MANCHESTER 1824

COMP27112

Computer Graphics and Image Processing

8: Rendering (2)

Toby.Howard@manchester.ac.uk

1

MANCHESTER 1824

Recap

- 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 for a triangle mesh?

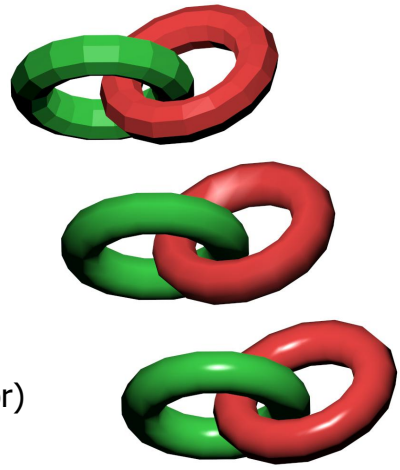
2

MANCHESTER 1824

Shading a surface

- We'll look at three shading methods:

1. Flat
   (aka constant)
2. Gouraud
   (aka intensity)
3. Phong
   (aka normal-vector)

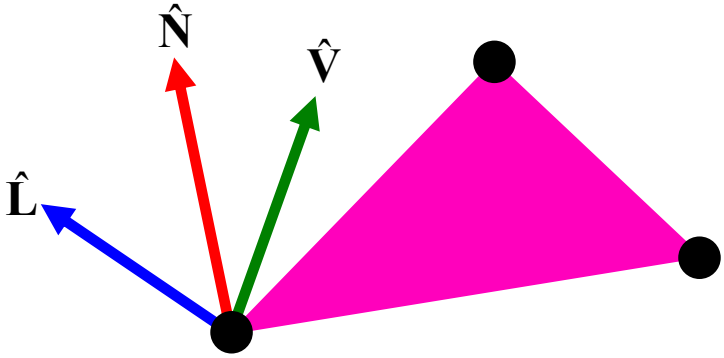


Vic Baker

3

MANCHESTER 1824

Flat (aka constant) shading



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

4

MANCHESTER 1824

Flat (aka constant) shading

Henri Gouraud

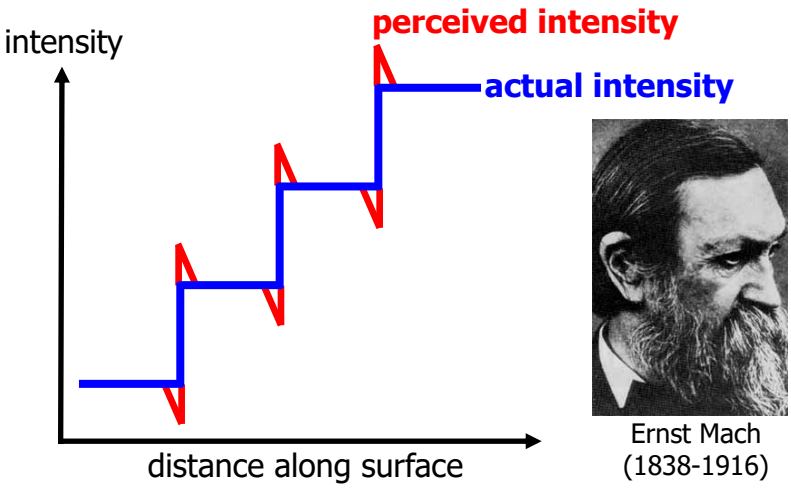
- Each polygon is uniformly coloured according to its orientation
- We clearly see the mesh
- This is made worse by the Mach Band effect

5

MANCHESTER 1824

Mach banding

- Our eyes are very good at finding edges

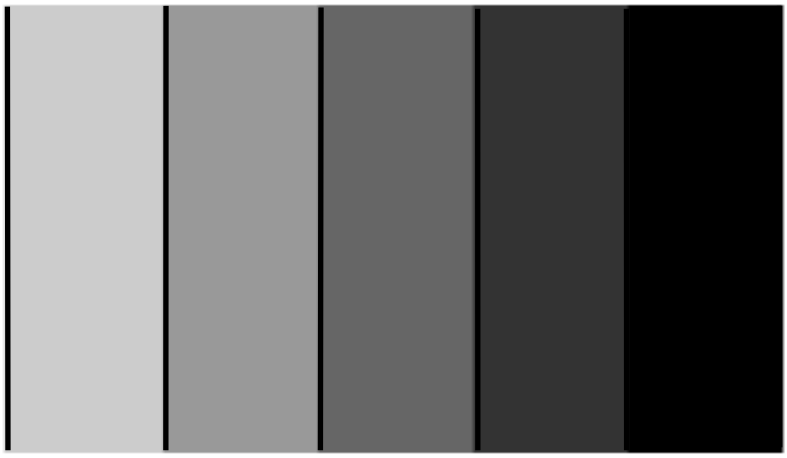


Ernst Mach
(1838-1916)

6

MANCHESTER 1824

Mach banding



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

7

MANCHESTER 1824

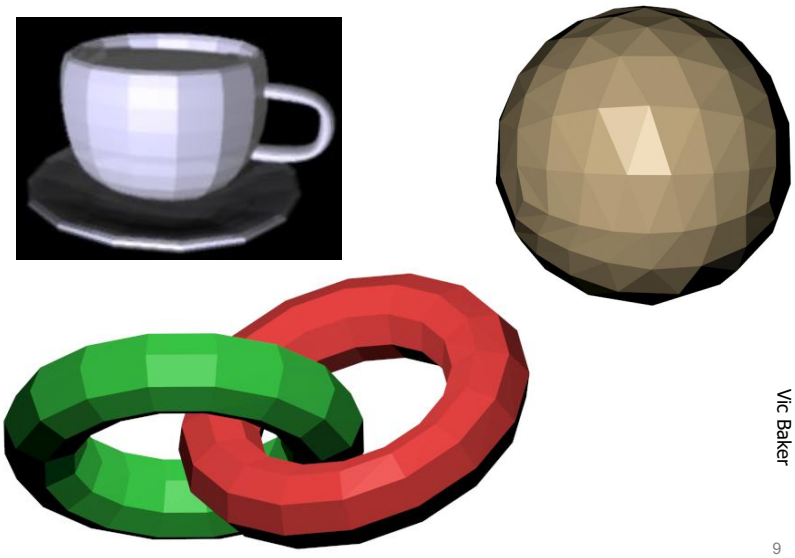
Mach banding

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

8

MANCHESTER 1824

Examples of flat shading



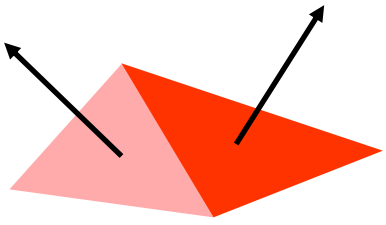
Vic Baker

9

MANCHESTER 1824

Gouraud (aka intensity) shading

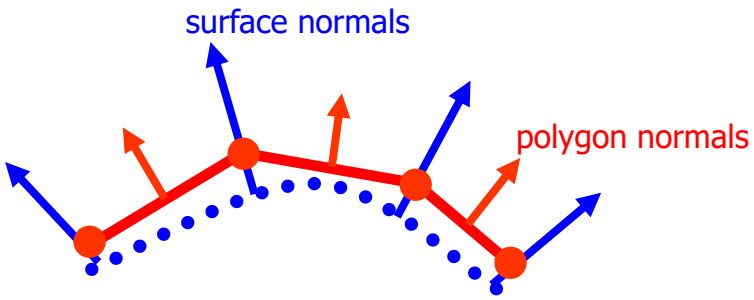
- Invented by Henri Gouraud in 1971
- Gouraud shading uses interpolation, to smooth out the discontinuities between polygons



10

MANCHESTER 1824

Approximating a surface

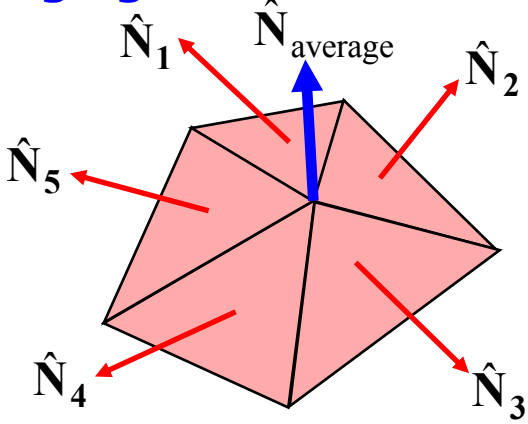


- We can approximate the normals of the underlying "surface" the polygons are modelling
- by averaging the normals where polygons share vertices

11

MANCHESTER 1824

Averaging the vertex normals



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

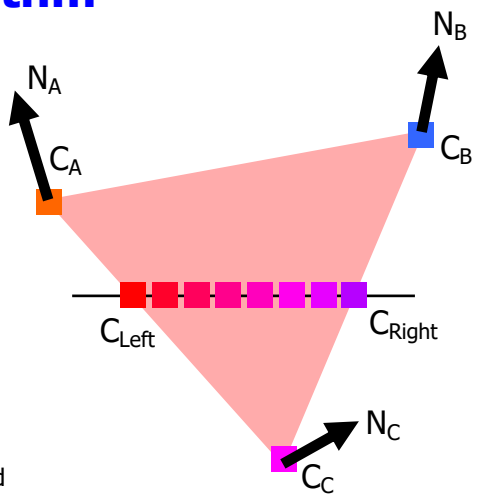
12

MANCHESTER 1824

Gouraud algorithm

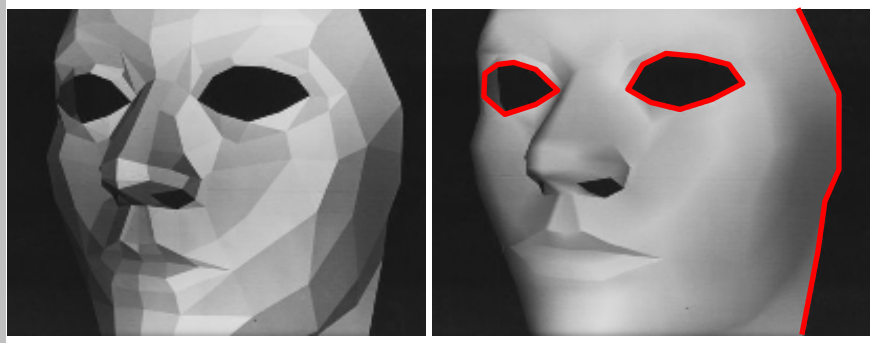
- compute average vertex normals at A, B and C
- compute pixel colours $\mathbf{C_A}$, $\mathbf{C_B}$, $\mathbf{C_C}$
- for each scanline {
  - average colour $\mathbf{C_{Left}}$ from $\mathbf{C_A}$ and $\mathbf{C_C}$
  - average colour $\mathbf{C_{Right}}$ from $\mathbf{C_B}$ and $\mathbf{C_C}$
  -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

  }



MANCHESTER 1824

Gouraud results



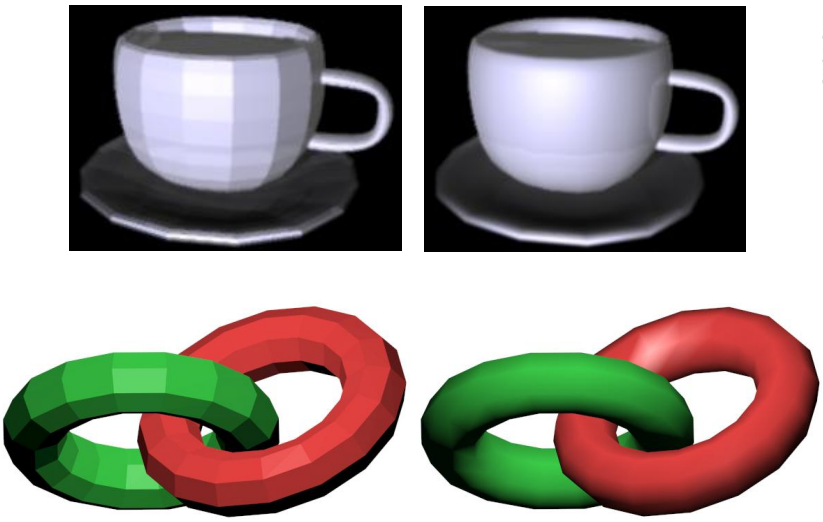
Henri Gouraud

flat shading **Gouraud shading**

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

MANCHESTER 1824

Flat shading versus Gouraud

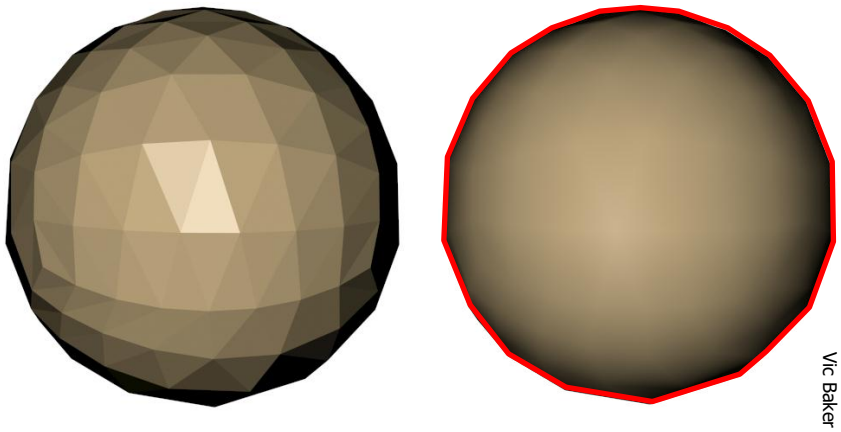


Vic Baker

15

MANCHESTER 1824

Flat shading versus Gouraud



Vic Baker

16

MANCHESTER 1824

Implementing Gouraud

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

```
deltaCol= (C_Right - C_Left)/(X_Right- X_Left);
Col= C_Left;
for (x= X_Left; X <= X_Right; x++) {
    TestAndSetPixel(x, y, Col);
    Col= Col + deltaCol;
    }
```

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

17

MANCHESTER 1824

Gouraud shading: problems

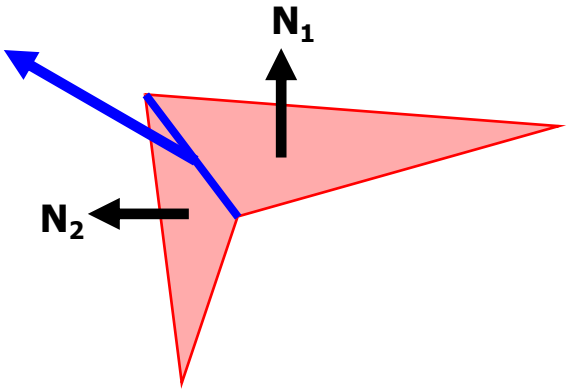
- While it's fast and efficient, the method has drawbacks:
  - Specular highlights may be distorted or averaged away altogether (because Gouraud shading averages between **vertex** colours)
  - Mach banding may still be visible

18

MANCHESTER 1824

Gouraud shading: problems

- Sometimes edges are "shaded away"

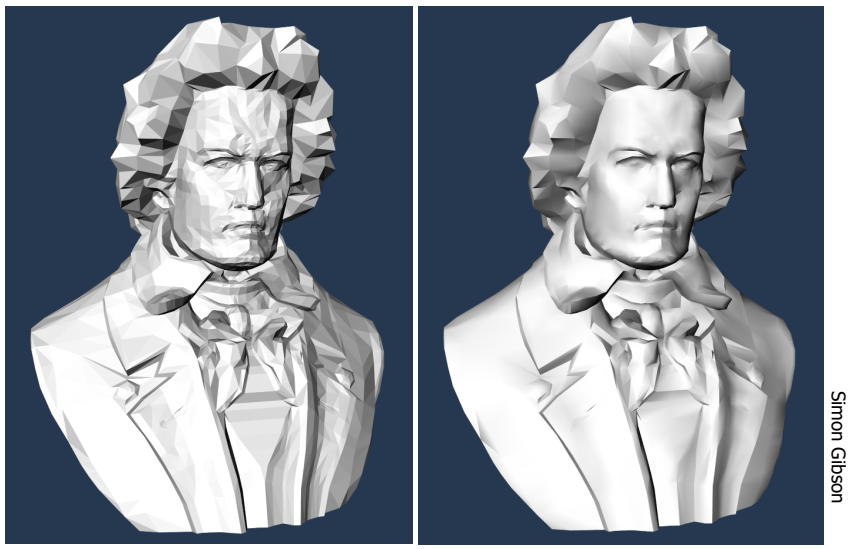


- Edge must be tagged in data structure to avoid interpolation across it

19

MANCHESTER 1824

Tagged edges to stop interpolation



Simon Gibson

20

MANCHESTER 1824

Phong (aka normal vector) interpolation

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

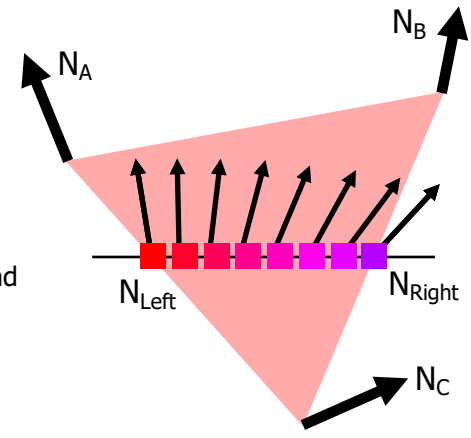
21

MANCHESTER 1824

Phong (aka normal vector) interpolation

- for each scanline {
  - compute average normal $\mathbf{N_{Left}}$ from $\mathbf{N_A}$ and $\mathbf{N_C}$
  - compute average normal $\mathbf{N_{Right}}$ from $\mathbf{N_B}$ and $\mathbf{N_C}$
  - average between $\mathbf{N_{Left}}$ and $\mathbf{N_{Right}}$ along the scanline, and compute colour C using illumination model

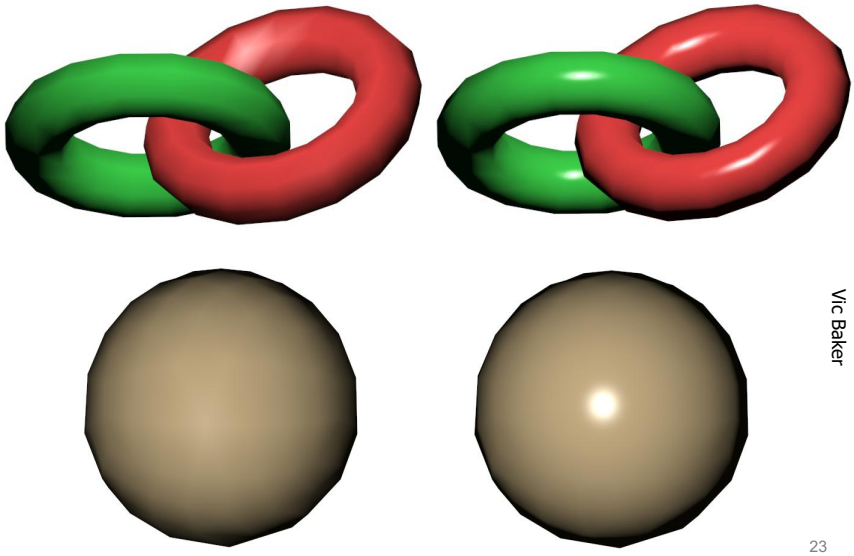
}



22

MANCHESTER 1824

Gouraud shading versus Phong



Vic Baker

23

MANCHESTER 1824

Comparison of shading methods

flat Gouraud Phong

24

MANCHESTER 1824

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

25