EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

Computer Graphics (Graphische Datenverarbeitung)

- Textures & Filtering -

WS 2021/2022

Corona

- Regular random lookup of the 3G certificates

- Contact tracing: We need to know who is in the class room
  - New ILIAS group for every lecture slot
  - Register via ILIAS or this QR code (only if you are present in this room)

Overview

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

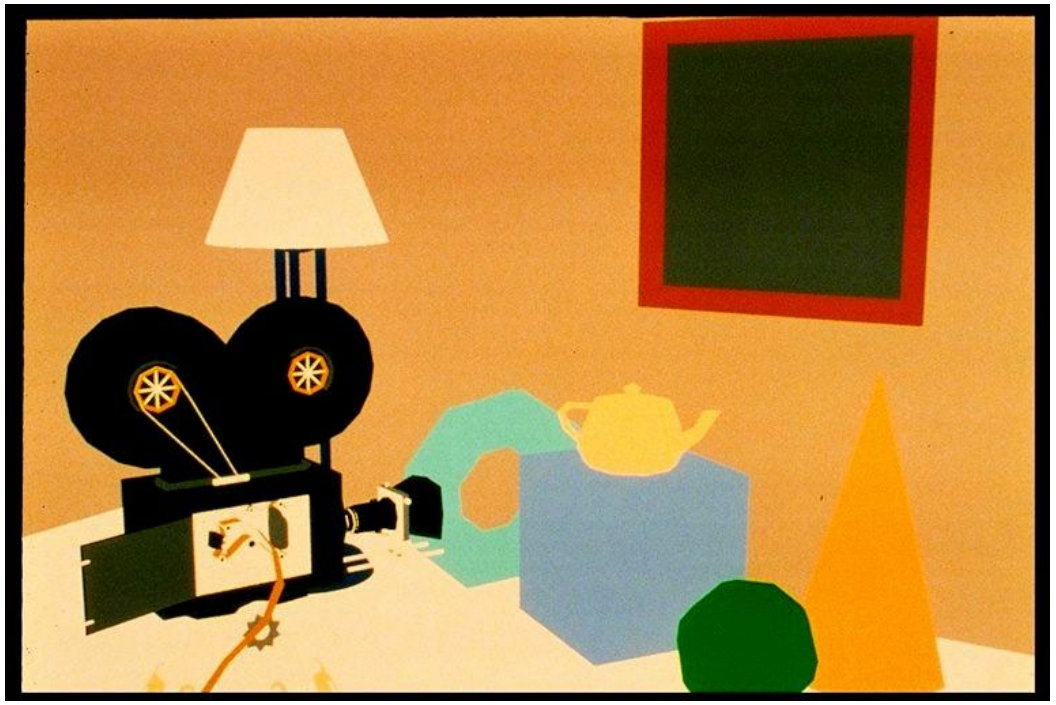
- Last time
  - BRDFs

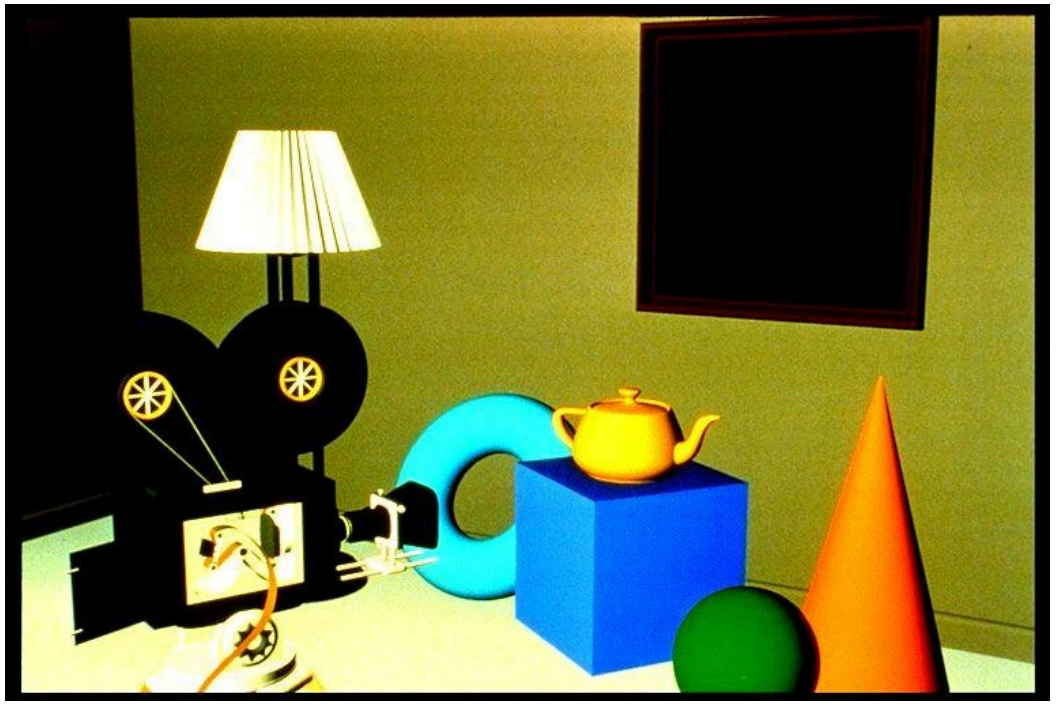
- Today
  - Textures to modify surface properties
  - Texture Parameterization
  - Procedural Shading
  - Texturing Filtering

Texturing

How can we make surfaces non-uniform

Simple Illumination

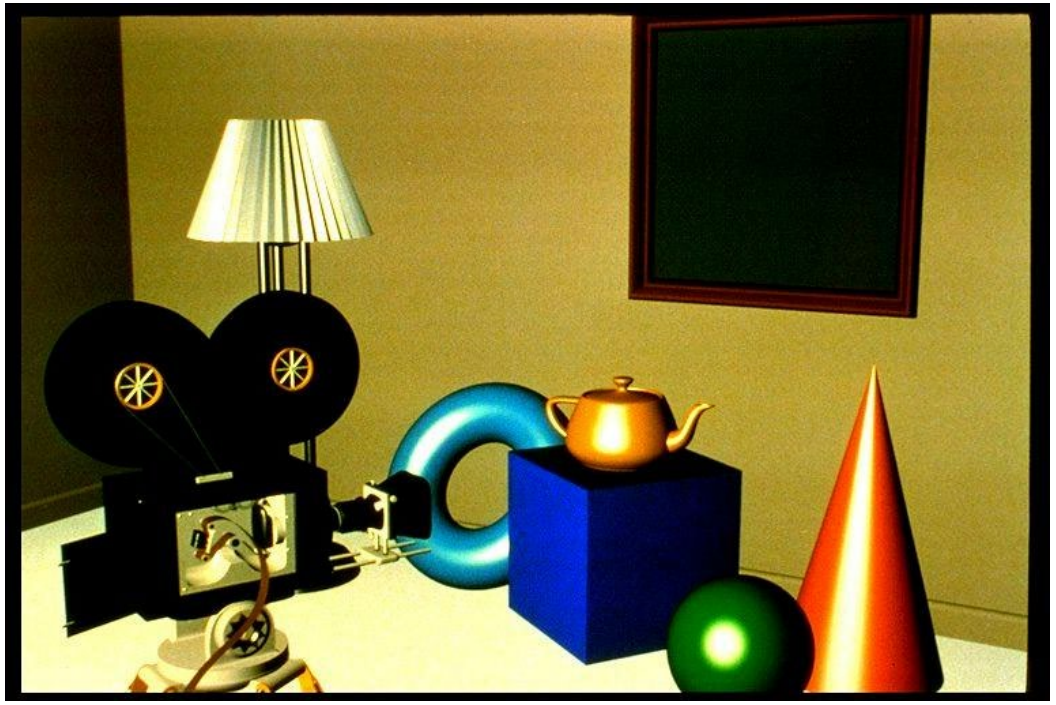




- No illumination
- Constant colors

- Parallel light
- Diffuse reflection

Standard Illumination



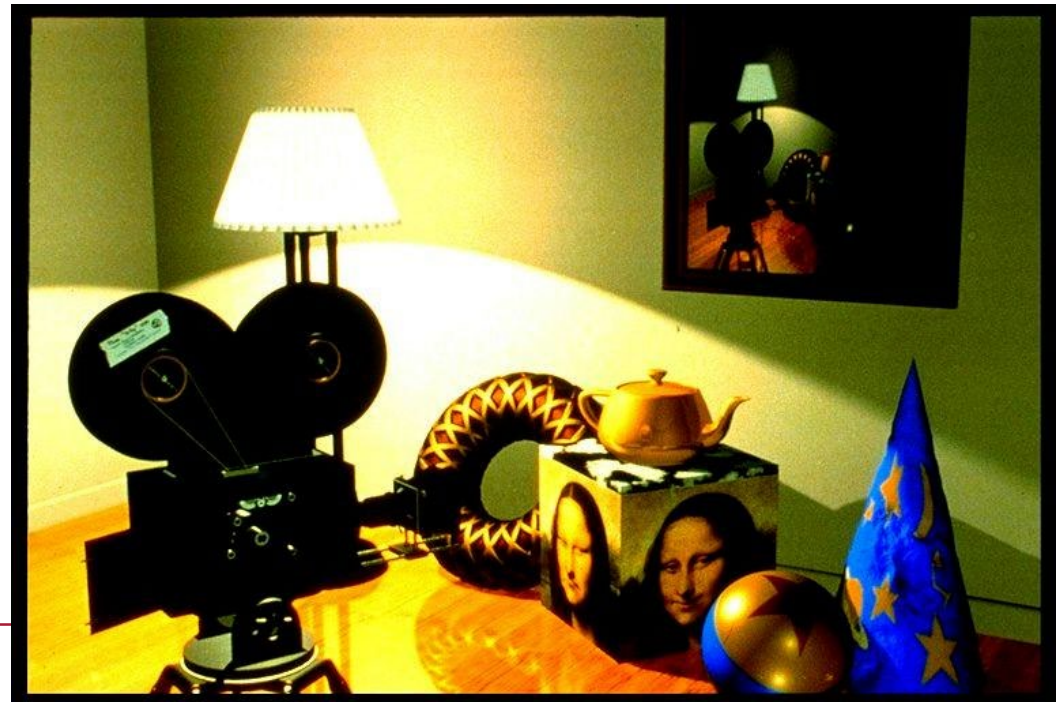
- Parallel light
- Specular reflection

- Multiple local light sources
- Different BRDFs

Object properties constant over surface

Texturing

- Locally varying object characteristics
- 2D Image Textures
- Shadows
- Bump-Mapping
- Reflection textures

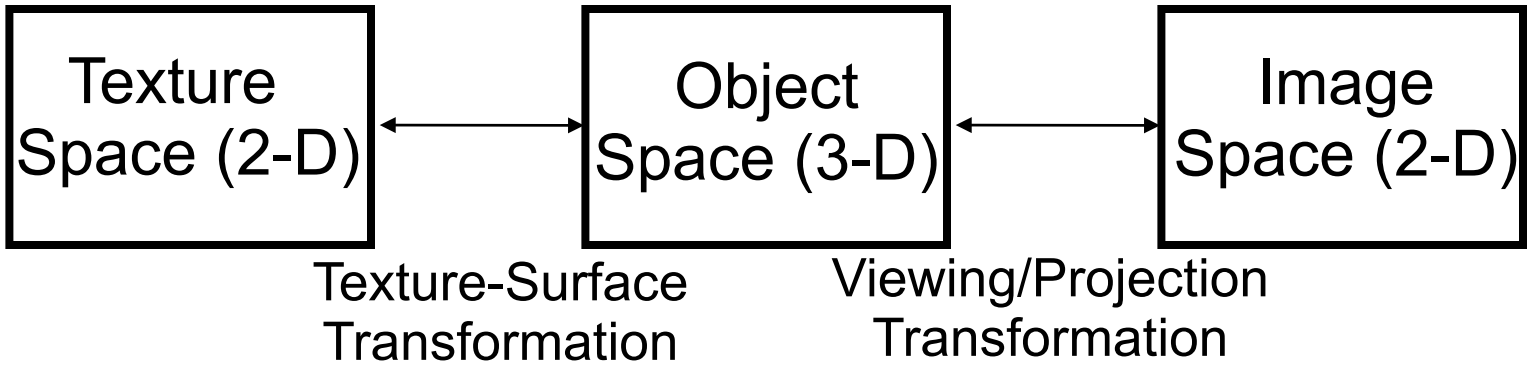


Texture-modulated Quantities

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- 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
  - virtually moving point vs. changing normal
  - Bump mapping or Normal mapping
- Geometry
  - P= P + dP
  - Displacement mapping
- Distant illumination
  - Environment mapping, Reflection mapping

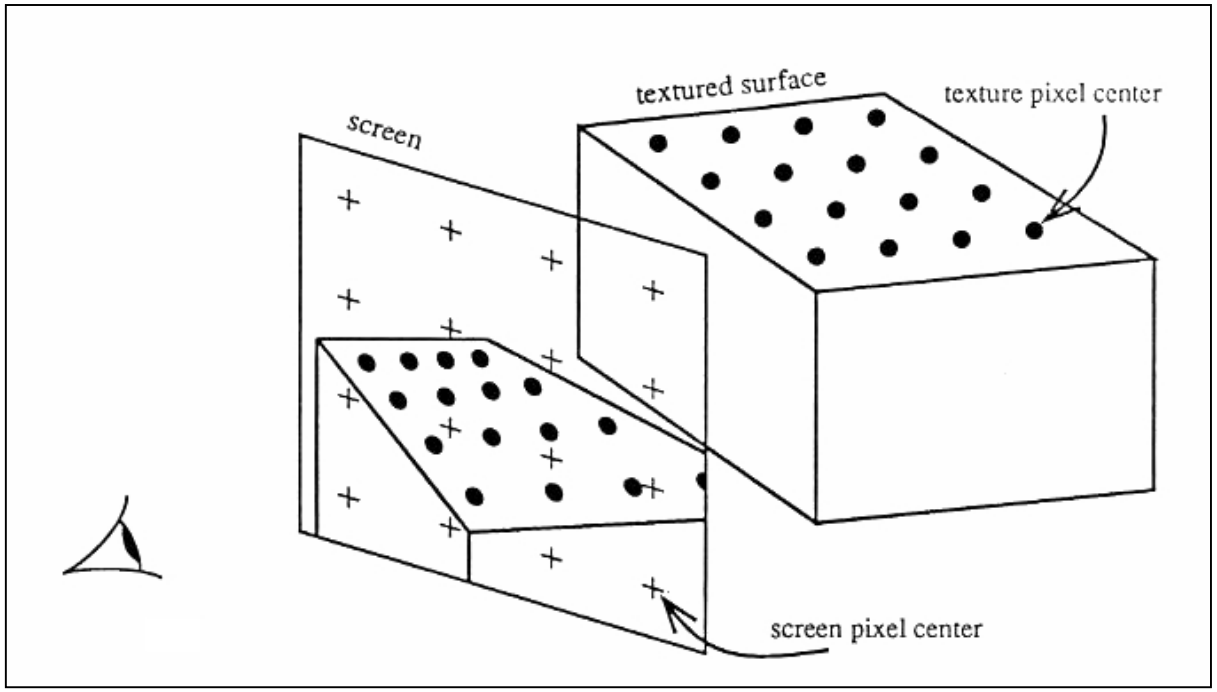
Texture Mapping Transformations

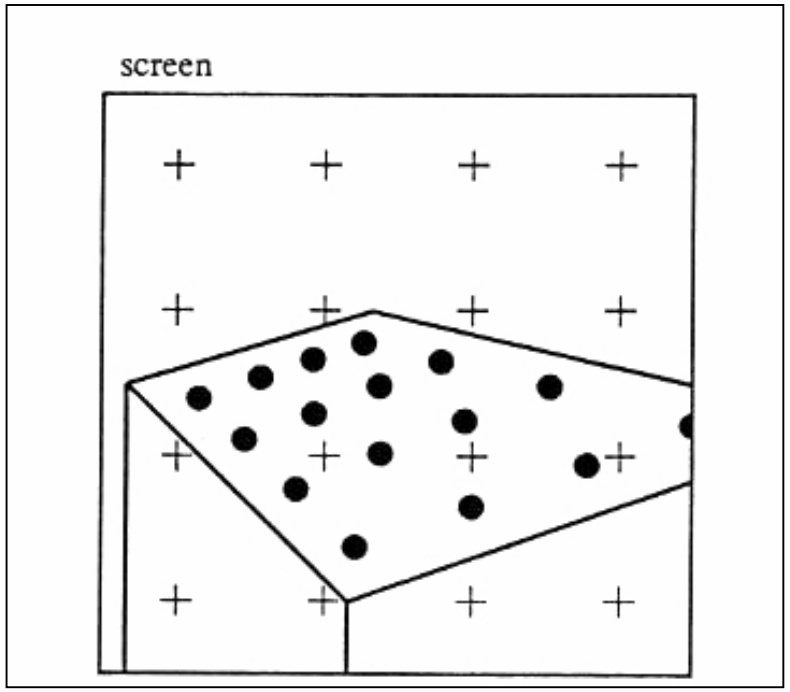


The texture is mapped onto a surface in 3-D object space, which is then mapped to the screen by the viewing projection. These two mappings are composed to find the overall 2-D texture space to 2-D image space mapping, and the intermediate 3-D space is often forgotten. This simplification suggests texture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

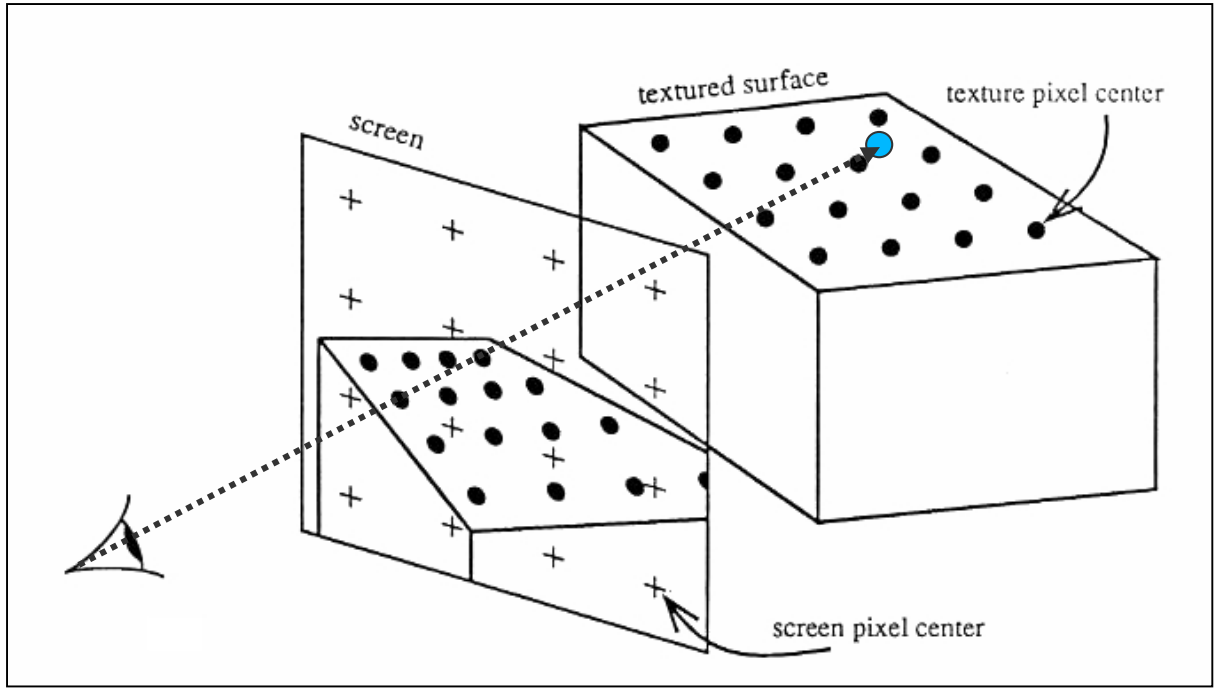
2D Texturing

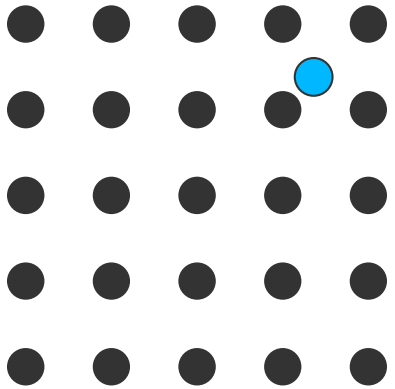




- 2D texture mapped onto object
- Object projected onto 2D screen
- 2D → 2D: warping operation
- Uniform sampling ?
- Hole-filling/blending ?

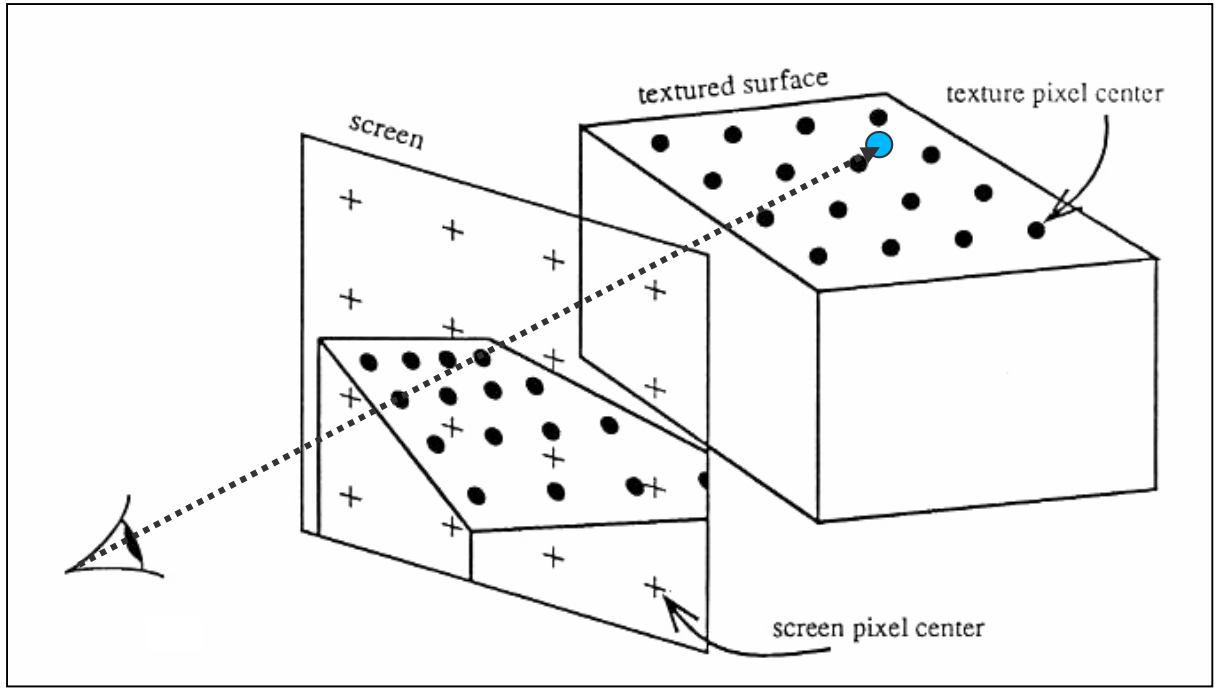
Texture Mapping in a Ray Tracer

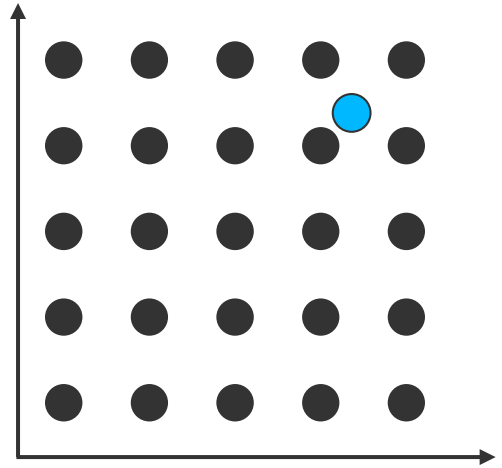




- approximation:
  - ray hits surface
  - surface location corresponds to coordinate inside a texture

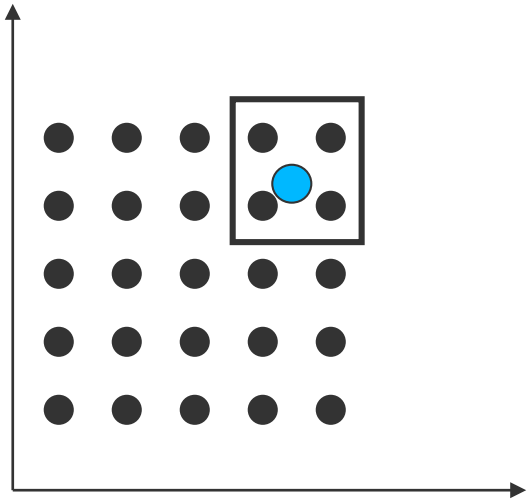
Texture Mapping in a Ray Tracer

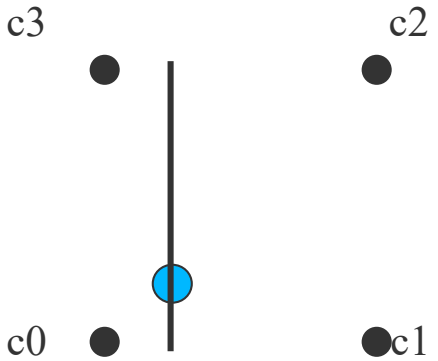




- approximation:
  - ray hits surface
  - surface location corresponds to coordinate inside a texture

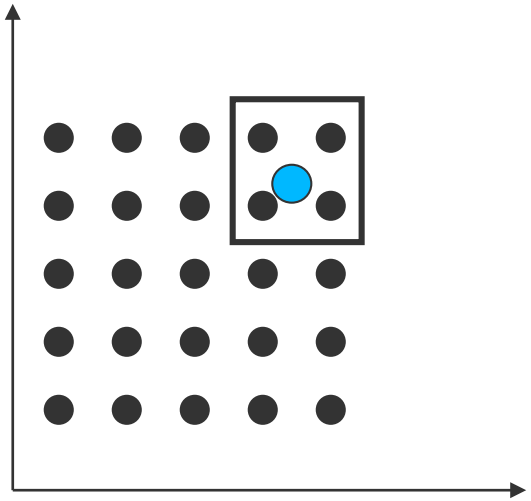
Interpolation 1D

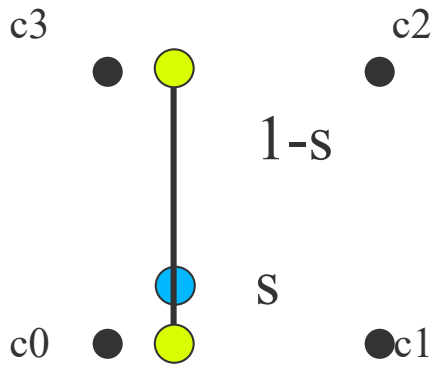




- How to interpolate the color of the pixel?

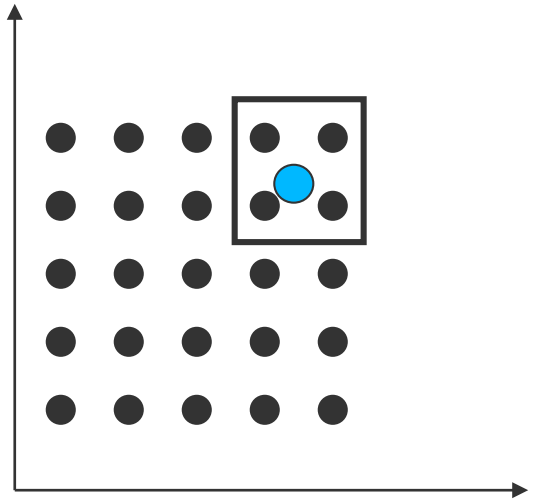
Interpolation 1D

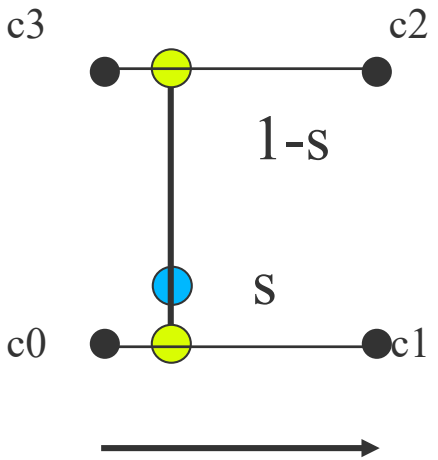




- How to interpolate the color of the pixel?

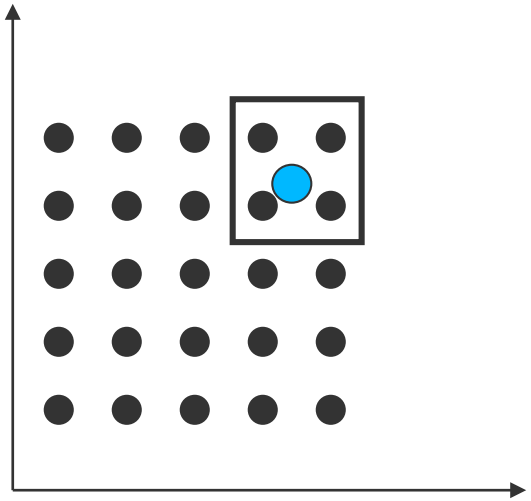
Interpolation 2D

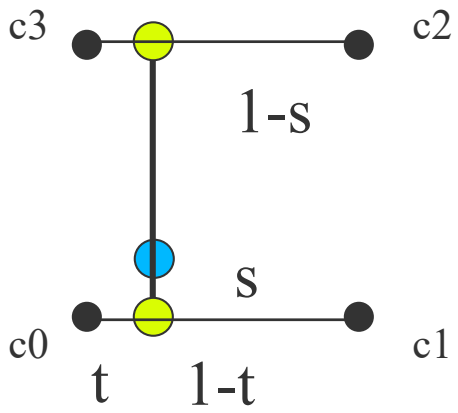




- How to interpolate the color of the pixel?

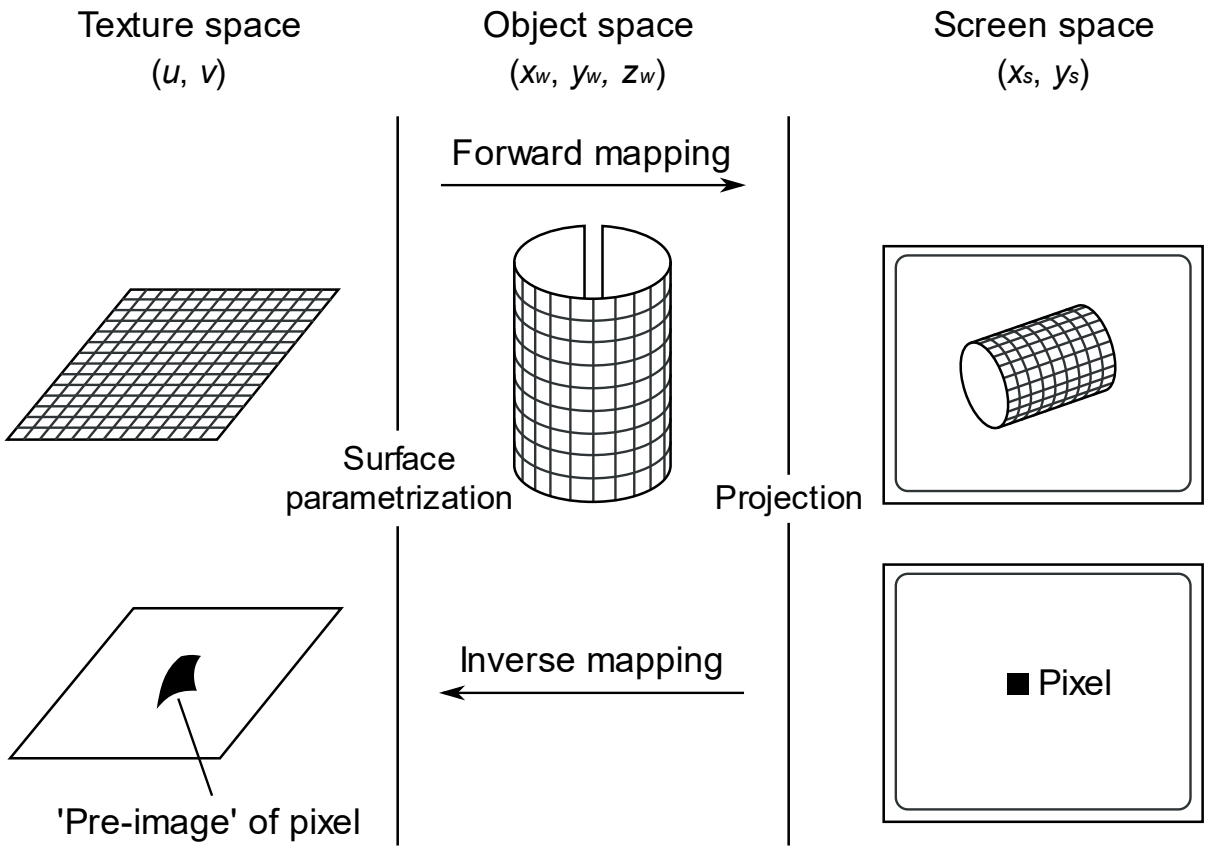
Interpolation 2D





- How to interpolate the color of the pixel?
- Lookup of four surrounding points
- 2 x 1D: i0 = (1-t)c0 + tc1
  i1 = (1-t)c3 + tc2
- 1 x 2D: c = (1-s) i0 + s i1

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

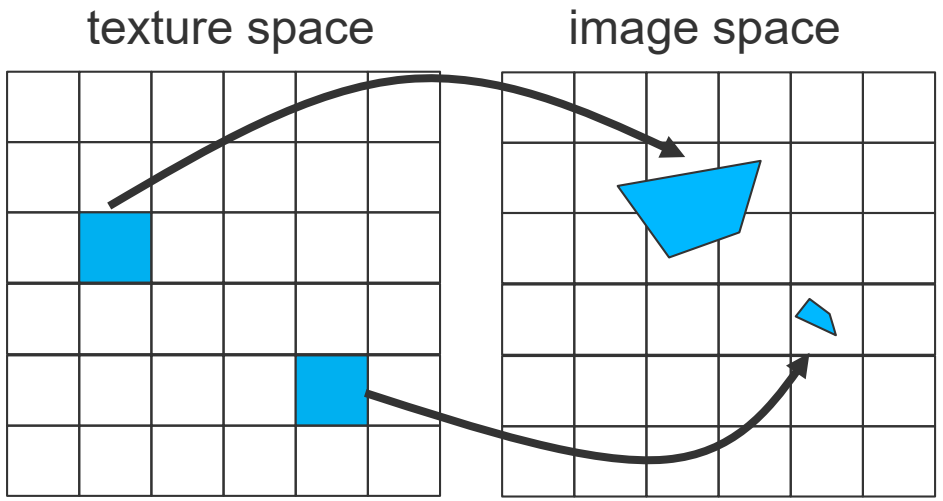
Forward Mapping

- 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:

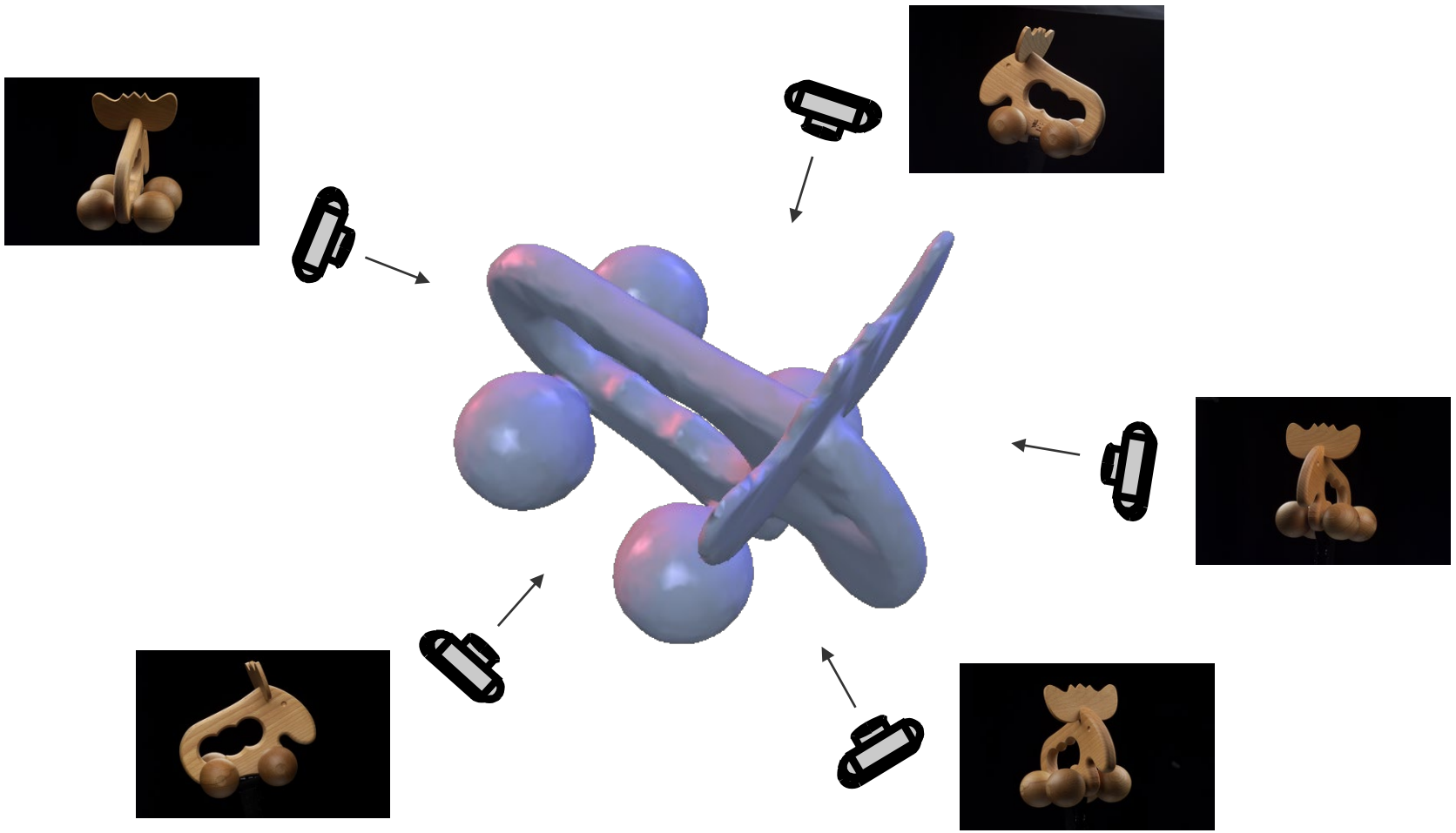
```
for v
  for u
    compute x(u,v) and y(u,v)
    copy TEX[u,v] to SCR[x,y]
```

(or in general
rasterize image of TEX[u,v])

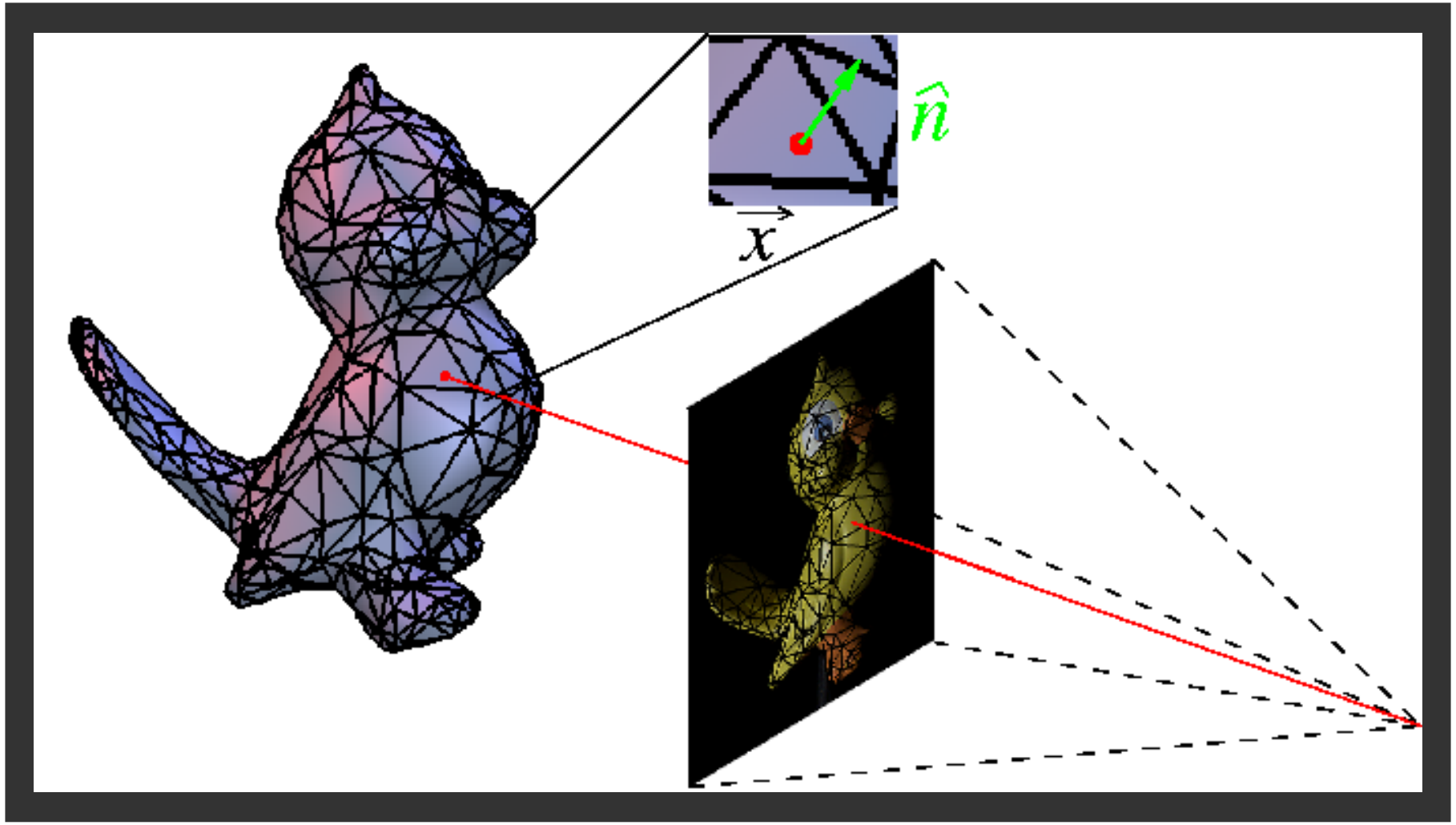


Texturing with real images

- Find the camera settings for each 2D image.



Resampling



- Image to surface mapping is given by projective mapping from camera position

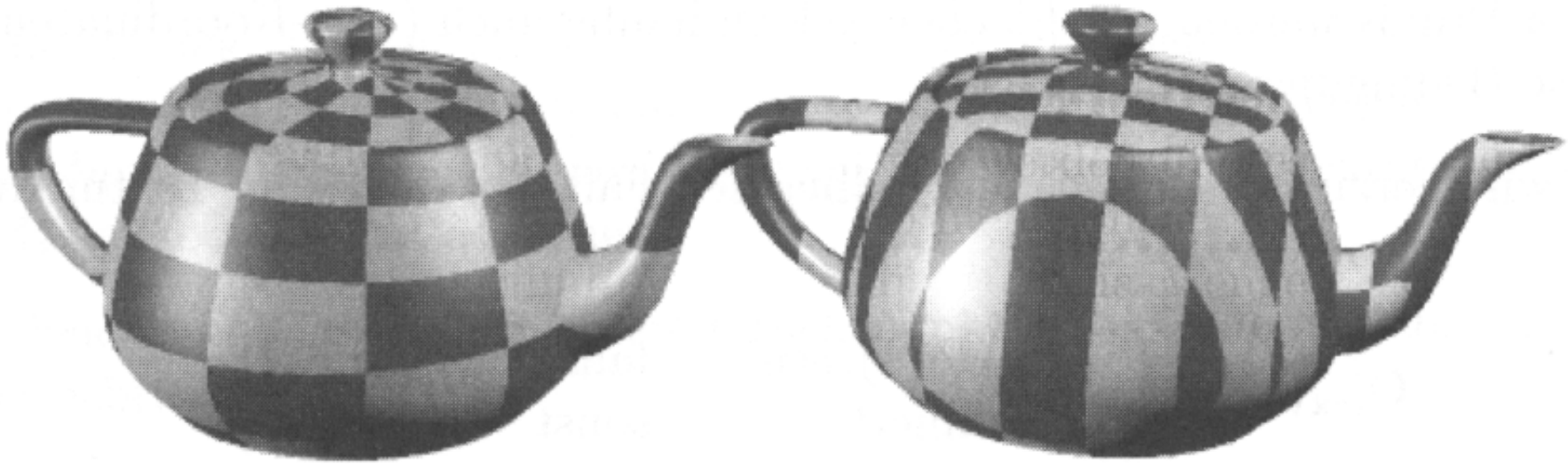
Merged Texture

Texture Parameterization

How to map the texture image onto the geometry?

Surface Parameterization

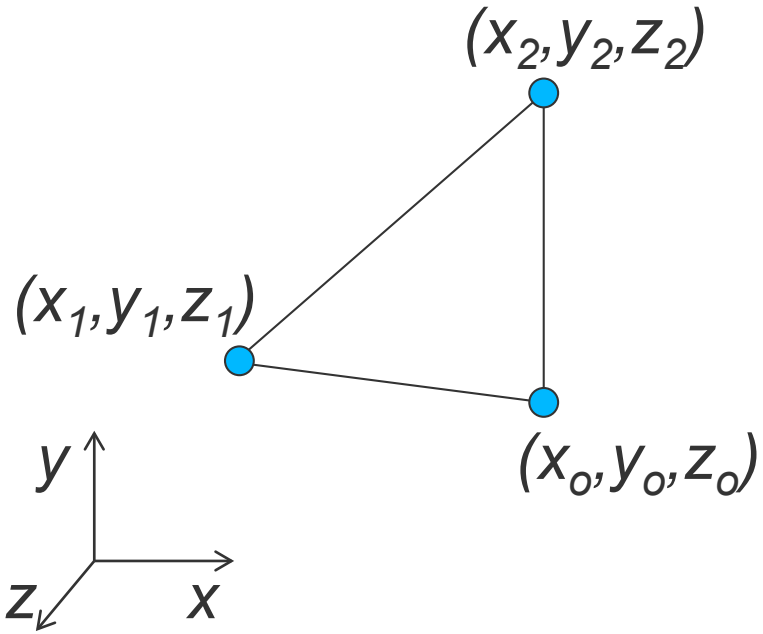
- To apply textures we need 2D coordinates on surfaces
  ➔ Parameterization
- Some objects have a natural parameterization
  - Sphere: spherical coordinates $(\varphi, \theta) = (2\pi\, u, \pi\, v)$
  - Cylinder: cylindrical coordinates $(\varphi, z) = (2\,\pi\, u, H\, v)$
  - Parametric surfaces (such as B-spline or Bezier surfaces → later)
- Parameterization is less obvious for
  - Polygons, implicit surfaces, …

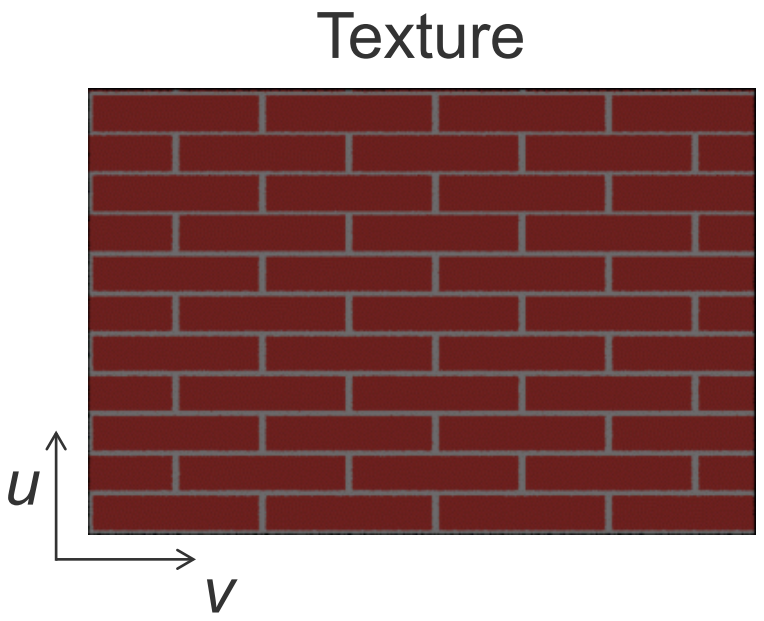


Triangle Parameterization

- Triangle is a planar object
  - Has implicit parameterization (e.g. barycentric coordinates)
  - But we need more control: Placement of triangle in texture space

- Assign texture coordinates (u_i, v_i) to each vertex $(x,y,z)_i$

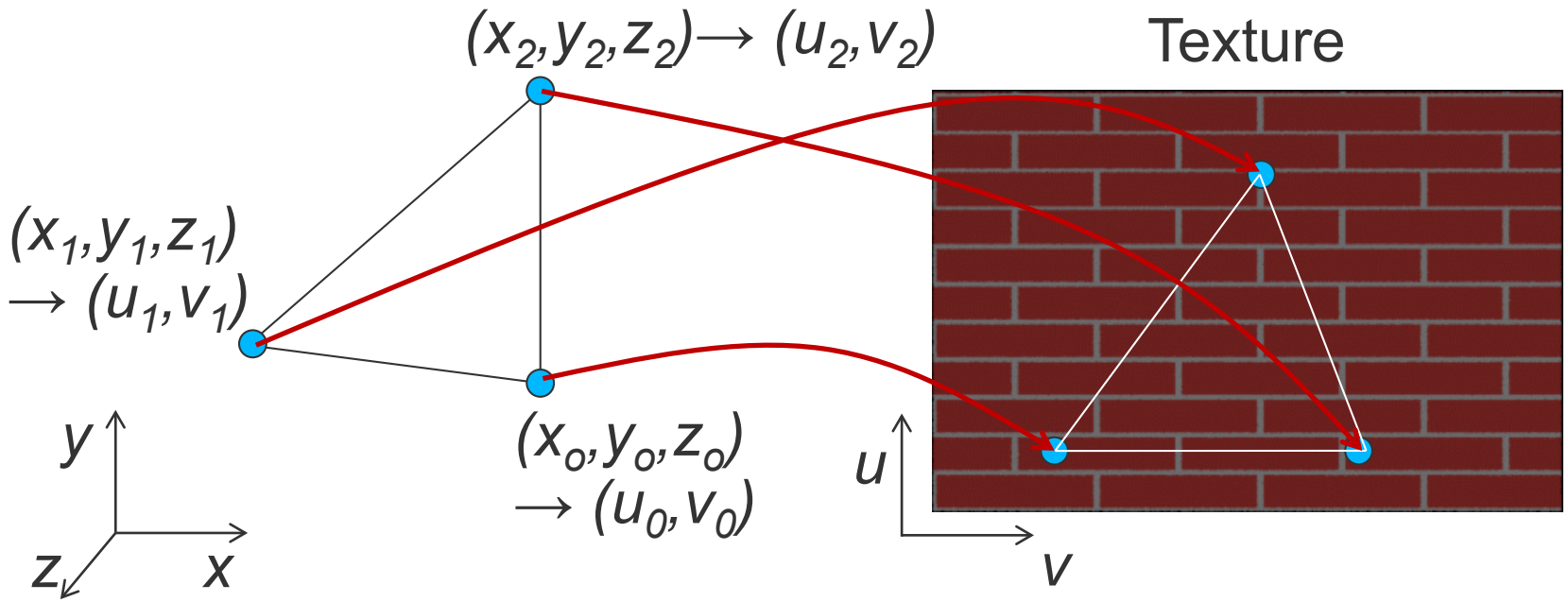




Triangle Parameterization

- Triangle is a planar object
  - Has implicit parameterization (e.g. barycentric coordinates)
  - But we need more control: Placement of triangle in texture space

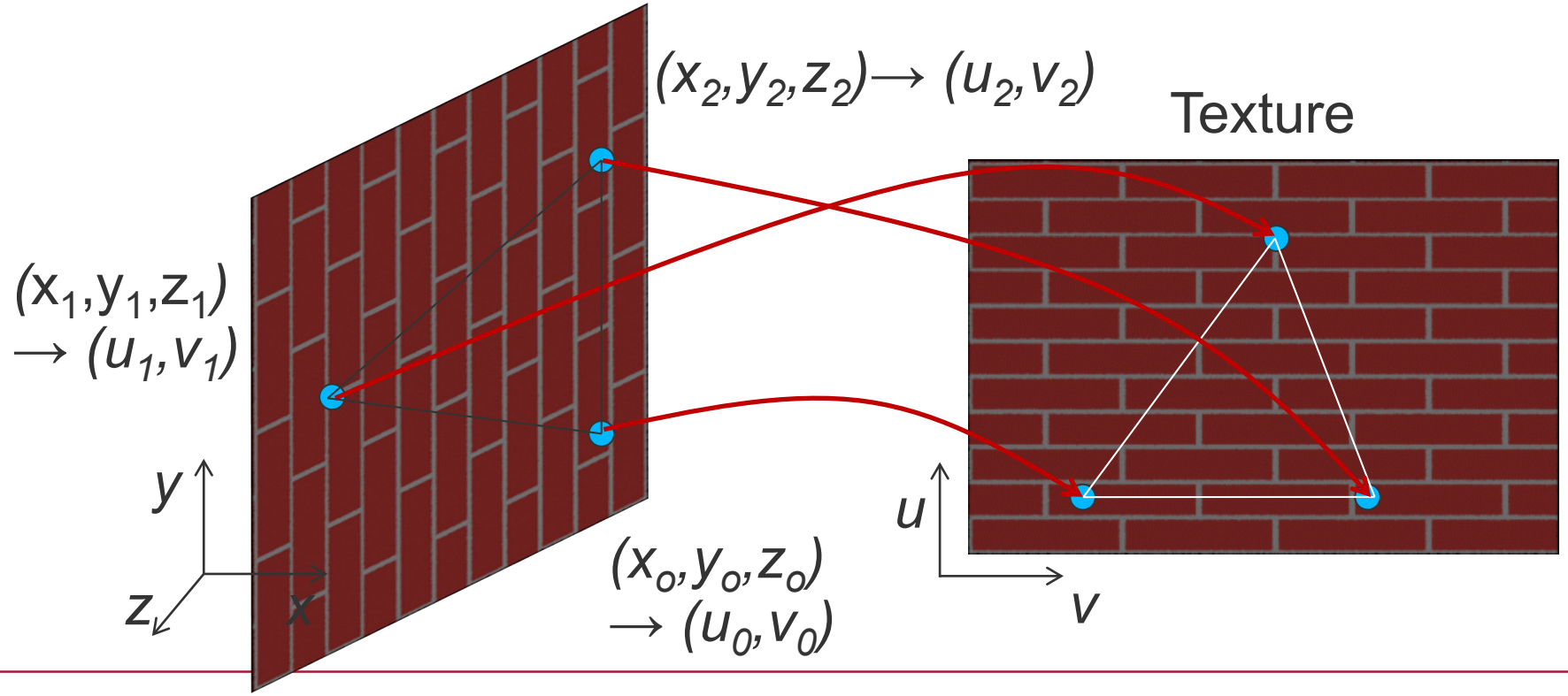
- Assign texture coordinates (u_i,v_i) to each vertex $(x,y,z)_i$



Triangle Parameterization

- Triangle is a planar object
  - Has implicit parameterization (e.g. barycentric coordinates)
  - But we need more control: Placement of triangle in texture space

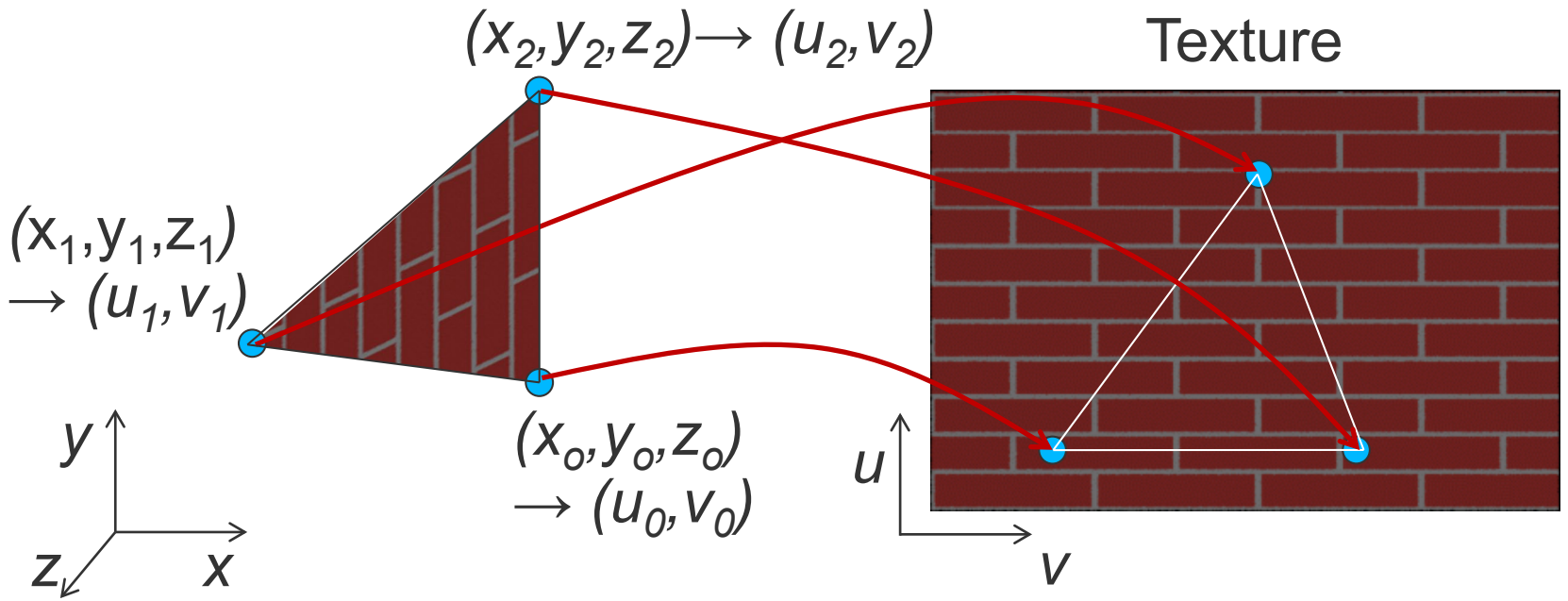
- Assign texture coordinates (u_i, v_i) to each vertex $(x,y,z)_i$



Triangle Parameterization

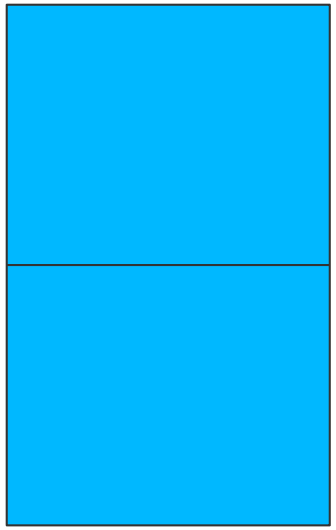
- Triangle is a planar object
  - Has implicit parameterization (e.g. barycentric coordinates)
  - But we need more control: Placement of triangle in texture space

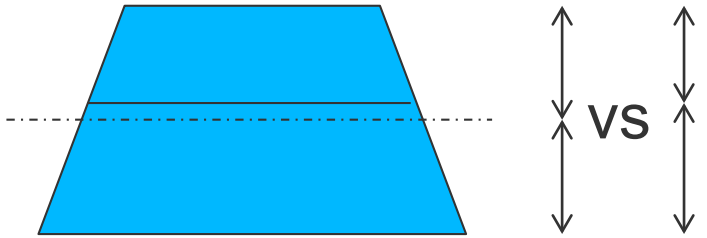
- Assign texture coordinates (u_i,v_i) to each vertex $(x,y,z)_i$



Influence of Perspective

- Due to perspective transformation the front patch is larger than the back patch. The center line of the texture is not at the center between top and bottom: ~ 1/z





orthographic perspective projection

- Need to calculate texture coordinate in 3D, then project!
  - no problem for a ray tracer

Triangle Parameterization

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Direct Perspective mapping
- Assign texture coordinates (u,v) to each vertex (x,y,z)
- Apply viewing projection (x,y,z) → (x,y)
- Yields full texture transformation (warping) (u,v)→(x,y)

$$x = \frac{au + bv + c}{gu + hv + i} \qquad y = \frac{du + ev + f}{gu + hv + i}$$

- In homogeneous coordinates (by embedding (u,v) as (u',v',1))

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} u' \\ v' \\ q \end{bmatrix}$$

$$(x, y) = (x'/w, y'/w)$$

$$(u, v) = (v'/q, v'/q)$$

- Transformation coefficients determined by 3 pairs (u,v)→(x,y)
  - Three linear equations
  - Invertible iff neither set of points is collinear

Triangle Parameterization II

- Given

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} u' \\ v' \\ q \end{bmatrix}$$

- the inverse transform (x,y)→(u,v) is defined as

$$\begin{bmatrix} u' \\ v' \\ q \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} \qquad \begin{bmatrix} u' \\ v' \\ q \end{bmatrix} = \begin{bmatrix} ei-fh & ch-bi & bf-ce \\ fg-di & ai-cg & cd-af \\ dh-eg & bg-ah & ae-bd \end{bmatrix} \begin{bmatrix} x' \\ y' \\ w \end{bmatrix}$$

- Important to account for perspective transformation!
- Coefficients must be calculated for each triangle
  - Rasterization
    - Incremental bilinear update of (u',v',q) in screen space
    - Using the partial derivatives of the linear function (i.e. constants)
  - vs. Ray tracing
    - Simply interpolate texture coordinates using barycentric coordinates of intersection point

Cylinder Parameterization

- Transformation from texture space to the cylinder parametric representation can be written as:

$$(\theta, h) = (2\pi u, vH)$$

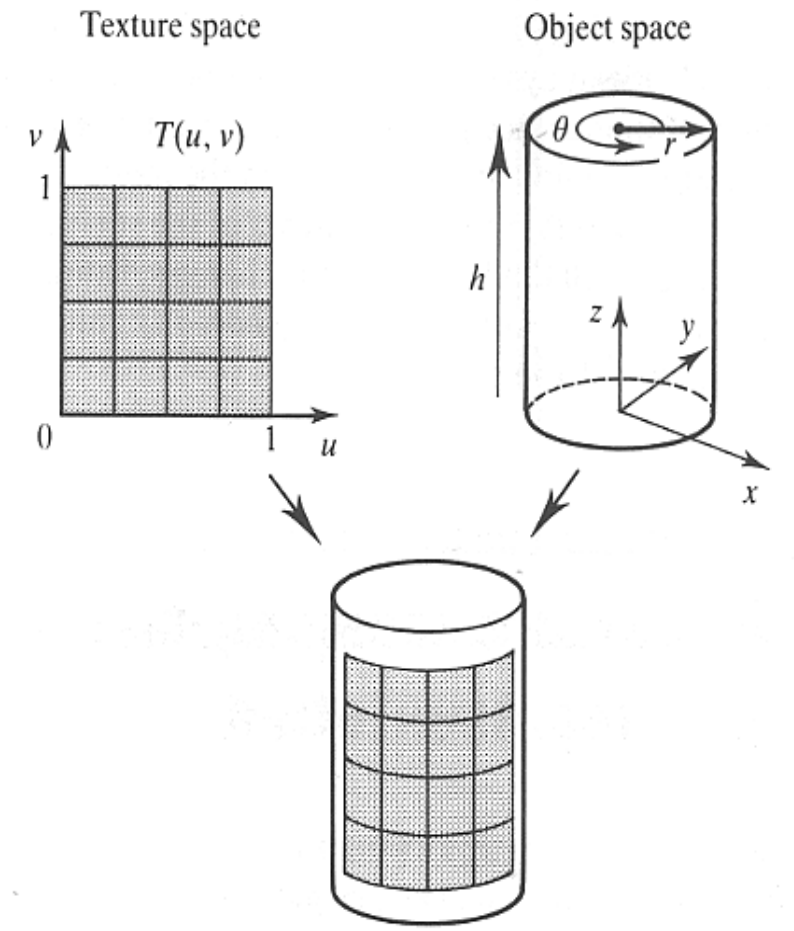
where *H* is the height of the cylinder.

- The surface coordinates in the Cartesian reference frame can be expressed as:

$$x_o = r\cos\theta$$

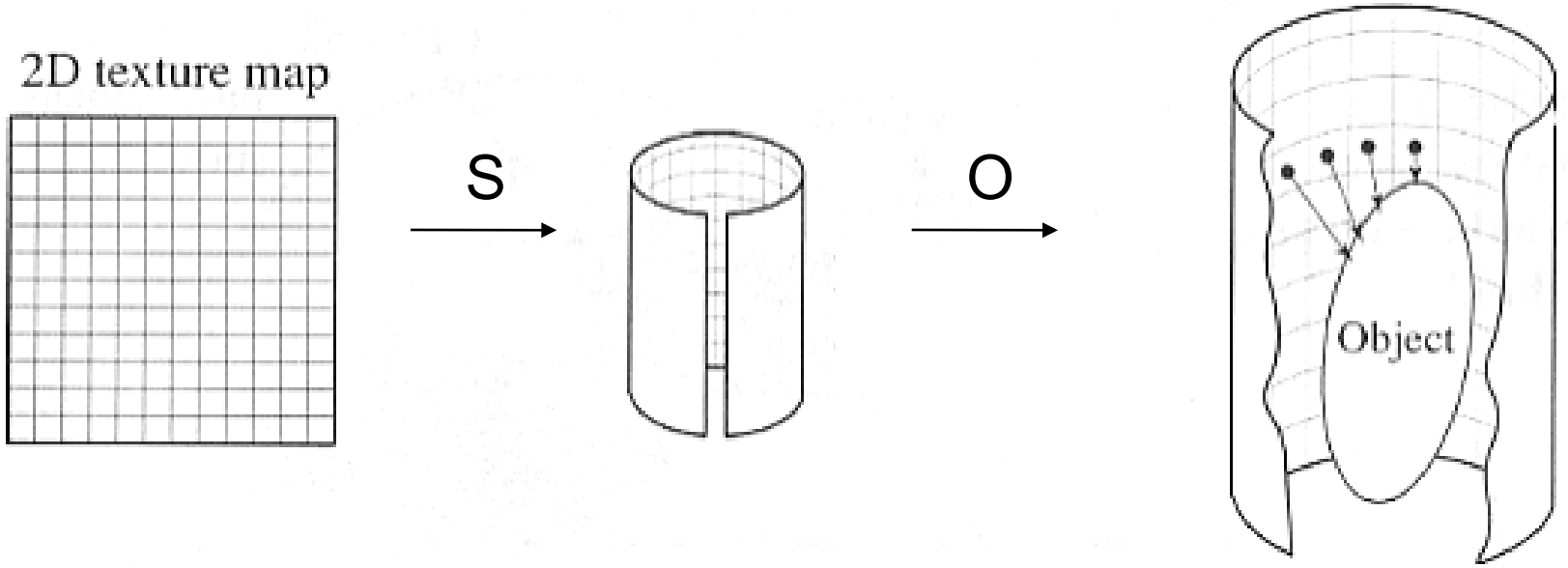
$$y_o = r\sin\theta$$

$$z_o = h$$



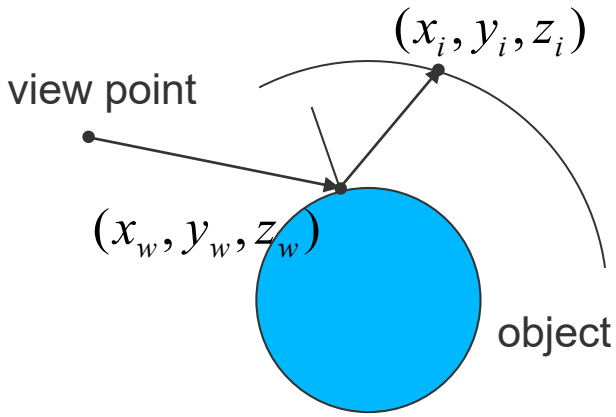
Two-Stage Mapping

- Inverse Mapping for arbitrary 3D surfaces too complex
- Approximation technique is used:
  - Mapping from 2D texture space to a simple 3D intermediate surface (S mapping)
    - Should be a reasonable approximation of the destination surface
    - E.g.: plane, cylinder, sphere, cube, ...
  - Mapping from the intermediate surface to the destination object surface (O mapping)

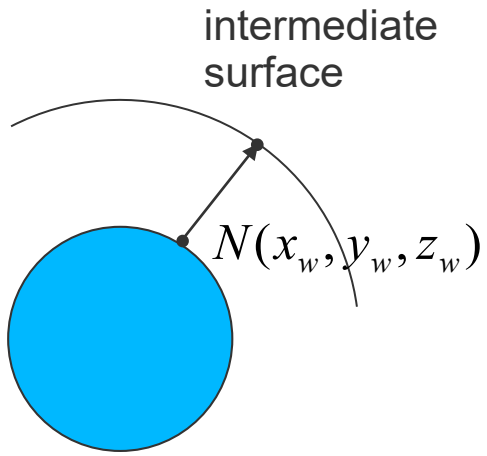


O-Mapping

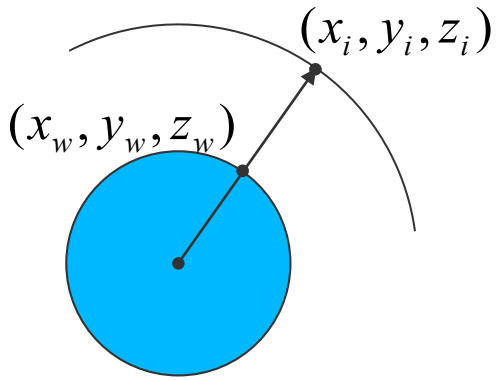
- Determine point on intermediate surface through
  - Reflected view ray
    - Reflection or environment mapping
  - Normal mapping
  - Line through object centroid
  - Shrinkwrapping
    - Forward mapping
    - Normal mapping from intermediate surface



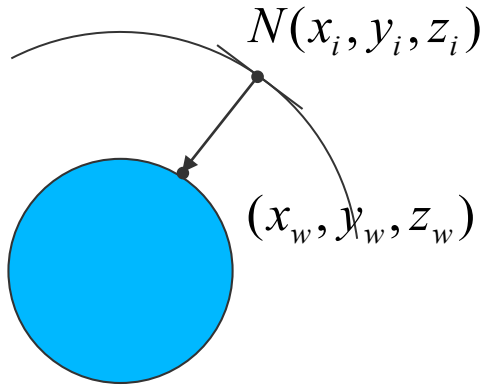
reflected ray



object normal



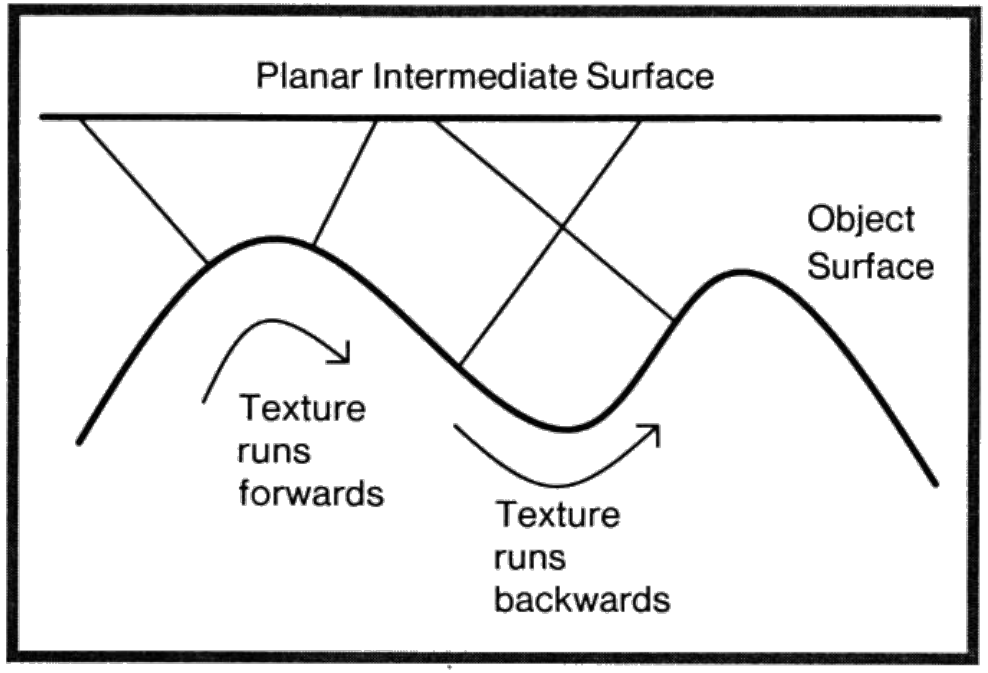
object centroid



intermediate surface normal

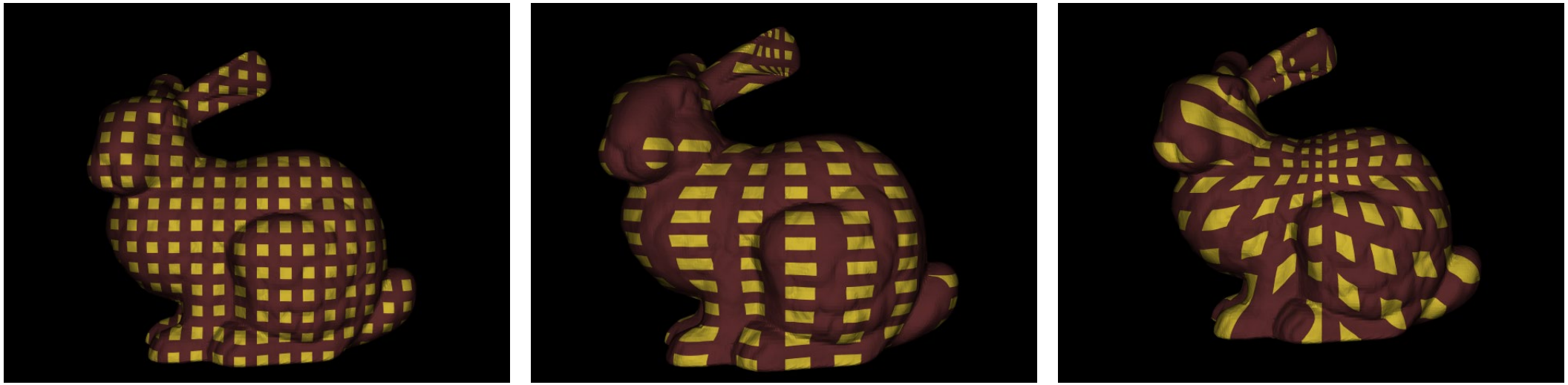
Two-Stage Mapping: Problems

- Problems
  - May introduce undesired texture distortions if the intermediate surface differs too much from the destination surface
  - Still often used in practice because of its simplicity



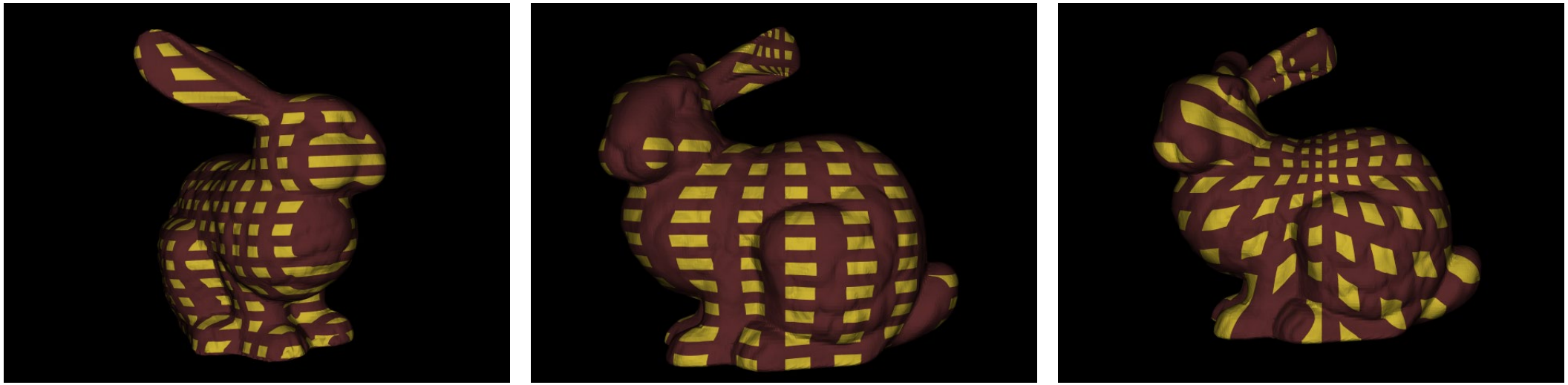
Surface concavities can cause the texture pattern to reverse if the object normal mapping is used.

Two-Stage Mapping: Example



- Different intermediate surfaces
- Plane
  - Strong distortion where object surface normal ≠ plane normal
- Cylinder
  - Reasonably uniform mapping (symmetry !)
- Sphere
  - Problems with concave regions

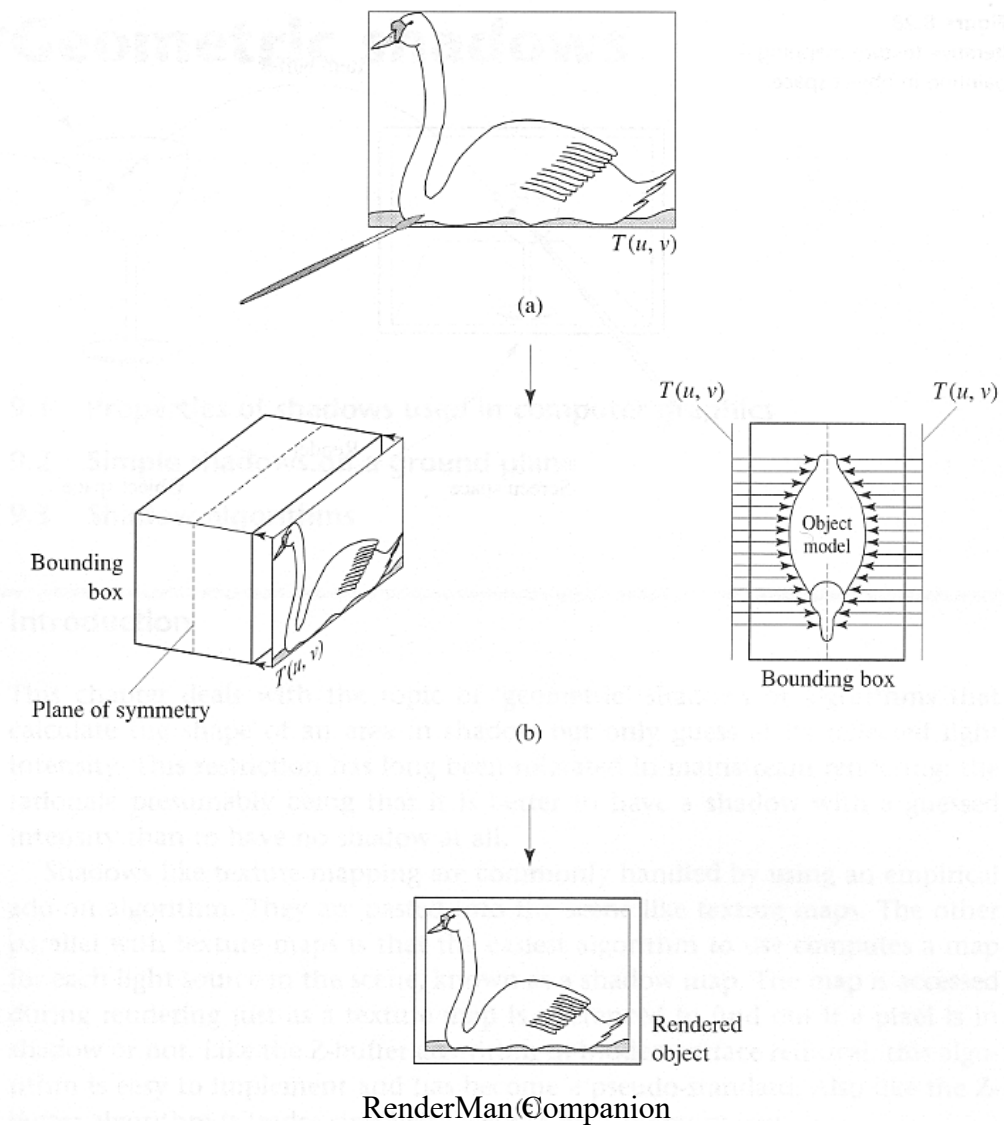
Two-Stage Mapping: Example



- Different intermediate surfaces
- Plane
  - Strong distortion where object surface normal ≠ plane normal
- Cylinder
  - Reasonably uniform mapping (symmetry !)
- Sphere
  - Problems with concave regions

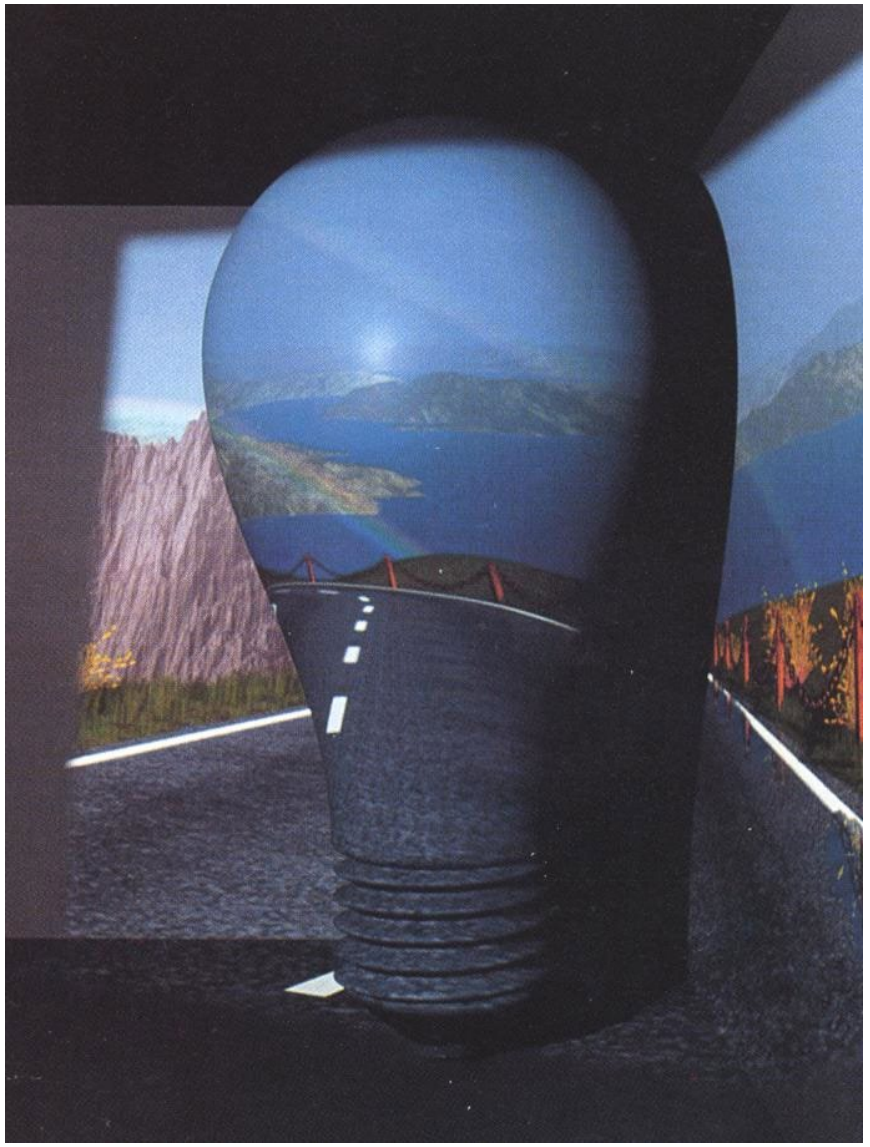
Projective Textures

- Project texture onto object surfaces
  - Slide projector
- Parallel or perspective projection
- Use photographs as textures
- Multiple images
  - View-dependent texturing
- Perspective Mapping



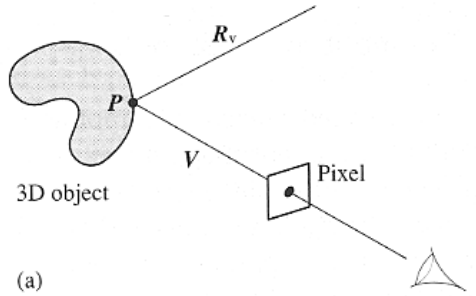
RenderMan Companion

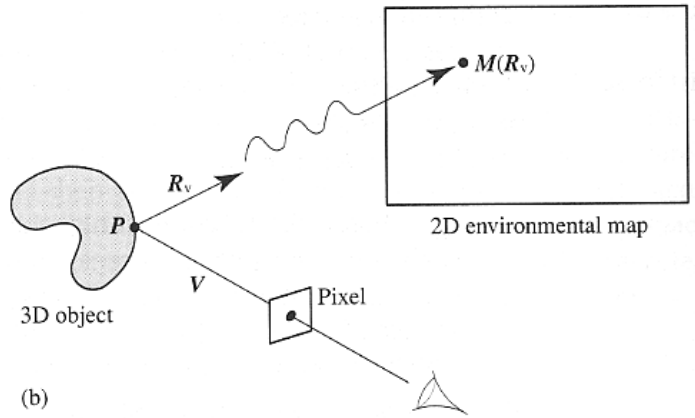
Projective Texturing: Examples

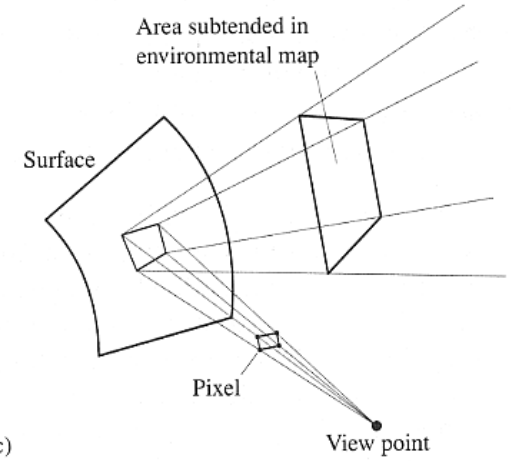


Reflection Mapping

- Also called Environment Mapping
- Mirror reflections
  - Surface curvature: beam tracing
  - Map filtering
- Reflection map parameterization
  - Intermediate surface in 2-stage mapping
  - Often cube, sphere, or double paraboloid
- Assumption: Distant illumination
  - Parallax-free illumination
  - No self-reflections, distortion of near objects
- Option: Separate map per object
  - Often necessary to be reasonable accurate
  - Reflections of other objects
  - Maps must be recomputed after changes

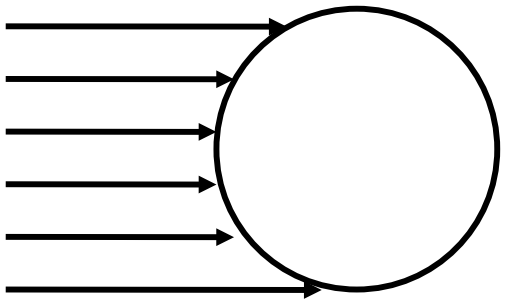






Reflection Map Acquisition

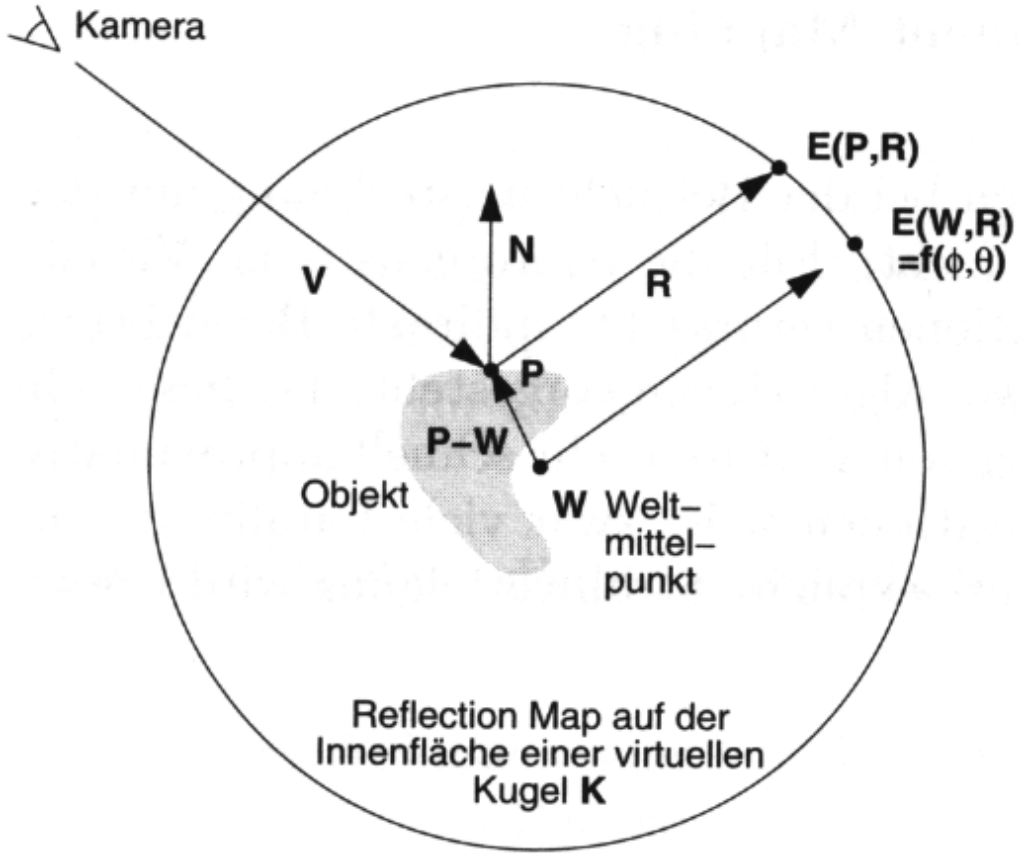
- Generating spherical maps (original 1982/83)
  - i.e. photo of a reflecting sphere (gazing ball)



Peter Chou

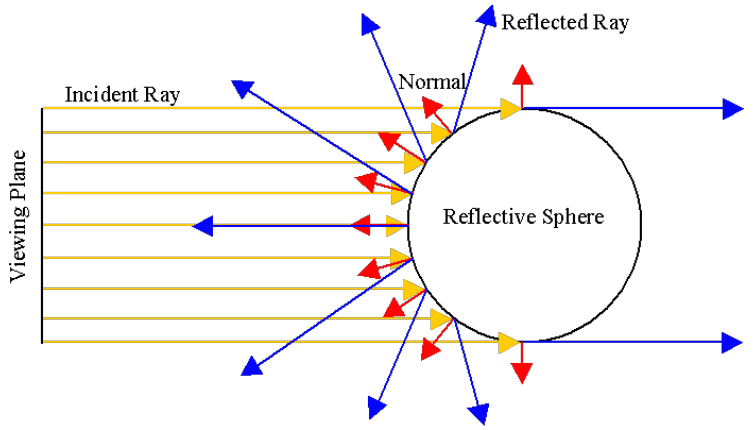
Reflection Map Rendering

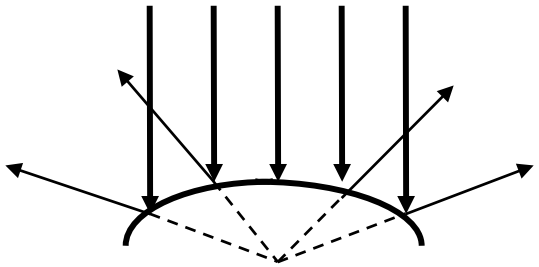
- Spherical parameterization
- O-mapping using reflected view ray intersection



Reflection Map Parameterization

- Spherical mapping
  - Single image
  - Bad utilization of the image area
  - Bad scanning on the edge
  - Artifacts, if map and image do not
  - have the same direction
- Double parabolic mapping
  - Subdivide in 2 images (facing and back facing side)
  - Less bias on the edge
  - Arbitrarily reusable
  - Supported by OpenGL extensions

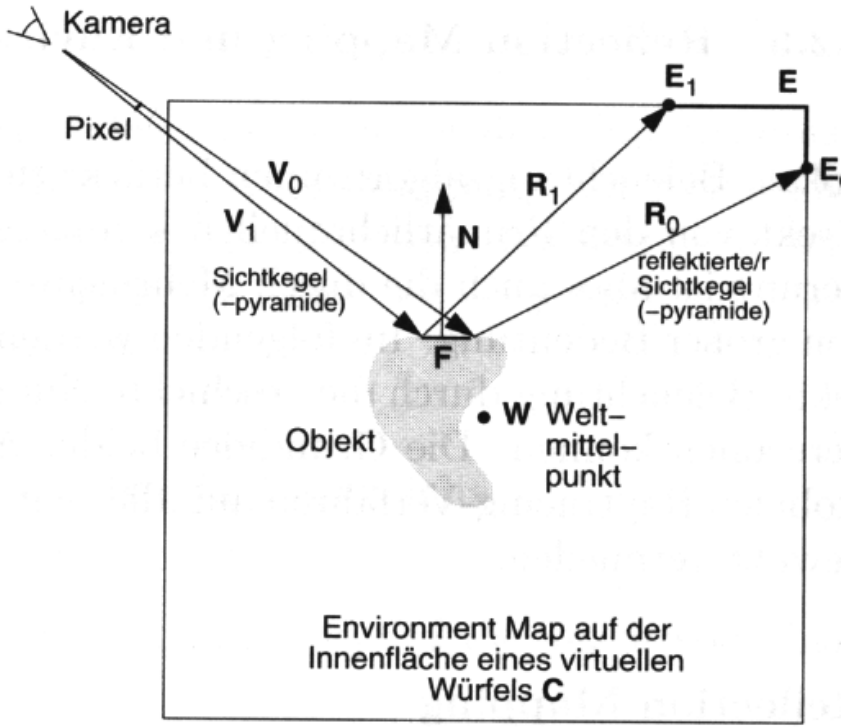


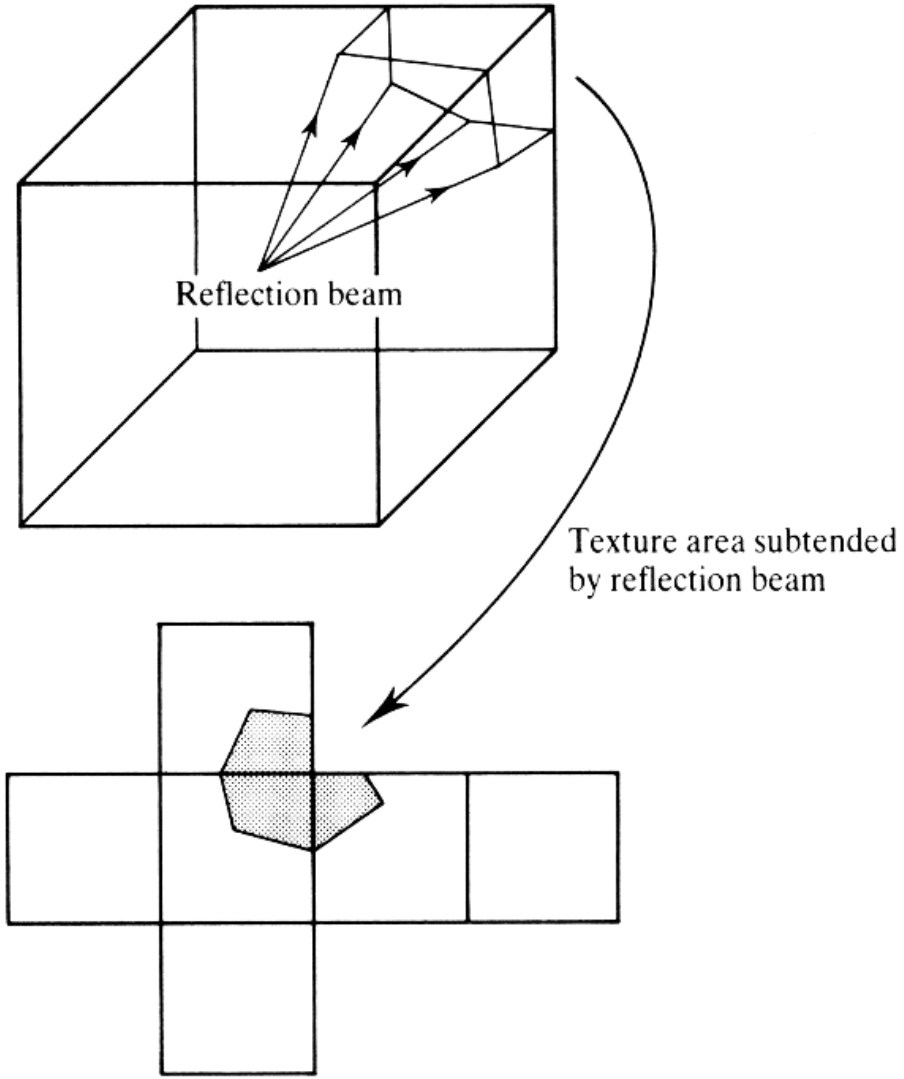


$$f(x,y) = \frac{1}{2} - \frac{1}{2}(x^2 + y^2)$$

Reflection Map Parameterization

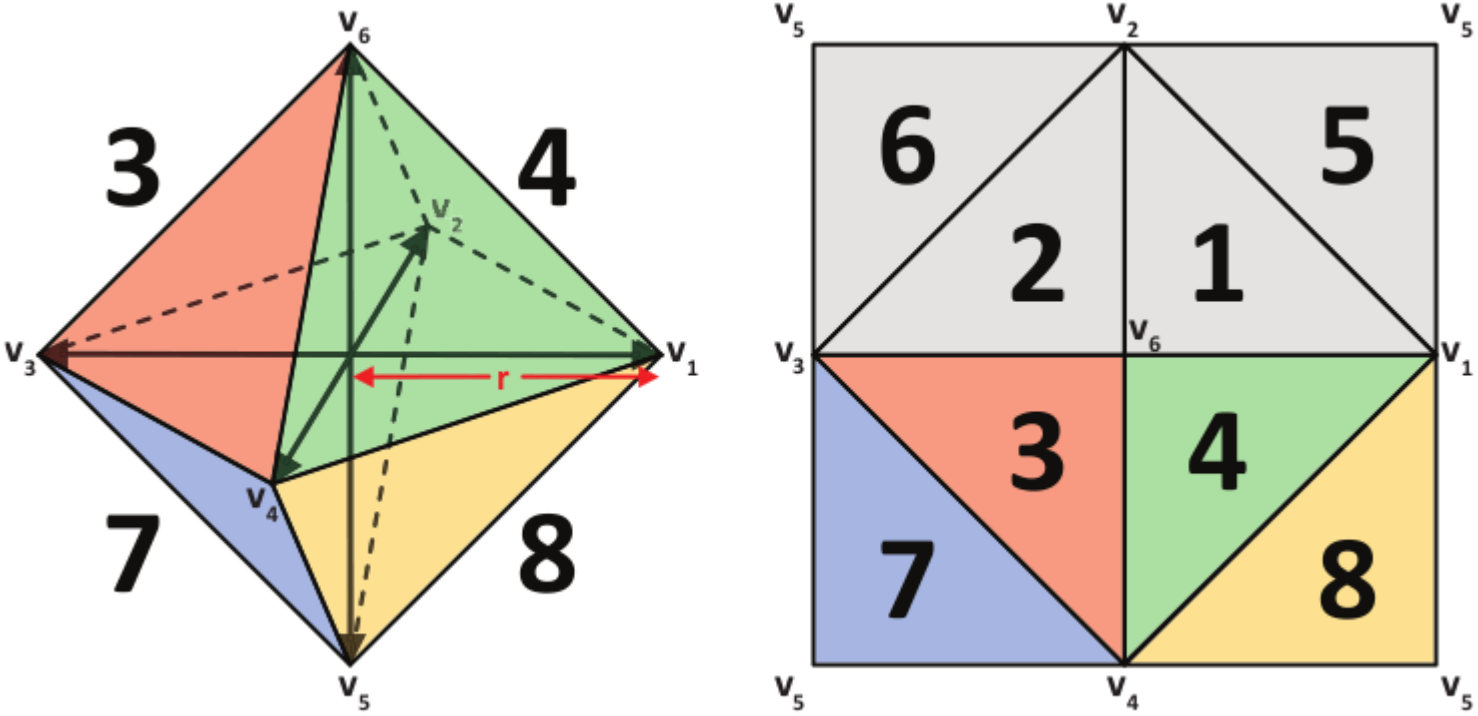
- Cubical environment map, cube map, box map
  - Enclose object in cube
  - Images on faces are easy to compute
  - Poorer filtering at edges
  - Support in OpenGL





Reflection Map Parameterization

- double pyramid [Hoppe et al. 2003]



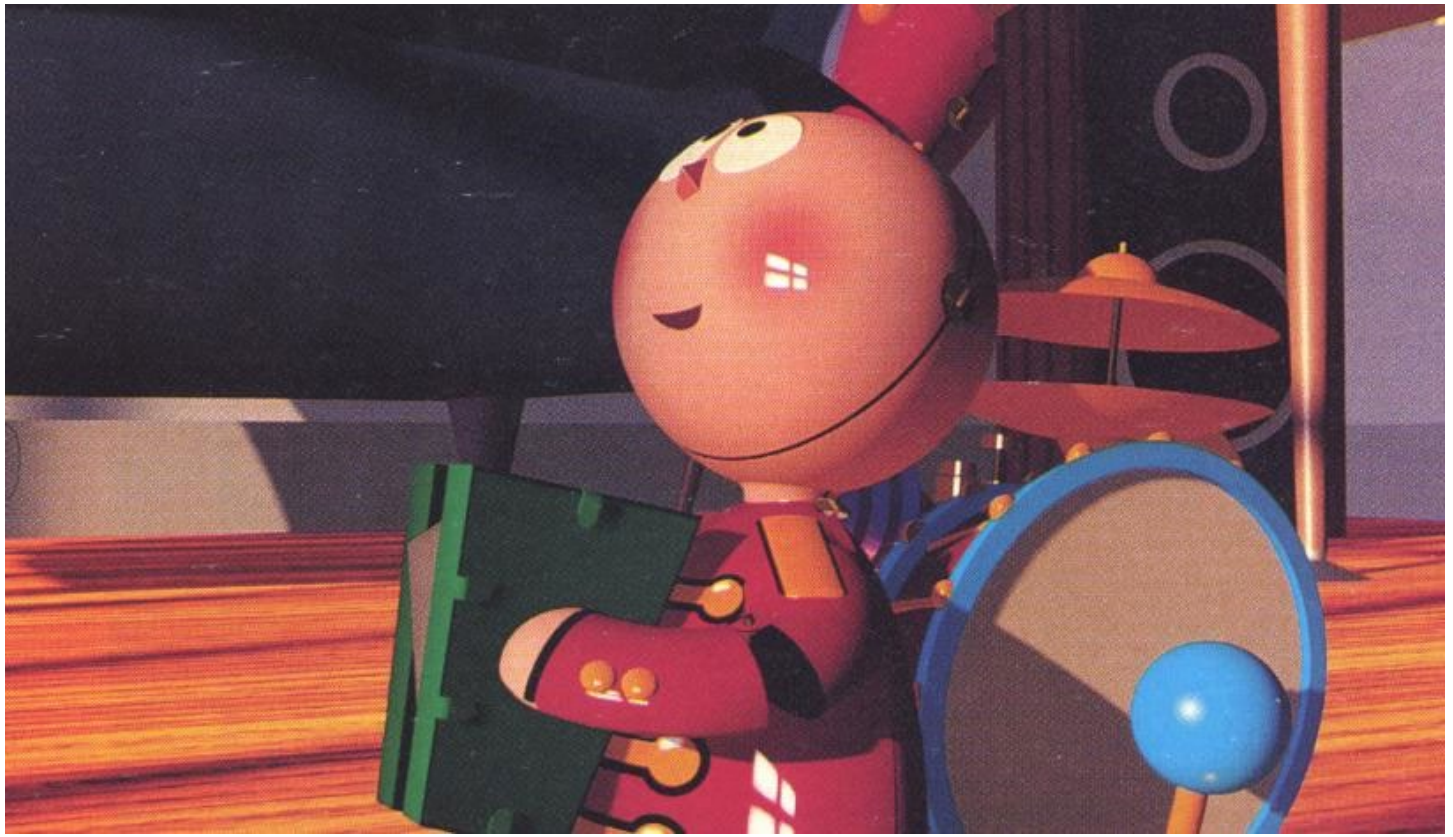
Reflection Mapping

Terminator II motion picture

Reflection Mapping Example II

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

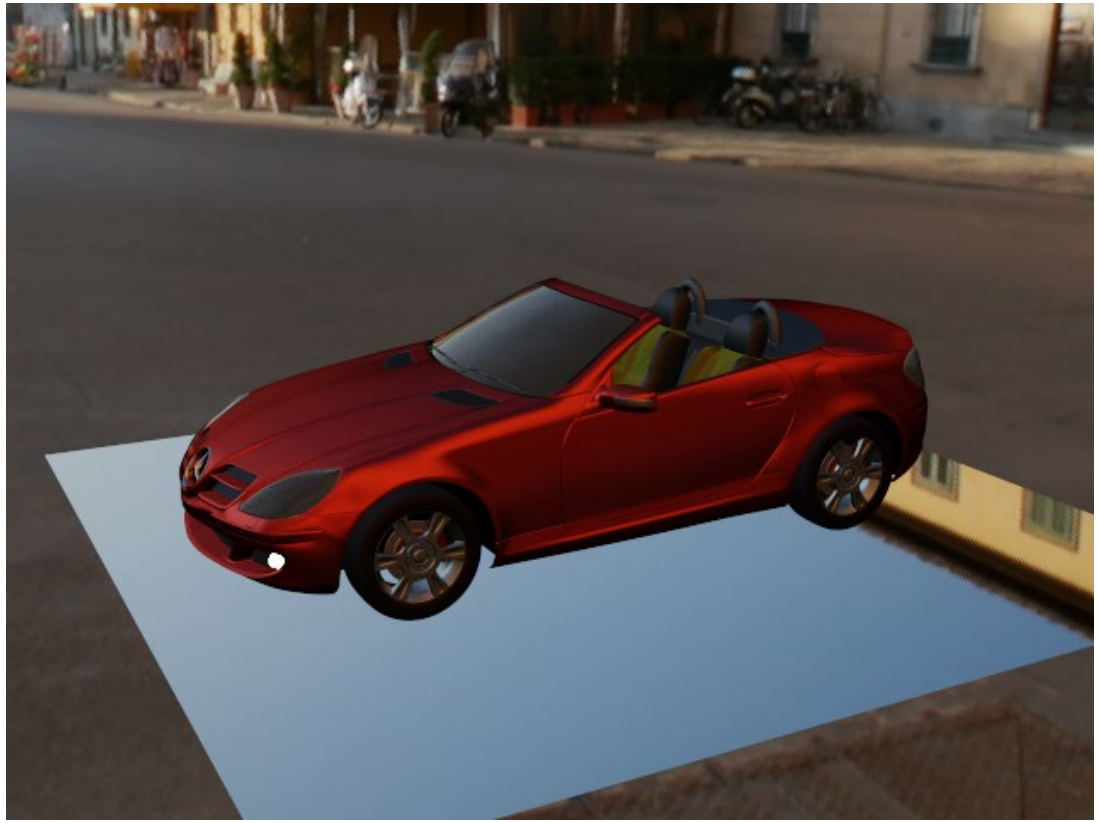
- Reflection mapping with Phong reflection
  - Two maps: diffuse & specular
  - Diffuse: index by surface normal
  - Specular: indexed by reflected view vector



RenderMan
Companion

Ray Tracing vs. Reflection Mapping

- Differences ?



Recursive Ray Tracing

- How to fake it with reflection mapping?

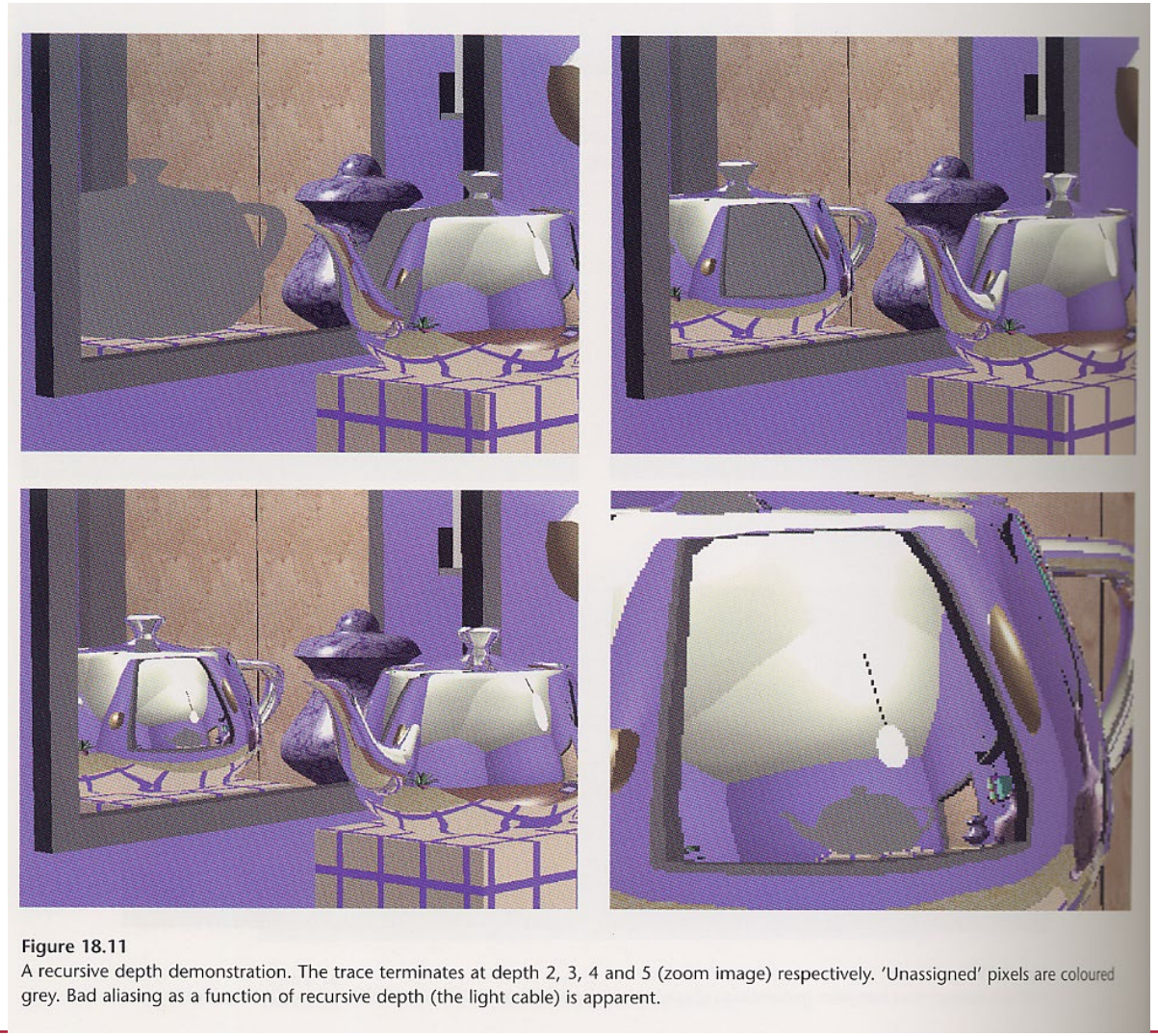
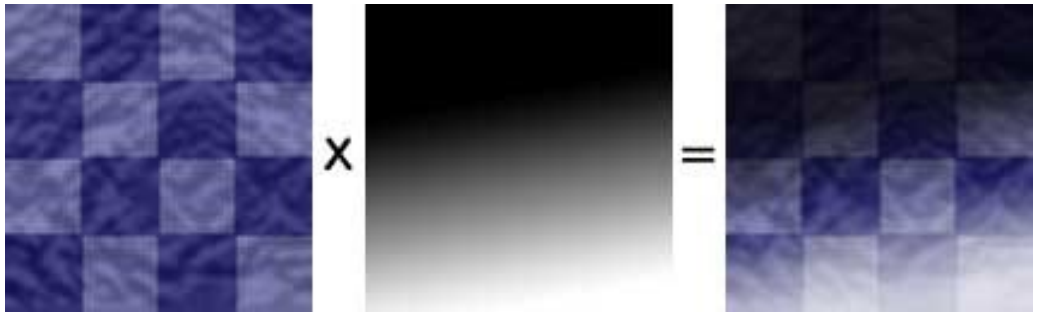


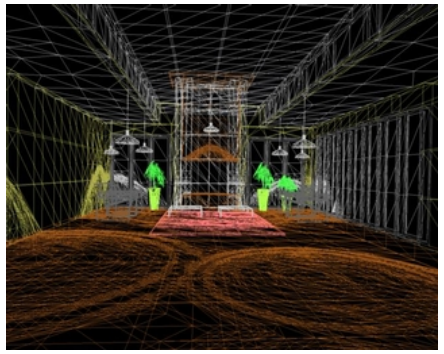
Figure 18.11
A recursive depth demonstration. The trace terminates at depth 2, 3, 4 and 5 (zoom image) respectively. 'Unassigned' pixels are coloured grey. Bad aliasing as a function of recursive depth (the light cable) is apparent.

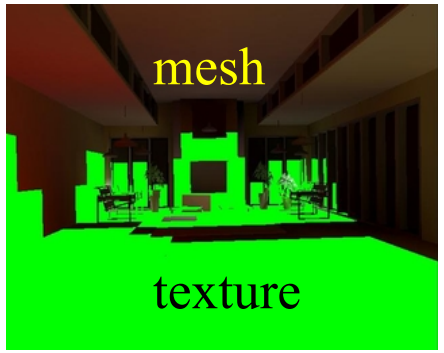
Light Maps

- Light maps (i.e. in Quake)
  - Pre-calculated illumination (local irradiance)
    - Often very low resolution
  - Multiplication of irradiance with base texture
    - Diffuse reflectance only
  - Provides surface radiosity
    - View-independent
  - Animated light maps
    - Animated shadows, moving light spots etc.



Reflectance Irradiance Radiosity





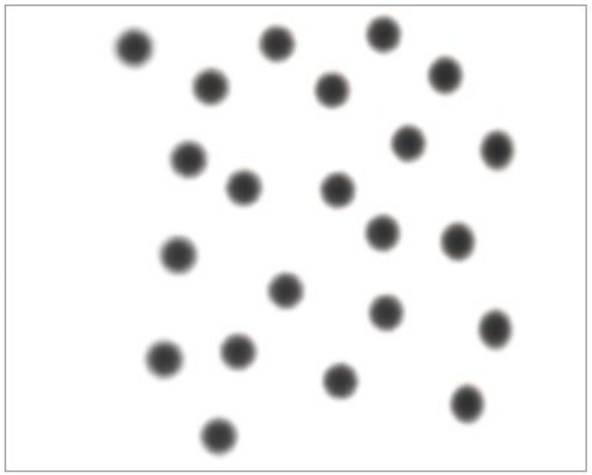
Representing radiosity in a mesh or texture

Bump Mapping

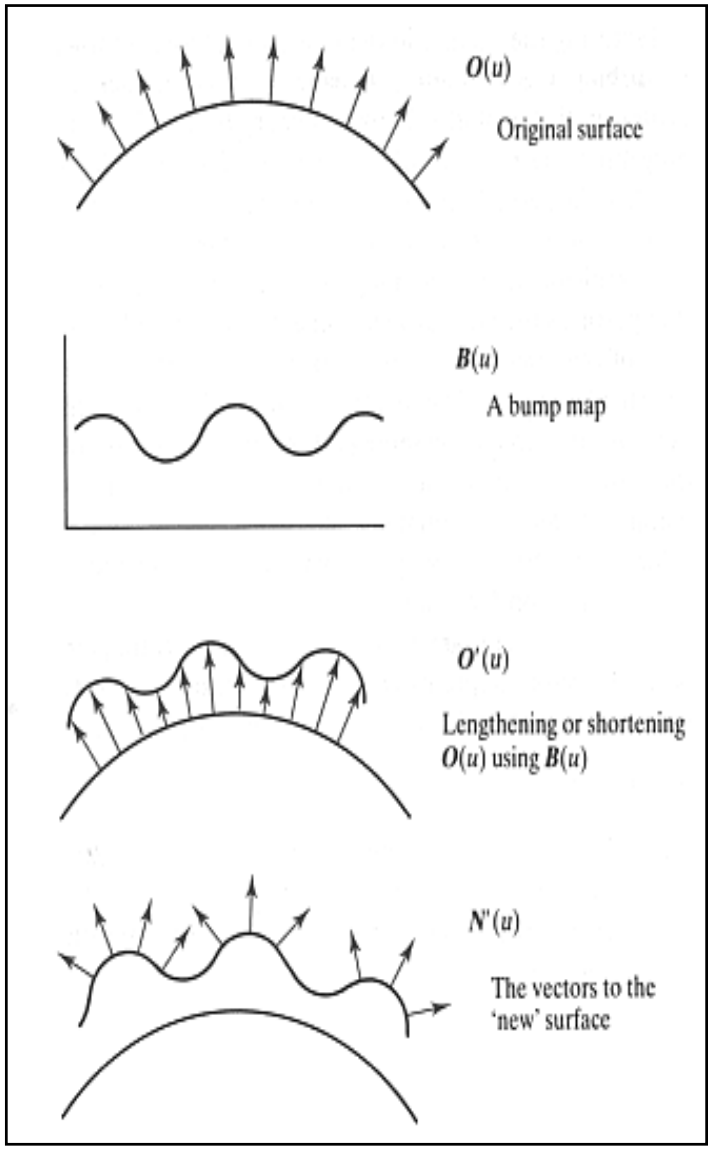
- Modulation of the normal vector
  - Surface normals changed only
    - Influences shading only
    - No self-shadowing, contour is not altered

Bump Mapping

- Original surface $O(u,v)$
  - Surface normals are known
- Bump map $B(u,v) \in R$
  - Surface is offset in normal direction according to bump map intensity
  - New normal directions $N'(u,v)$ are calculated based on *virtually* displaced surface $O'(u,v)$
  - Originals surface is rendered with new normals $N'(u,v)$



Grey-valued texture used for bump height



Bump Mapping

$$O'(u,v) = O(u,v) + B(u,v)\frac{N}{|N|}$$

Now differentiating this equation gives:

$$O'_u = O_u + B_u\frac{N}{|N|} + B\left(\frac{N}{|N|}\right)_u$$

$$O'_v = O_v + B_v\frac{N}{|N|} + B\left(\frac{N}{|N|}\right)_v$$

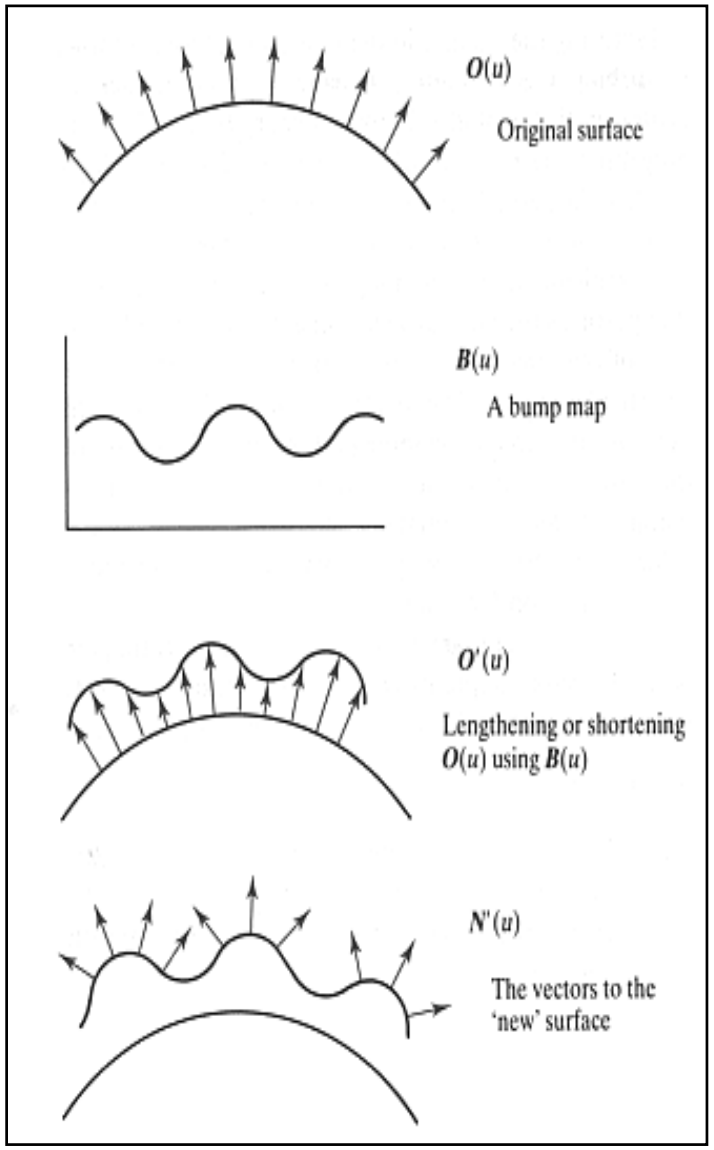
If B is small (that is, the bump map displacement function is small compared with its spatial extent) the last term in each equation can be ignored and

$$N'(u,v) = O_u \times O_v + B_u\left(\frac{N}{|N|} \times O_v\right) + B_v\left(O_u \times \frac{N}{|N|}\right) + B_u B_v\left(\frac{N \times N}{|N|^2}\right)$$

The first term is the normal to the surface and the last term is zero, giving:

$$D = B_u(N \times O_v) - B_v(N \times O_u)$$

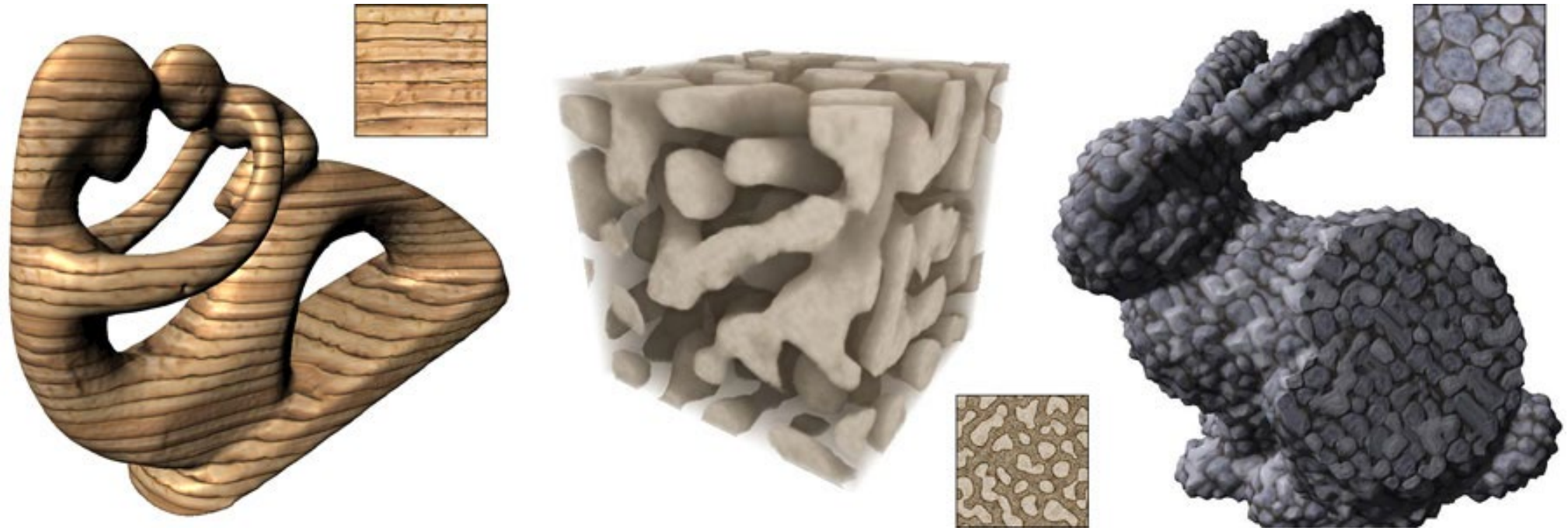
$N'=N+D$



[Watt&Watt 1992]

Solid Textures

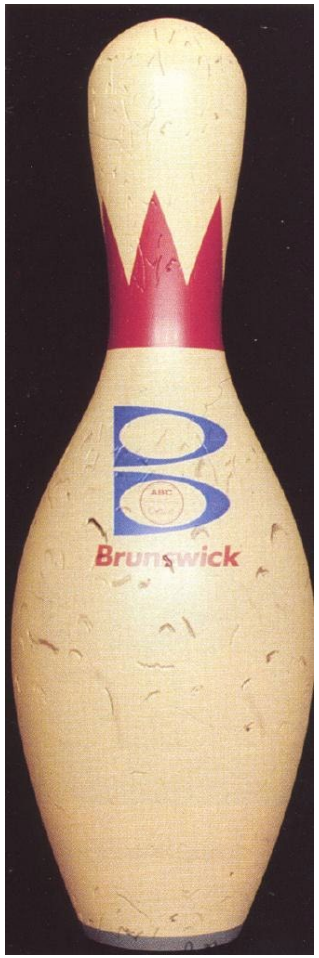
- Color (and transparency) per voxel
- Evaluated on visible surface



[Kopf et al. SIGGRAPH 2007]

Texture Examples

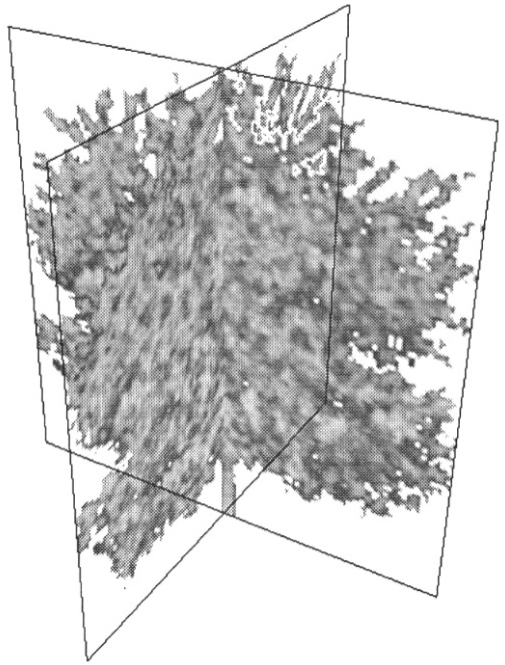
- Complex optical effects
  - Combination of multiple textures

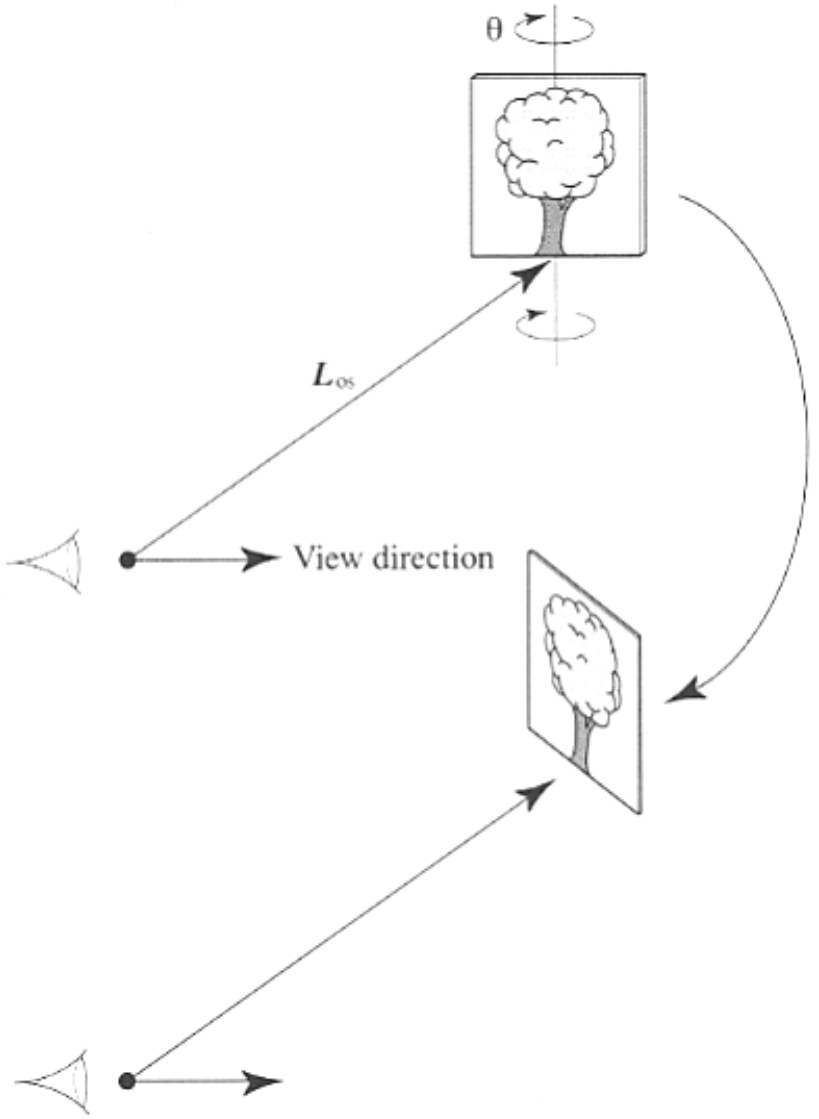


RenderMan Companion

Billboards

- Single textured polygons
  - Often with transparency texture
- Rotates, always facing viewer
- Used for rendering distant objects
- Best results if approximately radially or spherically symmetric

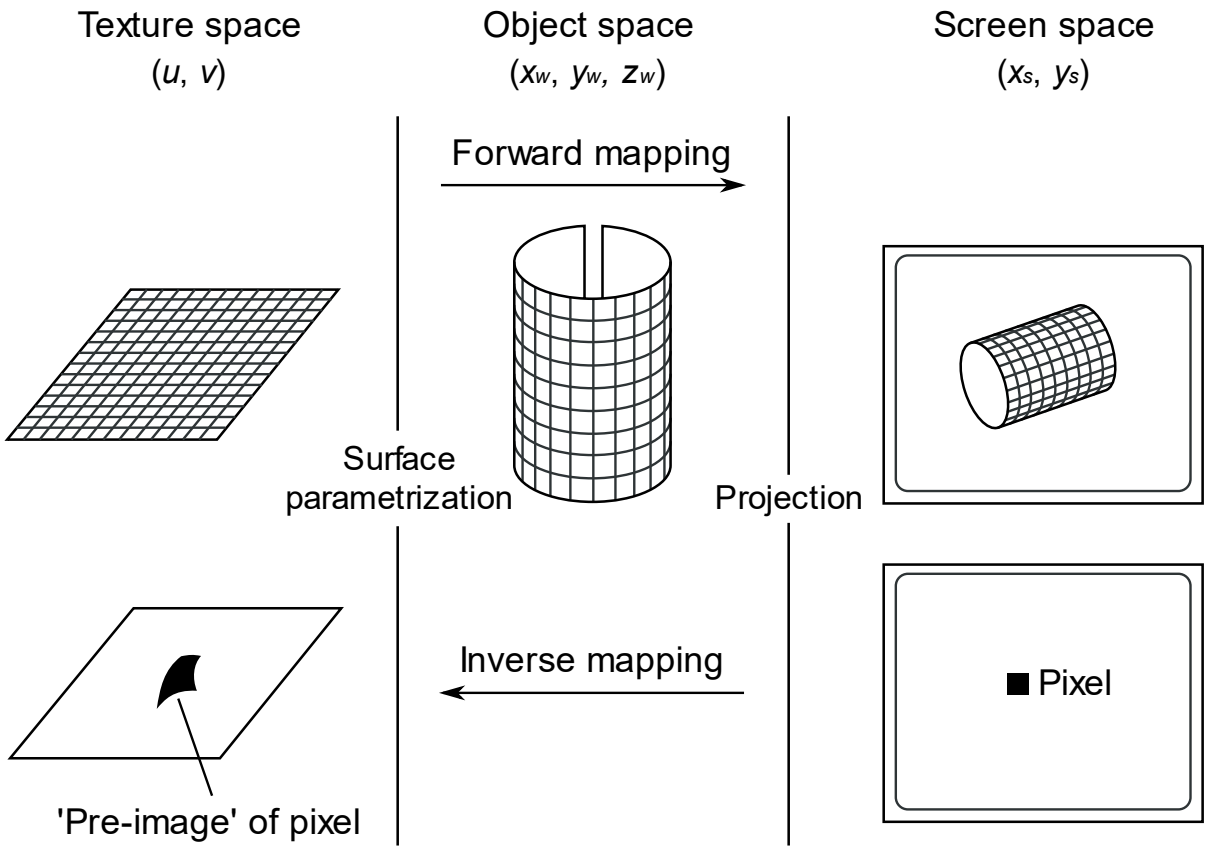




Texture Filtering

How to deal with varying texel size due to projection?

2D Texture Mapping



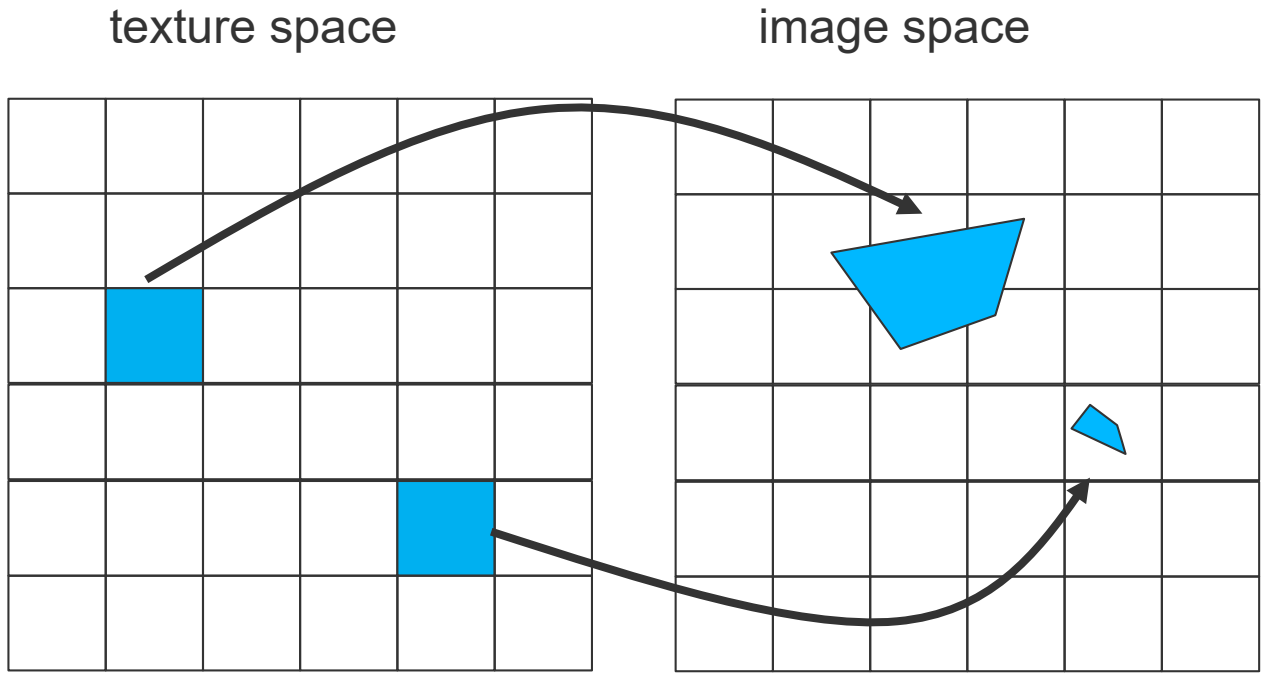
- Forward mapping
  - Object surface parameterization
  - Projective transformation
- Inverse mapping
  - Find corresponding pre-image/footprint of each pixel in texture
  - Integrate over pre-image

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:

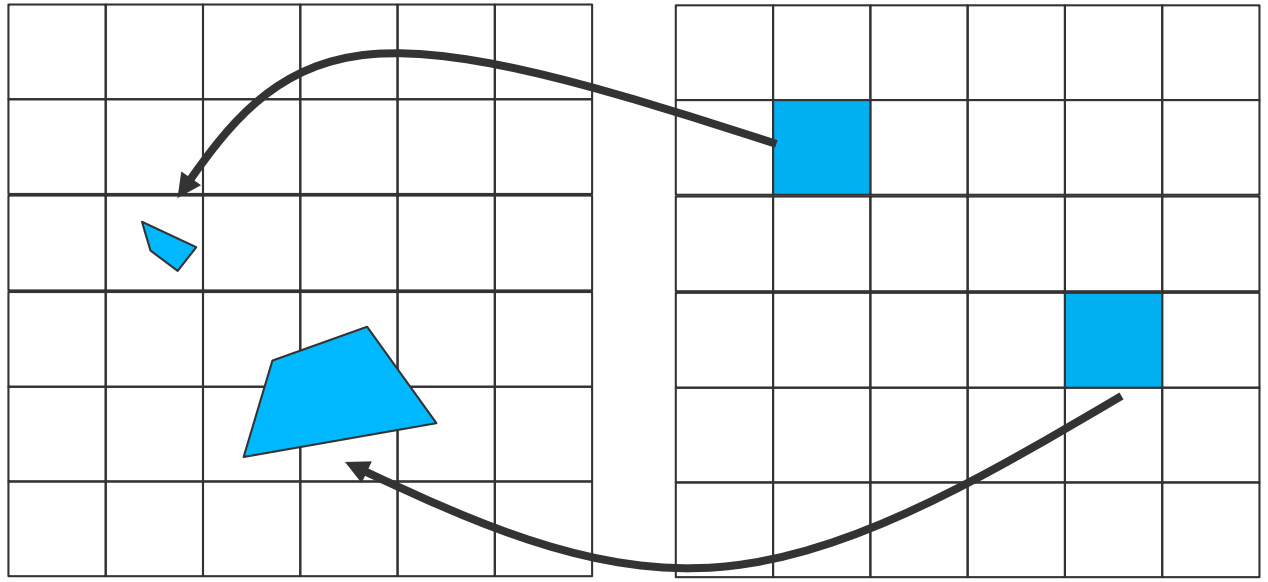
```
for v
  for u
    compute x(u,v) and y(u,v)
    copy TEX[u,v] to SCR[x,y]
(or in general
    rasterize image of TEX[u,v])
```



Inverse Mapping

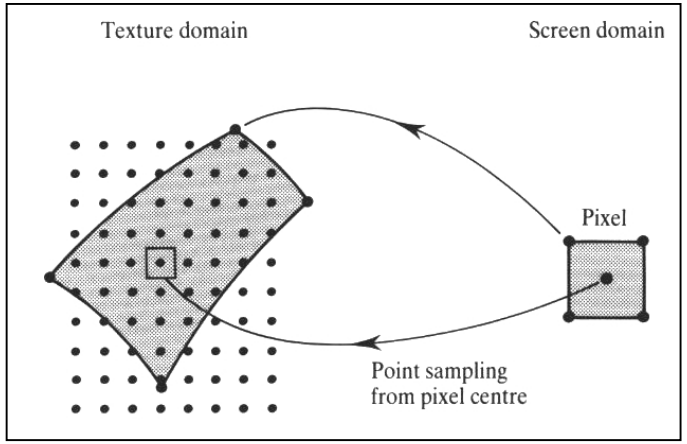
- Requires inverting the mapping transformation
- Preferable when the mapping is readily invertible and the texture image fits into memory
- The most common mapping method
  - for each pixel in screen space, the pre-image of the pixel in texture space is found and its area is integrated over

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,v])

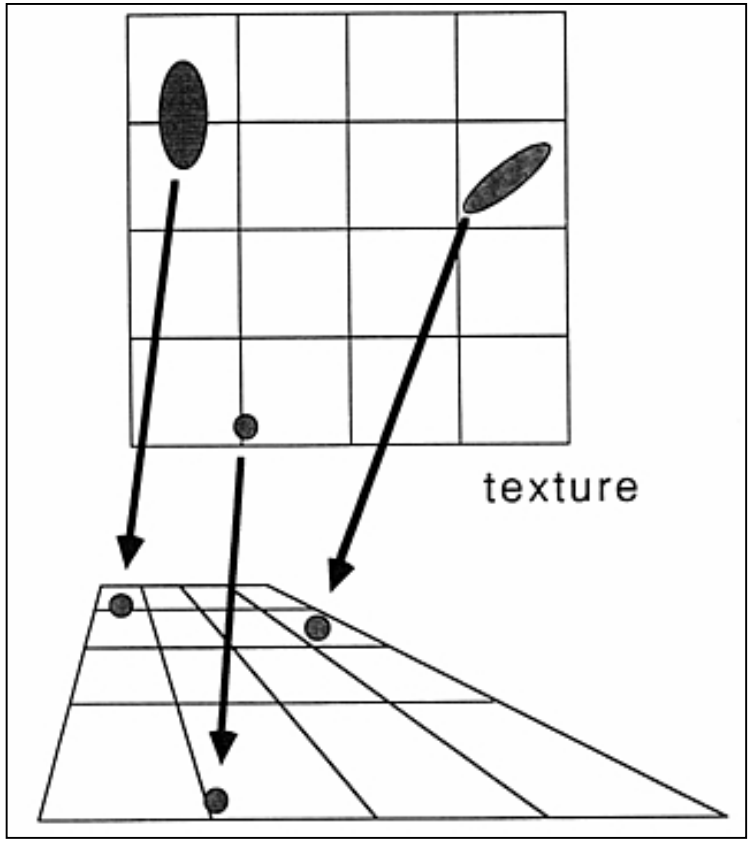


Pixel Pre-Image in Texture Space

- 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.

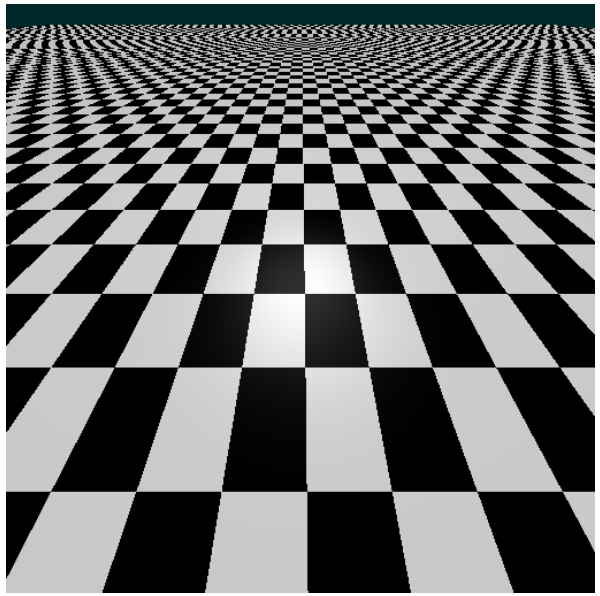


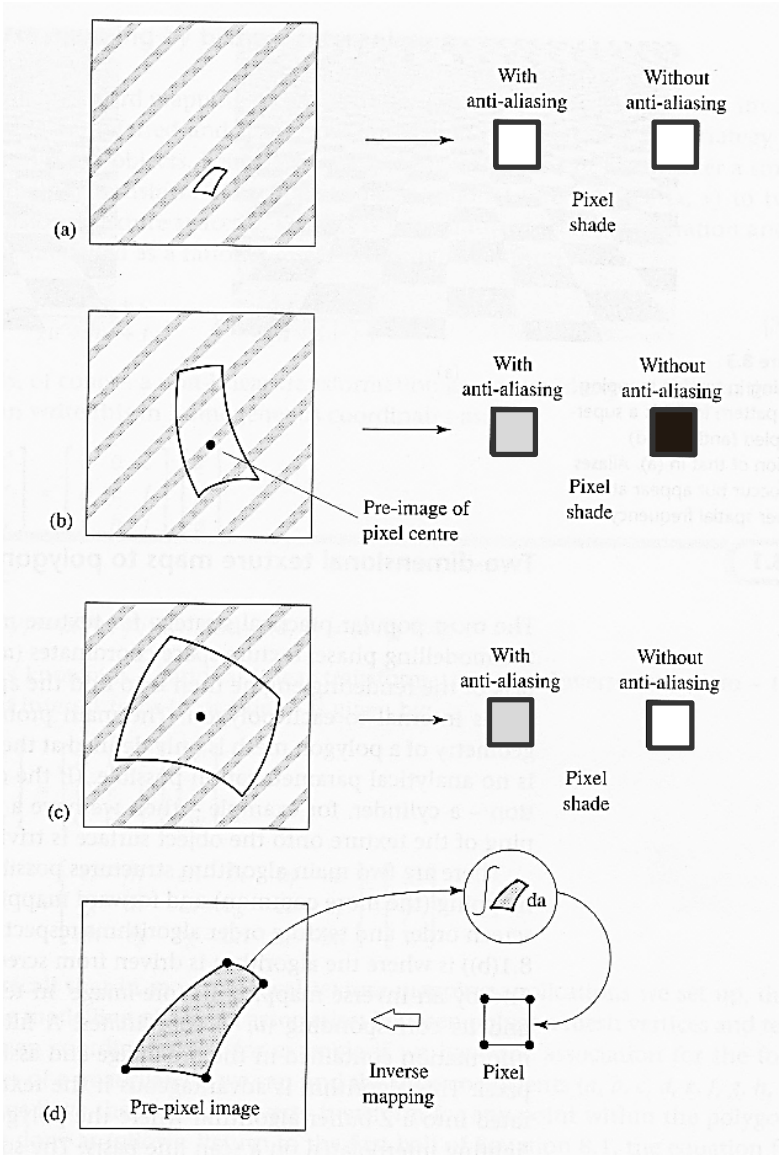
Approximating a quadrilateral texture area with (a) a square, (b) a rectangle, and (c) an ellipse. Too small an area causes aliasing; too large an area causes blurring.



Inverse Mapping: Filtering

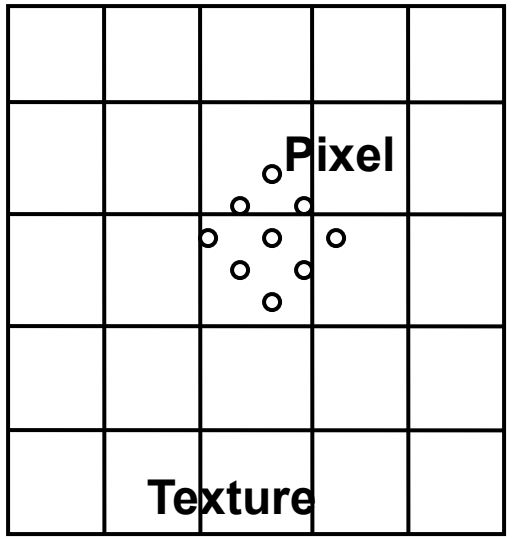
- Integration of Pre-image
  - Integration over pixel footprint in texture space
- Aliasing
  - Texture insufficiently sampled
  - Incorrect pixel values
  - “Randomly” changing pixels when moving

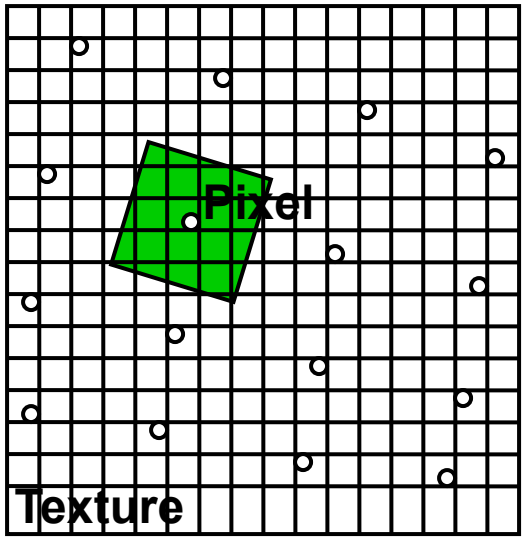




Filtering

- 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





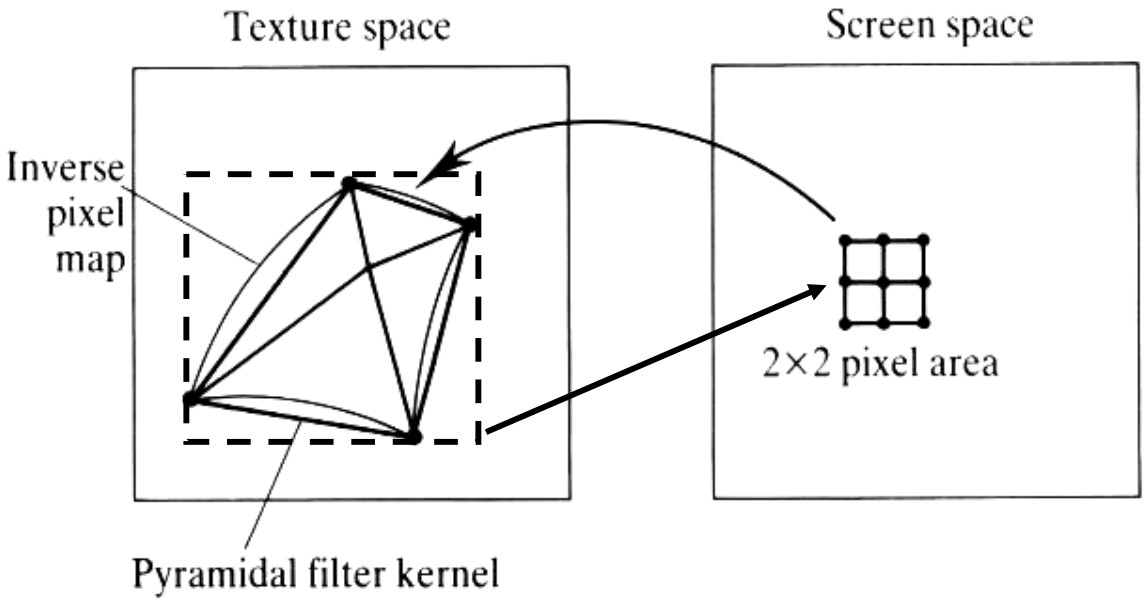
Filtering – Texture Minification

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- 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

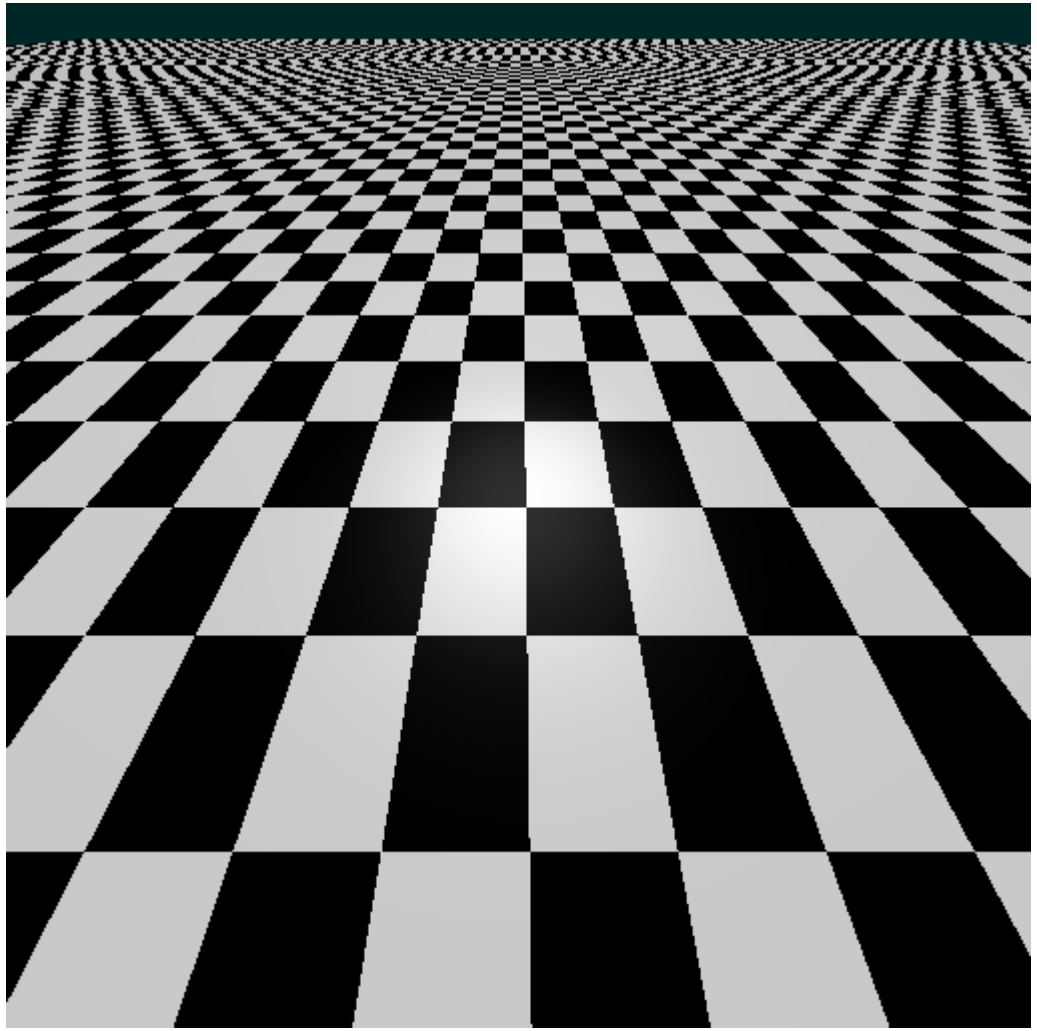
- Convolution in texture space
  - Texels weighted according to distance from pixel center (e.g. pyramidal filter kernel)



- 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.

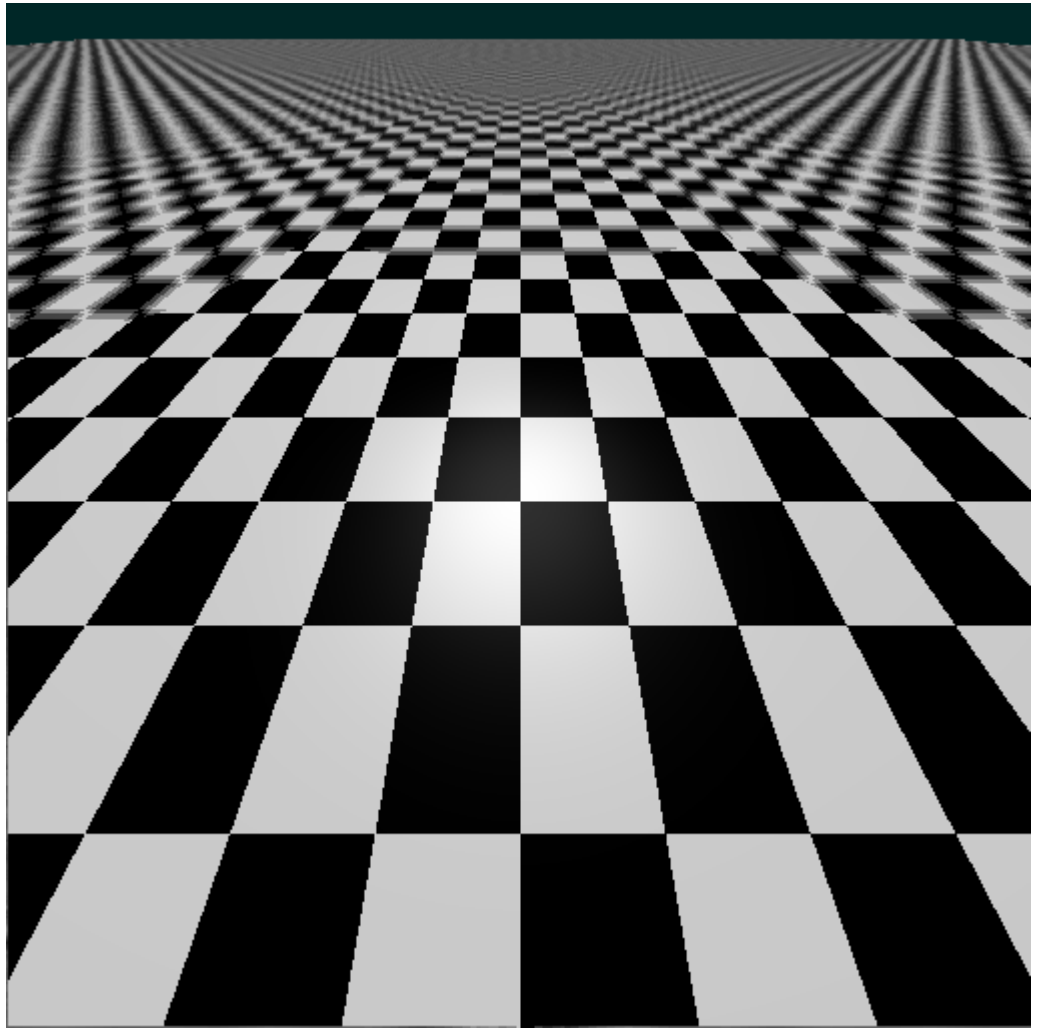
Aliasing – Sampling too sparse

- checker board gets distorted



EWA Filtering – Anti-Aliasing

- elliptical filtering plus Gaussian



Accelerated Filtering

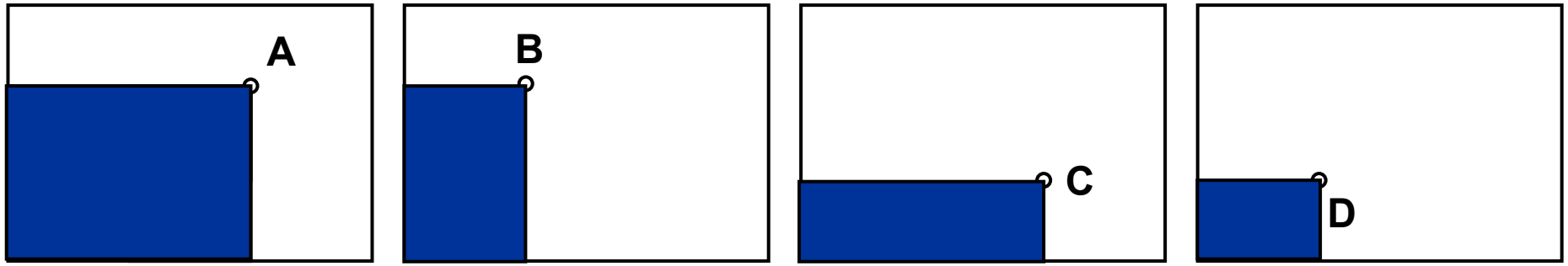
Filtering – Texture Minification

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- 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

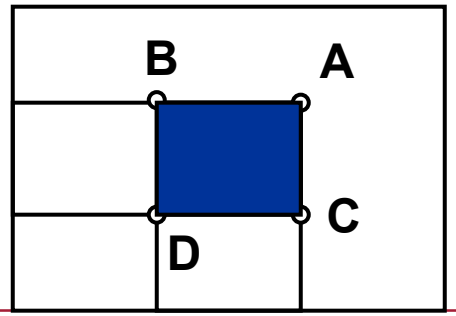
Summed Area Tables

- Per texel, store sum from *(0, 0)* to *(u, v)*



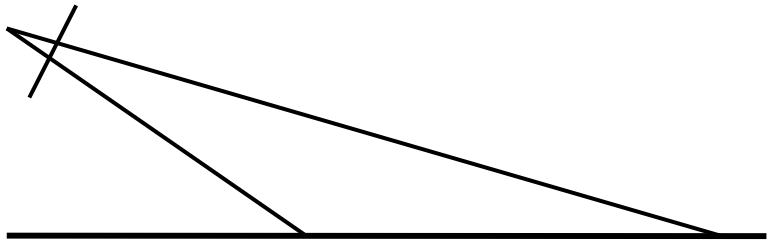
- Many bits per texel (sum !)
- Evaluation of 2D integrals in constant time!

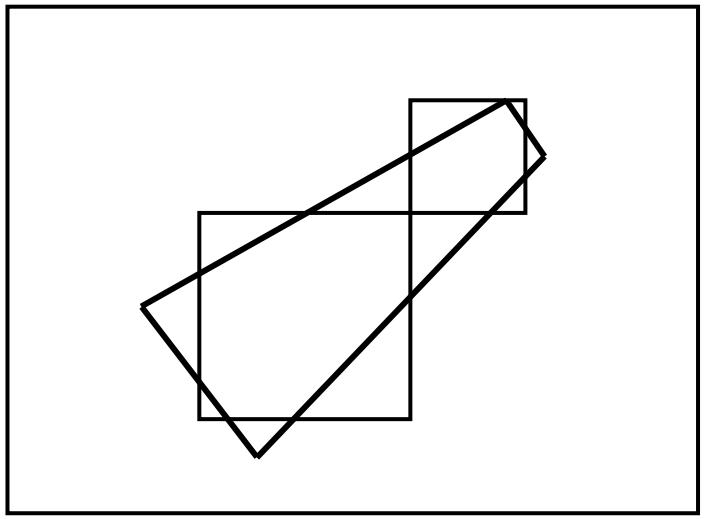
$$\int_{Bx}^{Ax}\int_{Cy}^{Ay} I(x,y)dxdy = A - B - C + D$$



Integrated Arrays

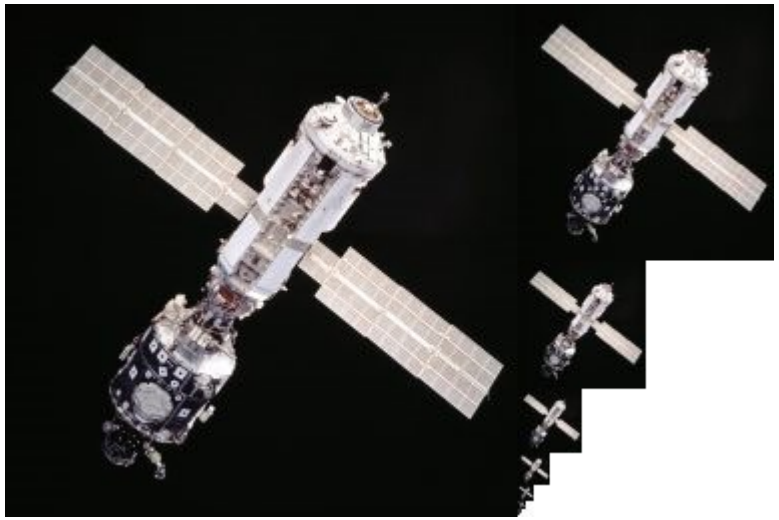
- Footprint assembly
  - Good for space variant filtering
    - e.g. inclined view of terrain
  - Approximation of the pixel area by rectangular texel-regions
  - The more footprints the better accuracy
  - Often fixed number of texels because of economical reasons

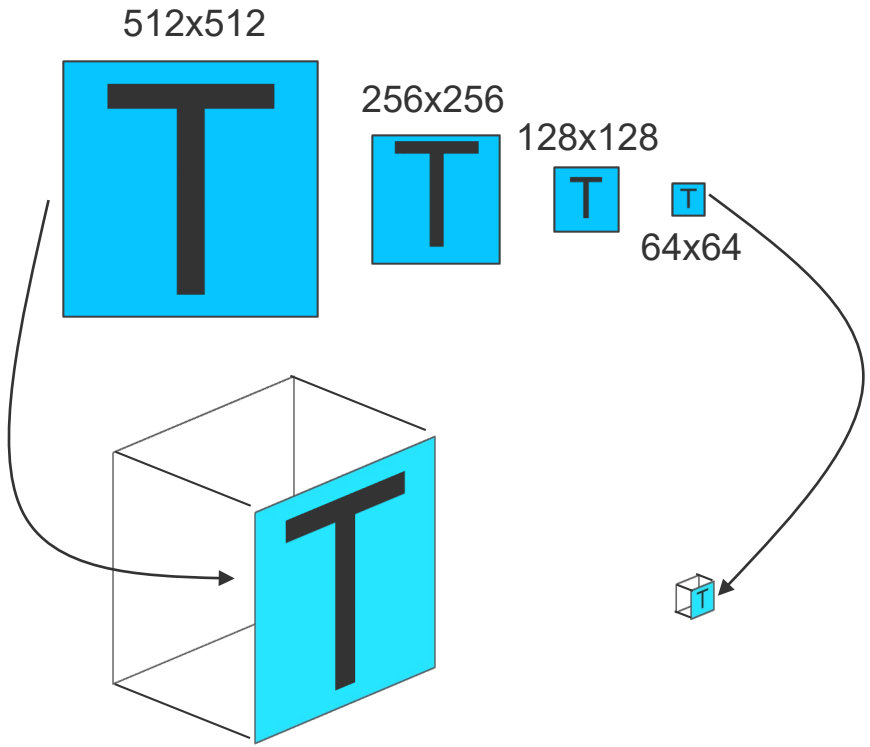




MipMapping

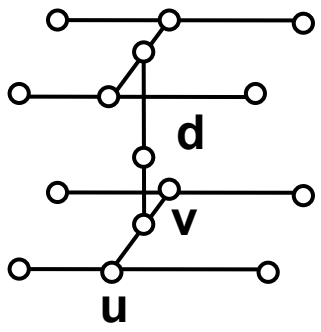
- Texture available in multiple resolutions
  - Pre-processing step
- Rendering: select appropriate texture resolution
  - Selection is usually per pixel !!
  - Texel size(n) < extent of pixel footprint < texel size(n+1)

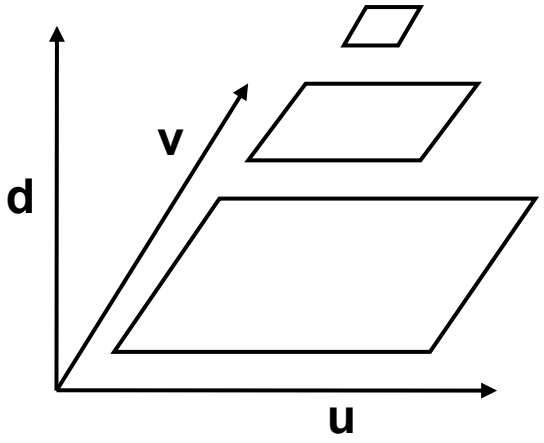


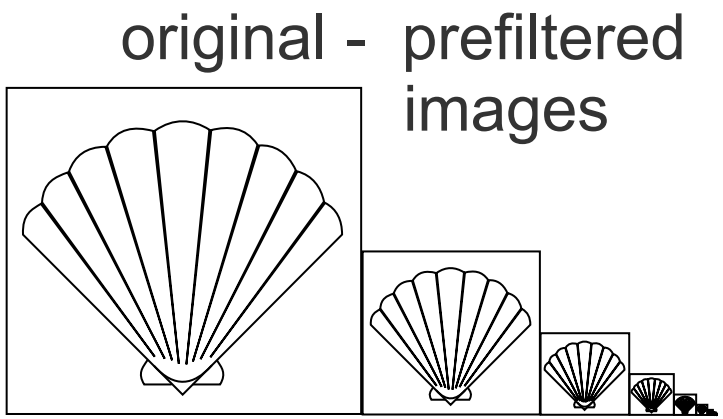


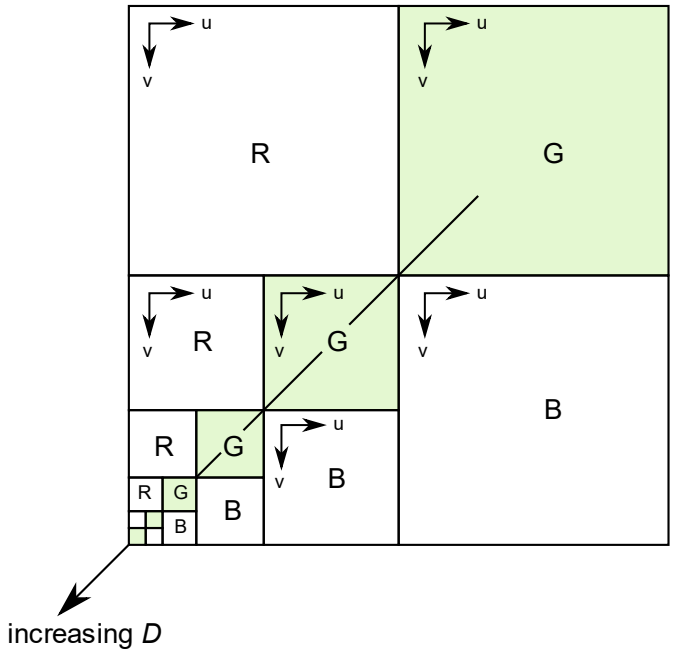
MipMapping II

- 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

- RGB → 4/3 the size

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

Procedural Textures

How to describe a natural 2D distribution of samples?

Texture Maps vs. Procedural Textures

- Texture maps (photos, simulations, videos, ...)
  - Simple acquisition
  - Illumination „frozen“ during acquisition
  - Limited resolution, aliasing
  - High memory requirements
  - Mapping issues
- Procedural textures
  - Non-trivial programming
  - Flexibility & parametric control
  - Unlimited resolution
  - Anti-aliasing possible
  - Low memory requirements
  - Low-cost visual complexity
  - Can adapt to arbitrary geometry

Ken Perlin

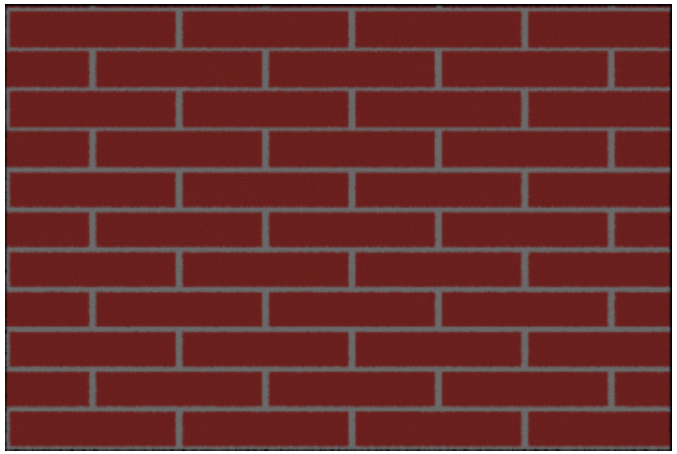
Procedural Textures

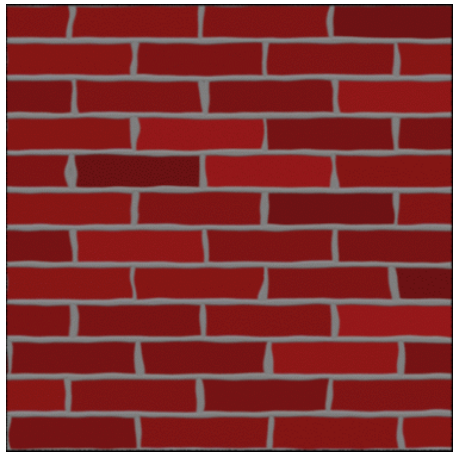
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Function of some shading parameter, e.g.
  - world space, texture coordinates, ...
- Texturing: evaluation of function on object surface
  - Ray tracing: At intersection point with surface
- Observation: Textures of natural objects
  - Similarity between patches at different locations
    - Repetitiveness, coherence (e.g. skin of a tiger)
  - Similarity on different resolution scales
    - Self-similarity
  - But never completely identical
    - Additional disturbances, turbulence, noise
- Goal: Generic procedural texture function
  - Mimics statistical properties of natural textures
  - Purely empirical approach
    - Looks convincing, but has nothing to do with material's physics

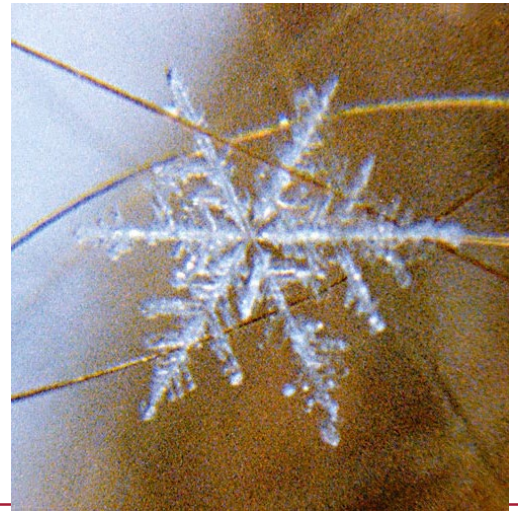
Texture Examples

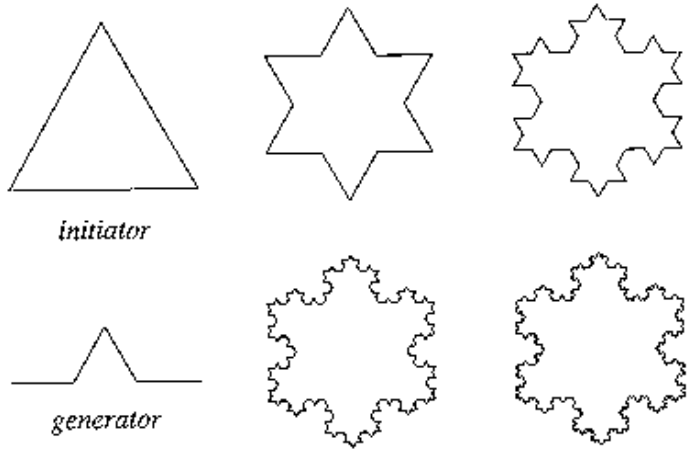
- Translational similarity





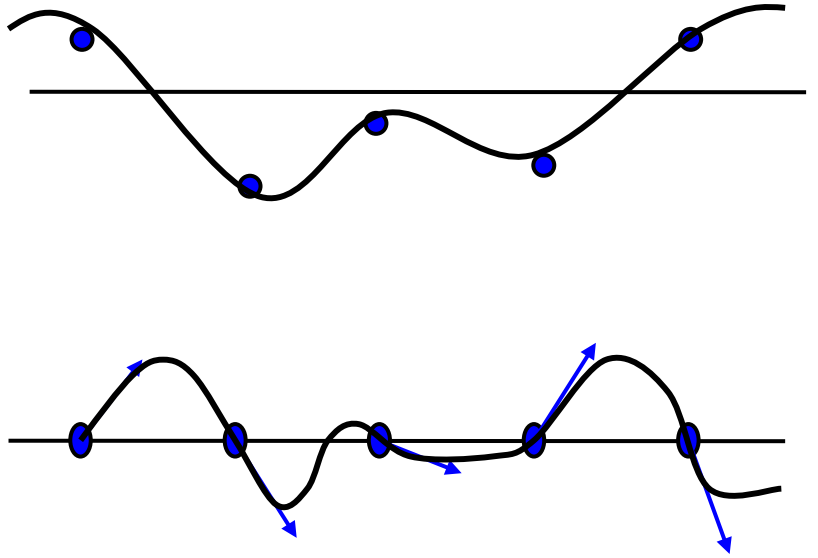
- Similarity on different scales





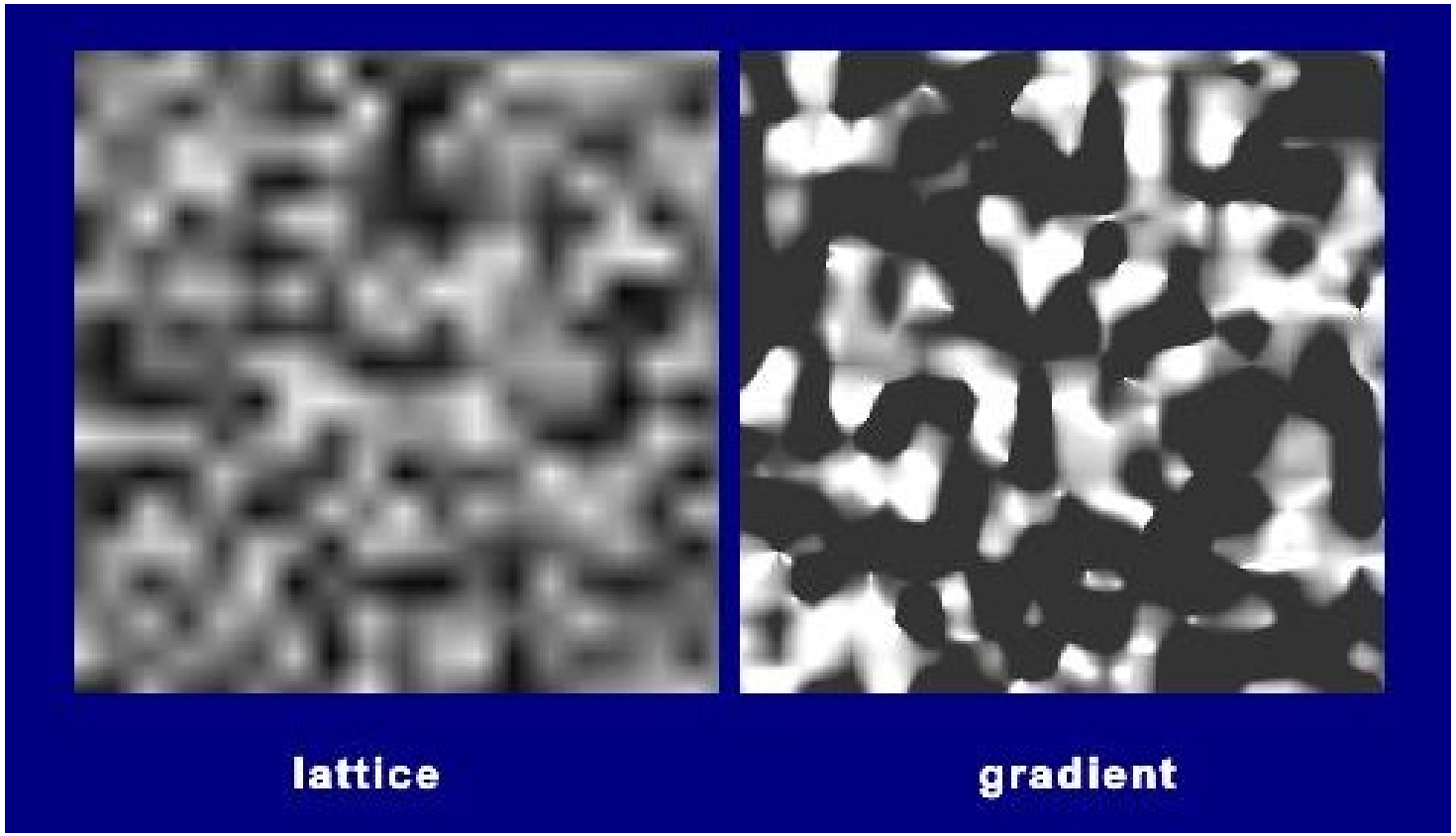
3D / Solid Noise: Perlin Noise

- Noise(x,y,z)
  - Statistical invariance under rotation
  - Statistical invariance under translation
  - Roughly one specific frequency
- Integer lattice (i,j,k)
  - Value noise: Random number at lattice
    - Look-up table or hashing function into hash map
  - Gradient lattice noise
    - Random (hashed) gradient vectors
  - Fixed fundamental frequency of ~1 Hz over lattice
- Evaluation at (x,y,z)
  - Tri-linear interpolation
  - Cubic interpolation (Hermite spline → later)
- Unlimited domain due to lattice and hashing
- Also see
  - http://freespace.virgin.net/hugo.elias/models/m_perlin.htm



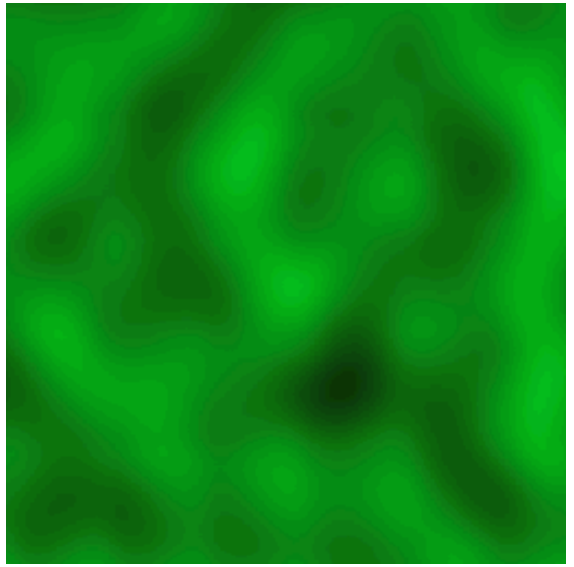
Gradient vs. Value Noise

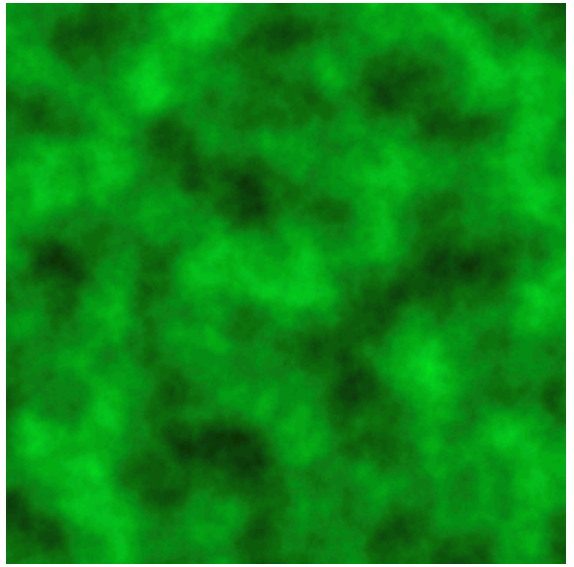
- Gradient noise better than value noise
  - Less regularity artifacts
  - More high frequencies in noise spectrum
  - Even tri-linear interpolation produces good results



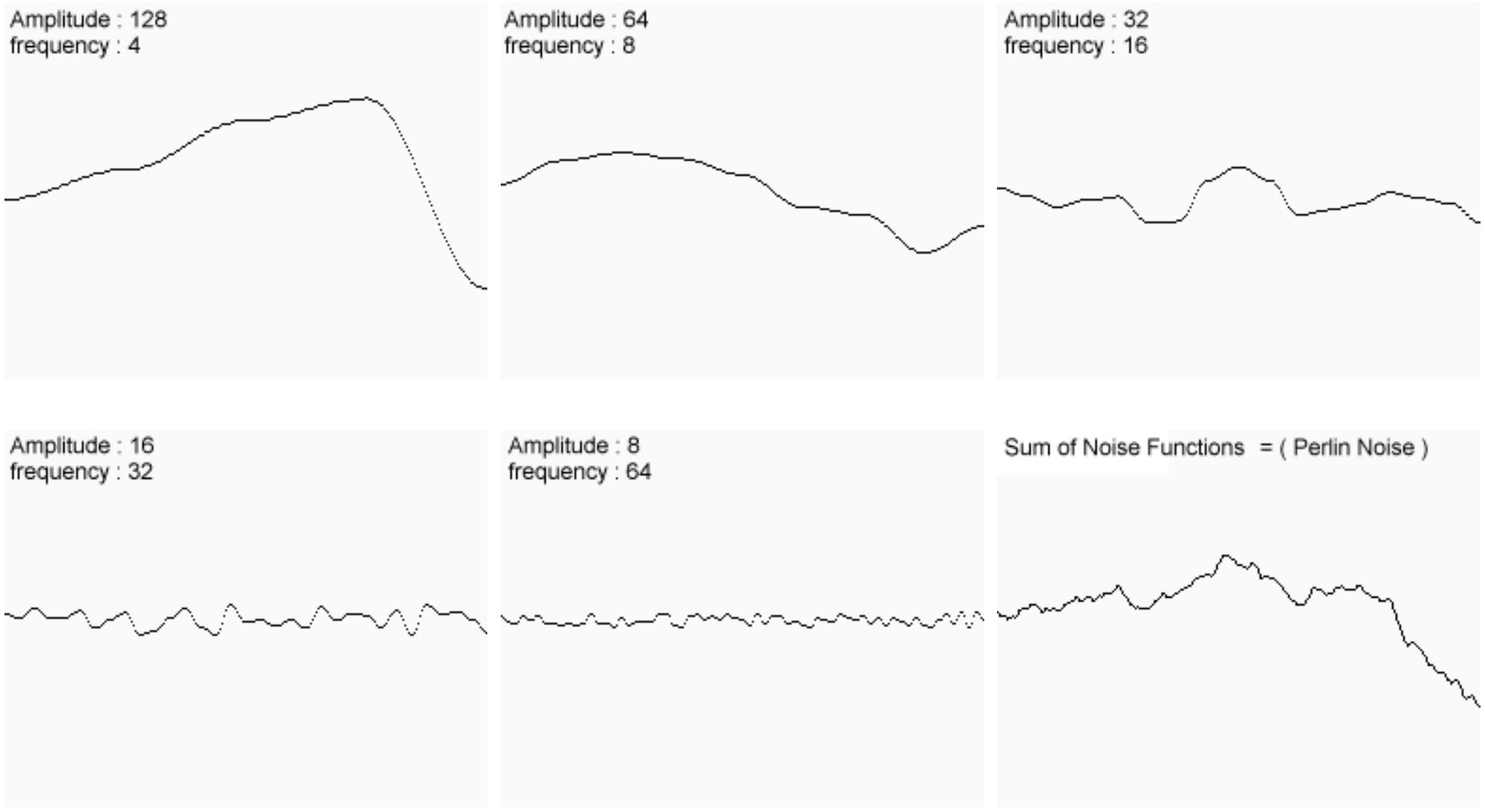
Turbulence Function

- Noise function
  - "White" frequency spectrum
- Natural textures
  - Decreasing power spectrum towards high frequencies
- Turbulence from noise
  - Turbulence(x) = $\sum^{k}_{i=0}$ abs(noise(2^i x) / p^i)
  - persistence p typically p=2
  - Summation truncation
    - $1/2^{k+1}$ < size of one pixel (band limit)
  - 1. Term: noise(x)
  - 2. Term: noise(2x)/2
  - …
  - Power spectrum: 1/f
  - (Brownian motion has $1/f^2$)

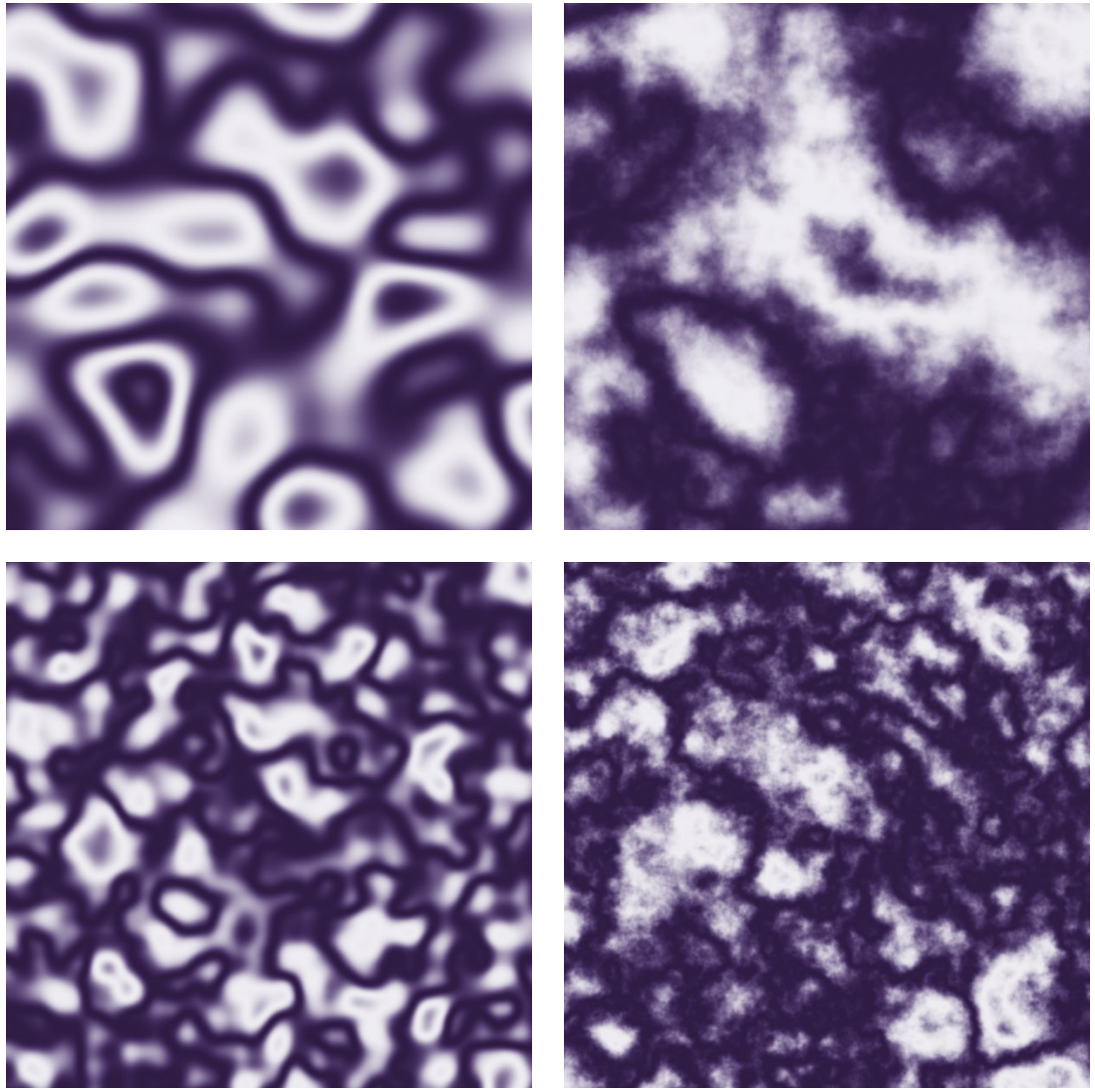




Synthesis of Turbulence (1D)

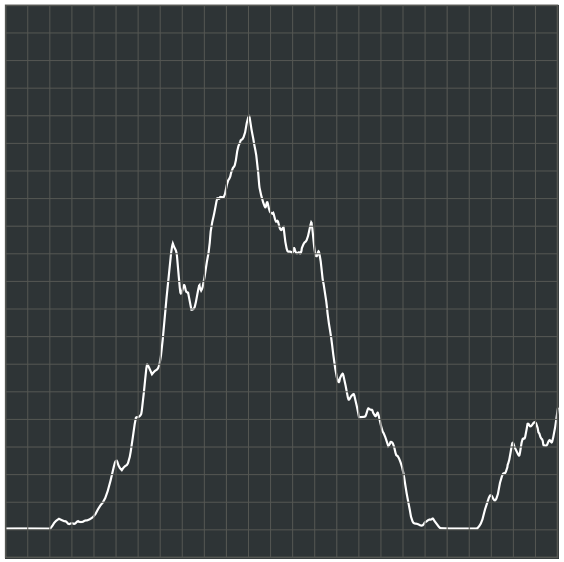


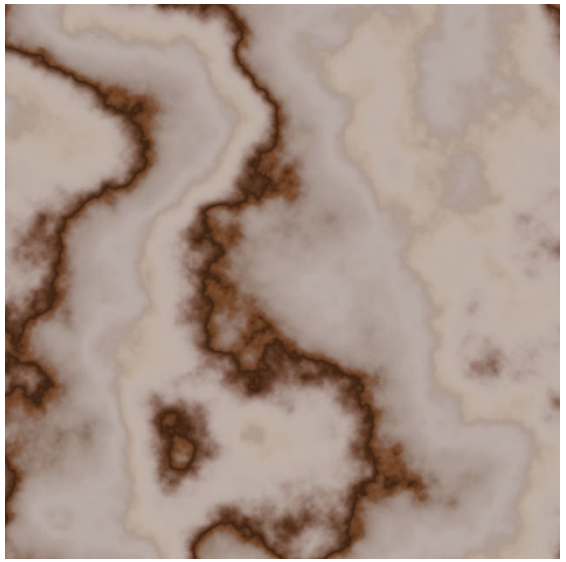
Synthesis of Turbulence (2D)



Example: Marble Texture Function

- Overall structure: alternating layers of white and colored marble
  - $f_{marble}(x,y,z)$:=marble_color(sin(x))
  - marble_color : transfer function (see lower left)
- Realistic appearance: simulated turbulence
  - $f_{marble}(x,y,z)$:=marble_color(sin(x+turbulence(x,y,z)))
- Moving object: turbulence function also transformed





Further Procedural Texturing Applications

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

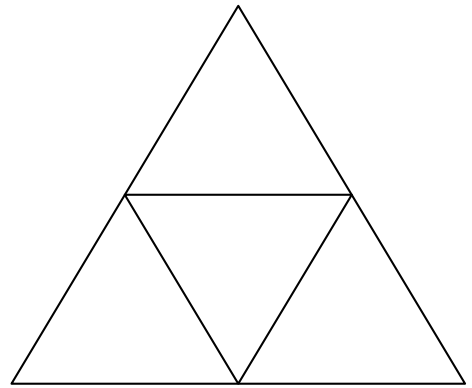
- Bark
  - Turbulated sawtooth function
  - Bump mapping
- Clouds
  - White blobs
  - Turbulated transparency along edge
  - Transparency mapping
- Animation
  - Vary procedural texture function’s parameters over time

Fractal Landscapes

- Procedural generation of geometry
- Complex geometry at virtually no memory cost
  - Can be difficult to ray trace !

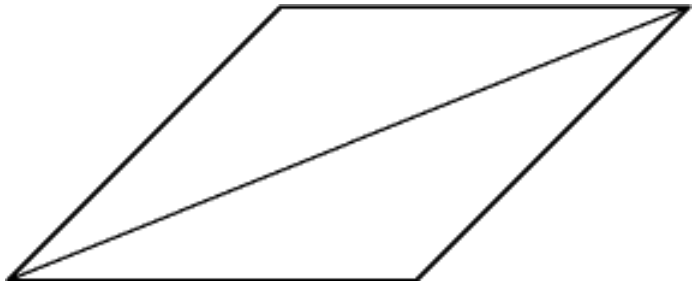
Fractal Landscapes

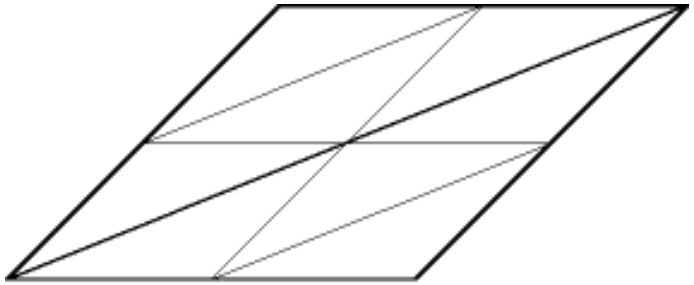
- Coarse triangle mesh approximation
- 1:4 triangle subdivision
  - Vertex insertion at edge-midpoints
- New vertex perturbation
  - Random displacement along normal
  - Scale of perturbation depends on subdivision level
    - Decreasing power spectrum
    - Parameter models surface roughness
- Recursive subdivision
  - Level of detail (LOD) determined by # subdivisions
- All done inside renderer
  - LOD generated locally when/where needed (bounding box test)
  - Minimal I/O cost (coarse mesh only)



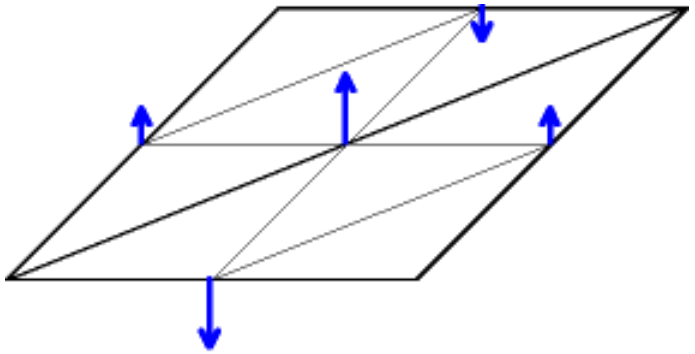
Fractal Landscapes

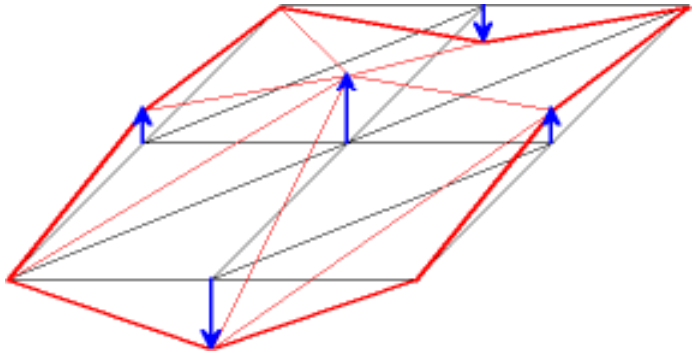
- Triangle subdivision
  - Insert new vertices at edge midpoints
  - 1:4 triangle subdivision





- Vertex displacement
  - Along original triangle normal

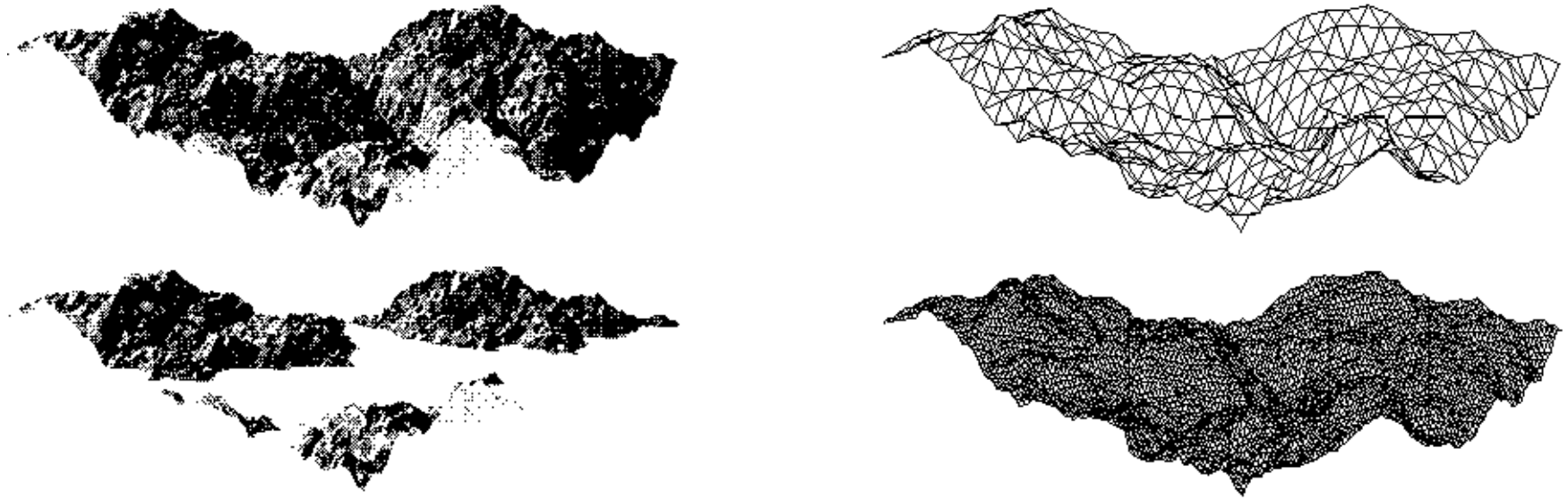




Courtesy http://www.uni-paderborn.de/SFB376/projects/a2/zBufferMerging/

Fractal Landscape Generation

- Base mesh
- Repeated subdivision & vertex displacement
- Shading
- + Water surface
- + Fog
- + …



Courtesy http://www.uwp.edu/academic/computer.science/morris.csci/CS.320/Week.11/Ch11b.www/Ch11b.html

Fractal Landscape Ray Tracing

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Fractal terrain generated on-the-fly
- Problem: where is the ray-surface interaction ?
  - Triangle mesh not a-priori known
- Solution: bounding boxes
  - Maximum possible bounding box around each triangle
  - Decreasing displacement amplitude: finite bounding box
- Algorithm
  - Intersect ray with bounding box
  - Subdivide corresponding triangle
  - Compute bounding boxes of 4 new triangles
  - Test against 4 new bounding boxes
  - Iterate until termination criterion fulfilled (LOD / pixel size)

Questions

- What spaces are involved in texture mapping?
- What is forward and inverse mapping?
- Why do we need filtering in texture mapping?
- How is a mip map organized?
- What are bump mapping and reflection mapping?

Wrap-Up

- Texture Parameterization
  - mapping vertices to texture coordinates
  - problem of perspective transformation

- Procedural Textures
  - Noise
  - Turbulence
  - Fractal Landscapes

- Texture Filtering
  - Minification
  - Magnification
  - EWA
  - Summed Area Tables
  - MipMaps