OpenGL12

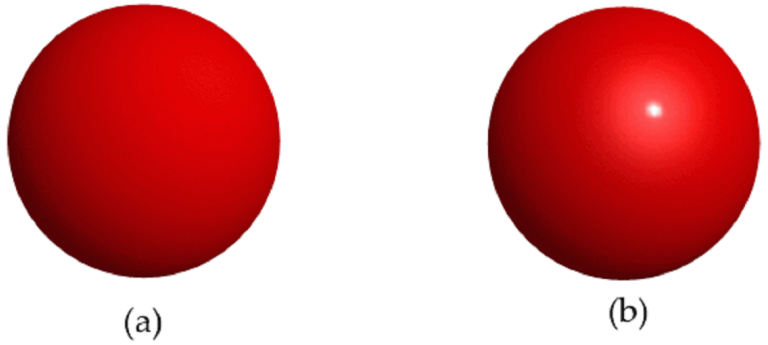
가상현실론

2021/04/01

목차

- 경면 반사
- 음영
- 광원

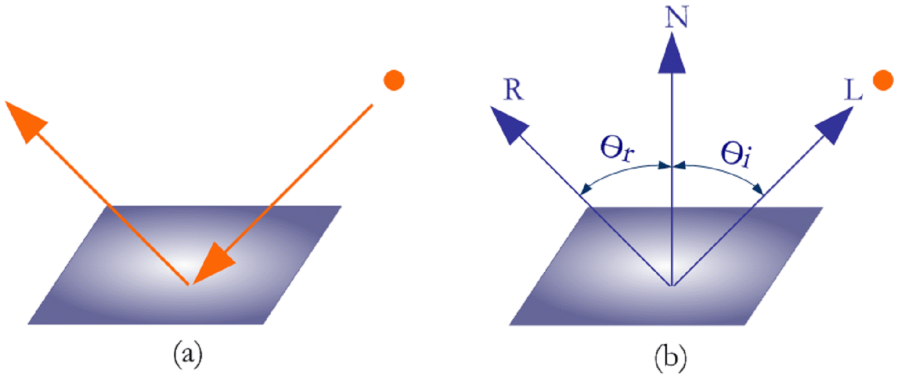
경면 반사



확산 반사 확산 반사+경면 반사

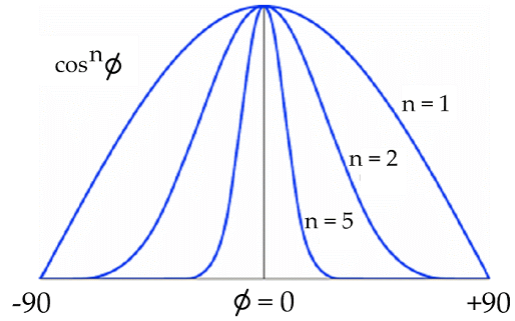
- 경면반사 (Specular Reflection): 특정한 방향에서 바라볼 때 거울면처럼 반사되는 반사모델. 경면광에 의해 물체면에 형성된 반짝이는 이미지를 하이라이트(highlight)라고 함.
- 경면광에 의한 물체면의 색은 물체 자체의 색이 아니라 광원에서 나오는 빛의 색
- (b)의 하이라이트 색이 물체색인 적색이 아니라 광원의 색인 백색

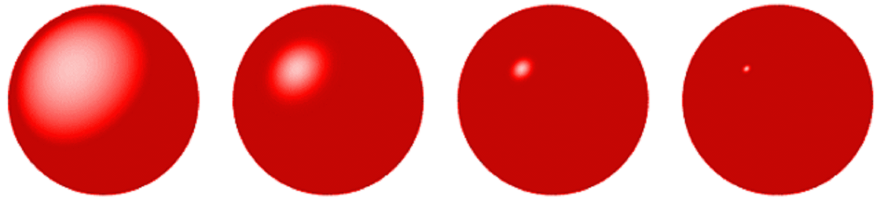
경면 반사



- 경면반사는 빛의 정반사에 의한 것.
- (a)와 같이 법선 벡터를 중심으로 입사광과 반사광이 정확히 대칭으로 진행.
- 법선 벡터 N이 광원 벡터 L과 이루는 각을 입사각(Incident Angle) θ_i라 하고, N이 경면 반사광 R과 이루는 각을 반사각(Reflection Angle) θ_r 이라 하면 $\theta_i = \theta_r$.
- 완벽한 경면에서 반사광 R은 광원 벡터 L, 수직벡터 N과 동일한 평면에 존재.

경면 반사





- 퐁 반사 모델 (Phong Reflection Model): 반사광 R과 시점벡터 V가 이루는 각을 ϕ라고 할 때 시점으로 들어오는 경면 반사의 양을 cos ϕ으로 간주.
- 퐁 반사 모델에서는 cos ϕ에 가해지는 승수 n에 의해 물체면의 매끄러운 정도를 반영. n을 광택 계수(Shineness Coefficient)
- 경면광의 세기

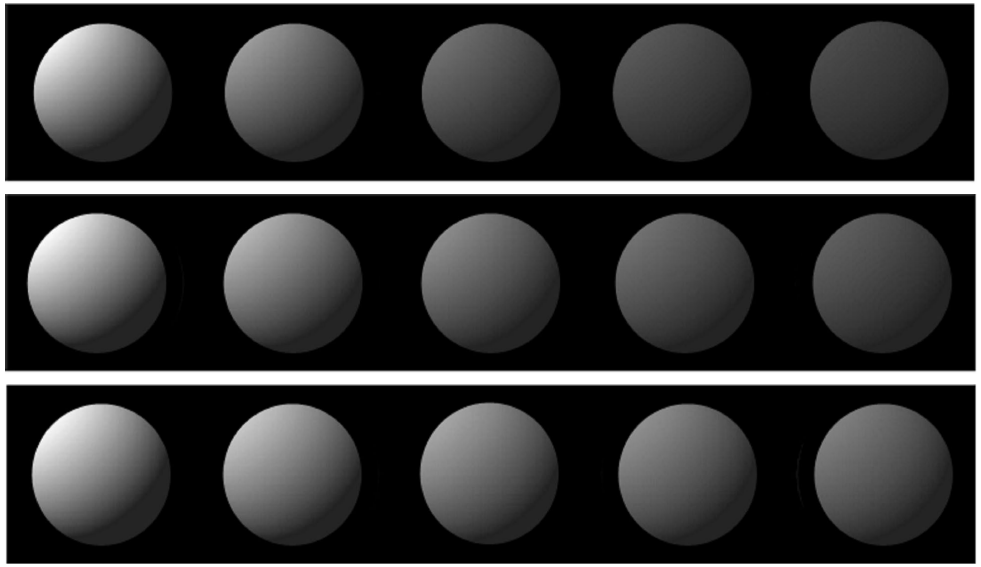
$$Specular\ \ Reflection = Ks\ \ Is\ \ (cos\phi)^n\ \ /\ \ D^2$$

$$= Ks\ \ Is\ \ (R \bullet\ \ V)^n\ \ /\ \ D^2$$

K_S : 경면 계수 (Reflective Coefficient)

I_S : 광원의 경면광 세기

반사광의 합성



a=b=0, c=1

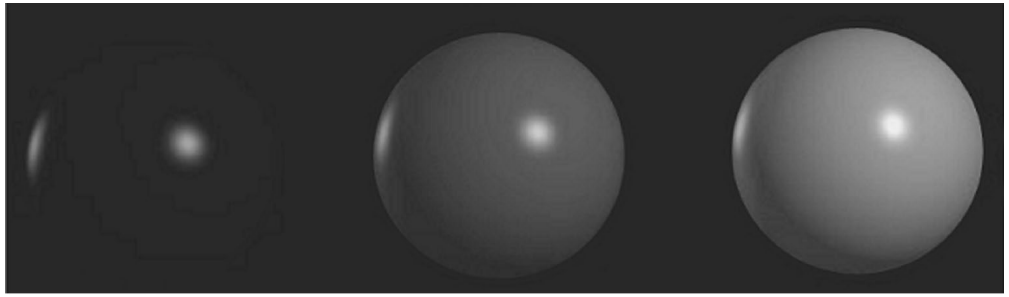
a=b=.25, c=.5

a=c=0 b=1

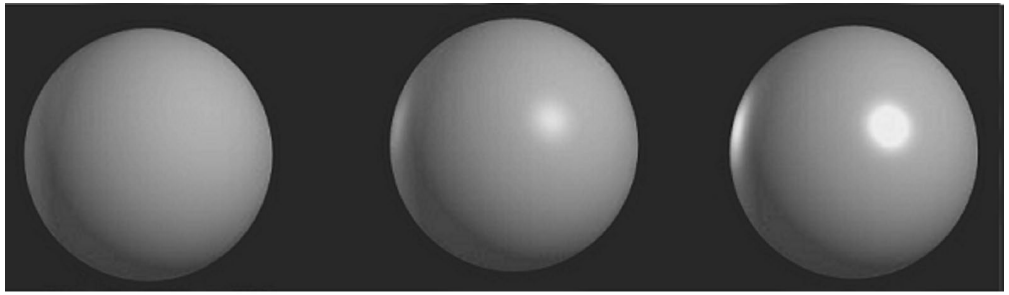
- GL에서 거리에 따른 빛의 세기

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

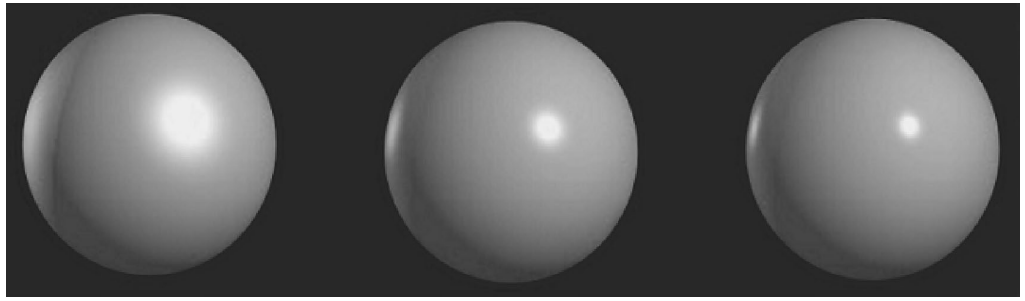
반사광의 합성



확산계수 0.01, 0.3, 0.7

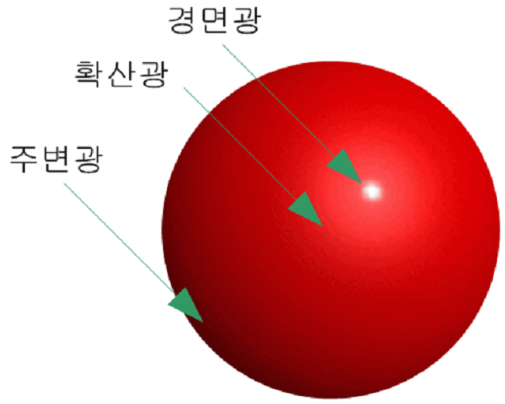


경면계수 0.0, 0.4, 0.8



광택 계수 5, 40, 100

반사광의 합성



- 지역반사모델: 주변 반사+ 확산 반사+경면 반사

$$I = Ambient\ \ Reflection + Diffuse\ \ Reflection + Specular\ \ Reflection$$

$$= \frac{1}{a + bD + cD^2}\ (Ka\ \ Ia + Kd\ \ Id\ \ (N \bullet\ L) + Ks\ \ Is\ (R \bullet\ V)^n)$$

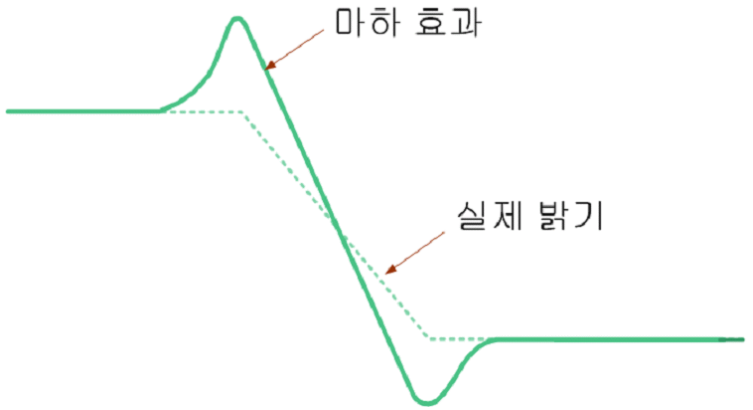
음영

- 음영(Shading): 계산된 정점의 색상으로부터 다각형 내부의 색을 칠하는 작업. 표면 렌더링(Surface Rendering)이라고도 함.

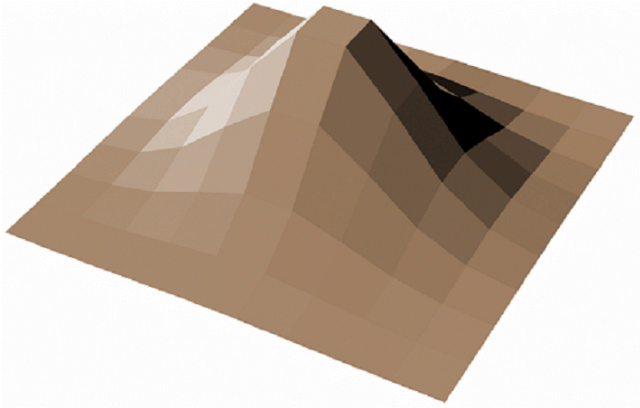
- 플랫 셰이딩(Flat Shading)
  - 주어진 하나의 다각형 전체를 동일한 색으로 칠함. 빠르고 간단
  - 상수 셰이딩(Constant Shading),깎은 면 셰이딩(Facet Shading)
  - 다각형을 구성하는 다각형 정점의 위치를 평균하여 중심점(Centroid)를 구함.중심점에서의 법선벡터, 광원벡터, 시점벡터를 기준으로 조명모델이 가해지며 그 결과 색이 면 내부를 모두 채움.

플랫 셰이딩

마하밴드 효과



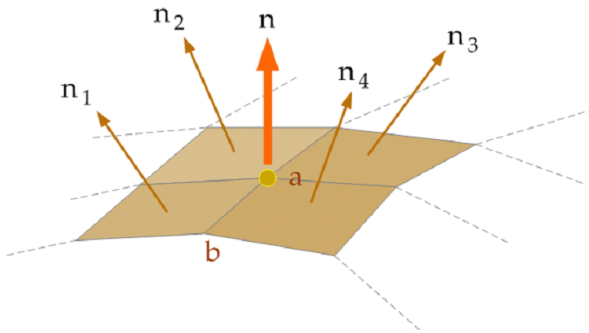
인식된 밝기



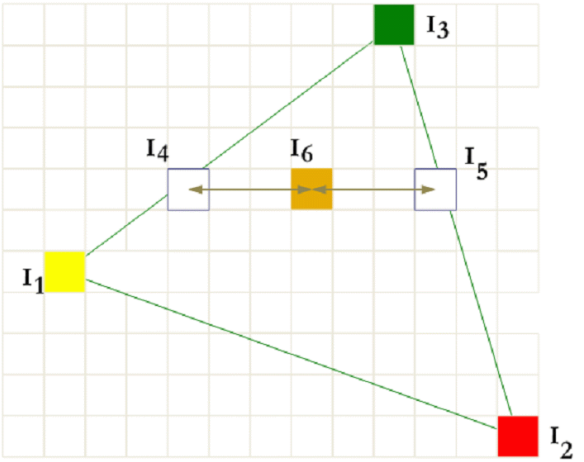
마하밴드

- 플랫셰이딩의 문제점
  - 다각형 사이의 경계면이 필요 이상으로 뚜렷하게 보임-> 마하밴드 효과 (Mach Band Effect)
  - 두 개의 면이 만나는 경계선 부근에서 어두운 면은 더욱 어두워지고 밝은 면은 더욱 밝게 보이는 착시효과

구로(Gouraund Shading) 셰이딩



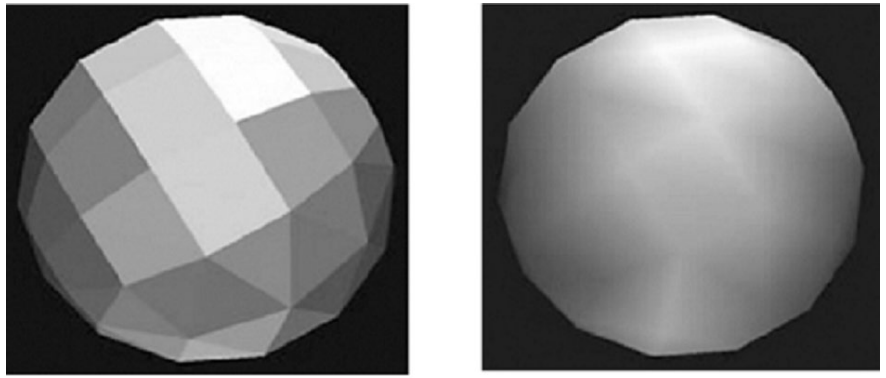
$$n = \frac{n_1 + n_2 + n_3 + n_4}{|n_1 + n_2 + n_3 + n_4|}$$



- 다각형 내부를 서로 다른 색으로 채우는 방법
- 정점의 색을 보간
  - 정점의 법선벡터를 요함. 인접면의 법선벡터를 평균하여 구함
  - 정점의 색으로부터 내부면의 색을 선형보간
- 경면광을 감안하지 않음: 실제적인 정점의 법선벡터와 근사적으로 계산된 법선벡터가 완전히 일치하지 않기 때문

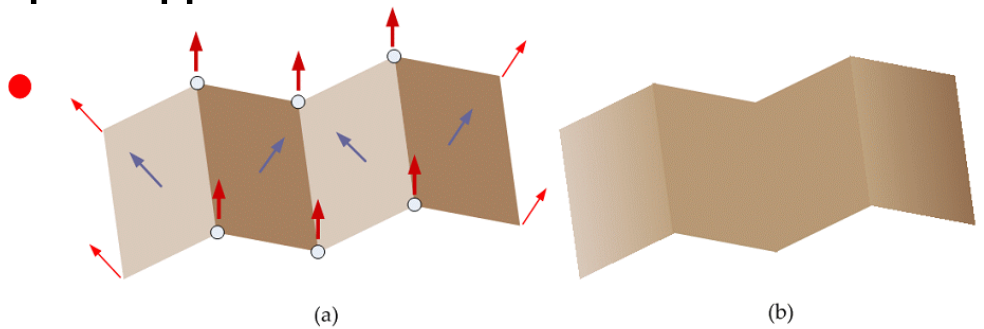
구로(Gouraund Shading) 셰이딩

- 플랫 셰이딩보다는 부드러움
  - 마하 밴드 효과는 그대로 남아있음.



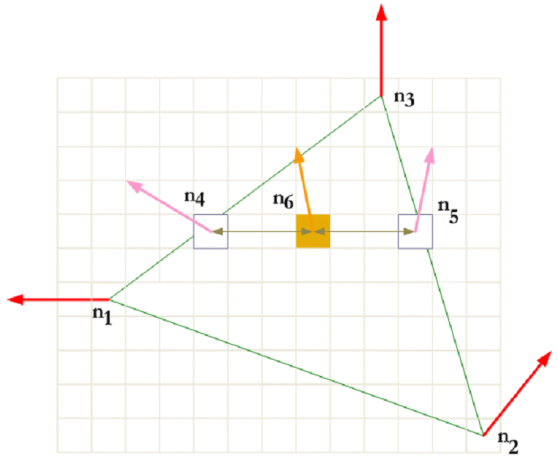
(a) (b)

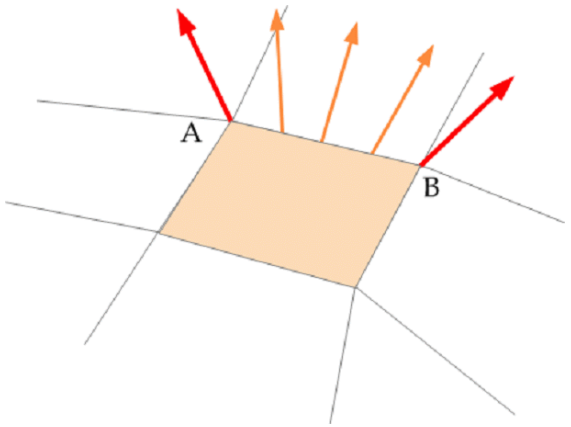
- 경우에 따라서 오류



(a) (b)

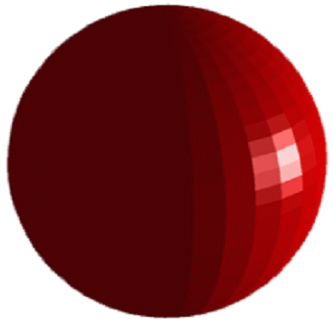
퐁(Phong) 셰이딩



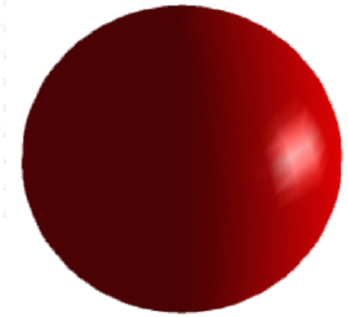


- 정점의 반사광 세기를 보간하는 구로 셰이딩과 달리 퐁(Bui-Tong Phong)에 의해 제안된 퐁 셰이딩(Phong Shading)은 정점의 법선 벡터를 보간.
- 일반적으로 구로셰이딩보다 사실적
- 경면광을 부여할 수 있음

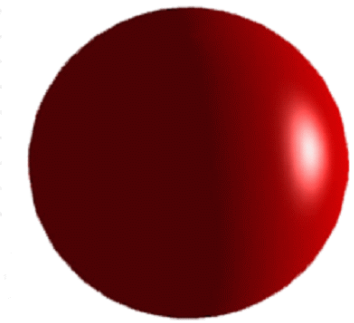
퐁 셰이딩



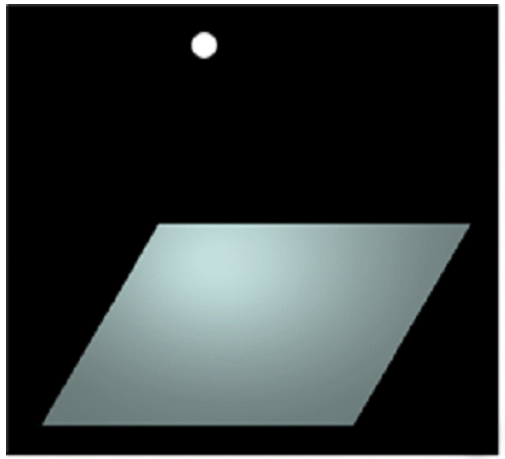
Flat



Gouraud

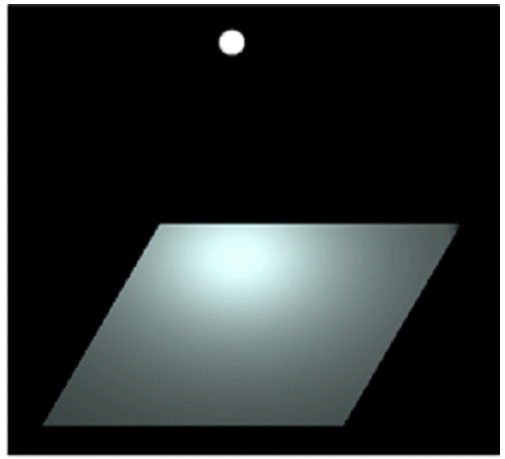


Phong



(a)

Gouraud



(b)

Phong

GL의 조명과 음영

- GL에서 조명과 음영을 가하기 위해서 필요한 작업
  1. 조명 기능 활성화 (Enable Lighting)
  2. 광원 정의 (Specify Lights)
  3. 음영 모드 정의 (Specify Shading Mode)
  4. 법선 벡터 정의 (Specify Surface Normals)
  5. 물체면 특성 정의 (Specify Surface Material Properties)
  6. 조명 모델 정의 (Specify Lighting Model)

조명 기능 활성화

- void glEnable(GLenum capability);
- void glDisable(GLenum capability);
  - 조명 기능 활성화 : glEnable(GL_LIGHTING);
  - 조명 기능 비활성화: glDisable(GL_LIGHTING);
  - 일단 조명 기능이 활성화되면 물체의 색은 광원과 물체의 특성에 의해서만 좌우되고 glColor*() 함수에 의해서 정의된 정점의 색은 무시된다.

광원 정의

- 광원의 활성화
  - glEnable(GL_LIGHT0)~glEnable(GL_LIGHT7)
    - GL_LIGHT0에서 GL_LIGHT7까지 최소 8개의 광원을 개별적으로 정의하거나 제어할 수 있음.
  - 현재 드라이버의 지원 광원 갯수 확인
    - GLint n;
    - glGetIntegerv(GL_MAX_LIGHT, &n);

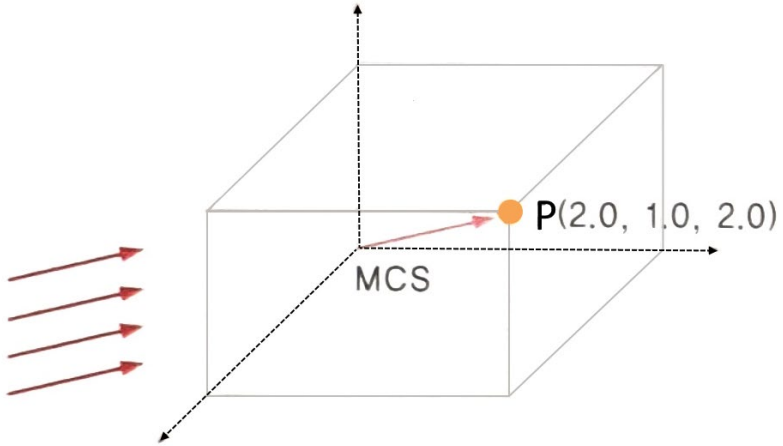
광원의 종류

- void glLightfv(GLenum LightSourceID, GLenum Parameter, TYPE *Valve);
  - LightSourceID: GL_LIGHT0, GL_LIGHT1, GL_LIGHT2 등 광원의 아이디
  - Parameter: GL_POSITON 명시
  - Valve: 광원의 위치

```
void InitLight() {
   GLfloat MyLightPosition[] = {1.0, 2.0, 3.0, 1.0};
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glLightfv(GL_LIGHT0, GL_POSITION, MyLightPosition);
}
```

광원의 위치
조명 기능 활성화
0번 광원 활성화
광원 위치 할당

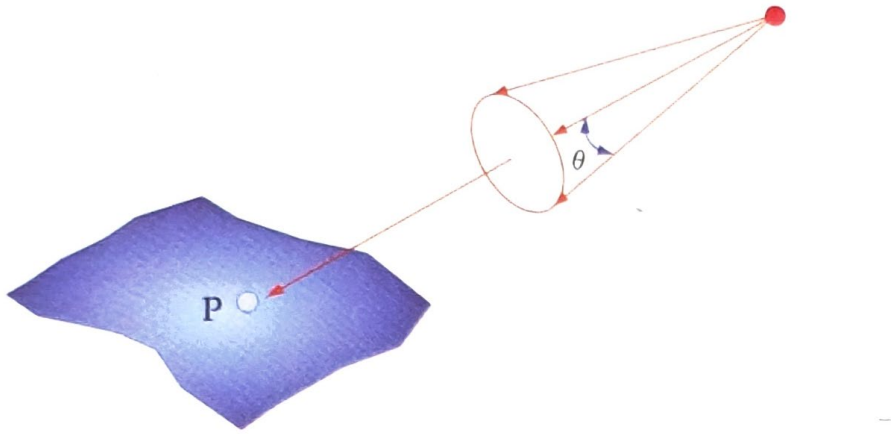
광원의 종류



- 광원을 (x, y, z, w) 로 표현할 때
  - w≠0 : (x/w, y/w, z/w)에 위치한 위치성 광원
  - w=0: 원점에서 (x, y, z)를 향하는 벡터 방향의 방향성 광원
- lightPosition을 (2.0, 1.0, 2.0, 0.0)으로 설정하면 무한대 거리에서 (2.0, 1.0, 2.0) 방향의 빛

위치성 광원

- 스포트라이트 설정




```
void InitLight() {
   GLfloat MyLightPosition[] = {1.0, 2.0, 3.0, 1.0};
   GLfloat MyLightDirection[] = {3.0, 4.0, 3.0};
   GLfloat MySpotAngle[] = {50.0};
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glLightfv(GL_LIGHT0, GL_POSITION, MyLightPosition);
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, MyLightDirection);
   glLightfv(GL_LIGHT0, GL_SPOT_CUTOFF, MySpotAngle);
   glLightfv(GL_LIGHT0, GL_SPOT_EXPONENT, 2.0);
}
```

광원의 위치
빛의 방향
조명각
조명 기능 활성화
0번 광원 활성화
광원 위치 할당
방향설정
조명각 설정
승수 설정

광원의 색

- 광원에서 나오는 빛의 색은 주변광(Ambient Light), 확산광(Diffusive Light), 경면광(Specular Light)에 대해서 입사광의 크기를 R, G, B, A로 나누어서 정의

```
void InitLight() {
   GLfloat MyLightPosition[] = {1.0, 2.0, 3.0, 1.0};                    광원의 위치
   GLfloat MyLightDirection[] = {3.0, 4.0, 3.0};                        빛의 방향
   GLfloat MyLightAmbient[] = {0.0, 1.0, 0.0, 1.0};                     주변반사의 입사광은 주로 녹색
   GLfloat MyLightDiffuse[] = {1.0, 0.0, 0.0, 1.0};                     확산반사의 입사광은 주로 적색
   GLfloat MyLightSpecular[] = {1.0, 0.0, 0.0, 1.0};                    경면반사의 입사광은 주로 적색
   GLfloat MySpotAngle[] = {50.0};                                      조명각
   glEnable(GL_LIGHTING);                                                       조명 기능 활성화
   glEnable(GL_LIGHT0);                                                 0번 광원 활성화
   glLightfv(GL_LIGHT0, GL_POSITION, MyLightPosition);                  광원 위치 할당
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, MyLightDirection);                   방향설정
   glLightfv(GL_LIGHT0, GL_SPOT_CUTOFF, MySpotAngle);                   조명각 설정
   glLightfv(GL_LIGHT0, GL_SPOT_EXPONENT, 2.0);                         승수 설정
   glLightfv(GL_LIGHT0, GL_AMBIENT, MyLightAmbient);
   glLightfv(GL_LIGHT0, GL_DIFFUSE, MyLightDiffuse);
   glLightfv(GL_LIGHT0, GL_SPECULAR, MyLightSpecular);
}
```

- 조명 모델의 관점에서 볼 때 위 array의 값들은 각각 광원의 I_a, I_d, I_s에 해당

거리에 따른 빛의 약화

- 약화함수 설정

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

  - a, b, c의 값을 아래와 같이 직접 정의
  - glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.0);
  - glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 2.0);
  - glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 3.0);

광원의 위치 제어

- 광원의 위치와 방향은 모델뷰 행렬을 통해서 자유롭게 조절할 수 있음. GL에서 광원은 기하학적 물체(Geometric Object)로 간주하기 때문.
- 광원과 물체 상호간의 위치 분류
  - 고정 위치 광원

```
glMatrixMode(GL_MODEIVIEW);                    (1)
glLoadIdentity();                              (2)
glLookAt(..... );                              (3)
glLightfv(...., GL_POSITION, ...);             (4)
glRotatef( );                                  (5)
glutSolidCube();                               (6)
```

  - 물체 관점에서 변환은 함수 호출의 역순이라고 생각하면 물체의 회전(5) 후에 조명 위치가 결정되고 시점이 결정.

광원의 위치 제어

- 움직이는 광원

```
glMatrixMode(GL_MODEIVIEW);                (1)
glLoadIdentity();                          (2)
glLookAt(..... );                          (3)
glPushMatrix();                            (4)
glRotatef(   );                            (5)
glLightfv(...., GL_POSITION, ...);         (6)
glPopMatrix();                             (7)
glutSolidCube();                           (8)
```

- (5)~(6)의 루틴 실행 후 기존의 CTM으로 복귀하기 때문에 (8)에 정의된 cube는 위치가 고정
- 반면 광원은 회전 변환(5)을 거친 후 적용

광원의 위치 제어

- 시점 위치의 광원

```
glMatrixMode(GL_MODEIVIEW);                    (1)
glLoadIdentity();                              (2)
glLightfv(...., GL_POSITION, MyLightPosition); (3)
glLookAt(..... );                              (4)
glRotatef(   );                                (5)
glutSolidCube();                               (6)
```

- 광원의 위치가 MyLightPosition에 고정됨. View Matix가 정의되기 전에 선언되기 때문에 시점과 함께 움직이게 됨.

음영 모드 정의

- void glShadeModel (GLenum mode);
  - GL은 플랫 셰이딩과 구로 셰이딩만 지원함
  - mode = GL_FLAT : 플랫 셰이딩
  - mode = GL_SMOOTH : 구로 셰이딩
  - 기본값은 GL_SMOOTH

```
void display() {
   glClear(GL_COLOR_BUFFER_BIT);
   glShadeMode(GL_SMOOTH);
   glBegin(GL_TRIANGLES);
      glColor3f(0.0, 1.0, 0.0);
      glVertex3f(-0.5, -0.5, 0.0);
      glColor3f(0.0, 0.0, 0.0);
      glVertex3f(0.5, -0.5, 0.0);
     glColor3f(1.0, 0.0, 0.0);
     glVertex3f(0.0, -0.5, 0.0);
   glEnd();
}
```

녹색
정점 1
흑색
정점 2
적색
정점 3

물체면 특성 정의

- GL에서는 광원의 특성과는 독립적으로 물체의 특성을 정의할 수 있음
- 물체의 특성은 색과 물체면의 매끄러움을 뜻함.
- 색은 주변 반사, 확산 반사, 경면 반사 등 반사의 종류별로 물체면에서 어떤 크기로 빛을 반사하는지를 뜻함.
- 물체면의 매끄러움은 경면광의 광택 계수를 뜻함.

```
GLfloat MyMaterialAmbient[] = {0.2, 0.2, 0.2, 1.0};
GLfloat MyMaterialDiffuse[] = {1.0, 0.0, 0.0, 1.0};
GLfloat MyMaterialSpecular[] = {.0, 1.0, 1.0, 1.0};
GLfloat MyShininess = 90.0;

glMaterialv(GL_FRONT, GL_AMBIENT, MyMaterialAmbient);
glMaterialv(GL_FRONT, GL_DIFFUSE, MyMaterialDiffuse);
glMaterialv(GL_FRONT, GL_SPECULAR, MyMaterialSpecular);
glMaterialv(GL_FRONT, GL_SHININESS, MyShininess);
```

예제 1

```
void init()
{
	GLfloat global_ambient[] = {0.1, 0.1, 0.1, 1.0};// 전역 주변 반사
	GLfloat light0_ambient[] = { 0.5, 0.4, 0.3, 1.0 };// 0번 광원 특성
	GLfloat light0_diffuse[] = { 0.5, 0.4, 0.3, 1.0 };
	GLfloat light0_specular[] = { 1.0, 1.0, 1.0, 1.0 };
	GLfloat light1_ambient[] = { 0.0, 0.0, 0.9, 1.0 };// 1번 광원 특성
	GLfloat light1_diffuse[] = { 0.5, 0.2, 0.3, 1.0 };
	GLfloat light1_specular[] = { 0.0, 0.0, 0.0, 1.0 };

	glShadeModel(GL_SMOOTH);// Gouraud shading
	glEnable(GL_DEPTH_TEST);
	glEnable(GL_LIGHTING);// 조명 기능 활성화

	glEnable(GL_LIGHT0);
	glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
	glLightfv(GL_LIGHT0, GL_SPECULAR, light0_specular);

	glEnable(GL_LIGHT1);
	glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
	glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
	glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);

	glLightfv(GL_FRONT, GL_AMBIENT, material_ambient);// 물체 특성
	glLightfv(GL_FRONT, GL_DIFFUSE, material_diffuse);
	glLightfv(GL_FRONT, GL_SPECULAR, material_specular);
	glLightfv(GL_FRONT, GL_SHININESS, material_shininess);

	glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
	glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
}
```

```
void display() {
	GLfloat LightPosition0[] = { 0.0, 0.0, 2.0, 1.0 };// 0번 광원 위치
	GLfloat LightPosition1[] = { 1.0, 1.0, 1.0, 1.0 };// 1번 광원 위치
	GLfloat LightDirection1[] = { -0.5, -1.0, -1.0, 1.0 };// 1번 광원 방향
	GLfloat SpotAngle1[] = { 20.0 };
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(0.5, 0.5, 0.5, 0.0, 0.0, -1.0, -.0, 1.0, 0.0);
	glTranslatef(0.3, 0.3, 0.0);// model transformation
	glLightfv(GL_LIGHT0, GL_POSITION, LightPosition0);// 위치성 광원
	glLightfv(GL_LIGHT1, GL_POSITION, LightPosition1);// 스포트 라이트
	glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, LightDirection1);// 방향
	glLightfv(GL_LIGHT1, GL_SPOT_CUTOFF, SpotAngle1);// 각도
	glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 1.0);
	glutSolidTorus(0.3, 0.6, 800, 800);
	glFlush();
}
```

예제 2

```
void init()
{
    GLfloat light0_ambient[] = { 0.5, 0.4, 0.3,
    1.0 };// 0번 광원 특성
    GLfloat light0_diffuse[] = { 0.8, 0.7, 0.6, 1.0 };
    GLfloat light0_specular[] = { 1.0, 1.0, 1.0, 1.0 };

    GLfloat material_ambient[] = { 0.4, 0.4, 0.4,
    1.0 };// 물체 특정
    GLfloat material_diffuse[] = { 0.9, 0.9, 0.9, 1.0 };
    GLfloat material_specular[] = { 1.0, 1.0, 1.0,
    1.0 };
    GLfloat material_shininess[] = { 25.0 };

    glShadeModel(GL_SMOOTH);// Gouraud shading
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);// 조명 기능 활성화

    glEnable(GL_LIGHT0);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light0_specular);
}
```

```
void display() {
    GLfloat LightPosition[] = { 0.0, 0.0, 1.5, 1.0 };// 0번 광원 위치
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glTranslatef(0.0, 0.0, -5.0);
    glPushMatrix();
    glRotatef(SpinAngle, 1.0, 0.0, 0.0);
    glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);
    glTranslatef(0.0, 0.0, 1.5);
    glDisable(GL_LIGHTING);
    glColor3f(0.9, 0.9, 0.9);
    glutWireSphere(0.06, 10, 10);
    glEnable(GL_LIGHTING);
    glPopMatrix();
    glutSolidSphere(1.0, 400, 400);
    glPopMatrix();
    glFlush();
}
```

예제 2

```
void mouse(int button, int state, int x,
int y)
{
    switch (button) {
    case GLUT_LEFT_BUTTON:
    if (state == GLUT_DOWN) {
    SpinAngle = (SpinAngle + 15) % 360;
    glutPostRedisplay();
    }
    break;
    default:
    break;
    }
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    //glClearColor(0.0, 0.0, 0.0, 0.0);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA |
    GLUT_DEPTH);
    glutCreateWindow("openGL lighting ex");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```