Shading Models
For each light source and each color component, the Phong model can be written (without the distance terms) as

\[ I = k_d L_d \cdot n + k_s L_s (v \cdot r)^\alpha + k_a L_a \]

For each color component we add contributions from all sources
Shading

Computing the internal colors for a polygon given the vertices, lights, viewer, etc.
Flat Shading

Polygons have a single normal
We assume distant viewer, light
Carry out lighting equations once for each polygon
Gouraud Shading

Each vertex has a normal

Compute color at vertices (e.g. using Phong Illumination Model)

*Interpolate* colors over the polygons

This is what we have been using in our v/f shaders
Interpolation

\[ R_h = R_a + \alpha (R_b - R_a) \]

where

\[ \alpha = \frac{y_h - y_a}{y_b - y_a} \]
Computing Mesh Normals (if you don’t have them!)

For polygonal models, Gouraud proposed we use the average of normals around a mesh vertex

\[
\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1| + |\mathbf{n}_2| + |\mathbf{n}_3| + |\mathbf{n}_4|}
\]

\[
\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)
\]
Gouraud Shading – problems?

Find average normal at each vertex (vertex normals)
Apply Phong Illumination model at each vertex
Interpolate vertex shades across each polygon
Phong Shading

Find vertex normals

Interpolate *vertex normals* across polygons

Compute Phong Illumination model over the surface of the polygon at each interpolated normal (e.g. per fragment lighting!)
Gouraud vs. Phong
Toon Shader

How would you create this shading effect?

wikipedia: cel shading
Finding Contours
Discretizing Shading Levels