Computer Graphics

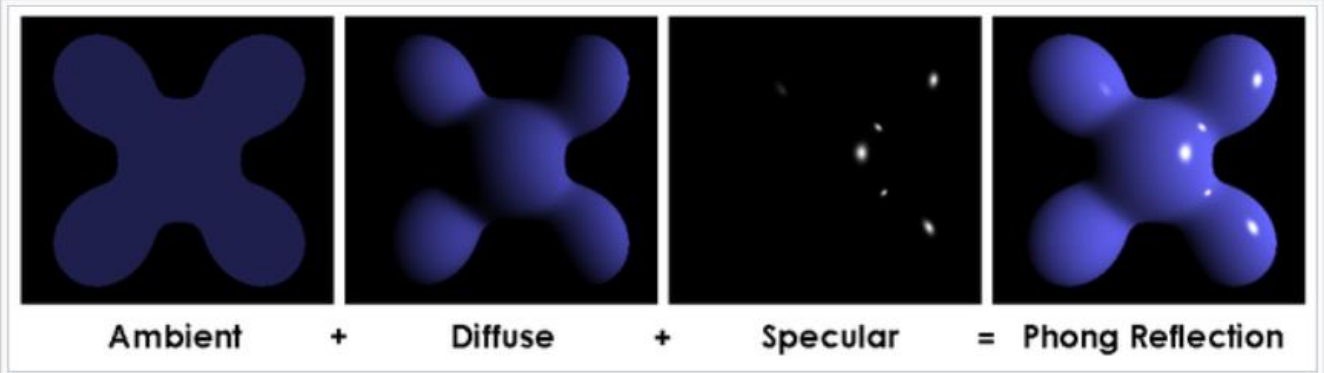
Prof. Jibum Kim

Department of Computer Science & Engineering

Incheon National University

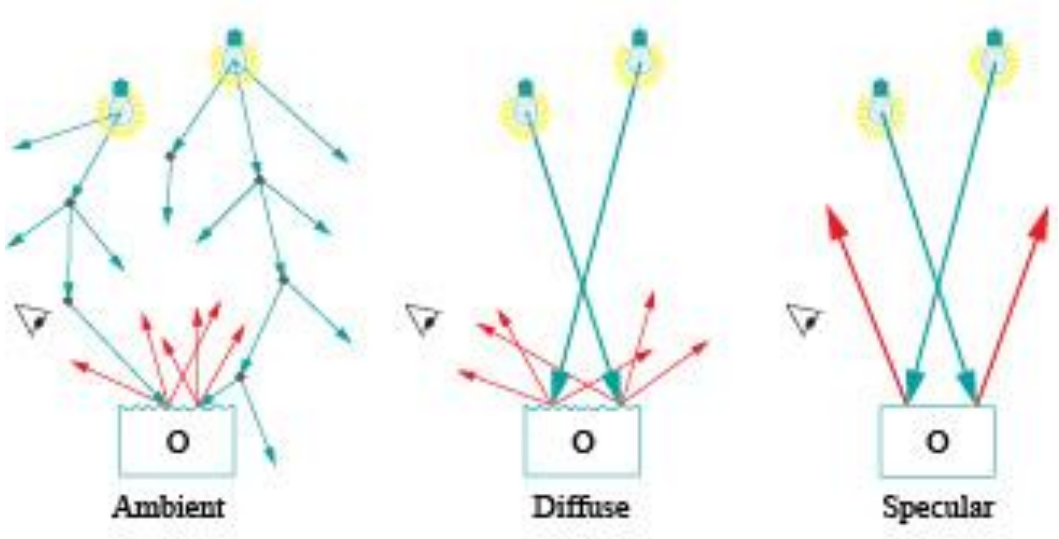
- **Phong's lighting model**

- **Phong's lighting model**
- **1975 Phong Bui Tuong이 제안한 지역 조명 모델로 OpenGL에서 사용하는 지역 조명 모델,계산이 아주 복잡하지 않으면서 꽤 현실적인 조명을 제공**
- **어떤 물체 (O)를 반사한 빛 (light) = Ambient + Diffuse + Specular**

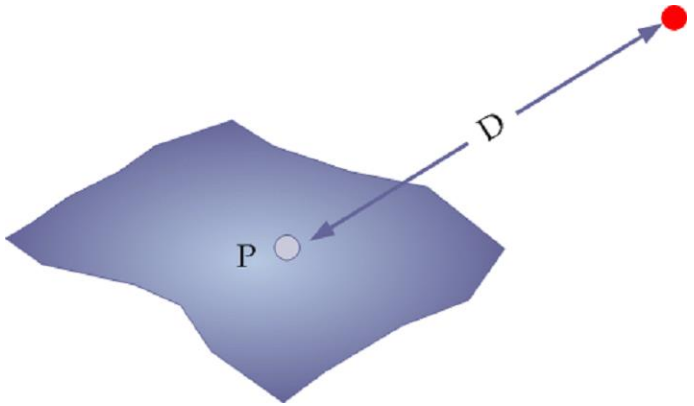


- **Ambient**

- **Ambient :** 광원에 의해 직접 빛이 들어오는 것이 아닌 여러 방향에서 최소 한번 이상 물체에 빛이 부딪쳐서 들어오는 빛이 물체 O에 반사되는 것
- 물체 면으로 입사되는 다양한 빛의 경로를 일일이 추적하지 않고
- 광원에 직접 노출되지 않는 면에 밝기를 부여하는 목적
- 지역 조명 모델에서 **전역 조명 모델 효과를 근사적으로 부여**하고자 함



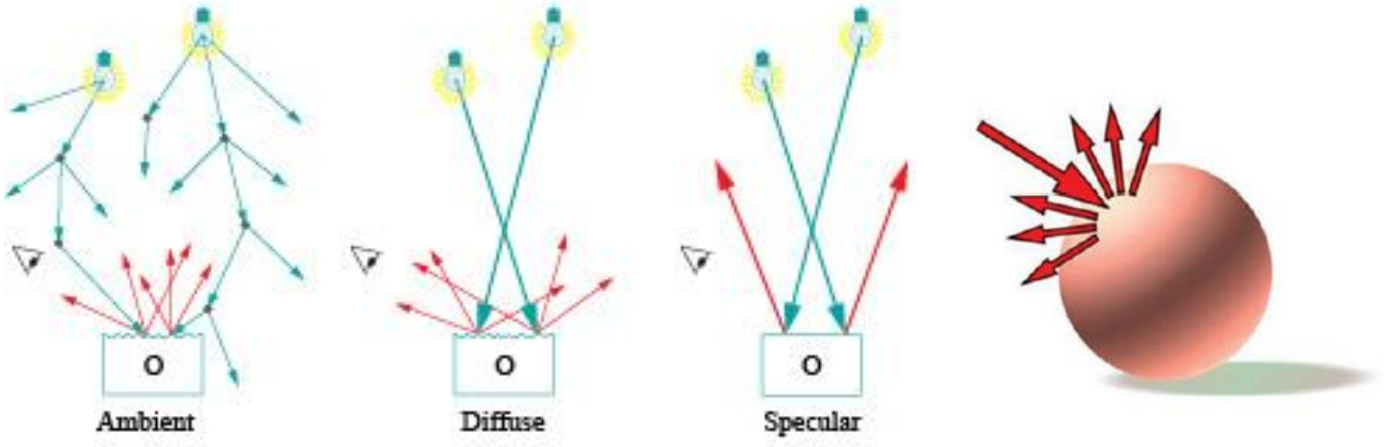
- 주변광 (ambient reflective light): 주변 반사에 의해 우리 눈으로 반사되는 빛을 의미
- Ia: ambient light의 세기
- Ka: ambient light의 반사율 (주변광 계수), 0< Ka < 1
- D: 물체와 광원과의 거리
- 주변광의 세기 (ambient reflection) =$K_a(I_a/D^2)$



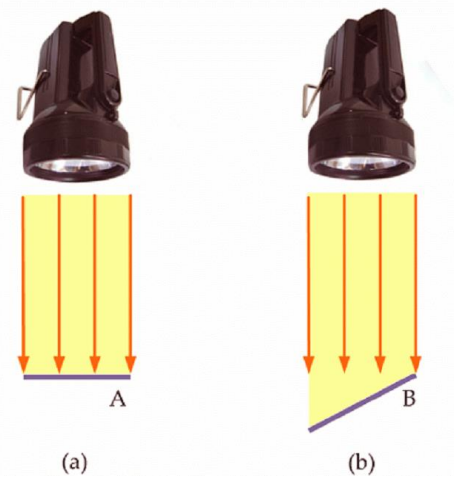
- Ka: 주변광 계수 (ambient coefficient)
- 이 계수를 증가시키면 광원의 위치나 물체 면의 방향과 무관하게 모든 물체 면의 밝기가 증가한다
- 아래 그림: 오른쪽으로 갈수록 주면광 계수 (Ka)를 증가시킴

- **Diffuse**

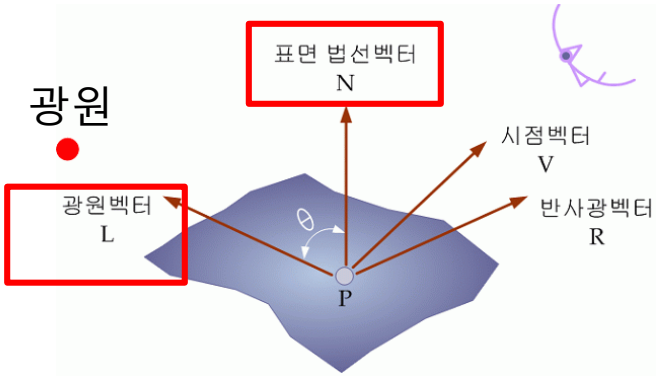
- 주변광만 사용시 물체의 모든 부분이 동일한 밝기로 보인다. 즉, 광원의 위치에 따른 명암차이가 드러나지 않기 때문에 입체감이 없다
- Diffuse 반사: 물체면과 광원과의 공간적인 관계에 따라 명암을 부여한다
- Diffuse: 광원으로부터의 빛이 물체 O로 직접 부딪쳐서 모든 방향으로 일정하게 scattered되면서 반사되는 것을 모델링, 난반사를 모델링
- Diffuse 고려 시에 viewer의 위치는 고려되지 않는다. Why?

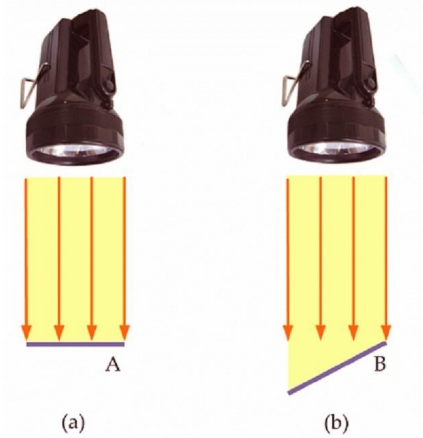


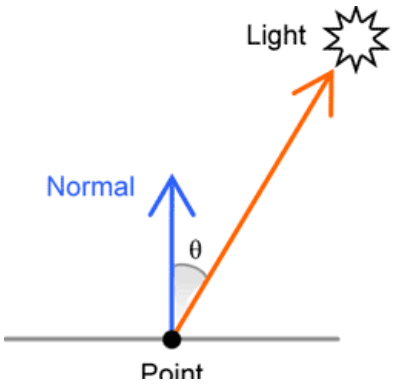
- **확산광 (diffusive reflection)세기는 물체 면이 서 있는 방향 (즉, 물체면의 법선 벡터의 방향)에 따라 달라진다**
- **방향상 광원이라고 할 때 아래 그림 (a)와 (b)중에 어떤 경우에 어떤 경우가 더 많은 확산광을 반사할까?**
- **(a) 광원에 정면으로 노출된 면**
- **(b) 광원에 비스듬히 놓인 면**



- **(a) 처럼 어떤 물체 면이 광원에 정면으로 노출되어 있다는 것을 수학적으로 어떻게 알까?**



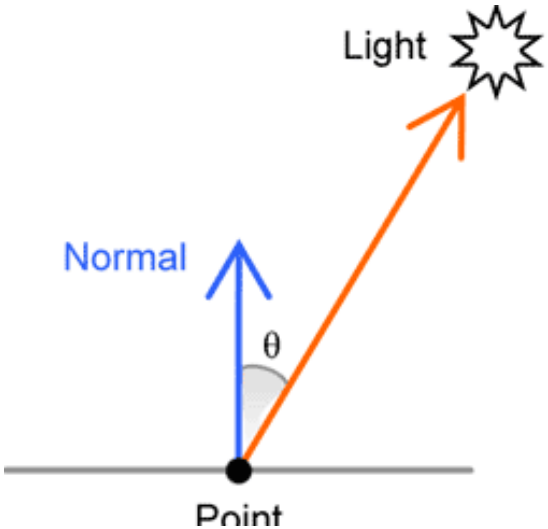




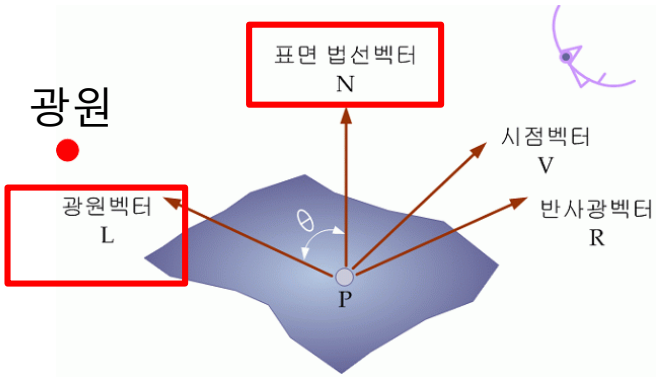
- 광원 벡터: 물체 면 (표면)에서 광원을 향하는 벡터
- θ : 물체면의 법선 벡터와 광원 벡터가 이루는 각
- (a)처럼 θ가 0도에 가까우면 광원에 정면으로 노출되는 면은 광원벡터와 법선벡터의 방향이 일치하는 면으로 가장 강한 반사광 (확산광)이 된다
- (b)처럼 θ가 점점 커질수록 반사광의 세기는 줄어든다
- θ가 90도이면 반사광의 세기는 0이 된다

즉, 반사광의 세기는 $\cos(\theta)$ 에 비례한다

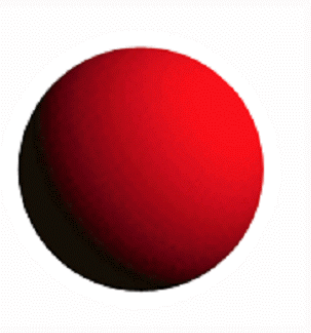
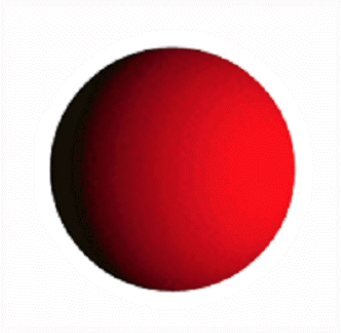
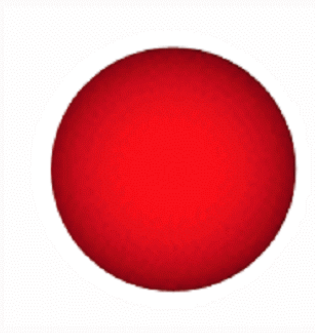
- **람베르트 법칙:** 확산광의 세기는 광원 벡터 (L)와 법선 벡터(N)가 이루는 각 (입사각, θ)의 코사인 값, 즉 $cos(\theta)$,에 비례한다



- **확산 광 (diffusive reflective light)**
- **I_d: diffuse light의 세기**
- **K_d: diffuse light의 반사율 (0< Kd<1), 확산 계수**
- **D: 물체와 광원과의 거리**
- **θ : 물체면의 법선 벡터와 광원 벡터가 이루는 각**
- **람베르트 법칙 이용**
- **Diffusive reflection (확산광)의 세기 $=K_d I_d cos(\theta) \,/\, D^2$**
- $$=K_d I_d \frac{N \cdot L}{|N||L|} \,/\, D^2$$
- **엄격하게 적용하면 내적 계산 값이 음이 나오는걸 방지 하기 위하여**
- **Diffusive reflection (확산광)의 세기 $=K_d I_d \,/\, D^2 \max\left(\frac{N \cdot L}{|N||L|}, 0\right)$**

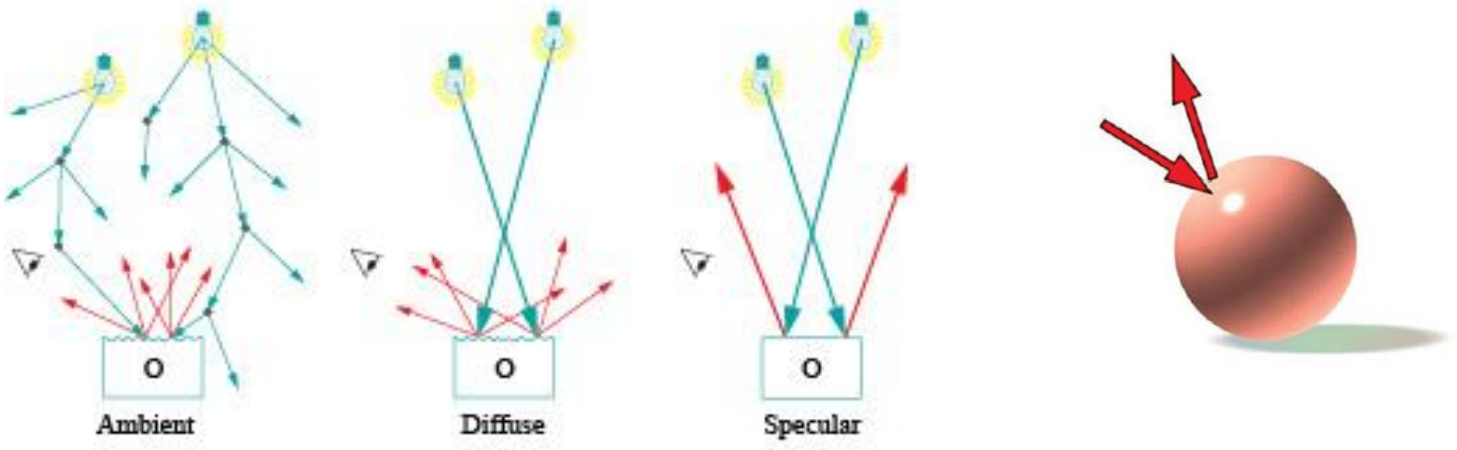


- 광원에 위치에 따른 확산광
- 주변광과 다르게 물체면에서의 광원벡터와 법선벡터가 이루는 각에 의해 차등적으로 밝기가 결정된다
- 예: 광원의 위치 (우상단), 광원의 위치 (우측 중앙), 광원의 위치 (정중앙)

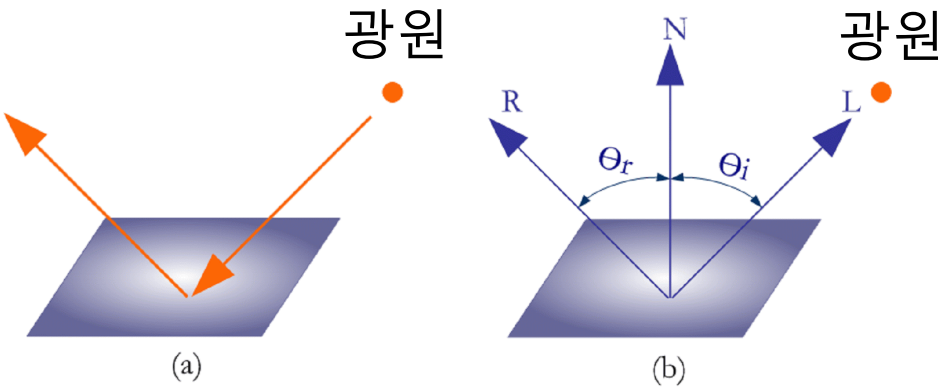




- **Specular**

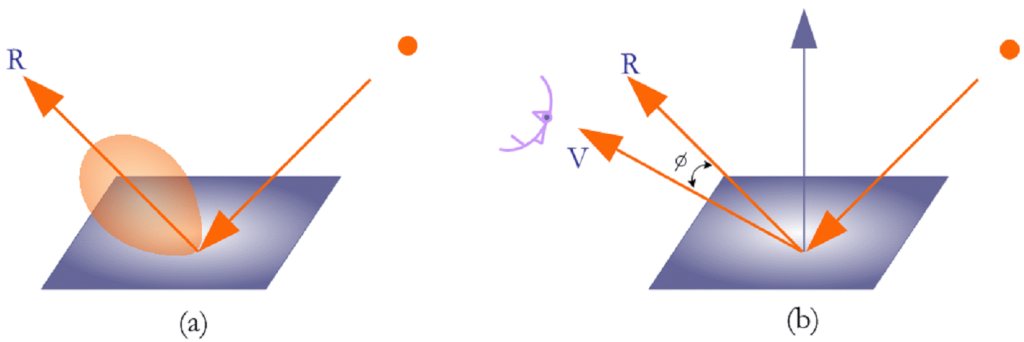
- **Specular:** 광원으로부터의 빛이 물체 O에 부딪쳐서 거울처럼 반사되는것을 모델링
- 빛의 정반사에 의한 것으로 광원에서 나오는 빛의 색이 그대로 우리 눈에 들어옴
- 반사시에 난반사와 다르게 모든 방향으로 균일하게 반사되지 않기 때문에 Diffuse와 다르게 이 경우 viewer의 위치가 중요하다



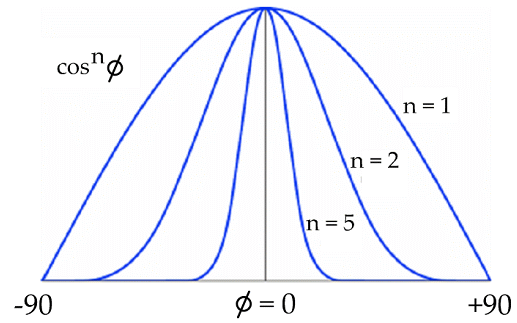
- 광원 벡터 (L): 물체 면 (표면, surface)에서 광원을 향하는 벡터
- 입사각 (θ_i): 광원 벡터 (L)와 물체의 법선벡터 (N)가 이루는 각
- 반사각 (θ_r): 경면 반사광 (R)과 법선벡터 (N)가 이루는 각
- 거울처럼 완벽하게 매끄러운 면의 경우
- 입사각 θ_i 과 반사각 θ_r이 완전히 동일. $\theta_i = \theta_r$



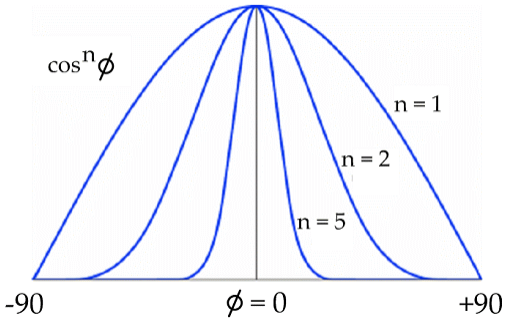
- 하지만, 실제는 어떤 물체 면이 완벽하게 매끄러울 수 없으므로 경면광도 약간은 다른 방향으로 흩어진다
- 시점 벡터 (V): 물체면으로 부터 viewer (카메라)위치 로의 벡터
- ϕ: 반사광 R과 시점 벡터 V가 이루는 각
- ϕ가 0일때 viewer에게 가장 많은 경면 반사량
- ϕ가 커지면 커질수록 viewer가 받는 경면 반사량을 줄어들게 된다
- Viewer에게 오는 경면 반사의 양은: $cos(\phi)$에 비례한다

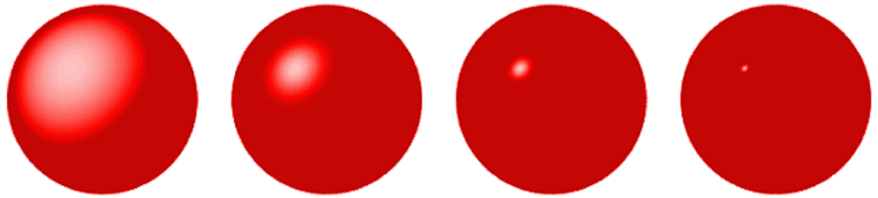


- Phong 반사모델에서는 $cos(\phi)$에 가해지는 승수 n에 의해 물체면의 매끄러운 정도를 반영한다
- 이 승수를 광택 계수 (shininess coefficient)라 한다
- Viewer에게 오는 경변 반사의 양은 광택 계수 고려시: $cos(\phi)^n$
- n값을 키우면 키울수록 $cos(\phi)^n$ 이 아래 그림과 같이 급속도로 좁아지므로 viewer가 정반사 되는 위치에서 조금만 벗어나도 viewer가 받는 경면 반사량은 급속히 줄게 된다

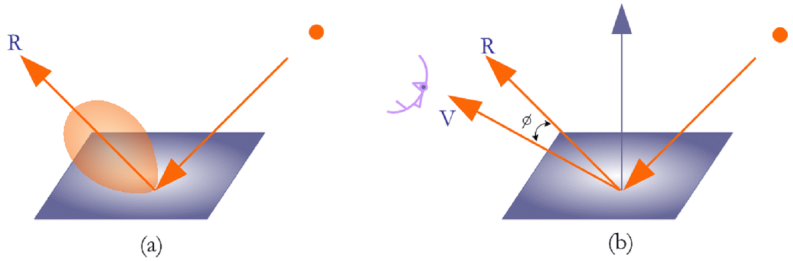


- 하이라이트: 경면광에 의해 물체면에 형성된 반짝이는 이미지
- Viewer에게 오는 경변 반사의 양은 광택 계수 고려시: $cos(\phi)^{n}$
- 예: 오른쪽으로 갈수록 n값 키운 것인데 하이라이트가 아주 좁은 시야에서 관찰 되게 된다





- 경면 광 (specular reflective light)
- I_s: specular light의 세기
- K_s: specular light의 반사율 (0< Ks<1), , 경면 계수
- D: 물체와 광원과의 거리
- ϕ: 반사광 R과 시점 벡터 V가 이루는 각
- n : 광택 계수 (shininess coefficient)



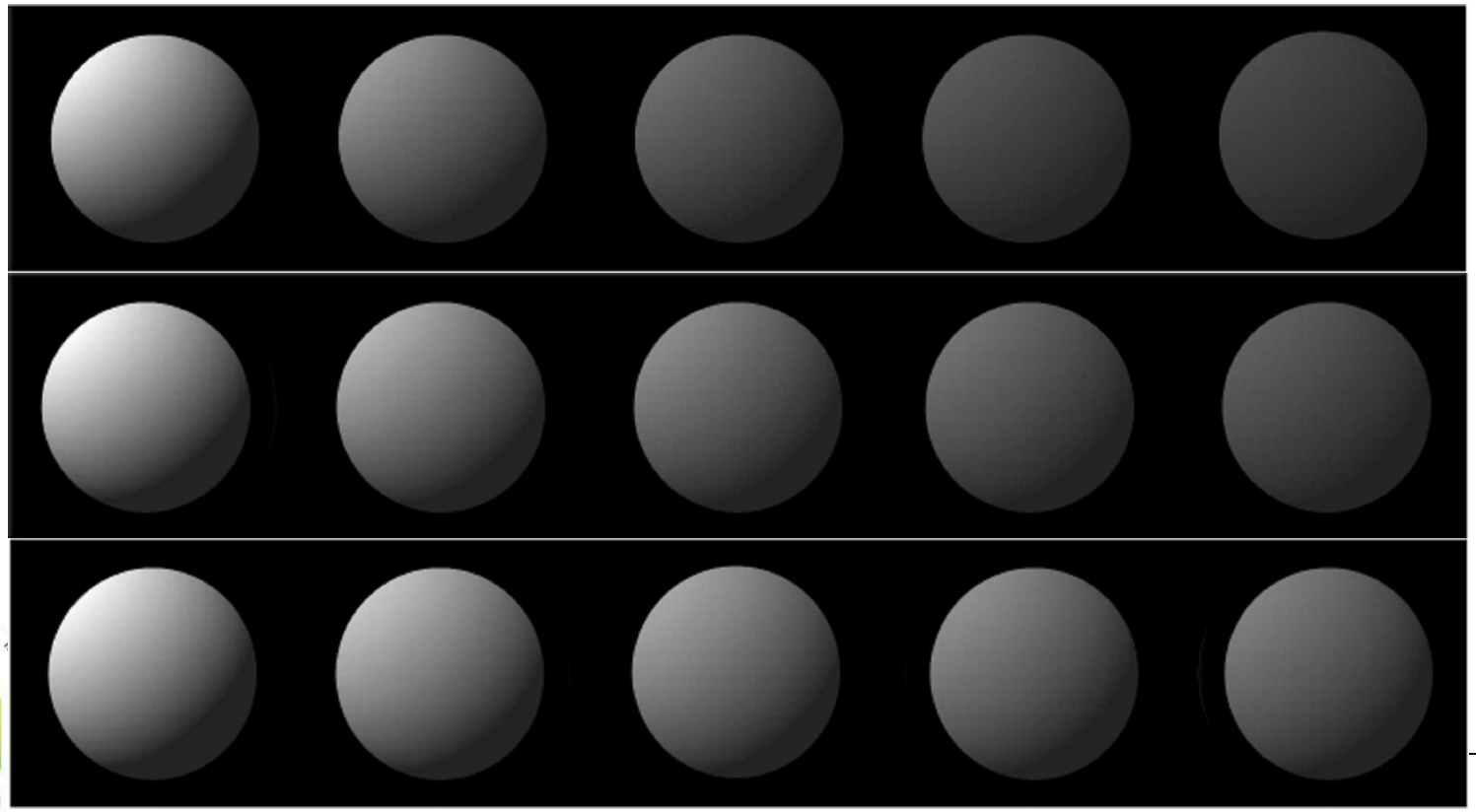
- Specular reflection (경면광)의 세기 $= K_s I_s cos(\phi)^{n} / D^2$
- $= K_s I_s (\frac{R \cdot V}{|R||V|})^{n} / D^2$
- 엄격하게 적용하면 내적 계산 값이 음이 나오는걸 방지 하기 위하여
- Specular reflection (경면광)의 세기=$K_s I_s / D^2$ max(0, $(\frac{R \cdot V}{|R||V|})^{n}$)

- 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

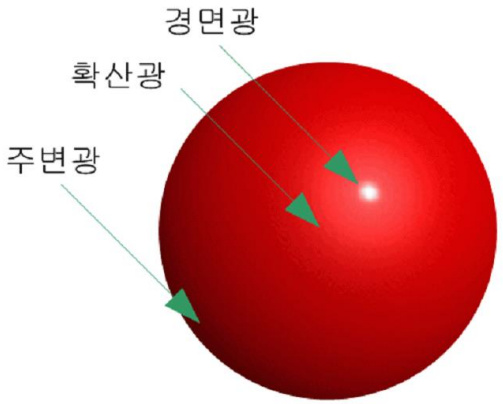
$$f_{attenuation} = \frac{1}{a + bD + cD^2}$$

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

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

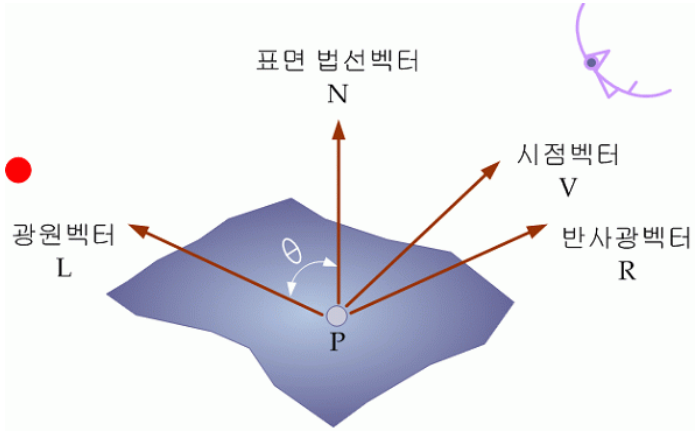


- **Phong (지역) 조명 모델**
- **모든 물체면에 대해서 주변광, 확상광, 경면광이라는 세 가지 반사광의 크기를 더하여 최종인 밝기가 결정된다**

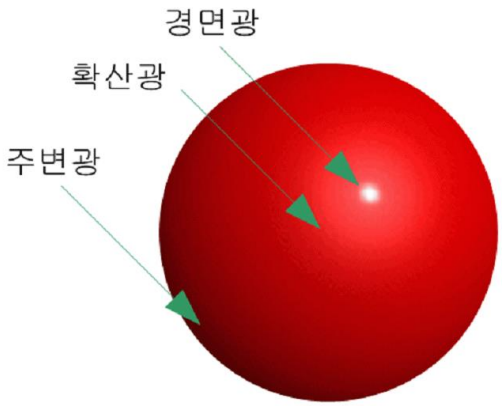


- **Phong’s lighting model 정리**

- **Phong ‘s lighting model**
- **I (물체에서 반사되는 빛)= Ambient 반사+ Diffuse 반사+ Specular 반사**
- $$I = \frac{K_a I_a + K_d I_d \max\left(\frac{N \cdot L}{|N||L|}, 0\right), + K_s I_s \max\left(0, \left(\frac{R \cdot V}{|R||V|}\right)^n\right)}{a + bD + cD^2}$$

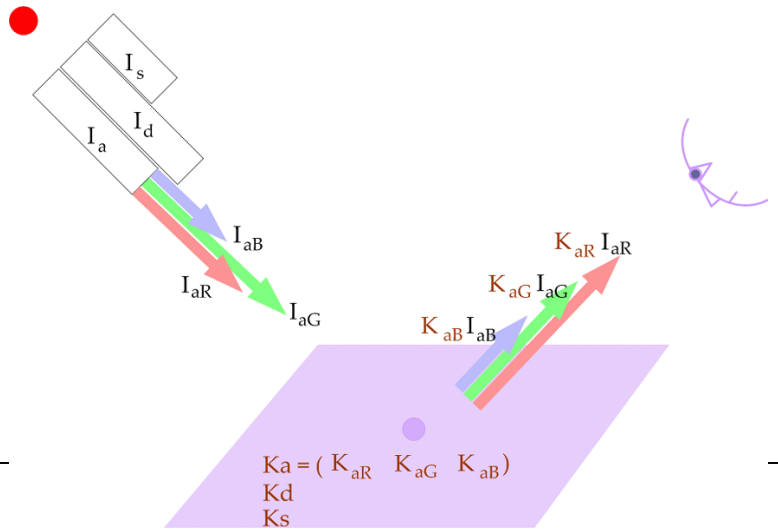


반사 종류	
Ambient (주변광)	Ia: ambient light의 세기 Ka: ambient light의 반사율
Diffuse (확산광)	Id: diffuse light의 세기 Kd: diffuse light의 반사율
Specular (경면광)	Is: specular light의 세기 Ks: specular light의 반사율
D: 물체와 광원과의 거리 θ: 광원벡터 L과 법선벡터 N이 이루는 각 ϕ: 반사광 R과 시점 벡터 V가 이루는 각	



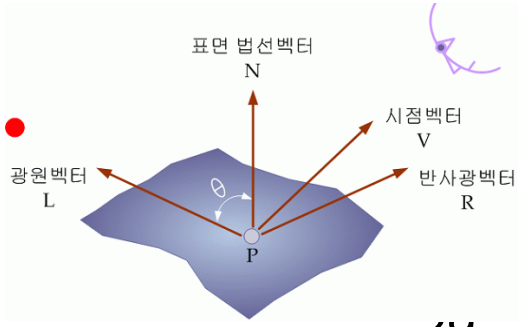
- **Colored light**

- **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**
- **실제 색은 (R, G, B)로 표현되고 각 색의 반사율은 물체 특성에 따라서 R, G, B색에 대해서 각각 별도로 적용해야 된다**
- **광원이 여러 개인 경우는 각각의 광원에서 나오는 빛을 모두 합산한다**



- 예: 한 개의 광원과 물체 (P)가 존재한다. 물체와 광원과의 거리 (D)는 1이다. 광원벡터 L과 법선벡터 N이 이루는 각은 30도이고, 반사광 R과 시점 벡터 V가 이루는 각은 0도이다
- 광원으로부터의 ambient light (R,G,B)=(0.4, 0.9, 0.2), diffuse light (R,G, B)=(0.3, 1.0, 1.0), specular light (R,G,B)=(1.0, 1.0, 1.0)이다.
- 물체에서의 ambient light의 반사계수 (R,G,B)=(0.9, 0.9, 0.1), diffuse light의 반사 계수 (R,G,B)=(0.8, 1.0, 0.8), specular light의 반사 계수 (R,G,B)=(0.0,1.0,0.6)이다
- 광택 계수=2, 물체로부터 반사되는 빛을 각각 구해보자

	R	G	B
Ambient			
Diffuse			
Specular			



- **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() {
  Glfloat MyLightPosition [ ] ={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

- 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

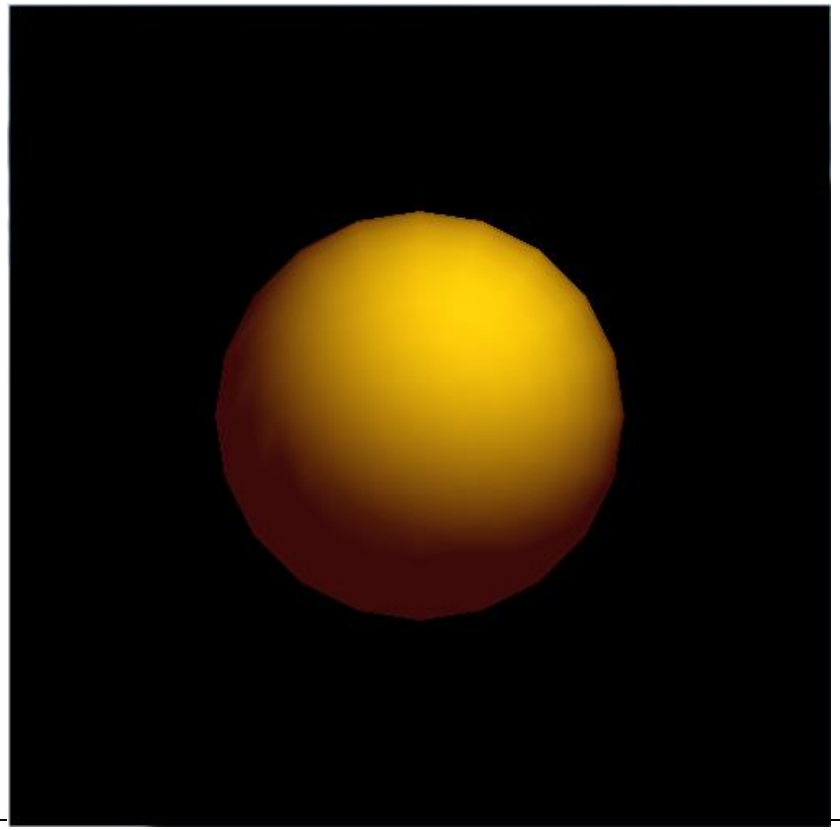
  (x, y, z, w)

  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);

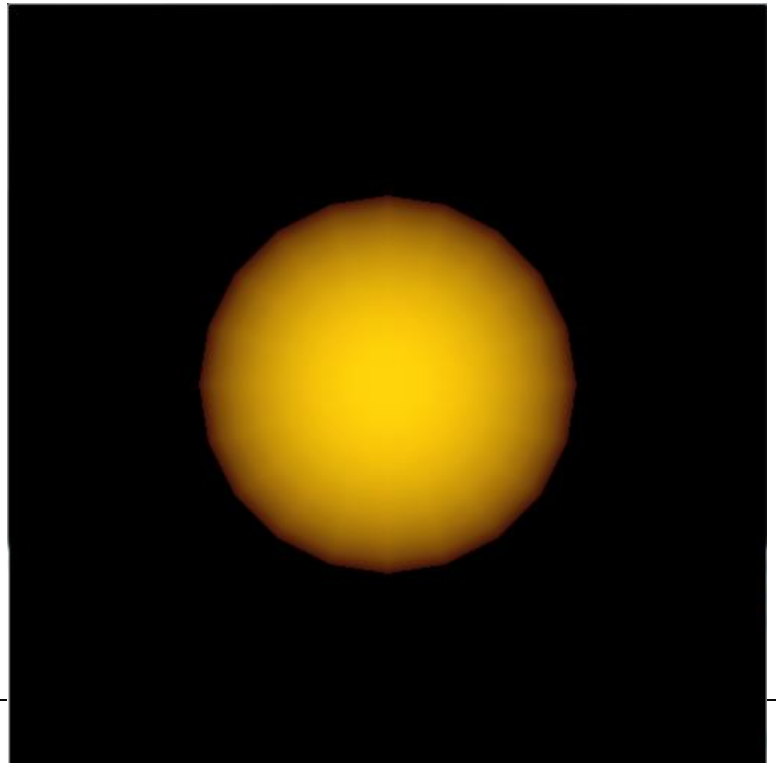
- **https://www.dropbox.com/s/819daoho6npp28o/light_0.txt?dl=0**



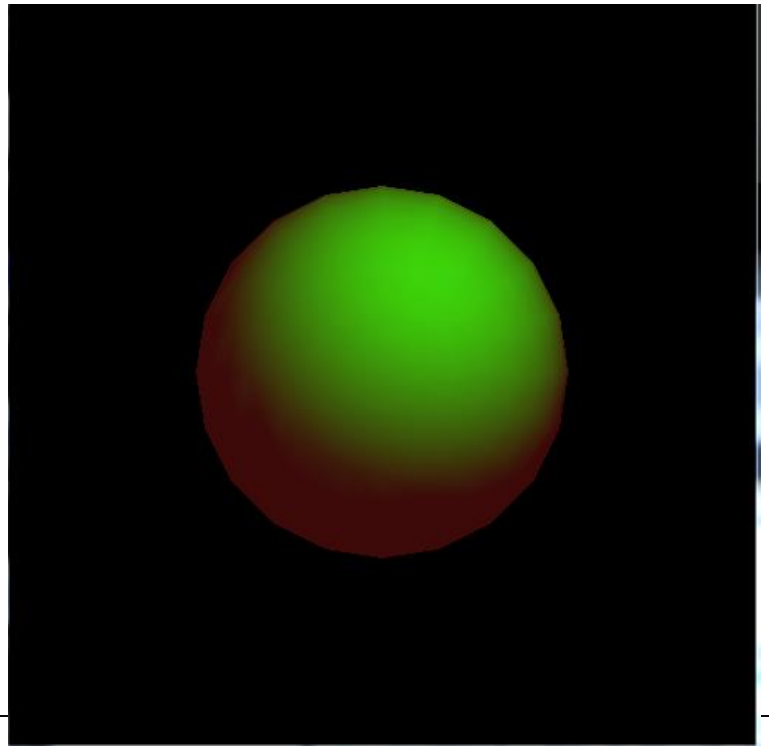
- 물체면 (material)에 대한 정보를 주지 않았으므로 default 값 사용
- Default로는 specular light이 (0.0, 0.0, 0.0, 1.0) 이다

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

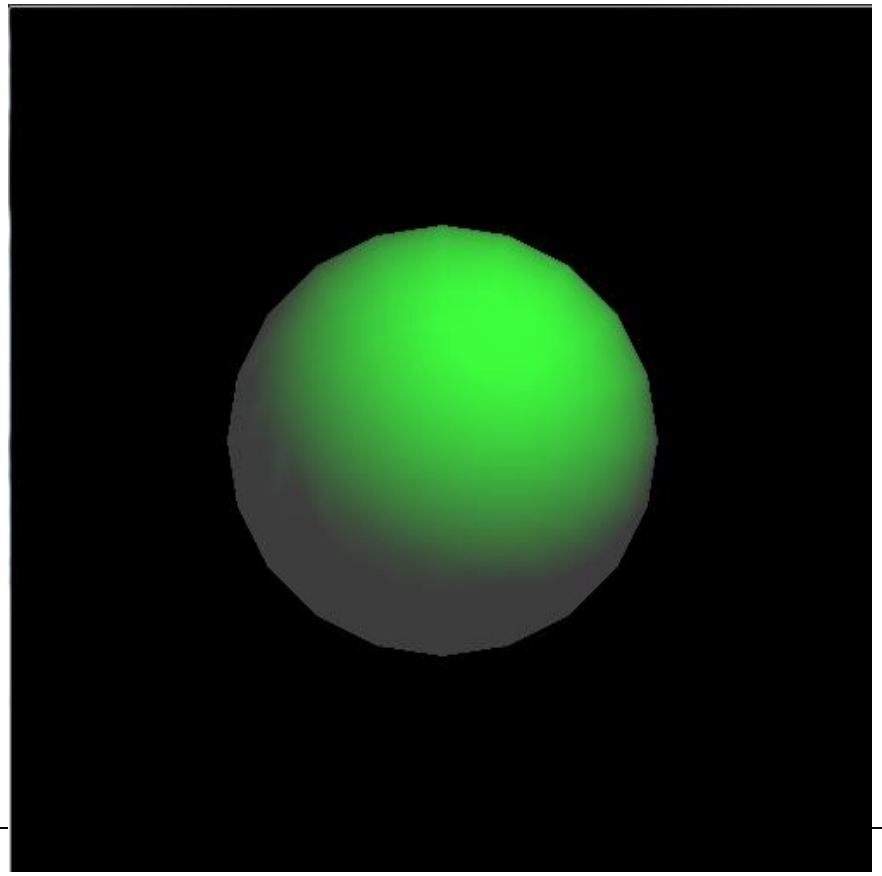
- 1) Now let’s modify the position of light source
- GLfloat MyLightPosition [ ] ={1.0, 2.0, 3.0, 1.0};
- => GLfloat MyLightPosition [ ] ={0.0, 0.0, 10.0, 1.0};



- 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**

- **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**

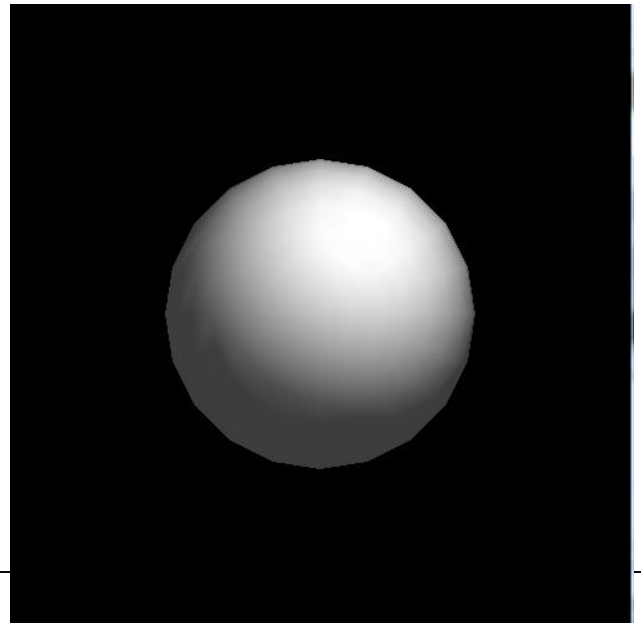
$$f_{attenuation} = \frac{1}{a + bD + cD^2}$$

- **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

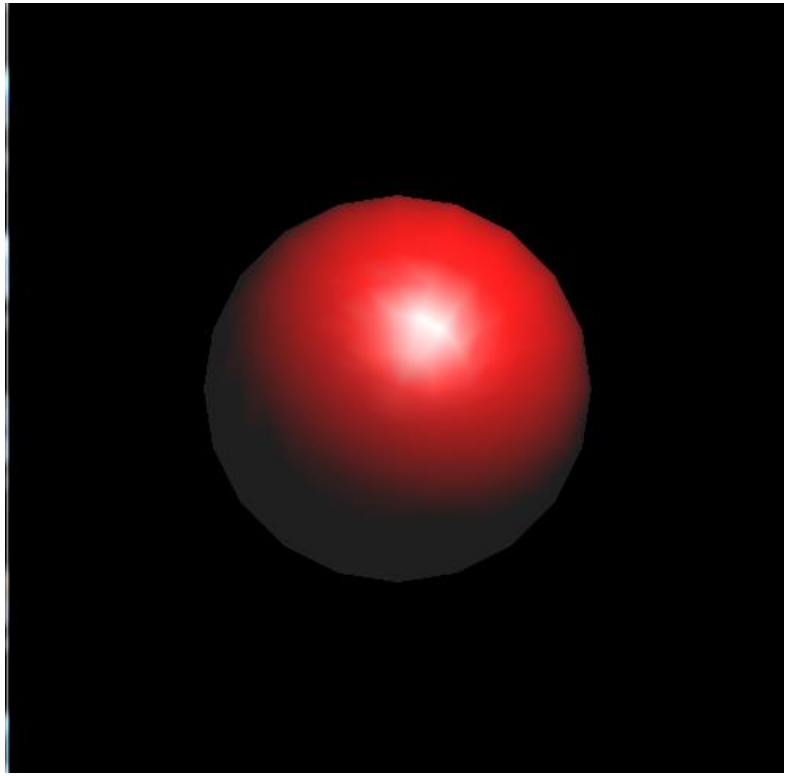
- Light color를 모두 white 로 하고 material의 특성을 변화시켜 보자
- 1. Ambient, diffuse, specular 모두 white로 바꿈
- GLfloat MyLightAmbient[ ] = {1.0, 1.0, 1.0, 1.0}; //ambient
- GLfloat MyLightDiffuse[ ] = {1.0, 1.0, 1.0, 1.0}; // diffuse
- GLfloat MyLightSpecular[ ] = {1.0, 1.0, 1.0, 1.0}; // specular



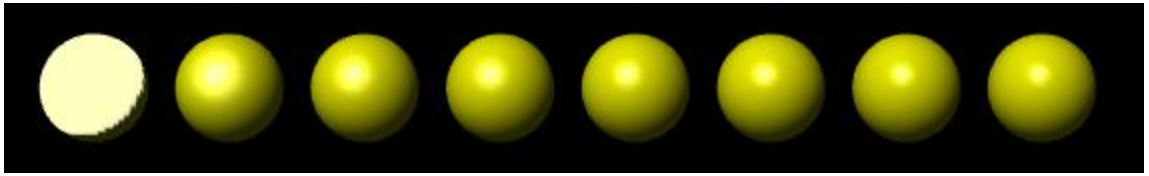
- 2. 물체의 반사 정도 결정 ambient, diffuse, specular, respectively
- shininess: 광택계수 (0-128사이의 값)
- GLfloat material_ambient[] = { 0.1, 0.1, 0.1, 1.0 }; // ambient (almost black)
- GLfloat material_diffuse[] = { 1.0, 0.0, 0.0, 1.0 }; // diffuse: red
- GLfloat material_specular[] = { 1.0, 1.0, 1.0, 1.0 }; //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

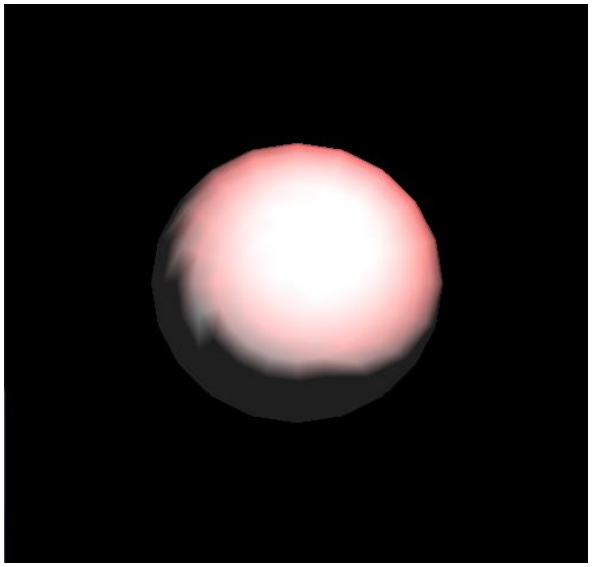
- **https://www.dropbox.com/s/v7z1xotx6lxqtx5/light_2.txt?dl=0**



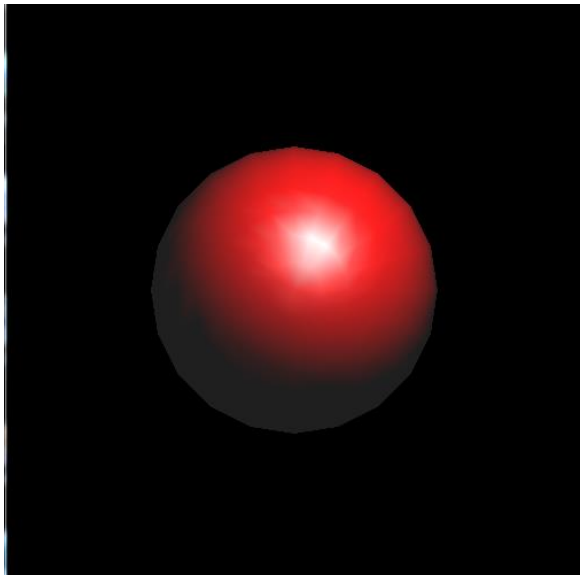
- 3. Shininess (SHARPNESS)조절
- (0-128까지 값가짐)
- shininess값 증가 (작고 날카로운 specular reflection)
- shininess값 감소 (반대로 넓고 덜 날카로운 specular reflection)
- shininess값을 줄여본다



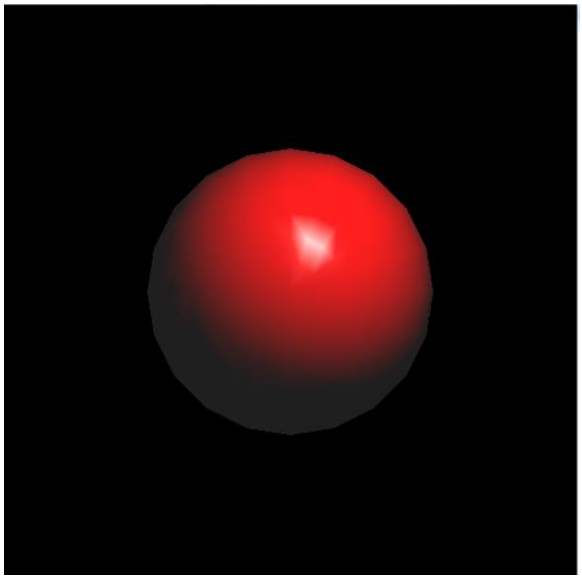
- **GLfloat material_shininess[] = { 1.0 };**



Shininess: 1



Shininess: 25

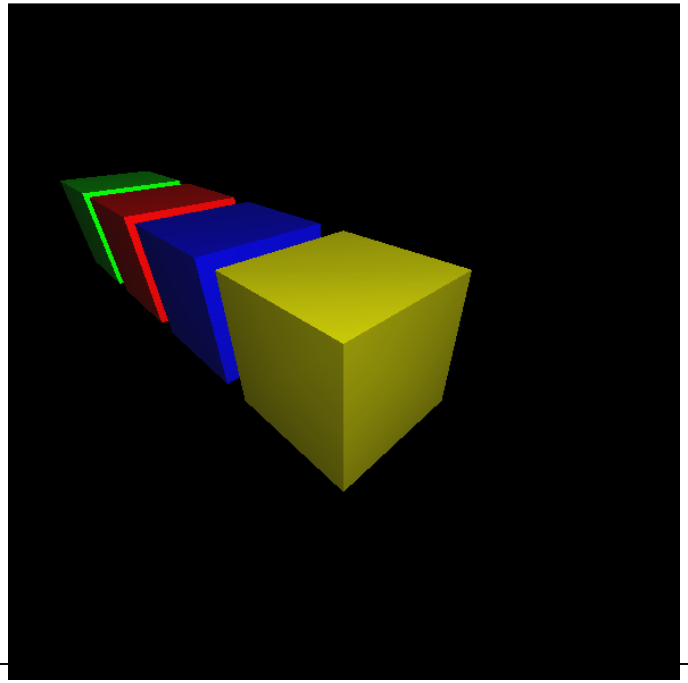


Shininess: 125

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

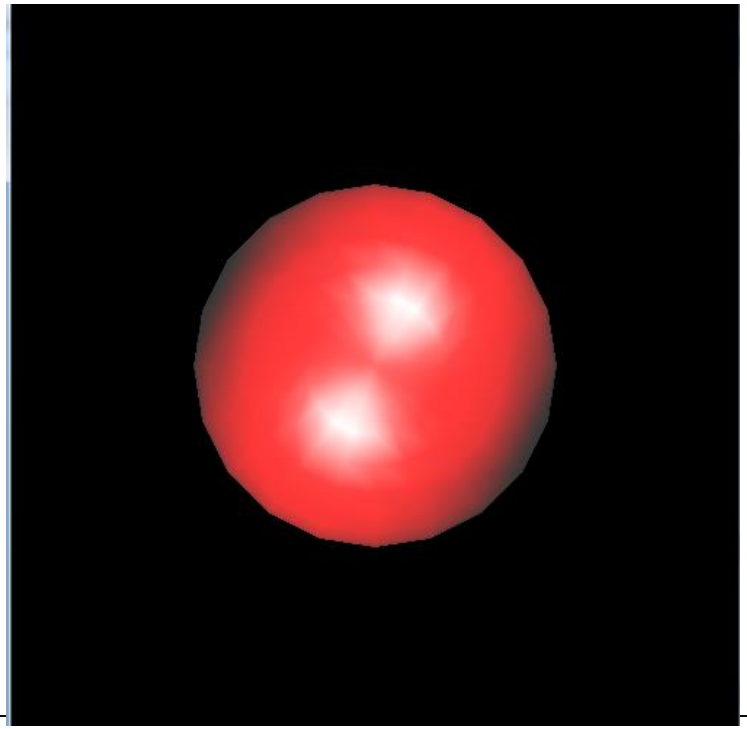
- **Summary: How to use light in OpenGL?**
- **1. Enable light (조명 기능 활성화)**
  - glEnable(GL_LIGHTING);
- **2. Light source의 종류 및 위치 정함**
  - 광원을 1개? 여러 개? glEnable(GL_LIGHT0); Positional? Directional?
  - glLightfv()
- **3. Light source의 color 정함**
  - Ambient, Diffuse, Specular 별로 color 부여
- **4. Material의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

- 하나의 광원과 4개의 material을 사용하였고 4개의 material에 서로 다른 lighting을 가하였다
- Viewing frustum과 gluLookAt() 함수를 사용하여 입체감 및 원근감을 주었다



- **https://www.dropbox.com/s/lfilk4wd5opsxdj/light_4.txt?dl=0**

- 예: Two light sources



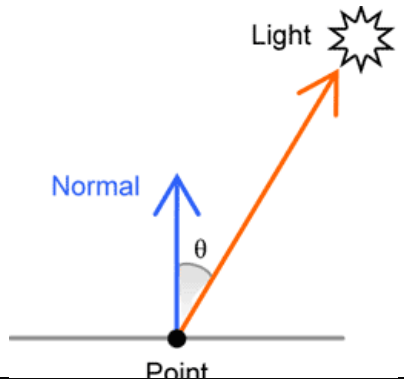
- Light source의 위치
- GLfloat light_position[] = { 0.0, -1, 1, 1.0 };

- Material 1 // diffuse: red
- glTranslatef(-1,0,0);
- GLfloat mat_diffuse [] = { 1.0, 0.0, 0.0, 1.0 };
- Material 2 // diffuse: green
- glTranslatef(1,0,0);
- GLfloat mat_diffuse1[] = { 0.0, 1.0, 0.0, 1.0 };

- **https://www.dropbox.com/s/gzg43445wsb5rxa/light_5.txt?dl=0**

- **GLUT objects and Lighting**

- OpenGL에서 Lighting은 vertex 단위로 동작한다
- Diffuse reflection의 경우 람베르트 법칙을 이용하여 그 vertex에서의 법선벡터 (N)와 광원벡터 (L)가 이루는 각 (θ)을 이용
- Diffusive reflection (확산광)의 세기 =$K_d I_d$ / D^2max$\left(\frac{N \cdot L}{|N||L|}, 0\right)$
- 그러면 각 vertex에서 normal vector를 주어야 정확한 lighting 계산이 되는데 지금까지 OpenGL 코드에서는 normal vector를 주지 않았다. 이유는 무엇일까?

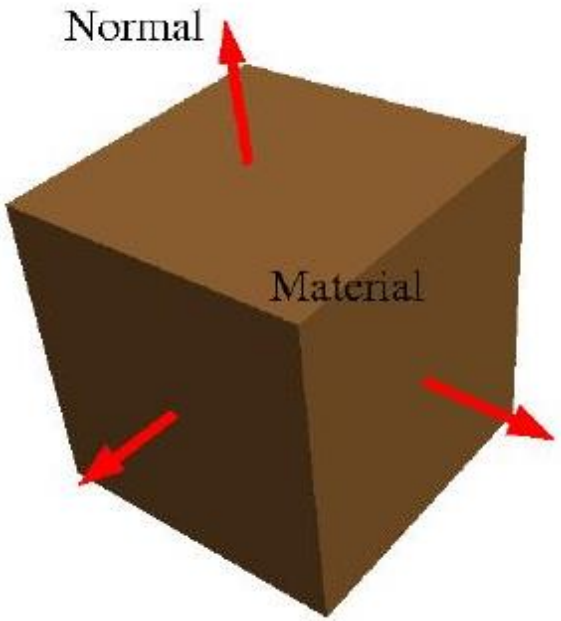


- OpenGL에서는 Lighting 사용시 원래 각 vertex 별로 normal vector를 정의 해야만 정확하게 lighting을 사용 가능 하지만
- 지금 까지 OpenGL에서 사용한 GLUT object는 각 vertex에서의 normal vector 계산이 내부적으로 이미 되어 있으므로 vertex 별로 normal vector를 정의할 필요가 없다
- GLUT objects가 아닌 다른 polygon을 사용자가 직접 정의시 normal vector를 직접 정해야 한다

- **Normal vector in OpenGL**

- glutObject가 아닌 polygon의 경우 programmer가 직접 normal vector를 glNormal3f()로 설정해 줘야 한다
- OpenGL normal vector 설정 함수
- glNormal3f(x, y, z);

- 곡면이 아닌 같은 평면에 있는 vertex들은 하나의 normal vector로, 즉, 하나의 glNormal3f()로 표현 가능 하다
- 아래 cube와 같은 closed surface에서는 면당 하나씩 정의



- 한 평면 위에 있는 vertex4개로 이루어진 polygon (quad)의 normal vector 설정 예

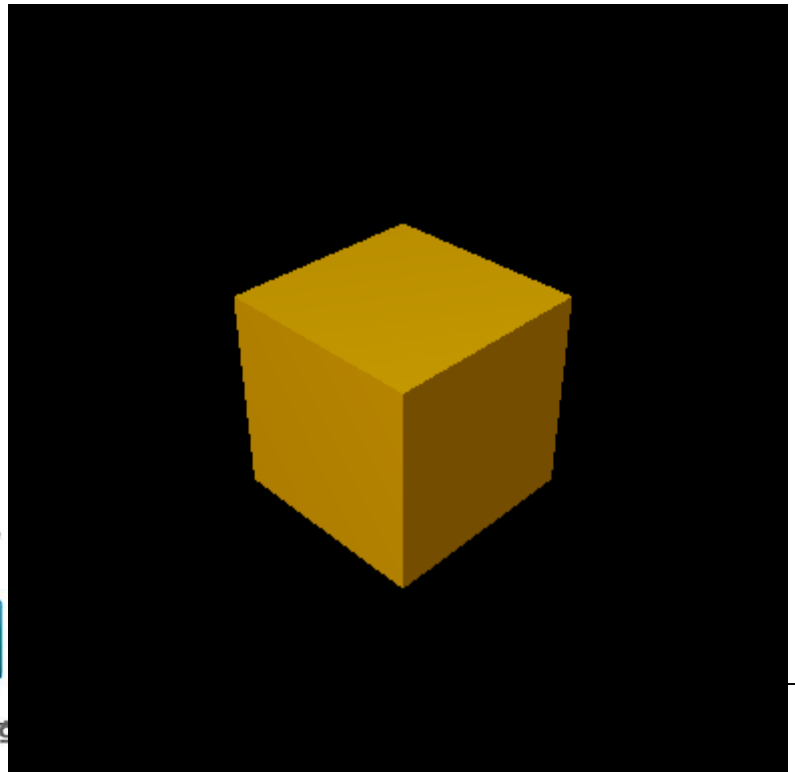
```
glBegin(GL_QUADS);
 glNormal3f(0.0f, 0.0f, 1.0f);
 glVertex3f(1.0f, 1.0f, 1.0f);
 glVertex3f(-1.0f, 1.0f, 1.0f);
 glVertex3f(-1.0f, -1.0f, 1.0f);
 glVertex3f(1.0f, -1.0f, 1.0f);
```

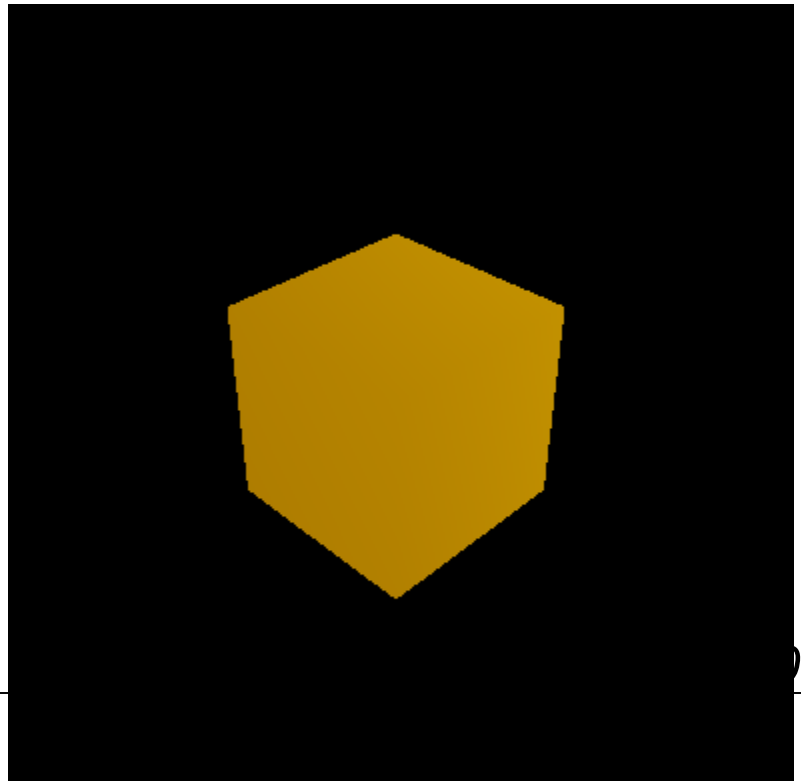
이 vertex4개로 이루어 지는 면의 normal vector를 계산해 보자

위의 glNormal3f(0.0, 0.0, 1.0) 이 맞는지 확인해 보자

- glNormal3f로 정의된 normal vector에 대해서 vector의 magnitude를 1로 자동으로 normalize하고 싶다면 OpenGL의 다음의 함수 사용 가능
- glEnable(GL_NORMALIZE);
- // normalized된 normal vector를 자동으로 계산

- Cube 모양의 polygon들을 2가지 방법으로 rendering 해봄
- (Left) using glNormal3f()
- (Right) without using glNormal3f()





- **https://www.dropbox.com/s/6mkldotu5ff8au4/normalvector.txt?dl=0**