### News
- Reminders
  - Don't need to tell us you're taking grace days; they're assumed if you turn in late.
  - Separate for written homework and project.
  - Exception: HW2 not accepted after 11am Fri.
  - Solutions posted then so you can use them when studying for midterm.
- Midterm
  - Topics:
    - All material through Rasterization (Wed Feb 10 lecture).
    - Format:
      - Closed book.
      - You may have simple (nongraphing) calculators.
      - You may have notes on one side of 8.5x11" sheet of paper.
      - Must be handwritten by you; cannot be xeroxed/printed.
      - You'll keep these notes; for final, can use back side of page as well.
  - Logistics:
    - Must have UBC ID face up.
    - Backpacks/coats at front of room.
    - Phones off.

### Midterm
- Review: Phong Lighting
  - Most common lighting model in computer graphics.
    - (Phong Bui-Tuong, 1975)
    - \[ I_{\text{specular}} = k_s I_{\text{light}} (\cos \phi)^n \]  
    - \( n \) : purely empirical constant, varies rate of falloff.
    - \( k_s \) : specular coefficient, highlight color.
    - No physical basis, works ok in practice.

### Light Source Falloff
- Quadratic falloff
  - Brightness of objects depends on power per unit area. The power per unit area for a point or spot light decreases quadratically with distance.
    - Area \( 4\pi r^2 \)
    - Area \( 4\pi (2r)^2 \)

### Lighting in OpenGL
- Light source: Amount of RGB light emitted.
- Value represents percentage of full intensity.
  - e.g., \( (1.0,0.5,0.5) \)
  - Every light source emits ambient, diffuse, and specular light.
  - Value represents percentage reflected.
    - e.g., \( (0.0,1.0,0.5) \)
  - Interaction: Multiply components.
    - Red light \( (1.0,0.0) \) x green surface \( (0,1,0) \) = black \( (0,0,0) \)

### Phong Lighting
- Phong Lighting Model
  - Combine ambient, diffuse, specular components.
    - \[ I_{\text{total}} = k_d I_{\text{ambient}} + \sum k_i (\hat{n} \cdot \hat{l}) + k_s (\hat{v} \cdot \hat{r})^n \]
  - Commonly called Phong lighting.
    - Once per light.
    - Once per color component.
  - Reminder: Normalize your vectors when calculating.
    - Normalize all vectors: \( n, l, r, v \)

### Lighting Review
- Lighting models
  - Ambient.
    - Normals don't matter.
  - Lambert/diffuse.
    - Angle between surface normal and light.
  - Phong/specular.
    - Surface normal, light, and viewpoint.

### Lighting in OpenGL
- Light source: Amount of RGB light emitted.
- Value represents percentage of full intensity.
  - e.g., \( (1.0,0.5,0.5) \)
  - Every light source emits ambient, diffuse, and specular light.
  - Value represents percentage reflected.
    - e.g., \( (0.0,1.0,0.5) \)
  - Interaction: Multiply components.
    - Red light \( (1.0,0.0) \) x green surface \( (0,1,0) \) = black \( (0,0,0) \)
Lighting in OpenGL

\[ \text{glLightfv(GL_LIGHT0, GL_AMBIENT, amb_light_rgba); } \]
\[ \text{glLightfv(GL_LIGHT0, GL_DIFFUSE, dif_light_rgba); } \]
\[ \text{glLightfv(GL_LIGHT0, GL_SPECULAR, spec_light_rgba); } \]
\[ \text{glLightfv(GL_LIGHT0, GL_POSITION, position); } \]
\[ \text{glEnable(GL_LIGHT0); } \]

- warning: glMaterial is expensive and tricky
  - use cheap and simple glColor when possible
  - see OpenGL Pitfall #14 from Kilgard's list
  \( \text{http://www.opengl.org/resources/features/KilgardTechniques/oglpitfall/} \)

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

- 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

Flat Shading Approximations

- if an object really is 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

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, and what OpenGL does
  - perform Phong lighting at the vertices
  - linearly interpolate the resulting colors over faces
    - along edges
    - along scanlines

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

- 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

- linearity 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