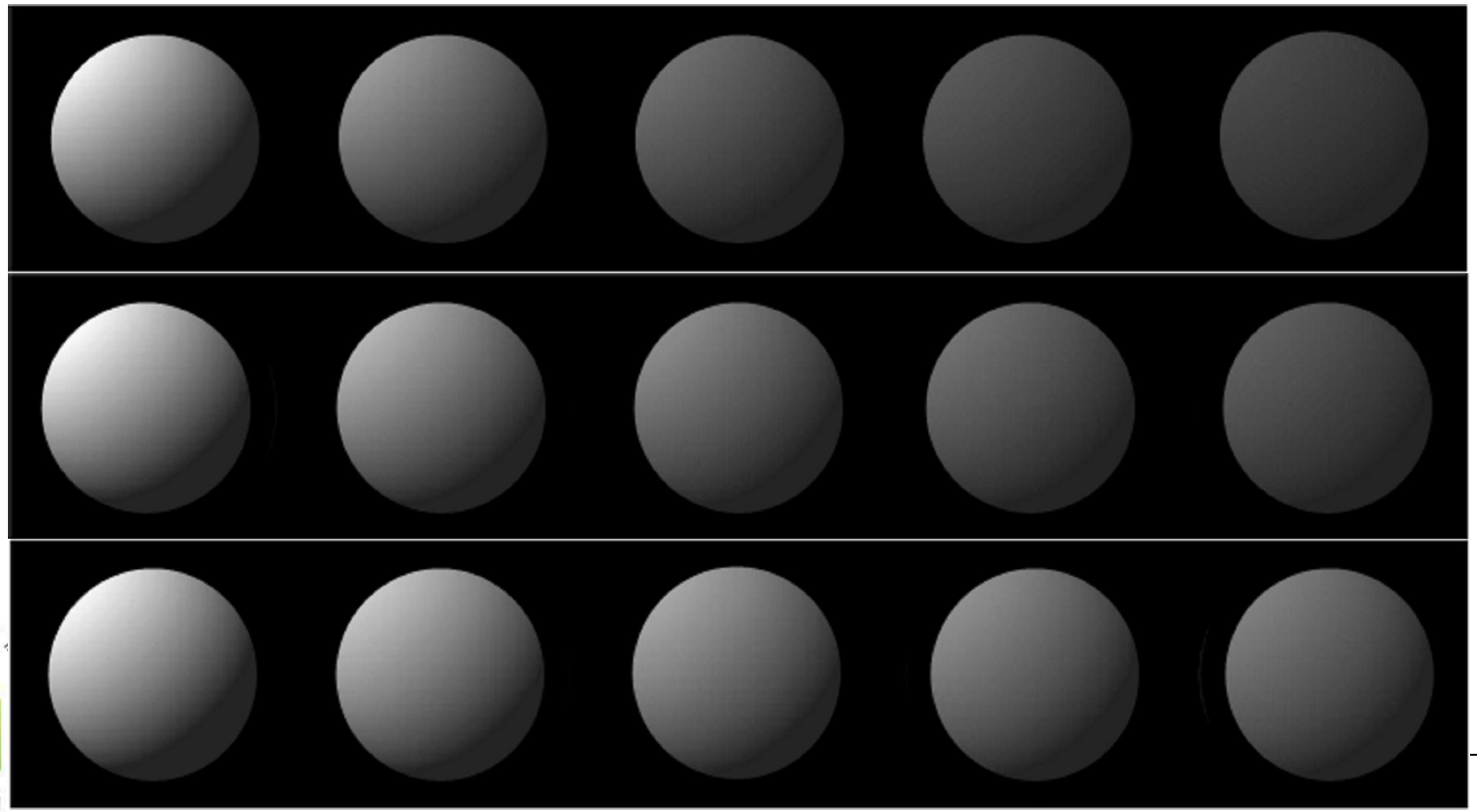


- **Distance term, 약화 함수 (attenuation function)**
- **광원과 물체와의 거리가 증가하면 할수록 빛은 약해진다**
- **그러나 프로그램에서 실제로 이러한 법칙을 적용하면 빛의 세기가 너무 급격히 약해지는 것을 알 수 있다**
- **OpenGL을 비롯하여 대부분의 그래픽스 소프트웨어에서는 다음과 같은 함수를 사용하여**
- **D: 물체와 광원과의 거리, a, b, c 조절 하는 계수**

$$f_{attenuation} = \frac{1}{a + bD + cD^2}$$

$$f_{attenuation} = \frac{1}{a + bD + cD^2}$$

- **a = b = 0, c = 1, a = b = .25, c = .5 , a = c = 0, b = 1**



- **Phong model including the distance term**
- I_a : ambient light, I_d : diffuse light, I_s : specular light
- k_a : ambient reflection coefficient
- $I = \frac{1}{a+bd+cd^2}(I_d + I_s) + I_a$
- $I = \frac{1}{a+bd+cd^2}(k_d L_d cos(\theta) + k_s L_s cos(\phi)^{\,n}) + k_a L_a$
- $I = \frac{1}{a+bd+cd^2}(k_d L_d \max\left(\frac{N \cdot L}{|N||L|}, 0\right), k_s L_s \max(0, (\frac{R \cdot V}{|R||V|})^n)) + k_a L_a$

- **Colored light**

- When dealing with colored sources and surfaces, we calculate each color component individually and simply add them to form the final color of reflected light

- Red: $I_r = \frac{1}{a+bd+cd^2}(I_{dr} + I_{sr}) + I_{ar}$

- Green: $I_g = \frac{1}{a+bd+cd^2}(I_{dg} + I_{sg}) + I_{ag}$

- Red: $I_b = \frac{1}{a+bd+cd^2}(I_{db} + I_{sb}) + I_{ab}$

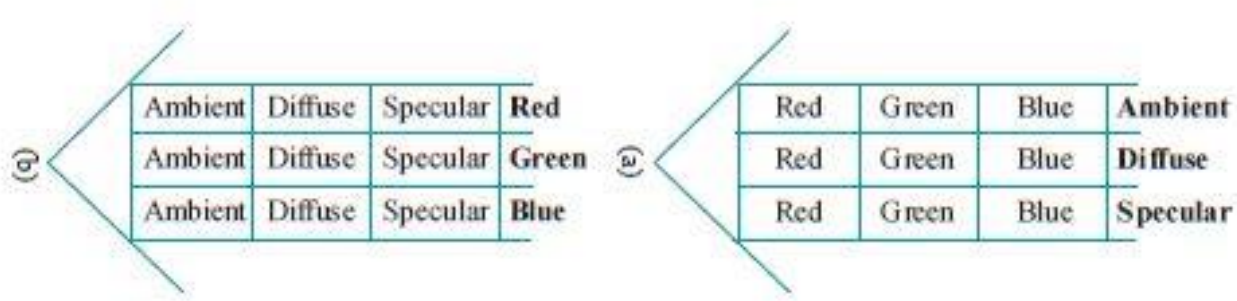


Figure 11.11: Orthogonal splitting of light: (a) Reflectance followed by color (b) Color followed by reflectance.

- 예: say the intensities of the diffuse light from source L are given by (R,G,B)=(0.3, 1.0, 1.0) and the diffuse coefficient (반사계수) of a vertex V by (R, G, B)=(0.8, 1.0, 0.8) and the angle θ of incidence at V is 60°
- Find the light emanating from V owing to the L diffuse
- $I_d = k_d L_d cos(\theta)$
- Red=0.5*0.3*0.8=0.12
- Green=0.5*1.0*1.0=0.5
- Blue=0.5*1.0.*0.8=0.4

- 예: say the intensities of the specular light from source L are given by (R,G,B)=(1.0, 1.0, 1.0) and the specular reflections of a vertex V by (R,G,B)=(0.0, 1.0, 0.6)
- The angle is $\phi = 60°$ and the shiness coeffiecient is 2.0
- Find the light emanating from V owing to the L specular
- $I_s = k_s L_s cos(\phi)^n$
- Red: 1.0*0.0*0.25=0.0
- Green: 1.0*1.0*0.25=0.25
- Blue: 1.0*0.6*0.25=0.15

- **OpenGL에서의 lighting**

- **OpenGL에서의 light (illumination)**
- **How to enable and disable light ?**

```
void glEnable(GL_LIGHTING);
```

```
void glDisable(GL_LIGHTING);
```

예: glEnable(GL_LIGHTING); // 조명 활성화

- **조명 기능이 활성화 되면 물체의 색은 light source와 material의 특성 (반사)에 의해서만 결정되고 glColor() 함수에 의해 정의된 vertex의 색은 무시된다**

- **How to enable and disable light source (광원)?**

```
void glEnable(LightSourceID);
void glDisable(LightSourceID);
```

예) glEnable(GL_LIGHT0); //0번 광원 활성화

예) glEnable(GL_LIGHT1); // 1번 광원 활성화

- **Total 8 light sources, 'GL_LIGHT0~GL_LIGHT7' are available to use**

1. Light source의 종류 및 위치 정함

예)

```
void InitLight() {
  GlfloatMyLightPosition [ ] ={1.0, 2.0, 3.0, 1.0}; // 광원위치
  glEnable(GL_LIGHTING);   //  조명 활성화
  glEnable(GL_LIGHT0);    // 0번 광원 활성화
  glLightfv(GL_LIGHT0, GL_POSITION, MyLightPosition);
  // 광원 위치 할당
}
```

- 2. Light source의 color 정함
- OpenGL에서 light source는 ambient, diffuse, specular

각각에 대하여 R, G, B, A 로 나누어서 정의

A: alpha component (후에 blending과 transparency에서 자세히 배움)

- 예) Set colors for each Light source

```
Glfloat MyLightAmbient[ ] = {1.0, 0.0, 0.0, 1.0}; //ambient = red
GLfloat MyLightDiffuse[ ] = {1.0, 1.0, 0.0, 1.0}; // diffuse = yellow
GLfloat MyLightSpecular[ ] = {1.0, 1.0, 1.0, 1.0}; // specular = white

 glLightv(GL_LIGHT0, GL_AMBIENT, MyLightAmbient);
 glLightv(GL_LIGHT0, GL_DIFFUSE, MyLightDiffuse);
 glLightv(GL_LIGHT0, GL_SPECULAR, MyLightSpecular);
// 첫 번째 인자: 광원, 두 번째 인자: 빛 종류, 세 번째 인자: RGBA
```

- **First OpenGL code using Light**

- Use just one light source (LIGHT_0)
- 1. Light source Position: (1, 2, 3, 1) => Positional light

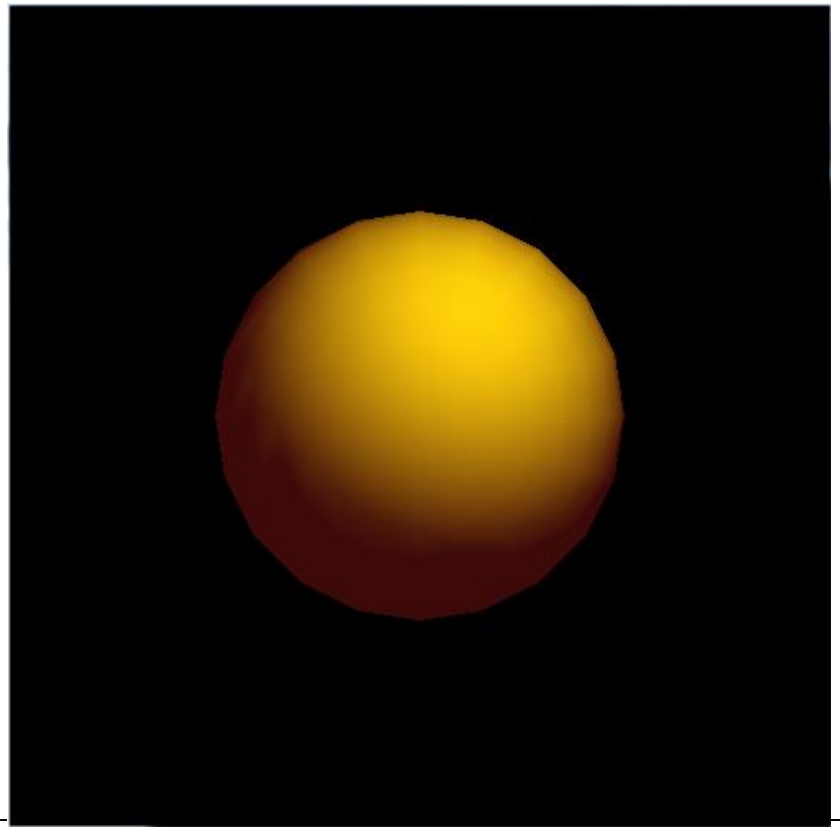
  (x, y, z, w)

  If w≠0, positional light, if w=0, directional light to (x, y, z) from (0,0,0)
- 2. Light color: Ambient(1,0,0) : red

  Diffuse(1,1,0) : yellow

  Specular(1,1,1) : white
- 3. Material : glutSolidSphere (1.0, 20, 16);
- 4. Projection: glOrtho(-2,2, -2, 2, -1, 1);

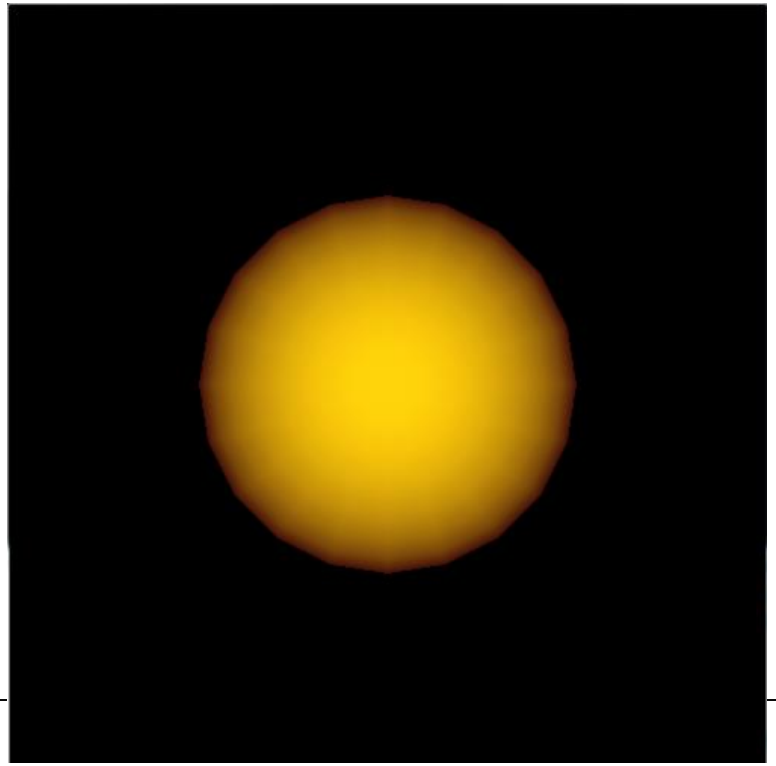
- **https://www.dropbox.com/s/819daoho6npp28o/light_0.txt?dl=0**



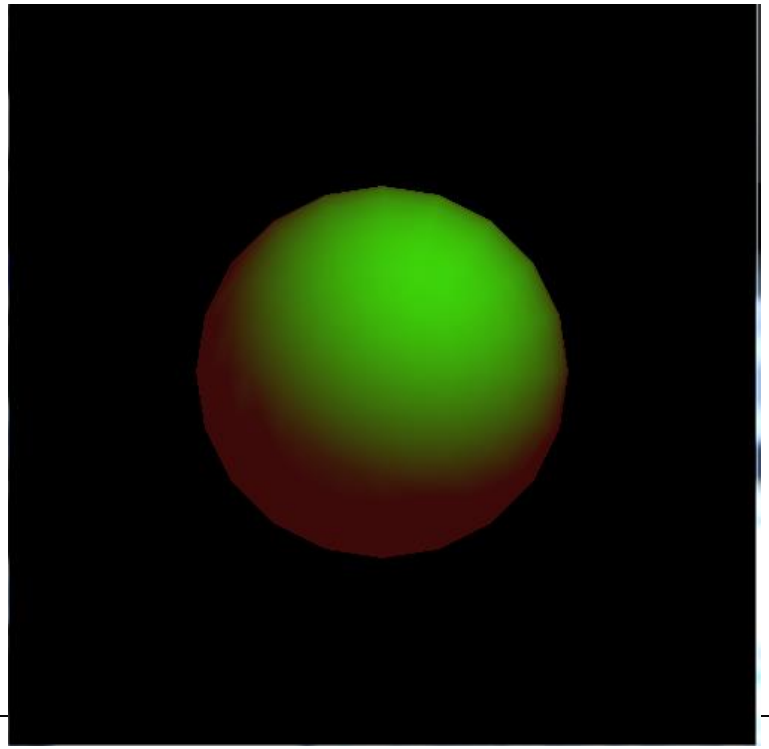
- 물체면 (material)에 대한 정보를 주지 않았으므로 default 값 사용
- Default로는 specular light이 (0.0, 0.0, 0.0, 1.0) 이다

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient color of material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	diffuse color of material
GL_AMBIENT_AND_DIFFUSE		ambient and diffuse color of material
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	specular color of material

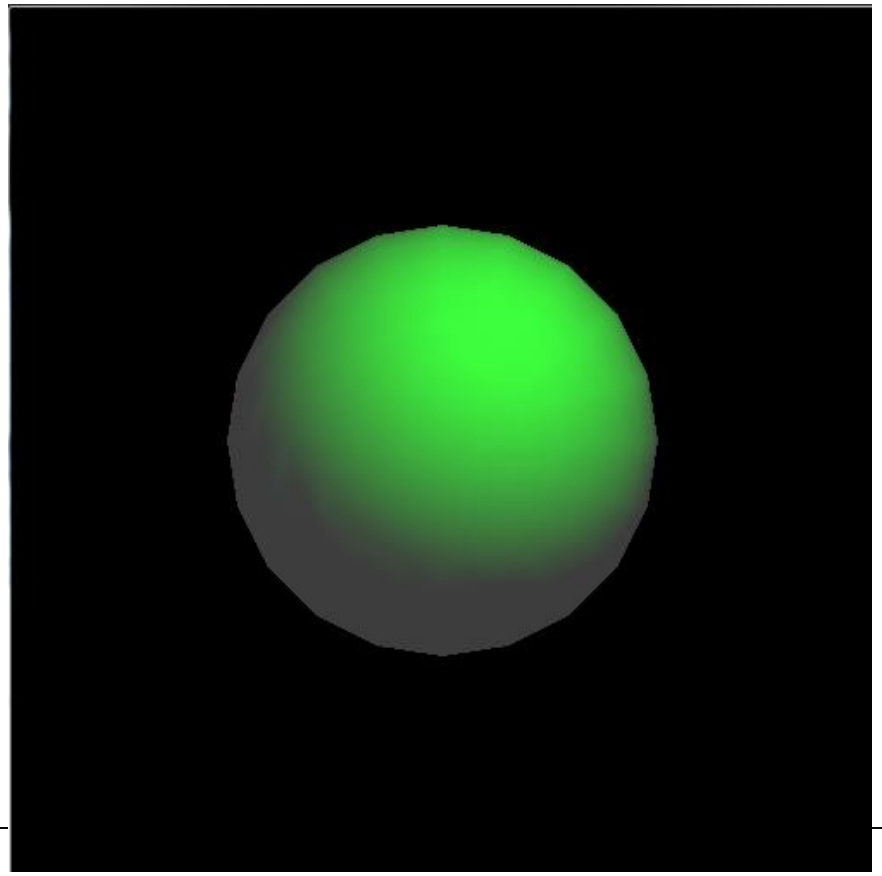
- 1) Now let’s modify the position of light source
- GLfloat MyLightPosition [ ] ={1.0, 2.0, 3.0, 1.0};
- => GLfloat MyLightPosition [ ] ={0.0, 0.0, 10.0, 1.0};



- 2) Now, let's modify the color of light (diffuse light)
- GLfloat MyLightPosition [ ] ={1.0, 2.0, 3.0, 1.0};
- GLfloat MyLightDiffuse[ ] = {0.0, 1.0, 0.0, 1.0};



- 3) Now, let’s modify the color of light (ambient light)
- GLfloat MyLightAmbient[ ] = {1.0, 1.0, 1.0, 1.0};
- // ambient : white



- **OpenGL에서의 light source의 default 값**

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	ambient RGBA intensity of light
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	diffuse RGBA intensity of light
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	specular RGBA intensity of light
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(*x*, *y*, *z*, *w*) position of light

- Positional light vs Directional light

- Light source Position: (x, y, z, w)

  If w≠0, positional light (위치성 광원)

  else, directional light (방향성 광원) to (x, y, z) from (0,0,0)

- 방향성 광원의 경우 광원의 위치는 생각하지 않고 (무한히 멀리있다고 생각함) 방향만 고려함

- 4D Homogeneous coordinate를 사용하는 경우, vector이면 마지막 cooridnate이 0으로 되어있는데 이를 지우면 원래 3D coordinate이 된다

- Default value of GL_POSITION is $[0, 0, 1, 0]^T$

- 거리에 따른 빛의 약화

- 현실적인 Lighting model은 광원과 물체와의 거리가 증가할수록 빛이 약해지는 것이다
- a: constant attenuation (상수 감쇄 계수)
- b: linear attenuation (1차 감쇄 계수)
- c: quadratic attenuation (2차 감쇄 계수)
- D: 물체와 광원과의 거리
- OpenGL default 값: a=1, b=0, c=0, 거리 감쇄 없음

$$f_{attenuation} = \frac{1}{a + bD + cD^2}$$

- glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 0.0);
- glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0);
- glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.0);
- a=0, b=1, c=0

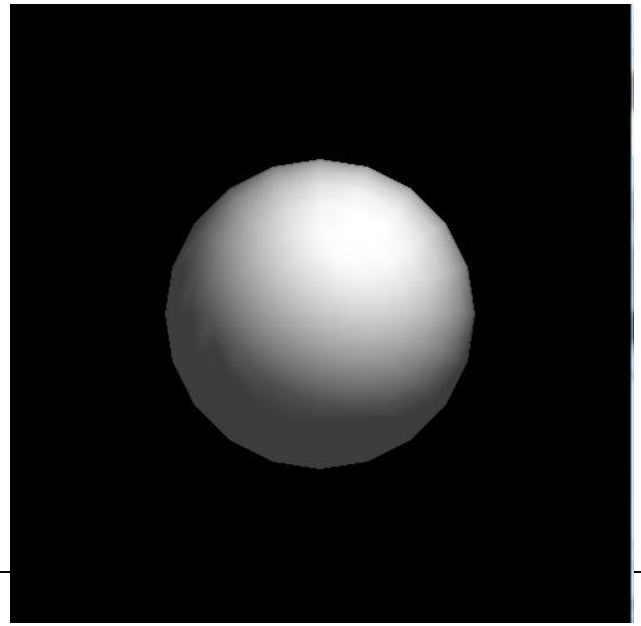
$$f_{attenuation} = \frac{1}{a + bD + cD^2}$$

- **https://www.dropbox.com/s/ws1kriws72pqp3f/light_1.txt?dl=0**

- 지금까지는 light source의 color, position등에 대해서 알아보았다

- 이번에는 material (물체)의 색과 물체 면의 매끄러움 (shininess)을 정의해보고 바꿔보자

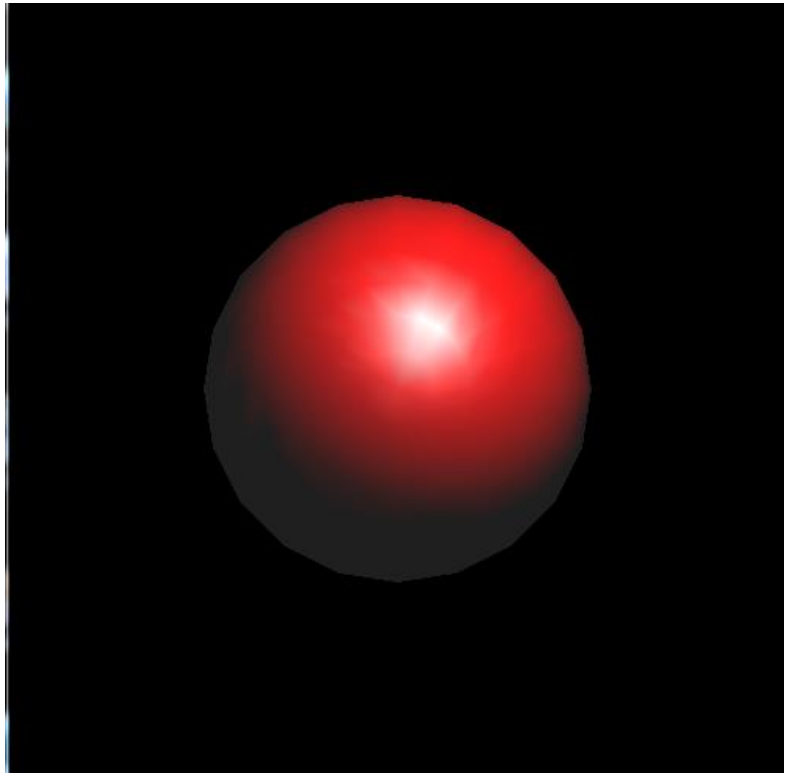
- Light color를 모두 white 로 하고 material의 특성을 변화시켜 보자
- 1. Ambient, diffuse, specular 모두 white로 바꿈
- GLfloat MyLightAmbient[ ] = {1.0, 1.0, 1.0, 1.0}; //ambient
- GLfloat MyLightDiffuse[ ] = {1.0, 1.0, 1.0, 1.0}; // diffuse
- GLfloat MyLightSpecular[ ] = {1.0, 1.0, 1.0, 1.0}; // specular



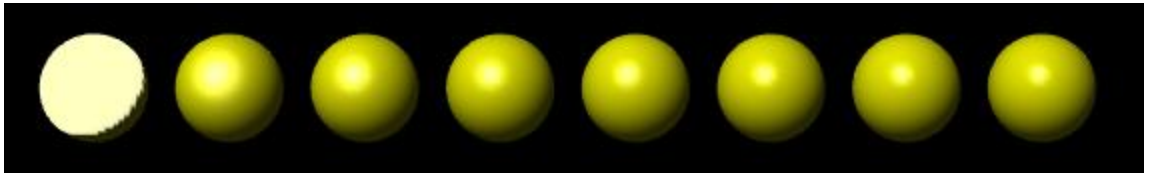
- 2. 물체의 반사 정도 결정 ambient, diffuse, specular, respectively
- shininess: 광택계수 (0-128사이의 값)
- GLfloat material_ambient[] = { 0.1, 0.1, 0.1, 1.0 }; // ambient (almost black)
- GLfloat material_diffuse[] = { 1.0, 0.0, 0.0, 1.0 }; // diffuse: red
- GLfloat material_specular[] = { 1.0, 1.0, 1.0, 1.0 }; // specular: white
- GLfloat material_shininess[] = { 25.0 }; // shininess:25

- glMaterialfv(GL_FRONT, GL_DIFFUSE, material_diffuse);
- glMaterialfv(GL_FRONT, GL_SPECULAR, material_specular);
- glMaterialfv(GL_FRONT, GL_AMBIENT, material_ambient);
- glMaterialfv(GL_FRONT, GL_SHININESS, material_shininess);

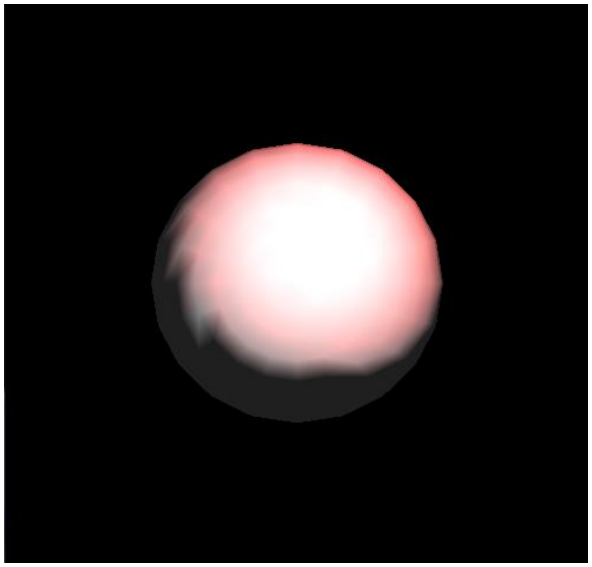
- **https://www.dropbox.com/s/v7z1xotx6lxqtx5/light_2.txt?dl=0**



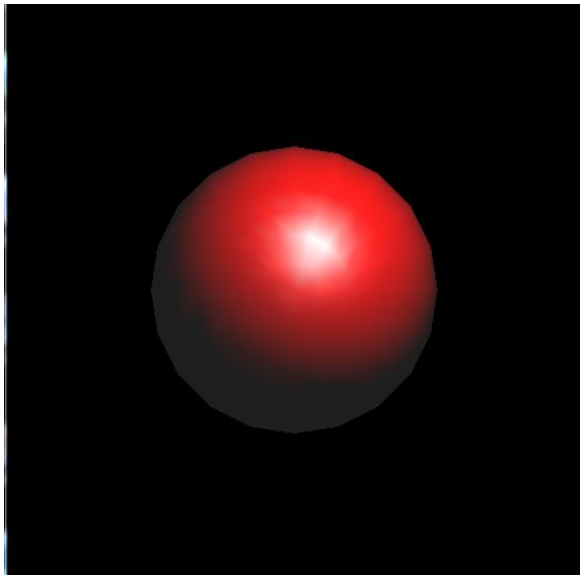
- 3. Shininess (SHARPNESS)조절
- (0-128까지 값가짐)
- shininess값 증가 (작고 날카로운 specular reflection)
- shininess값 감소 (반대로 넓고 덜 날카로운 specular reflection)
- shininess값을 줄여본다



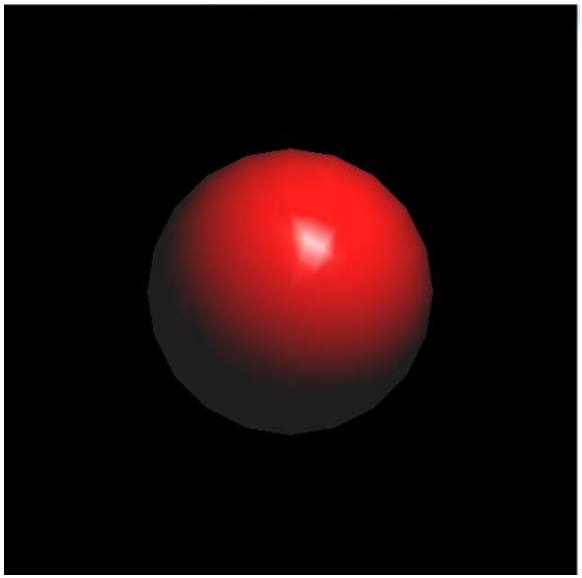
- **GLfloat material_shininess[] = { 1.0 };**



Shininess: 1



Shininess: 25



Shininess: 125

- **https://www.dropbox.com/s/z801l03g44a3cdm/light_3.txt?dl=0**

- **Summary: How to use light in OpenGL?**
- **1. Enable light (조명 기능 활성화)**
  - glEnable(GL_LIGHTING);
- **2. Light source의 종류 및 위치 정함**
  - 광원을 1개? 여러 개? glEnable(GL_LIGHT0); Positional? Directional?
  - glLightfv()
- **3. Light source의 color 정함**
  - Ambient, Diffuse, Specular 별로 color 부여
- **4. Material의 reflection 정함**
  - Ambient, Diffuse, Specular 별로 color 반사 정도 부여
  - glMaterialfv()

- **교재의 OpenGL 코드**
- **lightAndMaterial1.cpp**
- **처음 프로그램: blue ball의 material property 바꿀수 있고 움직일 수 있음**