Lecture 7:
Rasterization 2
(Z-Buffering and Antialiasing)
Last Week

- Viewing
  - View + Projection + Viewport

- Rasterizing triangles
  - Point-in-triangle test
  - Aliasing
Today

- Visibility / occlusion
  - Z-buffering

- Antialiasing
  - Sampling theory
  - Antialiasing in practice
Visibility / Occlusion
Painter’s Algorithm

Inspired by how painters paint
Paint from back to front, overwrite in the framebuffer

[Wikipedia]
Painter’s Algorithm

Requires sorting in depth (O(n log n) for n triangles)
Can have unresolvable depth order
Z-Buffer

This is the algorithm that eventually won.

Idea:

- Store current min. z-value for each sample (pixel)
- Needs an additional buffer for depth values
  - framebuffer stores color values
  - depth buffer (z-buffer) stores depth

IMPORTANT: For simplicity we suppose
z is always positive
(smaller z -> closer, larger z -> further)
Z-Buffer Example

Rendering

Depth / Z buffer

Image source: Dominic Alves, flickr.
Z-Buffer Algorithm

Initialize depth buffer to $\infty$

During rasterization:

for (each triangle $T$)

    for (each sample $(x,y,z)$ in $T$)

        if ($z < z\text{buffer}[x,y]$) // closest sample so far

            framebuffer[$x,y$] = rgb; // update color

            $z\text{buffer}[x,y] = z$; // update depth

        else

            ; // do nothing, this sample is occluded
Z-Buffer Algorithm

Slide courtesy of Prof. Ren Ng, UC Berkeley
Z-Buffer Complexity

Complexity

- $O(n)$ for $n$ triangles
- How can we sort $n$ triangles in linear time?

Most important visibility algorithm

- Implemented in hardware for all GPUs
Today

- Visibility / occlusion
  - Z-buffering

- Antialiasing
  - Sampling theory
  - Antialiasing in practice
Recap: Testing in/out △ at pixels’ centers
Pixels are uniformly-colored squares
Compare: The Continuous Triangle Function
What’s Wrong With This Picture?

Jaggies!

Slide courtesy of Prof. Ren Ng, UC Berkeley
Aliasing

Is this the best we can do?
Sampling is Ubiquitous in Computer Graphics
Rasterization = Sample 2D Positions
Photograph = Sample Image Sensor Plane
Video = Sample Time
Sampling Artifacts (Errors / Mistakes / Inaccuracies) in Computer Graphics
Jaggies (Staircase Pattern)

This is also an example of “aliasing” – a sampling error
Moiré Patterns in Imaging

[mwa:]
Wagon Wheel Illusion (False Motion)
Sampling Artifacts in Computer Graphics

Artifacts due to sampling - “Aliasing”

- Jaggies – sampling in space
- Moire – undersampling images
- Wagon wheel effect – sampling in time
- [Many more] …

Behind the Aliasing Artifacts

- Signals are changing too fast (high frequency), but sampled too slowly
Antialiasing Idea: Blurring (Pre-Filtering) Before Sampling
Rasterization: Point Sampling in Space

Note jaggies in rasterized triangle where pixel values are pure red or white
Rasterization: Antialiased Sampling

Pre-Filter
(remove frequencies above Nyquist) (?)

Sample

Note antialiased edges in rasterized triangle where pixel values take intermediate values
Point Sampling
Antialiasing
Point Sampling vs Antialiasing

Slide courtesy of Prof. Ren Ng, UC Berkeley
Antialiasing vs Blurred Aliasing

(Sample then filter, WRONG!)  (Filter then sample)

Slide courtesy of Prof. Ren Ng, UC Berkeley
But why?

1. Why undersampling introduces aliasing?
2. Why pre-filtering then sampling can do antialiasing?

Let’s dig into fundamental reasons
And look at how to implement antialiased rasterization
Frequency Space
Sines and Cosines

\[ \cos 2\pi x \]

\[ \sin 2\pi x \]

Slide courtesy of Prof. Ren Ng, UC Berkeley
Frequencies \( \cos 2\pi f x \)

\[ f = \frac{1}{T} \]

\( f = 1 \)

\( f = 2 \)

\( \cos 4\pi x \)

Slide courtesy of Prof. Ren Ng, UC Berkeley
Fourier Transform

Represent a function as a weighted sum of sines and cosines

\[
f(x) = \frac{A}{2} + \frac{2A \cos(t\omega)}{\pi} - \frac{2A \cos(3t\omega)}{3\pi} + \frac{2A \cos(5t\omega)}{5\pi} - \frac{2A \cos(7t\omega)}{7\pi} + \ldots
\]

Joseph Fourier 1768 - 1830

Slide courtesy of Prof. Ren Ng, UC Berkeley
Fourier Transform Decomposes A Signal Into Frequencies

\[ f(x) \quad F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} \, dx \quad F(\omega) \]

**Fourier transform**

**Inverse transform**

\[ f(x) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i \omega x} \, d\omega \]

Recall \( e^{ix} = \cos x + i \sin x \)

Slide courtesy of Prof. Ren Ng, UC Berkeley
Higher Frequencies Need Faster Sampling

Periodic sampling locations

$f_1(x)$

$f_2(x)$

$f_3(x)$

$f_4(x)$

$f_5(x)$

Low-frequency signal: sampled adequately for reasonable reconstruction

High-frequency signal is insufficiently sampled: reconstruction incorrectly appears to be from a low frequency signal
Undersampling Creates Frequency Aliases

High-frequency signal is insufficiently sampled: samples erroneously appear to be from a low-frequency signal.

Two frequencies that are indistinguishable at a given sampling rate are called “aliases.”
Filtering = Getting rid of certain frequency contents
Visualizing Image Frequency Content

Slide courtesy of Prof. Ren Ng, UC Berkeley
Filter Out Low Frequencies Only (Edges)

High-pass filter

Slide courtesy of Prof. Ren Ng, UC Berkeley
Filter Out High Frequencies (Blur)

Low-pass filter

Slide courtesy of Prof. Ren Ng, UC Berkeley
Filter Out Low and High Frequencies
Filter Out Low and High Frequencies
Filtering = Convolution
(= Averaging)
Convolution

Signal

Filter

Point-wise local averaging in a “sliding window”
Convolution

<table>
<thead>
<tr>
<th>Signal</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>3</th>
<th>7</th>
<th>1</th>
<th>3</th>
<th>8</th>
<th>6</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>1/4</td>
<td>1/2</td>
<td>1/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 x (1/4) + 3 x (1/2) + 5 x (1/4) = 3

<table>
<thead>
<tr>
<th>Result</th>
<th>3</th>
</tr>
</thead>
</table>
Convolution

Signal

Filter

Result

3 x (1/4) + 5 x (1/2) + 3 x (1/4) = 4
Convolution Theorem

Convolution in the spatial domain is equal to multiplication in the frequency domain, and vice versa

Option 1:
• Filter by convolution in the spatial domain

Option 2:
• Transform to frequency domain (Fourier transform)
• Multiply by Fourier transform of convolution kernel
• Transform back to spatial domain (inverse Fourier)
Convolution Theorem

Spatial Domain

Fourier Transform

Frequency Domain

\[ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \]

\[ * \frac{1}{9} \]

Inv. Fourier Transform

Slide courtesy of Prof. Ren Ng, UC Berkeley
Box Filter

Example: 3x3 box filter

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
Box Function = “Low Pass” Filter
Wider Filter Kernel = Lower Frequencies
Sampling = Repeating Frequency Contents
Sampling = Repeating Frequency Contents


Slide courtesy of Prof. Ren Ng, UC Berkeley
Aliasing = Mixed Frequency Contents

Dense sampling:

Sparse sampling:

Slide courtesy of Prof. Ren Ng, UC Berkeley
Antialiasing
How Can We Reduce Aliasing Error?

Option 1: Increase sampling rate

• Essentially increasing the distance between replicas in the Fourier domain
• Higher resolution displays, sensors, framebuffers…
• But: costly & may need very high resolution

Option 2: Antialiasing

• Making Fourier contents “narrower” before repeating
• i.e. Filtering out high frequencies before sampling
Antialiasing = Limiting, then repeating

Filtering

Then sparse sampling
Regular Sampling

Note jaggies in rasterized triangle where pixel values are pure red or white
Antialiased Sampling

Pre-Filter
(remove frequencies above Nyquist)

Sample

Note antialiased edges in rasterized triangle where pixel values take intermediate values
Thank you!