CSE 421

Lecture 04

Circle Drawings

Circle Drawing

Since the circle is a frequently used component in pictures and graphs, a procedure for generating either full circles or circular arc is included in most graphics packages.

Direct Algorithm

The equation for a circle is:

$$\mathbf{x^2 + y^2 = r^2}$$

Where **r** is the **radius** of the circle. So, we can write a direct circle drawing algorithm by solving the equation for **y** at unit **x** intervals using:

$$\mathbf{y = \pm(r^2 - x^2)^{1/2}}$$

Direct Algorithm

To draw a circle with radius=20

$y(0) = \pm(20^2 - 0^2)^{1/2} \cong 20$

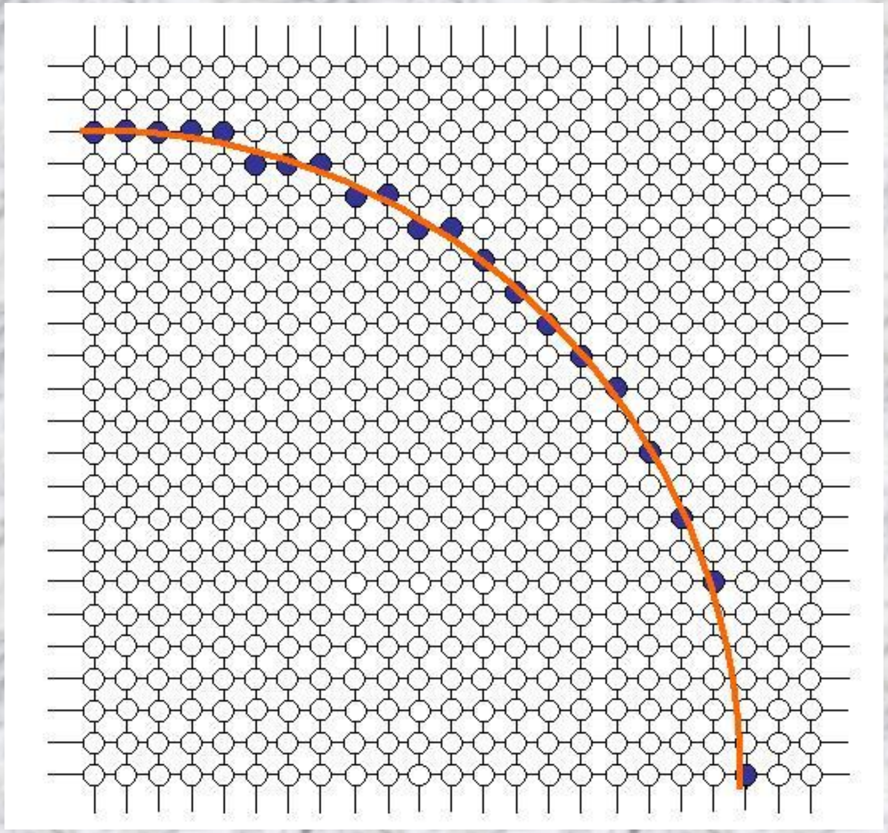
$y(1) = \pm(20^2 - 1^2)^{1/2} \cong 20$

$y(2) = \pm(20^2 - 2^2)^{1/2} \cong 20$

.....

$y(19) = \pm(20^2 - 19^2)^{1/2} \cong 6$

$y(20) = \pm(20^2 - 20^2)^{1/2} = 0$

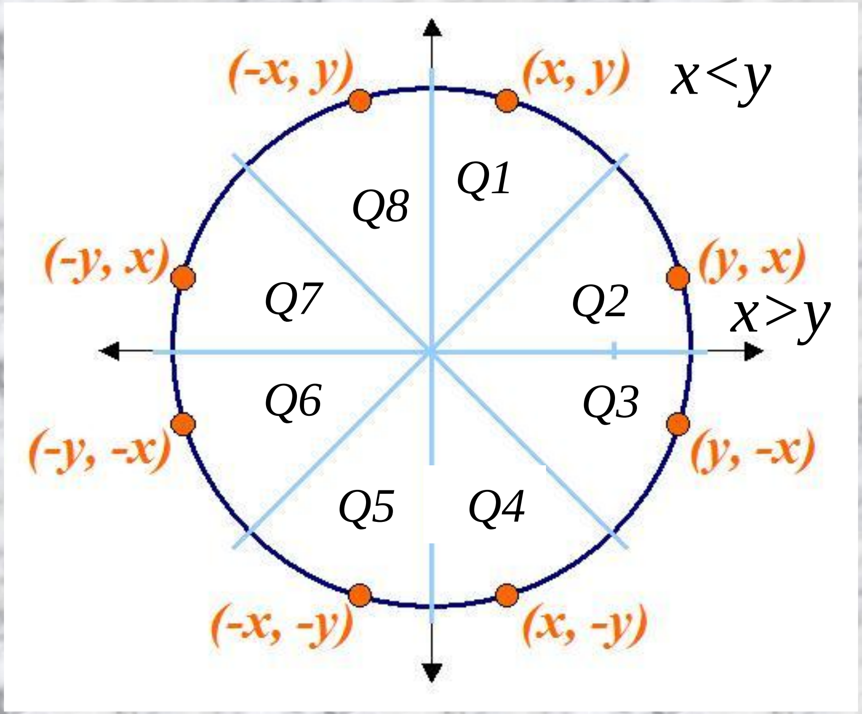


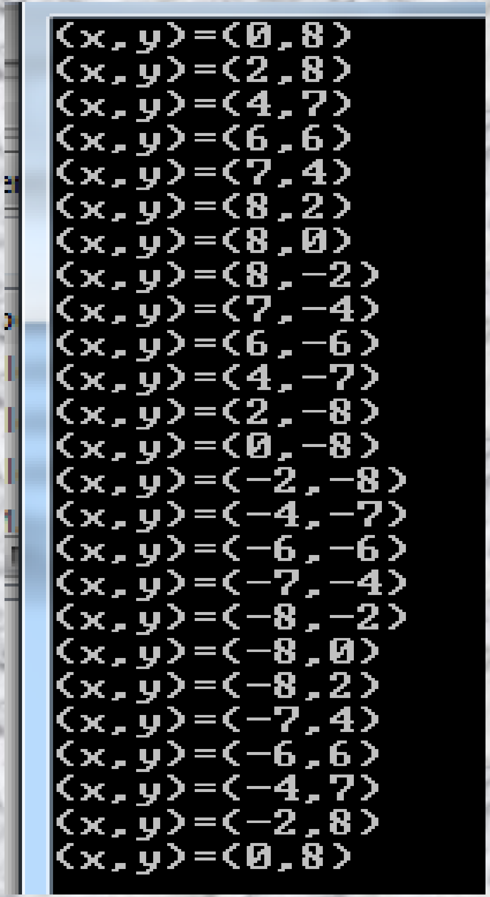
the resulting circle has large gaps where the slope approaches the vertical. And the calculations are not very efficient:

- ✓ The square (multiply) operations
- ✓ The square root operation

Eight-Way Symmetry

The first thing we can notice to make our circle drawing algorithm more efficient is that circles centred at (0, 0) have eight-way symmetry. The figure shows a circle with different coordinate values 15^0 apart with r=8.





```
(x,y)=(0,8)
(x,y)=(2,8)
(x,y)=(4,7)
(x,y)=(6,6)
(x,y)=(7,4)
(x,y)=(8,2)
(x,y)=(8,0)
(x,y)=(8,-2)
(x,y)=(7,-4)
(x,y)=(6,-6)
(x,y)=(4,-7)
(x,y)=(2,-8)
(x,y)=(0,-8)
(x,y)=(-2,-8)
(x,y)=(-4,-7)
(x,y)=(-6,-6)
(x,y)=(-7,-4)
(x,y)=(-8,-2)
(x,y)=(-8,0)
(x,y)=(-8,2)
(x,y)=(-7,4)
(x,y)=(-6,6)
(x,y)=(-4,7)
(x,y)=(-2,8)
(x,y)=(0,8)
```

x<y

Mid-Point Circle Algorithm

As in the raster line algorithm, we sample at unit intervals and determine the closest pixel position to the specified circle path at each step.

- For a given radius **r** and screen center position ($\mathbf{x}_c$, $\mathbf{y}_c$), we can first set up our algorithm to calculate pixel positions around a circle path centered at the coordinate (0, 0).

- Then each calculated position (x, y) is moved to its proper screen position by adding $\mathbf{x}_c$ to **x** and $\mathbf{y}_c$ to **y**.

Mid-Point Circle Algorithm

- Along the circle section from x = 0 to x = y in the first quadrant, the slope of the curve varies from 0 to –1. Therefore, we can take unit steps in the positive x direction over this octant and use a decision parameter to determine which of the two possible y positions is closer to the circle path at each step positions in the other seven octants are then obtained by symmetry.

- To apply the midpoint algorithm, we define a circle function:

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

Mid-Point Circle Algorithm

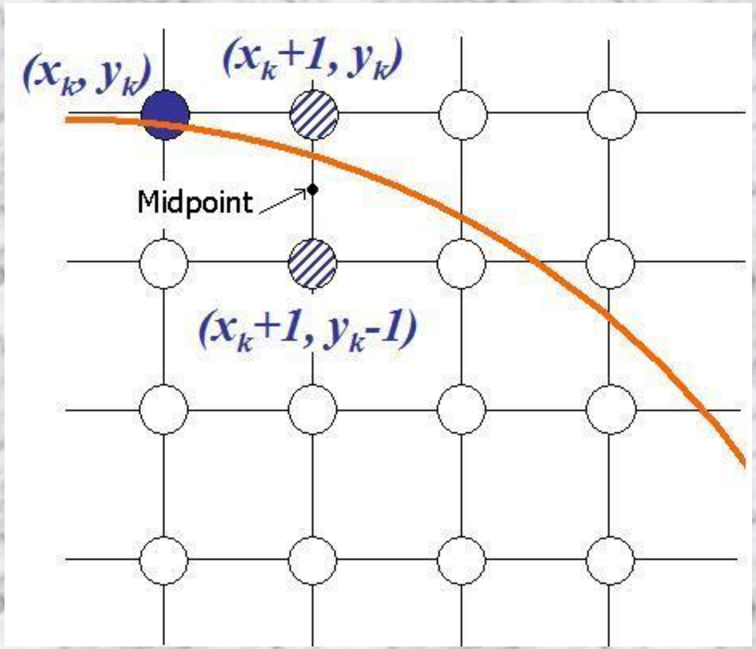
- The relative position of any point (x, y) can be determined by checking the sign of the on the boundary of the circle function:

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

The circle function tests are performed for the midpoints positions between pixels near the circle path at each sampling step. Thus, circle function is decision parameter in the midpoint algorithm, and we can set up incremental calculations for this function as we did in the line algorithm.

Mid-Point Circle Algorithm

Assuming we have just plotted point (**x_k, y_k**), we next need to determine whether the pixel at position (**x_k+1, y_k**) or the one at position (**x_k+1, y_k–1**) is closer to the circle path. Our decision parameter is The circle function evaluated at the midpoint between these two pixels:



$$\text{Mid point}=\left(\frac{x_k+1+x_k+1}{2}, \frac{y_k+y_k-1}{2}\right)$$

$$=\left(x_k+1, y_k-\frac{1}{2}\right)$$

$$\mathbf{X_{k+1}=X_k+1}$$

$$y_{k+1}= \begin{matrix} y_k \\ y_{k-1} \end{matrix}$$

Mid-Point Circle Algorithm

$$p_k = f_{circle}(x_k + 1, y_k - 1/2)$$

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

Similarly

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

Now

$$p_{k+1} - p_k = (x_{k+1} + 1)^2 + (y_{k+1} - 1/2)^2 - r^2 - (x_k + 1)^2 - (y_k - 1/2)^2 + r^2$$

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

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

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

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

$$\boxed{p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1}$$

Mid-Point Circle Algorithm

$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$

At Starting Point (0,r),

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

$= (0 + 1)^2 + (r - 1/2)^2 - r^2$

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

$= 5/4 - r$

$= 1 - r$ (take integral part)

Initial decision parameter $p_k = 1 - r$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$ (mid point outside the circle)
So Next Pixel $= (x_k + 1, y_k - 1)$

If $p_k < 0 \Rightarrow y_{k+1} = y_k$ (mid point inside the circle)
So Next Pixel $= (x_k + 1, y_k)$

Mid-Point Circle Algorithm

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

Initial decision parameter $\mathbf{p_k = 1 - r}$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$
So Next Pixel=$(\mathbf{x_k + 1, y_k - 1})$

If $p_k < 0 \Rightarrow y_{k+1} = y_k$
So Next Pixel=$(\mathbf{x_k + 1, y_k})$

Example, r=8
$P_0 = 1 - r = 1 - 8 = -7$

k	(x_k, y_k)	p_k	(x_{k+1}, y_{k+1})
0	(0,8)	-7	(1,8)

Mid-Point Circle Algorithm

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

Initial decision parameter $\mathbf{p_k = 1 - r}$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$
So Next Pixel=$(x_k + 1, y_k - 1)$

If $p_k < 0 \Rightarrow y_{k+1} = y_k$
So Next Pixel=$(x_k + 1, y_k)$

Example, r=8
$P_0 = 1 - r = 1 - 8 = -7$

k	(x_k, y_k)	p_k	(x_{k+1}, y_{k+1})
0	(0, 8)	-7	(1, 8)
1	(1, 8)	-4	(2, 8)

$$\begin{aligned} p_{k+1} &= p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \\ &= -7 + 2(0+1) + (8^2 - 8^2) - (8 - 8) + 1 \\ &= -7 + 2 + 1 = -4 \end{aligned}$$

Mid-Point Circle Algorithm

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

Initial decision parameter $p_k = 1 - r$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$
So Next Pixel $= (x_k + 1, y_k - 1)$

If $p_k < 0 \Rightarrow y_{k+1} = y_k$
So Next Pixel $= (x_k + 1, y_k)$

Example, r=8
$P_0 = 1 - r = 1 - 8 = -7$

k	(x_k, y_k)	p_k	(x_{k+1}, y_{k+1})
0	(0, 8)	-7	(1, 8)
1	(1, 8)	-4	(2, 8)
2	(2, 8)	1	(3, 7)

$$\begin{aligned} p_{k+1} &= p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \\ &= -4 + 2(1+1) + (8^2 - 8^2) - (8 - 8) + 1 \\ &= -4 + 4 + 1 = 1 \end{aligned}$$

Mid-Point Circle Algorithm

$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$

Initial decision parameter $p_k = 1 - r$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$
So Next Pixel $= (x_k + 1, y_k - 1)$

If $p_k < 0 \Rightarrow y_{k+1} = y_k$
So Next Pixel $= (x_k + 1, y_k)$

Example, r=8

$P_0 = 1 - r = 1 - 8 = -7$

k	(x_k, y_k)	p_k	(x_{k+1}, y_{k+1})
0	(0, 8)	-7	(1, 8)
1	(1, 8)	-4	(2, 8)
2	(2, 8)	1	(3, 7)
3	(3, 7)	-6	(4, 7)

$$\begin{aligned} p_{k+1} &= p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \\ &= 1 + 2(2+1) + (7^2 - 8^2) - (7 - 8) + 1 \\ &= 1 + 6 - 15 + 1 + 1 = -6 \end{aligned}$$

Mid-Point Circle Algorithm

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

Initial decision parameter $\mathbf{p_k = 1 - r}$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$
So Next Pixel=$\mathbf{(x_k + 1, y_k - 1)}$

If $p_k < 0 \Rightarrow y_{k+1} = y_k$
So Next Pixel=$\mathbf{(x_k + 1, y_k)}$

Example, r=8
$P_0 = 1 - r = 1 - 8 = -7$

k	(x_k, y_k)	p_k	(x_{k+1}, y_{k+1})
0	(0, 8)	-7	(1, 8)
1	(1, 8)	-4	(2, 8)
2	(2, 8)	1	(3, 7)
3	(3, 7)	-6	(4, 7)
4	(4, 7)	3	(5, 6)

$$\begin{aligned} p_{k+1} &= p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \\ &= -6 + 2(3+1) + (7^2 - 7^2) - (7 - 7) + 1 \\ &= -6 + 8 + 0 - 0 + 1 = 3 \end{aligned}$$

Mid-Point Circle Algorithm

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

Initial decision parameter $p_k = 1 - r$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$
So Next Pixel $= (x_k + 1, y_k - 1)$

If $p_k < 0 \Rightarrow y_{k+1} = y_k$
So Next Pixel $= (x_k + 1, y_k)$

Example, r=8
$P_0 = 1 - r = 1 - 8 = -7$

k	(x_k, y_k)	p_k	(x_{k+1}, y_{k+1})
0	(0, 8)	-7	(1, 8)
1	(1, 8)	-4	(2, 8)
2	(2, 8)	1	(3, 7)
3	(3, 7)	-6	(4, 7)
4	(4, 7)	3	(5, 6)
5	(5, 6)	2	(6,5)

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$
$$= 3 + 2(4+1) + (6^2 - 7^2) - (6 - 7) + 1$$
$$= 3 + 10 - 13 + 1 + 1 = 2$$

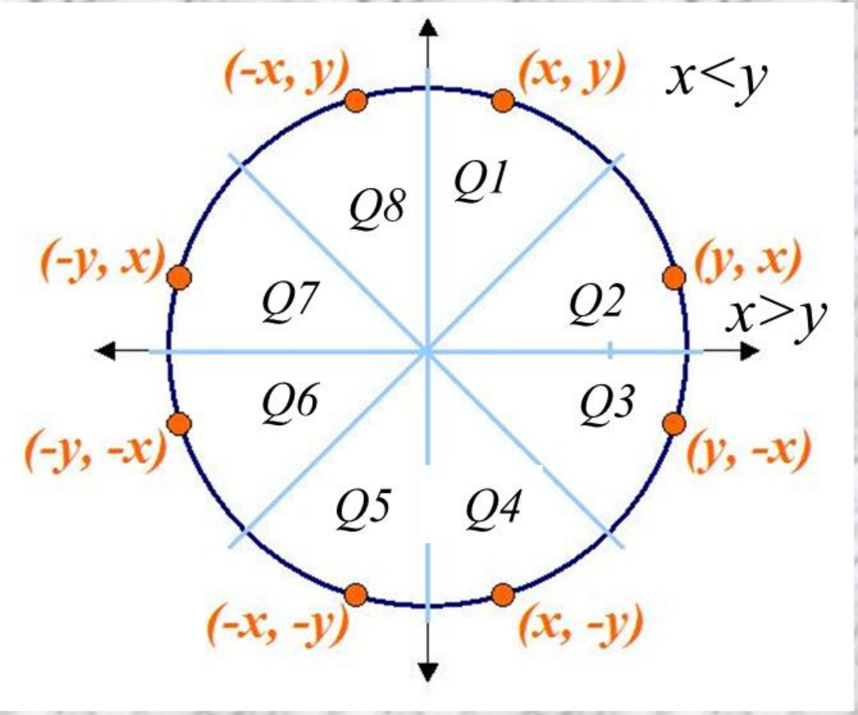
Mid-Point Circle Algorithm

$\mathbf{p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1}$

Initial decision parameter $\mathbf{p_k = 1 - r}$

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k - 1$
So Next Pixel=$\mathbf{(x_k + 1, y_k - 1)}$

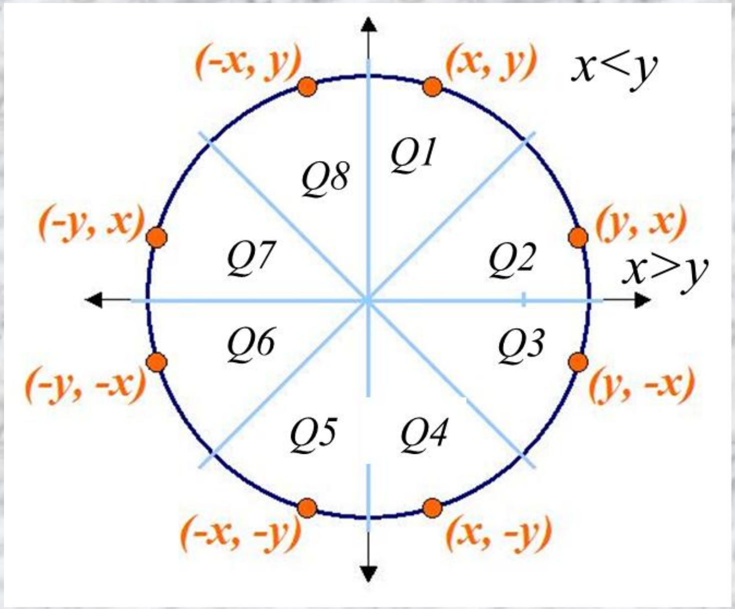
If $p_k < 0 \Rightarrow y_{k+1} = y_k$
So Next Pixel=$\mathbf{(x_k + 1, y_k)}$



k	**(x_k, y_k)**	**p_k**	**(x_{k+1}, y_{k+1})**
0	(0, 8)	-7	(1, 8)
1	(1, 8)	-4	(2, 8)
2	(2, 8)	1	(3, 7)
3	(3, 7)	-6	(4, 7)
4	(4, 7)	3	(5, 6)
5	(5, 6)	2	(6,5)

Outside Q1 as *x>y*

Mid-Point Circle Algorithm



Q1 (x , y)	Q2 (x , y)	Q3 (x , y)	Q4 (x , y)	Q5 (x , y)	Q6 (x , y)	Q7 (x , y)	Q8 (x , y)
(0, 8)	(8, 0)	(8, -0)	(0, -8)	(-0, -8)	(-8, -0)	(-8, 0)	(-0, 8)
(1, 8)	(8, 1)	(8, -1)	(1, -8)	(-1,- 8)	(-8, -1)	(-8, 1)	(-1, 8)
(2, 8)	(8, 2)	(8, -2)	(2, -8)	(-2,- 8)	(-8, -2)	(-8, 2)	(-2, 8)
(3, 7)	(7, 3)	(7, -3)	(3, -7)	(-3, -7)	(-7, -3)	(-7, 3)	(-3, 7)
(4, 7)	(7, 4)	(7, -4)	(4, -7)	(-4, -7)	(-7, -4)	(-7, 4)	(-4, 7)
(5, 6)	(6, 5)	(6, -5)	(5, -6)	(-5, -6)	(-6, -5)	(-6, 5)	(-5, 6)

Quadrant-wise pixel distribution

Mid-Point Circle Algorithm

$$x_{k+1} = x_k + 1 \qquad y_{k+1} = \begin{cases} y_k & p_k < 0 \\ y_{k-1} & p_k >= 0 \end{cases}$$

When $p_k >= 0$

$$\begin{aligned} p_{k+1} &= p_k + 2(x_k + 1) + (y_{k+1}{}^2 - y_k{}^2) - (y_{k+1} - y_k) + 1 \\ &= p_k + 2(x_k + 1) + ((y_k - 1)^2 - y_k{}^2) - (y_k - 1 - y_k) + 1 \\ &= p_k + 2(x_k + 1) + ((y_k{}^2 - 2y_k + 1) - y_k{}^2) - (y_k - 1 - y_k) + 1 \\ &= p_k + 2(x_k + 1) + (-2y_k + 1) + 1 + 1 \\ &= p_k + 2x_k - 2y_k + 5 \end{aligned}$$

When $p_k < 0$

$$\begin{aligned} p_{k+1} &= p_k + 2(x_k + 1) + (y_{k+1}{}^2 - y_k{}^2) - (y_{k+1} - y_k) + 1 \\ &= p_k + 2(x_k + 1) + (y_k{}^2 - y_k{}^2) - (y_k - y_k) + 1 \\ &= p_k + 2(x_k + 1) + (0) - (0) + 1 \\ &= p_k + 2x_k + 3 \end{aligned}$$