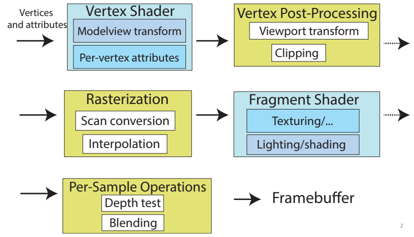
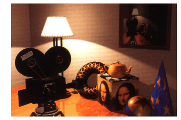
THE RENDERING PIPELINE



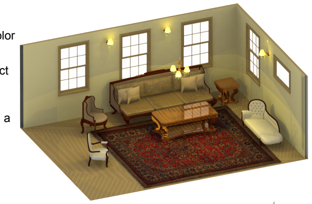
LIGHTING/SHADING

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



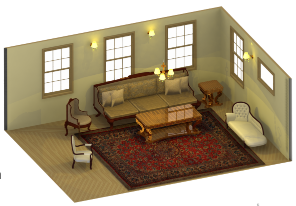
FACTORS

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



FACTORS

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

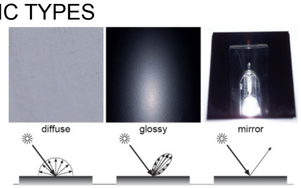
THE BIG PICTURE (BASIC)

- Light: energy in a range of wavelengths
  - White light – all wavelengths
  - Colored (e.g. red) – subset of wavelengths
- Surface "color" – reflected wavelength
  - White – reflects all lengths
  - Black – absorbs everything
  - Colored (e.g. red) absorbs all but the reflected color
- Multiple light sources add (energy sums)

MATERIALS

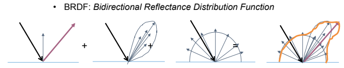
- Surface reflectance:
  - Illuminate surface point with a ray of light from different directions
  - How much light is reflected in each direction?

BASIC TYPES



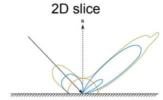
REFLECTANCE DISTRIBUTION MODEL

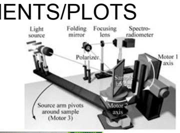
- Most surfaces exhibit complex reflectances
  - Vary with incident and reflected directions.
  - Model with combination – known as BRDF
    - BRDF: *Bidirectional Reflectance Distribution Function*

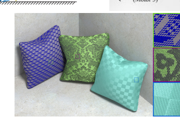


BRDF MEASUREMENTS/PLOTS

2D slice

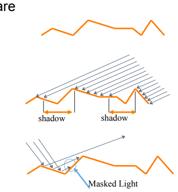




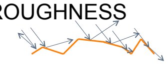


SURFACE ROUGHNESS

- at a microscopic scale, all real surfaces are rough
- cast shadows on themselves
- "mask" reflected light:



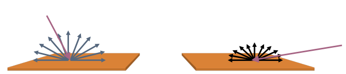
SURFACE ROUGHNESS



- notice another effect of roughness:
  - each "microfacet" is treated as a perfect mirror.
  - incident light reflected in different directions by different facets.
  - end result is mixed reflectance.
    - smoother surfaces are more specular or glossy.
    - random distribution of facet normals results in diffuse reflectance.

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
  - what does the reflected intensity depend on?

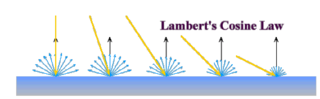


LAMBERT'S COSINE LAW

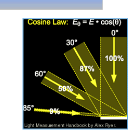
- ideal diffuse surface reflection

  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
- reflected intensity
- independent of viewing direction
- depends on surface orientation wrt light
- often called Lambertian surfaces

DIFFUSE (LAMBERT)



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



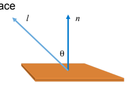
COMPUTING DIFFUSE REFLECTION

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

$I_{diffuse} = k_d I_{light} (\mathbf{n} \bullet \mathbf{l})$



Scalar (B/W intensity) or 3-tuple (color)

- k_d: diffuse coefficient, surface color
- I_{light}: incoming light intensity
- $I_{diffuse}$: outgoing light intensity (for diffuse reflection)

NB: **Always normalize vectors used in lighting!!**

- **n**, **l** should be unit vectors

DIFFUSE LIGHTING EXAMPLES

- Lambertian sphere from several lighting angles:

- need only consider angles from 0° to 90°

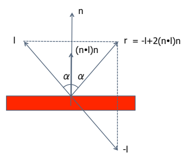
DIFFUSE INTERREFLECTION

SPECULAR HIGHLIGHTS

Michiel van de Panne

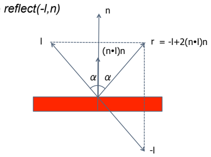
PHYSICS OF SPECULAR REFLECTION

- Geometry of specular (perfect mirror) reflection
  - Snell's law special case: Law of Reflection



PHYSICS OF SPECULAR REFLECTION

- Geometry of specular (perfect mirror) reflection
  - Snell's law special case: Law of Reflection
  - In GLSL: use *reflect(-l,n)*



CALCULATING R VECTOR

P = N cos θ |**L**| |**N**|

P = N cos θ

P = N (N · L)

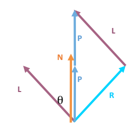
projection of **L** onto **N**

L, N are unit length

2 P = R + L

2 P – L = R

2 (N (N · L)) - L = R

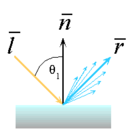


EMPIRICAL APPROXIMATION

- 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

EMPIRICAL APPROXIMATION

- Angular falloff



- How to model this falloff?

PHONG LIGHTING

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

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

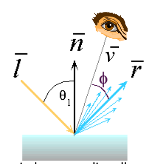
$$\mathbf{I}_{specular} = \mathbf{k}_s \mathbf{I}_{light} (\mathbf{v} \bullet \mathbf{r})^{n_s}$$

φ: angle between r and view direction v

n_s : purely empirical constant, varies rate of falloff

k_s: specular coefficient, highlight color

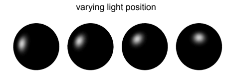
no physical basis, "plastic" look



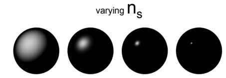
reminder: normalize all vectors: n,l,r,v

PHONG EXAMPLES

varying light position



varying n_s

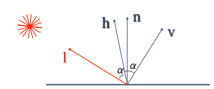


ALTERNATIVE MODEL

Blinn-Phong model (Jim Blinn, 1977)

- Variation with better physical interpretation
  - **h**: halfway vector; r: roughness

$$I_{specular} = k_s \cdot (\mathbf{h} \cdot \mathbf{n})^{1/r} \cdot I_{light}\text{; with } \mathbf{h} = (\mathbf{l} + \mathbf{v})/2$$



MULTIPLE LIGHTS

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

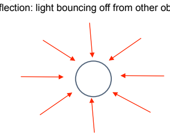
$$\sum_p I_p (k_d (n \cdot l_p) + k_s (r_p \cdot v)^n)$$

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

LIGHT SOURCES

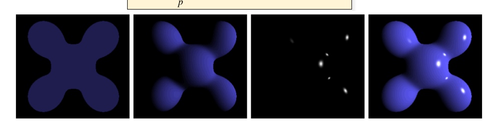
- ambient lights
  - no identifiable source or direction
  - hack for replacing true global illumination
    - (diffuse interreflection: light bouncing off from other objects)



ILLUMINATION EQUATION (PHONG)

- If we take the previous formula and add ambient component:

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



Ambient + Diffuse + Specular = Phong Reflection

LIGHT

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

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

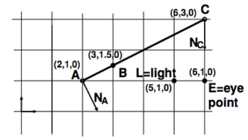
- Blue light on white surface?
- Blue light on red surface?

LIGHT AND MATERIAL SPECIFICATION

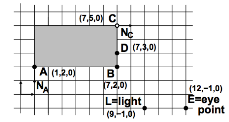
- Light source: amount of RGB light emitted
  - value = intensity per channel
    e.g., (1.0,0.5,0.5)
  - every light source emits ambient, diffuse, and specular light
- Materials: amount of RGB light reflected
  - value represents percentage reflected
    e.g., (0.0,1.0,0.5)
- Interaction: multiply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

WHITEBOARD EXAMPLE

$I_a = (.2, .5, .2), I_L = (1.0, 1.0, 1.0), k_a = (.1, .1, .1), k_d = (.3, .8, .7), k_s = (.8, .8, .8), n = 20.$



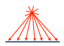
WHITEBOARD EXAMPLE

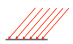


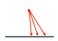
- ambient light color I_a is (.1,.1,.2)
- light color I_L is (1.0, .9, .9)
- diffuse material color k_d is (.9, .2, .9)
- ambient material color k_a is (.2, .2, .2)
- specular material color k_s is (1, 1, 0)
- shininess exponent is 30

LIGHT SOURCE TYPES

- Point Light
  - light originates at a point
  -
- Directional Light (point light at infinity)
  - light rays are parallel
  - Rays hit a planar surface at identical angles
  -
- Spot Light
  - point light with limited angles
  -

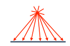


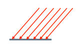


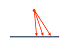


LIGHT SOURCE TYPES

- Point Light
  - light originates at a point
  - defined by **location** only
- Directional Light (point light at infinity)
  - light rays are parallel
  - Rays hit a planar surface at identical angles
  - defined by **direction** only
- Spot Light
  - point light with limited angles
  - defined by **location, direction, and angle range**

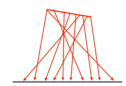






LIGHT SOURCES

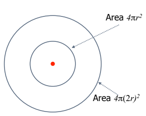
- area lights
  - light sources with a finite area
  - more realistic model of many light sources
  - much more complex!



WHICH LIGHTS/MATERIALS ARE USED HERE?

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



ILLUMINATION EQUATION WITH ATTENUATION

- For multiple light sources:

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

- $d_{\bar{p}}$: distance between surface and light source + distance between surface and viewer, A – attenuation function

WHEN TO APPLY LIGHTING MODEL?

per polygon "flat shading"	per vertex "Gouraud shading"	per pixel "per pixel lighting" "Phong shading"

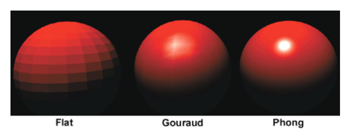
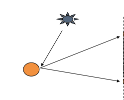


Image © Intergraph Computer Systems

LIGHTING VS. SHADING

- **lighting**
  - process of computing the luminous intensity (i.e., outgoing light) at a particular 3-D point, usually on a surface
- **shading**
  - the process of assigning colors to pixels
- (why the distinction?)



APPLYING ILLUMINATION

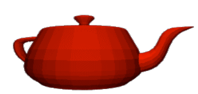
- we now have an illumination model for a point on a surface
- if surface defined as mesh of polygonal facets, *which points should we use?*
  - fairly expensive calculation
  - several possible answers, each with different implications for visual quality of result

APPLYING ILLUMINATION

- polygonal/triangular models
  - each facet has a constant surface normal
  - if light is directional, diffuse reflectance is constant across the facet
  - why?

FLAT SHADING

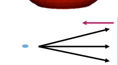
- simplest approach calculates illumination at a single point for each polygon

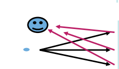


- obviously inaccurate for smooth surfaces

FLAT SHADING APPROXIMATIONS

- if an object really <u>is</u> faceted, is this accurate?
- no!
  - for point sources, the direction to light varies across the facet
  - for specular reflectance, direction to eye varies across the facet



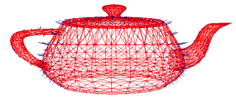


IMPROVING FLAT SHADING

- what if evaluate Phong lighting model at each pixel of the polygon?
  - better, but result still clearly faceted
- for smoother-looking surfaces we introduce *vertex normals* at each vertex
  - usually different from facet normal
  - used *only* for shading
  - think of as a better approximation of the *real* surface that the polygons approximate

VERTEX NORMALS

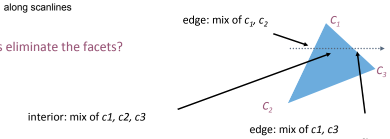
- vertex normals may be
  - provided with the model
  - computed from first principles
  - approximated by averaging the normals of the facets that share the vertex



GOURAUD SHADING

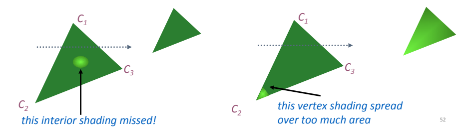
- most common approach
  - perform Phong lighting at the vertices
  - linearly interpolate the resulting colors over faces
    - along edges
    - along scanlines

does this eliminate the facets?



GOURAUD SHADING ARTIFACTS

- often appears dull, chalky
- lacks accurate specular component
  - if included, will be averaged over entire polygon

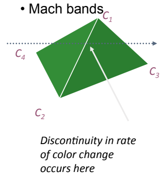


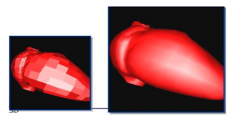
GOURAUD SHADING ARTIFACTS

- Mach bands
  - eye enhances discontinuity in first derivative
  - very disturbing, especially for highlights

GOURAUD SHADING ARTIFACTS

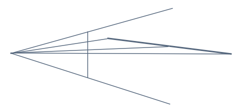
- Mach bands

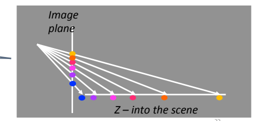




GOURAUD SHADING ARTIFACTS

- perspective transformations
  - affine combinations only invariant under affine, **not** under perspective transformations
  - thus, perspective projection alters the linear interpolation!





GOURAUD SHADING ARTIFACTS

- perspective transformation problem
  - colors slightly "swim" on the surface as objects move relative to the camera
  - usually ignored since often only small difference
    - usually smaller than changes from lighting variations
  - to do it right
    - either shading in object space
    - or correction for perspective foreshortening
    - expensive – thus hardly ever done for colors

PHONG SHADING

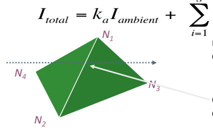
- linearly interpolating surface normal across the facet, applying Phong lighting model at every pixel
  - same input as Gouraud shading
  - pro: much smoother results
  - con: considerably more expensive
- **not** the same as Phong lighting
  - common confusion
  - Phong lighting: empirical model to calculate illumination at a point on a surface

PHONG SHADING

- linearly interpolate the vertex normals
  - compute lighting equations at each pixel
  - can use specular component

$$I_{total} = k_a I_{ambient} + \sum_{i=1}^{\# lights} I_i \left(k_d \left(\mathbf{n} \cdot \mathbf{l_i} \right) + k_s \left(\mathbf{v} \cdot \mathbf{r_i} \right)^{n_{shiny}} \right)$$

remember: normals used in diffuse and specular terms



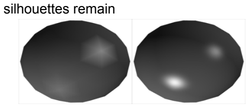
discontinuity in normal's rate of change harder to detect

PHONG SHADING DIFFICULTIES

- more computationally expensive
  - per-pixel vector normalization and lighting computation!
  - floating point operations required
  - straightforward with shaders
- lighting after perspective projection
  - messes up the angles between vectors
  - have to keep eye-space vectors around

SHADING ARTIFACTS: SILHOUETTES

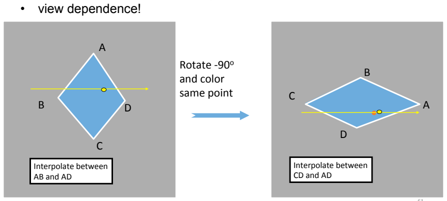
- polygonal silhouettes remain



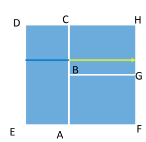
Gouraud *Phong*

SHADING ARTIFACTS: ORIENTATION

- interpolation dependent on polygon orientation
  - view dependence!



Shading Artifacts: Shared Vertices



vertex B shared by two rectangles on the right, but not by the one on the left

first portion of the scanline is interpolated between DE and AC

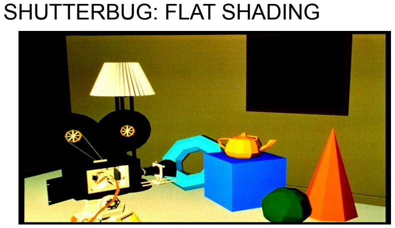
second portion of the scanline is interpolated between BC and GH

a large discontinuity could arise

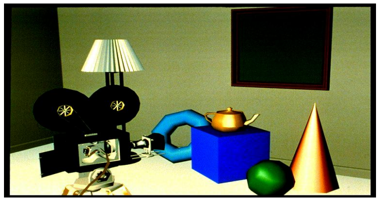
SHADING MODELS SUMMARY

- flat shading
  - compute Phong lighting once for entire polygon
- Gouraud shading
  - compute Phong lighting at the vertices and interpolate lighting values across polygon
- Phong shading
  - compute averaged vertex normals
  - interpolate normals across polygon and perform Phong lighting across polygon

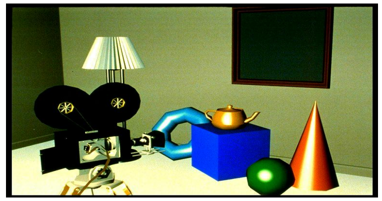
SHUTTERBUG: FLAT SHADING



SHUTTERBUG: GOURAUD SHADING

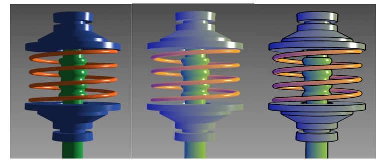


SHUTTERBUG: PHONG SHADING



NON-PHOTOREALISTIC SHADING

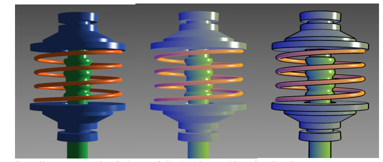
- cool-to-warm shading $k_w = \frac{1+\mathbf{n}\cdot\mathbf{l}}{2}, c = k_w c_w + (1-k_w)c_c$



http://www.cs.utah.edu/~gooch/SIG98/paper/drawing.html

NON-PHOTOREALISTIC SHADING

- draw silhouettes: if $(\mathbf{e}\cdot\mathbf{n_0})(\mathbf{e}\cdot\mathbf{n_1}) \le 0$, **e**=edge-eye vector
- draw creases: if $(\mathbf{n_0}\cdot\mathbf{n_1}) \le threshold$



http://www.cs.utah.edu/~gooch/SIG98/paper/drawing.html