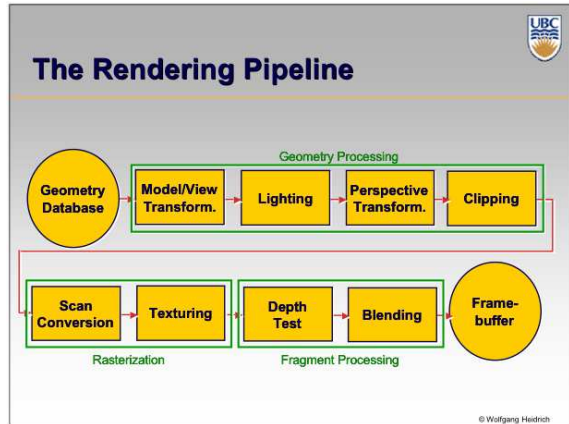


UBC

# Sampling & Reconstruction

## CPSC 314

© Wolfgang Heidrich



UBC

# Scan Conversion of Lines - Digital Differential Analyzer

**First Attempt:**

```

dda( float xs, ys, xe, ye ) {
    // assume xs < xe, and slope m between 0 and 1
    float m= (ye-ys)/(xe-xs);
    float y= round( ys );
    for( int x= round( xs ); x<= xe ; x++ ) {
        drawPixel( x, round( y ) );
        y= y+m;
    }
}
  
```

© Wolfgang Heidrich

UBC

# Texture Mapping

- Real life objects have nonuniform colors, normals
- To generate realistic objects, reproduce coloring & normal variations = **texture**
- Can often replace complex geometric details

© Wolfgang Heidrich

UBC

# Color Texture Mapping

**Define color (RGB) for each point on the surface**

**Two approaches**

- Surface texture map
- Volumetric texture

© Wolfgang Heidrich

UBC

# Texture Coordinates

**Texture Image: 2D array of color values (texels)**

**Assigning texture coordinates (s,t) at vertex with object coordinates (x,y,z,w)**

- Use interpolated (s,t) for texel lookup at each pixel
- Use value to modify a polygon's color
- Or other surface property
- Specified by programmer or artist

```

glTexCoord2f( s, t );
glVertexf( x, y, z, w );
  
```

© Wolfgang Heidrich

## Texture Mapping

The diagram illustrates the process of texture mapping. The top part shows a 3D hand model combined with a 2D texture of vertical yellow and black stripes to produce a striped hand. The bottom part shows a wireframe hand model with a grid of texture coordinates overlaid, and a vertical strip of the texture being mapped onto it.

## Reconstruction

- How to deal with:
  - Pixels that are much larger than texels?*
    - Apply filtering, "averaging"
  - Pixels that are much smaller than texels?*
    - Interpolate

© Wolfgang Heidrich

## Interpolating Textures

- Nearest neighbor
- Bilinear
- Hermite

The diagram shows a 3x3 grid of black and white squares. To its right are two grayscale images: the first is a sharp transition between black and white, and the second is a blurred version of the same transition, representing the result of bilinear interpolation.

© Wolfgang Heidrich

## MIPmapping

use "image pyramid" to precompute averaged versions of the texture

The image pyramid diagram shows a series of texture levels: 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, and 2x2. A small icon of a folder indicates that the entire pyramid is stored in a single block of memory.

Without MIP-mapping

With MIP-mapping

© Wolfgang Heidrich

## MIPmaps

*Multum in parvo -- many things in a small place*

- Prespecify a series of prefiltered texture maps of decreasing resolutions
- Requires more texture storage
- Avoid shimmering and flashing as objects move

**gluBuild2DMipmaps**

- Automatically constructs a family of textures from original texture size down to 1x1

without with

© Wolfgang Heidrich

## MIPmap storage

**only 1/3 more space required**

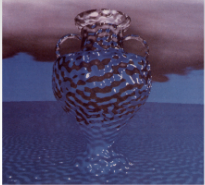
The diagram shows a 3x3 grid of colored spheres (red, blue, green) illustrating the storage of multiple MIPmap levels for each texel.

© Wolfgang Heidrich

**Texture Parameters**

*In addition to color can control other material/object properties*

- Surface normal (bump mapping)
- Reflected color (environment mapping)



© Wolfgang Heidrich

**Sampling & Reconstruction**

**CPSC 314**

© Wolfgang Heidrich

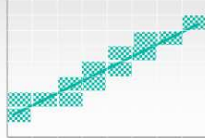
**Samples**

- Most things in the real world are **continuous**
- Everything in a computer is **discrete**
- The process of mapping a continuous function to a discrete one is called **sampling**
- The process of mapping a discrete function to a continuous one is called **reconstruction**
- The process of mapping a continuous variable to a discrete one is called **quantization**
- Rendering an image requires sampling and quantization
- Displaying an image involves reconstruction

© Wolfgang Heidrich

**Line Segments**

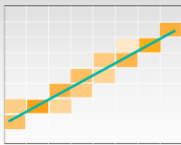
- We tried to sample a line segment so it would map to a 2D raster display
- We quantized the pixel values to 0 or 1
- We saw stair steps, or jaggies



© Wolfgang Heidrich

**Line Segments**

- Instead, quantize to many shades
- But what sampling algorithm is used?



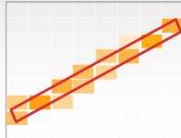
© Wolfgang Heidrich

**Unweighted Area Sampling**

*Shade pixels wrt area covered by thickened line*  
*Equal areas cause equal intensity, regardless of distance from pixel center to area*

- Rough approximation formulated by dividing each pixel into a finer grid of pixels

*Primitive cannot affect intensity of pixel if it does not intersect the pixel*



© Wolfgang Heidrich

**Weighted Area Sampling**

*Intuitively, pixel cut through the center should be more heavily weighted than one cut along corner*

**Weighting function,  $W(x,y)$**

- Specifies the contribution of primitive passing through the point  $(x, y)$  from pixel center

© Wolfgang Heidrich

**Images**

**An image is a 2D function  $I(x, y)$**

- Specifies intensity for each point  $(x, y)$
- (we consider each color channel independently)

An image seen as a continuous 2D function

© Wolfgang Heidrich

**Image Sampling and Reconstruction**

- Convert **continuous** image to **discrete** set of samples
- Display hardware **reconstructs** samples into continuous image
  - Finite sized source of light for each pixel

© Wolfgang Heidrich

**Point Sampling an Image**

- Simplest sampling is on a grid
- Sample depends solely on value at grid points

Sampling grid maps continuous to discrete

© Wolfgang Heidrich

**Point Sampling**

**Multiply sample grid by image intensity to obtain a discrete set of points, or samples.**

© Wolfgang Heidrich

**Sampling Errors**

**Some objects missed entirely, others poorly sampled**

- Could try unweighted or weighted area sampling
- But how can we be sure we show everything?

**Need to think about entire class of solutions!**

© Wolfgang Heidrich

**Image As Signal**

**Image as spatial signal**  
**2D raster image**

- Discrete sampling of 2D spatial signal

**1D slice of raster image**

- Discrete sampling of 1D spatial signal

Original signal

Pixel position across scanline

Examples from Foley, van Dam, Feiner, and Hughes  
 © Wolfgang Heidrich

**Sampling Theory**

**How would we generate a signal like this out of simple building blocks?**

**Theorem**

- Any signal can be represented as an (infinite) sum of sine waves at different frequencies

© Wolfgang Heidrich

**Sampling Theory in a Nutshell**

**Terminology**

- Wavelength – length of repeated sequence on infinite signal
- Frequency – 1/wavelength (number of repeated sequences in unit length)

**Example – sine wave**

- Wavelength =  $2\pi$
- Frequency =  $1/2\pi$

© Wolfgang Heidrich

**Summing Waves I**

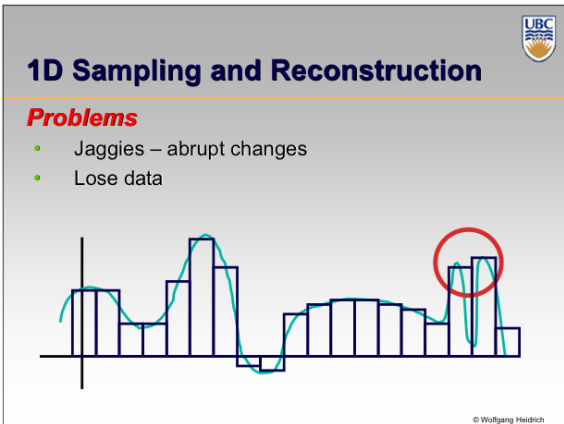
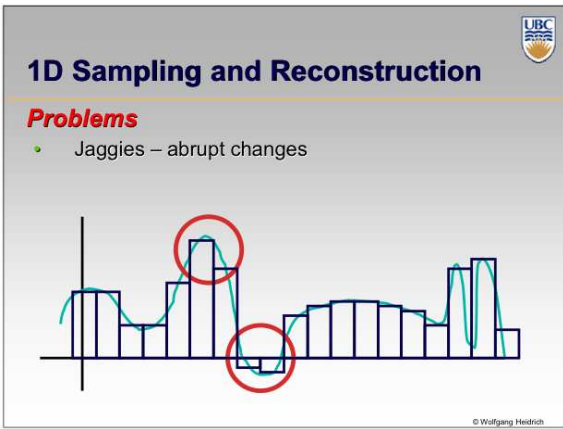
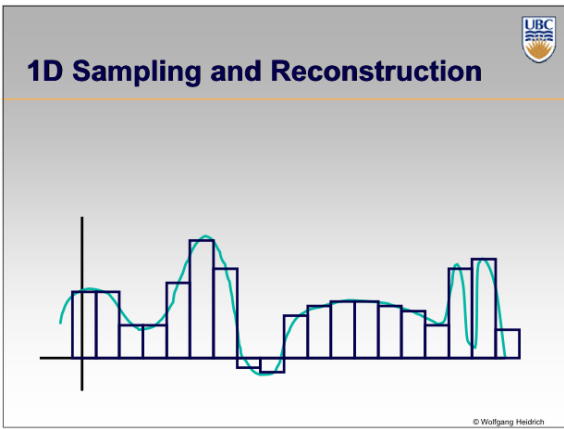
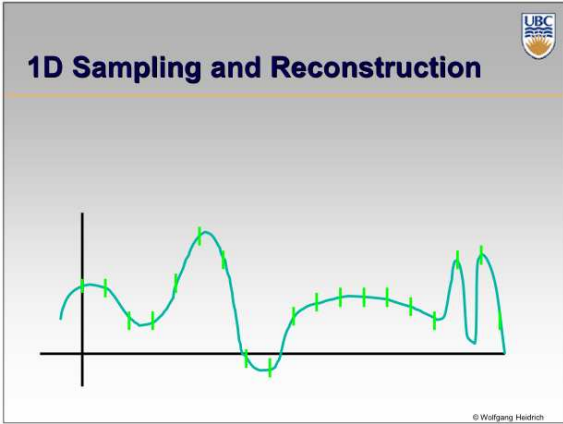
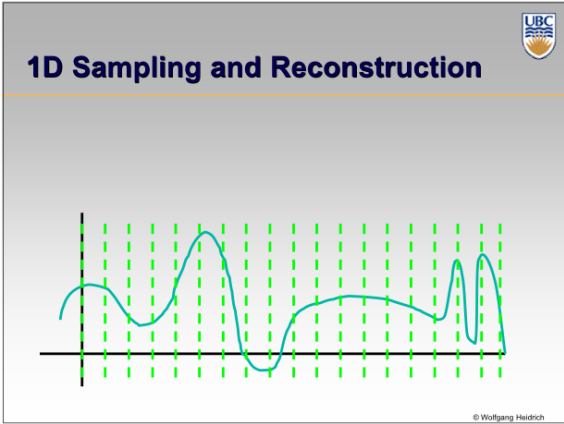
© Wolfgang Heidrich

**Summing Waves II**

© Wolfgang Heidrich

**1D Sampling and Reconstruction**

© Wolfgang Heidrich



### Sampling Theorem

- Continuous signal can be completely recovered from its samples

**Iff**

- Sampling rate greater than twice highest frequency present in signal

**- Claude Shannon**

© Wolfgang Heidrich

**Nyquist Rate**

**Lower bound on sampling rate**

- Twice the highest frequency component in the image's spectrum

(a)

© Wolfgang Heidrich

**Falling Below Nyquist Rate**

**When sampling below Nyquist Rate, resulting signal looks like a lower-frequency one**

- This is **aliasing!**

Fig. 14.17 Sampling below the Nyquist rate. (Courtesy of George Wolberg, Columbia University.)

© Wolfgang Heidrich

**Nyquist Rate**

$f_s < 2f$

$f_s = 2f$

$f_s > 2f$

© Wolfgang Heidrich

**Aliasing**

**Incorrect appearance of high frequencies as low frequencies**

**To avoid: anti-aliasing**

- Supersample
  - Sample at higher frequency
- Low pass filtering
  - Remove high frequency function parts
  - Aka prefiltering, band-limiting

© Wolfgang Heidrich

**Supersampling**

No antialiasing

3x3 supersampling  
3x3 unweighted filter

© Wolfgang Heidrich


**Low-Pass Filtering**

Original signal

Low-pass filtering

Low-pass filtered signal

© Wolfgang Heidrich



## Low-Pass Filtering

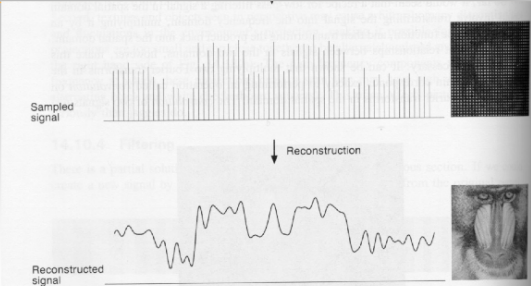

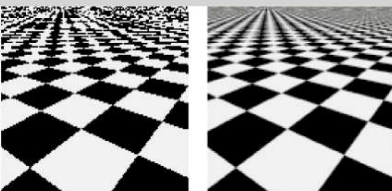


Fig. 14.20 The sampling pipeline with filtering. (Courtesy of George Wolberg, Columbia University.)




## Previous Antialiasing Example

**Texture mipmapping: low pass filter**



(a)                      (b)

© Wolfgang Heidrich




## Discussion

### Sampling & Reconstruction

- Fundamental issue in graphics, vision, and many other areas of computer science
  - Whenever continuous signals need to be represented in a computer
- Aliasing refers to the problem of reconstruction errors due to frequencies above the Nyquist limit
  - These frequencies show up as erroneous low frequency content

© Wolfgang Heidrich



## Discussion

### Anti-Aliasing Approaches

- Low-pass filtering (**before** sampling!)
  - Avoids aliasing
  - May not be practical in all settings
  - For images: artifacts around edges?!
- Supersampling
  - General algorithmic approach
  - However: even the higher resolution image has a Nyquist limit!
  - Slow

© Wolfgang Heidrich



## Coming Up...

**Tuesday:**

- Modern GPU Features

**Thursday:**

- Shadow Algorithms

© Wolfgang Heidrich