Polygon area filling :—

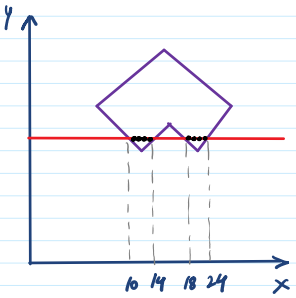
Examples with polygons since boundaries of polygon are linear.

① Scan-line Polygon Fill Algorithm

② Boundary Fill Algorithm.

Scan-Line Algorithm :—

By determining the overlap intervals for scan lines than crosses the area.

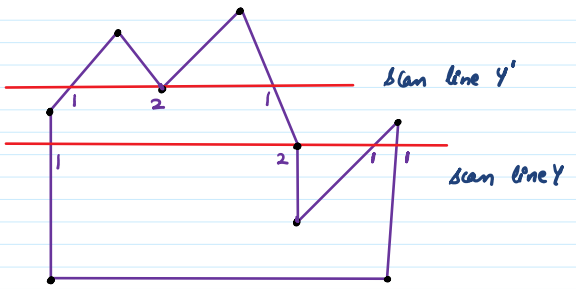


Interior pixel along a scan line passing through a polygon area.

① For each scan line crossing a polygon, algorithm locates the intersection points of the scan line with the polygon edges.

② Sort these intersection points from left-to-right.

③ For every pixel between these intersection points are set to specified fill color.



Scan line Y' generates odd no. of intersections.

Scan line Y generates even no. of intersections.

↳ Can be easily identified for pairs to fill the pixel along line Y.

⇒ Scan line intersection at polygon vertices require special handling.

→ The topological difference between scan line y and scan line y' is identified by noting the position of intersecting edges relative to the scan line.

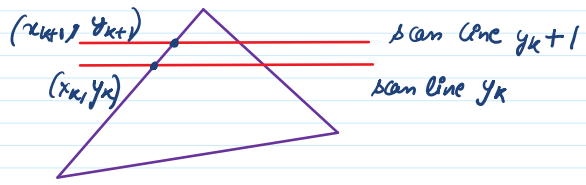
→ y → two intersecting edges sharing a vertex are on opposite sides of scan line
y' → two intersecting edges are both above the scan line.

Takes advantage of coherence properties of a scene.

Coherence Property :-

Properties of one part of a scene are related in some way to other parts of the scene so that the relationship can be used to reduce processing.

Involves incremental calculations applied along a single scan line or between successive scan lines.



Two successive scan lines intersecting a polygon boundary.

Slope of this polygon boundary line can be expressed as,

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

Since,

$$y_{k+1} - y_k = 1$$

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

Each successive x-intercept can thus be calculated by adding the inverse of the slope and rounding to the nearest integer.

Along an edge with slope m, the intersection x_k value for scan line k above the initial scan line can be calculated as

$$x_k = x_0 + \frac{k}{m}$$

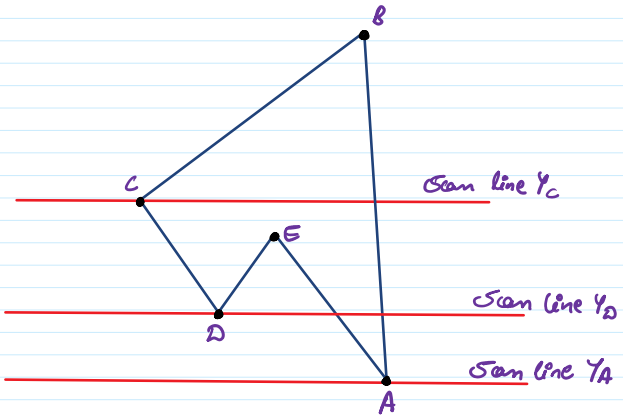
In a sequential fill algorithm, the increment of x values by the amount 1/m along an edge with integer operations as

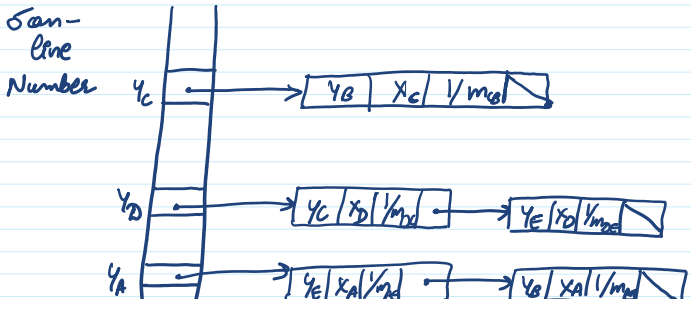
$$m = \frac{\Delta y}{\Delta x}$$

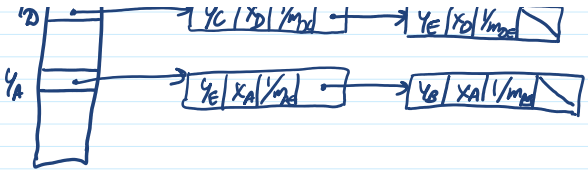
$\Rightarrow$ Incremental calculations of x-intercept becomes

$$x_{k+1} = x_k + \frac{\Delta x}{\Delta y}$$

$$= x_k + \frac{x_{k+1} - x_k}{y_{k+1} - y_k}$$







A polygon and its sorted edge table.

① Store the polygon boundary in a sorted edge table.

② Process the scan lines from bottom of the polygon to its top producing an an active edge list for each scan line crossing the polygon boundary.

③ The active edge list for a scan line contains all edges crossed by that scan line, with iterative coherence calculations used to obtain the edge intersections.

④ Fill from left-most x-intercept to one pixel before right-most x-intercept.

Boundary-Fill Algorithm :-

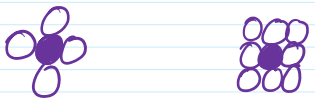
Start at a point inside a region and paint the interior outwards towards the boundary.

Inputs :-

① Coordinates of an interior point (x,y)

② Fill Color

③ Boundary Color.



(a) (b)

Fill methods applied to a
(a) 4-connected area.
(b) 8-connected area

Open circles are pixels to be tested from the current test position, shown as a solid color.

Fill algorithm :-

```
Boundary_fill4 (x, y, fill, boundary)
    {
        current = 1
        current = getpixel(x, y);
        if ((current != boundary && (current != fill))
            {
                setColor (fill);
                setpixel (x, y);
                Boundary-fill4 (x+1, y, fill, boundary);
                Boundary-fill4 (x-1, y, fill, boundary);
                Boundary-fill4 (x, y+1, fill, boundary);
                Boundary-fill4 (x, y-1, fill, boundary);
                    }
```

Flood-fill Algorithm :-

for area filling within multiple color boundries.

How?