Texture mapping

Computer Graphics

CSE 167

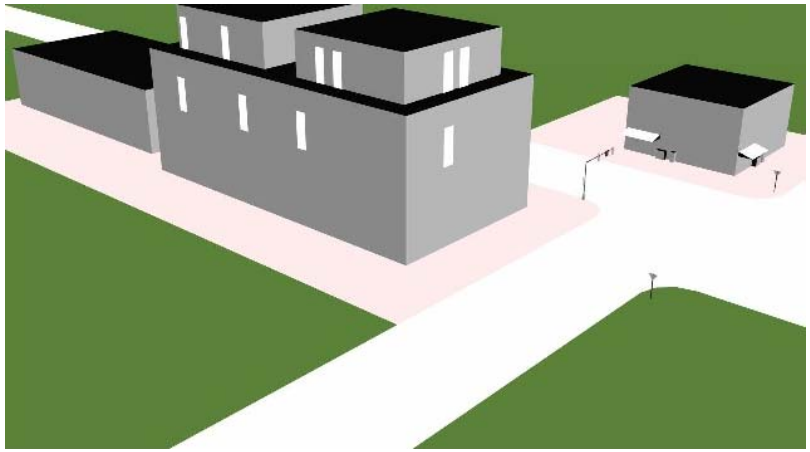
Lecture 9

CSE 167: Computer Graphics

- Texture Mapping
  - Overview
  - Interpolation
  - Wrapping
  - Texture coordinates
  - Anti-aliasing
    - Mipmaps
  - Other mappings
    - Including bump mapping

Large triangles

- Advantages
  - Often sufficient for simple geometry
  - Fast to render
- Disadvantages
  - Per vertex colors look boring and computer-generated

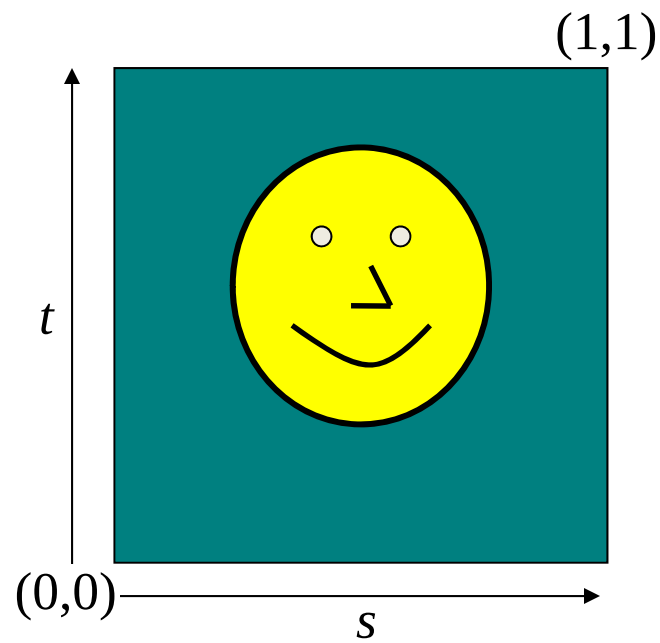


Texture mapping

- Map textures (images) onto surface polygons
- Same triangle count, much more realistic appearance

Texture mapping

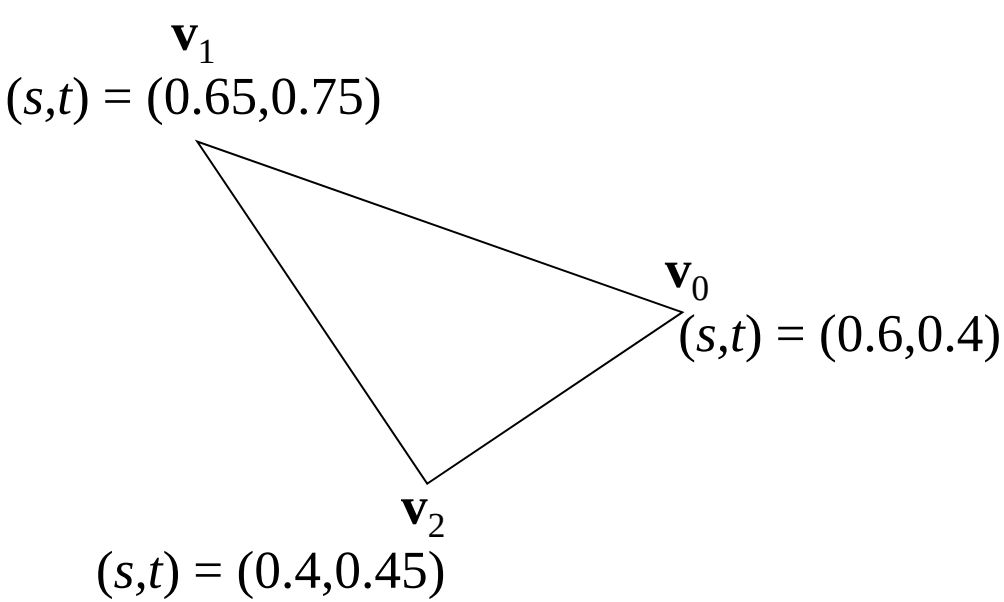
- Objective: map locations in texture to locations on 3D geometry
- Texture coordinate space
  - Texture pixels (texels) have texture coordinates (s,t)
- Convention
  - Bottom left corner of texture is at (s,t) = (0,0)
  - Top right corner is at (s,t) = (1,1)



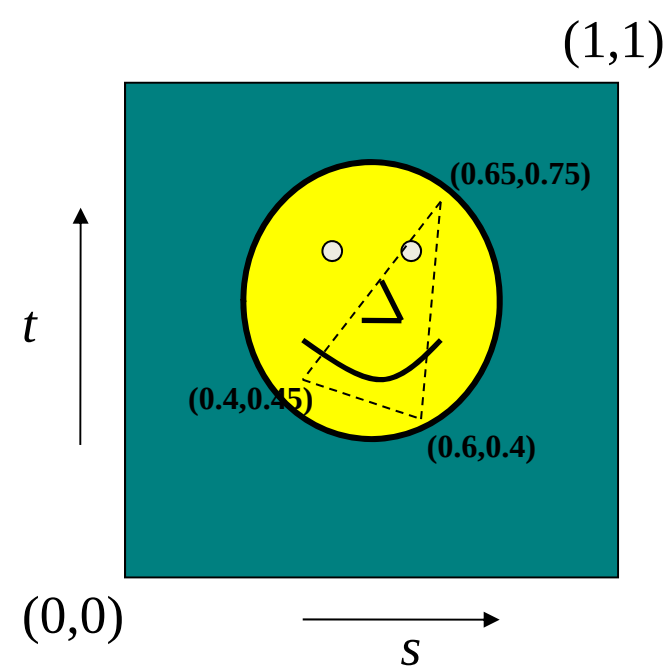
Texture coordinates

Texture mapping

- Store 2D texture coordinates *s*,*t* with each triangle vertex



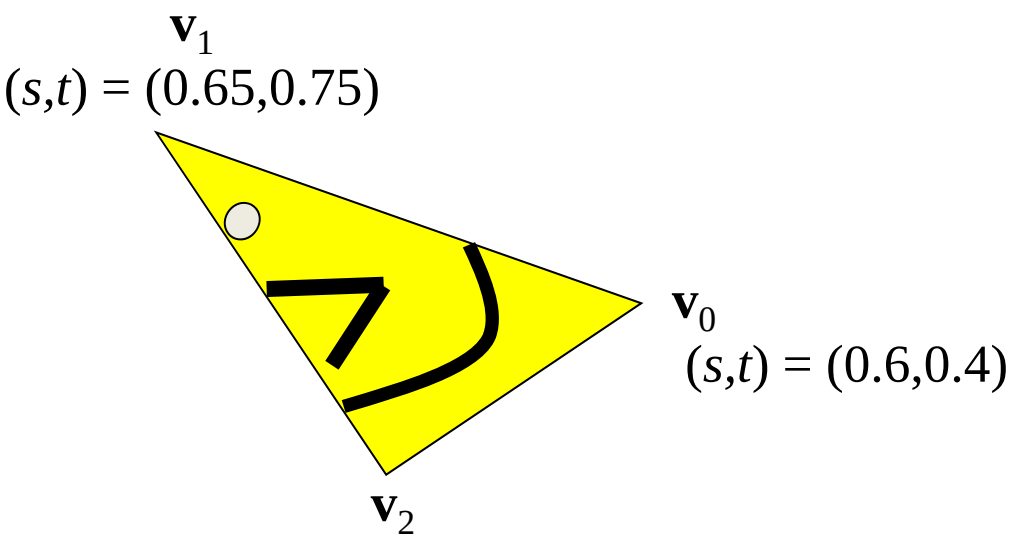
Triangle in any space before projection



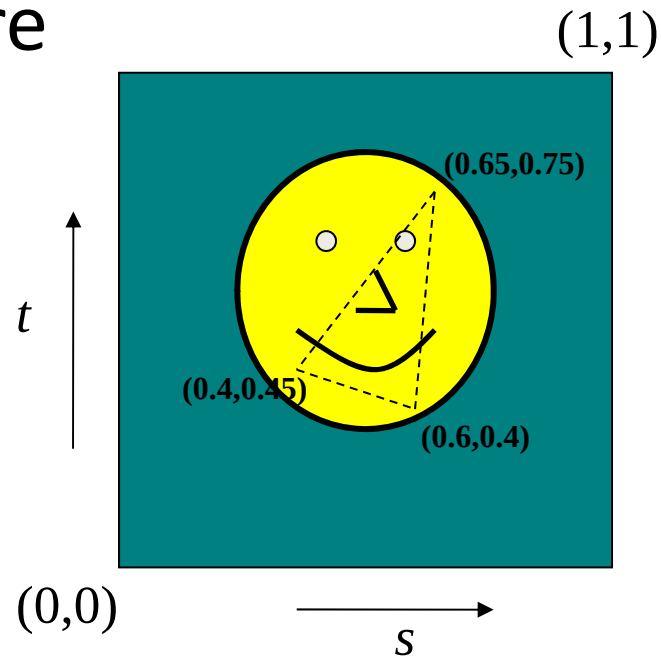
Texture coordinates

Texture mapping

- Each point on triangle gets color from its corresponding point in texture

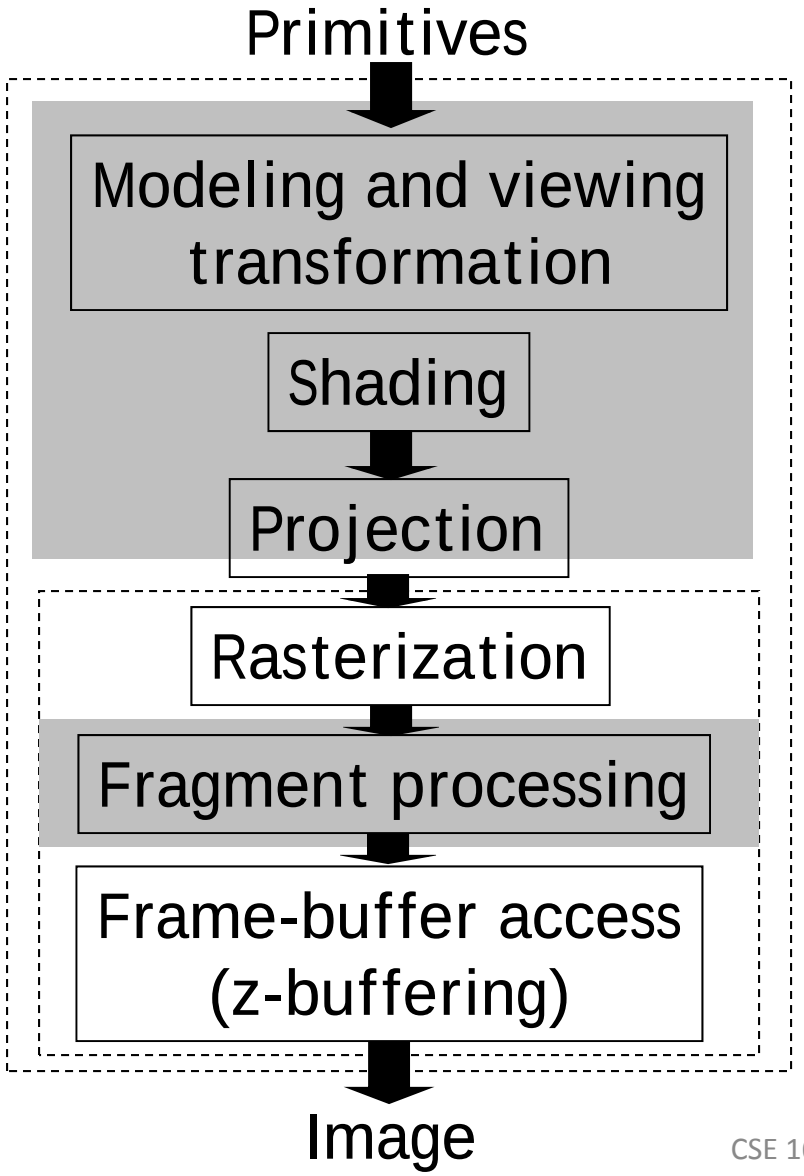


Triangle in any space before projection



Texture coordinates

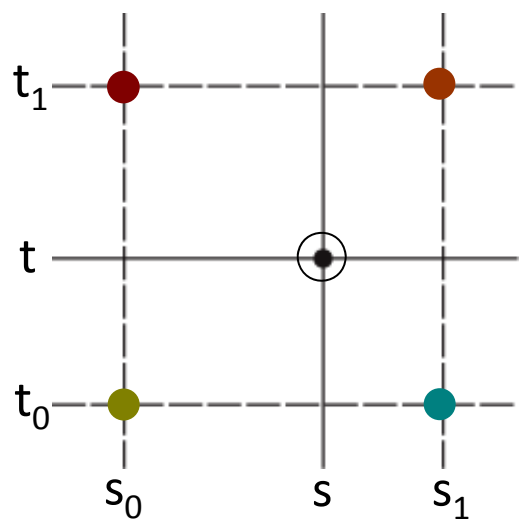
Texture mapping



Includes texture mapping

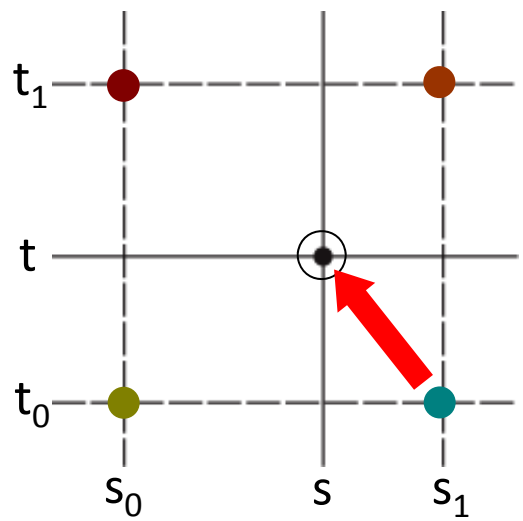
Texture look-up

- Given texture coordinates (s,t) at current pixel
- Closest four texels in texture space are at (s_0,t_0), (s_1,t_0), (s_0,t_1), (s_1,t_1)
- How to compute pixel color? Interpolate



Nearest neighbor interpolation

- Use color of closest texel



- Simple, but low quality and aliasing

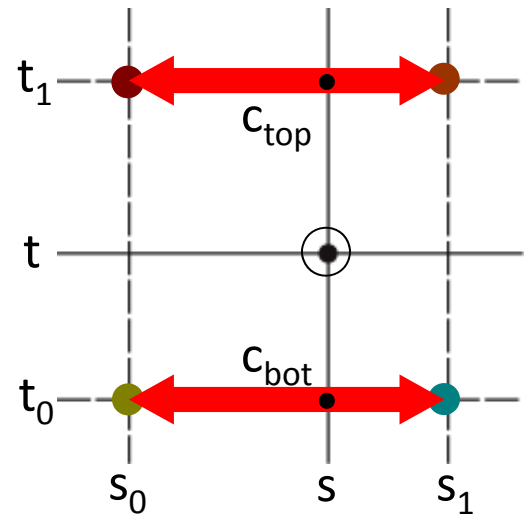
Bilinear interpolation

1. Linear interpolation horizontally

   Ratio in *s* direction is $r_s = \dfrac{s - s_0}{s_1 - s_0}$

   $c_{\text{top}} = \text{tex}(s_0,t_1)\,(1-r_s) + \text{tex}(s_1,t_1)\,r_s$

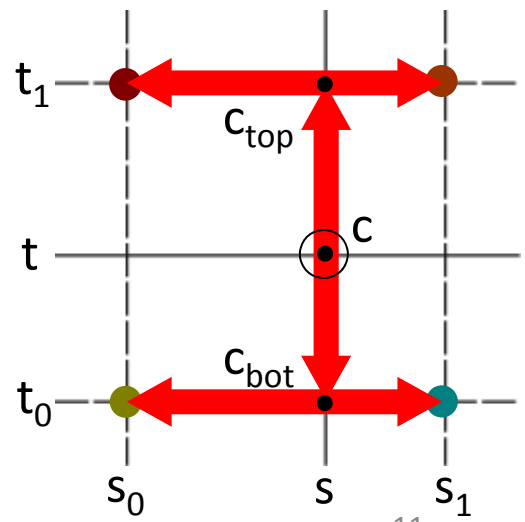
   $c_{\text{bot}} = \text{tex}(s_0,t_0)\,(1-r_s) + \text{tex}(s_1,t_0)\,r_s$



2. Linear interpolation vertically

   Ratio in *t* direction is $r_t = \dfrac{t - t_0}{t_1 - t_0}$

   $c = c_{\text{bot}}\,(1-r_t) + c_{\text{top}}\,r_t$



Interpolation

Nearest neighbor

Bilinear

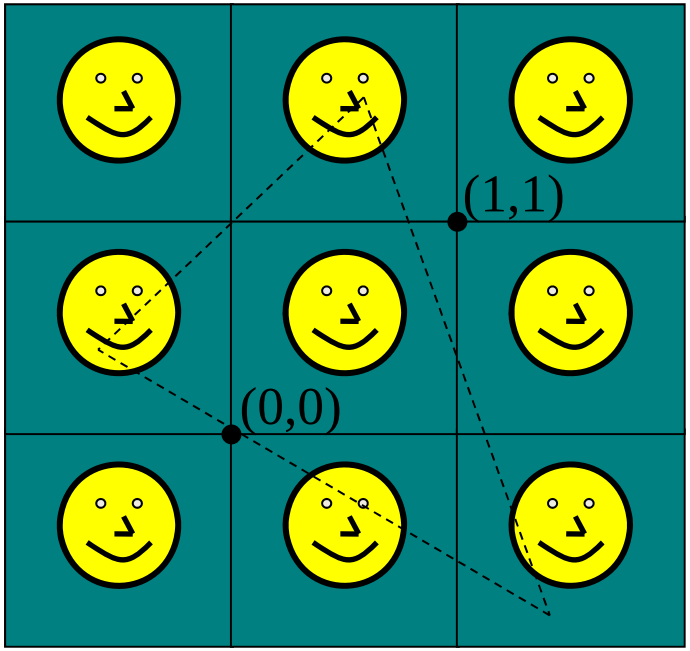
- OpenGL
  - GL_NEAREST: Nearest neighbor interpolation
  - GL_LINEAR: Bilinear interpolation

Wrapping

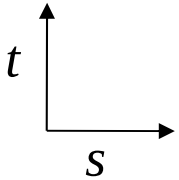
- Texture image extends from [0,0] to [1,1] in texture space
  - What if (s,t) texture coordinates are beyond that range? Wrap
    - Repeat (and optionally mirror) texture
    - Clamp texture

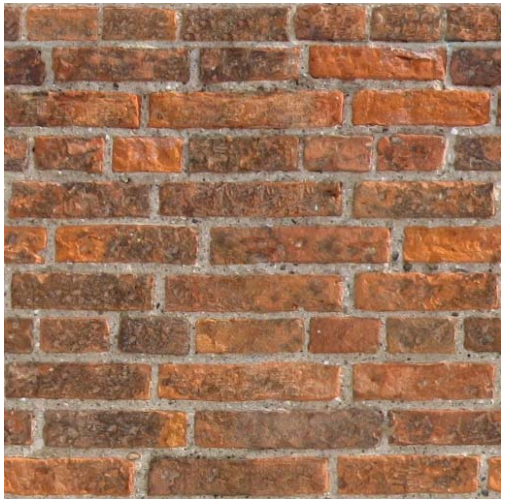
Wrapping: repeat

- Repeat the texture
  - Creates discontinuities at edges, unless texture is designed to line up



Texture Space

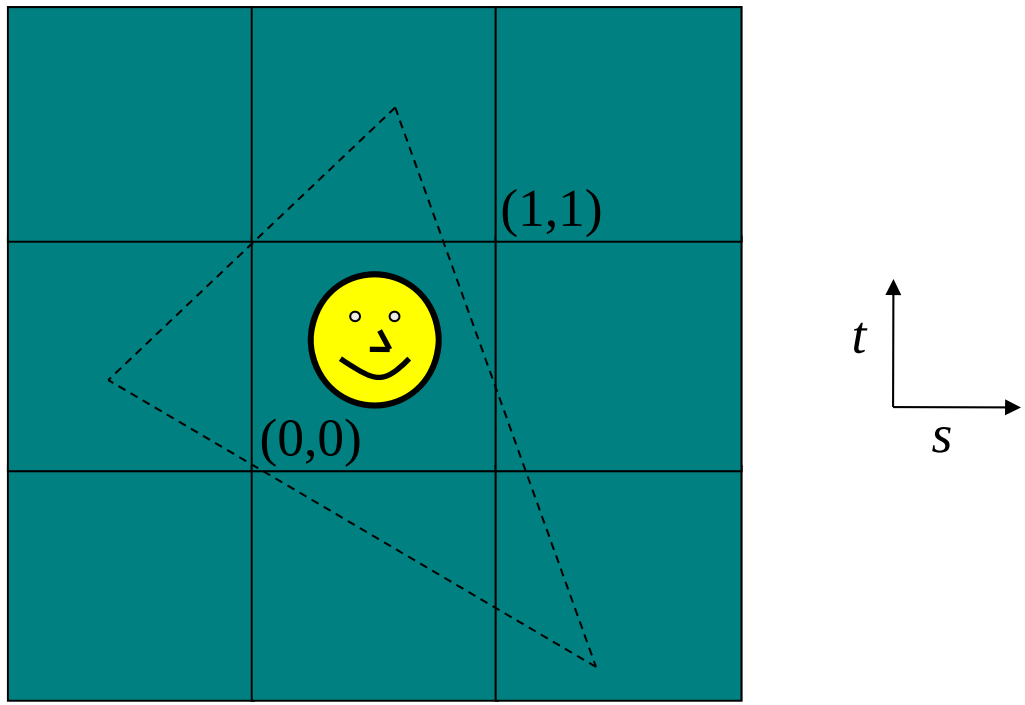




Seamless brick wall texture
(by Christopher Revoir)

Wrapping: clamp

- Use edge or specified color everywhere outside data range [0..1]



Texture Space

Wrapping

Repeat

Repeat and mirror

Clamp to edge color

Clamp to specified color

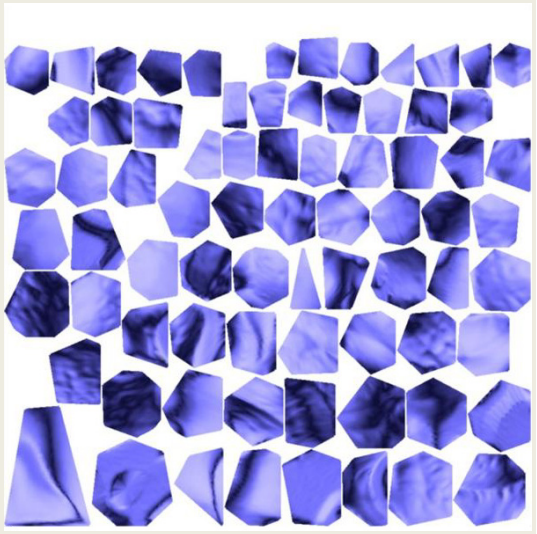
- OpenGL
  - GL_REPEAT
  - GL_MIRRORED_REPEAT
  - GL_CLAMP_TO_EDGE: repeats last pixel in the texture
  - GL_CLAMP_TO_BORDER: requires setting border color

Texture coordinates

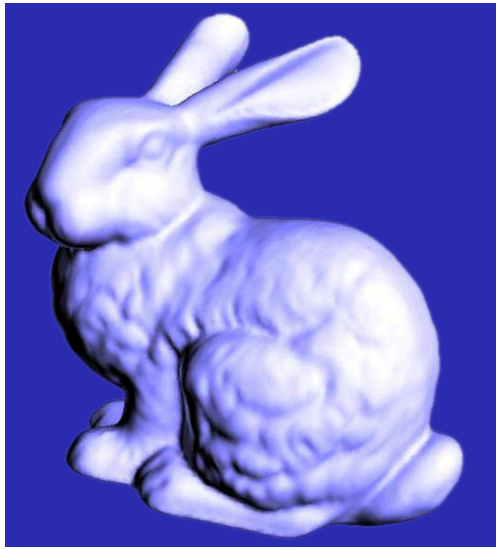
- What if texture extends across multiple polygons? Parameterize surface
  - Mapping between 3D positions on surface and 2D texture coordinates
    - Defined by texture coordinates of triangle vertices
  - Example mappings
    - Cylindrical
    - Spherical
    - Orthographic
    - Cube
    - Parametric
    - Skin

Texture atlas

charts



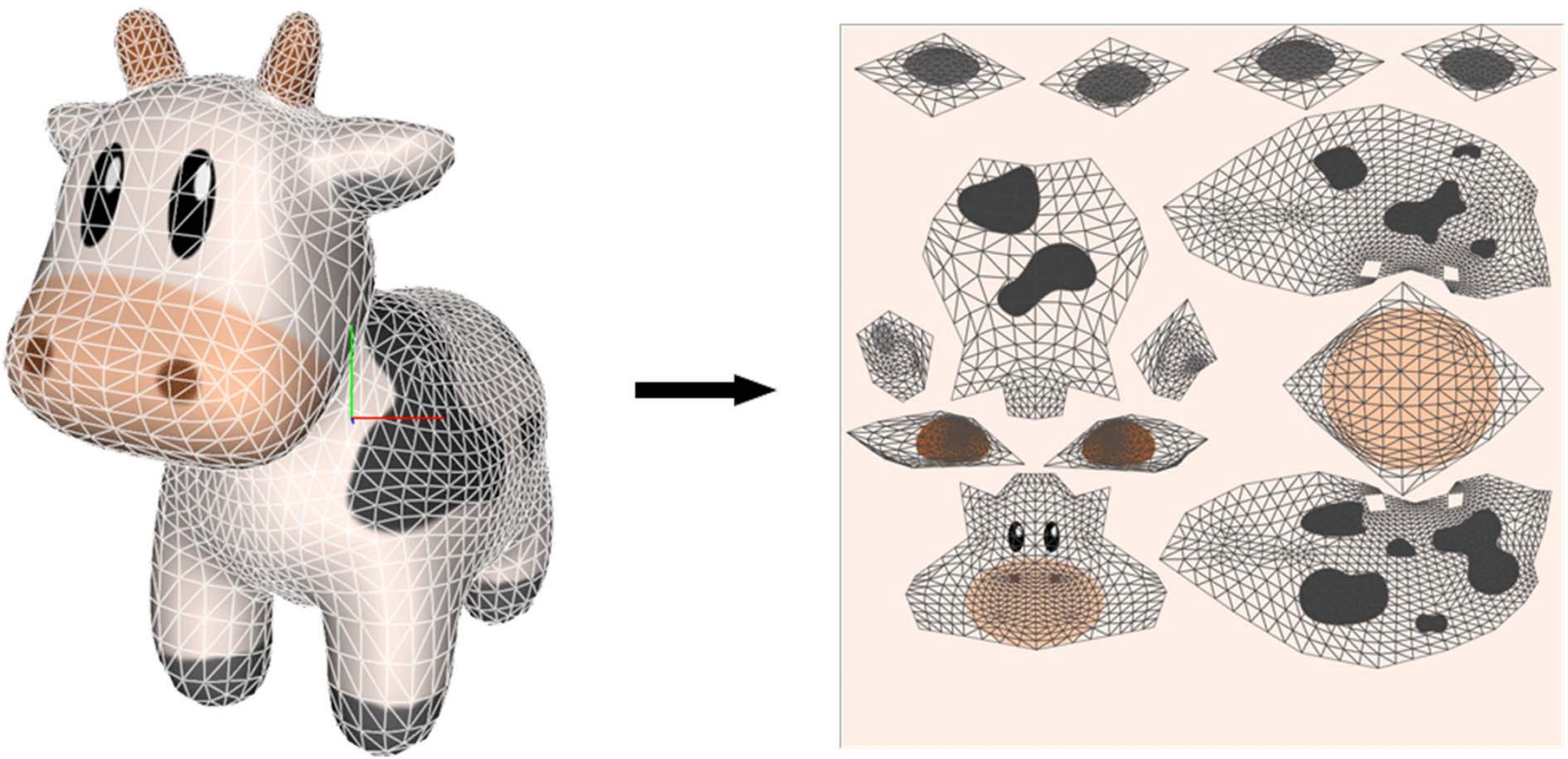
atlas



surface

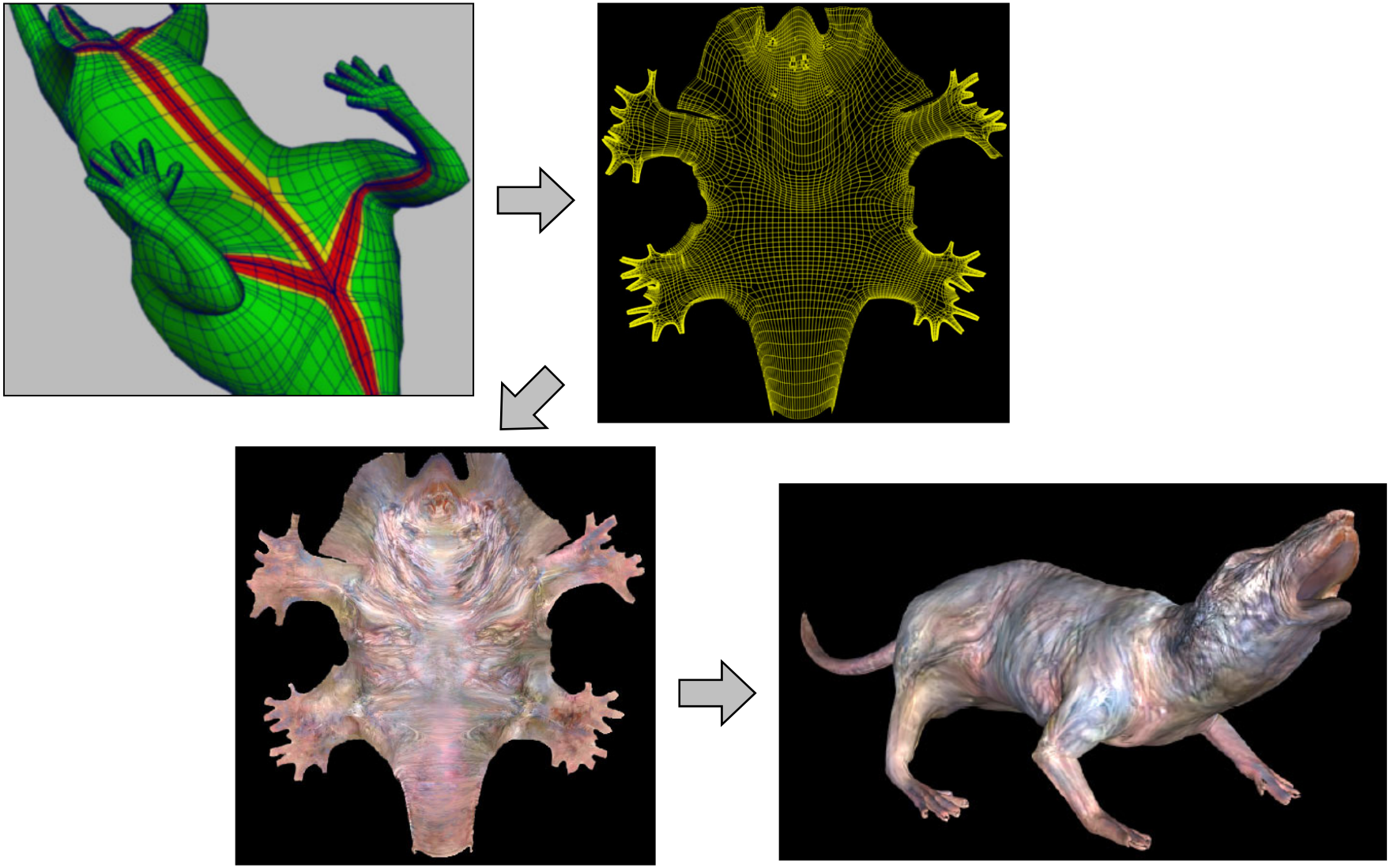
[Sander2001]

Skin mapping



Skin mapping

Skin mapping

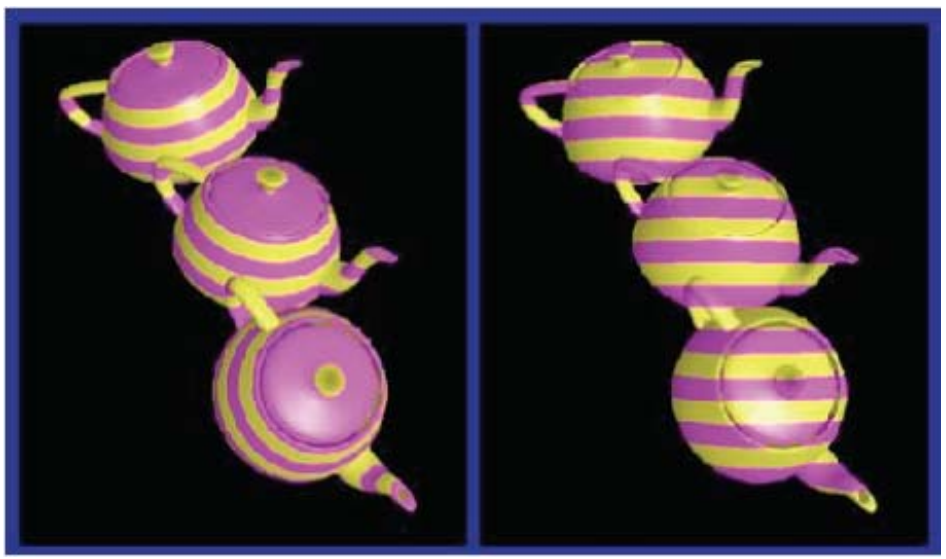


Skin mapping

Orthographic mapping

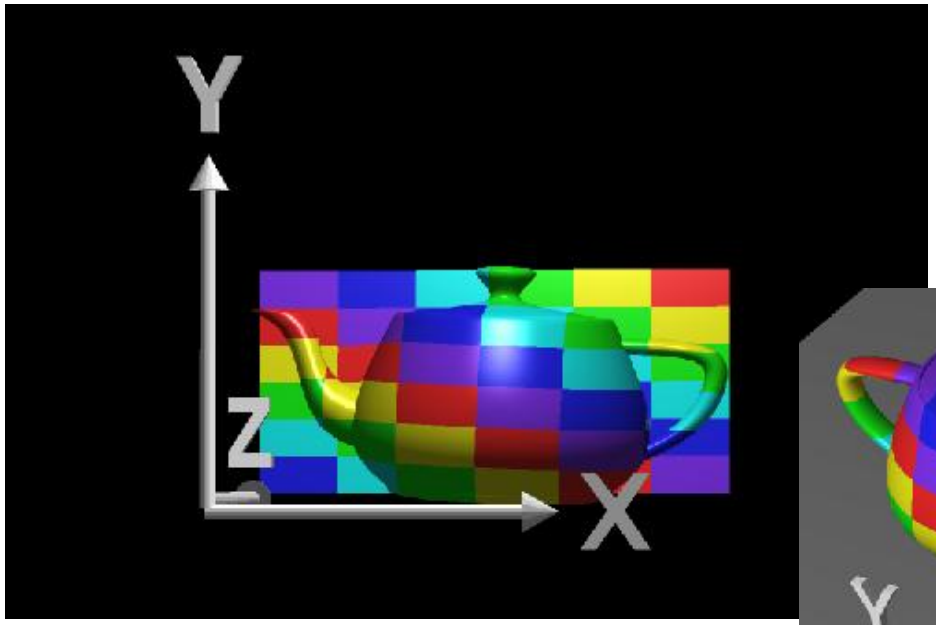
- Use linear transformation of object's XYZ coordinates

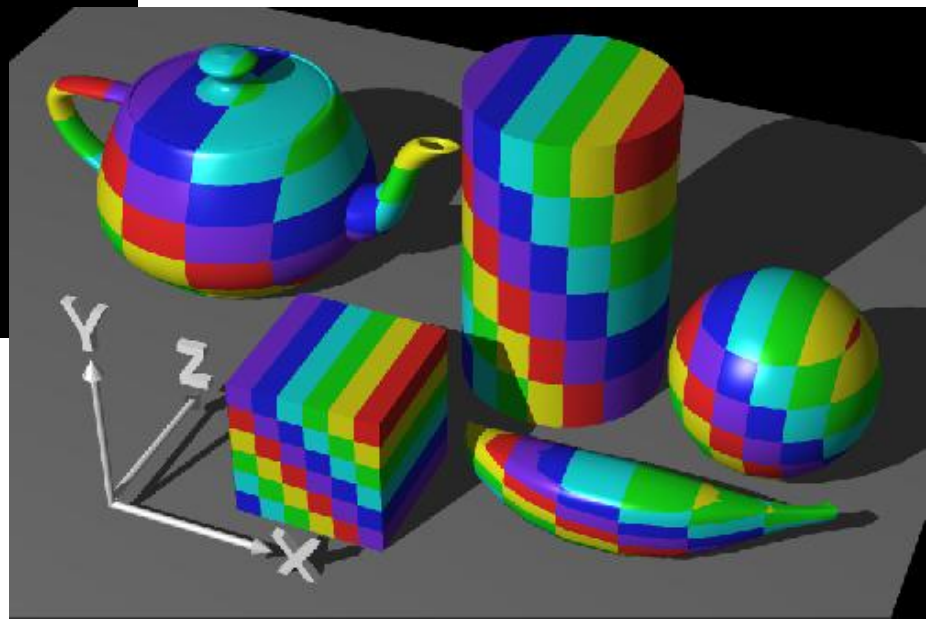
$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$



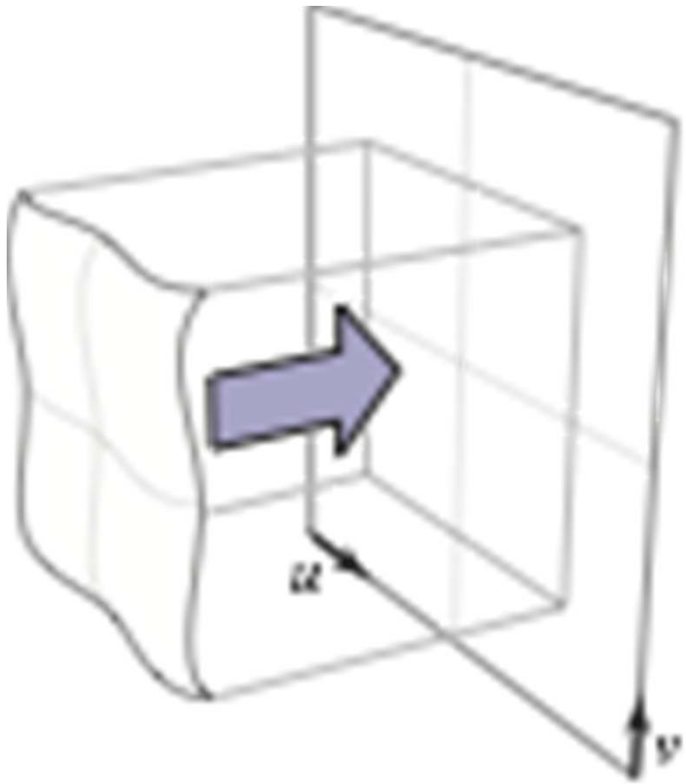
xyz in object space *xyz in camera space*

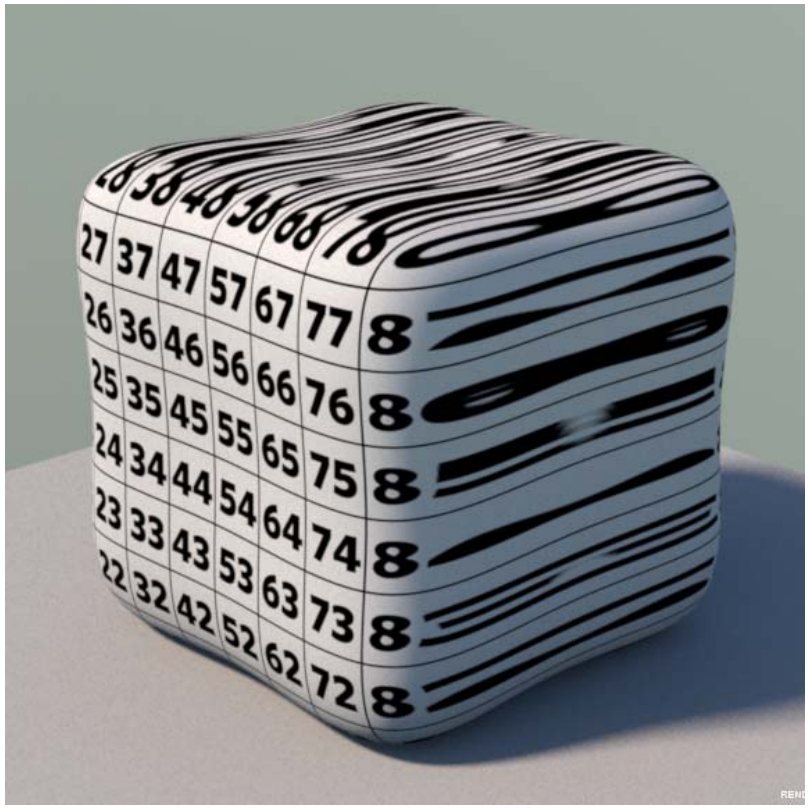
Planar mapping





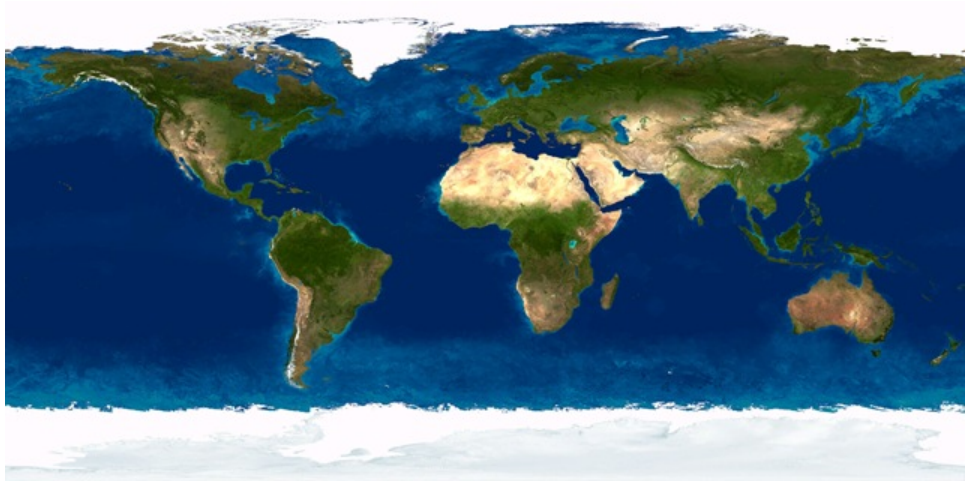
Planar mapping



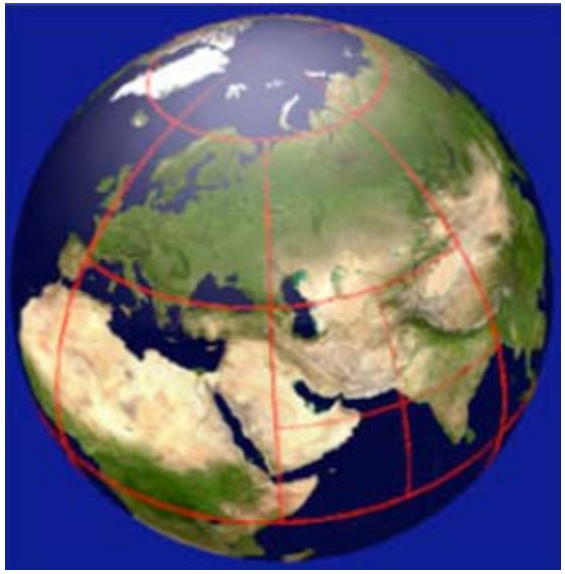


Spherical mapping

- Use spherical coordinates
- “Shrink-wrap” sphere to object

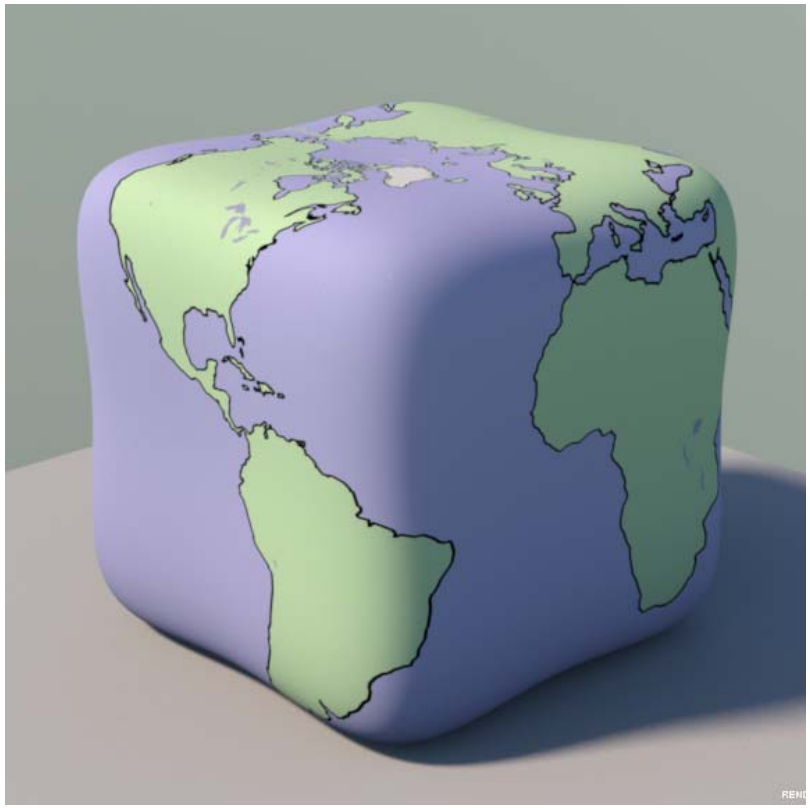


Texture map

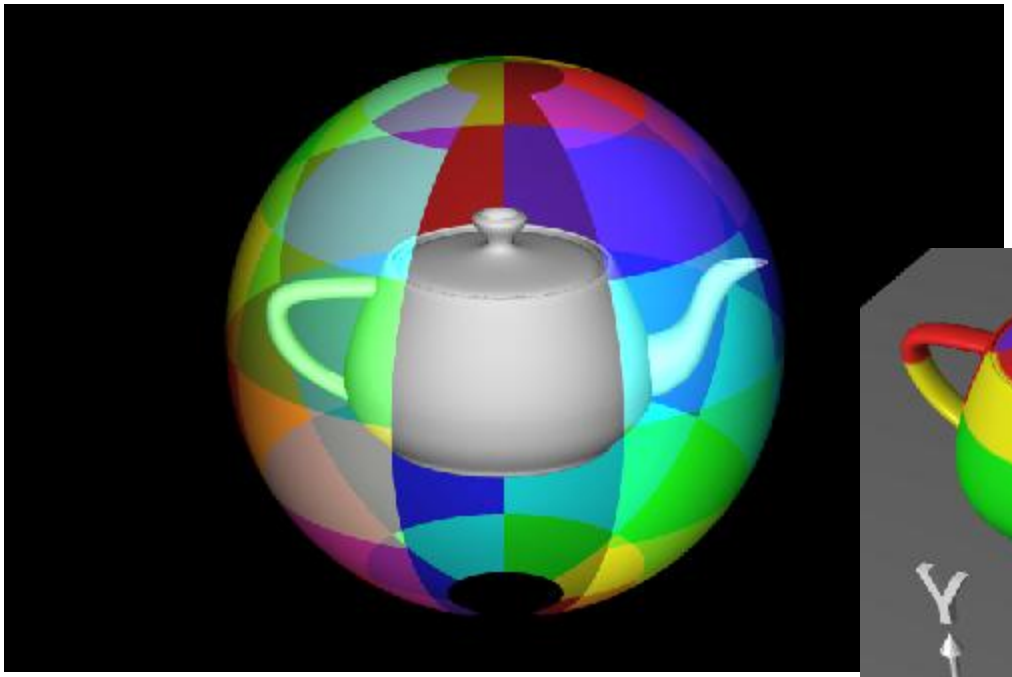


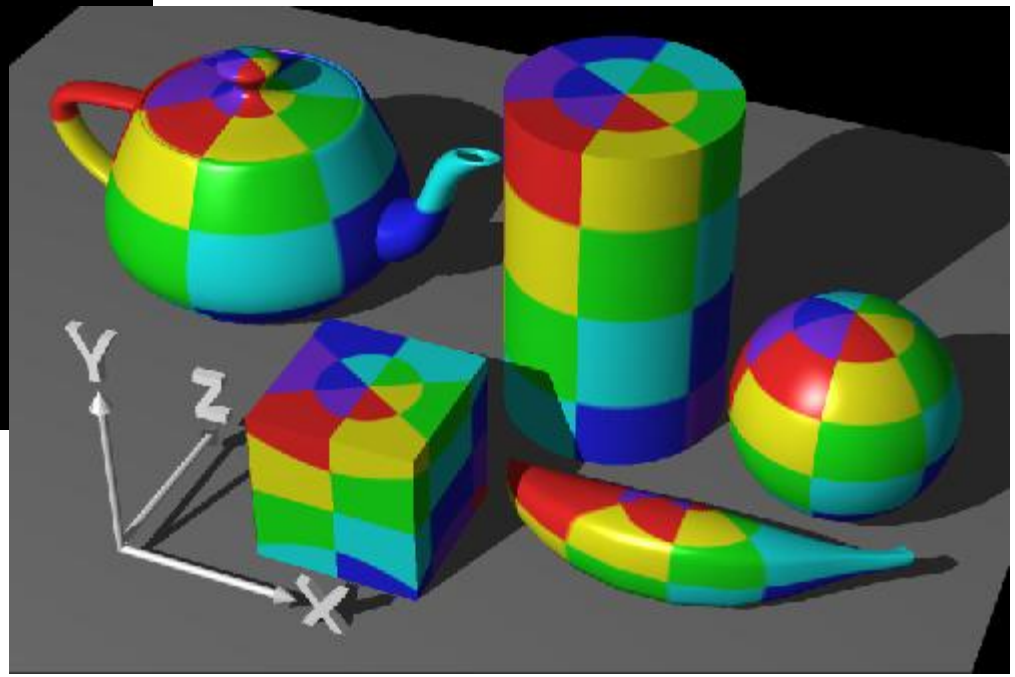
Mapping result

Spherical mapping



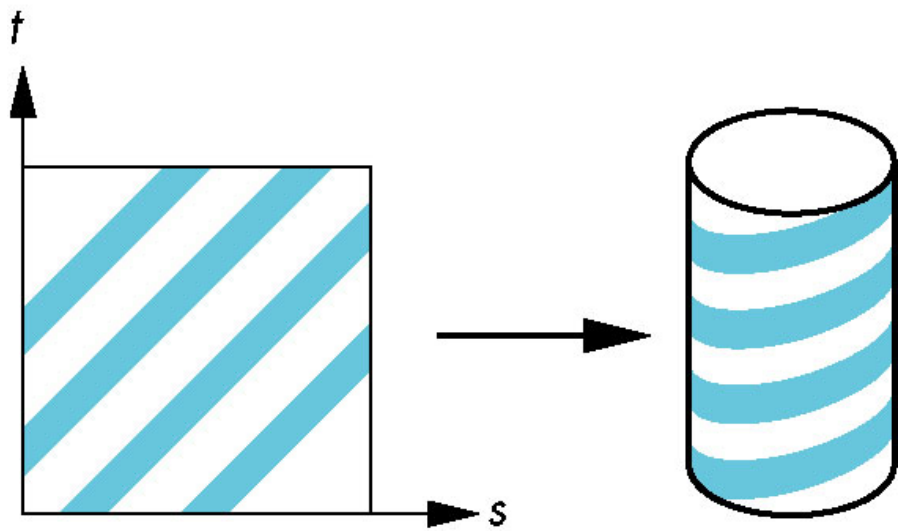
Spherical mapping

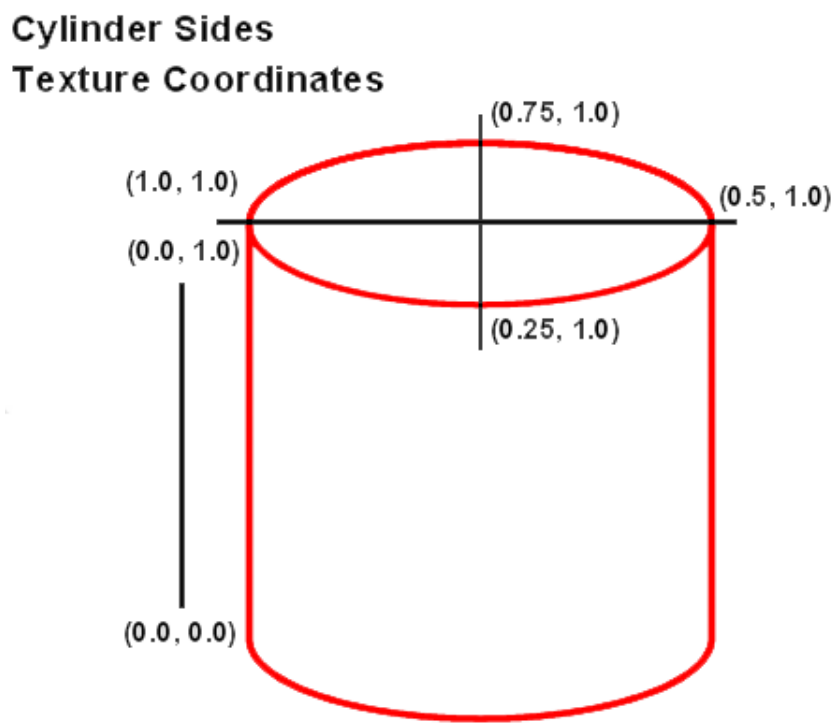




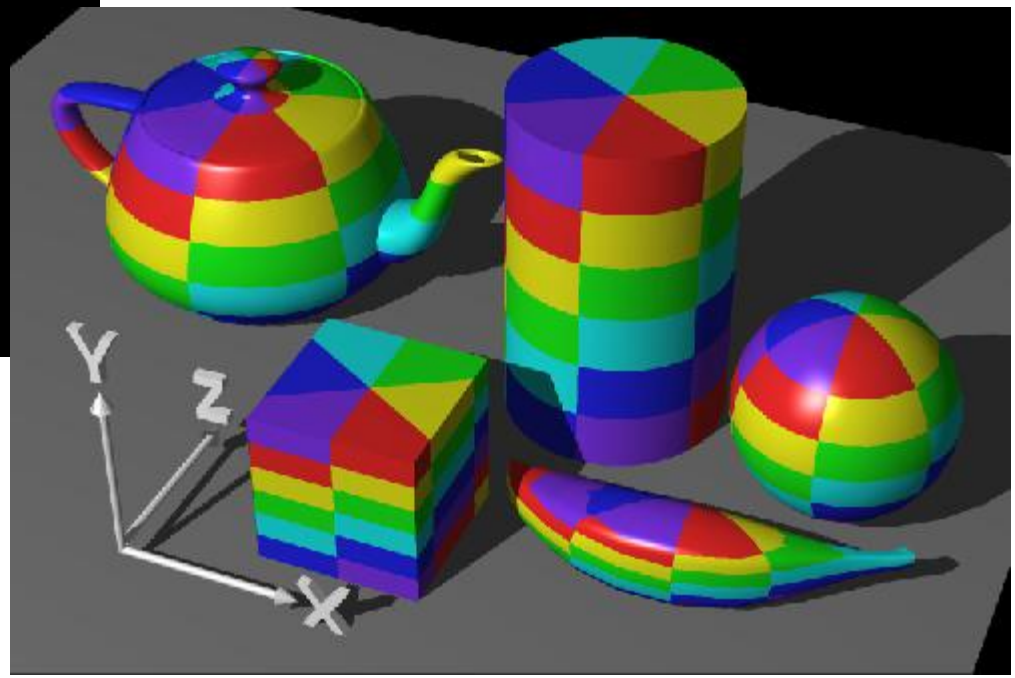
Cylindrical mapping

- Similar to spherical mapping, but with cylindrical coordinates



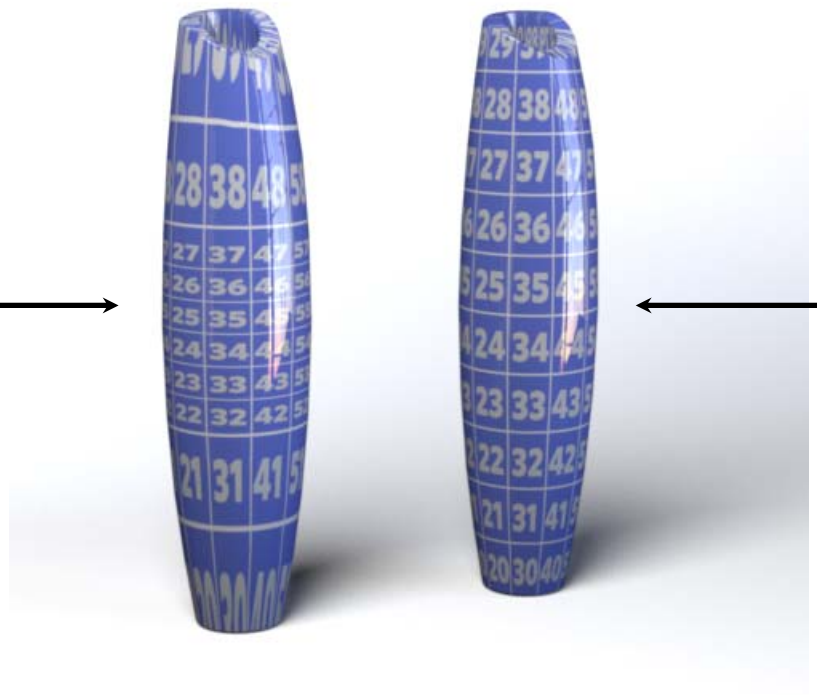


Cylindrical mapping



Spherical and cylindrical mapping

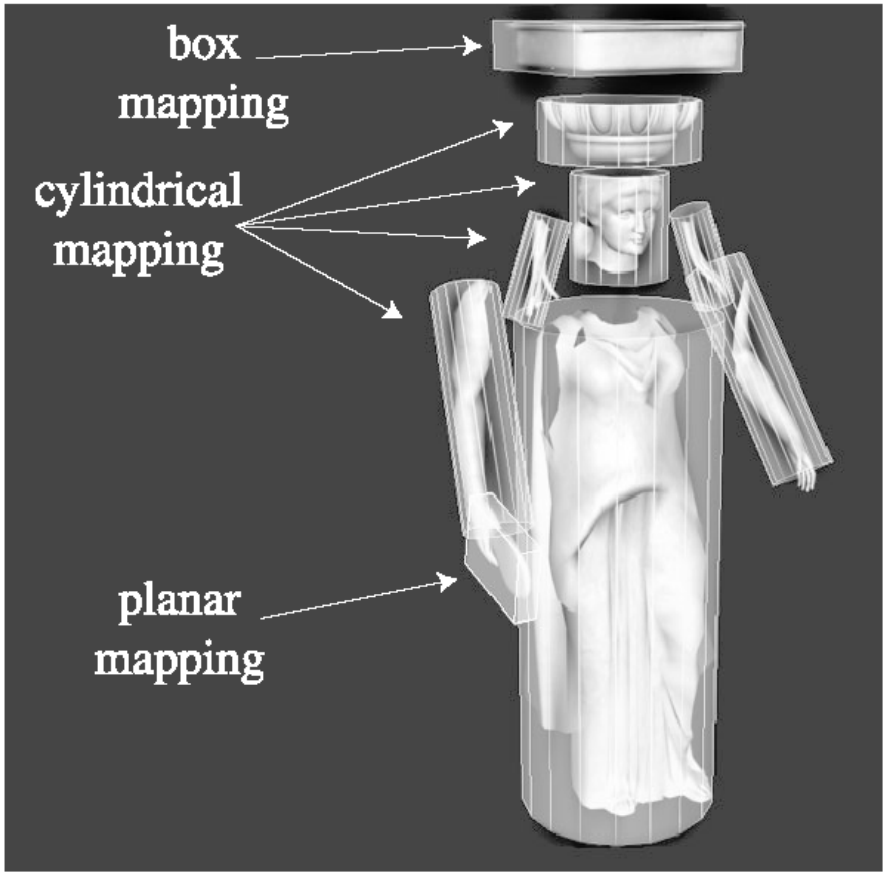
spherical



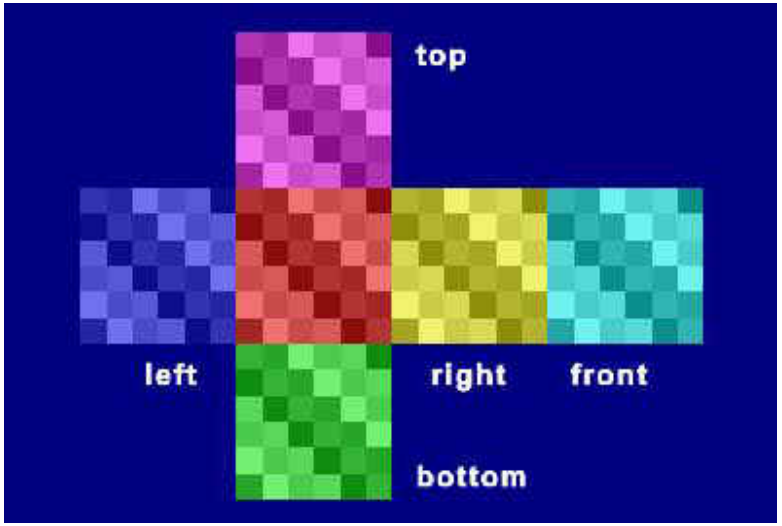
cylindrical

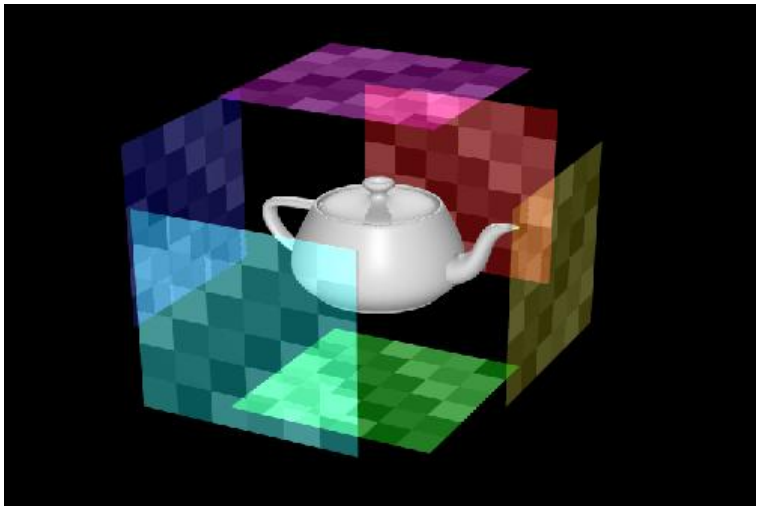
Multiple mappings

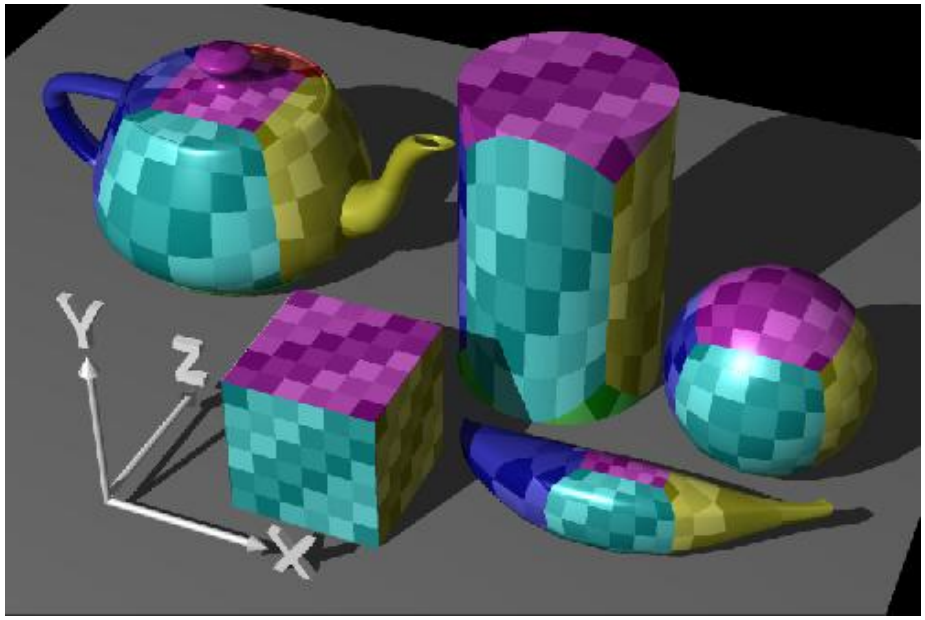
- Complex objects



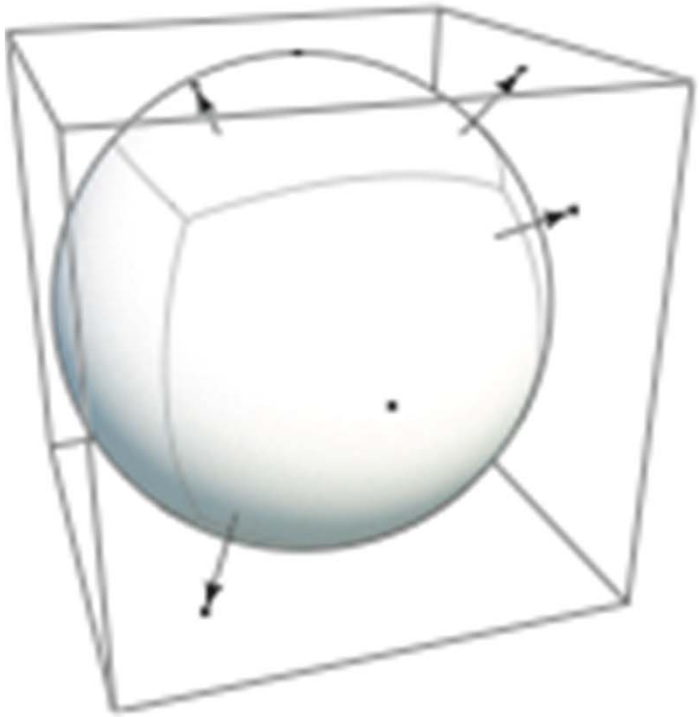
Cube mapping

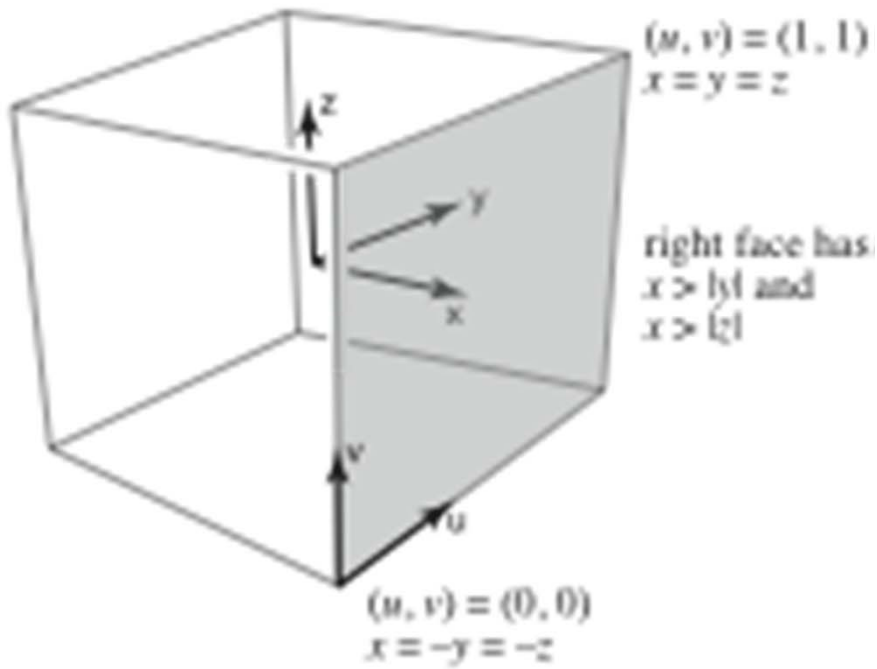




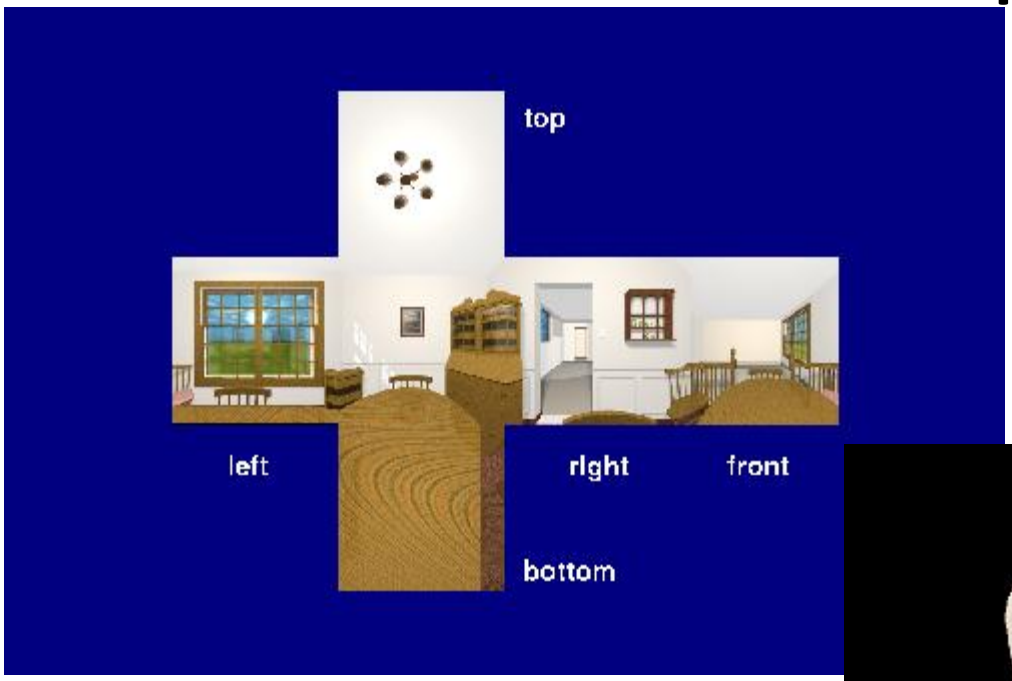


Cube mapping



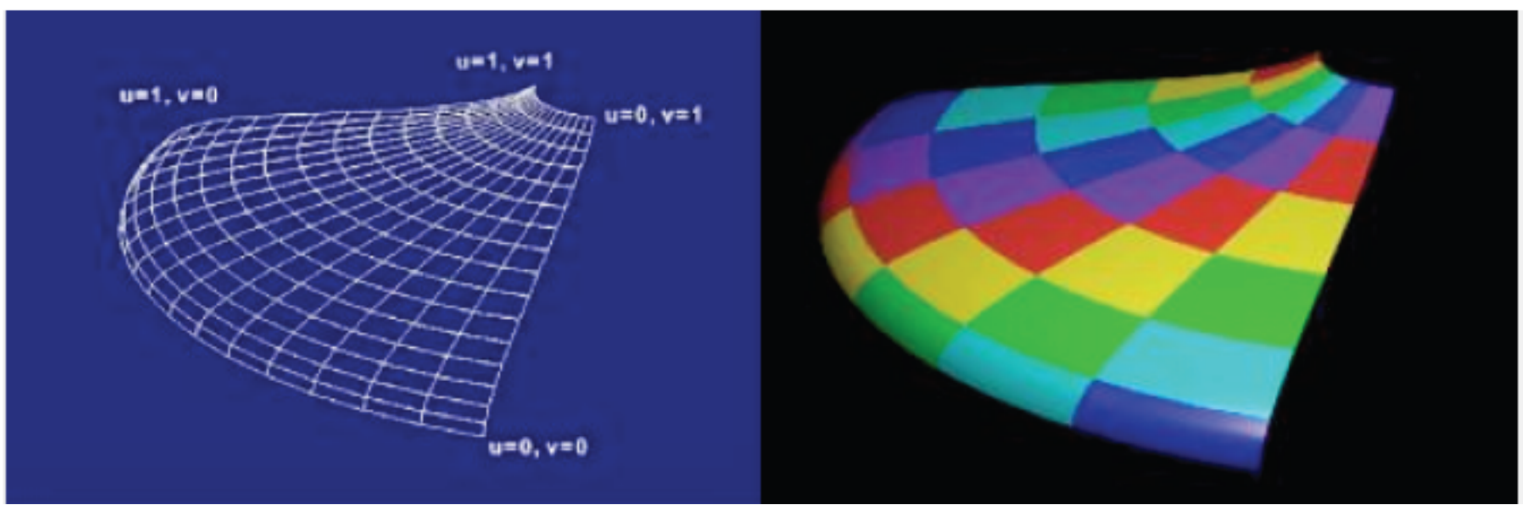


Cube mapping



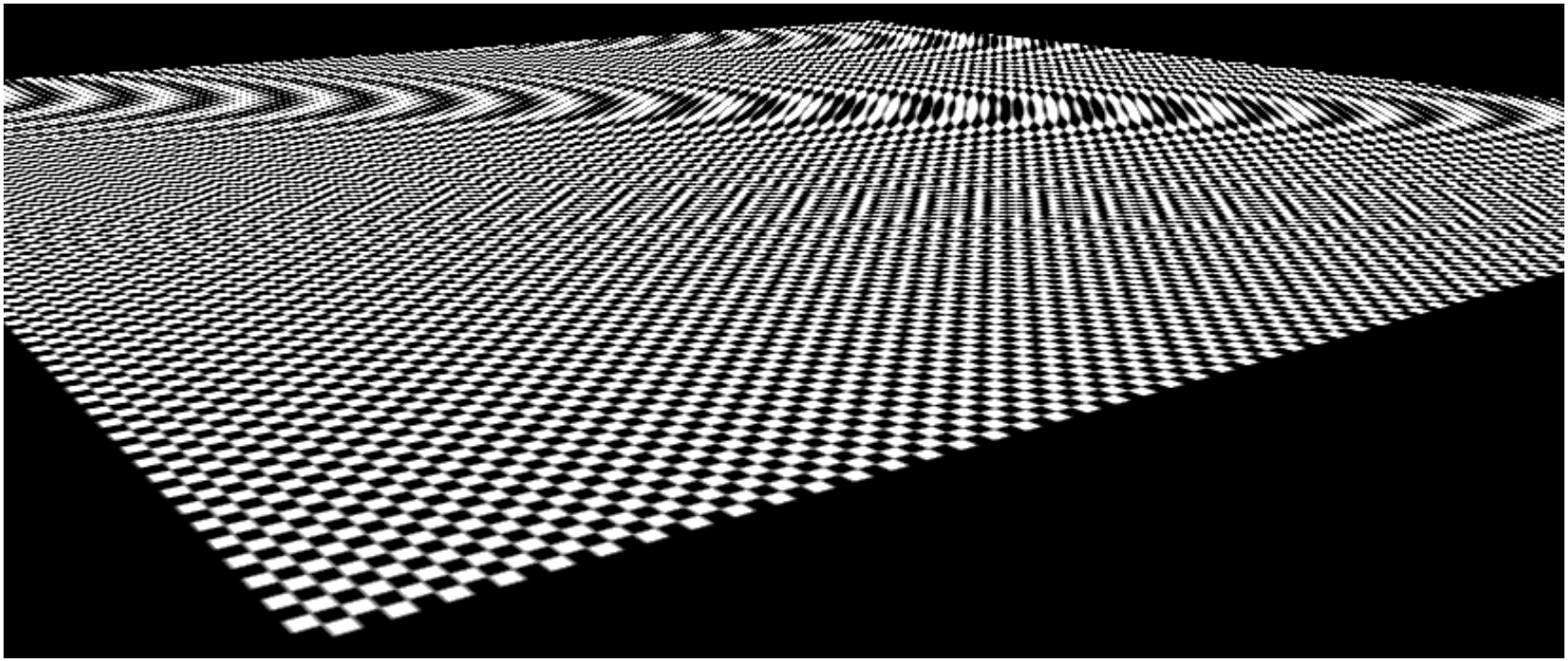
Parametric mapping

- Surface given by parametric functions
  $$x = f(u,v) \quad y = f(u,v) \quad z = f(u,v)$$
- Very common in CAD
- Clamp (*u*,*v*) parameters to [0..1] and use as texture coordinates (*s*,*t*)



Aliasing

- What could cause this aliasing effect?



Aliasing

Sufficiently sampled, no aliasing

Insufficiently sampled, aliasing

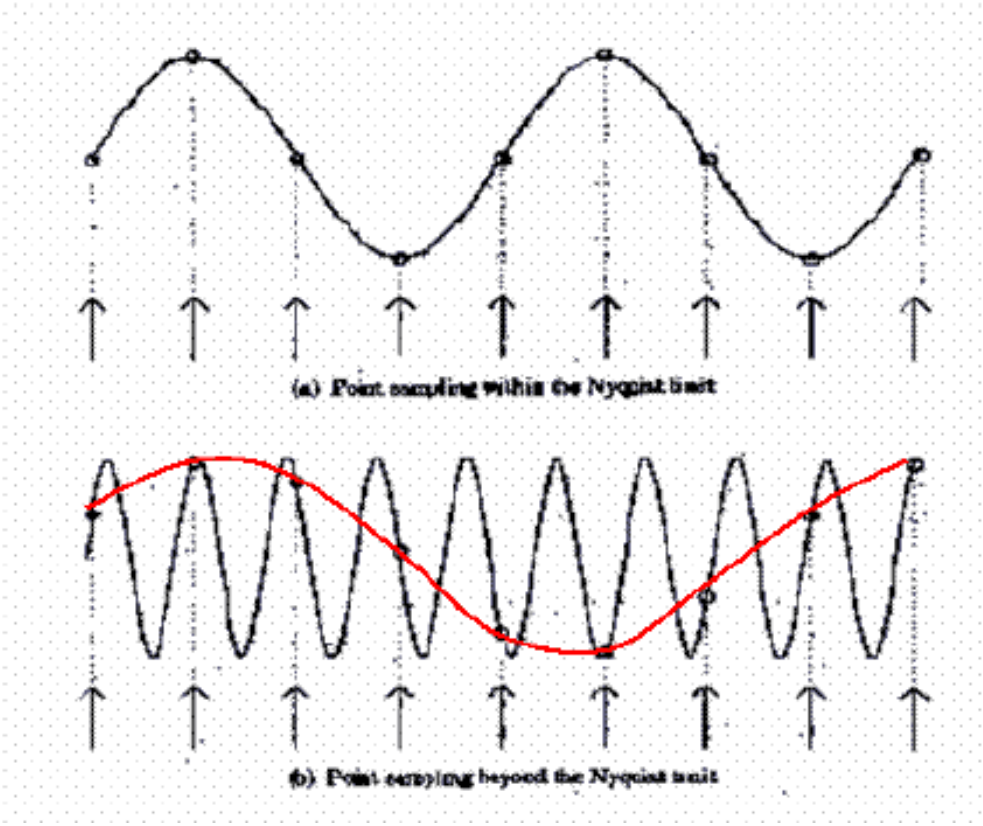
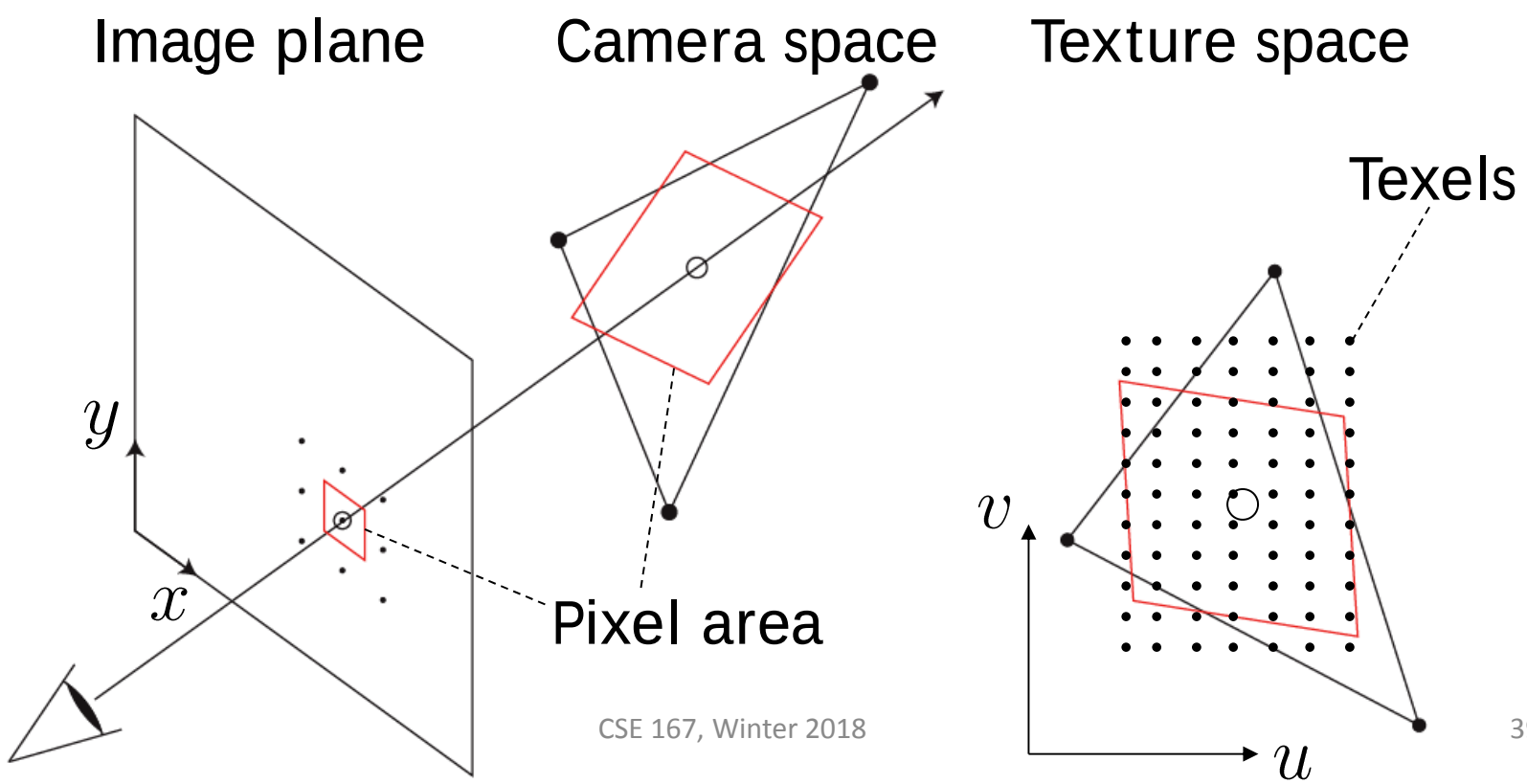


Image: Robert L. Cook

High frequencies in the input data can appear as lower frequencies in the sampled signal

Antialiasing: Intuition

- Pixel may cover large area on triangle in camera space
- Corresponds to many texels in texture space
- Need to compute average



Antialiasing using MIP maps

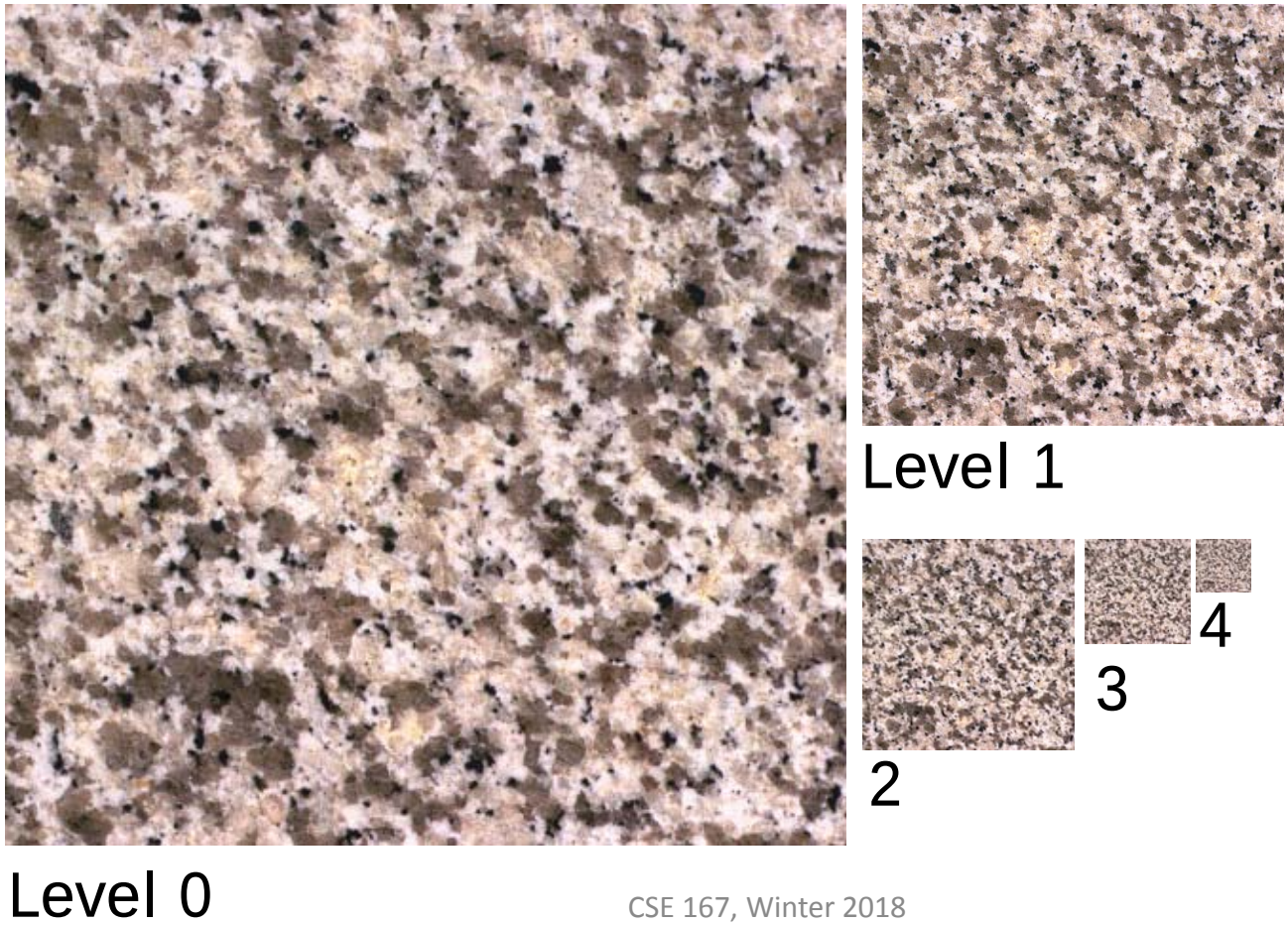
- Averaging over texels is expensive
  - Many texels as objects get smaller
  - Large memory access and computation cost
- Precompute filtered (averaged) textures
  - Texture at multiple resolutions
- Practical solution to aliasing problem
  - Fast and simple
  - Available in OpenGL, implemented in GPUs
  - Reasonable quality

MIP maps (mipmaps)

- MIP stands for *multum in parvo* (much in little) (Williams 1983)
- Before rendering
- Precompute and store down scaled versions of textures
  - Reduce resolution by factors of two successively
  - Use high quality filtering (averaging) scheme
- Increases memory cost by 1/3
  - 1/3 = ¼+1/16+1/64+…
- Width and height of texture should be powers of two (non-power of two supported since OpenGL 2.0)

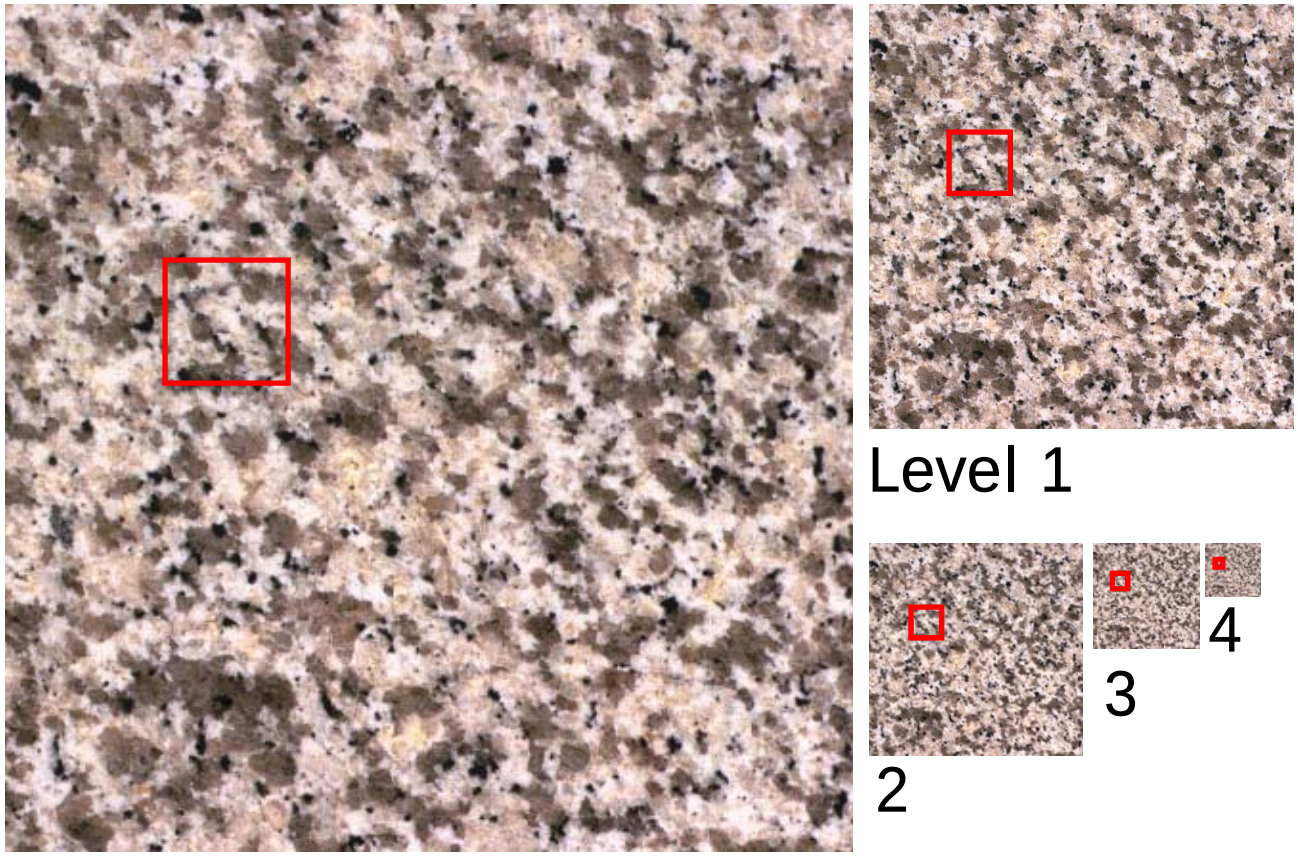
Mipmaps

- Example: resolutions 512x512, 256x256, 128x128, 64x64, 32x32 pixels

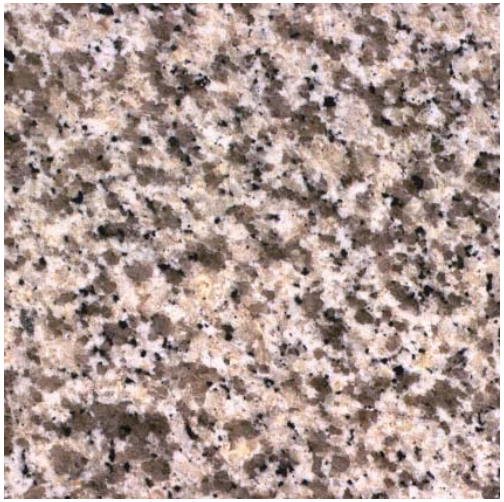


Mipmaps

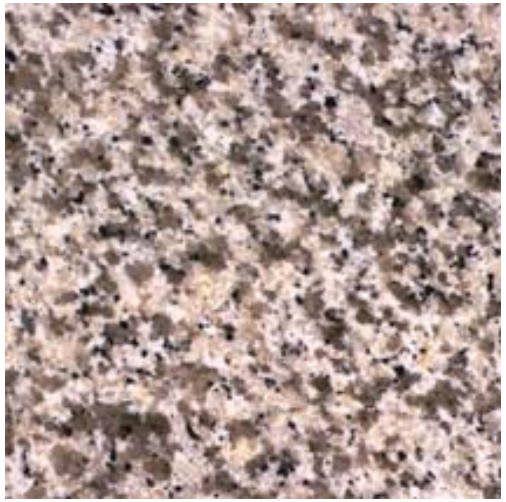
- One texel in level 4 is the average of 44=256 texels in level 0



Mipmaps



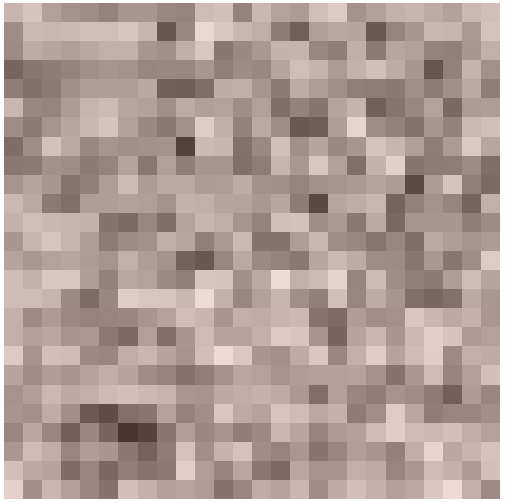
Level 0



Level 1

Level 2

Level 3

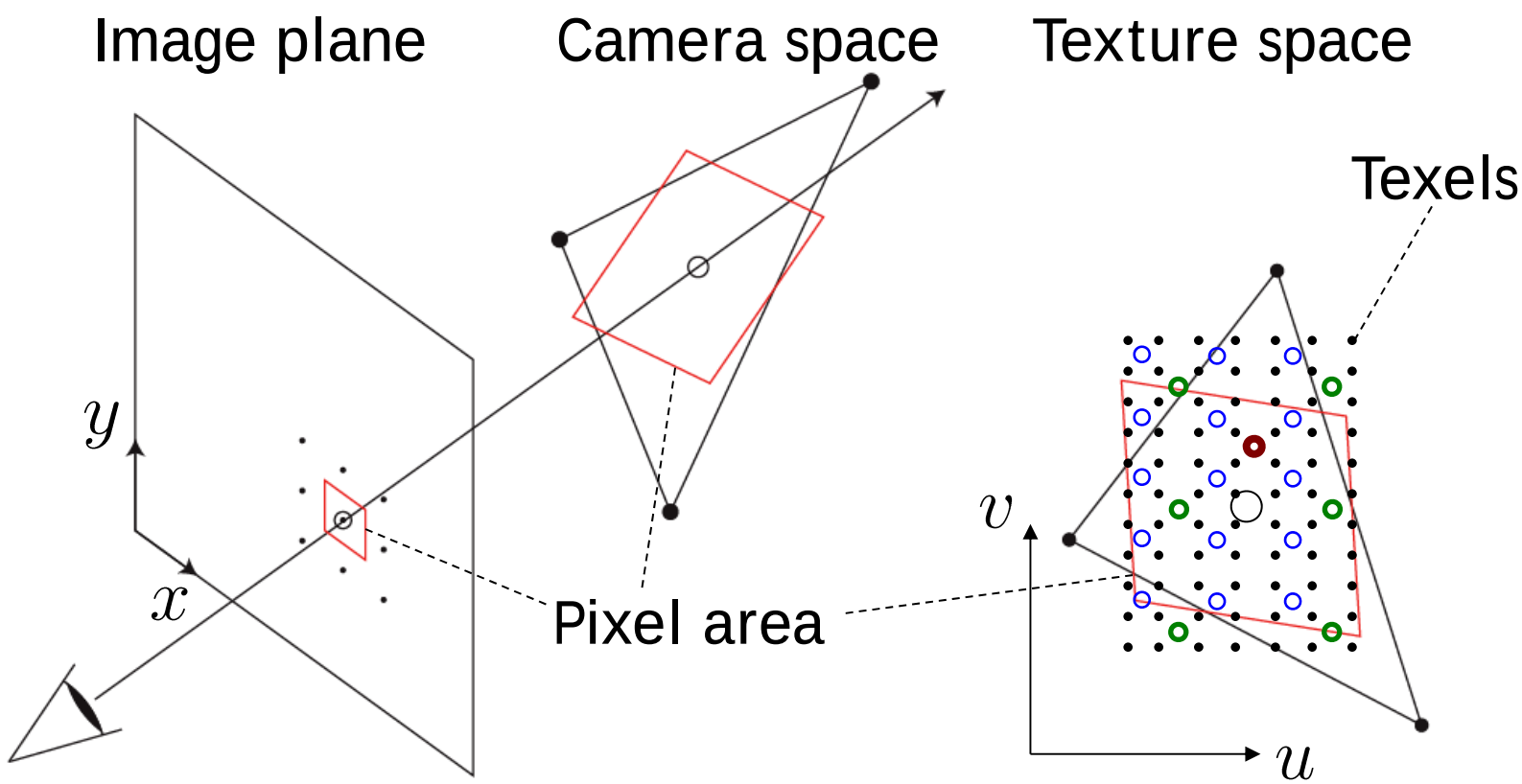


Level 4

Rendering with mipmaps

- "Mipmapping"
- Interpolate texture coordinates of each pixel as without mipmapping
- Compute approximate size of pixel in texture space
- Look up color in nearest mipmap
  - E.g., if pixel corresponds to 10x10 texels use mipmap level 3
  - Use interpolation as before

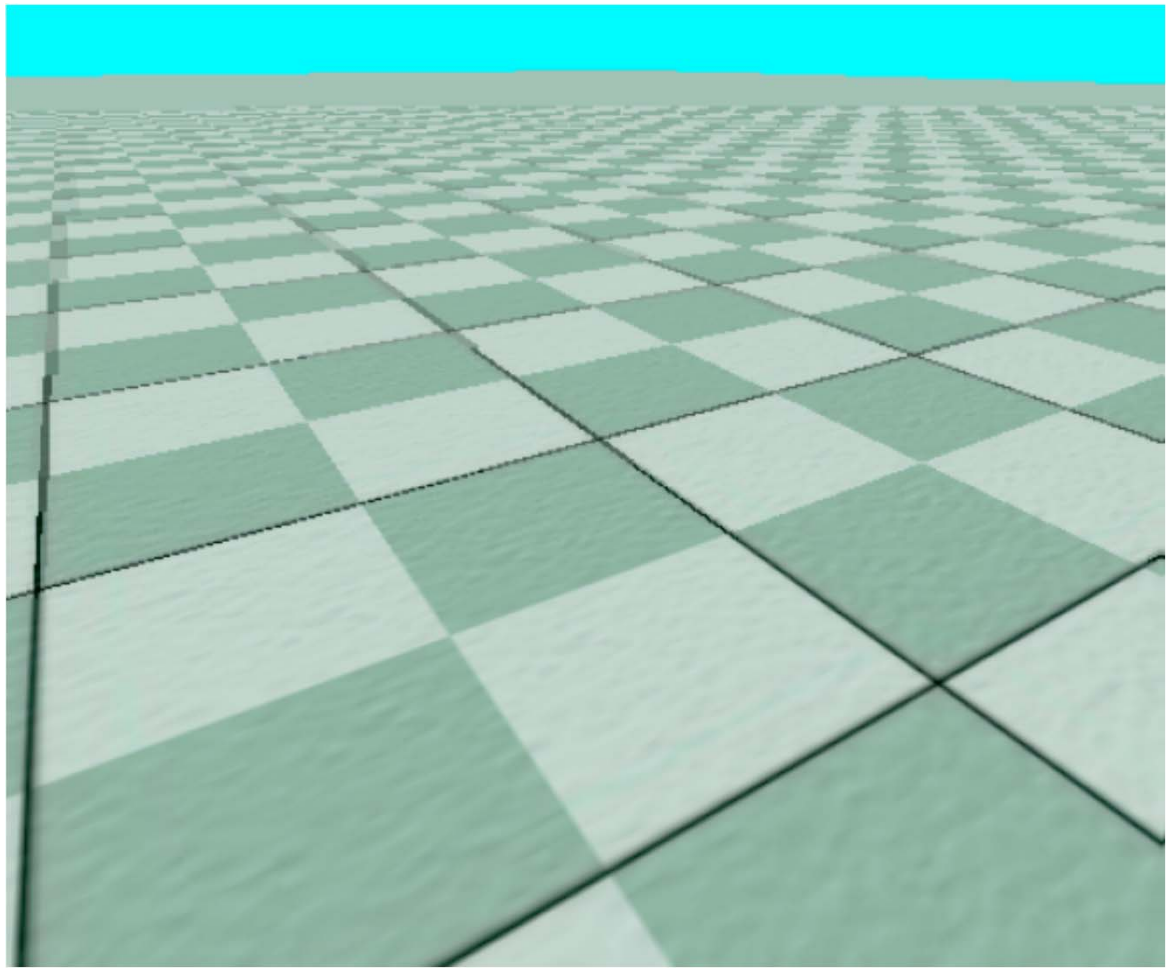
Mipmapping



- Mip-map level 0
- Mip-map level 1
- Mip-map level 2
- Mip-map level 3

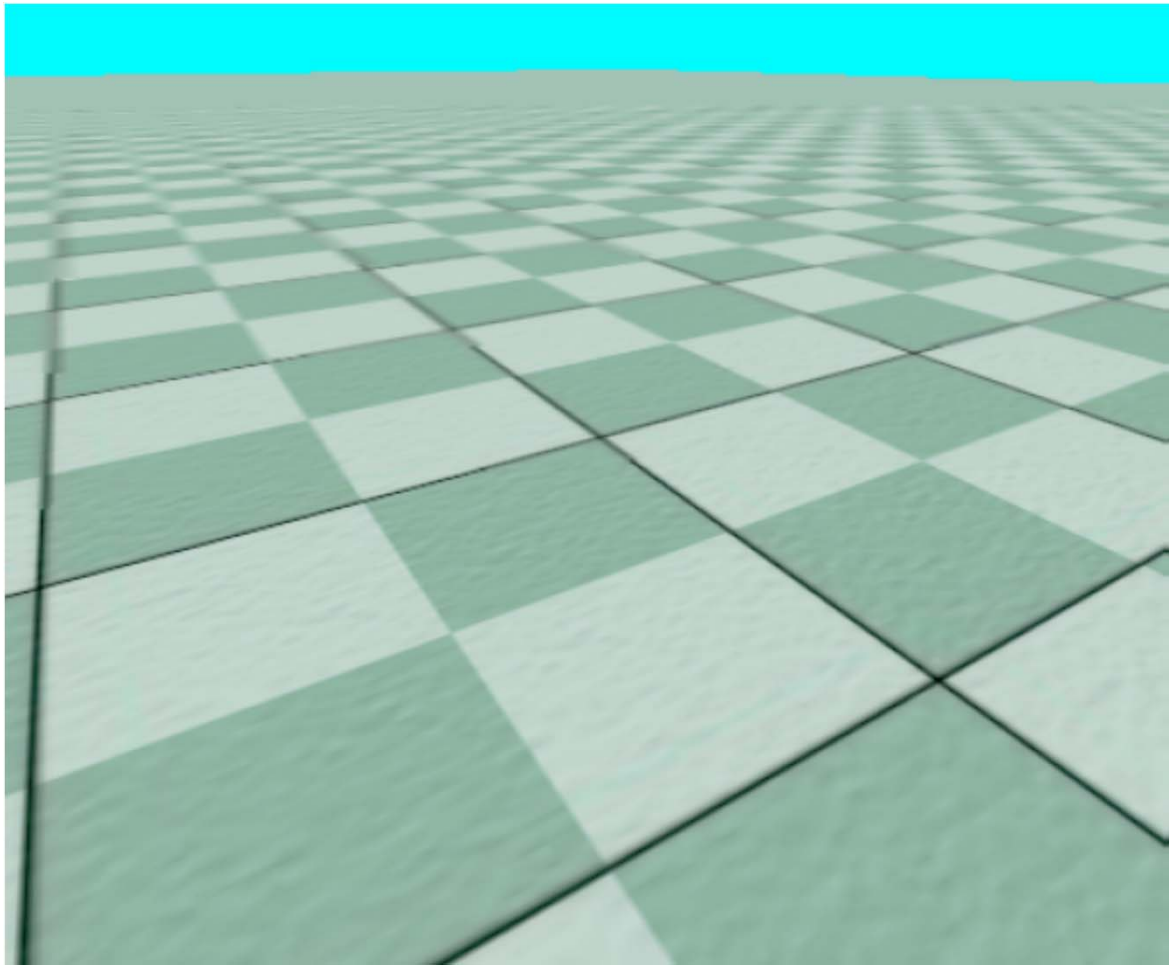
Nearest mipmap, nearest neighbor interpolation

Visible transition between mipmap levels



Nearest mipmap, bilinear interpolation

Visible transition between mipmap levels



Trilinear mipmapping

- Use two nearest mipmap levels
  - E.g., if pixel corresponds to 10x10 texels, use mipmap levels 3 (8x8) and 4 (16x16)
- 2-Step approach:
  1. Perform bilinear interpolation in both mip-maps
  2. Linearly interpolate between the results
- Requires access to 8 texels for each pixel
- Supported by hardware without performance penalty

Anisotropic filtering

- Method of enhancing the image quality of textures on surfaces that are at oblique viewing angles
- Different degrees or ratios of anisotropic filtering can be applied
- The degree refers to the maximum ratio of anisotropy supported by the filtering process. For example, 4:1 anisotropic filtering supports pre-sampled textures up to four times wider than tall

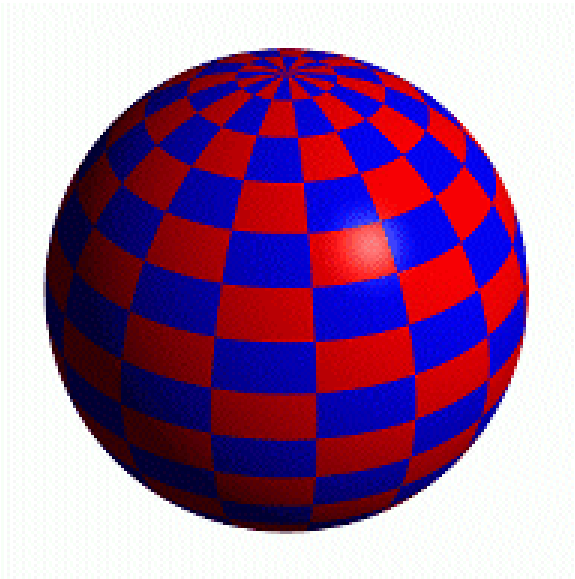
Mipmaps in OpenGL

- Mipmapping tutorial with source code

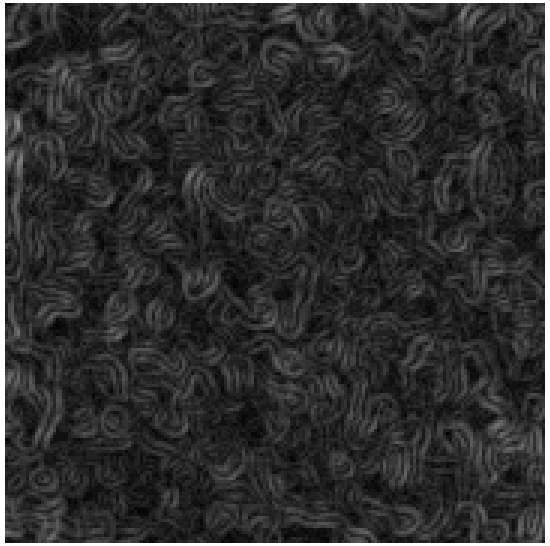
  http://www.videotutorialsrock.com/opengl_tutorial/mipmapping/text.php

Bump mapping

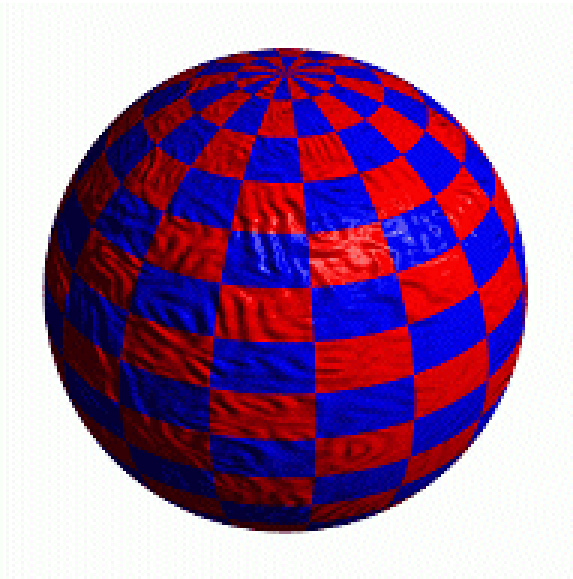
- Texture = change in surface normal



Sphere w/ diffuse texture

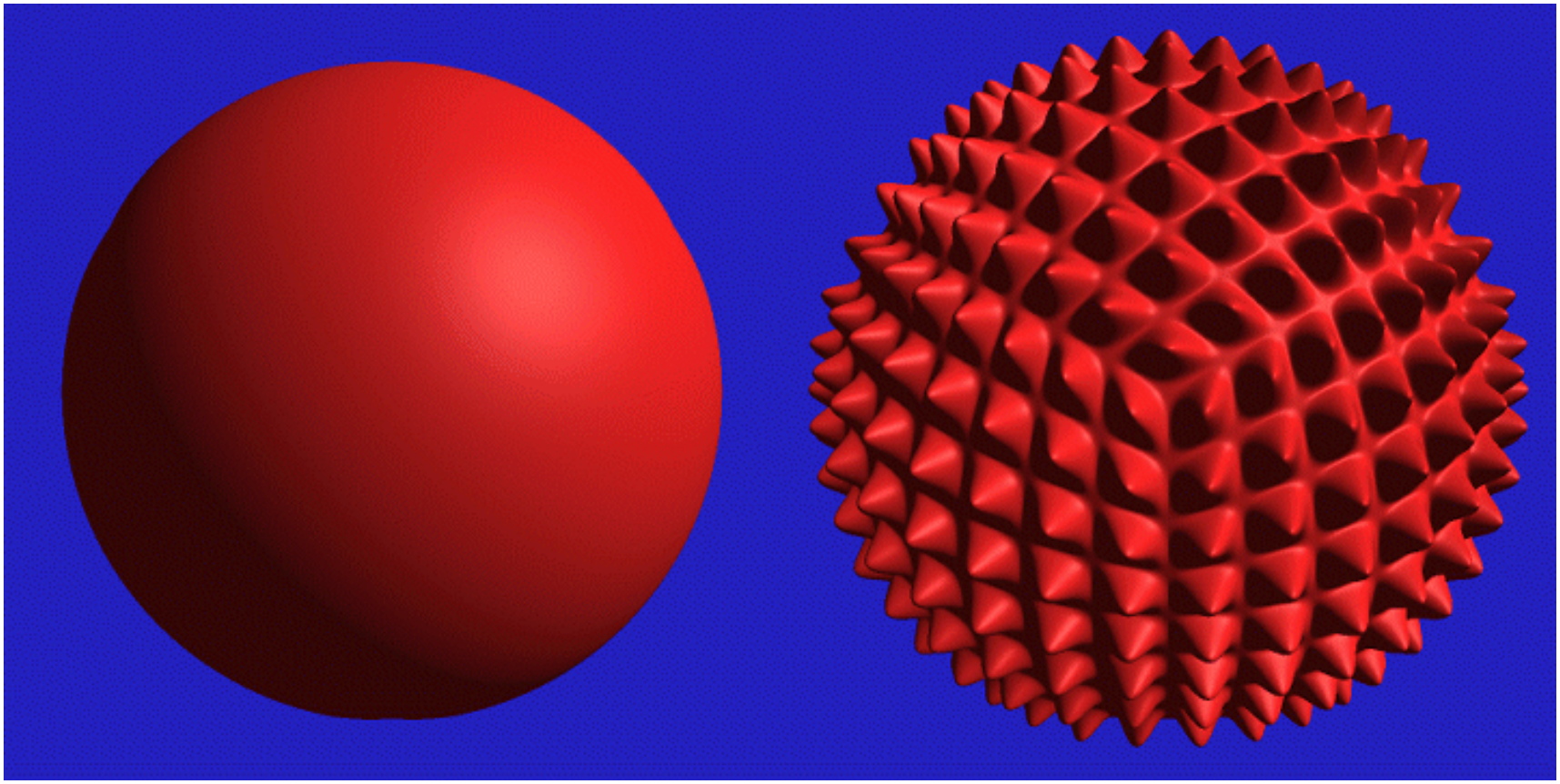


Swirly bump map

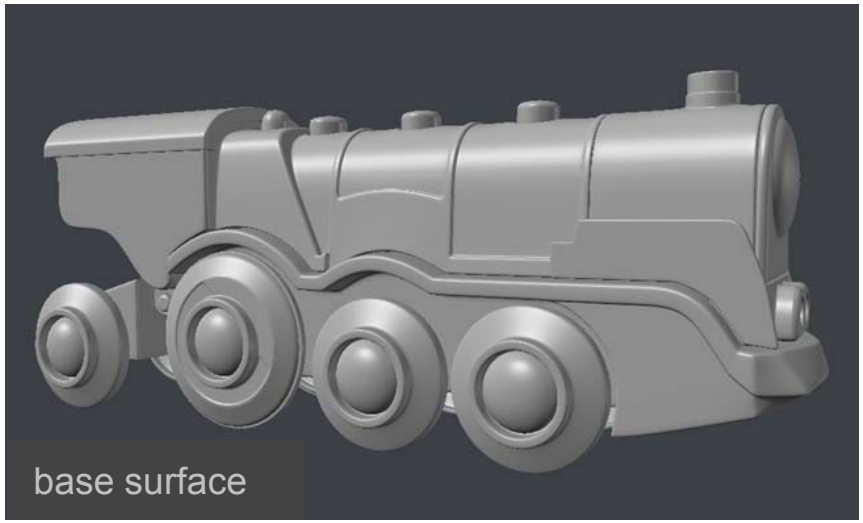


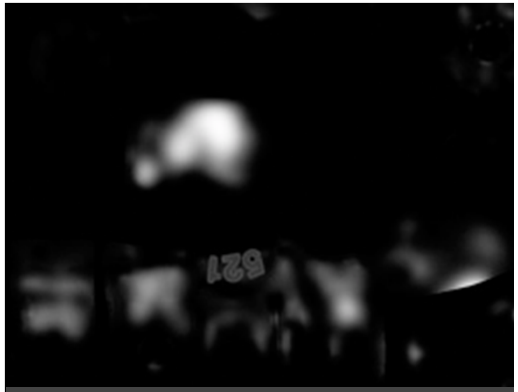
Sphere w/ diffuse texture and swirly bump map

Displacement mapping

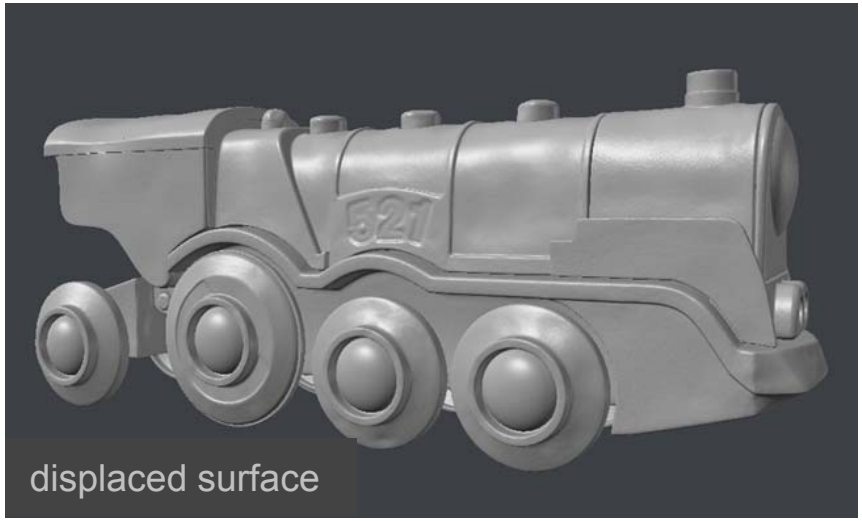


Displacement map





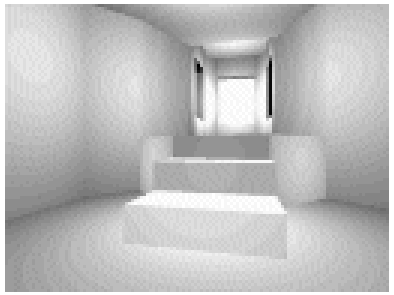
hand-painted displacement map (detail)



Paweł Filip
tolas.wordpress.com

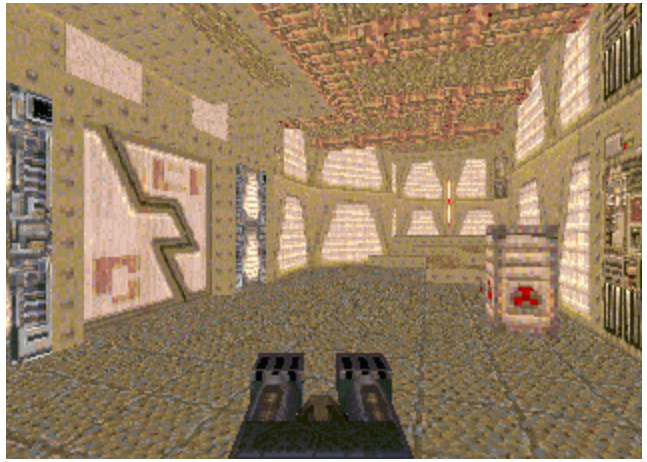
Illumination maps

- Quake introduced *illumination maps* or *light maps* to capture lighting effects in video games



Light map

Texture map
+ light map:



Environment maps

Images from *Illumination and Reflection Maps:*
Simulated Objects in Simulated and Real Environments
Gene Miller and C. Robert Hoffman
SIGGRAPH 1984 "Advanced Computer Graphics Animation" Course Notes

Environment map

Solid textures

- Texture values indexed by 3D location (x,y,z)
  - Expensive storage, or
  - Compute on the fly, e.g. Perlin noise