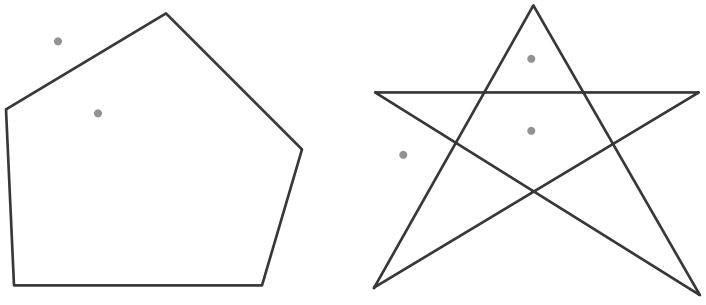
Computer Graphics Course 2005

Polygon Filling (Rasterization)

Point-in-polygon test

⌘How do we tell if a point is inside or outside a polygon?

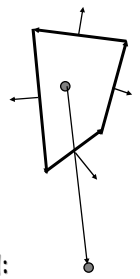


Point-in-polygon test

- Odd-even rule: count the number of times a line from the point of interest to a point known to be outside crosses the edges of the polygon.
  - Odd = inside
  - Even = outside

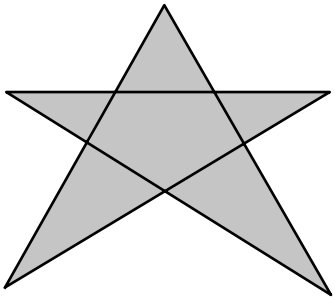
Point-in-polygon test

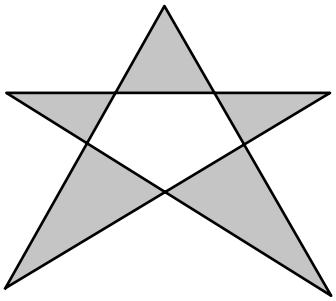
- **Non-zero winding number rule:** Consider the number of times the polygon edges wind around the point of interest (counter-clockwise direction).
  - Define edge direction to be counter-clockwise.
  - Define edge normal as the clockwise normal.
  - Choose a "far" point, outside polygon, and cast a ray from the point of interest to that point.
  - Calculate a winding number: for each edge crossed:
    - If *angle(ray, normal) < 90* → increase counter by 1.
    - If *angle(ray, normal) > 90* → decrease the count by 1.
  - If the winding number is non-zero the point is inside the polygon.



Point-in-polygon test

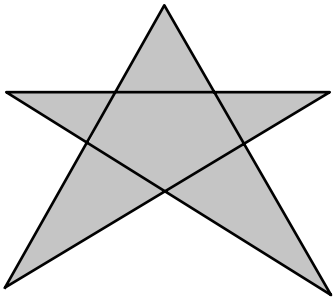
Which interior corresponds to which rule?

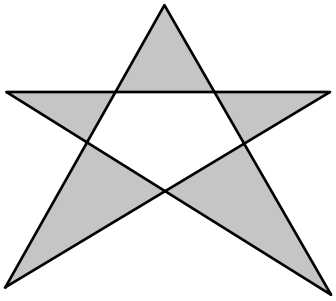




Point-in-polygon test

Which interior corresponds to which rule?





Winding number

Odd-even rule

Scan-Line Polygon Fill

- The scanline "span":
  A span is a group of adjacent pixels on a scanline, that are considered to be interior to the polygon.

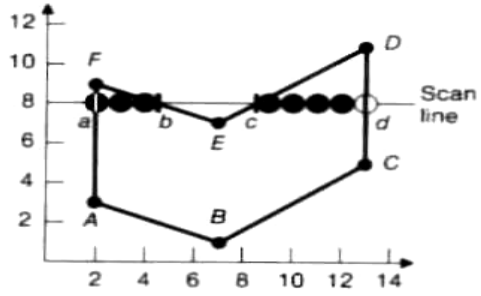
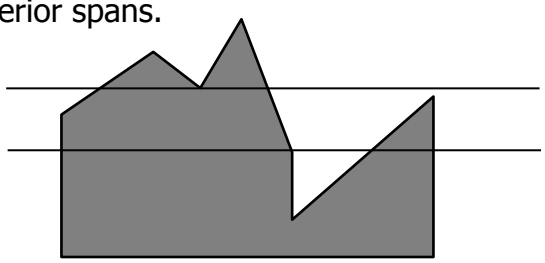


Fig. 3.22 Polygon and scan line 8.

- Two scan-line spans:
  from x=2 through 4
  from x=9 through 13

Polygon Fill-
Scan-line algorithm

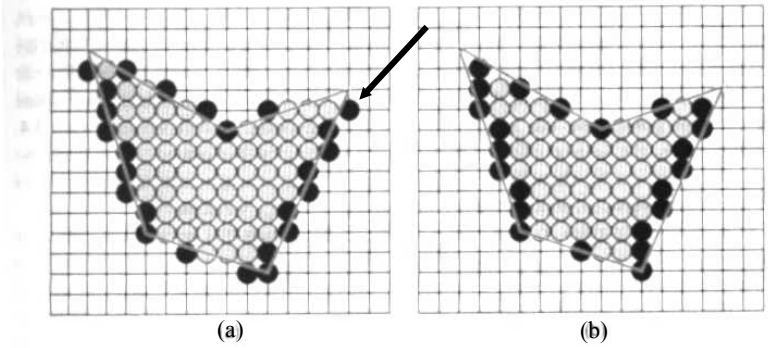
- For each scan-line crossing the polygon:
  - find intersections with polygon edges
  - sort from left to right
  - fill interior spans.



Polygon Fill-
Scan-line algorithm steps

- Find the intersection of the scan line with all edges of the polygon.
- Sort the intersections by increasing x coordinate.
- Fill all pixels between pairs of intersections that lie interior to the polygon, using the odd-parity rule:
  - Parity is initially even, and each intersection inverse the parity bit.
  - Draw when parity is odd.
  - Do not draw when parity is even.

Scan-Line Polygon Fill: span extrema



(a) (b)

MidPoint algorithm edge pixels

Problem: draws pixels outside polygon

Only Interior pixels

Polygon Fill - scanline algorithm
Special cases

- Q: Given an intersection with an arbitrary fractional x value, which pixel on either side of intersection is interior ?

- A: If we approach a fractional intersection to the right and are inside polygon, we round down the x coordinate to be inside polygon; and vice versa.

Polygon Fill - scanline algorithm
Special cases

- Q: How do we deal with intersection at integer pixel coordinate (think of 2 polygons sharing such pixel - to whom does it belong) ?

- A:
  Leftmost pixels of a span are considered to be interior.
  Rightmost pixels, are considered to be exterior.

Polygon Fill - scanline algorithm
Special cases

- Q: How do we deal with intersection at integer pixel coordinate which is also a shared vertex ?
  A: We will count only the Ymin vertex of an edge in the parity calculation, but not the Ymax.

- Q: How do we deal with intersection at integer pixel coordinate where the vertices also define a horizontal line ?
  A: We ignore horizontal edges in intersection calculations.

- Which edges will be drawn in the polygon below?

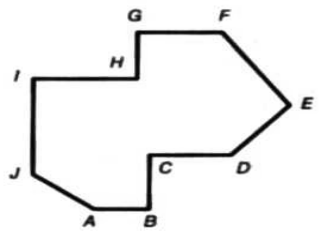


Fig. 3.24 Horizontal edges in a polygon.

Scan-Line Coherence

- Scan-line coherence means that the interior spans corresponding to two adjacent scan lines are usually very similar.
- We use scan-line coherence in order to compute these spans incrementally, rather than intersecting each scan line with the polygon.
- In particular, if a polygon edge intersects a scan line y=k at x_k, the intersection with y=k+1 is

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

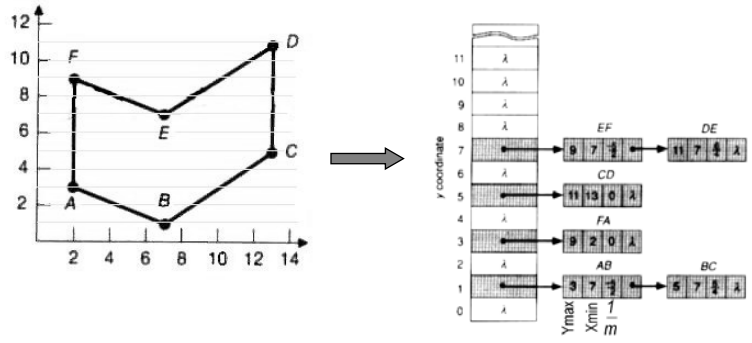
(where *m* is the slope of the edge).

Scan-Line Algorithm- Data structures

- Edge Table (ET):
  - An entry for each scan line crossing the polygon.
  - Each entry contains a list of all polygon edges whose lower end (Ymin) is on this scan line.
  - For each edge we store the following information: x_{min}, y_{max}, and slope (dx, dy or m).
  - The edges in each list are sorted from left to right (according to their x_{min}).

Scan-Line Algorithm- Data structures

⌘Edge-Table Example:



Scan-Line Algorithm- Data structures

Active Edge Table (AET):

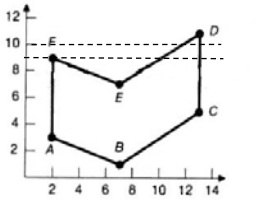
⌘ Maintain an *Active Edge* list that contains only the list of edges crossing the current scan line. Therefore, the edges held by the AET are updated each new scanline.

⌘ In this table each edge element holds the following information: Y_{max}, slope, (all taken from the ET), and the x coordinate of intersection point between the edge and current scanline (this should be updated for each new scanline).

⌘ The edges in this table are sorted by the x coordinate of the intersection points (left to right).

Scan-Line Algorithm

⌘ ET+AET Example (AET is given for scanlines 9 & 10):



Scanlines

9:

10:

AET:

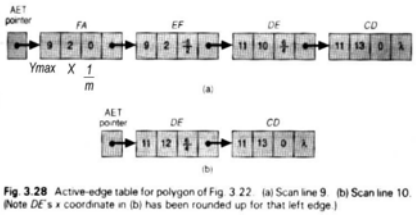
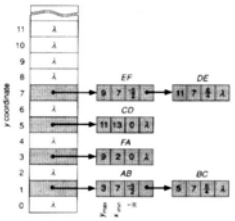


Fig. 3.28 Active-edge table for polygon of Fig. 3.22. (a) Scan line 9. (b) Scan line 10. (Note *DE*'s *x* coordinate in (b) has been rounded up for that left edge.)

ET:



Note that this AET example is general, and don't handle special cases (such as the intersection with edge DC, that according to our rules should be taken as x=12 rather than 13).

Scan-Line Algorithm steps

⌘ Initialize the ET.

⌘ Set 'y' (the current scanline) to the first non-empty entry in the ET.

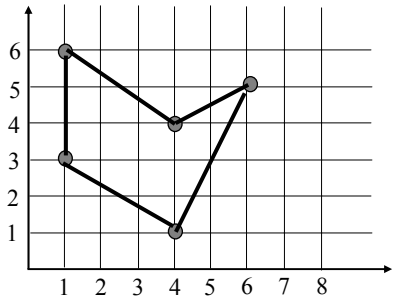
⌘ Repeat until the AET and ET are empty:

- Move new edges from ET to AET: take all edges in entry y of ET (recall that these edges has y_{min} == y).
- Update the x coordinate (intersection point with current scnline) of each edge in the AET.
- Re-sort the AET list, if necessary.
- Fill interior spans according to the edges on the AET list.
- Remove from AET those edges with y_{max} == y (will not intersect the next scanline).
- Increment y by 1 (move to next scanline).

Scan-Line Algorithm

Exercise:

Run the algorithm to fill the polygon below:



Result: