LEC22: Shading-part2

Ku-Jin Kim

School of Computer Science & Engineering

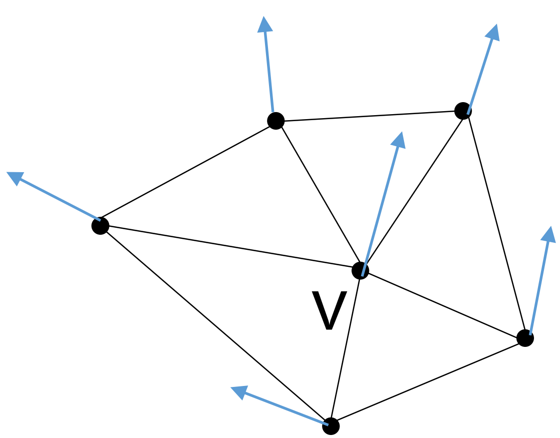
Kyungpook National University

Contents

- Normal vector computation
- Concept of Phong reflection model
- Computation of required vectors

Vertex Normal (1)

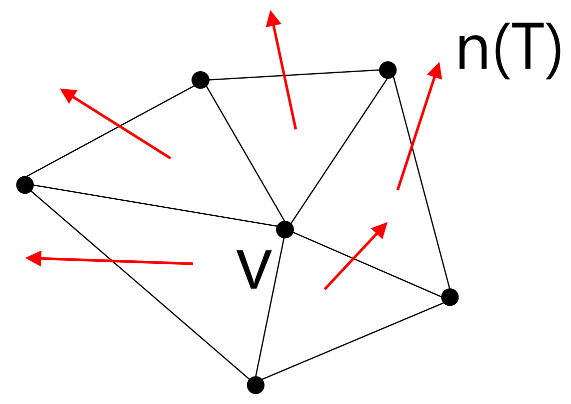
- You've learned about face normal in the previous class
- Vertex normal
  - Normal vector defined for each vertex
  - can be computed by face normals
  - can be used by given vertex normals in 3D model mesh file



Vertex Normal (2)

- Vertex normal computation
  - Basically vertex normal is computed by using 1-ring neighbor face normal

$$\mathbf{n}(v) = \frac{\sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T)}{\left\| \sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T) \right\|}.$$



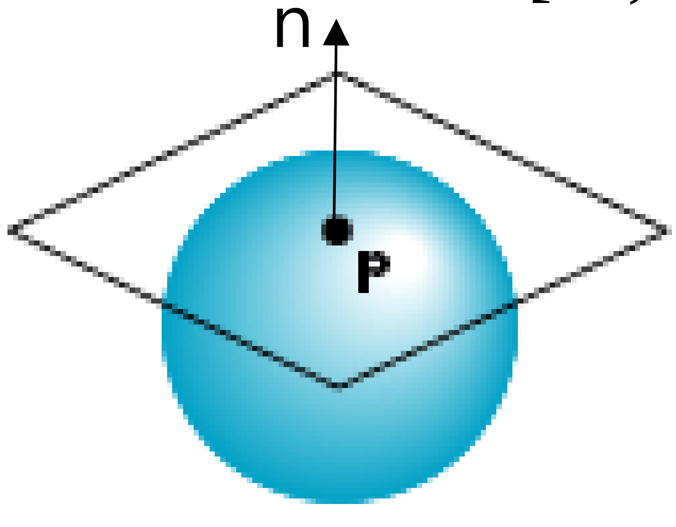
  - $N_1(v)$: 1-ring neighbor face
  - n(T) : face normal
  - α_T : weight
    - constant weight
    - area weight
    - angle weight

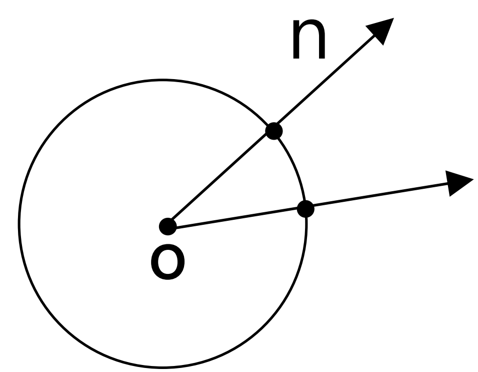
Surface Normal – Implicit Form(1)

- Implicit function $f(x, y, z) = 0$
- Normal vector is given by gradient
- Ex) Sphere

$$f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$$

$$n(x, y, z) = [\partial f/\partial x, \partial f/\partial y, \partial f/\partial z]^T = [2x, 2y, 2z]^T$$



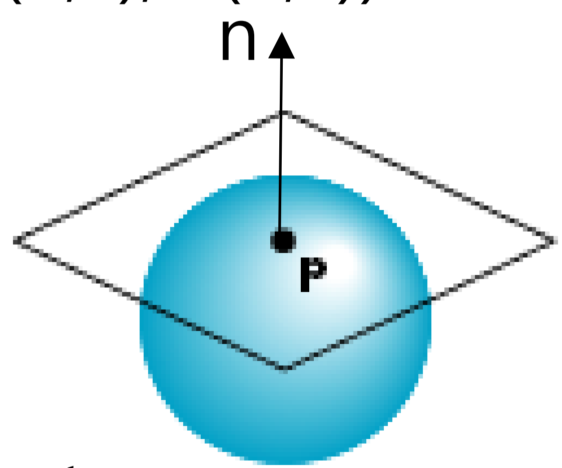


Surface Normal – Implicit Form(2)

- Ex) $f(x, y, z) = x^3 - xy^2 + 2z^2 + 2xy - 4yz = 0$
  - What is the normal vector at (1,1,1) ?

Surface Normal - Parametric Form (1)

- Parametric form: S(u, v) = (x(u,v), y(u,v), z(u,v))
- Ex1) sphere



$$x = x(u,v) = \cos u \sin v$$
$$y = y(u,v) = \cos u \cos v$$
$$z = z(u,v) = \sin u$$

  - Tangent plane determined by vectors

$$\partial \mathbf{p}/\partial u = [\partial x/\partial u, \partial y/\partial u, \partial z/\partial u]T$$
$$\partial \mathbf{p}/\partial v = [\partial x/\partial v, \partial y/\partial v, \partial z/\partial v]T$$

  - Normal given by cross product

$$\mathbf{n} = \partial \mathbf{p}/\partial u \times \partial \mathbf{p}/\partial v$$

Surface Normal - Parametric Form (2)

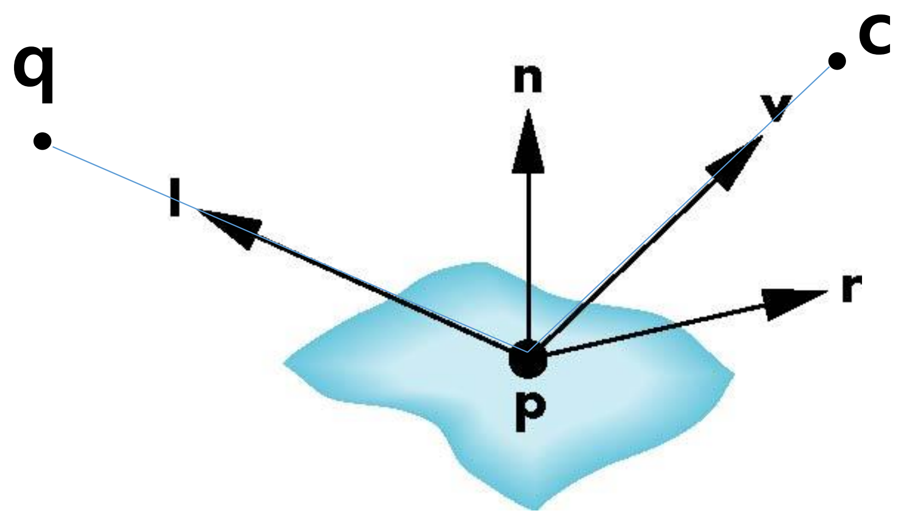
- Ex) $S(u, v) = (u^2v^2 + 3u,\ v^3 + uv + 2,\ u^2 + 3v^2)$
  - What is the normal vector at $S(1.0, 1.0)$?

Concept of Phong Reflection Model

- A simple model that can be computed rapidly
- Three types of material-light interactions
  - Diffuse
  - Specular
  - Ambient

Phong Reflection Model Vectors

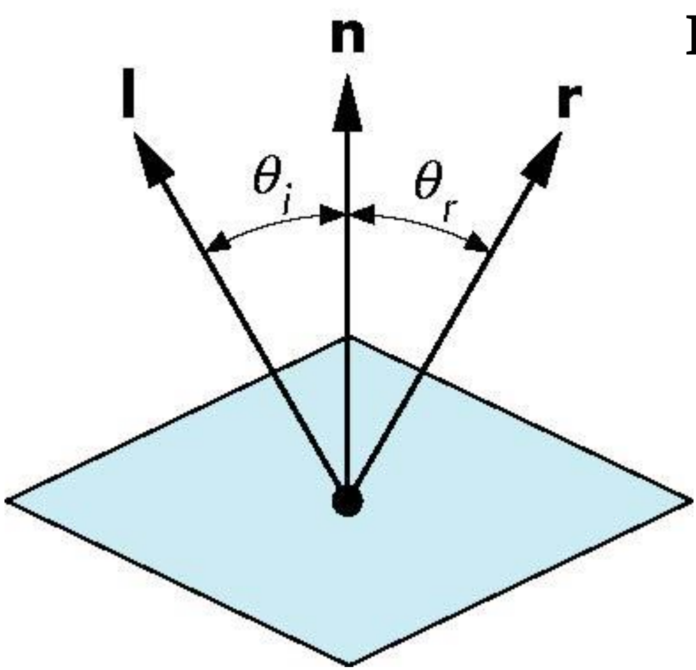
- Camera position: **c**
- Light source position: **q**
- Uses four vectors at **p**
  - To light source: **l**
  - To viewer: **v**
  - Normal: **n**
  - Perfect reflector: **r**
  - || **l** || = || **v** || = || **n** || = || **r** || = 1



Computing Reflection Direction

- Normal is determined by local orientation
- Angle of incidence = angle of reflection
- The three vectors must be coplanar
- Result r vector is ideal reflector

$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$

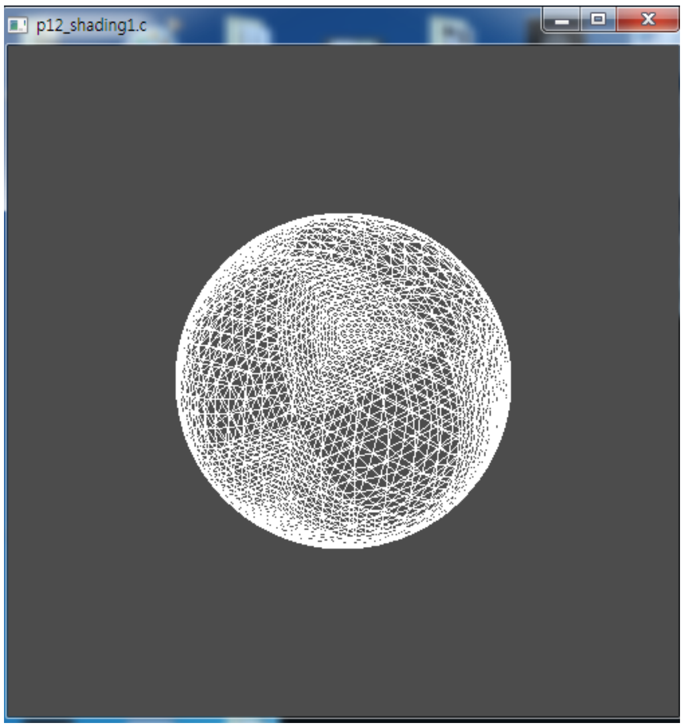


Ambient Reflection

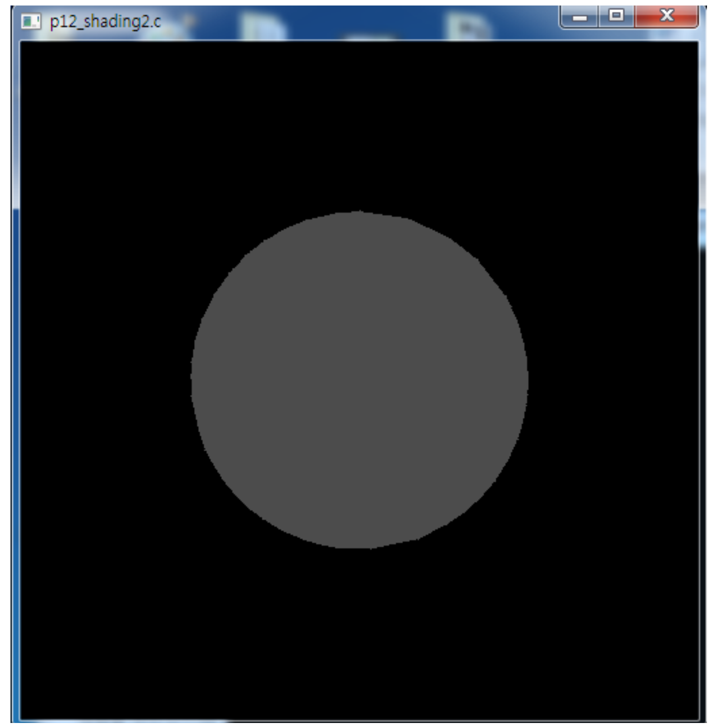
- Intensity of ambient light is the same at every point on the surface
- Some of the light is absorbed and some is reflected
- Ambient light coefficient: L_r, L_g, L_b
  - red, green, blue
- Amount of reflected light is given by ambient reflection coefficient k_r, k_g, k_b
  - red, green, blue

Ambient Reflection Example

Wireframe model



Ambient reflection

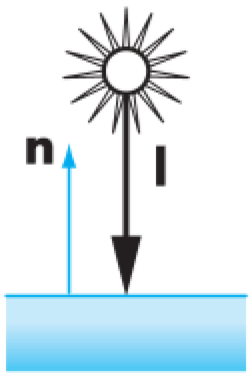


Diffuse Reflection (1)

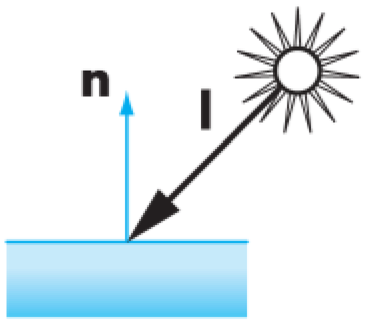
- Lambertian Surface
  - Perfectly diffuse reflector
  - Light scattered equally in all directions
- Amount of light reflected is proportional to the vertical component of incoming light
  - reflected light $\sim \cos \theta_i$
  - $\cos \theta_i = \mathbf{l} \cdot \mathbf{n}$ if vectors are normalized
  - There are also three coefficients, k_r, k_g, k_b that show how much of each color component is reflected

Diffuse Reflection (2)

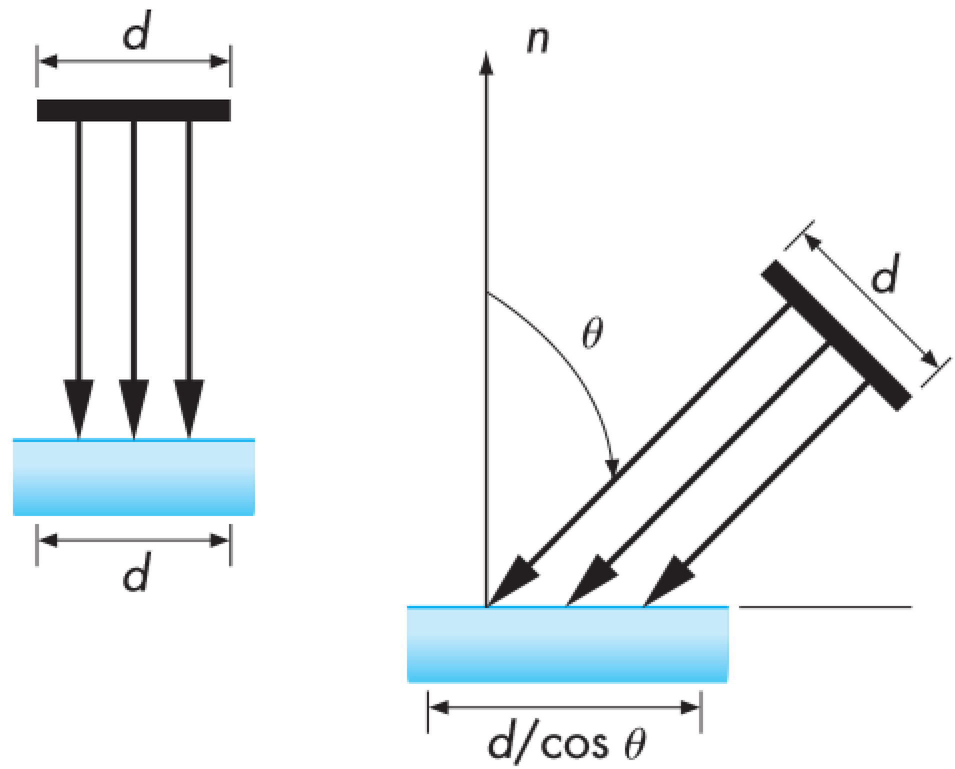
Vertical contributions by Lambert's law



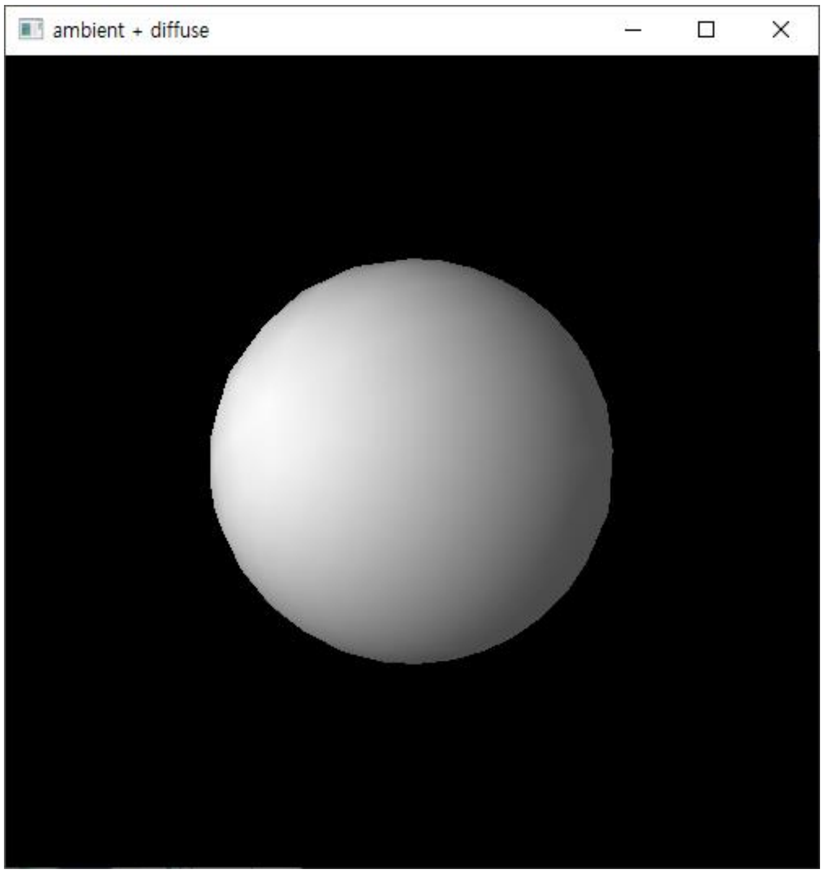
At noon

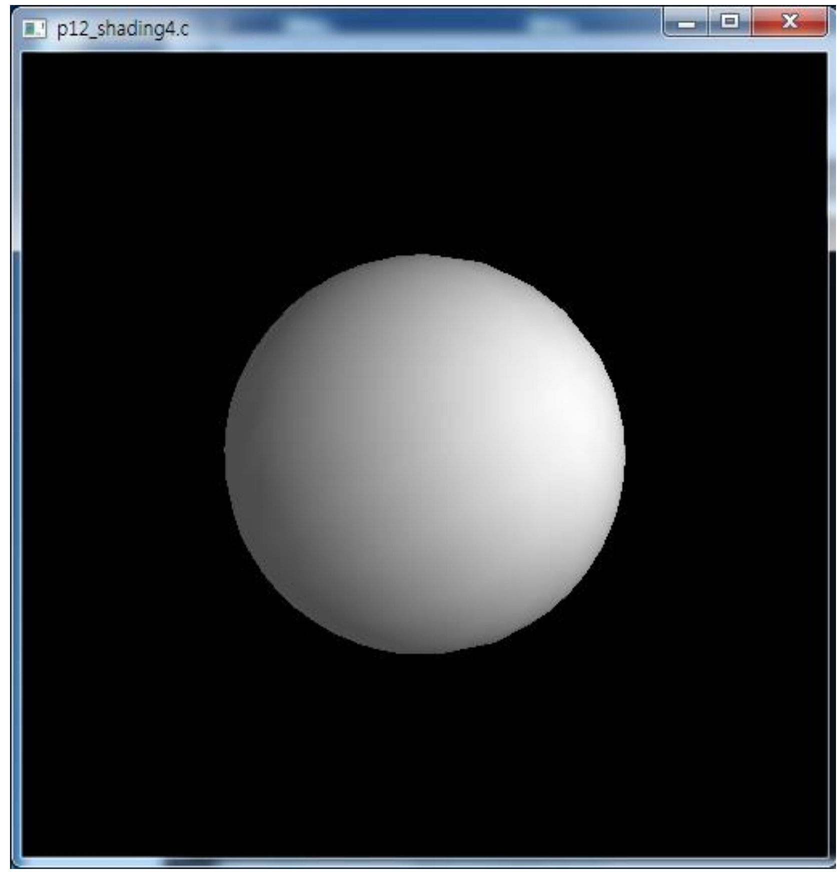


In the afternoon



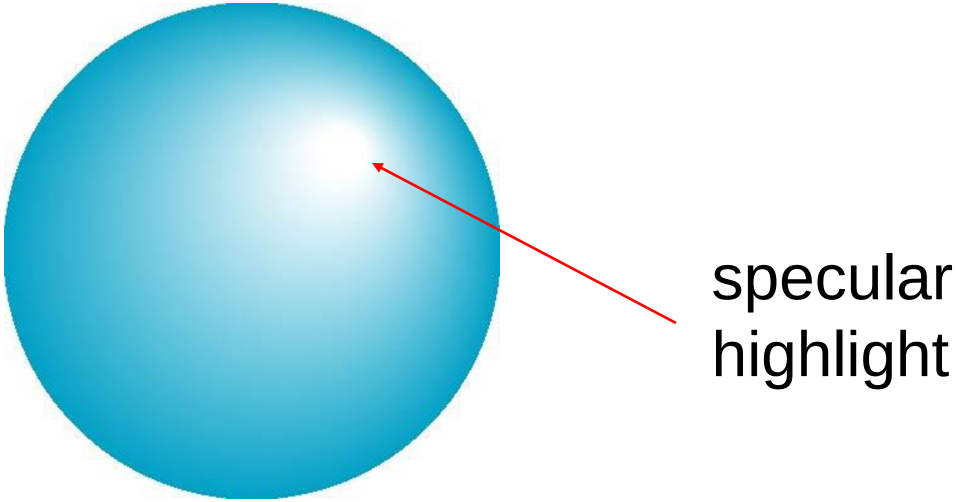
Diffuse Reflection Example



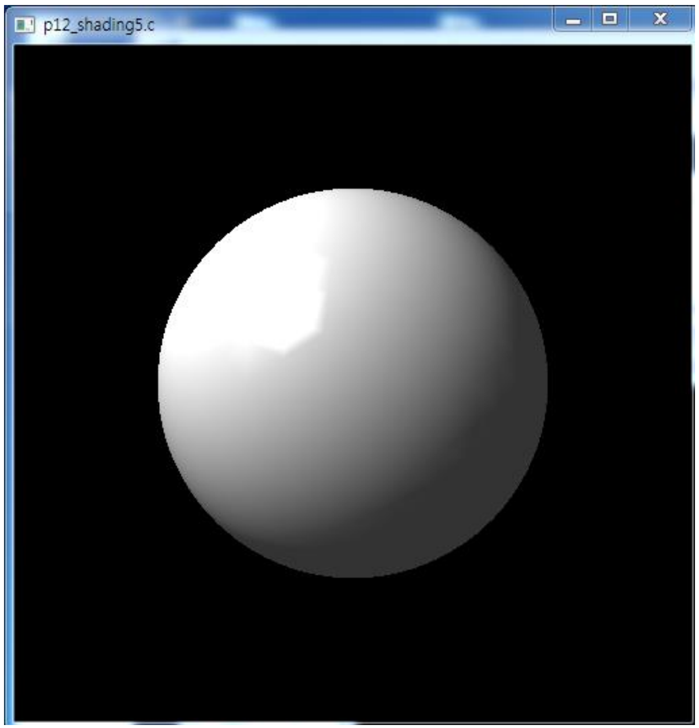


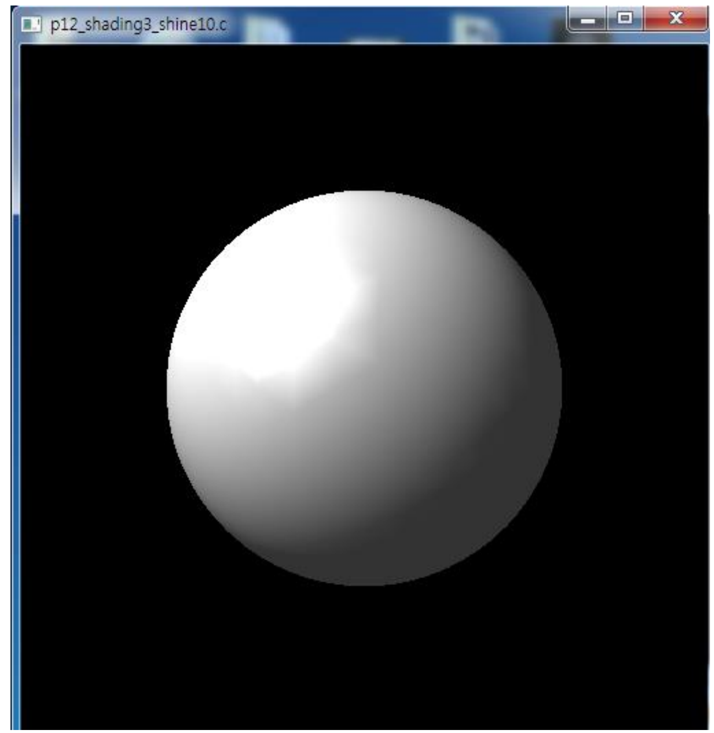
Specular Surfaces

- Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors)
- Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection



Specular Reflection Example





3D Mesh Model File (1)

Ex) teapot.obj

```
530 1024
40.6266   28.3457   -1.10804
40.0714   30.4443   -1.10804
40.7155   31.1438   -1.10804
42.0257   30.4443   -1.10804
...
-0.966742   -0.255752   9.97231e-09
-0.966824   0.255443   3.11149e-08
-0.092052   0.995754   4.45989e-08
0.68205   0.731305   0
...
7   6   1
1   2   7
8   7   2
2   3   8
...
```

3D Mesh Model File (2)

n m //# of vertices and # of faces

$v_0.x$ $v_0.y$ $v_0.z$ // x, y, z coordinate values of v_0

$v_1.x$ $v_1.y$ $v_1.z$ // x, y, z coordinate values of v_1

…

$v_{n-1}.x$ $v_{n-1}.y$ $v_{n-1}.z$ // x, y, z coordinate values of v_{n-1}

$N_0.x$ $N_0.y$ $N_0.z$ // N_0: vertex normal of v_0

$N_1.x$ $N_1.y$ $N_1.z$ // N_1: vertex normal of v_1

…

$N_{n-1}.x$ $N_{n-1}.y$ $N_{n-1}.z$ // N_{n-1}: vertex normal of v_{n-1}

$f_0.i$ $f_0.j$ $f_0.k$ // vertex indices for f_0

$f_1.i$ $f_1.j$ $f_1.k$ // vertex indices for f_1

…

$f_{m-1}.i$ $f_{m-1}.j$ $f_{m-1}.k$ // vertex indices for f_{m-1}

HW#21 3D Model File Input (1)

- Due date: This Friday 6:00pm
- Submit .c code that satisfies with the following requirements.
- Read the data for vertices, vertex normals, indices from the file 'teapot.obj', and fill the related arrays.
  - 'teapot.obj' was given in Lecture note board
  - Its format was explained in the class
  - Compute the bounding box for the vertices.
- Print out the center point of bounding box, and box lengths in x, y, z-directions in a command prompt window as follows:

```
Center:  (    ,    ,    )
Box length in x-direction:
Box length in y-direction:
Box length in z-direction:
```

HW#21 3D Model File Input (2)

- Draw the teapot by setting model matrix, where the view matrix and projection matrix must be given as identity matrices.
- Decide the object color as you want.
- Model matrix can be a concatenation of transformation matrices, and by using the model matrix, the entire teapot must be rendered within the window.
- This is one of the possible execution results of your homework program.