Sampling & Reconstruction

**CPSC 314**

The Rendering Pipeline

- **Geometry Database**
- **Model/View Transform.**
- **Lighting**
- **Perspective Transform.**
- **Clipping**

- **Scan Conversion**
- **Texturing**
- **Depth Test**
- **Blending**

- **Framebuffer**

*Geometry Processing*

*Rasterization*

*Fragment Processing*
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;
    }
}

Textured Mapping

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

Define color (RGB) for each point on the surface

Two approaches
- Surface texture map
- Volumetric texture

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

\[
g1TexCoord2f(s,t) \\
g1Vertexf(x,y,z,w)
\]
Texture Mapping

Reconstruction

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

- Nearest neighbor
- Bilinear
- Hermite

MIPmapping

use “image pyramid” to precompute averaged versions of the texture

store whole pyramid in single block of memory

Without MIP-mapping

With MIP-mapping
**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

---

**MIPmap storage**

*only 1/3 more space required*
Texture Parameters

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

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

Sampling & Reconstruction

**CPSC 314**
**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

**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
Line Segments

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

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*
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

Images

An image is a 2D function $I(x, y)$
- Specifies intensity for each point $(x, y)$
- (we consider each color channel independently)
**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*

![Diagram showing conversion from discrete input values to continuous light output](image)

**Point Sampling an Image**

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

![Sampling grid maps continuous to discrete](image)
Point Sampling

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

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!
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

---

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
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\(\pi\)

\[ \sin(t) \]

Summing Waves I
Summing Waves II

1D Sampling and Reconstruction
1D Sampling and Reconstruction

Problems

- Jaggies – abrupt changes
1D Sampling and Reconstruction

Problems

• Jaggies – abrupt changes
• Lose data

Sampling Theorem

• Continuous signal can be completely recovered from its samples

IFF

• Sampling rate greater than twice highest frequency present in signal

- Claude Shannon
Nyquist Rate

Lower bound on sampling rate

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

![Nyquist Diagram](image1.png)

Falling Below Nyquist Rate

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

- This is aliasing!

![Aliasing Diagram](image2.png)
Nyquist Rate

- $f_s < 2f$
- $f_s = 2f$
- $f_s > 2f$

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
Supersampling

No antialiasing

3x3 supersampling
3x3 unweighted filter

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
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

**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
Coming Up...

**Tuesday:**
- Modern GPU Features

**Thursday:**
- Shadow Algorithms