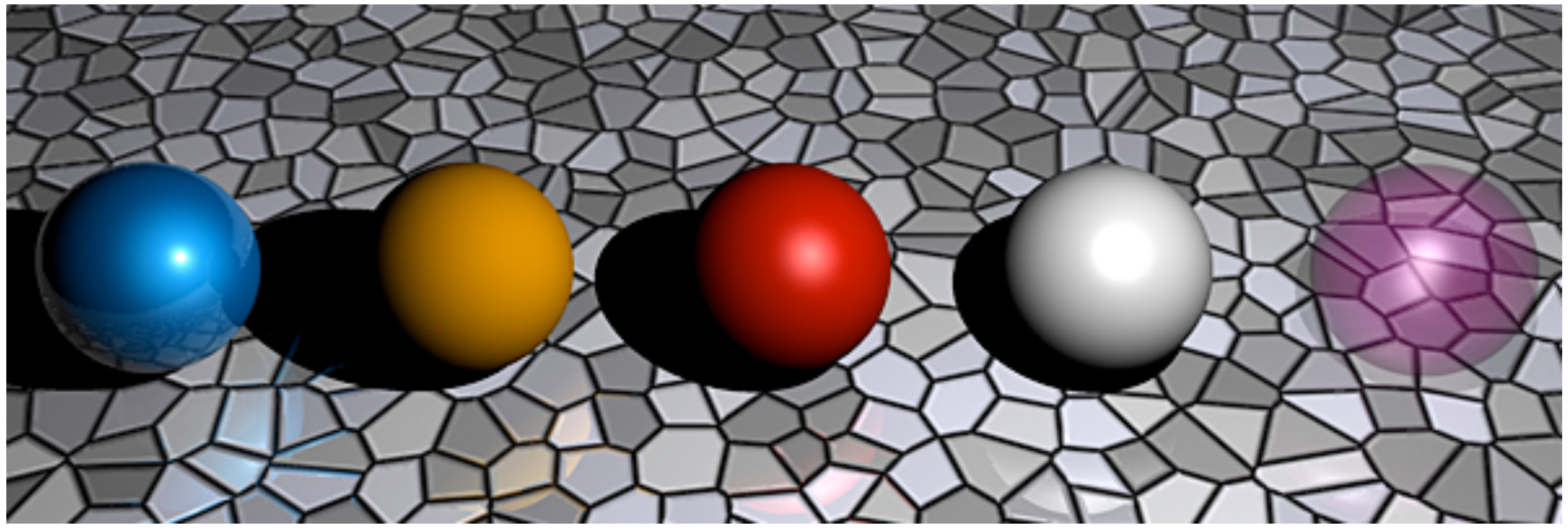


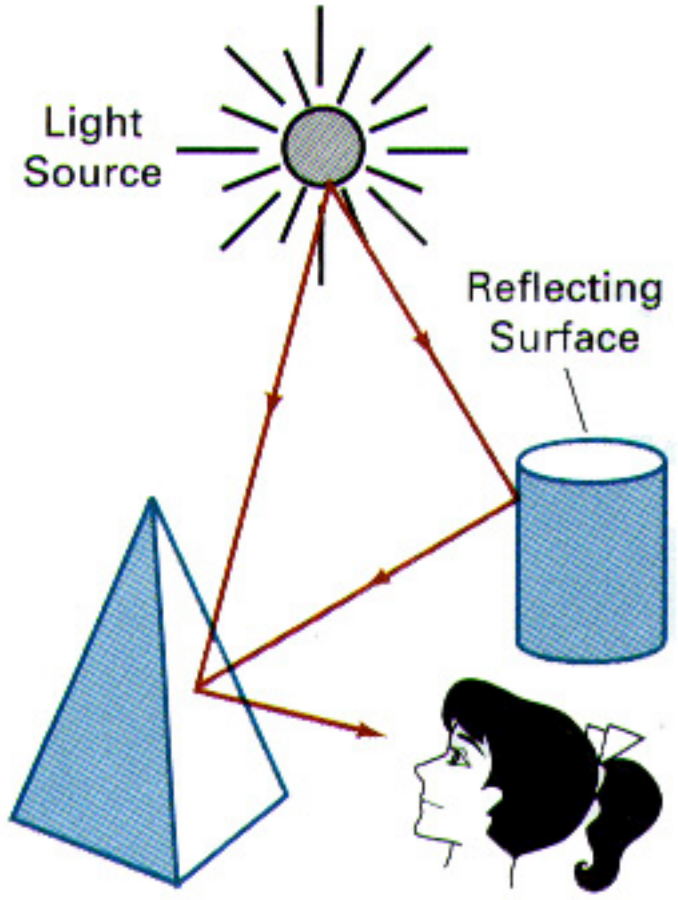
Monash University • Clayton's School of Information Technology

CSE3313 Computer Graphics

Lecture 22: Illumination Models

Illumination and Shading Models

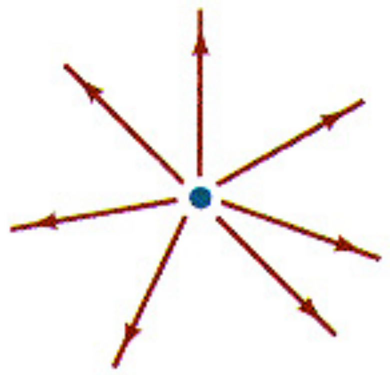
- In trying to depict a natural scene we need to model the effect of light hitting a surface. This may depend on:
  - the orientation of the surface;
  - the angle between the light source and the surface normal;
  - the surface properties of the object.
- A *light source* provides illumination of an object. Light may came from a point or area of *emission*, or may be transferred through *reflection*.



Light viewed from an opaque nonluminous surface is in general a combination of reflected light from a light source and reflections of light reflections from other surfaces.

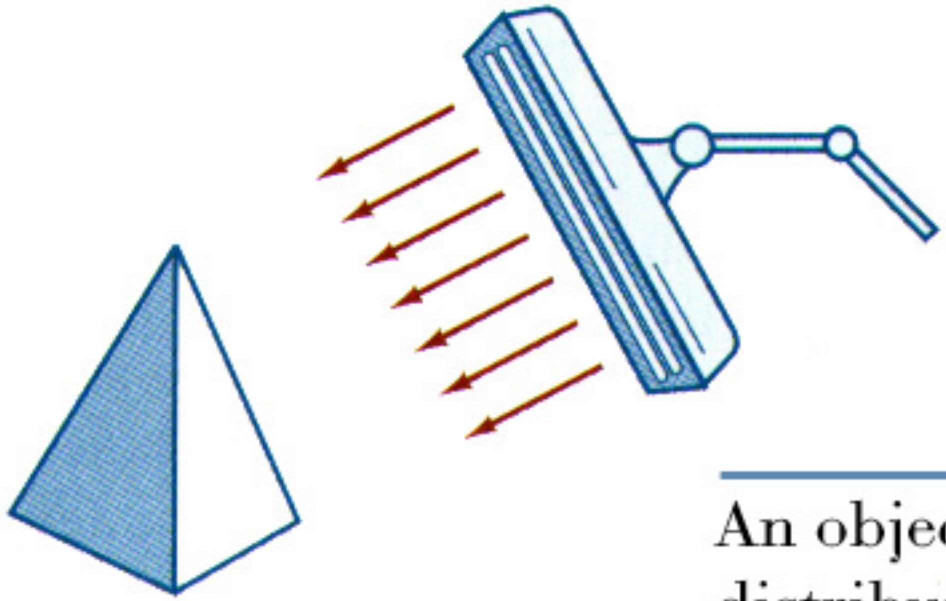
Light Sources

- The simplest light sources in computer graphics are known as *point light sources*. Rays of light originate from an single point.



Diverging ray paths from a point light source.

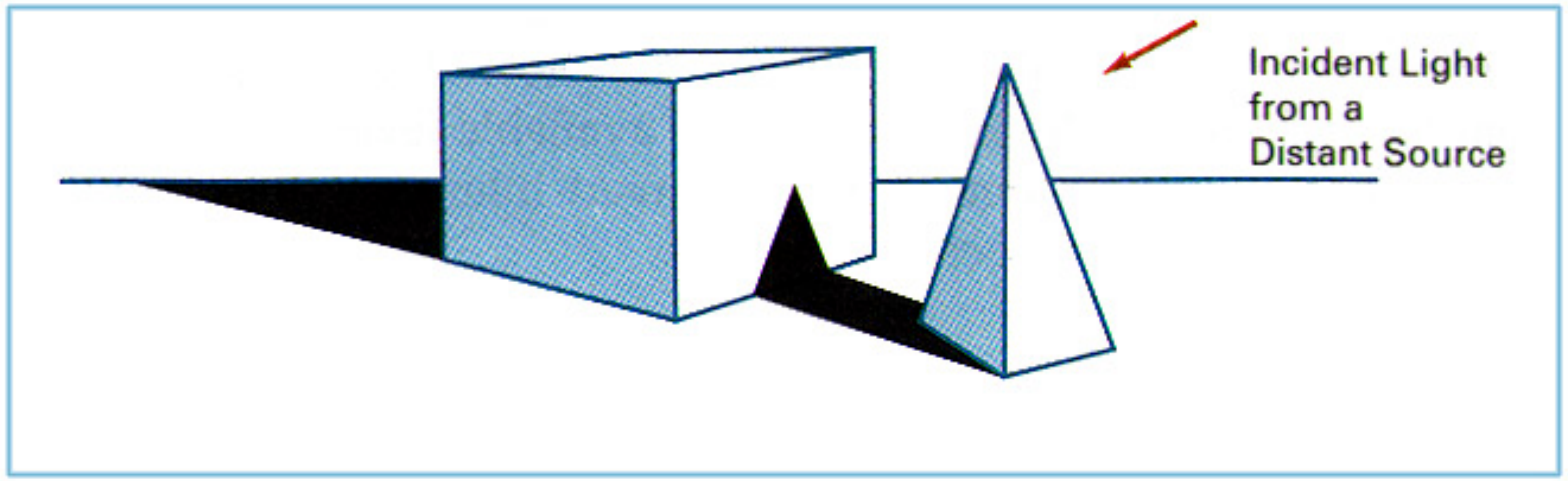
Other light sources include area or distributed light sources:



An object illuminated with a distributed light source

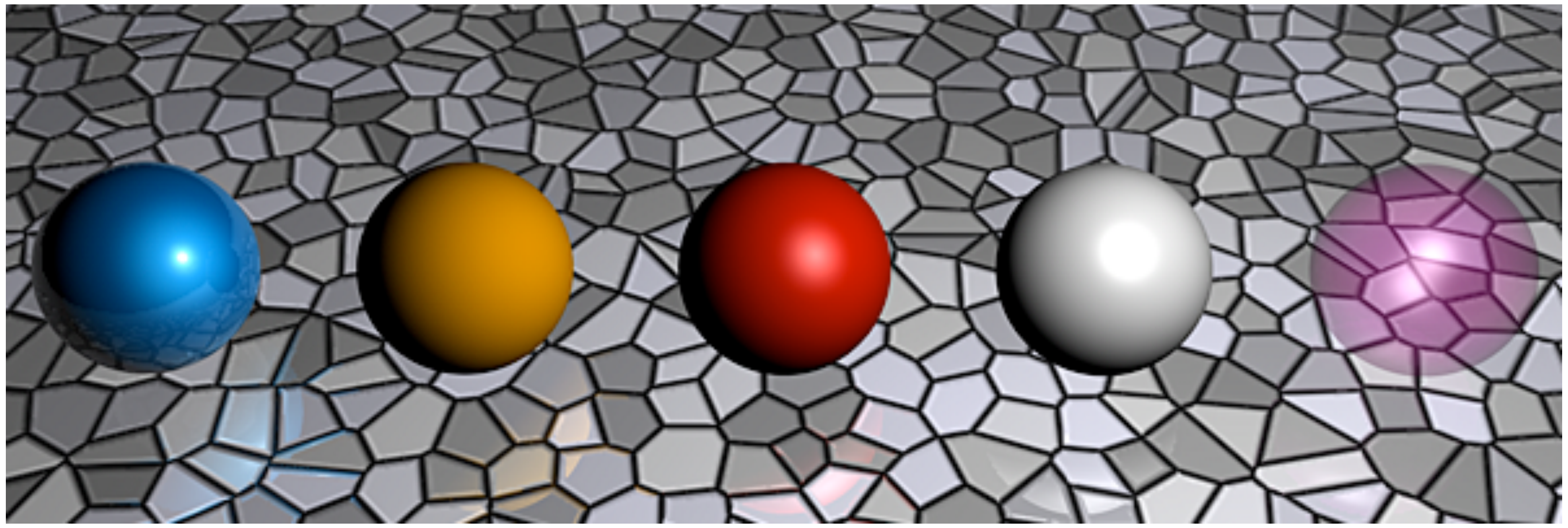
Light Sources (cont.)

- Light sources may also be constrained to effect only conic or rectangular areas (like a spotlight, sometimes with barn doors).
- Hidden surface methods may be used to locate areas where light sources produce *shadows*. By applying a hidden surface elimination method with the light source as the point of view, those areas which can not be seen from the light source are shadow areas – *shadow polygons*.
- There may be more than one light source in a scene. If the position of the lights does not change the position of the shadows do not change, even though the point of view may change.

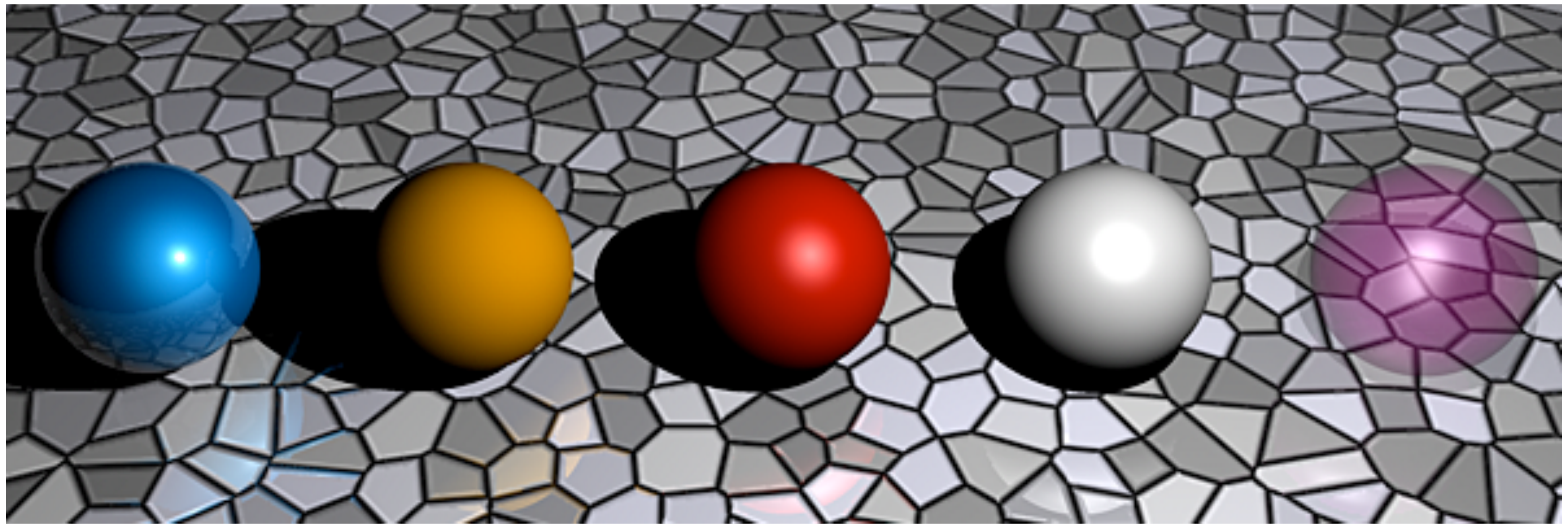


Objects modeled with shadow regions.

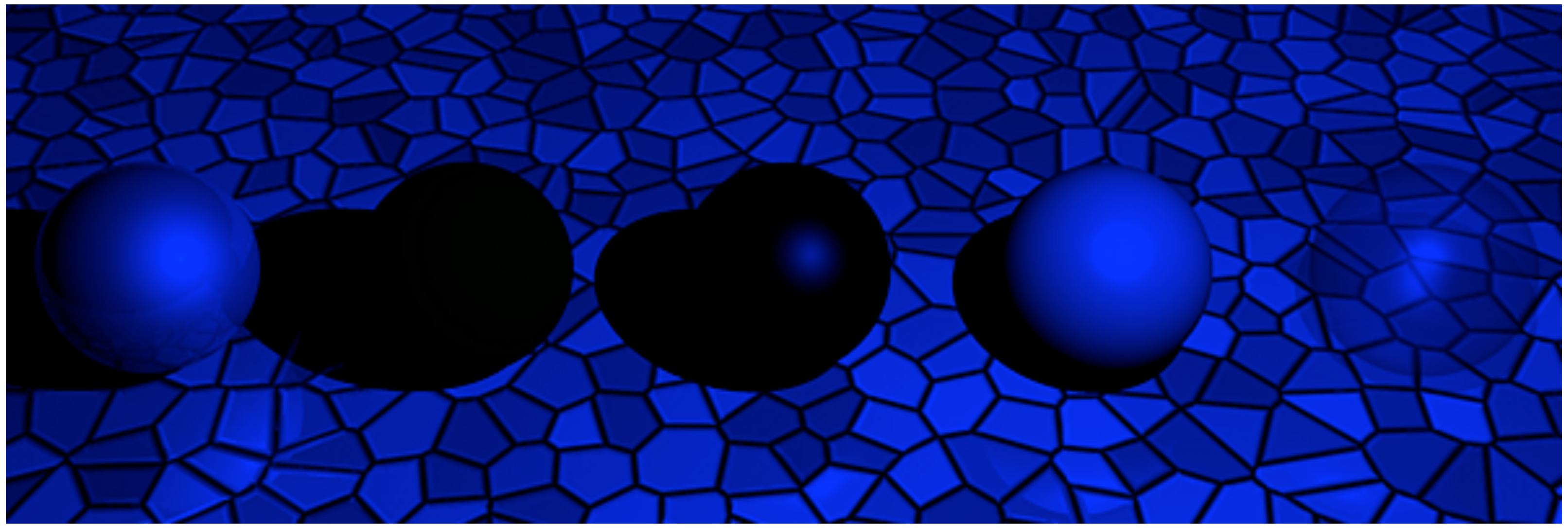
Point Light Source
(no
shadows)



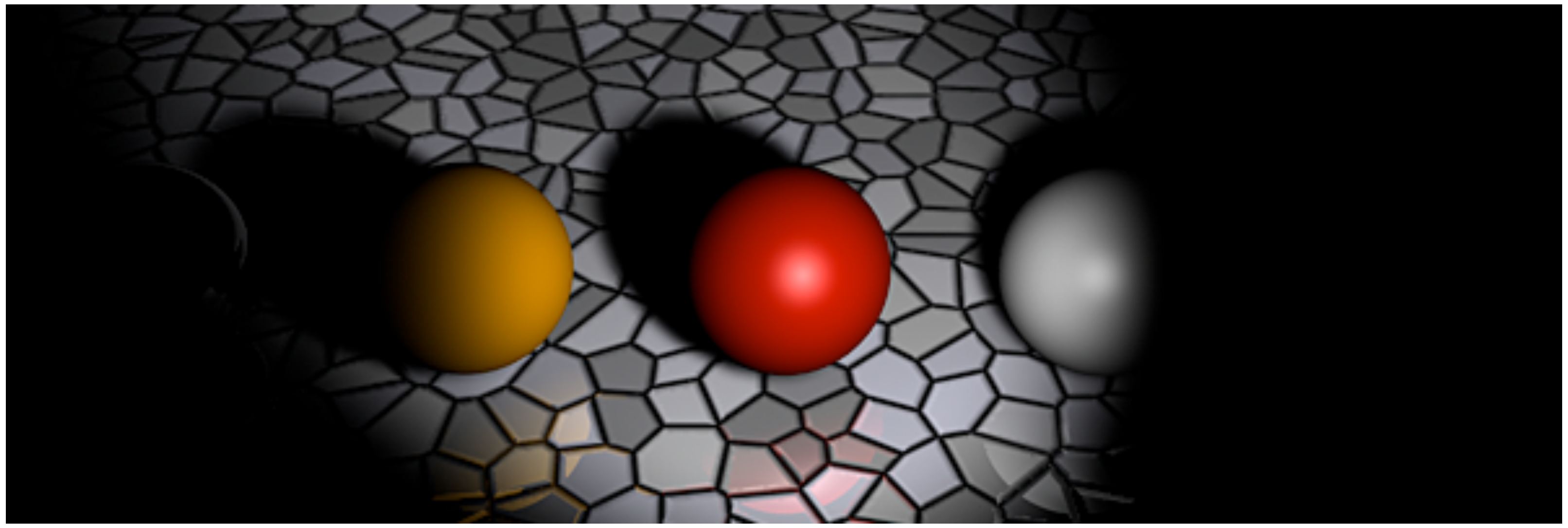
Point Light Source
(shadows)



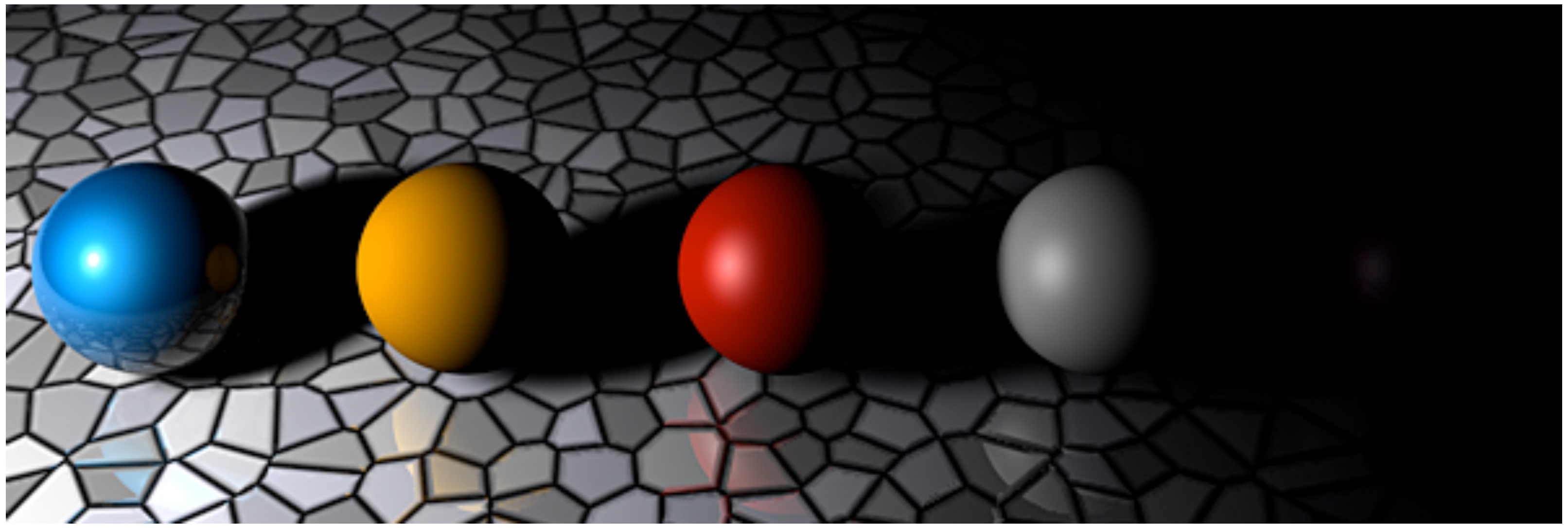
Blue Point Light
Source
(shadows)



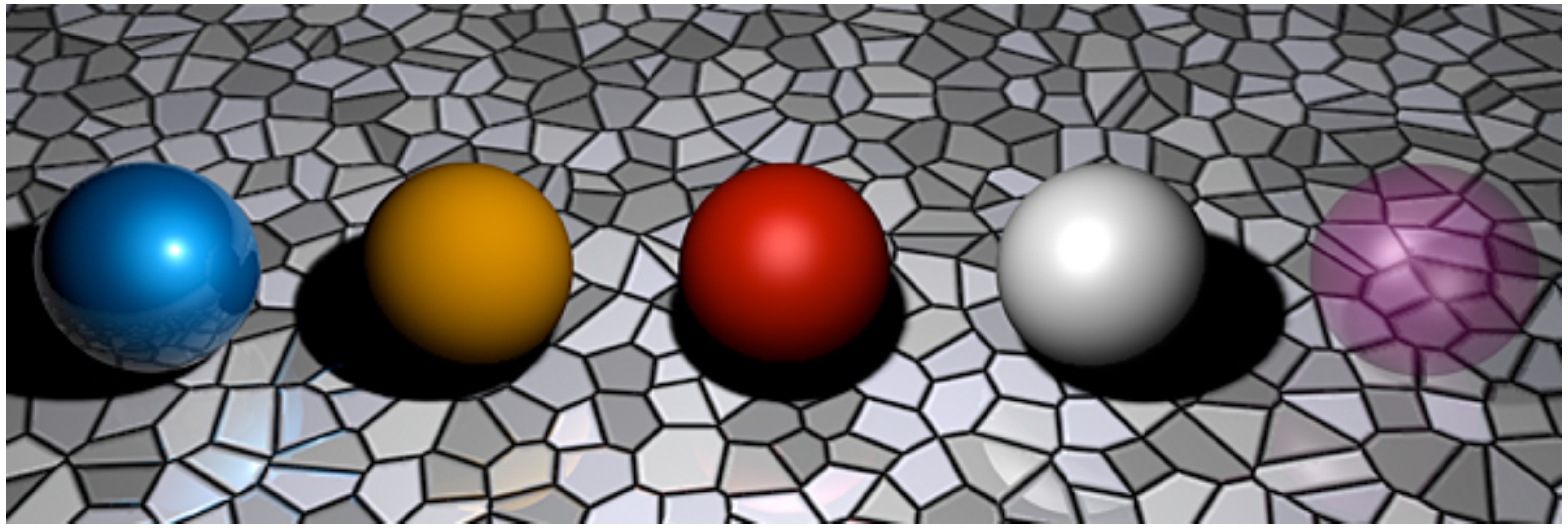
Spot Light (cone)



Point Light with
falloff

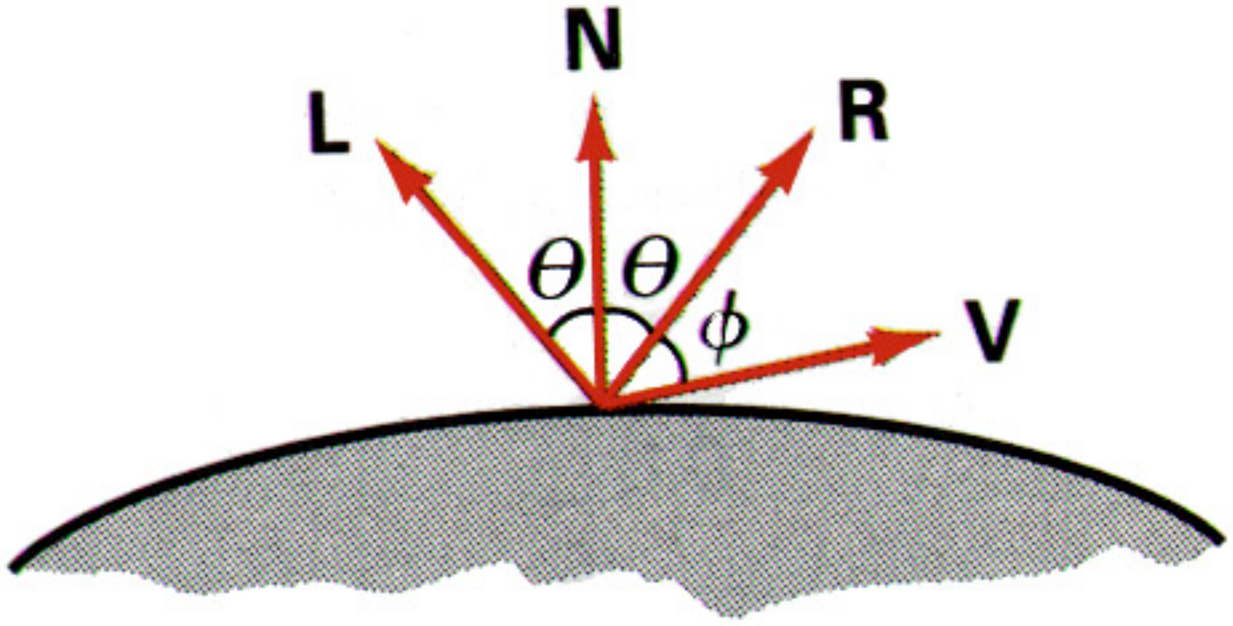


Area Light



Shading Model

- We will consider a surface, **S**, illuminated by a point light source.
- The surface is struck by the cast ray, **V**, where **V** gives the direction of the point of view.
- The orientation of the surface is given by the surface normal, **N**. The vector from the point on the surface to the light source is **L**.
- **R** is the direction of a perfect reflection. **R** is determined from **L** and **N** by noting that the angle of incidence equals the angle of reflection for a perfect mirror.



Shading Model (cont.)

- When the light hits the surface some energy is reflected while the rest is absorbed.
- An ideal mirror reflects light in a single direction.
- A diffuse reflector (surface) reflects an incident ray in all directions.
- Specular reflections are highlights about the angle of reflection.

Diffuse Reflections

- Light reflected from a perfectly diffuse surface is scattered equally in all directions. The surface appears the same to all viewers ($\mathbf{V}$ is not important).
- The component which is scattered depends on the cosine of the angle between $\mathbf{L}$ and $\mathbf{N}$ – *Lambert's Law*.

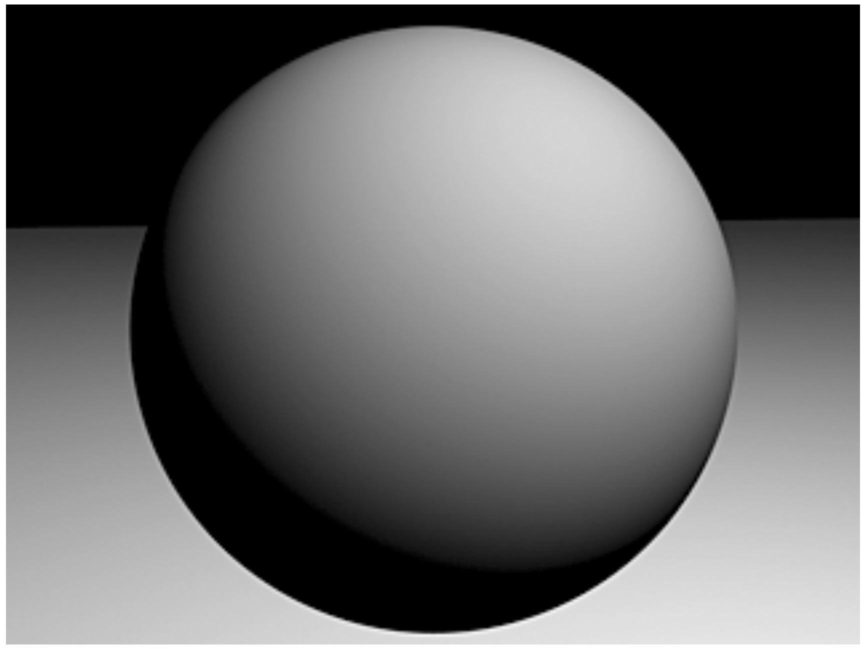
Diffuse Illumination

- Since: $\cos\theta = \frac{L \cdot N}{|L||N|}$

If the incoming light has intensity I_S and we allow for a fraction, k_d, $0 \le k_d \le 1$ which is diffused, then the reflected intensity is given by:

$$I_{diff} = \frac{I_s k_d L \cdot N}{|L||N|}$$

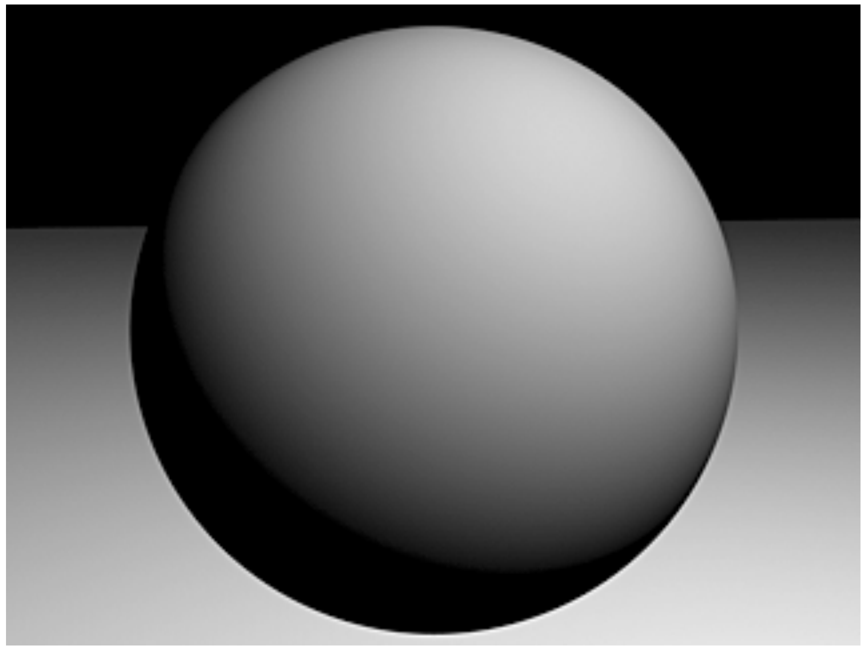
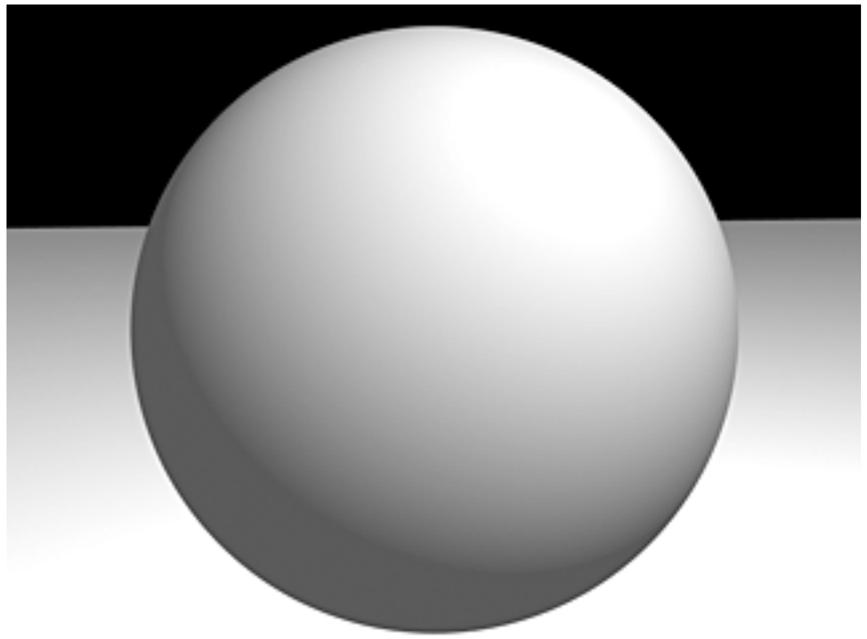
This model can be extended to include the effect of the light intensity diminishing with the square of the distance from the source.



Ambient Illumination

- Point sources for light can produce harsh images that appear unrealistic with high contrast. This problem can be alleviated to some extent by adding more point lights to a scene.
- However, in modelling real light sources and environments, we might need so many that it becomes impractical. A compromise is to use an ambient light level, $\mathbf{I_a}$. This light will be partially absorbed at a point on the surface, and so the contribution from the ambient light is of the form:

$$I_{amb} = I_a k_a$$

Specular Reflection

- The smoother an object is, the more like a mirror it will behave: the rougher the surface of an object is, the greater the diffuse reflection. If the surface is smooth enough, we can notice *specular highlights* in an image.

- This occurs because a significant amount of incoming light is being reflected at angles near that of a perfect reflector (around $\mathbf{R}$).

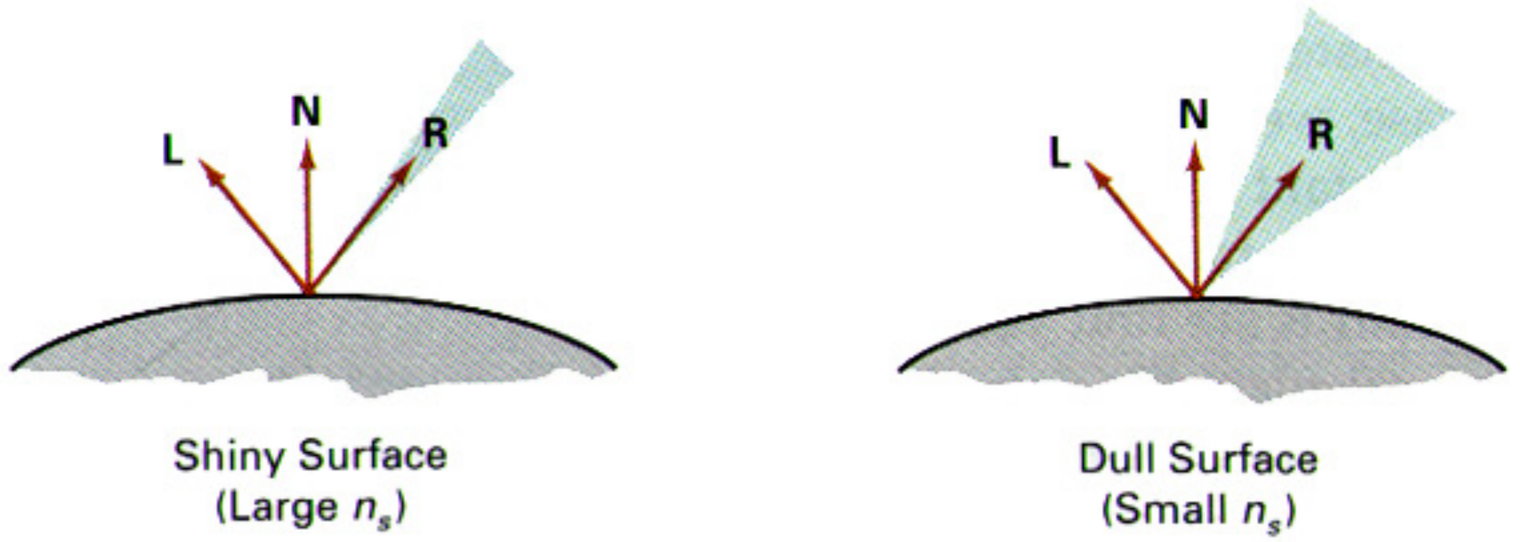
- Physical principles can be used to model such phenomena but this tends to be expensive in computational terms.

- Less computationally expensive models can be used to model the phenomenon *approximately*.

Specular Reflection (cont.)

- An empirical model, developed by Phong Bui Tuong, known as the Phong specular-reflection model or simply the Phong model, sets the intensity of specular reflection proportional to , $\cos^{n_s} \phi$ where Φ is the angle between **R** and **V**.

- Angle Φ may be assigned the values between 0 and 90°. The value assigned to the specular-reflection parameter n_S, is determined by the type of surface.



Modeling specular reflections (shaded area) with parameter n_s.

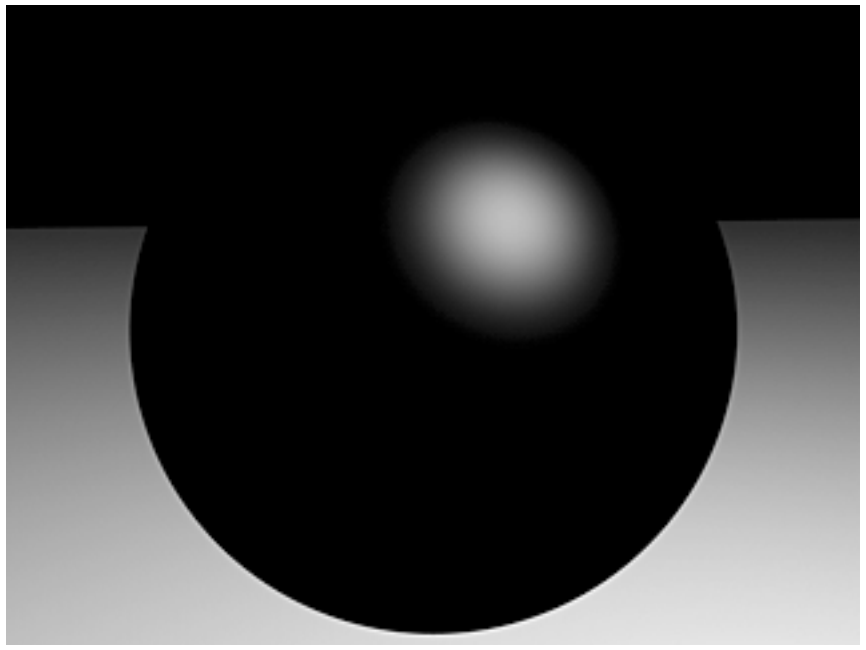
Specular Reflection (cont.)

- The specular reflection contribution can be written:

$$I_{spec} = I_s k_s \cos^{n_s} \phi$$

$$= I_s k_s \left(\frac{V \cdot R}{|V||R|} \right)^{n_s}$$

The intensity of specular reflection depends on an adsorption constant, k_s, dependent on the material.

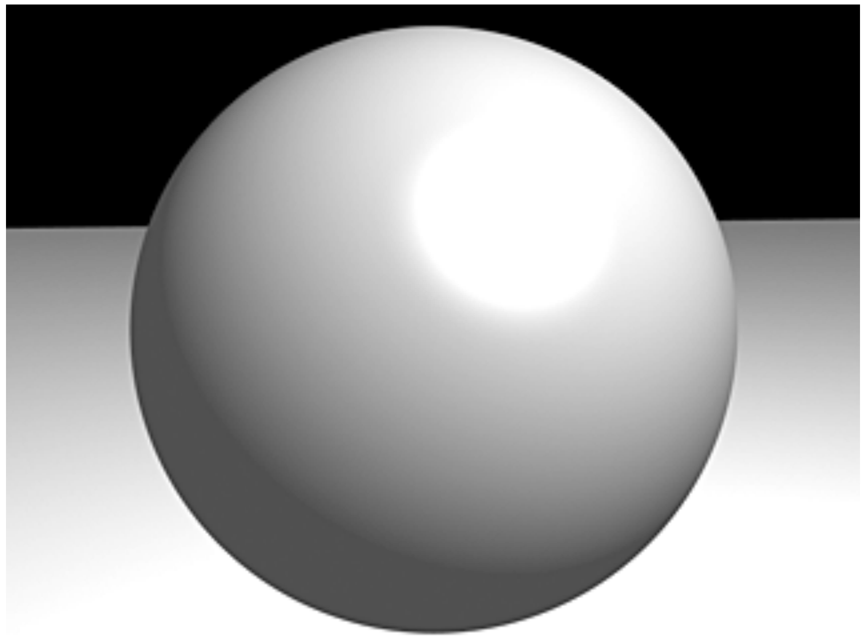


An Illumination Model

- Assuming unit vectors, a basic model with a single point light source incorporating ambient, diffuse and specular components is of the form:

$$I = \frac{I_s}{d + d_0}(k_d \hat{L} \cdot \hat{N} + k_s (\hat{R} \cdot \hat{V})^{n_s}) + I_a k_a$$

The intensity of the light source can be scaled according to the distance of the light to the surface. In reality this follows an inverse square law, however a linear approximation may provide better looking images and allow easier positioning of lights within the scene.



OpenGL Lighting

- Steps to specifying lighting in OpenGL:

  1. Define normal vectors for each vertex of every object.
  2. Create, select, and position one or more light sources.
  3. Create and select a *lighting model*, which defines the level of global ambient light and the effective location of the viewpoint (for the purposes of lighting calculations).
  4. Define material properties for the objects in the scene.

Lighting in OpenGL: Normals

- For surfaces to be correctly lit in OpenGL requires specification of surface normals:

```
        glNormal3f(nx, ny, nz);
```

- Normals are part of the OpenGL state and bound to vertices when the call to `glVertex` is made.
- OpenGL does not calculate normals for you (except for gluCurvedSurface). Your program needs to supply the normals to OpenGL.
- Normally we want to supply unit normals:
  - Normal length can be affected by transformations
  - In particular, scaling does not preserve normal length (or even direction)
  - `glEnable(GL_NORMALIZE)` allows for autonormalization at a performance penalty

OpenGL Lights

- OpenGL allows up to 8 lights to be active at any one time (may be implementation dependent);
- Two types of lights are supported:
  - Local (point) lights; Infinite (directional) lights;
  - The light type is determined by the *w* co-ordinate:

    w = 0 : Infinite light directed along (x,y,z);

    w ≠ 0 : Point light positioned at (x/w, y/w, z/w);

```
glLightfv(light, property, value)
```

specifies which light (e.g. GL_LIGHT0)

properties include: colour, position, type, attenuation

value of the property

OpenGL Lights: example

```
GLfloat light0_pos[] = {1.0, 2.0, 3.0, 1.0};

GLfloat ambient0[] = {1.0, 0.0, 0.0, 1.0};
GLfloat diffuse0[] = {1.0, 0.0, 0.0, 1.0};
GLfloat specular0[] = {1.0, 1.0, 1.0, 1.0};
GLfloat lmodel_ambient[]= {0.1, 0.1, 0.1, 1.0};
glEnable(GL_LIGHT0);

glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambient0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
                            lmodel_ambient);
```

Spotlights in OpenGL

- A local light can be converted into a spotlight
- By setting the `GL_SPOT_DIRECTION`, `GL_SPOT_CUTOFF`, and `GL_SPOT_EXPONENT`, the local light will shine in a direction and its light will be limited to a cone centred around that direction vector.

```
GLfloat light0_spotDir[3] = {0.0F, 0.0F, -1.0F};

glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 0.0F);

glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 180.0F);

glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, light0_spotDir);
```

Two-sided lighting in OpenGL

- Lighting calculations are performed for all polygons, both front and back facing. However, back facing polygons may not light correctly.
- You can render the front and back sides of a surface correctly, using front and back materials by invoking the function:
- `glLightModeli(GL_LIGHT_MODEL_TWO_SIDED, GL_TRUE);`
- You can control which faces are considered to be front facing using the command:

  `glFrontFace(GL_CCW)` — counter clockwise orientation

  `glFrontFace(GL_CW)` — clockwise orientation

Positing Light Sources in OpenGL

- Light sources are geometric objects whose positions or directions are affected by the model-view matrix
- Depending on where we place the position (direction) setting function, we can
  - ★ Move the light source(s) with the object(s)
  - ★ Fix the object(s) and move the light source(s)
  - ★ Fix the light source(s) and move the object(s)
  - ★ Move the light source(s) and object(s) independently
- The light source position is bound when `glLightfv(GL_LIGHT`*n*`, GL_POSITION, light_position)` is called.

Specifying Materials

- Material properties in OpenGL match the Phong reflection model
- Different material properties can be specified for the front and back sides of a surface
- The properties give slightly different results to what you would expect, the visual result is as follows:
  - `GL_DIFFUSE` - base color of object
  - `GL_SPECULAR` - color of highlights on object
  - `GL_AMBIENT` - color of object when not directly illuminated
  - `GL_EMISSION` - color emitted from the object (Unaffected by other light sources, adds a fixed colour to the surface)
  - `GL_SHININESS` - concentration of highlights on objects. Values range from 0 (very rough surface - no highlight) to 128 (very shiny)

OpenGL Material Example

```
GLfloat ambient[] = {1.0, 0.0, 0.0, 1.0};
GLfloat diffuse[] = {1.0, 0.0, 0.0, 1.0};
GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
glMaterialf(GL_FRONT, GL_SHININESS, 100.0);
```

- We can simulate a physical light source in OpenGL by giving a material an emissive component. This component is unaffected by any sources or transformations:

```
GLfloat emission[] = 0.0, 0.3, 0.3, 1.0);
glMaterialf(GL_FRONT, GL_EMISSION, emission);
```

Enabling Lighting in OpenGL

- Lighting is normally used in conjunction with depth buffering and normal RGBA colour mode.

- Lighting as a whole can be turned on and off, as can individual lights:

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

- Smooth shading (Gouraud interpolation) is enabled with:

```
glShadeModel(GL_SMOOTH);
```

  (use `GL_FLAT` to turn it off). Note that vertex normals must be correctly specified for Gouraud interpolation to work.