



Shading Interpolation

© 1995-2015 Josef Pelikán & Alexander Wilkie
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

<http://cgg.mff.cuni.cz/~pepca/>



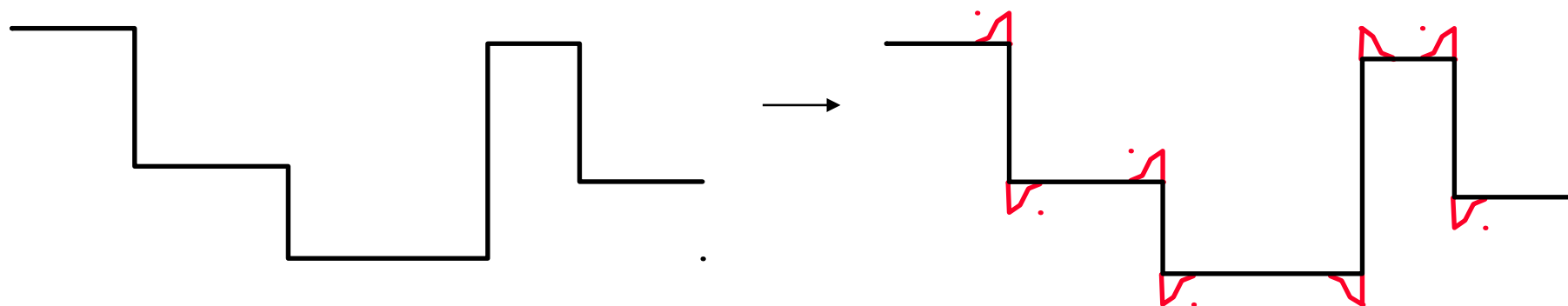
Shading Interpolation

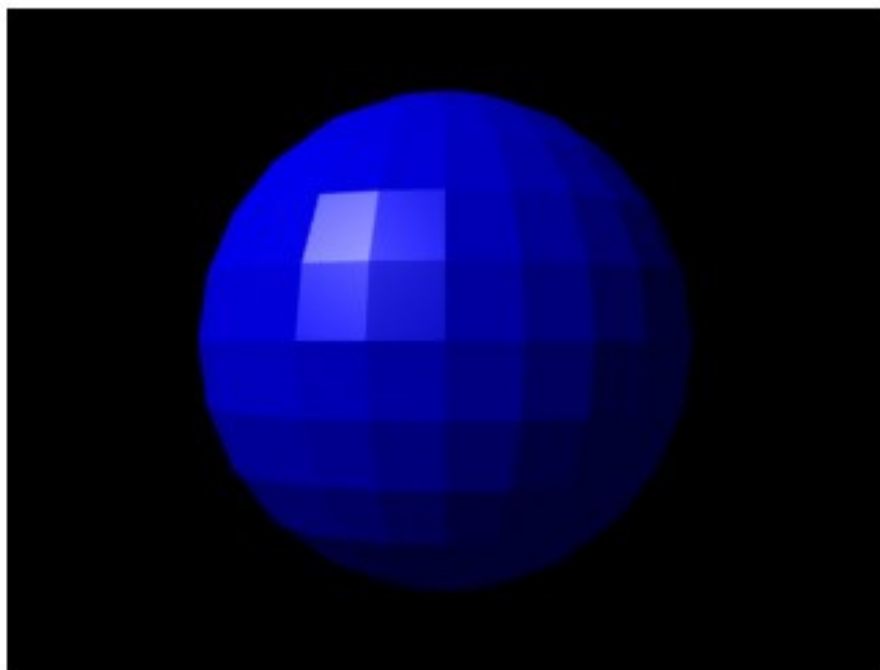
- ◆ Methods to **apply shading algorithms** when displaying surfaces (B-rep):
 - ➔ **Constant shading**
 - ➔ **Continuous shading**
 - Gouraud interpolation (colours)
 - Phong interpolation (normals)



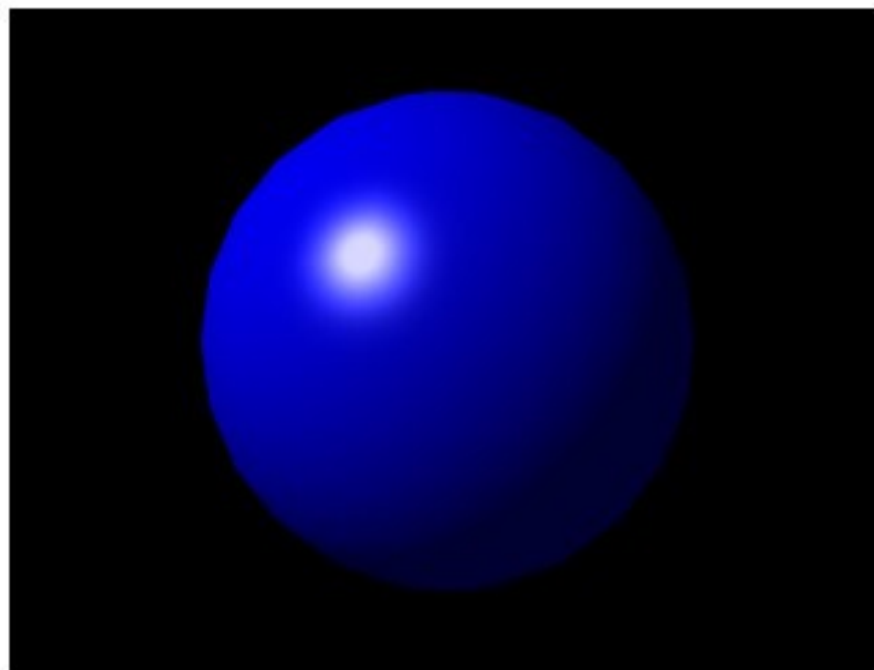
Constant Shading

- ◆ Compute **E** once for each polygon (e.g. in the center), and fill the polygon with **a single colour**
- ➔ Works well for **polygonal solids**
- ➔ **Curved surfaces** that are approximated via polygons:
 - Artificial edges of the solid are highlighted
 - This is made worse by the **Mach effect** (a human visual system feature)





FLAT SHADING



PHONG SHADING



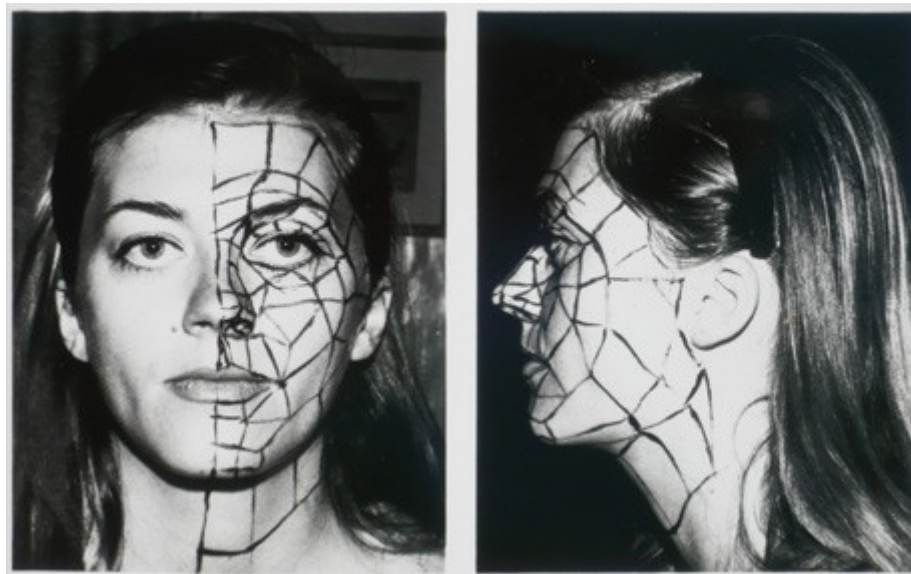
Continuous Shading

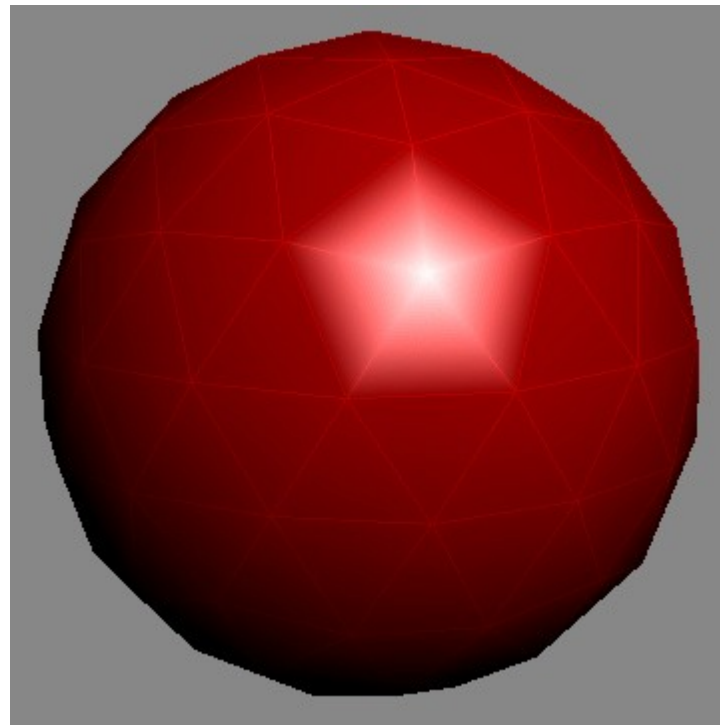
- ◆ Colour interpolation – **Gouraud shading**
 - Faster, more suited for diffuse surfaces
 - HW implementations available early on (Silicon Graphics)
- ◆ Normal interpolation – **Phong shading**
 - Slower, more accurate
 - Nowadays, also available in hardware - everywhere

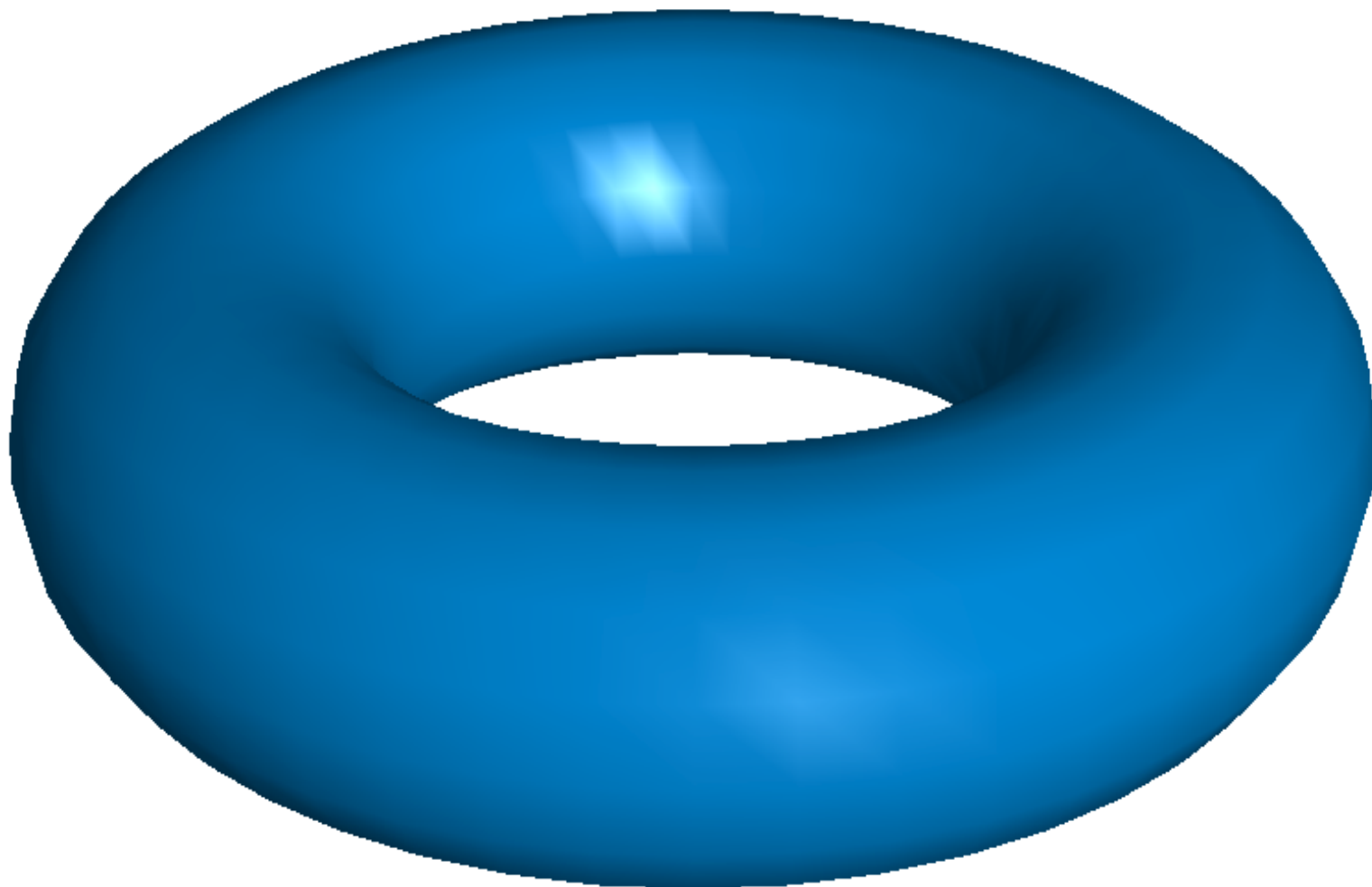


Gouraud Shading

- ◆ At **polygon vertices**, we calculate the normal vector, and the **colour** that results from shading
 - Any shading model can be used
- ◆ **Inside the polygon**, we only interpolate these colour values bi-linearly
 - This is done during polygon filling



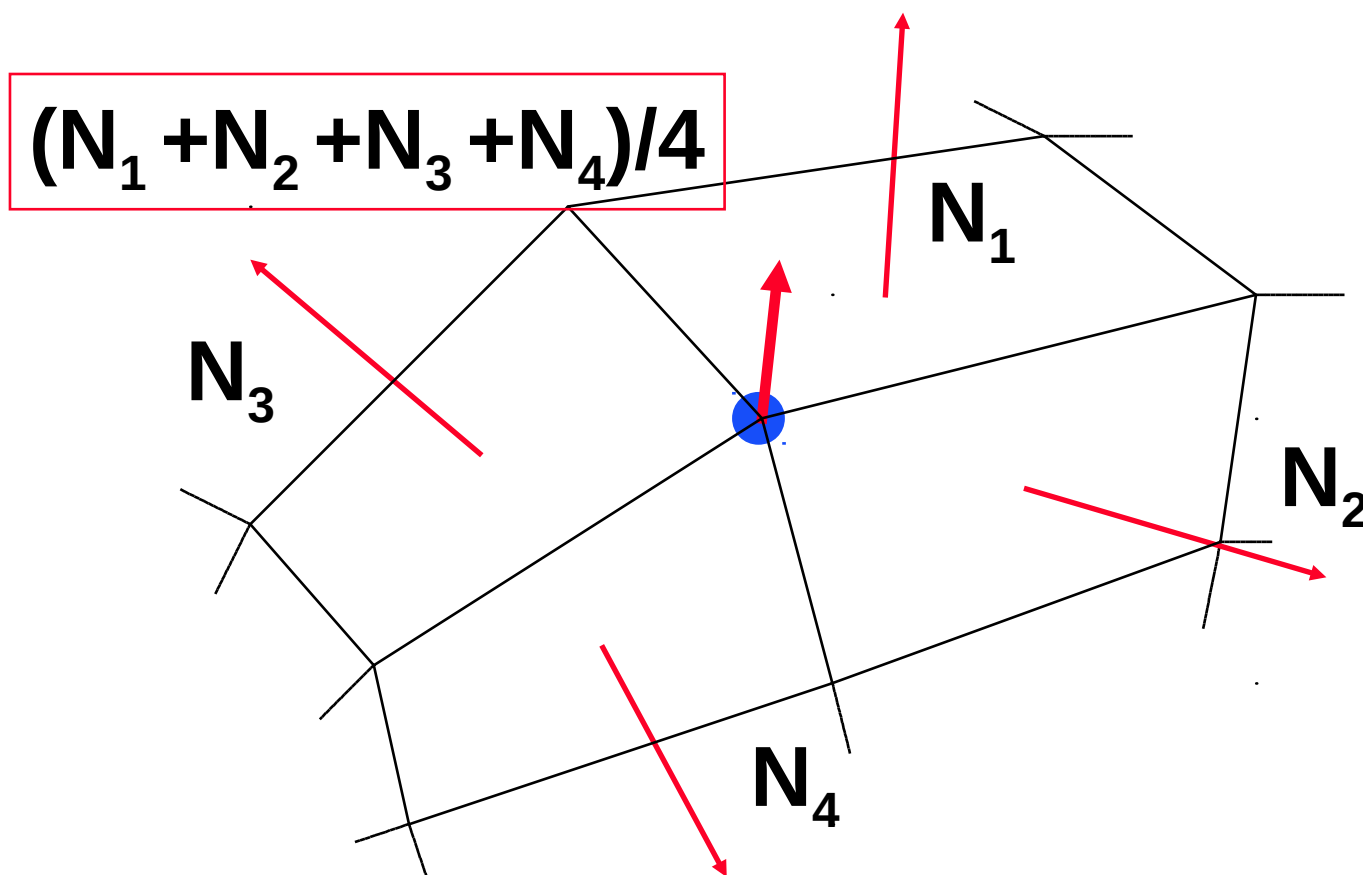






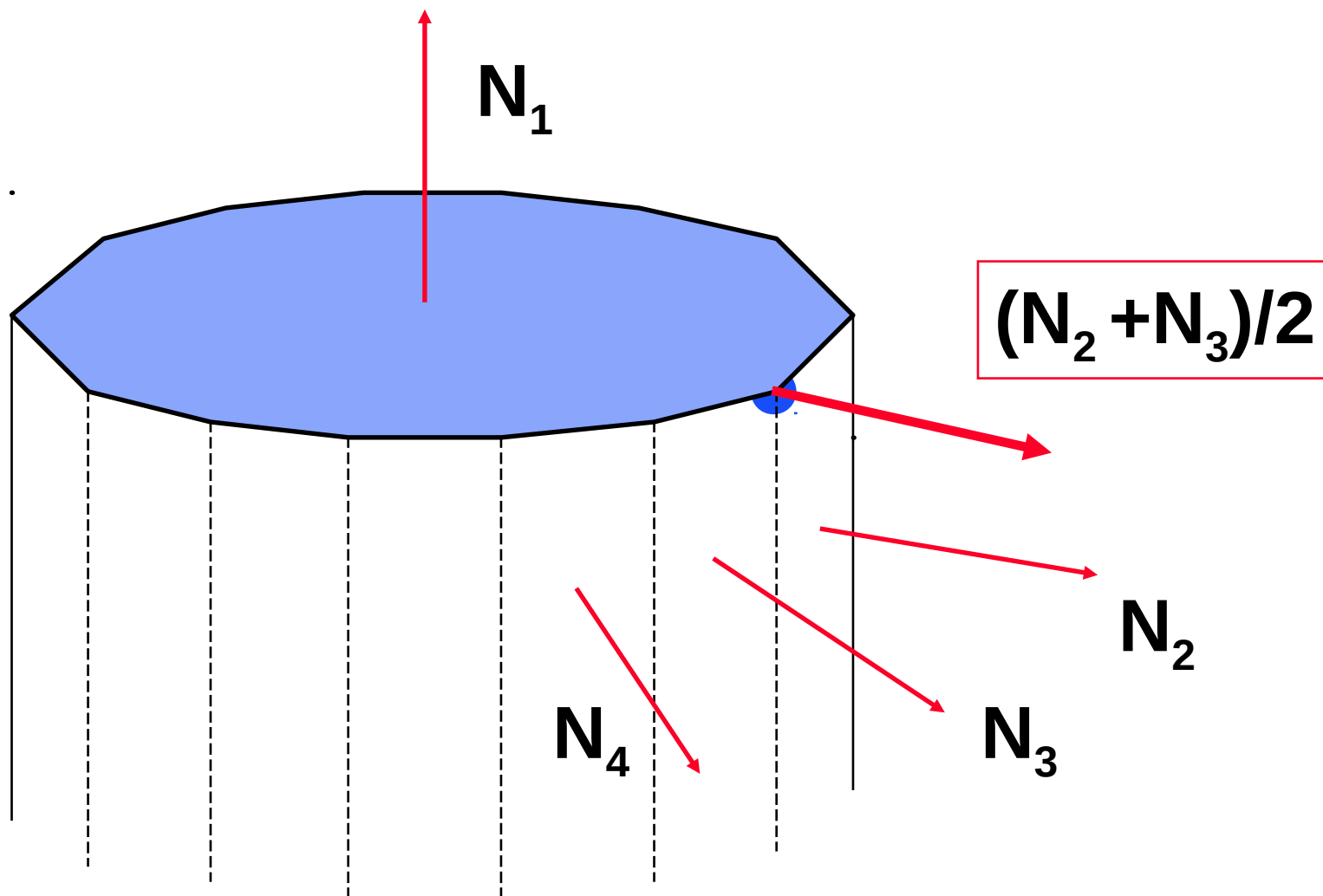
Calculation of Vertex Normals

- 1 **Analytically** – according to neighbour polygon area
- 2 **Aproximative** – normal average of neighbours:



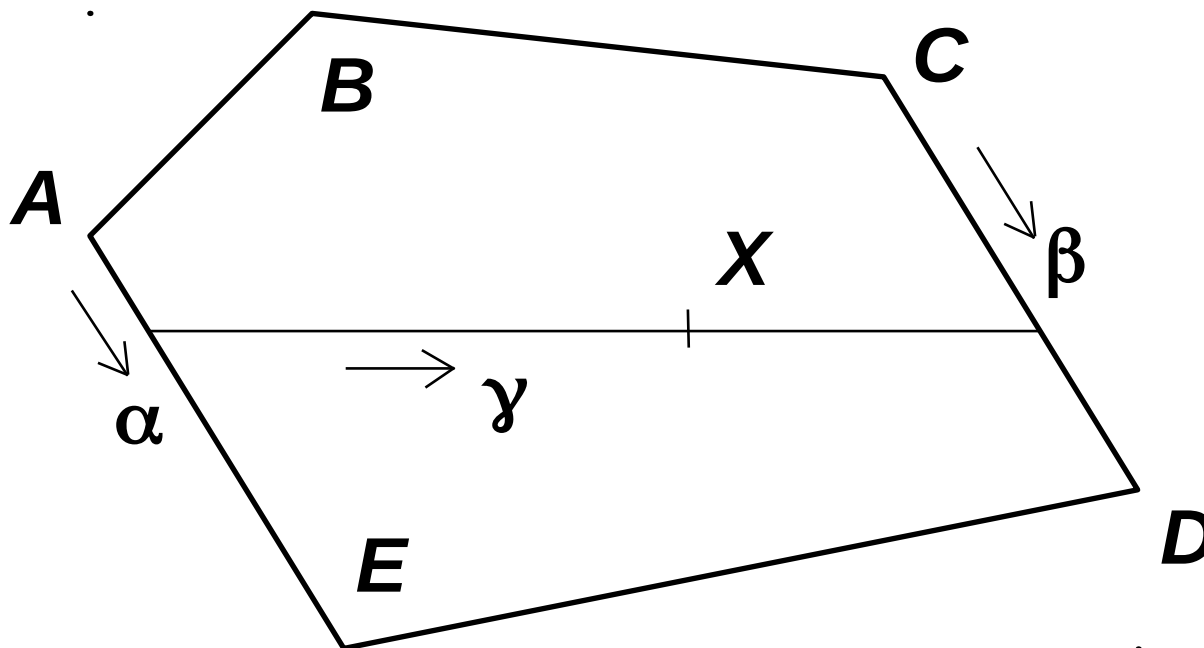


Real and Internal Edges





Bi-linear Interpolation

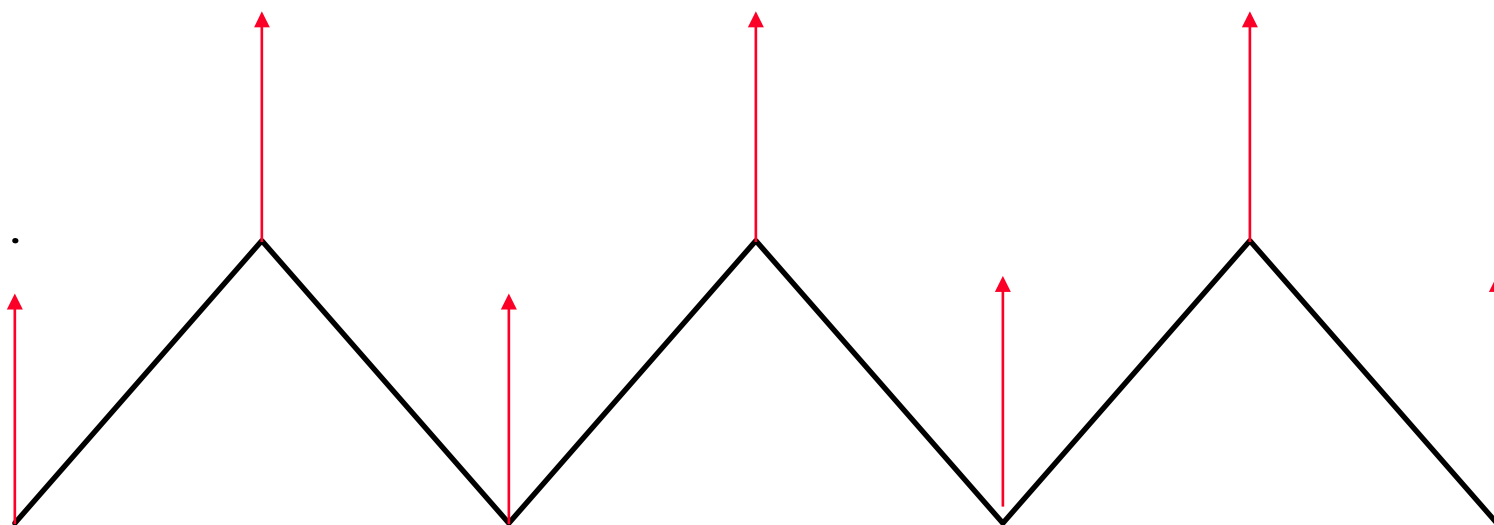


$$\mathbf{f}_X = (1-\gamma) \cdot [(1-\alpha) \cdot \mathbf{f}_A + \alpha \cdot \mathbf{f}_E] + \\ + \gamma \cdot [(1-\beta) \cdot \mathbf{f}_C + \beta \cdot \mathbf{f}_D]$$



Interpolation Issues (colour)

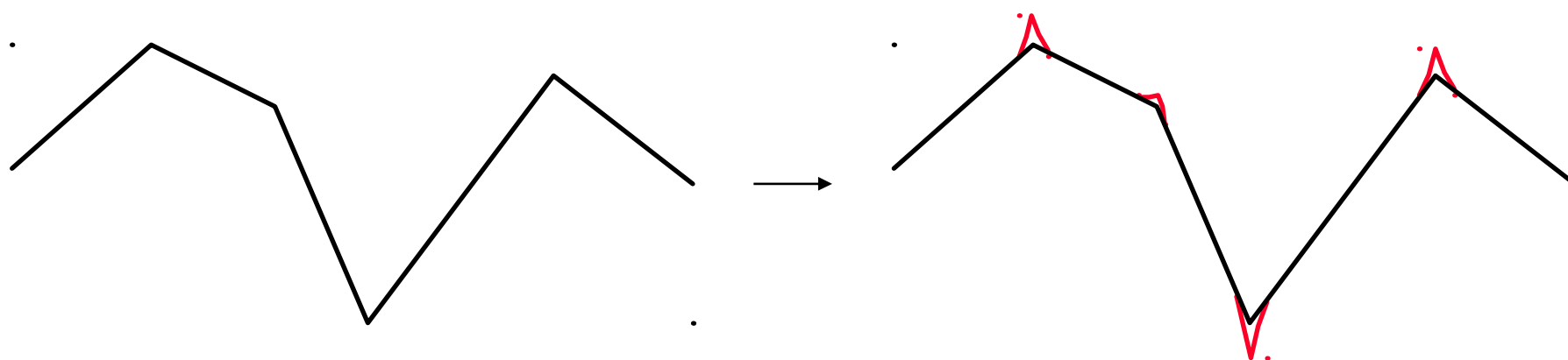
- ◆ Poorly captures **highlights** (particularly for mirror surfaces)
- ◆ Not **rotationally invariant!**
- ◆ Possibility of **badly calculated normals!**





The Mach Effect (1865)

- ◆ Highlight discontinuities due to **intensity** – or its **derivation!**
- ◆ This is caused by **lateral inhibition** of the photoreceptors in our retina
 - Excited cells lower the sensitivity of neighbouring cells



Phong Shading (Interpolation?)



- ◆ At the polygon vertices, we determine the correct (interpolated?) normal vectors
- ◆ Inside the polygon, we interpolate the normal vectors via bi-linear interpolation for each pixel
 - This is done in parallel with polygon filling
- ◆ For **each internal pixel** we compute the shading (colour)
 - According to the chosen shading model



Larger Computational Cost

- ◆ Normals are computed for each pixel
 - Bi-linear interpolation and normalisation – this needs a square root calculation
 - There exist approximate interpolations w/o square root
- ◆ In each pixel, we compute the shading model
 - Scalar product, square of floats, division

End



Further Information:

- **J. Foley, A. van Dam, S. Feiner, J. Hughes:**
Computer Graphics, Principles and Practice, 734-741
- **Jiří Žára a kol.:** *Počítačová grafika*, principy a algoritmy, 355-361