

Handouts: Projections

Lines and Polygons

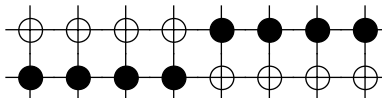
- And Assignment #2!

Antialiasing

- Aliasing caused by finite addressability of CRT
- Approximation of lines with discrete points can result in a staircase appearance or "Jaggies"
- Desired line



- Aliased rendering of the line

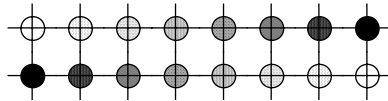


Handouts: Projections

Antialiasing - Solutions



- Aliasing can be smoothed out by using higher addressability.
- Problem: addressability usually fixed
- Solution: intensity is variable, so use it
- ⇒ two adjacent pixels can give impression of point part way between
- ⇒ perceived location of point dependent upon ratio of intensities

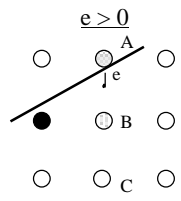


- An antialiased line has virtual pixels "located" at the proper addresses

Antialiased Bresenham Lines



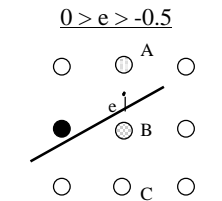
- Use the distance ($e = di/a$) value to determine pixel intensities.
- Three possible cases for the Bresenham algorithm:



$$A = 0.5 + e$$

$$B = 1 - \text{abs}(e+0.5)$$

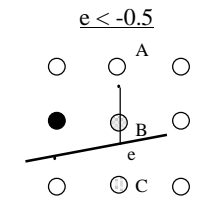
$$C = 0$$



$$A = 0.5 + e$$

$$B = 1 - \text{abs}(e+0.5)$$

$$C = 0$$



$$A = 0$$

$$B = 1 - \text{abs}(e+0.5)$$

$$C = -0.5 - e$$

- What about color?

Handouts: Projections

Processing Polygons

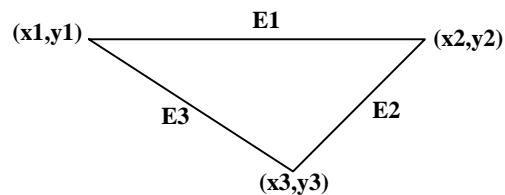


Polygons



- Polygon: many-sided planar figure of vertices and edges
- Vertices: represented by points (x,y)
- Edges: represented as line segments between two points, (x_1,y_1) and (x_2,y_2)

$$P = \{ (x_i, y_i) \} \quad i=1,n$$

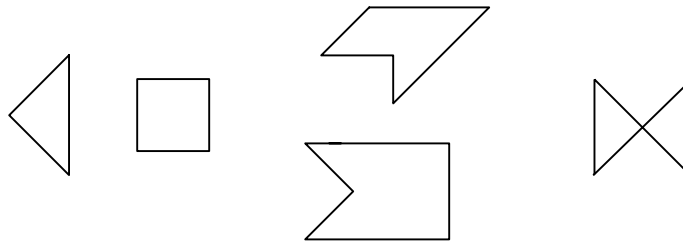


Handouts: Projections

Convex and Concave Polygons



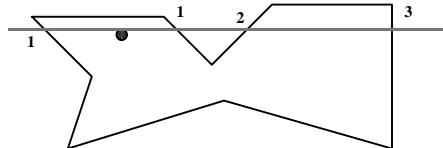
- Convex Polygon:
 - Given P_1, P_2 inside polygon
 - All $P = uP_1 + (1-u)P_2, u \text{ in } [0,1]$ is inside polygon
- Concave = not convex!



How do we know a point is "inside" a polygon?

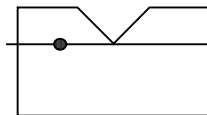


- P is inside a polygon iff a scanline intersects the polygon edges an odd number of times moving from P in either direction

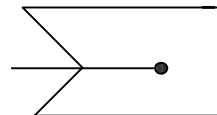


- Problem when scan line crosses a vertex:

Does the vertex count as two points?



Or should it count as one point?

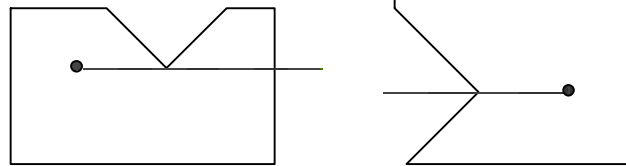
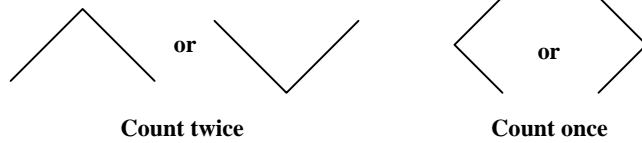


Handouts: Projections

Max-Min Test



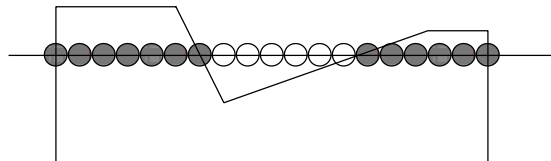
- Vertex = local max or min
- Count it twice, else count it once.



Filling Polygons



- Fill polygon 1 scanline at a time



- Set pixels inside polygon on each scanline to the appropriate value
- Look only for those pixels at which changes occur

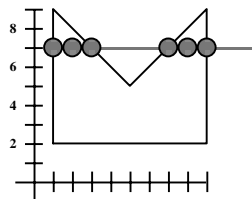
Handouts: Projections

Scan-Line Algorithm



- For each scan-line:

1. Find intersections of scan line with all edges
2. Sort intersections by increasing x-coordinate
3. Fill all pixels between pairs of intersections



For scan-line number 7 the sorted list of x-coordinates is (1,3,7,9)

How do we know a point is "inside" a polygon?

Possible Problems



- Horizontal edges

- Ignore

- Vertices

- If local max or min, count twice, else count once (Implemented by shortening one edge by one pixel)

- Calculating intersections is slow

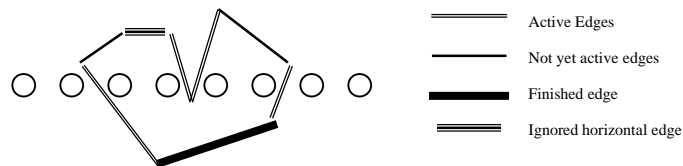
Handouts: Projections

Edge Coherence

- Observations:
 - Not all edges intersect each scanline
 - If edge intersected in scanline i , will probably be intersected by scanline $i+1$
- Consider scanline s , the line $y = s$
$$s = mx^s + b$$
$$\Rightarrow x^s = (s-b)/m$$
- For scanline $s + 1$,
$$x^{s+1} = (s+1 - b)/m = x^s + 1/m$$

Processing Polygons

- Polygon edges sorted according to minimum Y
- Scan lines processed in increasing (decreasing) Y order
- When current scan line reaches edge, it becomes active
- When current scan line passes edge, it becomes inactive



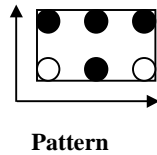
- Active edges sorted according to increasing X
- Fill the scan line between alternating edge intersections

Handouts: Projections

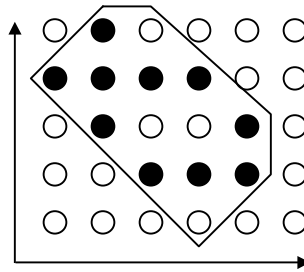
Fill Patterns: Simple "textures"



- Defined as a 0-based, $m \times n$ array
- Pixel (x,y) is assigned the value found in:
 - $\text{pattern}((x \bmod m), (y \bmod n))$



Pattern

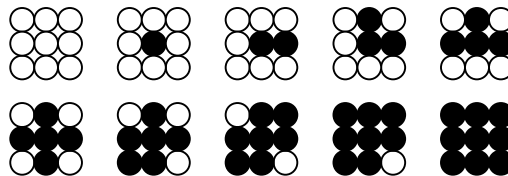


Pattern filled polygon

Halftoning



- Mimic greyscale on bitmapped displays
- Tradeoff resolution (addressability) for range of intensities
- Patterns should be designed to avoid being noticed



Handouts: Projections

Dithering



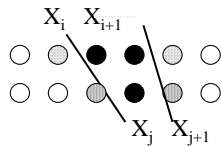
- Another way to mimic grey on bit-mapped displays
- Ordered dither: turn pixel on or off at (x,y) based on
 - desired intensity $I(x,y)$ at that point
 - an $(n \text{ by } n)$ dither matrix D_n
- Each integer 0 to $n^2 - 1$ appears once in the matrix D_n
e.g. D_4

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5
- let $i = x \text{ MOD } 4$, $j = y \text{ MOD } 4$
- if $I(x,y) > D_4(i, j)$ then (x,y) is turned on; otherwise it is not

Antialiasing Polygons



- Polygon edges suffer from aliasing as lines
- Similar method can be used on the scan line fill



- If odd intersection between two pixels $X_i < X < X_{i+1}$
 - pixel X_i is assigned the intensity $(X_{i+1} - X)$
 - pixel X_{i+1} is assigned intensity 1.0
- At even intersection, reverse is true