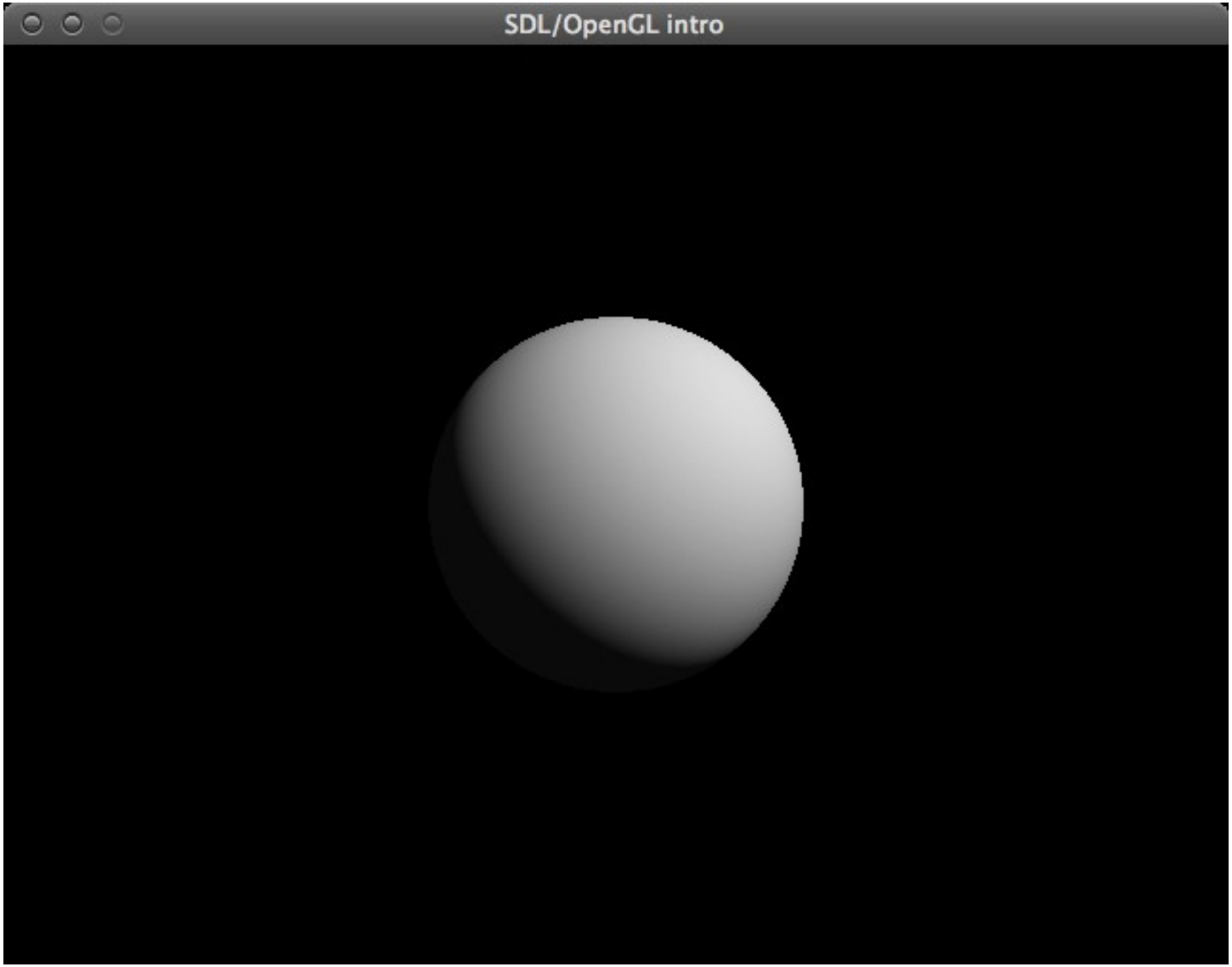
Lightning

Light in OpenGL consists of...

- 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
- emitted light
  - originates from object – unaffected by light sources

Lightning 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();
}
```

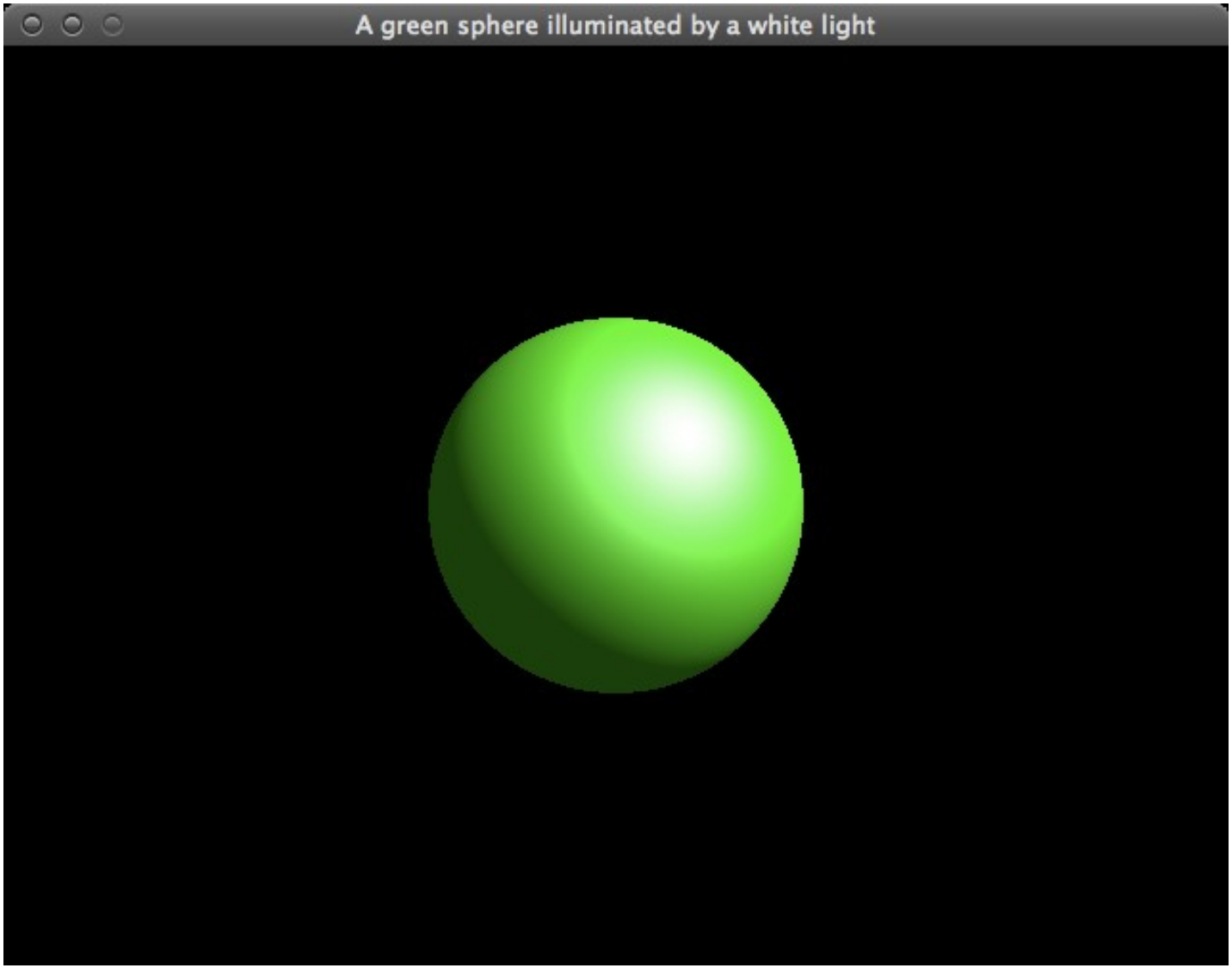

SDL/OpenGL intro

Material Properties

- The color of a material depends on the percentage of incoming red, green, and blue light it reflects.
- Like lights, materials have different ambient, diffuse, and specular colors.
  - Material colors determine reflectance of the light component
  - Ambient and diffuse reflectances define the color of the material (typically similar or identical)
  - Specular reflectance is usually white or

Lighting Example

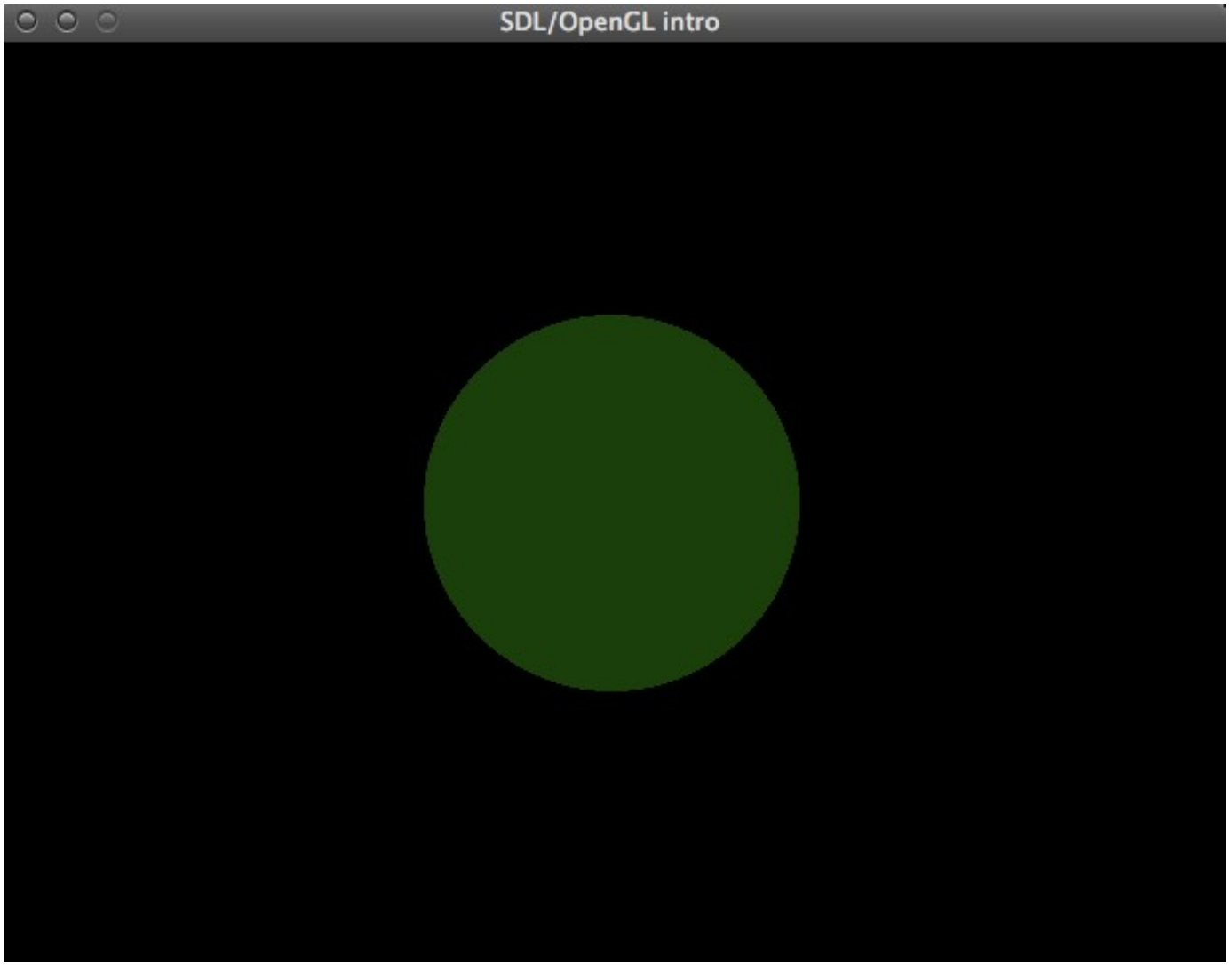
```
void myinit(int width, int height)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 10.0 };
   GLfloat mat_ambient_and_diffuse[] = { 0.0, 1.0, 0.0, 1.0 };
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   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

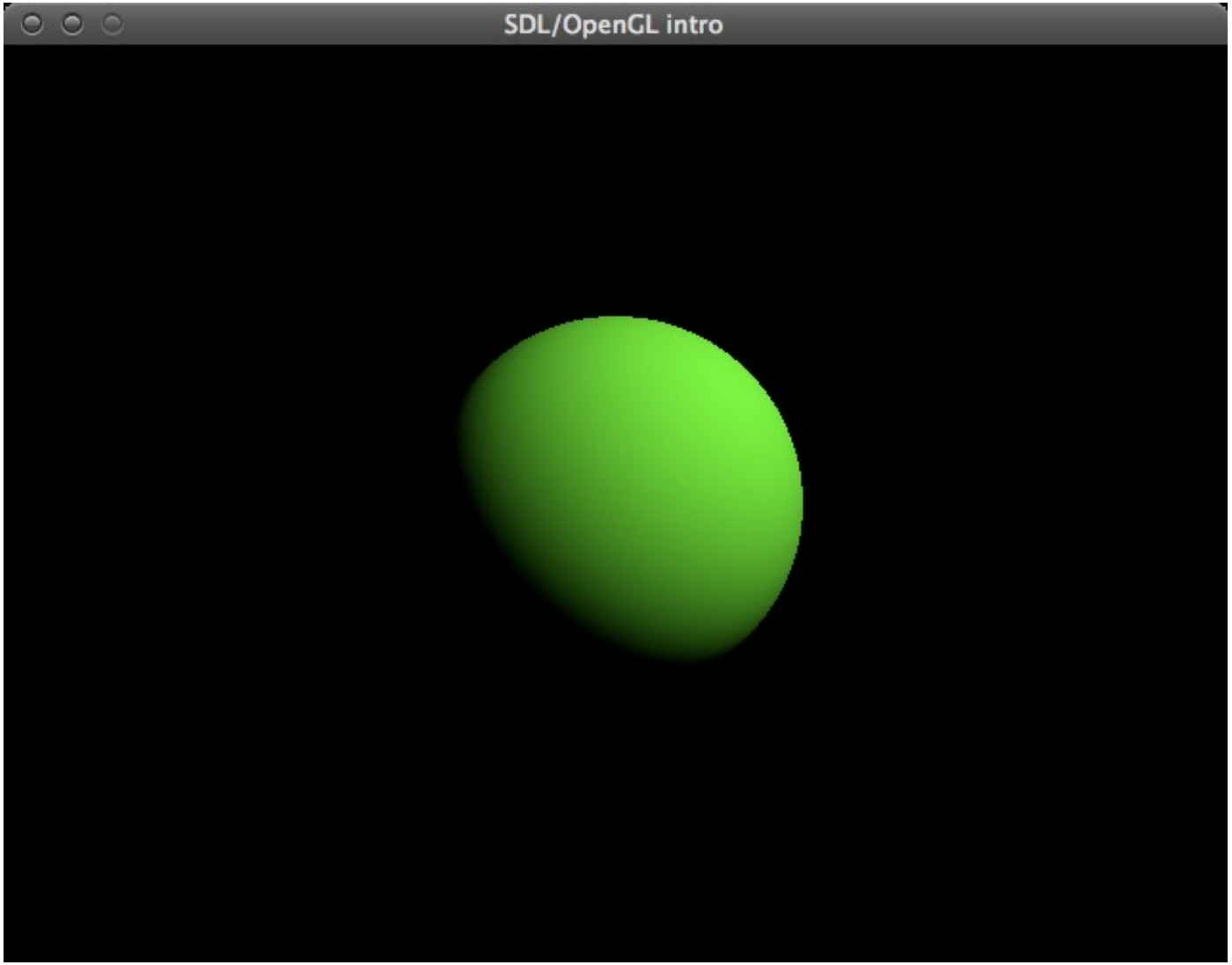
let’s take a closer look at the light components…

Light in OpenGL consists of...

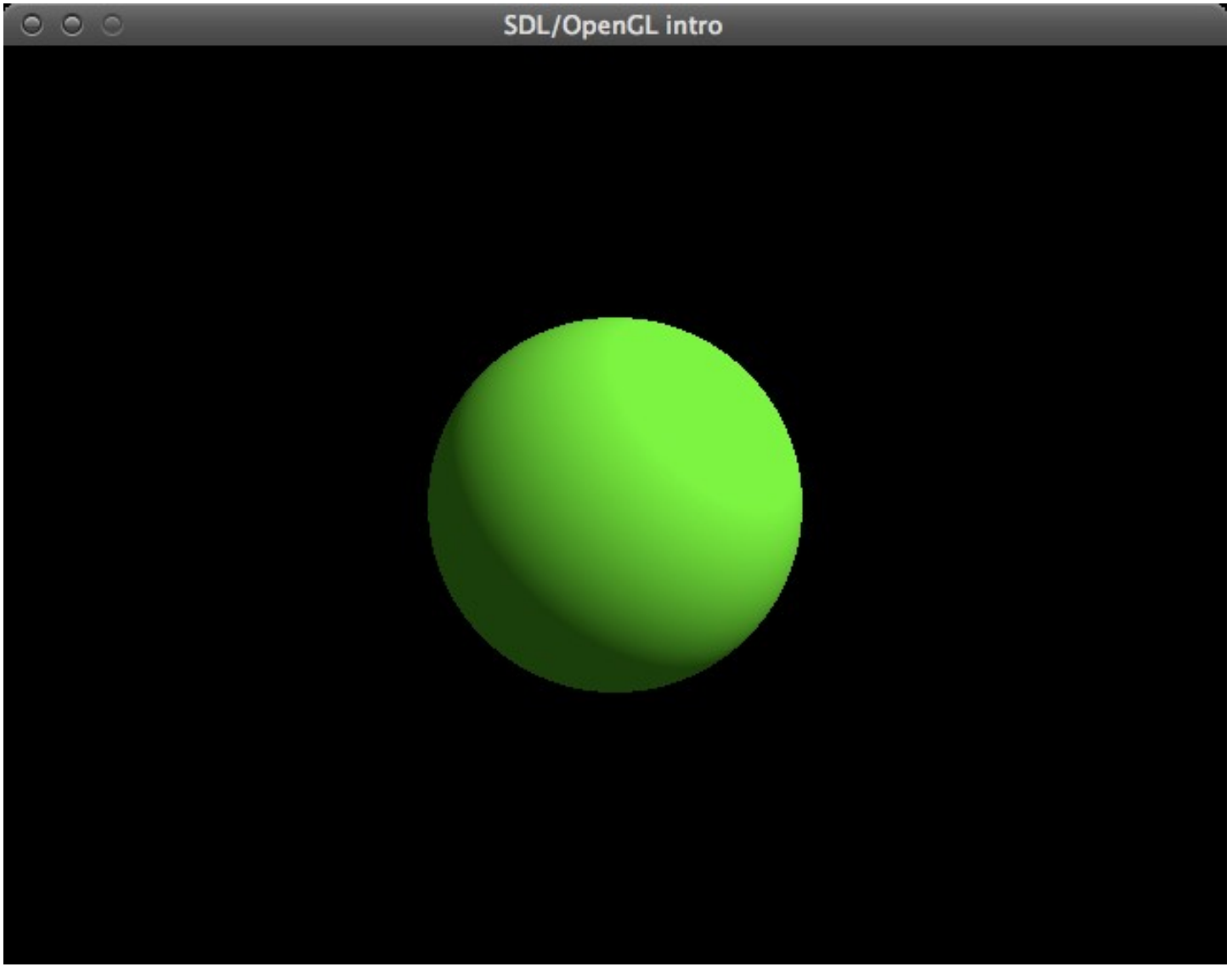
- ambient light
  - scattered light (seemingly coming from all directions)
- diffuse light
  - light coming from one direction
  - scattered evenly when bouncing off a surface
- specular light (“shininess”)
  - light coming from one direction
  - bounces off the surface in a preferred direction
- emitted light
  - originates from object – unaffected by light sources



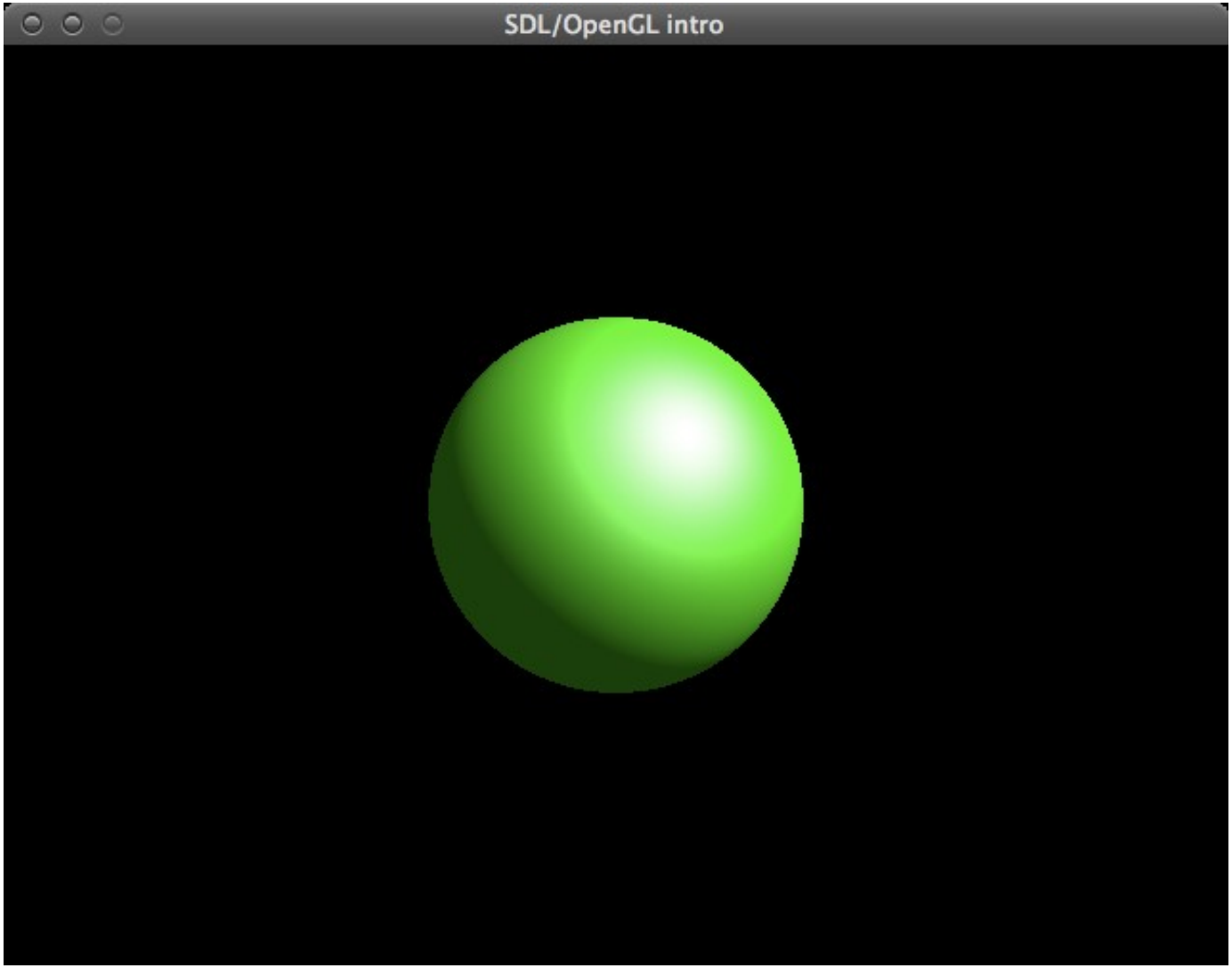
ambient light only



diffuse light only



ambient and diffuse light

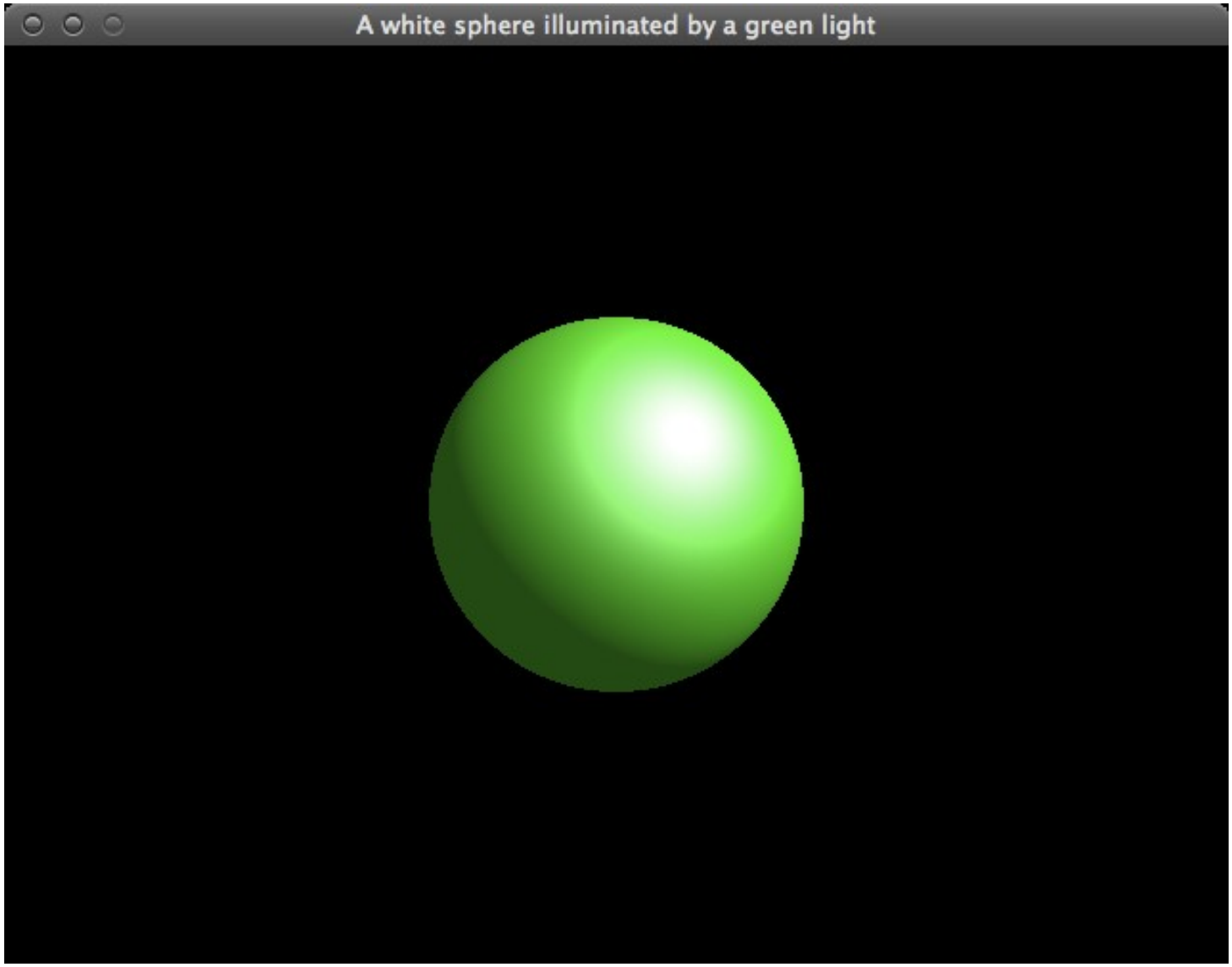


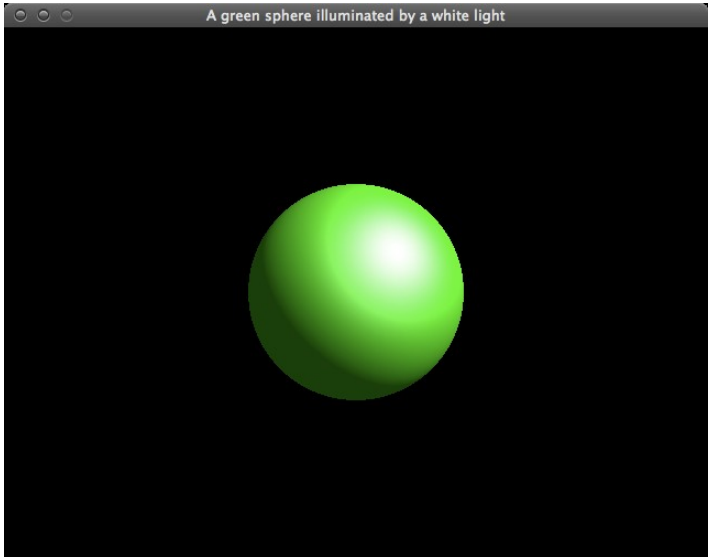
ambient , diffuse and specular light

Light Sources 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)**

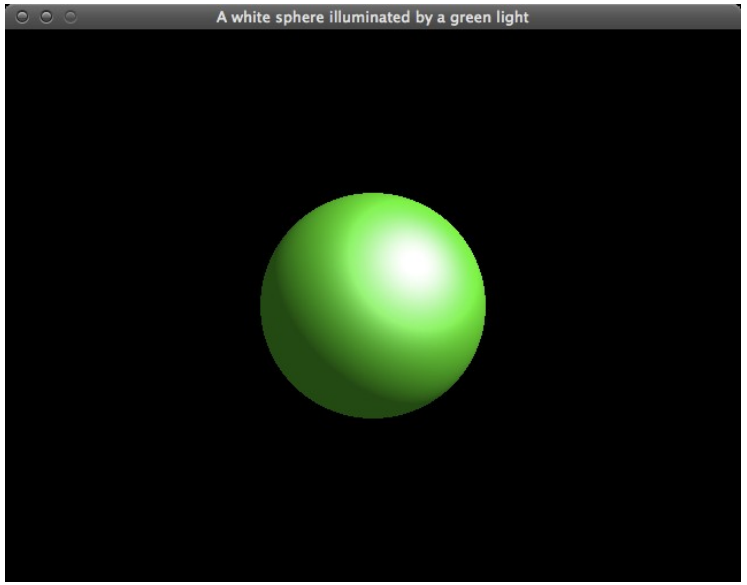
```
void myinit(int width, int height)
{
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 10.0 };
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

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



A green sphere illuminated by a white light

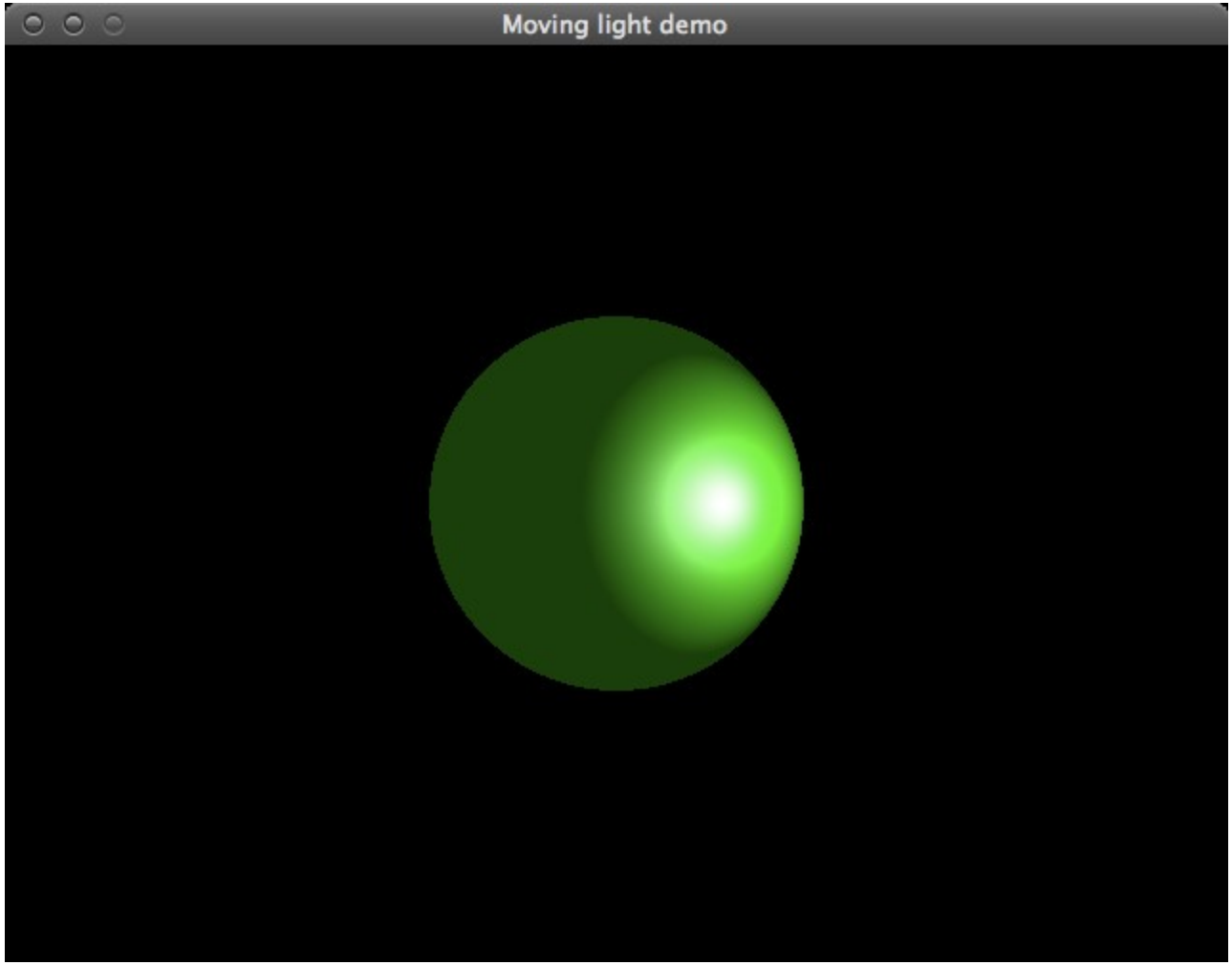
A white sphere illuminated by a green light



Moving the Light

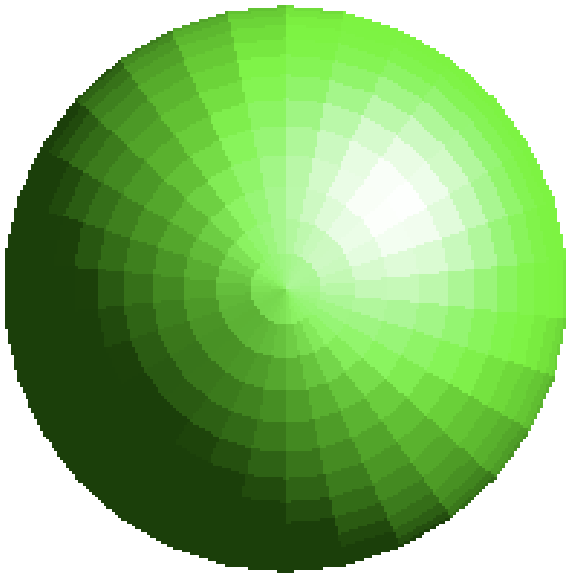
- Lights are influenced by the modelview matrix like any other object
- Translating the light relative to a stationary object?
  - Change model transform to specify the light position
  - Set light position after this
  - Something like this:

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

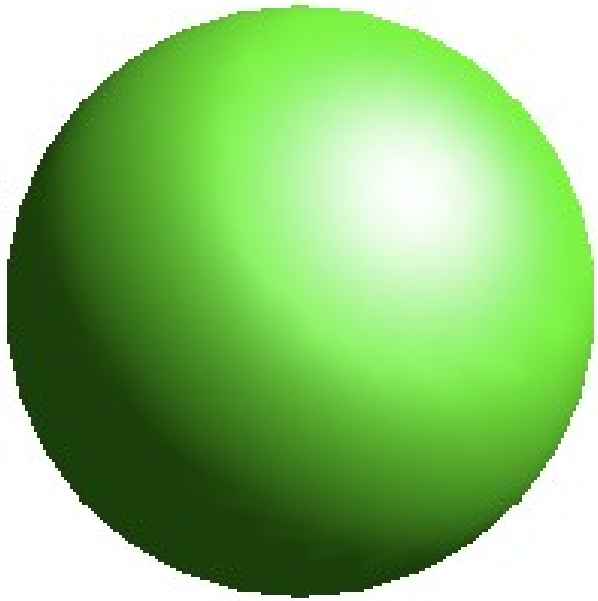

Moving light demo

Shade Models

- flat shading
  - face normals (one color per polygon)
  - GL_FLAT
- gouraud shading
  - vertex normals (one colour per vertex, interpolated
  - over the polygon along edges and scanlines)
  - GL_SMOOTH

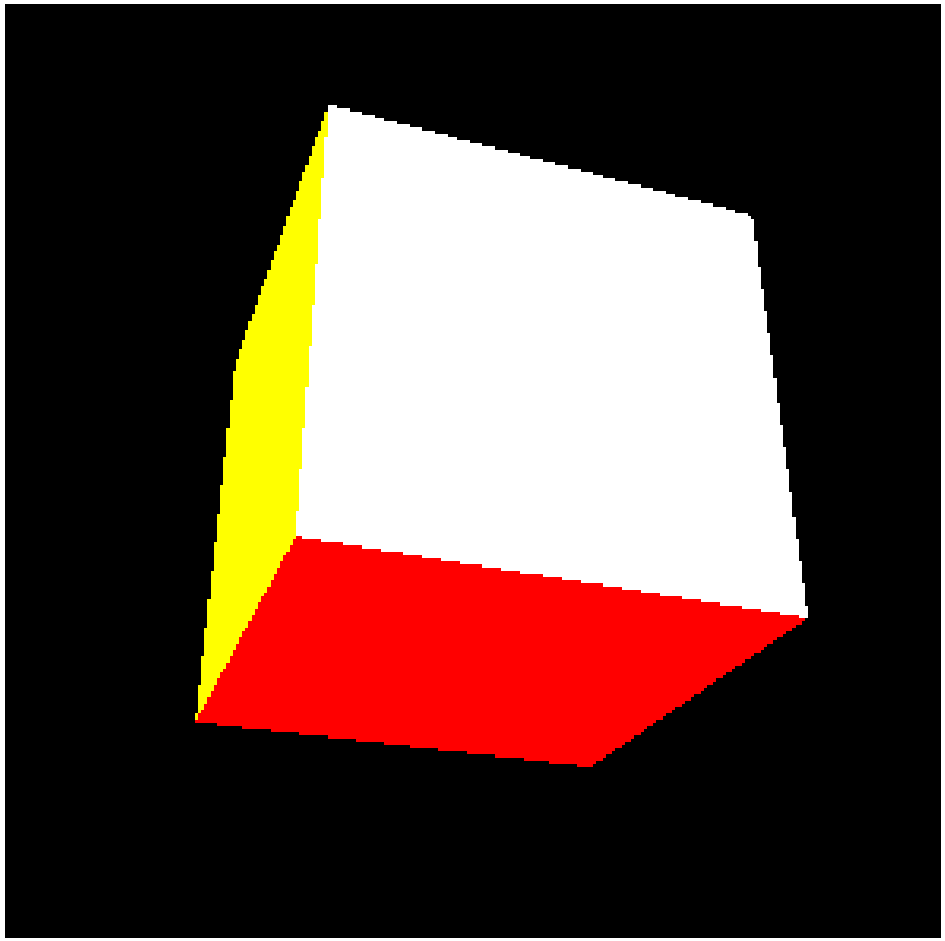


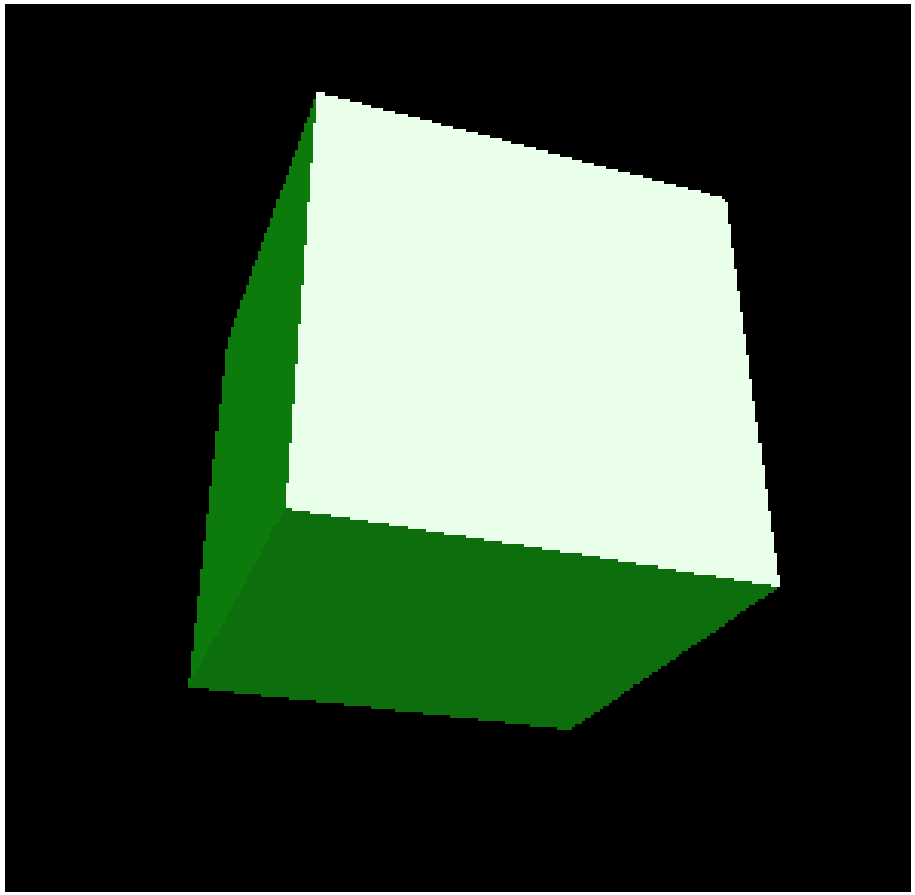
glShadeModel(GL_FLAT);



glShadeModel(GL_SMOOTH);

Let’s add light to our 3D example..



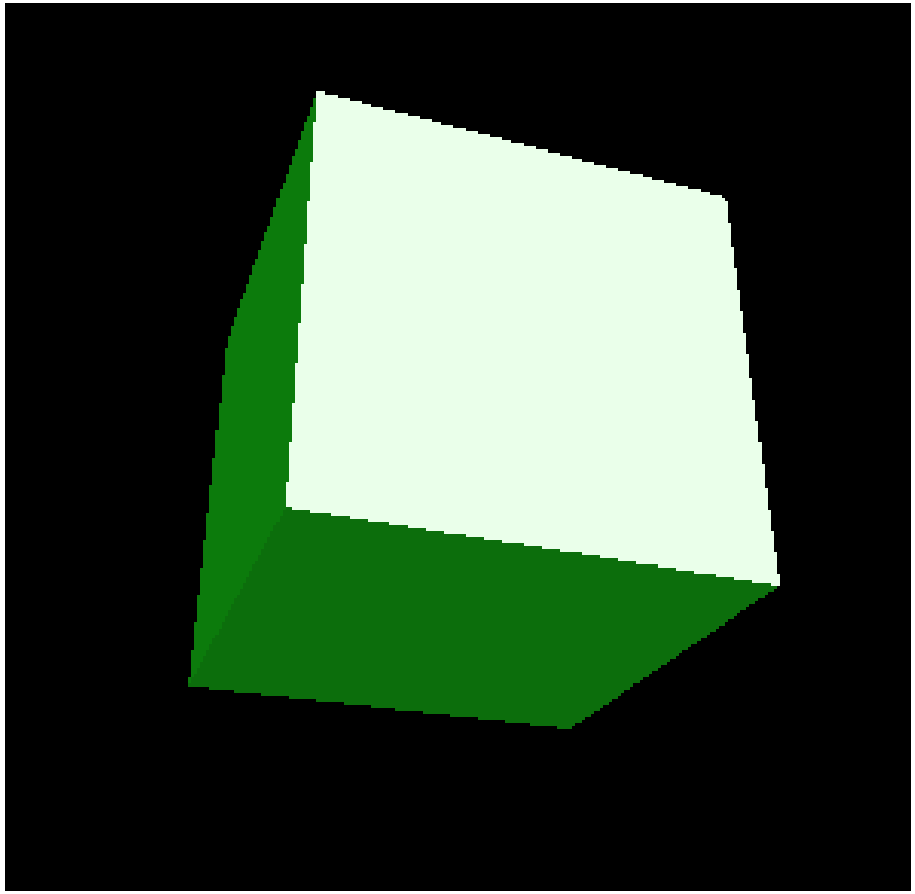


What you need

- set up a light source
- use glMaterial instead of glColor
- calculate normal vectors
  - faces must be defined in counter-clockwise order
  - to test: glEnable(GL_CULL_FACE); glFrontFace(GL_CCW);
  - normals should be unit length
    - either do normalisation yourself (recommended)
    - or let OpenGL do it for you: glEnable(GL_NORMALIZE);

```
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