Lights and Materials

Woo, Neider et Al., Chapter 4

Enabling Lights

- To enable lights from a single source (Light0):

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```

Defaults:

Light 0: white light (1.0, 1.0, 1.0, 1.0) in RGBA
diffuse and specular components

Other lights: black (0.0, 0.0, 0.0, 1.0)

Position: (0.0, 0.0, 1.0, 0.0) in HC, w=0.0 means ∞ distance,
i.e. the HCs represent a direction, not a point

- Once lighting is enabled, colors assigned by **glColor*( )** are no longer used

Normals

- When specifying polygon vertices, we must supply the normal vectors to each vertex:

  void glNormal3*(TYPE dx, TYPE dy, TYPE dz)
  void glNormal3*v(TYPE norm);

- Lighting calculations require unitary normals
- Scaling changes the length of the normals
- To enable automatic normalization, use:

  glEnable(GL_NORMALIZE);

Specifying a Distant Light Source

- Main functions to set scalar or vector parameters for OpenGL light sources:

```
void glLight*(GLenum light,
              GLenum param, TYPE value);
void glLight*v(GLenum light,
              GLenum param, TYPE *value);
```

light: GL_LIGHT0, GL_LIGHT1, GL_LIGHT2,…

param: GL_POSITION, GL_DIFFUSE,
GL_AMBIENT, GL_SPECULAR, others…

Specifying a Distant Light Source

- **Example:** source position never changes

```
void init( ) {
    GLfloat light_pos[] = {1.0, 2.0, 3.0, 1.0};
    // ---- Rest of init( ) ----
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity( );
    glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
}
```

Specifying a Distant Light Source

- **Example:** specifying light source parameters

$$I = I_{amb} + I_{diff} + I_{spec} = I_a K_a + I_d K_d N \cdot L + I_s K_s (R \cdot V)^n$$

```
GLfloat position0[] = {1.0, 1.0, 1.0, 0.0};
GLfloat diffuse0[] = {1.0, 0.0, 0.0, 1.0};        // Id term - Red
GLfloat specular0[] = {1.0, 1.0, 1.0, 1.0};       // Is term - White
GLfloat ambient0[] = {0.1, 0.1, 0.1, 1.0};        // Ia term - Gray

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

glLightfv(GL_LIGHT0, GL_POSITION, position0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambient0);
```

Specifying a Spot Light

- Spot light is a close light source with direction, angle and exponential drop off

void glLight*(GLenum light, GLenum param, TYPE value);
void glLight*v(GLenum light, GLenum param, TYPE *value);

param: GL_SPOT_DIRECTION
GL_SPOT_CUTOFF (default 180^{o})
GL_SPOT_EXPONENT (default 0)

GL_POSITION has the form (x, y, z, 1.0)

Specifying a Spot Light

- Attenuation with distance generates a softer and more realistic image:

$$f_{att}(d) = \frac{1}{a + bd + cd^2}$$

d is the distance computed by OpenGL

a = GL_CONSTANT_ATTENUATION (default 1.0)
b = GL_LINEAR_ATTENUATION (default 0.0)
c = GL_QUADRATIC_ATTENUATION (default 0.0)

Default is no attenuation: a=1, b=0, c=0

Specifying a Material

- Once lighting is enabled, colors are no longer used
- Main functions to set scalar or vector parameters

for OpenGL faces:

void glMaterial*(GLenum face,
GLenum param, TYPE value);

void glMaterial*v(GLenum face,
GLenum param, TYPE *value);

face: GL_FRONT,
GL_BACK,
GL_FRONT_AND_BACK

Specifying a Material

$$I = I_{amb} + I_{diff} + I_{spec} = I_a K_a + I_d K_d N \cdot L + I_s K_s (R \cdot V)^n$$

param: GL_AMBIENT // **K_a** term

GL_DIFFUSE // **K_d** term

GL_SPECULAR // **K_s** term

GL_SHININESS // exponent **n**

GL_AMBIENT_AND_DIFFUSE // **K_a** = **K_d**

GL_EMISSION // No light calculations

// simulates sources

Specifying a Material

- **Example:**

```
typedef struct materialStruct {
      GLfloat Ka [4];
      GLfloat Kd[4];
      GLfloat Ks[4];
      GLfloat n;
} materialStruct;

void set_material(materialStruct *mat) {
     glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, mat->Ka);
     glMaterialfv (GL_FRONT_AND_BACK, GL_DIFFUSE, mat->Kd);
     glMaterialfv (GL_FRONT_AND_BACK, GL_SPECULAR, mat->Ks);
     glMaterialfv (GL_FRONT_AND_BACK, GL_SHININESS, mat->n);
      }
```

Specifying a Material

- **Example:**

```
materialStruct brass {
      {0.33, 0.22, 0.03, 1.0},          // Ka
      {0.78, 0.57, 0.11, 1.0},          // Kd
      {0.99, 0.91, 0.81, 1.0},          // Ks
      27.8                              // n
};

materialStruct red_shiny_plastic {
      {0.3, 0.0, 0.0, 1.0},             // Ka
      {0.6, 0.0, 0.0, 1.0},             // Kd
      {0.8, 0.6, 0.6, 1.0},             // Ks
      100.0                             // n
};

set_material(red_shiny_plastic);        // Usage
```

Controlling Lighting Calculations

- Light for back faces will not be computed unless we set:

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDED, GL_TRUE);
```

- If all sources are off, but we still want ambient light:

```
GLfloat glob_ambient[] = {0.2, 0.2, 0.2, 1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, glob_ambient);
```

- By default, the viewer is at infinite distance. In this case, calculation for specular component is done once for each face. When this is not the case, set:

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
```

Smooth Shading

- Flat shading (no interpolation):
  glShadeModel(GL_FLAT);

- Phong shading:
  glShadeModel(GL_SMOOTH);

NOTE: Shading may be unsatisfactory when polygons are too big. If this is the case, divide into smaller polygons