
CSE 472

Shading Models

Shading Models

Where to compute colors

Simulating Curved Surfaces

Smooth Shading

- Gouraud Shading
- Phong Shading

Problems with Smooth Shading

Computing Colors

We are drawing primitives!

Where do we compute a primitive's color?

- Once for the entire primitive?
- Once at each vertex or other point?
 - Vertex Shader (GPU)
- At every pixel?
 - Fragment Shader (GPU)
- Any other options?

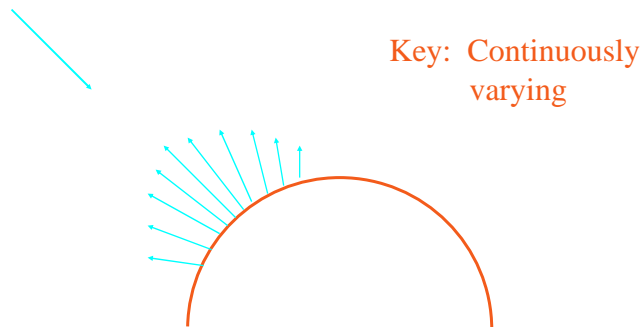
When will these options vary the color?

- Remember what affects the color...

Light on a curve

When light hits a curved surface, we are varying the angle of the surface relative to the light.

- Remember our illumination model?



But, we don't have curves!

Computer graphics systems usually tessellate curved surfaces into polygons

- Why?

Many systems only compute color at vertices

- Why?

So, how can we make a surface look curved?



One Solution (not a good one)

Tessellate very finely

- At least one polygon per pixel
- But, how do we do this?
- How many polygons will we require?

Bad Solution 2

Compromise between fine and coarse

- How about getting polygons down to, say, 5 pixel range on average?
 - About 1/25 as many

Problem: **Mach Banding**

Mach Banding

The eye is much more sensitive to edges than gradual changes

- We are VERY sensitive to edges!!

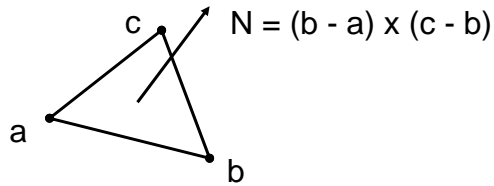
Mach Banding

- The perception of bands of color due to edges
- Demonstration program...

Vertex Normals vs. Face Normals

What are the normals to the surface?

Each polygonal face has a normal.

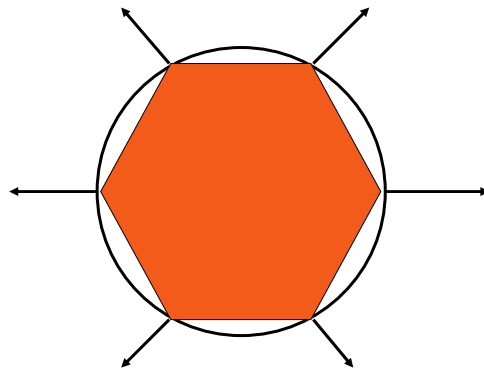


We call these **face normals**.

Polygon mesh is only an approximation.
Can we do better?

Vertex Normals vs. Face Normals

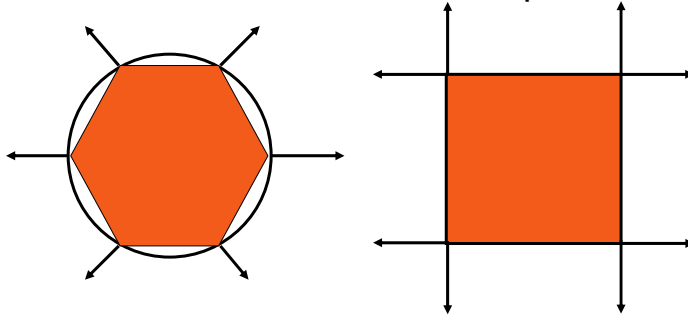
Should use the actual surface's normals



Usually stored at the vertices of the object
Can calculate as averages of face normals

Vertex Normals vs. Face Normals

Vertex normals are normals that are specified each time a vertex of a face is specified



One normal per vertex for smooth shading,
Multiple normals per vertex for hard edges

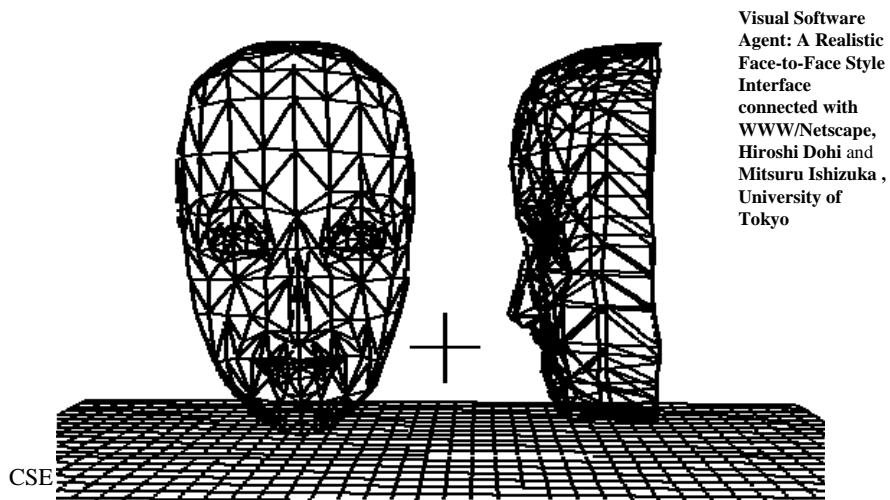
Example - Faces



What if you wanted to render your face?

- Collect a set of points on your face
- Triangulate those points
- Compute normals for each triangle
 - Using simple formula from before
- Compute normals for each vertex
 - By averaging normals of incident polygons

Visual Software Agent (VSA)



Interpolation

Given vertex normals, how to color the interior:

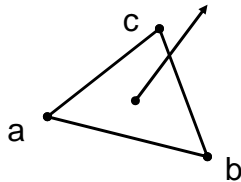
Flat shading

Smooth Shading

- Gouraud Interpolation
- Phong Interpolation

Flat Shading

Assume a constant color across the polygon



Uses face normals

Equivalent to single point sampling

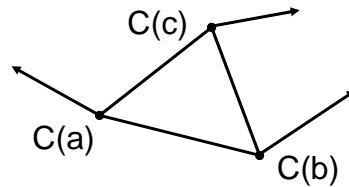
Gouraud Shading

Gouraud Shading

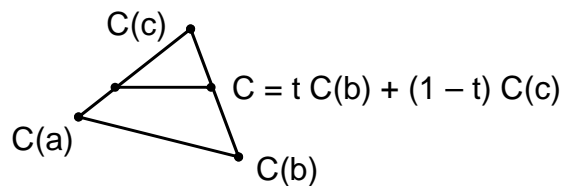
- Compute the color at the vertices
- Interpolate the color over the face of the primitive
- Most implementations of OpenGL

Gouraud Interpolation

Calculate the color at each vertex

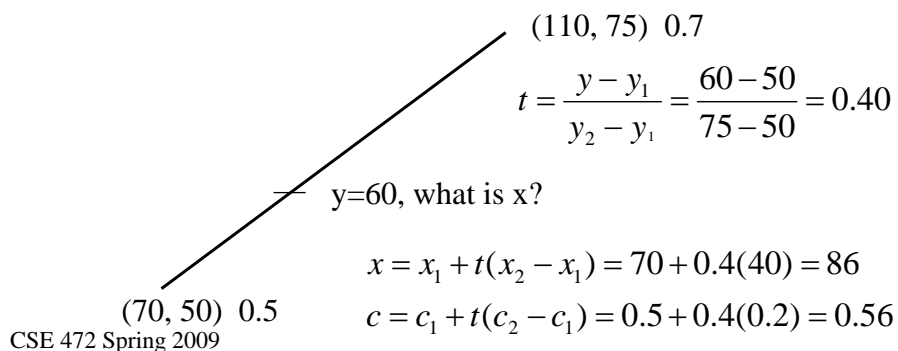


Interpolate the colors

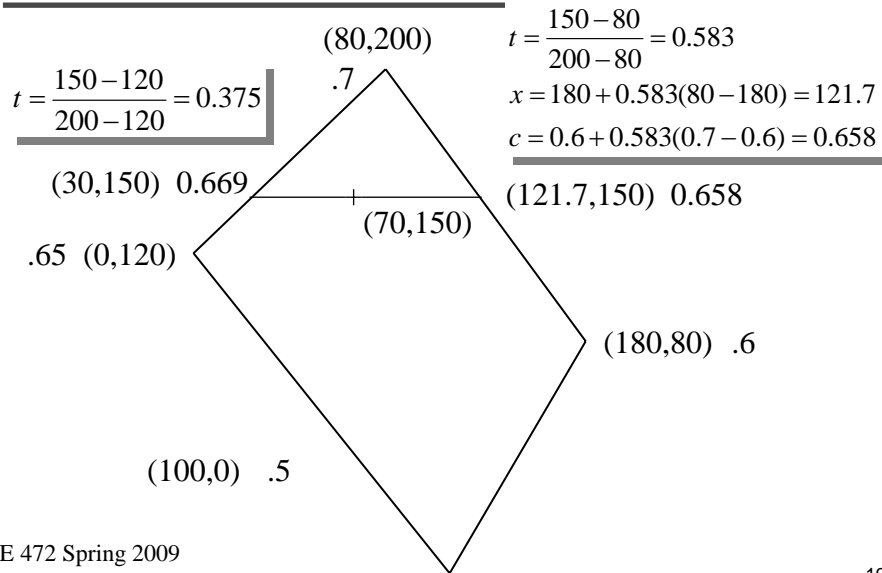


Linear Interpolation Details

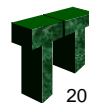
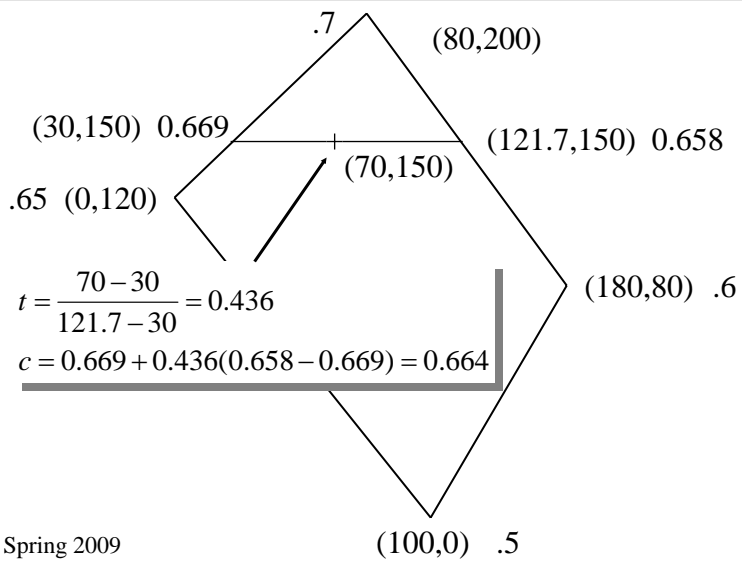
in one dimension, we can figure
where we are in the other
dimensions



Bilinear Interpolation Details



Bilinear Interpolation Details



Gouraud Shading

Advantages

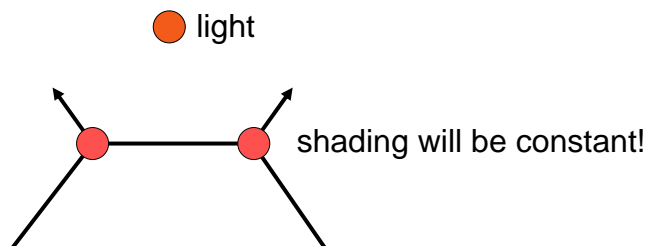
- Color math only at vertices
 - Illumination at vertex shader
 - Simple interpolation at fragment shader
- Fast and easy (if we are drawing primitives)

Disadvantages

- Color math only at vertices
- Does not simulate the actual curve, only the fact that colors interpolate
 - Color cannot increase beyond vertex colors.
 - When would that happen?

Gouraud Interpolation Problems

Misses some highlights

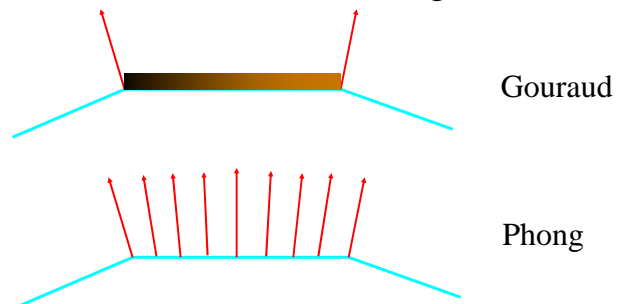


Shading is **not** linear

Phong Shading

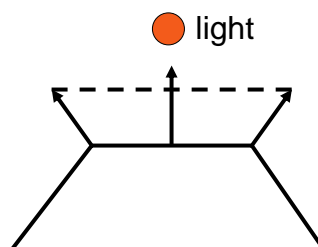
Phong Shading

- Interpolate Normals
- Compute color at each pixel
- Illumination model used at Fragment Shader



Phong Interpolation

Interpolate the normals, then compute the colors



Interpolation is usually done component-wise

Interpolation

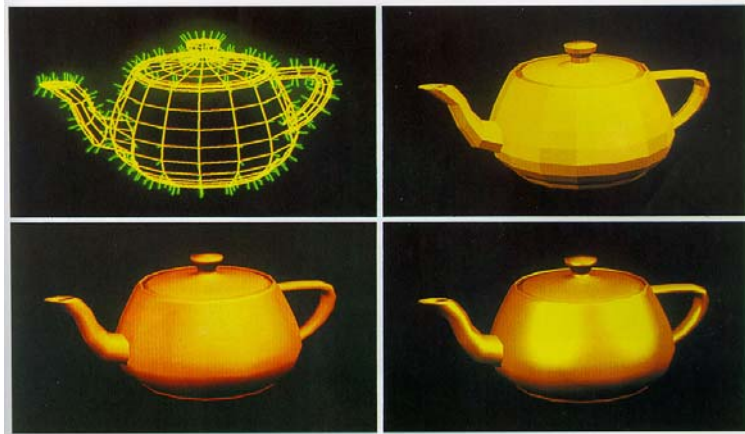


Image courtesy of Watt & Watt, Advanced Animation and Rendering Techniques

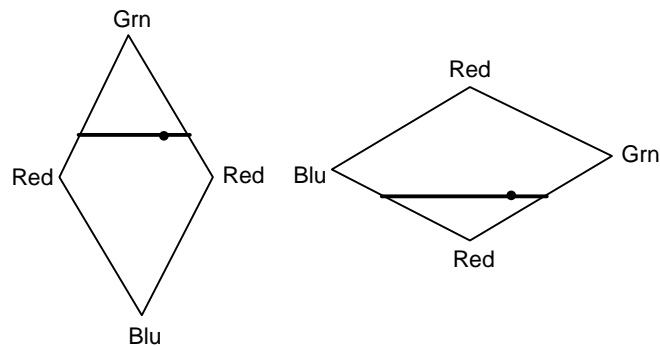
CSE 472 Spring 2009

25

Problems with Smooth Shading

Polygonal Silhouette

Orientation Dependence



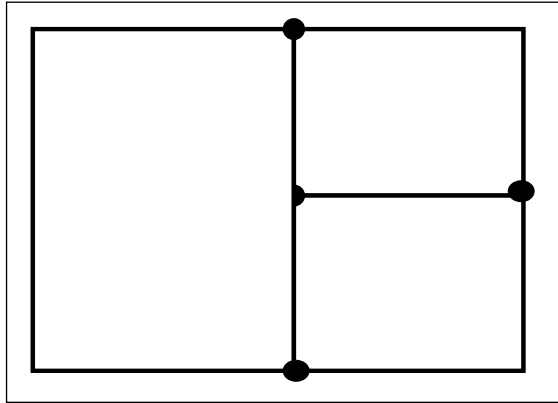
CSE 472 Spring 2009



26

Problems with Smooth Shading

Lights in the scene with large polygons
Unshared or missing vertices

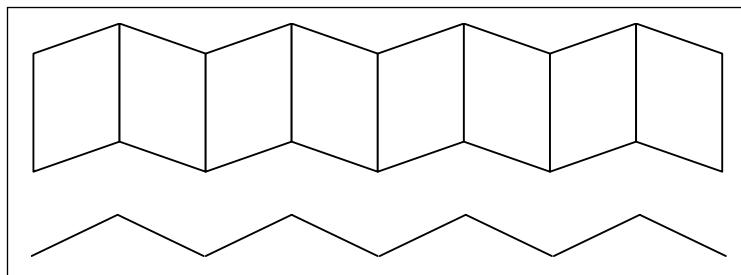


CSE 472 Spring 2009

27

Problems with Smooth Shading

Unrepresentative Vertex Normals



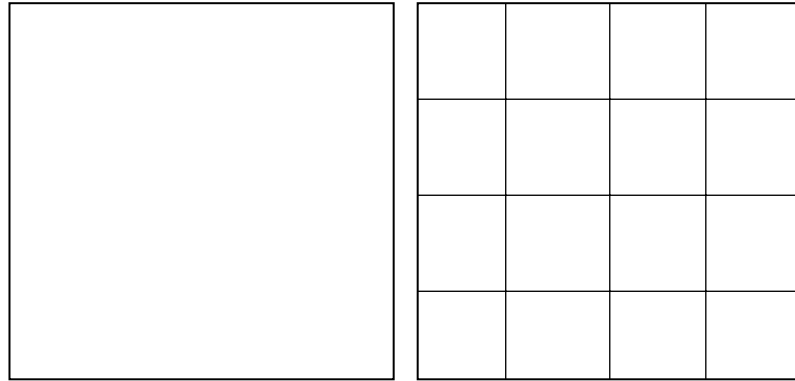
CSE 472 Spring 2009



28

Problems with Smooth Shading

Too large a polygon



Problems with Smooth Shading

Concave Polygons

