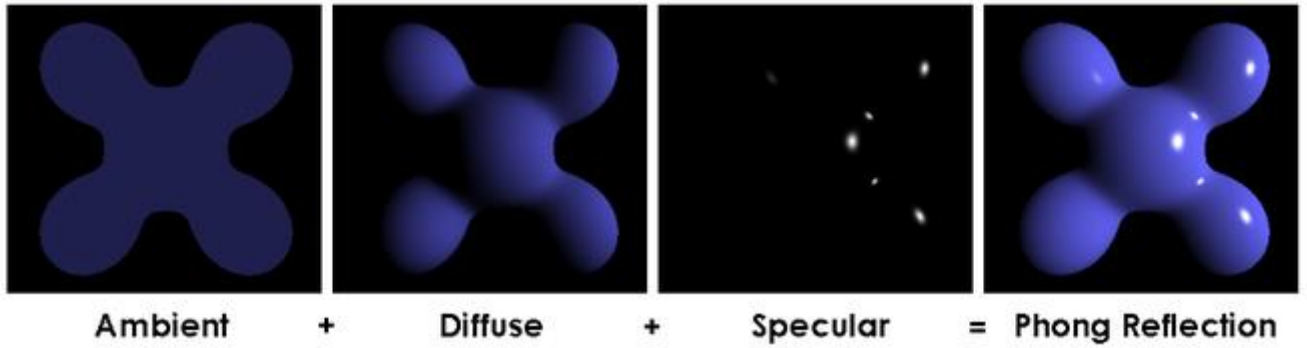
OpenGL Lighting

Mark Daniel Dacer

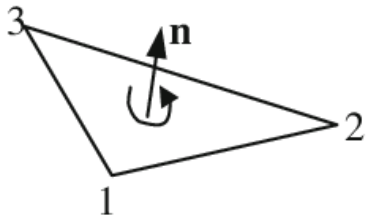
HOW OPENGL SIMULATES LIGHTS

- The Phong reflection model (also called Phong illumination or Phong lighting) is an empirical model of the local illumination of points on a surface designed by the computer graphics researcher Bui Tuong Phong. In 3D computer graphics, it is sometimes referred to as "Phong shading", particularly if the model is used with the interpolation method of the same name and in the context of pixel shaders or other places where a lighting calculation can be referred to as “shading”.

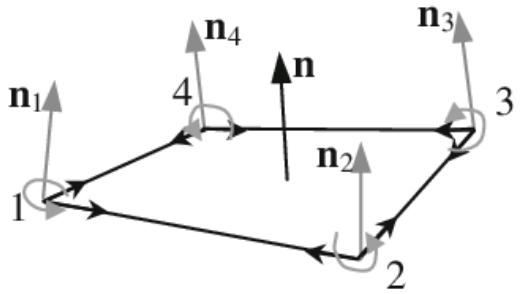


OPENGL NORMALS

- NORMALS DEFINE HOW A SURFACE REFLECTS LIGHT
  - glNormal3f(X, Y, Z)
- USE UNIT NORMALS FOR PROPER LIGHTING
  - SCALING AFFECTS A NORMAL'S LENGTH
  - GLENABLE(GL_NORMALIZE) OR GLENABLE(GL_RESCALE_NORMAL)



Triangle facet



$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$

Rectangle facet

FACE NORMAL VS VERTEX NORMAL

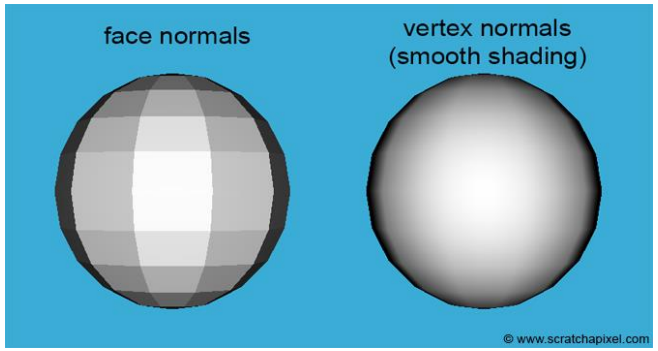
- Formula to get vertex normals

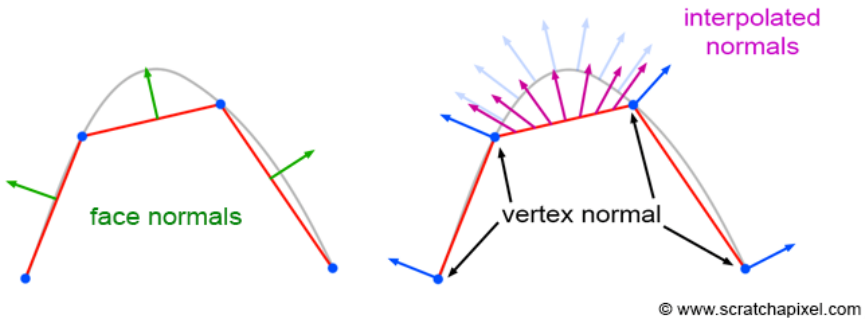
$$m = \sqrt{(x^2 + y^2 + z^2)}$$

$$x = \frac{x}{m} \qquad y = \frac{y}{m} \qquad z = \frac{z}{m}$$

glNormal3f(X, Y, Z)

```
glBegin(GL_QUADS);
    //front
    glColor3f(1.0f,0.0f,0.0f); //red
    //glNormal3f(0.0f,0.0f,1.0f);//face normals
    glNormal3f(-0.7f,-0.7,0.0f);//normalized vertex
        glVertex3f(-1.0f, -1.0f, 0.0f);
    glNormal3f(0.7f,-0.7,0.0f);//
        glVertex3f(1.0f, -1.0f, 0.0f);
    glNormal3f(0.7f,0.7,0.0f);//
        glVertex3f(1.0f, 1.0f, 0.0f);
    glNormal3f(-0.7f,0.7,0.0f);//
        glVertex3f(-1.0f, 1.0f, 0.0f);
```



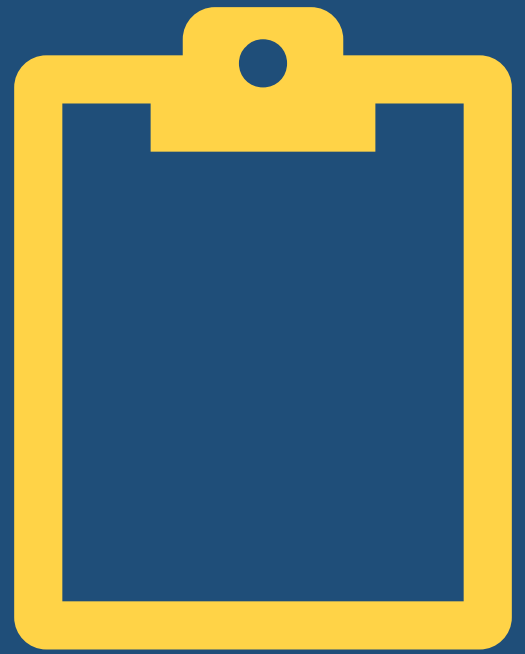


MATERIAL PROPERTIES

- DEFINE THE SURFACE PROPERTIES OF A PRIMITIVE
- glmaterialfv(*FACE, PROPERTY, VALUE*);

GL_DIFFUSE	Base color
GL_SPECULAR	Highlight Color
GL_AMBIENT	Low-light Color
GL_EMISSION	Glow Color
GL_SHININESS	Surface Smoothness

- SEPARATE MATERIALS FOR FRONT AND BACK

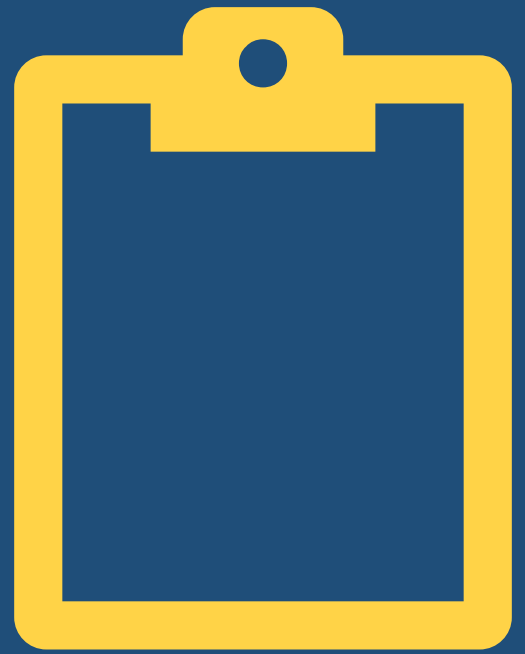


LIGHT PROPERTIES

- glLightfv(*LIGHT, PROPERTY, VALUE*);
  - ***LIGHT*** SPECIFIES WHICH LIGHT
    - MULTIPLE LIGHTS, STARTING WITH **GL_LIGHT0**

      GLGETINTEGERV(*GL_MAX_LIGHTS*, *&N*);
  - ***PROPERTIES***
    - COLORS
    - POSITION AND TYPE
    - ATTENUATION
- Sample

```
//setting world lighting
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient); //add ambient lighting
glLightfv(GL_LIGHT0, GL_POSITION, light_position);//set light position
glLightfv(GL_LIGHT0, GL_DIFFUSE, light ); //add diffuse specular lighting
glLightfv(GL_LIGHT0, GL_SPECULAR, light ); //add specular lighting
```



LIGHT SOURCES

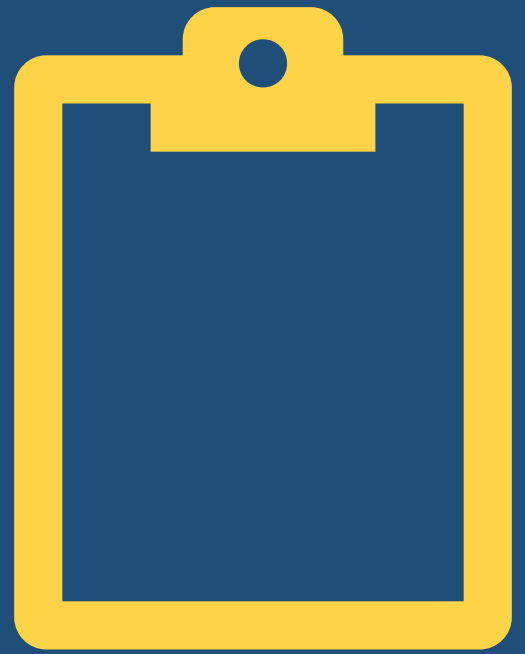
- LIGHT COLOR PROPERTIES
  - GL_AMBIENT
  - GL_DIFFUSE
  - GL_SPECULAR
- TYPES OF LIGHTS
  - LOCAL (POINT) LIGHT SOURCES
  - INFINITE (DIRECTIONAL) LIGHT SOURCES

- TYPE OF LIGHT CONTROLLED BY W COORDINATE

$w = 0$ ***Infinite Light directed along*** $(x \quad y \quad z)$

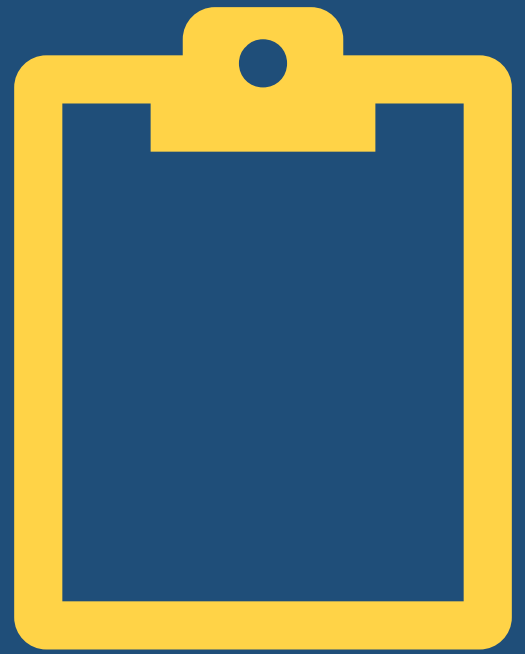
$w \neq 0$ ***Local Light positioned at*** $(x/w \quad y/w \quad z/w)$

```
Light_position[]={1,1,1,w}
glLightfv(LIGHT, GL_POSITION, Light_position);
```



TURNING ON THE LIGHTS

- FLIP EACH LIGHT'S SWITCH
  - GLENABLE(GL_LIGHTN);
- TURN ON THE POWER
  - GLENABLE(GL_LIGHTING);



Preview

```
//Initializes 3D rendering
void initRendering() {

    glClearColor(0.0,0.0,0.0,1.0); //set background to black

    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING); //Enable lighting
    glEnable(GL_LIGHT0); //Enable light #0
    glEnable(GL_NORMALIZE); //Automatically normalize normals
    glEnable(GL_AUTO_NORMAL);
    glShadeModel(GL_SMOOTH); //Enable smooth shading
}
```

```
 //setting world lighting
 glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient); //add ambient lighting
 glLightfv(GL_LIGHT0, GL_POSITION, light_position);//set light position
 glLightfv(GL_LIGHT0, GL_DIFFUSE, light ); //add diffuse specular lighting
 glLightfv(GL_LIGHT0, GL_SPECULAR, light ); //add specular lighting

 glTranslatef(0.0f, 0.0f, -15.0f); //move object in -z axis to seen in display

 glPushMatrix();
 glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular); //set object specular mateial
 glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);// set shininess
 glColor3f(0.0f, 1.0f, 0.0f);//add color material to object
 glTranslatef(3.0f, -3.0f, 0.0f); //location
 glRotatef(_angle, 0.0f, 1.0f, 0.0f); //rotation animation
 glutSolidTeapot (1.5);
 glPopMatrix();

 glPushMatrix();
 glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular); //set object specular mateial
 glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);// set object shininess
 glColor3f(0.0f, 0.0f, 1.0f);//add color material to object
 glTranslatef(-3.0f, -3.0f, 0.0f); //location
 glRotatef(_angle, 1.0f, 0.0f, 0.0f); //rotation animation
 glutSolidTorus(0.5,1,24,48);
 glPopMatrix();

 glPushMatrix();
 glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular); //set object specular mateial
 glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);// set object shininess
 glColor3f(0.0f, 0.0f, 1.0f);//add color material to object
 glTranslatef(0.0f, 1.0f, 6.0f); //location
 glRotatef(_angle, 0.0f, 1.0f, 0.0f); //rotation animation
 cube();
 glPopMatrix();
```