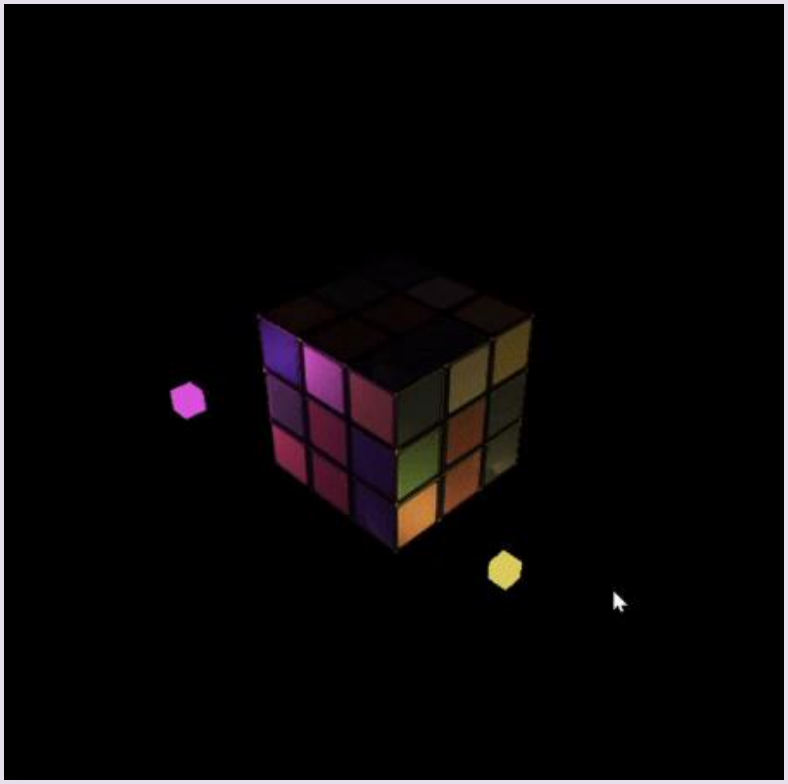
디지털 그래픽스 강의 12주차

Lighting & Shading *(3)*

Illumination model



How to Shade in OpenGL?

1. Light Source : Position & Type
2. **Illumination Model : Light** & Material

   Ambient, Diffuse, Specular

Illumination model

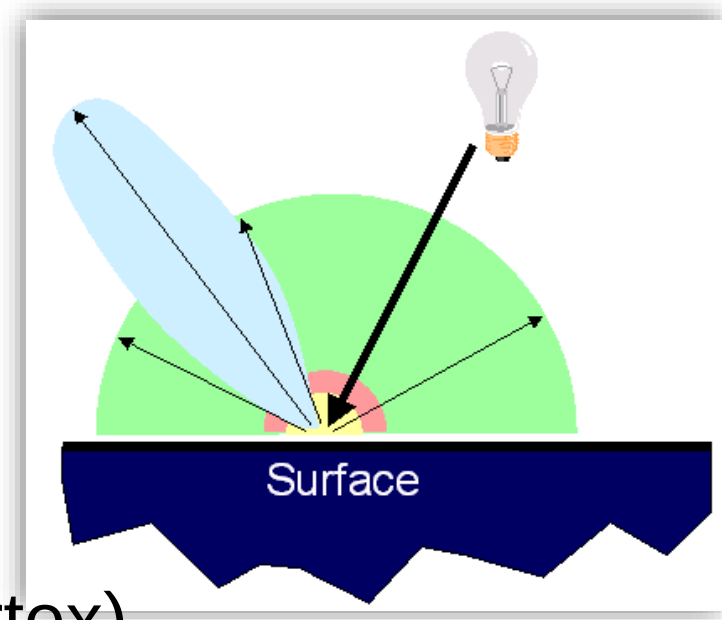
조명 빛이 물체에 닿아 어떻게 반사되어 눈에 어떻게 보이는지를 모델링해둔것.
어떤 빛인지, 어떤 재질의 물체인지에 따라 illumination이 다름

- Simple analytic model in OpenGL

모든방향으로 일정반사(난반사)가 대부분의 경우.
그러나 어떤 경우는 전반사(한쪽으로만 많이)

  - The model used by OpenGL – consider three types of light contribution to compute the final illumination of an object

    - **Ambient light (주변광, 간접광)**
      조명에서 직접 들어오는 게 아니라.
      주변 환경에 따라 간접적으로 들어오는 빛
    - **Diffuse reflection (난반사)**
      모든방향으로 일정반사
      물체에 직접 닿는 것
    - **Specular reflection (전반사)**
      이 덕에 하이라이트를 볼 수 있음.
      물체에 직접 닿는 것.

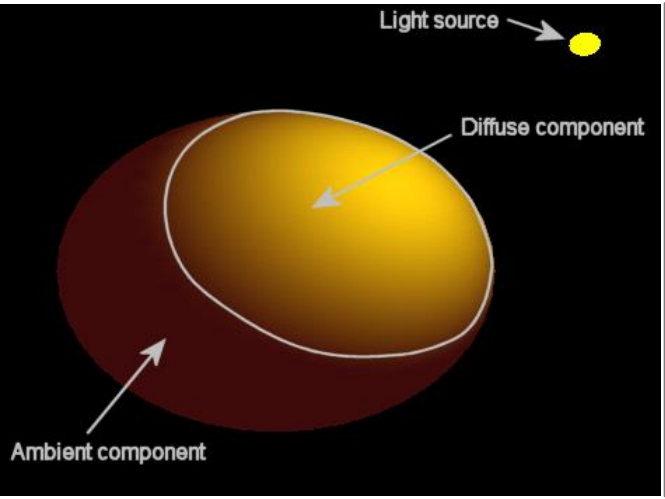


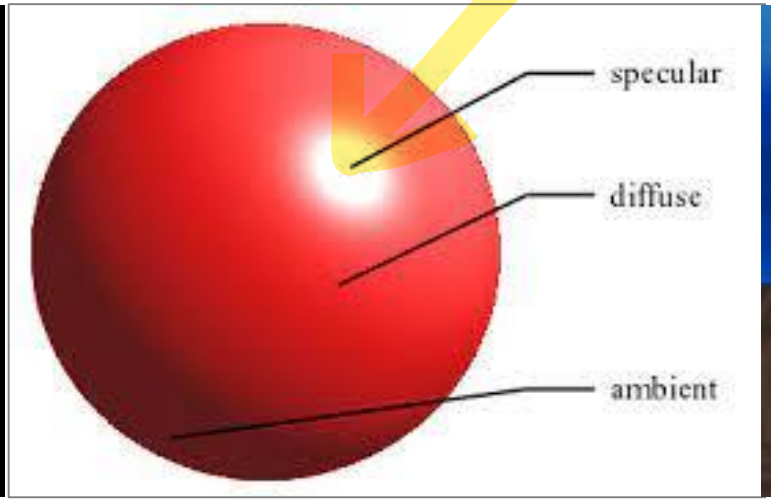
- Final illumination of a point (vertex)

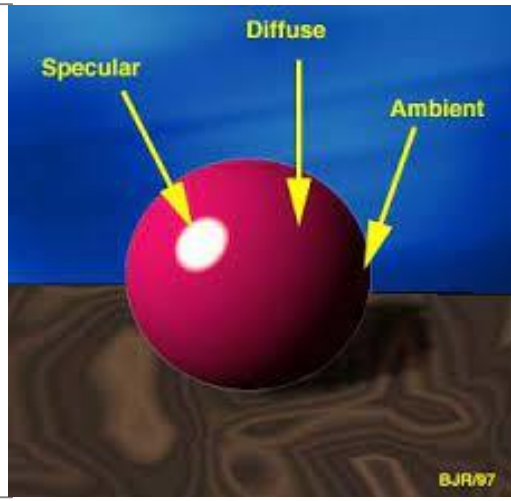
  = ambient + diffuse + specular

Ambient vs. Diffuse component

- Ambient component
  - the light that is scattered by the environment
  - 간접적인 빛, ambient component가 없으면 그림자가 진 부분은 완전히 어둡게 보일 수 있다.
- Diffuse reflection
  - reflects equally in all direction
  - 표면에서 '빛'이 모든 방향으로 균일하게 반사





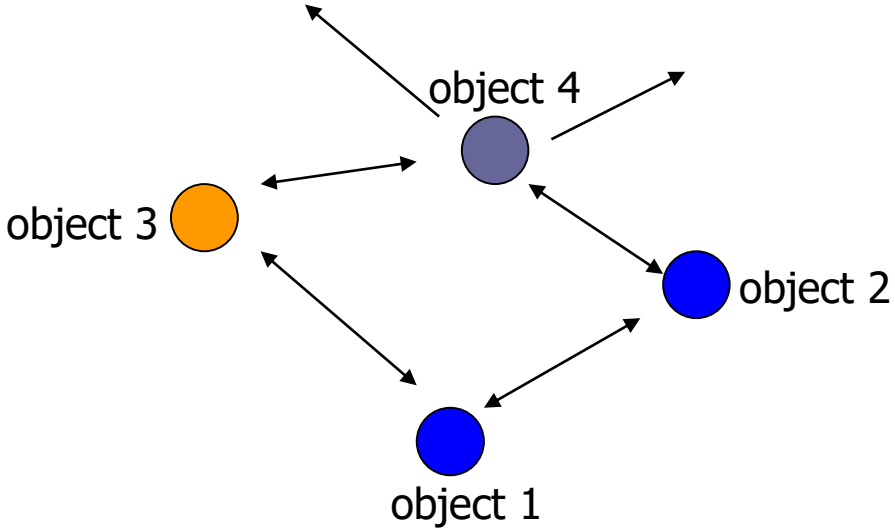


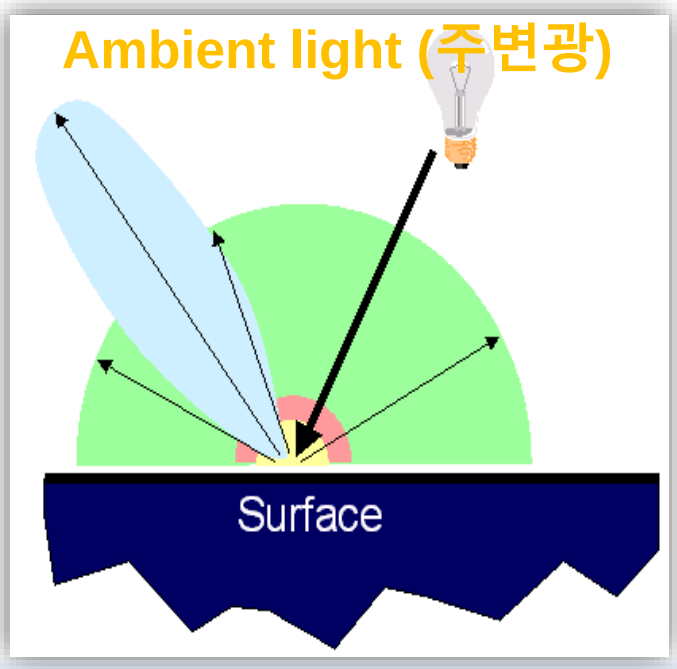
Ambient Light

Ambient light

간접적인 빛, 조명의 위치/ 방향과 무관

- **Ambient light (background light)**
  - the light that is scattered by the environment
- Independent of the light position, object orientation, observer's position or orientation
- ambient light has no direction





Ambient Lighting Example

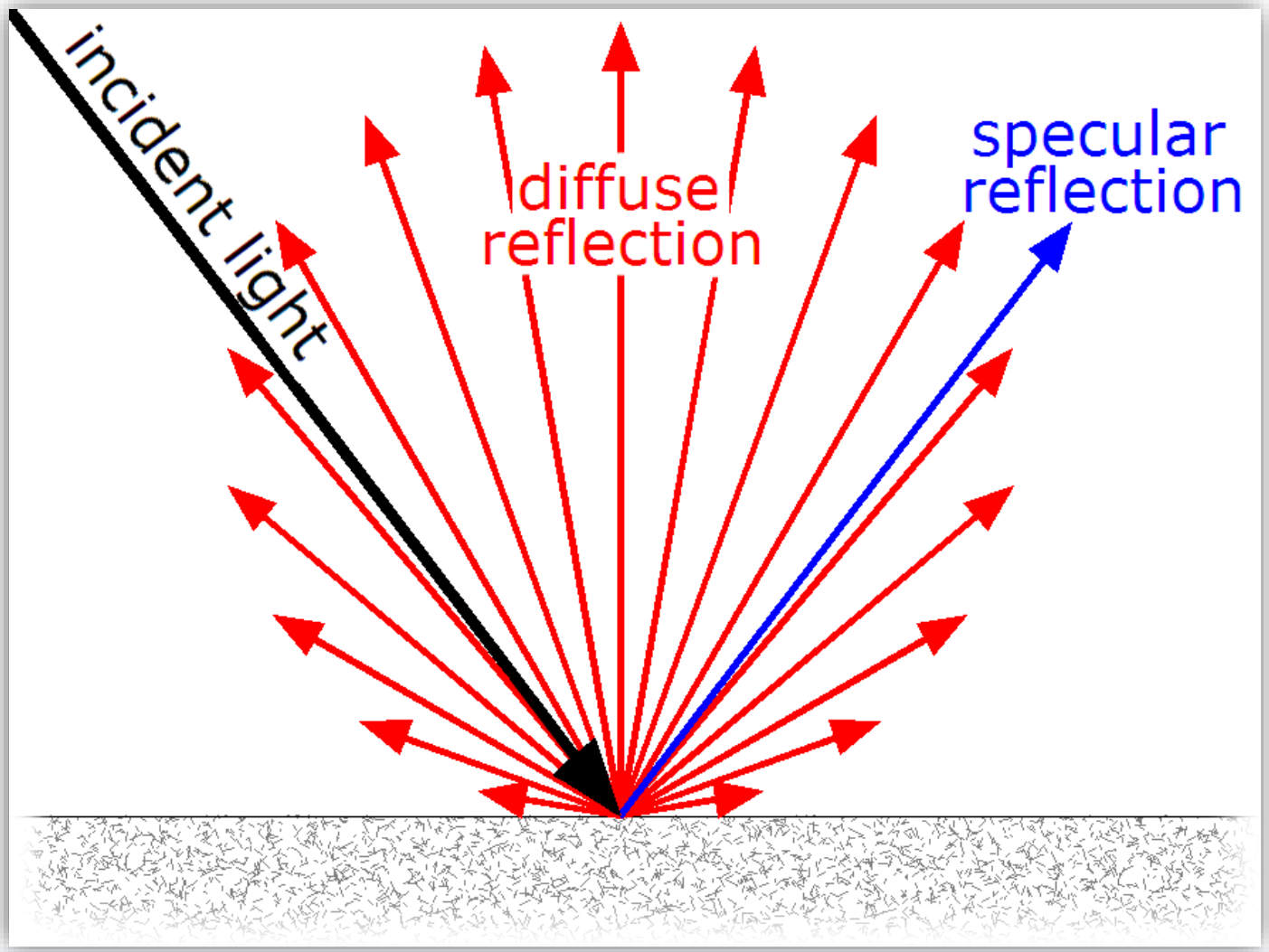
직접광이 없어 전반적으로 어둡다

- The object will have a **uniform color**
  - It looks similar to the result **without lighting**

Diffuse Reflection

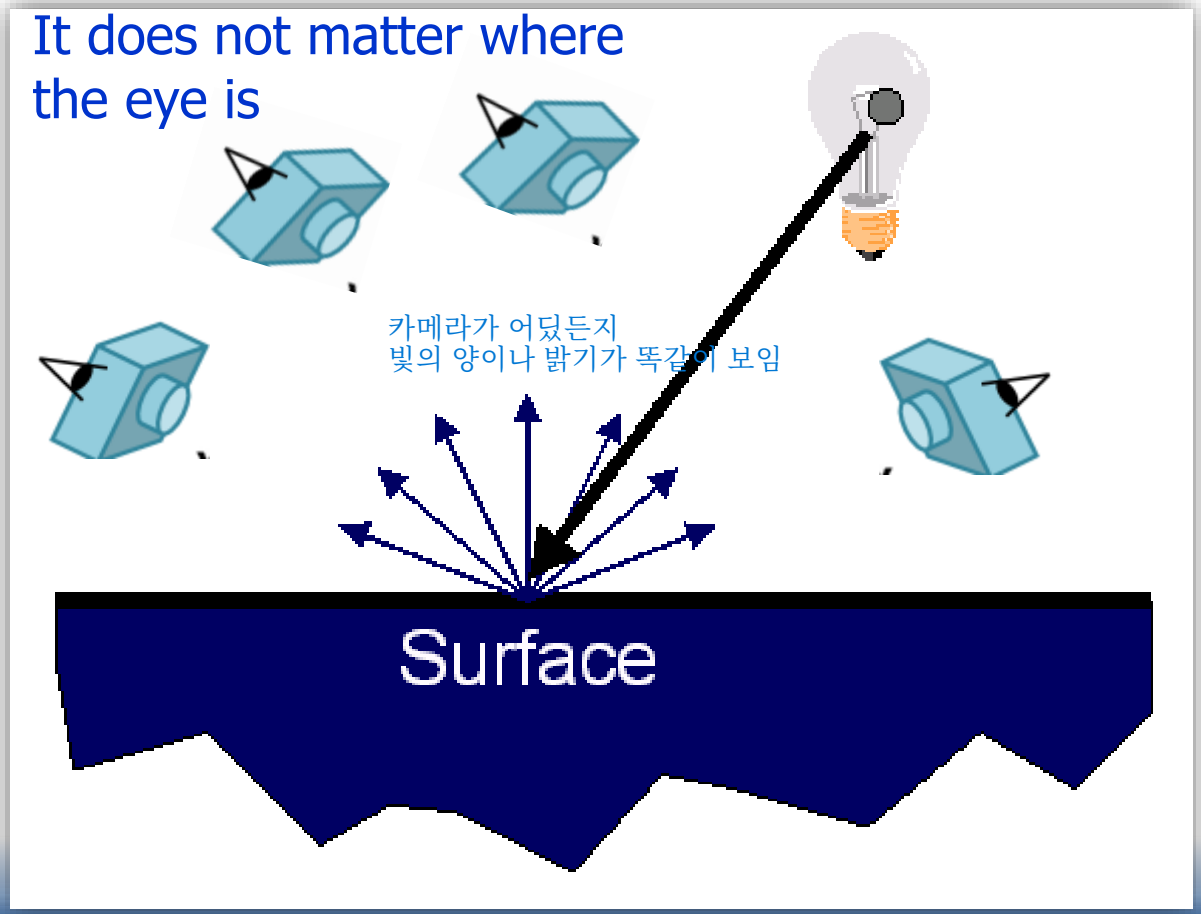
Diffuse & Specular Reflection

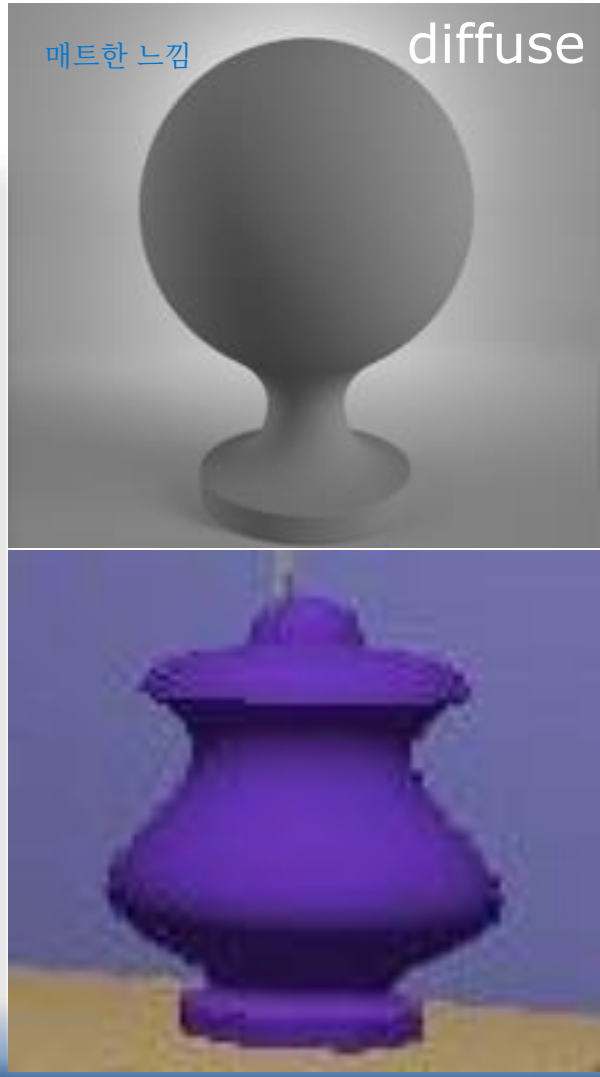
- 난반사 & 전반사 모든방향으로 일정 빛 반사: diffuse >> 매트한 느낌으로 렌더링됨



Diffuse Reflection (난반사)

- **Diffuse light** : The illumination that a surface receives from a light source and **reflects equally in all direction**
  - With equal intensity → view independent



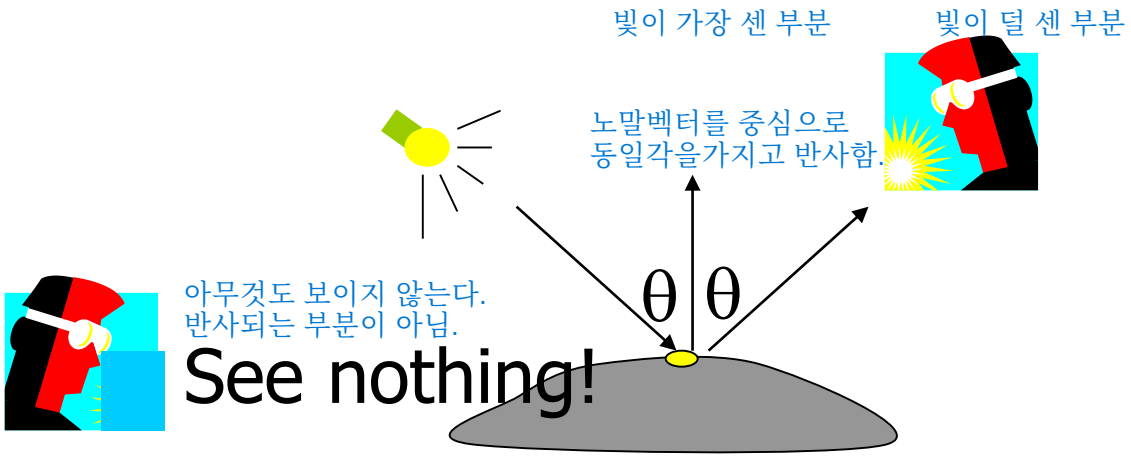


Diffuse lighting example

Specular Reflection

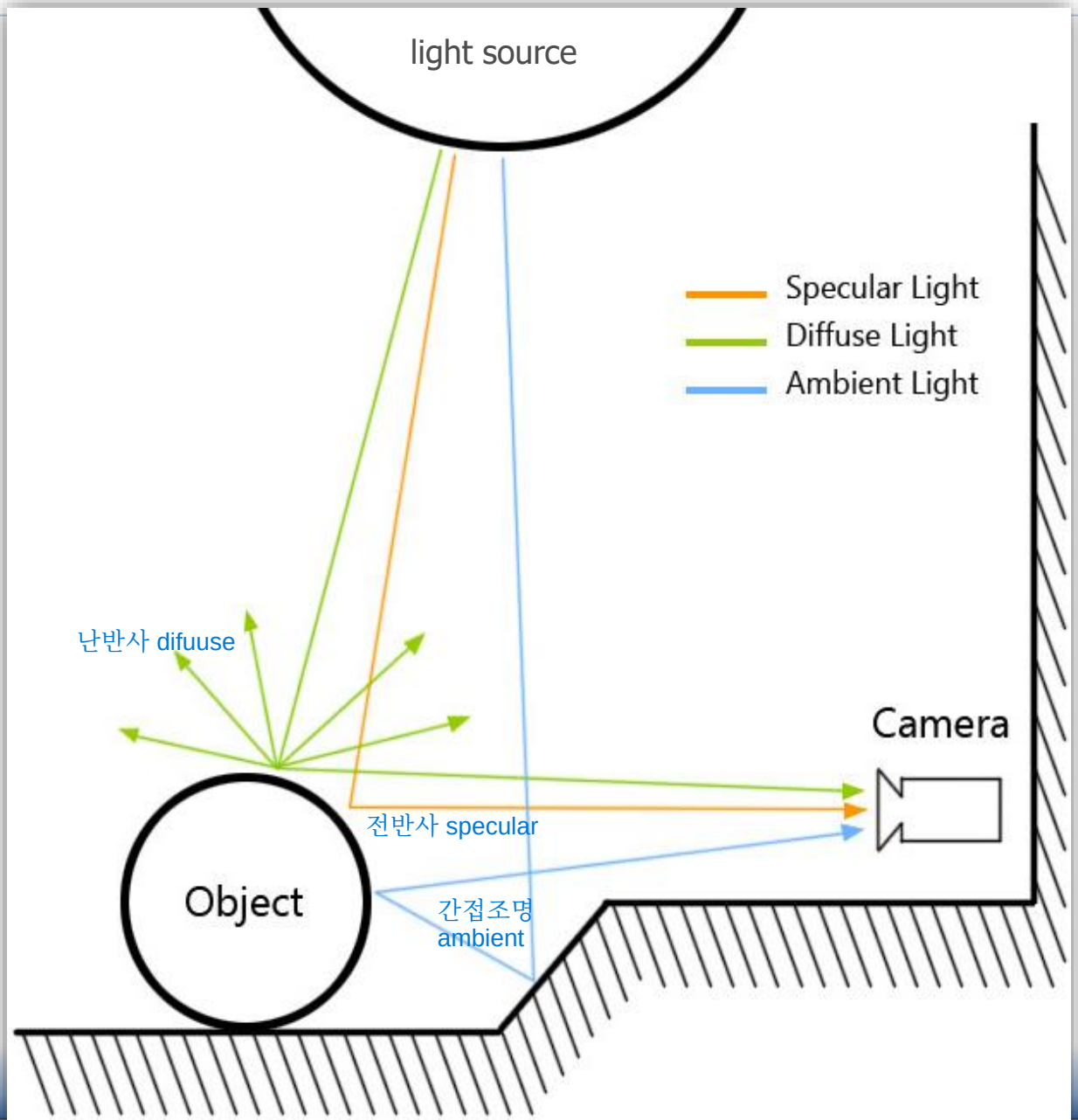
Specular light contribution

- **Specular reflection** : **regular reflection**, is the mirror-like reflection of light from a surface.
- Highlight effect : the bright spot on the object
  - ex) mirrors, metals
- The result of total reflection of the incident light in a concentrate region
- Each incident ray is reflected at the same angle to the surface normal as the incident ray,



Specular light example

Ambient & Diffuse & Specular Review



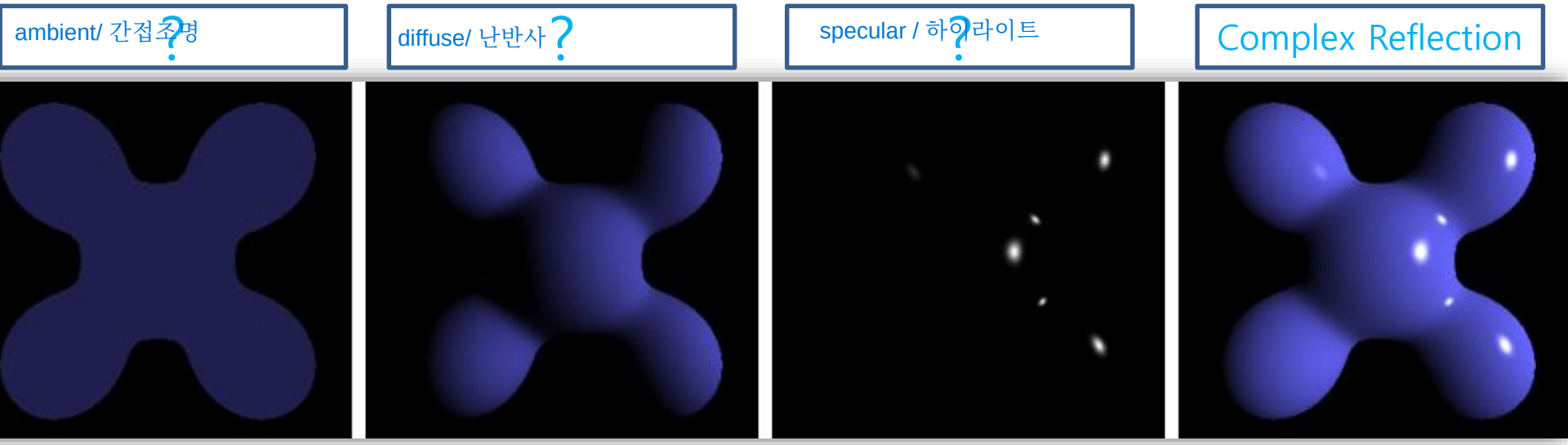
Put it all together

- Illumination from a light:

  Illum = ambient + diffuse + specular

- If there are N lights

  Total illumination for a point P = $\sum$ (Illum)



OpenGL implementation for light reflection

OpenGL Light

- OpenGL set light source parameters like this :

glposition대신

```
glLightfv(GLenum light, GLenum pname, GLfloat param)
```

- light : GL_LIGHT0 ~ GL_LIGHT7 (8개까지 가능)
- pname : GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR
- param : A vector array, {0.1, 0.0, 0.0, 1.0}; RGBA

a는 알파. 여기서는 사용하지 않음.

- 값) ambient, diffuse, specular light의 color & intensity

{0.0, 0.0, 1.0} blue

{0.2, 0.2, 0.2, 1.0} gray 조명색

{1.0, 1.0, 1.0, 1.0} white 하이라이트 컬러

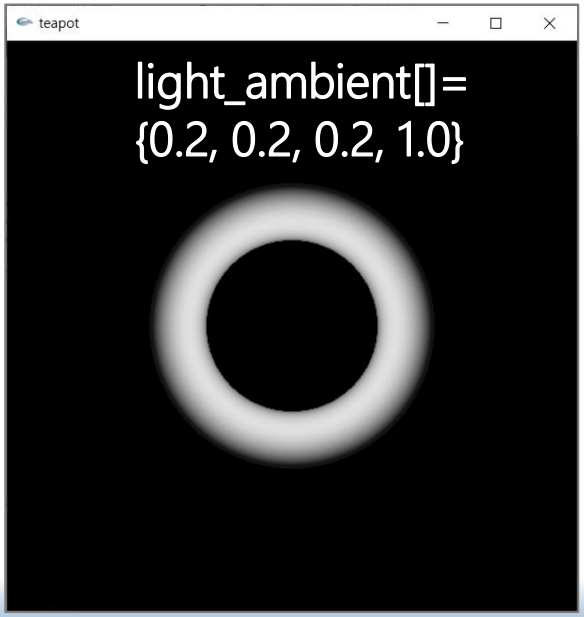
OpenGL Light Model

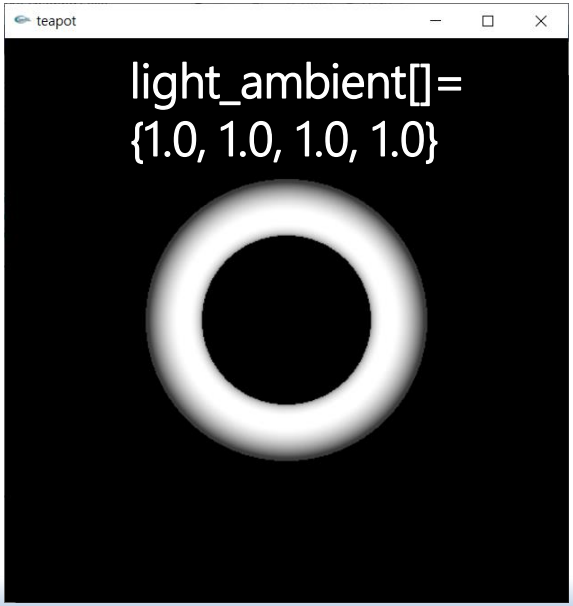
- Set individual attribute for a single light source

```
GLfloat light_ambient[] = {0.2, 0.2, 0.2, 1.0};   // ambient light is gray
GLfloat light_diffuse[]  = {1.0, 1.0, 1.0, 1.0};   // color of light (white)
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0};   // color of highlight on surface (white)

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
```

float, vector(배열)

ambient값이 커질수록 간접조명이 밝아짐.



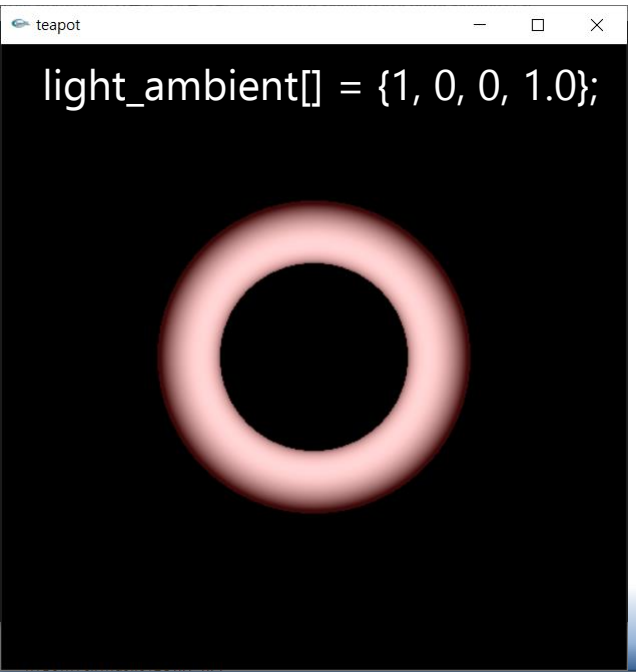


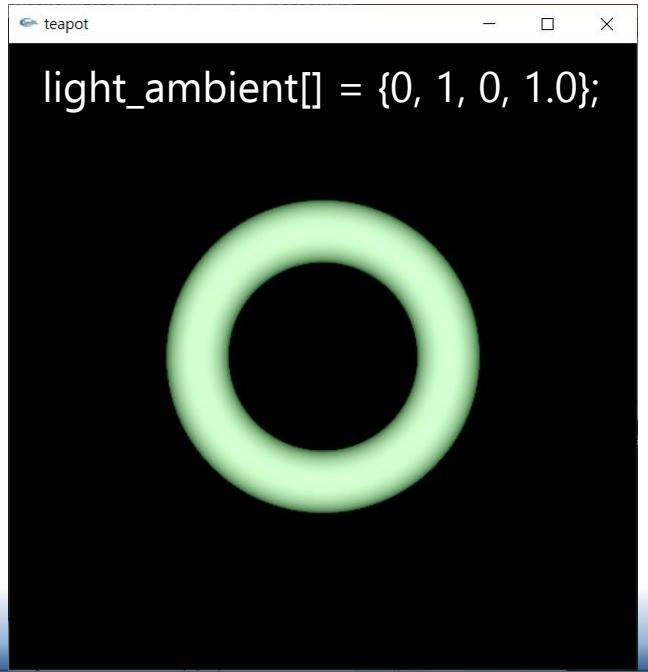
Ambient Light

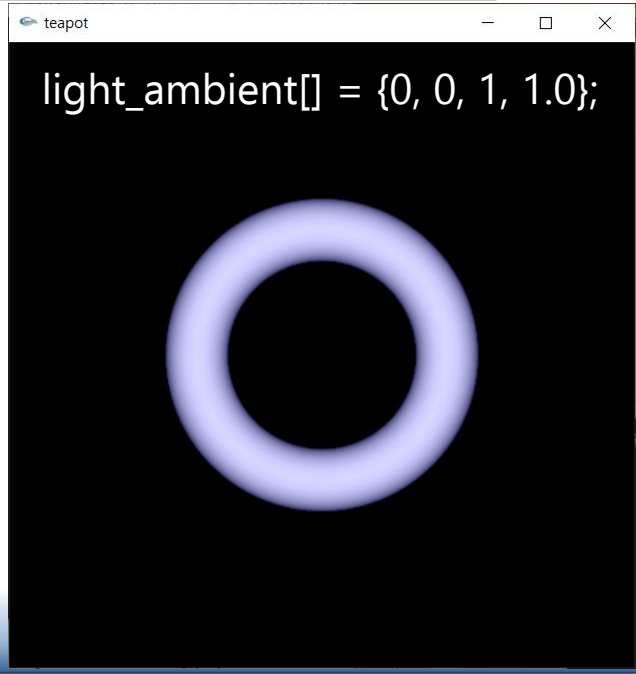
Ambient, diffuse, specular

- Light Reflectance model에서 **ambient** 요소를 조절하면서 solid torus 쉐이딩에 어떤 영향을 미치는지 확인해 보자. (키보드 x, y, z)

```
GLfloat light_ambient[] = {0.2, 0.2, 0.2, 1.0};  // ambient : gray
GLfloat light_diffuse[]   = {1.0, 1.0, 1.0, 1.0};
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
```





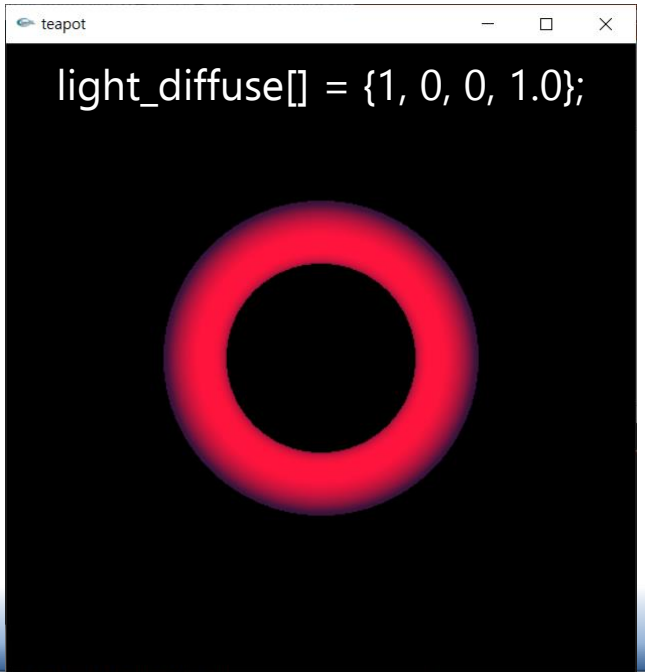


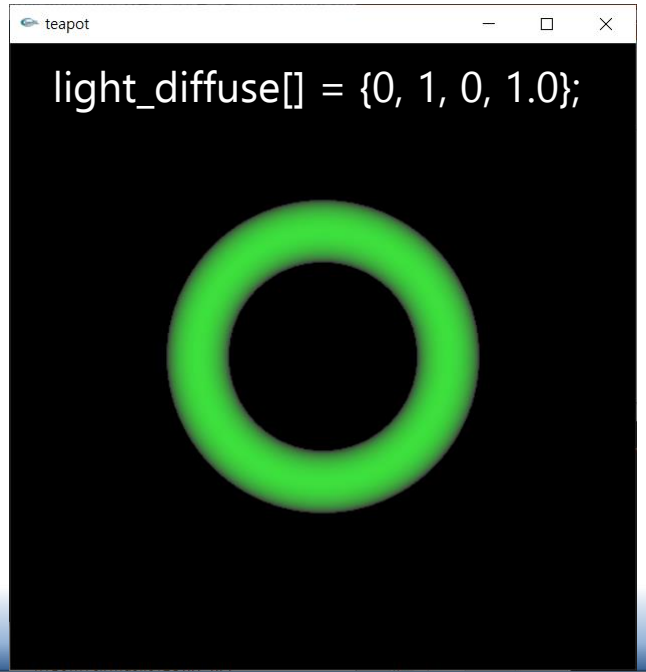
Diffuse Light

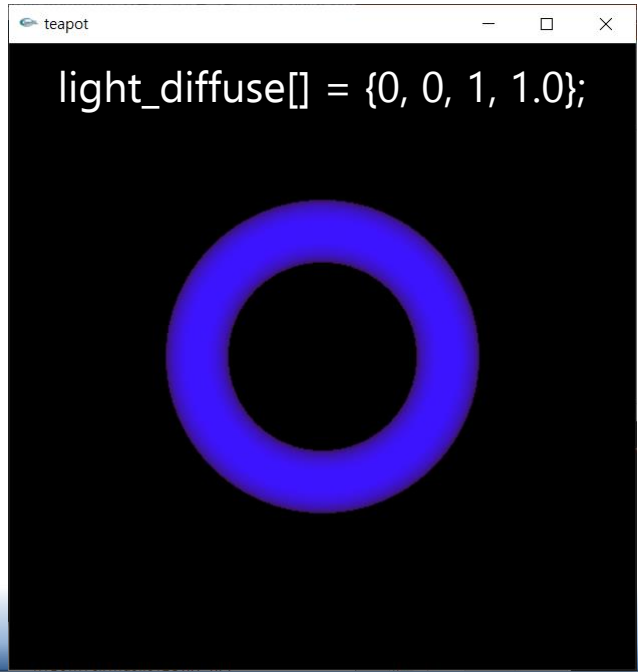
Ambient, diffuse, specular

- Light Reflectance model에서 **diffuse** 요소를 조절하면서 solid torus 쉐이딩에 어떤 영향을 미치는지 확인해 보자. (키보드 r, g, b)

```
GLfloat light_ambient[] = {0.2, 0.2, 0.2, 1.0};
GLfloat light_diffuse[]   = {1.0, 1.0, 1.0, 1.0};   // diffuse : white
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
```





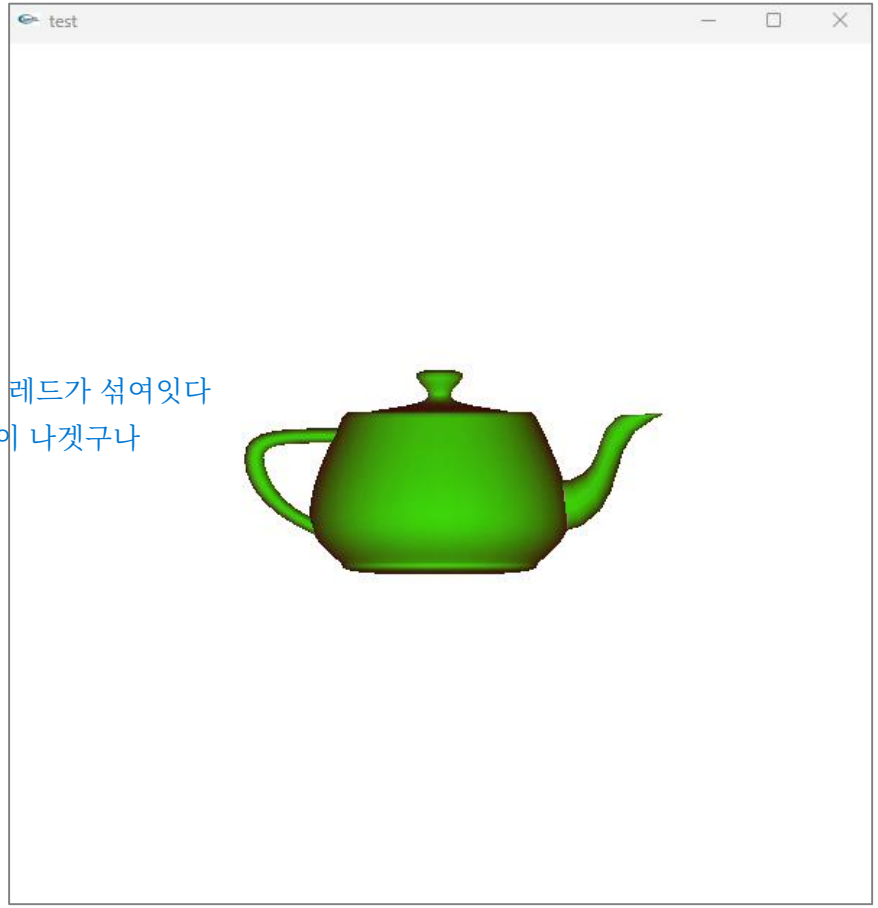


OpenGL Light Model: *code*

Ambient, diffuse, specular

```
#include <iostream>
#include <math.h>
#include <gl/glut.h>
using namespace std;

#define WIDTH 600
#define HEIGHT 600

GLfloat ambient[] = { 1.0, 0, 0, 1.0 };
GLfloat diffuse[] = { 0, 1.0, 0, 1.0 };
GLfloat specular[] = { 0, 0, 1.0, 1.0 };

float spin=0;

void display()
{
	GLfloat position[] = { 0, 0, 2, 1 };

	glClear(GL_COLOR_BUFFER_BIT
		| GL_DEPTH_BUFFER_BIT);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	// View Volume
	glOrtho(-1, 1, -1, 1, 1, 30);
```

간접조명에는 레드가 섞여잇다
대부분 초록빛이 나겟구나



OpenGL Reflectance Model: *code*

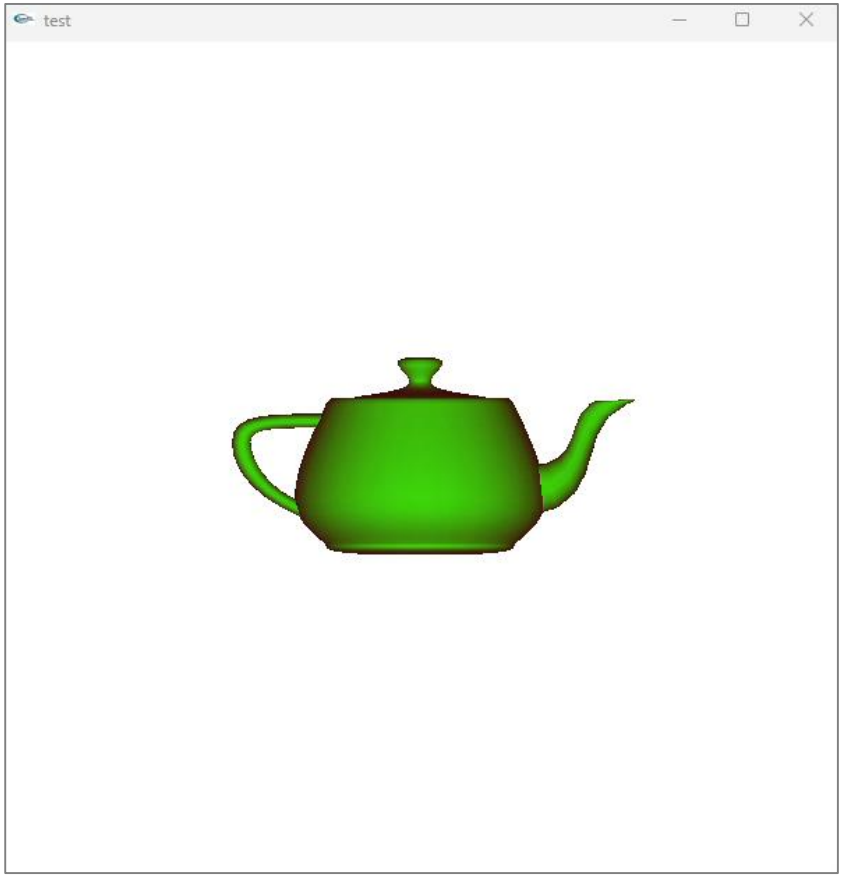
Ambient, diffuse, specular

```
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glPushMatrix();
    glRotatef(spin += 0.3, 1, 0, 0);
    // Light Position
    glLightfv(GL_LIGHT0, GL_POSITION, position);
    glPopMatrix();

    // Camera
    gluLookAt(0, 0, 2, 0, 0, 0, 0, 1, 0);

    // Light Type
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);

    // Teapot
    glColor3f(1, 0, 1); 바이올렛 컬러임. / 그러나 조명의 색상이 너무 세서
    glutSolidTeapot(0.3); 물체의 본래 색상이 잘 보이지 않음.
    glFlush();
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA
            | GLUT_DEPTH | GLUT_SINGLE);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(WIDTH, HEIGHT);
    glutCreateWindow("test");

    glClearColor(1, 1, 1, 0);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glutDisplayFunc(display);
    glutIdleFunc(display);
    glutMainLoop();
    return 0;
}
```

Quiz1. Illumination Model

- 이전 코드를 활용하여 다음 light를 구현하세요.
  - Violet 조명을 받는 주전자



How to Shade in OpenGL?

1. Light Source : Position & Type
2. Illumination Model : Light & **Material**

   Ambient, Diffuse, Specular, Emission

Illumination Model

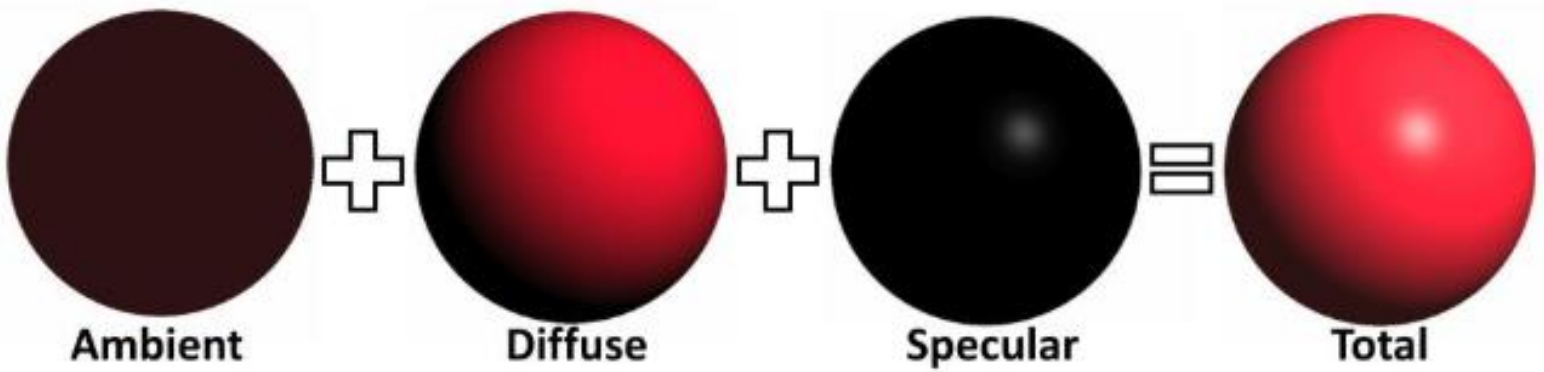
- **Light components** 전반사, 난반사, 간접조명 중 무엇이냐
  - **ambient, diffuse, specular**
  - Describe the properties of light sources.
  - 빛의 반사 특성

- **Material components** 물체의 재질과 어떤 조명인지에 따라 반사 정도가 다름. 물체쪽에서 반사하는 정도 조절 가능
  - **ambient, diffuse, specular**
  - Describe how a surface interacts with these light sources.
  - 물체(재질)의 빛 반사 특성

OpenGL Material

- OpenGL separates surface reflection into three components:

material 에서
specular 반사 정도가 0이다라고 하면 하이라이트 사라진다.



- Therefore, you should define three material properties. Each of them contributes to a reflection component:
  - Ambient
  - Diffuse
  - Specular

OpenGL Material

- OpenGL defines material properties like this :

물체의 앞면 혹은 뒷면 혹은 둘다 RGBA

```
glMaterialfv("face", "property", "value")
```

- face : GL_FRONT, GL_BACK, GL_FRONT_AND_BACK (물체의 앞면/뒷면)
- property : GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_SHININESS
- value : A vector array, {0.1, 0.0, 0.0, 1.0}; RGBA
- Shininess is from 0 (dull) to 128 (shiny)

얼마나 반짝거리는 것인지

Defining Material Properties

light는 white or gray

- Set material properties (ambient, diffuse, specular, shininess)

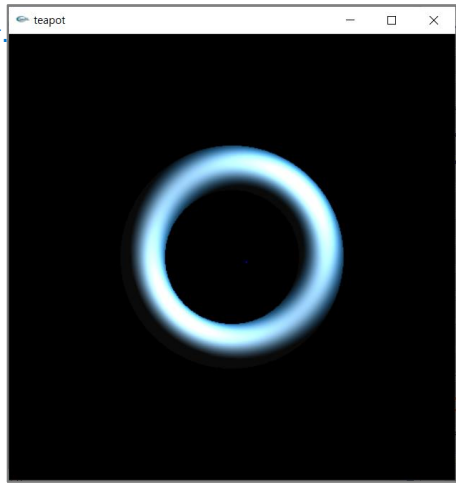
ambient의 R값을 0.1만 반사

```
GLfloat mat_a[] = {0.1, 0.1, 0.1, 1.0};      // ambient : gray – 영향 적음
GLfloat mat_d[] = {0.1, 0.5, 0.8, 1.0};      // diffuse : skyblue – 색상 결정
GLfloat mat_s[] = {1.0, 1.0, 1.0, 1.0};      // specular : white – 영향 많음
GLfloat low_sh[] = {5.0};                    // shininess n=5
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_a);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_d);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_s);
glMaterialfv(GL_FRONT, GL_SHININESS, low_sh);
```

diffuse
하늘색을 많이 반사함.

넓은 흰 부분이 하이라이트

얼마나 반짝거리는지.
값이 높을 수록 반짝거리며, 하이라이트 면적이 줄어든다.
값이 작을 수록 반짝거리며, 하이라이트 면적이 커진다.



물체 재질이 incident light를 얼마만큼 반사하는가 ?

Light & Material

- light & material interaction

```
GLfloat light_ambient[] = { 0.2, 0.2, 0.2, 1.0 };  // gray
GLfloat light_diffuse[]  = { 1.0, 0.5, 0.5, 1.0 };  // red
GLfloat light_specular[] = { 0.0, 1.0, 0, 0.0 };  // green
```

어두운

둔탁한

깽한

```
GLfloat material_ambient[] = {0.1, 0.1, 0.1, 1.0 };  // gray
GLfloat material_diffuse[]  = { 0.1, 0.2, 0.2, 1.0 };  // red
GLfloat material_specular[] = { 0.0, 1.0, 0.0, 1.0 }; // green
GLfloat material_shininess[] = { 50.0 }; // shininess
```

거의 영향 없음

0.1 > 1로 변경

하이라이트 영역 줄고 더 반짝인다.

What happen ?

```
GLfloat material_specular[] = { 0, 0, 1, 1.0 };
```

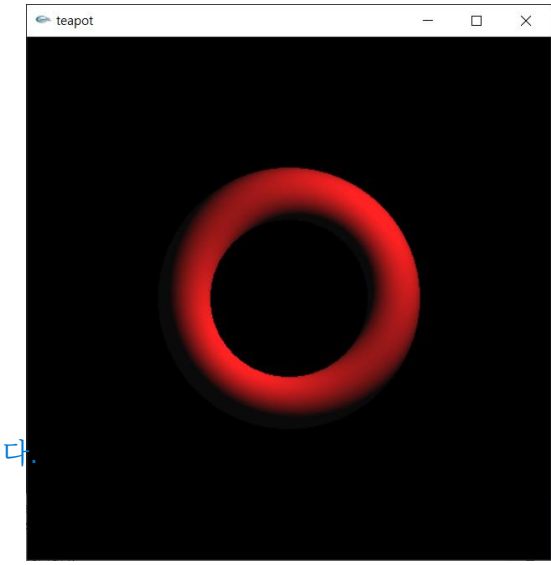
들어오는 라이트가 그린 색장이 들어오고, 블루를전체 반사하면 아무것도 없으니 하이라이트 없다.

What happen ?

```
GLfloat material_specular[] = { 0, 0, 0, 1.0 };
```

0.0.1로 주는 것과 0.0.0으로 주는 것 둘 다 하이라이트가 없다.

물체 재질이 incident light를 얼마만큼 반사하는가 ?

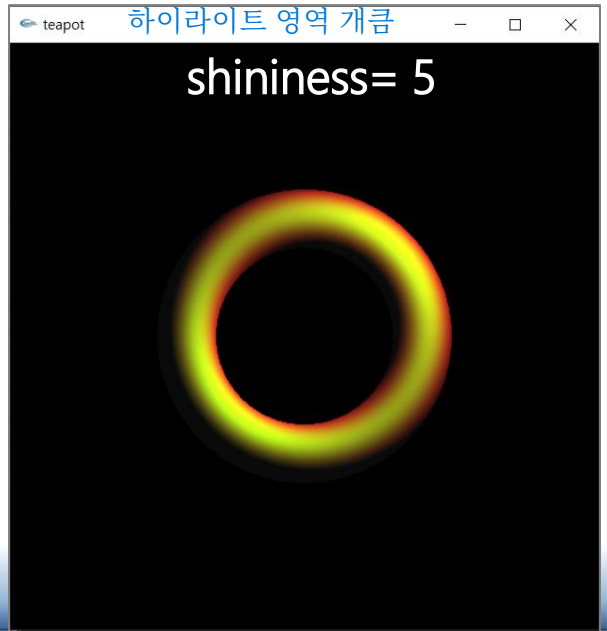


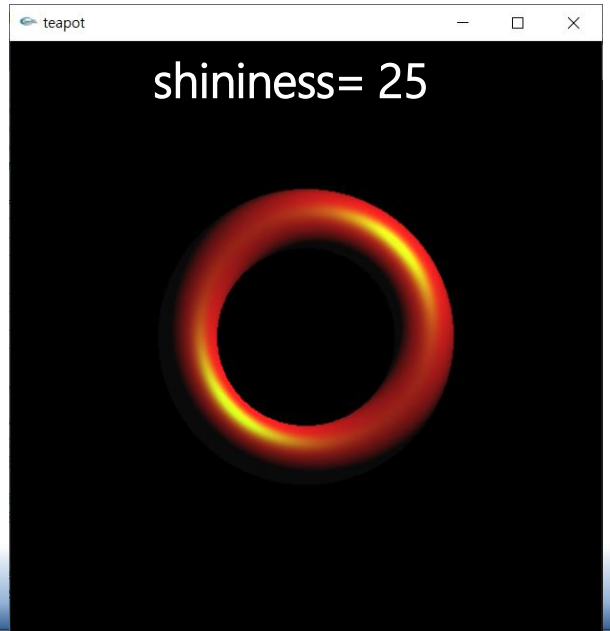
Shininess

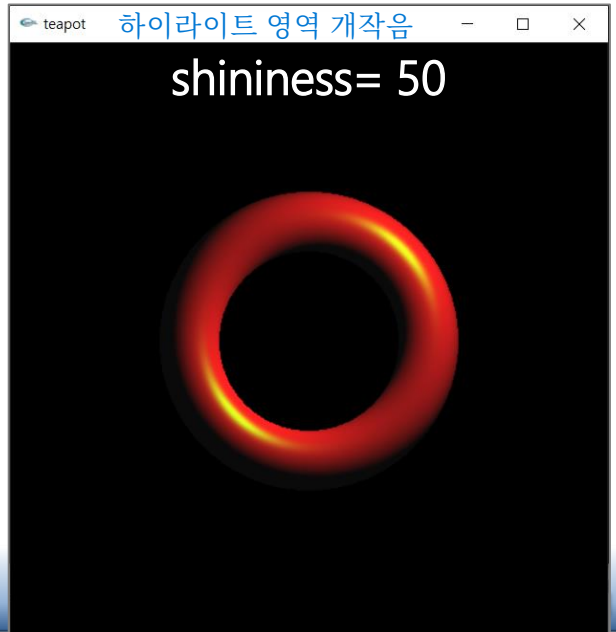
- light & material interaction

```
GLfloat light_ambient[] = { 0.2, 0.2, 0.2, 1.0 };  // gray
GLfloat light_diffuse[]   = { 1, 0.5, 0.5, 1.0 };     // red
GLfloat light_specular[] = { 0, 1, 0, .0 };           // green

GLfloat material_ambient[] = {0.1, 0.1, 0.1, 1.0 };  // gray
GLfloat material_diffuse[]   = { 1, 0.2, 0.2, 1.0 };   // red
GLfloat material_specular[] = { 0, 1, 0, 1.0 };       // green
GLfloat material_shininess[] = { 50.0 };
```

하이라이트 100퍼센트 반사







Emission 발광. 방출

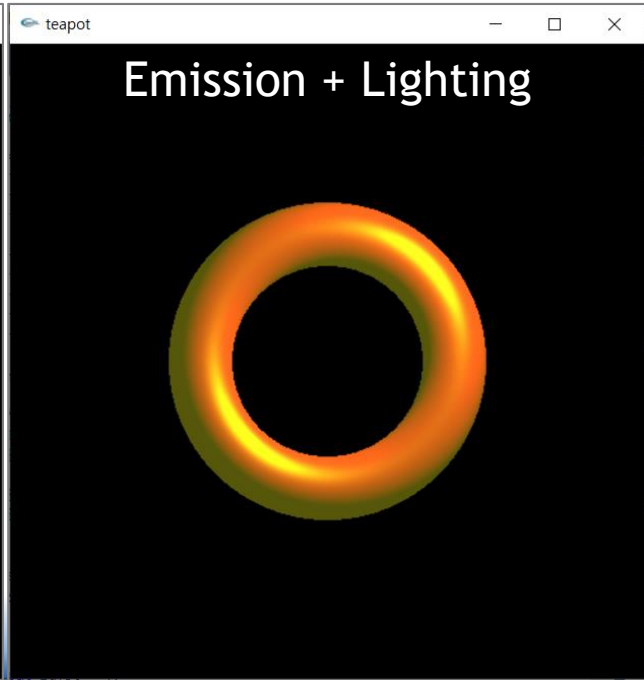
- specify the RGBA emitted light intensity of the material

```
// Material Properties - Emission
GLfloat material_emission[]={1,1,0,1};   // emission : yellow
glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, material_emission);
```

No Emission

material_emission[]
={1, 1, 0, 1};
노란빛방출하도록

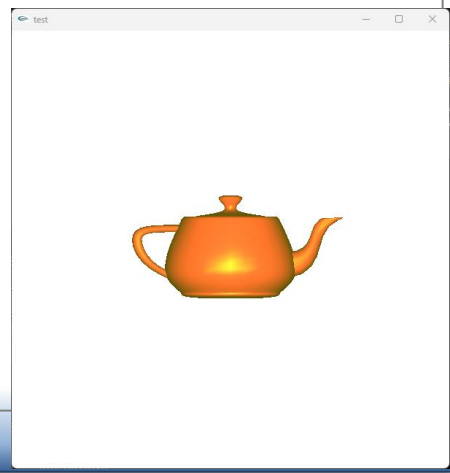
material_emission[]
={0.3, 0.3, 0, 1};



OpenGL Material: *code*

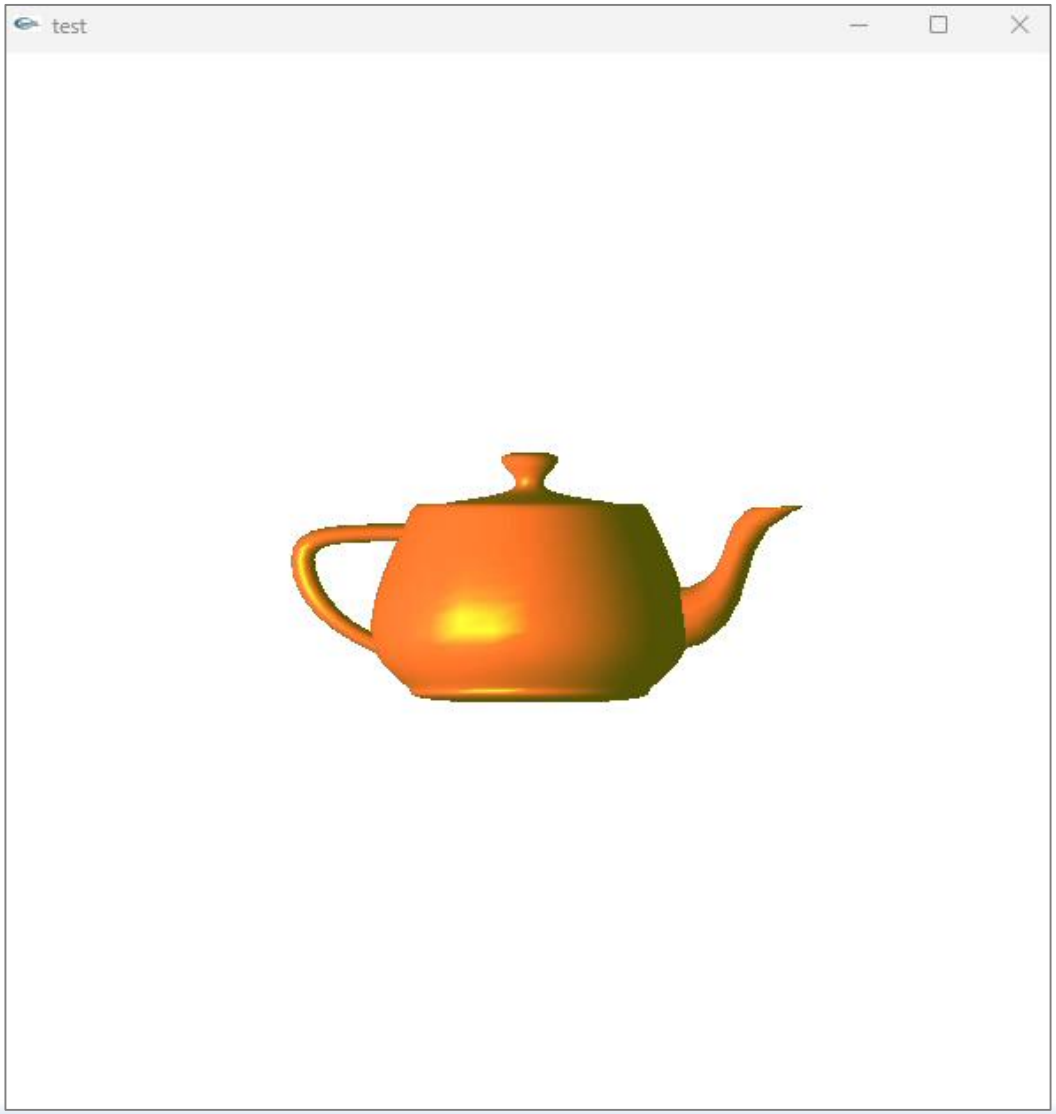
Ambient, diffuse, specular, shininess, emission

```
GLfloat ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat_a[] = { 0.1, 0.1, 0.1, 1.0 };
GLfloat mat_d[] = { 1, 0.2, 0.2, 1.0 };   diffuse red많이 반사
GLfloat mat_s[] = { 0, 1, 0, 1.0 };   yellow 하이라이트
GLfloat low_sh[] = { 50.0 };   값 하나 있는 배열로 만들어서
GLfloat material_emission[] = { 0.3,0.3,0,1 };   yellow 발광

void display()                    fv를 원하지 않는다면
{                                 배열 말고 그냥 선언 ㄱ
    GLfloat position[] = { 0, 0, 2, 1 };

    glClear(GL_COLOR_BUFFER_BIT
        | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // View Volume
    glOrtho(-1, 1, -1, 1, 1, 30);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    // Light Position
    glLightfv(GL_LIGHT0, GL_POSITION, position);

    // Camera
    gluLookAt(0, 0, 2, 0, 0, 0, 0, 1, 0);

    // Light
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);

    // Material
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_a);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_d);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_s);
    glMaterialfv(GL_FRONT, GL_SHININESS, low_sh);
                          여기서 v가 붙음.
    glMaterialfv(GL_FRONT_AND_BACK,
                 GL_EMISSION, material_emission);

    // Teapot
    glColor3f(1, 0, 1);
    glutSolidTeapot(0.3);
    glFlush();
}
```



OpenGL Material: *code*

Ambient, diffuse, specular, shininess, emission



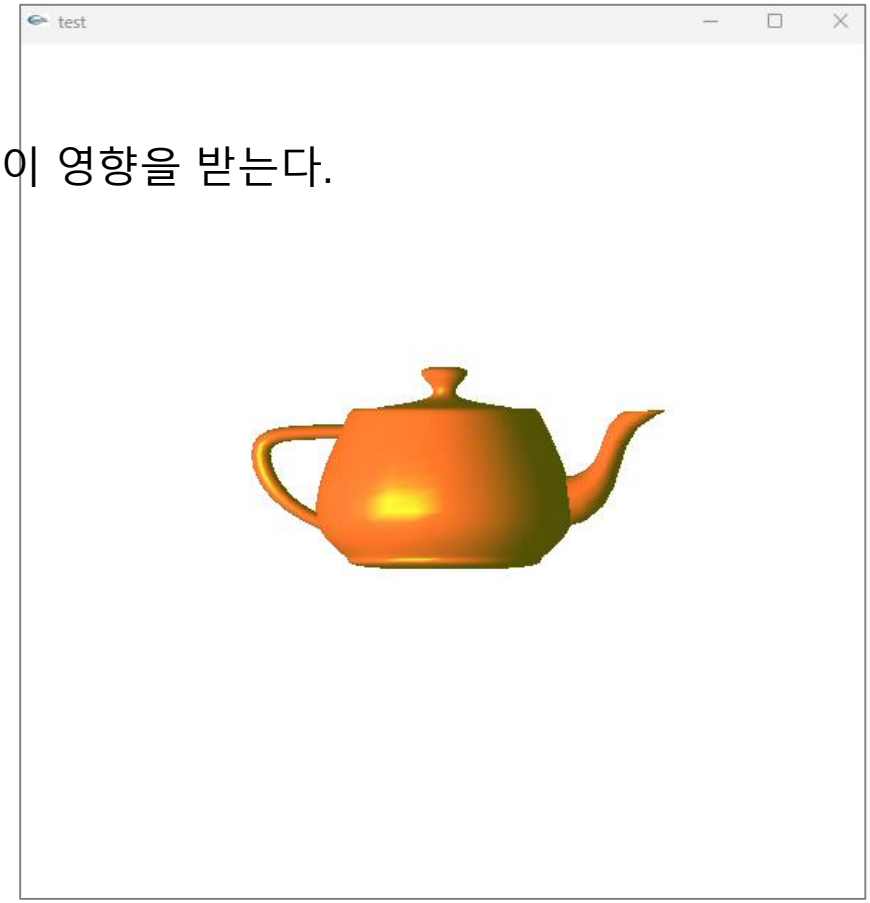
OpenGL Material: *code*

Ambient, diffuse, specular, shininess, emission

```
GLfloat ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat_a[] = { 0.1, 0.1, 0.1, 1.0 };
GLfloat mat_d[] = { 1, 0.2, 0.2, 1.0 };
GLfloat mat_s[] = { 0, 1, 0, 1.0 };
GLfloat low_sh[] = { 50.0 };
GLfloat material_emission[] = { 0.3,0.3,0,1 };

GLfloat position[] = { 0, 0, 2, 1 };
bool dir = true;

void display()
{
    glClear(GL_COLOR_BUFFER_BIT
        | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // View Volume
    glOrtho(-1, 1, -1, 1, 1, 30);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    // Light Position
    if (dir) position[0] += 0.005;
    else position[0] -= 0.005;
    if (position[0] >= 5.0) dir = false;
    if (position[0] <= -5.0) dir = true;
    glLightfv(GL_LIGHT0, GL_POSITION, position);

    // Camera
    gluLookAt(0, 0, 2, 0, 0, 0, 0, 1, 0);

    // Light
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);

    // Material
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_a);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_d);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_s);
    glMaterialfv(GL_FRONT, GL_SHININESS, low_sh);
    glMaterialfv(GL_FRONT_AND_BACK,
        GL_EMISSION, material_emission);

    // Teapot
    glColor3f(1, 0, 1);
    glutSolidTeapot(0.3);
    glFlush();
}
```

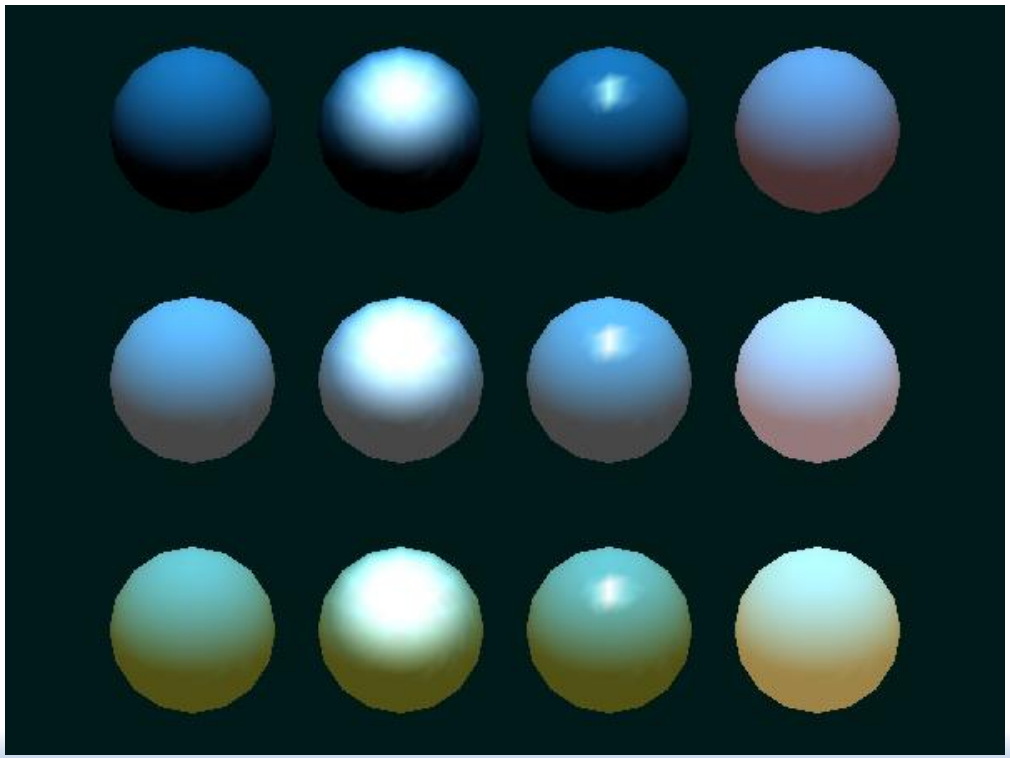
Quiz2. Illumination Model

- 이전 코드를 활용하여 다음 light & Material을 구현하세요.
  - Violet 조명을 받는 주전자
  - SkyBlue 하이라이트 표현
  - 그림보다 더 반짝이는 주전자 표현
  - Green color 발광으로 Violet 주전자 색이 영향을 받는다.



과제. Light & Material

- Try this !
  - 아래 그림과 유사하도록 자신만의 Magic light와 Material을 만들어 보세요~
  - Light를 물체 앞에서 두고 효과적으로 움직여서 specular component가 변화하는 현상을 애니메이션으로 제작하세요.



Shading – OpenGL Light & Material

Thank you~