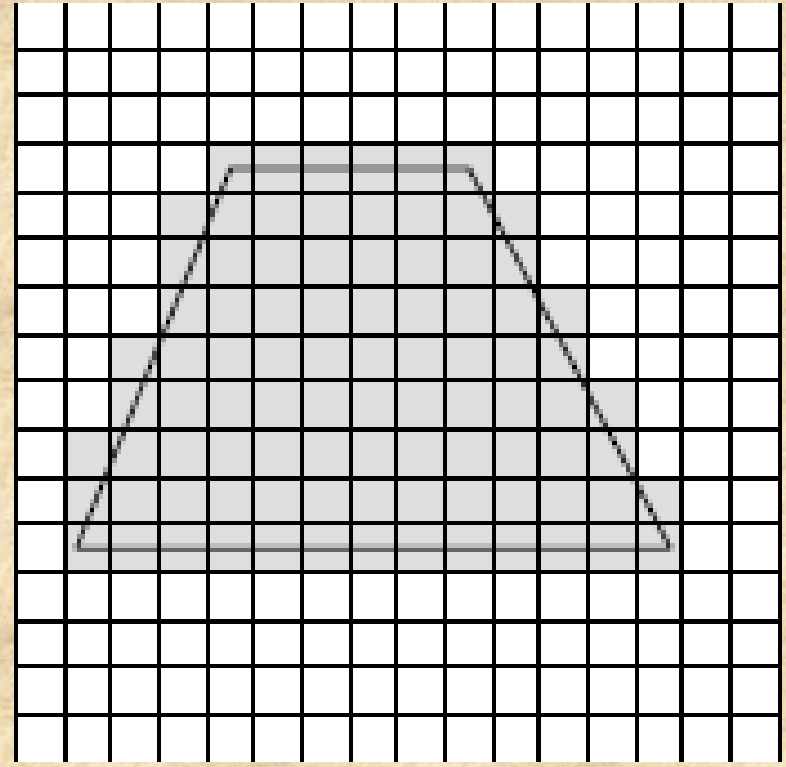
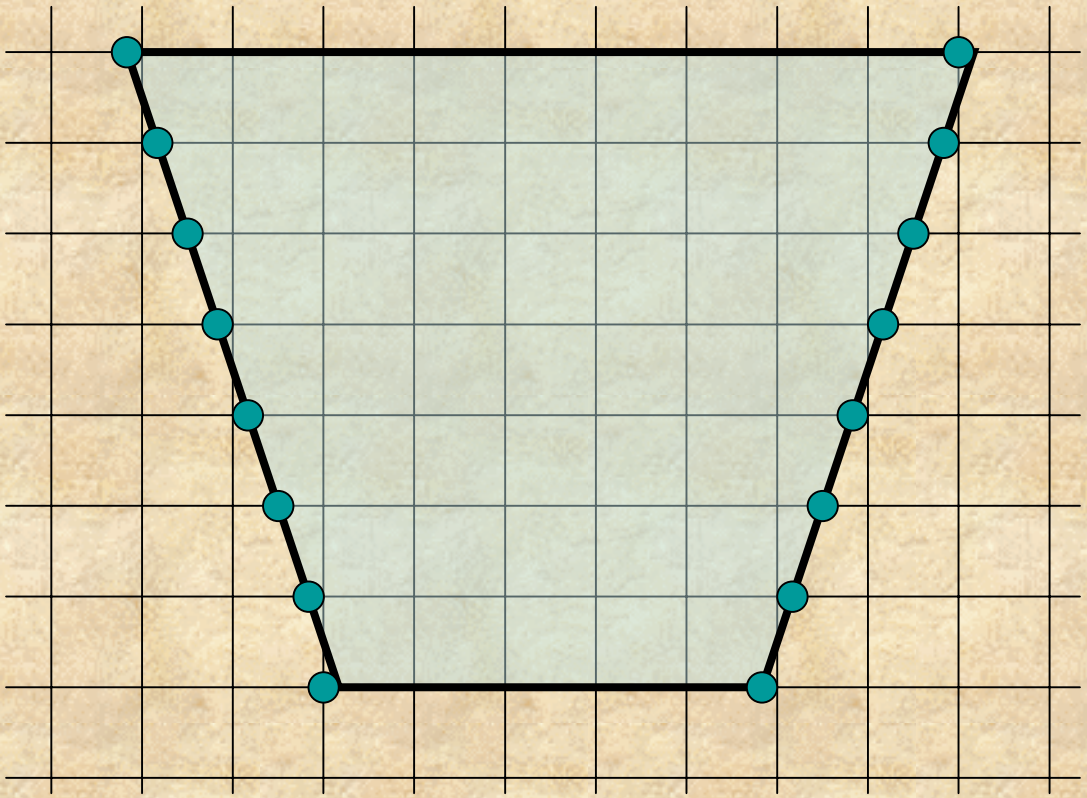
SCAN CONVERSION - POLYFILL



Pixels are not at the center of the grid, but at the intersection of two orthogonal scan lines (on the grid intersection points).

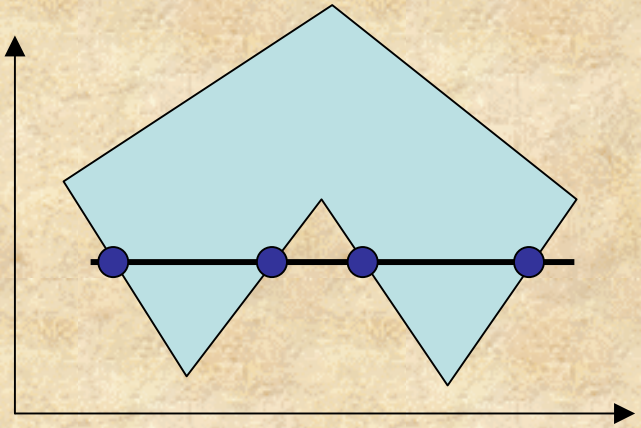


SCANLINE POLYFILL ALGORITHM

Steps (conceptual):

- **Find minimum enclosed rectangle**
- **No. of scanlines = $Y_{max} - Y_{min} + 1$**
- **For each scanline do**
  - **Obtain intersection points of scanline with polygon edges.**
  - **Sort intersections from left to right**
  - **Form pairs of intersections from the list§**
  - **Fill within pairs**
  - **Intersection points are updated for each scanline**
  - **Stop when scanline has reached Y_{max}**

§ - Intersections at vertices require special handling

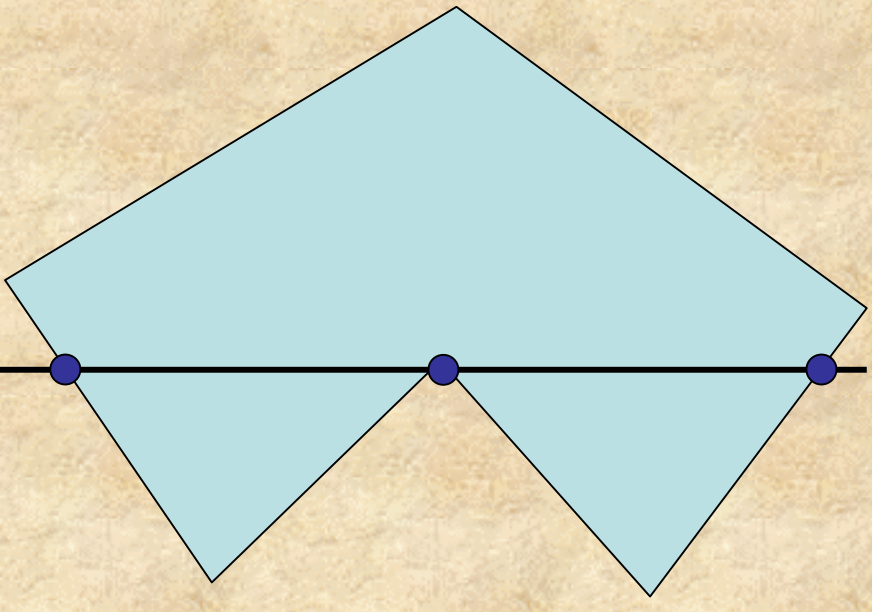


Check if a point is within a Poly?

Look left or right, and count the number of intersection points of the scanline with the edges of the Poly.

If the number is ODD,
Points is Inside
else Outside

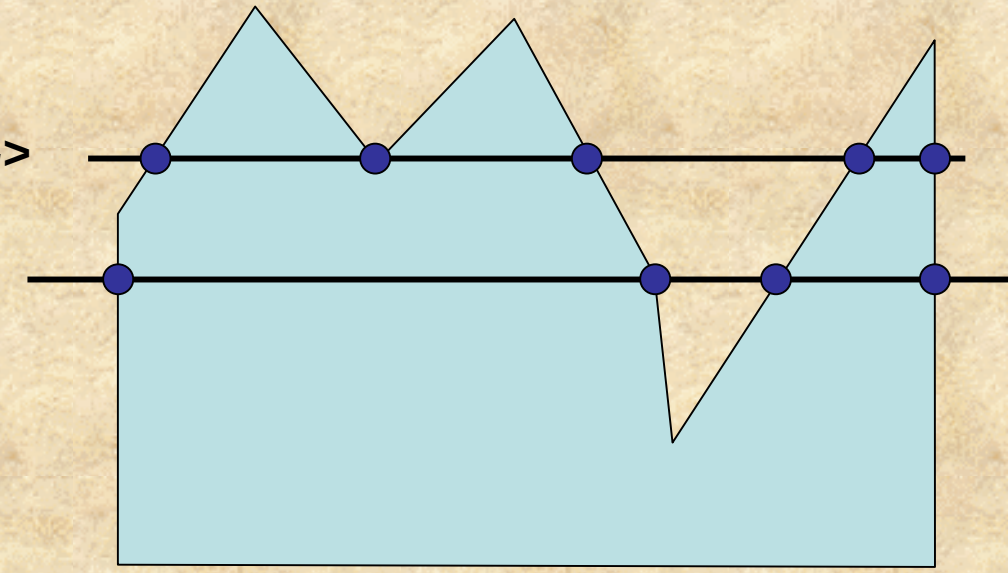
Two different cases of scanlines passing through the vertex of a polygon

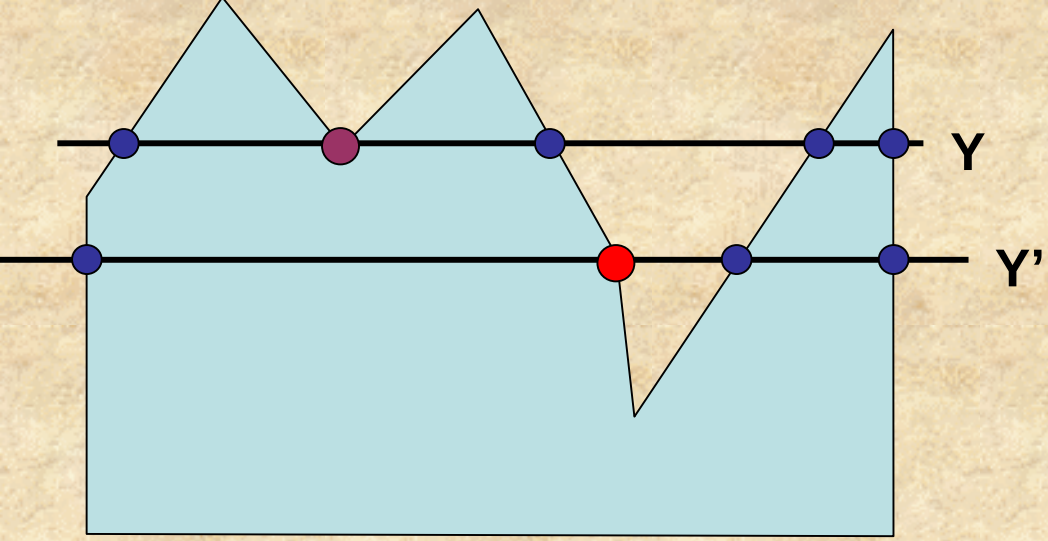


← *Add one more intersection: 3 -> 4*

Add one more intersection: 5 -> 6; ---->

Do not add intersection, keep 4; ---->
HOW ??





What is the difference between the intersection of the scanlines Y and Y', with the vertices?

For Y, the edges at the vertex are on the same side of the scanline, whereas for Y' the edges are on either/both sides of the vertex.
In this case, we require additional processing:

Traverse along the polygon boundary clockwise (or counter-clockwise) and observe the relative change in Y-value of the edges on either side of the vertex (i.e. as we move from one edge to another).

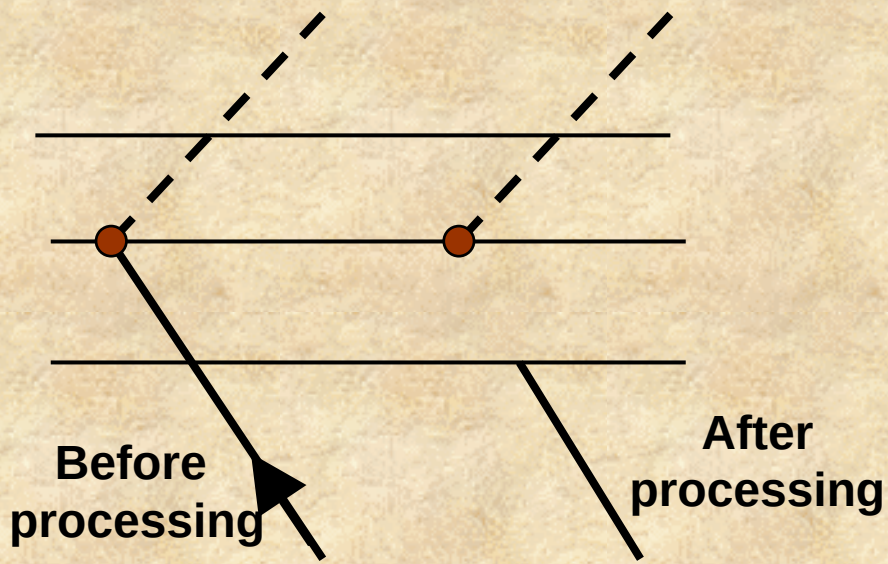
Check the condition:

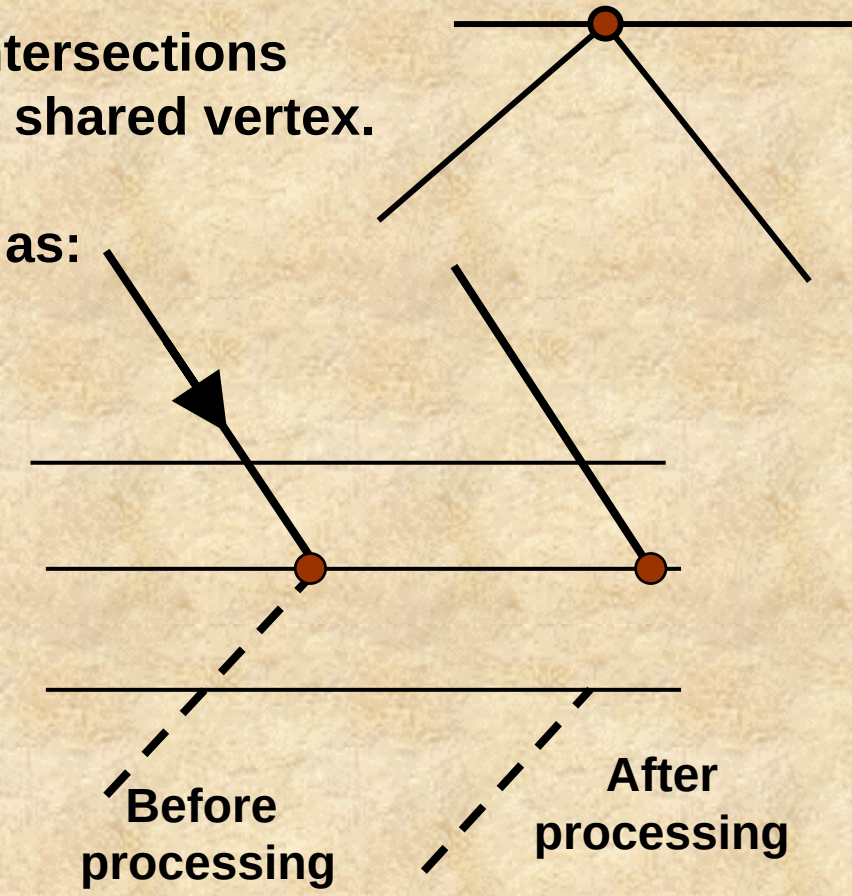
If end-point Y values of two consecutive edges monotonically increase or decrease, count the middle vertex as a single intersection point for the scanline passing through it.

Else the shared vertex represents a local maximum (or minimum) on the polygon boundary.

If the vertex is a *local extrema*, add two intersections for the scan line corresponding to such a shared vertex.

Must avoid this to happen in cases, such as:





To implement the above:

While processing non-horizontal edges (generally) along a polygon boundary in any order, check to determine the condition of monotonically changing (increasing or decreasing) endpoint Y values. If so:

Shorten the lower edge to ensure only one intersection point at the vertex.

Scanline Fill Algorithm (revisited, in brief)

Intersect scanline with polygon edges.
Fill between pairs of intersections

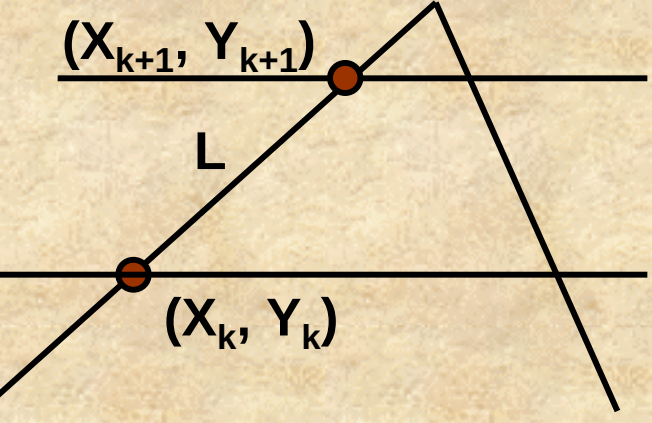
Basic Structure:
For y = Y_{min} to Y_{max}

1) intersect scanline with each edge
2) sort intersections by increasing X
3) fill pairwise (int0->int1, int2->int3, ...)

This is the basic structure, but we are going to handle some special cases to make sure it works right.

- ***scanline coherence*** **- values don't change much from one scanline to the next - the coverage (or visibility) of a face on one scanline typically differs little from the previous one.**

- ***edge coherence*** **- edges intersected by scanline i are typically intersected by scanline i+1**



Slope of the line L (polygon edge) is:

$$m = \frac{Y_{k+1} - Y_k}{X_{k+1} - X_k}$$

where, $Y_{k+1} = Y_k + 1$; Thus, $X_{k+1} = X_k + 1/m$

Thus the intersection for the next scanline is obtained as:
X_{k+1} = round (X_k + 1/m),
where m = $\Delta Y/\Delta X$. How to implement this using integer arithmetic ?

Take an example: m = $\Delta Y/\Delta X$ = 7/3.
Set Counter, C = 0 and counter increment, $\Delta C = \Delta X = 3$;

For the next three scan lines, successive values of C are : 3, 6, 9.
Thus only at 3rd scanline C > ΔY.
X_k is incremented by 1 only at 3rd scanline
and set C = C – ΔY = 9 – 7 = 2.

Repeat the above step(s) till Y_k reaches Y_{max}.

Data Structure Used (typical example):

SET (Sorted Edge table):

Contains all information necessary to process the scanlines efficiently. SET is typically built using a bucket sort, with a many buckets as there are scan lines. All edges are sorted by their Y_{min} coordinate. with a separate Y bucket for each scanline. Within each bucket, edges sorted by increasing X of Y_{min} point.

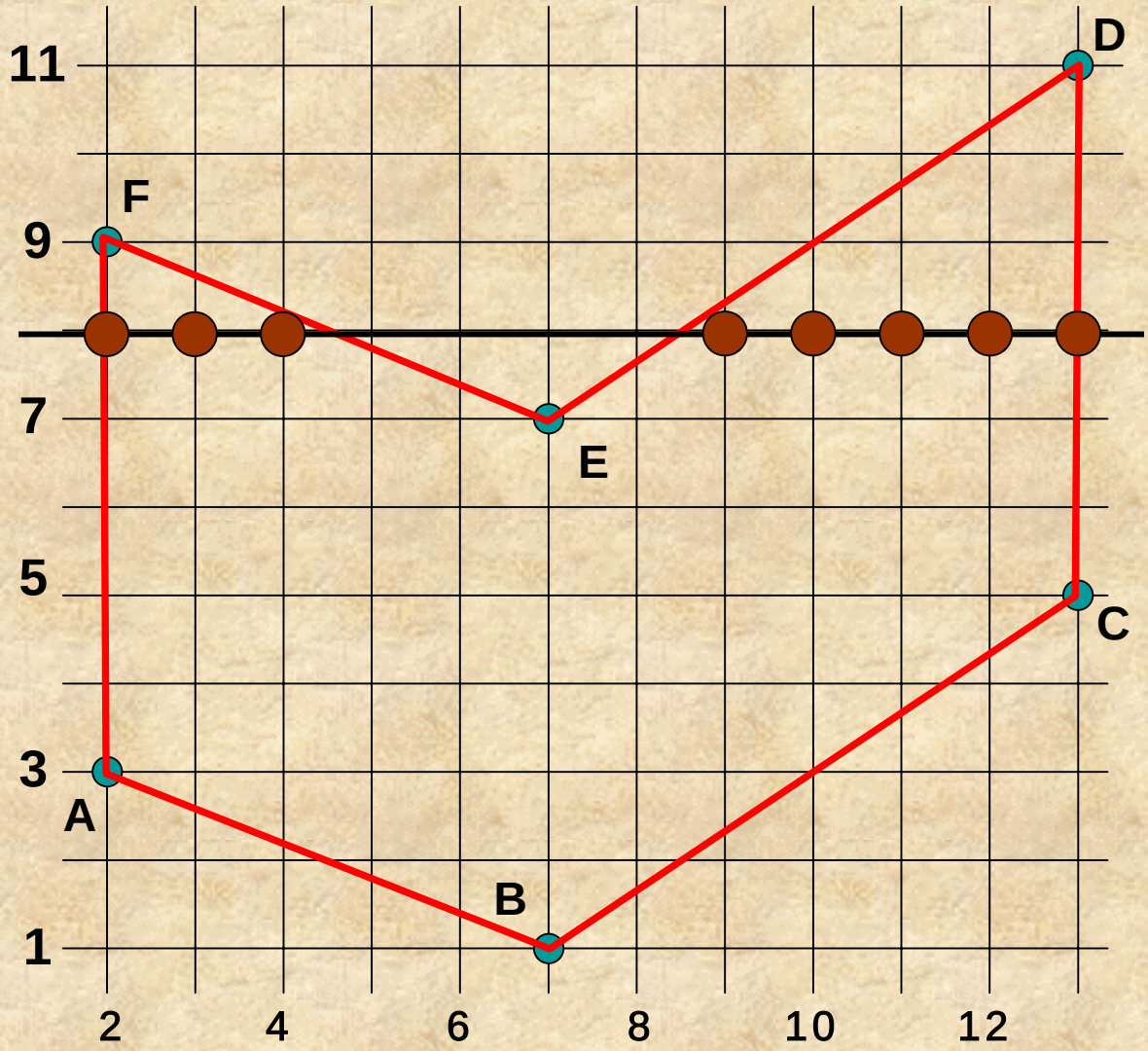
Only non-horizontal edges are stored. Store these edges at the scanline position in the SET.

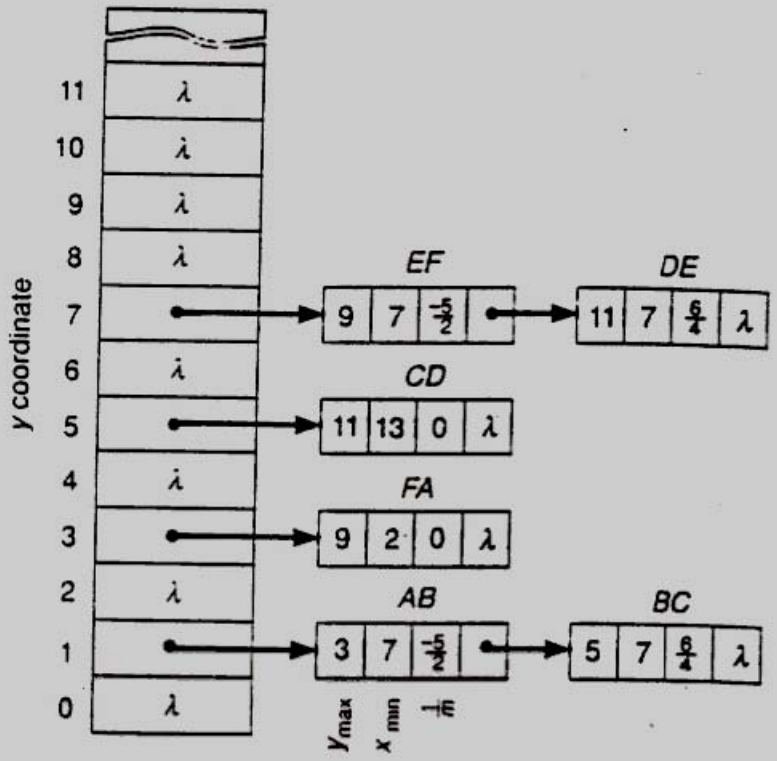
Edge structure (sample record for each scanline):

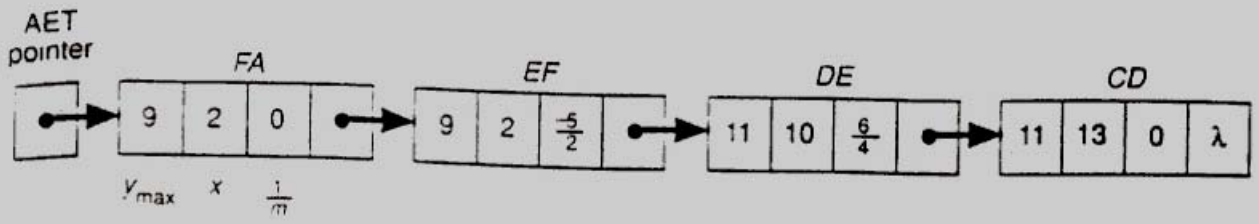
(Y_{max}, X_{min}, $\Delta X/\Delta Y$, pointer to next edge)

AEL (Active edge List):

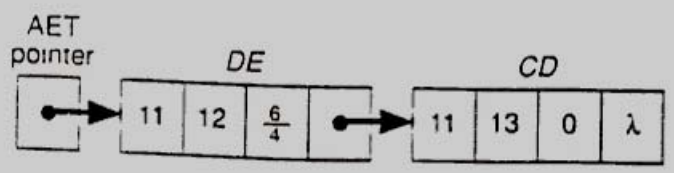
Contains all edges crossed by a scanline at the current stage of iteration. This is a list of edges that are active for this scanline, sorted by increasing X intersections. Also called: Active Edge Table (AET).


11
9
7
5
3
1
2
4
6
8
10
12
A
B
C
D
E
F

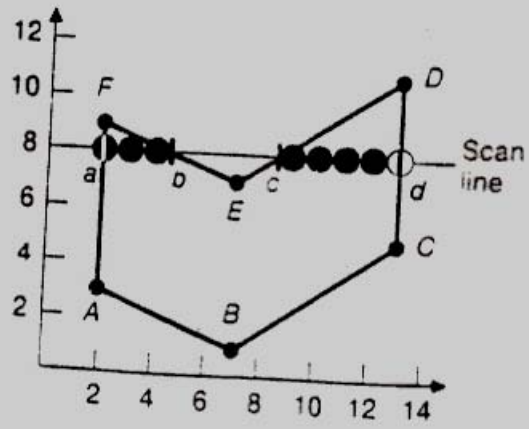

y coordinate
11
10
9
8
7
6
5
4
3
2
1
0
λ
EF
9 7 -5/2
DE
11 7 6/4 λ
CD
11 13 0 λ
FA
9 2 0 λ
AB
3 7 -5/2
BC
5 7 6/4 λ
ymax
xmin
1/m


AET pointer
FA
9 2 0
EF
9 2 -5/2
DE
11 10 6/4
CD
11 13 0 λ
ymax
x
1/m

(a)


AET pointer
DE
11 12 6/4
CD
11 13 0 λ

(b)


12
10
8
6
4
2
F
D
a
b
E
c
d
Scan line
C
A
B
2 4 6 8 10 12 14

Precautions:

Intersection has an integer Y coordinate

If this point is the Y_{min} of the edge's endpoints, count it.

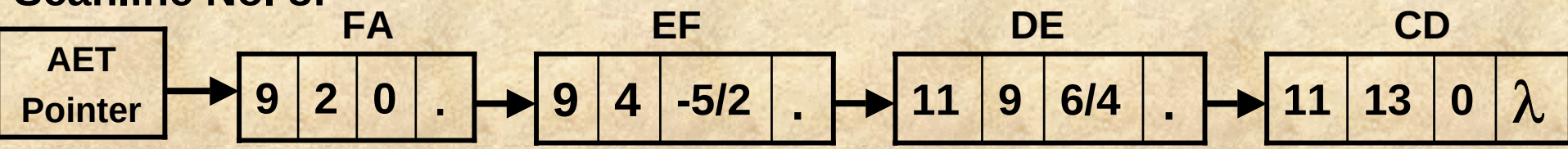
If the edge is horizontal and on the scanline, don't count it.

During iteration process with each scanline, the *AET is updated*.

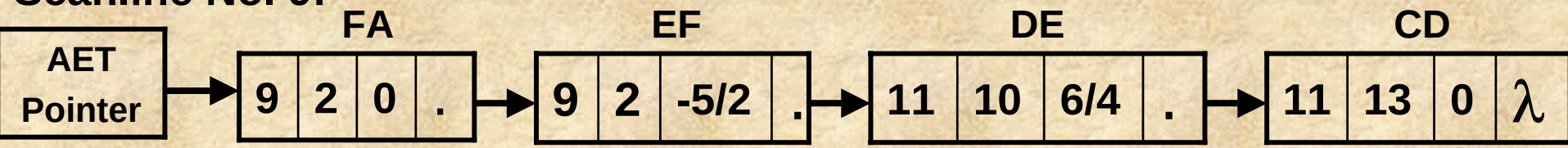
For each scanline the AET keeps track of the set of edges it has to intersect and stores the intersection points in it. The sorting of the entries is w.r.t the X-intersection values.

Have a closer look at it, w.r.t. the previous figure:

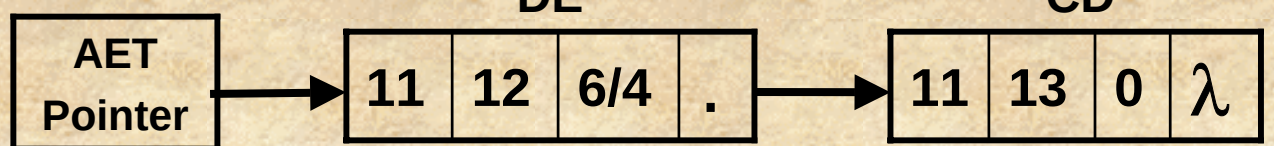
Scanline No. 8:



Scanline No. 9:



Scanline No. 10:



Processing Steps:

- **Set Y to smallest Y in SET entry (first non-empty bucket)**
- **Initialize AET to be empty**
- **Repeat until both AET and SET are empty**

  (i) Move from SET bucket Y to AET, those edges whose Y_{min} = Y.

  (ii) Sort AET on X (simple, as SET is pre-sorted anyway).

  (iii) Fill pixels in scanline Y using pairs of X-coordns. from AET.

  (iv) Increment scanline by 1.

  (v) Remove from AET those entries for which Y = Y_{max} (edges not involved).

  (vi) For each non-vertical edge in AET, update X for new Y.

The algorithm:

scan-fill(polygon)

Construct the Edge table (ET)
Y = min(all Y in the ET)
AET= null

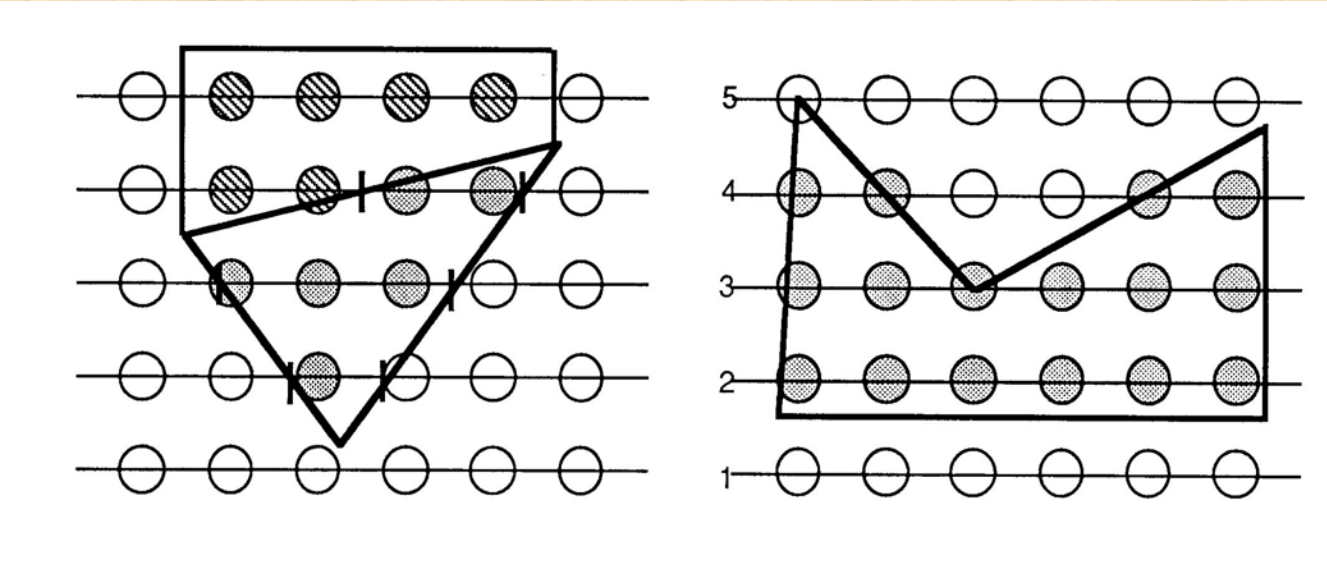
for Y = Y_{min} to Y_{max}

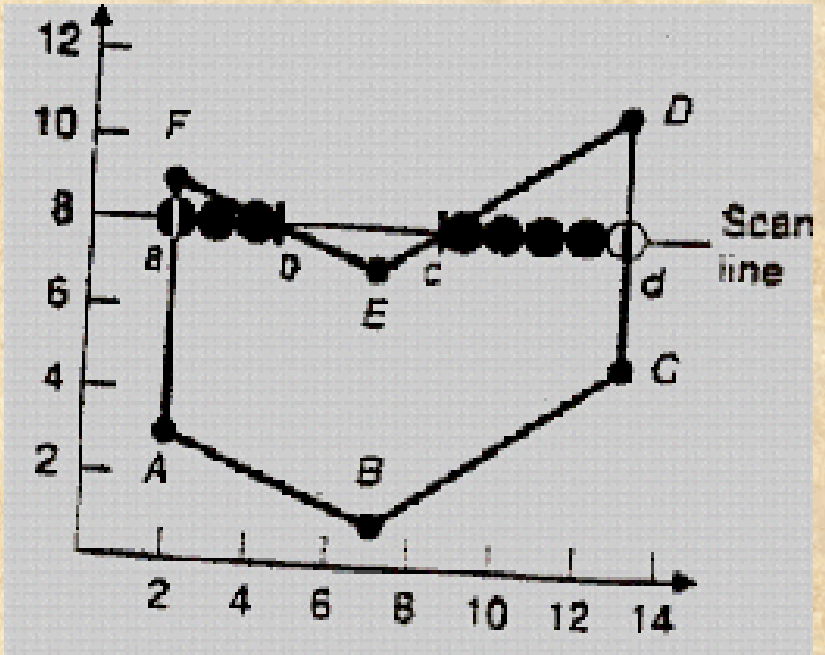
Merge-sort ET[Y] into AET by X value
Fill between pairs of X in AET.

for each edge in AET
if edge.Y_{max} = Y
remove edge from AET
else
edge.X = edge.X + dx/dy
endif

sort AET by X value

end scan_fill


5
4
3
2
1


12
10
8
6
4
2
F
D
B
E
c
d
C
A
B
Scan
line
2
4
6
8
10
12
14