

COMP27112

Computer
Graphics
and
Image Processing

8: Rendering (2)

Toby.Howard@manchester.ac.uk



Shading a surface

- We now have a local illumination model

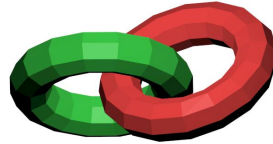
$$I_R = k_{aR}I_{aR} + \frac{I_{pR}}{d'} \left[k_{dR}(\hat{\mathbf{N}} \cdot \hat{\mathbf{L}}) + k_s(\hat{\mathbf{R}} \cdot \hat{\mathbf{V}})^n \right]$$

- How do we use it?

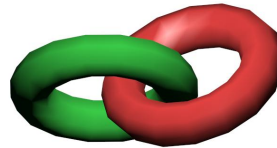
Shading a surface

- We'll look at three shading methods:

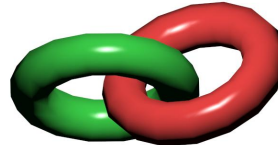
- Flat
(aka constant)



- Gouraud
(aka intensity)



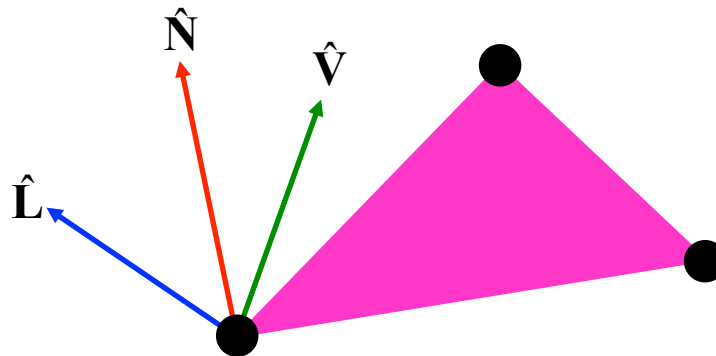
- Phong
(aka normal-vector)



Vic Baker

3

Flat shading



- This is the simplest approach
- We compute colour C at one vertex and use it for all pixels in the polygon

4

Flat shading



Henri Gouraud

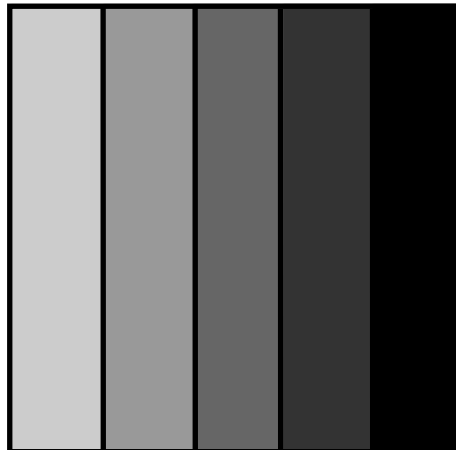
- Each polygon is uniformly coloured according to its orientation
- And we clearly see the mesh
- This is made worse by the "Mach Band" effect

5

Mach banding



Ernst Mach
(1838-1916)



Here, we see separate strips, each with a different intensity

6

MANCHESTER
1824

Mach banding



Ernst Mach
(1838-1916)

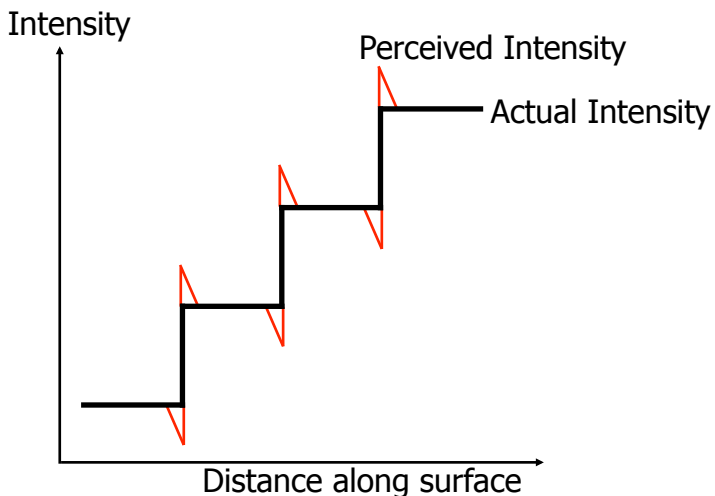


Now, we still see separate strips, but the edges between them “stand out”

7

MANCHESTER
1824

Mach banding



Intensity

Perceived Intensity

Actual Intensity


Distance along surface

- Our eyes are good at finding edges

8

MANCHESTER
1824

Mach banding applet



Repaint ☐ Non-uniform bars ☐ Smooth transitions ☐ Draw separators

Number of bars: Separator width:

	Red	Green	Blue
Left Color:	<input type="text" value="255"/>	<input type="text" value="0"/>	<input type="text" value="255"/>
Right Color:	<input type="text" value="0"/>	<input type="text" value="255"/>	<input type="text" value="255"/>
Separator Color:	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>


David Anson

- <http://www.nbb.cornell.edu/neurobio/land/OldStudentProjects/cs490-96to97/anson/MachBandingApplet/>

9

MANCHESTER
1824

Flat shading: examples



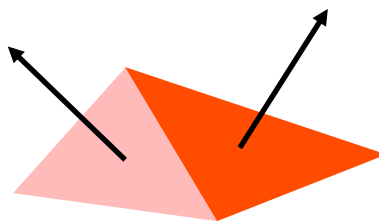
The image displays three 3D models rendered with flat shading. On the left is a white cup on a saucer. In the center is a sphere composed of many small triangles. On the right are two interlocking rings, one green and one red.

Vic Baker

10

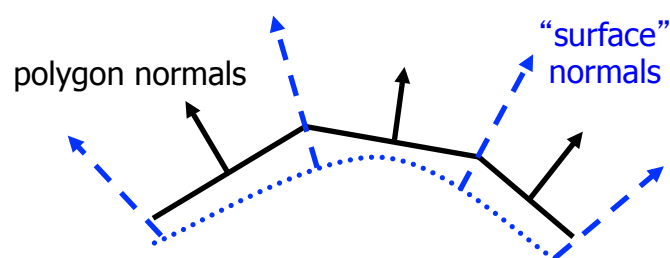
Gouraud shading

- Invented by Henri Gouraud in 1971
- Gouraud shading uses interpolation, to smooth out the discontinuities between polygons



11

Approximating a surface

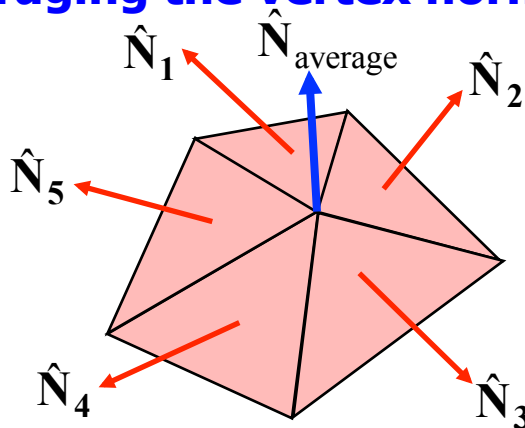


- We can approximate the normal of the underlying “surface” ...
- ...by averaging the normals where polygons share vertices

12

MANCHESTER
1824

Averaging the vertex normals



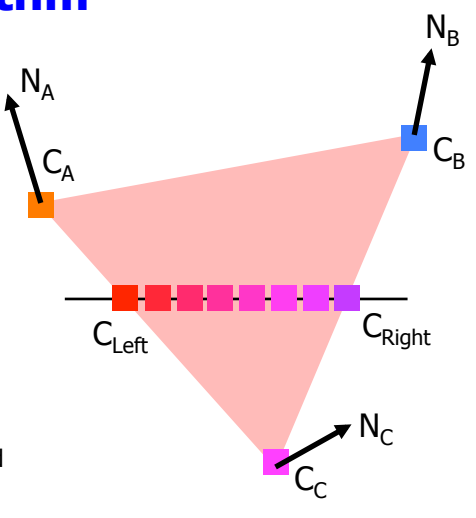
$$\hat{N}_{\text{average}} = \frac{\hat{N}_1 + \hat{N}_2 + \hat{N}_3 + \hat{N}_4 + \hat{N}_5}{|\hat{N}_1 + \hat{N}_2 + \hat{N}_3 + \hat{N}_4 + \hat{N}_5|}$$

13

MANCHESTER
1824

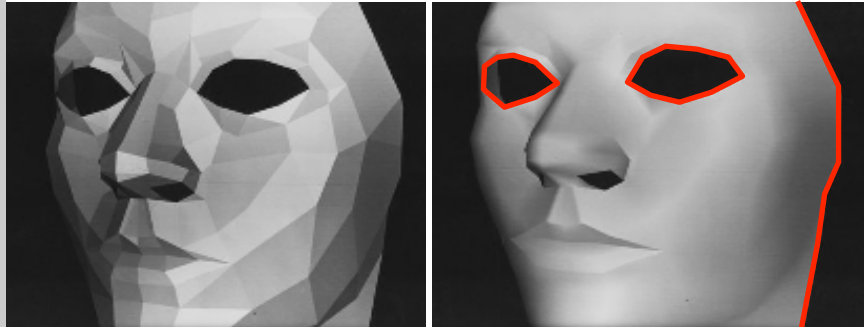
Gouraud algorithm

- compute average vertex normals at A, B and C
- compute pixel colours C_A , C_B , C_C
- for each scanline {
 - average colour C_{Left} from C_A and C_C
 - average colour C_{Right} from C_B and C_C
 - average between C_{Left} and C_{Right} along the scanline



14

Gouraud results



Henri Gouraud

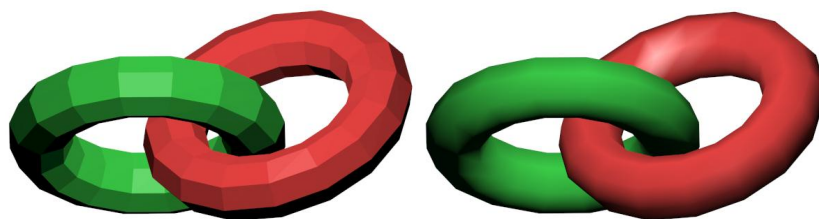
- Mesh now appears smooth
- But notice the silhouette edges, still polygonal

15

Flat shading versus Gouraud

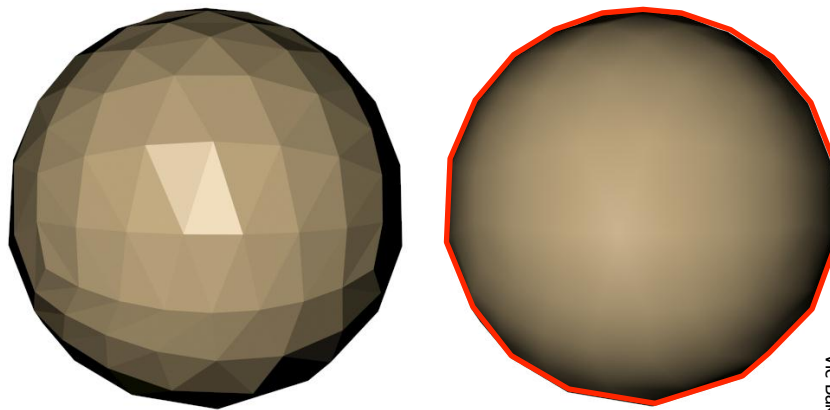


Vic Baker



16

Flat shading versus Gouraud



Vic Baker

17

Implementing Gouraud

- We need to optimise the computation as much as possible
- For each scanline we compute the colour increment between pixels:

```
deltaCol= (CRight - CLeft) / (XRight - XLeft);  
Col= CLeft;  
for (x= XLeft; X <= XRight; x++) {  
    TestAndSetPixel(x, y, Col);  
    Col= Col + deltaCol;  
}
```

- Similarly, we can also optimise by computing C_{Right} and C_{Left} incrementally

18

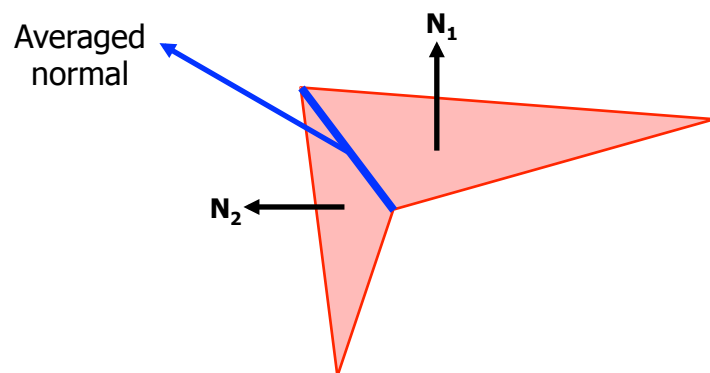
Gouraud shading: problems

- While it's fast and efficient, the method has drawbacks:
 - Specular highlights may be distorted or averaged away altogether (because Gouraud shading averages between **vertex** colours)
 - Mach banding may still be visible

19

Gouraud shading: problems

- Sometimes, edges are "shaded away"

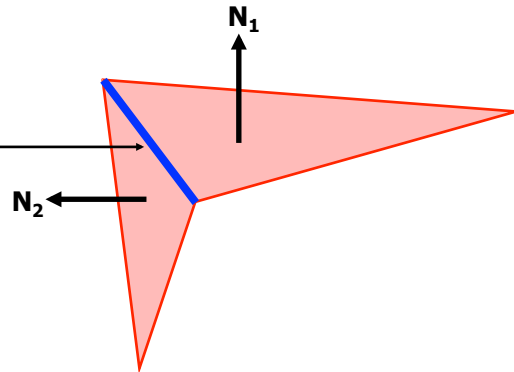


20

Gouraud shading: problems

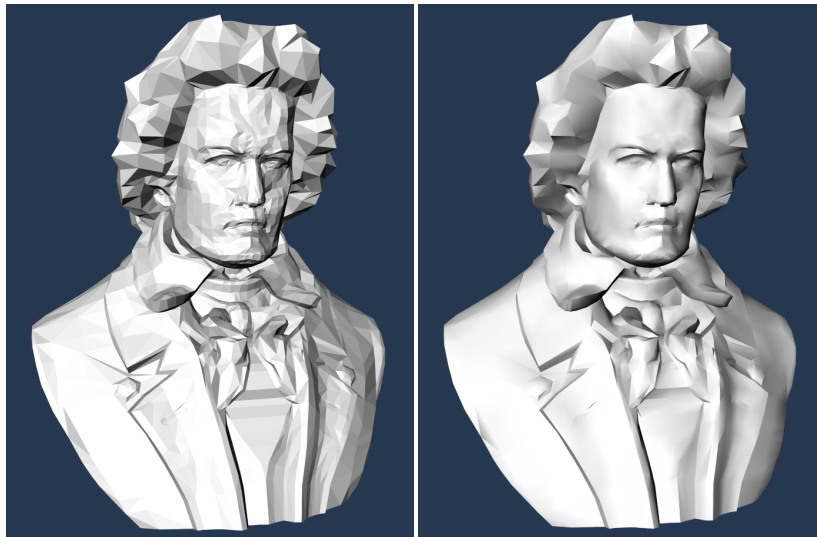
- Sometimes, edges are “shaded away”

Edge must be **tagged** in data structure to avoid interpolation across it



21

Tagged edges to stop interpolation



22

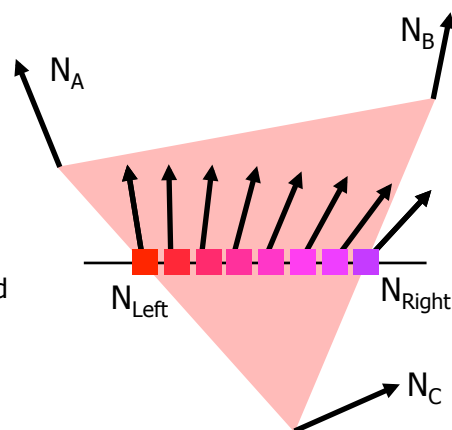
Phong interpolation

- Instead of interpolating colours, Phong suggested interpolating normal vectors
- We interpolate the normal vector along the scanline
- Compute illumination model for every pixel

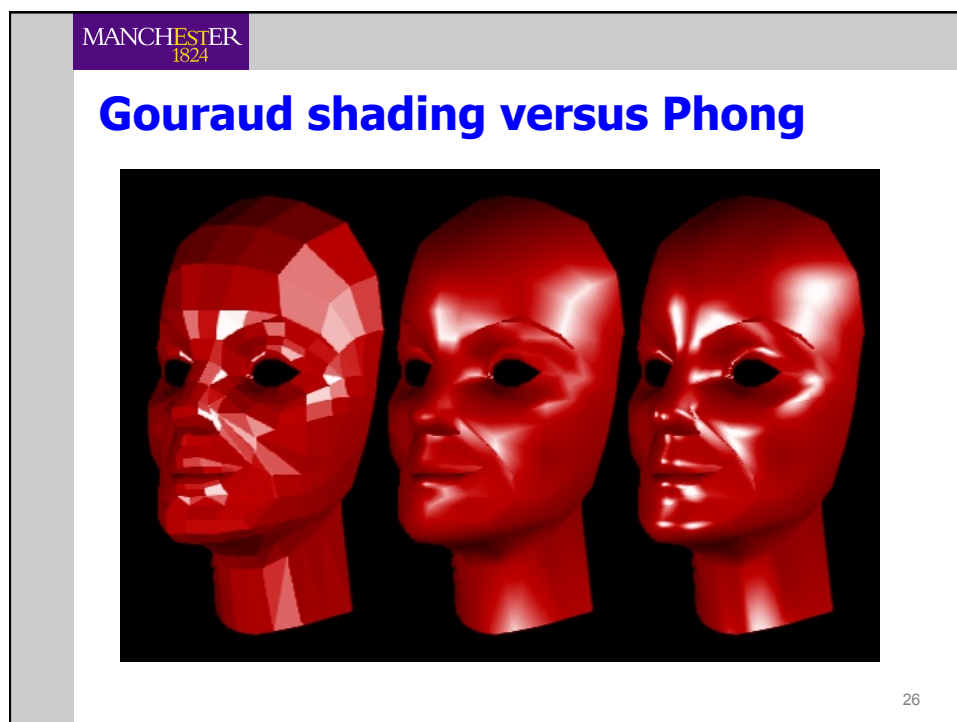
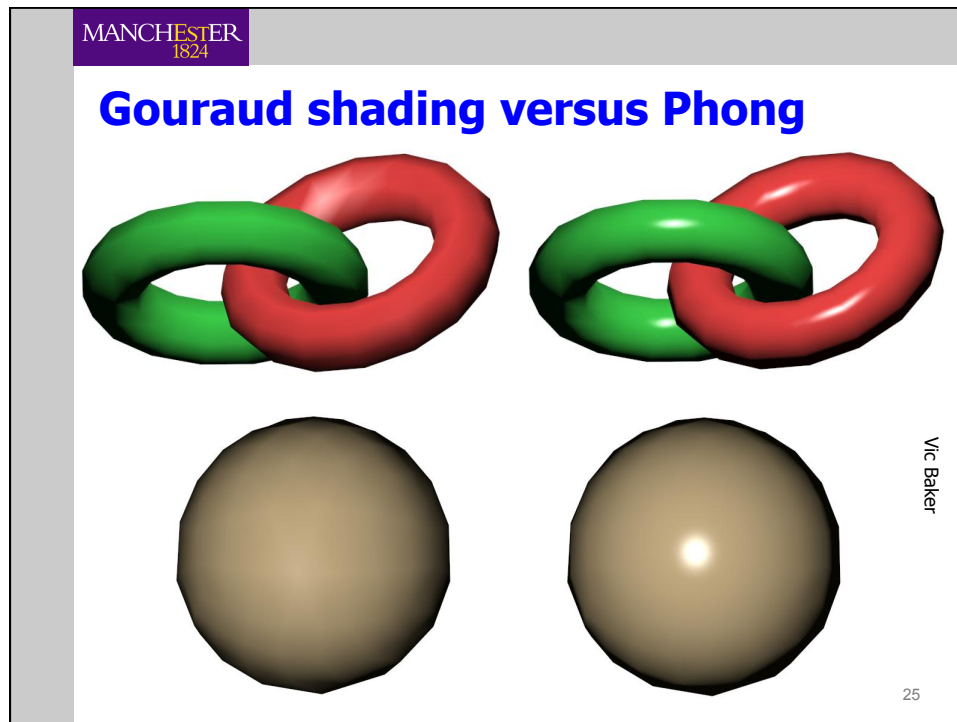
23

Phong algorithm

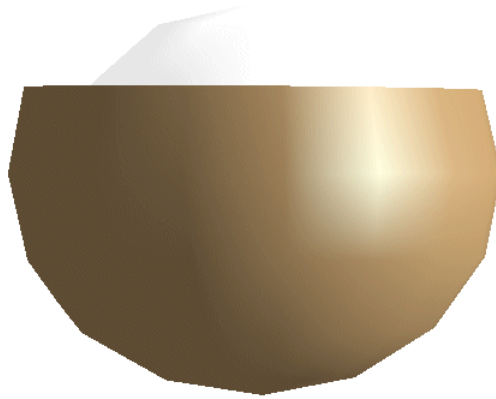
- for each scanline {
 - compute average normal \mathbf{N}_{Left} from \mathbf{N}_A and \mathbf{N}_C
 - compute average normal $\mathbf{N}_{\text{Right}}$ from \mathbf{N}_B and \mathbf{N}_C
 - average between \mathbf{N}_{Left} and $\mathbf{N}_{\text{Right}}$ along the scanline, and compute colour C using illumination model
- }



24



Gouraud shading versus Phong

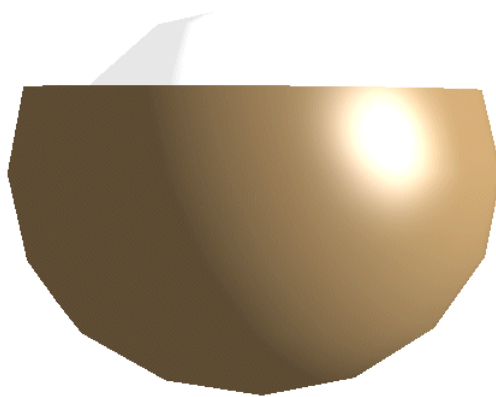


Alan Watt

- Gouraud-shaded, highlight distorted

27

Phong shading



Alan Watt

- Phong-shaded, highlight now correct

28

Rendering expense

- Roughly, our local illumination model takes about **60** floating-point operations to compute a colour for a pixel
- For a Gouraud-shaded triangle, that's **180** flops, then about **2** per pixel
- For a Phong-shaded triangle, that's **60 flops for every pixel**

29