CHAPTER 10

Based on material by Werner Purgathofer

Illumination Models and Surface-Rendering Methods

Illumination Models

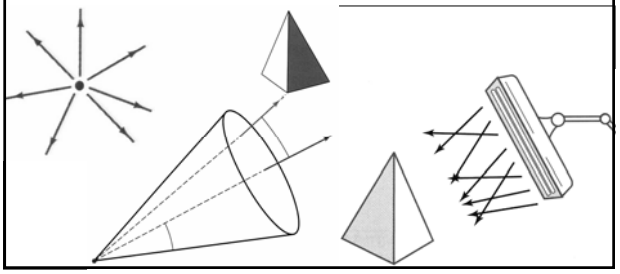
- light sources
- basic models
- effects
  - transparency
  - atmospheric effects
  - …

Surface-Rendering Methods

- polygon rendering methods
- environment mapping
- texture mapping
- bump mapping

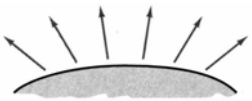
Light Sources

- point light source
- directional point light source
- distributed light source (“area light source”)

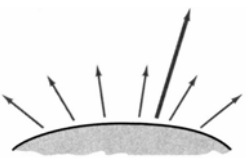


Surface Lighting Effects

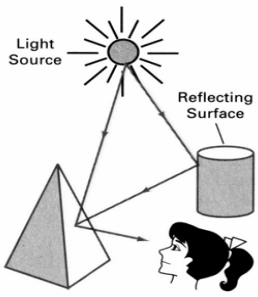
diffuse reflection



specular reflection



reflections from other surfaces



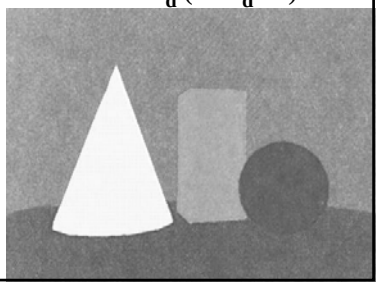
Basic Illumination Models

- empirical models
- lighting calculations
  - surface properties (glossy, matte, opaque,…)
  - background lighting conditions
  - light-source specification
  - reflection, absorption
- ambient light (background light) I_a
  - approximation of global diffuse lighting effects

Ambient Light Reflection

- constant over a surface
- independent of viewing direction
- diffuse-reflection coefficient k_d $(0 \le k_d \le 1)$

$$I_{ambdiff} = k_d I_a$$



Illumination and Shading

- shaded surfaces generate a spatial impression

the flatter light falls on a surface, the darker it will appear

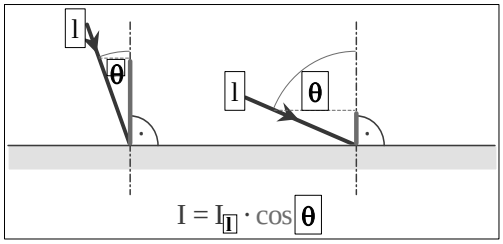
- therefore:

we need the incident light direction

or

the position of the (point) light source

Lambert's Law



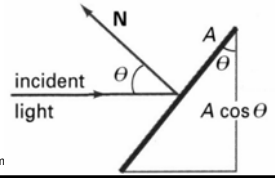
when considering the material:

$$\mathbf{I = k_d \cdot I_l \cdot \cos\theta}$$

Lambertian (Diffuse) Reflection

- ideal diffuse reflectors (Lambertian reflectors)
- brightness depends on orientation of surface

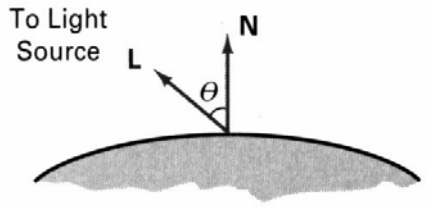
- Lambert's cosine law



$$I_{l,diff} = k_d I_l \cos\theta$$

Diffuse Reflection

- angle of incidence $\theta \in [0°,90°] \Rightarrow$ surface illuminated

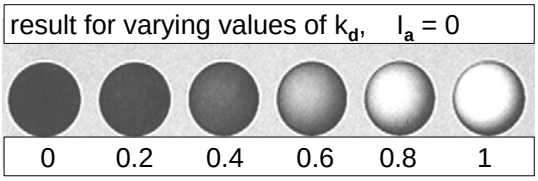


$$I_{l,diff} = k_d I_l (N \cdot L)$$

Diffuse Reflection Coefficient

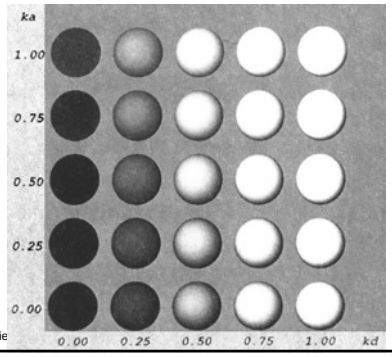
- varying k_d

result for varying values of k_d, $I_a = 0$



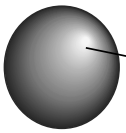
Ambient and Diffuse Reflection

- total diffuse reflection $I_{l,diff} = k_a I_a + k_d I_l (N \cdot L)$



(sometimes k_a for ambient light)

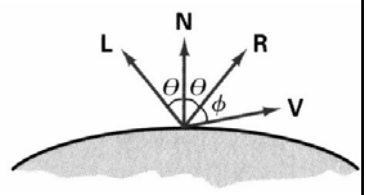
Specular Highlights



this area must be lighter than the shading model calculates, because the light source is reflected directly into the viewer's eye

Specular Reflection Model

- reflection of incident light around specular-reflection angle

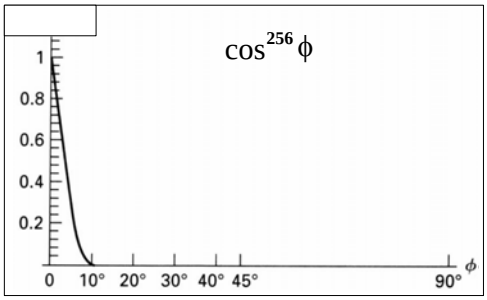


- empirical Phong model

$$I_{l,spec} = I_l \cos^{n_s} \phi$$

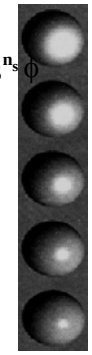
Specular Reflection Coefficient n_s

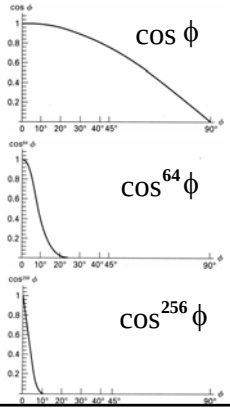
$\cos^{n_s} \phi$

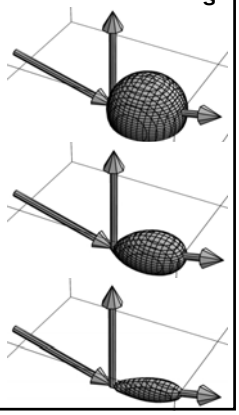


Specular Reflection Coefficient n_s

$\cos^{n_s} \phi$

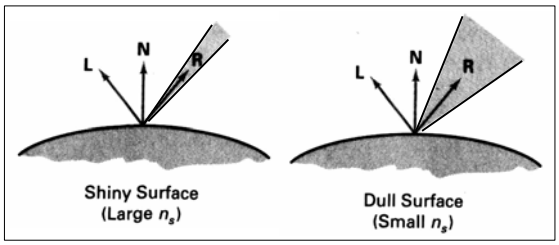






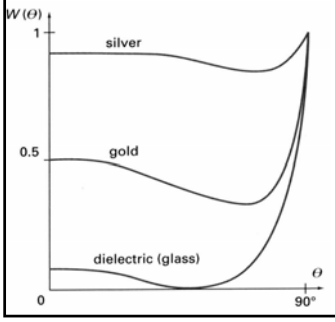
Specular Reflection Coefficient

- empirical Phong model $I_{l,spec} = I_l \cos^{n_s} \phi$
  - n_s large ⇒ shiny surface
  - n_s small ⇒ dull surface



Fresnel Specular Refl. Coefficient

- Fresnel's laws of Reflection
  - specular reflection coefficient $W(\theta)$

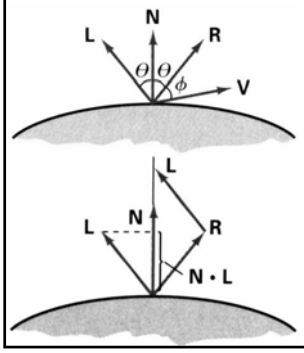


$$I_{l,spec} = W(\theta) I_l \cos^{n_s} \phi$$

specular reflection coefficient as a function of angle of incidence for different materials

Simple Specular Reflection

- $W(\theta) \approx$ constant for many opaque materials (k_s)



$$I_{l,spec} = k_s I_l (V \cdot R)^{n_s}$$

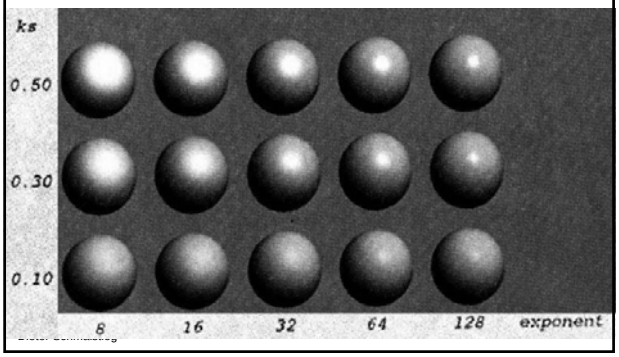
calculation of R:

$$R + L = (2N \cdot L)N$$

$$R = (2N \cdot L)N - L$$

Specular Reflection Results

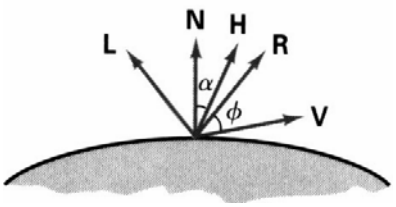
$$I_{l,spec} = k_s I_l (V \cdot R)^{n_s}$$



Simplified Specular Reflection

- simplified Phong model with halfway vector H

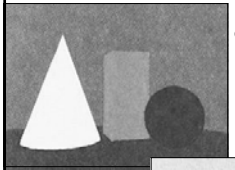
$$I_{spec} = k_s I_l (V \cdot R)^{n_s} \quad \rightarrow \quad I_{spec} = k_s I_l (N \cdot H)^{n_s}$$



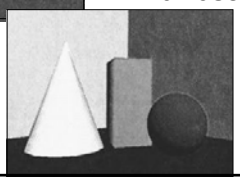
$$H = \frac{L+V}{|L+V|}$$

Diffuse and Specular Reflection

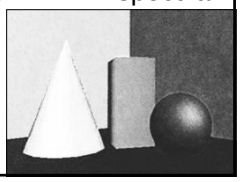
$$I = k_a I_a + \sum_{l=1}^{n} I_l \left[k_d (N \cdot L_l) + k_s (N \cdot H_l)^{n_s} \right]$$



ambient



ambient + diffuse



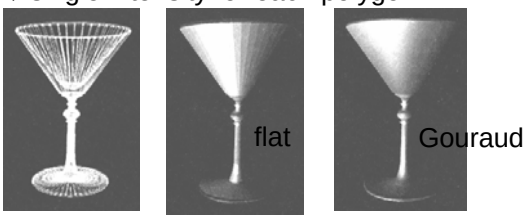
ambient + diffuse + specular

Other Aspects

- intensity attenuation with distance
- anisotropic light sources (Warn model)
- transparency (Snell's law)
- atmospheric effects
- shadows
- …

Polygon-Rendering Methods

- application of illumination model to polygon rendering
- constant-intensity shading (flat shading)
  - single intensity for each polygon

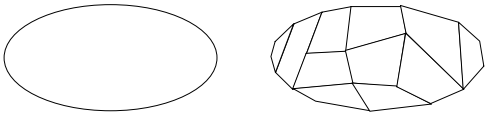


Polygon Shading: Interpolation

- the shading of a polygon is not constant, because it normally is only an approximation of the real surface ⇒ **interpolation**

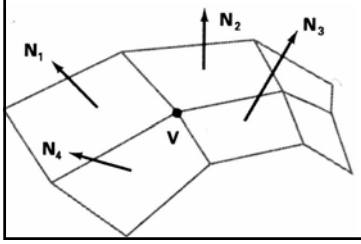
Gouraud shading: intensities

Phong shading: normal vectors



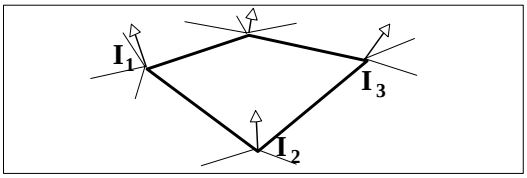
Gouraud Shading Overview

- intensity-interpolation
  - determine average unit normal vector at each polygon vertex
  - apply illumination model to each vertex
  - linearly interpolate vertex intensities



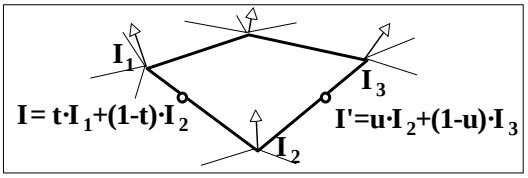
$$N_V = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

Gouraud Shading



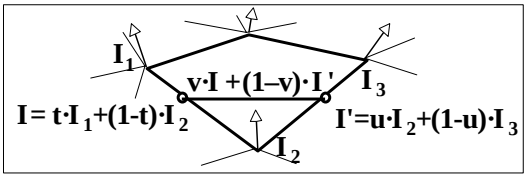
1. find normal vectors at corners and calculate shading (intensities) there: I_i

Gouraud Shading



1. find normal vectors at corners and calculate shading (intensities) there: I_i
2. interpolate intensities along the edges linearly: I, I'

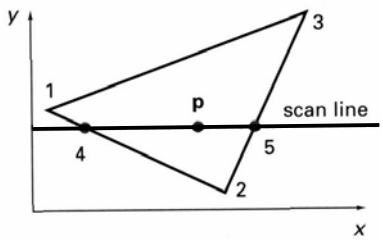
Gouraud Shading



1. find normal vectors at corners and calculate shading (intensities) there: I_i
2. interpolate intensities along the edges linearly: I, I'
3. interpolate intensities along scanlines linearly: I_p

Gouraud Shading

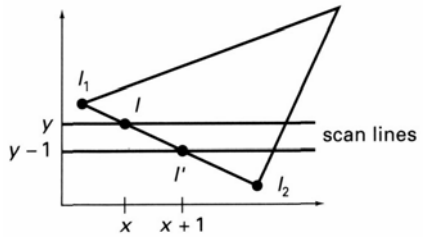
- interpolating intensities



$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2 \qquad I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Gouraud Shading

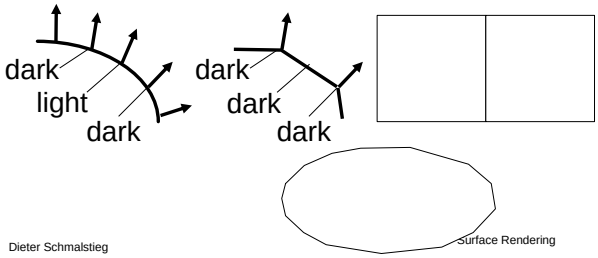
- incremental update



$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2 \qquad I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$

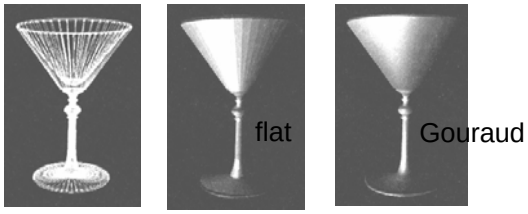
Problems of Gouraud Shading

- highlights can get lost or grow
- corners on silhouette remain
- Machband effect is visible at some edges



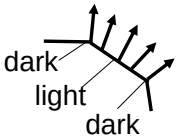
Gouraud Shading Results

- no intensity discontinuities
- Mach bands due to linear intensity interpolation
- problems with highlights



Phong Shading

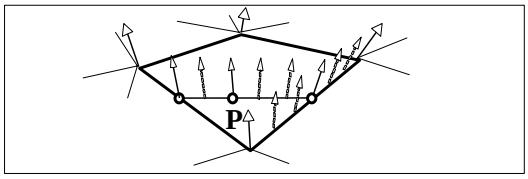
- instead of intensities the normal vectors are interpolated, and for every point the shading calculation is performed separately



Phong Shading Principle

- normal-vector interpolation
  - determine average unit normal vector at each polygon vertex
  - linearly interpolate vertex normals
  - apply illumination model along each scan line

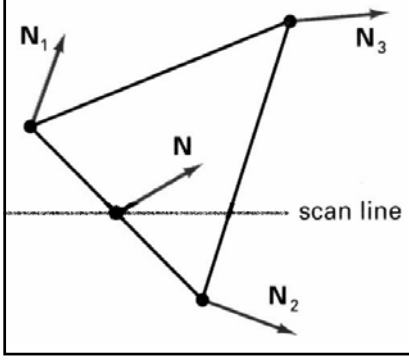
Phong Shading Overview



1. normal vectors at corner points
2. interpolate normal vectors along the edges
3. interpolate normal vectors along scanlines & calculate shading (intensities) for every pixel

Phong Shading Normal Vectors

- normal-vector interpolation



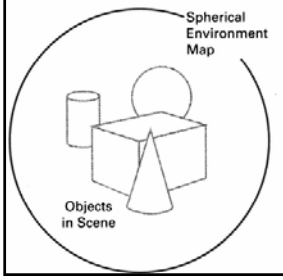
$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

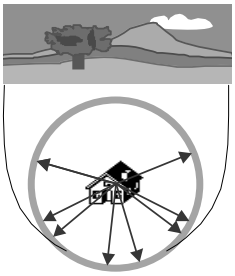
Phong Shading

- incremental normal vector update along and between scan lines
- comparison to Gouraud shading
  - better highlights
  - less Mach banding
  - more costly

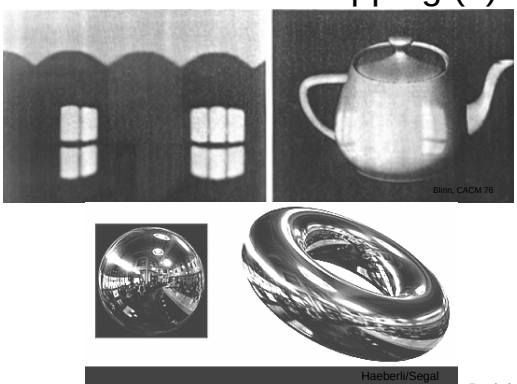
Environment Mapping (1)

- reflection mapping
- defined over the surface of an enclosing universe (sphere, cube, cylinder)



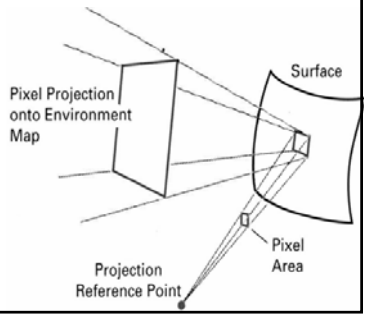


Environment Mapping (2)

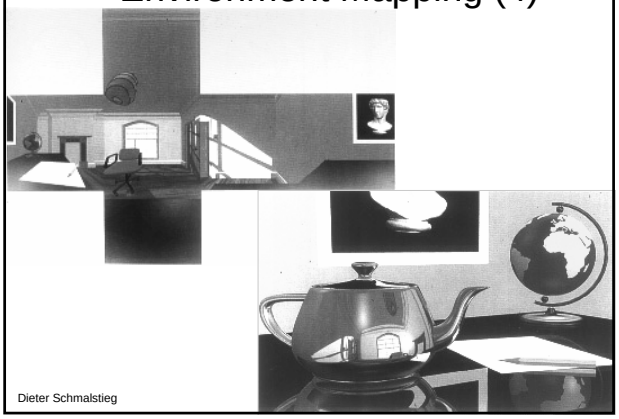


Environment Mapping (3)

- information in the environment map
  - intensity values for light sources
  - sky
  - background objects
- pixel area
  - projected onto surface
  - reflected onto environment map



Environment Mapping (4)



Dieter Schmalstieg

Adding Surface Detail (1)

- most objects do not have smooth surfaces
  - brick walls
  - gravel roads
  - shag carpets
- surface texture required

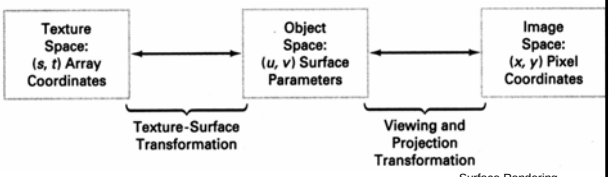
Adding Surface Detail (2)

- modeling surface detail with polygons
  - small polygon facets (e.g., checkerboard squares)
  - facets overlaid on surface polygon (parent)
  - parent surface used for visibility calculations
  - facets used for illumination calculations
  - impractical for intricate surface structure

Dieter Schmalstieg

Texture Mapping (1)

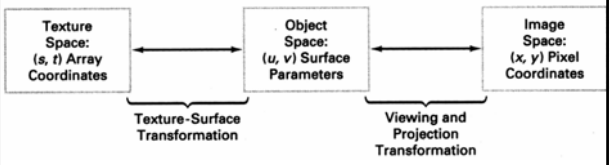
- texture patterns mapped onto surfaces
- texture pattern:
  - raster image
  - or procedure (modifies surface intensities)



Dieter Schmalstieg

Texture Mapping (2)

- texture mapping
  - texture scanning (s,t)→(x,y)
  - inverse scanning (x,y)→(s,t)



- texture-surface transformation

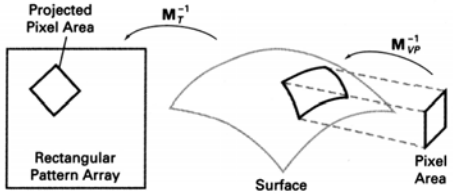
$$u = u(s,t) = a_u s + b_u t + c_u$$

$$v = v(s,t) = a_v s + b_v t + c_v$$

Dieter Schmalstieg

Texture Mapping (3)

- projecting pixel areas to texture space =
  = inverse scanning (x,y)→(s,t)



- calculation of M_{VP}^{-1} M_T^{-1}
- anti-aliasing with filter operations

Dieter Schmalstieg

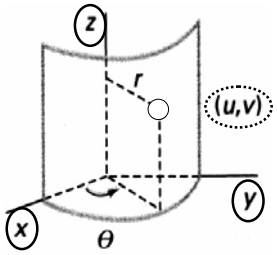
Texture-Map.: Cylindrical Surface

- M_{VP}

$u = \theta$, with $0 \le \theta \le \pi/2$

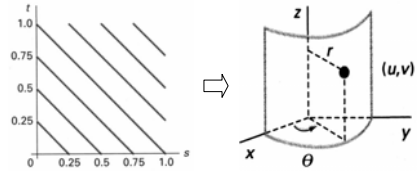
$v = z$ with $0 \le z \le 1$

$x = r \cdot \cos u$,
$y = r \cdot \sin u$,
$z = v$



Texture-Map.: Cylindrical Surface

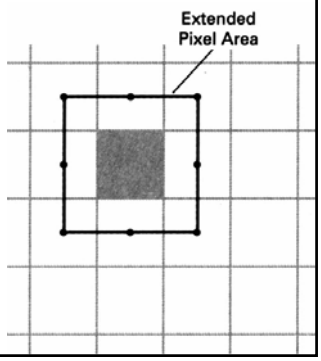
- M_T $u = s\,\pi/2, \quad v = t$



- M^{-1}_{VP}
  - pixel → surface point (x,y,z)
  - (x,y,z) → (u,v): $u = \tan^{-1}(y/x), \quad v = z$
- M^{-1}_{T} $\boxed{s = 2u/\pi, \quad t = v}$

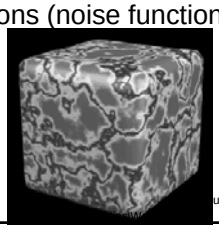
Texture Mapping: Anti-aliasing

- anti-aliasing with filter operations
  - project larger pixel area into texture space
  - mip-mapping
  - summed-area table method



Procedural Texturing (1)

- procedural texture definition
  - texture_function (x,y,z) returns intensity
  - avoid M_T
- 2D (surface texturing) or 3D (solid texturing)
- stochastic variations (noise function)
- examples
  - wood grains
  - marble
  - leaves



Procedural Texturing (2)

a scene with surface characteristics generated using solid-texture methods

Procedural Texturing (3)

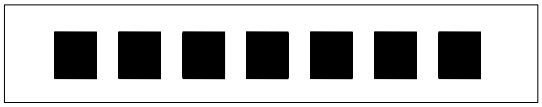
gem faces:
polygonal facets
rest: quadric & bicubic patches

surface texturing

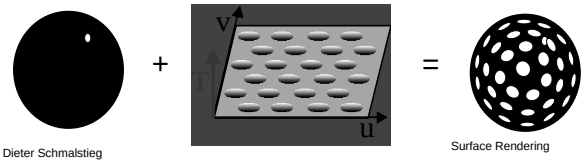
procedural texturing
for steamy jungle atmosphere

Bump Mapping (1)

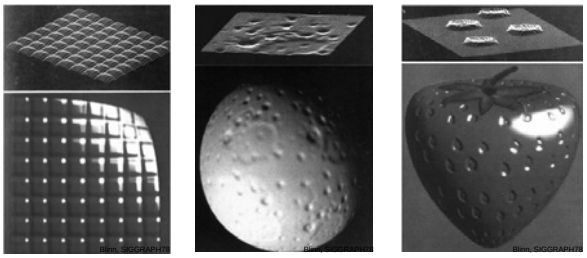
bumps are visible because of shading



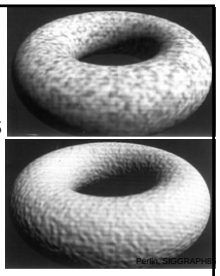
modeling of bumps is very costly.
trick: insert a detail structure T:



Bump Mapping Examples



Bump Mapping (2)



- surface roughness simulated
  - perturbation function varies surface normal locally
  - bump map b(u,v)

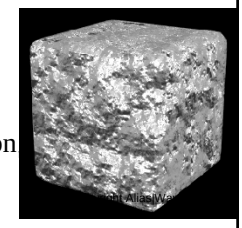
$P(u,v)$ surface point

$N = P_u \times P_v \qquad n = N/|N|$ surface normal

$P'(u,v) = P(u,v) + b(u,v)n$ modified surface point

Bump Mapping (3)

$$P'(u,v) = P(u,v) + b(u,v)n$$



$$N' = P_u' \times P_v'$$

$$P_u' = \frac{\partial}{\partial u}(P + bn) = P_u + b_u n + bn$$

$$P_u' \approx P_u + b_u n, \qquad P_v' \approx P_v + b_v n,$$

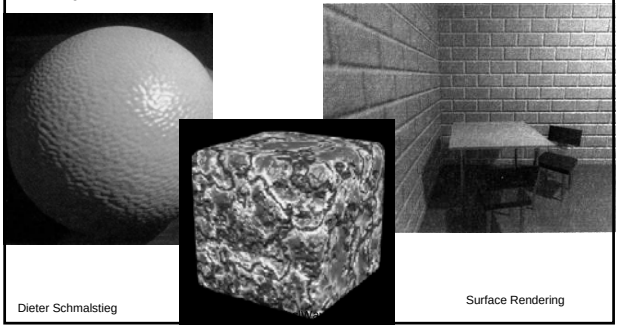
$$N' = P_u \times P_v + b_v(P_u \times n) + b_u(n \times P_v) + b_u b_v(n \times n)$$

$$n \times n = 0$$

$$\mathbf{N' = N + b_v(P_u \times n) + b_u(n \times P_v)}$$

Bump Mapping (4)

- bump map b(u,v) defined as raster image
- b_u, b_v: approximated with finite differences



Combination of Mapping Methods

armor surf
combinati
bump map
& environment mapping
& texture mapping