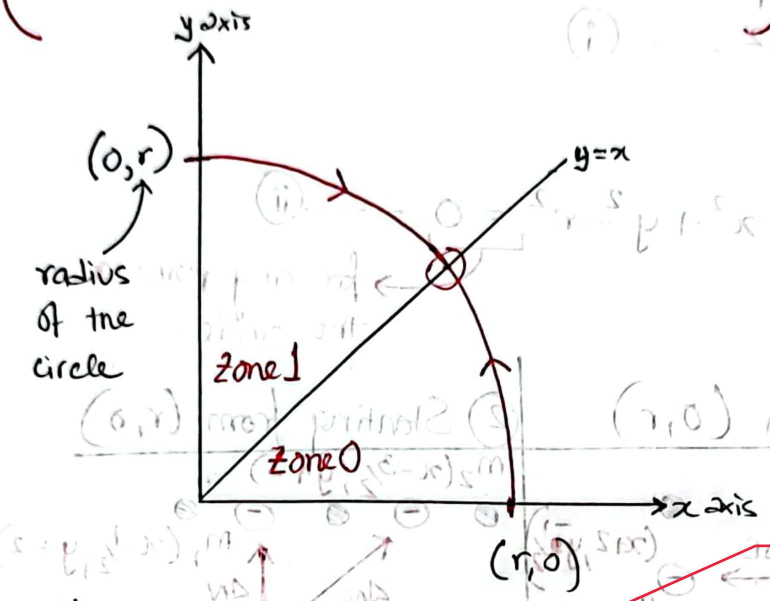
(MID POINT CIRCLE ALGORITHM)



Ø In midpoint circle algorithm we can follow two approaches

① Starting from (0,r)

② Starting from (r,0)

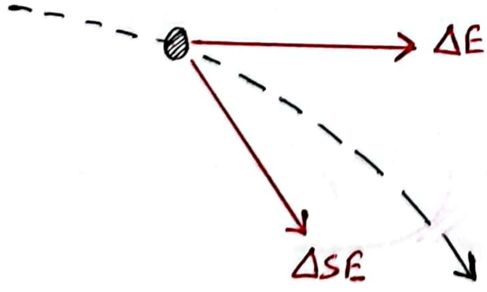
Have to start with zone 0 / zone 1

Concept :

① Starting from (0,r)

We calculate the pixels belonging to zone 1 segment of the circle by starting from (0,r), and then map them onto the other zones using 8-way symmetry

For Zone 1 (0,r)

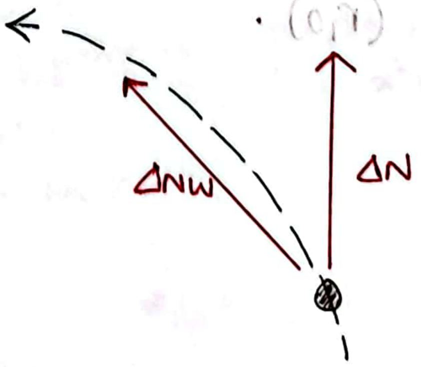


Ø There can be two movements, ΔE & ΔSE

② Starting from (r,0)

We calculate the pixels belonging to zone 0 segment of the circle by starting from (r,0), and then map them onto the other zones using 8-way symmetry

For Zone 0 (r,0)



Ø ΔN & ΔNW movements

The general equation of a circle:

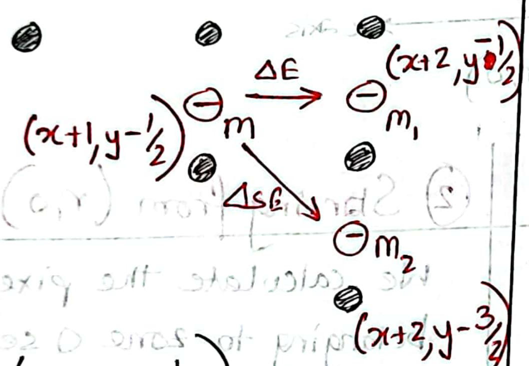
$$x^2+y^2=r^2 \quad \text{-----} \ (i)$$

Explicit form:

$$f(x,y) = x^2+y^2-r^2 \underbrace{= 0}_{} \quad \text{-----} \ (ii)$$

→ for any point on the circle.

(1) Starting from (0,r)

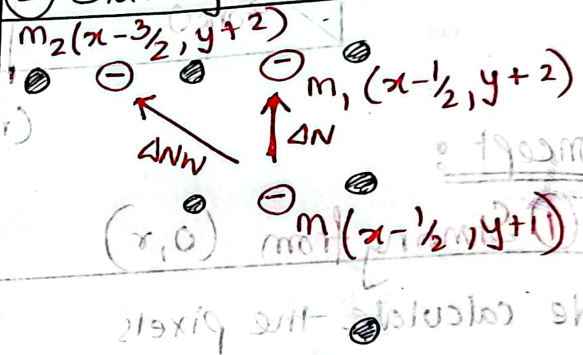


$m = (x+1, y-\frac{1}{2})$

$m_1 = (x+2, y-\frac{1}{2})$

$m_2 = (x+2, y-\frac{3}{2})$

(2) Starting from (r,0)



$m = (x-\frac{1}{2}, y+1)$

$m_1 = (x-\frac{1}{2}, y+2)$

$m_2 = (x-\frac{3}{2}, y+2)$

(*) We only need the mapping algorithm for zone 0 to all other zone, if we start at (r,0).

Derivation for (0, r) → zone 1

For ΔE movement:

$$f(m_1) \longrightarrow (x+2)^2 + (y-1/2)^2 - r^2 = d_1$$

$$f(m) \longrightarrow \underset{(-)}{(x+1)^2} + \underset{(-)}{(y-1/2)^2} \underset{(+)}{-\,r^2} = \underset{(-)}{d}$$

$$2x+3 \;+\; 0 \;+0 = d_1 - d = \Delta E$$

$$\therefore \boxed{\Delta E = 2x+3} \rightarrow \text{(i)}$$

$(a+b)^2 = a^2 + 2ab + b^2$

$(a-b)^2 = a^2 - 2ab + b^2$

For ΔSE movement:

$$f(m_2) \longrightarrow (x+2)^2 + (y-3/2)^2 - r^2 = d_2$$

$$f(m) \longrightarrow \underset{(-)}{(x+1)^2} + \underset{(-)}{(y-1/2)^2} \underset{(+)}{-\,r^2} = \underset{(-)}{d}$$

$$(2x+3) + (-2y+2) + 0 = d_2 - d = \Delta SE$$

$$\therefore \boxed{\Delta SE = 2x - 2y + 5} \rightarrow \text{(ii)}$$

For dinit:

$$f(m) \longrightarrow (x_0+1)^2 + (y_0 - 1/2)^2 - r^2 = dinit$$

at x_0, y_0

$$\Rightarrow \underbrace{(x_0)^2} + 2x_0 + 1 + \underbrace{(y_0)^2} - y_0 + 1/4 \underbrace{-(r)^2} = dinit$$

These three terms accumulate to give 0, according to the general eqn of a circle: $x^2+y^2-r^2=0$

$$\Rightarrow dinit = 2x_0 - y_0 + 5/4$$

$$\Rightarrow dinit = 2(0) - r + 5/4$$

Since, the starting point is $(0, r)$

So we know,

$x_0 = 0,$

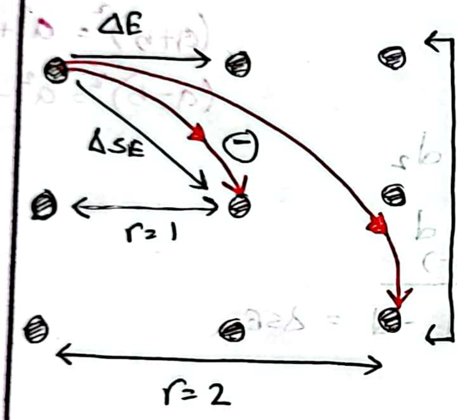
$y_0 = r,$

$$\therefore \boxed{dinit = 5/4 - r} \rightarrow \text{(iii)}$$

How to decide if ΔE/ΔSE ?

For that we need the value of dinit,

$d = 5/4 - r$



So,

for r=2, curve goes over the mid point → ΔE

for r=1, curve goes below the mid point → ΔSE

(i) For zone 0,3,4,7:

if dinit = -ve, x increase/decrease by 1. means it will move to dE

if dinit = +ve, x and y both increase/decrease by 1. means it will move to dSE/dNE

(ii) For zone 1,2,5,6:

Opposite to (i)

(*) r=1 & r=2 are arbitrary values used just to understand the algorithm.

ΔE (when, r=2)	ΔSE (when, r=1)
$d = 5/4 - r$	$d = 5/4 - r$
$\Rightarrow d = 5/4 - 2 = -0.75$ (-ve)	$\Rightarrow d = 5/4 - 1 = 0.25$ (+ve)
if d is -ve, negative the movement is ΔE	if d is +ve, Positive the movement is ΔSE

Pseudo Code:

```
Void  drawCircle_zone1 (int r) {
          int d = 1-r   → since, 1 & 1.25 are significantly close, we'll use 1.
      OR
          int d = 5-(4*r)  → multiplied with 4 to remove fraction, if we
                             use this, we need to multiply ΔE & ΔSE
                             with 4 as well.
                             [ ΔE = 4x(2x+3)
                             [ ΔSE = 4x (2x-2y+5)
          int x = 0
          int y = r
draw the pixel on ← draw8way (x,y)
all 8 zones
          while (x≤y){
              if (d<0) { // ΔE  since -ve, increase x by 1
                    d = d+2x+3
                    x = x+1
              }
              else { // ΔSE,  increase x by 1, decrease y by 1
                    d = d+2x-2y+5
                    x = x+1
                    y = y-1
              }
              draw8way (x,y);
          }
}
```

Q1 Draw a circle a Circle with radius 10 and origin (0,0), slanting with (0,10)

$d = 1 - r$, $\longrightarrow$ $(1 - 10 = -9)$

$\Delta E = 2x + 3$

$\Delta SE = 2x - 2y + 5$

x	y	d	$\Delta E/\Delta SE$	d update	PIXEL
0	10	−9	ΔE	$= -9 + (2\times0+3) = -6$	(0,10)
1	10	−6	ΔE	$= -6 + (2\times1+3) = -1$	(1,10)
2	10	−1	ΔE	$= -1 + (2\times2+3) = 6$	(2,10)
3	10	6	ΔSE	$= 6 + (2\times3 - 2\times10+5) = -3$	(3,10)
4	9	−3	ΔE	$= -3 + (2\times4+3) = 8$	(4,9)
5	9	8	ΔSE	$= 8 + (2\times5 - 2\times9+5) = 5$	(5,9)
6	8	5	ΔSE	$= 5 + (2\times6 - 2\times8+5) = 6$	(6,8)
7	7	6	~	~	(7,7)

answer.

When x >= y then stop

Q2) Calculate the points for zone 1 of a circle with radius = 6 and origin centered at (10,10)?

↳ we'll carry out the normal calculations but since the origin is at (10,10) we need to add +10 to the x coordinate & +10 to the y coordinate of each pixel to translate the circle.

$d = 1 - r$

$\Delta E = 2x + 3$

$\Delta SE = 2x - 2y + 5$

x	y	d	ΔE/ΔSE	d update	Actual PIXEL $(x+10, y+10)$
0	6	−5	ΔE	$= -5 + (2\times0+3) = -2$	$(0+10, 6+10) = (10,16$
1	6	−2	ΔE	$= -2 + (2\times1+3) = 3$	(11,16)
2	6	3	ΔSE	$= 3 + (2\times2 - 2\times6 + 5) = 0$	(12,16)
3	5	0	ΔSE	$= 0 + (2\times3 - 2\times5 + 5) = 1$	(13,15)
4	4	1	~	~	(14,14)

X

Derivation for (r,0) ⟶ zone 0

ΔN

$f(m_1) \longrightarrow (x-\tfrac{1}{2})^2 + (y+2)^2 - r^2 = d_1$

$f(m) \rightarrow \underset{(-)}{}(x-\tfrac{1}{2})^2 \underset{(-)}{+} (y+1)^2 \underset{(+)}{} - r^2 \underset{(-)}{=} d$

$0 + (2y+3) + 0 = d_1 - d = \Delta N$

$\therefore \Delta N = 2y+3$

ΔNW

$f(m_2) \longrightarrow (x-\tfrac{3}{2})^2 + (y+2)^2 - r^2 = d_2$

$f(m) \rightarrow (x-\tfrac{1}{2})^2 + (y+1)^2 - r = d$

$(-2x+2) + (2y+3) = d_2 - d = \Delta NW$

$\therefore \Delta NW = 2y - 2x + 5$

dinit

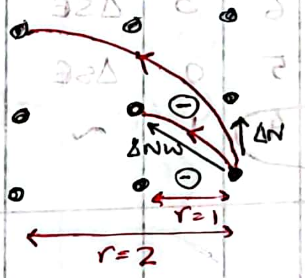
$f(m) \longrightarrow (x_0 - \tfrac{1}{2})^2 + (y_0+1)^2 - r^2 = dinit$

at (x_0, y_0) $dinit = \tfrac{5}{4} - r$

$x_0 = r,$

$y_0 = 0,$

So, $d > 0$, ΔNW ; $d < 0$, ΔN



Pseudo

```
void drawCircle_zone0 (int r) {
    int d = 1 - r
    x = r
    y = 0
    draw 8way (x,y)
    while (x >= y) {
        if (d > 0) { //ΔNW
            d+ = 2y - 2x + 5
            x--
            y++
        }
        else { //ΔN
            d+ = 2y + 3
            y++
        }
        draw 8way (x,y)
    }
}
```

Q3) For a circle with radius = 20, and origin (0,0). Draw the circle using MPC algo for zone 0, starting at (20,0). [For 12 pixels]

d = − 19

dN = 2y+3 ⟶ for d<0

dNW = 2y −2x+5 ⟶ for d>0

SL	x	y	d	ΔN/ΔNW	d update	PIXEL
1.	20	0	−19	ΔN	= −19 +(2×0+3) = −16	(20,0)
2.	20	1	−16	ΔN	= −16+ (2×1+3) = −11	(20,1)
3.	20	2	−11	ΔN	= −11+ (2×2+3) = −4	(20,2)
4.	20	3	−4	ΔN	= −4+ (2×3+ 3) = 5	(20,3)
5.	20	4	5	ΔNW	= 5+ (2×4 − 2×20+5) = −22	(20,4)
6.	19	5	−22	ΔN	= −22+(2×5 + 3) = −9	(19,5)
7.	19	6	−9	ΔN	= −9+(2×6+3) = 6	(19,6)
8.	19	7	6	ΔNW	= 6+(2×7 −2×19+5) = −13	(19,7)
9.	18	8	−13	ΔN	= −13+(2×8 + 3) = 6	(18,8)
10.	18	9	6	ΔNW	= 6+(2×9 −2×18 +5) = −7	(18,9)
11.	17	10	−7	ΔN	= −7+(2×10+3) = 16	(17,10)
12.	17	11	16	—	—	(17,11)