Polygon Filling

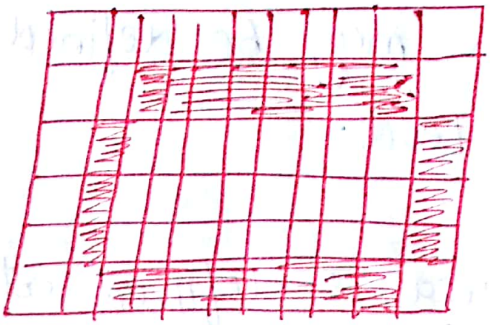
→ Process of coloring the area of polygon.

→ An area or region is defined as a collection of pixels.

→ Area/region are of 2 types:-

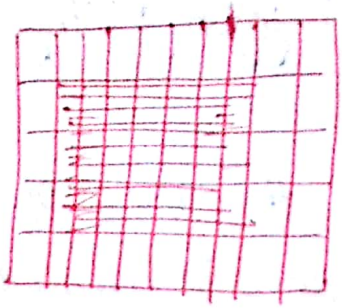
1) Boundary defined Region (BDR)

- Pixels that marks the boundary/outline of the region are called BDR pixels.
- Unique color not same as color of interior pixels.
- Algorithms that are used to fill the boundary defined region are called as Boundary fill alg.



2) Interior Defined Region

- Collection of same color continuous pixels.
- Pixels exterior to a region have different colors.
- Algorithms used to fill interior defined region are called Interior/Flood fill algorithms.



Both these algorithms are together called as Seed-fill algorithms.

(Because both start from a pixel within a region)
(Pixel where you start)

→ Principle of seed fill algorithms.

"start at a sample pixel called as seed pixel from the area, fill color value and the boundary color value".

→ Area/Region may be defined at pixel level or Geometric Level.

→ When the area is defined at pixel level, we have 4 different algorithms:-

1) Boundary fill
2) Flood "
3) Edge "
4) Fence "

→ When defined a geometric level know as Scanline Approach.

Boundary fill Algorithms

→ Recursive algorithm, begins with starting point inside the region. (seed)

→ Algo checks whether this is a boundary pixel or has been filled. already

→ If answer is No, it fills pixel and make recursive call to itself using each & every neighbouring pixel as a new seed.

→ If answer is YES, algo. simply return to caller.

Adv:- works well for arbitary shaped region

DisAd:- Takes time & memory beco2 of so many recursive calls.

```
Void boundaryFill (intx, inty, intfill, intboundary)
    {
      int current = getpixel(x,y)
      if (current != boundary && current != Fill)
        {
          setcolor(fill);
          setpixel(x,y);
      boundaryfill (x+1, y, fill, Boundary)
                   (x-1, y,  ,,    ,,  )
      }; (x, y-1, ,, ,,) (x, y+1,  ,,   ,,  )
```

Flood fill Algorithm

→ Begins with seed inside the region.
→ checks, if pixel has region's original color

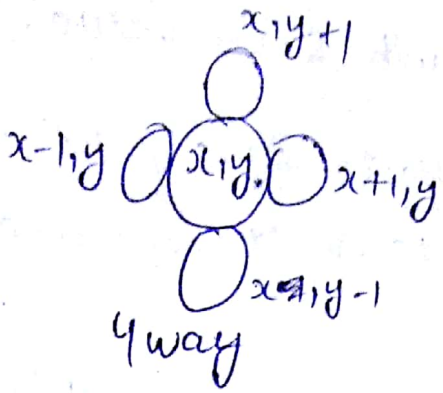
If yes :- it fills pixel with a new color & uses each of pixels neighbour as a new seed in a recursive call.

If No:- it return to caller.

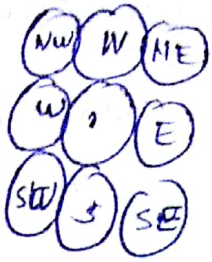
Ad:- useful if region has non-uniform coloured boundary.

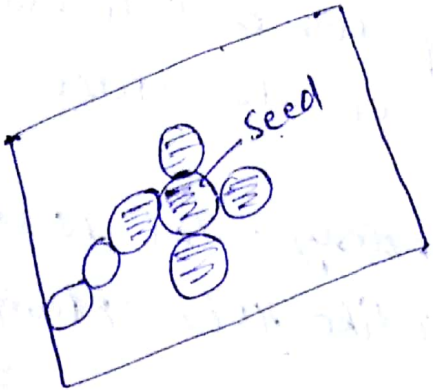
DisAd:- same as Boundary fill.

Boundary filling



4 way





Boundary-fill (x, y, f-color, b-color)
(seed coordinates, new, background (0) black)

[

if (get pixel (x,y) != b_color && get pixel (x,y) != f-color)

[

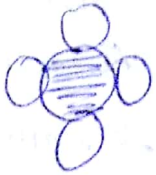
Put Pixel (x,y f-color)

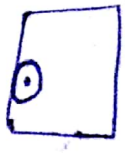
Boundary-fill (x+1, y, f-color, b-color)

" (x, y+1, " ")

" (x-1, y, " ")

" (x, y-1, " ").





Flood-fill

→ we start with seed value.

→ then we inspects all 8 points surrounding the seed. If these points aren't on boundary, each will be filled.

→ Now, each of these points become seed. The algo continues to work in this fashion until all points successively all seeds enter into boundary.

↳ Some cases where boundary color is diff than fill color. For situations like this Flood fill is used.

↳ Here instead of matching color with a boundary color, a specified color is matched.

→ flood-fill (x, y, old-color, new-color)

{

putpixel (x, y, new-color)

flood fill (x+1, y, old-c, nc)

(x-1, y,

(x, y+1

(x, y-1

---

x+1, y+1

x-1, y-1

x+1, y-1

x-1, y+1

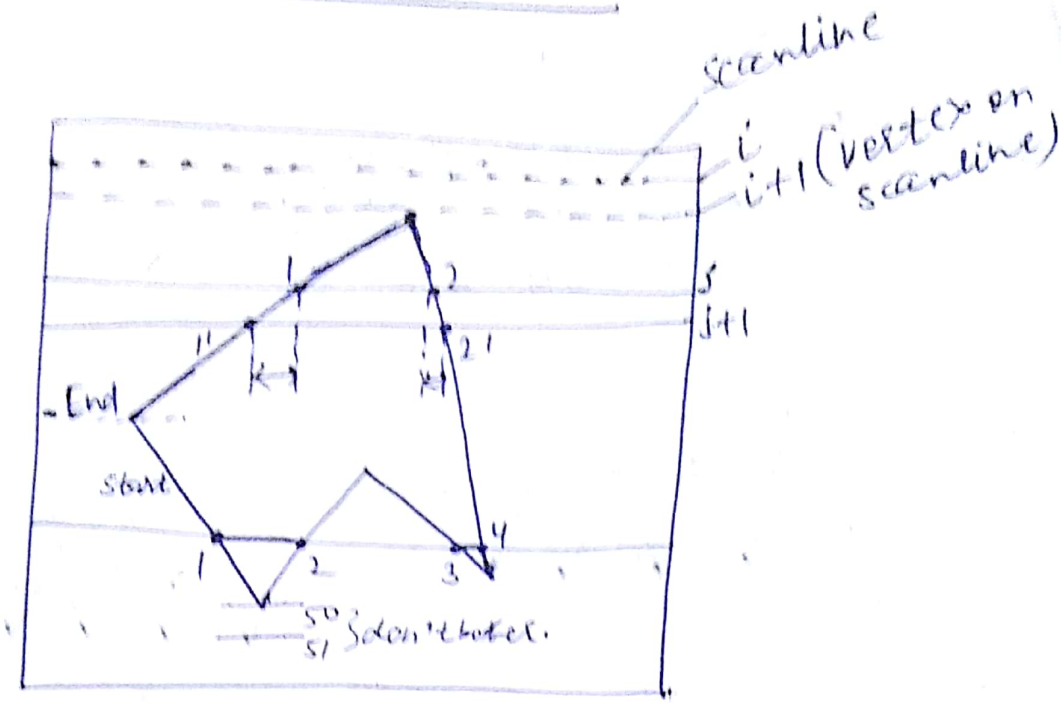
}

}

→ doesn't work for large polygons.

→ If an inside pixel is in some other color then fill terminates & polygon remain unfilled.

Scan Conversion



→ Pixels within the polygon are displayed

→ Row of pixels are called Scanline.

→ All pixel b/w 1-2 are displayed
" " " 1&2 / 3-4 " "

→ moving from ISL to another SL.

→ 1 ON 11 modified
2 — 21

→ Point 1 is modified by the slope of line

→ when we move from ISL to next the position of edge should be an amt $= \frac{dy}{dx} = \frac{\Delta y}{\Delta x}$.

From ISL to another :-

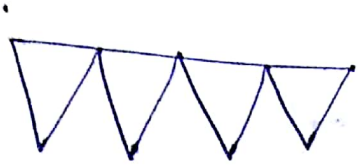
1) Intersection pt. change by $\Delta y/\Delta x$
2) edges can start / end.

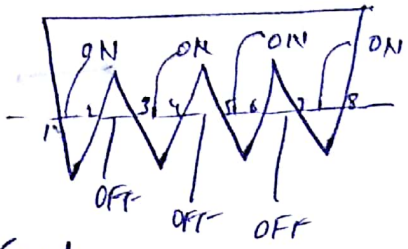
Intersection Points.

$I_1, I_2 \dots I_n$

B/w 1 & 2 — ON — Pixels.
2 & 3 — OFF
3 & 4 — ON

→





→ Every scanline keep track of intersection pts.

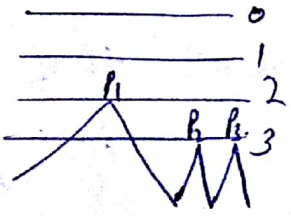
Algorithm

Given :- A polygon as a list of points.

TODO :- To fill the polygon.

Algo :-

1. Sort vertices as per y-coordinate and place them in bucket for that co-ordinate.



Buc[i] →

B[0] → - -

B[1] → - -

B[2] — P_1 (vertices).

B[3] — P_2, P_3

If y coordinate is i

we keep track of Buc[i].

A polygon defined by a set of vertices $v_1 - v_n$ and color of polygon color.

for all polygon edges do

calculate $\frac{\Delta x}{\Delta y}$ for each edge.

end for

for all scan lines in polygon (top-bottom) do

for all polygon edges do

if edge intersects with scanline then

cal. x co-ordinate of scan extrema

end if

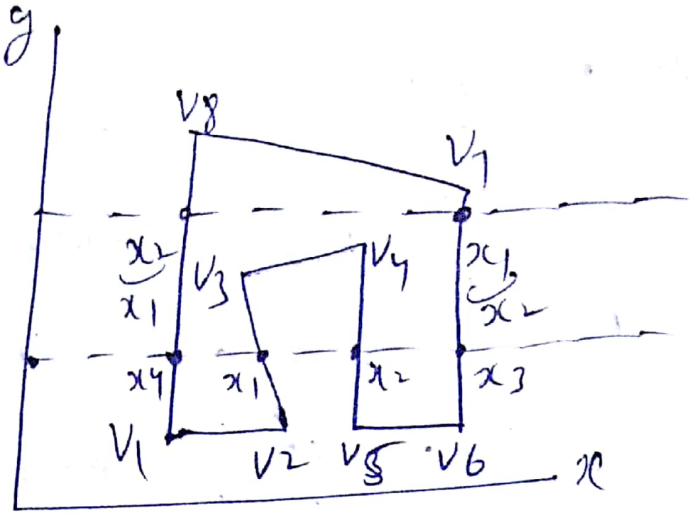
end for

sort the scan extrema in ascending order

illuminate pixels s/w scan extrema pairs

end for.

- A scanline is a horizontal row of pixels.
- The algo loops through row of pixel starting at top of polygon.
- Points where polygon edge intersect with scanline are called scan extrema.
- If an edge intersects with scaline x-coordinate of scan extrema are called cal. as its I.P.



$x_4 > x_2$

edge $V_1 - V_2$ doesn't intesect scanline.

$V_2 — V_3$

$V_3 — V_4$))

$V_4 — V_5$

$V_5 - V_6$

$V_6 - V_7$ — intesects scanline at x_1

$V_7 - V_8$ — no.

$V_8 - V_1$ — " " " x_2

→ Scan extents sorted in ascending order.

→ $V_1 \rightarrow V_2$

$V_2 - V_3 - x_1$

$V_3 - V_4$

$V_4 - V_5 - x_2$

$V_5 - V_6$

$V_6 - V_7 - x_3$

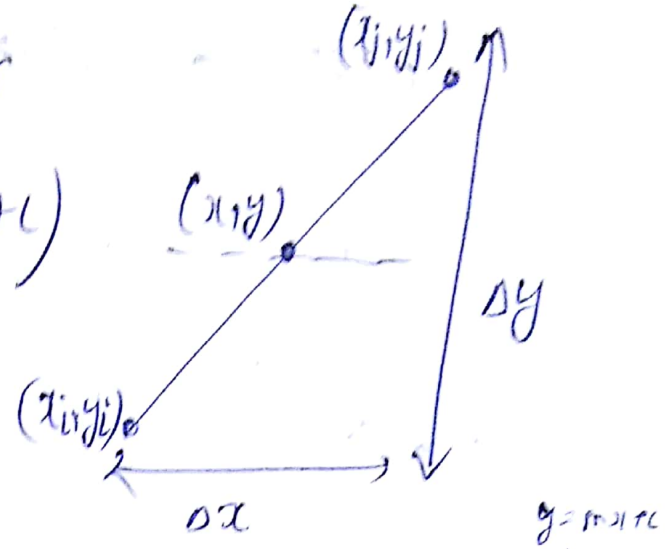
$V_7 - V_8$

$V_8 - V_8 - x_4$

Linear Interpolation

→ cartesian eq. of line is

$$y = \frac{\Delta y}{\Delta x} x + c \quad (y = mx + c)$$



given (x_i, y_i) is on line then

$$c = y_i - \frac{\Delta y}{\Delta x} x_i$$

→ Substitute c into eq. of a straight line.

$$\boxed{x = x_i + \frac{\Delta x}{\Delta y}(y - y_i)}$$

Scan extrema

Co-ordinate of scan extrema are:

$$y = y - 1$$

$$x = x_i + \frac{\Delta x}{\Delta y}(y - y_i)$$

$\frac{\Delta x}{\Delta y}$ = constant along edge, ∴ it can be pre-cal.

If an edge is horizontal $\Delta y = 0$

$$m = \begin{cases} \frac{\Delta x}{\Delta y} & \text{if } \Delta y \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

→ Scan explanation sorted in ascending order of (x) and all pixels b/w pairs of scan explanation are filled.