shading concepts

• light sources and illumination in OpenGL
  – separate R, G, B channel computations
  – “local illumination” model
    • no notion of obscuration, scattering
      – only direction from source to surface considered, not intervening surfaces
      – shadows cast by objects between source and illuminated surface
      – no mutual illumination
    • no sense of reflections, let alone multiple reflections
    • transparency without refraction
    • surface emission not contributing as light source
  • later: global (ray tracing and radiosity) methods
shading concepts

- **discrete light sources**
  - ambient + maximum eight point sources
  - limited dynamic range of intensity representation
- **Phong model of reflectance for each source**, sum of:
  - intensity due to reflectance of ambient light
    - \( f(\text{ambient intensity, ambient reflectance term}) \)
  - intensity due to diffuse reflectance
    - \( f(\text{point source diffuse intensity/direction/distance, surface diffuse reflectance, surface normal/position}) \)
  - intensity due to specular reflectance
    - \( f(\text{point source specular intensity/direction, surface specular reflectance, surface normal, camera position, shininess}) \)
shading concepts

• light sources
  – ambient (global, omni-directional, additive)
  – point sources (positioned, with direction at least)
    • spherical distribution of luminances
    • inverse square law for proximate sources
    • no modeling of umbra (full shadow) versus penumbra
  – spot lights
    • position of apex of cone (w=0 for infinite source)
    • direction vector, cone cutoff angle, concentration $e$ for $\cos^e$
shading concepts

• material properties
  – ambient reflectance (often same as diffuse)
  – diffuse reflectance (matte, Lambertian)
  – specular reflectance
    • shininess (0-128)
  – translucent (fractionally opaque, alpha channel)
  – emission (hack, hack!)
shading concepts

- surface modeled by facets bounded by vertices
  - normals attributed to vertices only
    - creases where not desired
    - blending where creases desired
    - failure of Gouraud interpolation algorithm
  - no modeling of *anisotropic* reflectance functions
    - surface striations, grain, ..
the following are due to

COMS 4160, Lecture 17: Shading 1
Ravi Ramamoorthi
http://www.cs.columbia.edu/~cs4160

Guest Lecturer:
Aner Benartzi
General Considerations

• Surfaces are described as having a position, and a normal at every point.

• Other vectors used
  – \( L \) = vector to the light source
    • light position minus surface point position
  – \( E \) = vector to the viewer (eye)
    • viewer position minus surface point position
Diffuse reflection (Lambertian)

- Many materials are rough on a small scale.
  - matte paint, cloth, unfinished wood, etc.
- Incoming light is reflected equally in all directions.
- Outgoing radiance is dependent on irradiance angle

\[ R = I \cos(\theta) = I(\mathbf{\hat{N}} \times \mathbf{\hat{L}}) \]

(real materials don’t follow the cosine drop-off exactly)
Mirror reflection (Specular)

• Many materials are also smooth even on a small scale
  – glass, glossy paint, plastic, polished metal, etc.
• Incoming light is reflected mainly in the mirror direction.

\[ \theta_{in} = \theta_{out} \]
Phong reflection

- Specularities are only singular points in perfect mirrors.
- Phong model allows for some light reflected in off-specular directions.

\[ R = I \cos^n(\theta) = I(E \times Lr)^n \]

(Real materials don’t follow the cosine drop-off exactly, nor use \( Lr \))
Meaning of negative dot products

• If \((N \cdot L)\) is negative, then the light is behind the surface, and cannot illuminate it.

• If \((N \cdot E)\) is negative, then the viewer is looking at the underside of the surface and cannot see its front-face.

• In both cases, I is clamped to Zero.
Triangle Meshes as Approximations

• Most geometric models are large collections of triangles.
• Triangles have 3 vertices, each with a position, color, normal, and other parameters (such as n for Phong reflection).
• The triangles are an approximation to the actual surface of the object.
Coloring Between the Lines

• We know how to calculate the light intensity given:
  – surface position
  – normal
  – viewer position
  – light source position (or direction)

• How do we shade a triangle between it’s vertices, where we aren’t given the normal?
Flat vs. Gouraud Shading

Flat - Determine that each face has a single normal, and color the entire face a single value, based on that normal.

Gouraud – Determine the color at each vertex, using the normal at that vertex, and interpolate linearly for the pixels between the vertex locations.
Flat Shading

The simplest approach, flat shading, calculates illumination at a single point for each polygon:

If an object really is faceted, is this accurate?
Is flat shading realistic for faceted object?

No:

- For point sources, the direction to light varies across the facet
- For specular reflectance, direction to eye varies across the facet
Flat Shading

We can refine it a bit by evaluating the Phong lighting model at each pixel of each polygon, but the result is still clearly faceted:

To get 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 – Details 1

• Inter-vertex interpolation can be done in object space (along the face), but it is simpler to do it in image space (along the screen).

• 2 ways for a vertex to get its normal:
  – given when the vertex is defined.
  – take all the normals from faces that share the vertex, and average them.
The intensity at the 3 vertices, $I_1$, $I_2$, $I_3$, is calculated. Each $I_k$ is projected onto pixel $(x_k, y_k)$.

The intensities of 2 points on the same scan, $y=s$, $I_a$, $I_b$, are found. These points lie on the edges of the triangle.

\[
I_a = I_1 - (I_1 - I_2) \frac{y_1 - y_s}{y_1 - y_2} \quad I_b = I_1 - (I_1 - I_3) \frac{y_1 - y_s}{y_1 - y_3}
\]

The intensity at an interior point at $(x_p, s)$, $I_p$ is found as follows:

\[
I_p = I_b - (I_b - I_a) \frac{x_b - x_p}{x_b - x_a}
\]
Gouraud Shading

This is the most common approach

- Perform Phong lighting at the vertices
- Linearly interpolate the resulting colors over faces
  - Along edges
  - Along scanlines

- This is what OpenGL does

Does this eliminate the facets?
Gouraud Shading

**Artifacts**

- Often appears dull, chalky
- Lacks accurate specular component
  - If included, will be averaged over entire polygon

Can’t shade that effect!
Gouraud Shading

- Artifacts
  - Mach Banding
    - Artifact at discontinuities in intensity or intensity slope

Beware of Mach Banding

Discontinuity in rate of color change occurs here
Gouraud and Errors

- $I_1 = 0$ because $(N \text{ dot } E)$ is negative.
- $I_2 = 0$ because $(N \text{ dot } L)$ is negative.
- Any interpolation of $I_1$ and $I_2$ will be 0.
2 Phongs make a Highlight

• Besides the Phong Reflectance model (\(\cos^n\)), there is a Phong Shading model.

• Phong Shading: Instead of interpolating the intensities between vertices, interpolate the normals.

• The entire lighting calculation is performed for each pixel, based on the interpolated normal.
Phong Shading

Phong shading is not the same as Phong lighting, though they are sometimes mixed up

• Phong lighting: the empirical model we’ve been discussing to calculate illumination at a point on a surface
• Phong shading: linearly interpolating the surface normal across the facet, applying the Phong lighting model at every pixel
  – Same input as Gouraud shading
  – Usually very smooth-looking results:
  – But, considerably more expensive
**Phong Shading**

*Linearly interpolate the vertex normals*

- Compute lighting equations at each pixel
- Can use specular component

\[
I_{\text{total}} = k_a I_{\text{ambient}} + \sum_{i=1}^{\#\text{lights}} I_i \left( k_d \left( \hat{N} \times \hat{L}_i \right) + k_s \left( \hat{V} \times \hat{R}_i \right) \right)^{\text{shiny}}
\]

Remember: Normals used in diffuse and specular terms

Discontinuity in normal’s rate of change is harder to detect
Shortcomings of Shading

Polygonal silhouettes remain

Gouraud

Phong
Interpolation dependent on polygon orientation

Interpolate between AB and AD

Interpolate between CD and AD

Rotate \(-90^\circ\) and color same point
Problems at Shared Vertices

Vertex B is shared by the two rectangles on the right, but not by the one on the left.

The first portion of the scanline is interpolated between DE and AC.

The second portion of the scanline is interpolated between BC and GH.

A large discontinuity could arise.
Bad Vertex Averaging