Image Processing and Computer Graphics

Texture Mapping

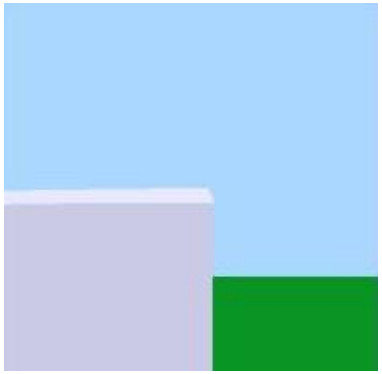
Matthias Teschner

Computer Science Department
University of Freiburg

UNI FREIBURG

Motivation

- adding per-pixel surface details without raising the geometric complexity of a scene
- keep the number of vertices and primitives low
- add detail per fragment
- detail refers to any property that influences the final radiance, e.g. object or light properties
- modeling and rendering time is saved by keeping the geometrical complexity low



scene without texture

scene with color texture

colors can be stored in a texture (represented as an image) and applied to a rendered scene.

[Rosalee Wolfe]

Concept

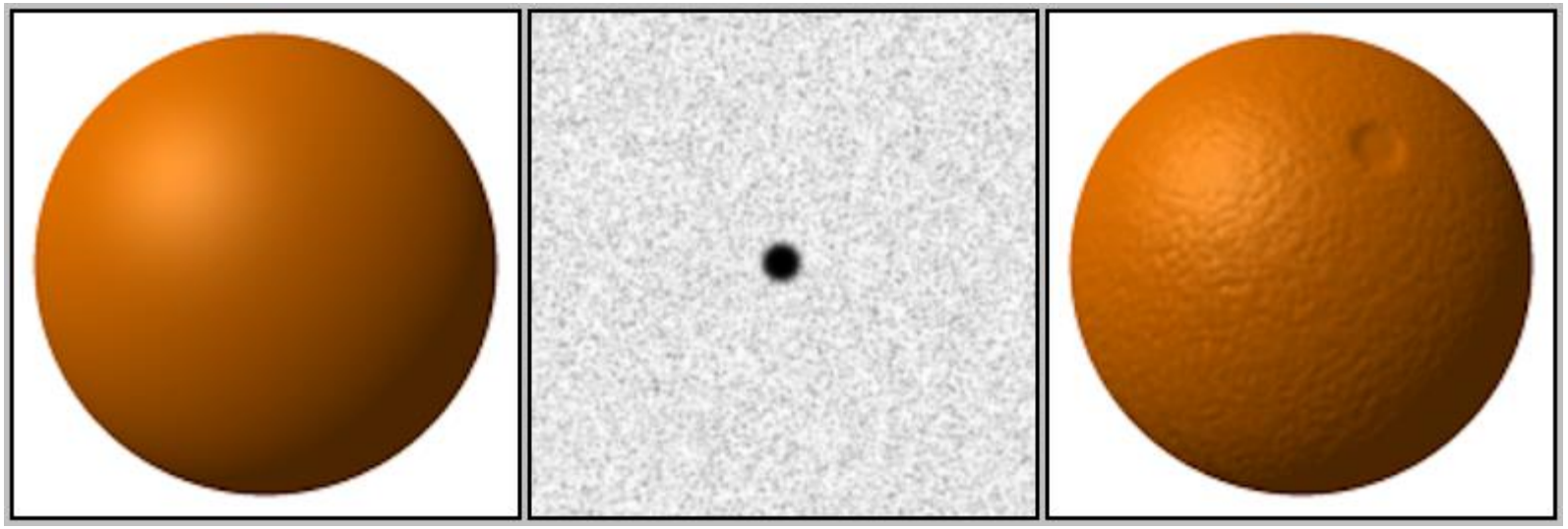
- 2D textures are represented as 2D images
- textures can store a variety of properties, i.e. colors, normals
- positions of texture pixels, i. e. texels, are characterized by texture coordinates (u, v) in texture space
- texture mapping is a transformation from object space to texture space (x, y, z) → (u, v)
  - texture coordinates (u, v) are assigned to a vertex (x, y, z)
- texture mapping is generally applied per fragment
  - rasterization determines fragment positions and interpolates texture coordinates from adjacent vertices
  - texture lookup is performed per fragment using interpolated texture coordinates

Outline

- applications
- concept
- image texturing

Bump Mapping

- normals / normal deviations
  can be stored in a texture



object without
bump mapping

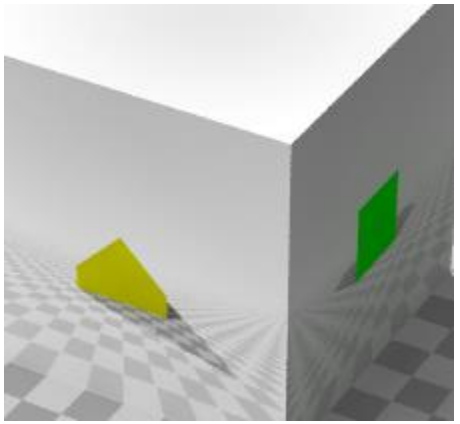
bump texture encodes
normals or normal deviations

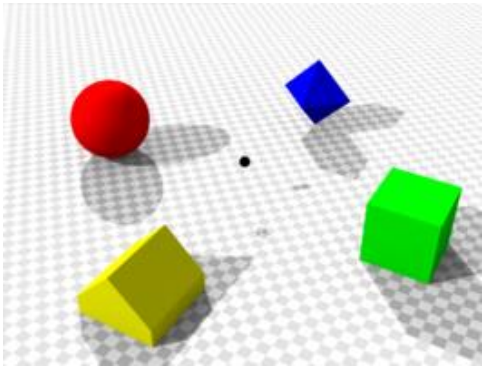
object with
bump mapping

[Wikipedia: Bump Mapping]

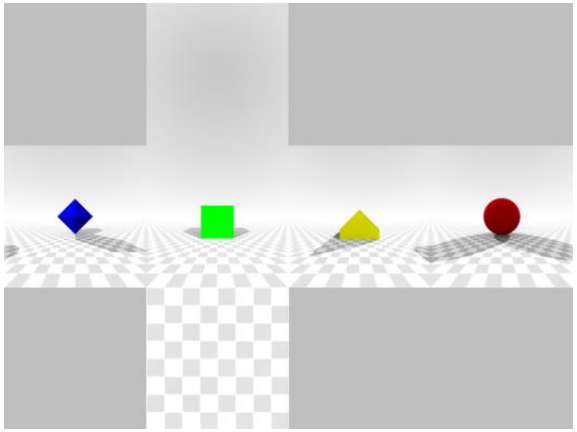
Environment Mapping

- cube mapping
- approximates reflections of the environment

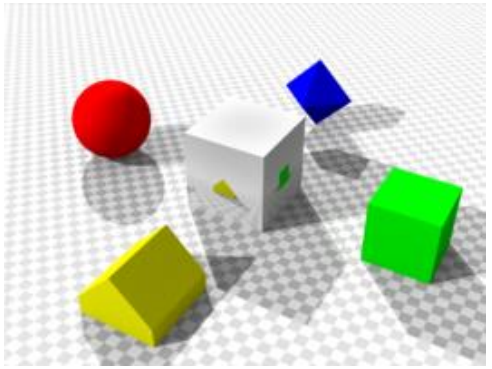




place a viewer in a scene



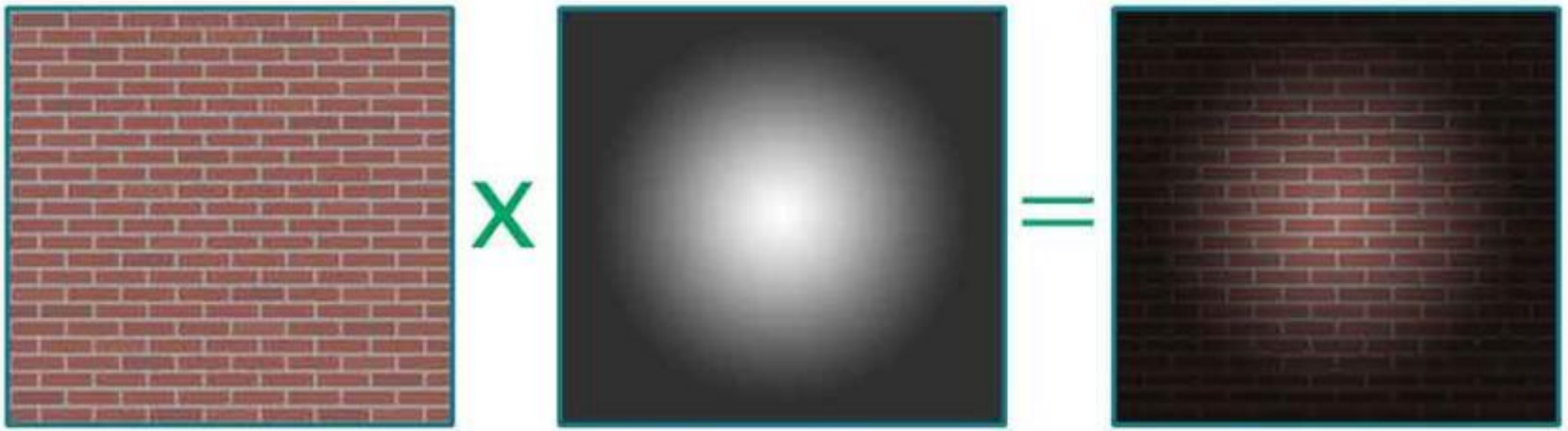
generate the environment texture from six view directions



apply the texture to an object at the position of the viewer

[Wikipedia: Cube Mapping]

Light Mapping



diffuse object color
(color texture)

light map
(light texture)

diffuse reflection component

[Keshav Channa: Light Mapping - Theory and Implementation]

Light Mapping

scene with color texture

scene with color and light texture

[Keshav Channa: Light Mapping - Theory and Implementation]

Billboards

color texture

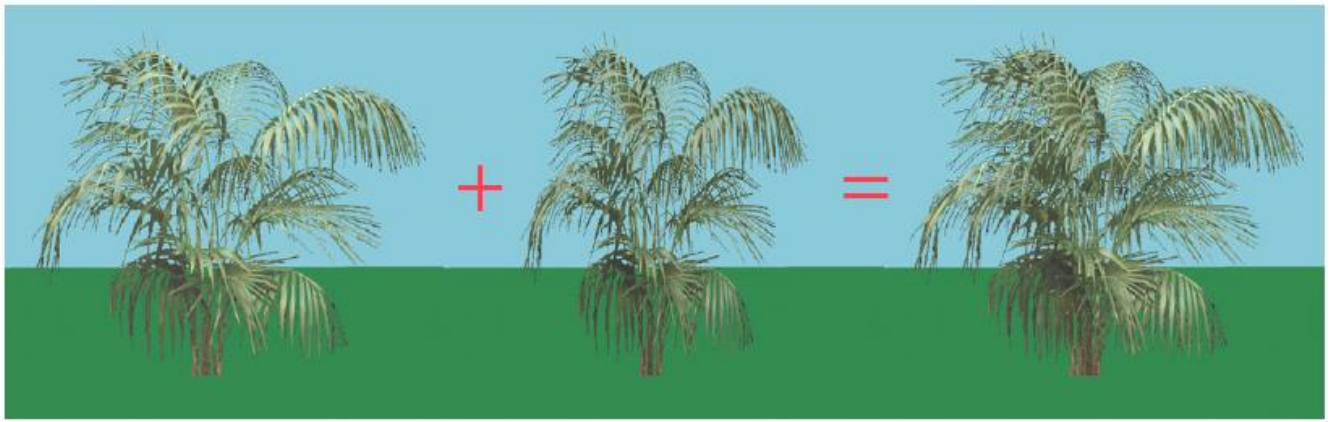
alpha texture representing transparency

billboard: simple primitive with color and alpha mapping

[Virtual Terrain Project: www.vterrain.org]

Billboards

color and
alpha texture



billboards at different orientations

combined 3D billboard

viewed at appropriate directions, a small number of combined 2D billboards provides a realistic 3D impression

[Akenine-Moeller et al.: Real-time Rendering]

Outline

- applications
- concept
- image texturing

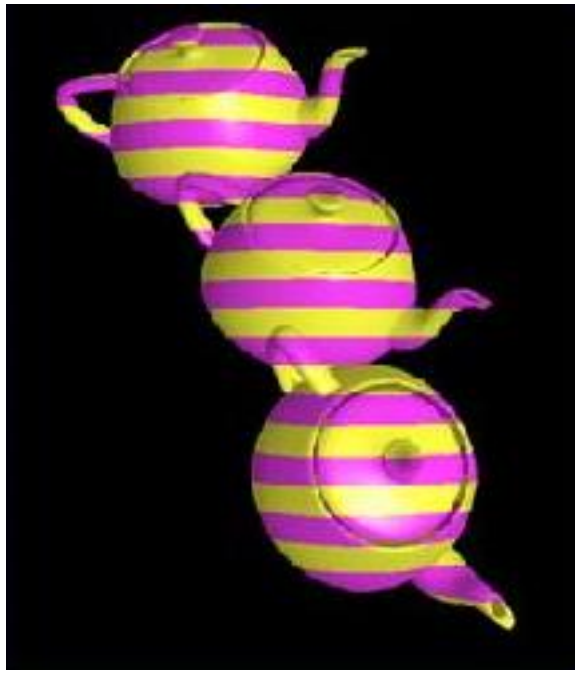
Texturing Pipeline

- object space location
  - texture coordinates are defined for object space locations
  - if the object moves in world space,
    the texture moves along with it
- projector function
  - maps a 3D object space location
    to a (2D) parameter-space value
- corresponder functions
  - map from parameter space to texture space
  - texture-space values are used to obtain values from a texture
- value transform
  - a function that transforms obtained texture values

Object vs. World Space

- if the object moves in world space, the texture should move along with it

texture coordinates assigned to object space positions

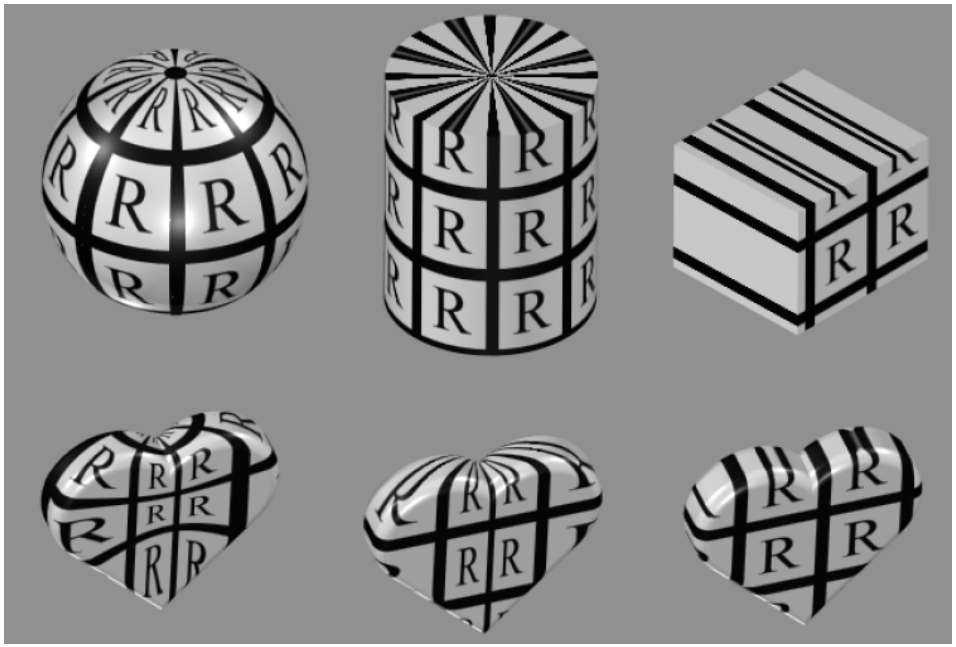


texture coordinates assigned to world space positions

[Rosalee Wolfe]

Projector Functions

- a method to generate texture coordinates, i.e. a method for surface parameterization
- planar projection
  $(x, y, z) \rightarrow (u, v)$
- cylindrical projection
  $(x, y, z) \rightarrow (r, \theta, h) \rightarrow (u, v)$
- spherical projection
  $(x, y, z) \rightarrow (latitude, longitude) \rightarrow (u, v)$
- can be an initialization step for the surface parameterization

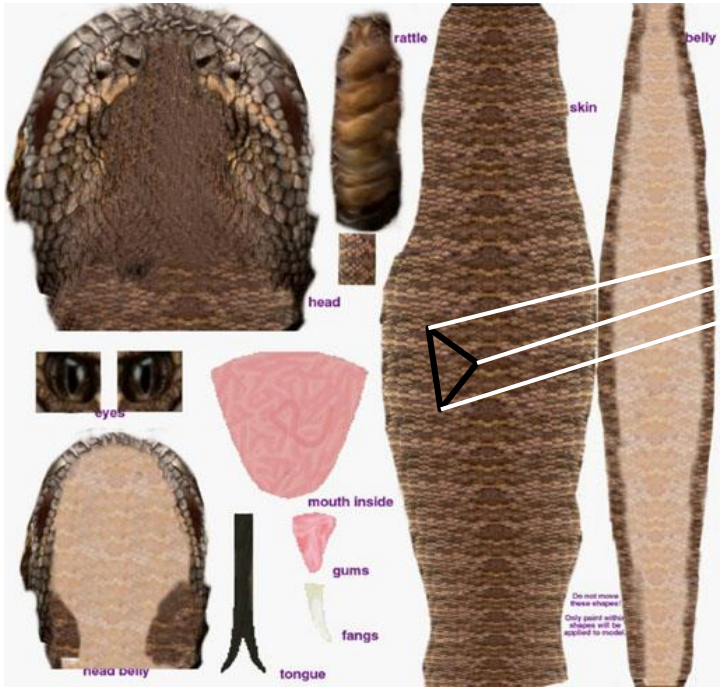


spherical projection | cylindrical projection | planar projection

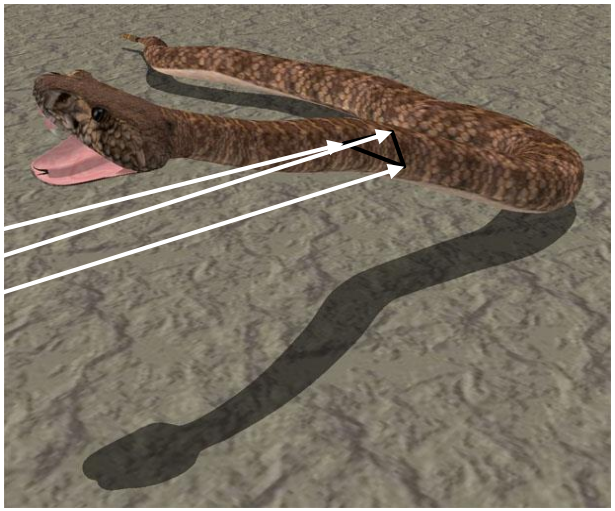
[Akenine-Moeller et al.: Real-time Rendering]

Projector Functions

- use of different projections for surface patches
  - minimize distortions, avoid artifacts at seams
  - several texture coordinates can be assigned to a single vertex



texture

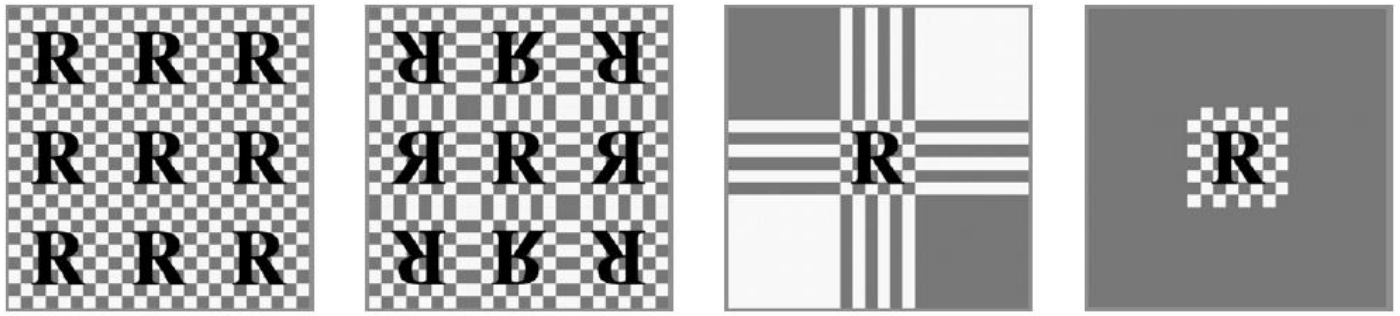


textured object

Corresponder Functions

- one or more corresponder functions transform from parameter space to texture space
- transform of (u, v) into a valid range from 0 to 1, e.g.
  - repeat (the texture repeats itself, integer value is dropped)
  - mirror (the texture repeats itself, but is mirrored)
  - clamp to edge (values outside [0,1] are clamped to this range)
  - clamp to border (values outside [0,1] are rendered with a border color)

[Akenine-Moeller et al.: Real-time Rendering]



Value Transform / Texture Blending Functions

- value transform
  - arbitrary transformations, e.g. represented by a matrix in homogeneous form, can be applied to texture values
- texture blending functions
  - define how to use the texture value, e.g.
  - replace the original surface color with the texture color
  - linearly combine the original surface color with the texture
  - multiply, add, subtract surface color and texture color
- alternatively, texture values can be used in the computation of the illumination model

Outline

- applications
- concept
- image texturing
  - perspective-correct interpolation
  - magnification
  - minification

Image Texturing

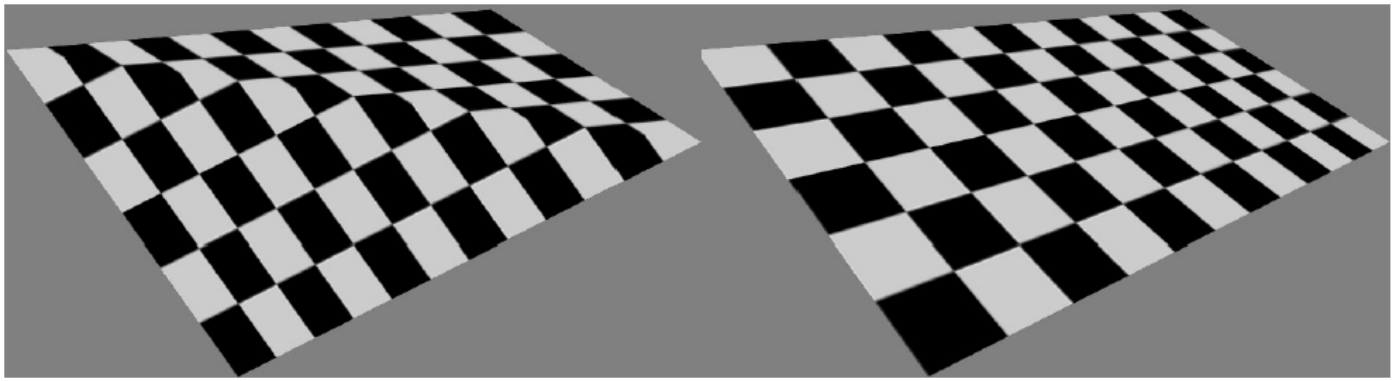
- 2D image is mapped / glued to a triangulated object surface
- certain aspects affect the quality
- interpolation of texture coordinates
  - has to consider the nonlinear transformation of the z-component from camera space to clip space in case of perspective projection
- sampling
  - textures can be change their size when applied to the object (magnification / minification)

Outline

- applications
- concept
- image texturing
  - perspective-correct interpolation
  - magnification
  - minification

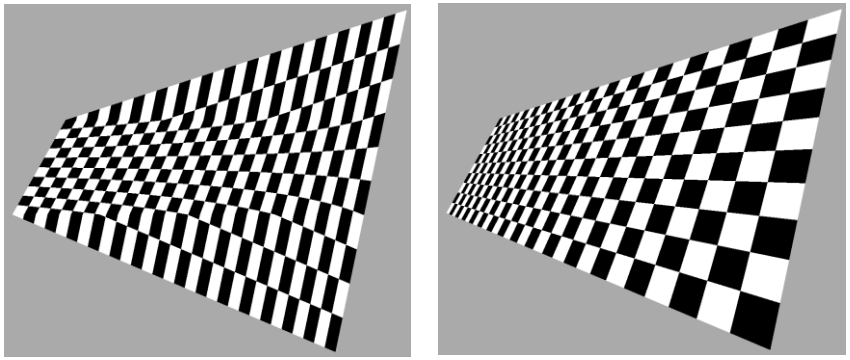
Linear vs. Perspective-Correct Interpolation

- two triangles in the same plane



[Akenine-Moeller et al.: Real-time Rendering]

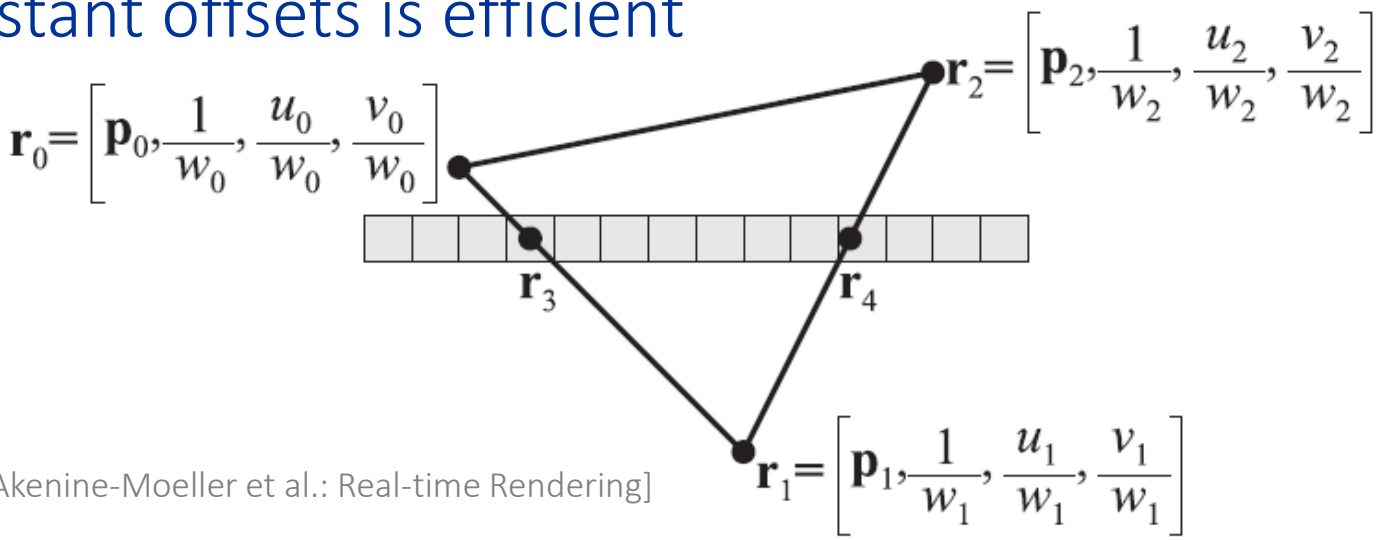
- a quadrilateral



[Heckbert, Moreton]

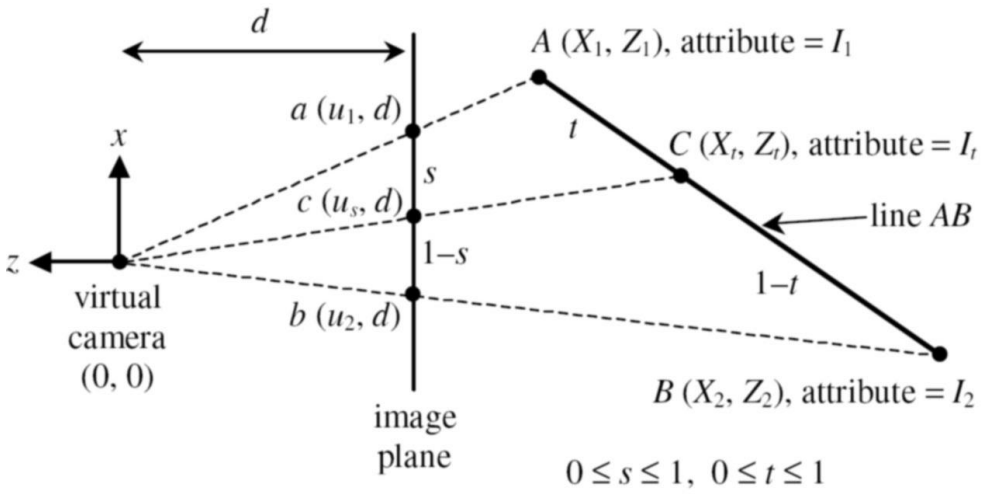
Perspective-Correct Interpolation Solution

- u and v are not linearly interpolated
- instead, u / w_{clip}, v / w_{clip}, and $1 / w_{clip}$ are interpolated
- the resulting interpolated values for u / w_{clip} and v / w_{clip} are divided by the interpolated value for $1 / w_{clip}$
- motivated by the fact that linear interpolation with constant offsets is efficient



[Akenine-Moeller et al.: Real-time Rendering]

Perspective-Correct Interpolation



Perspective projection of a line AB. t / (1-t) is not equal to s / (1-s). Therefore, linear interpolation between a and b does not correspond to a linear interpolation between A and B.

$$I_t = I_1 + t(I_2 - I_1)$$

linear interpolation in camera space

$$I_t = I_1 + \frac{sZ_1}{sZ_1+(1-s)Z_2}(I_2 - I_1)$$

non-linear interpolation in screen space

$$I_t = \frac{\frac{I_1}{Z_1}+s\left(\frac{I_2}{Z_2}-\frac{I_1}{Z_1}\right)}{\frac{1}{Z_1}+s\left(\frac{1}{Z_2}-\frac{1}{Z_1}\right)}$$

linear interpolation of I / Z and 1 / Z in screen space

[Kok-Lim Low: Perspective-Correct Interpolation]

Perspective-Correct Interpolation

- perspective projection transform

$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} X_{camera} \\ Y_{camera} \\ Z_{camera} \\ 1 \end{pmatrix}$$

- linear relation between w in clip space and Z in camera space $w_{clip} = -Z_{camera}$
- therefore, Z_{camera} or w_{clip} can be used in the interpolation

$$I_t = \frac{\frac{I_1}{Z_1} + s\left(\frac{I_2}{Z_2} - \frac{I_1}{Z_1}\right)}{\frac{1}{Z_1} + s\left(\frac{1}{Z_2} - \frac{1}{Z_1}\right)} \qquad I_t = \frac{\frac{I_1}{w_1} + s\left(\frac{I_2}{w_2} - \frac{I_1}{w_1}\right)}{\frac{1}{w_1} + s\left(\frac{1}{w_2} - \frac{1}{w_1}\right)}$$

Perspective-Correct Interpolation

- is only an issue for perspective projection
  - for parallel projection, linear interpolation in screen space corresponds to linear interpolation in camera space
- is not an issue for lines and polygons parallel to the near plane
- can be applied to all vertex attributes, i.e. color, depth, normal, texture coordinates
- interp. is commonly based on w_{clip} instead of Z_{camera}
  - Z_{camera} is not available in screen space
  - w_{clip} can be kept when converting from clip to NDC space

Outline

- applications
- concept
- image texturing
  - perspective-correct interpolation
  - magnification
  - minification

Magnification

- adjacent pixel positions in window space map to the same texel, i.e. pixel position in an image texture
- texture is magnified

[Akenine-Moeller et al.: Real-time Rendering]

nearest neighbor
(nearest texel)

bilinear interpolation
(weighted average
of 2x2 texels)

bicubic interpolation
(weighted average
of 5x5 texels)

Filtering

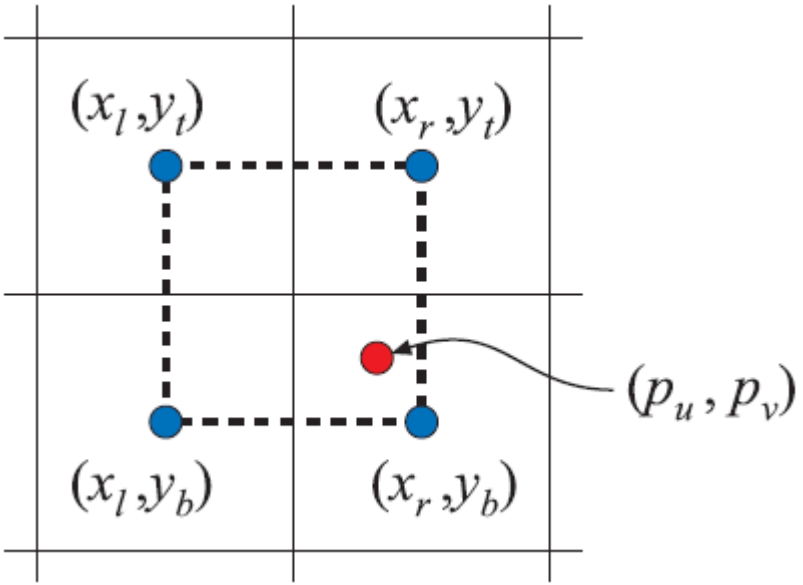
- bilinear interpolation

$$(u', v') = (p_u - \lfloor p_u \rfloor, p_v - \lfloor p_v \rfloor)$$

relative position within four adjacent texel positions

$$\begin{aligned}\mathbf{b}(p_u, p_v) = (1-u')(1-v')\mathbf{t}(x_l, y_b) \\ +u'(1-v')\mathbf{t}(x_r, y_b) \\ +(1-u')v'\mathbf{t}(x_l, y_t) \\ +u'v'\mathbf{t}(x_r, y_t)\end{aligned}$$

b (p_u, p_v) is interpolated from four texels
t (x_l, y_b), t (x_r, y_b), t (x_l, y_t), t (x_r, y_t)

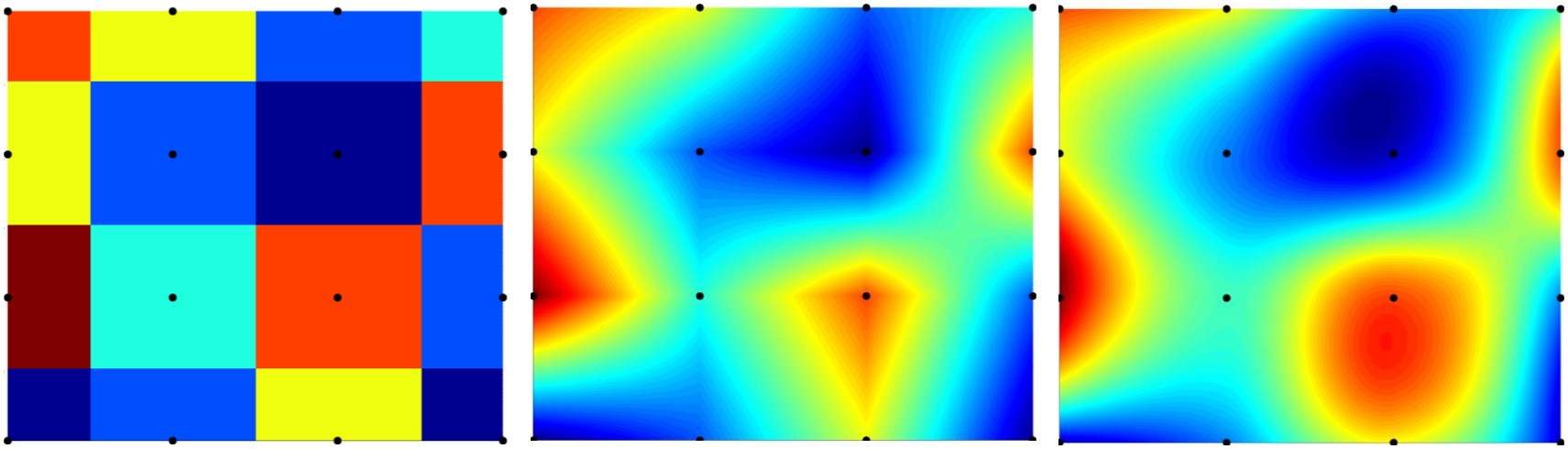


[Akenine-Moeller et al.: Real-time Rendering]

- bicubic interpolation

$$\mathbf{b}(p_u, p_v) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} (u')^i (v')^j$$

Examples



nearest neighbor
(nearest texel)

bilinear interpolation
(weighted average
of 2x2 texels)

bicubic interpolation
(weighted average
of 2x2 texels)

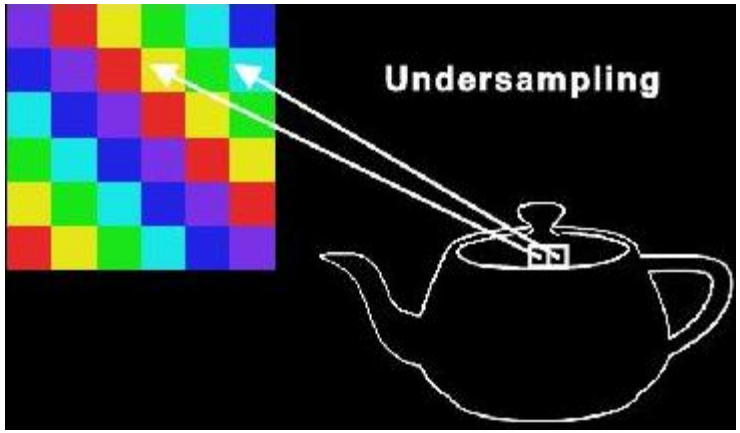
[Wikipedia: Bicubic interpolation]

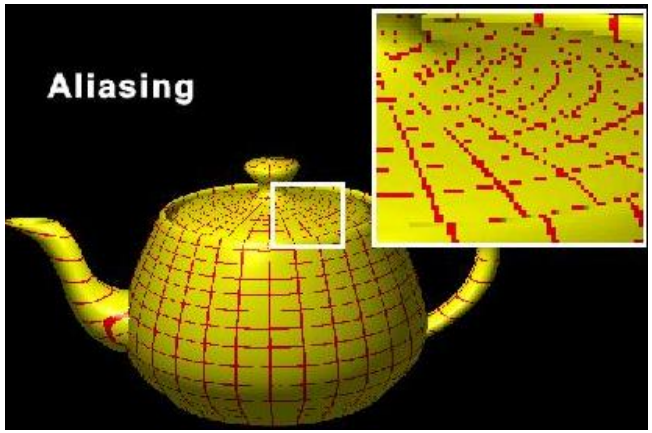
Outline

- applications
- concept
- image texturing
  - perspective-correct interpolation
  - magnification
  - minification

Minification

- adjacent texels map to the same pixel position in window space
- texture is minimized
- adjacent pixel positions do not map to adjacent texels (undersampling), which can cause aliasing

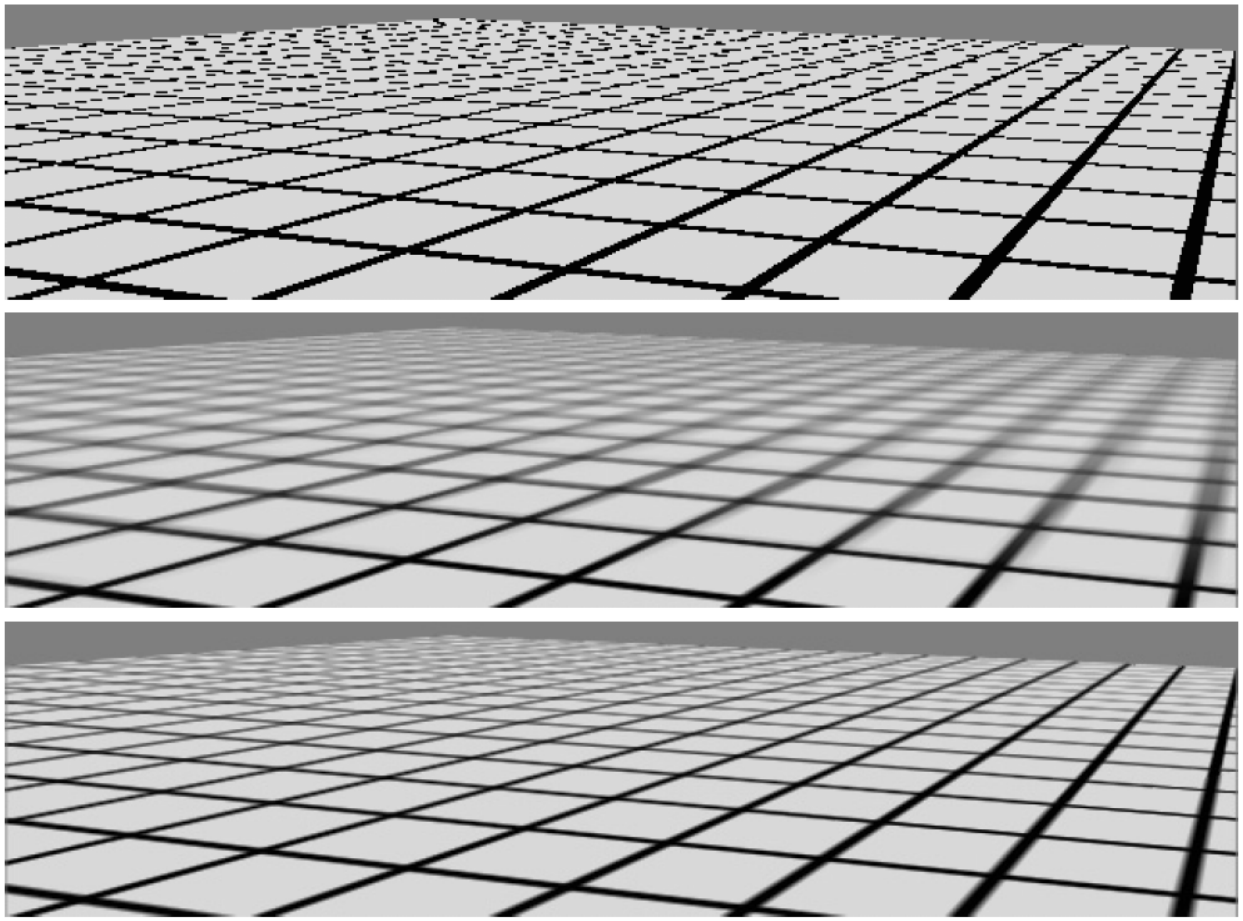




Undersampling can lead to artifacts, i. e. aliasing, as information from the texture is lost

[Rosalee Wolfe]

Filtering



nearest neighbor

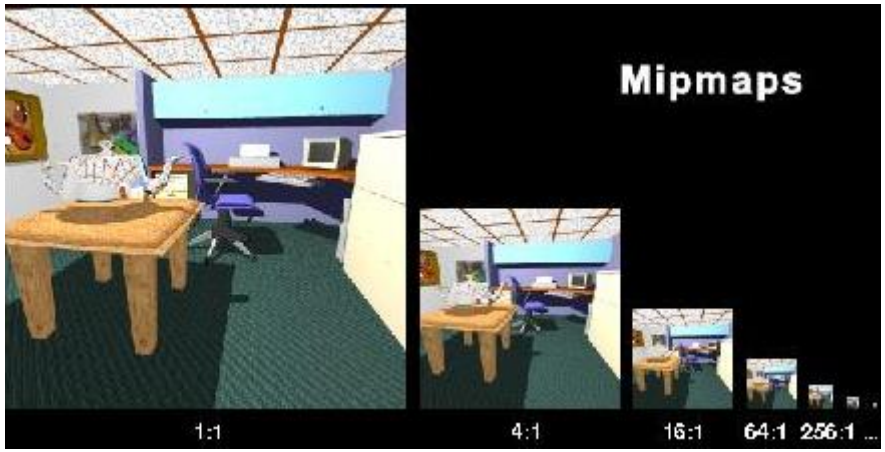
mipmapping

summed area tables

[Akenine-Moeller et al.: Real-time Rendering]

Mipmapping

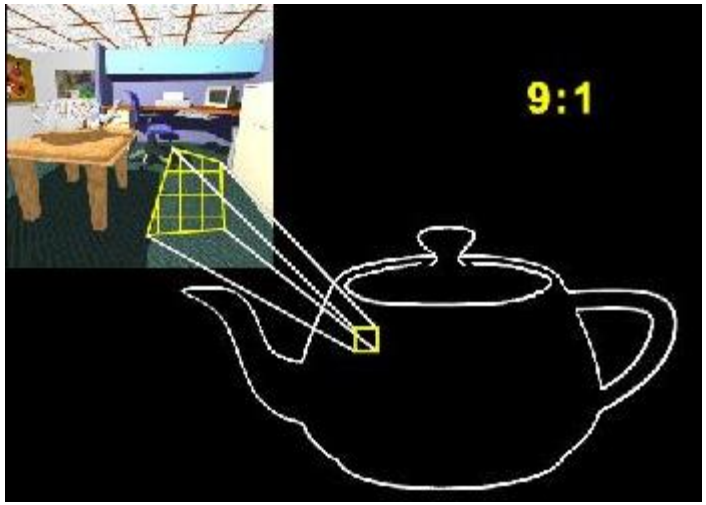
- mip - multum in parvo, many things in a small space
- a set of smaller, low-pass filtered textures is generated



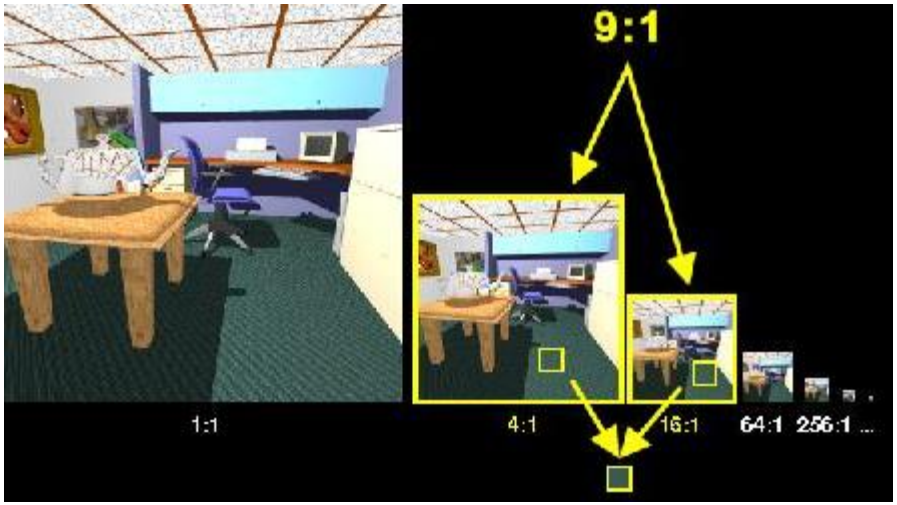
[Rosalee Wolfe]

- mipmaps can be extended to ripmaps with sub-textures of rectangular areas to avoid overblurring

Mipmapping



estimation of the texture area covered by a pixel in window space, e. g. by considering the texture coordinates of 2x2 pixels in window space

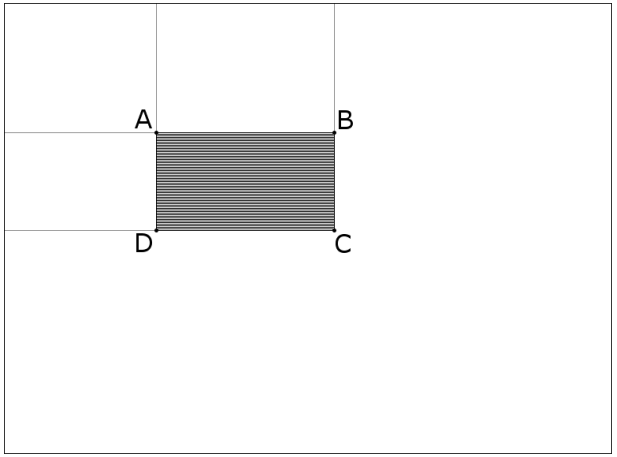


choosing the appropriate subtexture, applying additional filters, i. e. interpolation of texels of two adjacent subtextures.

[Rosalee Wolfe]

Summed Area Table

- reduced overblurring due to the consideration of a more accurate texture area
- generate a second texture that stores $\mathbf{sum}(x, y) = \sum_{x' \leq x, y' \leq y} \mathbf{t}(x', y')$
- the sum of all texels within the rectangle A, B, C, D is $\mathbf{sum(C)} - \mathbf{sum(B)} - \mathbf{sum(D)} + \mathbf{sum(A)}$
- if a pixel in window space covers rectangle A, B, C, D in the texture, the texture value can be, e. g. , $\mathbf{b}(x, y) = \frac{\mathbf{sum(C)} - \mathbf{sum(B)} - \mathbf{sum(D)} + \mathbf{sum(A)}}{\mathrm{area}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})}$



[Wikipedia: Summed area table]

Summary

- textures add detail without raising the geom.complexity
- textures can influence a variety of properties
- textures can be 1D, 2D, 3D, …, or procedural
- texture coordinates at vertices or fragments are used to lookup texels
- quality of applied textures can be improved by
  - perspective-correct interpolation
  - considering magnification and minification
- examples
  - color, alpha, environment mapping, light, bump, parallax, relief mapping, …