CSE 167:
Introduction to Computer Graphics
Lecture #13: Advanced Texture Mapping

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2019

Announcements

- Homework 4 grading tomorrow

Lecture Overview

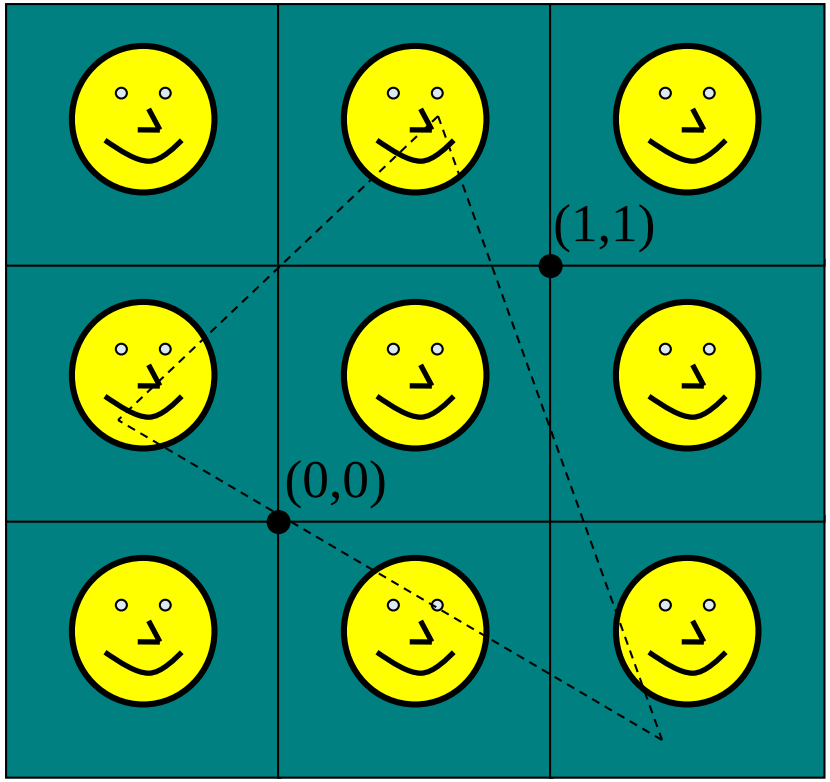
- Texture Mapping
  - Wrapping
  - Texture coordinates
  - Anti-aliasing

Wrap Modes

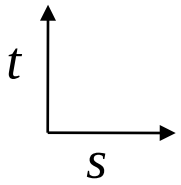
- Texture image extends from [0,0] to [1,1] in texture space
  - What if (s,t) texture coordinates are beyond that range?
- → Texture wrap modes

Repeat

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



Texture Space

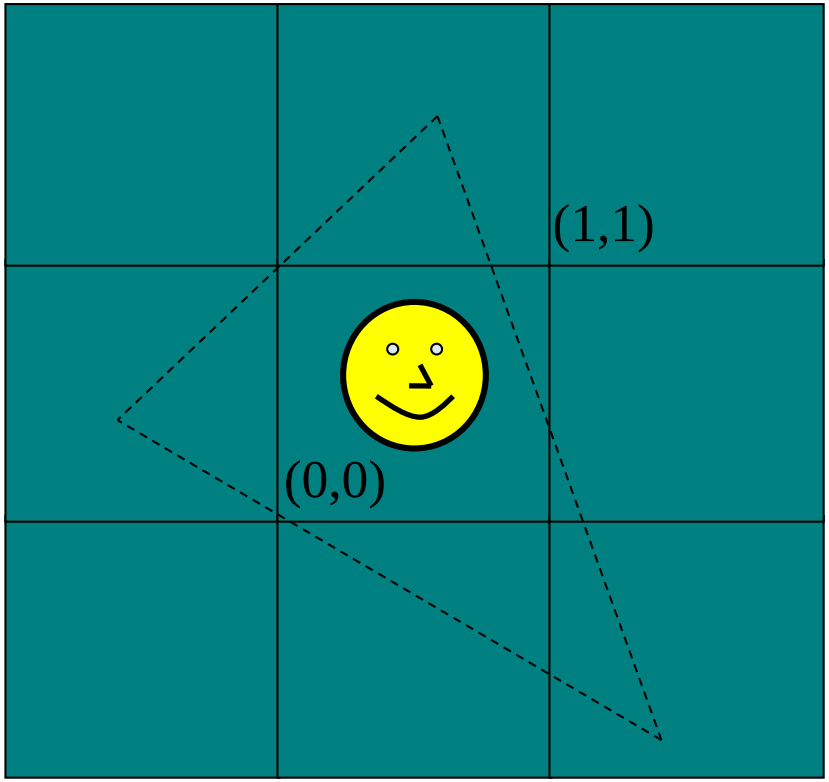


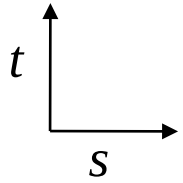
Seamless brick wall texture
(by Christopher Revoir)

UCSD

Clamp

- Use edge value everywhere outside data range [0..1]
- Or use specified border color outside of range [0..1]

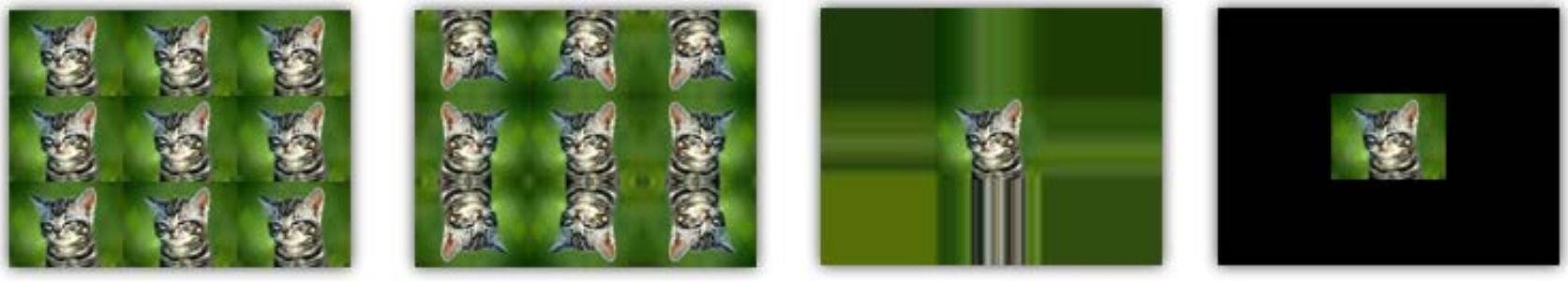




Texture Space

Wrap Modes in OpenGL

- Default:
  - glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
  - glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
- Options for wrap mode:
  - GL_REPEAT
  - GL_MIRRORED_REPEAT
  - GL_CLAMP_TO_EDGE: repeats last pixel in the texture
  - GL_CLAMP_TO_BORDER: requires border color to be set



Source: https://open.gl/textures

Lecture Overview

- Texture Mapping
  - Wrapping
  - Texture coordinates
  - Anti-aliasing

Texture Coordinates

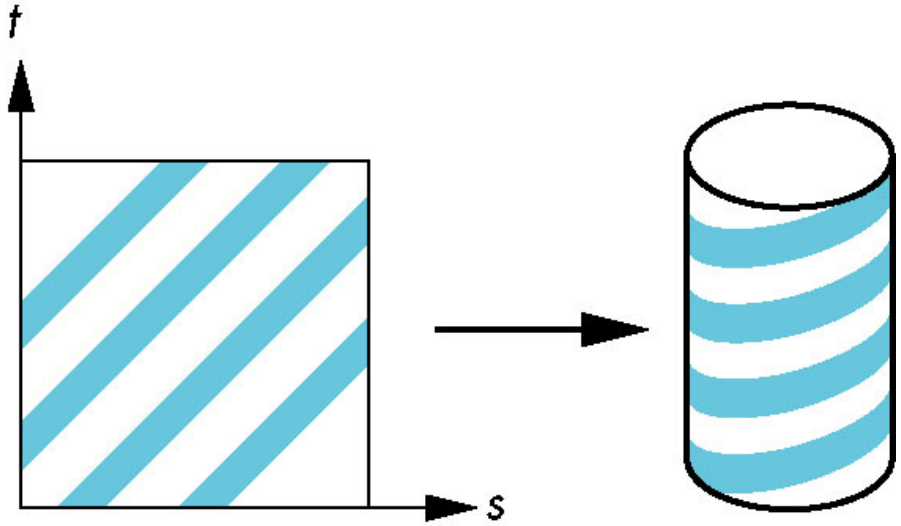
What if texture extends across multiple polygons?

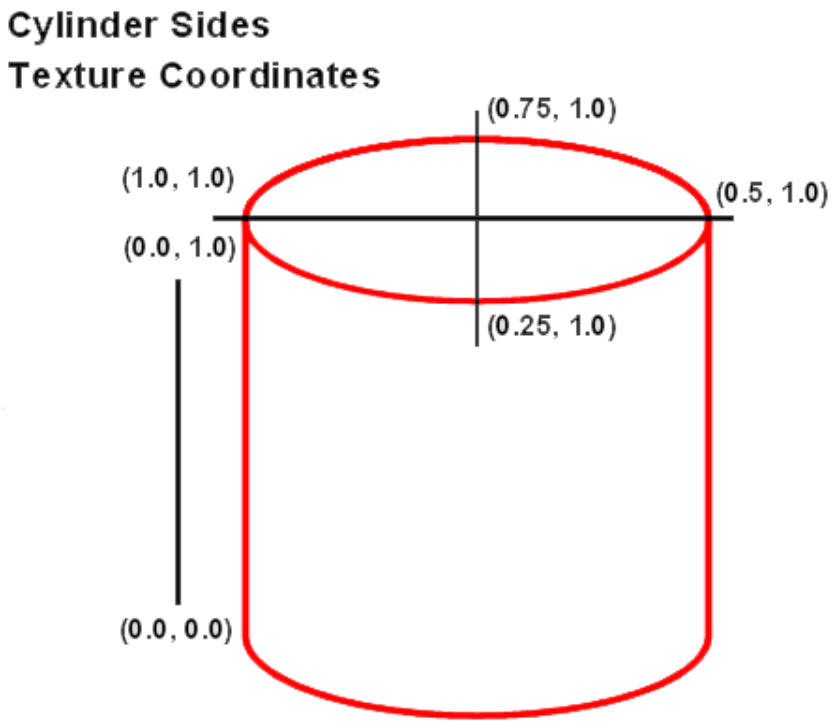
→ Surface parameterization

- Mapping between 3D positions on surface and 2D texture coordinates
  - Defined by texture coordinates of triangle vertices
- Options for mapping:
  - Cylindrical
  - Spherical
  - Orthographic
  - Parametric
  - Skin

Cylindrical Mapping

- Similar to spherical mapping, but with cylindrical coordinates

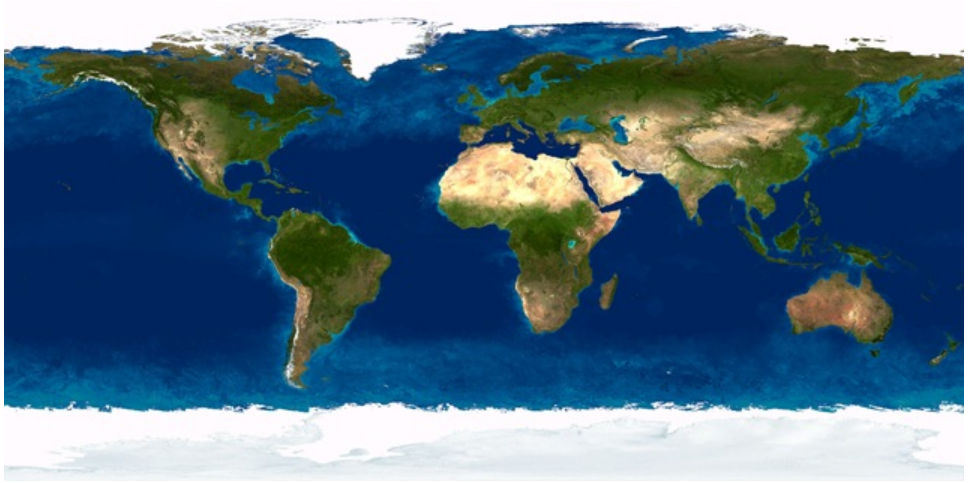




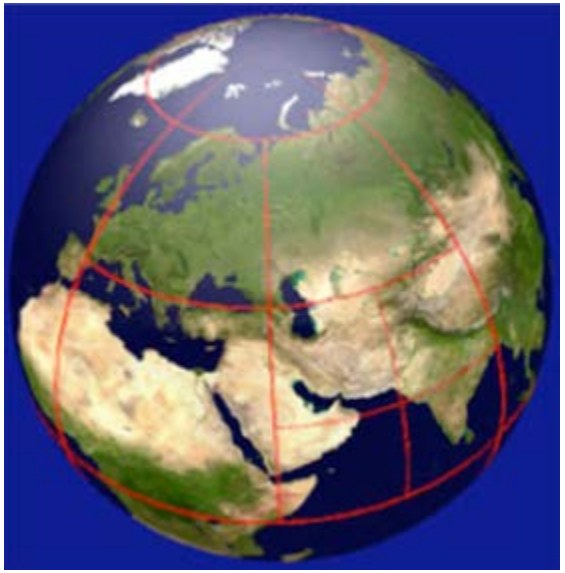
UCSD

Spherical Mapping

- Use spherical coordinates
- "Shrink-wrap" sphere to object



Texture map

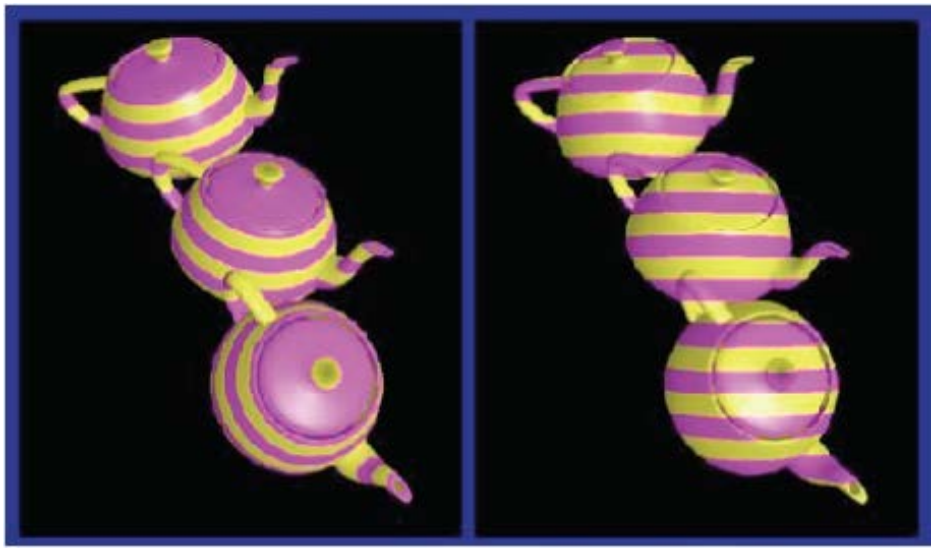


Mapping result

Orthographic Mapping

- Use linear transformation of object's xyz coordinates
- Example:

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



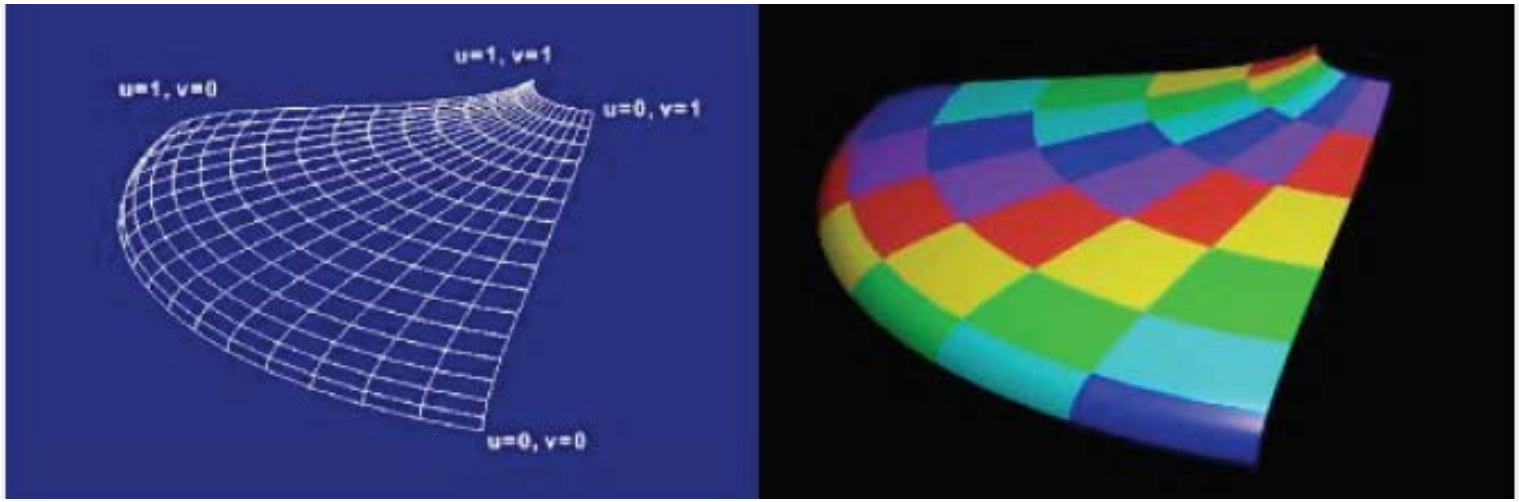
xyz in object space *xyz in camera space*

Parametric Mapping

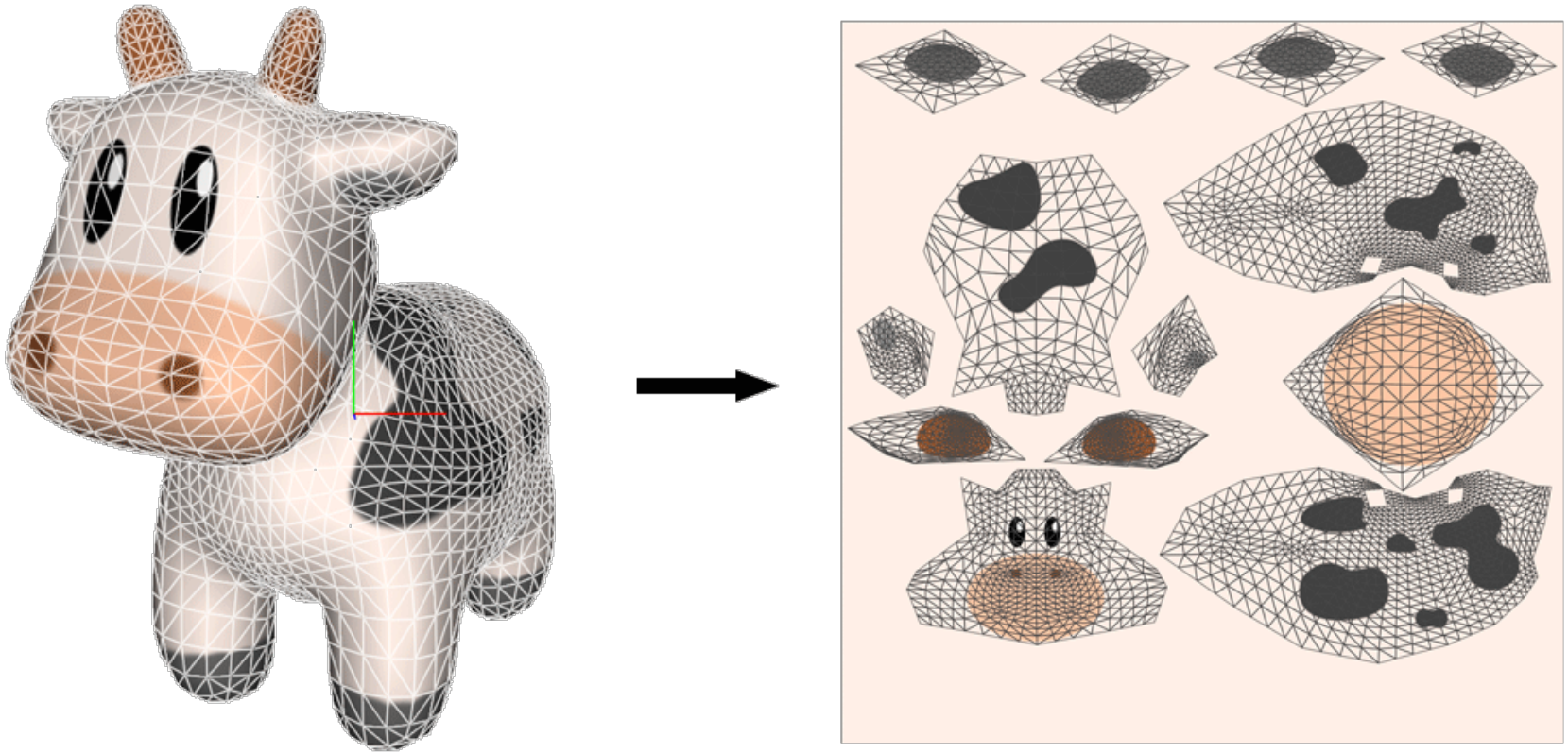
- Surface given by parametric functions

$$x = f(u,v) \quad y = f(u,v) \quad z = f(u,v)$$

- Very common in CAD
- Clamp (u,v) parameters to [0..1] and use as texture coordinates (s,t)



Skin Mapping

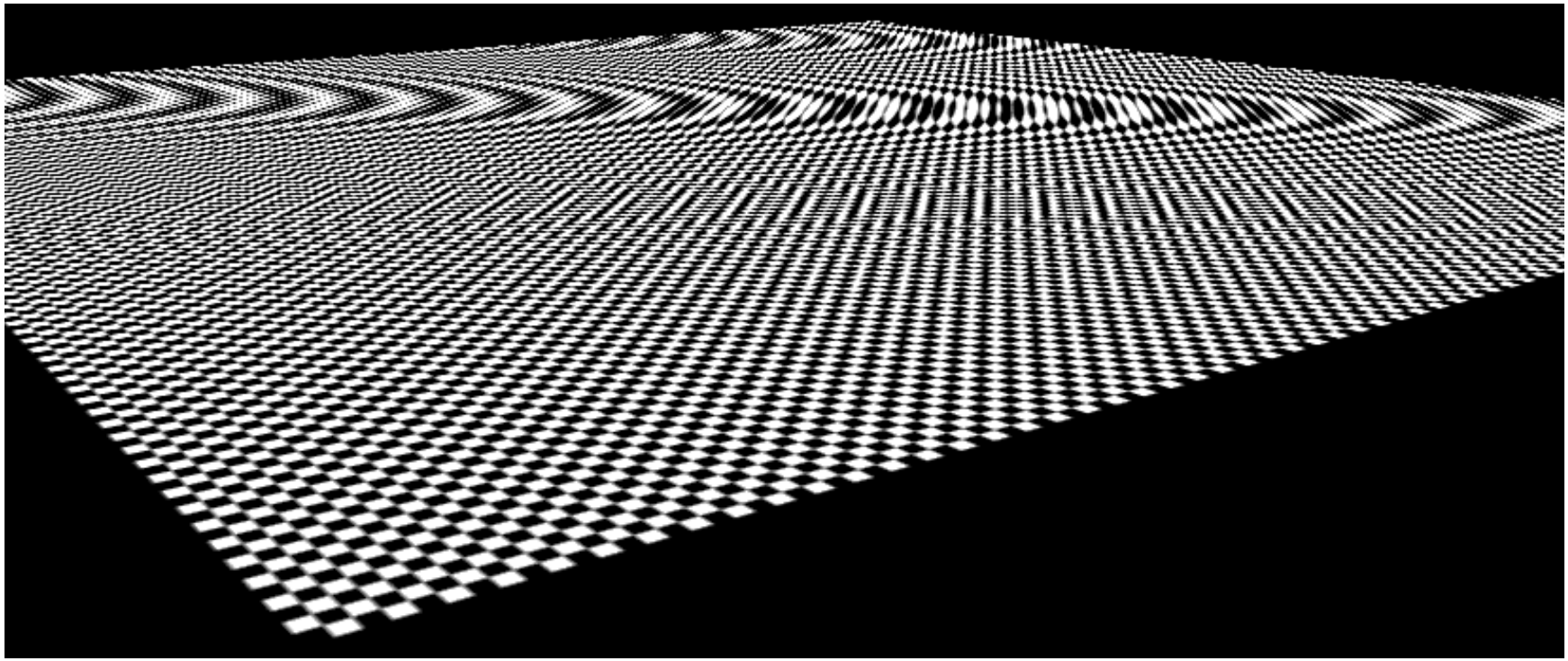


Lecture Overview

- Texture Mapping
  - Wrapping
  - Texture coordinates
  - Anti-aliasing

Aliasing

- What could cause this aliasing effect?



Aliasing

Sufficiently sampled, no aliasing

Insufficiently sampled, aliasing

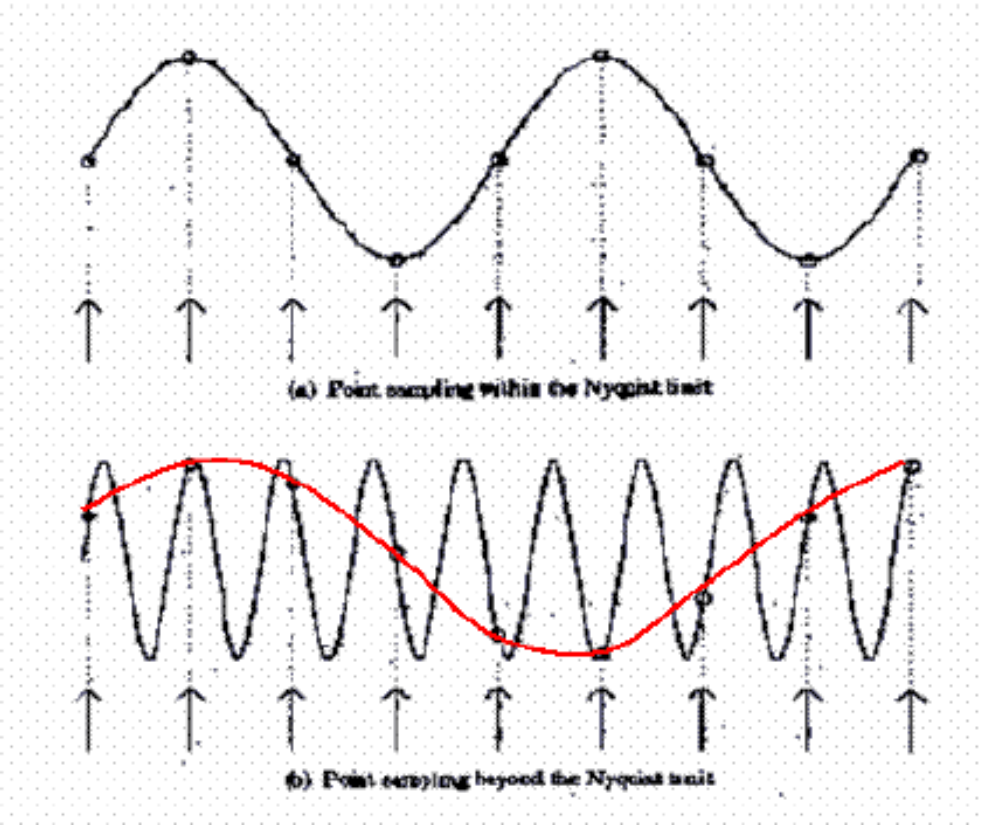
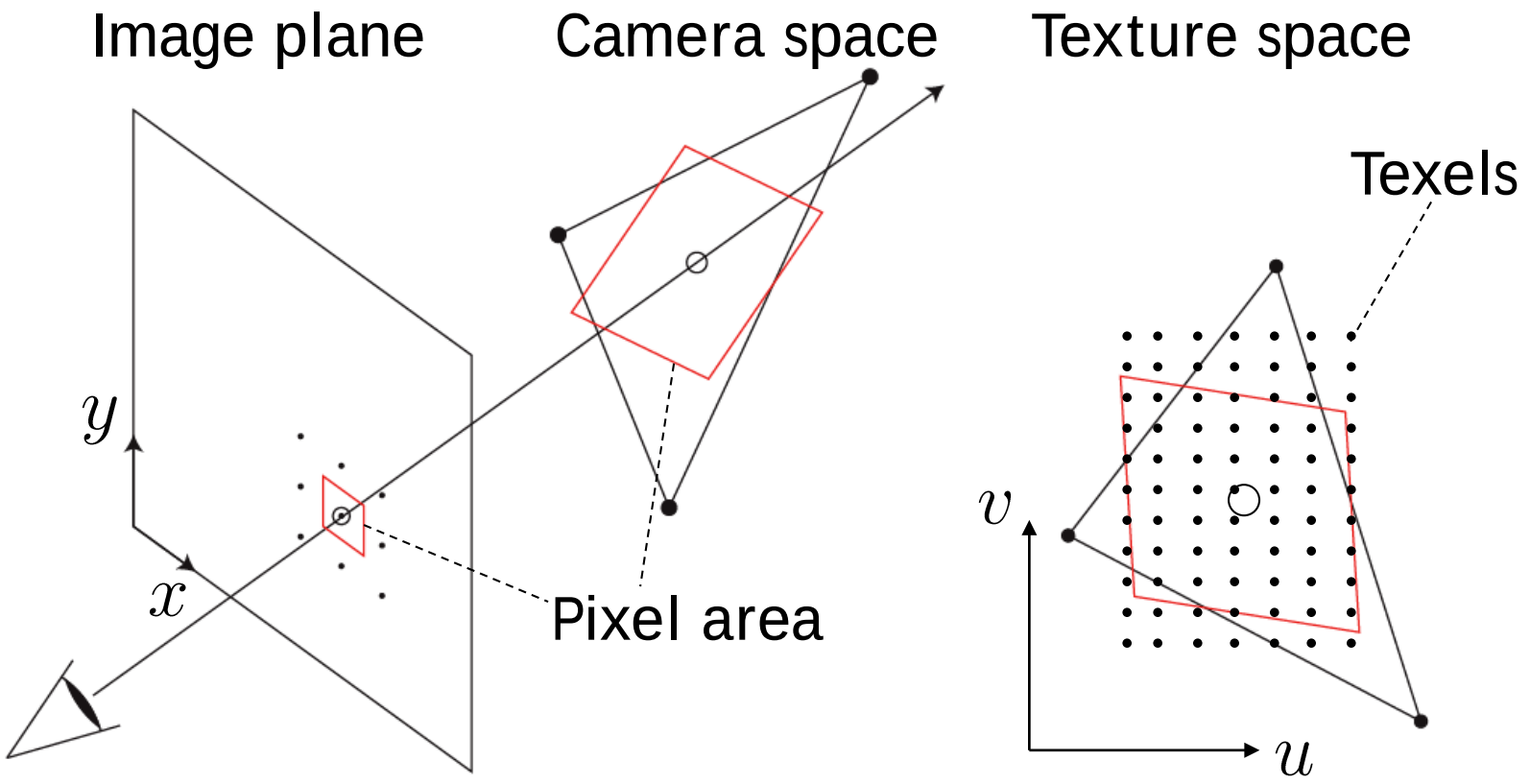


Image: Robert L. Cook

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



UCSD

Antialiasing Using Mip-Maps

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

Mipmaps

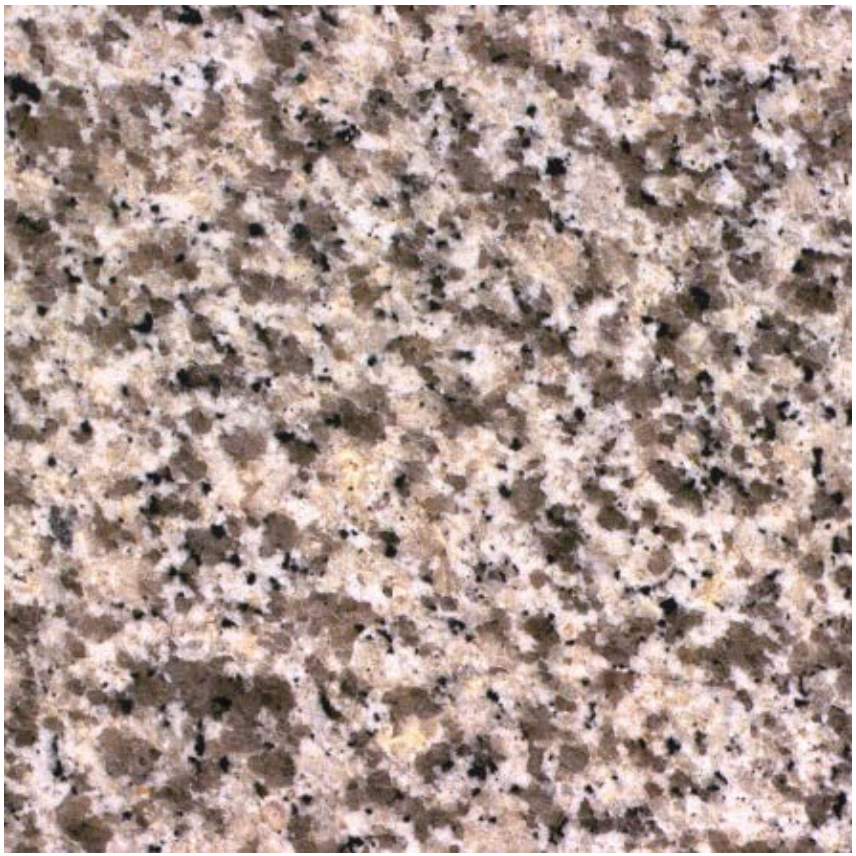
- MIP stands for *multum in parvo* = *"much in little"* (Williams 1983)

Before rendering

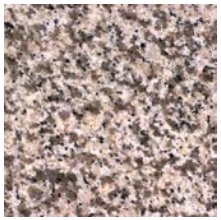
- Pre-compute and store down scaled versions of textures
  - Reduce resolution by factors of two successively
  - Use high quality filtering (averaging) scheme
- Increases memory cost by 1/3
  - 1/3 = ¼+1/16+1/64+…
- Width and height of texture should be powers of two (non-power of two supported since OpenGL 2.0)

Mipmaps

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



Level 1



2

3

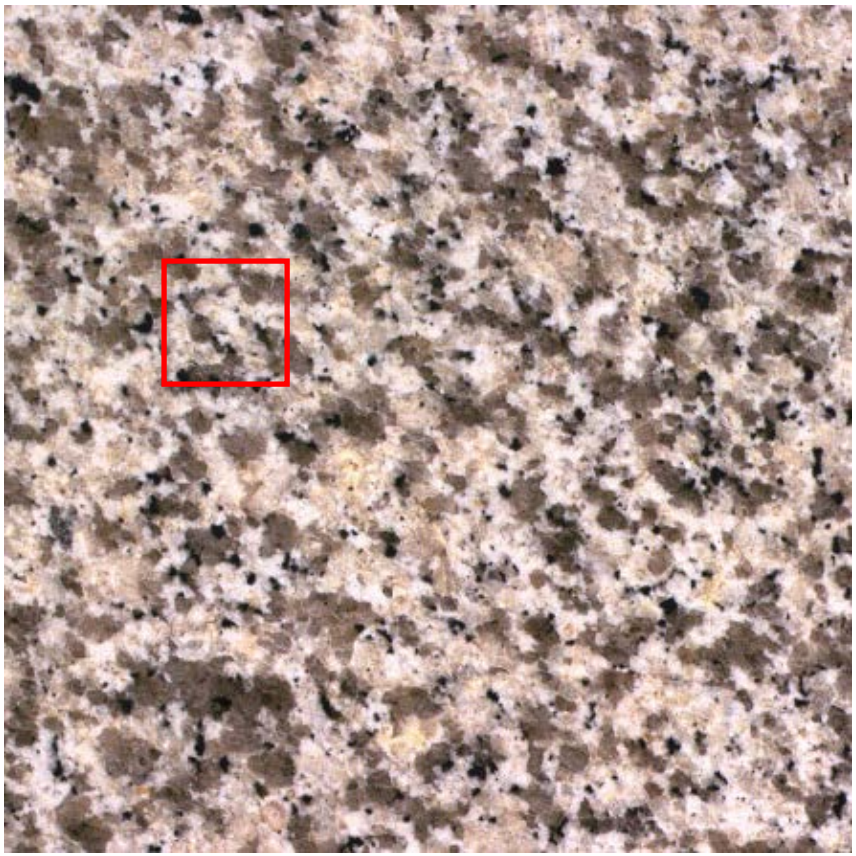
4

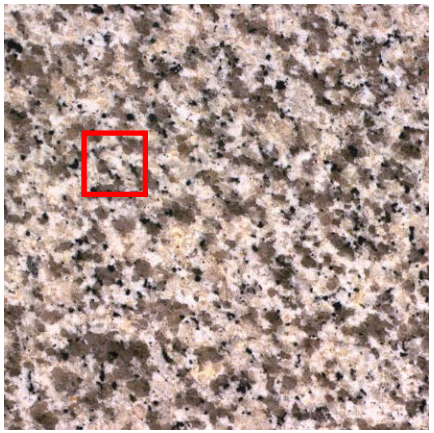
"multum in parvo"

 Level 0

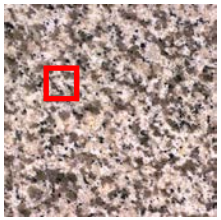
Mipmaps

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





Level 1



4

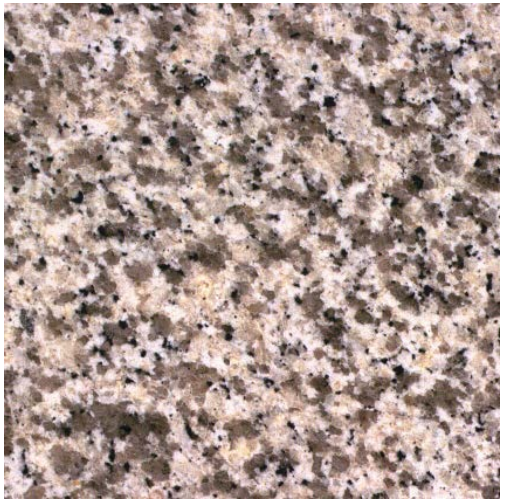
3

2

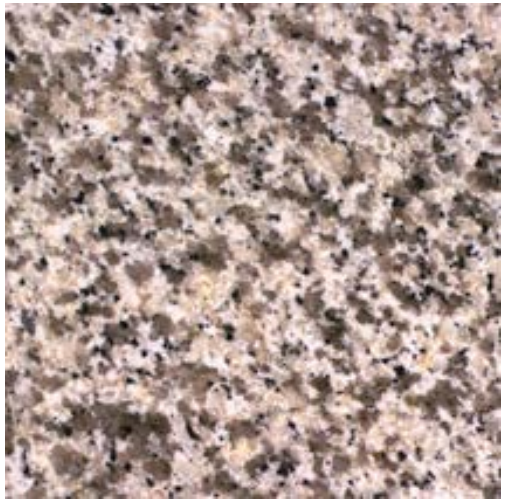
"multum in parvo"

Level 0

Mipmaps



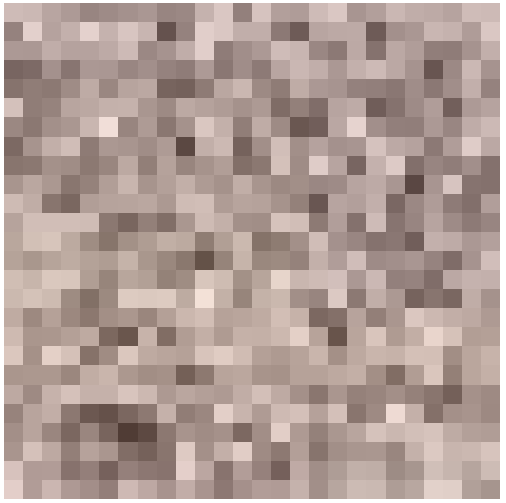
Level 0



Level 1

Level 2

Level 3

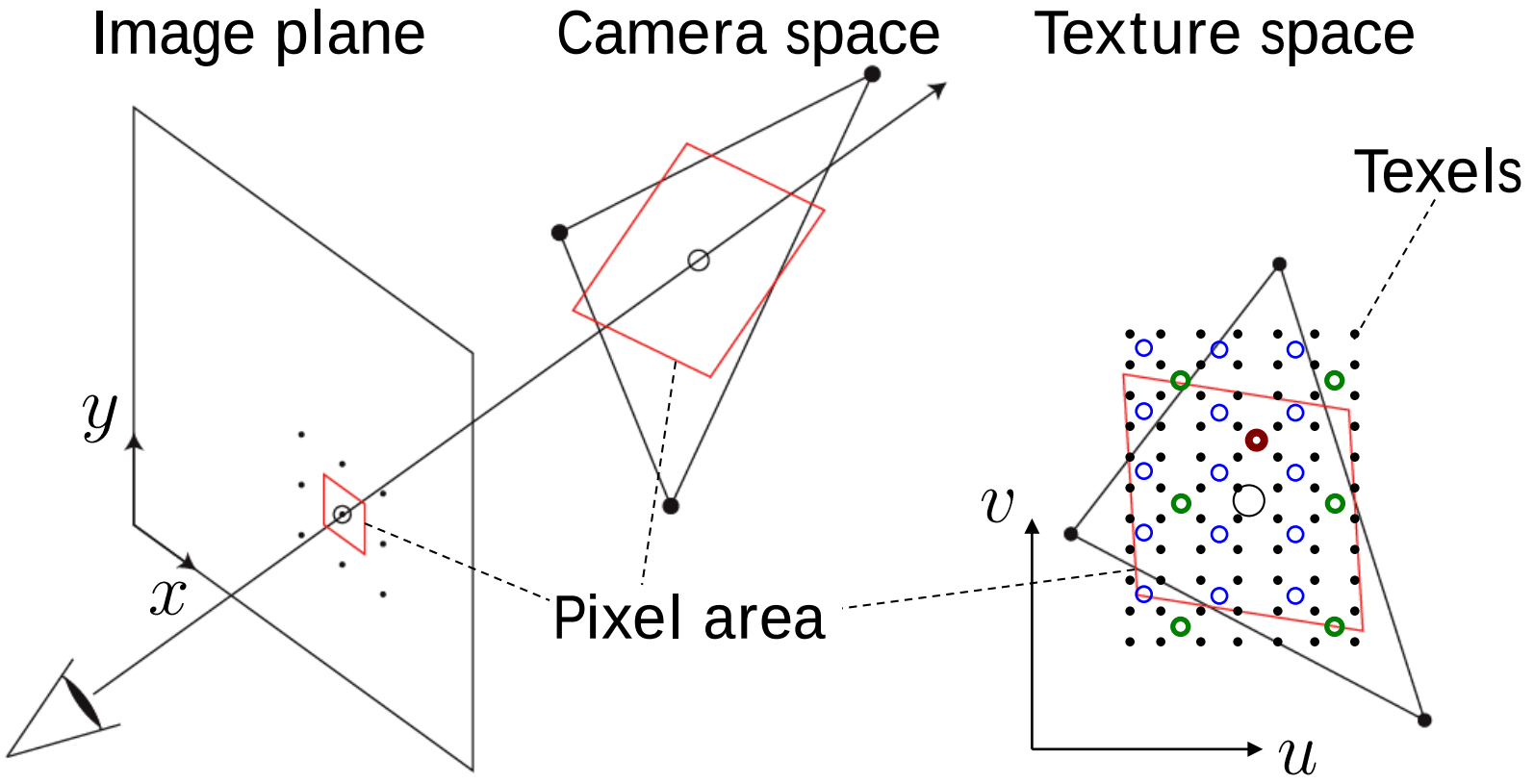


Level 4

Rendering With Mipmaps

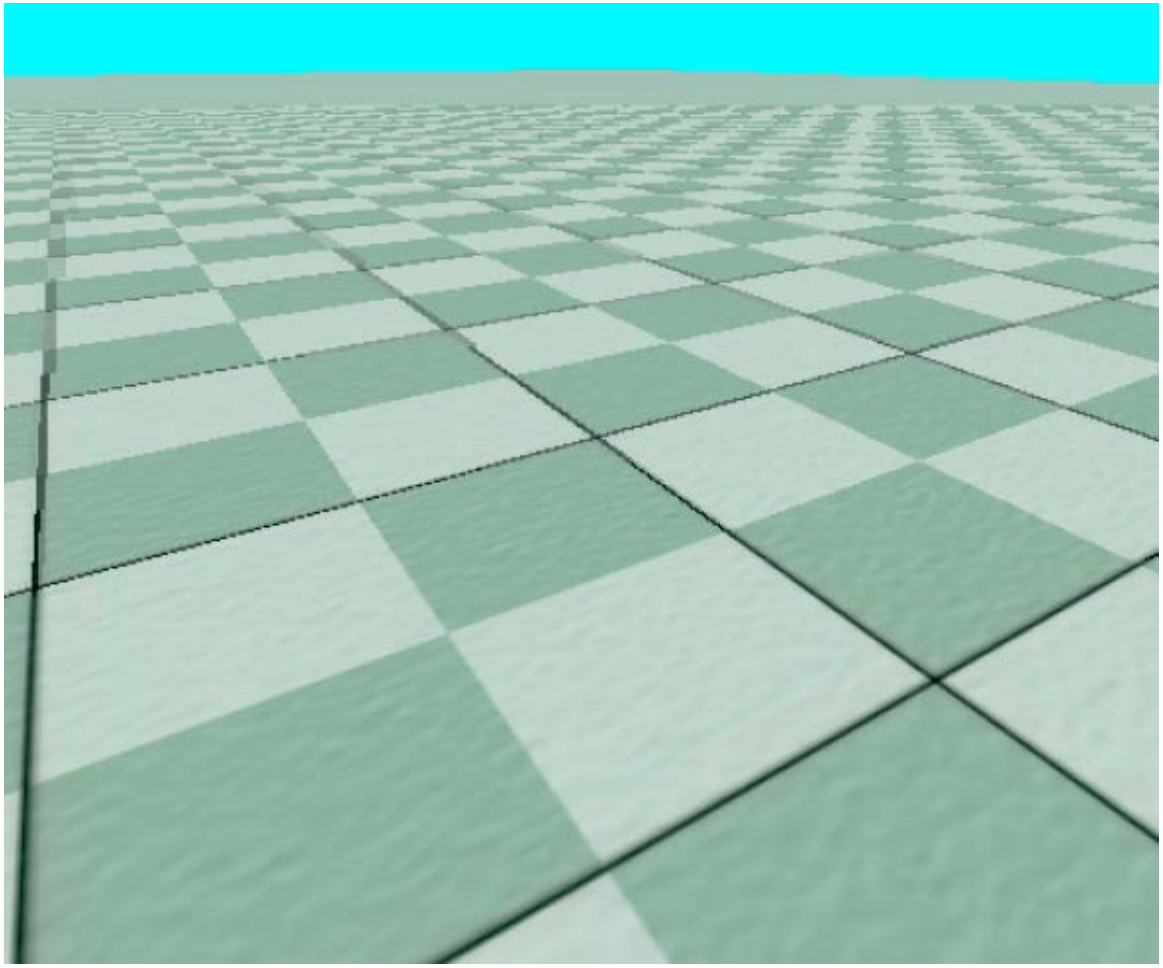
- "Mipmapping"
- Interpolate texture coordinates of each pixel as without mipmapping
- Compute approximate size of pixel in texture space
- Look up color in nearest mipmap
  - E.g., if pixel corresponds to 10x10 texels use mipmap level 3
  - Use nearest neighbor or bilinear interpolation as before

Mipmapping



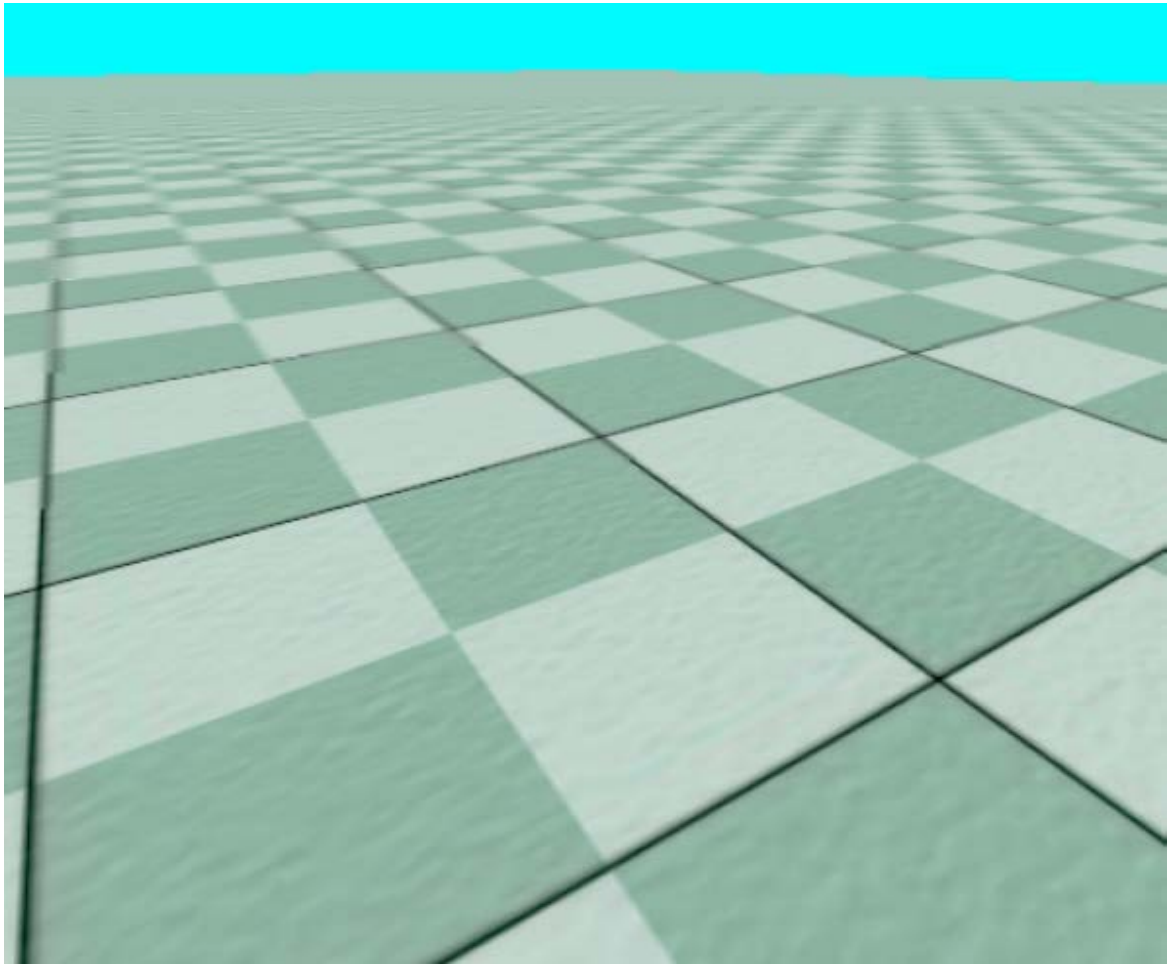
Nearest Mipmap, Nearest Neighbor

- Visible transition between mipmap levels



Nearest Mipmap, Bilinear

- 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:
  - Step 1: perform bilinear interpolation in both mip-maps
  - Step 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

More Info

- Mipmapping tutorial w/source code:
  - http://www.videotutorialsrock.com/opengl_tutorial/mipmapping/text.php