Why we need shading

• Suppose we build a model of a sphere using many polygons and color each the same color. We get something like

• But we want

The more realistically lit sphere has gradations in its color that give us a sense of its three-dimensionality
Shading

• Why does the image of a real sphere look like

• Light-material interactions cause each point to have a different color or shade

• Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation (normal)

We are going to develop a local lighting model by which we can shade a point independently of the other surfaces in the scene. Our goal is to add this to a fast graphics pipeline architecture.
General rendering

- The most general approach is based on physics - using principles such as conservation of energy
- A surface either emits light (e.g., light bulb) or reflects light for other illumination sources, or both
- Light interaction with materials is recursive
- The rendering equation is an integral equation describing the limit of this recursive process

http://en.wikipedia.org/wiki/Rendering_equation
Fast local shading models

- the rendering equation can’t be solved analytically
- numerical methods aren’t fast enough for real-time
- for our fast graphics rendering pipeline, we’ll use a local model where shade at a point is independent of other surfaces
- use Phong reflection model
  - shading based on local light-material interactions

Some approximations to the rendering equation include radiosity and ray tracing, but they are still not as fast as the local model in the pipeline architecture
**Local shading model**

**direct light** is the color of the light source

**reflected light** is the color of the light reflected from the object surface

For rendering, color of light source and reflected light determines the colors of pixels in the frame buffer.

Only need to consider the rays that leave the source and reach the viewer’s eye.
Global Effects

- Shadow
- Multiple reflection
- Translucent surface

[Angel and Shreiner]
Light-material interactions

at a surface, light is absorbed, reflected, or transmitted

**specular**: shiny, smooth surface. Light scattered in narrow range close to angle of reflection. E.g., mirror is perfectly specular.

**diffuse**: matte, rough surface. Light scattered in all directions.

**translucent**: allows some light to pass through object. Refraction: e.g., glass or water.
General light source

Illumination function:

\[ L(\mathbf{x}, \omega, \lambda) \]

\[ \mathbf{x} = (x, y, z) \]

\[ \mathbf{\omega} = \theta, \phi \]

integrate contributions from all sources to shade the point
Idealized light sources

- Ambient light
- Point light
- Spotlight
- distant (directional) light

luminance: \( L = \begin{bmatrix} L_r \\ L_g \\ L_b \end{bmatrix} \)

Source will be described through three component intensity or luminance
decompose into red, green, blue channels
e.g., use the red component of source to calculate red component of image
use a single scalar equations – each equation applied independently to each channel
Ambient light source

- achieve a uniform light level
- no black shadows
- ambient light intensity at each point in the scene

\[ L_a = \begin{bmatrix} L_{ar} \\ L_{ag} \\ L_{ab} \end{bmatrix} \]

use scalar \( l_a \) to denote any component of \( \text{vec} I_a \)
ambient light is the same everywhere but different surfaces will reflect it differently
Point light source

\[ L(p_0) = \begin{bmatrix} L_r(p_0) \\ L_g(p_0) \\ L_b(p_0) \end{bmatrix} \quad L(p_0) \]

illumination intensity at \( p \):

\[ l(p, p_0) = \frac{1}{|p - p_0|^2} L(p_0) \]

- use scalar \( l(\vec{p}_0) \) to denote any of three components
- points sources alone aren’t too realistic looking -- tend to be high contrast
- most real-world scenes have large light sources
- add ambient light to mitigate high contrast
Point light source

Most real-world scenes have large light sources.

Point light sources alone aren't too realistic. Add ambient light to mitigate high contrast.

- umbra is fully in shadow, penumbra is partially in shadow.
Most real-world scenes have large light sources.

Point light sources alone aren’t too realistic:
- **drop off intensity more slowly**

\[
l(p, p_0) = \frac{1}{d^2} L(p_0)
\]

\[
l(p, p_0) = \frac{1}{a + bd + cd^2} L(p_0)
\]

In practice, we also replace the \(1/d^2\) term by something that falls off more slowly.
Spotlights

Intensity

\[ \cos(\phi) \]

\[ \theta \]

\[ \phi \]

\[ p_s \]

[Angel and Shreiner]
Spotlights

\[ \cos^e(\phi) \ l(p, p_s) \]

add an exponent for greater control
final result is like point light but modified by this cone
Distant light source

characterized by direction

most shading calculations require direction from the surface point to the light source position. If the light source is very far, the direction vectors don't change.

example: sun
characterized by direction rather than position.