

## Radiometry

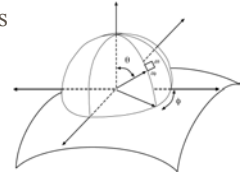
- Relates light-source energy, reflected light and light registered by sensor.
- Core Questions:
  - how “bright” will surfaces be?
  - what is “brightness”?
    - measuring light
    - interactions between light and surfaces



Slides by Edward Angel © 2002

## Radiometry

- around any point is a hemisphere of directions
- Simplest problems can be dealt with by reasoning about this hemisphere
- Core idea - think about light arriving at a surface



Slides by Edward Angel © 2002

- **Solid Angle**  $\omega$ : area of the projection of surface patch at point  $x$ , onto the unit sphere centered at  $x$ . Unit: steradians.

$$d\omega = \sin(\theta) d\theta d\phi$$

- **Radiance**: power (energy per unit time) traveling at some point in a specified direction, per unit area perpendicular to direction of travel, per unit solid angle.
  - Unit for measuring the distribution of light in space ( $W \times m^{-2} \times sr^{-1}$ )



Slides by Edward Angel © 2002

## Solid Angle

- Defined by analogy with angle (in radians)
  - solid angle subtended by a patch is the area covered by the patch on a unit sphere
- The solid angle subtended by an infinitesimal patch area  $dA$  is given by
 
$$d\omega = \frac{dA \cos \theta}{r^2}$$
- Another useful expression:  $d\omega = \sin \theta (d\theta) (d\phi)$
- $\theta$ -polar angle,  $\phi$  - azimuth



Slides by Edward Angel © 2002

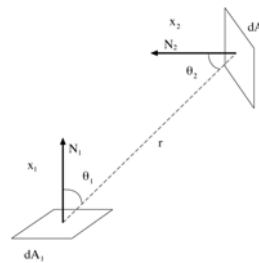
## Radiance

- Measure the “amount of light” at a point, in a direction
  - Function of position and direction
- Definition: *Radiant power per unit foreshortened area per unit solid angle*
  - *Foreshortened* implies perpendicular to the direction of travel
- Units: watts per square meter per steradian ( $Wm^{-2}sr^{-1}$ )
- Usually written as:  $L(\underline{x}, \theta, \phi)$
- Crucial property: In a vacuum, radiance leaving  $p$  in the direction of  $q$  is the same as radiance arriving at  $q$  from  $p$ 
  - hence the units



Slides by Edward Angel © 2002

## Radiance is constant along straight lines



- Assume vacuum
- $p_1$  and  $p_2$  have a line of sight between them
- Power  $1 \Rightarrow 2$ , leaving 1:
 
$$L(\underline{x}_1, \theta, \phi) (dA_1 \cos \theta_1) \left( \frac{dA_2 \cos \theta_2}{r^2} \right)$$
- Power  $1 \Rightarrow 2$ , arriving at 2:
 
$$L(\underline{x}_2, \theta, \phi) (dA_2 \cos \theta_2) \left( \frac{dA_1 \cos \theta_1}{r^2} \right)$$



Slides by Edward Angel © 2002

## Irradiance

- How much light is arriving at a surface?
- Sensible unit is *Irradiance*
- Incident power per unit area *not foreshortened* ( $W/m^2$ )
- This is a function of incoming angle.
- A surface experiencing radiance  $L(x, \theta, \phi)$  coming in from solid angle  $d\omega$  experiences irradiance  $L(x, \theta, \phi) \cos \theta d\omega$



Slides by Edward Angel © 2002

## Irradiance

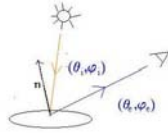
- Crucial property:
  - Total power arriving at the surface is given by adding irradiance over all incoming angles --- this is why it's a natural unit
- Total power is  $\int L(x, \theta, \phi) \cos \theta \sin \theta d\theta d\phi$
- Irradiance at an image point depends on light arriving from projected object point.
- Image irradiance is proportional to scene radiance



Slides by Edward Angel © 2002

## Surfaces and the BRDF

- Many possible effects when light strikes a surface:
  - absorbed
  - transmitted
  - reflected
  - scattered
- Assume that
  - All effects are local
  - surfaces don't fluoresce
  - surfaces don't emit light (i.e. are cool)
  - all the light leaving a point is due to that arriving at that point



Slides by Edward Angel © 2002

## The BRDF

- Given assumptions, we can model effects at a surface with a record of outgoing vs incoming illumination
  - the Bidirectional Reflectance Distribution Function (BRDF)
- Definition  $\rho$ :
  - the ratio of the radiance in the outgoing direction to the incident irradiance

$$\rho_{bd}(x, \theta_o, \phi_o, \theta_i, \phi_i) = \frac{L_o(x, \theta_o, \phi_o)}{L_i(x, \theta_i, \phi_i) \cos \theta_i d\omega}$$



Slides by Edward Angel © 2002

## The BRDF

- Units: inverse steradians ( $sr^{-1}$ )
- Symmetric in incoming and outgoing directions (Helmholtz reciprocity principle)
- Radiance leaving in a particular direction:
  - add contributions from every incoming direction

$$\int_{\Omega} \rho_{bd}(x, \theta_o, \phi_o, \theta_i, \phi_i) L_i(x, \theta_i, \phi_i) \cos \theta_i d\omega_i$$



Slides by Edward Angel © 2002

## Suppressing Angles - Radiosity

- In many situations, we do not really need angle coordinates
  - e.g. cotton cloth, where the reflected light is not dependent on angle
- Appropriate radiometric unit is radiosity
  - total power leaving a point on the surface, per unit area on the surface ( $Wm^{-2}$ )
- Radiosity from radiance?
  - sum radiance leaving surface over all exit directions

$$B(x) = \int_{\Omega} L_o(x, \theta, \phi) \cos \theta d\omega$$



Slides by Edward Angel © 2002

## Radiosity

- Important relationship:

- radiosity of a surface whose radiance is independent of angle (e.g. that cotton cloth)

$$\begin{aligned} B(\underline{x}) &= \int_{\Omega} L_o(\underline{x}, \vartheta, \varphi) \cos \vartheta d\omega \\ &= L_o(\underline{x}) \int_{\Omega} \cos \vartheta d\omega \\ &= L_o(\underline{x}) \int_0^{\pi/2} \int_0^{2\pi} \cos \vartheta \sin \vartheta d\varphi d\vartheta \\ &= \pi L_o(\underline{x}) \end{aligned}$$



Slides by Edward Angel © 2002

## Directional hemispheric reflectance

- BRDF is a very general notion
  - some surfaces need it (underside of a CD; tiger eye; eg oriented microstructure)
  - very hard to measure and often an unstable measurement
  - for many surfaces, light leaving the surface is largely independent of exit angle (surface roughness is one source of this property)



Slides by Edward Angel © 2002

## Directional Hemispheric Reflectance

- Directional hemispheric reflectance:
  - the fraction of the incident irradiance in a given direction that is reflected by the surface (whatever the direction of reflection)
  - unitless, range is 0-1

$$\begin{aligned} \rho_{dh}(\vartheta_i, \varphi_i) &= \frac{\int_{\Omega} L_o(\underline{x}, \vartheta_o, \varphi_o) \cos \vartheta_o d\omega_o}{L_i(\underline{x}, \vartheta_i, \varphi_i) \cos \vartheta_i} \\ &= \int_{\Omega} \rho_{brdf}(\underline{x}, \vartheta_o, \varphi_o, \vartheta_i, \varphi_i) \cos \vartheta_o d\omega_o \end{aligned}$$



Slides by Edward Angel © 2002

## Lambertian surfaces and albedo

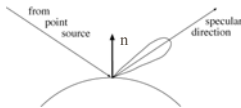
- For some surfaces, the DHR is independent of direction
  - cotton cloth, carpets, matte paper, matte paints, etc.
  - radiance leaving the surface is independent of angle
  - **Lambertian surfaces or ideal diffuse surfaces**
  - Use radiosity as a unit to describe light leaving the surface
  - DHR is often called diffuse reflectance, or **albedo** for a Lambertian surface, BRDF is independent of angle, too.
- Useful fact:  $\rho_{brdf} = \frac{\rho_d}{\pi}$
- Appears equally bright from any direction



Slides by Edward Angel © 2002

## Specular surfaces

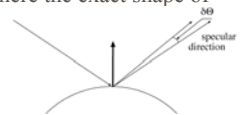
- Another important class of surfaces is specular, or mirror-like.
  - radiation arriving along a direction leaves along the specular direction
  - reflect about normal
  - some fraction is absorbed, some reflected
  - on real surfaces, energy usually goes into a lobe of directions
  - can write a BRDF, but requires the use of funny functions



Slides by Edward Angel © 2002

## Phong's model

- There are very few cases where the exact shape of the specular lobe matters.
- Typically for lobe shape:
  - very, very small --- mirror
  - small -- blurry mirror
  - bigger -- see only light sources as "specularities"
  - very big -- faint specularities
- Phong's model
  - reflected energy falls off with  $\cos^n(\delta\vartheta)$



Slides by Edward Angel © 2002

## Lambertian + specular model

- Widespread model
    - all surfaces are Lambertian plus specular component
  - Advantages
    - easy to manipulate
    - very often quite close true
  - Disadvantages
    - some surfaces are not
      - e.g. underside of CD's, feathers of many birds, blue spots on many marine crustaceans and fish, most rough surfaces, oil films (skin!), wet surfaces
    - Generally, very little advantage in modelling behaviour of light at a surface in more detail -- it is quite difficult to understand behaviour of L+S surfaces
- distant bright sky).

Slides by Edward Angel © 2002

## Normals

- In OpenGL the normal vector is part of the state
- Set by `glNormal*()`
  - `glNormal3f(x, y, z);`
  - `glNormal3fv(p);`
- Usually we want to set the normal to have unit length so cosine calculations are correct
  - Length can be affected by transformations
  - Note the scale does not preserved length
  - `glEnable(GL_NORMALIZE)` allows for autonormalization at a performance penalty

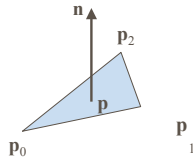
Slides by Edward Angel © 2002

## Normal for Triangle

$$\text{plane } \mathbf{n} \cdot (\mathbf{p} - \mathbf{p}_0) = 0$$

$$\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_0)$$

$$\text{normalize } \mathbf{n} \leftarrow \mathbf{n} / |\mathbf{n}|$$



Note that right-hand rule determines outward face

Slides by Edward Angel © 2002

## Enabling Shading

- Shading calculations are enabled by
  - `glEnable(GL_LIGHTING)`
  - Once lighting is enabled, `glColor()` ignored
- Must enable each light source individually
  - `glEnable(GL_LIGHTi) i=0,1,...`
- Can choose light model parameters
  - `glLightModeli(parameter, GL_TRUE)`
    - `GL_LIGHT_MODEL_LOCAL_VIEWER` do not use simplifying distant viewer assumption in calculation
    - `GL_LIGHT_MODEL_TWO_SIDED` shades both sides of polygons independently

Slides by Edward Angel © 2002

## Defining a Point Light Source

- For each light source, we can set an RGB for the diffuse, specular, and ambient parts, and the position

```
GL float diffuse0[]={1.0, 0.0, 0.0, 1.0};
GL float ambient0[]={1.0, 0.0, 0.0, 1.0};
GL float specular0[]={1.0, 0.0, 0.0, 1.0};
GLfloat light0_pos[]={1.0, 2.0, 3.0, 1.0};
```

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightv(GL_LIGHT0, GL_AMBIENT, ambient0);
glLightv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
glLightv(GL_LIGHT0, GL_SPECULAR, specular0);
```

Slides by Edward Angel © 2002

## Distance and Direction

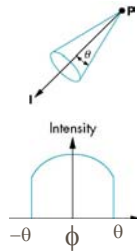
- The source colors are specified in RGBA
- The position is given in homogeneous coordinates
  - If  $w=1.0$ , we are specifying a finite location
  - If  $w=0.0$ , we are specifying a parallel source with the given direction vector
- The coefficients in the distance terms are by default  $a=1.0$  (constant terms),  $b=c=0.0$  (linear and quadratic terms). Change by

```
a= 0.80;
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
```

Slides by Edward Angel © 2002

## Spotlights

- Use `glLightfv` to set
  - Direction  
`GL_SPOT_DIRECTION`
  - Cutoff `GL_SPOT_CUTOFF`
  - Attenuation  
`GL_SPOT_EXPONENT`
    - Proportional to  $\cos^{\alpha}\phi$



Slides by Edward Angel © 2002

## Global Ambient Light

- Ambient light depends on color of light sources
  - A red light in a white room will cause a red ambient term that disappears when the light is turned off
- OpenGL allows a global ambient term that is often helpful
  - `glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient)`



Slides by Edward Angel © 2002

## Moving Light Sources

- 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



Slides by Edward Angel © 2002

## Material Properties

- Material properties are also part of the OpenGL state and match the terms in the Phong model
- Set by `glMaterialv()`

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



Slides by Edward Angel © 2002

## Front and Back Faces

- The default is shade only front faces which works correct for convex objects
- If we set two sided lighting, OpenGL will shaded both sides of a surface
- Each side can have its own properties which are set by using `GL_FRONT`, `GL_BACK`, or `GL_FRONT_AND_BACK` in `glMaterialf`



Slides by Edward Angel © 2002

## Emissive Term

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

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



Slides by Edward Angel © 2002

## Transparency

- Material properties are specified as RGBA values
- The A value can be used to make the surface translucent
- The default is that all surfaces are opaque regardless of A
- Later we will enable blending and use this feature



Slides by Edward Angel © 2002

## Efficiency

- Because material properties are part of the state, if we change materials for many surfaces, we can affect performance
- We can make the code cleaner by defining a material structure and setting all materials during initialization

```
typedef struct materialStruct {  
    GLfloat ambient[4];  
    GLfloat diffuse[4];  
    GLfloat specular[4];  
    GLfloat shininess;  
} MaterialStruct;
```



- We can then select a material by a pointer

Slides by Edward Angel © 2002

## Polygonal Shading

- Shading calculations are done for each vertex
  - Vertex colors become vertex shades
- By default, vertex colors are interpolated across the polygon
  - `glShadeModel(GL_SMOOTH);`
- If we use `glShadeModel(GL_FLAT);` the color at the first vertex will determine the color of the whole polygon



Slides by Edward Angel © 2002

## Polygon Normals

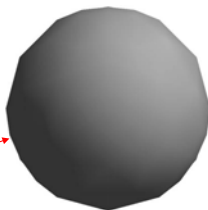
- Polygons have a single normal
  - Shades at the vertices as computed by the Phong model can be almost same
  - Identical for a distant viewer (default) or if there is no specular component
- Consider model of sphere
- Want different normals at each vertex even though this concept is not quite correct mathematically



Slides by Edward Angel © 2002

## Smooth Shading

- We can set a new normal at each vertex
- Easy for sphere model
  - If centered at origin  $\mathbf{n} = \mathbf{p}$
- Now smooth shading works
- Note *silhouette edge*

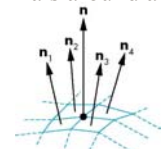


Slides by Edward Angel © 2002

## Mesh Shading

- The previous example is not general because we knew the normal at each vertex analytically
- For polygonal models, Gouraud proposed we use the average of normals around a mesh vertex

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



Slides by Edward Angel © 2002



## Gouraud and Phong Shading

- Gouraud Shading
  - Find average normal at each vertex (vertex normals)
  - Apply Phong model at each vertex
  - Interpolate vertex shades across each polygon
- Phong shading
  - Find vertex normals
  - Interpolate vertex normals across edges
  - Find shades along edges
  - Interpolate edge shades across polygons



Slides by Edward Angel © 2002



## Comparison

- If the polygon mesh approximates surfaces with a high curvatures, Phong shading may look smooth while Gouraud shading may show edges
- Phong shading requires much more work than Gouraud shading
  - Usually not available in real time systems
- Both need data structures to represent meshes so we can obtain vertex normals



Slides by Edward Angel © 2002