Three Shading Options

What about the space between vertices?

Flat Shading

• Illumination is computed at every vertex
• The average illumination is the average of the illumination at the vertices
• Polygon filling then paints this color
• Downside:
  – 3D structure (from angle changes) is lost
  – Boundaries between surfaces become artifacts
Illumination at a vertex?

• Wait a minute
  – *Illumination depends on the surface normal*
  – What’s the surface normal at a vertex?
  – It’s ambiguous – vertex shared by surfaces!

• Solution #1:
  – User-set vertex normals
  – OpenGL uses this solutions

• Solution #2:
  – Average adjoining surface normals
How users set normals

- How do you pick a normal?
- Case #1: polygonal approximation to a smooth surface

- Set normals to underlying “true” normal
Rule(s) to set normals (II)

- Case #2: Truly polygonal object
  - Double up: create multiple vertices at one position, one for each adjacent surface.
  - Each vertex now has normal of associated surface.
Smooth (Gouraud) Shading

• Compute the illumination at every vertex

• Interpolate colors along edges
  – Between vertices

• When filling the polygon, interpolate colors between scan-line intersections
Smooth Shading Example

\[(r_1, g_1, b_1)\]
\[((r_1+r_2)/2,...)\]
\[(r_2, g_2, b_2)\]
Phong Shading

• Calculate normals at vertices
• Interpolate **normals** along edges
• When polygon filling:
  – Interpolate normals between scan-line intersections
  – **Calculate color using interpolated normals**

**Warning:** do not confuse *Phong Shading* with *Phong Reflectance*
Selecting a Shading Model

• Case 1: Object is curved
  – Phong shading (most realistic)
  – Smooth shading (slightly faster)

• Case 2: Large flat surface, divided into multiple polygons
  – Same as above

• Case 3: Flat surface, true boundaries
  – Flat shading
  – Replicate vertices (for normals)
Not so subtle distinction

• Smooth shading discards 3-D normals
  – Operates solely with R,G,B values.
  – Direction to lights within surfaces fixed.

• Phong shading adjust 3-D normals.
  – Illumination better within surfaces.
  – What about direction to lights?
Example of Case #2

Impact of polygon size on appearance

Source: www.opengl.com

*In this example, is illumination being recomputed internal to surface faces?*
More on Normals - Blender

Normals

Introduction

In geometry, a normal is a direction or line that is perpendicular to something, typically a triangle or surface but can also be relative to a line, a tangent line for a point on a curve, or a tangent plane for a point on a surface.

A visualization of the face normals of a torus.

In the figure above, each blue line represents the normal for a face on the torus. The lines are each perpendicular to the face on which they lie. The visualization can be activated in the Mesh Display panel.
Part of P4 – A Smoother Cow

Your P3 ray tracer essentially creates this illumination of the cow object.

Put essentially the Phong model into your ray tracer.
P4 - Smoothing Approach

- Identify shared vertices
  - Lookup vertex numbers for a given triangle
- Compute true normal for every surface
  - Here assume A then B then C traversal
  - Compute the average normal at a vertex
    - Exclude adjacent faces too far off in orientation
- Use beta and gamma to interpolate normals

\[ N_i = (1 - \beta - \gamma) N_A + \beta N_B + \gamma N_C \]
A Glimpse at Shaders

• Original OpenGL and the Fixed Pipeline
  – Example: Gouraud Shading
• Modern OpenGL means writing shaders
  – Vertices
  – Fragments

In CS410, know the role shaders play. Do not expect to know how to write shaders.
One More Glimpse

https://www.learnopengles.com/tag/gouraud-shading/