CSE 167:
Introduction to Computer Graphics
Lecture 11: Textures 2

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2011

Announcements

- Homework assignment #5 due Friday, Nov 4, grading in lab 260 starts as usual at 1:30pm
- StarCAVE demonstration Nov 15 after class
  - Instructor's office hour moves from Nov 15 to Nov 17 (same time)
- Next lecture:
  - Midterm review
  - Midterm exams distributed

Lecture Overview

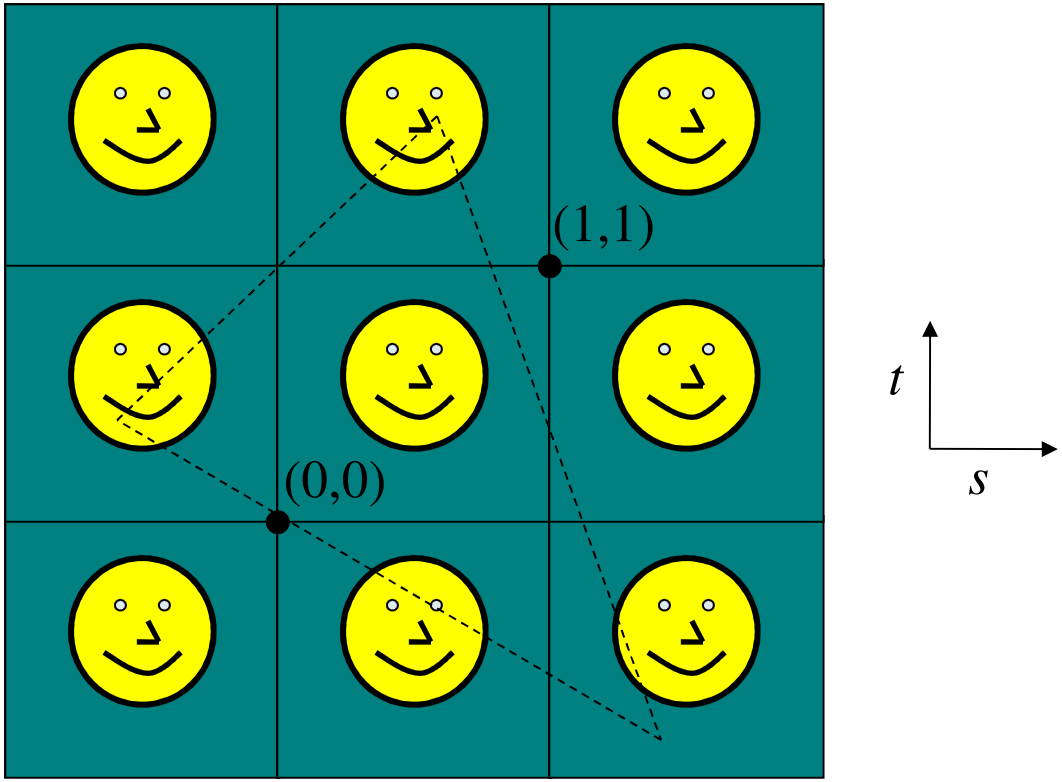
- Texturing
  - Wrapping
  - Texture coordinates
  - Anti-aliasing

Wrap Modes

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

Repeat

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

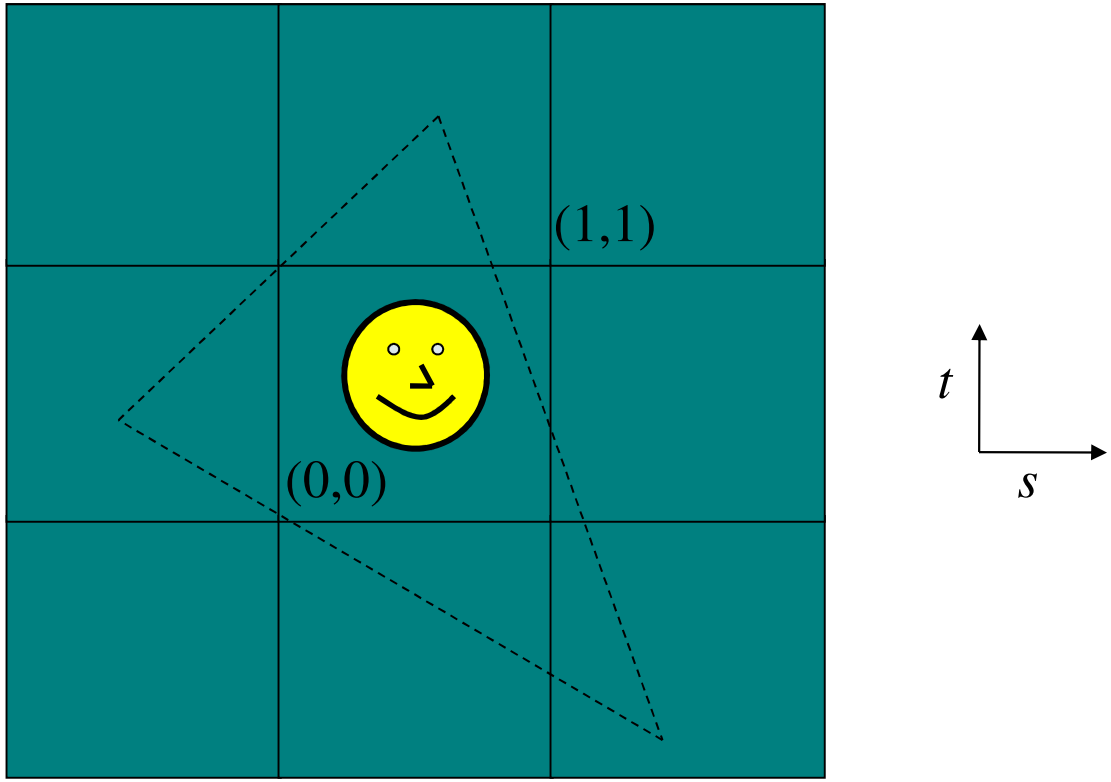


Texture Space

Seamless brick wall texture
(by Christopher Revoir)

Clamp

- Use edge value everywhere outside data range [0..1]
- Or, ignore the texture outside [0..1]



Texture Space

Wrap Mode Specification 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_CLAMP (requires border to be set) GL_CLAMP_TO_EDGE (repeats last pixel in texture), GL_REPEAT

Lecture Overview

- Texturing
  - Wrapping
  - Texture coordinates
  - Anti-aliasing

Texture Coordinates

What if texture extends across multiple polygons?

→ Surface parameterization

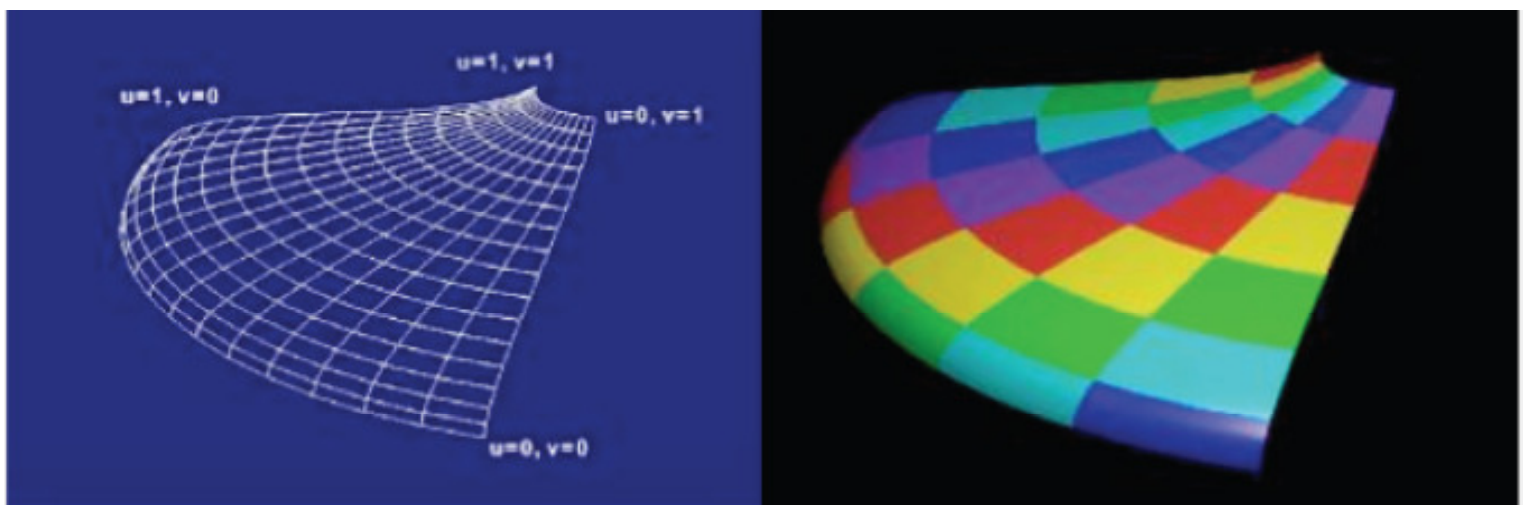
- Mapping between 3D positions on surface and 2D texture coordinates
  - In practice, defined by texture coordinates of triangle vertices
- Options for mapping:
  - Parametric
  - Orthographic
  - Projective
  - Spherical
  - Cylindrical
  - Skin

Parametric Mapping

- Surface given by parametric functions

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

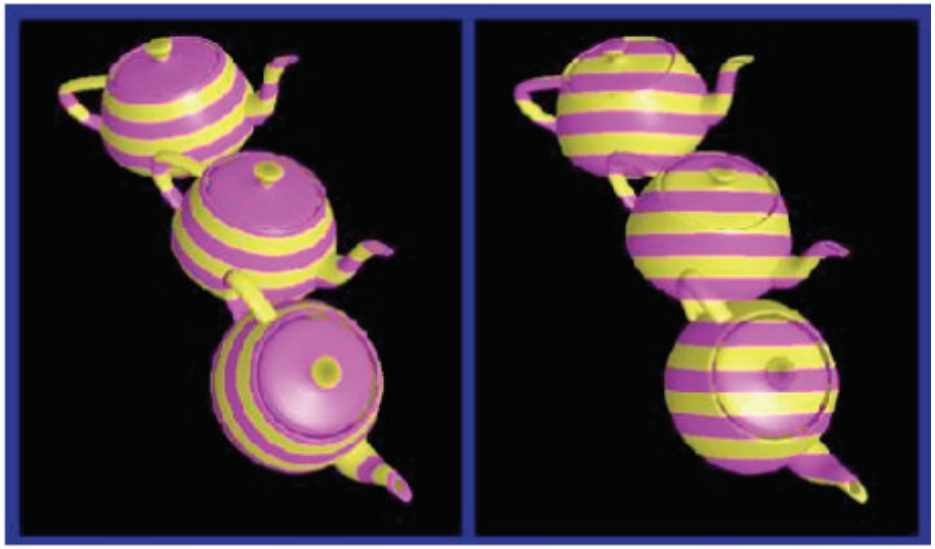
- Very common in CAD
- Clamp (u,v) parameters to [0..1] and use as texture coordinates (s,t)



Orthographic Mapping

- Use linear transformation of object's xyz coordinates
- For example:

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

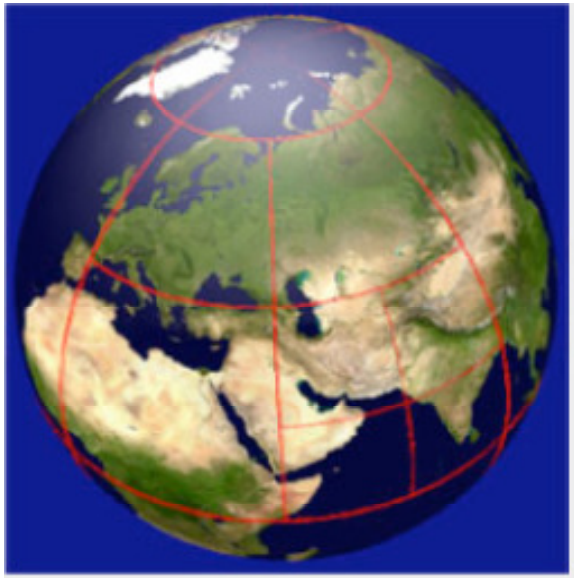


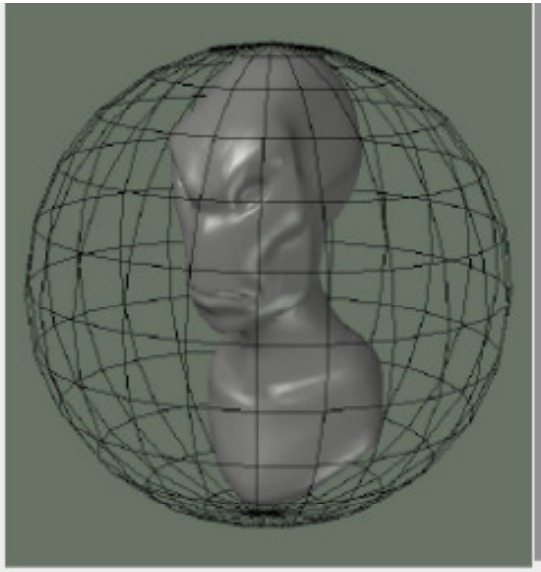
Projective Mapping

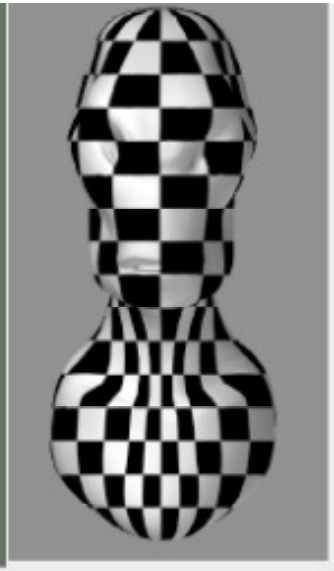
- Use perspective projection of xyz coordinates
  - OpenGL provides GL_TEXTURE matrix to apply perspective projection on texture coordinates
- Can be used for "fake" lighting effects

Spherical Mapping

- Use spherical coordinates for sphere
- Place object in sphere
- "shrink-wrap" sphere to object

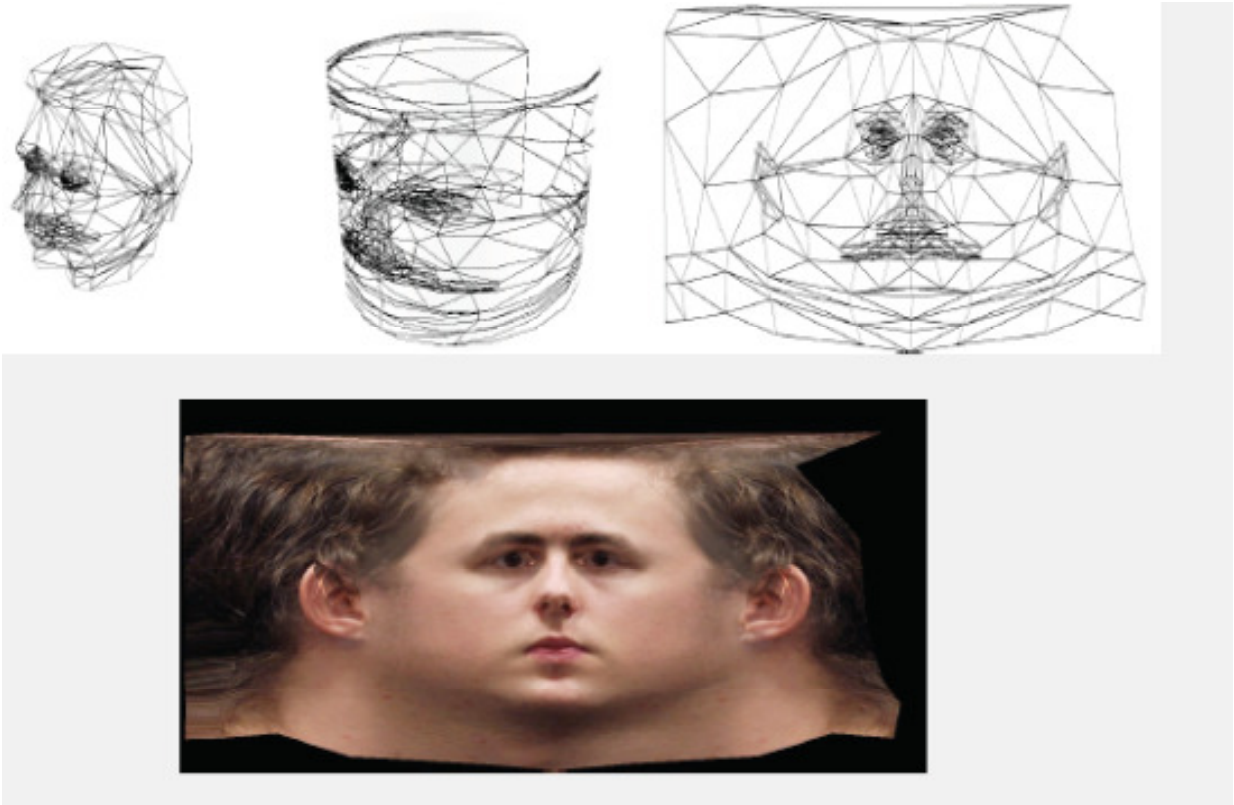






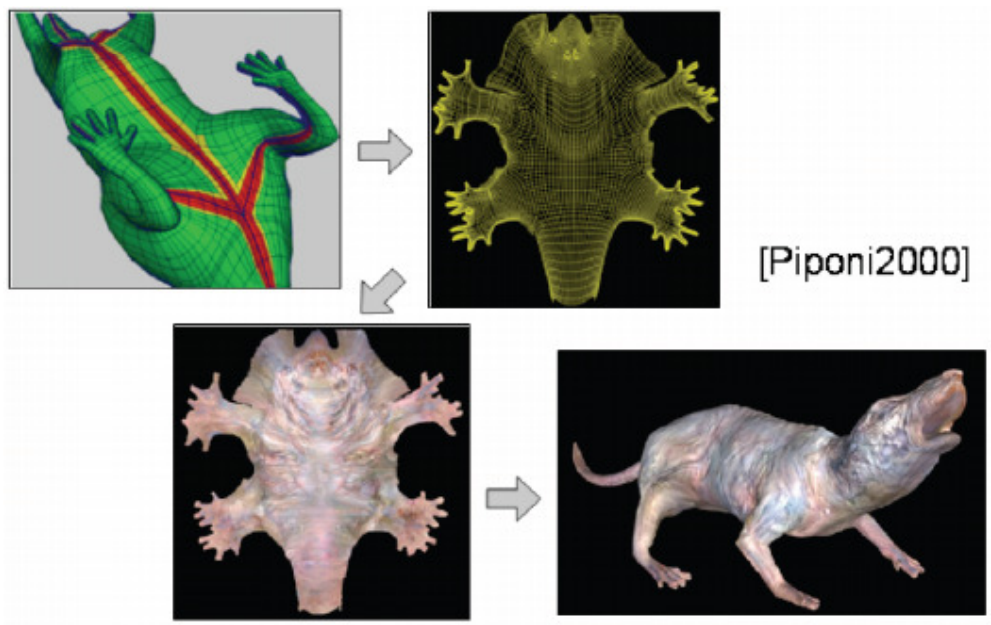
Cylindrical Mapping

- Similar as spherical mapping, but using cylinder
- Useful for faces



Skin Mapping

- Complex technique to unfold surface onto plane
- Preserve area and angle
- Sophisticated mathematics

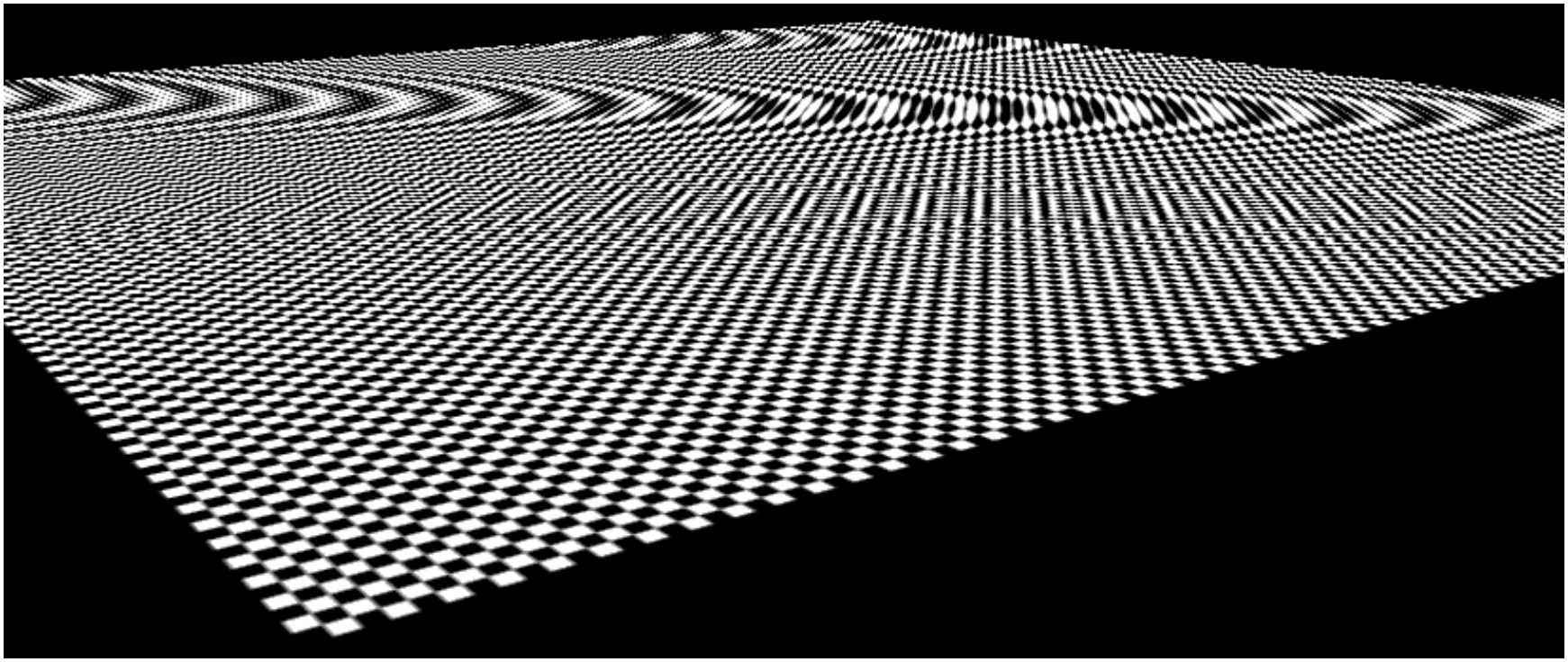


Lecture Overview

- Texturing
  - Wrapping
  - Texture coordinates
  - Anti-aliasing

Example for Aliasing

- What could cause this effect?



Aliasing

Sufficiently sampled, no aliasing

Insufficiently sampled, aliasing

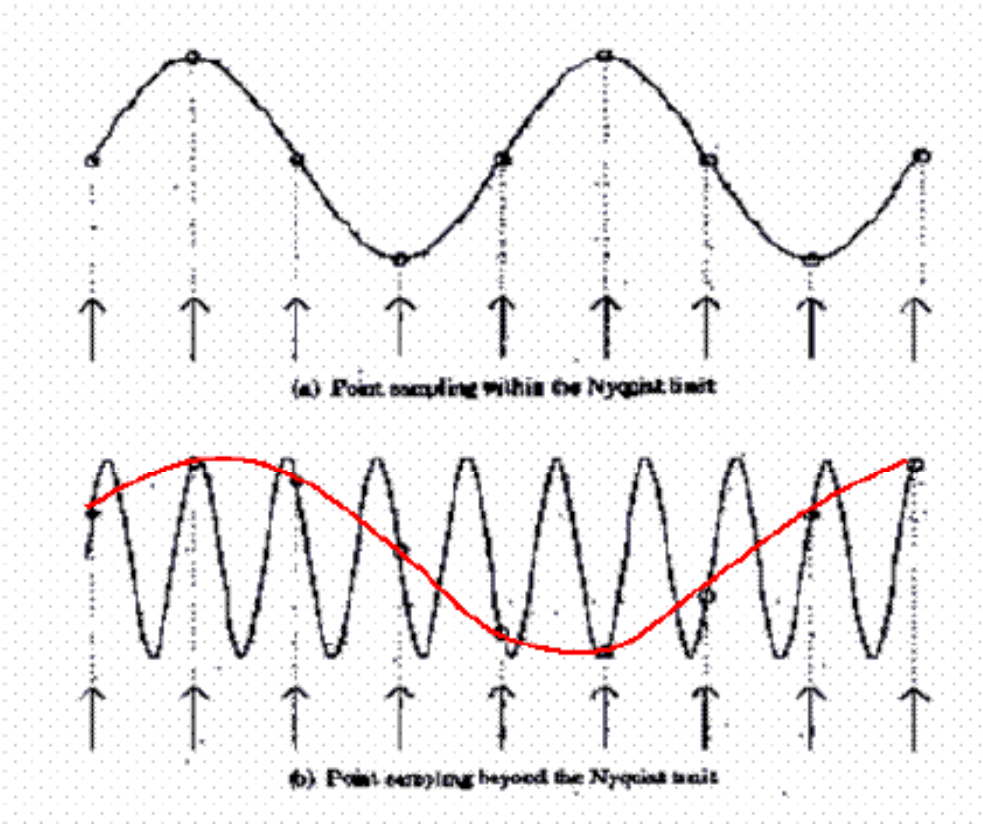
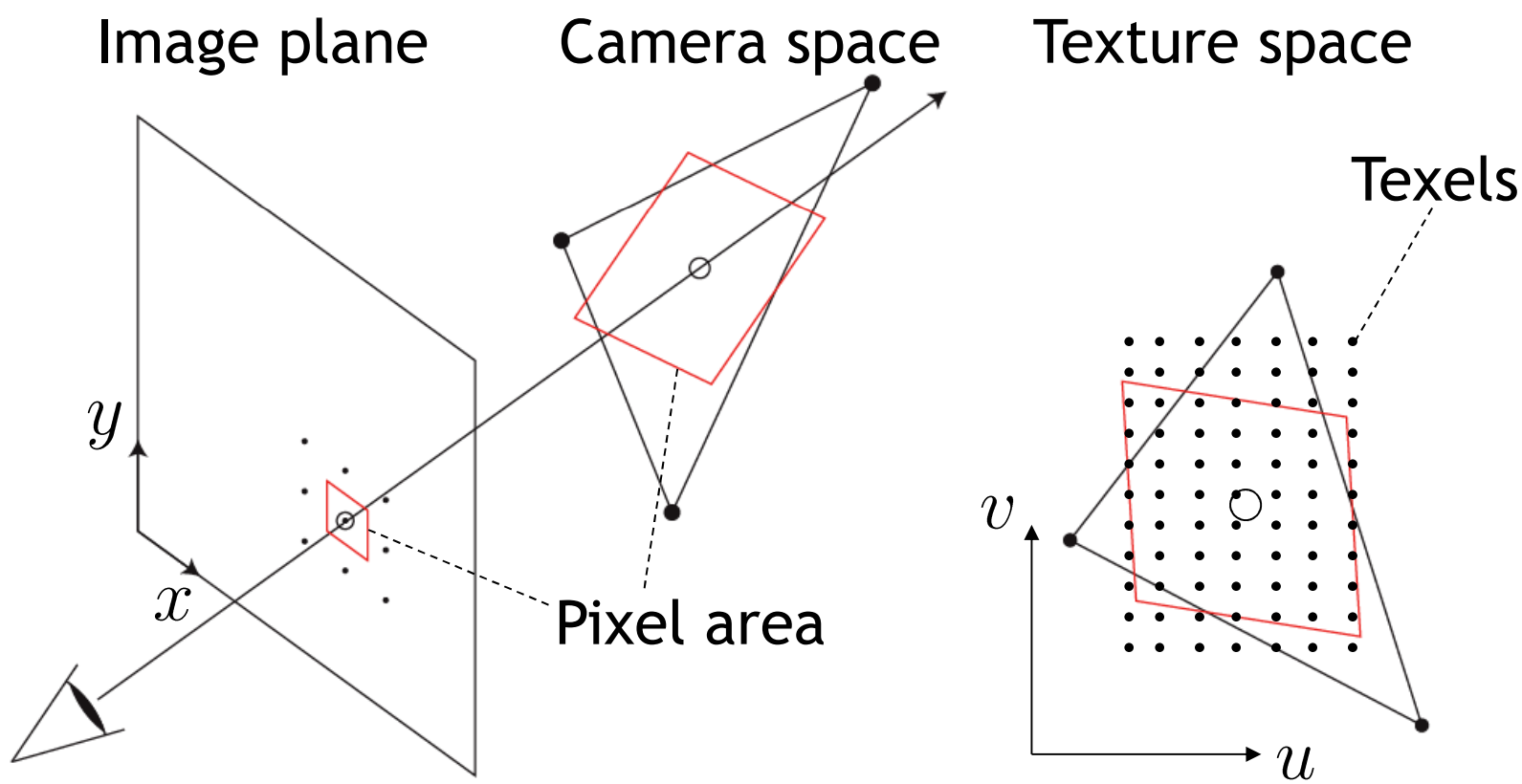


Image: Robert L. Cook

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: Mathematics

- Pixels are sample points, not little squares!
  - See article by Dr. Alvy Ray Smith, Microsoft: http://alvyray.com/Memos/CG/Microsoft/6_pixel.pdf
- Antialiasing is achieved through low-pass filtering
- For more information:
  - Master's Thesis: Fundamentals of Texture Mapping and Image Warping, Paul Heckbert, UC Berkeley: http://www.cs.cmu.edu/~ph/texfund/texfund.pdf

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

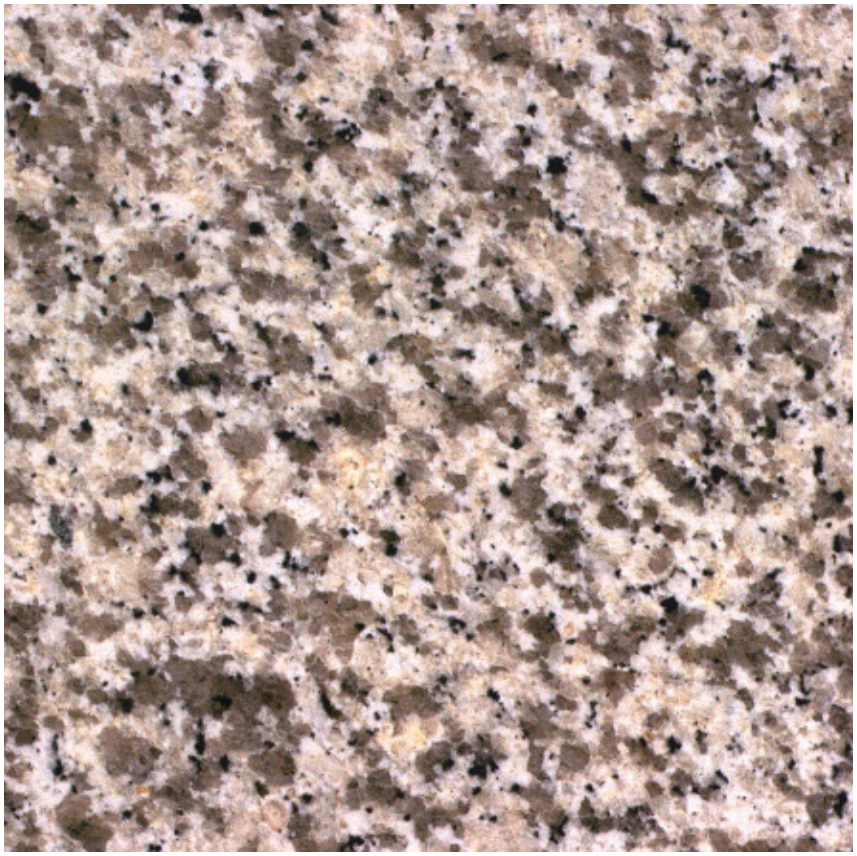
- MIP stands for *multum in parvo* = "*much in little*" (Williams 1983)

Before rendering

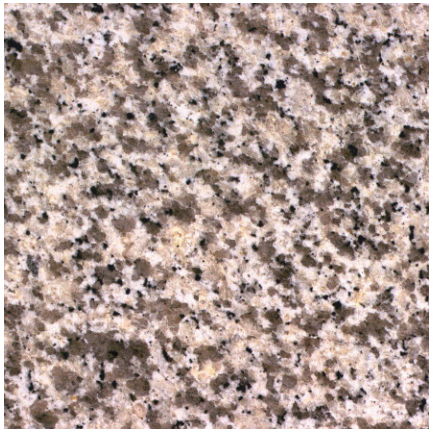
- Pre-compute and store down scaled versions of textures
  - Reduce resolution by factors of two successively
  - Use high quality filtering (averaging) scheme
- Increases memory cost by 1/3
  - 1/3 = ¼+1/16+1/64+…
- Width and height of texture should be powers of two (non-power of two supported since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

Mipmaps

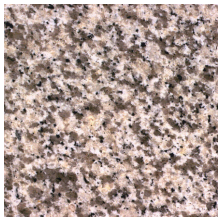
- Example: resolutions 512x512, 256x256, 128x128, 64x64, 32x32 pixels



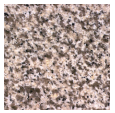
Level 0



Level 1



2



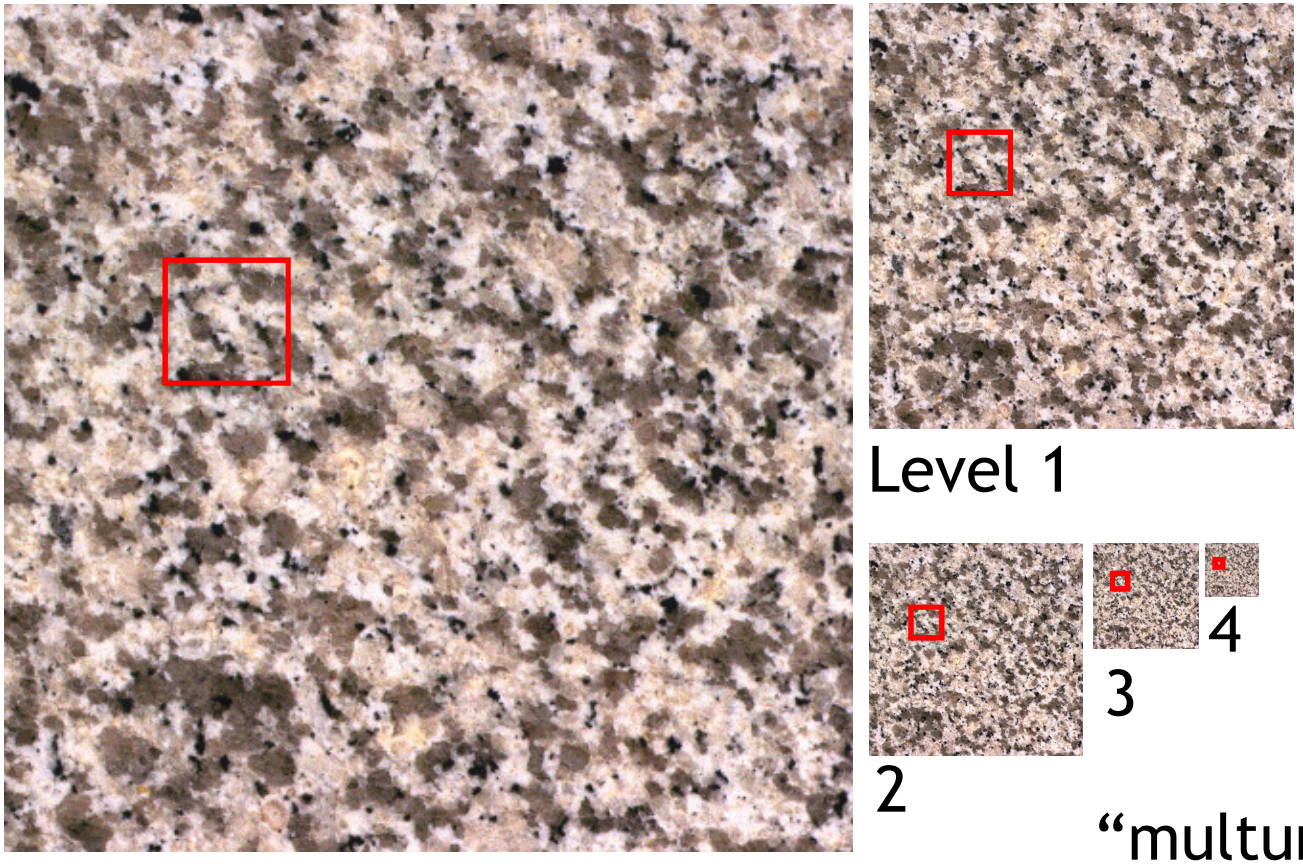
3

4

"multum in parvo"

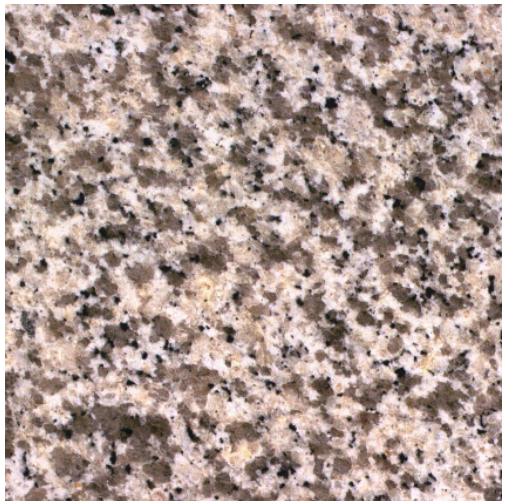
Mipmaps

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

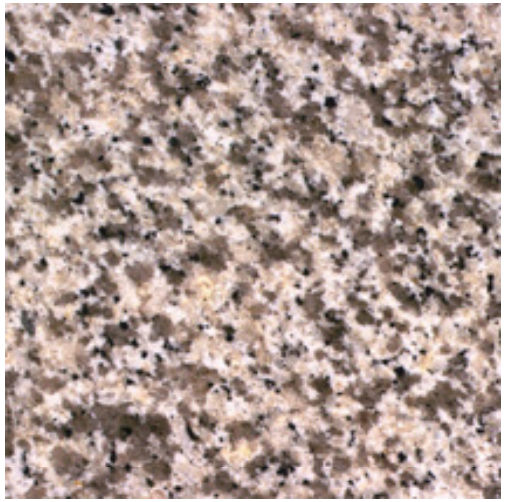


"multum in parvo"

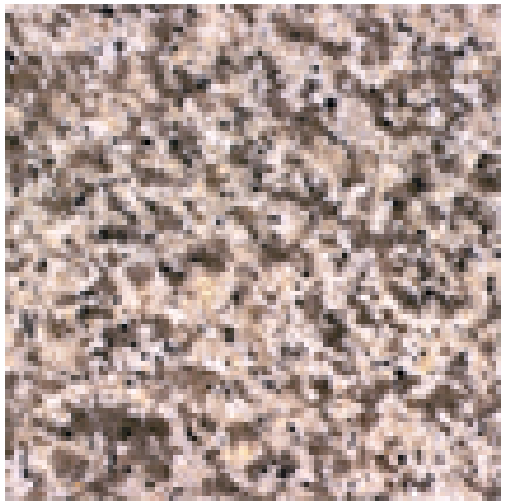
Mipmaps



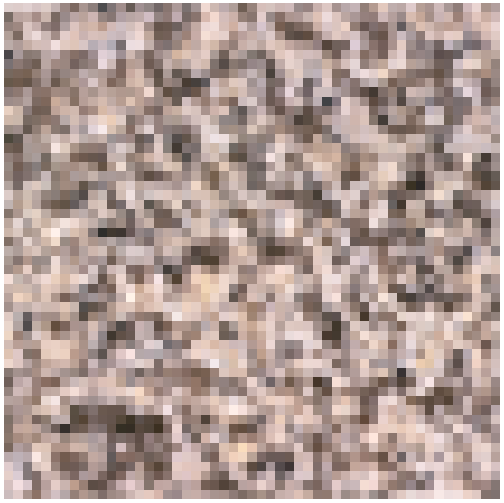
Level 0



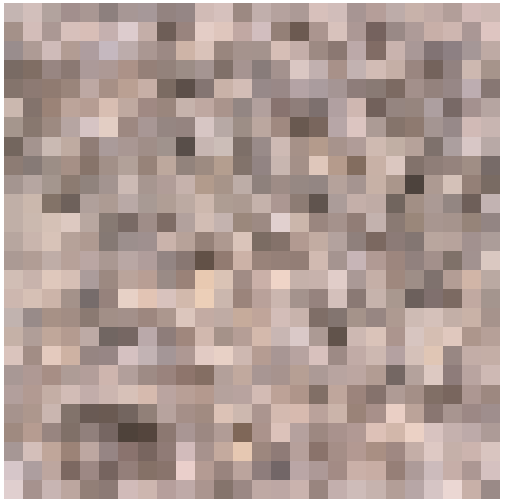
Level 1



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

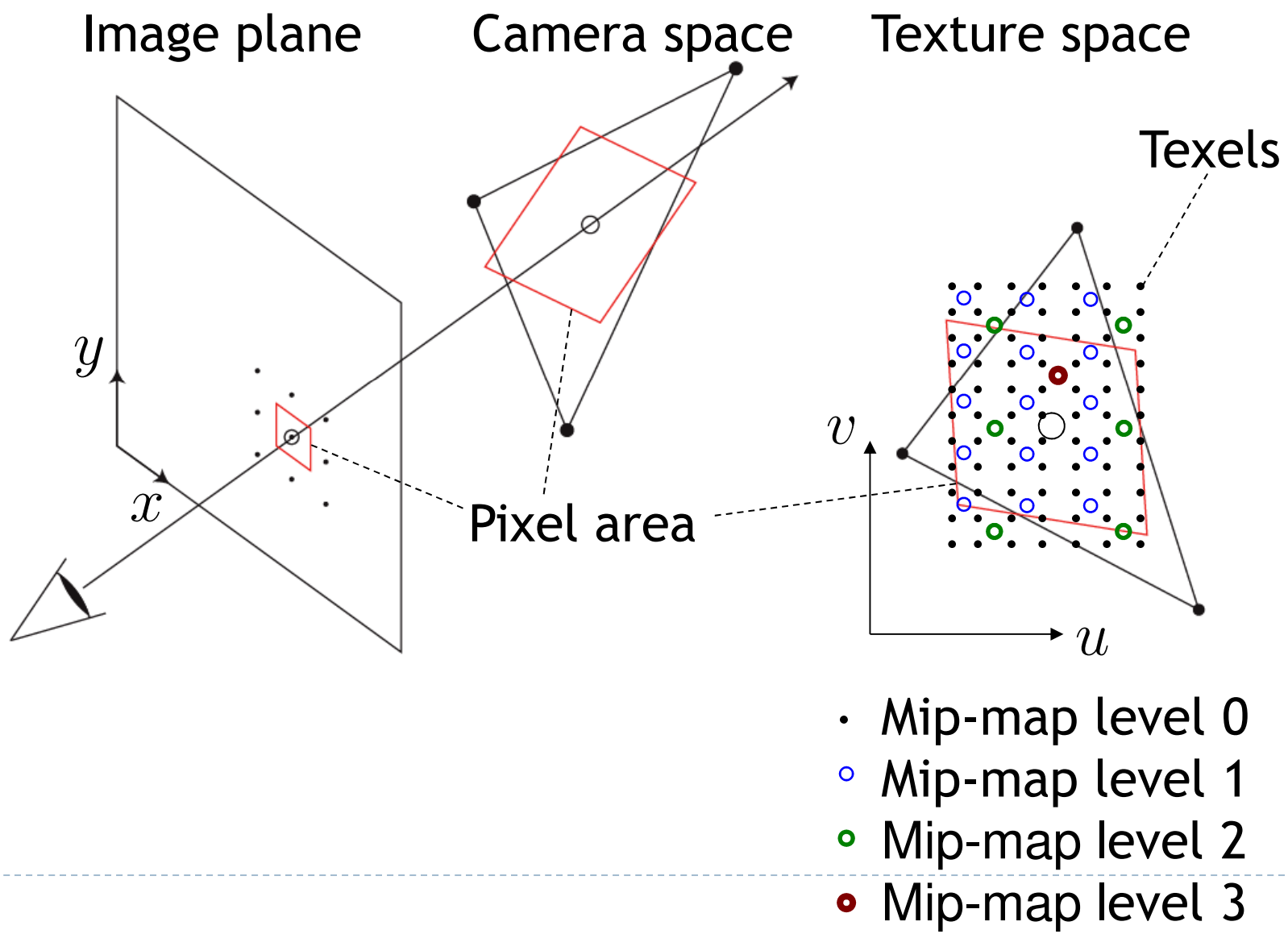
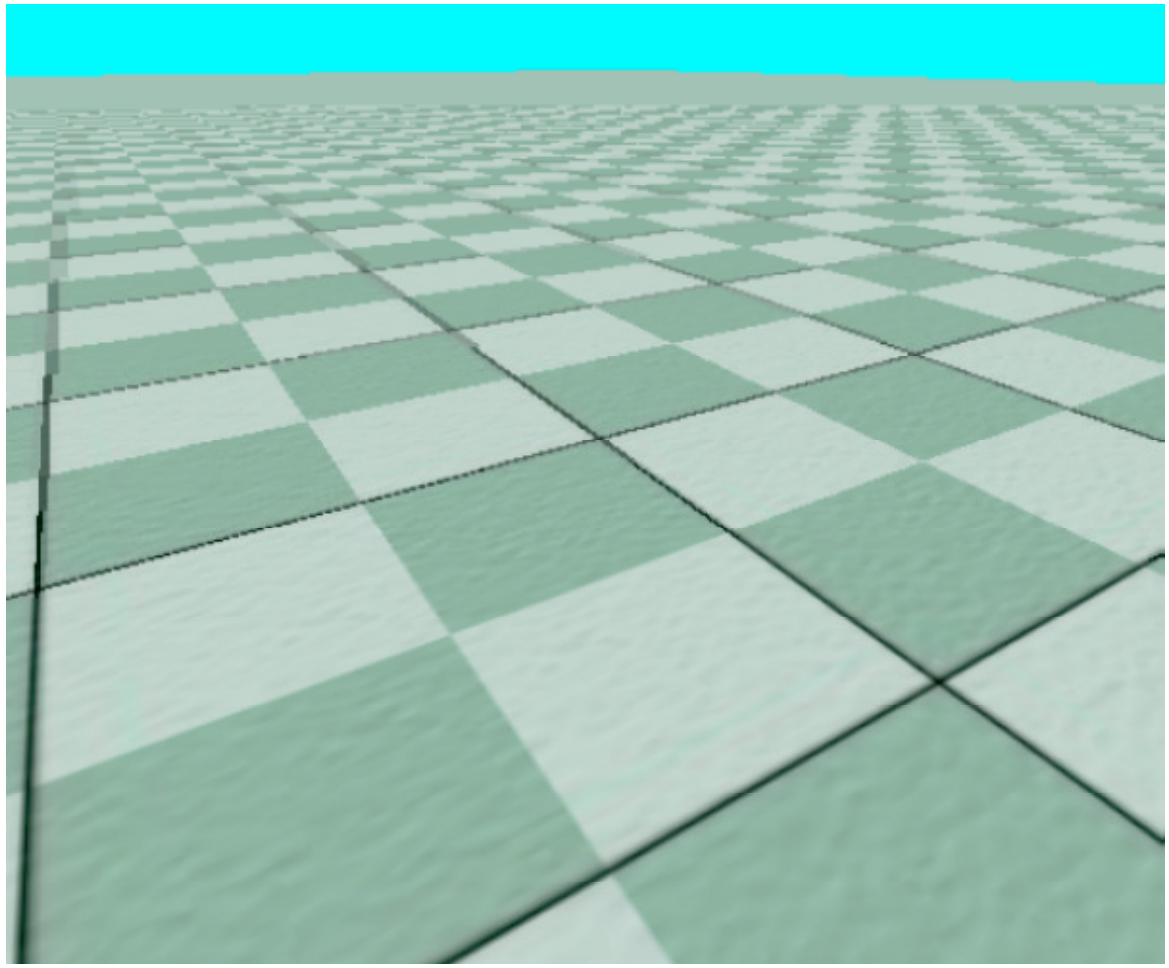


Image plane
Camera space
Texture space
Texels
y
x
Pixel area
v
u
Mip-map level 0
Mip-map level 1
Mip-map level 2
Mip-map level 3

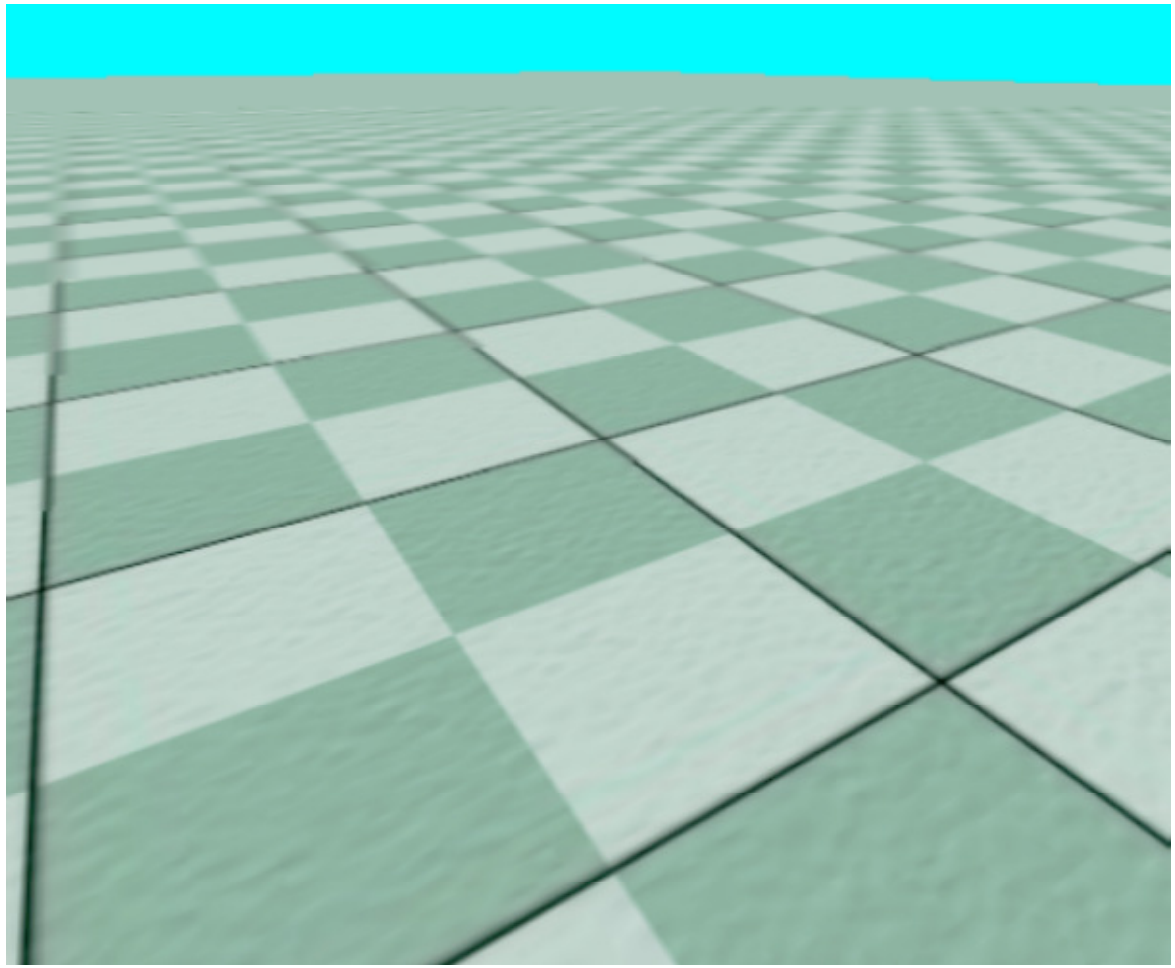
Nearest Mipmap, Nearest Neighbor

- Visible transition between mipmap levels



Nearest Mipmap, Bilinear

- Visible transition between mipmap levels

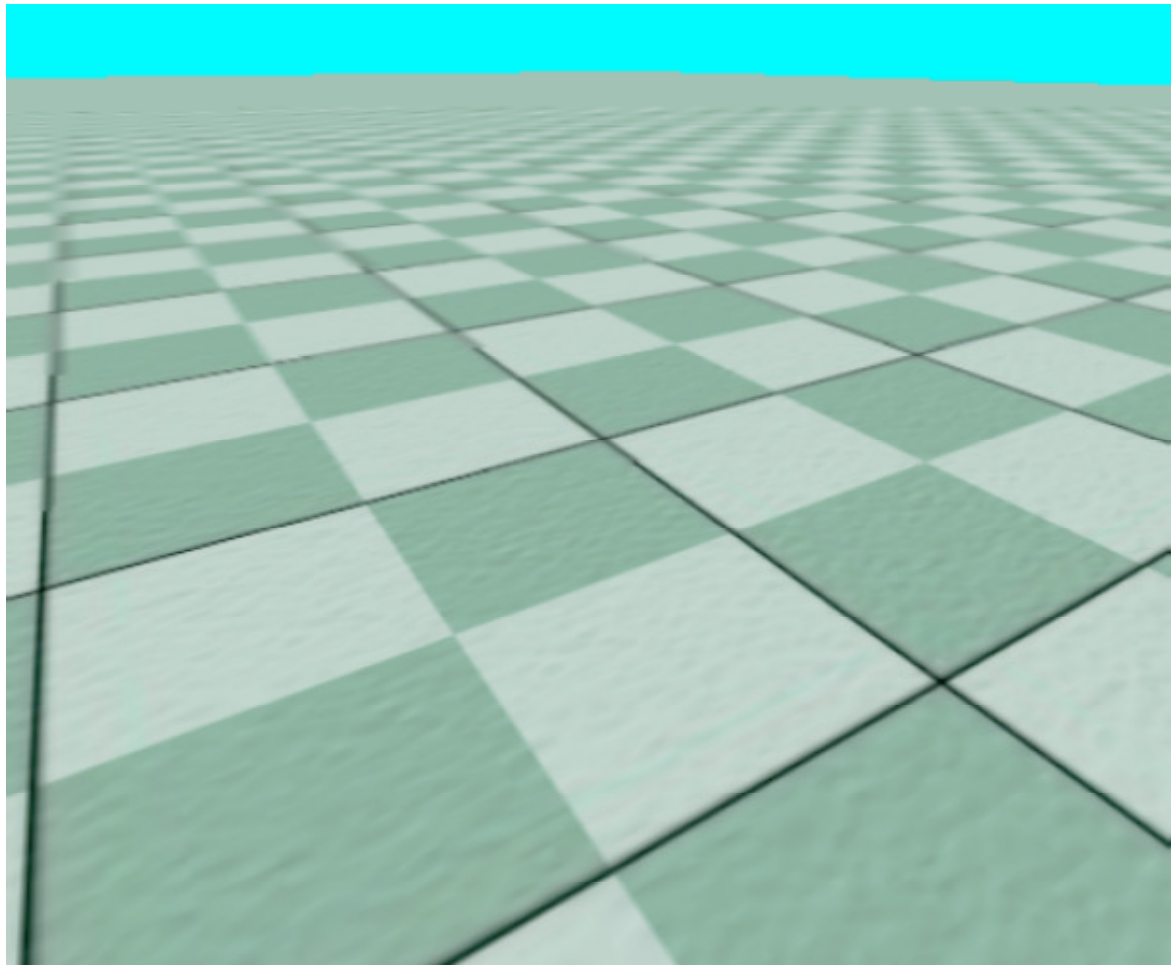


Trilinear Mipmapping

- Use two nearest mipmap levels
  - E.g., if pixel corresponds to 10x10 texels, use mipmap levels 3 (8x8) and 4 (16x16)
- Perform bilinear interpolation in both mip-maps
- Linearly interpolate between the results
- Requires access to 8 texels for each pixel
- Standard method, supported by hardware with no performance penalty

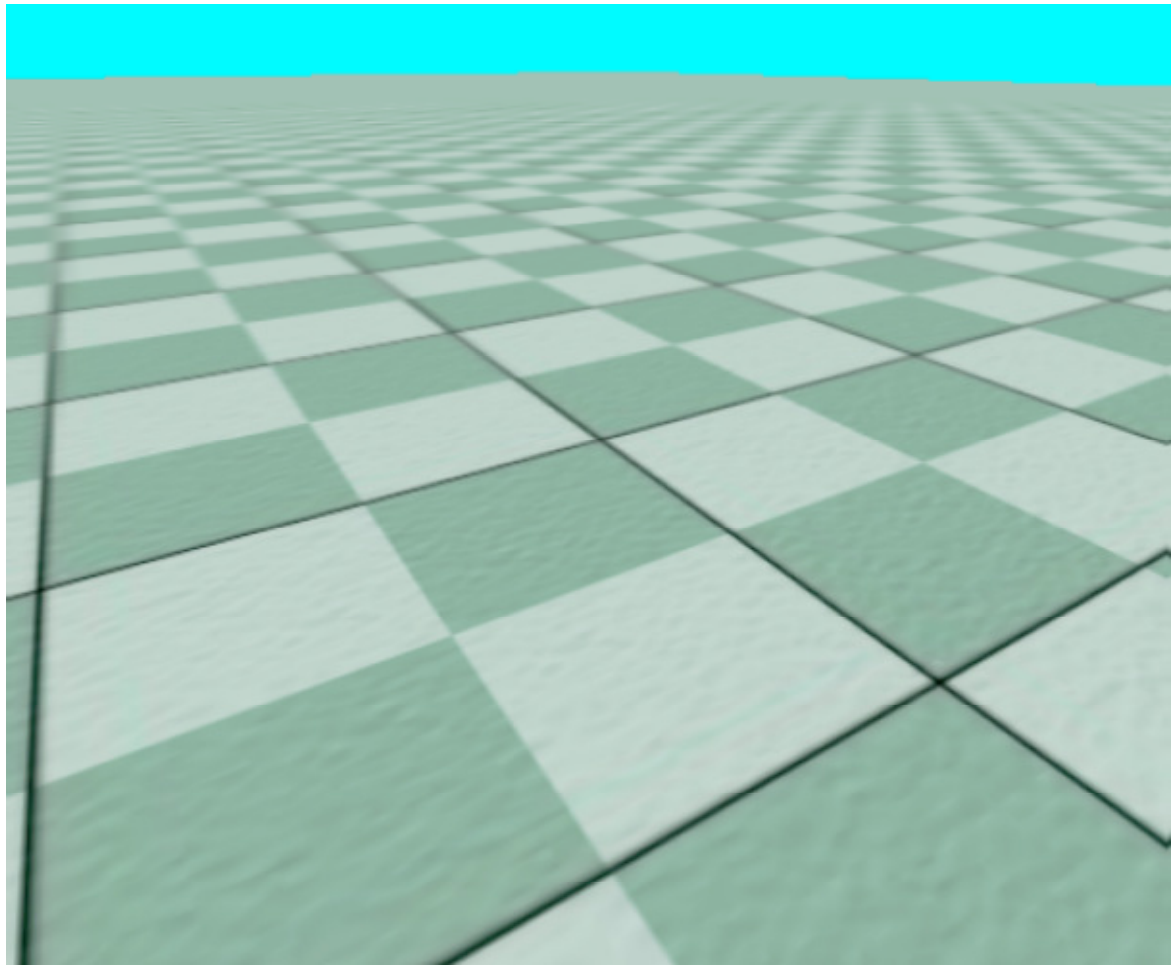
Nearest Mipmap, Bilinear

- Visible transition between mipmap levels



Trilinear Mipmapping

- Smooth transition between mipmap levels



Next Lecture

- Midterm review
- Curves