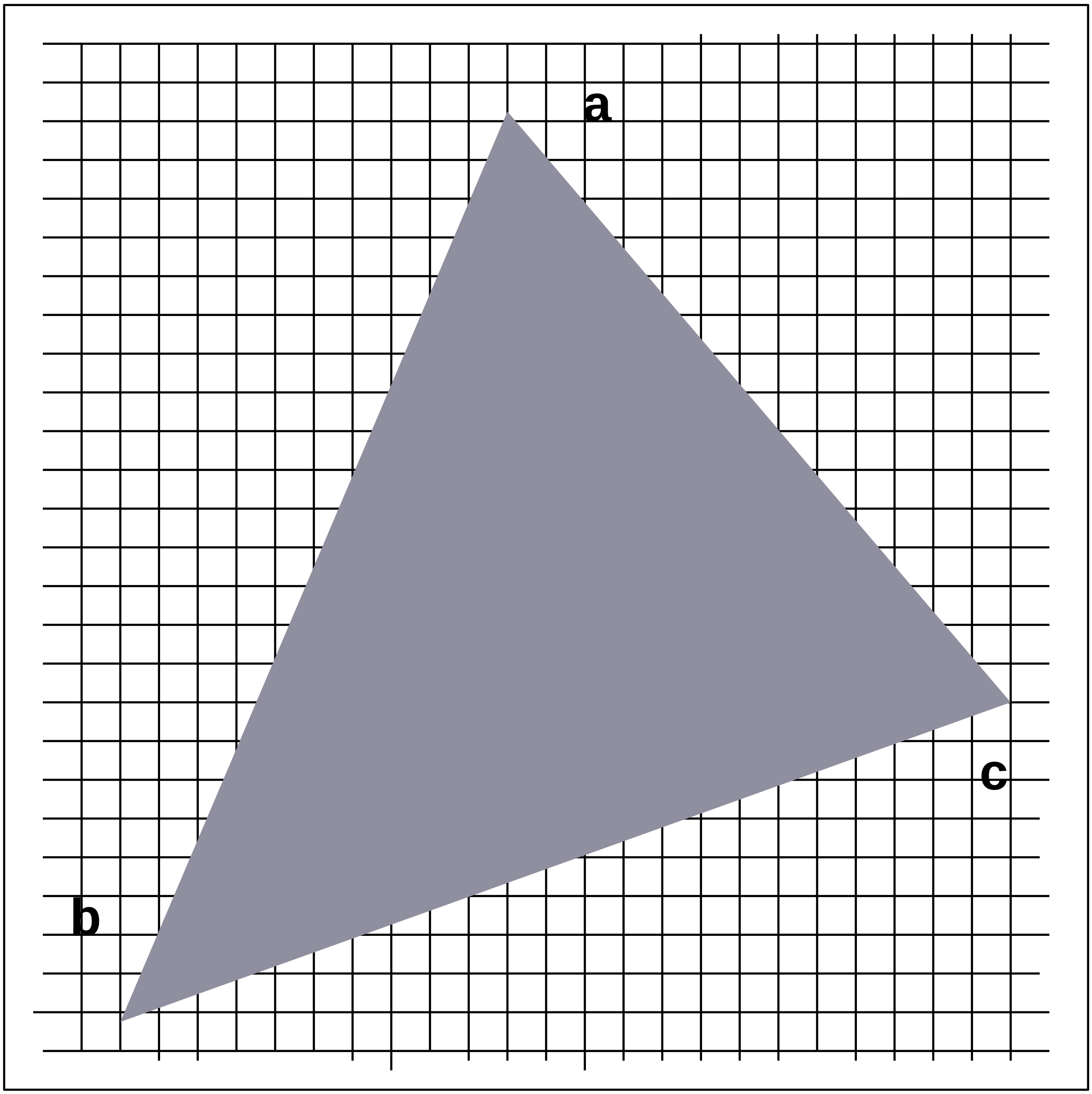
Scan Conversion



Simple Algorithms

- We start from a triangle T
  (x_1, y_1), (x_2, y_2), and (x_3, y_3)

- Find all pixels inside T

- Method 1 (the worst algorithm)
  For each pixel p do
  If $p \in T$ then draw-pixel (p) end if
  End for

- Method 2 (a slight improvement)
  $B = bounding - box(T)$
  For each pixel $p \in B$ do
  If $p \in T$ then draw-pixel (p) end if
  End for

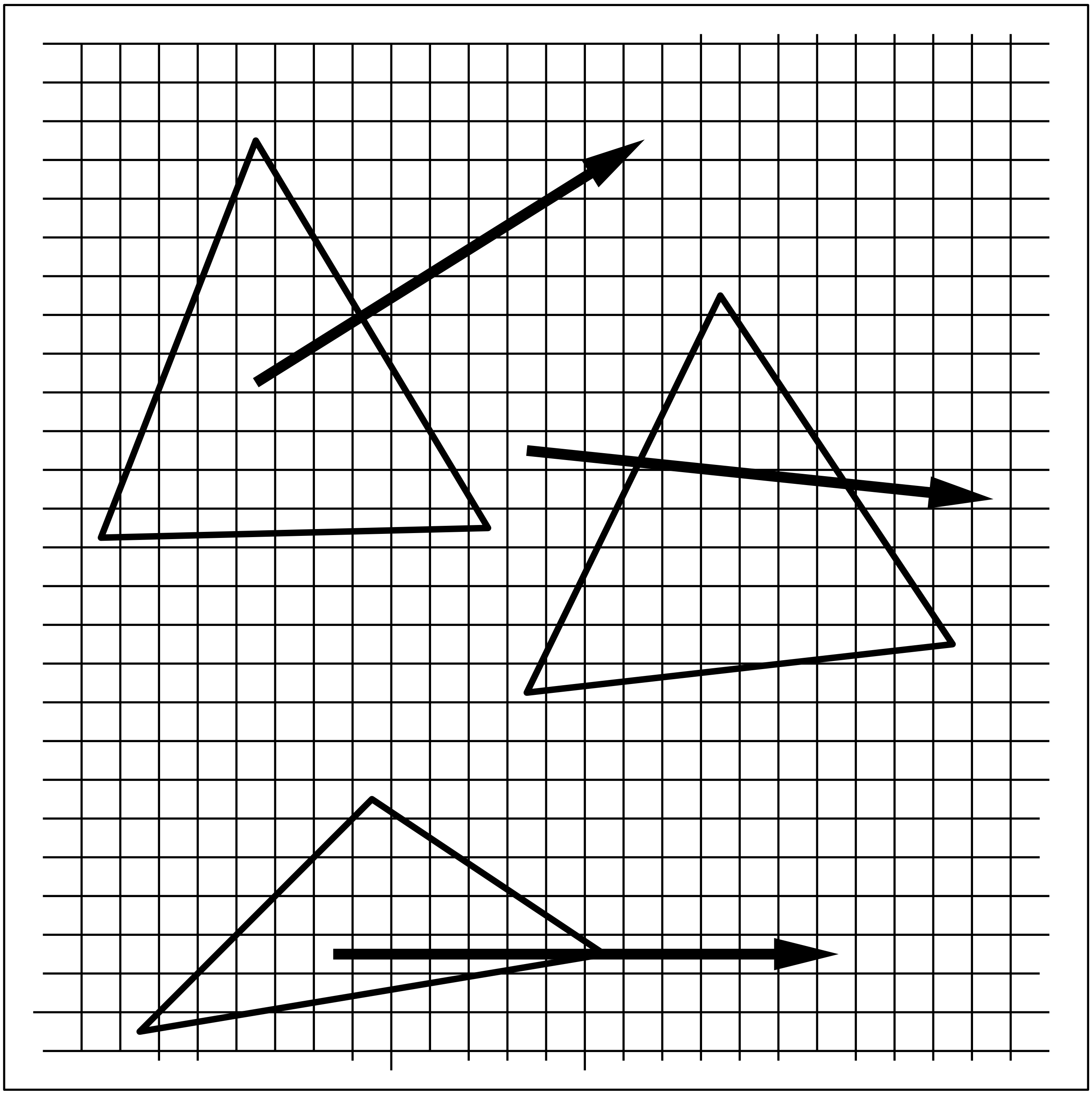
- The given previous algorithms suggest
  an important sub-problem:
  Given a triangle T, and $p = (p_x, p_y)$
  How to determine: $p \in T$

Ray Firing

- **Here's a simple approach to test if $p \in T$**
  (1) draw a ray from p outward in any direction
  (2) count number of intersections of this ray with boundaries of T
  (3) If odd, then $p \in T$, otherwise, p is not in T

- **Is this method correct?**
  What happens if the ray crosses at a vertex?

Polygon Scan Conversion



Implicit Line Formula

- A slightly easier method
- Consider the edge $v_1 v_2$
- Write down the implicit function of this line

$$l_{1,2}(x, y) = a_{1,2}x + b_{1,2}y + c_{1,2}$$

- Pick the sign of $l_{1,2}$ so that $l_{1,2}(x_3, y_3) < 0$
- This defines a half-plan $h_{1,2}$

$$h_{1,2} = \{(x, y) : l_{1,2}(x, y) <= 0\}$$

- Apply the similar process shown above to $l_{1,3}$ and $l_{2,3}$
- Construct half-planes $h_{1,3}$ and $h_{2,3}$
- The important observation

$$T = h_{1,2} \cap h_{1,3} \cap h_{2,3}$$

- **Therefore,** $p \in T$ **is equivalent to** $(p \in h_{1,2})$ **and** $(p \in h_{1,3})$ **and** $(p \in h_{2,3})$
- **It is the same to say**

$$l_{1,2}(p_x, p_y) <= 0$$

$$l_{1,3}(p_x, p_y) <= 0$$

$$l_{2,3}(p_x, p_y) <= 0$$

- **Question:** **does this algorithm work for concave polygon ?**

Sweep-line Algorithm

- Observation
  If $p \in T$, then neighboring pixels are probably
  in the triangle, too
  (Coherence)

- Idea
  (1) sweep from top to bottom
  (2) maintain intersections of T and sweep-line “span”
  (3) paint pixels in the span

- Algorithm
  Initialize x_l and x_r
  For each scan line covered by T do Paint pixels
  $(x_l, y), \ldots, \ldots, (x_r, y)$ on the current span
  Incrementally update x_l and x_r
  End for

- Question: how do we update x_l and x_r ?

- Answer: midpoint algorithm !

Polygon Scan Conversion

- Given a simple polygon P with vertices
  $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
  Find all pixels inside P

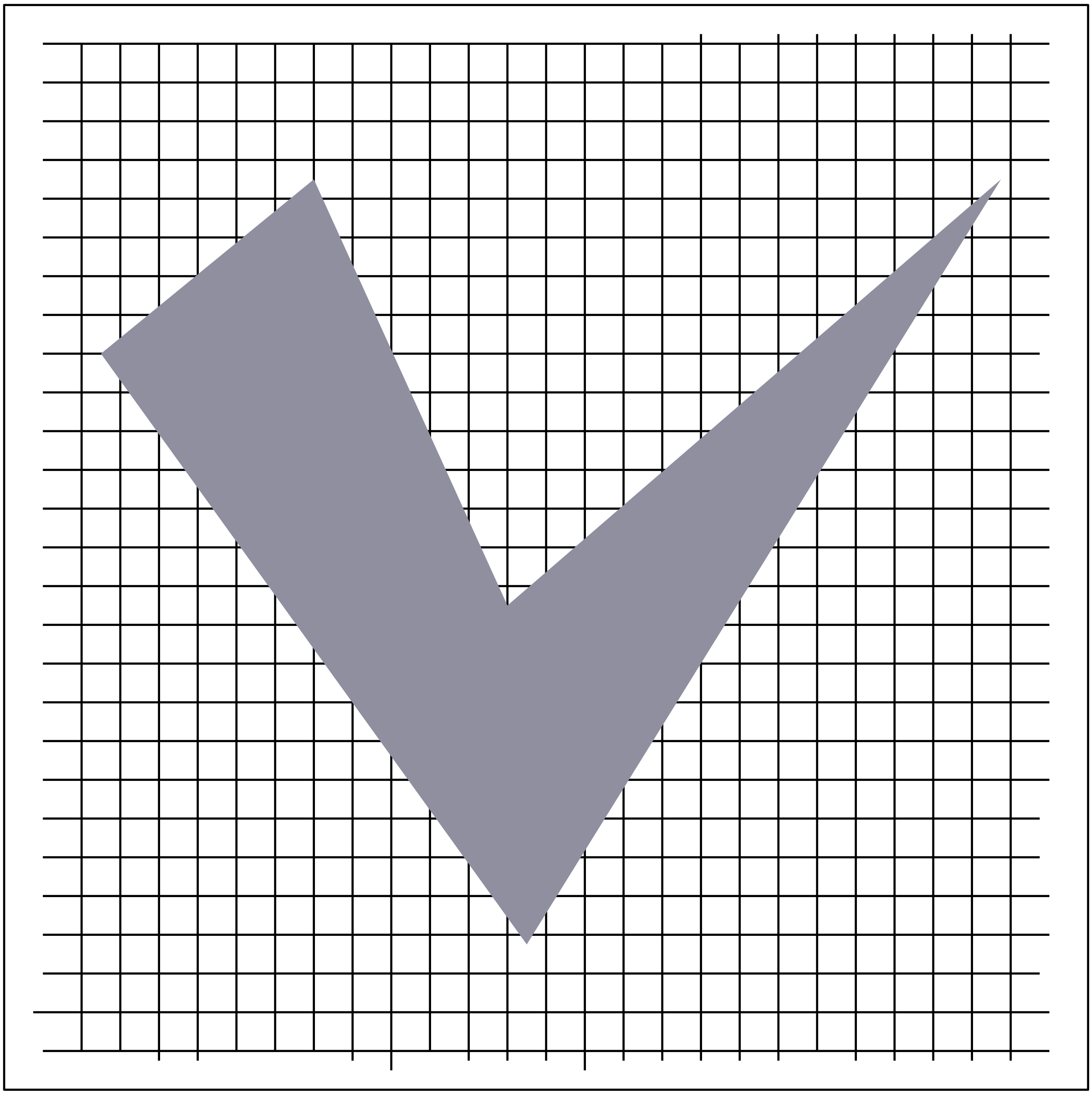
- Polygon classification
  simple convex
  simple concave
  non-simple (self-intersection)

- Once again, we could compute a bounding box and use ray casting
  $B = bounding - box(P)$
  For each pixel $p \in B$ do
  If $p \in P$ then paint (p) end if
  End for

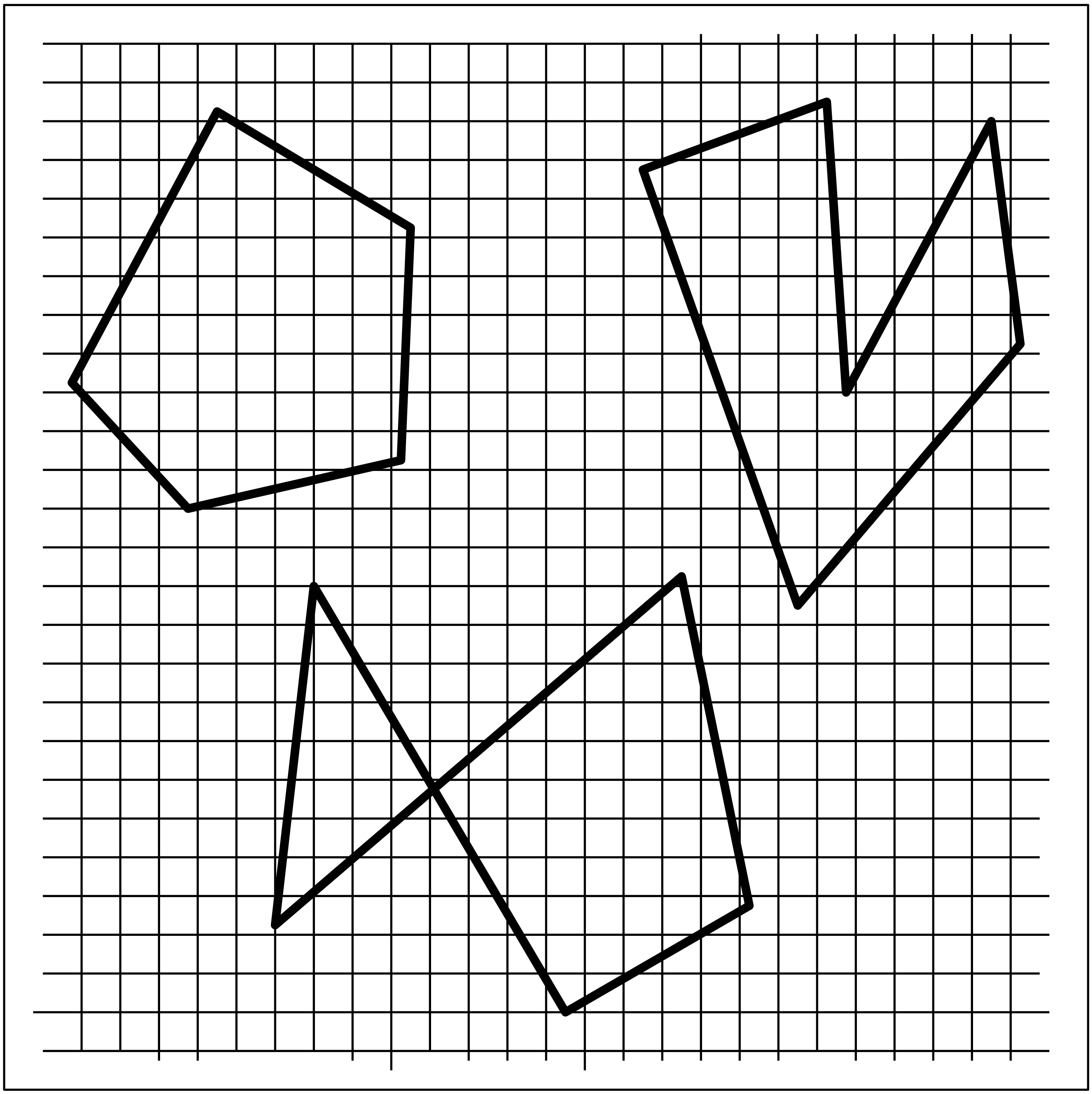
- But this would NOT take advantage of coherence

- Coherence
  Adjacent pixels in image space are likely sharing the similar graphic properties such as color

Polygon Scan Conversion



Polygon Classification



Scan Conversion

- More efficient algorithm
  For each scanline
  Identify all intersections $x_0, x_1, \ldots, x_{k-1}$
  Sort intersections from left to right
  Fill pixels between consecutive pairs of intersection

$$(x_{2i}, y), (x_{2i+1}, y)$$

- We must deal with "special cases" !
  - horizontal lines
  - intersecting a vertex (double intersection)
  - unwanted intersection

- We must speed up the edge intersection detection

- Data structure for efficient implementation
  A sorted edge table
  The active edge list
  From bottom to the top

Figure 3.39

- Practical polygon scan conversion
  Many implementations just triangulate the polygon and then convert the triangles

- Extremely easy to do for convex polygons

- Triangles are often particularly nice to work with because they are always planar and simple

Special Cases