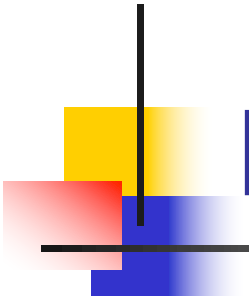
CS 112 - Polygon Scan Conversion

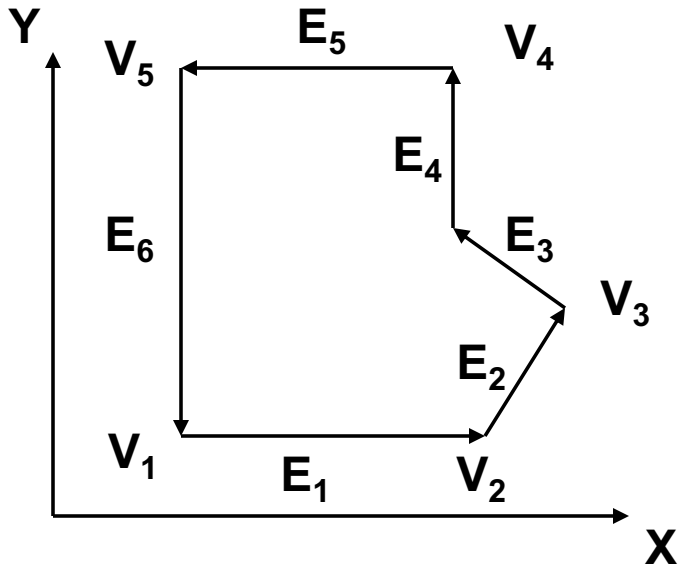


Polygon Classification

- Convex
  - All interior angles are less than 180 degrees
- Concave
  - Interior angles can be greater than 180 degrees
- Degenerate polygons
  - If all vertices are collinear

Identifying Concave Polygons

- Using direction of cross products of adjacent edges
  - If all same sign, then convex, else concave



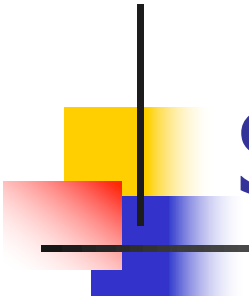
$E_1 \times E_2 > 0$

$E_2 \times E_3 > 0$

$E_3 \times E_4 < 0$

$E_4 \times E_5 > 0$

$E_5 \times E_6 > 0$

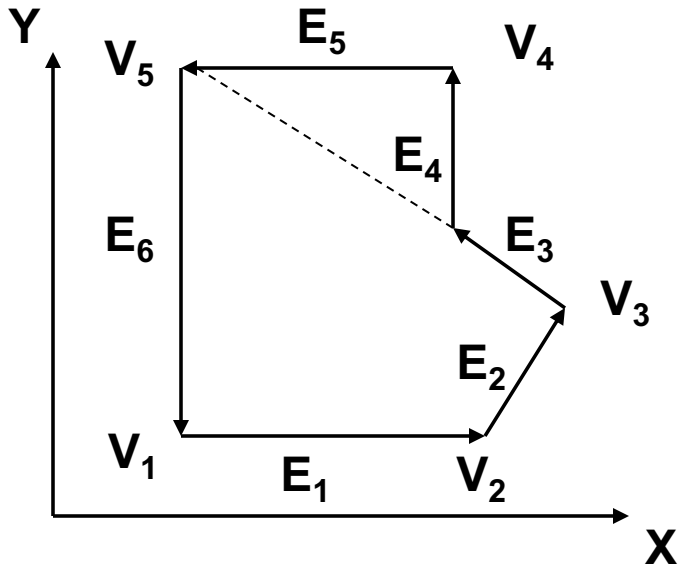


Splitting Concave Polygons

- Assume that the polygon is on XY plane
- Find the cross products of adjacent edges
- If it has a negative z component
  - Split the polygon along the line of the first edge vector of the cross product

Identifying Concave Polygons

- Using direction of cross products of adjacent edges
  - If all same sign, then convex, else concave



$E_1 \times E_2 > 0$

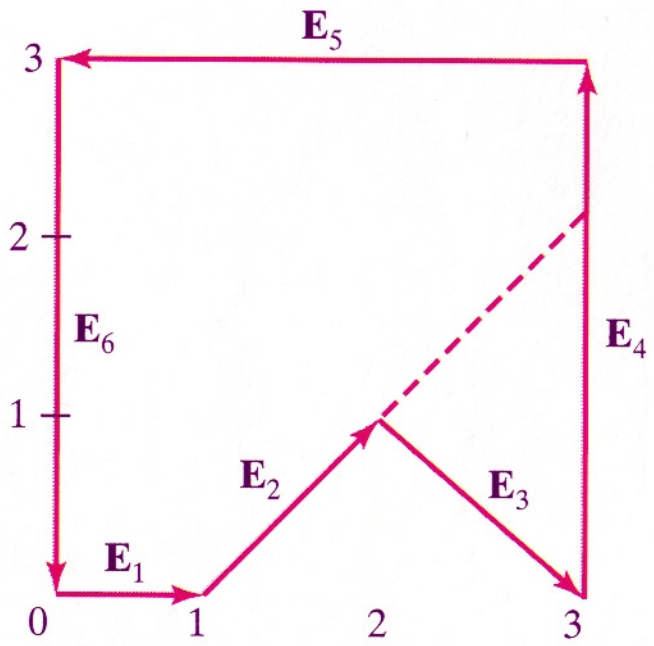
$E_2 \times E_3 > 0$

$E_3 \times E_4 < 0$

$E_4 \times E_5 > 0$

$E_5 \times E_6 > 0$

Example

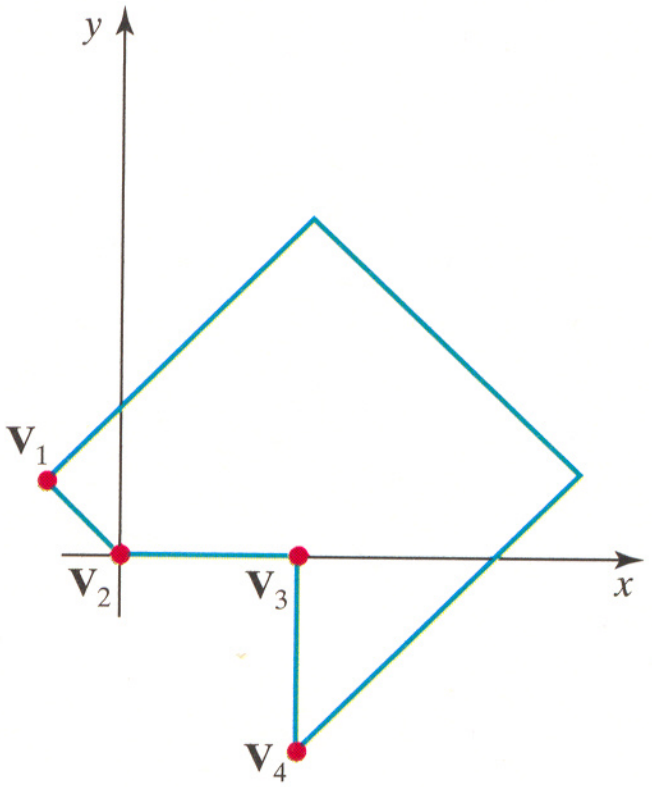


$$\mathbf{E}_1 = (1, 0, 0) \qquad \mathbf{E}_2 = (1, 1, 0)$$
$$\mathbf{E}_3 = (1, -1, 0) \qquad \mathbf{E}_4 = (0, 2, 0)$$
$$\mathbf{E}_5 = (-3, 0, 0) \qquad \mathbf{E}_6 = (0, -2, 0)$$

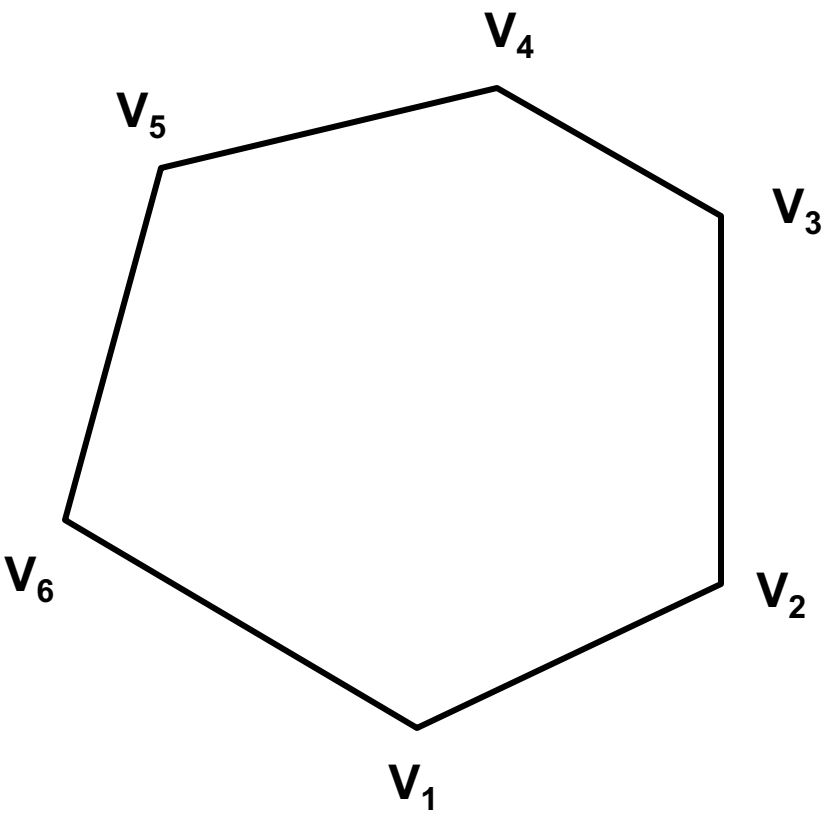
$$\mathbf{E}_1 \times \mathbf{E}_2 = (0, 0, 1) \qquad \mathbf{E}_2 \times \mathbf{E}_3 = (0, 0, -2)$$
$$\mathbf{E}_3 \times \mathbf{E}_4 = (0, 0, 2) \qquad \mathbf{E}_4 \times \mathbf{E}_5 = (0, 0, 6)$$
$$\mathbf{E}_5 \times \mathbf{E}_6 = (0, 0, 6) \qquad \mathbf{E}_6 \times \mathbf{E}_1 = (0, 0, 2)$$

Rotational Method

- For each vertex V_k
  - Translate so that V_k coincides with origin
  - Rotate clockwise so that V_{k+1} lies on the x-axis
  - If V_{k+2} is below the x-axis, then concave
  - Split along x-axis to get two polygons
  - Apply inverse transformations

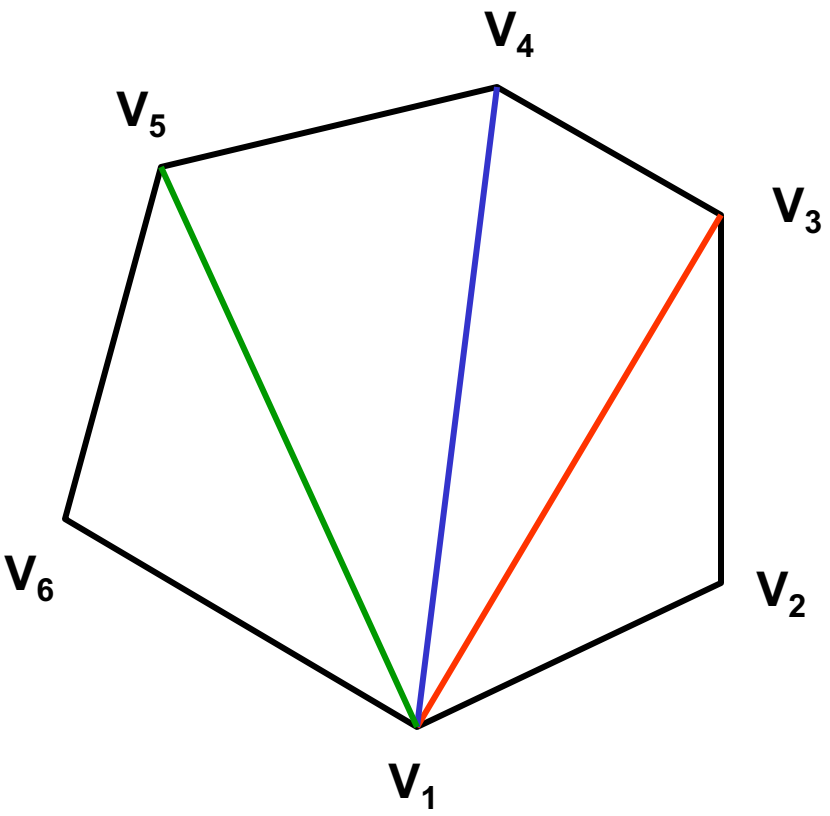


Triangulating a Convex Polygon



- Take any three vertices in order
- Throw away the middle vertex from the list
- Apply to the same procedure
- Until only three vertices remain

Triangulating a Convex Polygon



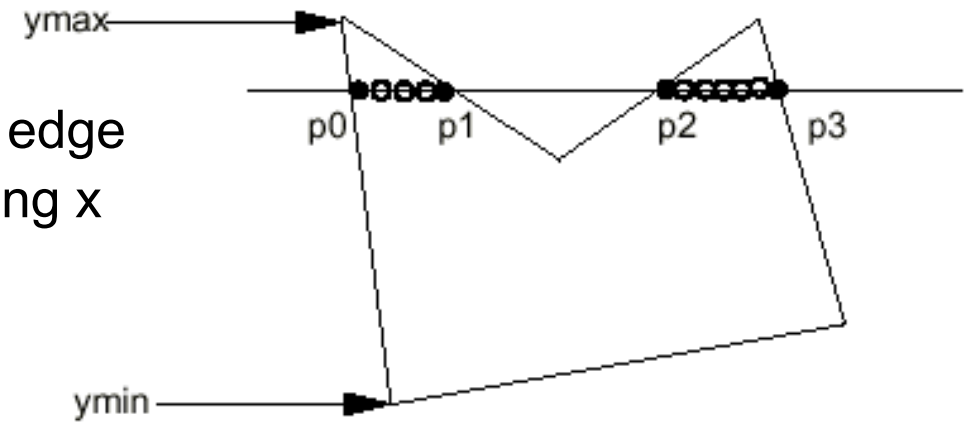
- $V_1, V_2, V_3, V_4, V_5, V_6$
- V_1, V_2, V_3 – tri 1
- V_1, V_3, V_4, V_5, V_6
- V_1, V_3, V_4 – tri 2
- V_1, V_4, V_5, V_6
- V_1, V_4, V_5 – tri 3
- V_1, V_5, V_6 – tri 4

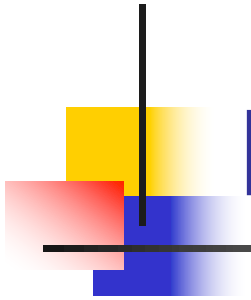
Polygon Rasterization

- Works for both convex and concave polygons
- Basic Idea: Intersect scanline with polygon edges and fill between pairs of intersections

For y = ymin to ymax

1) intersect scanline y with each edge
2) sort interesections by increasing x
[p0,p1,p2,p3]
3) fill pairwise (p0 - p1, p2 - p3)

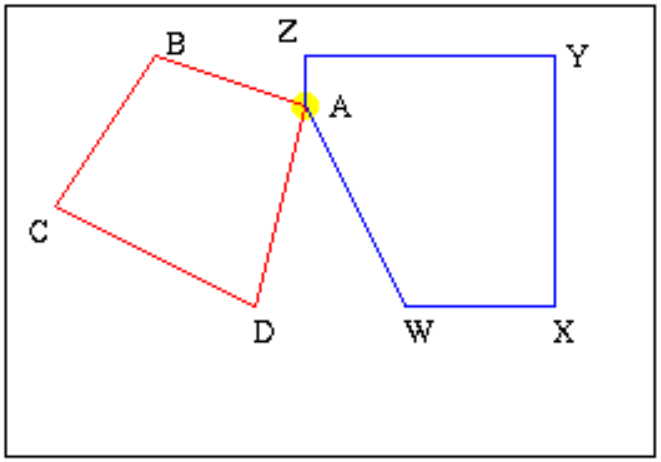




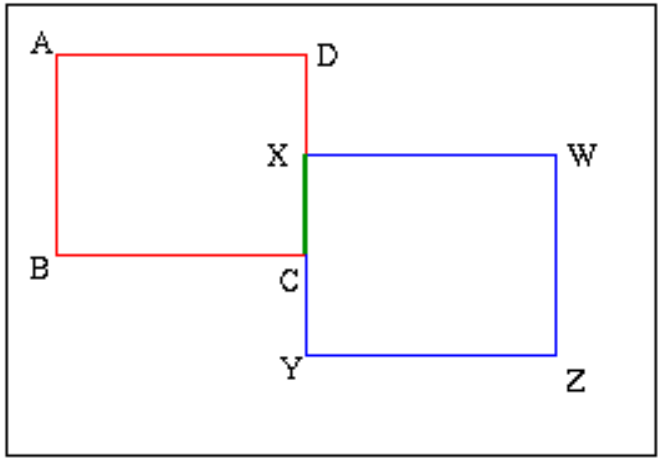
Filling in – Odd Parity Rule

- Sort the intersections
- Start with even parity, and traverse scanline from left to right
- Whenever an intersection is encountered, flip parity
- If parity is odd, draw the pixel
- Continue till the end of the scanline

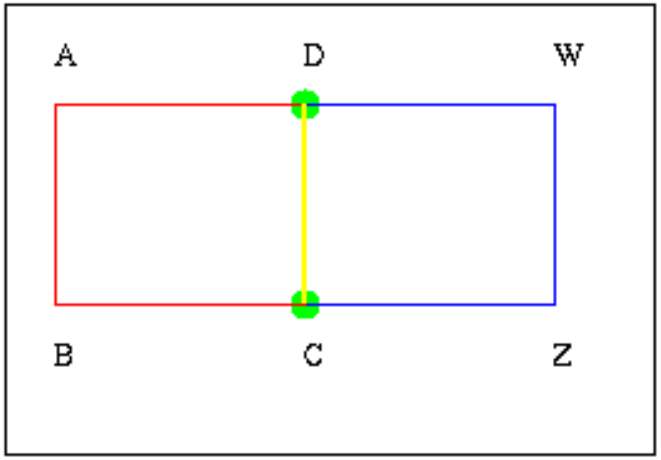
Problem: Shared Boundaries



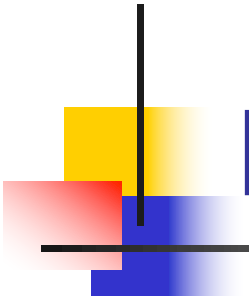
Case 1: Shared vertices



Case 2: Shared edges

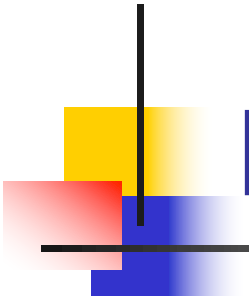


Case 3: Shared vertices and edges



How to solve this?

- Let the pixel be colored in the order the polygons are scan converted
- Rightmost polygon is the dominant one
- Problems
  - Coloring same pixel multiple times
  - Can cause coloring problems depending on what exactly the coloring function does
    - Can create an amalgam of colors

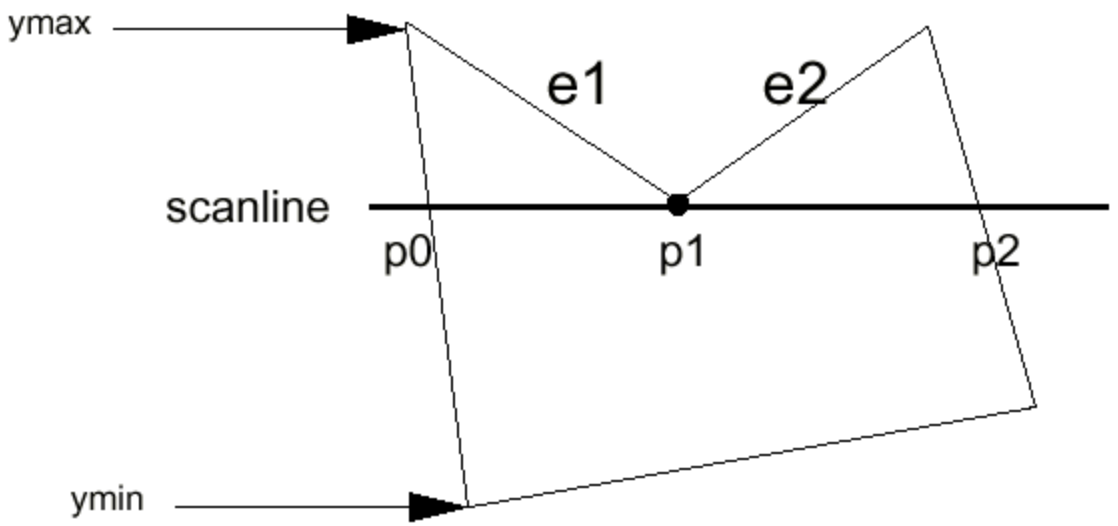


How to solve this?

- Define interior: For a given pair of intersection points (Xi, Y), (Xj, Y)
- Fill ceiling(Xi) to floor(Xj)
  - Will resolve shared boundary problems
- Intersection has an integer X coordinate
  - if Xi is integer, we define it to be interior (fill)
  - if Xj is integer, we define it to be exterior (don't fill)

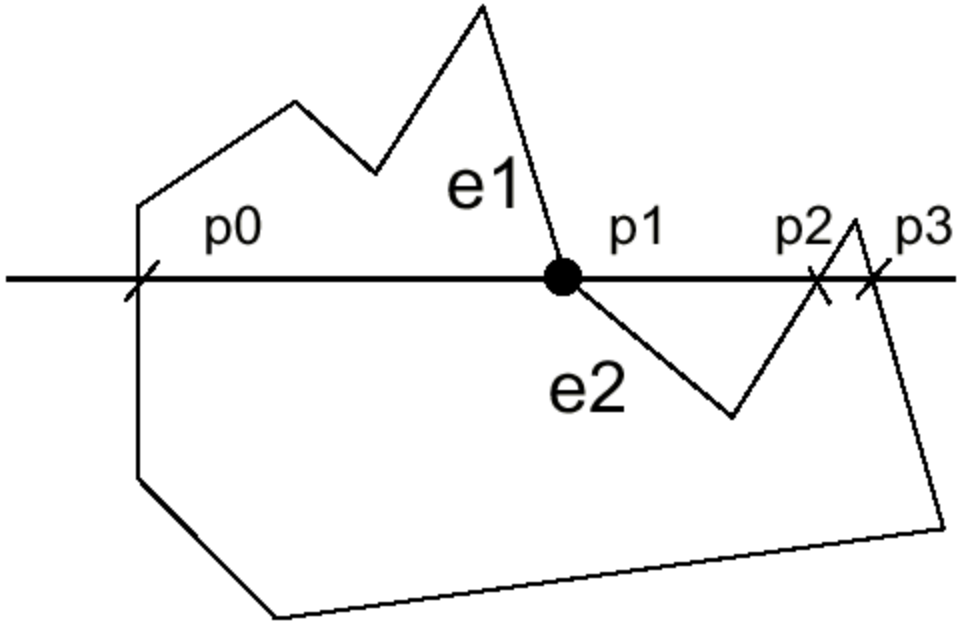
Problem: Edge endpoint

- Intersection is an edge end point, say: (p0, p1, p2) ??
- (p0,p1,p1,p2), so we can still fill pairwise
- In fact, if we compute the intersection of the scanline with edge e1 and e2 separately, we will get the intersection point p1 twice. Keep both of the p1.



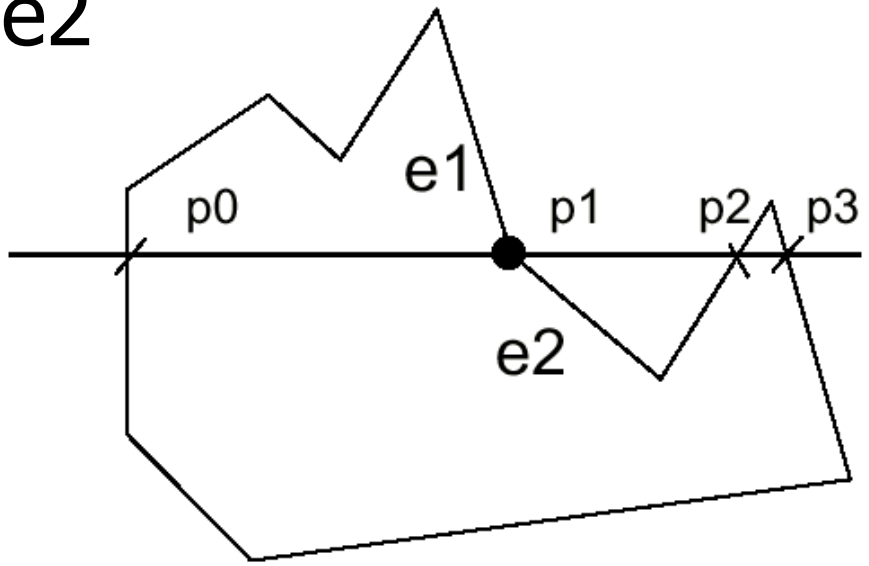
Problem: Edge endpoint

- But what about this case: still (p0,p1,p1,p2)



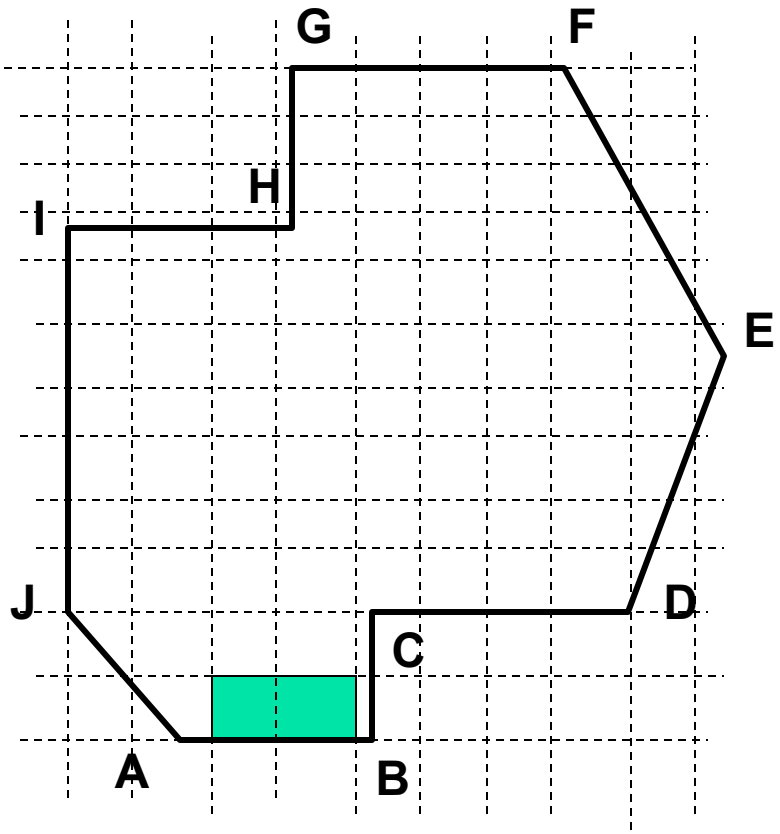
Rule

- Rule:
  - If the intersection is the ymin of the edge's endpoint, count it. Otherwise, don't.
- Don't count p1 for e2



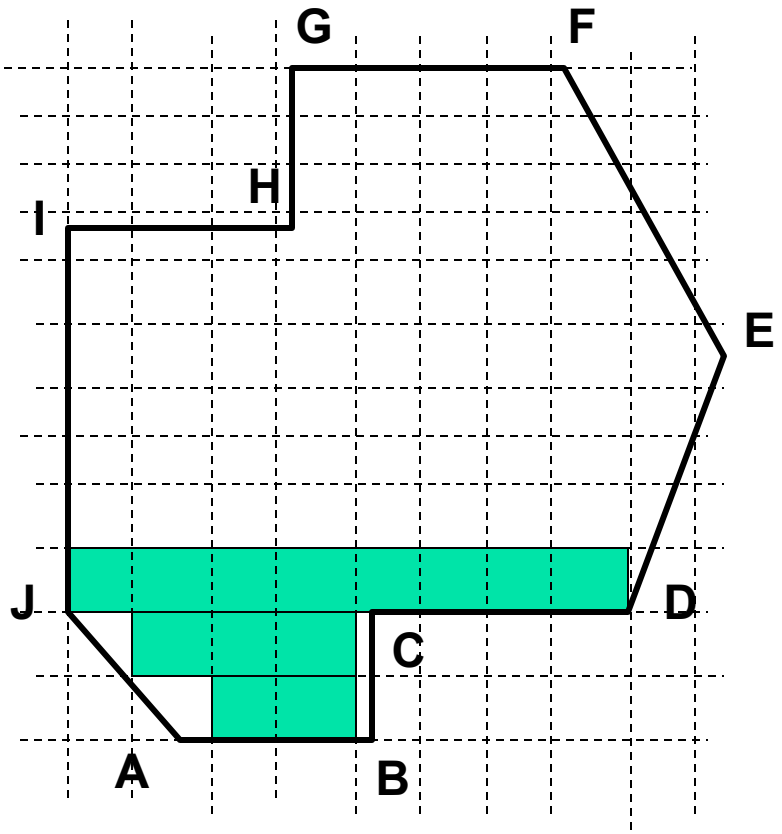
Horizontal edges

- Need not be considered
- AB
  - A – intersection of JA
    - Parity becomes odd at A
  - B – intersection of BC
    - Parity becomes even at B
  - Hence AB is drawn



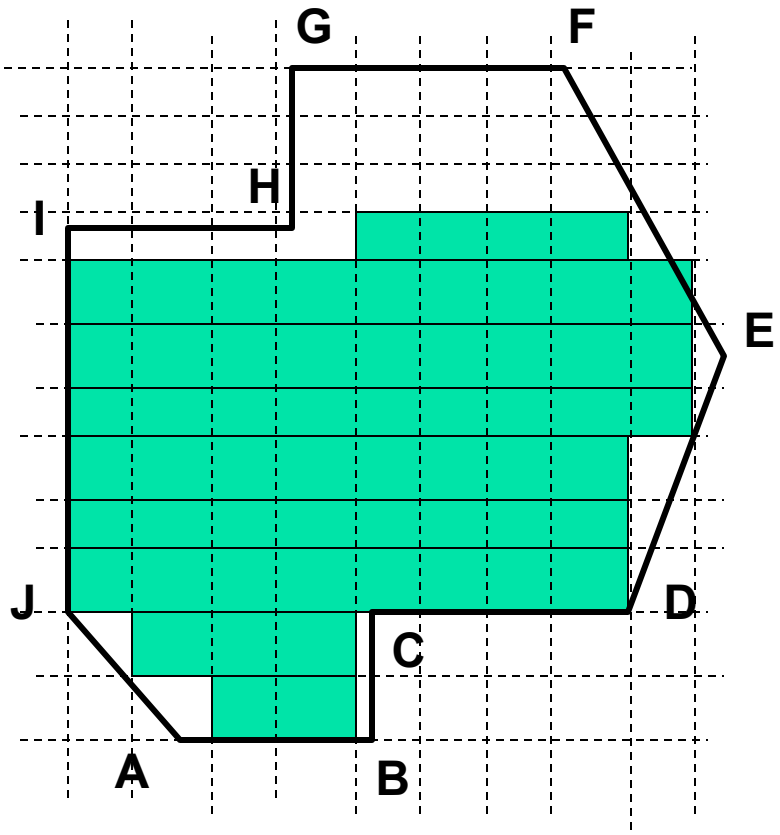
Horizontal edges

- CD
  - J – intersection of IJ
    - Not of JA
    - Parity becomes odd at J
  - C – no change seen
    - Parity remains odd
  - D – intersection of ED
    - Parity changes to even
  - CD is drawn



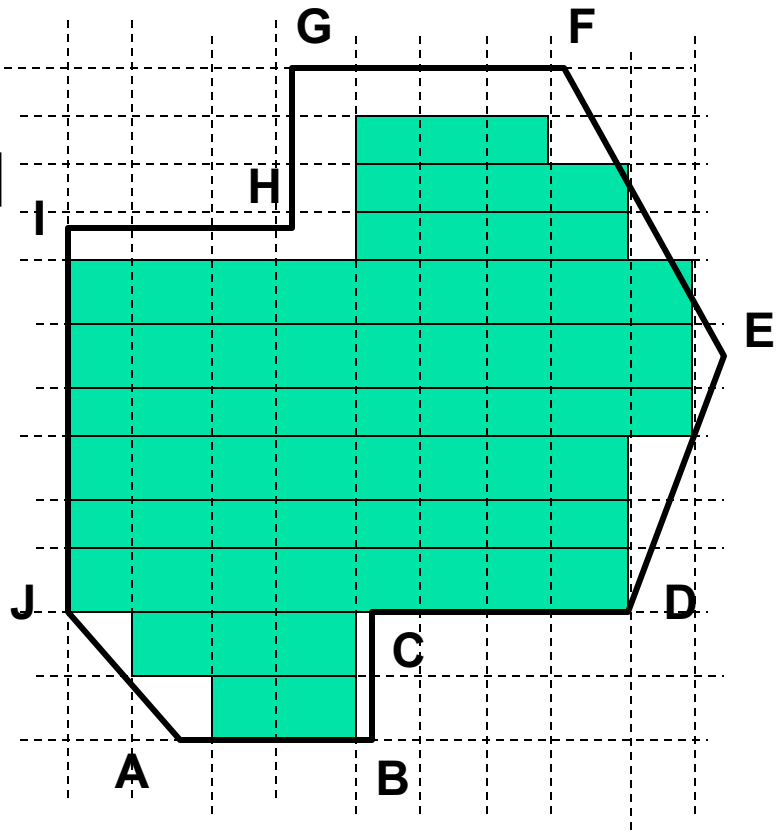
Horizontal edges

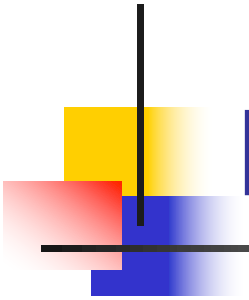
- IH
  - I – no intersection of IJ
    - Parity remains even
  - H – intersection of GH
    - Parity becomes odd
  - IH is not drawn
  - But right span of IH is drawn



Horizontal edges

- GF
  - G – no intersection of GH
    - Parity remains even
  - F – no intersection of FE
    - Parity becomes even
  - GF is not drawn



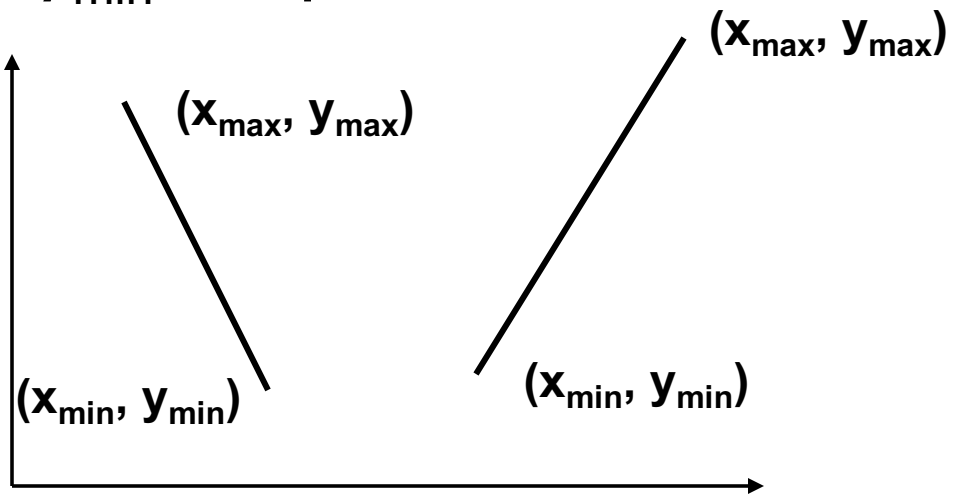


Performance Improvement

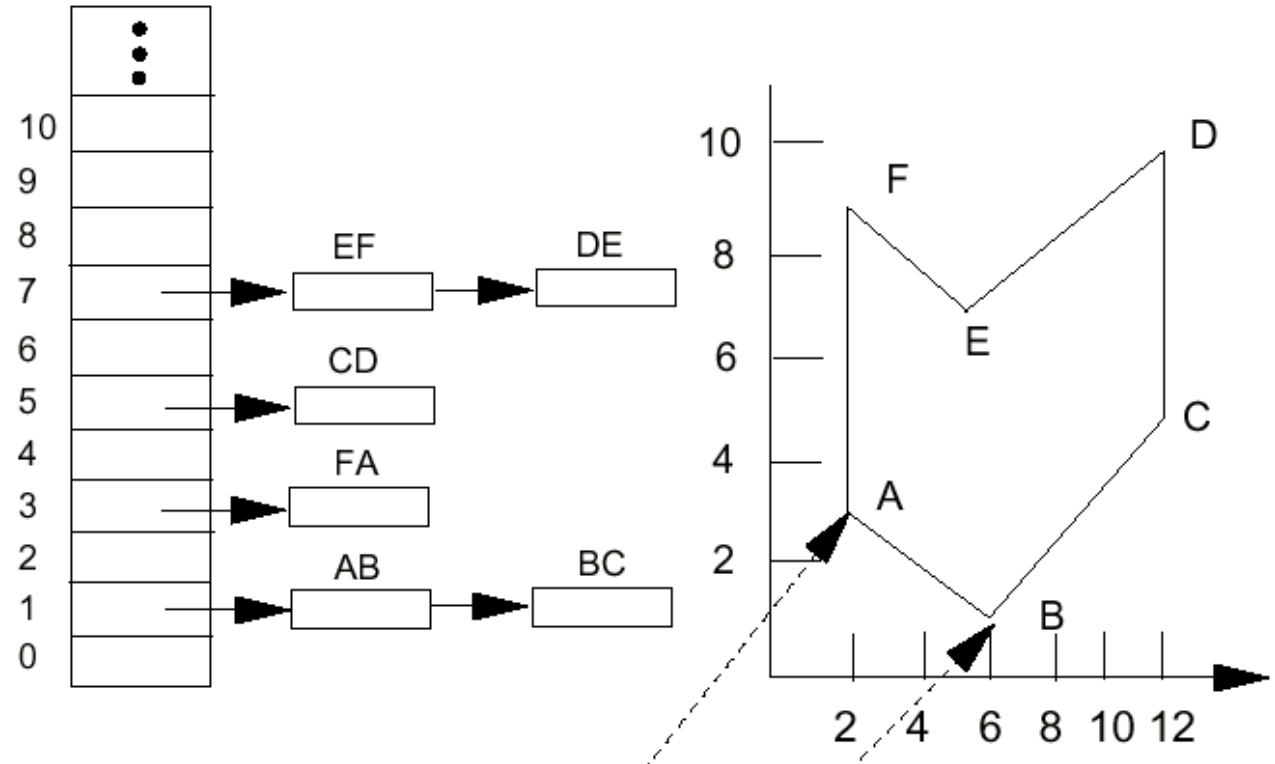
- Brute force: intersect all the edges with all scanline
- Goal:compute the intersections more efficiently
  - find the ymin and ymax of each edge and intersect the edge only when it crosses the scanline
  - only calculate the intersection of the edge with the first scan line it intersects
  - calculate dx/dy
  - for each additional scanline, calculate the new intersection as x = x + dx/dy

Data Structure

- Edge table: Bucket Sort
  - A separate bucket for each scanline
  - Each edge goes to the bucket of its y_{min} scanline
  - Within each bucket, edges are sorted by increasing x of the y_{min} endpoint

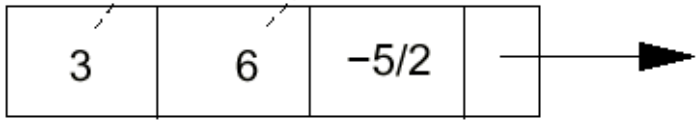


Edge Table



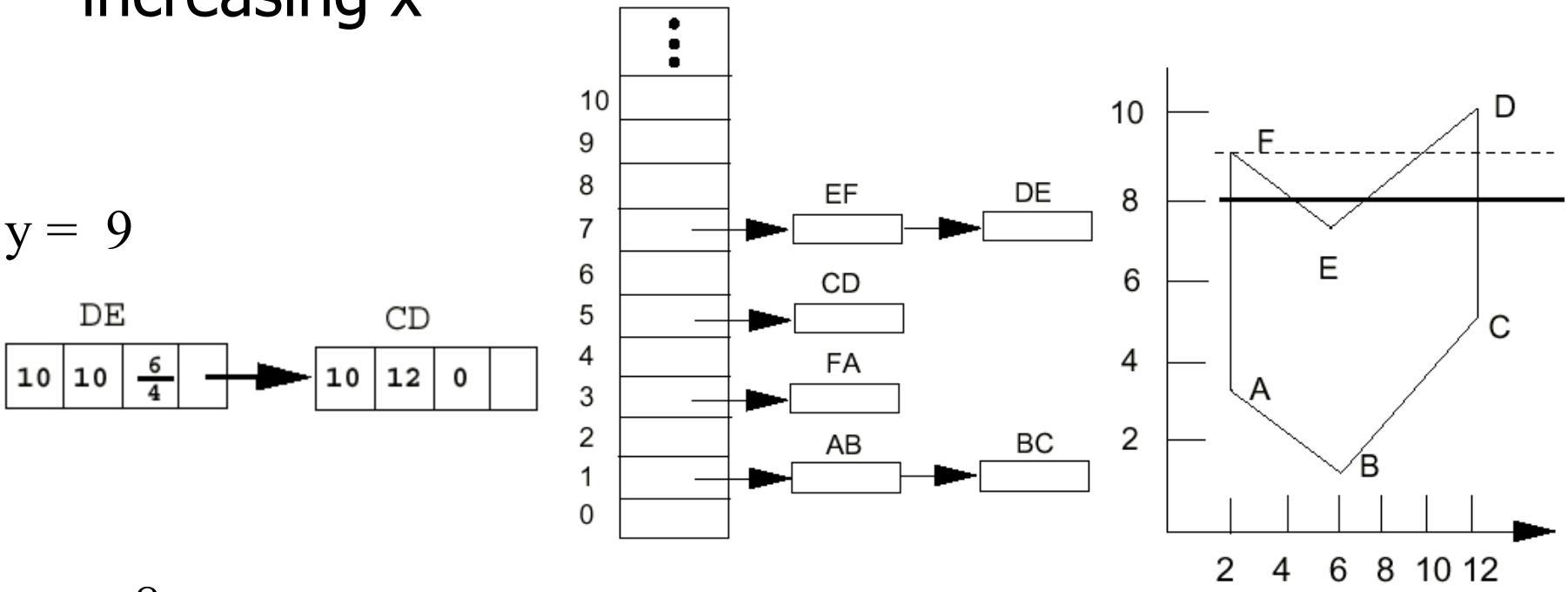
– Edge structure: ymax, xmin, dx/dy, next

AB:



Active Edge Table (AET)

- A list of edges active for current scanline, sorted in increasing x



Algorithm

```
Construct the Edge Table (ET);
Active Edge Table (AET) = null;
for y = Ymin to Ymax
    Merge-sort ET[y] into AET by x value
    for each edge in AET
        if edge.ymax = y
            remove edge from AET
    Fill between pairs of x in AET
    for each edge in AET
        edge.x = edge.x + dx/dy
    sort AET by x value
end scan_fill
```