23 Jan, 19

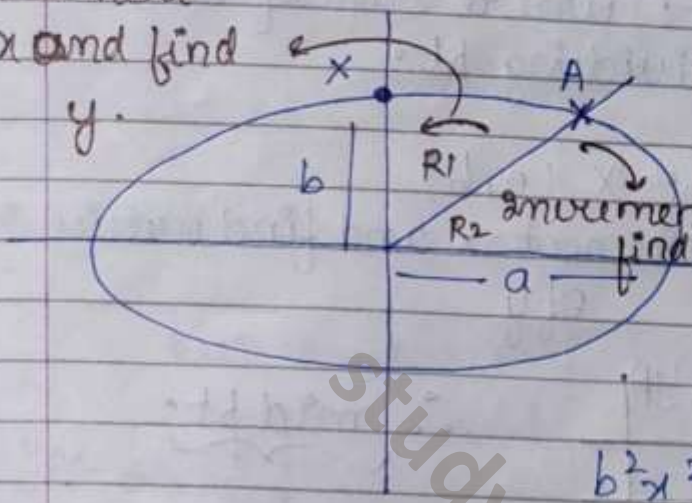
CLASSTIME | Page No.
Date

→ Symmetric figure. In 4 parts.

Ellipse Drawing Algorithm

through mid pt. drawing algorithm.

Increment x and find y.



major and minor axis decide the eqn ↓

$$f(x,y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$b^2x^2 + a^2y^2 - a^2b^2 = 0$$

Along I quadrant, the slope is varying in ellipse.

At pt. A ⟶ slope of tangent is -1 and gradient is 1 ($\perp$)

⟶ The deciding factor for dividing into regions is by finding gradient.
At pt. where gradient = 1, then we move from R_1 to R_2 i.e sloped of tangent is opp.

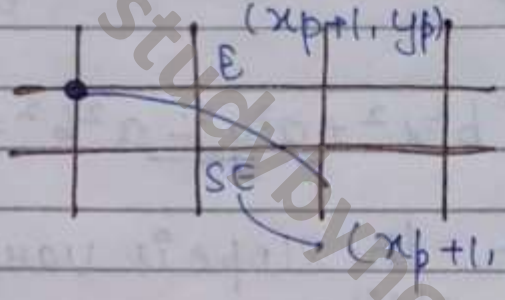
- Gradient ↓ (partial derivative wrt x + partial derivative wrt y).

$$f(x,y) = \frac{\partial f}{\partial x}\hat{i} + \frac{\partial f}{\partial y}\hat{j}$$

$$= 2b^2x\hat{i} + 2a^2y\hat{j}$$

→ In region 1, the j component of the gradient is greater / larger than the i component. Both the components are equal at A i.e at pt. where slope of curve is -1 or the dividing pt.

- We start from pt. X (0, b)
  Thus, we increment x and find values of y.



∴ mid pt.

$x_{p+1}, y_p - 1/2$.

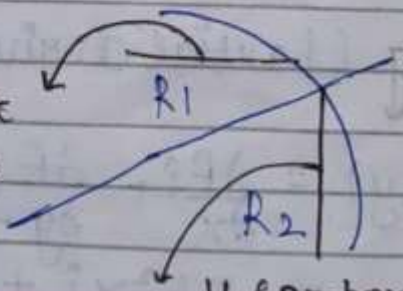
We keep on checking this value ensuring j component is greater than i component. The moment we get $j < i$ we move to Region 2.

$a^2(y_p - 1)^2 \leq b^2(x_p + 1)$

deciding factor how to move from R_1 to R_2.

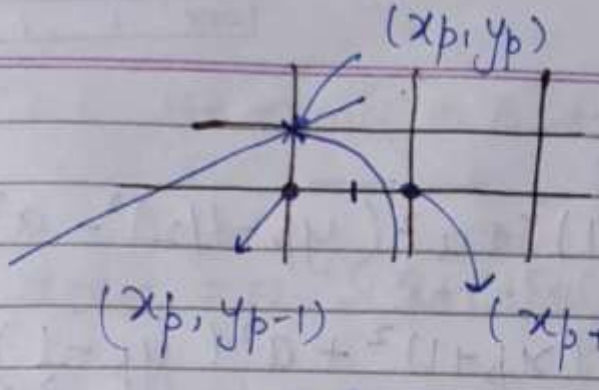
j component i component.

Now for region 2 we decrement y and find all values of x



x component has more pts.

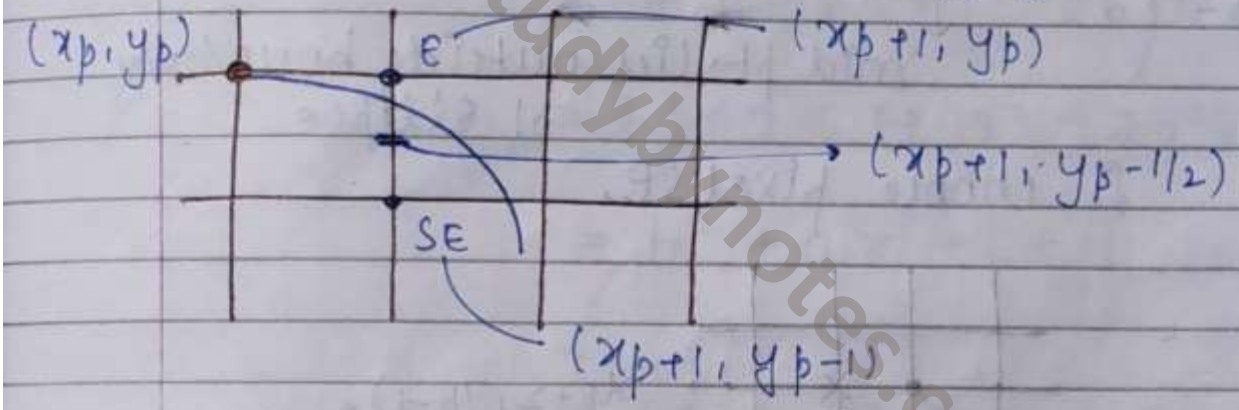
y component has more pts.



mid pt → $(x_p + 1/2, y_p - 1)$

→ Now we need to find initial decision variable. (d_o)

but derivations are diff. for R_1 and R_2.

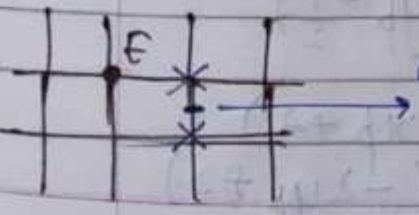


$\therefore d_{old} = f(x_p + 1, y_p - 1/2)$

Now $d_{old} < 0$ then mid pt. lies inside the boundary of ellipse.

$\therefore$ Boundary is closer to pixel E and thus choose pt. E

If E is chosen ↓



$d_{new} = f(x_p + 2, y_p - 1/2)$

$= b^2(x_p+2)^2 + a^2(y_p - 1/2)^2$

$-a^2b^2 = 0$

$b^2(x_p+1+1)^2 + a^2(y_p-1/2)^2 - a^2b^2$

$= b^2(x_p+1)^2 + a^2(y_p-\frac{1}{2})^2 -$

$a^2b^2 + b^2(2x_p+3)$

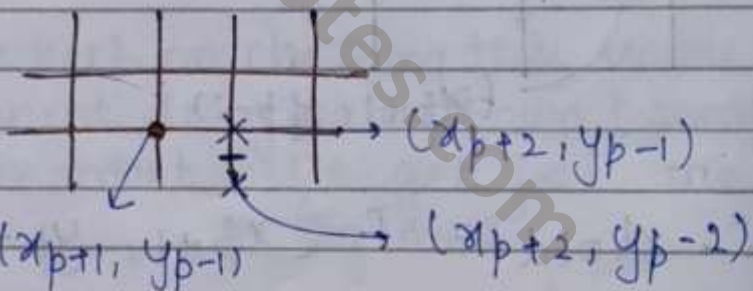
$= d_{old} + \Delta E$ — extra part

If $d_{old} > 0$

↓

mid pt. lies outside boundary of ellipse

∴ choose pixel SE.



∴ mid pt → $(x_p+2, y_p-3/2)$

$d_{new} = F(x_p+2, y_p-3/2)$

$= b^2(x_p+2)^2 + a^2(y_p-3/2)^2$

$= b^2(x_p+1)^2 + a^2(y_p-\frac{1}{2})^2$

$-a^2b^2 + b^2(2x_p+3) +$

$+ a^2(-2y_p+2)$

= dold + ΔSE ——— (2).

dold = 0 (Initial deciding factor).

Pt. known (0, b)

(for region 1) → $d_0 = F(0+1, b-\frac{1}{2})$

$= b^2(1)^2 + a^2(b-\frac{1}{2})^2 - a^2b^2$

$= b^2 + a^2(b^2 + \frac{1}{4} - 2b) - a^2b^2$

$= b^2 + a^2b^2 + \frac{a^2}{4} - 2a^2b - a^2b^2$

$= b^2 + a^2(-b + \frac{1}{4})$ ——— (3)

Now we find whether this value is

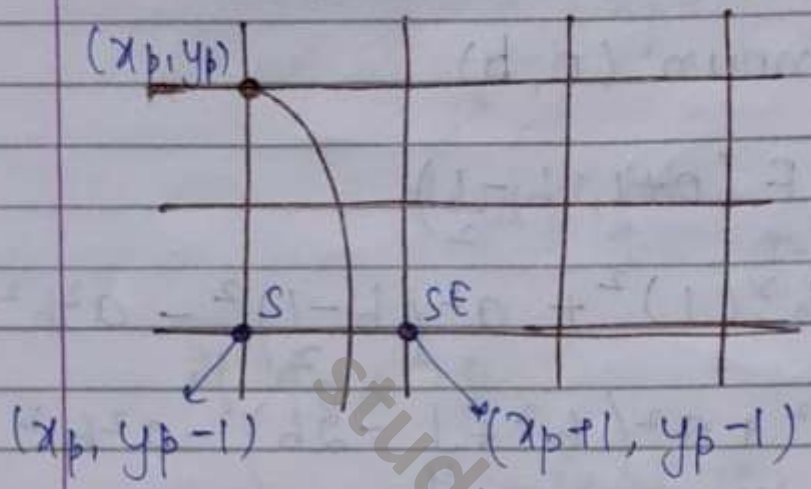
< 0
add ΔE

> 0
add ΔSE
pixel plotted will be
(x_p+1, y_p-1)

∴ plotting of 4 pixels.
(4 quadrants)

Now for region 2 (Slope is in -ve dirⁿ)

We will decrement y and find value of x.



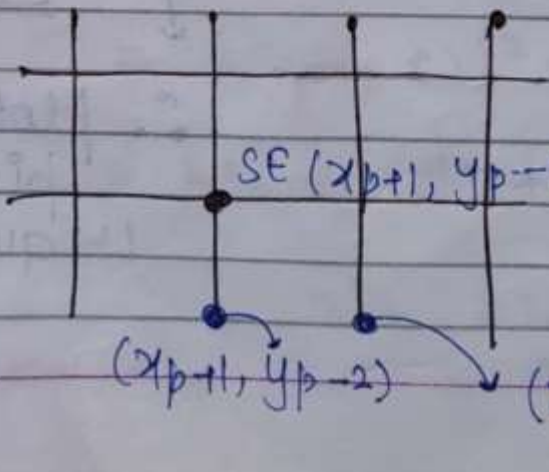
∴ midpt → $(x_p+1/2, y_p-1)$

$d_{old} = f(x_p+\frac{1}{2}, y_p-1)$

$= b^2\left(x_p+\frac{1}{2}\right)^2 + a^2(y_p-1)^2 - a^2b^2$

If $d_{old} < 0$, choose SE as mid pt. lies inside boundary & boundary is closer to SE

If we choose SE



mid pt → $x_p + \frac{3}{2}, y_p - 2$

$d_{new} = F\left(x_p + \frac{3}{2}, y_p - 2\right)$

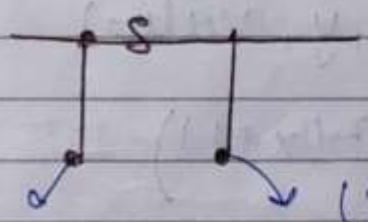
$= b^2\left(x_p + \frac{3}{2}\right)^2 + a^2(y_p - 2)^2 - a^2b^2$

$= b^2\left(x_p + \frac{1}{2}\right)^2 + b^2(+2x_p + 2) + a^2(y_p - 1)^2 + a^2(-2y_p + 3)$

$= d_{old} + \underbrace{b^2(2x_p + 2) + a^2(-2y_p + 3)}_{\Delta SE}$

(4)

If $d_{old} > 0$ choose $S(x_p, y_p - 1)$



midpt → $\left(x_p + \frac{1}{2}, y_p - 2\right)$

$d_{new} = F\left(x_p + \frac{1}{2}, y_p - 2\right)$

$= b^2\left(x_p + \frac{1}{2}\right)^2 + a^2(y_p - 2)^2 - a^2b^2.$

$= b^2\left(x_p + \frac{1}{2}\right)^2 + a^2(y_p - 1)^2 + \underbrace{a^2(-2y_p + 3)}_{\Delta S}$

(5)

d_o for region 2

↓

$(x_p + 1/2,\ y_p - 1)$

Where the region 1 finishes, the last pt. $\left(x_p + \frac{1}{2}, y_p - 1\right)$ will give us the d_o for region 2.

$d_o = F\left(x + \frac{1}{2}, y - 1\right)$ where (x, y) is last pt. of Region 1.

* ~~Algorithm~~ ↓

$x = 0,\ y = b$

$d_o = d_{OR1} = b^2 - a^2 b + \frac{a^2}{4}$

Plot ellipse pts. $(x, y$ value)

while $\left(a^2\left(y - \frac{1}{2}\right) > b^2(x+1)\right)$

{

if $d < 0$ // choose E

$d_o += b^2(2x+3)$

else

{

$d_o += b^2(2x+3) + a^2(-2y+3)$; // select SE.

$y--$;

}

x++;
plot ellipse pts.
}

last x and y values from 1st while loop.

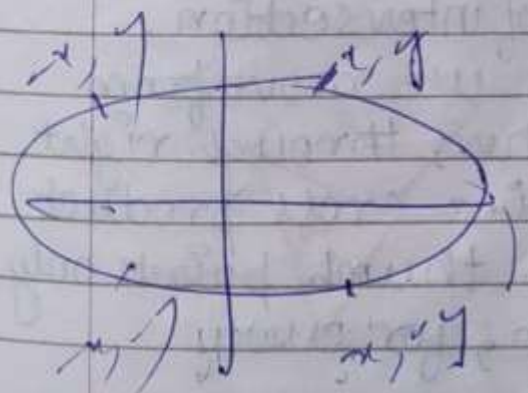
$d_0 = d_{OR2} = b^2\left(x+\frac{1}{2}\right)^2 + a^2(y-1)^2 - a^2b^2$

while (y > 0)
{
if ($d_0 < 0$)
{
$d_0 += b^2(2x+2) + a^2(-2y+3)$;
x++;
}
else
{ $d_0 += a^2(-2y+3)$;
y--; }
plot ellipse ( );
}
}

at y = 0 we have touched the x axis and thus now we are in R2.

NOTE:- Take all values of variables as double.

as there is problem of precision.



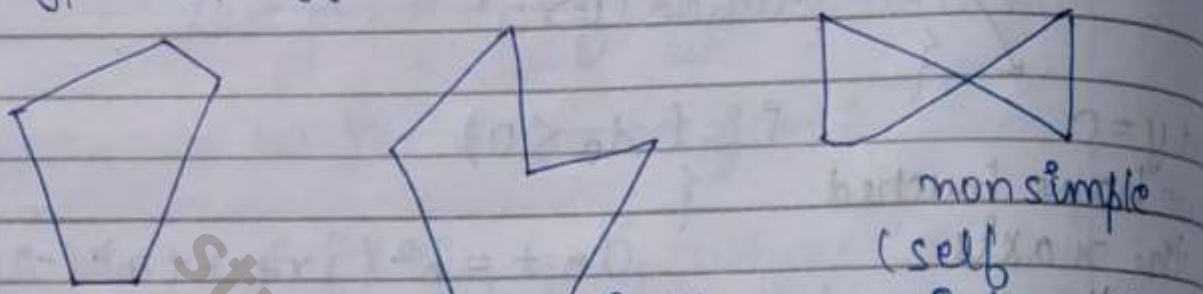
28 Jan 19

Filling Polygons

↓

Filling colours in polygons.

3 types of polygons



Simple convex

Simple concave

non simple (self intersection)

On a system, we don't first make outline and then fill colours. But filling and drawing goes hand in hand
∴ colouring is done with scan conversions.

all algos of DDA, Bresenhams
(The name is so because we scan each line i.e scan line).

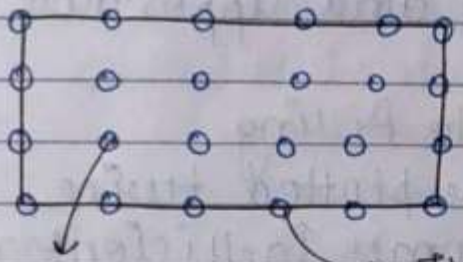
- Convex polygon - only 2 pts. of intersection. For a given edge vector, we move from one edge to another, through right hand thumb rule i.e cross product of 2 edge vectors, then thumb points only in one direction, for every edge vector.

- Concave polygon

If polygon is concave, the part where concavity starts, the z component comes out to be -ve.
And for a concave polygon, we can have multiple pts. of intersection.
Also, acc. to thumb rule, thumb doesn't always points in one direction.

NOTE- Filling polygon algo. are only based for convex polygons.
For concave polygons, we divide them into diff. convex polygons.

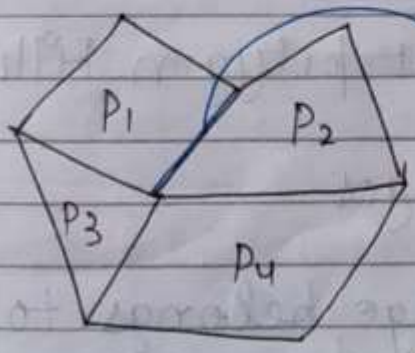
What pixels to be considered ↓.



These pixels lie inside polygon

They lie on the boundary.

Now we need to decide whether these pts. must be painted or not.

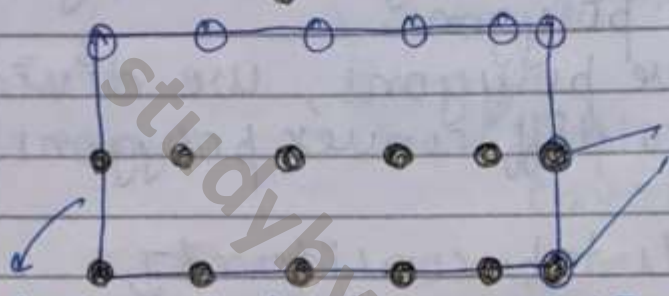


This is a shared edge/ Shared pixels. of P_1 and P_2
If P_1 is drawn first these pixels get red color or else yellow if P_2 is plotted.

To get rid of this problem, we need to have some common criteria.

So, for shared pixels, we must have some method.

For adjacent polygon figures



plot the left and bottommost pixels and leave the right and uppermost pixels.

→ Disadvantages of Double Filling

(1) If shared pixels are plotted twice, algo becomes inefficient

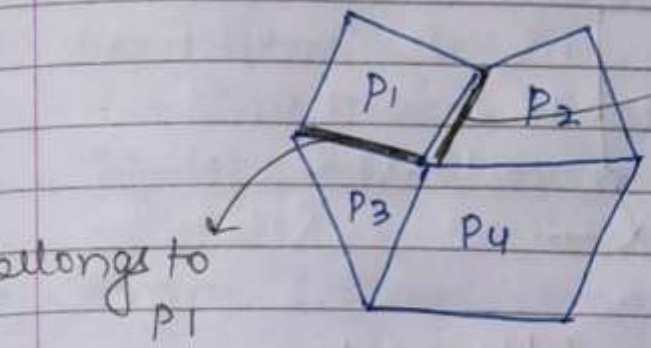
(2) If polygon have diff color, then final color depends on the order in which polygon is drawn.

→ General Ideas about polygon Filling

• Rules for Shared edges

- A shared vertical edge belongs to the rightmost of the two sharing shapes

- A shared non vertical edge belongs to upper shape.



(non vertical edge)

- Fill in polygons by computing intersections of boundaries with scan lines.

→ Edges of polygon are to be stored in form of vertices.

- Span
- coherence
- Spatial coherence
- edge coherence

{collection of adjacent pixels on single scan line which lie inside primitive}
(Span)

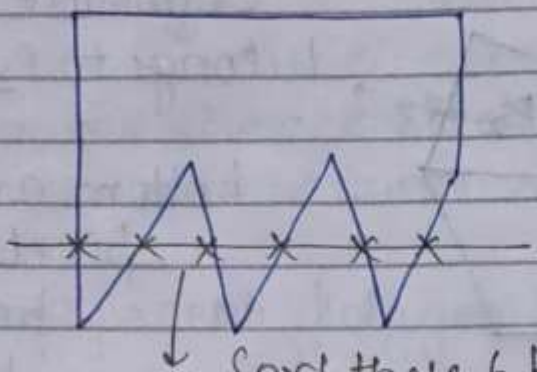
Spatially coherent (as they will be getting same colour)

→ The algo stops when y attains its max. value.

Algorithm -

(1) Find intersection of scan line with all edges of polygon. (min=0 and max=∞)

2. Sort intersections by increasing x coordinates.



Sort these 6 pts acc to ascending order.

3. Fill in all pixels b/w pairs of intersections that lie in interior to polygon.

→ How do we find and sort intersections.

→ How to find whether a pt. is inside or outside

imp. Parity (odd-Even) Rule (6-7 marks)

Begin from a pt. outside the polygon, counting no. of edges crossed so far, a pixel is inside polygon if no. of edges crossed so far (parity) is odd and outside if no. of edges crossed is even. This is known as odd-even parity rule.

Q Dry-run the algo given initial & final pts. (Circle, line, ellipse)
Make table. (6-7 marks)

CLASSTIME | Page No.
Date / /

Q How to deal with special case of intersection at integer pixel coordinates

↓

→ Use criteria we are using for avoiding conflicting b/w shared edges

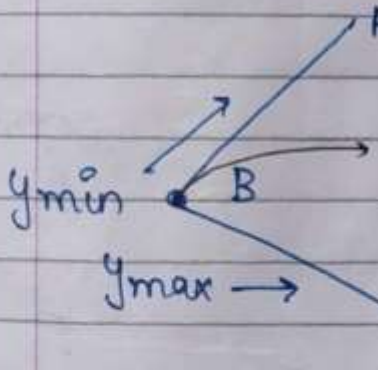
~~If upmost pixel in a span has~~

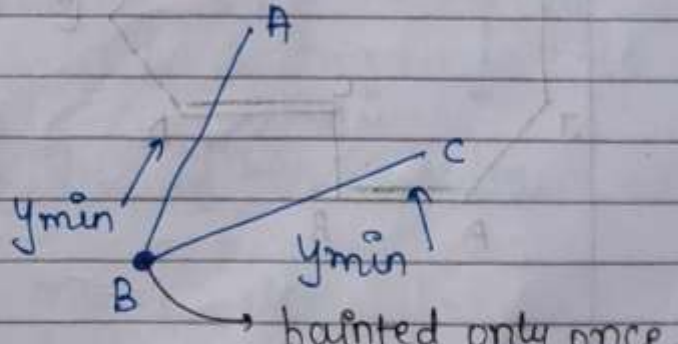
→ draw leftmost and bottommost pixels.

→ How to deal with special case of shared vertices

↓

count ymin vertex of an edge in parity calculations but not ymax vertex.

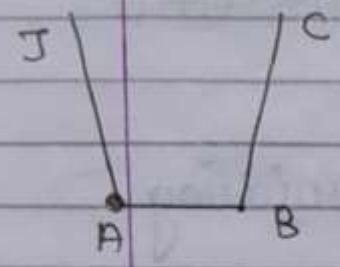




→ Bottom edges are drawn but not top edges.

{ Ymin - contribute to parity
Ymax - no contribution of parity. }

CLASSTIME | Page No.
Date / /



A is ymin for AJ ∴ parity becomes odd from initial 0 parity. Now we encounter line AB, which is a horizontal line and no horizontal line, we have no concept of ymin and ymax ∴ horizontal line doesn't contribute to parity rule.

A to B → B is ymin for BC ∴ parity changes from odd to even.
∴ do not print B.

NOTE:- Parity will only change for a particular scan line with ymin as intersection or any other pixel except ymax.