

RENDERING:

- Wire, Flat, Smooth
- Gouraud, Phong
- Materials in OpenGL

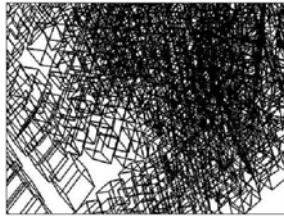
Acknowledgements: <http://graphics.cs.ucdavis.edu/GraphicsNotes> and Chris Hoffman

Basic Ways to Render

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

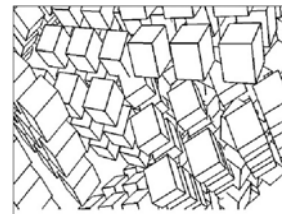
2

Wire Frame



3

No Hidden Lines



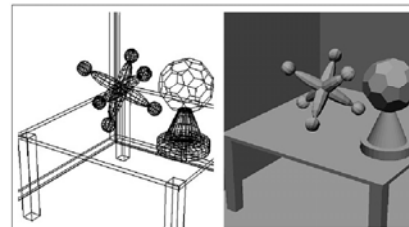
4

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

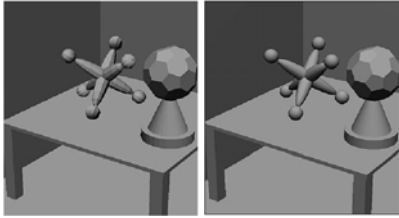
5

Flat Shading



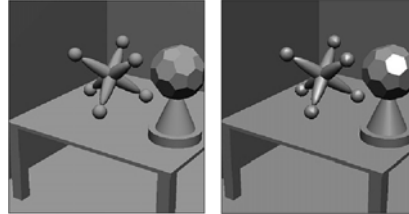
6

Smooth Shading



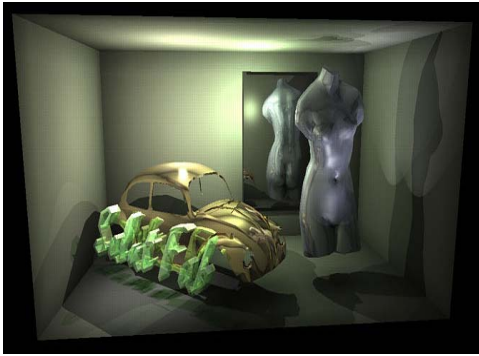
7

Specular Highlights



8

Advanced Rendering Techniques



9

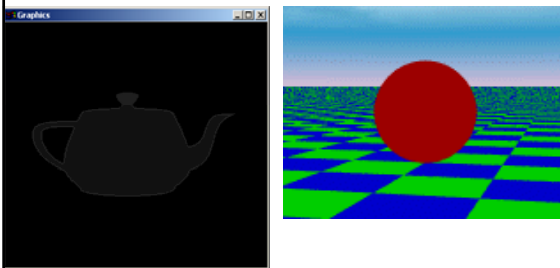
A Simple Shading Model

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

10

Ambient Light

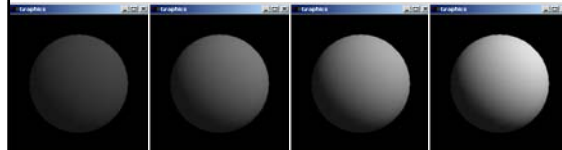
Uniform intensity everywhere



11

Diffuse Light (Lambert)

- A fraction of light is radiated in **every** direction
- Intensity varies with cosine of the angle with normal



12

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?

13

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

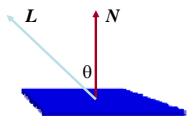
- The reflected intensity is **independent** of the viewing direction, but is **dependent** on how the surface is oriented with respect to the light source.

14

Computing Diffuse Reflection

1. Determine the angle of incidence θ
2. Compute $I_{diff} = I_{Light} \rho_{diff} \cos(\theta)$
 - In practice we use vector arithmetic:

$$I_{diff} = I_{Light} \rho_{diff} (N \cdot L)$$



15

Shading So Far

- Intensity is $I = I_a \rho_a + I_d \rho_d \cos(\theta)$
- This is computed for each color component – recall (kR, kG, kB) is perceived roughly as (R, G, B) with different intensity.

- **Missing: Highlights**

16

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

17

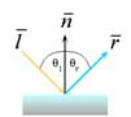
The Physics of Reflection

- At the microscopic level a specularly 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

18

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:



$\theta_i = \theta_r$

19

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?

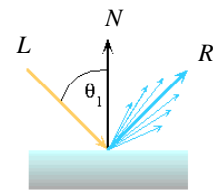
20

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

21

- Intuition of the angular falloff:



- How might we model this falloff?

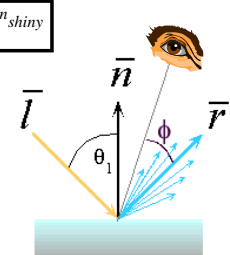
22

Phong Lighting

- The most common lighting model was suggested by Phong

$$I_{spec} = \rho_{spec} I_{Light} (\cos \phi)^{n_{shiny}}$$

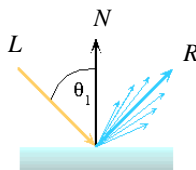
- The n_{shiny} term is an empirical constant to model the rate of falloff
- The model has no physical basis, but it sort of works

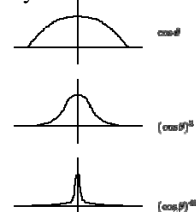


23

The Specular Exponent

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





24

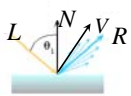
Specular Intensity Computation

- The **cos** term of Phong lighting can be computed using vector arithmetic:

$$I_{spec} = \rho_{spec} I_{Light} (V \cdot R)^{n_{shiny}}$$

- V is the unit vector towards the viewer
 - Common simplification: V is constant R is the ideal reflectance direction (means what?)
- Calculation of R :

$$R = (2(N \cdot L))N - L$$



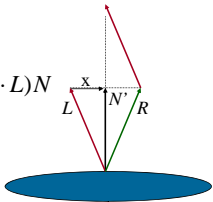
25

Calculating R:

because

$$R = (2(N \cdot L))N - L$$

$$R + L = (2(N \cdot L))N$$

$$L + x = N' = (N \cdot L)N$$


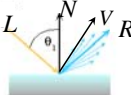
26

Final Phong Model

- Intensity computed as

$$I = I_a \rho_a + I_d \rho_d \cos(\theta) + I_s \rho_s (\cos(\varphi))^n$$

- where $\cos(\varphi) = I_a \rho_a + I_d \rho_d (L \cdot N) + I_s \rho_s (V \cdot R)^n$

$$R = (2(N \cdot L))N - L$$


27

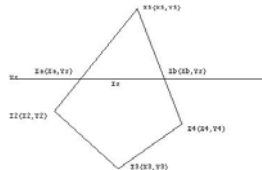
How to Apply the Model

- 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

28

Gouraud Shading

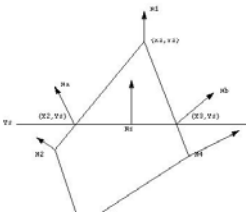
- Normals averaged at vertices, colors computed using a shading model, say Phong
- Colors linearly interpolated along edges, then again along scan lines.



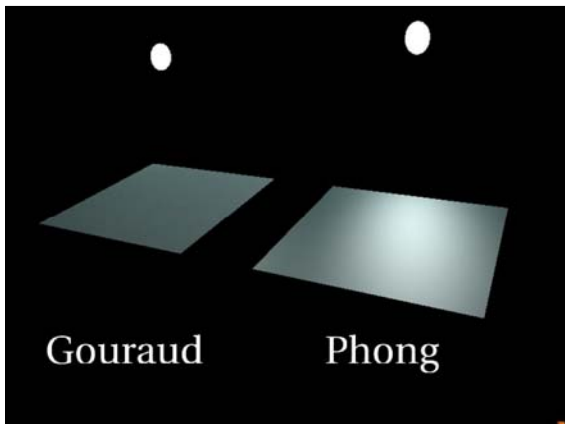
29

Phong Shading

- Instead of interpolating intensity, interpolate normals



30



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

32

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

33

A Simple Model

- Consider (R_M, G_M, B_M) to be spectral reflectivity coefficients. Color of facet is then

$$I(R_L, G_L, B_L) \cdot (R_M, G_M, B_M)$$

where I is the intensity computed using the illumination model

34

OpenGL Model

- Refine by considering each intensity component separately: ambient, diffuse, specular, and shininess.
- 4th coordinate, the A in (R,G,B,A):
 - Called α , it determines how to mix colors at the pixel level
 - Common use: $(R, G, B, A) =$
 $(R_s A_s + R_d (1 - A_s), G_s A_s + G_d (1 - A_s),$
 $B_s A_s + B_d (1 - A_s), A_s A_s + A_d (1 - A_s))$

35

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)

36

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

37