
CM0304 Graphics

II Geometric Modelling

II.2 Illumination Models

Xianfang Sun

F. C. Langbein

School of Computer Science
Cardiff University



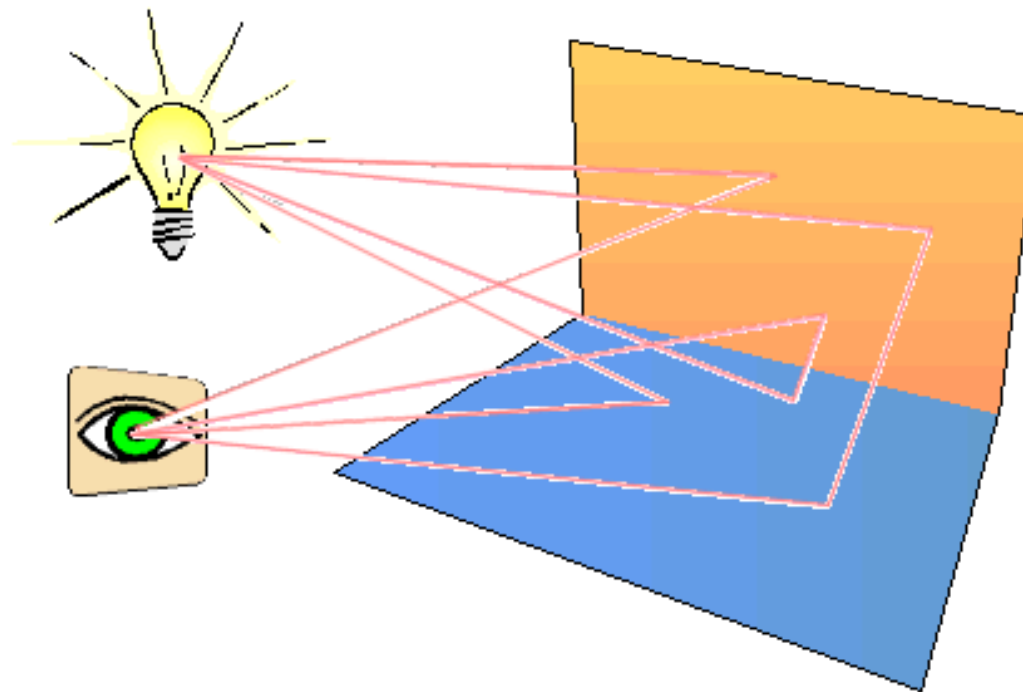


Overview

- Phong illumination model
 - Light source types
 - Reflection models
- OpenGL lighting

■ Illumination Models

- *Illumination*: transport of luminous flux from light sources between points via direct and indirect paths
- *Lighting*: computing luminous intensity reflected from a specific 3D point
- *Shading*: assigning colours to a pixel



Luxo Jr.

➤ First Pixar animation:



- See <http://www.pixar.com/shorts/index.html> for this and more...



Light-Material Interaction

- Light that strikes an object is partially absorbed and partially reflected
- The amount reflected determines the colour and brightness of the object (subtractive colours)
- Reflected light is scattered depending on smoothness and orientation of surface
 - *Ambient* light: comes from all directions, is scattered in all directions
 - *Diffuse* light: comes from one direction, is scattered in all directions
 - *Specular* light: comes from one direction, reflected in preferred direction (highlights)

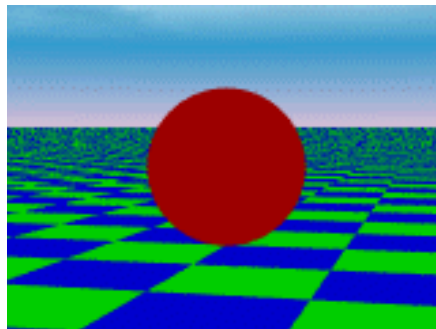
Light Sources and Material

- Light source types: *ambient*, *directional*, *point*, *spot* light
- For each light source of type T (ambient, directional, point, spot) emitting light of type t (ambient, diffuse, specular) and colour c (red, green, blue) we have to model $I_T(p, L_{t,c})$:
 - *Intensity* of light in $[0, 1]$ of type t , colour c
 - *emitted* from light source of type T
 - at *intensity* $L_{t,c} \in [0, 1]$
 - *arriving* at 3D position $p \in \mathbb{R}^3$
- Surface *material*:
 - Specifies how incoming light at intensity $I_T(p, L_{t,c})$ is *reflected* by surface for each light type and colour:
 - $R_{t,c} \in [0, 1]$ for all light types t and all colours c

Ambient Light Source

- An object not directly lit is still visible
 - Caused by light reflected from other surfaces
- Modelled by a single ambient light source
 - Instead of computing surface reflections, specify *constant ambient light* for all surfaces
 - Defined solely by ambient RGB light *intensities*
- Intensity of ambient light of intensity $L_{\text{ambient},c}$ arriving at point p :

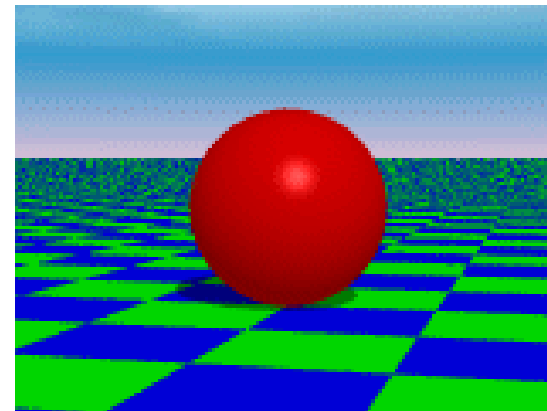
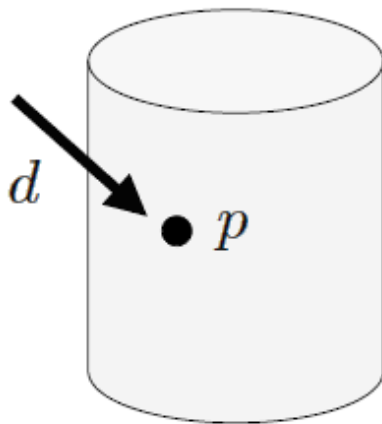
$$I_{\text{ambient}}(p, L_{\text{ambient},c}) = L_{\text{ambient},c}$$



Directional Light Source

- Light from a source *infinitely far away*
 - Defined by *intensities* of emitted RGB light of all types,
 - *direction* \mathbf{d} , $\|\mathbf{d}\| = 1$ (and no position)
- Intensity of directional light of intensity $L_{t,c}$ arriving at point p :

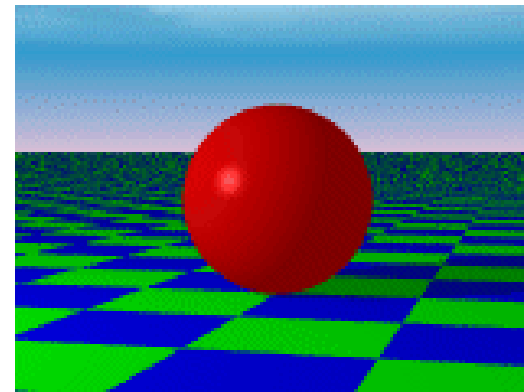
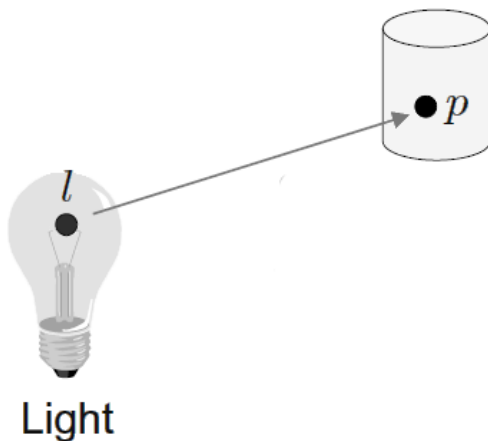
$$I_{\text{directional}:\mathbf{d}}(p, L_{t,c}) = L_{t,c}$$



Point Light Source

- Light emitted *radially* from single point *in all directions*
 - Defined by *intensities* of emitted RGB light for all types,
 - *position* l (and no direction),
 - constant, linear and quadratic *attenuation* (k_c, k_l, k_q)
- Intensity of point light of intensity L_c arriving at point p :

$$I_{\text{point:l,k}_c,\text{k}_l,\text{k}_q}(p, L_{t,c}) = \frac{1}{k_c + k_l \|p - l\| + k_q \|p - l\|^2} L_{t,c}$$

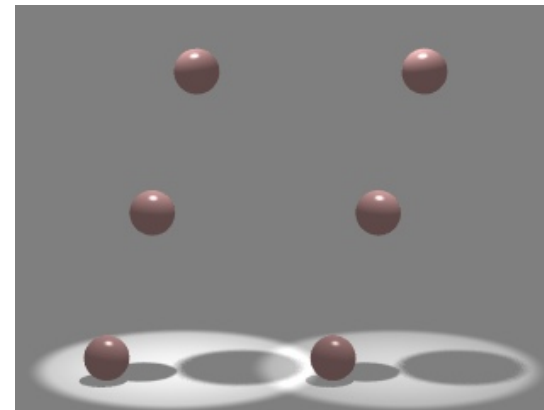
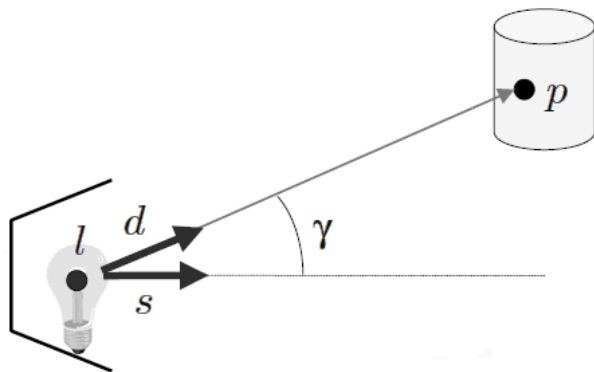


Spot Light Source

- Light emitted in a *cone* (e.g. Luxo Jr. lamp)
 - Defined by *intensities* of emitted RGB light for all types,
 - *position* \mathbf{l} , *unit cone direction* \mathbf{s} , *spot cut-off exponent* τ ,
 - constant, linear and quadratic *attenuation* (k_c, k_l, k_q)

- Intensity of point light of intensity $L_{t,c}$ arriving at point p :

$$I_{\text{spot:l,s,\tau,k_c,k_l,k_q}}(\mathbf{p}, L_{t,c}) = \frac{(s^t((\mathbf{p} - \mathbf{l}) / \|\mathbf{p} - \mathbf{l}\|))^\tau}{k_c + k_l \|\mathbf{p} - \mathbf{l}\| + k_q \|\mathbf{p} - \mathbf{l}\|^2} L_{t,c}$$



Light Source “Visibility”

➤ *Angle cut-off* for spot lights:

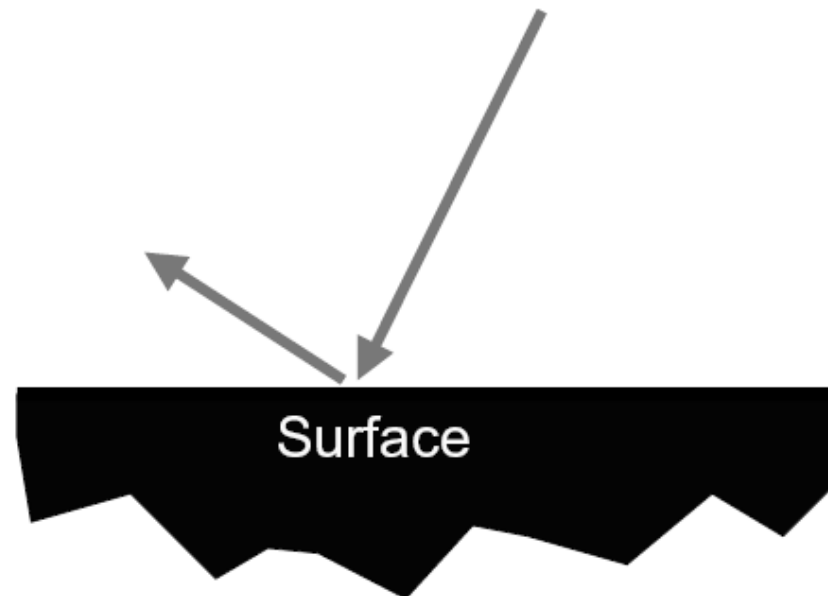
- If position p is outside light cone ($s^t d = \cos \gamma < \cos \delta$ with $d = (p - l) / \|p - l\|$ and cone semi-angle δ), set I to 0

➤ Light source *behind* surface:

- Diffuse and specular light only reflected if light source is in front of surface at p
- Set diffuse and specular light intensities from light sources to 0 if $n^t d \leq 0$
 - n : unit surface normal at p
 - d : unit direction from p to light source
- This distinguishes between front and back of surfaces / polygons (also see two-sidedness)

Modelling Surface Reflectance

- Compute light *reflected* by surface as *observed by viewer*
- Surface material tells *how much* of the incoming light is reflected
 - Type of light determines reflection model
- Intensity of observed light depends on *direction to light source* and *direction to viewer*





Ambient Reflection

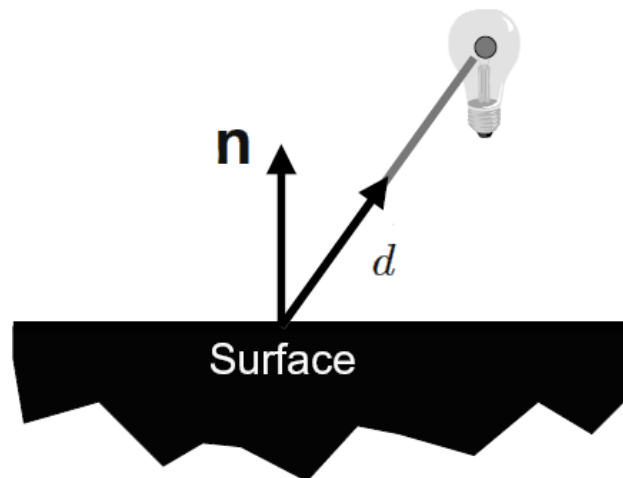
- Ambient light is the same everywhere
 - Amount of reflected light of incoming intensity $\mathbb{I}_{\text{ambient},c}$ is *independent* of direction to light source and viewer
- Intensity of reflected light observed by a viewer:

$$L_{\text{ambient},c} = R_{\text{ambient},c} \mathbb{I}_{\text{ambient},c}$$

- $R_{\text{ambient},c}$ is ambient material property for colour c
(percentage of red, green or blue ambient light reflected by surface)

Diffuse Reflection

- Light is reflected in all directions
 - Amount of reflected light of incoming intensity $I_{\text{diffuse},c}$ *depends only on direction to light source*
- *Lambertian model* (use cosine law / scalar product):
$$L_{\text{diffuse},c} = R_{\text{diffuse},c}(\mathbf{n}^t \mathbf{d}) I_{\text{diffuse},c}$$
 - \mathbf{d} : unit direction from surface point to light source
 - \mathbf{n} : unit surface normal



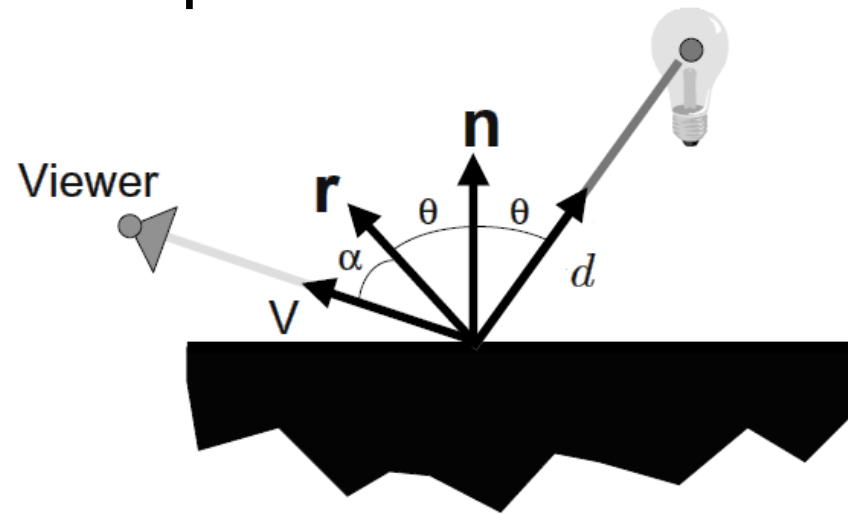
Specular Reflection

- Light is reflected preferably in *direction of perfect reflection*
 - Amount of reflected light of incoming intensity $\mathbb{I}_{\text{specular},c}$ *depends on direction to light source and to viewer*

- Observed light intensity:

$$L_{\text{specular},c} = R_{\text{specular},c} (\mathbf{r}^t \mathbf{v})^\sigma \mathbb{I}_{\text{specular},c}$$

- \mathbf{r} : unit direction of perfect reflection of d
- \mathbf{v} : unit direction towards viewer position
- σ is shininess exponent





Surface Light Emissions

- Can make surface emit light, not just reflect light
- Simple model:
 - Add *emissive* light intensities $E_{t,c}$ to light intensities for each light type t and colour c
 - Does not illuminate other surfaces
(but can add a multiple point light sources behind surface or a directional light source for larger light emitting surfaces)

Phong Illumination Model

- Putting everything together gives the *Phong Illumination Model*
- Consider monochromatic light (e.g. red, green or blue) and a single light source:
 - Depending on light source type, at a surface point the incoming intensity of different light types is $\mathbb{I}_a, \mathbb{I}_d, \mathbb{I}_s$
 - The *intensity of reflected light* is:
$$R_a \mathbb{I}_a + R_d (\mathbf{n}^t \mathbf{d}) \mathbb{I}_d + R_s (\mathbf{r}^t \mathbf{v})^\sigma \mathbb{I}_s$$
 - *Summation* over all light sources for red, green, blue gives total intensity for all colors
- Note, Phong's illumination model is *not* physically accurate

OpenGL Lighting

- Activate lighting computations in OpenGL:

```
glEnable (GL_LIGHTING);
```

- Each light source must be enabled individually:

```
glEnable (GL_LIGHT $x$ );
```

with $x = 0, 1, \dots$ (at least 8 light sources)

- For each light source the diffuse, specular and ambient light and the position can be set
 - Colour of emitted light specified in RGBA, where A is the alpha channel (for transparency, not used here)



Defining a Point Light Source

```
GLfloat diffuse[] = {1.0, 0.0, 0.0, 1.0};
GLfloat ambient[] = {1.0, 0.0, 0.0, 1.0};
GLfloat specular[] = {1.0, 0.0, 0.0, 1.0};
GLfloat light_pos[] = {1.0, 2.0, 3.0, 1.0};

glEnable (GL_LIGHTING);
glEnable (GL_LIGHT0);
glLightfv (GL_LIGHT0, GL_POSITION,
           light_pos);
glLightfv (GL_LIGHT0, GL_AMBIENT, ambient);
glLightfv (GL_LIGHT0, GL_DIFFUSE, diffuse);
glLightfv (GL_LIGHT0, GL_SPECULAR, specular);
```



Position and Attenuation

- Light position in homogeneous coordinates (x, y, z, w)
 - If $w = 1$, (x, y, z) is the *location*
 - If $w = 0$, (x, y, z) is the *direction* of a *directional light*
- *Attenuation:*
 - Intensity drops with distance from light source
 - `glLight*` attenuation attributes:
 - `GL_CONSTANT_ATTENUATION`
 - `GL_LINEAR_ATTENUATION`
 - `GL_QUADRATIC_ATTENUATION`

Spotlights

- Spotlights defined like point light with `glLight*` attributes:
 - `GL_SPOT_DIRECTION`: spot light direction
 - `GL_SPOT_CUTOFF`: semi-angle ϕ of spot light cone
 - `GL_SPOT_EXPONENT`: concentration of light in cone
 - highest intensity in centre
 - attenuated towards edge proportional to $(\cos(\phi))^\tau$



Global Ambient Light

➤ *Global ambient light:*

- Single special light source of global ambient light

```
GLfloat global_ambient[] = {  
    0.2, 0.2, 0.2, 1.0  
};  
  
glLightModelfv (GL_LIGHT_MODEL_AMBIENT,  
               global_ambient);
```

Material Properties

- OpenGL material properties part of the pipeline state

```
GLfloat ambient[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat diffuse[] = { 1.0, 0.8, 0.0, 1.0 };
GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat shine = 100.0;
glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
glMaterialfv(GL_FRONT, GL_SHININESS, &shine);
```

- Shininess is specular exponent

Surface Normal Vectors

- For lighting computations OpenGL requires *normal vectors* of polygonal primitives
 - Orthogonal to surface pointing outwards
 - Used to compute reflection angle
- Set normal vector for vertices with `glNormal* (...);`
 - Sets current normal as state for each subsequent vertex
- Normals should be unit vectors
 - Vectors automatically normalised, if enabled explicitly:

```
glEnable (GL_NORMALIZE) ;
```



Summary

- What is the Phong illumination model?
- What are ambient, directional, point and spot light sources? How is the light intensity arriving from one of these light sources at a surface point computed?
- What is ambient, diffuse and specular light? How is the amount of specular light for each light type computed?