

그래픽스 강의노트 06 - 조명 1

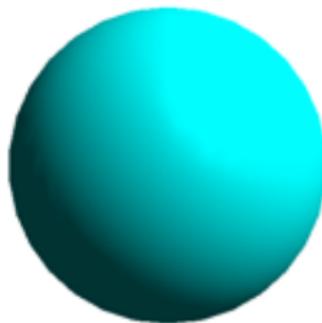
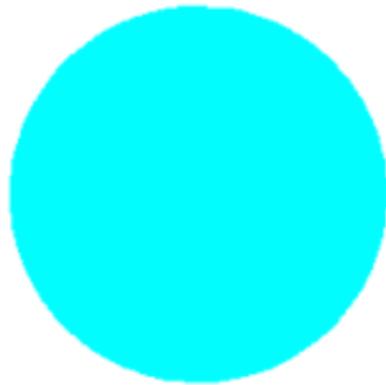
강영민

동명대학교

2015년 2학기

음영 계산의 필요성

- 음영(陰影) 계산, 혹은 셰이딩(shading)은 어떤 물체의 표면에서 어두운 부분과 밝은 부분을 서로 다른 밝기로 그려내는 것
- 모든 면을 동일한 색으로 그리면 입체감이 없다.



조명과 재질

음영 계산에 필요한 두 가지 핵심 요소

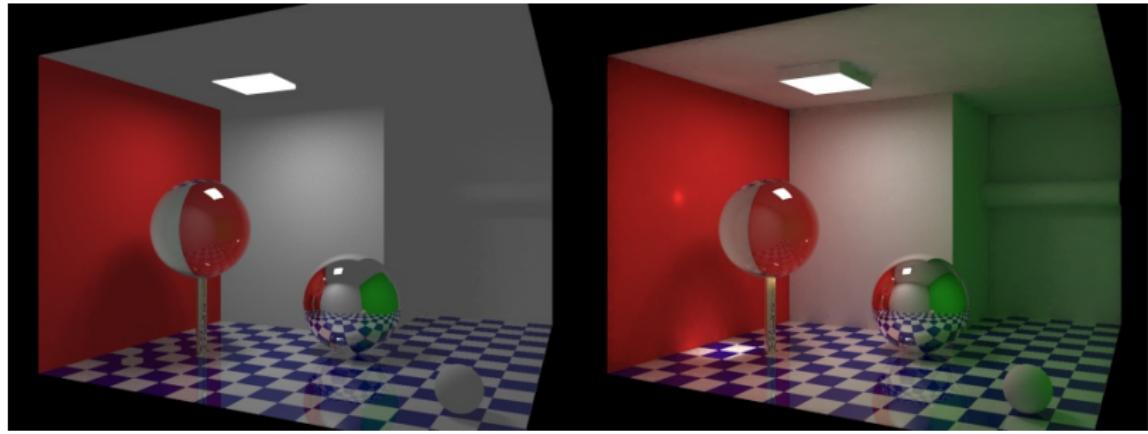
- 빛

- 컴퓨터 그래픽스의 최종 결과는 우리의 눈을 통해 입력되는 시각 정보
- 눈이 인지하는 신호는 전자기파
- 우리가 인지할 수 있는 영역의 전자기파: 가시광선(可視光線)
- 이 가시광선을 일상적으로 ‘빛’이라 부름

- 재질(材質,material)

- 같은 빛이라고 모든 물체가 같은 색으로 보이지는 않음
- 물체는 저마다의 특성에 따라 서로 다른 방식으로 빛을 반사
- 물체가 가진 반사 특성을 재질이라고 함

지역조명과 전역조명



광원 모델

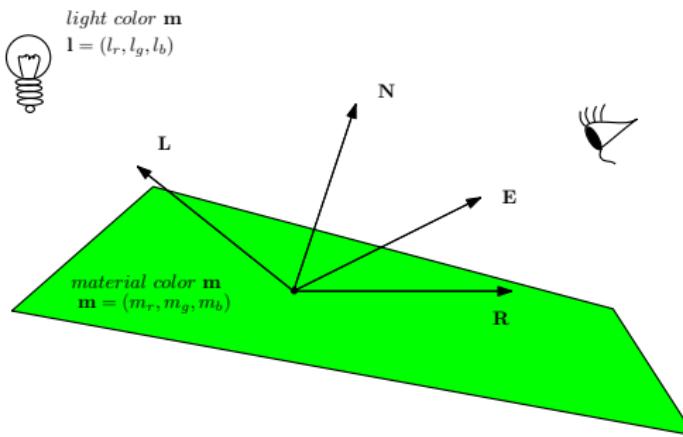
일반적인 광원(光源, light source)은 다루기가 쉽지 않다. 하나의 광원은 일정한 면적이나 체적을 가지기 때문에 이 광원에서 나온 빛들을 적분해야 한다. 실제 실시간 렌더링에서는 광원을 하나의 점이나 방향으로 보는 단순화된 모델을 사용한다. 사용되는 광원은 다음과 같은 것들이 있다.

광원 종류	특징
점광원	광원의 위치와 색으로 결정. 전방향(omnidirection)으로 빛 진행.
집중광원	점광원에서 일부 입체각으로 빛을 제한.
방향광원	특정한 위치가 아니라 방향에 광원 존재.
주변광원	모든 곳에 동일하게 가해지는 빛.

지역 조명 모델 - 풍(Phong) 모델

정반사와 난반사, 주변광 반사 특성을 표현할 수 있는 간단하고 빠른 모델
다음과 같은 정보가 필요

- 광원으로 향하는 벡터 **L**
- 카메라(시점)을 향하는 벡터 **E**
- 법선 벡터 **N**
- 이상적인 반사 방향 **R**



퐁(Phong) 모델: 광원과 재질 - 색상

- 퐁 모델은 난반사, 정반사, 주변광을 각각 독립적으로 계산하여 합성
- 각각의 반사 요소에 따라 결정해야 하는 것은 눈을 향해 오는 빛의 강도(intensity)와 색상
- 색상의 결정
 - 어떤 광원에서 나오는 빛의 색상이 $\mathbf{l} = (l_r, l_g, l_b)$
 - 물체의 재질 색상이 $\mathbf{m} = (m_r, m_g, m_b)$
 - 빛의 색상은 이 빛이 눈에 감지되었을 때 우리가 느끼는 색이
 - 빨간 색을 가진 물체의 재질 색상 $(1.0, 0.0, 0.0)$ 은 도착하는 빛의 RGB 3 개 채널에서 R 채널의 빛을 100% 반사한다는 것을 의미

반사되는 빛의 색상은 다음과 같다.

$$\mathbf{c} = (l_r m_r, l_g m_g, l_b m_b)$$

이를 이렇게 표현한다.

$$\mathbf{c} = \mathbf{l} \otimes \mathbf{m}$$

퐁(Phong) 모델: 광원과 재질 - 광강도

- 이 계산은 눈에 감지되는 빛의 색상을 결정
- 같은 색상이라도 밝고 어두울 수 있음
- 이러한 음영을 고려해야 입체적인 물체로 보임
- 이러한 음영은 광강도(光強度, light intensity)에 의해 결정
- 이 광강도의 값은 스칼라(scalar) 값으로 I 로 표현
- 실제로 눈에 보이는 색은 광원과 재질에 의해 결정되는 색상 \mathbf{c} 와 광강도 I 에 의해 다음과 같이 결정

$$\kappa = I\mathbf{c}$$

퐁(Phong) 모델: 최종 결정 반사색

- 퐁 모델은 각각의 반사 요소를 모두 따로 계산
- 난반사 요소와 관련된 값에는 아래첨자 d
- 정반사에는 s
- 주변광 반사에는 a
- 눈에 관측되는 색상은 다음과 같음

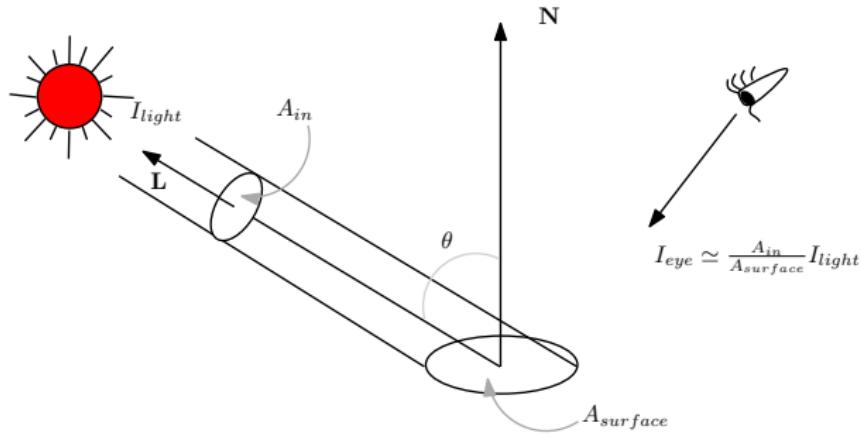
$$\kappa = I_a \mathbf{c}_a + I_d \mathbf{c}_d + I_s \mathbf{c}_s$$

이때, 주변광의 광강도는 상수 값을 가지므로 I_a 는 1로 설정할 수 있다.
따라서 다음 식으로 바꿀 수 있다.

$$\begin{aligned}\kappa &= \mathbf{c}_a + I_d \mathbf{c}_d + I_s \mathbf{c}_s \\ &= \mathbf{l}_a \otimes \mathbf{m}_a + I_d \mathbf{l}_d \otimes \mathbf{m}_d + I_s \mathbf{l}_s \otimes \mathbf{m}_s\end{aligned}$$

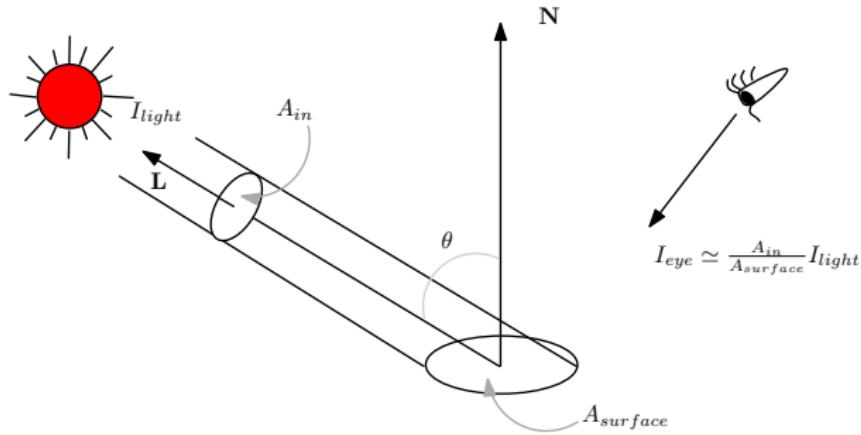
퐁 모델 광강도(intensity)의 계산 - 난반사

- 퐁 모델에서 계산해야 하는 광강도는 난반사 I_d 와 정반사 I_s
- 난반사는 모든 방향에 동등하게 빛이 퍼지는 것으로 가정
- 눈이 어디에 있든지 동일한 색상 관찰
- 눈의 움직임에 따라 변하는 하일라이트(highlight)는 표현하지 못 함
- 색을 칠하려고 하는 한 지점에 대해 어디서 쳐다 보든지 동일한 밝기
- 밝기는 얼마나 많은 에너지가 해당 지점에 떨어지는지에 달려 있음



퐁 모델 광강도(intensity)의 계산 - 난반사 1/2

- 퐁 모델에서 계산해야 하는 광강도는 난반사 I_d 와 정반사 I_s
- 난반사는 모든 방향에 동등하게 빛이 퍼지는 것으로 가정
- 눈이 어디에 있든지 동일한 색상 관찰
- 눈의 움직임에 따라 변하는 하일라이트(highlight)는 표현하지 못 함
- 색을 칠하려고 하는 한 지점에 대해 어디서 쳐다 보든지 동일한 밝기
- 밝기는 얼마나 많은 에너지가 해당 지점에 떨어지는지에 달려 있음

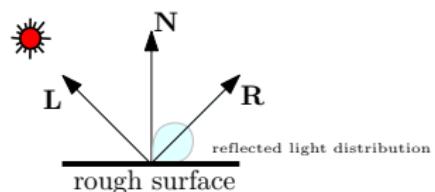
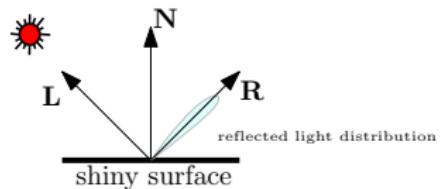
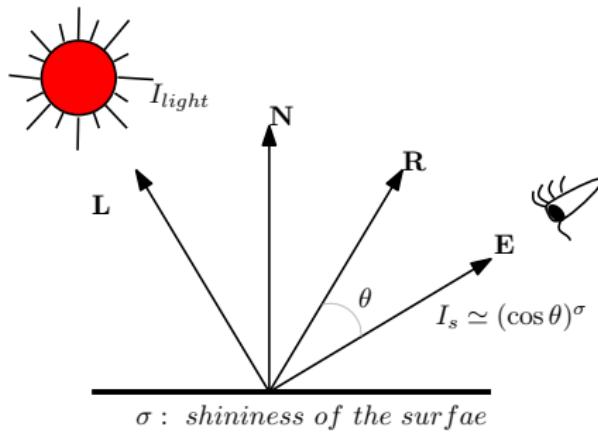


퐁 모델 광강도(intensity)의 계산 - 난반사 2/2

- 이 값은 광원 벡터 \mathbf{L} 과 법선 벡터 \mathbf{N} 이 일치할 때 최대이며, 90도를 이룰 때 0이 됨
- 이 값은 두 벡터의 내적, 즉 두 벡터 사잇각의 코사인(cosine)에 비례한다는 것이 램버트 반사(Lambertian reflectance) 모델
- 따라서 광강도를 계산해야하는 지점에서 빛을 향하는 방향벡터 \mathbf{L} 과 표면 법선벡터 \mathbf{N} 의 내적으로 I_d 를 구할 수 있음

$$I_d = \cos \theta = \mathbf{L} \cdot \mathbf{N} \quad (1)$$

퐁 모델 광강도(intensity)의 계산 - 정반사 1/2



정반사는 거울과 같이 입사각에 대칭되는 방향으로 반사되는 것이다. 그런데, 실제 물체들은 이런 이상적인 정반사가 아니라 반사 방향으로 빛이 강하게 진행하기는 하지만 다른 방향으로 조금씩 빛이 나간다. 퐁 모델에서 사용하는 정반사 모델은 거울과 같은 반사가 아니라 반사 벡터 R 중심으로 퍼지는 반사를 표현한다.

퐁 모델 광강도(intensity)의 계산 - 정반사 2/2

- 정반사는 반사 벡터 \mathbf{R} 근처에서 강하게 관찰되기 때문에 눈을 \mathbf{R} 근처로 가져가야 강한 반사
- 정반사의 광강도 I_s 는 \mathbf{R} 과 \mathbf{E} 의 사잇각의 코사인에 연관
- 물체의 재질에 따라 \mathbf{R} 방향으로 집중되는 정도가 달라짐 - 물체의 반질함(shininess) σ 에 의해 결정

$$I_d = \cos \theta = (\mathbf{R} \cdot \mathbf{E})^\sigma$$

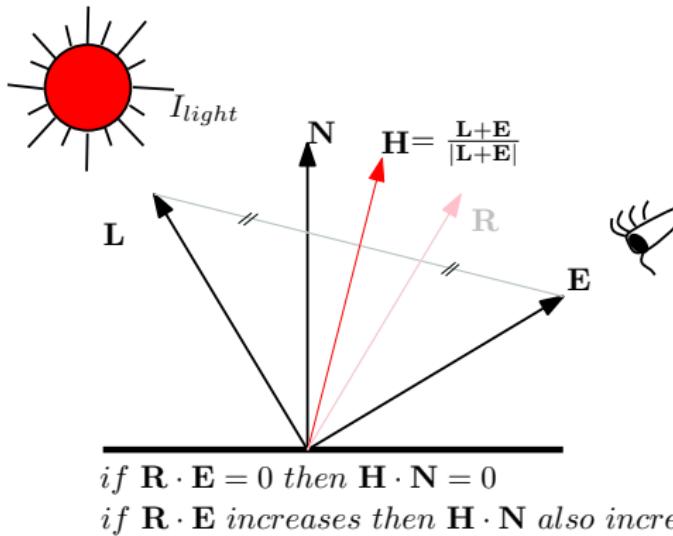
퐁 모델 광강도(intensity)의 계산 - 주변광

주변광의 광강도는 언제나 1이다. 따라서 모든 물체는 주변광과 주변광 반사 재질에 의한 색상 $\mathbf{l}_a \otimes \mathbf{m}_a$ 을 기본적으로 갖게 된다. 이것은 지역조명 기법의 단점인 지나치게 어두운 부분을 없애는 역할을 수행한다.

$$\begin{aligned}\kappa &= \mathbf{c}_a + I_d \mathbf{c}_d + I_s \mathbf{c}_s \\ &= \mathbf{l}_a \otimes \mathbf{m}_a + I_d \mathbf{l}_d \otimes \mathbf{m}_d + I_s \mathbf{l}_s \otimes \mathbf{m}_s\end{aligned}$$

수정된 풍 모델 - 블린(Blinn) 모델

- 풍 모델은 정반사 광강도 계산에서 반사 벡터 \mathbf{R} 을 계산해야 함
- 블린(Blinn)은 이 반사벡터 대신에 반 벡터(halfway vector)라는 것을 도입
- 반 벡터 \mathbf{H} 는 광원 벡터와 시선 벡터 \mathbf{E} 를 더해서 정규화: $\mathbf{H} = \frac{\mathbf{L}+\mathbf{E}}{|\mathbf{L}+\mathbf{E}|}$



OpenGL 조명 - 조명과 재질 적용할 데이터 준비

- 조명의 위치는 동차좌표(homogeneous coordinate)로 표현
- 마지막 성분이 1이면 점광원이고, 0이면 방향광원(directional light source)

```
// 재질의 정반사, 난반사, 주변광, 반질거림 특성으로 사용될 데이터
// values for material specification
GLfloat mat_specular[] = { 1.0, 1.0f, 1.0f, 1.0f };
GLfloat mat_diffuse[] = { 0.0, 1.0f, 1.0f, 1.0f };
GLfloat mat_ambient[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 120.0 };

// 광원의 정반사, 난반사, 주변광 특성으로 사용될 데이터
// values for light specification
GLfloat lit_specular[] = { 1.0, 1.0f, 1.0f, 1.0f };
GLfloat lit_diffuse[] = { 0.0, 1.0f, 1.0f, 1.0f };
GLfloat lit_ambient[] = { 0.5, 0.0f, 0.0f, 1.0f };

// 광원의 위치로 사용될 데이터
// values for light positioning
GLfloat light_position[] = { 1.0, 1.0f, 1.0f, 0.0f };
```

OpenGL 조명 - 조명과 재질 적용

실제 조명과 재질에 적용

- 조명의 특성을 설정하는 함수는 `glLight*`
- 재질 특성을 설정하는 것은 `glMaterial*`

```
// 조명과 재질의 특성을 준비된 데이터로 설정하는 함수
void LightSet( void ) {
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

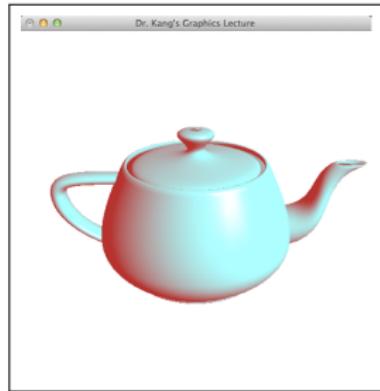
    glLightfv(GL_LIGHT0, GL_SPECULAR, lit_specular);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lit_diffuse);
    glLightfv(GL_LIGHT0, GL_AMBIENT, lit_ambient);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
}

// 조명의 위치를 설정하는 함수
void LightPositioning( void ) {
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
}
```

OpenGL 조명 - 렌더링

```
void init( int argc , char **argv ) {
    [[각종 초기화 작업]]
    LightSet(); // 조명의 특성과 재질의 특성을 설정한다
    glEnable(GL_DEPTH_TEST);
}

void display() {
    [[GL_MODELVIEW 모드 설정]]
    gluLookAt(0,1,2,0,0,0,0,1,0);
    LightPositioning(); // 점광원의 위치를 설정한다
    glutSolidTeapot(0.5);
    glutSwapBuffers();
}
```



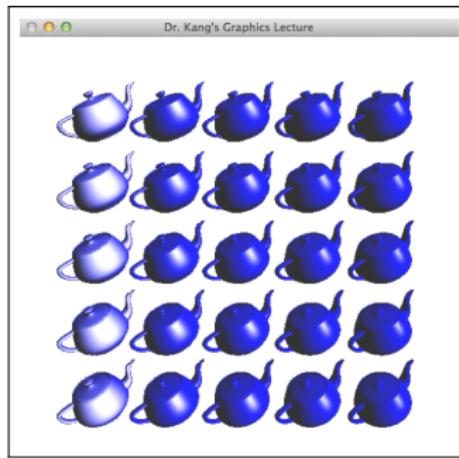
(a) 주변광 포함



(b) 주변광 제외

OpenGL 조명 - 반질거림 조정

```
for ( int i=0; i < 5; i++ ) {
    // 재질의 반질거림을 변경하여 설정하고 주전자를 그림
    glMaterialf(GL_FRONT, GL_SHININESS, 4.0f + 123.0f * ( i / 4.0 ) );
    for ( int j=0; j < 5; j++ ) {
        glPushMatrix();
        glTranslated( float(i) + 0.5, float(j) + 0.5, 0.0 );
        glRotated( 45, 1, 1, 1 );
        glutSolidTeapot( 0.4 );
        glPopMatrix();
    }
}
```



OpenGL 조명 - 점광원 구현 1/2

```
[[광원과 재질의 색상 설정]]
// 광원의 위치를 설정한다.
// 광원의 좌표 가운데 w 성분이 1로 설정되어 있다
// 디스플레이 콜백에서 광원의 위치를 바꾸지만, w 좌표는 변경하지 않는다
GLfloat lit_position[] = { 0.0f, 0.0f, 0.0f, 1.0f }; // light position

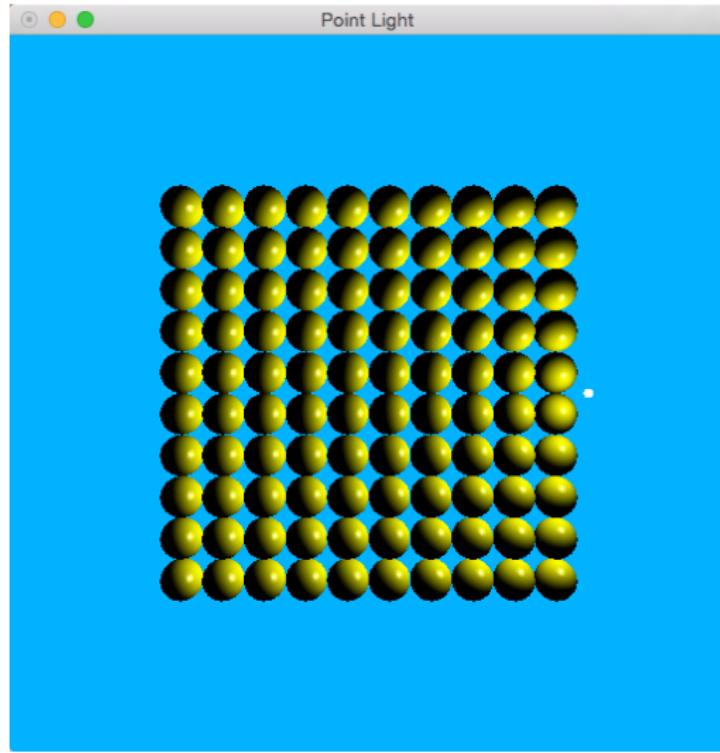
void SetLight() { 이전 코드와 동일 }

void init(void) {
    [[윈도우, 버퍼 초기화, 카메라 설정 등의 초기 작업]]
    SetLight();
}

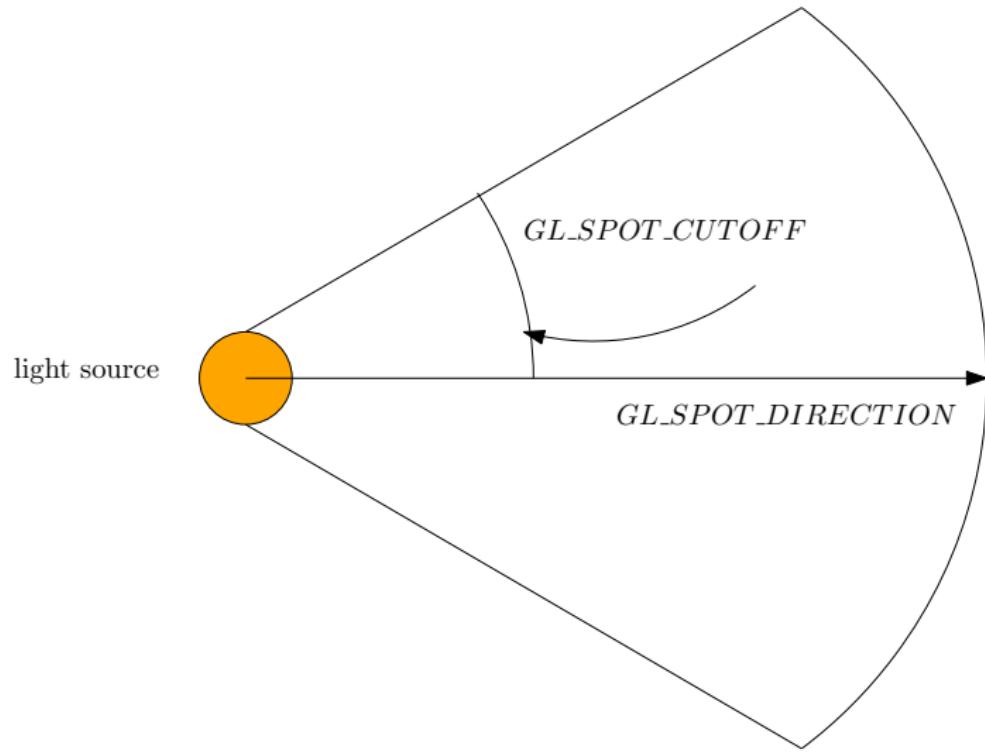
void display() {
    // [[버퍼 지우기, 모델뷰 행렬 모드 설정, 카메라 위치 설정 등 수행]]
    static float t = 0.0; t+=0.01;
    // 광원의 위치를 설정함. w 좌표는 1로 고정하고 회전하도록 함
    lit_position[0] = 5.0*sin(t); lit_position[2] = 5.0*cos(t);
    glLightfv(GL_LIGHT0, GL_POSITION, lit_position);

    for(int i=0;i<10;i++) {
        for(int j=0;j<10;j++) {
            glPushMatrix();
            glTranslatef(i-4.5,j-4.5,0);
            glutSolidSphere(0.5, 30, 30);
            glPopMatrix();
        }
    }
    // [[광원의 위치 등을 표시하여 확인할 수 있도록 함]]
    glutSwapBuffers();
}
```

OpenGL 조명 - 점광원 구현 2/2



OpenGL 조명 - 집중광원에 필요한 요소



OpenGL 조명 - 집중광원 1/2

```
// 집중 광원의 방향으로 사용될 데이터를 준비한다
GLfloat spotDir[] = { 0.0f, 0.0f, -1.0f };

void SetLight() {
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    // set material properties
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

    // set light properties
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lit_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, lit_specular);
    glLightfv(GL_LIGHT0, GL_AMBIENT, lit_ambient);

    // 집중광원에 필요한 데이터를 설정한다
    glLightf (GL_LIGHT0,GL_SPOT_CUTOFF,20.0f);
    glLightfv (GL_LIGHT0,GL_SPOT_DIRECTION,spotDir);
    // Exponent 집중광의 테두리를 부드럽게 한다. 자세한 내용은 Reference API 등을 참고하라
    glLightf (GL_LIGHT0,GL_SPOT_EXPONENT, 20.0f);
}
```

OpenGL 조명 - 집중광원 2/2

