

Vertex Complications

○ 1 intersection
● 2 intersections

What is the difference between these cases? 1

Vertex Intersection Types

Inside/outside change No change

2

Vertex Workaround

If monotonic, shorten one edge by one scan If not monotonic, leave edges alone

3

Horizontal Edges

Ignore horizontal segments (but don't fill over them?)

Check monotonicity of adjacent edges

4

Calculating Intersections

- Edge line equations?
 - Expensive
 - Function evaluated every scan line
- Coherence
 - Each scan line similar to previous
 - Intersections calculated incrementally

5

Incremental Calculation

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \quad y_{k+1} - y_k = 1$$

$$x_{k+1} - x_k = \frac{1}{m}$$

$$x_{k+1} = x_k + \frac{1}{m}$$

Similar to DDA algorithm?
Floating point calculation?

6

Integer Calculation

$$m = \frac{\Delta y}{\Delta x} \quad x_{k+1} = x_k + \frac{\Delta x}{\Delta y}$$

Accumulate Δx at each step
If sum $> \Delta y$, reduce modulo Δy and update x

$$x_{k+1}\Delta y = x_k\Delta y + \Delta x$$

7

Integer Calculation Example

$$m = \frac{\Delta y}{\Delta x} = -\frac{12}{9}$$

y	x_k	sum		x_{k+1}
0	3	-9	-9	3
1	3	-18	-6	2
2	2	-15	-3	1
3	1	-12	0	0
4	0			

To round instead of truncate, compare sum with $\Delta y/2$ or compare sum of $2\Delta x$ with Δy

Intersection Data Structure

See text, page 200

Y_c → y_e | x_c | Δx/Δy

Y_b → y_e | x_b | Δx/Δy

1

0

9

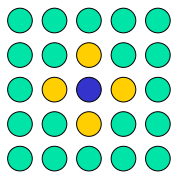
Using Intersection Data Structure

- Odd/even: sort by x and toggle
- Non-zero winding
 - Add an up/down (CCW/CW) data member to the edge data structure
 - Proceed as before
 - Including shortening and horizontals
 - But instead of toggling in/out
 - Start with winding number = 0 and add...
 - -1 for edge crossing upward
 - +1 for edge crossing downward

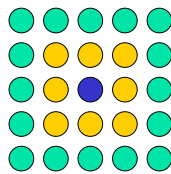
Boundary Fill

- Start at interior point
- Paint interior outward toward boundary
- Stop on boundary encounter
 - Determined by pixel color
 - Also stop recursion on "fill" color
- Can change fill pattern (with proper choice of colors)

Boundary Fill Patterns



4-connected



8-connected

Boundary Fill Algorithm

- Stop if current position is:
 - Boundary color
 - Fill color
- Otherwise
 - Set fill color
 - Recursively try neighbors
 - North, East, South, West (arbitrary)
 - Each neighbor recursively performs algorithm until "stop"

13


Boundary Fill Example N, E, S, W

14

Boundary Fill Problems

- Recursive algorithm
 - Stack space?
- Terminates on
 - Boundary color (good)
 - Fill color (may be a problem)
- What if some fill color present?
 - Pixels in area to be filled


15



Flood Fill

- Similar to boundary fill
- But replaces "interior" color
 - Floods through an area
 - Area initially consistent color
 - Only one significant color
- Paint bucket tool
 - In common drawing programs

16



Fill Algorithm Summary

- Based on vector edge model
 - Scan line fill
 - Add integer coherence
 - Even-odd or winding
- Based on pixel model
 - 4- or 8- connected
 - Recurses over
 - Boundary fill: is not a (boundary or new internal) color
 - Flood fill: is an old internal color

17
