Basic Ways to Render

- Wire frame
- Wire, no hidden lines, but silhouettes
- Flat shaded image
- Smooth shaded image
- Image with shadows

Acknowledgements:
http://graphics.cs.ucdavis.edu/GraphicsNotes
and Chris Hoffman

Wire Frame

No Hidden Lines

Hidden Lines Eliminated in OpenGL

- A trick you can use for polyhedra
  1. Set polygon mode to GL_FRONT_AND_BACK, GL_LINE
  2. Render polygons once in black (or whatever line color you want)
  3. Set polygon mode to GL_FRONT, GL_FILL
  4. Render polygons again in background color

- Note: depth resolution conflicts may arise. If so, try shrinking the object in step 4

Flat Shading
Smooth Shading

Specular Highlights

Advanced Rendering Techniques

A Simple Shading Model

- Ambient and point light sources
  - No extended light sources…
- Diffuse illumination
- Specular illumination
- Smooth shading (Phong)

Ambient Light

Uniform intensity everywhere

Diffuse Light (Lambert)

- A fraction of light is radiated in every direction
- Intensity varies with cosine of the angle with normal
Ideal Diffuse Reflection
- An ideal diffuse reflector, at the microscopic level, is a very rough surface (real-world example: chalk)
- Because of these microscopic variations, an incoming ray of light is equally likely to be reflected in any direction over the hemisphere:

What does the reflected intensity depend on?

Lambert’s Cosine Law
- Ideal diffuse surfaces reflect according to Lambert’s cosine law:
  The energy reflected by a small portion of a surface from a light source in a given direction is proportional to the cosine of the angle between that direction and the surface normal
- The reflected intensity is independent of the viewing direction, but is dependent on how the surface is oriented with respect to the light source.

Computing Diffuse Reflection
1. Determine the angle of incidence $\theta$
2. Compute $I_{\text{diff}} = I_{\text{Light}} \rho_{\text{diff}} \cos(\theta)$
   - In practice we use vector arithmetic:
     $I_{\text{diff}} = I_{\text{Light}} \rho_{\text{diff}} (N \cdot L)$

Shading So Far
- Intensity is $I = I_{\text{diff}} + I_{\text{spec}} \rho_{\text{spec}} \cos(\theta)$
- This is computed for each color component – recall $(k_R, k_G, k_B)$ is perceived roughly as $(R, G, B)$ with different intensity.
- Missing: Highlights

Specular Reflection
- Shiny surfaces reflect predominantly in a particular direction, creating highlights
  - Polished metal
  - Glossy car finish
- Where the highlights appear depends on the viewer’s position

The Physics of Reflection
- At the microscopic level a speculally reflecting surface is very smooth
- Thus rays of light are likely to bounce off the micro-geometry as in a mirror
- The smoother the surface, the more it behaves like a perfect mirror
Snell’s Law of Reflection

- The incoming ray and reflected ray lie in a plane with the surface normal.
- The angle that the reflected ray forms with the surface normal equals the angle formed by the incoming ray and the surface normal:

\[ \theta_i = \theta_r \]

Imperfect Specular Reflectance

- Snell’s law applies to perfect mirror surfaces, but few surfaces exhibit perfect specularity.
- How can we capture the “softer” reflections of surface that are glossy rather than mirror-like – without modeling microgeometry?

Empirical Approximation

- We expect most reflected light to travel in direction predicted by Snell’s Law.
- But because of microscopic surface variations, some light may be reflected in a direction slightly off the ideal reflected ray.
- The larger the deviation from the ideal reflected ray, the less light is reflected.

Phong Lighting

- The most common lighting model was suggested by Phong:

\[ I_{spec} = \rho_{spec} I_{Light} \cos \phi^{n_{shiny}} \]

- The \( n_{shiny} \) term is an empirical constant to model the rate of falloff.
- The model has no physical basis, but it sort of works.

The Specular Exponent

- This diagram shows how the Phong reflectance term drops off with divergence of the viewing angle from the ideal reflected ray.
Specular Intensity Computation

- The \( \text{cos} \) term of Phong lighting can be computed using vector arithmetic:
  \[
  I_{\text{spec}} = \rho_{\text{spec}} I_{\text{light}} (V \cdot R)^{\text{shiny}}
  \]
- \( V \) is the unit vector towards the viewer
  - Common simplification: \( V \) is constant
    - Calculation of \( R \):
      \[
      R = (2(N \cdot L))N - L
      \]

Calculating \( R \):

\[
R = (2(N \cdot L))N - L
\]

Final Phong Model

- Intensity computed as
  \[
  I = I_a \rho_a + I_d \rho_d \cos(\theta) + I_s \rho_s (\cos(\varphi))^s
  \]
- where
  \[
  R = (2(N \cdot L))N - L
  \]

How to Apply the Model

- At every visible pixel, compute the surface normal and do the shading
  - Good, but takes too long
  - Cannot make flat adjacent surfaces make curved in appearance
  - So, consider “averaged” normals at vertices

Gouraud Shading

- Normals averaged at vertices, colors computed using a shading model, say Phong
- Colors linearly interpolated along edges, then again along scan lines.

Phong Shading

- Instead of interpolating intensity, interpolate normals
OpenGL

- Implements Gouraud shading
  - To do Phong, you need to do pixel painting
- Shininess of specular light set as a material property of the objects rendered

Material Properties?

- Light has color, but objects reflect differently, so we need to express material properties as well.
- OpenGL approach: Assign properties to objects that modify illumination

A Simple Model

- Consider \((R_M, G_M, B_M)\) to be spectral reflectivity coefficients. Color of facet is then
  \[
  I(R_I, G_I, B_I) \cdot (R_M, G_M, B_M)
  \]
  where \(I\) is the intensity computed using the illumination model

OpenGL Model

- Refine by considering each intensity component separately: ambient, diffuse, specular, and shininess.
- 4th coordinate, the A in \((R, G, B, A)\):
  - Called \(\alpha\), it determines how to mix colors at the pixel level
  - Common use: \((R, G, B, A) = (R_A + R_d (1-A), G_A + G_d (1-A), B_A + B_d (1-A), A_A + A_d (1-A))\)

Materials in OpenGL

- Materials Demo
- Identify the following:
  - black plastic
  - brass
  - bronze
  - chrome
  - copper
  - gold
  - pewter
  - polished silver
  - silver
  - slate
  (but make allowance for the projector)
Materials in OpenGL

- Materials Demo
- Identify the following:
  - 2 - black plastic
  - 3 - brass
  - 1 - bronze
  - 8 - chrome
  - 9 - copper
  - 7 - gold
  - 4 - pewter
  - 6 - polished silver
  - 5 - silver
  - 10 - slate