Illumination and shading (2)

Adding "reality" to images
- Shading algorithms
- Flat
- Gouraud
- Phong

Algorithms for shading of surfaces

- Shading model so far showed how to compute reflectance for individual points on a surface
- Shading varies across surfaces
- Point-by-point computation very expensive
- Three approaches for computing shading for polygonal surfaces
  - Flat shading
  - Gouraud shading
  - Phong shading

Flat shading

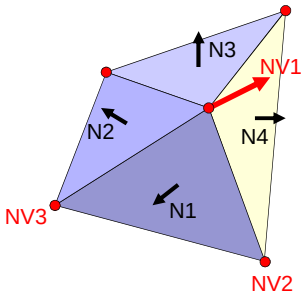
- One reflectance value per polygon surface
- Advantages
  - Computationally simple
- Drawbacks
  - Not very realistic for curved surfaces
  - Polygon structure visibly obvious

Gouraud shading

- Interpolates the reflectance (colour) across the surface of each polygon from a subset of points for which reflectance has been computed from the model
- Advantages
  - Visually more realistic than flat shading
- Drawbacks
  - Some artifacts may still be visible
  - Computationally more expensive than flat shading

Gouraud shading

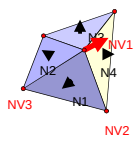
- Input
  - Surface normals
- Algorithm
  1. For each surface calculate vertex normals by interpolating surface normals

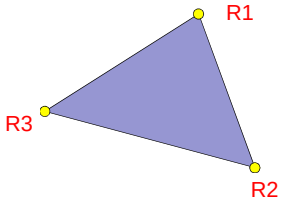


Gouraud shading

- Algorithm
  2. Compute reflectance at all the vertices

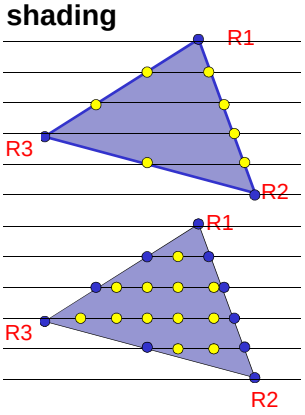
$$I = K_a I_a + K_d I_d \cos\theta_d + K_s I_s (\cos\theta_s)^n$$





Gouraud shading

- Algorithm
  3. Compute reflectance for points on the edges for every scan-line by interpolating vertex intensities
  4. Compute reflectances for every point within the patch of surface by interpolating along each scan-line between edge reflectances

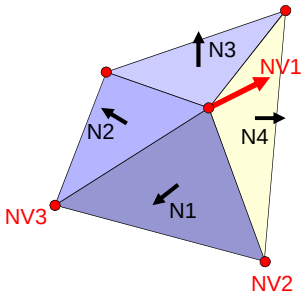


Phong shading

- Interpolates the surface normals across the surface of each polygon from a subset of points for which the normal has been computed from the model
- Advantages
  - Visually more realistic than Gouraud shading
- Drawbacks
  - Computationally more expensive than Gouraud shading

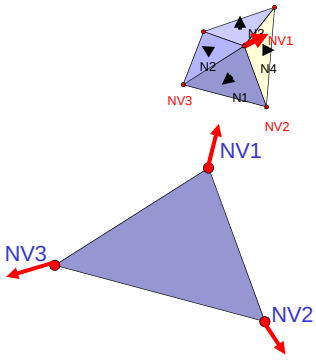
Phong shading

- Input
  - Surface normals
- Algorithm
  1. For each surface calculate vertex normals by interpolating surface normals



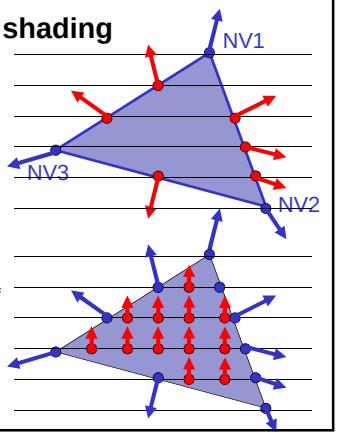
Phong shading

- Algorithm
  1. For each surface calculate vertex normals by interpolating surface normals



Phong shading

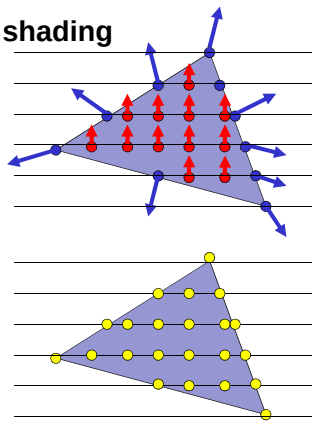
- Algorithm
  3. Compute normal vectors for points on the edges for every scan-line by interpolating vertex normals
  4. Compute normal vectors for every point within the patch of surface by interpolating along each scan-line between edge normal vectors

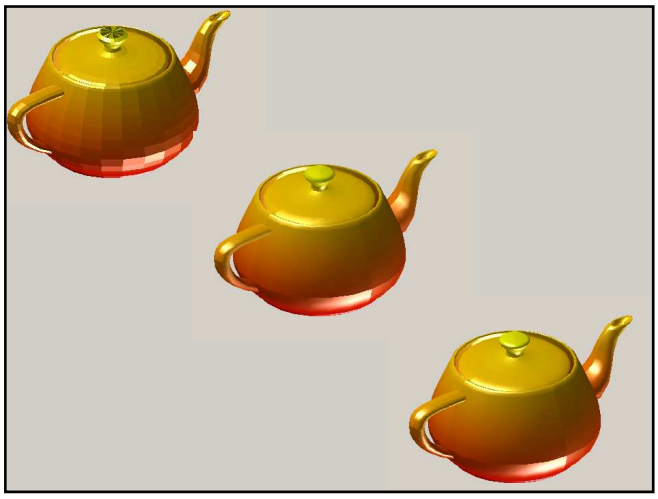


Phong shading

- Algorithm
  5. Compute surface reflectance directly from normal vectors for every point within the patch of surface

$$I = K_a I_a + K_d I_d \cos\theta_d + K_s I_s (\cos\theta_s)^n$$





Technical issues

- Normal vectors must be normalised (i.e. length always set to 1) (WHY?)
- After interpolation vectors must be re-normalised
- Transforming normal vectors (e.g. after change of the view or after changing position of an object) is not straighforward
- see e.g. http://groups.csail.mit.edu/graphics/classes/6.837/F01/Lecture15/lecture15.ppt

Credits

- This presentation has used slides from various web sources, including:
  - www.classes.cec.wustl.edu/~cse452/lectures/lect11_Illumination_2pp.pdf
  - http://www1.cs.columbia.edu/~cs4160/slides/lecture15.ppt#767,2,Rendering: 1960s (visibility)
  - groups.csail.mit.edu/graphics/classes/6.837/F01/Lecture15/lecture15.ppt
  - http://artis.imag.fr/~Nicolas.Holzschuch/cours/class9.pdf

Homework

- A surface is of a uniform red colour.

  Given two vertices at
  - V1 = [-80 00 58]
  - V2 = [-65 -47 58]

  their vertex normals
  - N1 = [-0.80 -0.04 0.60]
  - N2 = [-0.65 -0.50 0.60]

  a vector specifying the direction of light
  - [-0.30 -2.20 2.80]

  and light colour vector
  - [1 0.5 0.5]

  compute the colours (RGB vectors) of the 10 points lying on the line joining the two vertices V1 and V2

Next lecture

Colour