CSE-303: COMPUTER GRAPHICS

PROFESSOR DR. SANJIT KUMAR SAHA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAHANGIRNAGAR UNIVERSITY, SAVAR, DHAKA

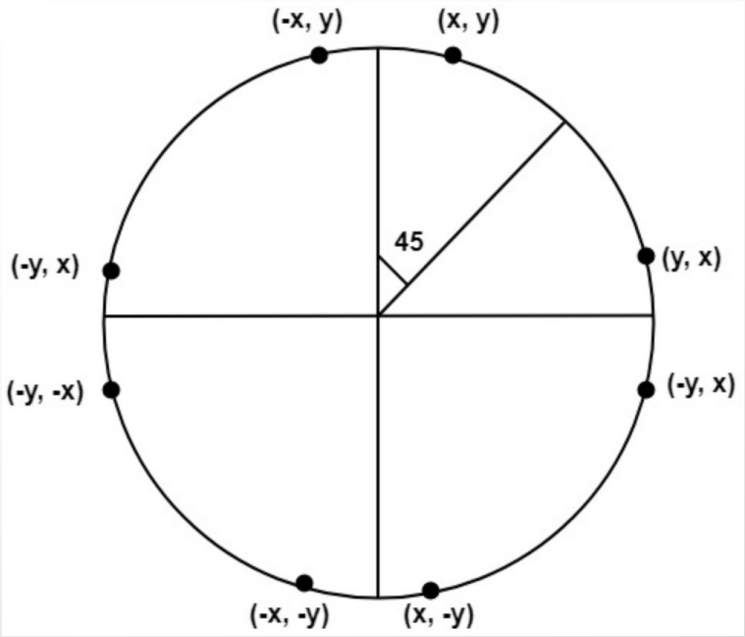
SCAN CONVERSION II

DIFFERENCES BETWEEN LINE ALGORITHM

	Vector Generation Algorithm/ DDA Line Algorithm	**Bresenham's Line Algorithm**
1	It uses Floating point arithmetic	It uses only integer addition, subtraction and multiplication by 2.
2	Due to floating point arithmetic and rounding function it takes more time.	It is quicker than vector generation algorithm
3	Less efficient	More efficient
4	Where speed is important this algorithm need to be implemented in hardware	Hardware implementation is not required.

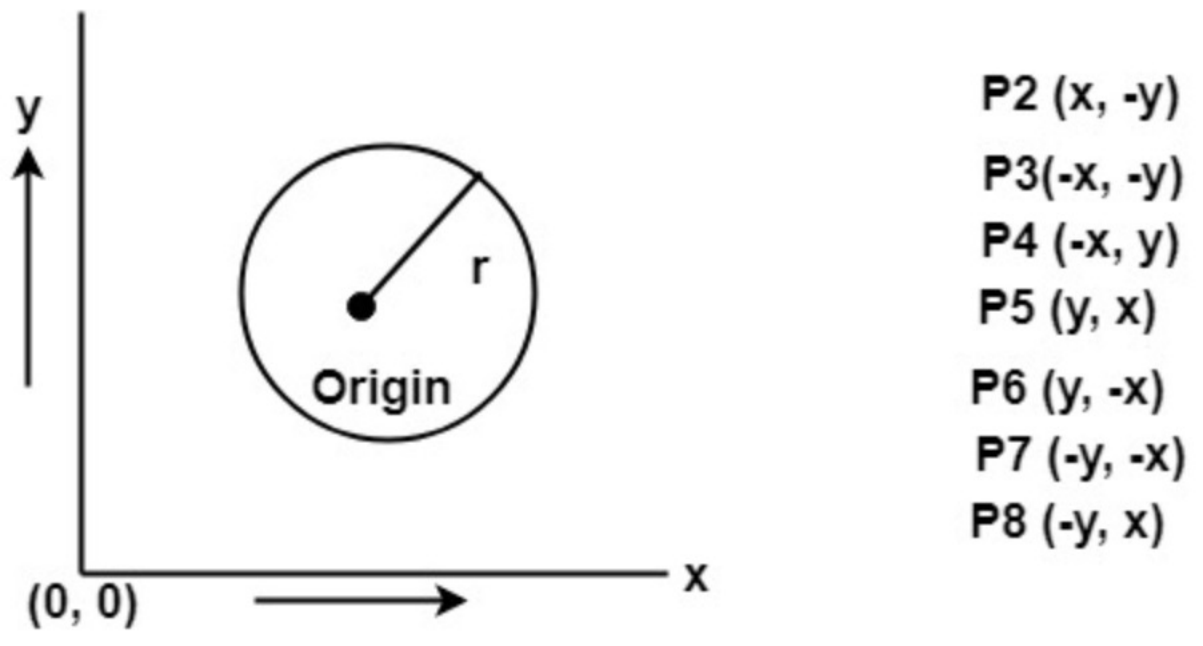
SCAN CONVERSION OF CIRCLE

- Circle is an eight-way symmetric figure.
- The shape of circle is the same in all quadrants.
- In each quadrant, there are two octants.
- If the calculation of the point of one octant is done, then the other seven points can be calculated easily by using the concept of eight-way symmetry.



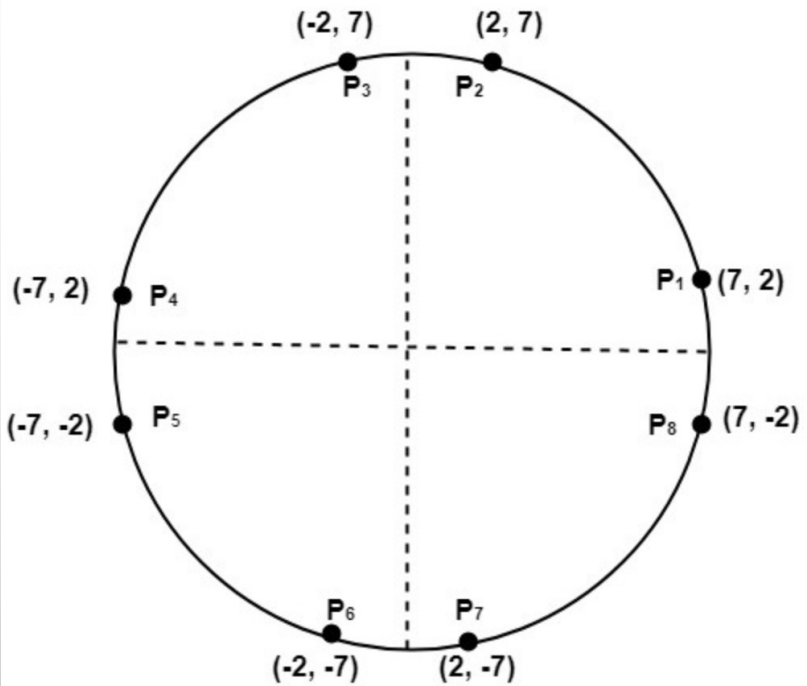
SCAN CONVERSION OF CIRCLE

For drawing, circle considers it at the origin. If a point is $p_1(x, y)$, then the other seven points will be



EXAMPLE

- Let we determine a point (2, 7) of the circle then other points will be (2, -7), (-2, -7), (-2, 7), (7, 2), (-7, 2), (-7, -2), (7, -2)
- These seven points are calculated by using the property of reflection. The reflection is accomplished in the following way:
  - The reflection is accomplished by reversing x, y co-ordinates.



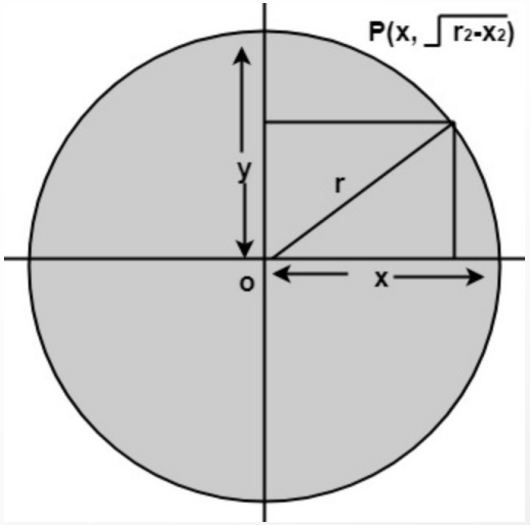
Eight way symmetry of a Circle

BASICS OF CIRCLE GENERATION

Circle can be defined in two ways:

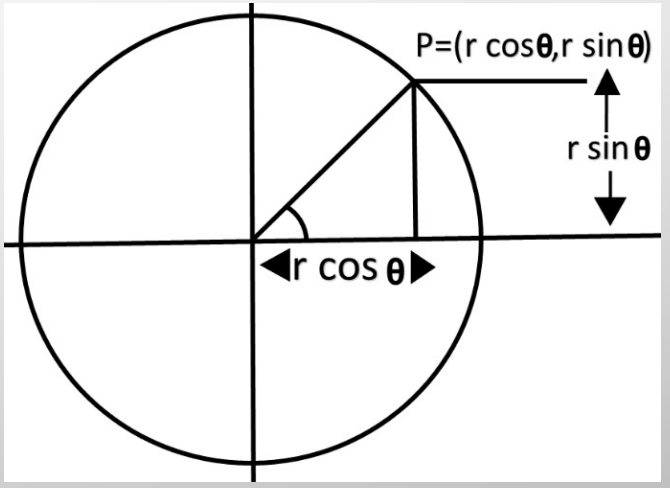
- Polynomial method:

$$x^2 + y^2 = r^2$$



- Trigonometric or polar co-ordinates method

$$x = r\cos\theta \qquad y = r\sin\theta$$



POLYNOMIAL METHOD

- The equation for a circle is:

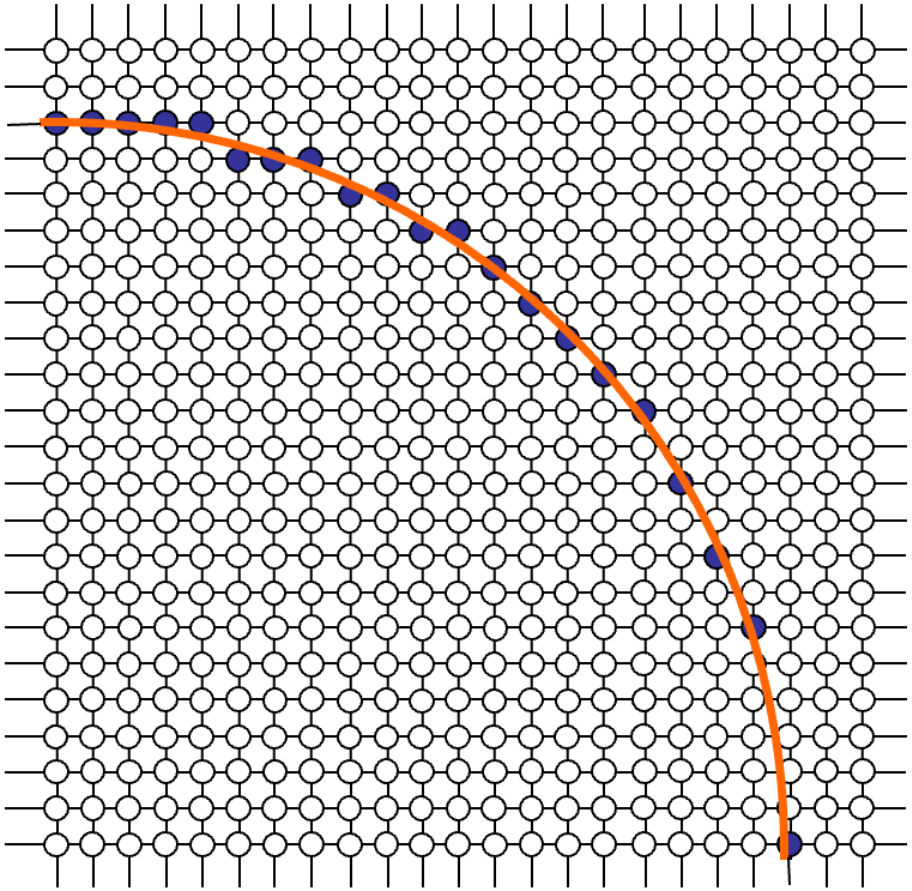
$$x^2 + y^2 = r^2$$

where r is the radius of the circle

- So, we can write a simple circle drawing algorithm by solving the equation for y at unit x intervals using:

$$y = \pm\sqrt{r^2 - x^2}$$

POLYNOMIAL METHOD



$$y_0 = \sqrt{20^2 - 0^2} \approx 20$$

$$y_1 = \sqrt{20^2 - 1^2} \approx 20$$

$$y_2 = \sqrt{20^2 - 2^2} \approx 20$$

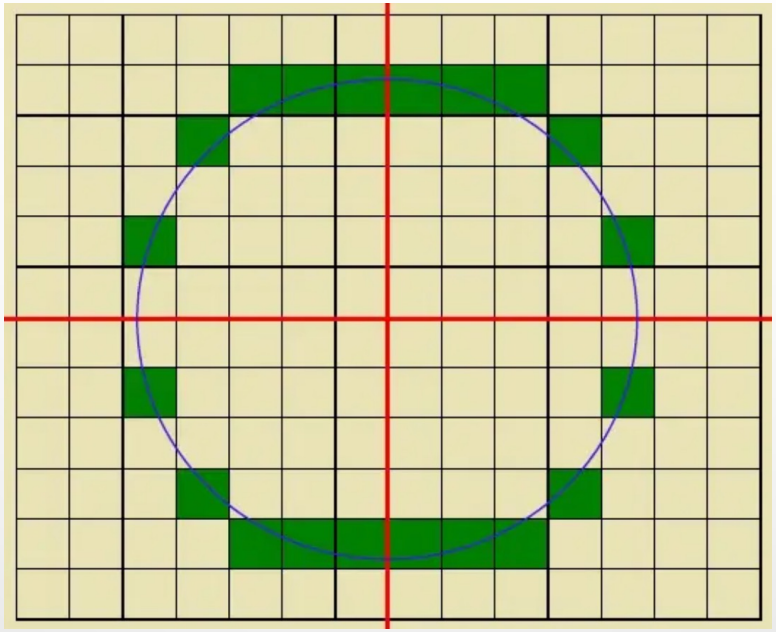
$$\vdots$$

$$y_{19} = \sqrt{20^2 - 19^2} \approx 6$$

$$y_{20} = \sqrt{20^2 - 20^2} \approx 0$$

POLYNOMIAL METHOD

- Step through x-axis to determine y values



Disadvantages:

- Not all pixel filled in.
- Square root function is slow.

POLYNOMIAL METHOD

- However, unsurprisingly this is not a brilliant solution!
- Firstly, the resulting circle has large gaps where the slope approaches the vertical
- Secondly, the calculations are not very efficient

  The square (multiply) operations

  The square root operation – try really hard to avoid these!
- We need a more efficient, more accurate solution

TRIGONOMETRIC METHOD

$\theta = 0°$
while ($\theta < 360°$)
$x = r\text{Cos}\theta$
$y = r\text{Sin}\theta$
setPixel(x,y)
$\theta = \theta + 1°$
end while

Disadvantages:

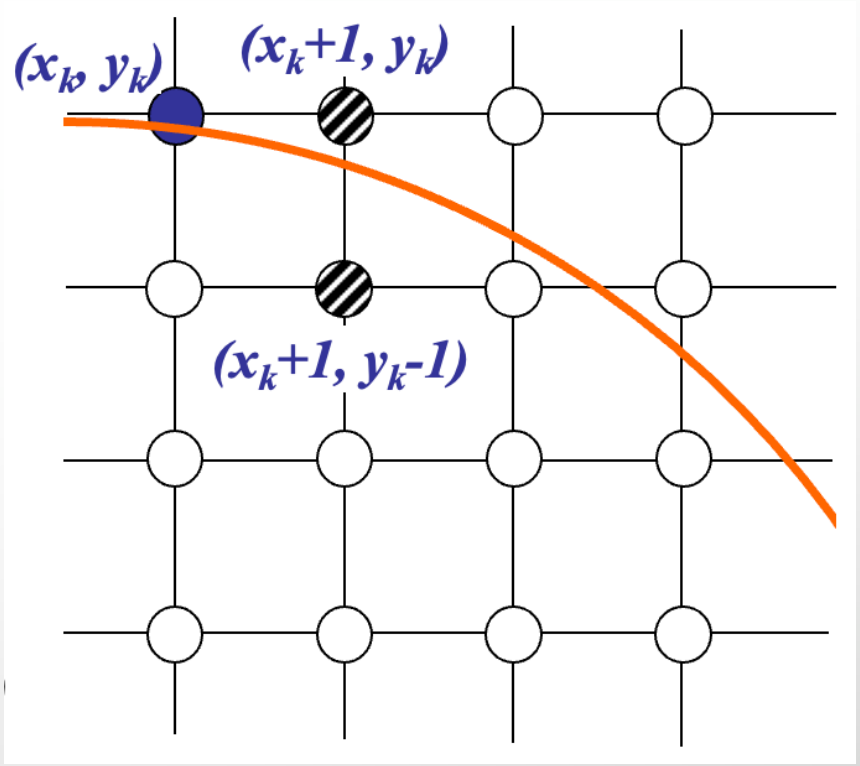
- To find a complete circle θ varies from 0 to 360 degree.
- The calculation is slow.

ALGORITHMS

- Midpoint circle algorithm
- Bresenham's circle algorithm

MIDPOINT CIRCLE ALGORITHM

- Assume that we have just plotted point (x_k, y_k)
- The next point is a choice between (x_k+1, y_k) and (x_k+1, y_k-1)
- We would like to choose the point that is nearest to the actual circle
- So how do we make this choice?



MIDPOINT CIRCLE ALGORITHM

- Let's re-jig the equation of the circle slightly to give us:

$$f_{circ}(x, y) = x^2 + y^2 - r^2... \qquad (1)$$

- The equation evaluates as follows:

$$f_{circ}(x, y) \begin{cases} < 0, \text{ if } (x, y) \text{ is inside the circle boundary} \\ = 0, \text{ if } (x, y) \text{ is on the circle boundary} \\ > 0, \text{ if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

- By evaluating this function at the midpoint between the candidate pixels we can make our decision

MIDPOINT CIRCLE ALGORITHM

- Assuming we have just plotted the pixel at (x_k,y_k) so we need to choose between (x_k+1,y_k) and (x_k+1,y_k-1)
- Our decision variable can be defined as: mid point between 2 points (x_k+1, y_k) and (x_k+1, y_k-1) is $[x_k+1, y_k-1/2]$

$$p_k = f_{circ}(x_k+1, y_k - \tfrac{1}{2})$$

$$= (x_k+1)^2 + (y_k - \tfrac{1}{2})^2 - r^2 \qquad \ldots(2)$$

- If $p_k < 0$ the midpoint is inside the circle and the pixel at y_k is closer to the circle
- Otherwise the midpoint is outside and y_k-1 is closer

MIDPOINT CIRCLE ALGORITHM

- To ensure things are as efficient as possible we can do all of our calculations incrementally
- First consider: (since $x_k+1 = x_{k+1}$)

$$p_{k+1} = f_{circ}\left(x_{k+1}+1, y_{k+1}-\tfrac{1}{2}\right)$$

$$= [(x_k+1)+1]^2 + \left(y_{k+1}-\tfrac{1}{2}\right)^2 - r^2$$

where y_{k+1} is either y_k or y_k-1 depending on the sign of p_k

MIDPOINT CIRCLE ALGORITHM

- The initial value of pk is given by the circle function at the position (0,r) as,

$$p_k = f_{circ}(x_k + 1, y_k - \tfrac{1}{2})$$
$$= (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2$$

- Substituting k=0, x_k=0, y_k=r in the above function results in,

MIDPOINT CIRCLE ALGORITHM

- The first decision variable is given as:

$$p_0 = f_{circ}(1, r - 1/2)$$
$$= 1 + (r - 1/2)^2 - r^2$$
$$= 5/4 - r$$

If r is an integer, then p_0 can be rounded to $p_0 = 1 - r$.

- Then if $p_k < 0$ then the next decision variable for next point (x_k+1, y_k) is given as:

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

- If $p_k > 0$ then the decision variable for next point (x_k+1, y_k-1) is:

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_k + 1$$

MIDPOINT CIRCLE ALGORITHM

Mid-point circle algorithm

- Input radius r and circle centre (x_c, y_c), then set the coordinates for the first point on the circumference of a circle centred on the origin as:

$$(x_0, y_0) = (0, r)$$

- Calculate the initial value of the decision parameter as:

$$p_0 = 5/4 - r$$

- Perform the test, starting with $k = 0$ at each position x_k, perform the following test.

- (i) if $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and:

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

MIDPOINT CIRCLE ALGORITHM

- (ii) if pk >0 then the next point along the circle is (x_k+1, y_k-1) and:

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$

- Identify the symmetry points in the other seven octants
- Move (x, y) according to:

$$x = x + x_c \qquad y = y + y_c$$

- Repeat steps 3 to 5 until $x >= y$

MIDPOINT CIRCLE ALGORITHM

- To see the mid-point circle algorithm in action lets use it to draw a circle centred at (0,0) with radius 10
- Determine the positions along the circle octant in the first quadrant from x=0 to x=y.
- The intial value of the decision parameter is

$$P_0 = 1 - r = 1 - 10 = -9$$

- For circle centred on the coordinate origin, the initial point is $(X_0, Y_0) = (0,10)$ and initial increment terms for calculating the decision parameters are

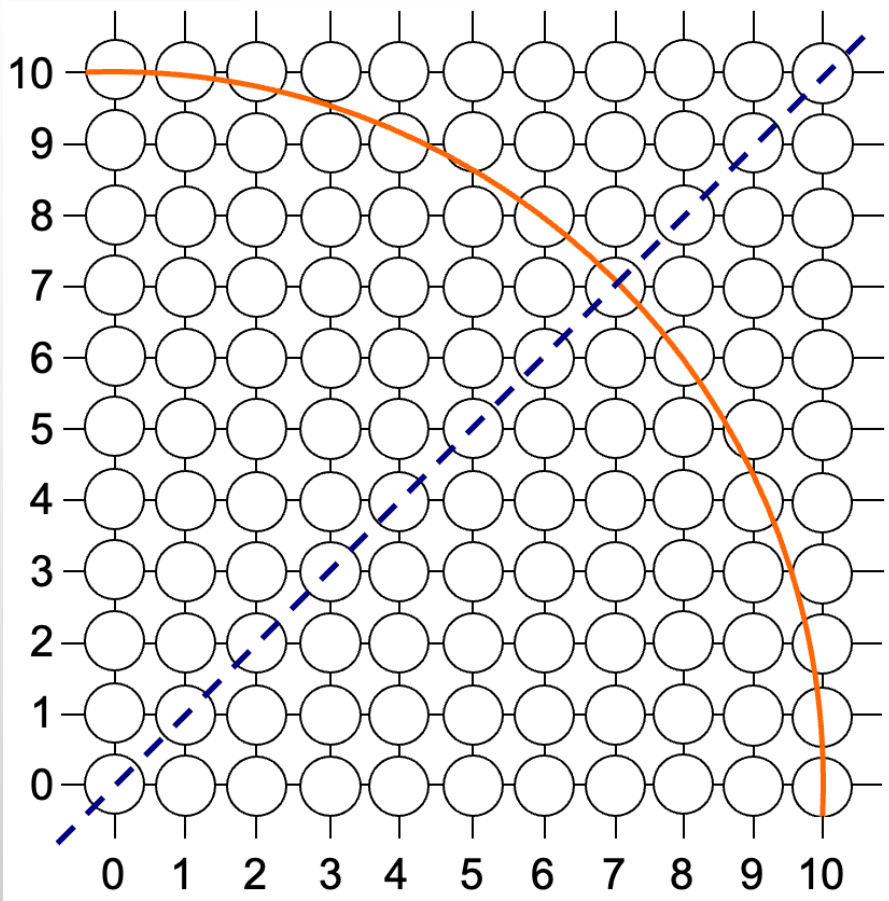
$$2X_0 = 0 \text{ and } 2Y_0 = 20$$

MIDPOINT CIRCLE ALGORITHM

- k=0 and $p_0 = -9$ (1,10)
- $p_{k+1} = p_k + 2x_{k+1} + 1$ ($p_k<0$)
- k=1, $p_1=p_0+2x_k+1$ => -9+2(1)+1 =-9+3=-6 (2,10)
- k=2, $p_2= p_1 +2(2)+1$ = -6+4+1 = -1 (3,10)
- k=3 $p_3=p_2+2(3)+1$ = -1+7= 6 (4,9)
- k=4 $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$ ($p_k>0$)

  $p_4= p_3+2(4)+1-2(9)$ => 6+8+1-18 = -3 (5,9)
- k=5 $p_5= p_4+2(5)+1$ => -3+10+1 = 8 (6,8)
- k=6 $p_6=8+2(6)+1-2(8)$ => 8+12+1-16 = 5 (7,7)
- k=7 $p_7= 6$ (8,6)
- k=8 $p_8=11$ (9,5)
- k=9 $p_9 =20$ (10,4)

MIDPOINT CIRCLE ALGORITHM

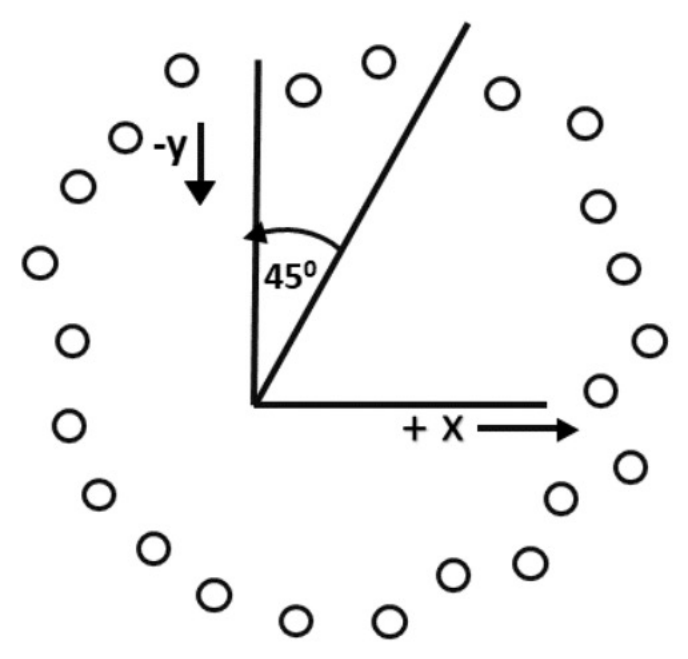


k	p_k	(x_{k+1},y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1,10)	2	20
1	-6	(2,10)	4	20
2	-1	(3,10)	6	20
3	6	(4,9)	8	18
4	-3	(5,9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14

BRESENHAM'S CIRCLE ALGORITHM

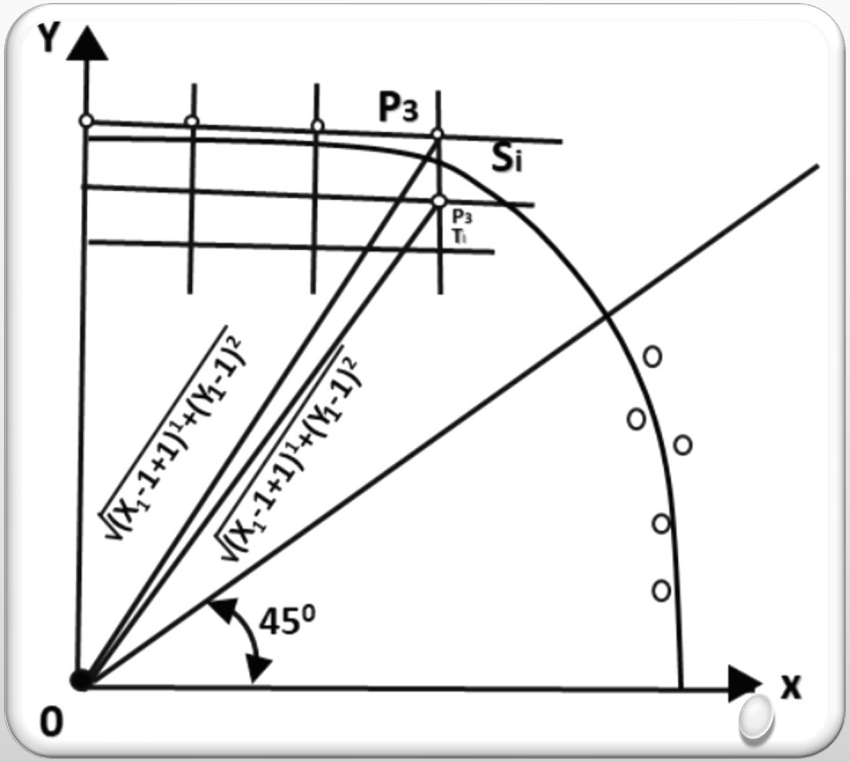
- It considers the eight-way symmetry of the circle to generate it.
- It plots $1/8^{th}$ part of the circle, i.e. from 90° to 45°.
- As circle is drawn from 90° to 45°, the x moves in positive direction and y moves in negative direction.

BRESENHAM'S CIRCLE ALGORITHM

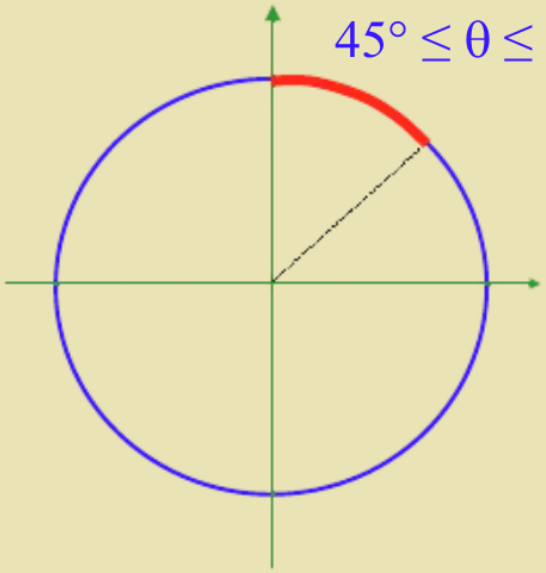


BRESENHAM'S CIRCLE ALGORITHM

- The best approximation of the true circle will be described by those pixels in the raster that falls the least distance from the true circle.
- We want to generate the points from 90° to 45°.
- Assume that the last scan-converted pixel is P1 as shown in fig. Each new point closest to the true circle can be found by taking either of two actions.
  - Move in the x-direction one unit
  - Move in the x- direction one unit & move in the negative y-direction one unit.



BRESENHAM'S CIRCLE ALGORITHM



General Principle

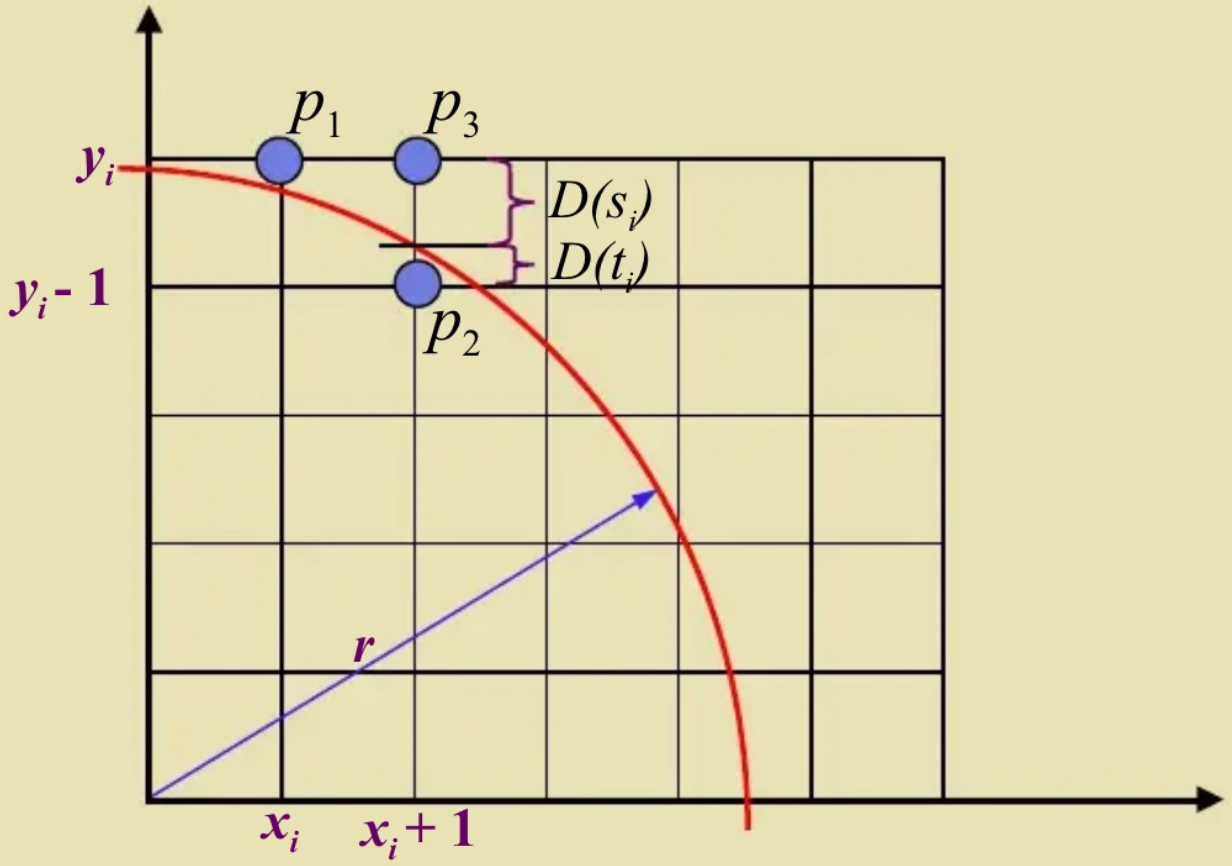
- The circle function:

$$f_{circle}(x,y) = x^2 + y^2 - r^2$$

- and

$$f_{circle}(x,y) = \begin{cases} < 0 & \text{if (x,y) is inside the circle boundary} \\ = 0 & \text{if (x,y) is on the circle boundary} \\ > 0 & \text{if (x,y) is outside the circle boundary} \end{cases}$$

BRESENHAM'S CIRCLE ALGORITHM



After point p_1, do we choose p_2 or p_3?

BRESENHAM'S CIRCLE ALGORITHM

Define: $D(s_i)$ = distance of p_3 from circle

$D(t_i)$ = distance of p_2 from circle

i.e. $D(s_i) = (x_i + 1)^2 + y_i^2 - r^2$ [always +ve]

$D(t_i) = (x_i + 1)^2 + (y_i - 1)^2 - r^2$ [always -ve]

- Decision Parameter $p_i = D(s_i) + D(t_i)$

so if $p_i < 0$ then the circle is closer to p_3 (point above)

if $p_i \geq 0$ then the circle is closer to p_2 (point below)

BRESENHAM'S CIRCLE ALGORITHM

$$p_i = D(s_i) + D(t_i)$$

$$p_i = 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 - 2r^2$$

Calculating next decision parameter,

$$p_{i+1} = 2(x_i+2)^2 + y_{i+1}^2 + (y_{i+1}-1)^2 - 2r^2$$

$$p_{i+1} - p_i = 2((x_i+2)^2-(x_i+1)^2) + (y_{i+1}^2 - y_i^2) + ((y_{i+1}-1)^2 + (y_i-1)^2)$$

$$p_{i+1} = p_i + 2((x_i+2+x_i+1)(x_i+2-x_i-1)) + ((y_{i+1}+y_i)(y_{i+1}-y_i)) + ((y_{i+1}-1+y_i-1)(y_{i+1}-1-y_i+1))$$

BRESENHAM'S CIRCLE ALGORITHM

if ($p_i <= 0$)

$x_{i+1} = x_i + 1$ and $y_{i+1} = y_i$

so that $p_{i+1} = p_i + 2(2x_i + 3) + ((y_{i+1}+y_i)(y_i - y_i)) + ((y_i - 1 + y_i - 1)(y_i - 1 - y_i + 1))$

$p_{i+1} = p_i + 2(2x_i + 3) + ((y_{i+1}+y_i)(0)) + ((y_i - 1 + y_i - 1)(0))$

$p_{i+1} = p_i + 4x_i + 6$

Else

$x_{i+1} = x_i + 1$ and $y_{i+1} = y_{i-1}$

$p_{i+1} = p_i + 2(2x_i + 3) + ((y_i - 1 + y_i)(y_i - 1 - y_i)) + ((y_i - 2 + y_i - 1)(y_i - 2 - y_i + 1))$

$p_{i+1} = p_i + 4x_i + 6 + ((2y_i - 1)(-1)) + ((2y_i - 3)(-1))$

$p_{i+1} = p_i + 4x_i + 6 - 2y_i - 2y_i + 1 + 3$

$p_{i+1} = p_i + 4(x_i - y_i) + 10$

CALCULATE INITIAL VALUE

The initial value of p_i can be obtained by replacing x=0 and y=r. Thus, we get,

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

$$p_o = 2 + r^2 + r^2 + 1 - 2r - 2r^2$$

$$p_o = 3 - 2r$$

BRESENHAM'S CIRCLE ALGORITHM

$x_0 = 0$

$y_0 = r$

$p_0 = [1^2 + r^2 - r^2] + [1^2 + (r\text{-}1)^2 - r^2] = 3 - 2r$

if $p_i < 0$ *then*

$y_{i+1} = y_i$

$p_{i+1} = p_i + 4x_i + 6$

else if $p_i \geq 0$ *then*

$y_{i+1} = y_i - 1$

$p_{i+1} = p_i + 4(x_i - y_i) + 10$

$x_{i+1} = x_i + 1$

- Stop when $x_i \geq y_i$ and determine symmetry

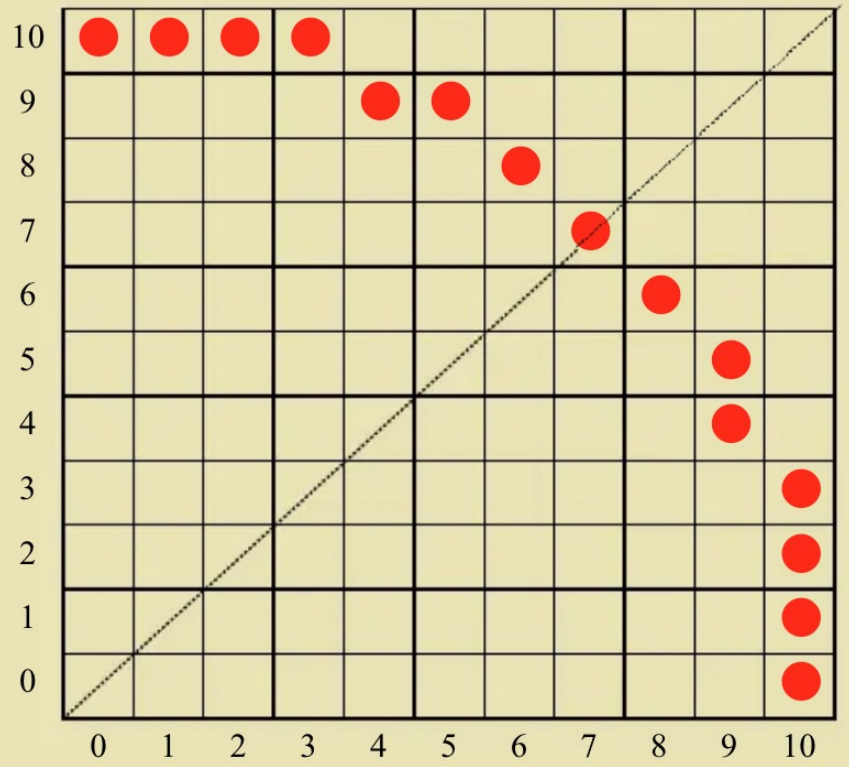
EXAMPLE

$r = 10$

$p_0 = 3 - 2r = -17$

Initial point $(x_0, y_0) = (0, 10)$

i	p_i	x_i, y_i
0	-17	(0, 10)
1	-11	(1, 10)
2	-1	(2, 10)
3	13	(3, 10)
4	-5	(4, 9)
5	15	(5, 9)
6	9	(6, 8)
7		(7,7)



BRESENHAM'S CIRCLE ALGORITHM

- Step 1 : set initial values of (x_c, y_c) and (x, y)
- Step 2 : calculate decision parameter d to d = 3 – (2 * r).
- Step 3 : call drawcircle(int x_c, int y_c, int x, int y) method to display initial(0,r) point.
- Step 4 : repeat steps 5 to 8 until x < = y
- Step 5 : increment value of x.
- Step 6 : if d < 0, set d = d + (4*x) + 6
- Step 7 : else, set d = d + 4 * (x – y) + 10 and decrement y by 1.
- Step 8 : call drawcircle(int x_c, int y_c, int x, int y) method.
- Step 9 : exit.

BRESENHAM'S CIRCLE ALGORITHM

```
drawcircle(int xc, int yc, int x, int y)
{
    putpixel(xc+x, yc+y, RED);
    putpixel(xc -x, yc +y, RED);
    putpixel(xc +x, yc -y, RED);
    putpixel(xc -x, yc -y, RED);
    putpixel(xc +y, yc +x, RED);
    putpixel(xc -y, yc +x, RED);
    putpixel(xc +y, yc -x, RED);
    putpixel(xc -y, yc -x, RED);
}
```