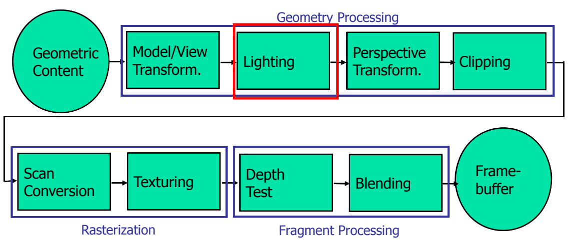
Chapter 6

Lighting

Factors

- Light sources
  - Location, type & color
- Surface materials
  - How surfaces reflect light
- Transport of light
  - How light moves in a scene
- Viewer position

Rendering Pipeline



Illumination Models/Algorithms

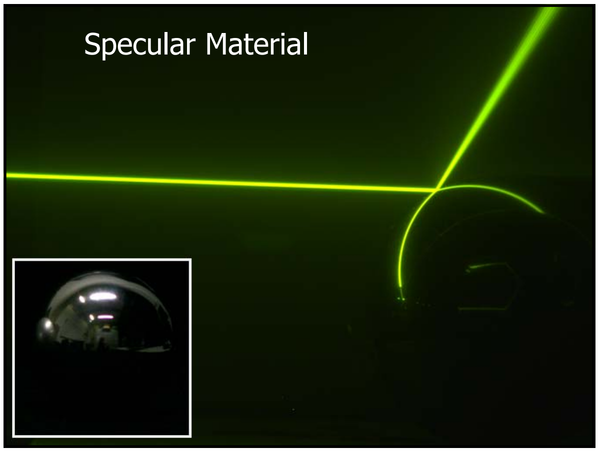
- Local illumination - Fast
  - Ignore real physics, approximate the look
  - Interaction of each object with light
    - Compute on surface (light to viewer)
- Global illumination – Slow
  - Physically based
  - Interactions between objects

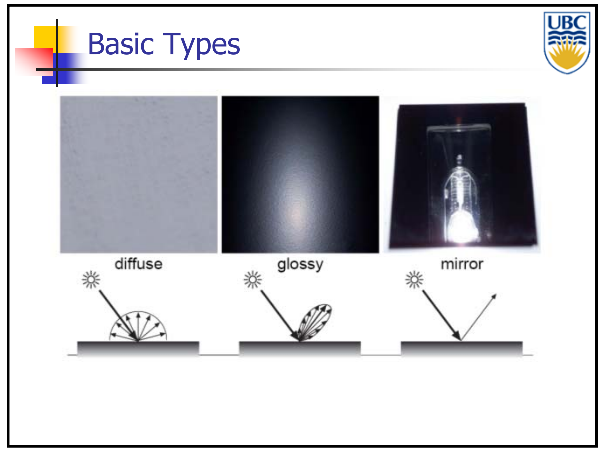
Lighting/Shading

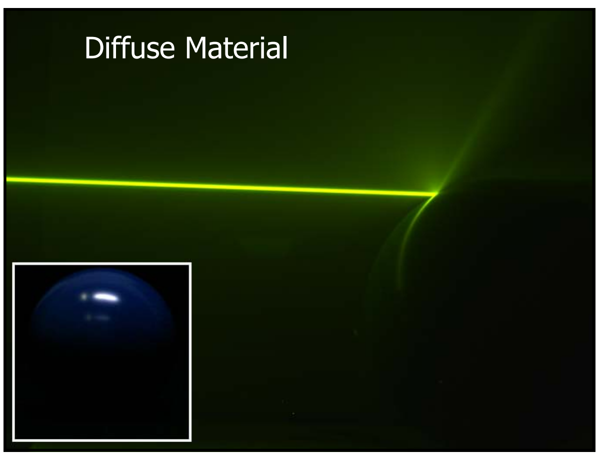
- Goal
  - Model the interaction of light with surfaces to render realistic images
  - Generate per (pixel/vertex) color

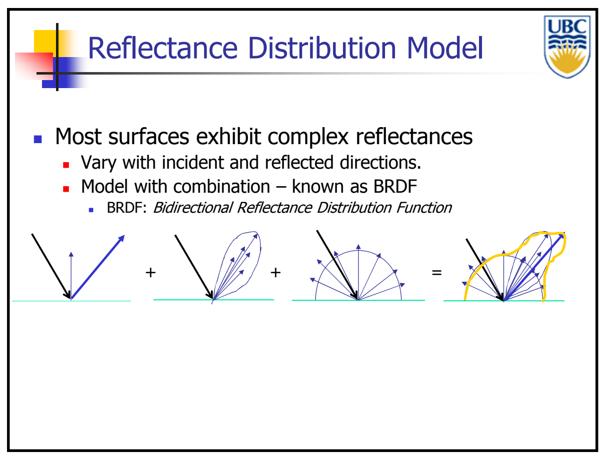
Materials

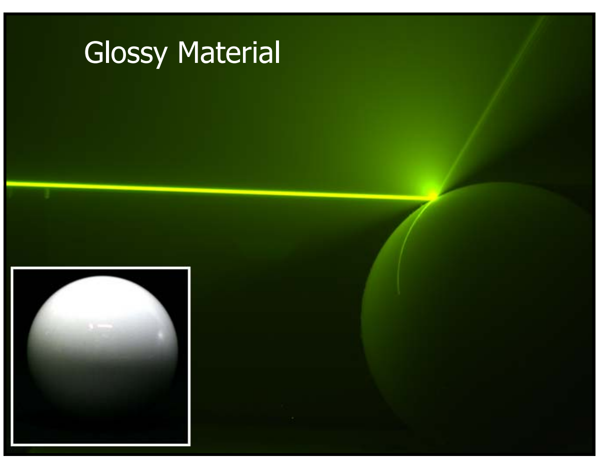
- Surface reflectance:
  - Illuminate surface point with a ray of light from different directions
  - How much light is reflected in each direction?

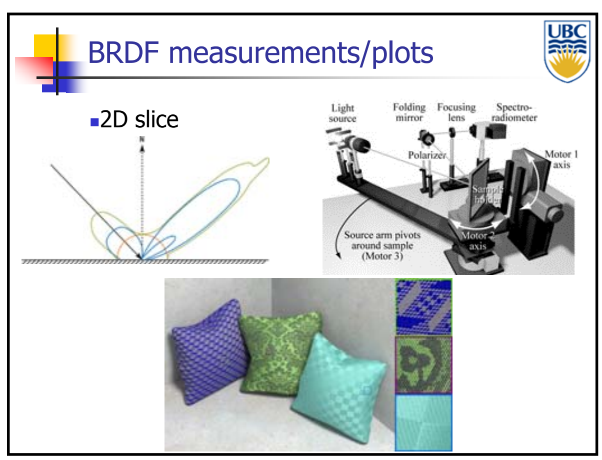

Specular Material


Basic Types
diffuse
glossy
mirror


Diffuse Material


Reflectance Distribution Model
Most surfaces exhibit complex reflectances
Vary with incident and reflected directions.
Model with combination – known as BRDF
BRDF: *Bidirectional Reflectance Distribution Function*
+
+
=


Glossy Material

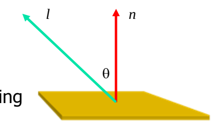

BRDF measurements/plots
2D slice
Light source
Folding mirror
Focusing lens
Spectro-radiometer
Polarizer
Motor 1 axis
Source arm pivots around sample (Motor 3)

Practical Considerations

- In practice, often simplify (computational efficiency)
- Derive specific formulas that describe basic reflectance behaviors
  - diffuse, glossy, specular
  - OpenGL choice

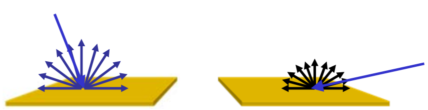
Computing Diffuse Reflection

- Depends on angle of incidence: angle between surface normal and incoming light
  - $I_{diffuse} = k_d I_{light} \cos\theta$
- In practice use vector arithmetic
  - $I_{diffuse} = k_d I_{light} (\mathbf{n} \bullet \mathbf{l})$
- Always normalize vectors used in lighting
  - **n, l** should be unit vectors
- Scalar (B/W intensity) or 3-tuple or 4-tuple (color)
  - k_d: diffuse coefficient, surface color
  - I_{light}: incoming light intensity
  - $I_{diffuse}$: outgoing light intensity (for diffuse reflection)



Physics of Diffuse Reflection

- Ideal diffuse reflection
  - Very rough surface at the microscopic level
    - Real-world example: chalk
    - Microscopic variations mean incoming ray of light equally likely to be reflected in any direction over the hemisphere
  - Reflected intensity only depends on light direction!

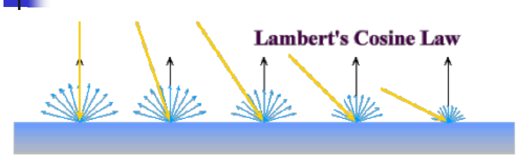


Diffuse Lighting Examples

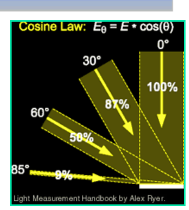
- Lambertian sphere from several lighting angles:

- need only consider angles from 0° to 90°
  - *Why?*

Lambert's "Law"

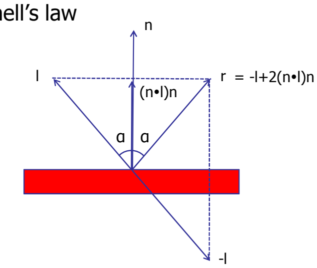


Intuitively: cross-sectional area of the "beam" intersecting an element of surface area is smaller for greater angles with the normal.



Physics of Specular Reflection

- Geometry of specular (perfect mirror) reflection
  - Snell's law



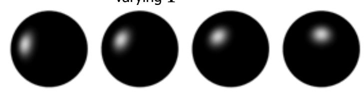
Empirical Approximation

- Snell's law = perfect mirror-like surfaces
  - But ..
    - few surfaces exhibit perfect specularity
    - Gaze and reflection directions never EXACTLY coincide
- Expect **most** reflected light to travel in direction predicted by Snell's Law
- But some light may be reflected in a direction slightly off the ideal reflected ray
- As angle from ideal reflected ray increases, we expect less light to be reflected

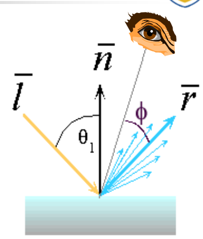
Phong Examples

$$\mathbf{I}_{\text{specular}} = \mathbf{k}_s \mathbf{I}_{\text{light}} (\cos\phi)^{n_s}$$

varying l

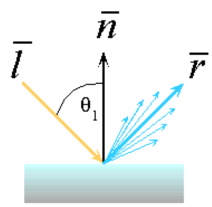


varying n_s



Empirical Approximation

- Angular falloff



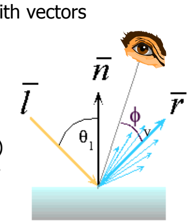
- How to model this falloff?

Calculating Phong Lighting

- compute cosine term of Phong lighting with vectors

$$\mathbf{I}_{\text{specular}} = \mathbf{k}_s \mathbf{I}_{\text{light}} (\mathbf{v} \bullet \mathbf{r})^{n_s}$$

- v: unit vector towards viewer/eye
- r: ideal reflectance direction (unit vector)
- k_s: specular component = highlight color
- I_{light}: incoming light intensity



Phong Lighting

- Most common lighting model in computer graphics
  - (Phong Bui-Tuong, 1975)

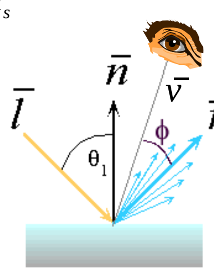
$$\mathbf{I}_{\text{specular}} = \mathbf{k}_s \mathbf{I}_{\text{light}} (\cos\phi)^{n_s}$$

ϕ: angle between r and view direction v

n_s : purely empirical constant, varies rate of falloff

k_s: specular coefficient, highlight color

no physical basis, works ok in practice



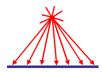
Materials (last bit)

- Light is **linear**
  - If multiple rays illuminate the surface point the result is just the sum of the individual reflections for each ray

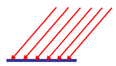
$$\sum_p I_p (k_d (n \cdot l_p) + k_s (r_p \cdot v)^n)$$

Light Sources

- Point source
  - light originates at a point
  - Rays hit planar surface at different angles

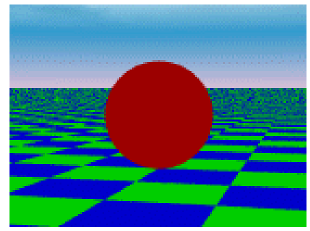


- Parallel source
  - light rays are parallel
  - Rays hit a planar surface at identical angles
  - Can model as point source at infinity
  - *Directional light*



Ambient Light Sources

- Scene lit only with an ambient light source



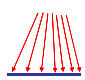
Light Position Not Important

Viewer Position Not Important

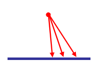
Surface Angle Not Important

Light Sources

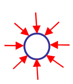
- Area source
  - Light originates at finite area in space.
  - In-between point and parallel sources



- Spotlights
  - position, direction, angle



- Ambient light (environment light)
  - Hack for replacing true global illumination
    - (light bouncing off from other objects)

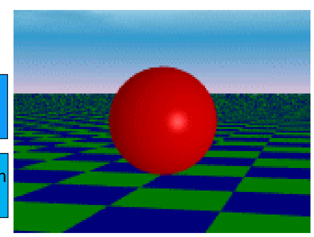


Directional Light Sources

- Scene lit with directional and ambient light

Surface Angle Important

Viewer Position Important



Light Position Not Important

Ambient Light

- Non-directional light – environment light
- Object illuminated with same light everywhere
  - Looks like silhouette
- Illumination equation $I = I_a k_a$
  - I_a - ambient light intensity
  - k_a- fraction of this light reflected from surface

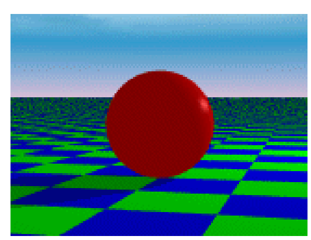
Point Light Sources

- Scene lit with ambient and point light source

Light Position Important

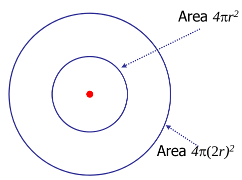
Viewer Position Important

Surface Angle Important



Light Source Falloff

- Quadratic falloff (point- and spot lights)
  - Brightness of objects depends on power per unit area that hits the object
  - The power per unit area for a point or spot light decreases quadratically with distance



Light

- Light has color
- Interacts with object color (r,g,b)

$$I = I_a k_a$$
$$I_a = (I_{ar}, I_{ag}, I_{ab})$$
$$k_a = (k_{ar}, k_{ag}, k_{ab})$$
$$I = (I_r, I_g, I_b) = (I_{ar}k_{ar}, I_{ag}k_{ag}, I_{ab}k_{ab})$$

- Blue light on white surface?
- Blue light on red surface?

Light Source Falloff

- Non-quadratic falloff
  - Many systems allow for other falloffs
  - Allows for faking effect of area light sources
  - OpenGL / graphics hardware
    - I_o: intensity of light source
    - $\mathbf{x}$: object point
    - r: distance of light from $\mathbf{x}$

$$I_{in}(\mathbf{x}) = \frac{1}{ar^2 + br + c} \cdot I_0$$

Lighting in OpenGL

- Light source: amount of RGB light emitted
  - value = percentage of full intensity, e.g., (1.0,0.5,0.5)
  - every light source emits ambient, diffuse, and specular light
- Materials: amount of RGB light reflected
  - value represents percentage reflected e.g., (0.0,1.0,0.5)
- Interaction: multiply components
  - Red light (1,0,0) x green surface (0,1,0) = black (0,0,0)

Illumination Equation

- For multiple light sources:

$$I = I_a k_a + \sum_p \frac{I_p}{A(d_p)}(k_d (n \cdot l_p) + k_s (r_p \cdot v)^n)$$

- d_p - distance between surface and light source + distance between surface and viewer, A – attenuation function

In OpenGL

- k_a, k_d, k_s - surface color (RGB)
- Modify by glMaterialfv(GL_FRONT_AND_BACK , *pname*, RGB[])
- *pname* - GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR
- Light source properties (also RGB) glLightfv(GL_LIGHTi,*pname*,light[])

Lighting in OpenGL

```
glLightfv(GL_LIGHT0, GL_AMBIENT, amb_light_rgba );
glLightfv(GL_LIGHT0, GL_DIFFUSE, dif_light_rgba );
glLightfv(GL_LIGHT0, GL_SPECULAR, spec_light_rgba );
glLightfv(GL_LIGHT0, GL_POSITION, position);
glEnable(GL_LIGHT0);

glMaterialfv( GL_FRONT, GL_AMBIENT, ambient_rgba );
glMaterialfv( GL_FRONT, GL_DIFFUSE, diffuse_rgba );
glMaterialfv( GL_FRONT, GL_SPECULAR, specular_rgba );
glMaterialfv( GL_FRONT, GL_SHININESS, n );
```

Light Sources - OpenGL

- Specify parameters
  glLightfv(GL_LIGHTi,GL_POSITION,light[])
  i – between 0 & 8 (or more)
- Directional $[x \quad y \quad z \quad 0]$
- Point source $[x \quad y \quad z \quad 1]$
- Spotlight has extra parameters:
  - GL_SPOT_DIRECTION, GL_SPOT_EXPONENT, GL_SPOT_CUTOFF
- Area source – too complex for projective pipeline (e.g. OpenGL)

Lighting in Rendering Pipeline

- Notes:
  - Lighting is applied to every **vertex**
    - i.e. the three vertices in a triangle
    - Per-vertex lighting
  - Will later see how the interior points of the triangle obtain their color
    - This process is called **shading**
    - Will discuss in the context of scan conversion