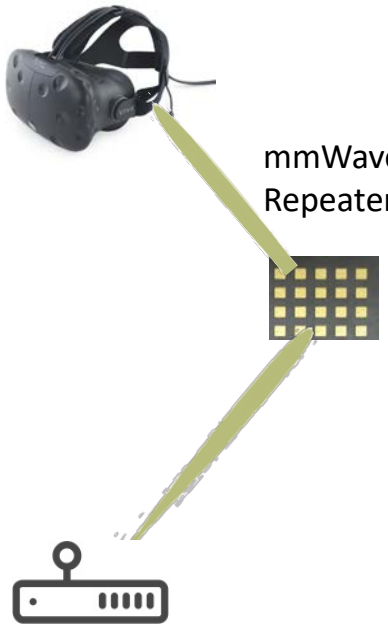
CSE 167:
Introduction to Computer Graphics
Lecture #9: Mipmapping

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2017

Announcements

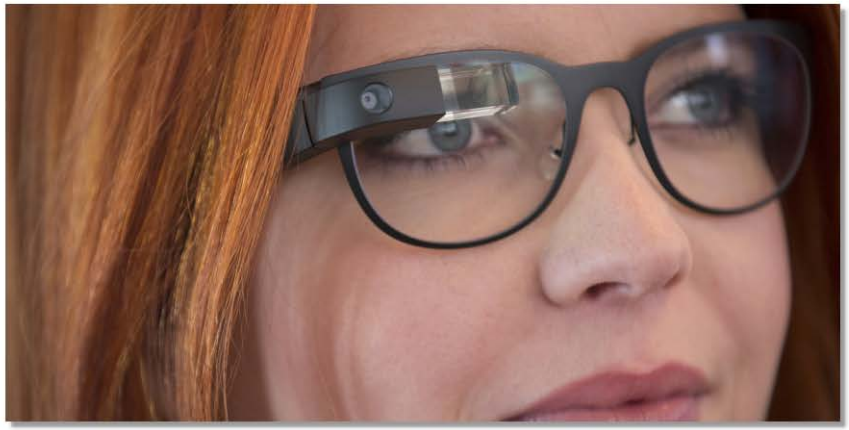
- Project 2 due tomorrow at 2pm
  - Grading in basement labs B260 and B270
  - Will use separate sign up lists on whiteboards in each room
- Midterm #1 next Tuesday in class
- Next Friday: late grading for project 2

Wireless VR – Professor Dinesh Bharadia



Wireless Virtual Reality

Can we untether the VR experience? What we need:



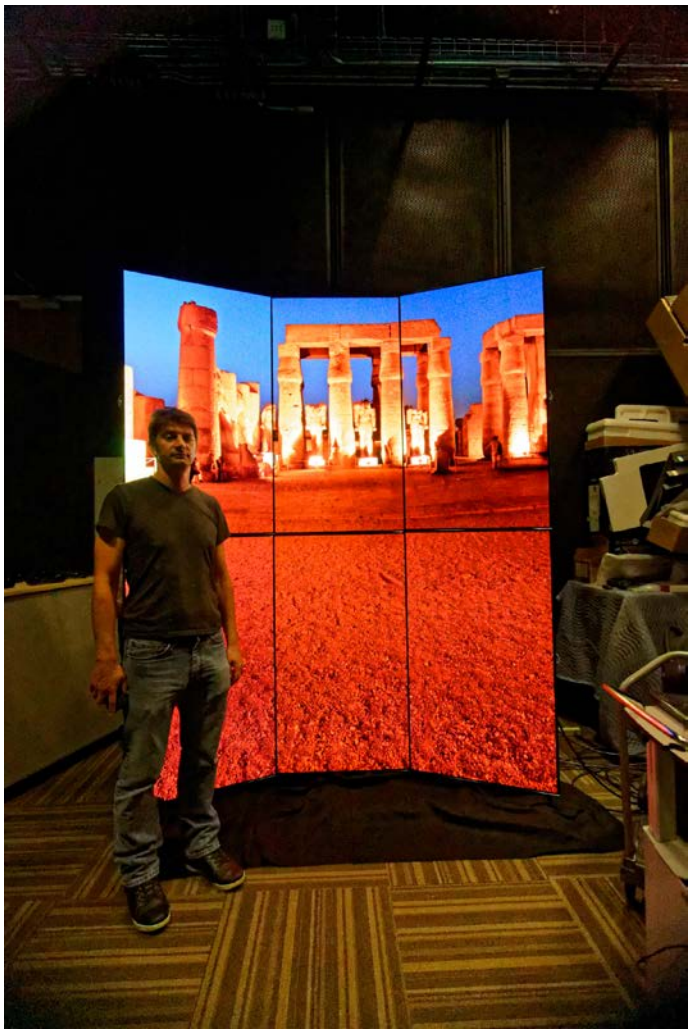
- Off-load compute and sensing to existing network infrastructure
- Fine-grained user context and sensing (head position, direction, speed) provided by the network
- Requires < 20ms RTT latency including compute, high bandwidth, move with user and rich video content

Can the wireless network become the platform for VR / AR?

UCSD

Unity Programmers Wanted

- Need Unity programmer(s) to improve UI on CAVE Kiosk at Geisel library
- Collaboration with Archaeology Department
- Windows 10 PC with 3 Nvidia GTX 1080 GPUs to drive 6 4k displays in 3D stereo
- Interaction with Xbox controller
- Starts immediately
- Can do for CSE 198 Independent Study credit



Ivy Film Festival

My son, Adam Hersko-RonaTas, currently a student at Brown University, is chief coordinator of the New Media section of the Ivy Film Festival, an annual event for student artists.
http://ivyfilmfestival.org/

He is soliciting new media artwork (eg. **360° films, VR games, AR applications**, projection/sound installations, etc.) with a strong narrative component. He wants to invite submissions from students at UCSD and asked me to help.

Who should he contact at UCSD? Where should he send the call?

Akos Rona-Tas
Professor of Sociology
University of California, San Diego

Lecture Overview

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

Texture Coordinates

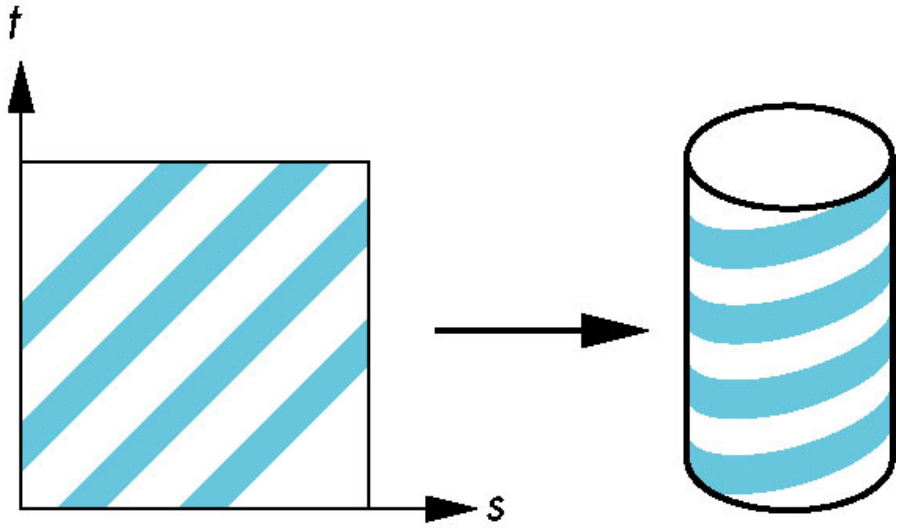
What if texture extends across multiple polygons?

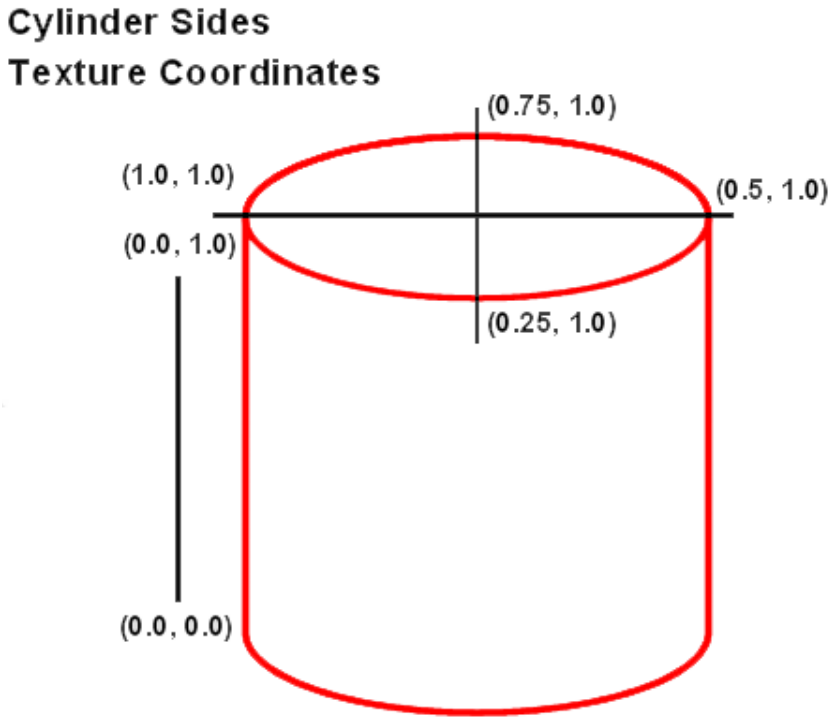
→ Surface parameterization

- 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

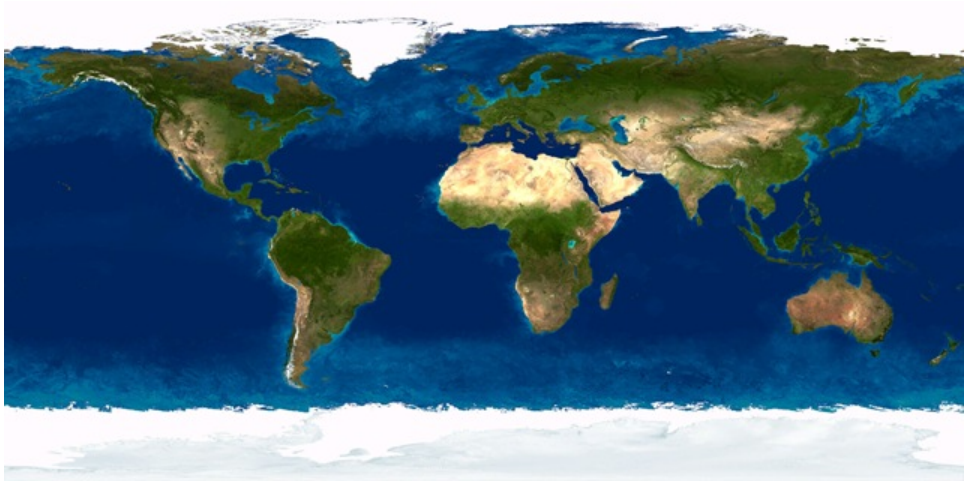




UCSD

Spherical Mapping

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



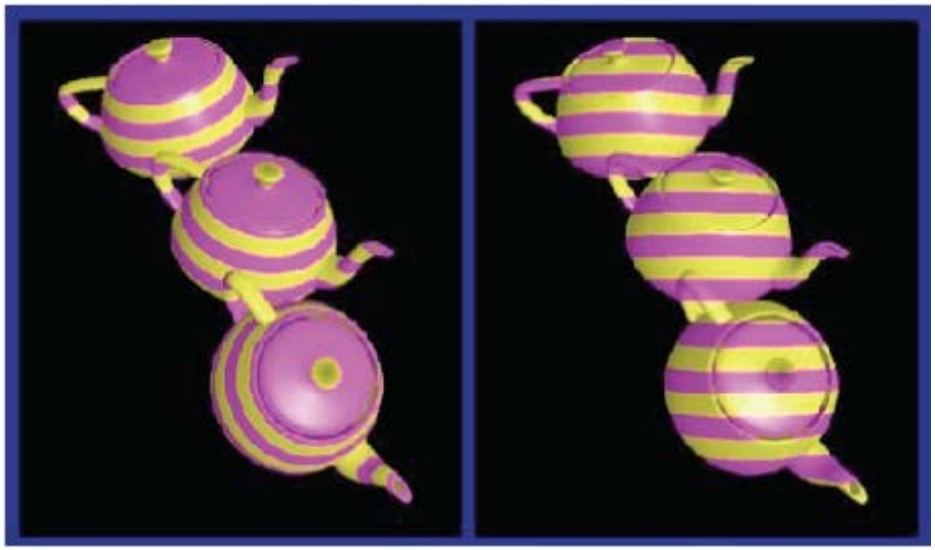
Texture map

Mapping result

Orthographic Mapping

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

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



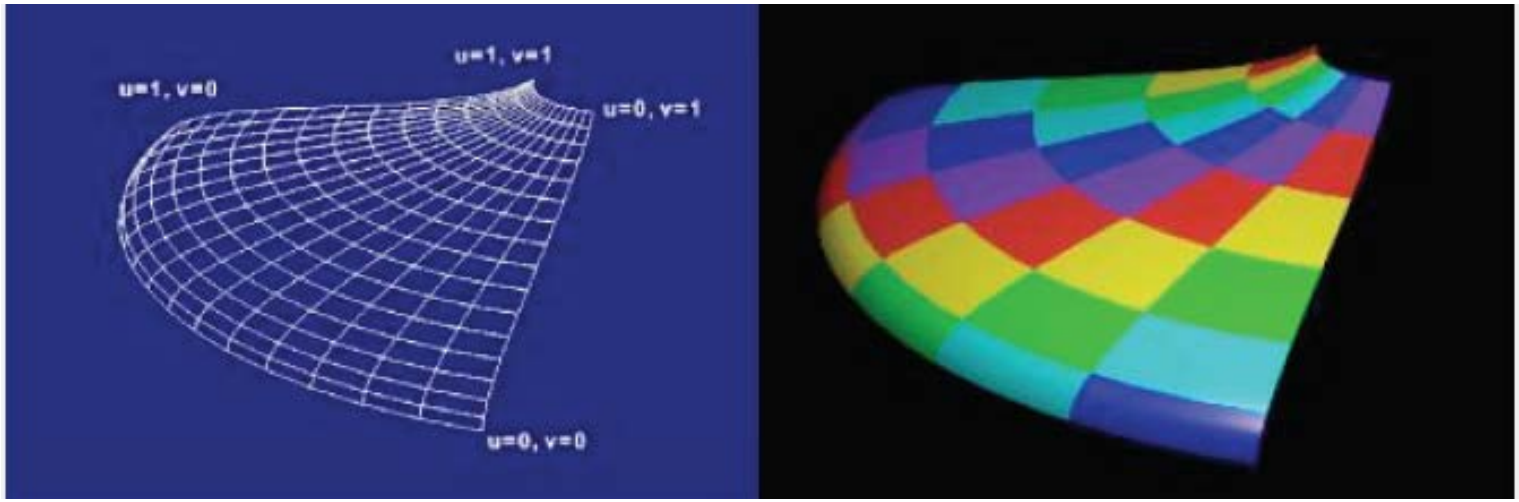
xyz in object space *xyz in camera space*

Parametric Mapping

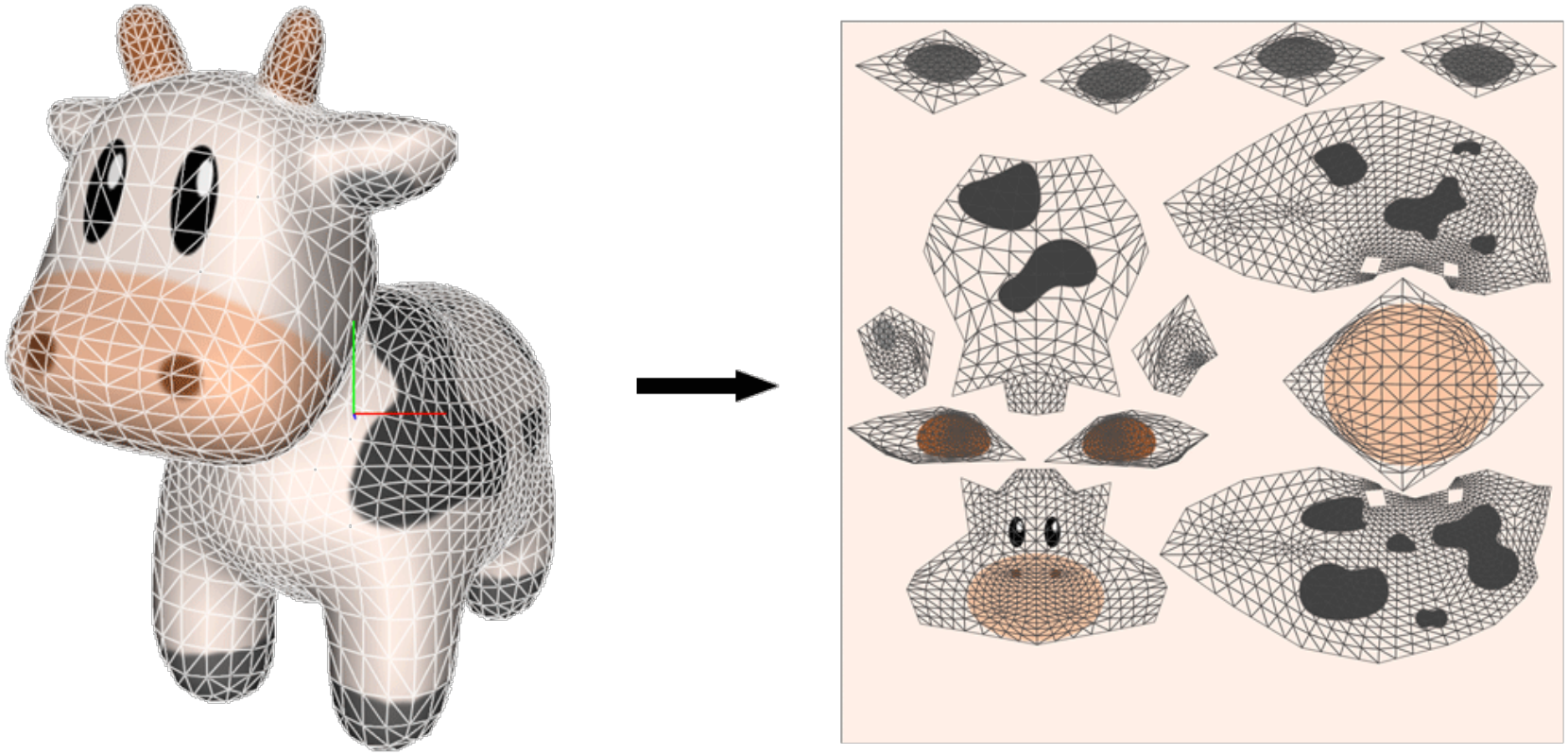
- Surface given by parametric functions

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

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



Skin Mapping

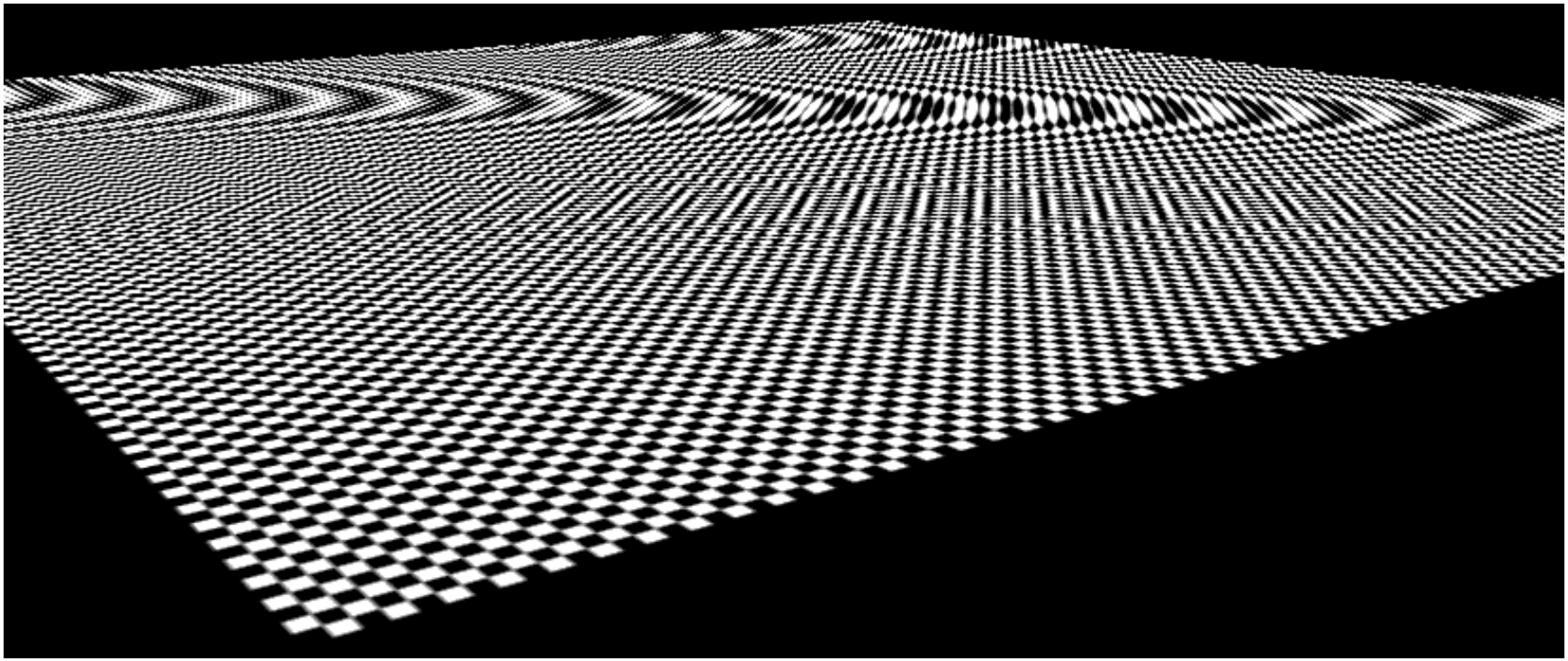


Lecture Overview

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

Aliasing

- What could cause this aliasing effect?



Aliasing

Sufficiently sampled, no aliasing

Insufficiently sampled, aliasing

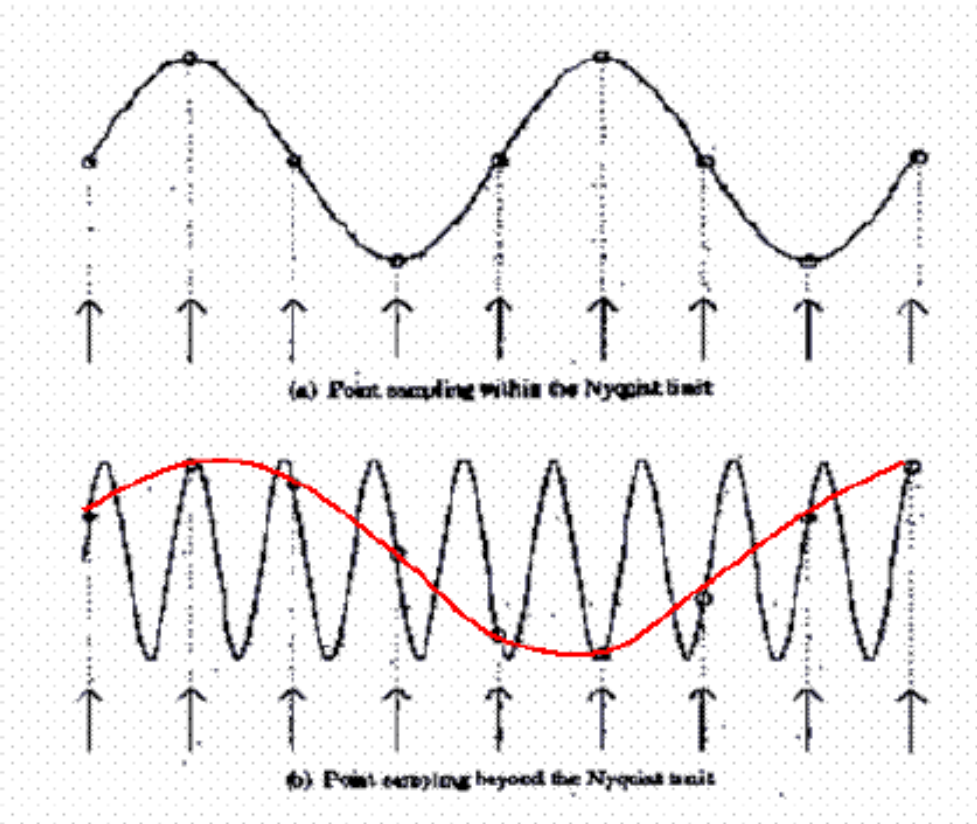
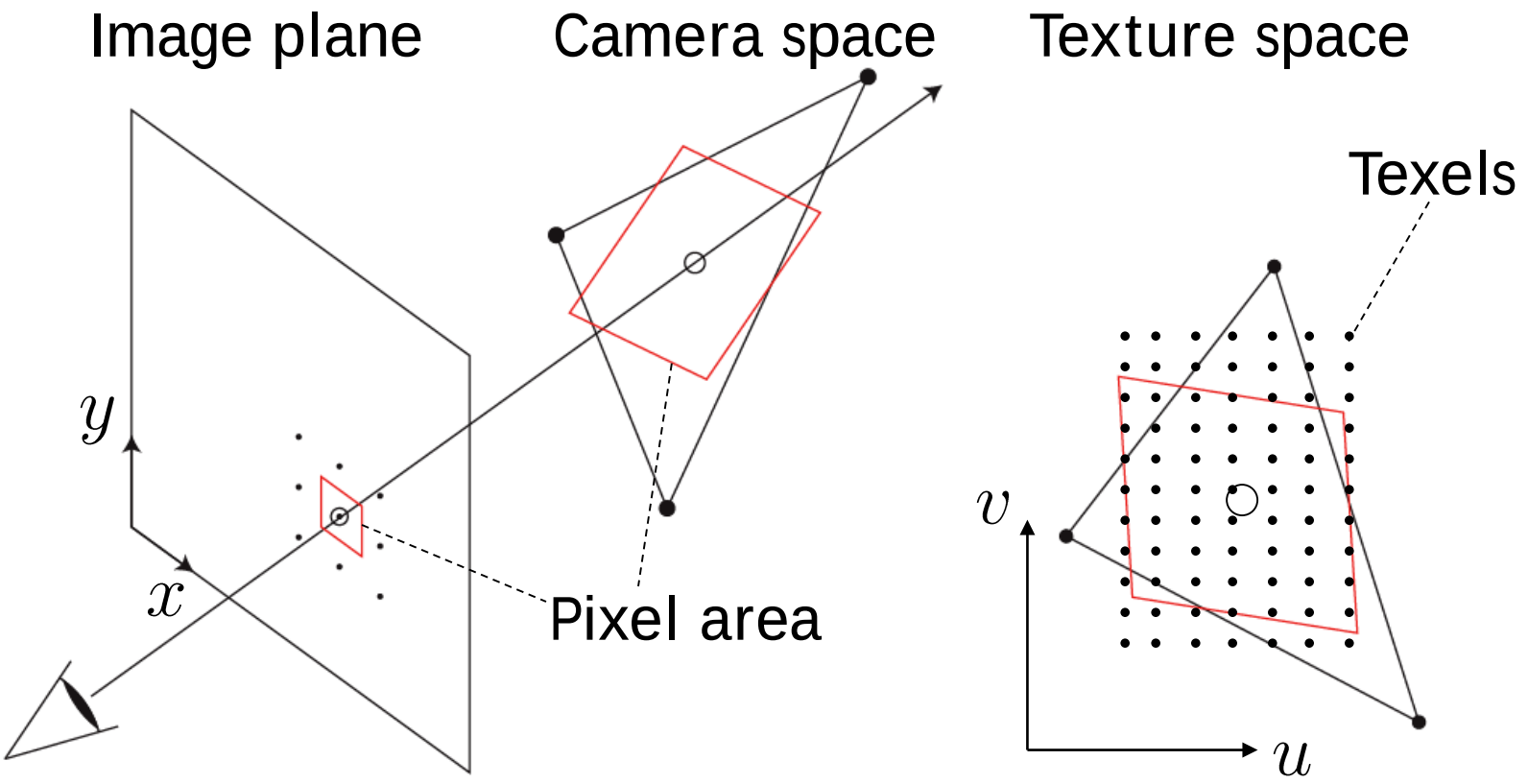


Image: Robert L. Cook

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

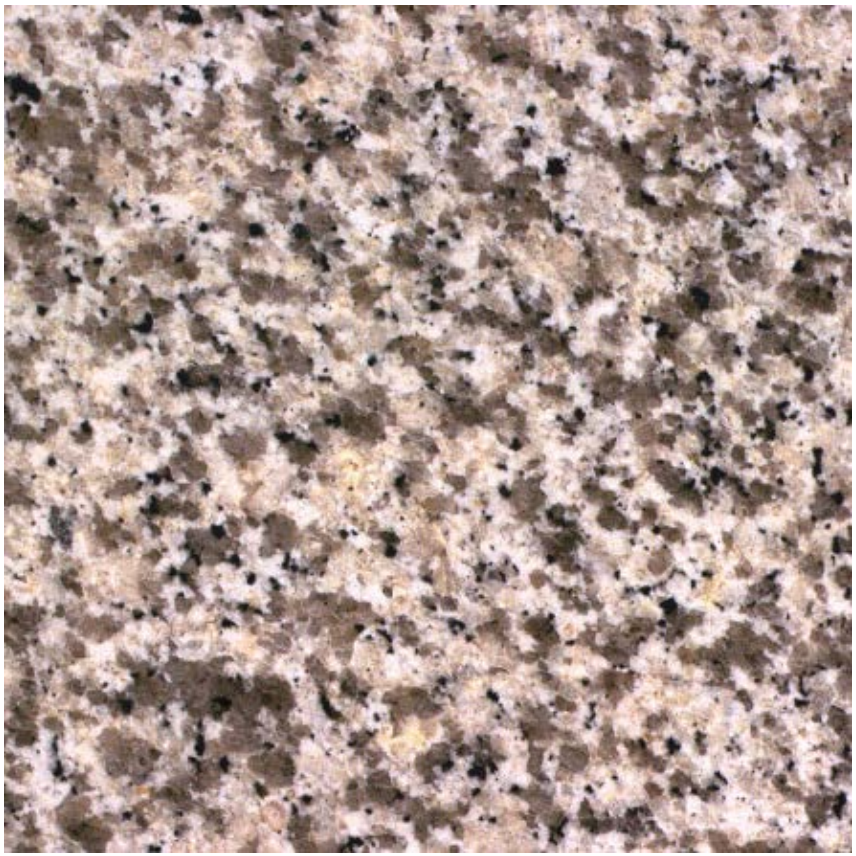
- MIP stands for *multum in parvo* = *"much in little"* (Williams 1983)

Before rendering

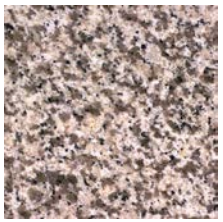
- Pre-compute and store down scaled versions of textures
  - Reduce resolution by factors of two successively
  - Use high quality filtering (averaging) scheme
- Increases memory cost by 1/3
  - 1/3 = ¼+1/16+1/64+…
- Width and height of texture should be powers of two (non-power of two supported since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

Mipmaps

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



Level 1



2

3

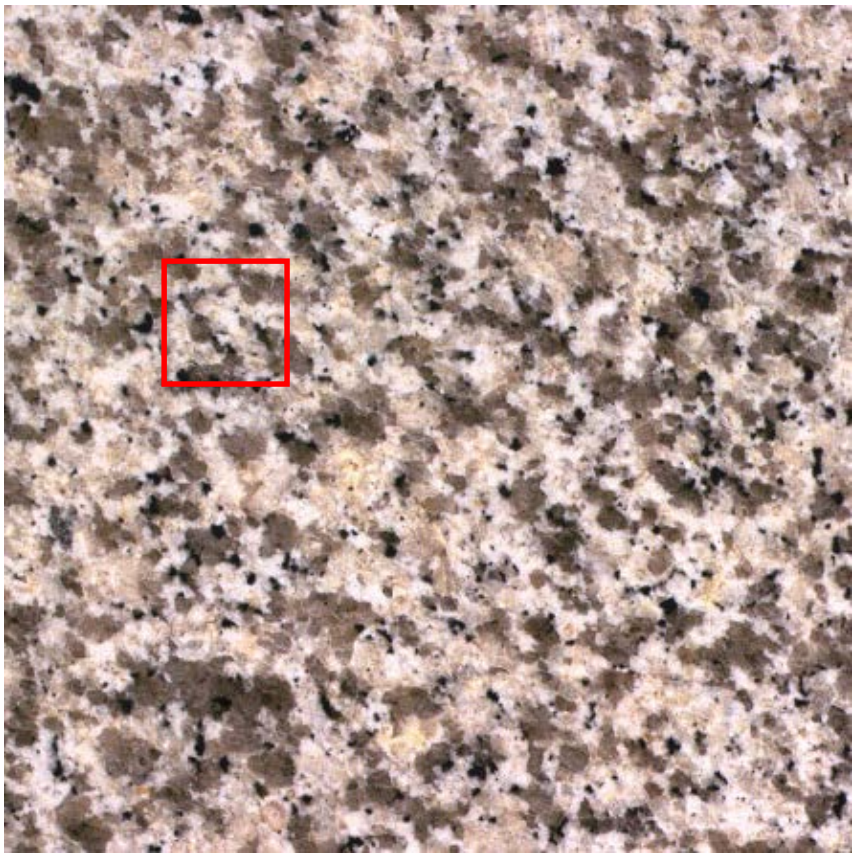
4

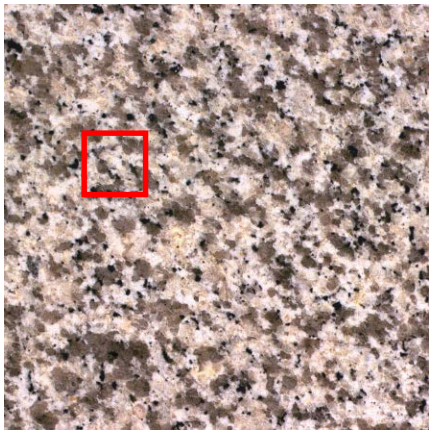
"multum in parvo"

Level 0

Mipmaps

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





Level 1

4

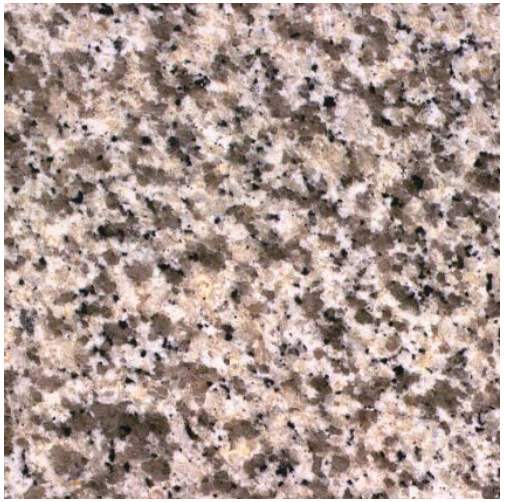
3

2

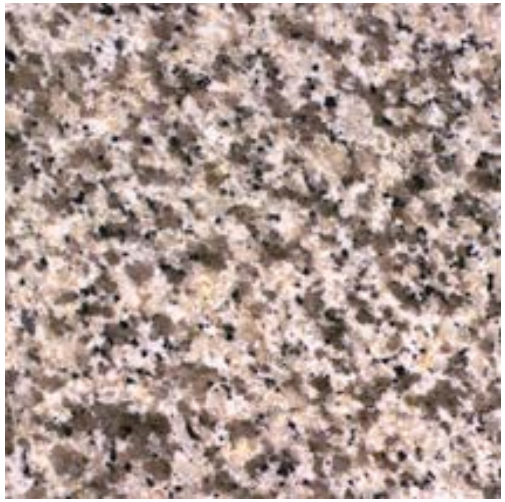
"multum in parvo"

Level 0

Mipmaps



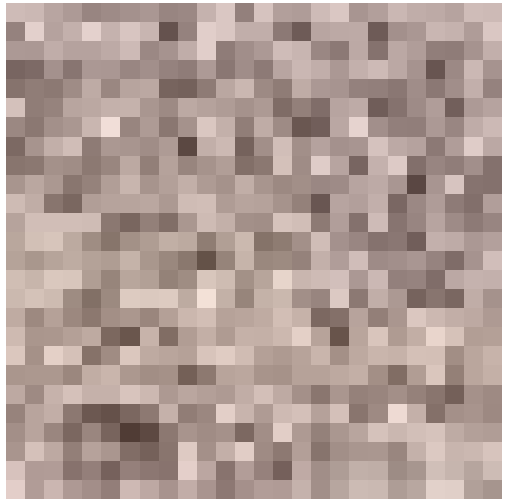
Level 0



Level 1

Level 2

Level 3

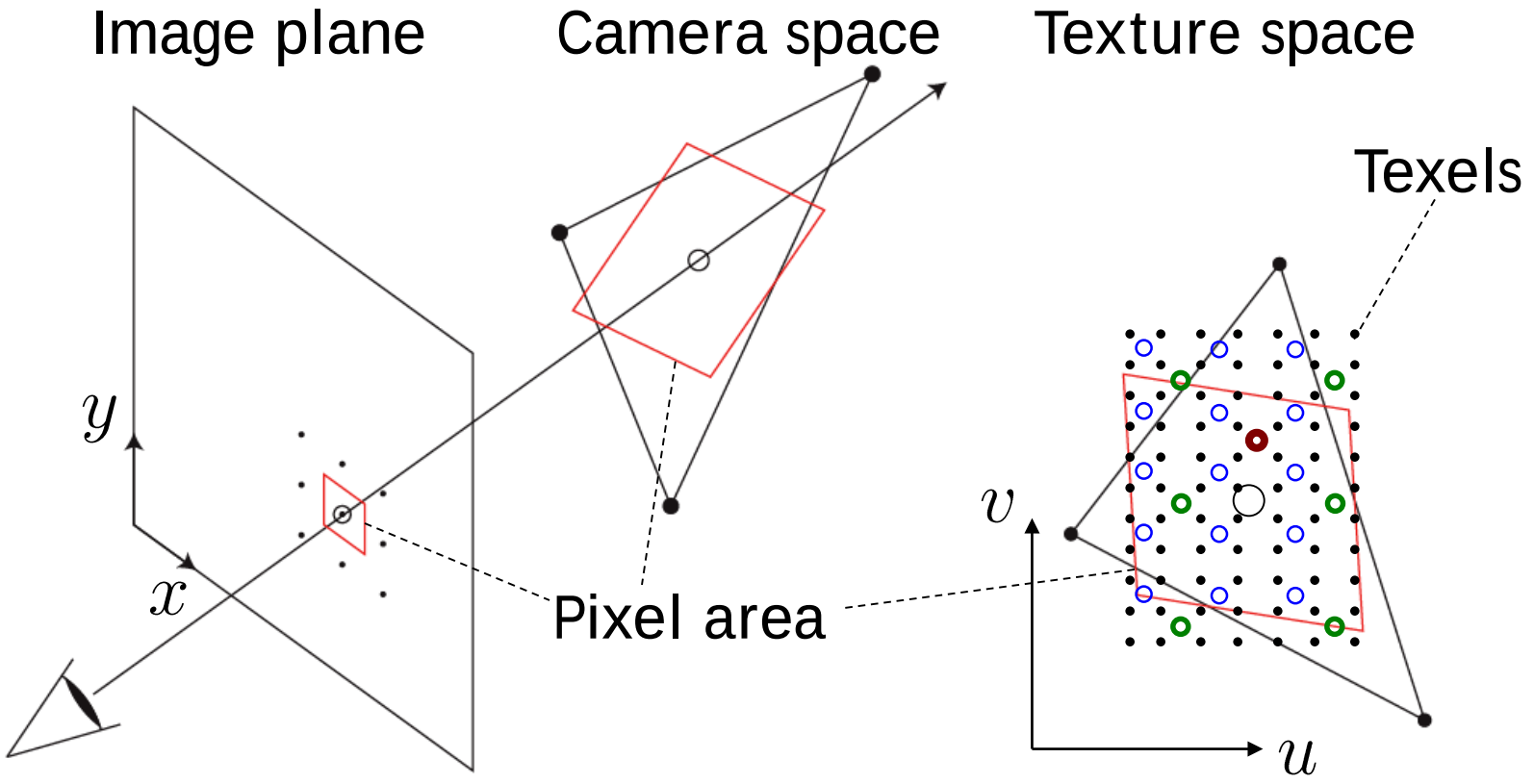


Level 4

Rendering With Mipmaps

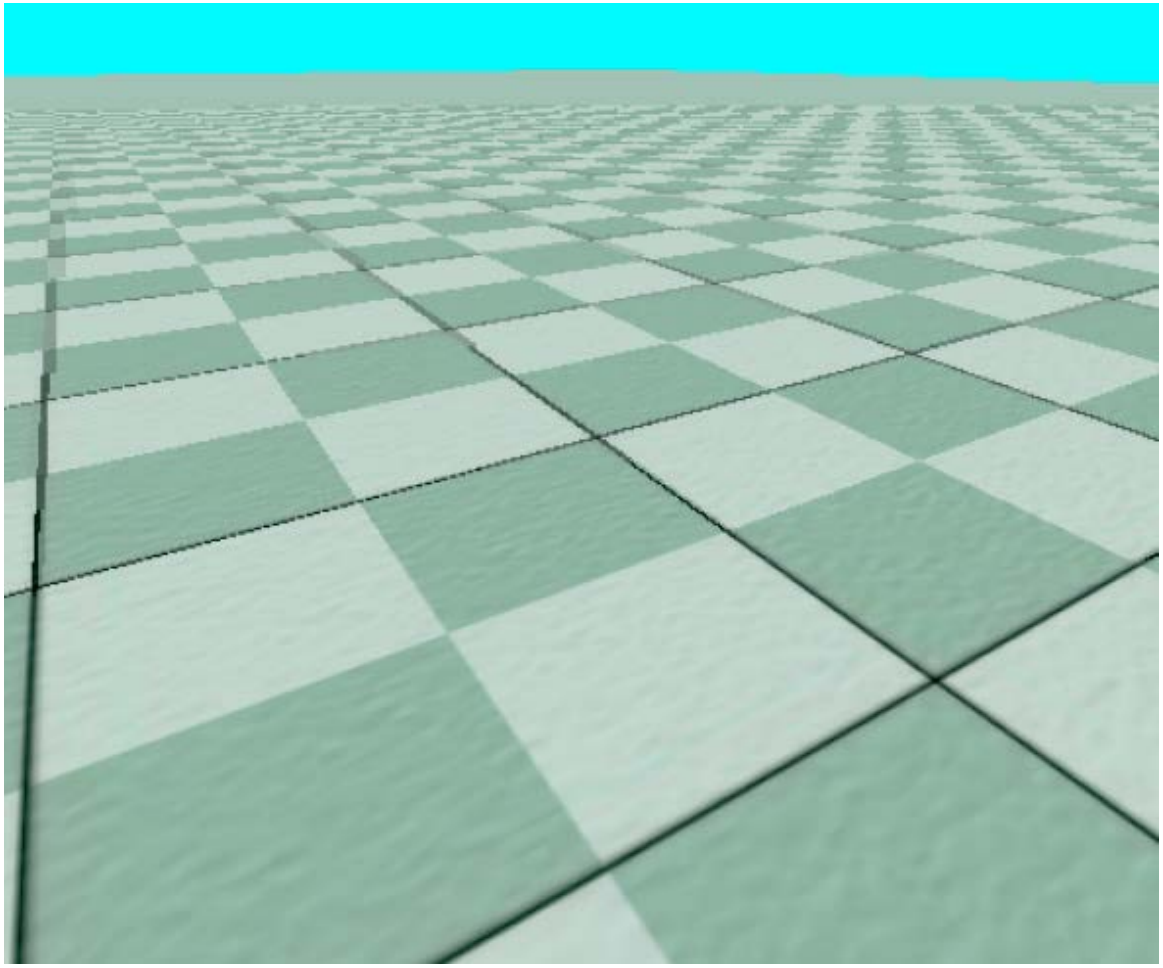
- "Mipmapping"
- Interpolate texture coordinates of each pixel as without mipmapping
- Compute approximate size of pixel in texture space
- Look up color in nearest mipmap
  - E.g., if pixel corresponds to 10x10 texels use mipmap level 3
  - Use nearest neighbor or bilinear interpolation as before

Mipmapping



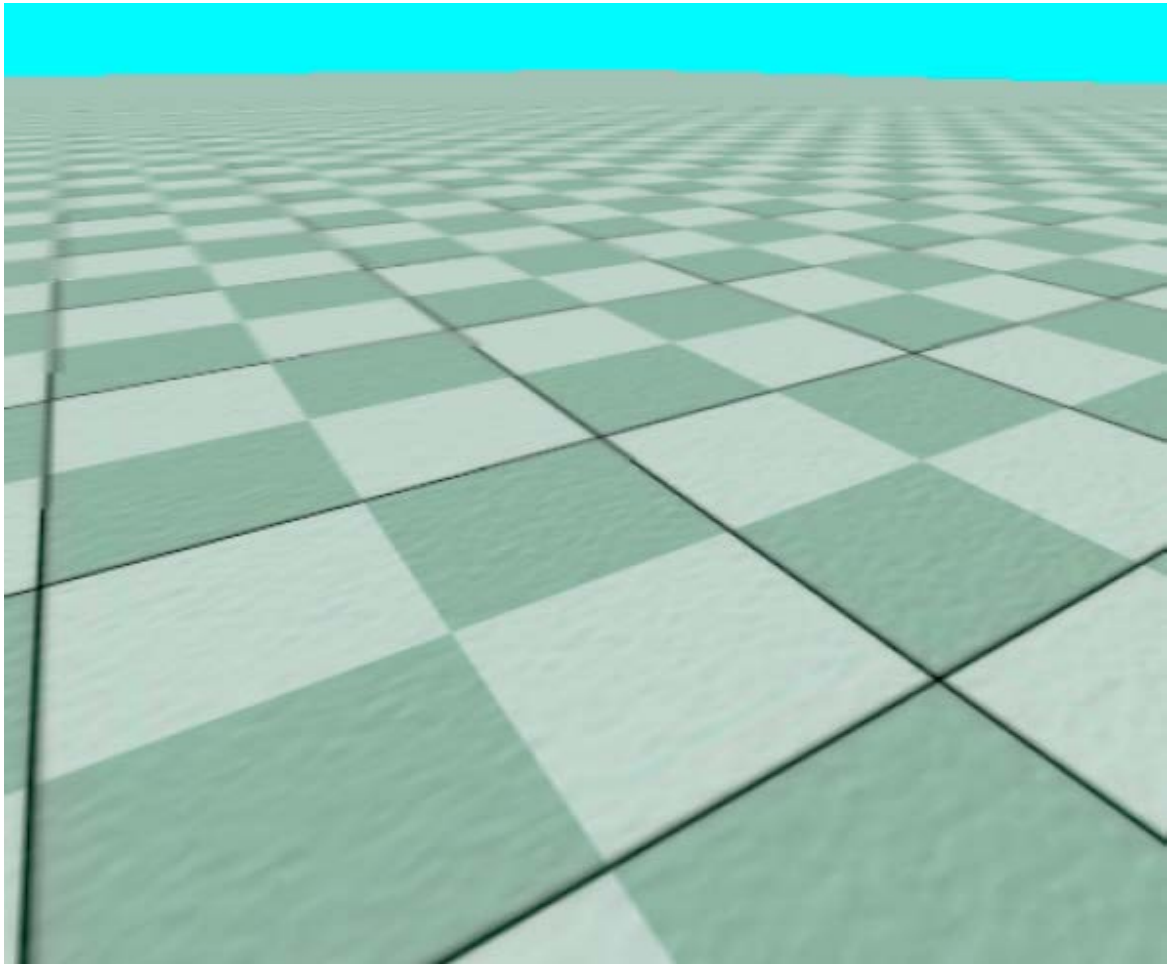
Nearest Mipmap, Nearest Neighbor

- 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

OpenGL Example: Loading a Texture

```
// Loads image as texture, returns ID of texture
GLuint loadTexture(Image* image)
{
  GLuint textureId;

  glGenTextures(1, &textureId); // Get unique ID for texture
  glBindTexture(GL_TEXTURE_2D, textureId); // Tell OpenGL which texture to edit
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR); // set bi-linear interpolation
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR); // for both filtering modes
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE); // set texture edge mode
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

  Image* image = loadJPG("photo.jpg");  // load image from disk; uses third party Image library

  // Depending on the image library, the texture image may have to be flipped vertically

  // Load image into OpenGL texture in GPU memory:
  glTexImage2D(GL_TEXTURE_2D,      // Always GL_TEXTURE_2D for image textures
     0,                            // 0 for now
     GL_RGB,                       // Format OpenGL uses for image without alpha channel
     image->width, image->height,  // Width and height
     0,                            // The border of the image
     GL_RGB,           // GL_RGB, because pixels are stored in RGB format
     GL_UNSIGNED_BYTE, // GL_UNSIGNED_BYTE, because pixels are stored as unsigned numbers
     image->pixels);               // The actual RGB image data

  return textureId; // Return the ID of the texture
}
```

Vertex Shader

```
#version 150

in vec3 vert;
in vec2 vertTexCoord;
out vec2 fragTexCoord;

void main()
{
  // Pass the tex coord straight through to the fragment shader
  fragTexCoord = vertTexCoord;

  gl_Position = vec4(vert, 1);
}
```

Fragment Shader

```
#version 150

uniform sampler2D tex; // this is the texture
in vec2 fragTexCoord;  // these are the texture coordinates
out vec4 finalColor;   // this is the output color of the pixel

void main()
{
  finalColor = texture(tex, fragTexCoord);
}
```