Texturing

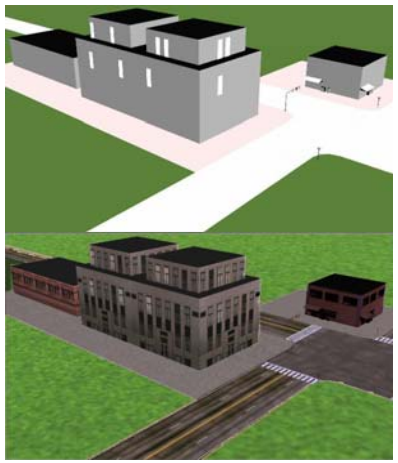
Eduard Gröller
(today: Stefan Jeschke)

Institute of Computer Graphics and Algorithms

Vienna University of Technology

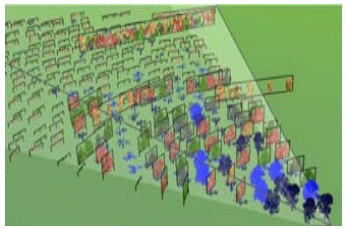
Why Texturing?

- Idea: enhance visual appearance of plain surfaces by applying fine structured details

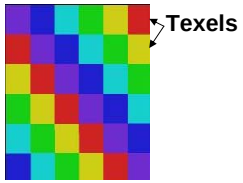


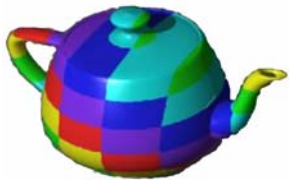
Why Texturing?

- Also possible: model very complex objects just by using simple textured geometry



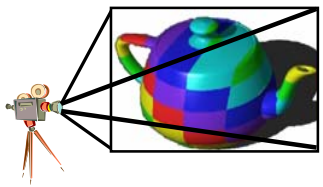
Texturing: General Approach





Texture space (u,v) **Object space** (x_O,y_O,z_O) **Image Space** (x_I,y_I)

Parametrization **Rendering (Projection etc.)**

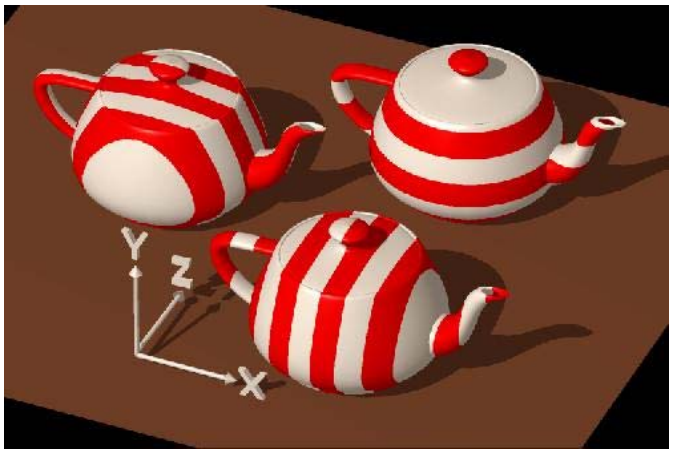


Texture Definition

- A texture is a **function**:
  - *Algebraic*: fast to evaluate but limited variety
  - *Sampled*: most common method
    - Raster images that are taken with a digital camera, scanned or synthesized
- Textures can be defined:
  - In 3D object space: *"solid texturing"*
  - On the 2D object surface: *"texture mapping"*

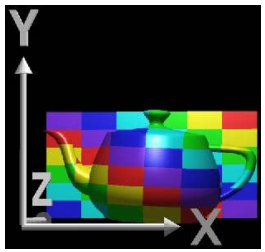
Solid Texturing: Parametrization

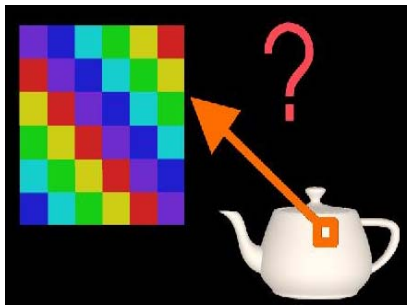
- ***1D*** texture: parameter can have arbitrary domain (along one axis, incident angle, etc.)

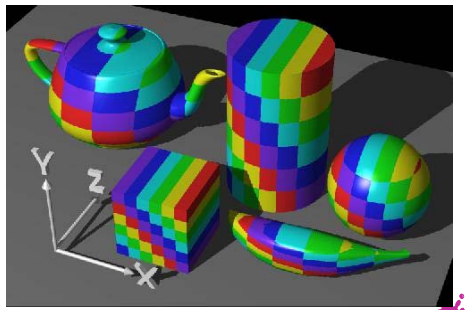


Solid Texturing: Parametrization

- ***2D*** texture
  - Projection of 2D data
  - Many possibilities for definition

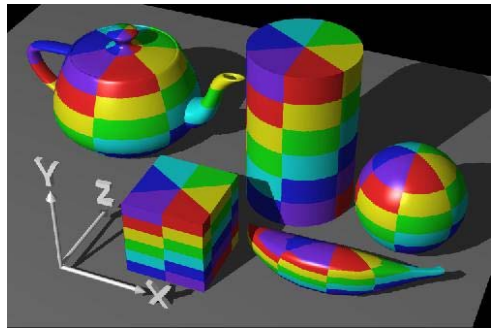






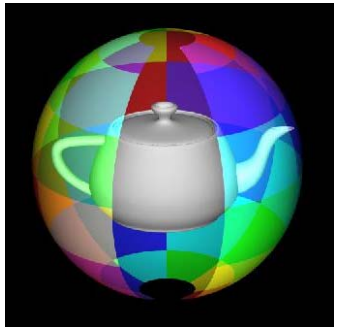
Solid Texturing: Parametrization

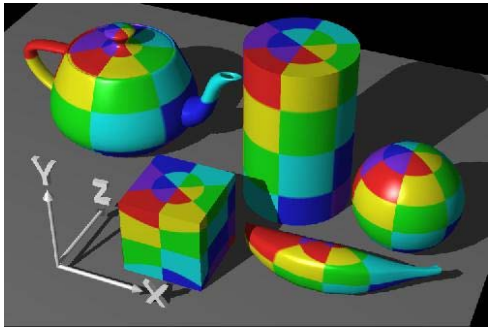
- *Other 2D texture: cylindrical parametrization*
  - Depending on cylindrical coordinates of each point



Solid Texturing: Parametrization

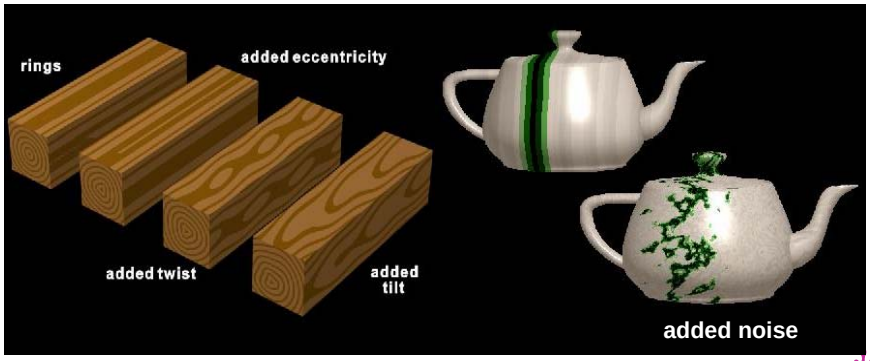
- *Other 2D texture: spherical parametrization*
  - Depending on spherical coordinates of each point



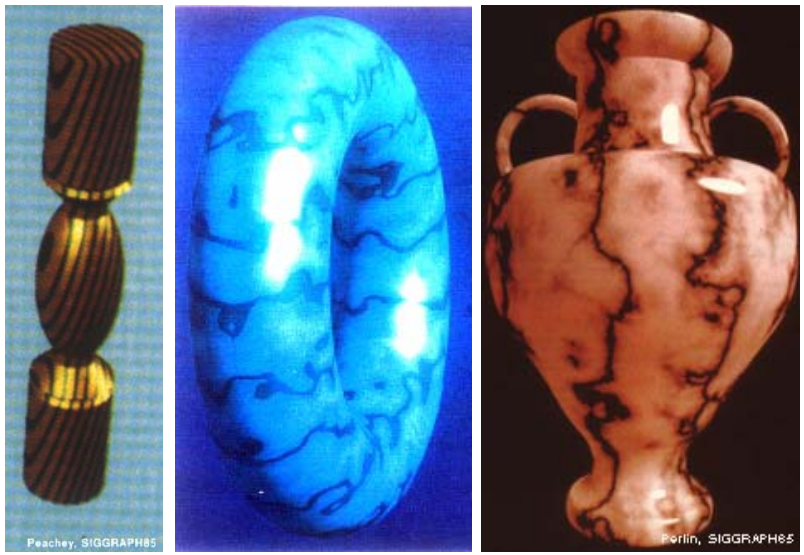


Solid Texturing: Parametrization

- ***3D*** texture: *T(u,v,w)*
  - $0 \le x^2+y^2 < 1$ ➡ light brown
    $1 \le x^2+y^2 < 2$ ➡ dark brown
    $2 \le x^2+y^2 < 3$ ➡ yellow etc.



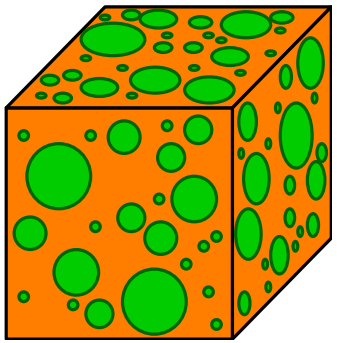
Solid Texturing Examples (1)



Solid Texturing Examples (2)

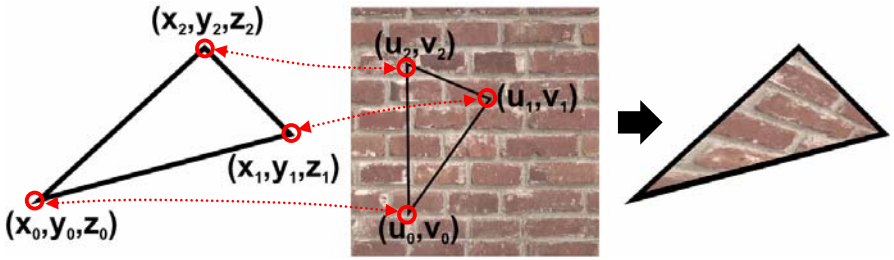
Solid Texturing Examples (3)

- *"Bombing"*: special solid texture function
  - Random sized spheres define areas of different material (like cheese holes)
  - Precalculation of position and radius of all spheres
  - Fast rendering



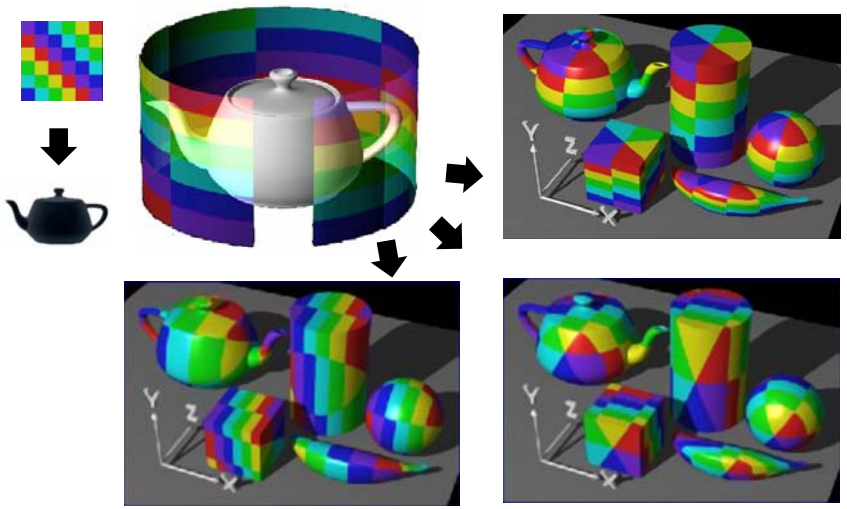
Texture Mapping

- Parametrization per vertex
  - Apply *(u,v)* texture coordinates to every vertex
- At runtime, ***(bi)linearly interpolate*** between these coordinates during rasterization



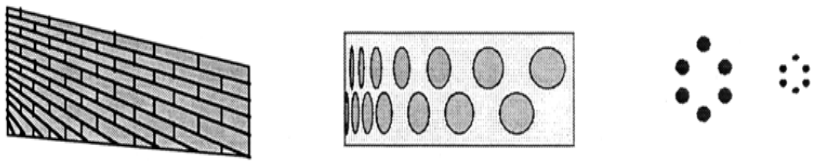
Texture Mapping: Parametrization

- Difficulty: how to minimize texture distortions?



Texture Mapping: Parametrization

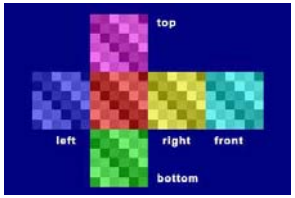
- ***Texture gradient***
  - Defines the amount of texture pattern distortion due to the spatial position in 3D-space
    - Size / Shape / Density

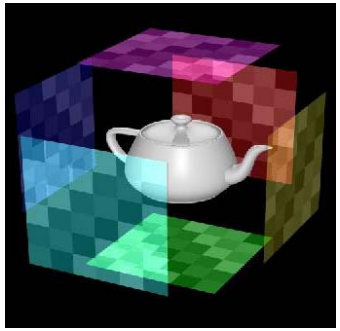


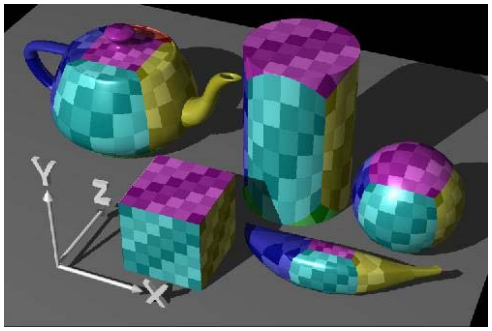
- ***Angular*** and ***area distortions*** can not be minimized at the same time! (except for some special cases)

Texture Mapping: Parametrization

- Example: box parametrization







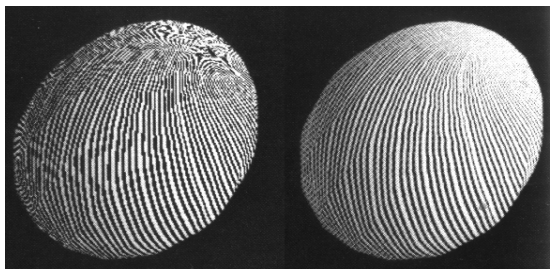
Texture Mapping vs. Solid Texturing

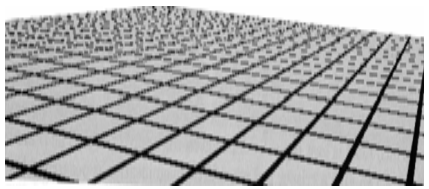
Solid texturing

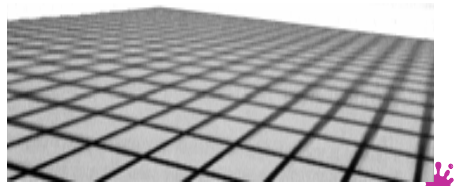
Texture mapping

Texturing: Rendering

- Problem: aliasing
  - One pixel in image space covers many texels

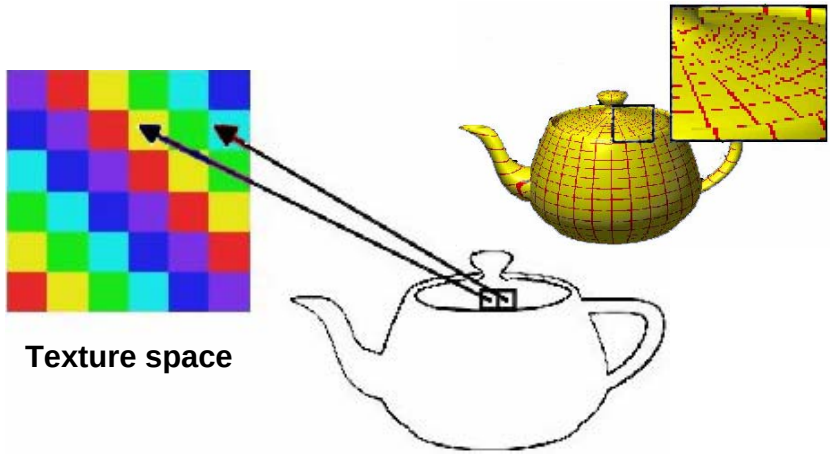






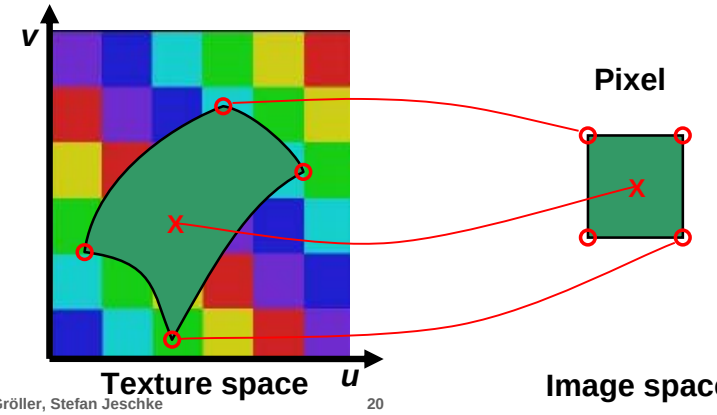
Aliasing

- Caused by *undersampling*: texture information is lost



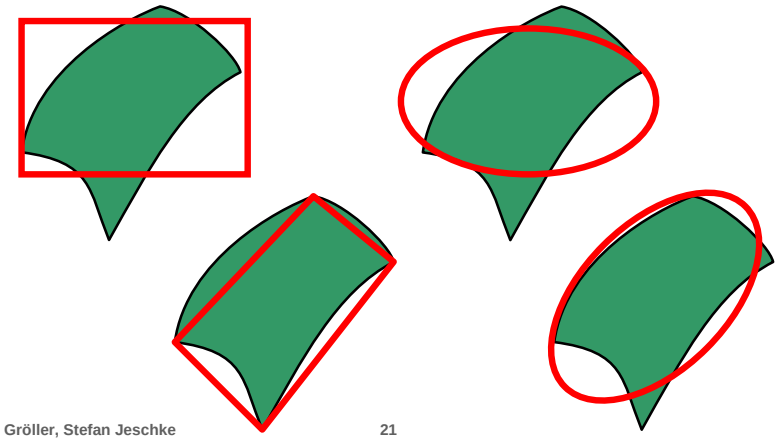
Anti-Aliasing

- A good pixel value is the weighted mean of the pixel area projected into texture space
  - Calculation at *runtime* or in a *preprocess*



Anti-Aliasing

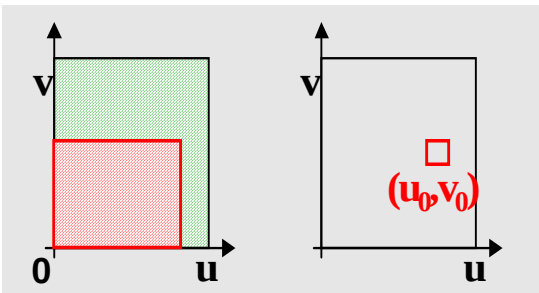
- Calculation of the weighted mean at runtime (called *"direct convolution"*)
  - Often used approximations



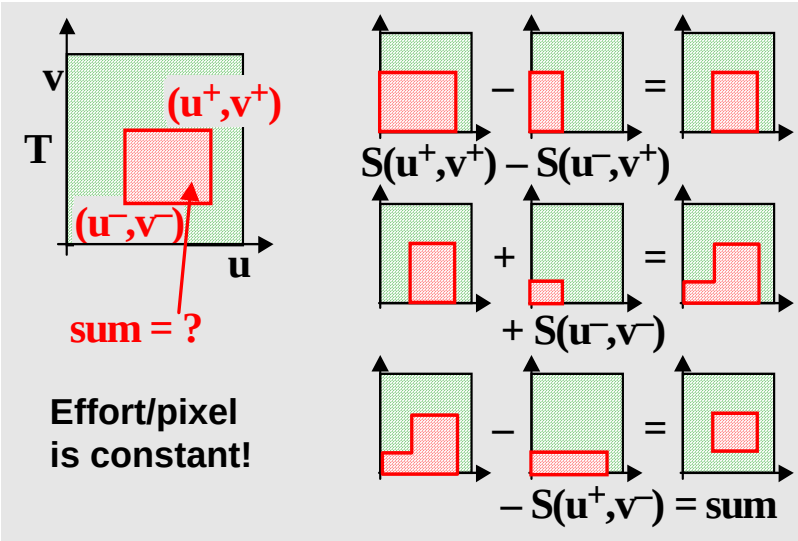
Anti-Aliasing

- Calculation of the weighted mean in a *preprocess* (called *"prefiltering"*)
  - *Summed area table*
  - *Mip Mapping* (+ optional *anisotropic filtering*)
- Summed area table S
  - precalculation of rectangle sums for each pixel

$$S(u_0,v_0) = \sum_{\substack{u \le u_0 \\ v \le v_0}} T(u,v)$$



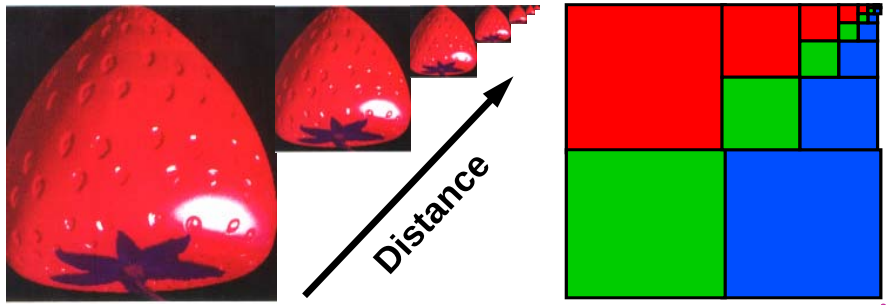
Anti-Aliasing: Summed Area Table



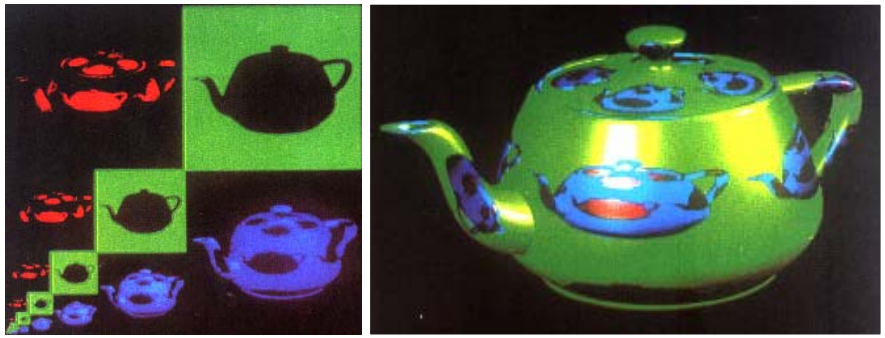
Anti-Aliasing: Example (Summed Area T.)

Anti-Aliasing: MIP Mapping

- MIP Mapping (“Multum In Parvo”)
  - Texture size is reduced by factors of 2 (*downsampling* = "much info on a small area")
  - Simple (4 pixel average) and memory efficient
  - Last image is only ONE texel

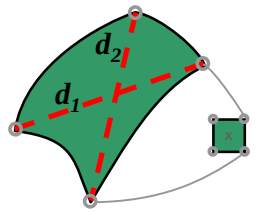


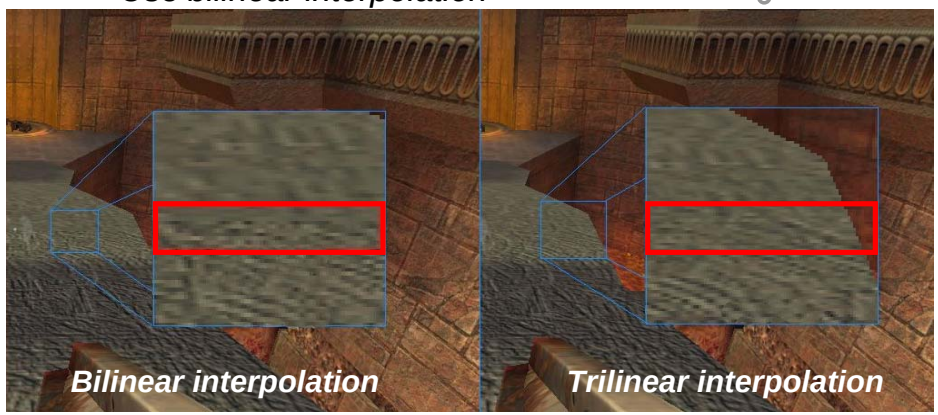
Anti-Aliasing: Example (Mip Mapping)



Anti-Aliasing: MIP Mapping

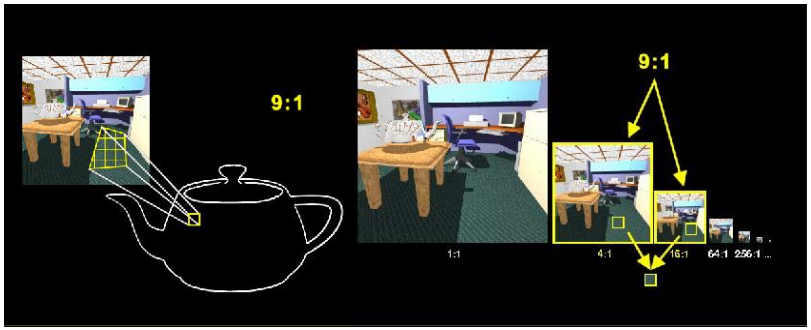
- MIP Mapping Algorithm
- $D := ld(max(d_1,d_2))$ *"Mip Map level"*
- $T_0 :=$ value from texture $D_0 = trunc\ (D)$
  - Use *bilinear interpolation*





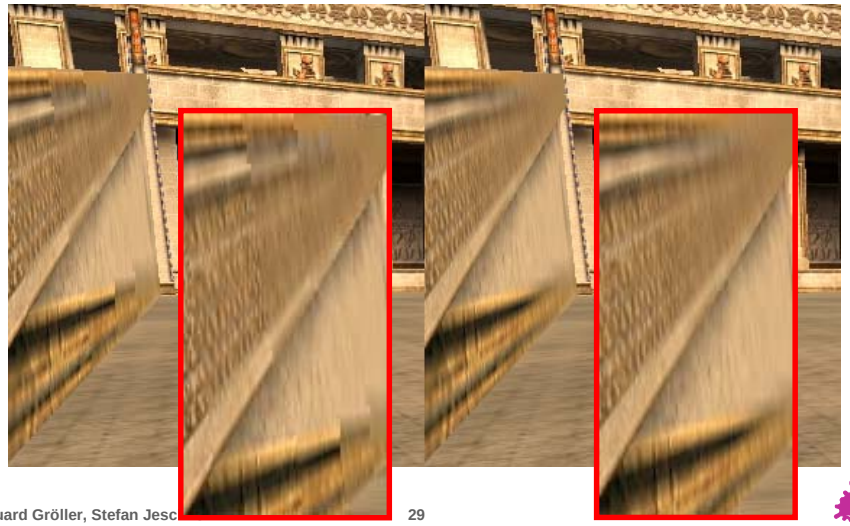
Anti-Aliasing: MIP Mapping

- Trilinear interpolation:
  - T_1 := value from texture $D_1 = D_0+1$ (bilin.interpolation)
  - Pixel value := $(D_1–D)\cdot T_0 + (D–D_0)\cdot T_1$
    - Linear interpolation between successive MIP Maps
  - Avoids "Mip banding" (but doubles texture lookups)



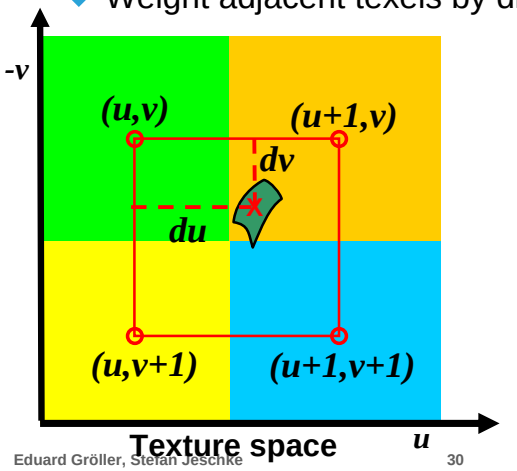
Anti-Aliasing: Mip Mapping (Example)

- Other example for bilinear vs. trilinear filtering



Anti-Aliasing

- Bilinear reconstruction for texture magnification ($D<0$) (*"upsampling"*)
  - Weight adjacent texels by distance to pixel position



$$T(u+du,v+dv) = du\cdot dv\cdot T(u+1,v+1) + du\cdot(1–dv)\cdot T(u+1,v) + (1-du)\cdot dv\cdot T(u,v+1) + (1-du)\cdot(1-dv)\cdot T(u,v)$$

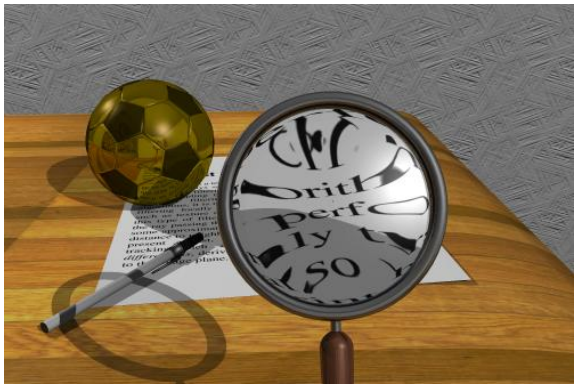
Anti-Aliasing (Bilinear Filtering Example)

Original image

Nearest neighbor

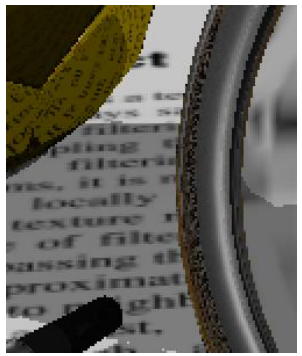
Bilinear filtering

Anti-Aliasing Examples (1/2)



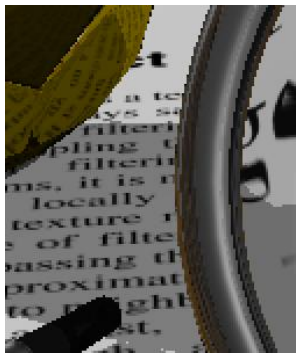
No mip-mapping

Anti-Aliasing Examples (2/2)



simple mip-mapping

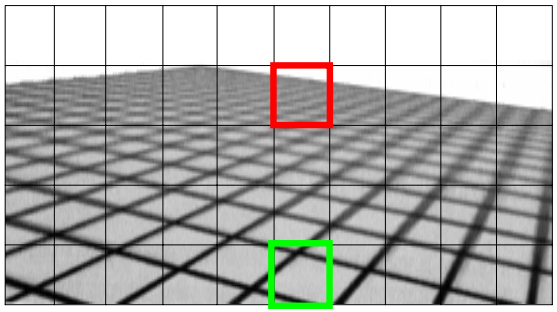
ray differentials

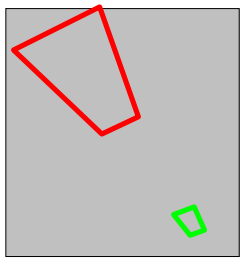


+ anisotropic filtering

Anti-Aliasing: Anisotropic Filtering

- Anisotropic Filtering
  - View dependent filter kernel
  - Implementation: *summed area table* (see above), "*RIP Mapping*", "*footprint assembly*"

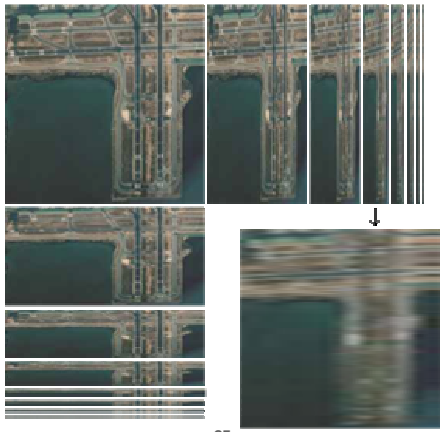




Texture space

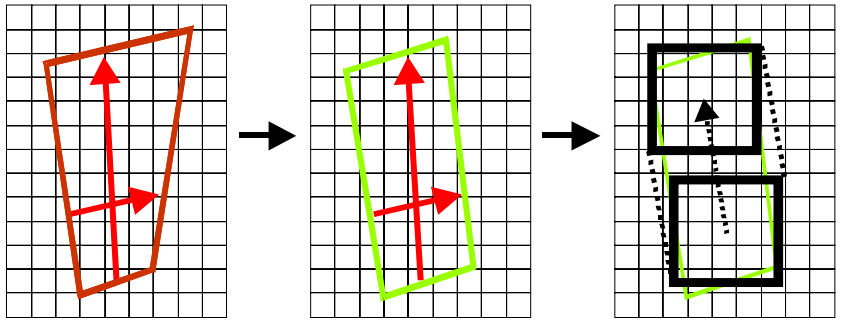
Anti-Aliasing: Anisotropic Filtering

- *"RIP Mapping"*: like MIP Maps but half resolution in x and y direction *separately*
  - Fast, but needs more memory than MIP Mapping



Anti-Aliasing: Anisotropic Filtering

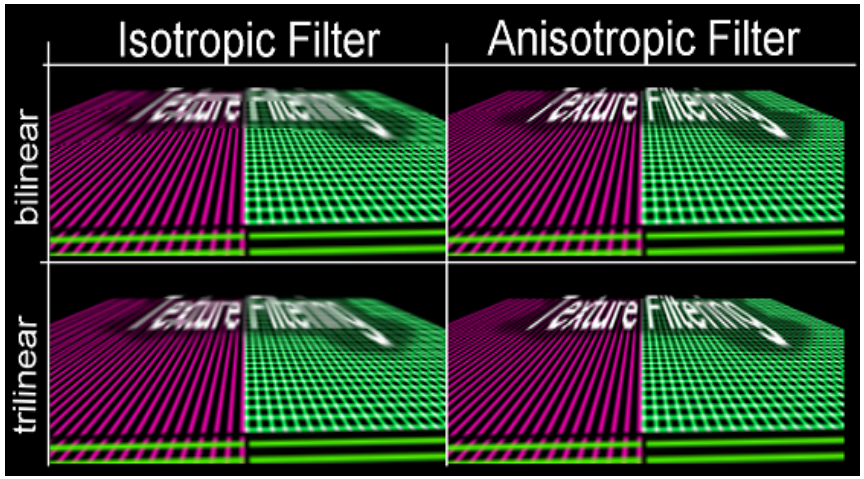
- *"Footprint assembly"*
  - Approximate pixel footprint by parallelogram
  - Sample along the parallelogram using MIP mapping



Footprint in texture space | **Approximation by parallelogram** | **Sampling using Mip Mapping**

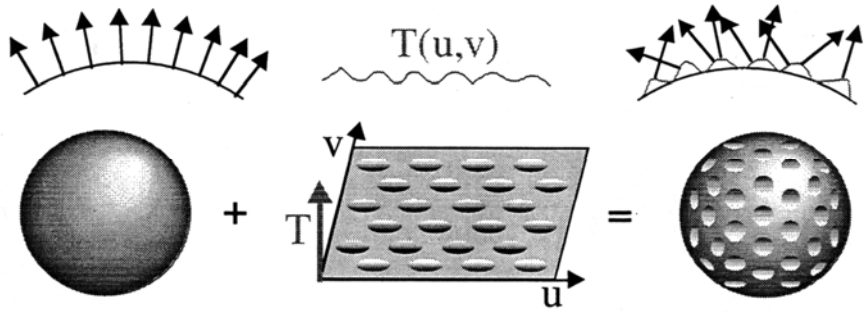
Anti-Aliasing: Anisotropic Filtering

- Example

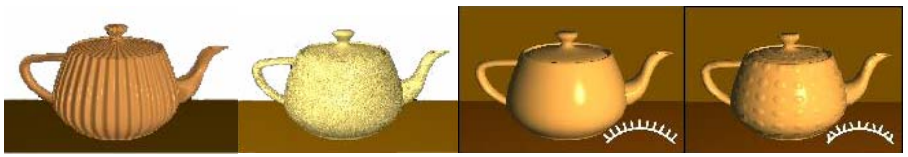


Bump Mapping

- Preprocess: compute normal vectors from height field
- Runtime: use computed normals for illumination

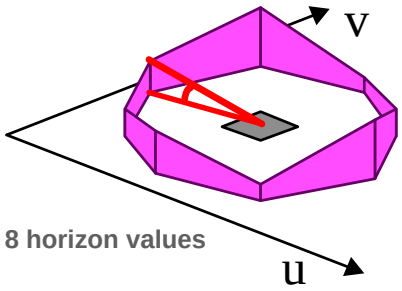


Bump Mapping Examples

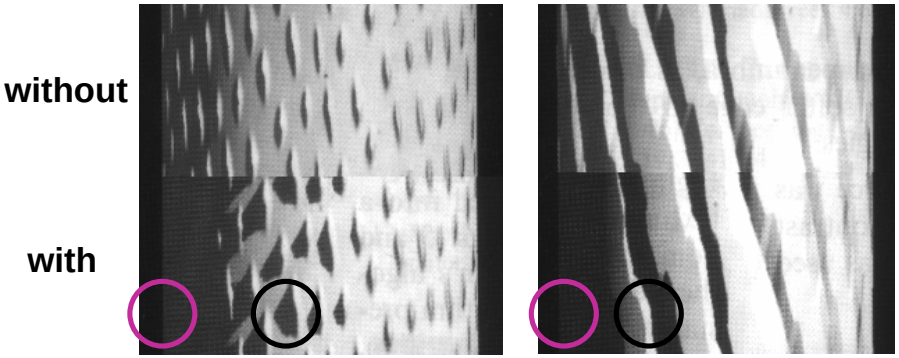


Horizon Mapping

- Improve bump mapping with (local) shadows
- Preprocess: compute n horizon values per texel
- Runtime:
  - Interpolate horizon values
  - Shadow accordingly



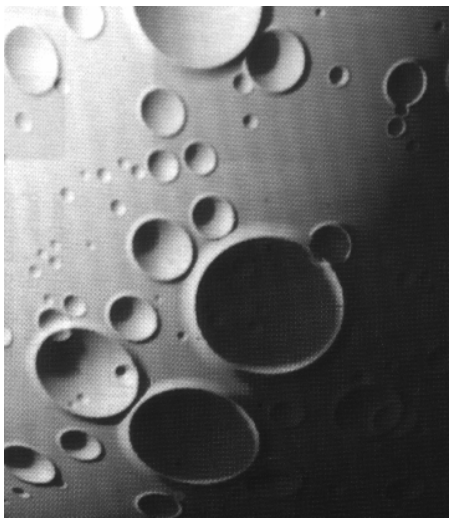
Horizon Mapping Examples (1/2)

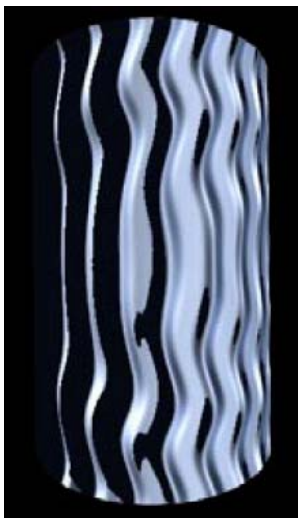


Shadows from bumps

No light on rear side of the object

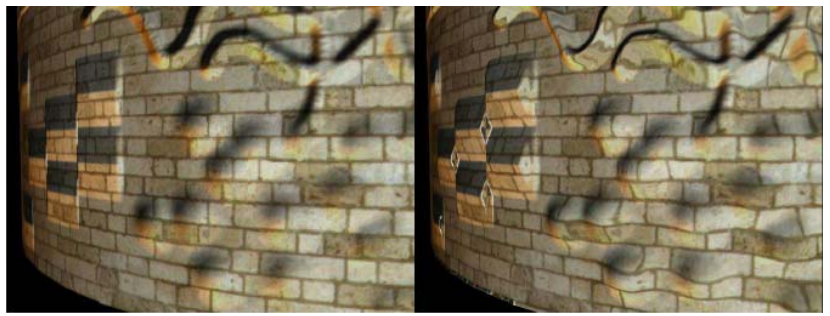
Horizon Mapping Examples (2/2)





Parallax Mapping

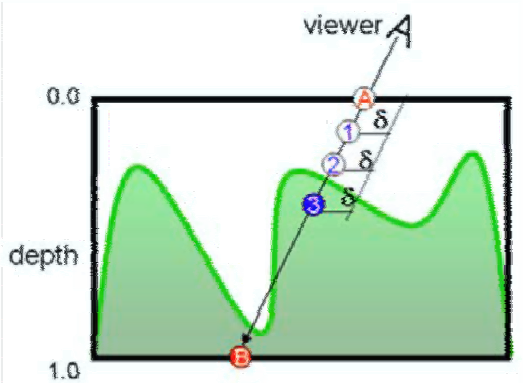
- Parallax: apparent movement of close objects in front of further objects
  - Preprocess: calculate *"texture coordinate shifts"*
  - Runtime: “crd. shifts” alter the texture lookup position
    - Problem: occlusions cannot be modelled

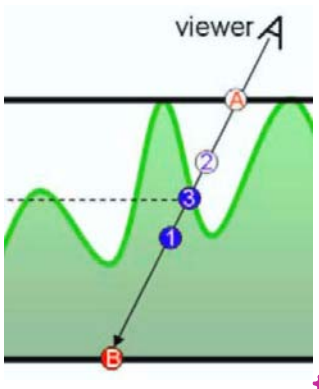


Bump mapping **Parallax + Bump mapping**

Relief Mapping

- At runtime: perform ray casting in the pixel shader
  - Calculate entry (A) and exit point (B)
  - March along ray until intersection with height field is found
  - Binary search to refine the intersection position





Relief Mapping Examples (1)

Texture mapping

Parallax mapping

Relief mapping

Relief Mapping Examples (2)

Links

- http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_2.htm
- http://www.3dcenter.org/artikel/grafikfilter/index3.php