Computer Graphics

- Shading & Texturing -

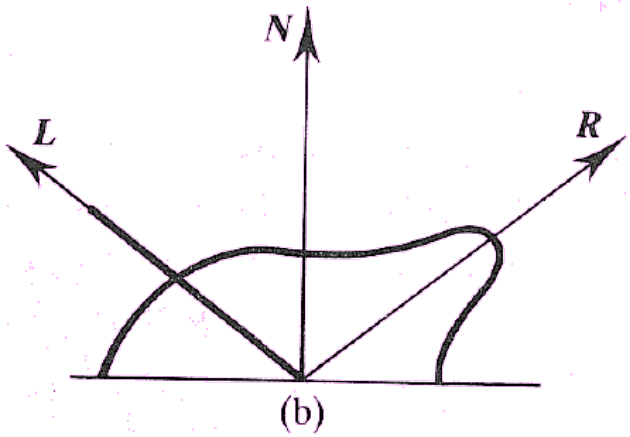
Philipp Slusallek

Overview

- **Last time**
  - Light Transport Equations
  - Bidirectional Reflectance Distribution Function (BRDF)
- **Today**
  - Shading (Cont.)
  - Texturing
- **Next lecture**
  - Procedural Shading

Empirical BRDF Approximation

- Purely heuristic model
  - Initially without units (values $\in [0,1]$)
- $L_r = L_{r,a} + L_{r,d} + L_{r,s}$ ($+ L_{r,m} + L_{r,t}$)
- $L_{r,a}$: Ambient term
  - Approximate indirect illumination
- $L_{r,d}$: Diffuse term (Lambert)
  - Uniform reflection
- $L_{r,s}$: Specular term
  - Mirror-reflection on a rough surface
- $L_{r,m}$: Perfect reflection
  - *Only possible with Ray-Tracing*
- $L_{r,t}$: Perfect transmission
  - *Only possible with Ray-Tracing*



Phong Illumination Model

- **Extended light sources: l point light sources**

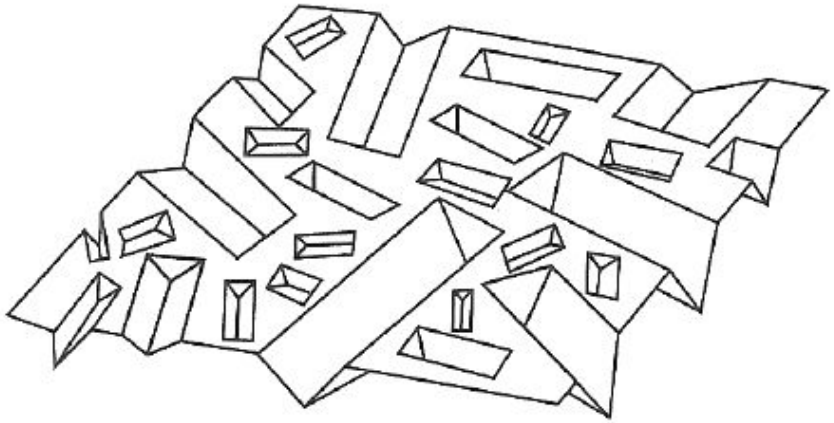
$$L_r = k_a L_{i,a} + k_d \sum_l L_l (I_l \cdot N) + k_s \sum_l L_l (R(I_l) \cdot V)^{k_e} \quad \text{(Phong)}$$

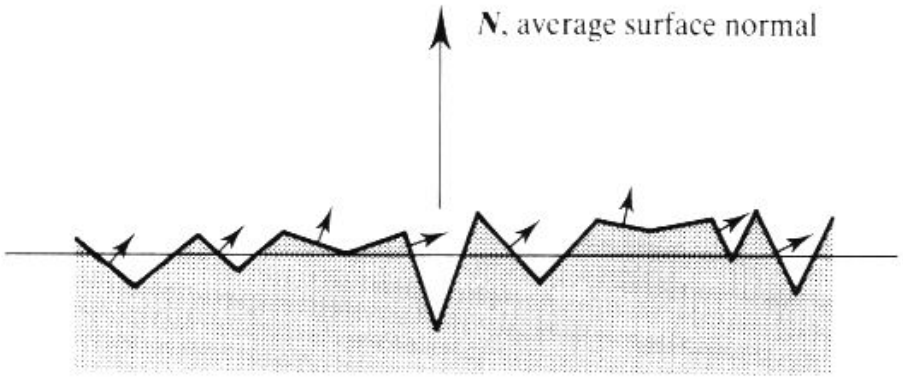
$$L_r = k_a L_{i,a} + k_d \sum_l L_l (I_l \cdot N) + k_s \sum_l L_l (H_l \cdot N)^{k_e} \quad \text{(Blinn)}$$

- **Color of specular reflection equal to light source**
- **Heuristic model**
  - Contradicts physics
  - Purely local illumination
    - Only direct light from the light sources
    - No further reflection on other surfaces
    - Constant ambient term
- **Often: light sources & viewer assumed to be far away**

Microfacet Model

- **Isotropic microfacet collection**
- **Microfacets assumed as perfectly smooth reflectors**
- **BRDF**
  - Distribution of microfacets
    - Often probabilistic distribution of orientation or V-groove assumption
  - Planar reflection properties
  - Self-masking, shadowing





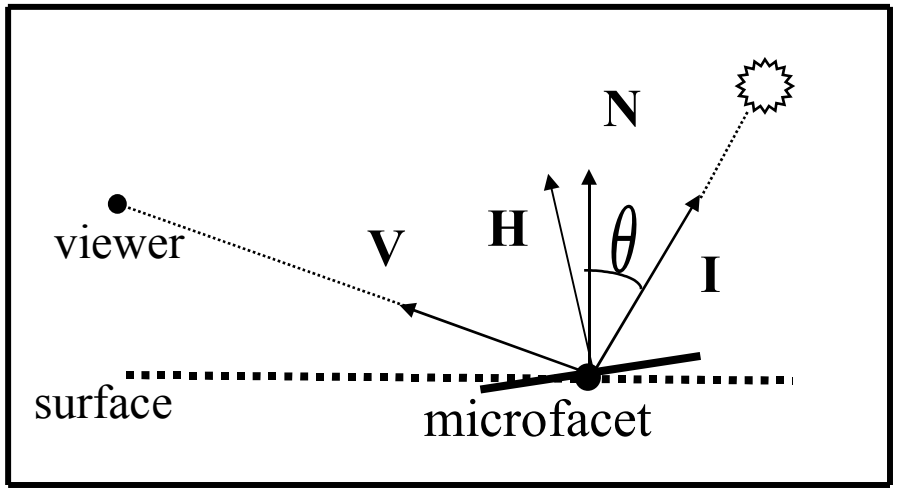
Ward Reflection Model

- **BRDF**

$$f_r = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{(I \bullet N)(V \bullet N)}} \bullet \frac{\exp(-\tan^2 \angle (H,N)/\sigma^2)}{4\pi\sigma^2}$$

σ standard deviation (RMS) of surface slope

- Simple expansion to anisotropic model (σ_x, σ_y)
- Empirical, not physics-based
- Inspired by notion of reflecting microfacets
- Convincing results
- Good match to measured data



Physics-inspired BRDFs

- **Notion of reflecting microfacet**
- **Specular reflectivity of the form**

$$f_r = \frac{D \cdot G \cdot F_\lambda(\lambda, \theta_i)}{\pi \ \underline{N} \cdot \underline{V}}$$

  - D : statistical microfacet distribution
  - G : geometric attenuation, self-shadowing
  - F : wavelength, angle dependency of reflection along mirror direction
  - $\underline{N} \bullet \underline{V}$: flaring effect at low angle of incidence

- **Cook-Torrance model**
  - F : wavelength- and angle-dependent reflection
  - Metal surfaces

Cook-Torrance Reflection Model

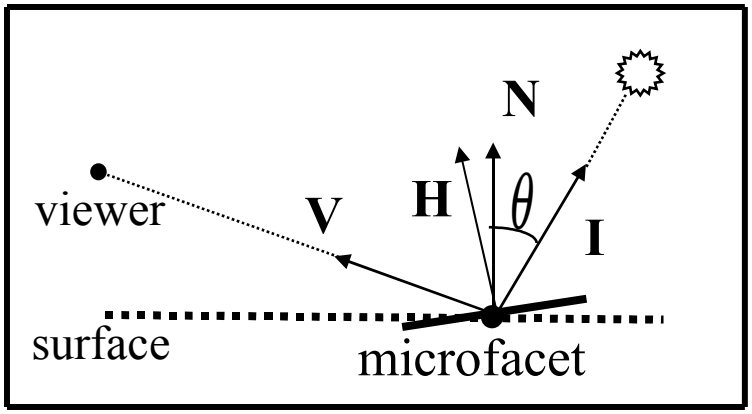
- Cook-Torrance reflectance model is based on the *microfacet* model. The BRDF is defined as the sum of a diffuse and specular components:

$$f_r = k_d \rho_d + k_s \rho_s; \qquad k_d + k_s \leq 1$$

  where k_s and k_d are the specular and diffuse coefficients.
- Derivation of the specular component ρ_s is based on a **physically derived** theoretical reflectance model

Cook-Torrance Specular Term

$$\rho_s = \frac{F_\lambda DG}{\pi\,(\underline{N}\cdot\underline{V})(\underline{N}\cdot\underline{I})}$$

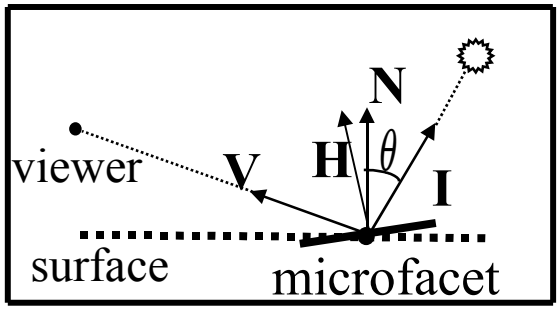


- **D : Distribution function of microfacet orientations**
- **G : Geometrical attenuation factor**
  - represents self-masking and shadowing effects of microfacets
- **F_λ : Fresnel term**
  - computed by Fresnel equation
  - relates incident light to reflected light for each planar microfacet
- **N·V : Proportional to visible surface area**
- **N·I : Proportional to illuminated surface area**

Microfacet Distribution Functions

- **Isotropic Distributions** $D(\underline{\omega}) \Rightarrow D(\alpha) \quad \alpha = \mathbf{N} \cdot \mathbf{H}$

  α : angle to average normal of surface
  - Characterized by half-angle β



$$D(\beta) = \frac{1}{2}$$

- **Blinn**

$$D(\alpha) = \cos^{\frac{\ln 2}{\ln \cos \beta}} \alpha$$

- **Torrance-Sparrow**

$$D(\alpha) = e^{-\left(\frac{\sqrt{2}}{\beta}\alpha\right)^2}$$

- **Beckmann**
  - m : root mean square
  - Used by Cook-Torrance

$$D(\alpha) = \frac{1}{4m^2 \cos^4 \alpha} e^{-[\tan \alpha / m]^2}$$

Geometric Attenuation Factor

- **V-shaped grooves**
- Fully illuminated and visible

$$G = 1$$

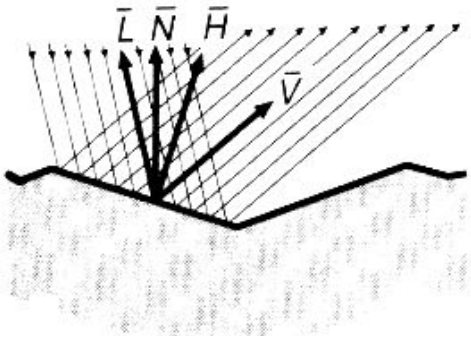
- Partial masking of reflected light

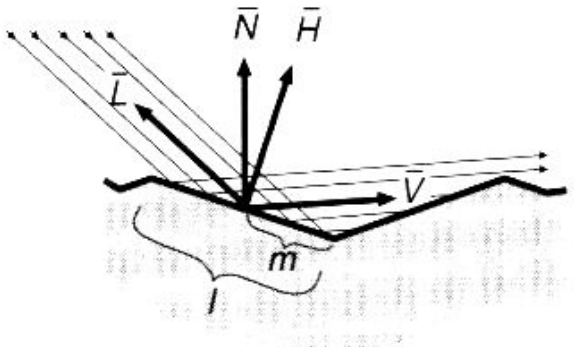
$$G = \frac{2(\underline{N} \cdot \underline{H})(\underline{N} \cdot \underline{V})}{(\underline{V} \cdot \underline{H})}$$

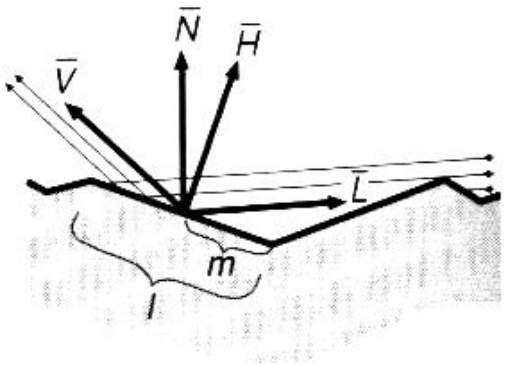
- Partial shadowing of incident light

$$G = \frac{2(\underline{N} \cdot \underline{H})(\underline{N} \cdot \underline{I})}{(\underline{V} \cdot \underline{H})}$$

$$G = \min\left\{1, \frac{2(\underline{N} \cdot \underline{H})(\underline{N} \cdot \underline{V})}{(\underline{V} \cdot \underline{H})}, \frac{2(\underline{N} \cdot \underline{H})(\underline{N} \cdot \underline{I})}{(\underline{V} \cdot \underline{H})}\right\}$$

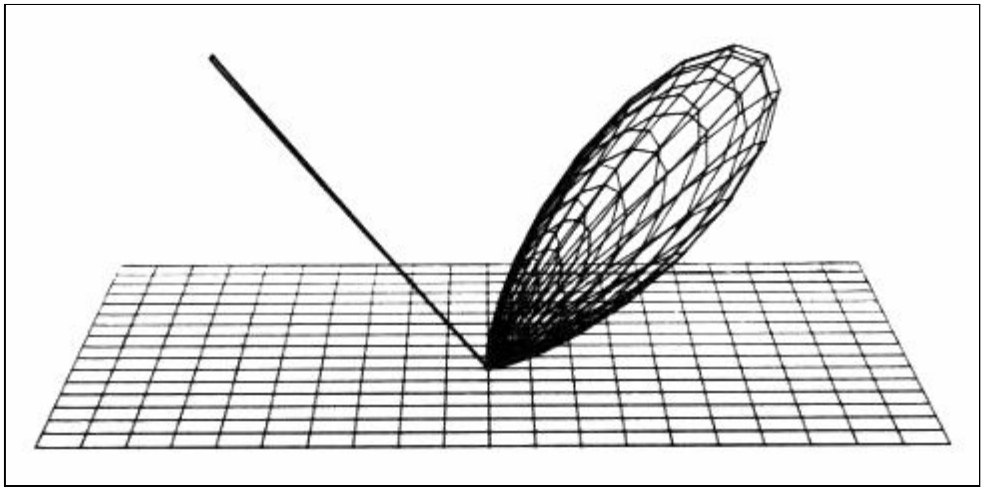




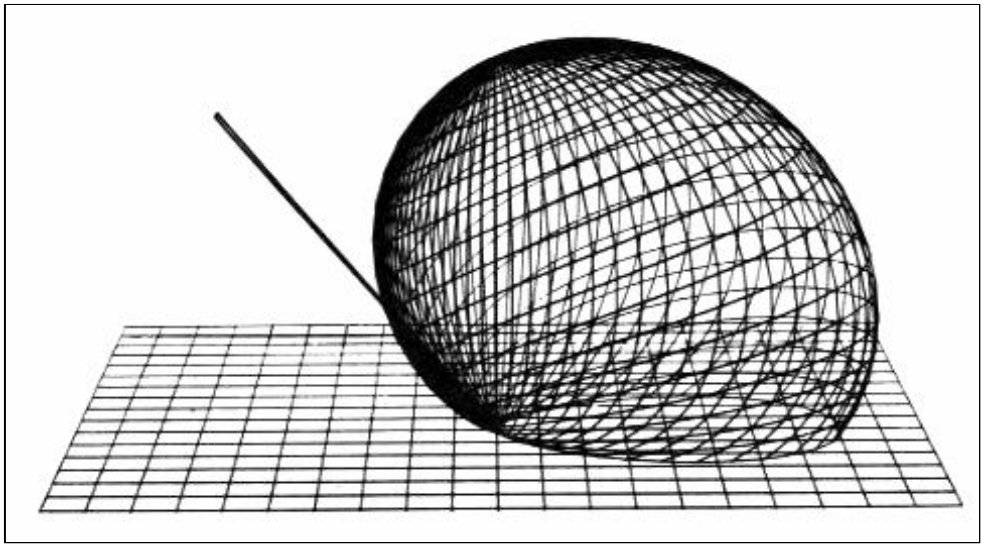


Beckman Microfacet Distribution Function

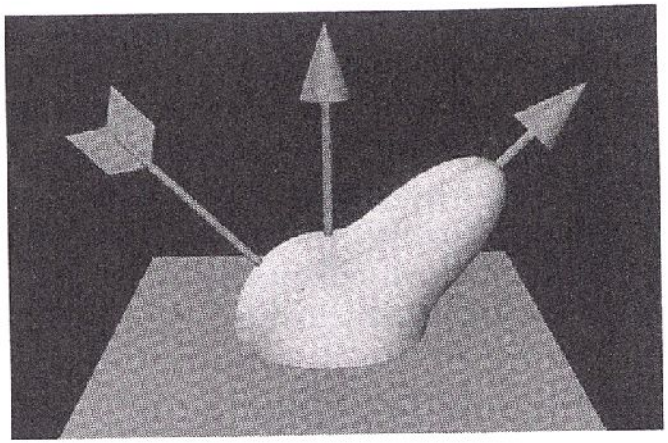
$m=0.2$



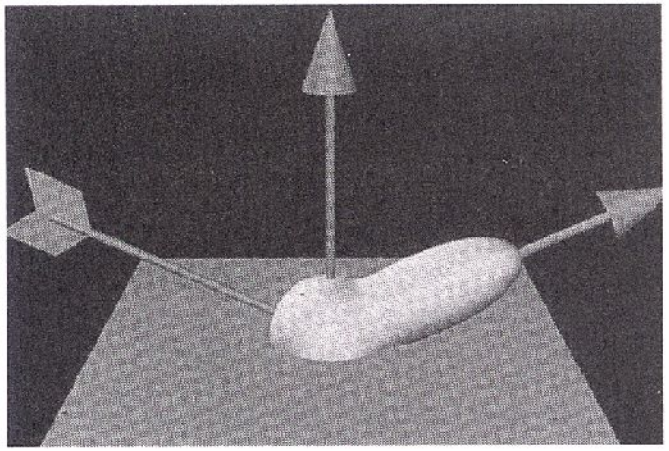
$m=0.6$



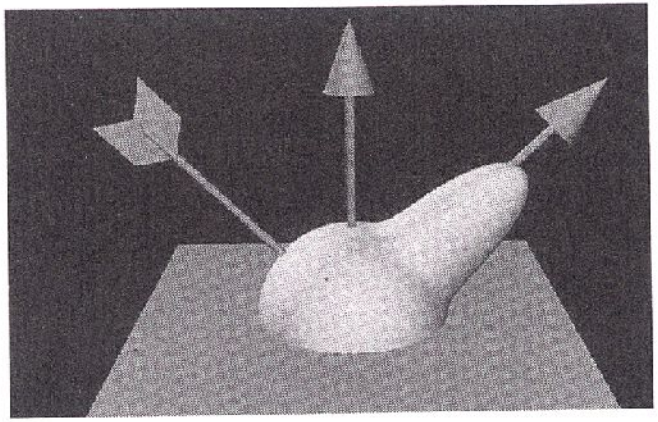
Comparison Phong vs. Torrance



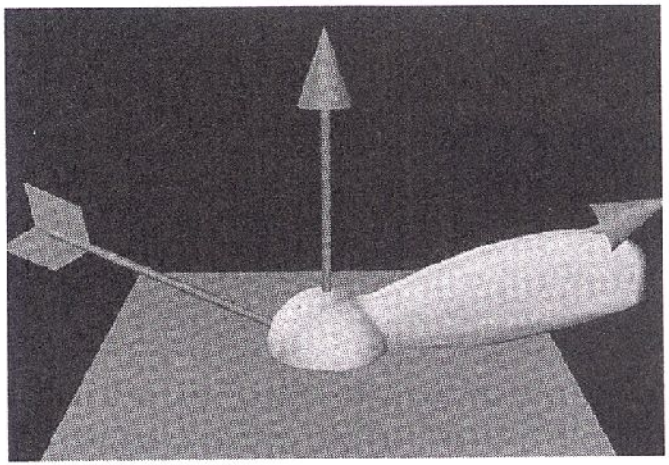
(a)



(b)



(c)



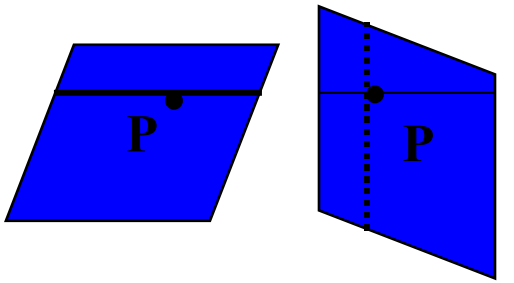
(d)

Polygon-Shading Methods

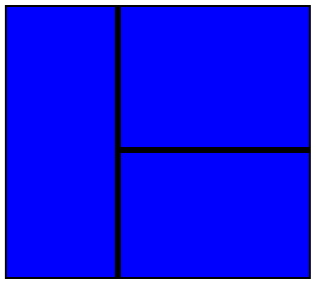
- Application of an illumination model to compute intensity for every pixel has been time consuming.
- Intensity of adjacent pixels is usually very similar (the so called shading coherence), which allows for less frequent shading evaluations.
- Each polygon can be rendered with a single intensity or intensity can be obtained at each point of the surface using an interpolation scheme:
  - ***Flat shading***, single intensity is calculated for each polygon
  - ***Gouraud shading*** (per vertex shading), intensity calculated at vertices is interpolated across the surface
  - ***Phong shading*** (per pixel shading), normal vectors are calculated at vertices; then normal vectors are interpolated across the surface and an illumination model using these normal vectors is applied for every point of the surface
- With modern hardware this is no big issue any more
  - Often even the normal is calculated per pixel
    - Bump or displacement maps

Problems in Interpolated Shading

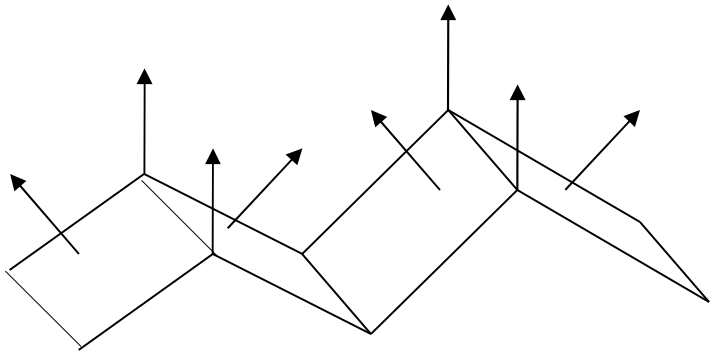
- **Problems**
  - Polygonal silhouette may not match the smooth shading
  - Perspective distortion
    - Interpolation may be performed after perspective transformation in the 2-D screen coordinate system, rather than world coordinate system.
  - Orientation dependence.
    - This problem does not concern triangles for which linear interpolation is rotation-invariant.
  - Shading discontinuities at shared vertices (T-edges).
  - Unrepresentative normal vectors.



Shading at **P** is interpolated along different scan-lines when polygon rotates.



T-edges

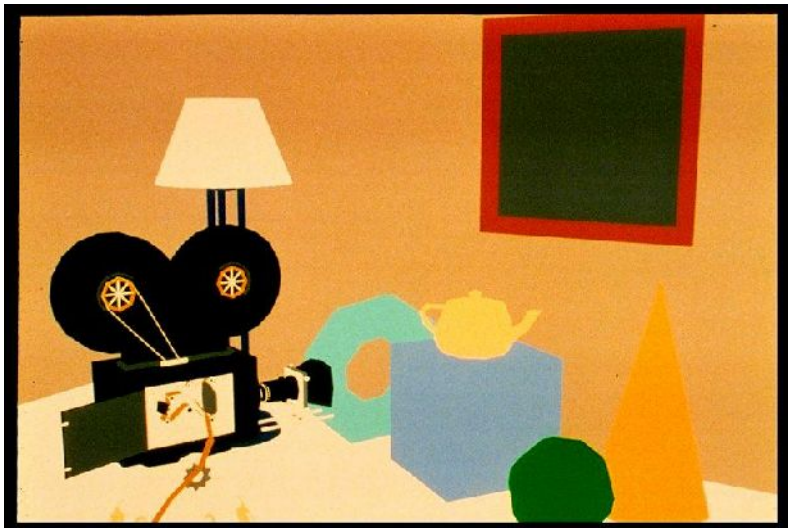


Vertex normals are all parallel

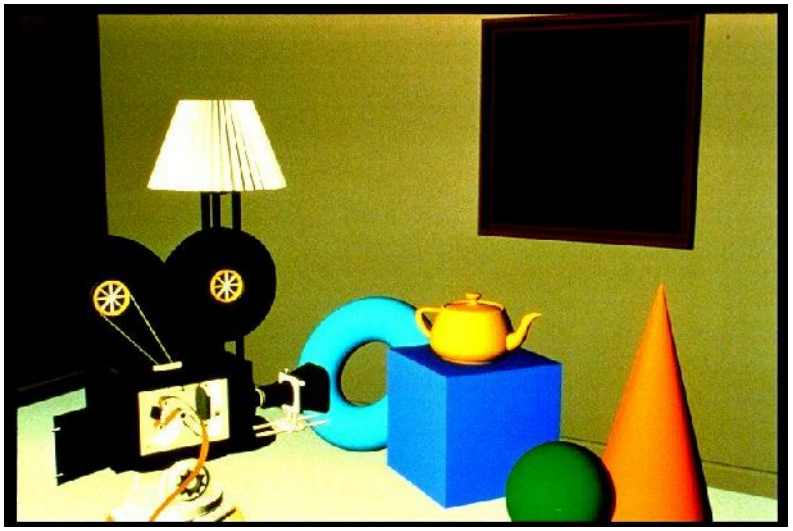
Texturing

Simple Illumination

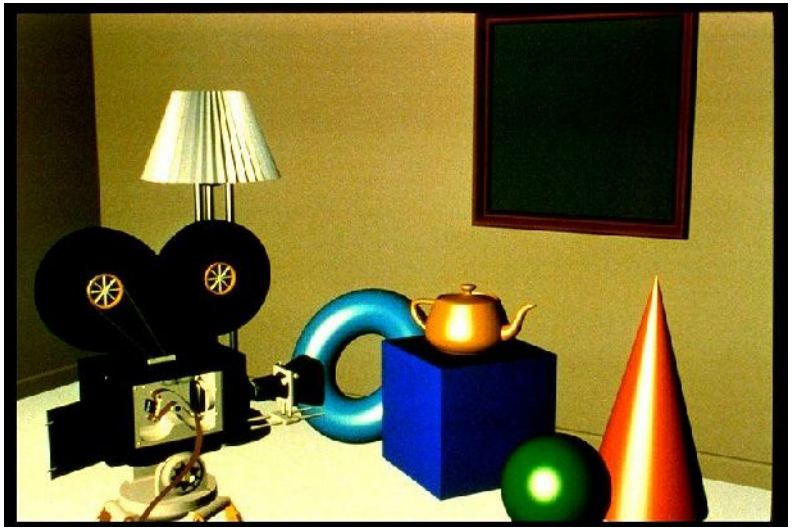
- **No illumination**
- **Constant colors**



- **Parallel light**
- **Diffuse reflection**



Standard Illumination



- **Parallel light**
- **Specular reflection**

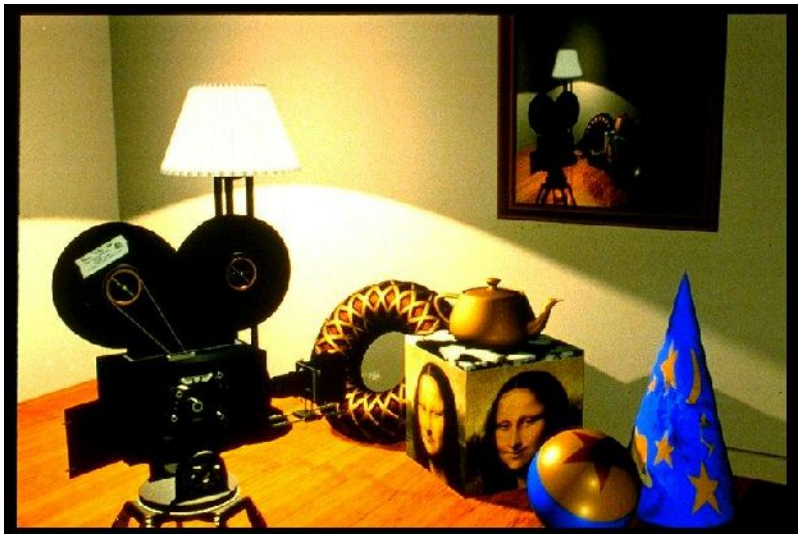
- **Multiple local light sources**
- **Different BRDFs**

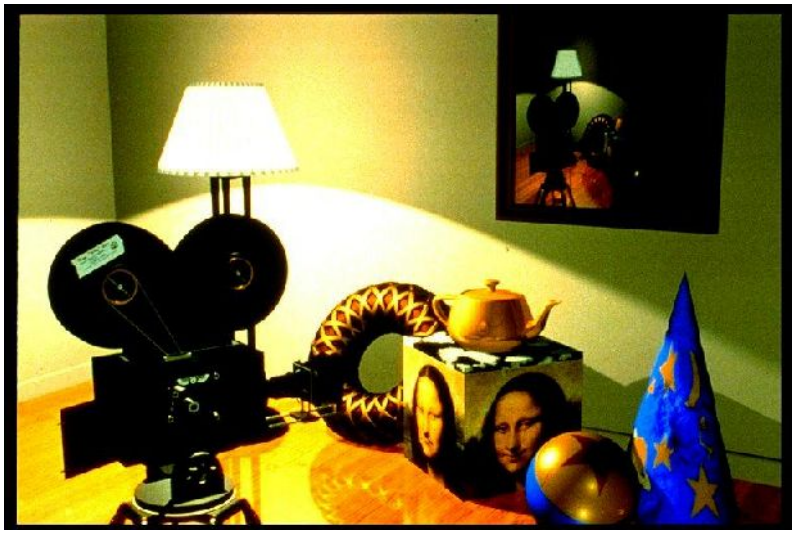
Object properties constant over surface

Texturing

Locally varying object characteristics

- **2D Image Textures**
- **Shadows**
- **Bump-Mapping**
- **Reflection textures**

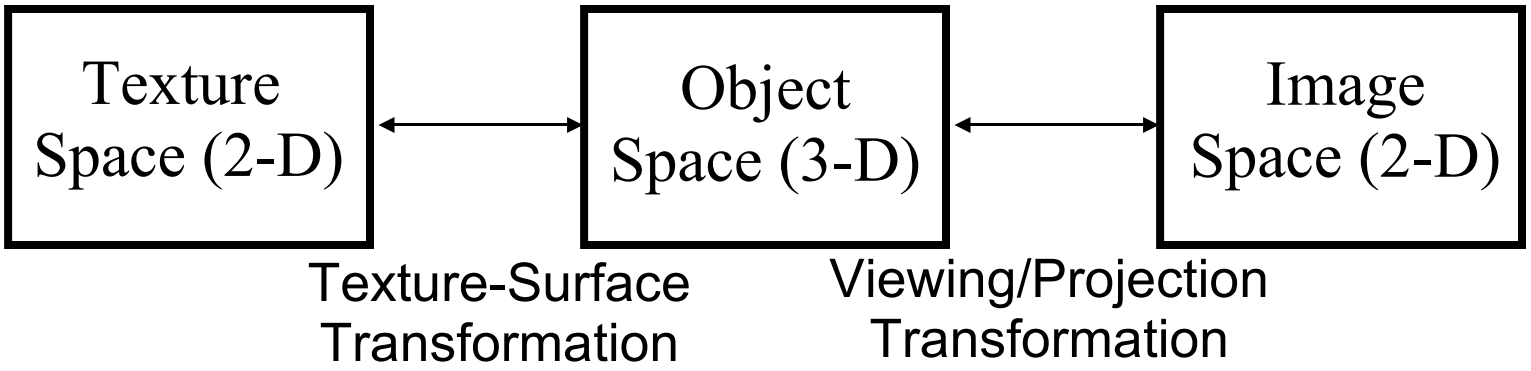




Texture-modulated Quantities

- **Modulation of object surface properties**
- **Reflectance**
  - Color (RGB), diffuse reflection coefficient kd
  - Specular reflection coefficient ks
- **Opacity (α)**
- **Normal vector**
  - N(P)= N(P+ t N) or N= N+dN
  - „Bump mapping“ or „Normal mapping“
- **Geometry**
  - P= P + dP
  - „Displacement mapping“
- **Distant illumination**
  - “Environment mapping“, “Reflection mapping“

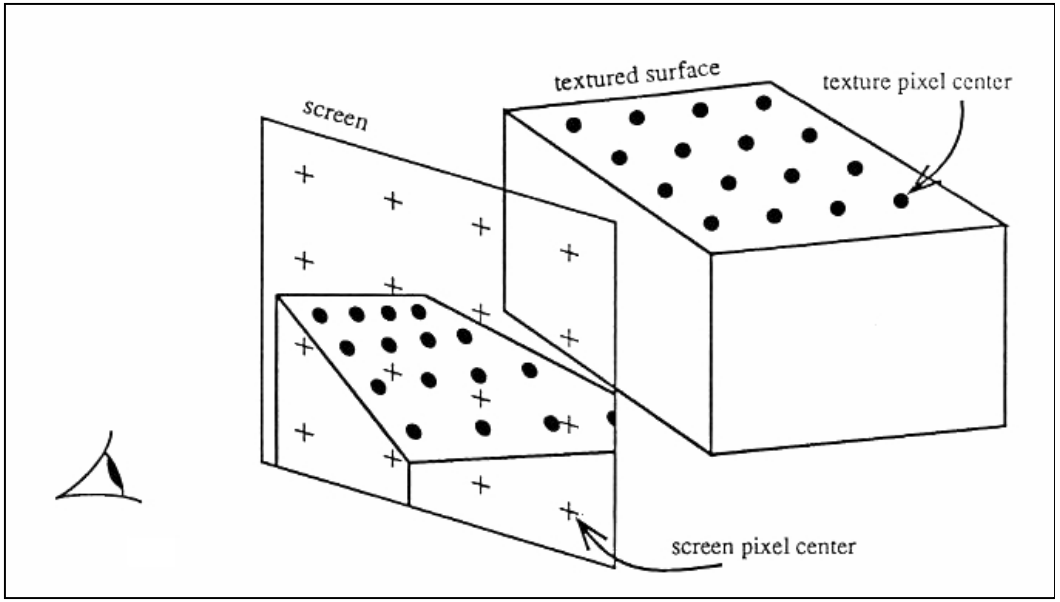
Texture Mapping Transformations

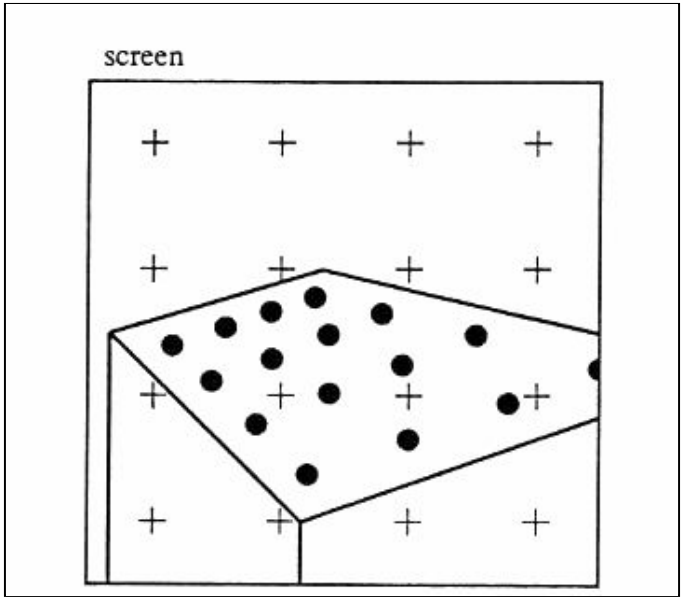


The texture is mapped onto a surface in 3-D object space, which is then mapped to the screen by the viewing projection. These two mappings are composed to find the overall 2-D texture space to 2-D image space mapping, and the intermediate 3-D space is often forgotten. This simplification suggests texture mapping's close ties with image warping and geometric distortion.

Texture space (u,v)
Object space (x_o,y_o,z_o)
Screen space (x,y)

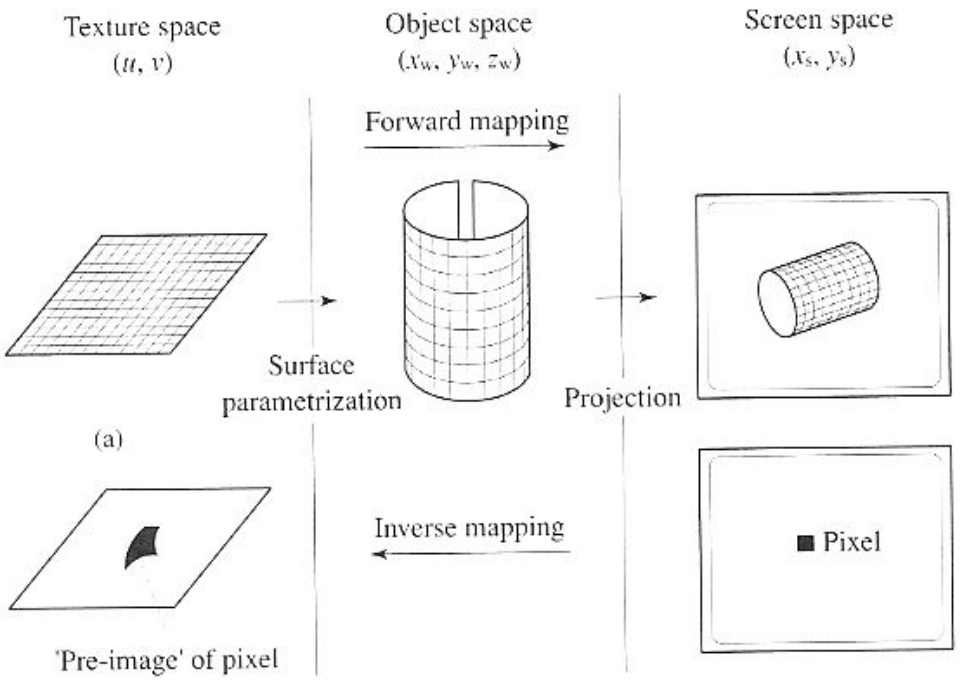
2D Texturing





- **2D texture mapped onto object**
- **Object projected onto 2D screen**
- **2D→2D: warping operation**
- **Uniform sampling ?**
- **Hole-filling/blending ?**

2D Texture Mapping



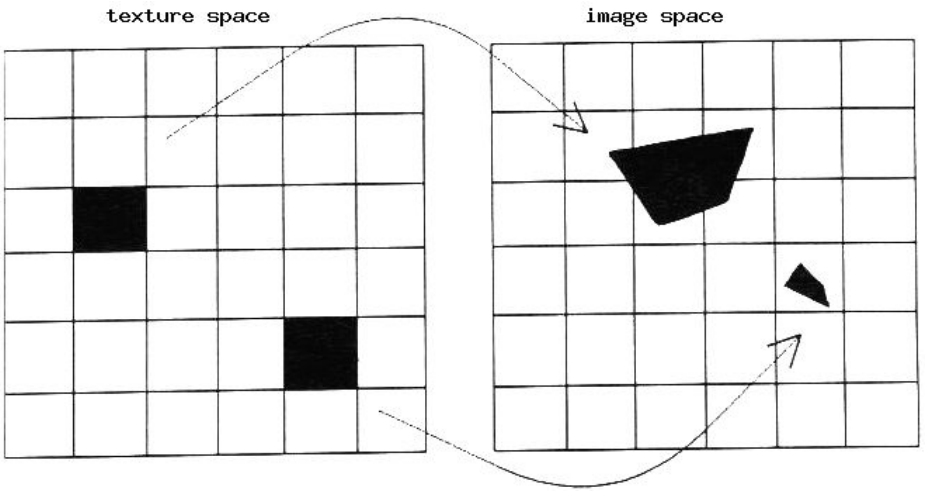
- **Forward mapping**
  - Object surface parameterization
  - Projective transformation
- **Inverse mapping**
  - Find corresponding pre-image/footprint of each pixel in texture
  - Integrate over pre-image

Forward Mapping

- **Maps each texel to its position in the image**
- **Uniform sampling of texture space does not guarantee uniform sampling in screen space**
- **Possibly used if**
  - The texture-to-screen mapping is difficult to invert
  - The texture image does not fit into memory

Texture scanning:

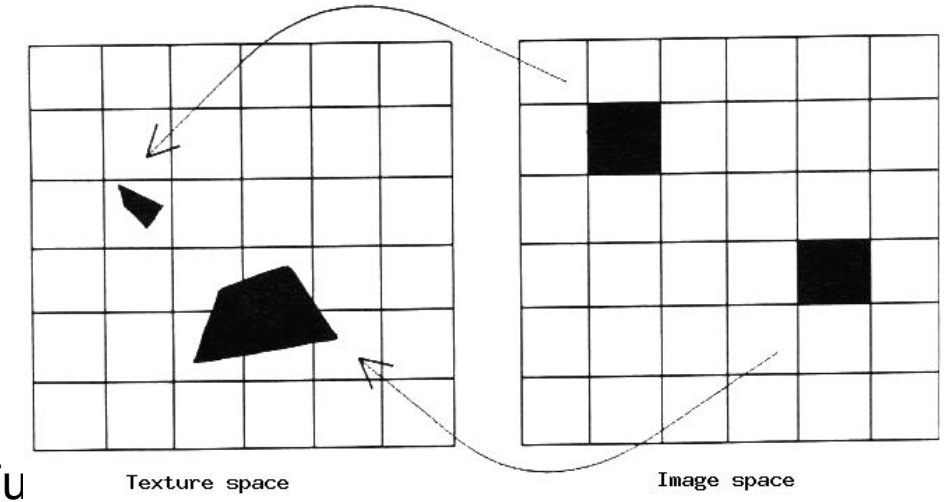
```
for v
  for u
    compute x(u,v) and y(u,v)
    copy TEX[u,v] to SCR[x,y]
(or in general
     rasterize image of TEX[u,v])
```



Inverse Mapping

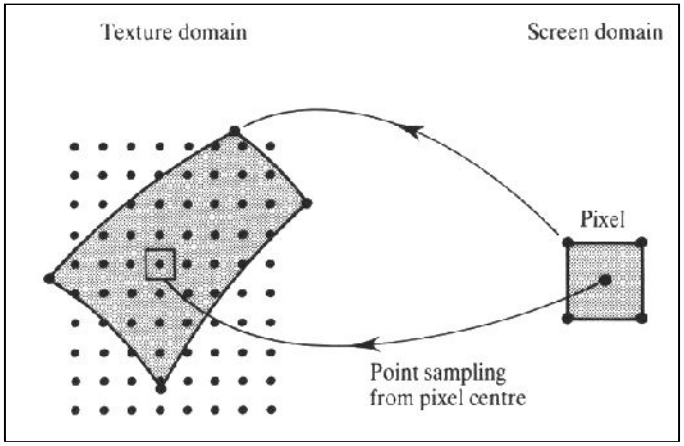
- **Requires inverting the mapping transformation**
- **Preferable when the mapping is readily invertible and the texture image fits into memory**
- **The most common mapping method**
  - for each pixel in screen space, the pre-image of the pixel in texture space is found and its area is integrated over

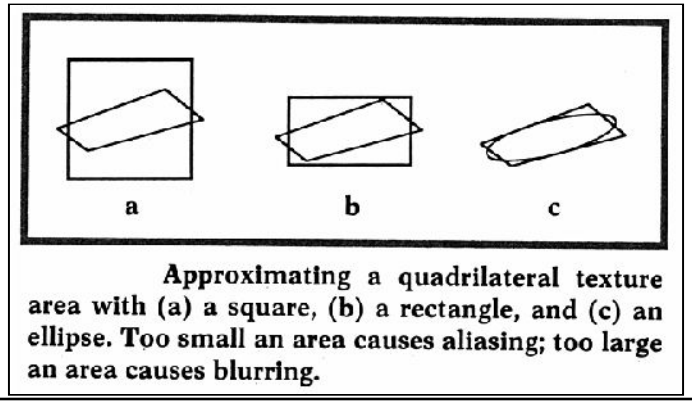
Screen scanning:
for y
  for x
    compute u(x,y) and v(x,y)
    copy TEX[u,v] to SCR[x,y]
(or in general
    integrate over image of SCR[u



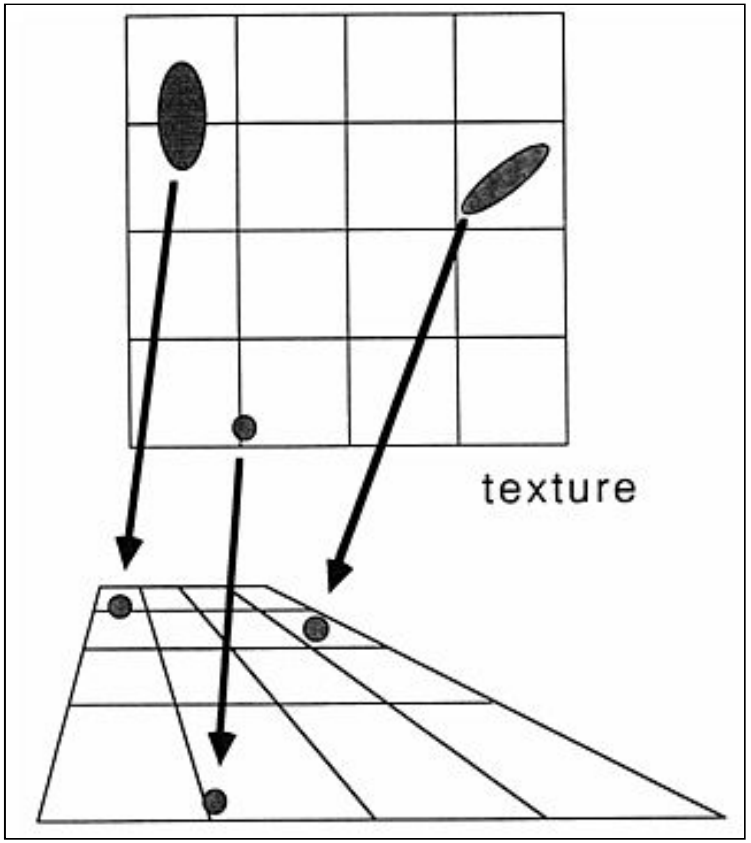
Pixel Pre-Image in Texture Space

A square screen pixel that intersects a curved surface has a curvilinear quadrilateral pre-image in texture space. Most methods approximate the true mapping by a quadrilateral or parallelogram. Or they take multiple samples within a pixel. If pixels are instead regarded as circles, their pre-images are ellipses.



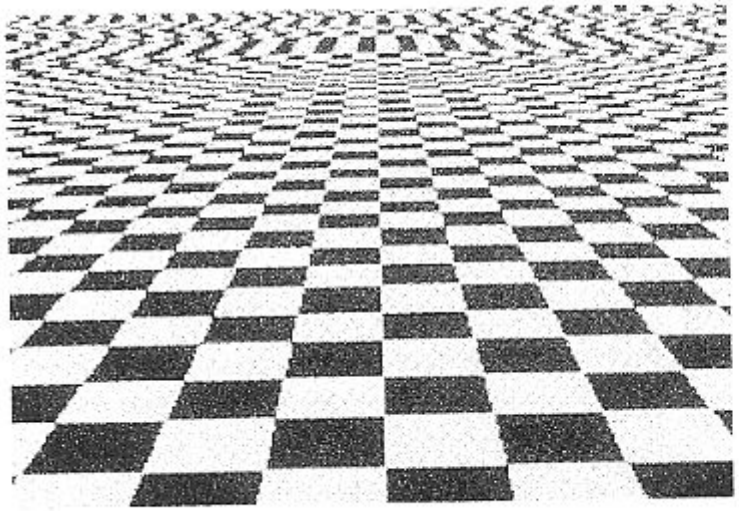


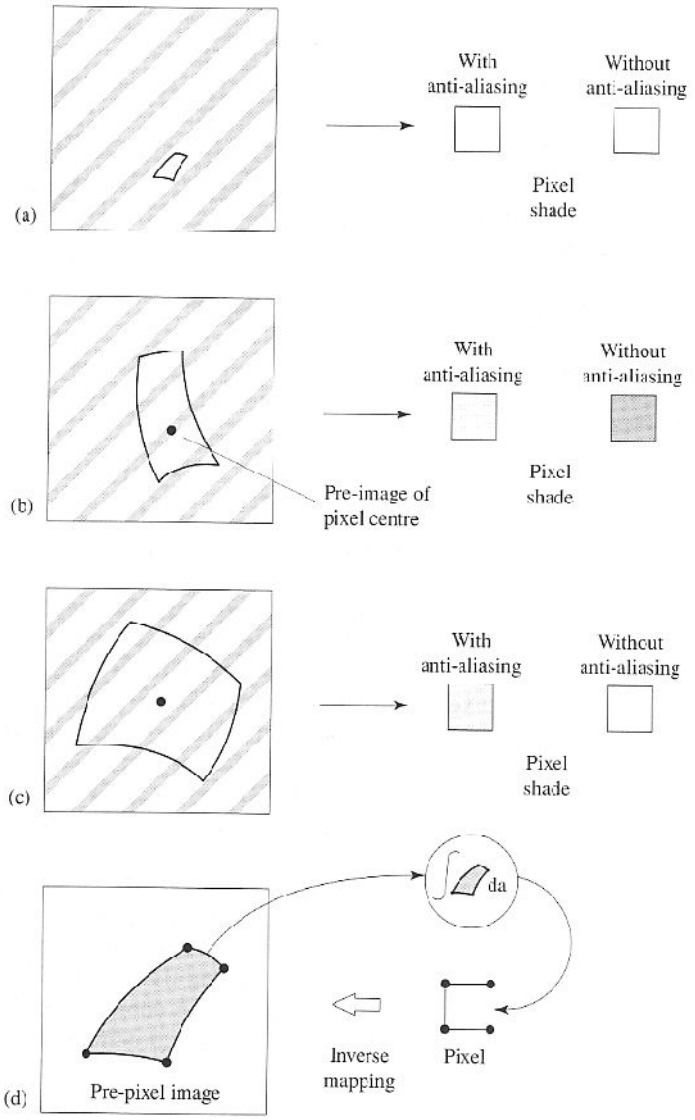
Approximating a quadrilateral texture area with (a) a square, (b) a rectangle, and (c) an ellipse. Too small an area causes aliasing; too large an area causes blurring.



Inverse Mapping: Filtering

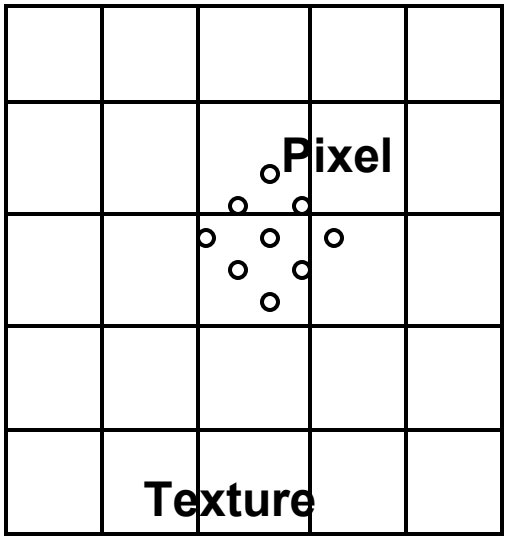
- **Integration of Pre-image**
  - Integration over pixel footprint in texture space
- **Aliasing**
  - Texture insufficiently sampled
  - Incorrect pixel values
  - “Randomly” changing pixels when moving

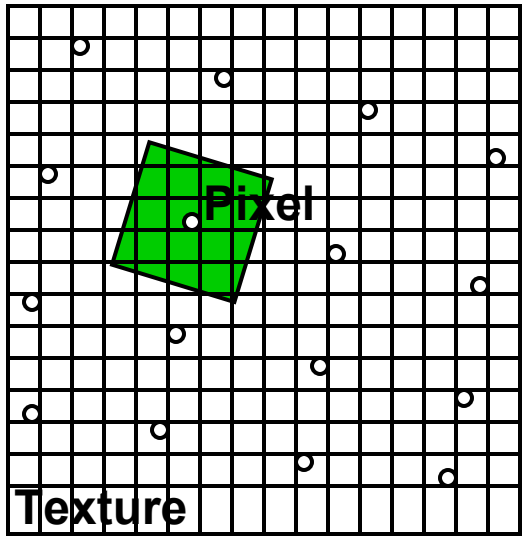




Filtering

- **Magnification**
  - Map few texels onto many pixels
  - Nearest:
    - Take the nearest texel
  - Bilinear interpolation:
    - Interpolation between 4 nearest texels
    - Need fractional accuracy of coordinates
- **Minification**
  - Map many texels to one pixel
    - Aliasing:
      - Reconstructing high-frequency signals with low level frequency sampling
  - Filtering
    - Averaging over (many) associated texels
    - Computationally expensive



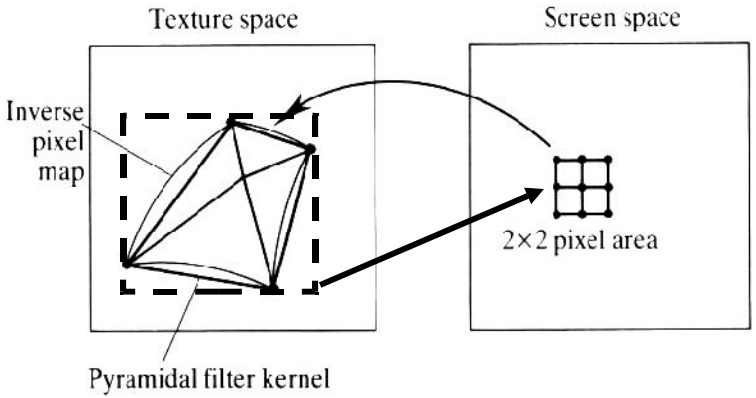


Filtering – Texture Minification

- **Space-variant filtering**
  - Mapping from texture space *(u,v)* to screen space *(x,y)* not affine
  - Filtering changes with position
- **Space variant filtering methods**
  - ***Direct convolution***
    - Numerically compute the Integral
  - ***Pre-filtering***
    - Precompute the integral for certain regions -- more efficient
    - Approximate footprint with regions

Direct Convolution

- **Convolution in texture space**
  - Texels weighted according to distance from pixel center (e.g. pyramidal filter kernel)



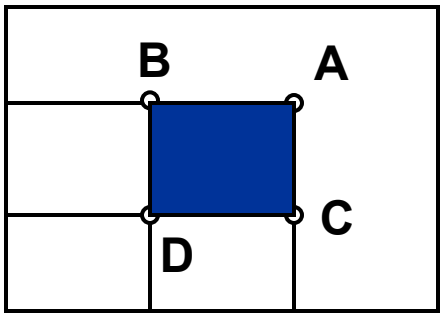
- **Convolution in image space**
  1. **Center the filter function on the pixel (in image space) and find its bounding rectangle.**
  2. **Transform the rectangle to the texture space, where it is a quadrilateral. The sides of this rectangle are assumed to be straight. Find a bounding rectangle for this quadrilateral.**
  3. **Map all pixels inside the texture space rectangle to screen space.**
  4. **Form a weighted average of the mapped texture pixels using a two-dimensional lookup table indexed by each sample's location within the pixel.**

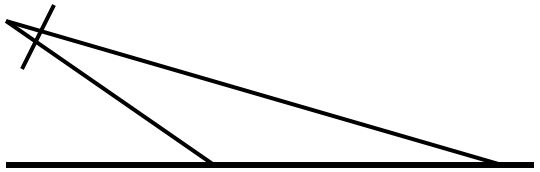
Filtering – Texture Minification

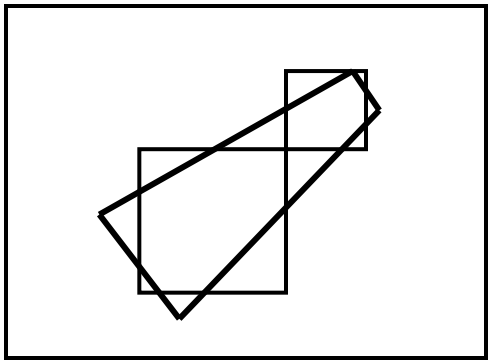
- **Direct convolution methods are slow**
  - A pixel pre-image can be arbitrarily large along silhouettes or at the horizon of a textured plane
  - Horizon pixels can require averaging over thousands of texture pixels
  - Texture filtering cost grows in proportion to projected texture area
- **Speed up**
  - The texture can be prefiltered so that during rendering only a few samples will be accessed for each screen pixel
- **Two data structures are commonly used for prefiltering:**
  - Integrated arrays (***summed area tables***)
  - Image pyramids (***mipmaps***) Space-variant filtering

Integrated Arrays

- **Summed-Area-Tables**
  - Per texel, store sum from (0, 0) to (u, v)
  - Arbitrary rectangle:
    - Area = A – B – C + D
  - Many bits per texel (sum !)
- **Footprint assembly**
  - Good for space variant filtering
    - e.g. inclined view of terrain
  - Approximation of the pixel area by rectangular texel-regions
  - The more footprints the better accuracy
  - Often fixed number of texels because of economical reasons

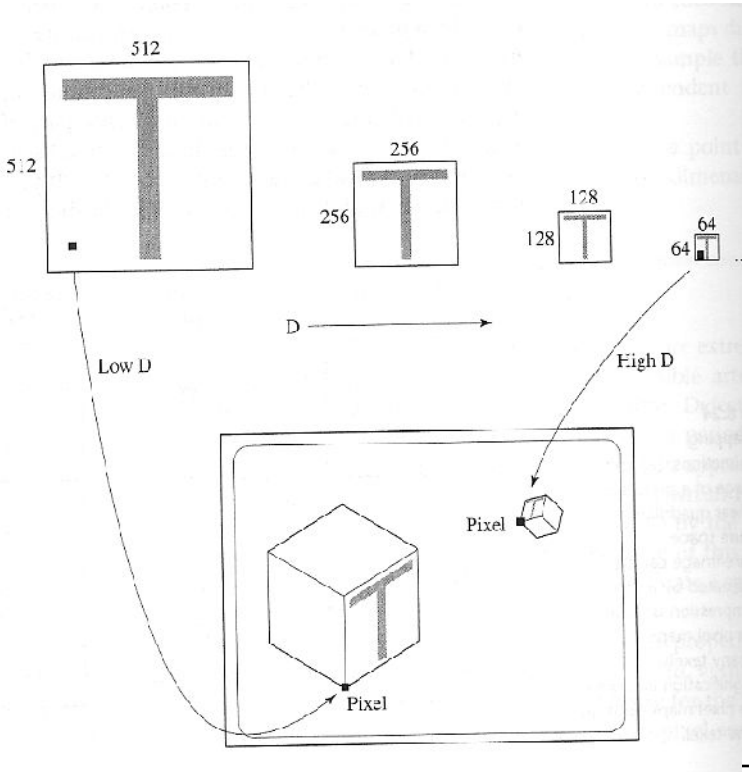






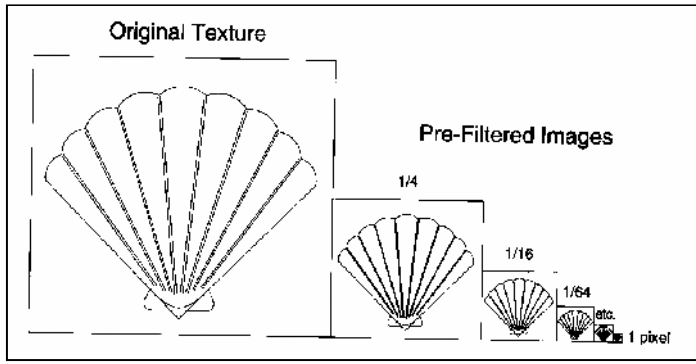
MipMapping

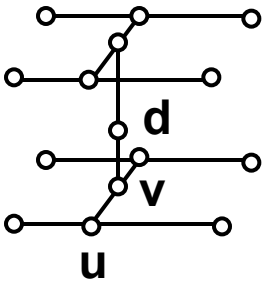
- **Texture available in multiple resolutions**
  - Pre-processing step
- **Rendering: select appropriate texture resolution**
  - Selection is usually per pixel !!
  - Texel size(n) < extent of pixel footprint < texel size(n+1)

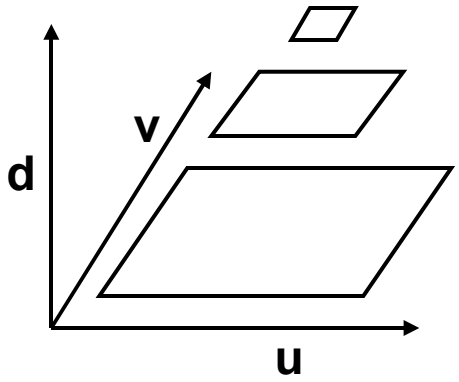


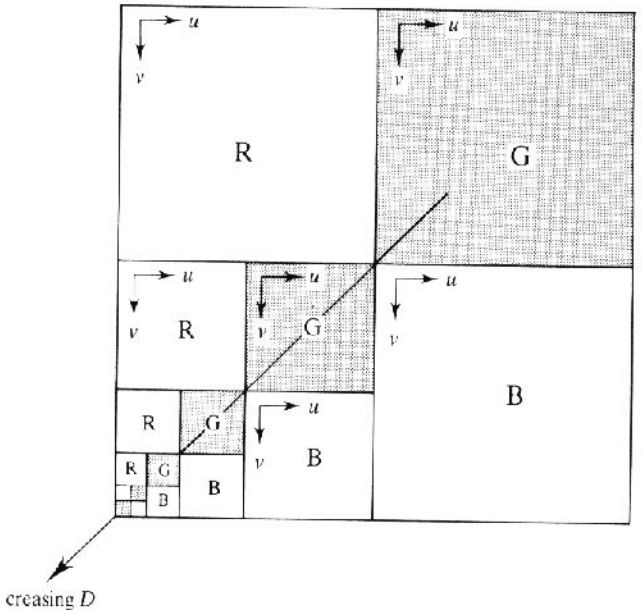
MipMapping II

- **Multum In Parvo (MIP): much in little**
- **Hierarchical resolution pyramid**
  - Repeated averaging over 2x2 texels
- **Rectangular arrangement (RGB)**
- **Reconstruction**
  - Tri-linear interpolation of 8 nearest texels

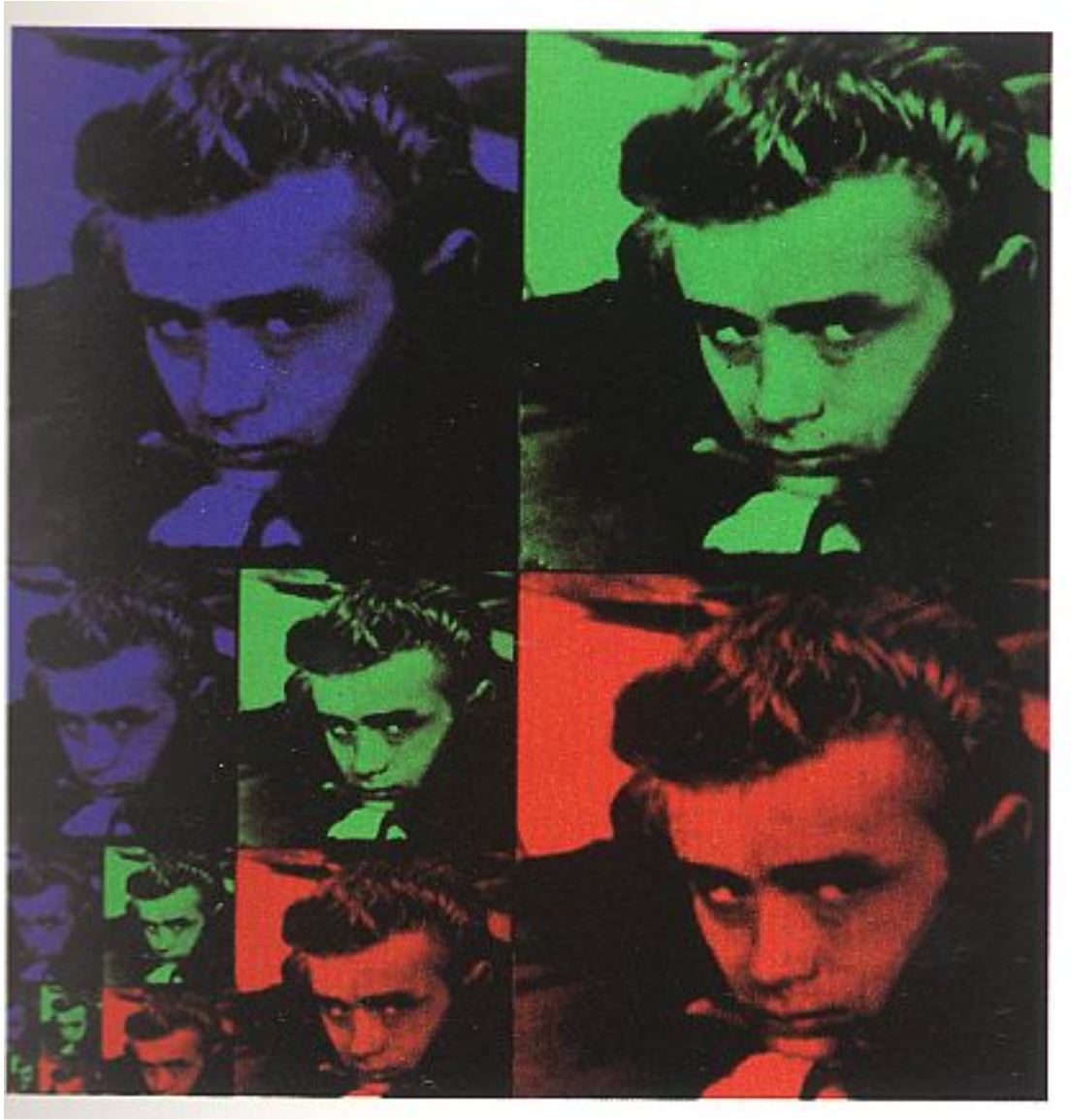








MipMap Example

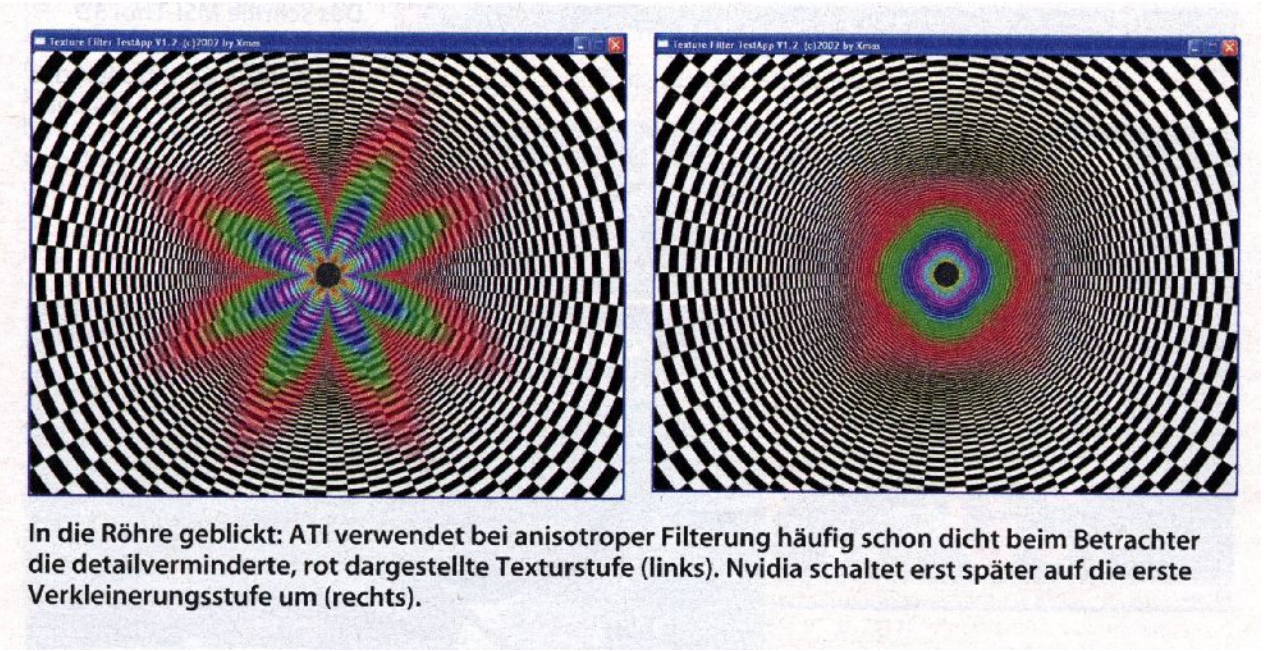


MipMaps

- **Why is MipMapping sometimes faster?**
  - Bottleneck is memory bandwidth
  - Using of texture caches
  - Texture minification required for far geometry
- **No MipMap**
  - „Random“ access to texture
  - Always 4 new texels
- **MipMap**
  - Next pixel at about one texel distance (1:1 mapping)
  - Access to 8 texels at a time, but
    - Most texels are still in the cache

Anisotropic Filtering

- **Footprint Assembly on GPUs**
  - Integration Across Footprint of Pixel
  - HW: Choose samples that best approximate footprint
  - Weighted average of samples



In die Röhre geblickt: ATI verwendet bei anisotroper Filterung häufig schon dicht beim Betrachter die detailverminderte, rot dargestellte Texturstufe (links). Nvidia schaltet erst später auf die erste Verkleinerungsstufe um (rechts).

© C`t Magazine