COSC 414/519I: Computer Graphics

2023W2

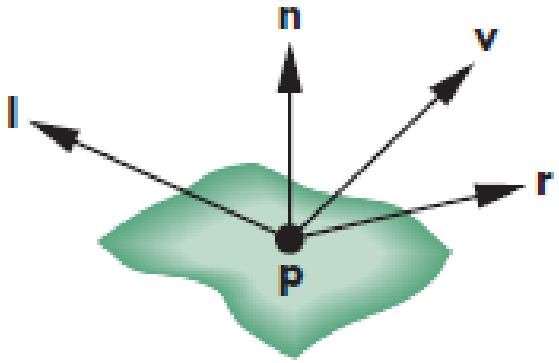
Shan Du

Phong Reflection Model

- Although we could approach light-material interactions through physical models, we have chosen to use a model that leads to efficient computations.
- The reflection model that we present was introduced by Phong and later modified by Blinn.
- It has proved to be efficient and a close enough approximation to physical reality to produce good renderings under a variety of lighting conditions and material properties.

Phong Reflection Model

- The Phong model uses four vectors to calculate a color for an arbitrary point *p* on a surface.

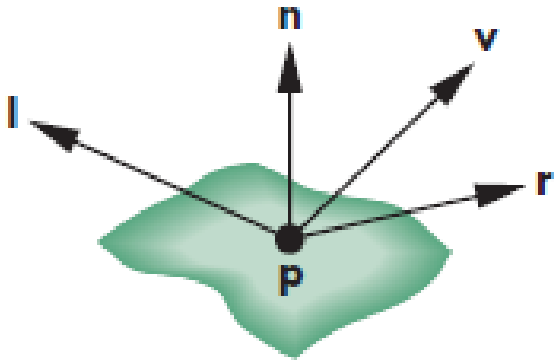


Vectors used by the Phong model.

- If the surface is curved, all four vectors can change as we move from point to point.

Phong Reflection Model

- *n* is the normal at *p*.
- *v* is the direction from *p* to COP.
- *l* is the direction of the light source.
- *r* is the direction of the reflection. *r* is determined by *n* and *l*.



Vectors used by the Phong model.

Phong Reflection Model

- The Phong model supports three types of material-light interactions – ambient, diffuse, and specular.
- Suppose we have a set of point sources, we assume each source can have separate ambient, diffuse, and specular components for each of the three primary colors.
- Our light-source model has ambient, diffuse, and specular terms.

Phong Reflection Model

- We need nine coefficients to characterize these terms at any point *p* on the surface. We can place these nine coefficients in a 3×3 illumination matrix for the *i*th light source:

$$L_i = \begin{bmatrix} L_{ira} & L_{iga} & L_{iba} \\ L_{ird} & L_{igd} & L_{ibd} \\ L_{irs} & L_{igs} & L_{ibs} \end{bmatrix}$$

- In practice, we use

```
light_i_ambient = vec4();
light_i_diffuse = vec4();
light_i_specular = vec4();
```

Phong Reflection Model

- For each light term, e.g., L_{ird}, we can compute a reflection terms R_{ird}, and the intensity at *p* is $R_{ird}L_{ird}$.
- The value of R_{ird} depends on the material properties, the orientation of the surface, the direction of the light source, and the distance between the light source and the surface.

Phong Reflection Model

- For each point, we have nine coefficients in a matrix that represent the reflection terms.

$$R_i = \begin{bmatrix} R_{ira} & R_{iga} & R_{iba} \\ R_{ird} & R_{igd} & R_{ibd} \\ R_{irs} & R_{igs} & R_{ibs} \end{bmatrix}$$

We can compute the contribution of each color source by adding the ambient, diffuse, and specular components.

Phong Reflection Model

- For example, the red intensity that we see at *p* from source *i* is

$$I_{ir} = R_{ira}L_{ira} + R_{ird}L_{ird} + R_{irs}L_{irs}$$
$$= I_{ira} + I_{ird} + I_{irs}$$

- We obtain the total intensity by adding the contributions of all sources and possibly a global ambient terms. Thus, the red term is

$$I_r = \sum_i (I_{ira} + I_{ird} + I_{irs}) + I_{ar}$$

Phong Reflection Model

- We can simplify our notation by observing that the necessary computations are the same for each source and for each primary color. They differ depending on whether we are considering the ambient, diffuse, or specular terms. Hence, we can omit the subscripts *i*, *r*, *g*, and *b*. We write

$$I = I_a + I_d + I_s = R_a L_a + R_d L_d + R_s L_s$$

with the global ambient term can be added at the end.

Ambient Reflection

- The intensity of ambient light I_a is the same at every point on the surface.
- Some of this light is absorbed and some is reflected.
- The amount reflected is given by the ambient reflection coefficient, $R_a = k_a$, where $0 \leq k_a \leq 1$. Thus, $I_a = k_a L_a$.

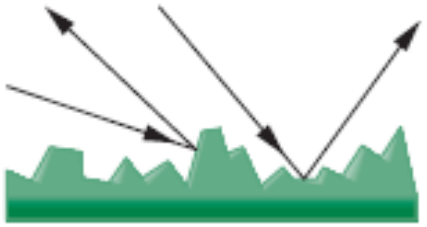
Here L_a can be any of the individual light sources, or it can be a global ambient term.

Diffuse Reflection

- A perfectly diffuse reflector

scatters the light that it reflects

equally in all directions.

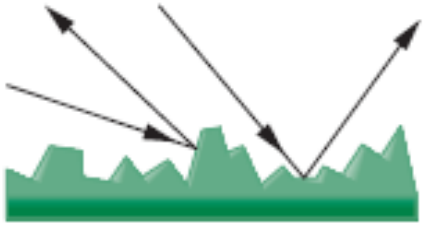


Rough surface.

- Such a surface appears the same to all viewers.
- The reflection of light depends both on the material and on the position of the light source relative to the surface.

Diffuse Reflection

- Diffuse reflections are characterized by rough surfaces.

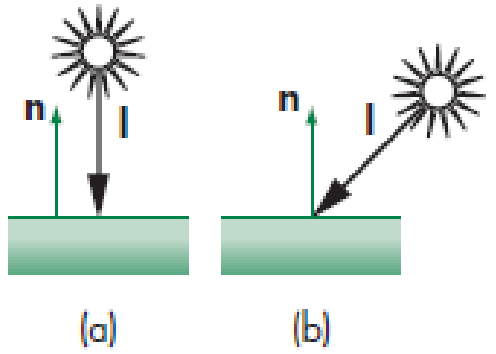


Rough surface.

- Perfectly diffuse surfaces are so rough that there is no preferred angle of reflection.
- Such surface, sometimes called Lambertian surfaces, can be modeled mathematically with Lambert's law.

Diffuse Reflection

- Consider a diffuse planar surface illuminated by the sun. The surface is brightest at noon and dimmest at dawn and dusk, because according to Lambert's law, we see only the vertical component of the incoming light.



Illumination of a diffuse surface. (a) At noon. (b) In the afternoon.

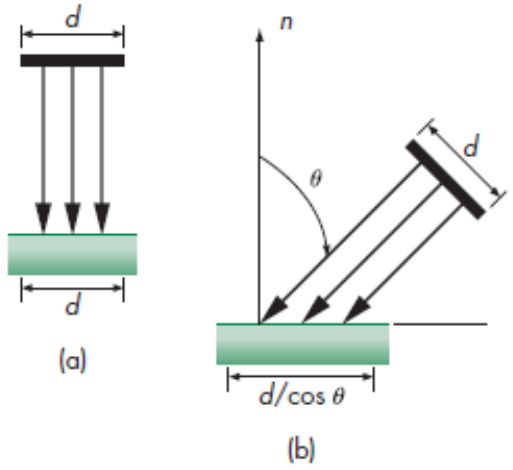
Diffuse Reflection

- Lambert's law

states that

$R_d \propto cos\theta$

where θ is the angle



Vertical contributions by Lambert's law. (a) At noon. (b) In the afternoon.

between the normal *n* at the point of interest and the direction of the light source *l*. If both *l* and *n* are unit-length vectors, then

$$cos\theta = \mathrm{l} \cdot n$$

Diffuse Reflection

- If we add in a reflection coefficient k_d representing the fraction of incoming diffuse light that is reflected, we have the diffuse reflection terms:

$$I_d = k_d(l \cdot n)L_d$$

- If we incorporate a distance term, *d*,

$$I_d = \frac{k_d}{a + bd + cd^2}(l \cdot n)L_d$$

Shading Due to Directional Light and Its Diffuse Reflection

- Surface color is determined by light direction and the orientation of the surface it strikes when considering diffuse reflection.
- The calculation of the color due to directional light is easy because its direction is constant.
- The formula for calculating the color of a surface by diffuse reflection is

$$\langle surface\ color\ by\ diffuse\ reflection \rangle = \langle light\ color \rangle \times \langle base\ color\ of\ surface \rangle \times \cos\theta$$

Shading Due to Directional Light and Its Diffuse Reflection

- The following three pieces of information are used:
  - The color of the light source (directional light)
  - The base color of the surface
  - The angle (θ) between the light and the surface

Calculating Diffuse Reflection Using the Light Direction and the Orientation of a Surface

- cos θ is defined as follows:

$$\cos\theta = \langle light\ direction \rangle \bullet \langle orientation\ of\ a\ surface \rangle$$

- Therefore

$$\langle surface\ color\ by\ diffuse\ reflection \rangle =$$
$$\langle light\ color \rangle \times \langle base\ color\ of\ surface \rangle \times$$
$$(\langle light\ direction \rangle \bullet \langle orientation\ of\ a\ surface \rangle)$$

Calculating Diffuse Reflection Using the Light Direction and the Orientation of a Surface

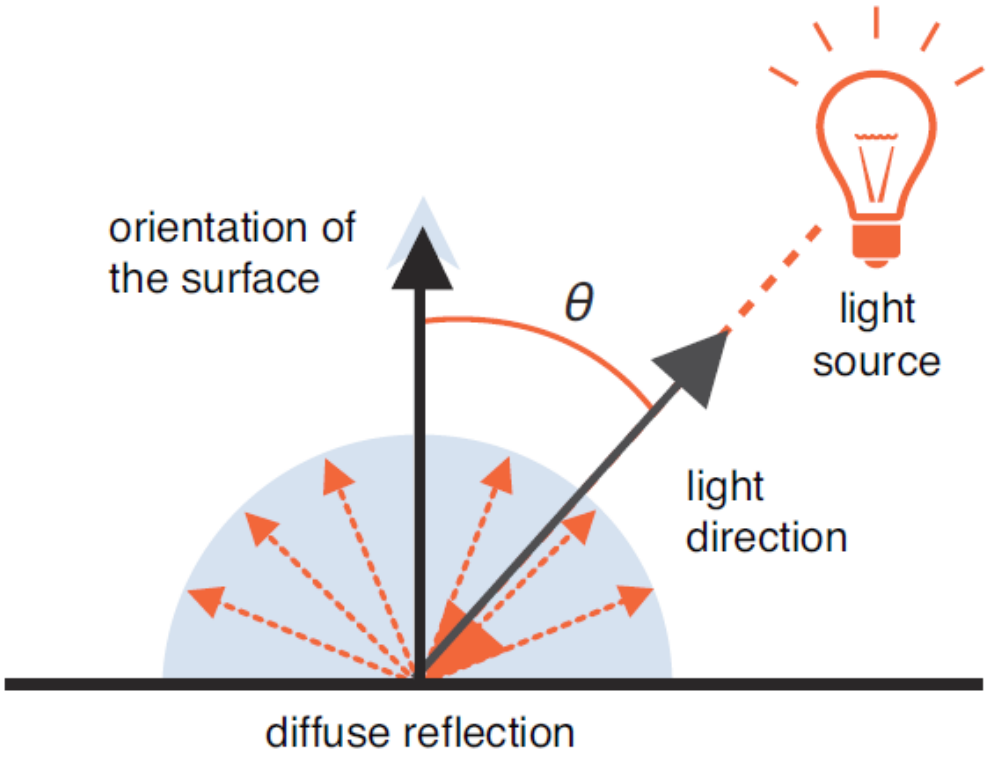


Figure 8.6 The light direction is from the reflecting surface to the light source

The Orientation of a Surface: What is the Normal?

- A surface has two normal:

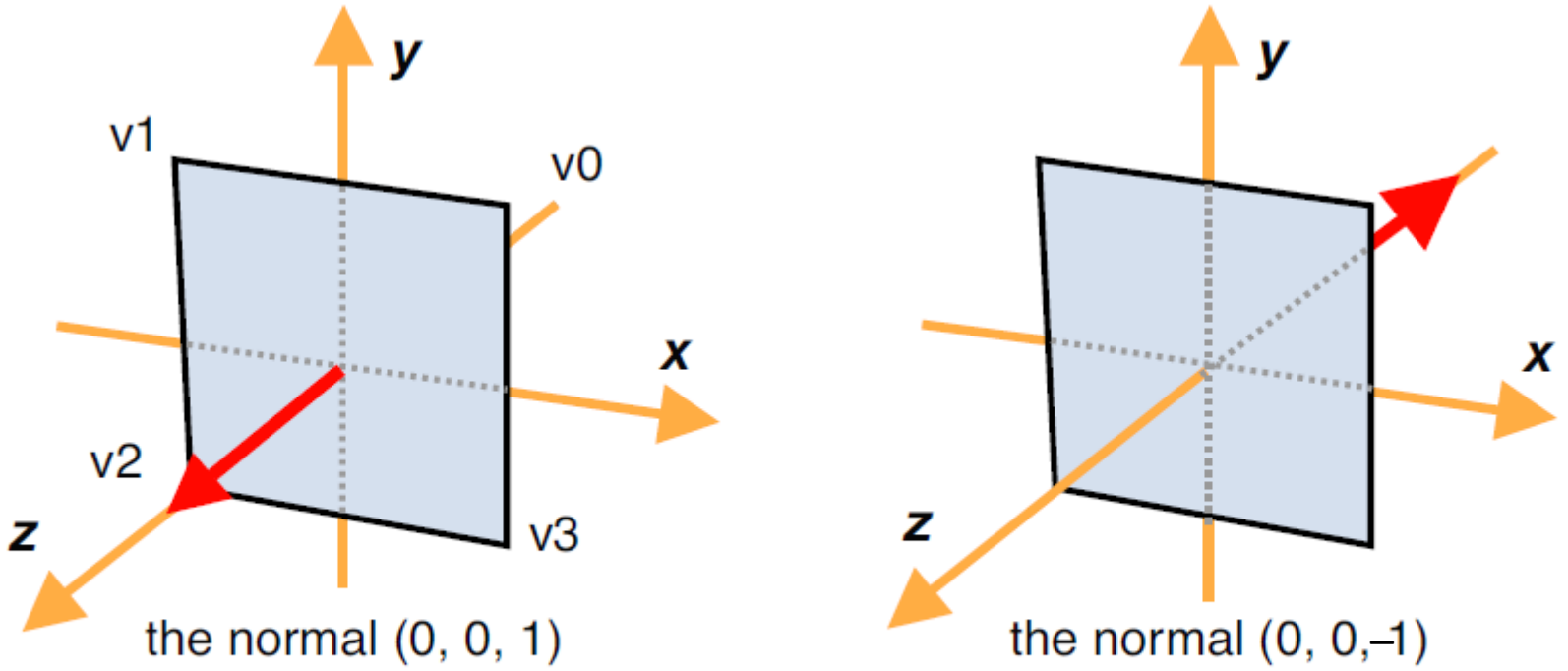
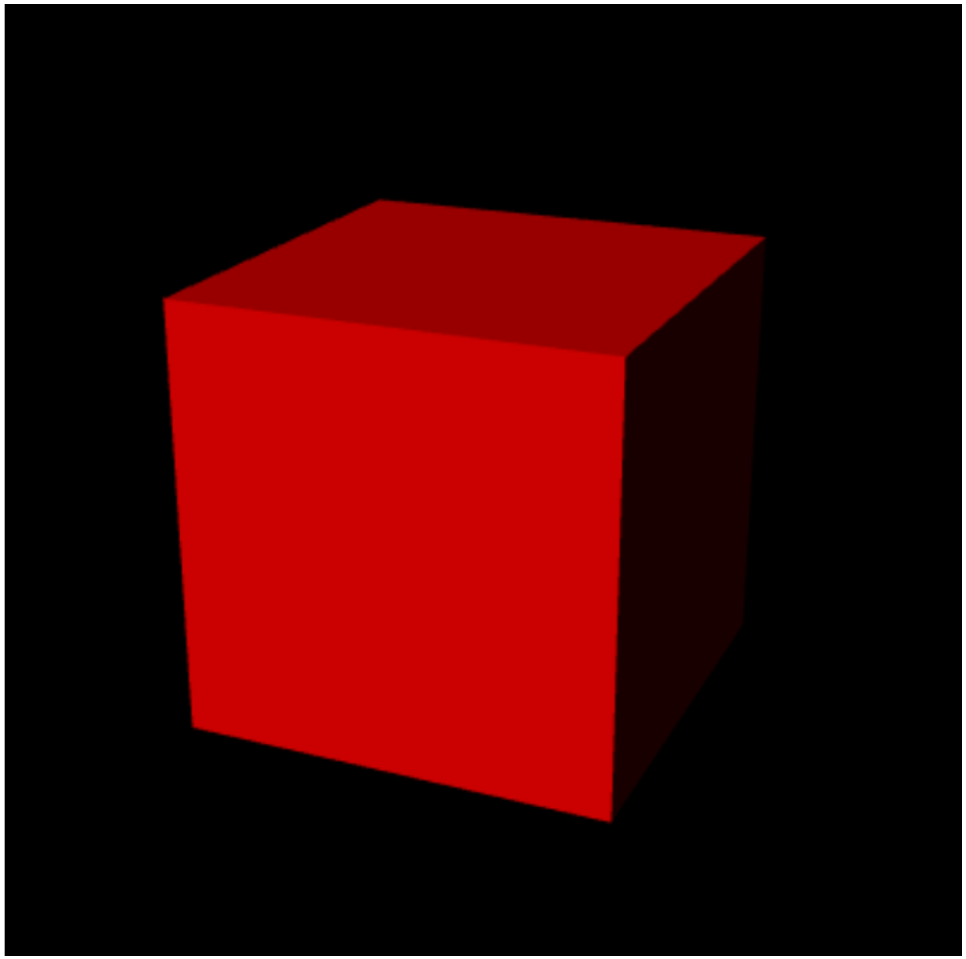


Figure 8.7 Normals

Lighted Cube



Lighted Cube

```
// Vertex shader program
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n' +
  'attribute vec4 a_Color;\n' +
  'attribute vec4 a_Normal;\n' +       // Normal
  'uniform mat4 u_MvpMatrix;\n' +
  'uniform vec3 u_LightColor;\n' +     // Light color
  'uniform vec3 u_LightDirection;\n' + // Light direction (in the world coordinate, normalized)
  'varying vec4 v_Color;\n' +
  'void main() {\n' +
  '  gl_Position = u_MvpMatrix * a_Position ;\n' +
  // Make the length of the normal 1.0
  '  vec3 normal = normalize(a_Normal.xyz);\n' +
  // Dot product of the light direction and the orientation of a surface (the normal)
  '  float nDotL = max(dot(u_LightDirection, normal), 0.0);\n' +
  // Calculate the color due to diffuse reflection
  '  vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;\n' +
  '  v_Color = vec4(diffuse, a_Color.a);\n' +
  '}\n';
```

Lighted Cube

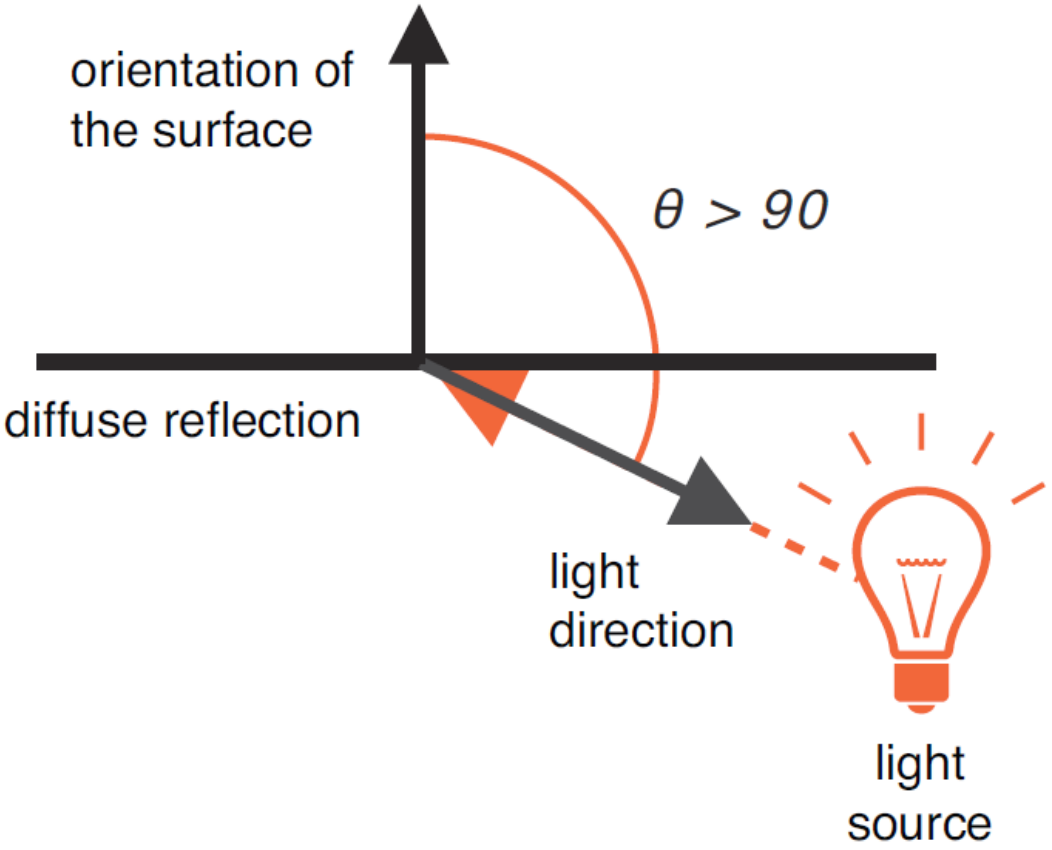
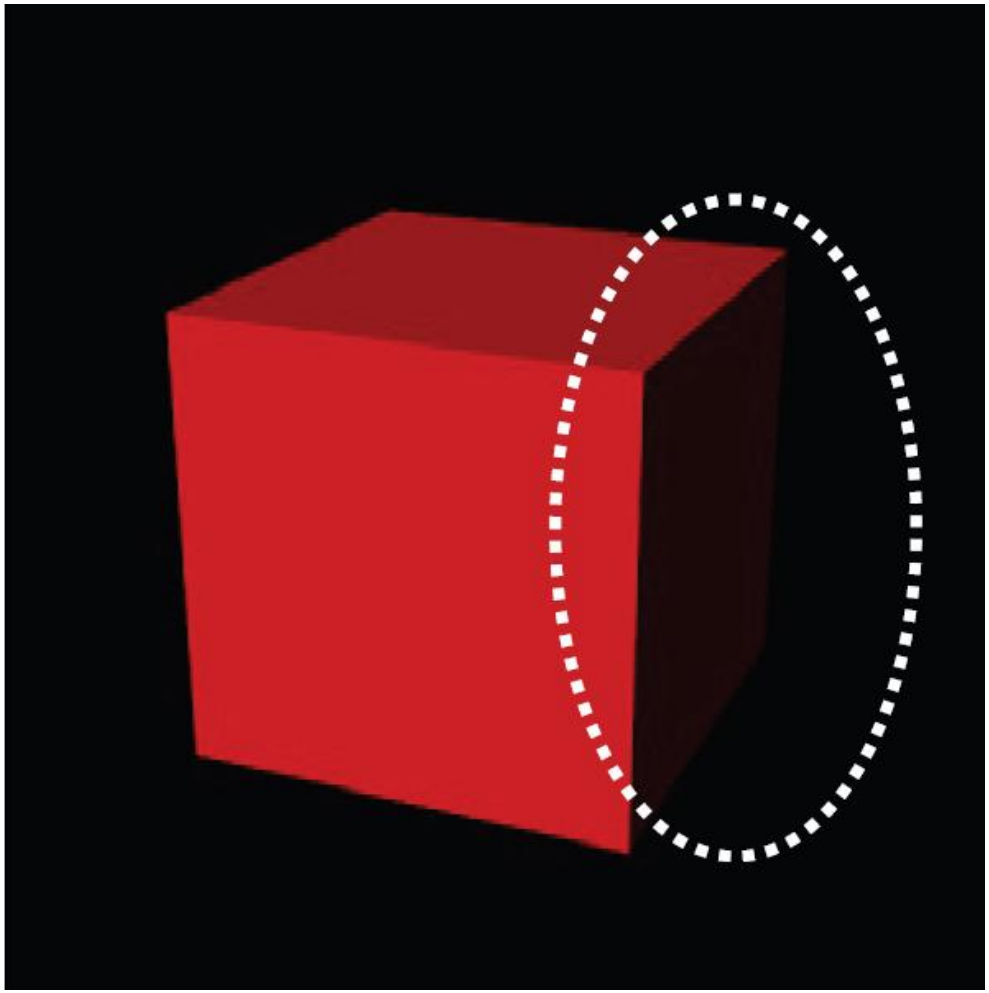


Figure 8.11 A normal and light in case θ is greater than 90 degrees

Lighted Cube

```
function main() {
……
 // Set the light color (white)
  gl.uniform3f(u_LightColor, 1.0, 1.0, 1.0);
  // Set the light direction (in the world coordinate)
  var lightDirection = new Vector3([0.5, 3.0, 4.0]);
  lightDirection.normalize();     // Normalize
  gl.uniform3fv(u_LightDirection, lightDirection.elements);
……
 var normals = new Float32Array([    // Normal
    0.0, 0.0, 1.0,   0.0, 0.0, 1.0,   0.0, 0.0, 1.0,   0.0, 0.0, 1.0,  // v0-v1-v2-v3 front
    1.0, 0.0, 0.0,   1.0, 0.0, 0.0,   1.0, 0.0, 0.0,   1.0, 0.0, 0.0,  // v0-v3-v4-v5 right
    0.0, 1.0, 0.0,   0.0, 1.0, 0.0,   0.0, 1.0, 0.0,   0.0, 1.0, 0.0,  // v0-v5-v6-v1 up
   -1.0, 0.0, 0.0,  -1.0, 0.0, 0.0,  -1.0, 0.0, 0.0,  -1.0, 0.0, 0.0,  // v1-v6-v7-v2 left
    0.0,-1.0, 0.0,   0.0,-1.0, 0.0,   0.0,-1.0, 0.0,   0.0,-1.0, 0.0,  // v7-v4-v3-v2 down
    0.0, 0.0,-1.0,   0.0, 0.0,-1.0,   0.0, 0.0,-1.0,   0.0, 0.0,-1.0   // v4-v7-v6-v5 back
  ]);
```

Add Shading Due to Ambient Light



Add Shading Due to Ambient Light

- Because ambient light models the light that hits an object from all directions with constant intensity, the surface color due to the reflection is determined only by the light color and the base color of the surface.

$$\langle \textit{surface color by ambient reflection} \rangle = \langle \textit{light color} \rangle \times \langle \textit{base color of surface} \rangle$$

- Then

$$\langle \textit{surface color by diffuse and ambient reflection} \rangle = \langle \textit{surface color by diffuse reflection} \rangle + \langle \textit{surface color by ambient reflection} \rangle$$

Lighted Cube_ambient

```
// Vertex shader program
var VSHADER_SOURCE =
……
'uniform vec3 u_AmbientLight;\n' +   // Color of an ambient light
……
 'void main() {\n' +
……
 // Calculate the color due to ambient reflection
  '  vec3 ambient = u_AmbientLight * a_Color.rgb;\n' +
    // Add the surface colors due to diffuse reflection and ambient reflection
  '  v_Color = vec4(diffuse + ambient, a_Color.a);\n' +
…… '}\n';
function main() {
……
var u_AmbientLight = gl.getUniformLocation(gl.program, 'u_AmbientLight');
……
// Set the ambient light
  gl.uniform3f(u_AmbientLight, 0.2, 0.2, 0.2);
…… }
```

Lighted Cube_ambient