MAE 473/573

Lecture 25
Shading and Lighting in OpenGL
Lighting requires 3 things

- light source
- material properties
- surface normals
Light Sources

- OpenGL computes 4 types of lighting
  - ambient
  - diffuse
  - specular
  - emissive
- OpenGL computes all 4 and adds them
Ambient Light

- No incoming direction
- No outgoing (bouncing) direction
- baseline lighting in a room or outdoors
Diffuse Light

- Comes from specific direction
- no outgoing direction (bouncing)
- for example, a flood light
Specular Light

- Specific incoming direction
- Specific bounce direction
- for example, a laser beam on a mirror
Emmissive Light

- A specific light source, like a headlight
- OpenGL doesn’t use it in lighting calculations
- For effect only
How does OpenGL computing appearance?

- Combine light sources with material properties
  - ambient properties with ambient light component
  - diffuse light with diffuse component
  - specular light with specular component
A simple example - a lit cube

- enable lighting
- enable a specific light
- assign light properties
- assign light-specific material properties
Enable Lighting

- Usually done globally (done once)

```c
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```
Assign Light Properties

GLfloat light_ambient[] = {.5, .5, .5, 1.0};
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_position[] = {50, 50, 50};
...
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
Assign light-specific material properties

GLfloat Material_Specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat Material_Ambient[] = {1.0, 0.0, 0.0, 0.0};
GLfloat Material_Shininess = 50;

glMaterialfv(GL_FRONT, GL_SPECULAR, Material_Specular);

glMaterialfv(GL_FRONT, GL_AMBIENT, Material_Ambient);

glMaterialf(GL_FRONT, GL_SHININESS, Material_Shininess);
Surface Normals Define Reflection Behavior

```c
glBegin(GL_QUADS);
// back face
glNormal3d(0,0,-1);
glVertex3d(-WIDTH/2,-WIDTH/2,-WIDTH/2);
glVertex3d(WIDTH/2,-WIDTH/2,-WIDTH/2);
glVertex3d(WIDTH/2,WIDTH/2,-WIDTH/2);
glVertex3d(-WIDTH/2,WIDTH/2,-WIDTH/2);
// front face
glNormal3d(0,0,1);
glVertex3d(-WIDTH/2,-WIDTH/2,WIDTH/2);
glVertex3d(WIDTH/2,-WIDTH/2,WIDTH/2);
glVertex3d(WIDTH/2,WIDTH/2,WIDTH/2);
...```