Discussion Session 3 (supplementary)

Sikun LIN
sikun@ucsb.edu

Topics

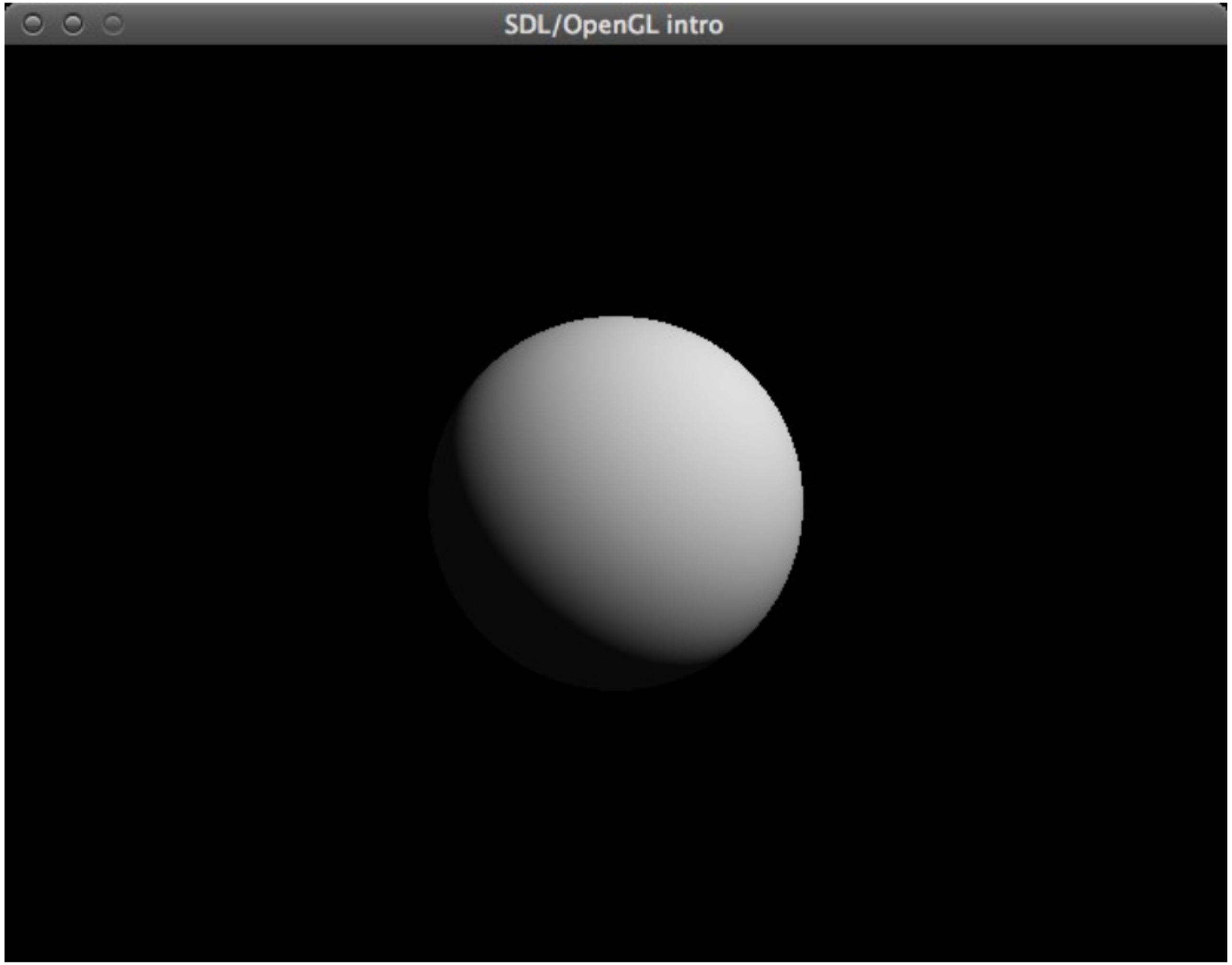
- Define light sources and lighting model
- Define the material properties of the objects
- Control the position of light sources

Light in OpenGL

- **Ambient light**
  - Scattered light (seemingly coming from all directions)
- **Diffuse light**
  - Light coming from one direction
  - Scattered evenly when bouncing off a surface
- **Specular light (“shininess”)**
  - Light coming from one direction
  - Bounces off the surface in a preferred direction

Basic Example

```
void myinit(int width, int height)
{
  GLfloat light_position[] = { 1.0,1.0,1.0,0.0 };
  glLightfv(GL_LIGHT0, GL_POSITION,light_position);
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  glShadeModel(GL_SMOOTH);
// continue with initialisation code as before
// ....

void dispay()
{
  glutSolidSphere(1.0, 100, 100);
  glFlush();
}
```

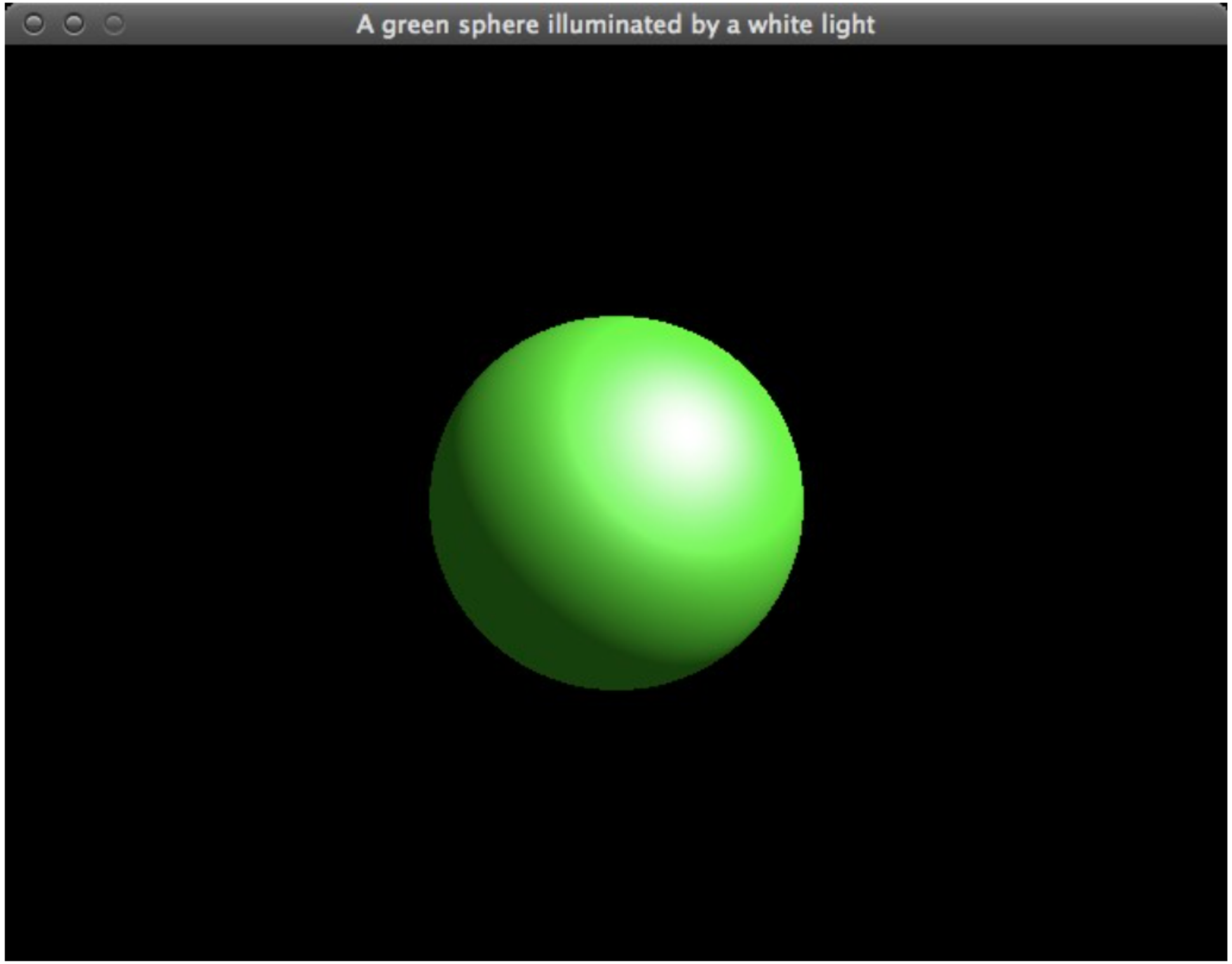

SDL/OpenGL intro

Material Properties

- The color of a material depends on the percentage of incoming red, green and blue light it reflects
- Material colors determine reflectance of the light component:
    - **Ambient and diffuse reflections**
        - define the color of the material (normally they are same color)
    - **Specular reflection**
        - produces highlights (usually white)
        - the amount of specular reflection depends on the location of the viewpoint -- brightest along the direct angle of reflection
    - **Emissive Color**
        - Light originating from an object (ex. simulating lamps)
        - Unaffected by any light sources

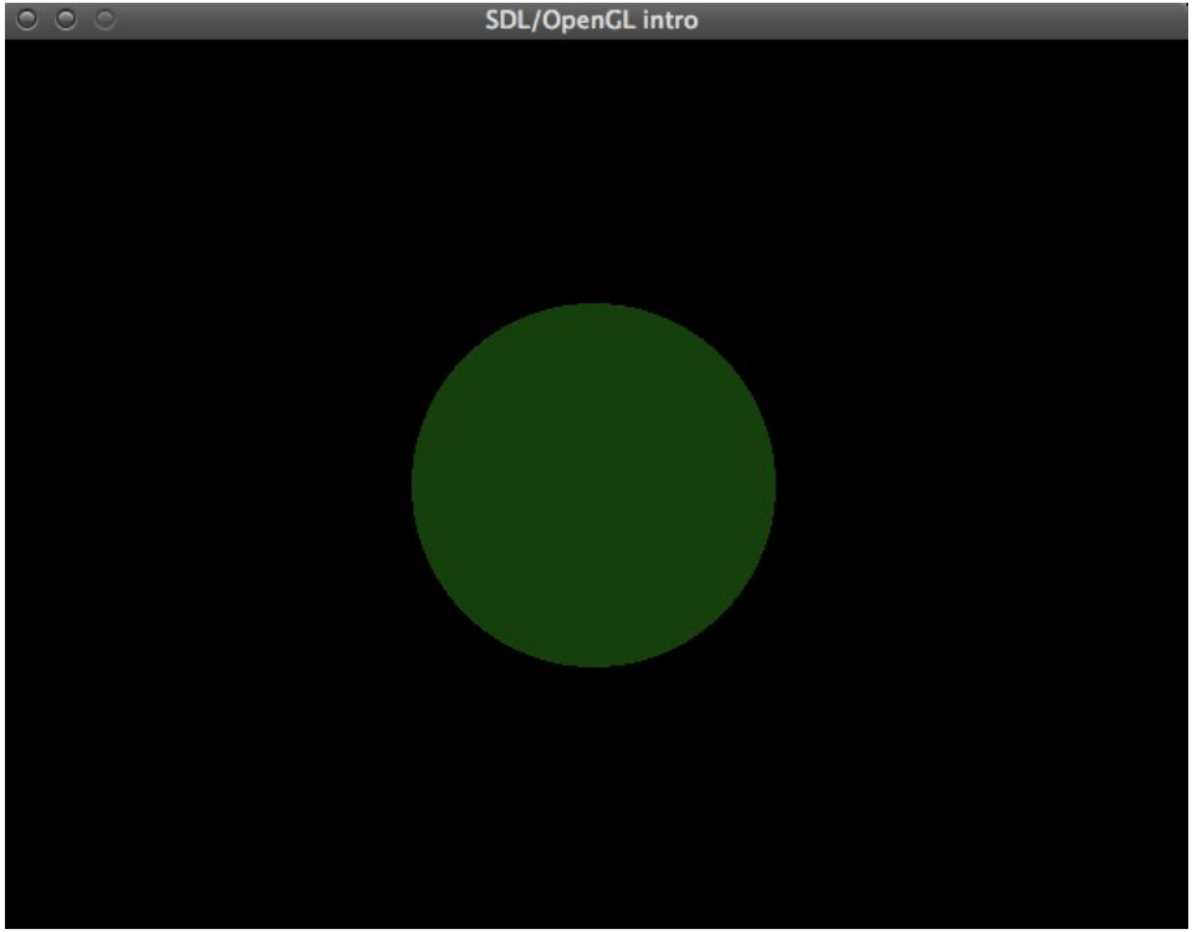
Lighting Example

```
void myinit(int width, int height)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 10.0 };
   GLfloat mat_ambient_and_diffuse[] = { 0.0, 1.0, 0.0, 1.0 };
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_and_diffuse);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_ambient_and_diffuse);
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glShadeModel(GL_SMOOTH);
// continue with initialisation code as before
// ....
```

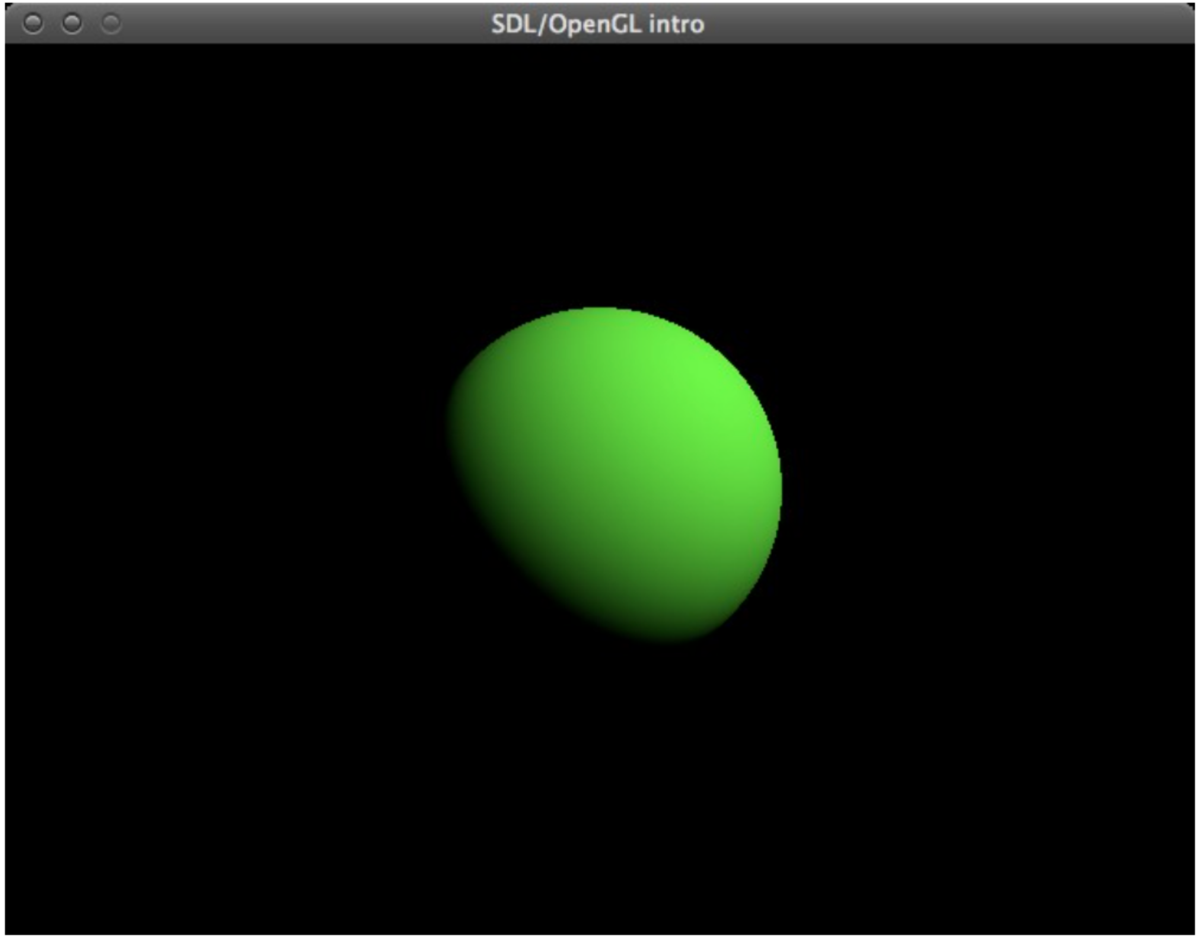

A green sphere illuminated by a white light

Let’s take a closer look at the light components...

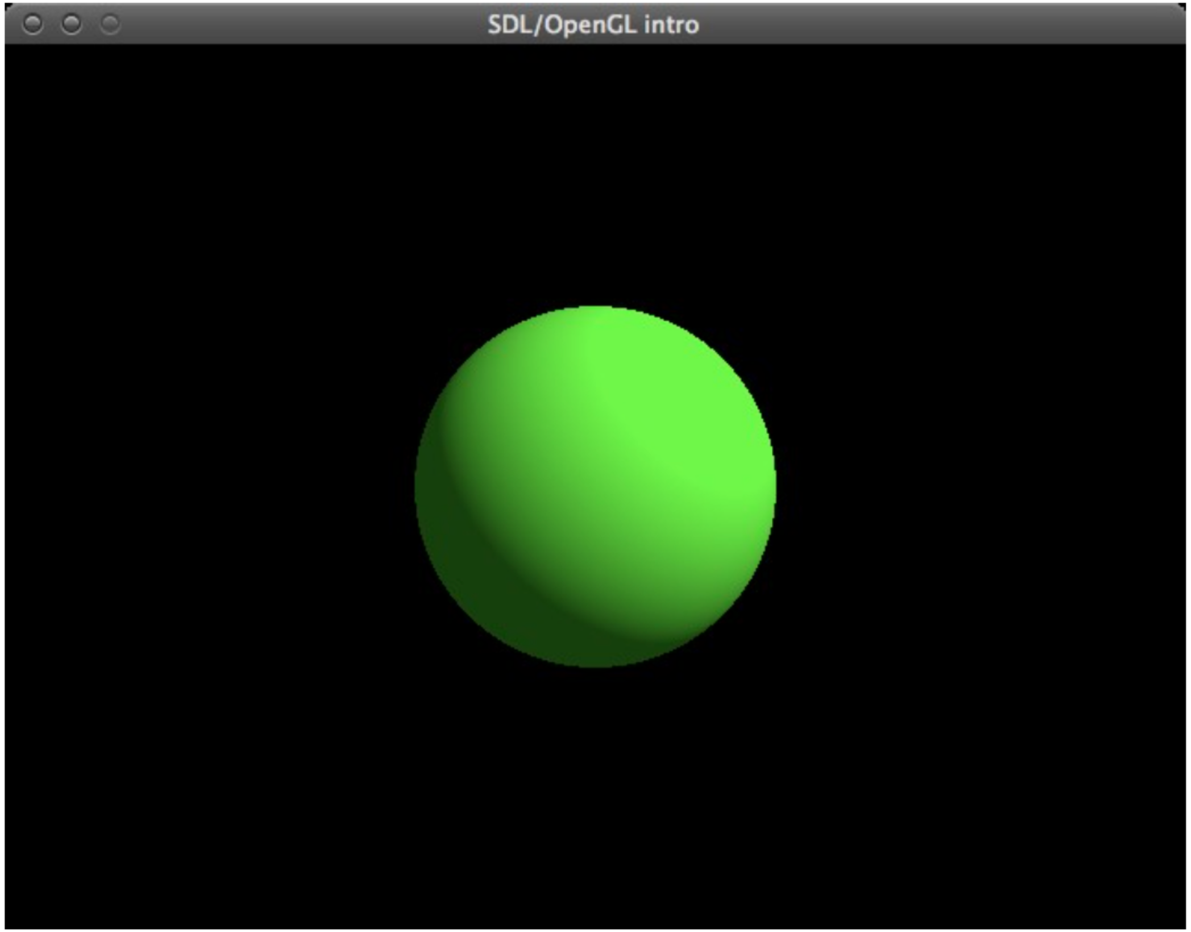
- **Ambient light**
  - Scattered light (seemingly coming from all directions)
- **Diffuse light**
  - Light coming from one direction
  - Scattered evenly when bouncing off a surface
- **Specular light (“shininess”)**
  - Light coming from one direction
  - Bounces off the surface in a preferred direction



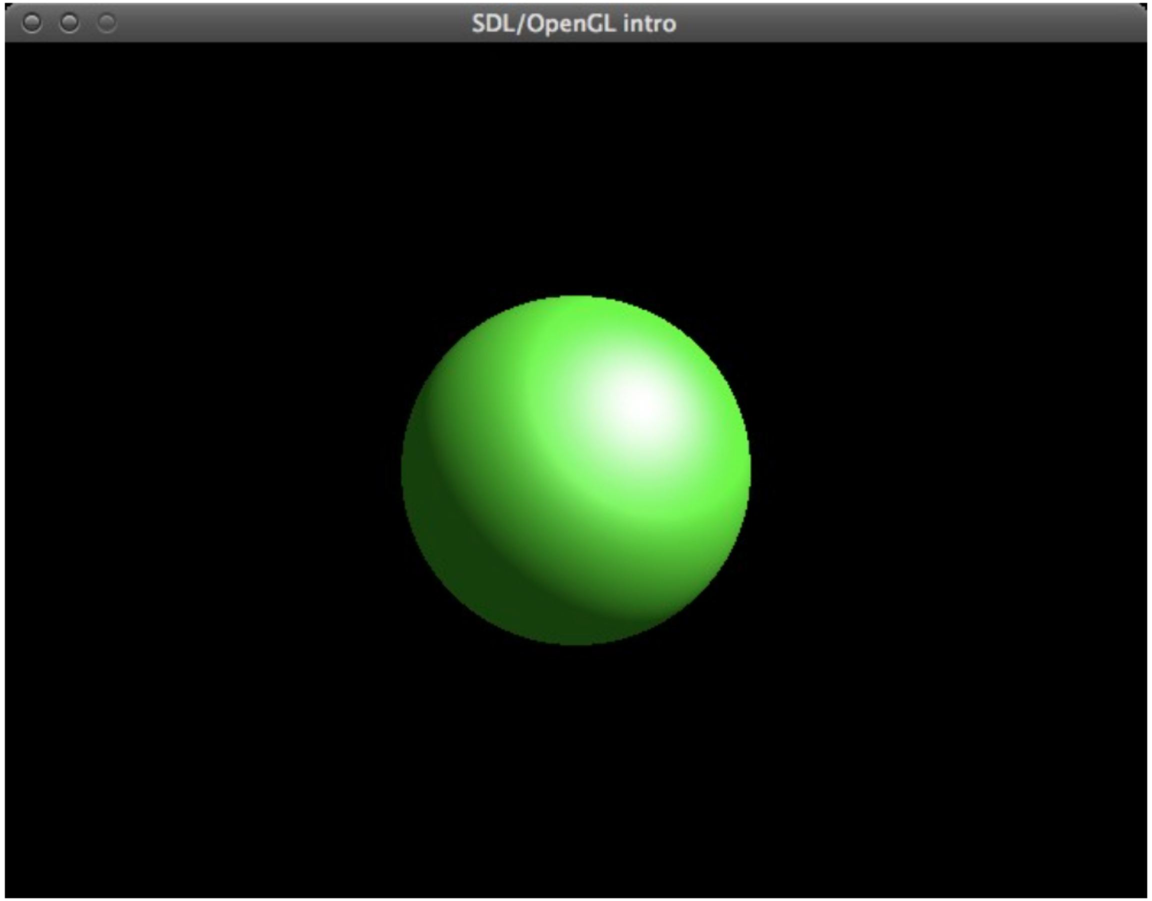
ambient light only



diffuse light only



ambient and diffuse light

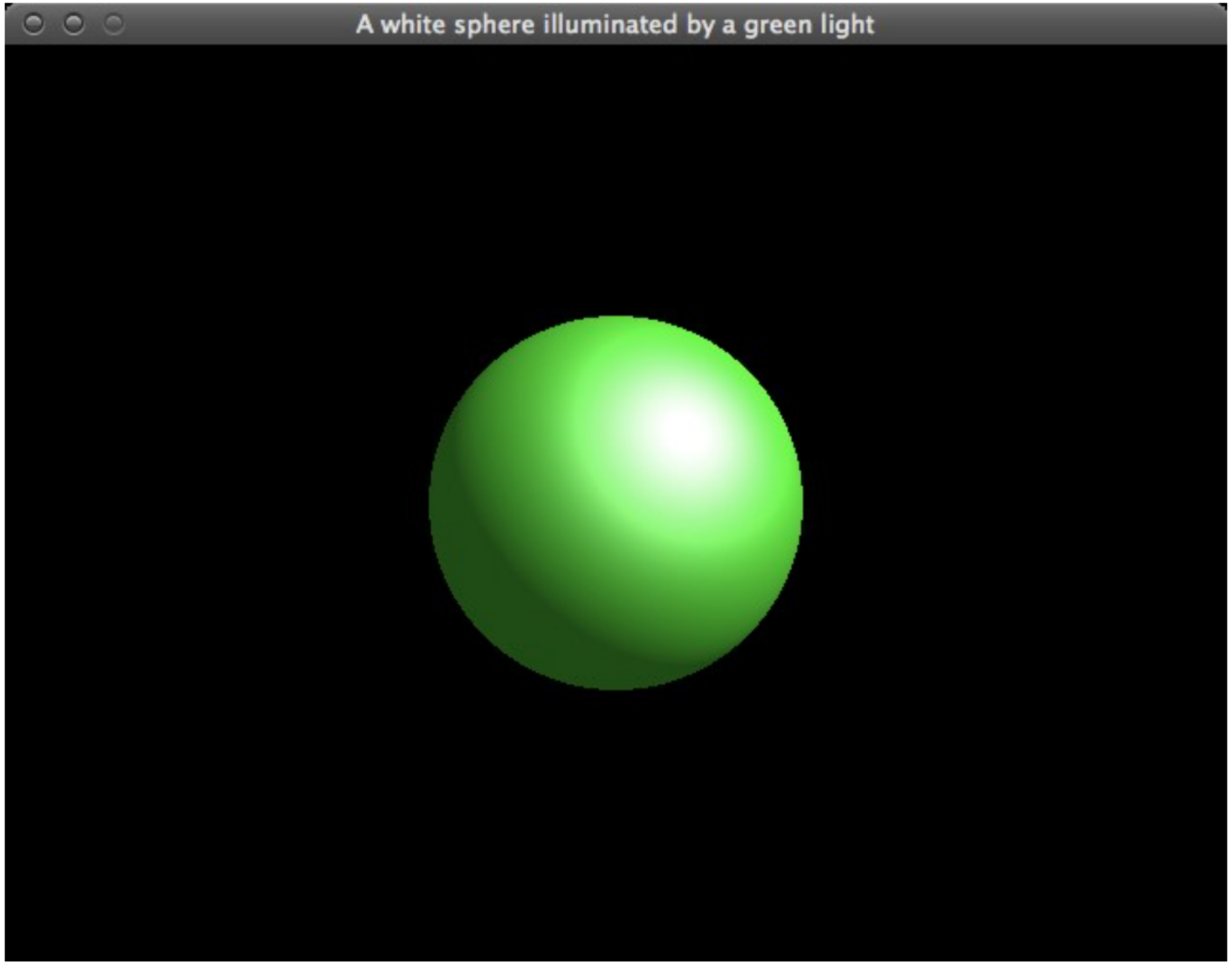


ambient , diffuse and specular light

Light Source Properties

- Properties of light sources can be changed using
- **glLight*() calls**
- Available properties:
  - **GL_AMBIENT (r, g, b, a – default: 0 0 0 1)**
  - **GL_DIFFUSE (r, g, b, a – default: 1 1 1 1)**
  - **GL_SPECULAR (r, g, b, a – default: 1 1 1 1)**
  - **GL_POSITION (x, y, z, w position – default: 0 0 1 0)**

```
void myinit(int width, int height)
{
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 10.0 };
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

GLfloat light_ambient[] = { 0.0, 1.0, 0.0, 1.0 };
GLfloat light_diffuse[] = { 0.0, 1.0, 0.0, 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);


GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
// ...
```

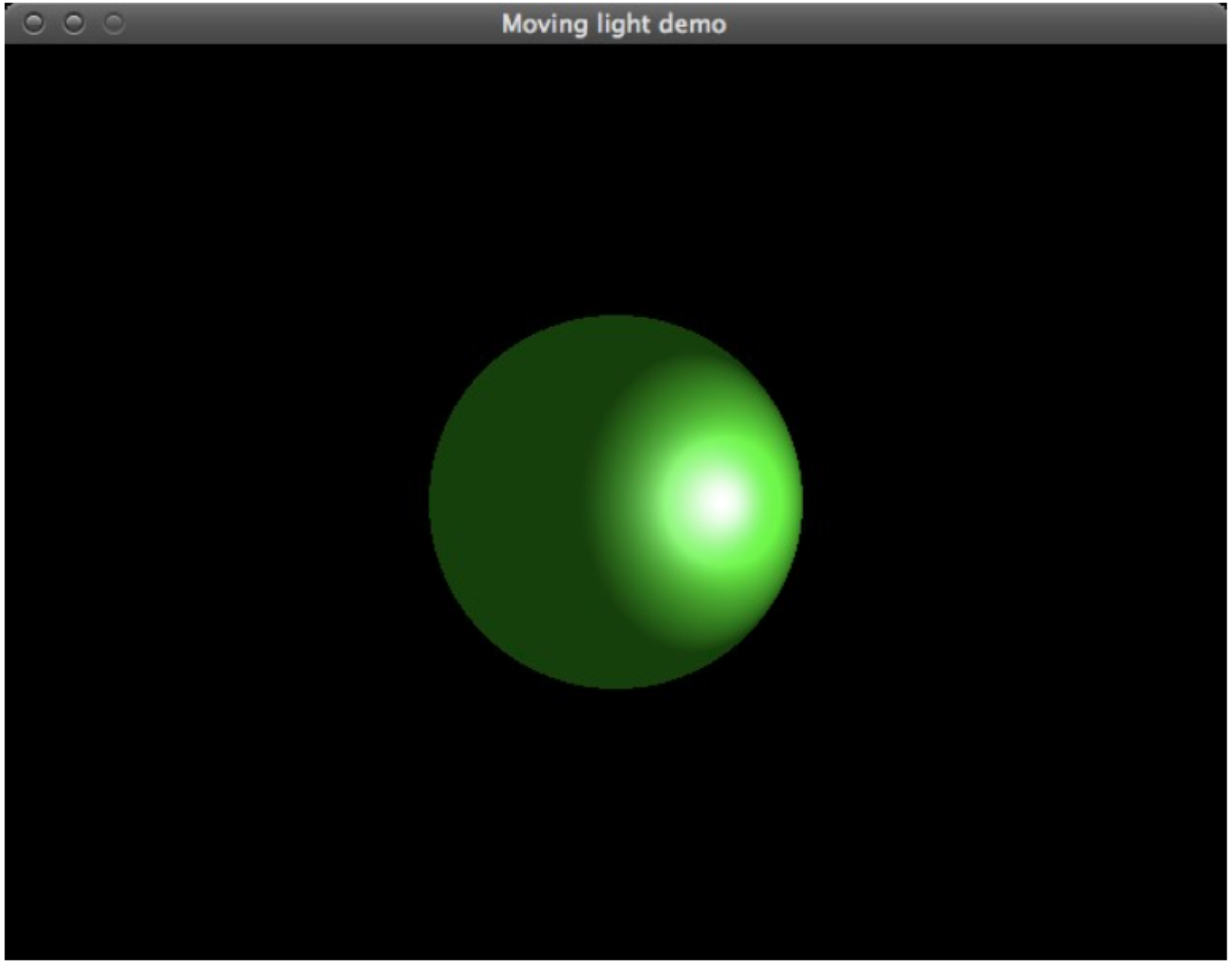

A white sphere illuminated by a green light

Moving the Light

- Lights are influenced by the modelview matrix like any other objects
- To move the light relative to a stationary object:
    - Change model transformation to specify the light position
    - Then set up a light source

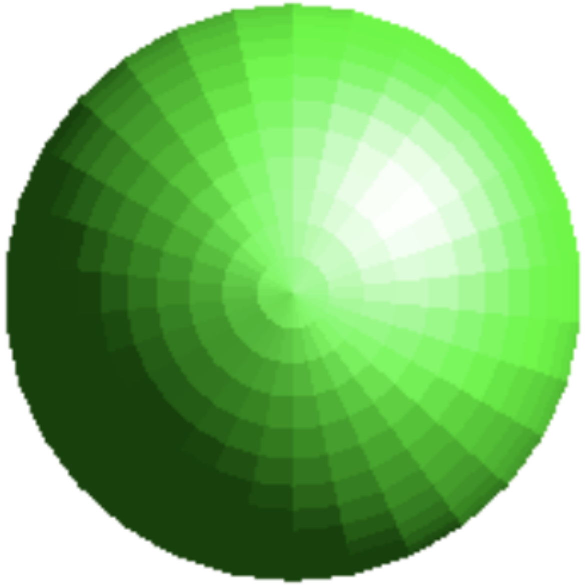
EX.

```
          glPushMatrix();
          glRotatef (angle, 0.0, 1.0, 0.0);
      glLightfv (GL_LIGHT0, GL_POSITION, light_position);
      glPopMatrix();
      drawScene();
```

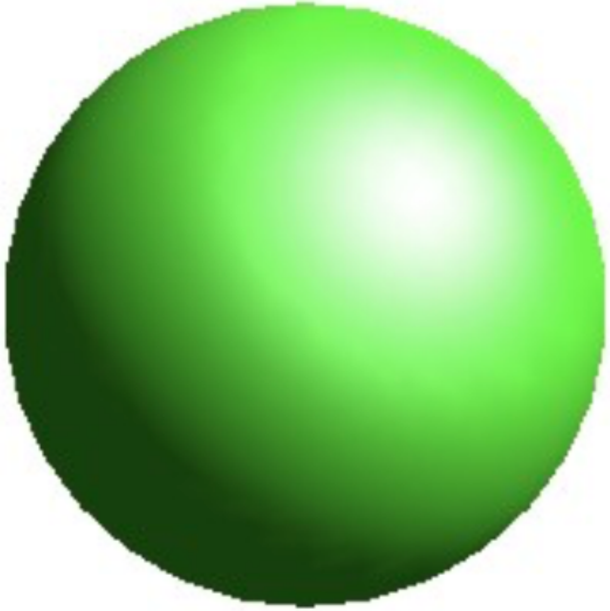

Moving light demo

Shade Models

- **Flat shading**
  - Face normals
  - One color per polygon
  - GL_FLAT
- **Gouraud shading**
  - Vertex normals
  - One color per vertex, interpolated over the polygon along edges and scanlines
  - GL_SMOOTH



glShadeModel(GL_FLAT);



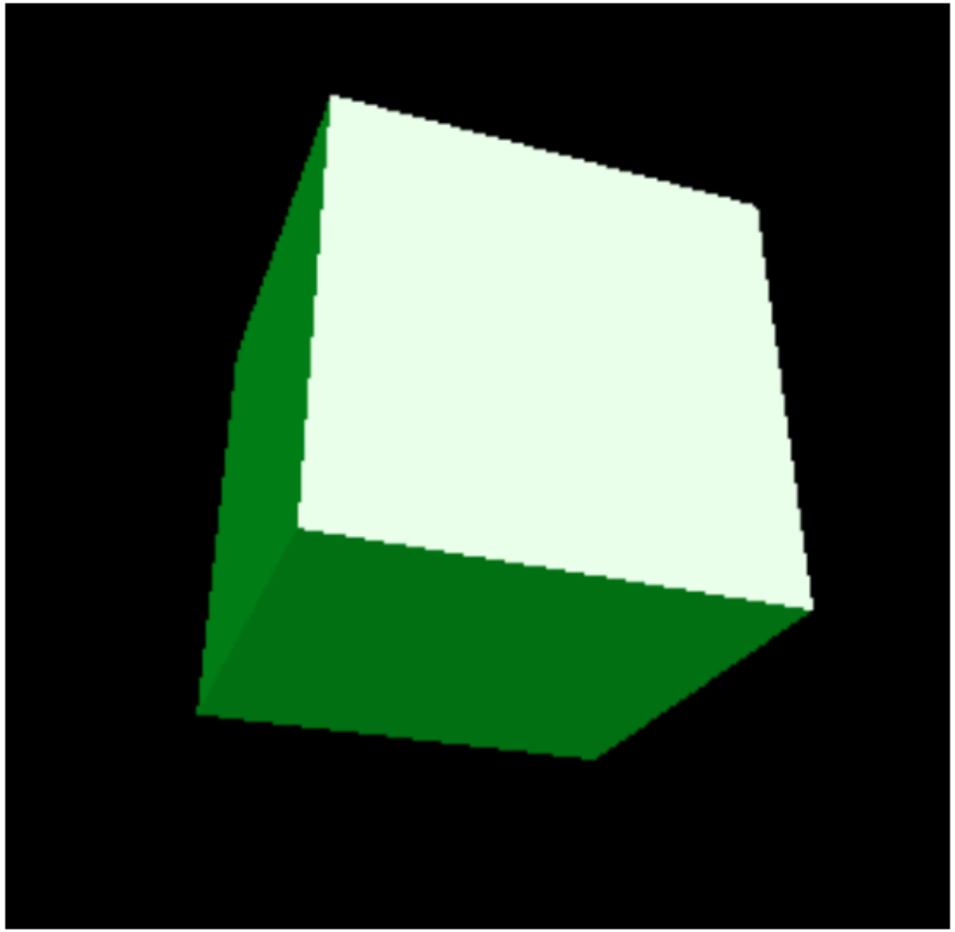
glShadeModel(GL_SMOOT
H);

You need to

- Set up a light source
- Use glMaterial instead of glColor
- Calculate normal vectors:
    - Should be unit length
    - Use glEnable(GL_CULL_FACE) to test and improve performance
    - glFrontFace(GL_CCW) -- *default value*
        - Faces in counter-clockwise order are front faces

```
void
drawBox(void)
{
        glPolygonMode(GL_FRONT_AND_BACK,
GL_FILL);
        glBegin(GL_QUAD_STRIP);
        glNormal3f(1,0,0);
        glVertex3f(-1,-1,-1);
        glVertex3f(-1,-1, 1);
        glVertex3f(-1,1, -1);
        glVertex3f(-1,1,1);

        glNormal3f(-1,0,0);
        glVertex3f( 1, 1,-1);
        glVertex3f( 1, 1, 1);

        glVertex3f( 1,-1,-1);
        glVertex3f( 1,-1, 1);

        glVertex3f(-1,-1,-1);
        glVertex3f(-1,-1, 1);
        glEnd();
```

```
glBegin(GL_QUADS);
        glNormal3f(0,0,-1);

        glVertex3f(-1,-1, 1);
        glVertex3f(-1, 1, 1);
        glVertex3f( 1, 1, 1);
        glVertex3f( 1,-1, 1);

        glNormal3f(0,0,1);
        glVertex3f(-1,-1,-1);
        glVertex3f(-1, 1,-1);
        glVertex3f( 1, 1,-1);
        glVertex3f( 1,-1,-1);
   glEnd();
```



Q & A