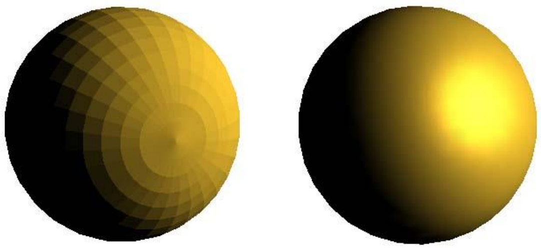
Let there be light...

5 Illumination

Department of Computer Science and Software Engineering
University of Canterbury, New Zealand.

R. Mukundan (mukundan@canterbury.ac.nz)

Shades of Colour



The surface colour at a point on an illuminated object depends on:

- Surface orientation, view position relative to the light source
  - Normal vector, light source vector, view vector
- Illumination Model
  - Local illumination, global illumination
- Light Characteristics
  - Colour, position, direction, attenuation.
  - Point light, area light, spotlight.
- Material Properties
  - Ambient, diffuse and specular reflections, advanced models
- Shading Model (introduced later)
  - Flat shading, Gouraud Shading, Phong Shading

Light-Material Interaction

- In OpenGL, the interaction between the light and the material properties is simply modelled as the component-wise product of the light colour and the material colour.

Light colour (0., 1., 1.) * Material colour (1., 0., 1.) = Perceived colour of the object (0., 0., 1.)

(l_1, l_2, l_3) * (m_1, m_2, m_3) = $(l_1m_1,\ l_2m_2,\ l_3m_3)$

Ambient, Diffuse and Specular Components

- Both light and material have 3 components: ambient, diffuse and specular. Each component is a colour value.

	Light	Material	Result	
Ambient:	L_a (0.2, 0.2, 0.2, 1.0)	M_a (0.0, 0.0, 1.0, 1.0)	$L_a M_a$ (0.0, 0.0, 0.2, 1.0)	
Diffuse:	L_d (1.0, 1.0, 1.0, 1.0)	M_d (0.0, 0.0, 1.0, 1.0)	$L_d M_d\,(\boldsymbol{l}\bullet\boldsymbol{n})$ (0.0, 0.0, 1.0, 1.0)	
Specular:	L_s (1.0, 1.0, 1.0, 1.0)	M_s (1.0, 1.0, 1.0, 1.0)	$L_s M_s\,(\boldsymbol{r}\bullet\boldsymbol{v})^f$ (1.0, 1.0, 1.0, 1.0)	

Ambient: Provides constant illumination, but no surface details.

Diffuse: Varies with respect to the angle between the light source vector (l) and the normal vector (n).

Specular: Varies with respect to the angle between the reflection vector (r) and the view vector (v).

Total:

Setting Up Light Sources

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHTn);
glLightfv(GL_LIGHTn,  GL_AMBIENT   ,  array );
                      GL_DIFFUSE
                      GL_SPECULAR
                      GL_POSITION
```

Light number (0 .. 7)

Colour or position

Position is a *point* (d = 1) for a positional light source or a *vector* (d = 0) for a directional one.

- The light position can be transformed just like any other vertex using OpenGL transformations.
- Always specify colours as 4-tuples: (red, green, blue alpha)
  - Alpha: 1 = opaque, 0 = transparent

Setting Material Properties

To set material properties, use:

$$\texttt{glMaterialf[v](}\left\{\begin{array}{l}\texttt{GL\_FRONT}\\ \texttt{GL\_BACK}\\ \texttt{GL\_FRONT\_AND\_BACK}\end{array}\right\}\texttt{,}\left\{\begin{array}{l}\texttt{GL\_EMISSION}\\ \texttt{GL\_AMBIENT}\\ \texttt{GL\_DIFFUSE}\\ \texttt{GL\_AMBIENT\_AND\_DIFFUSE}\\ \texttt{GL\_SPECULAR}\\ \texttt{GL\_SHININESS}\end{array}\right\}\texttt{, array)}$$

The polygon face(s) affected

The parameter(s) being set

OpenGL Lighting: Example

```
void initialize()
{
  float lgt_amb[4] = {0.0, 0.0, 0.0, 1.0};
  float lgt_dif[4] = {1.0, 1.0, 1.0, 1.0};
  float lgt_spe[4] = {1.0, 1.0, 1.0, 1.0};

  float mat_amb[3] = {0.0, 0.0, 1.0, 1.0};
  float mat_dif[3] = {0.0, 0.0, 1.0, 1.0};
  float mat_spe[3] = {1.0, 1.0, 1.0, 1.0};

  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  glLightfv(GL_LIGHT0, GL_AMBIENT, lgt_amb);
  glLightfv(GL_LIGHT0, GL_DIFFUSE, lgt_dif);
  glLightfv(GL_LIGHT0, GL_SPECULAR, lgt_spe);

  glMaterialfv(GL_FRONT, GL_AMBIENT, mat_amb);
  glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_dif);
  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spe);
  glMaterialf(GL_FRONT, GL_SHININESS, 50);
  ...
}
```

OpenGL Lighting: Example

= 0 for directional light

```
void display()
{
  float lgt_pos[4]={0., 10., 10., 1.};
  glClear(GL_COLOR_BUFFER_BIT |  GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();

  gluLookAt(5., 3., 2., 0., 0., 0., 0., 1., 0.);

  glLightfv(GL_LIGHT0, GL_POSITION, lgt_pos);

  glTranslatef(0.0, 1.2, 0.0);
  glRotatef(angle, 0.0, 1.0, 0.0);

  glutSolidTeapot(1.0);
  glFlush();
}
```

Light source moves with the object

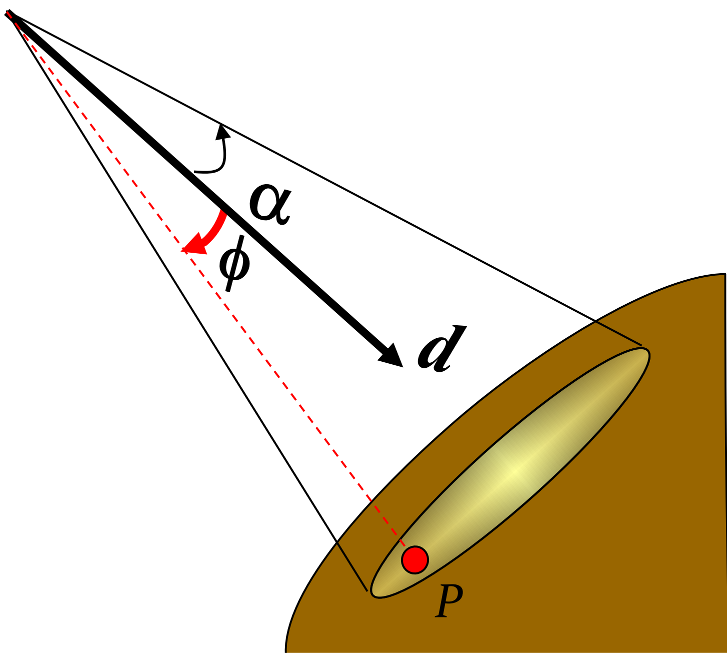
Light source's position fixed in the scene

Light source fixed relative to the camera.

A point light source is transformed like any other point

Spot Light

- Spotlights emit light in set of directions restricted to a cone whose axis direction is ***d***, and half cone angle α.
- The intensity at a vertex *P* at an angle ϕ is attenuated by a spotlight attenuation factor $\cos^{\varepsilon}(\phi)$, where ε is a constant.

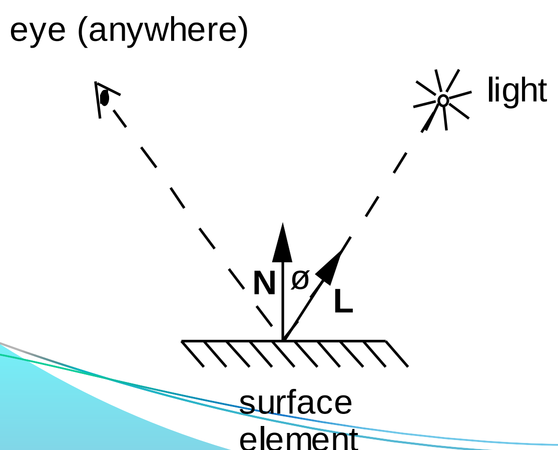


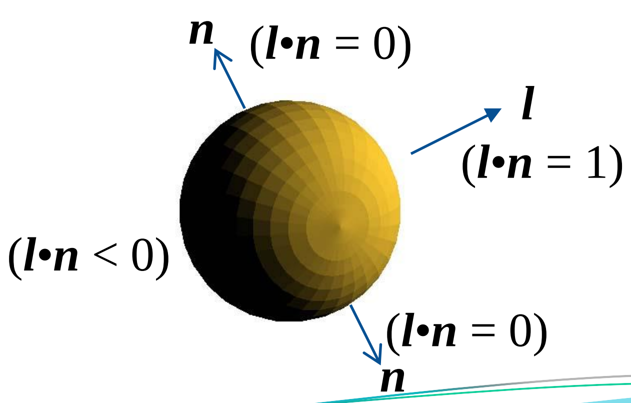

```
float spotdir[]={5.0, -2.0, -4.0};
glLightf(GL_LIGHT2, GL_SPOT_CUTOFF, 10.0);
glLightf(GL_LIGHT2, GL_SPOT_EXPONENT, 2.0);
glLightfv(GL_LIGHT2, GL_SPOT_DIRECTION,spotdir);
```

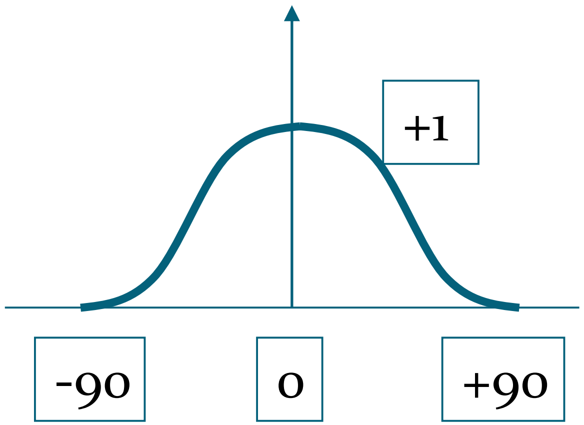
Diffuse Reflection

- An ideal diffuse surface is a "perfect scatterer" of light.
  - The outgoing direction of a scattered photon is uncorrelated with its initial direction of arrival
- Lambert's Law: Surface is seen to have the same intensity (i.e. shade of grey) from all view directions
  - Hence terms Lambert surface, Lambertian reflection

$$\boxed{\boldsymbol{I}_d = L_d M_d \max(\boldsymbol{l}\bullet\boldsymbol{n}\ ,\ 0)}$$

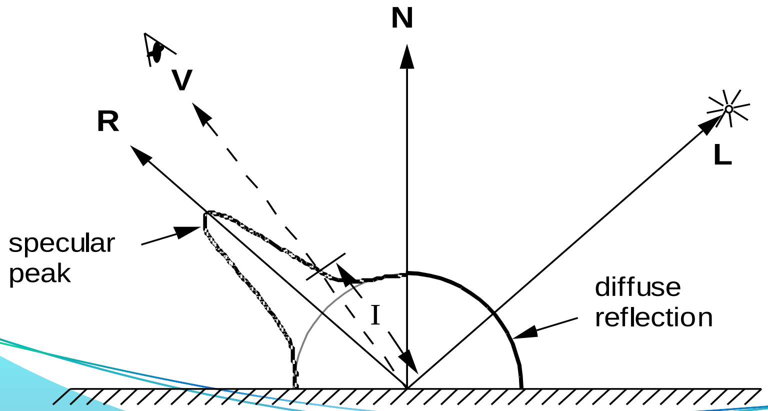






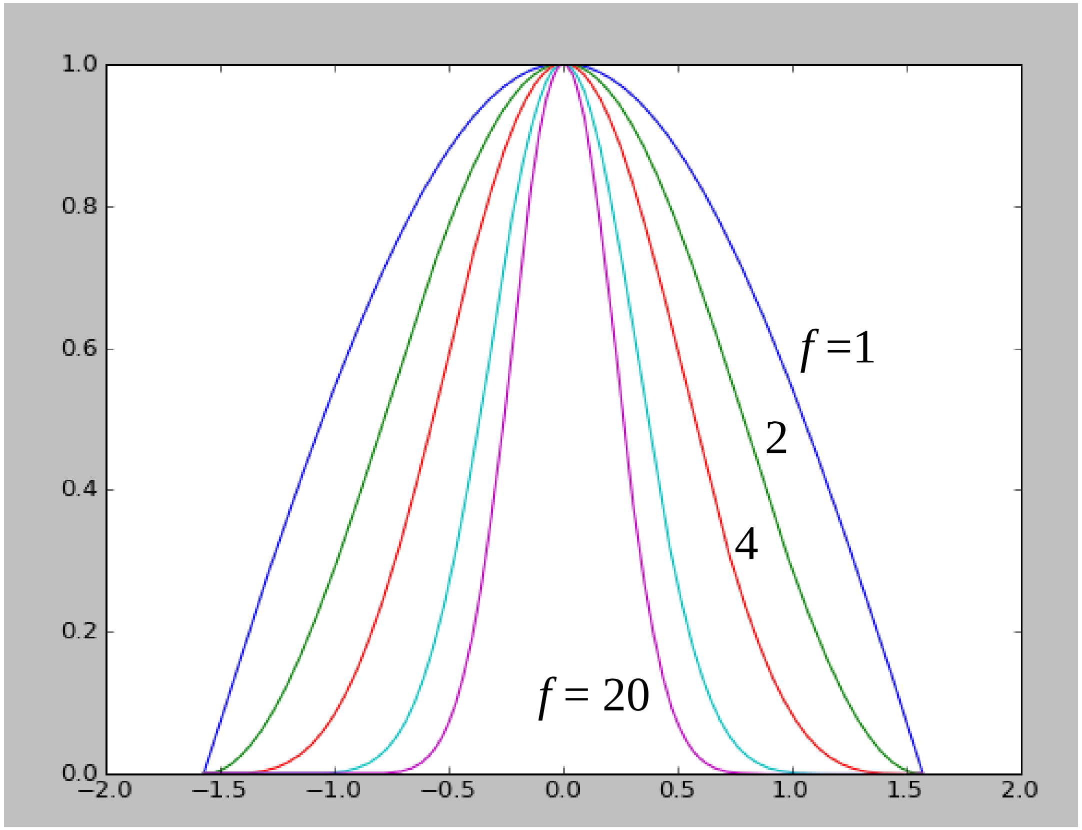
Specular Reflection

- Specular reflection or "highlight" provides shininess to the surface.
- Highlight occurs at mirror reflection angle, along the reflection of $\boldsymbol{l}$ about $\boldsymbol{n}$. This vector is denoted by $\boldsymbol{r}$.
- Intensity of the highlight reduces as the viewer $\boldsymbol{v}$ moves away from $\boldsymbol{r}$.
- The Phong's constant (or shininess) f controls the width (or spread) of the highlight. Ref. Code on slide 7.



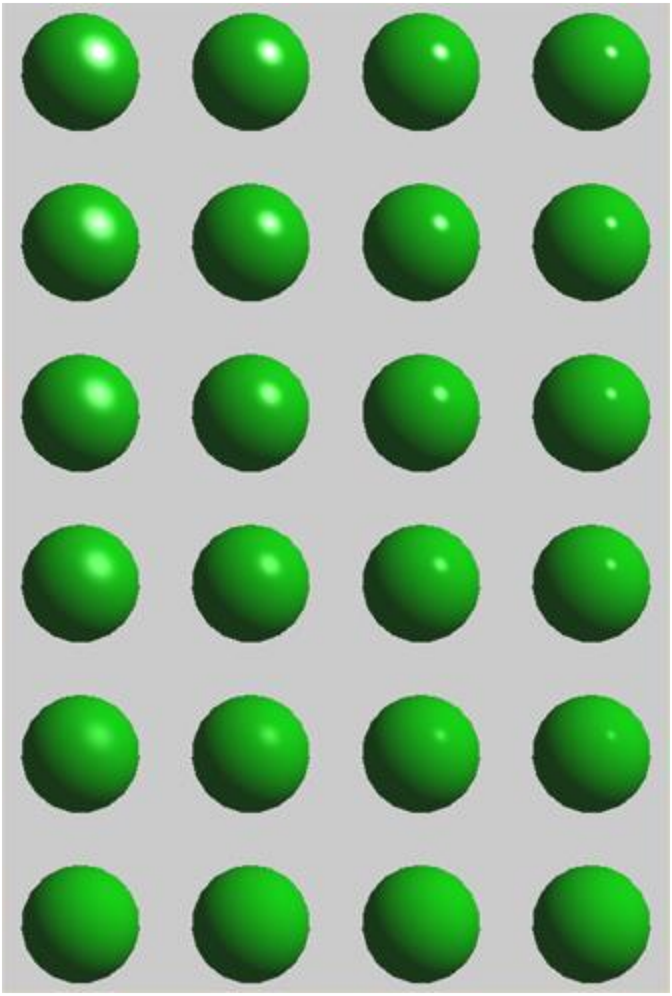
$$\boldsymbol{I}_s = L_s M_s (\boldsymbol{r} \bullet \boldsymbol{v})^f$$

Phong's Constant



- Large values of the exponent f gives highly concentrated highlights.

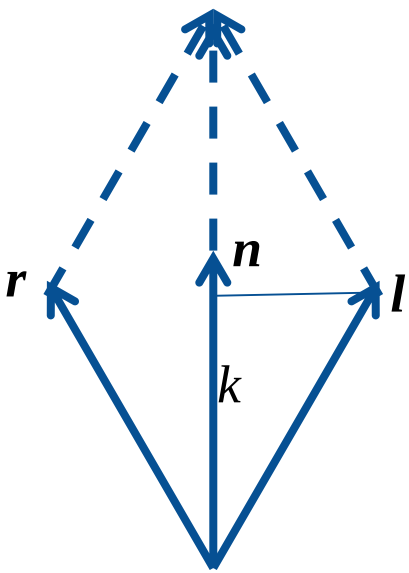
Specular Reflection: Example



$L_a = (0.2, 0.2, 0.2)$
$L_d = (1.0, 1.0, 1.0)$
$L_s = (1.0, 1.0, 1.0)$

$M_a = (0.0, 1.0, 0.0)$
$M_d = (0.0, 1.0, 0.0)$

Computation of Reflection Vector

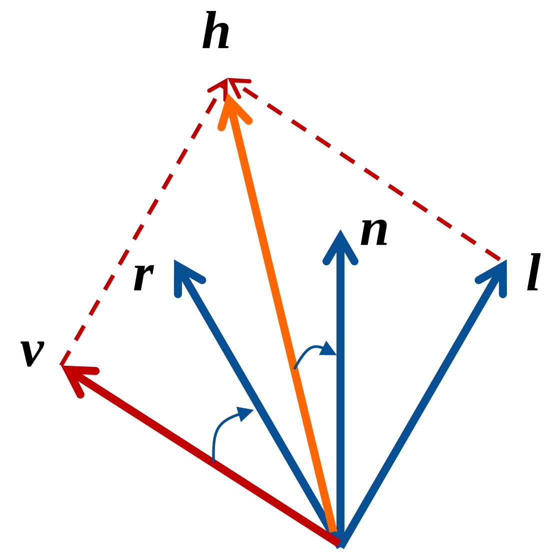


$k = (\boldsymbol{l} \bullet \boldsymbol{n})$

$\boldsymbol{r} + \boldsymbol{l} = 2k\,\boldsymbol{n}$

Therefore,

$\boldsymbol{r} = 2\,(\boldsymbol{l} \bullet \boldsymbol{n})\,\boldsymbol{n} - \boldsymbol{l}$



- $\boldsymbol{I}_s = L_s M_s\,(\boldsymbol{r} \bullet \boldsymbol{v})^f$
- OpenGL uses an approximation of $(\boldsymbol{r} \bullet \boldsymbol{v})$, by the term $(\boldsymbol{h} \bullet \boldsymbol{n})$
- The vector $\boldsymbol{h}$ is called the "Half-way Vector", and computed as $\boldsymbol{h} = (\boldsymbol{l} + \boldsymbol{v})$ normalized.
- Simplified formula for specular reflection: $\boldsymbol{I}_s = L_s M_s\,(\boldsymbol{h} \bullet \boldsymbol{n})^f$

Geometrical Considerations

- The rendering speed is increased if $\boldsymbol{v}$ is made constant for all vertices (infinite viewpoint). This is the default setting in OpenGL.

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_FALSE);
```

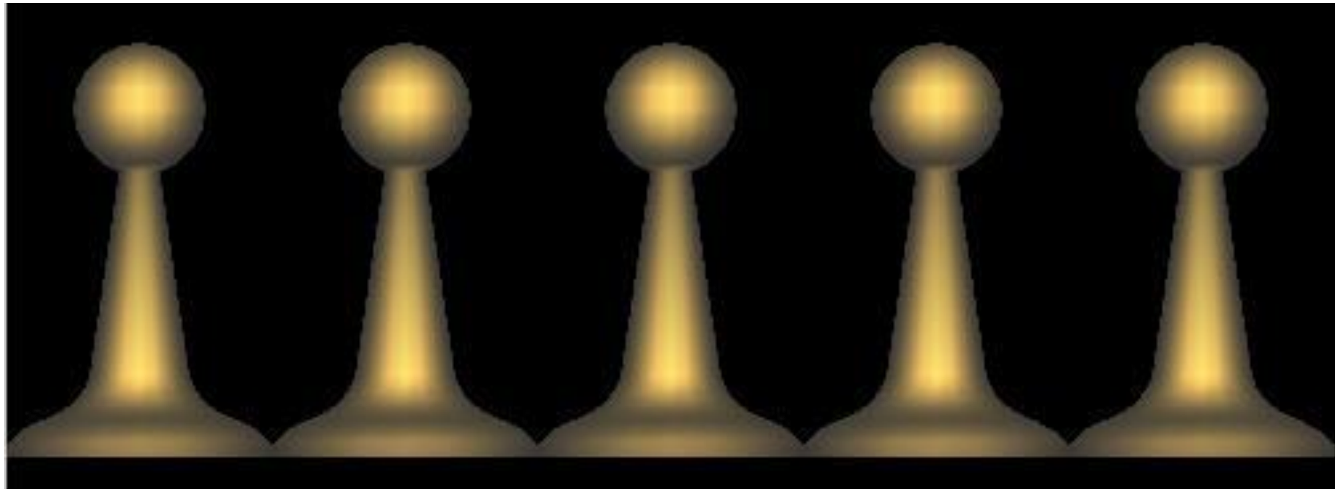
- You can force the computation of the true value of $\boldsymbol{v}$ for each vertex (local viewpoint) for more realistic results:

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
```

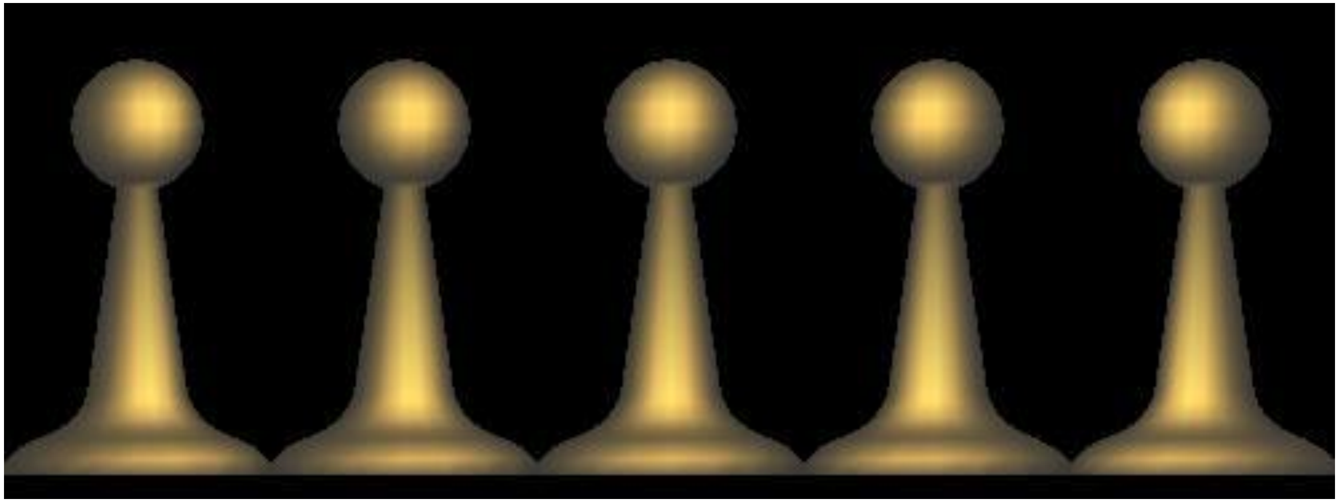
- For a light source at infinity (directional source) the vector $\boldsymbol{l}$ is constant for all vertices in the scene.

Local and Infinite Viewpoints

Infinite viewpoint



Local viewpoint



Light Attenuation

- Distance attenuation: The ambient, specular and diffuse contributions of a positional light source can all be attenuated based on the distance *D* of the vertex from the light source.

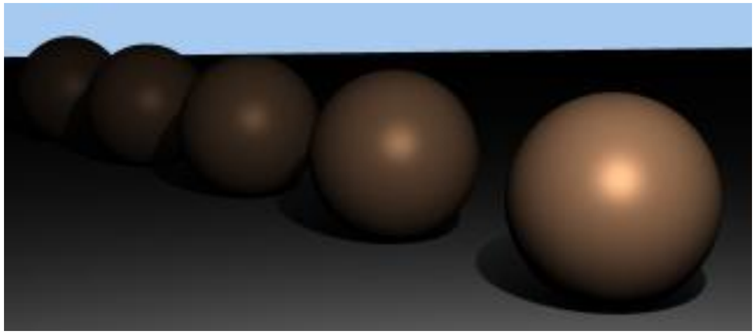
$$\frac{1}{k_c + k_l D + k_q D^2}$$

k_c = Constant attenuation factor
k_l = Linear attenuation factor
k_1 = Quadratic attenuation factor

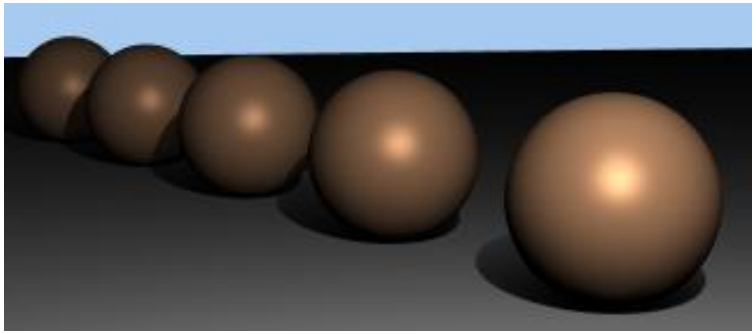
```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 0);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0);
glLightf(GL_LIGHT0,GL_QUADRATIC_ATTENUATION,.1);
```

Distance Attenuation

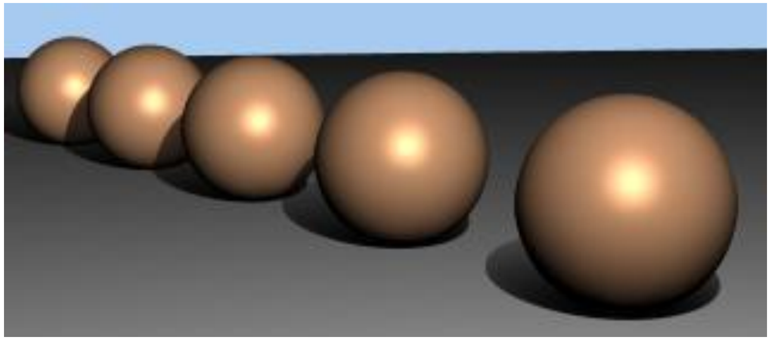
Quadratic:
$k_c = 0, \;\; k_l = 0, \;\; k_q = 1$



Linear:
$k_c = 0, \;\; k_l = 1, \;\; k_q = 0$



Constant:
$k_c = 1, \;\; k_l = 0, \;\; k_q = 0$



Notes

- For GL_LIGHT0, the default diffuse and specular color is (1,1,1,1), while for all other light sources the value is (0,0,0,0). Therefore, if you are using a light source other than GL_LIGHT0, you will need to specify its diffuse and specular colours:

```
float white[] = {1.0, 1.0, 1.0, 1.0};
glLightfv(GL_LIGHT1, GL_DIFFUSE, white);
glLightfv(GL_LIGHT1, GL_SPECULAR, white);
```

- If lighting is enabled, use 4-components for colour values and the light position. Defining colour using just 3 components may generate unpredictable results.

Notes

- If lighting is enabled, OpenGL uses the colour values defined by `glMaterialfv` (slides 6,7). Colour definitions using `glColor3f()`, `glColor4f()` etc, are ignored.
- However, we can force the material colour to track the current colour specified using `glColor4f()`. This is done by calling `glColorMaterial()` as follows:

```
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
glColor4f(0., 0., 1., 1.);
```

- The above method is usually preferred as it does not require separate array initializations.

Notes

- Lighting computation also requires the surface normal components for each primitive or each vertex. Assigning per-vertex surface normal vectors results in a smooth shading of the object. Examples:

```
glEnable(GL_NORMALIZE);
glNormal3f(nx, ny, nz);
glBegin(GL_TRIANGLES);
  glVertex3f(...);
  glVertex3f(...);
  glVertex3f(...);
glEnd();
```

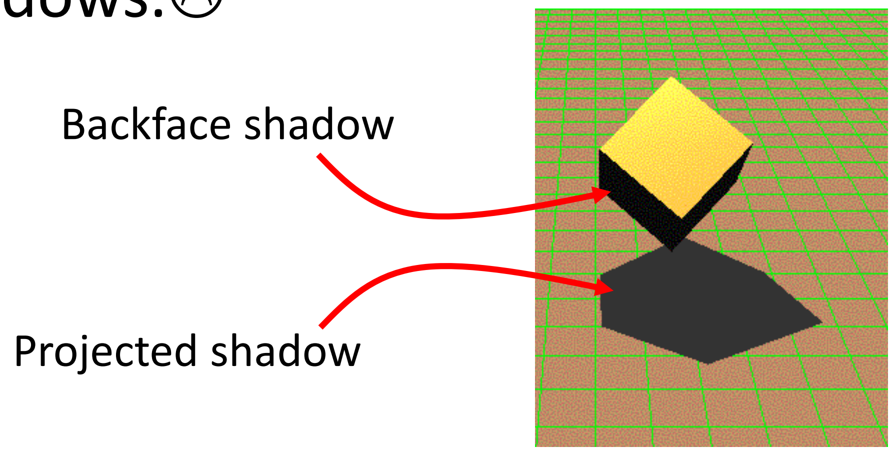
Face Normals

```
glEnable(GL_NORMALIZE);
glBegin(GL_TRIANGLES);
  glNormal3f(...);
  glVertex3f(...);
  glNormal3f(...);
  glVertex3f(...);
  glNormal3f(...);
  glVertex3f(...);
glEnd();
```

Vertex Normals

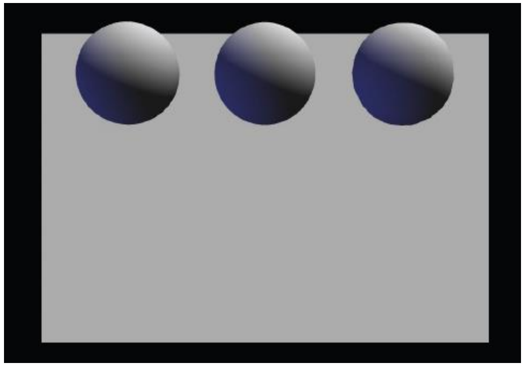
Shadows

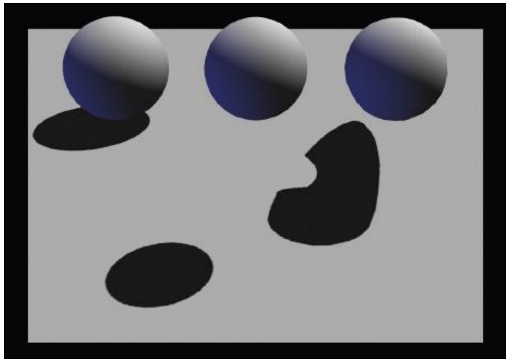
- Two types of shadows:
  - Backface shadows: A shadow on an object's surface that is oriented away from light. This type of shadows are automatically generated by the illumination model ($\boldsymbol{l} \bullet \boldsymbol{n} < 0$)
  - Projected shadows or cast shadows: Shadows cast by a part of an object's surface on either the same or a different object.
- OpenGL's lighting model cannot generate projected shadows.☹

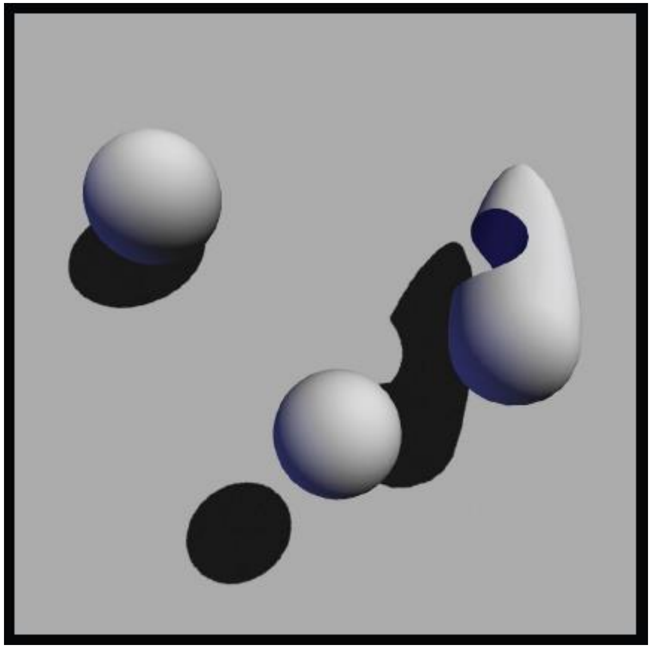


Projected Shadows

- Shadows add a great amount of realism to a scene.
- Shadows provide a second view of an object.
- Shadows convey additional information such as depth cues (eg. object's height from a floor plane) and object's shape.

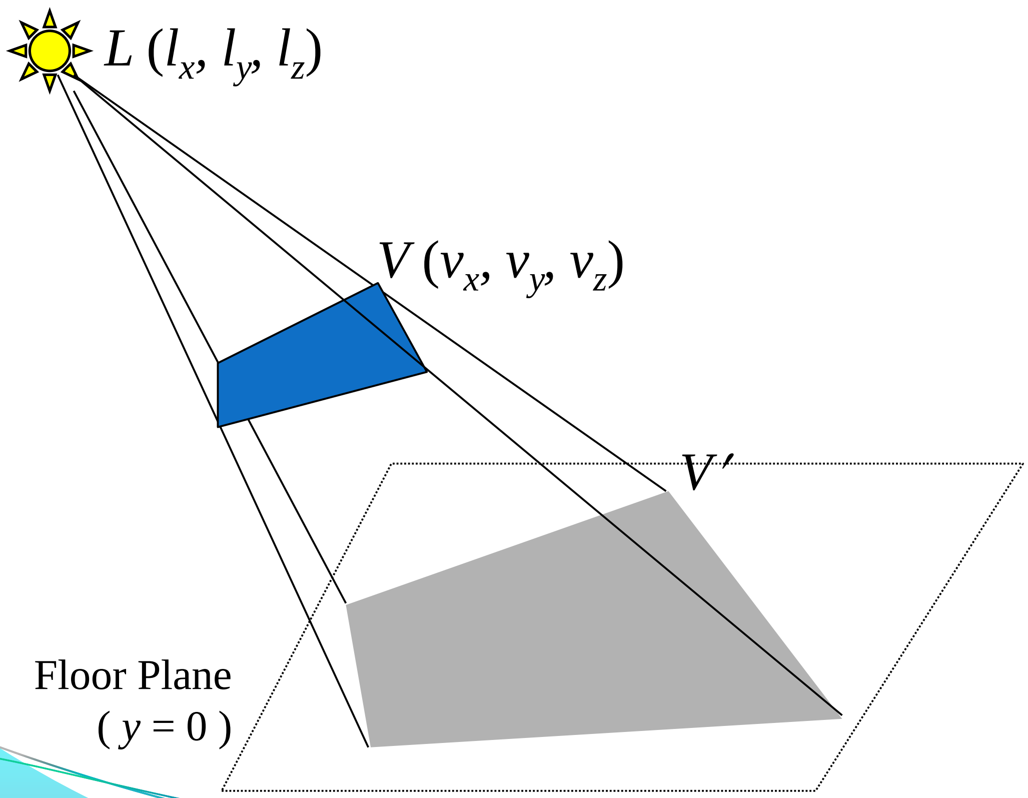






Planar Projected Shadows

- Project each of the polygonal faces onto the floor plane, using the light source (*L*) as the centre of projection.
- Use only the ambient light to draw the projected object.



$$\boxed{V' = (1-t)L + tV, \qquad t > 1.}$$

$$v'_x = (1-t)l_x + tv_x$$

$$v'_y = (1-t)l_y + tv_y = 0$$

$$v'_z = (1-t)l_z + tv_z$$

$$\therefore t = \frac{l_y}{l_y - v_y}$$

Planar Shadows

- The projection V' of the vertex $V = (v_x, v_y, v_z)$ on the floor-plane is given by the following coordinates:

$$\left.\begin{aligned} v'_x &= \frac{-l_x v_y + v_x l_y}{l_y - v_y} \\ v'_y &= 0 \\ v'_z &= \frac{-l_z v_y + v_z l_y}{l_y - v_y} \end{aligned}\right\} \xrightarrow{\text{Homogeneous Coordinates}} \begin{aligned} s_x &= -l_x v_y + v_x l_y \\ s_y &= 0 \\ s_z &= -l_z v_y + v_z l_y \\ w &= l_y - v_y \end{aligned}$$

- The above equations can be written as a transformation:

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ w \end{bmatrix} = \begin{bmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix}$$

Planar Shadows: Code

```
// Light source position = (lx, ly, lz)
float shadowMat[16] = { ly,0,0,0, -lx,0,-lz,-1,
                        0,0,ly,0,  0,0,0,ly };
 glDisable(GL_LIGHTING);
 glPushMatrix();     //Draw Shadow Object
   glMultMatrixf(shadowMat);
   /* Transformations */
   glColor4f(0.2, 0.2, 0.2, 1.0);
   drawObject();
 glPopMatrix();

 glEnable(GL_LIGHTING);
 glPushMatrix();     //Draw Actual Object
    /* Transformations */
    drawObject();
 glPopMatrix();
```