Scan Conversion.

9.30 Am
14-07-2018.

Pixel :- (Pel, Picture element)

In digital image processing, pixel is a physical point in an image.

- Pixel is the smallest addressable and controllable element of a picture represented on the screen.

Rasterisation / Scan Conversion :-

Rasterisation is the task of taking an image described in vector graphics format (shapes) and converting it into a raster image (pixel or dots) for output on video display or printer.

It is also known as scan conversion.

Suppose (x, y) is a mathematical point and it is scan converted to a pixel location (x', y'). where x' is the integer part of x and y' is the integer part of y.

Here, $x' = \text{floor}(x)$ and

$y' = \text{floor}(y)$.

Scanned by CamScanner

** Describe DDA Line drawing algorithm.

Digital differential Analyzer is an incremental scan conversion method. It is characterized by performing calculation at each step using result from previous step.

Now at step i point is (x_i, y_i) and in the next step point is (x_{i+1}, y_{i+1}).

Now, $m = \frac{\Delta y}{\Delta x} = \frac{\cancel{y_2 - y_1}\ y_{i+1} - y_i}{x_{i+1} - x_i}$

For $m \leq 1$,

$y_{i+1} - y_i = m(x_{i+1} - x_i)$

$\Rightarrow y_{i+1} = y_i + m(\Delta x)$ $[\Delta x = 1]$

$\therefore y_{i+1} = y_i + m$.

For $m > 1$.

$\cancel{y_{i+1} - y_i}$

$x_{i+1} - x_i = \frac{\Delta y_{i+1} - y_i}{m}$

$\Rightarrow x_{i+1} = x_i + \frac{\Delta y}{m}$ $[\Delta y = 1]$.

$\Rightarrow x_{i+1} = x_i + \frac{1}{m}$

This process continues until x reaches x_2' (for $|m| \leq 1$) and y reaches y_2' for $(|m| > 1)$ and all points found are scan converted into pixel co-ordinate.

Scanned by CamScanner

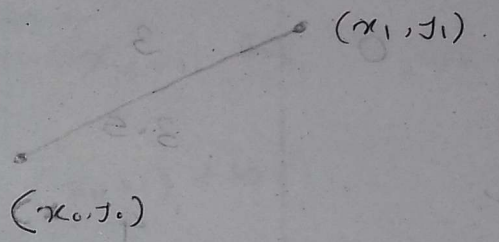
Merits :-

1. Simple line drawing algorithm.
2. Faster than direct use of line equation.

Demerits :-

1. Not very efficient.
2. Floating point operations and rounding operations are expensive.
3. Cumulative round off error may changes the line.

Pseudocode :-




```
Compute m;
if (m ≤ 1) {
    float y = y0;
    for ( x = x0 ; x ≤ x1 ; x++) {
        y = ~~floor~~ round (y + m).
        setpixel (x, y);
    }
else {
    float x = x0;
    for (y = y0 ; y ≤ y1 ; y++) {
        x = round (x + m);
        setpixel (x, y);
    }
}
```

Scanned by CamScanner

** Draw a line starting pixel (3,2) and ending pixel (8,12).

Here, $m = \frac{y_2 - y_1}{x_2 - x_1}$

$m = \frac{12-2}{8-3}$

$= 2.$

Ans, $|m| > 1$

So, $y_{i+1} = y_i + 1$

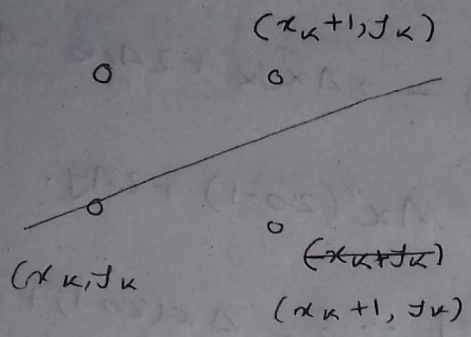
$x_{i+1} = x_i + 1/m$. $(\Delta y = 1)$

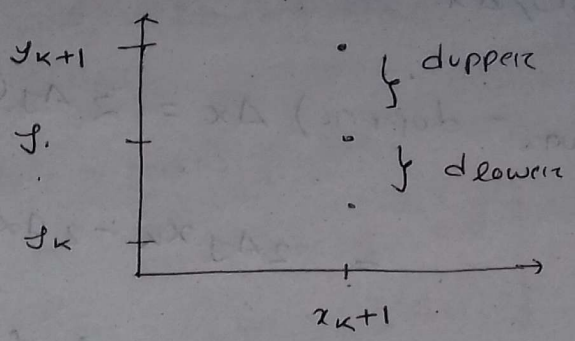
t	x	y	R(x)	R(y)
0	3	2	3	2
	3.5	3	3	3
	4	4	4	4
	4.5	5	4	5
	5	6	5	6
	5.5	7	5	7
	6	8	6	8
	6.5	9	6	9
	7	10	7	10
	7.5	11	7	11
	8	12	8	12

(CT এর সময় এইভাবে এবং একটা math দেয়া আছে).

Scanned by CamScanner

** Derive bresenham line drawing algorithm. (Imp)





Let our current position is (x_k, y_k). In bresenham line drawing algorithm our next choice will be either (x_k+1, y_k+1) or (x_k+1, y_k).

Now orginial line equation at x_{k+1} is,

$$y = m \cdot x_{k+1} + b = m(x_k+1) + b.$$

Let dupper is at (x_k+1, y_k+1) and dlower is at (x_k+1, y_k):

$$\text{dupper} = y_{k+1} - y$$

$$= y_k + 1 - m(x_k+1) - b.$$

$$\text{dlower} = y - y_k$$

$$= m(x_k+1) + b - y_k.$$

$$\therefore \text{dlower} - \text{dupper} = 2m(x_k+1) - 2y_k + 2b - 1 \quad (i)$$

Scanned by CamScanner

Now multiplying equation (1) by Δx and replacing m by $\Delta y/\Delta x$.

$$(d_{lower} - d_{upper})\,\Delta x = 2\Delta y(x_k+1) - 2\Delta x y_k + 2\Delta x b - \Delta x$$

$$= 2\Delta y x_k - 2\Delta x y_k + \Delta x(2b-1) + 2\Delta y.$$

$$= 2\Delta y x_k - 2\Delta x y_k + c \quad \left[\because c = \Delta x(2b-1) + 2\Delta y\right]$$

Now, Decision parameter, $P_k = 2\Delta y x_k - 2\Delta x y_k + c$.

If, P_k is ~~positive~~ negative then chose lower pixel.

P_k is positive then chose upper pixel.

$$P_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + c.$$

$$P_{k+1} - P_k = 2\Delta y(x_k+1) - 2\Delta x\, y_{k+1} + c - 2\Delta y x_k + 2\Delta x y_k - c$$

$$= 2\Delta y(x_k+1-x_k) - 2\Delta x(y_{k+1} - y_k)$$

$$= 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

Now, $y_{k+1} - y_k$ is either 0 or 1 because $y_{k+1} = y_k + 1$ or $y_{k+1} = y_k$.

$$\therefore P_{k+1} = \boxed{P_k + 2\Delta y} \quad (\text{If } y_{k+1} = y_k) \quad (P_k < 0)$$

$$P_{k+1} = \boxed{P_k + 2\Delta y - 2\Delta x} \quad (\text{If } y_{k+1} = y_k + 1) \quad (P_k \geq 0)$$

Scanned by CamScanner

Now first decision parameter at (x_0, y_0) is,

$$P_0 = \{ 2m(x_0+1) - 2y_0 + 2b - 1 \} \Delta x.$$

$$= \{ 2(mx_0 - y_0 + b) + 2m - 1 \} \Delta x.$$

$$= \boxed{2\Delta y - \Delta x} \qquad \left[\begin{array}{l} mx_0 + b - y_0 = 0 \\ m = \Delta y / \Delta x. \end{array} \right].$$

** Advantage of bresenham algorithm.

1. An fast incremental algorithm.
2. uses only integer calculation.

** Comparison of DDA and Bresenham algorithm.

1. There are rounding and floating point operation in DDA which is time consuming. calculation of bresenham algorithm are simple.

2. Accumulated round of error may change the ~~line~~ orginial line in DDA. No rounding is required in bresenham algorithm.

Scanned by CamScanner

** How can we apply Bresenham's line algorithm in the region out of $0<m<1$.

Bresenham's line algorithm works in the region $0<m<1$ where m stands for slope of the line.

→ we can use use bresenham line algorithm out of region $0<m<1$ by mirroring either horizontally, vertically or diagonally.

** Is it possible to apply bresenham's line algorithm when $-1<m<0$? How?

Yes, bresenham algorithm can be applied in the region $-1<m<0$.

A line from (x_1', y_1') to (x_2', y_2') with $-1<m<0$ has a horizontal mirrored counterpart from $(x_1', -y_1')$ to $(x_2', -y_2')$ with $0<m<1$.

we can simply use bresenham algorithm to scan convert counter part but negate y coordinate at the end of each iteration to get right pixel for the line.

Scanned by CamScanner

** For a line slope between 45° to 90°.

For a line whose slope is in the 45° to 90° range, we can obtain its mirrored counterpart by exchanging the x and y co-ordinates of its endpoints.

we can then scan convert the counterpart.

** Pseudocode for bresenham line algorithm $(0<m<1)$.

Now scan converting a line from $P_1'(x_1', y_1')$ to $P_2'(x_2', y_2')$,

Now,

```
int   x = x1' , y = y1'
int   dx = x2' - x1' , dy = y2' - y1'
int   P0 = 2 dx + dy .

while (x < x2') {
        setpixel (x,y) ;
        x++ ;
        if ( P < 0) .
            P = P+2dx . ;

        else
            P = P + 2dx - 2dy ;
            Y++;
}.
```

Scanned by CamScanner

(**) using bresenham line drawing algorithm digitize the line with endpoints (20, 10) and (30, 18).

Here, $\Delta x = 30 - 20 = 10$

$\Delta y = 18 - 10 = 8$

$P = 2 \cdot 8 - 10 = 6 > 0$

if $P_k < 0$,

$x_{k+1} = x_k + 1$;

$y_{k+1} = y_k$

$P_{k+1} = P_k + 2\Delta y$

else,

$x_{k+1} = x_k + 1$

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

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$.

First point $(x_0, y_0) = (20, 10)$.

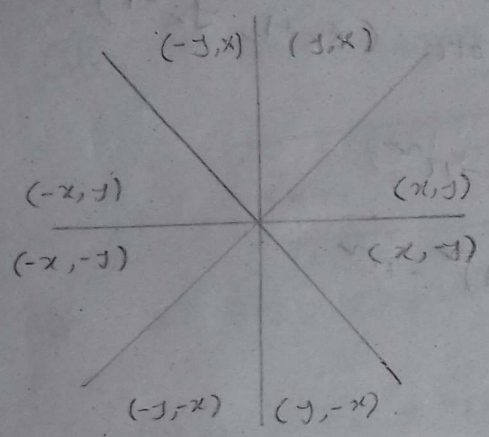
k	P_k	(x_{k+1}, y_{k+1})	P_{k+1}
0	6	(21, 11)	$6 + 16 - 20 = 2 > 0$
1	2	(22, 12)	$2 - 4 = -2 < 0$
2	-2	(23, 12)	$-2 + 16 = 14 > 0$
3	14	(24, 13)	$14 - 4 = 10 > 0$
4	10	(25, 14)	$10 - 4 = 6 > 0$
5	6	(26, 15)	$6 - 4 = 2 > 0$
6	2	(27, 16)	$2 - 4 = -2 < 0$
7	-2	(28, 16)	$-2 + 16 = 14 > 0$
8	14	(29, 17)	$14 - 4 = 10 > 0$
9.	10	(30, 18)	

Scanned by CamScanner

** what is 8-way symmetry? write short note on its application.

→ 8-way symmetry is the circle's symmetry to plot eight points. For each value the algorithm has calculated.

- It is created by reflecting each calculated point around each 45° axis.



Now,

$P_1 = (x, y)$ $\quad$ $P_5 = (-x, y)$

$P_2 = (y, x)$ $\quad$ $P_6 = (-x, -y)$

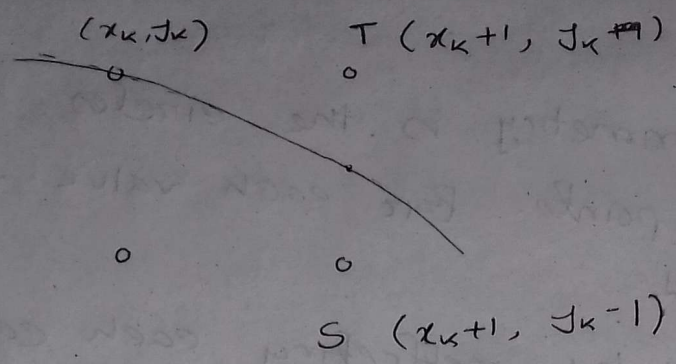
$P_3 = (-y, x)$ $\quad$ $P_7 = (-y, -x)$

$P_4 = (-x, y)$ $\quad$ $P_8 = (y, -x)$.

The main application of 8-way symmetry is that by calculating only one of 8-region of a circle we can draw the entire circle. by only replacing co-ordinate values and their sign.

Scanned by CamScanner

** Derive bresenham circle algorithm formula.



Let our current pixel position is (x_k, y_k). In bresenham's circle drawing algorithm our next position will either (x_k+1, y_k) or (x_k+1, y_k-1).

Now,

$$D(T) = (x_k+1)^2 + y_k^2 - r^2$$

$$D(S) = (x_k+1)^2 + (y_k-1)^2 - r^2$$

$$P_k = D(T) + D(S)$$

$$= 2(x_k+1)^2 + y_k^2 + (y_k-1)^2 - 2r^2$$

$$P_{k+1} = 2(x_{k+1}+1)^2 + y_{k+1}^2 + (y_{k+1}-1)^2 - 2r^2$$

$$P_{k+1} - P_k = 2\left\{(x_{k+1}+1)^2 - (x_k+1)^2\right\} + y_{k+1}^2 - y_k^2 + (y_{k+1}-1)^2 - (y_k-1)^2$$

As, $x_{k+1} = x_k + 1$.

Scanned by CamScanner

$$P_{k+1} - P_k = 2\{(x_k+2)^2 - (x_k+1)^2\} + y_{k+1}^2 - y_k^2$$
$$+ y_{k+1}^2 - 2y_{k+1} + \cancel{1} - y_k^2 + 2y_k - \cancel{1}$$

$$= 2\{\cancel{x_k^2} + 4x_k + 4 - \cancel{x_k^2} - 2x_k - 1) + 2(y_{k+1}^2 - y_k^2) - 2(y_{k+1} - y_k)$$

$$\therefore P_{k+1} = P_k + 4x_k + 6 + 2(y_{k+1}^2 - y_k^2) - 2(y_{k+1} - y_k).$$

If T is chosen then, $y_{k+1} = y_k$.

$$\therefore \boxed{P_{k+1} = P_k + 4x_k + 6.}$$

If S is chosen they $y_{k+1} = y_k - 1$

$$P_{k+1} = P_k + 4x_k + 6 + 2(y_k^2 - 2y_k + 1 - y_k^2)$$
$$- 2(y_k - 1 - y_k)$$

$$= P_k + 4x_k + 6 - 4y_k + 2 + 2$$

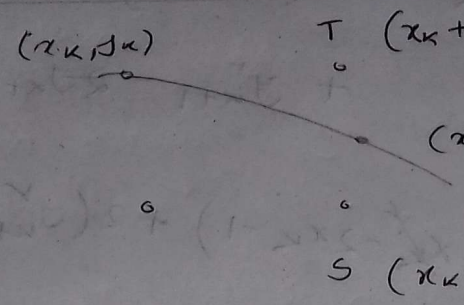
$$= \boxed{P_k + 4(x_k - y_k) + 10.}$$

we get $(0, r)$ as starting pixel co-ordinate.

$$\therefore P_0 = 2(0+1)^2 + r^2 + (r-1)^2 - 2r^2$$
$$= 2 + r^2 + r^2 - 2r + 1 - 2r^2$$
$$= \boxed{3 - 2r.}$$

Scanned by CamScanner

** Derive midpoint circle Algorithm.



Let our current pixel position is (x_k, y_k). In midpoint circle algorithm our next position will be either (x_k+1, y_k) or (x_k+1, y_k-1).

Decision variable can be defined as,

$$P_k = F(x_k+1, y_k-1/2)$$

$$= (x_k+1)^2 + (y_k-1/2)^2 - r^2.$$

$$P_{k+1} = (x_{k+1}+1)^2 + (y_{k+1}-1/2)^2 - r^2.$$

$$P_{k+1} - P_k = (x_k+2)^2 - (x_k+1)^2 + (y_{k+1}-1/2)^2 - (y_k-1/2)^2$$

$$= \cancel{x_k^2} + 4x_k + 4 - \cancel{x_k^2} - 2x_k - 1 + y_{k+1}^2 - y_{k+1} + 1/4 - y_k^2 + y_k - 1/4.$$

$$\therefore P_{k+1} = P_k + 2x_k + 3 + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k)$$

Scanned by CamScanner

Now first decision variable at $(0, r)$.

$$P_0 = f(0, r)$$

$$= 1 + (r - 1/2)^2 - r^2$$

$$= 1 + r^2 - r + 1/4 - r^2$$

$$= \frac{5}{4} - r.$$

If $P_k < 0$ then, $y_{k+1} = y_k$.

$$P_{k+1} = P_k + 2x_k + 3$$

$$= P_k + 2(x_k + 1) + 1 \qquad [\because x_{k+1} = x_k + 1].$$

$$= \boxed{P_k + 2x_{k+1} + 1}$$

If $P_k > 0$ then $y_{k+1} = y_k - 1$.

$$P_{k+1} = P_k + 2x_k + 3 + (\cancel{y_k^2} - 2y_k + 1 - \cancel{y_k^2}) - (y_k - 1 - y_k)$$

$$= P_k + 2x_k + 3 - 2y_k + 1 + 1$$

$$= P_k + 2(x_k + 1) - 2(y_k - 1) + 1$$

$$= \boxed{P_k + 2(x_{k+1} - y_{k+1}) + 1}$$

Scanned by CamScanner

2016
2(b)

** Explain Differences between midpoint and bresenham circle algorithm.

1. Bresenham circle algorithm results in much more smoother circle.

2. In bresenham decision parameter depends on previous decision parameter. In midpoint decision parameter depends on both previous decision parameter and corresponding pixel.

3. In bresenham initial decision parameter is determined by $\boxed{P_0 = 3-2r}$. In midpoint intial decision parameter is determined by $\boxed{P_0 = 1-r}$.

Bresenham

```
Let,
    x=0, y=r, P0=3-2r.
while (x<=y) {
    setpixel (x,y);
    if (P<0)
        P = P+4x+6;
    ~~if (P>0)~~
    else {
        P= P+4(x-y)+10;
        y--;
    }
    x++;
}
```

midpoint

```
Let x=0, y=r, P= 1-r;
while ( x <= y){
    setpixel (x,y);
    if (P<0)
        P= P+ 2x+3;
    else{
        P= P+ 2(x-y)+5
        y--
    }
    x++;
}.
```

Scanned by CamScanner

** Given a circle radius $r = 15$, demonstrate midpoint circle algorithm.

Here, $P_0 = 1 - r = 1 - 15 = -14$.

if $(P_k < 0)$,

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$

$P_{k+1} = P_k + 2x_{k+1} + 1$

else

$x_{k+1} = x_k + 1$

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

$P_{k+1} = P_k + 2(x_{k+1} - y_{k+1}) + 1$

Plot initial point $(0, 15)$.

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	P_{k+1}
0	-14	(1, 15)	2	30	-11
1	-11	(2, 15)	4	30	-6
2	-6	(3, 15)	6	30	1
3	1	(4, 14)	8	28	-18
4	-18	(5, 14)	10	28	-7
5	-7	(6, 14)	12	28	6
6	6	(7, 13)	14	26	-5
7	-5	(8, 13)	16	26	12
8	12	(9, 12)	18	24	7
	7	(10, 11)	20	22	6
	6	(11, 10)	22	20	

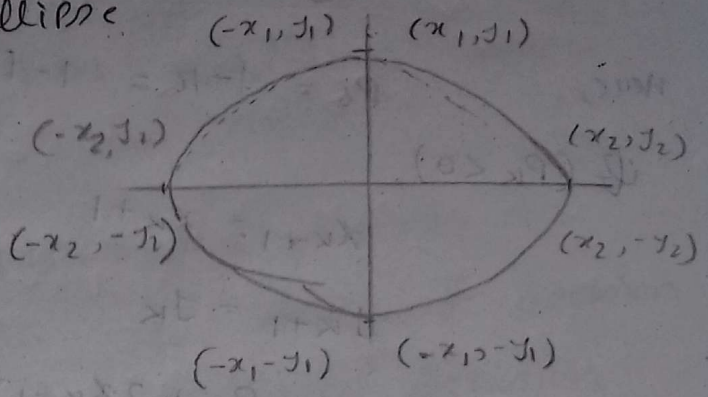
Scanned by CamScanner

* Derive decision parameter for midpoint ellipse algorithm.

Now, Implicit equation for ellipse

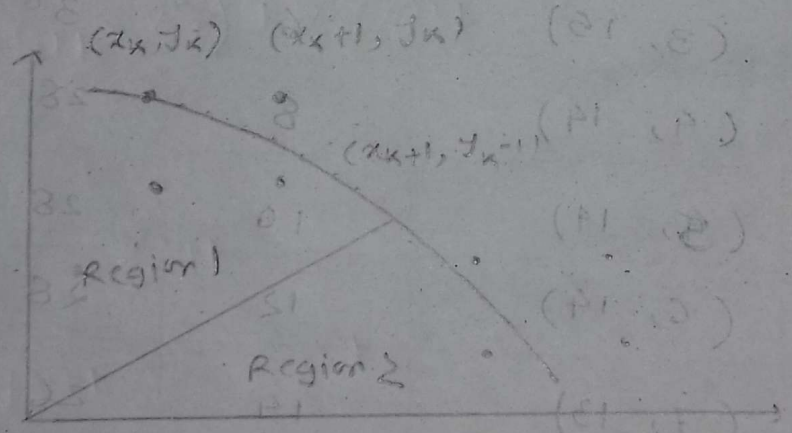
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$\Rightarrow \quad b^2x^2 + a^2y^2 - a^2b^2 = 0$$



There exist 2 regions.

1. In Region 1, $dx > dy$, increase x at each step y may decrease.
2. In region 2 $dx < dy$, decrease y at each step x may increase.



Scanned by CamScanner

At region 1.

$$P_k = f(x_k+1, y_k-1/2).$$

$$= b^2(x_k+1)^2 + a^2(y_k-1/2)^2 - a^2b^2.$$

$$P_{k+1} = b^2(x_{k+1}+1)^2 + a^2(y_{k+1}-1/2)^2 - a^2b^2.$$

$$P_{k+1} - P_k = b^2\{(x_k+2)^2 - (x_k+1)^2\} + a^2(y_{k+1}-1/2)^2 - a^2(y_k-1/2)^2$$

$$= b^2(x_k^2+4x_k+4-x_k^2-2x_k-1) + a^2(y_{k+1}^2 - y_{k+1} + 1/4 - y_k^2 + y_k - 1/4)$$

$$\therefore P_{k+1} = P_k + 2b^2x_k + 3b^2 + a^2(y_{k+1}^2 - y_k^2) - a^2(y_{k+1} - y_k).$$

If $P_k < 0$, $y_{k+1} = y_k$.

$$\boxed{P_{k+1} = P_k + 2b^2x_k + 3b^2}$$

If $P_k > 0$ $y_{k+1} = y_k - 1$.

$$P_{k+1} = P_k + 2b^2x_k + 3b^2 + a^2\{(y_k-1)^2 - y_k^2\} + a^2(y_k - 1 - y_k)$$

$$\boxed{P_{k+1} = P_k + 2b^2x_k + 3b^2 - 2a^2y_k}.$$

$$\therefore P_0 = \cancel{b^2(0+1)^2 +} \boxed{b^2 + a^2/4 - a^2b^2}.$$

Scanned by CamScanner

At Region 2.

$$P_j = f(x_j + 1/2, y_j - 1)$$

$$= b^2(x_j + 1/2)^2 + a^2(y_j - 1)^2 - a^2b^2.$$

$$P_{j+1} = b^2(x_{j+1} + 1/2)^2 + a^2(y_{j+1} - 1)^2 - a^2b^2.$$

$$P_{j+1} - P_j = b^2\{(x_{j+1} + 1/2)^2 - (x_j + 1/2)^2\} + a^2\{(y_{j+1} - 1)^2 - (y_j - 1)^2\}$$

$$= b^2(x_{j+1}^2 + x_{j+1} + 1/4 - x_j^2 - x_j - 1/4) + a^2(y_{j+1}^2 - 2y_{j+1} + 1 - y_j^2 + 2y_j - 1)$$

$$P_{j+1} = P_j + b^2(x_{j+1}^2 - x_j^2) + b^2(x_{j+1} - x_j) + a^2(y_{j+1}^2 - y_j^2) - 2a^2(y_{j+1} - y_j).$$

If $P_j > 0$, $x_{j+1} = x_j$.

$$P_{j+1} = P_j + a^2(y_j^2 - 2y_j + 1 - y_j^2) + 2a^2$$

$$= P_j + 2a^2 y_j + 3a^2.$$

If $P_j < 0$, $x_{j+1} = x_j + 1$

Scanned by CamScanner

$$P_{j+1} = P_j + b^2(x_j^2 + 2x_j + 1 - x_j^2) + b^2 + 2a^2 y_j + 3a^2$$

$$= \boxed{P_j + 2b^2 x_j + 3b^2 + 2a^2 y_i + 3a^2}$$

$$\therefore P_0 = b^2(0 + 1/2)^2 + a^2(-1)^2 - a^2 b^2$$

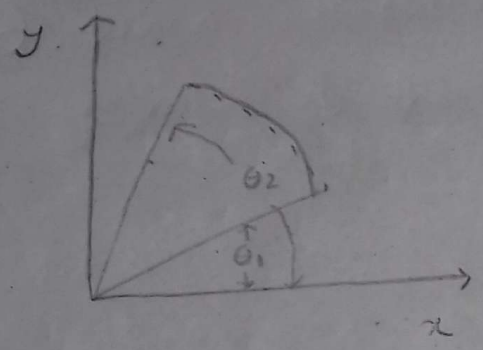
$$= \boxed{b^2/4 + a^2 - a^2 b^2}$$

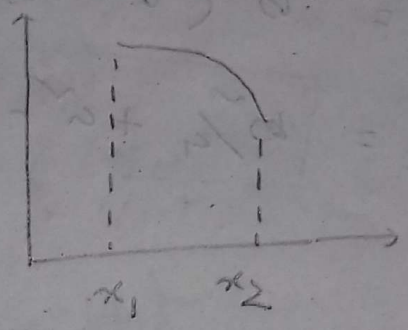
Scanned by CamScanner

* Define Arcs and Sectors:-

Arcs:- An arc may be generated by either polynomial or trigonometric method.

when trigonometric method is used the starting value is set as θ_1 and ending value is set as θ_2.



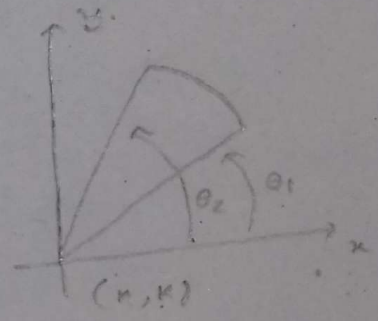


when polynomial method is used the value of x is varied from x_1 to x_2 and values of y are found by evaluating the expression $\sqrt{r^2 - x^2}$.

The rest of the steps are similar as scan converting a circle except symmetry is not used.

Sectors:

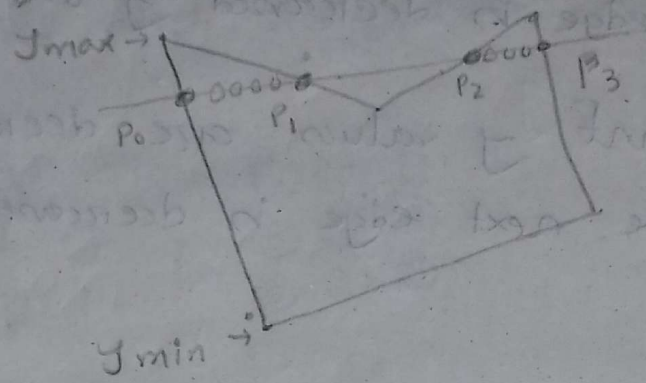
A sector can be scan converted by using any of the method of scan converting an arc. and then scan converting two lines from the center of the arc to endpoints of the arc.



Scanned by CamScanner

** Describe scan line algorithm to scan convert a polygon.

This algorithm works by intersecting scan-line with polygon edges. and fill the polygon between pairs of intersection.



Steps :-

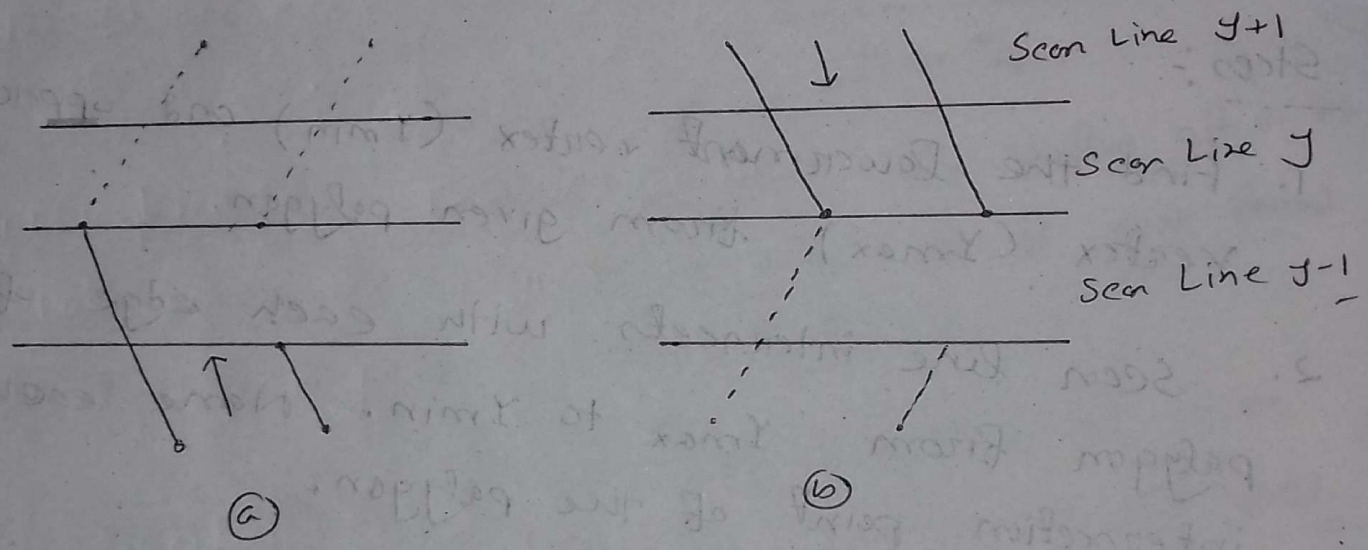
1. Find the lowermost vertex (Ymin) and uppermost vertex (Ymax) from given polygon.
2. Scan line intersects with each edge of the polygon from Ymax to Ymin. Name each intersection point of the polygon.
3. Sort the intersection point of the increasing order of x- coordinate. (P0, P1), (P1, P2) and (P2, P3).
4. Fill all those pairs that are inside the polygon and ignore alternate ones.

Scanned by CamScanner

Dealing with the vertices :-

(a) when endpoint y co-ordinate of two edges are increasing, the y value of the upper endpoint for the current edge is decreased by one.

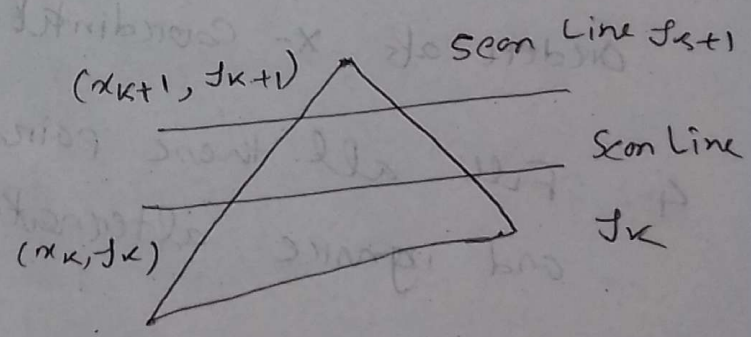
(b) when endpoint y values are decreasing, the y value of the next edge is decreased by one.



Determining edge intersection :-

$$m = (y_{k+1} - y_k)/(x_{k+1} - x_k)$$

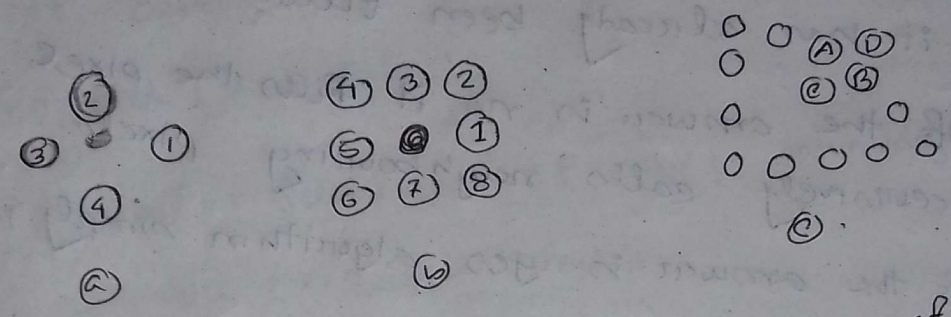
$$\therefore \; x_{k+1} = x_k + \frac{1}{m}$$



Scanned by CamScanner

** Distinguish between 4-connected and 8-connected pixel and state their applications.

1. 4-Connected pixel may have upto 4-neighbours, 8-connected pixel may have upto 8-neighbours.



2. using 4-connected defination we do not have any region in fig(c) but by using 8-connected defination we identify a triangular region.

3. 4-Connected pixel may fail to fill some region which can be filled by 8-connected pixel.

Application :- 4-connected and 8-connected pixel are mainly used to fill region and connect different regions.

Scanned by CamScanner

** Describe Boundary Fill Algorithm

This recursive algorithm begins with,

1. A starting pixel called seed inside the region.
2. The algorithm checks if the pixel is on boundary or it has already been filled.
3. If the answer is no it fills the pixel and recursively calls neighbouring pixels.
4. If the answer is yes algorithm simply return to its caller.

Application :- This algorithm is particularly useful when the region to be filled has uniformly colored boundary.

Disadvantages :- (This disadvantage is also applicable for flood fill Algo).

1. Straightforward implementation is time consuming.
2. Require more memory for high number of recursive call.
3. Recursive call is not feasible for relatively large region.

Scanned by CamScanner

pseudo code for boundary fill algorithm for 4-connected defination.

```
BoundaryFill (int x, y, fill-color, boundary-color).
{
    int color;
    getpixel (x, y, color);
    if ( color != fill-color && color != boundary-color){
        setpixel (x, y, fill-color);
        BoundaryFill ( x+1, y, fill-color, boundary-color);
        Boundary Fill (x, y+1, fill-color, boundary-color);
        Boundary Fill (x-1, y, fill-color, boundary-color);
        Boundary Fill (x, y-1, fill-color, boundary-color);
    }
}
```

** 8 Connected ~~ज~~ Just ~~8 ~~ 4 ~~ 4 recursive call 2 ~~

** Describe flood fill algorithm. (Same as boundary fill just condition diff)

This recursive algorithm begins with,

1. A starting pixel called seed inside the region.
2. The algorithm checks if the pixel has regions original color.
3. If the answer is yes it fills the pixel with new color and call neighbouring pixel.
4. If no, the algorithm simply returns to its caller.

Application:-

1. This method is useful when the region to be filled has no uniformly colored boundary.
2. in puzzle games.
3. to determine which parts of a bitmap to fill with color.

Scanned by CamScanner

pseudo code.

```
FloodFill (int x, y, fill-color, orginial-color)
{
    int color;
    getpixel (x, y, color);
    if (color == orginial-color)
    {
        setpixel (x, y, fill-color);
        FloodFill ( x+1, y, fill-color, orginial-color);
        Flood Fill (x, y+1, fill-color, orginial-color);
        Flood Fill (x-1, y, fill-color, orginial-color);
        Flood Fill (x, y-1) fill-color, orginial-color);
    }
}
```

Example 3.27

3.28

3.29

** How limit number of recursive calls in BF and FF Algo.
v.v.g
(Scan Blood fill algo)

→ we can limit the number of recursive call by structuring the order of neighbouring pixel.

1). we can first fill the pixel to the left and right of the seed on same scan line untill boundary pixel hit.

2) Then inspect each pixel above and below the line just drawn to see if it can be used as a new seed for the next horizontal line to fill.

** Short note on Bitmap and outline font.

Bitmap Font :-

1. Character is represented by the pixel in a bi-level pixel grid pattern called a bitmap.
2. characters are already in scan converted form.
3. size of bitmap font is dependent on image resolution.

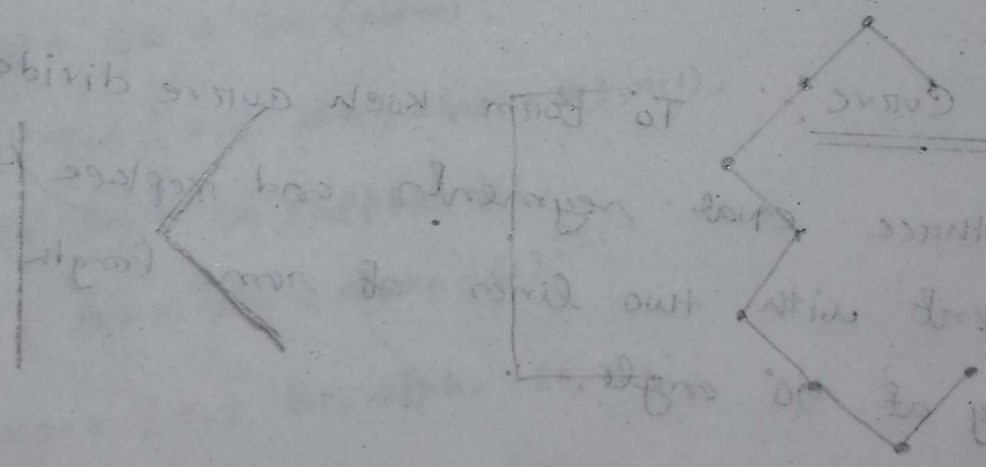
Scanned by CamScanner

outline Font :-

1. The graphical primitives such as lines and arcs are used to define outline of each character.
2. Requires scan conversion operations.
3. Can be used to produce characters of varying size, appearance and even orientation.

** Recursively Defined drawings:

C curve :- To form C curve, replace a line by two shorter equal-length lines joining each other at 90° angle.



C_0 C_1 C_2 C_3

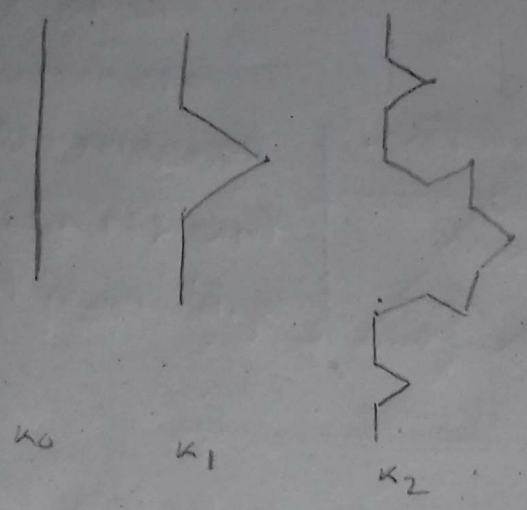
Scanned by CamScanner

pseudo code.

```
C- curve ( float x, y, len, alpha, int n).
{
    if (n>0){
        len = len/sqrt(2.0);
        C-curve ( x, y, len, alpha+45, n-1);
        x = x + len * cos(alpha +45);
        y = y + len * sin (apha+45);
        C-curve (x,y, len, alpha-45, n-1);
    }
    else {
        line (x, y, x + len * cos(alpha), y+len * sin(alpha))
    }
}
```

Koch Curve:- To form koch curve divide a line into three equal segments and replace the middle segment with two lines of same length ~~which~~ joining at 90° angle.

Scanned by CamScanner



(similarly)

(প্রতি step এ 4 টা করে সুনির্দিষ্ট
করে line বাড়ে)

Pseudocode :- (Easy)

```
koch-curve (float x, y, len, alpha, int n).
{ if (n>0) {
        len = len/3;
    koch-curve (x, y, len, apha, n-1);
    x = x + len * cos(alpha);
    y = y + len * sin(alpha);
    koch-curve (x, y, len, alpha-60, n-1);
    x = x + len * cos(alpha-60);
    y = y + len * sin(alpha-60);
    koch-curve (x, y, len, alpha+60, n-1);
    x = x + len * cos(apha+60);
    y = y + len * sin(appha+60);
    koch-curve (x, y, len, alpha, n-1);
}
```

Scanned by CamScanner