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

- Viewing
  - View + Projection + Viewport

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

• Antialiasing
  - Sampling theory
  - Antialiasing in practice

• Visibility / occlusion
  - Z-buffering
Recap: Testing in/out $\triangle$ at pixels’ centers
Pixels are uniformly-colored squares

![Red square grid diagram]
Compare: The Continuous Triangle Function
What’s Wrong With This Picture?

Jaggies!
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

Harold Edgerton Archive, MIT
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

[mwəː]  

Skip odd rows and columns
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
• Changing not just in time but also in space
Antialiasing Idea:

averaging instead of sampling
Antialiasing By Averaging Values in Pixels

General observation:

• The value at a pixel’s center may not “represent” all values in a pixel

• But the average can

Solution:

• Compute the average value of inside(triangle,x,y) for all possible points (x, y) in a pixel
Antialiasing by Computing Average Pixel Value

In rasterizing one triangle, the average value inside a pixel area of $f(x,y) = \text{inside}(\text{triangle},x,y)$ is equal to the area of the pixel covered by the triangle.
Antialiasing By Averaging Values in Pixels

General observation:

• The value at a pixel’s center may not “represent” all values in a pixel
• But the average can

Solution:

• Compute the average value of inside(triangle,x,y) for all possible points (x, y) in a pixel
• But how?

Approximation

• Compute the average value of inside(triangle,x,y) for a few points (x, y) in a pixel
Antialiasing By Supersampling (MSAA)
Supersampling

Approximate the effect of the 1-pixel box filter by sampling multiple locations within a pixel and averaging their values:

4x4 supersampling
Point Sampling: One Sample Per Pixel
Supersampling: Step 1

Take $N \times N$ samples in each pixel.

2x2 supersampling
Supersampling: Step 2

Average the NxN samples “inside” each pixel.

Averaging down
Supersampling: Step 2

Average the NxN samples “inside” each pixel.
Supersampling: Step 2

Average the NxN samples “inside” each pixel.
Supersampling: Result

This is the corresponding signal emitted by the display
Point Sampling
4x4 Supersampling
Sampling Food for Thought
Off-Grid Sampling?
Random Sampling?
Use Samples “Outside” Pixel?
Non-Uniform Sample Weighting?
Antialiasing Today

No free lunch!

• What’s the cost of MSAA?

Milestones (personal idea)

• FXAA (Fast Approximate AA)
• TAA (Temporal AA)

Super resolution / super sampling

• From low resolution to high resolution
• Essentially still “not enough samples” problem
• DLSS (Deep Learning Super Sampling)
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
  - frame buffer 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 ∞

During rasterization:

for (each triangle T)
  for (each sample (x,y,z) in T)
    if (z < zbuffer[x,y]) // closest sample so far
      framebuffer[x,y] = rgb; // update color
      zbuffer[x,y] = z; // update depth
    else
      ; // do nothing, this sample is occluded
Z-Buffer Algorithm
Z-Buffer Complexity

Complexity
• $O(n)$ for $n$ triangles (assuming constant coverage)
• How is it possible to sort $n$ triangles in linear time?

Drawing triangles in different orders?

Most important visibility algorithm
• Implemented in hardware for all GPUs
Thank you!