Outline

- Drawing with Thick Primitives
- Halftone Approximation
- Anti-aliasing

Drawing with Thick Primitives

- How do we thicken the line stroke width?
- Ideas:
  - Place the center of the circular “brush” on the pixel
  - Place the upper corner of the square “marker” on the pixel (issues of orientation)
  - Then do scan conversion algorithm

Three Basic Methods

1. Column Replication
   - Use >1 pixel per col/row
2. Trace brush outline across 1-pixel primitive
3. Trace two copies, \( t \) apart, and fill in

Column(Row) Replication

- Idea: duplicate pixels in
  - Columns, when \(-1 < \text{slope} < 1\)
  - Rows, otherwise
- Thickness \( t \) is from primitive’s boundaries perpendicular to its tangent
- What happens for even \( t \)?
- Issues when lines meet at angles, when octants merge, brightness for sloped lines, etc.

Moving the Pen

- Example:
  - a rectangular pen
  - Center or corner follows scan algorithm
- How to implement?
  - Idea 1: fill the box at each point
  - Problem: pixels get colored more than once
  - Idea 2: fill by using a span of the pen primitive at each step
Halftone Approximation

- Not all devices can display all colors
  – e.g. GIF is only 256 colors
- Idea: With few available shades, produce illusion of many colors/shades?
- Technique: **Halftone Approximation**
- Example: How do we do greyscale with black-and-white monitors?

Halftone Approximation

- Technique: **Dithering**
  – Idea: create meta-pixels, grouping base pixels into 3x3s or 4x4s
  – Example: a 2x2 dither matrix for greyscale

Halftone Approximation

- Issues with Dithering
  – Image is now 4x in size
- How do we keep image the same size?
- Technique: **Error Diffusion**
  – Idea: When approximating pixel intensity, keep track of error and try to make up for errors with later pixels

Halftone Approximation: Error Diffusion Example #1

- Problem: draw 1D line with 1/3 gray tone
- Pixel #1: round to black, 0... error 1/3
- Pixel #2: value 1/3+1/3=2/3, color white
- Pixel #3: value 1/3+1/3=0, color black
- Pixel #4: value 1/3+0=1/3, color black
- Color sequence: 01001001001...

Halftone Approximation: Error Diffusion Example #2

- Consider a 2D image/primitive
  – Goals: Spread errors out in x and y pixels
  Nearby gets more error than far away
- Floyd-Steinberg Error Distribution Method
  – Let the current pixel be \((x,y)\)
  – Distribute error as follows:
    - **Right**: 7/16 of the error to \((x+1,y)\).
    - **Below left**: 3/16 of the error to \((x-1,y-1)\).
    - **Below**: 5/16 of the error to \((x,y-1)\).
    - **Below right**: 1/16 of the error to \((x+1,y-1)\).

Pics/Math courtesy of Dave Mount @ UMD-CP
Halftone Approximation: Error Diffusion Example #2

- Let $S[x][y]$ be the shade of pixel $(x, y)$
- To draw $S[x][y]$ we round pixel to nearest shade $K$ and set $err = S[x][y] - K$
- Then, diffuse the errors throughout surrounding pixels, e.g.
  
  $$
  S[x + 1][y] += (7/16) \times err
  S[x - 1][y - 1] += (3/16) \times err
  S[x][y - 1] += (5/16) \times err
  S[x + 1][y - 1] += (1/16) \times err
  $$

Antialiasing

- Converting an idealized line to a discrete grid is an approximation
- "Jaggies" are the result

The Aliasing Problem

- General problem in Analog-to-Digital conversion
  - When sampling, one needs to sample at a higher frequency than the analog signal
  - Aliasing shows up as spurious low frequencies

Zone plate

- Zone plate $f(x, y) = x^2 + y^2$
- Black if floor(f) odd, else white
- Outer rings occur too often to be sampled correctly
- Moiré patterns resemble the zone plate
Aliasing in Computer Graphics

- Mathematical model of image: analog
- Screen: digital
- Result: visual effects, jaggies, lost textures and detail

Antialiasing in use...

Antialiasing

- How to create the visual effect of smoothing the line?
- We need to find a way to simulate the display of partial pixels

Antialiasing: PreFiltering

- Idea:
  - each pixel has area
  - compute color based on overlap with the object's area
- Result: smoother borders for objects

Antialiasing: PostFiltering

- Filters: the weighted measure of color
  - weighted or unweighted average of the pixels

Antialiasing: PostFiltering (SuperSampling)

- Idea:
  - take multiple samples for each pixel
  - create weighted measure of color
  - samples can be stochastic or regular
  - stochastic makes for more pleasing results
Antialiasing of Lines & Curves

- Example: draw 1-pixel thick black line
  - 2 different screen resolutions ⇒
    - 2x jags, jags 1/2 size, but 4x memory!

Weighted Area Sampling: Box Filters

- Pixel intensity
  - $I_{\text{max}} = 1$, $I_{\text{min}} = 0$
- Box over the pixel: weighting function
- The volume $W_x$ is the weight
  - $W_x \cdot I_{\text{max}}$
- (achieves unweighted area sampling)

Antialiasing of Circles and Curves

- Super-sampling can be applied to anti-aliasing of circles and curves
- Double (triple, quadruple) the resolution of your buffer.
- Use any polygon filling algorithm to scan-fill the polygon in BW
- Compute the color of each pixel by counting the number of colored sub-pixels

Weighted Area Sampling: Cone Filters

- Idea:
  - Cone centered at each pixel
  - Radius of cone bigger than single pixel
  - Volume of cone is 1
- Intensity is volume $W_x$
- Achieves
  - Linear decrease of intensity vs distance
  - Smoother effects by decreasing pixel contrasts
  - Rotational symmetry

Antialiasing Area Boundaries

- Supersampling:
  - Double (triple, quadruple) the resolution of your buffer.
  - Use any polygon filling algorithm to scan-fill the polygon in BW
  - Compute the color of each pixel by counting the number of colored sub-pixels
Antialiasing with Grayscale

Antialiasing Example

Anti-aliasing from CG II

Steven Thomas

Copyright 1993-1997, Education Slide Set