Computer Graphics
Lecture-5
Scan Conversion-3

Tasnimatul Jannah
Lecturer, SoSET
East Delta University

A Simple Circle Drawing Algorithm

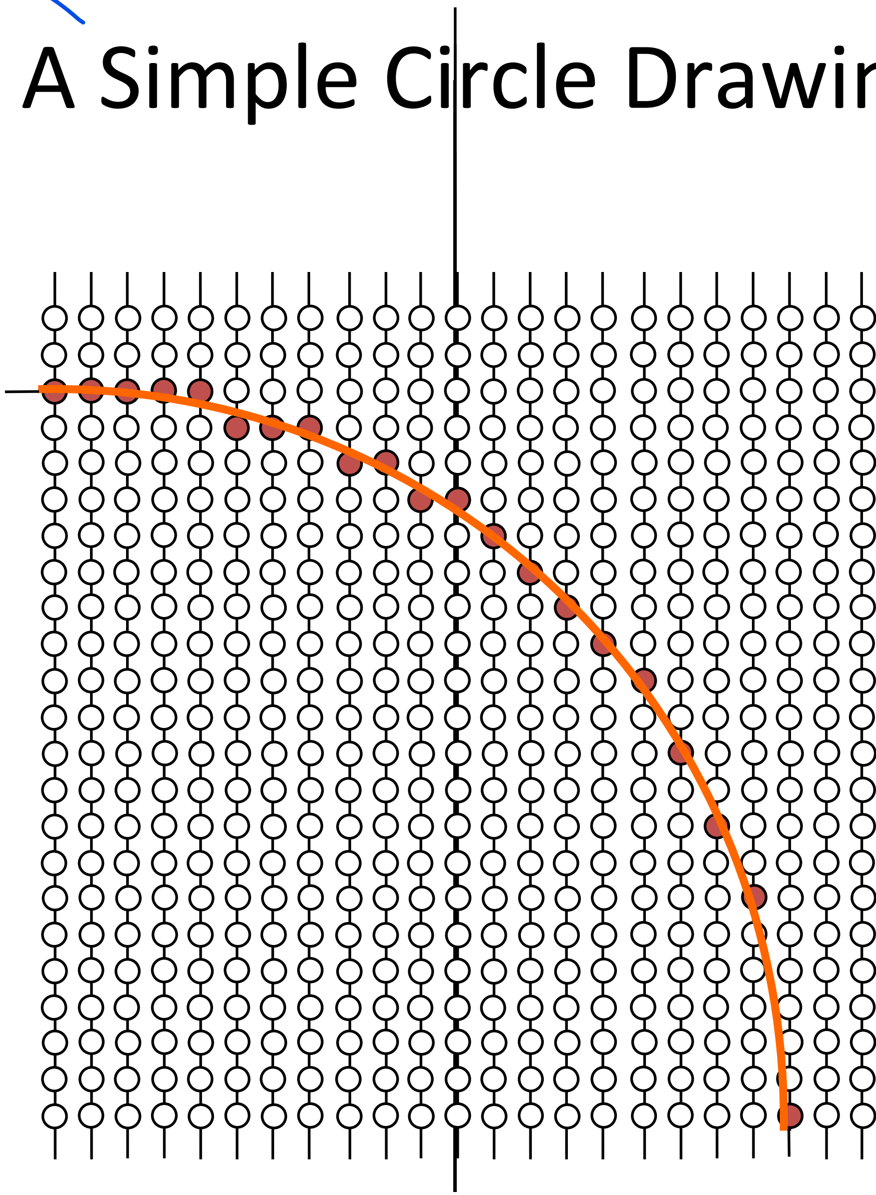
- The equation for a circle is:

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

- 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:

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

A Simple Circle Drawing Algorithm (cont…)



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

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

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

$$\vdots$$

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

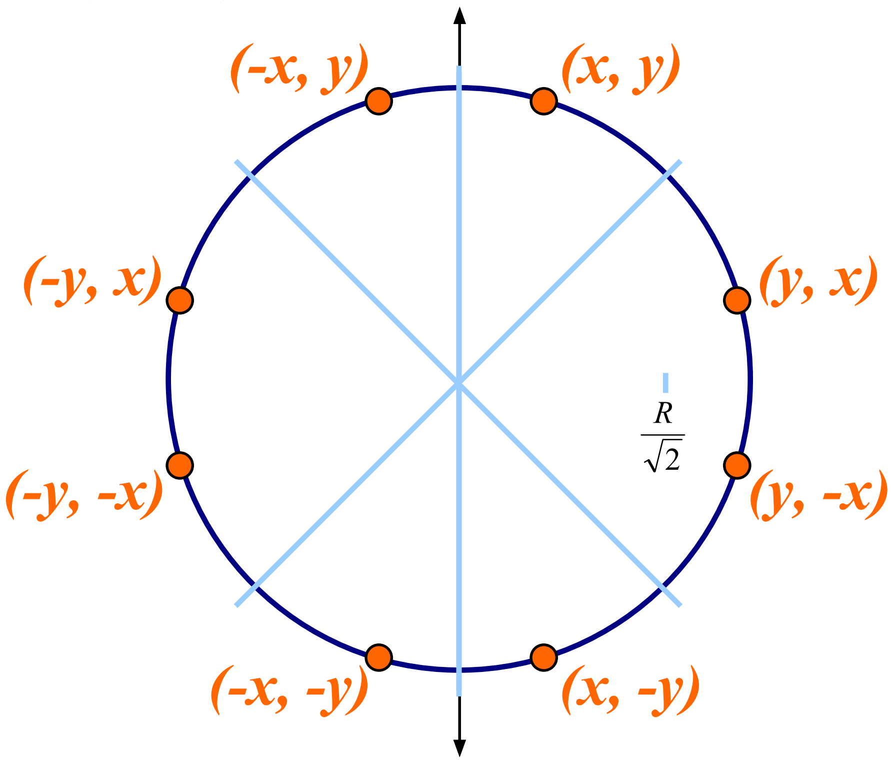
$$y_{20} = \sqrt{20^2 - 20^2} \approx 0$$

A Simple Circle Drawing Algorithm (cont…)

- 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

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*

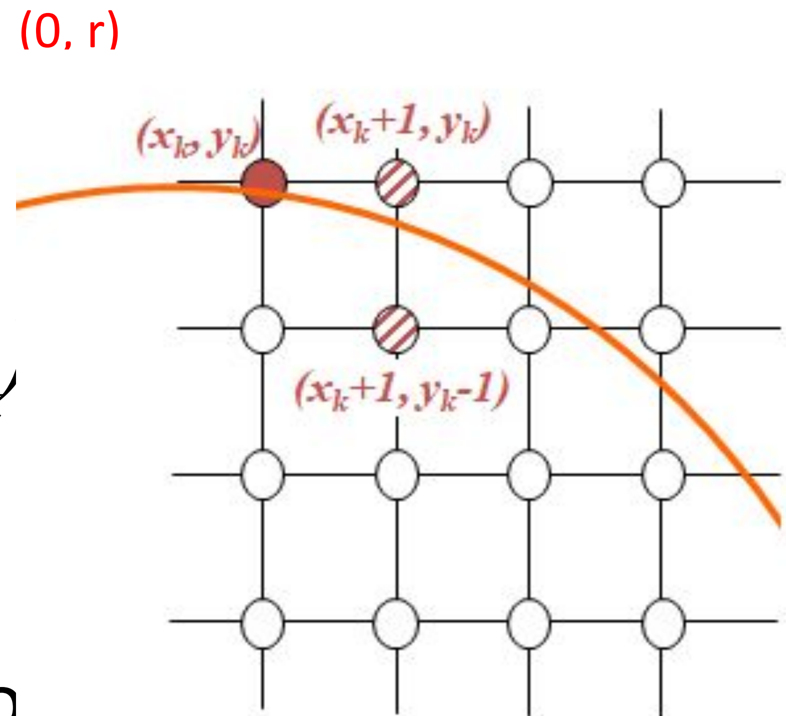


Mid-Point Circle Algorithm

- Similarly to the case with lines, there is an incremental algorithm for drawing circles – the *mid-point circle algorithm*
- In the mid-point circle algorithm we use eight-way symmetry so only ever calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points

Mid-Point Circle Algorithm (cont…)

- 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?



Mid-Point Circle Algorithm (cont…)

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

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

- 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

Mid-Point Circle Algorithm (cont…)

- 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:

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

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

Mid-Point Circle Algorithm (cont…)

- To ensure things are as efficient as possible we can do all of our calculations incrementally
- First consider: $p_{k+1} = f_{circ}\left(x_{k+1}+1, y_{k+1} - \frac{1}{2}\right)$

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

- or:

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

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

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

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

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

$$= x_k^2 + 4x_k + 4 - x_k^2 - 2x_k - 1 + y_{k+1}^2 - y_{k+1} + \tfrac{1}{4}$$
$$- y_k^2 + y_k + \tfrac{1}{4}$$

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

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

$$\left[\; 2x_k + 3 = 2x_k + 2 + 1 = 2(x_k + 1) + 1 \;\right]$$
$$= 2x_{k+1} + 1$$

Mid-Point Circle Algorithm (cont…)

- The first decision variable is given as:

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

- Then if $p_k < 0$ then the next decision variable is given as: $p_{k+1} = p_k + 2x_{k+1} + 1$

- If $p_k > 0$ then the decision variable is:

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

$P_k > 0$, plot $(x_{k+1}, y_k - 1)$

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

$$\{y_k - 1 - y_k\}$$

$$= P_k + 2x_{k+1} + 3 + (y_k^2 - 2y_k + 1 - y_k^2) + 1$$

$$= P_k + 2x_{k+1} + 3 - 2y_k + 1 + 1$$

$$= P_k + 2x_{k+1} + 3 - 2y_k + 2$$

$$= P_k + (2x_{k+1} + 3) + 1 - 2y_k + 1$$

$$= P_k + 2x_{k+1} + 1 - 2y_k + 1$$

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

$$\therefore 2y_{k+1} = 2(y_k - 1) = 2y_k - 2$$

Mid-point Circle Algorithm - Steps

1. Input radius $\mathbf{r}$ and circle center $(\mathbf{x_c}, \mathbf{y_c})$. set the first point $(x_0, y_0) = (0, r)$.

1. Calculate the initial value of the decision parameter as $\mathbf{p_0 = 1 - r}$.
   $(p_0 = 5/4 - r \cong 1 - r)$

3. If $\mathbf{p_k < 0}$,
   plot $(\mathbf{x_k + 1, y_k})$ and $\mathbf{p_{k+1} = p_k + 2x_{k+1} + 1}$,

   Otherwise,

   plot $(\mathbf{x_k + 1, y_k - 1})$ and $\mathbf{p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}}$,

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

Mid-point Circle Algorithm - Steps

4. Determine symmetry points on the other seven octants.

4. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values: $\boldsymbol{x = x + x_c}$ **,** $\boldsymbol{y = y + y_c}$

4. Repeat steps 3 though 5 until $\boldsymbol{x \geq y}$.

4. For all points, add the center point (x_c, y_c)

Mid-point Circle Algorithm - Steps

- Now we drew a part from circle, to draw a complete circle, we must plot the other points.
- We have $(x_c + x , y_c + y)$, the other points are:
  - $(x_c - x , y_c + y)$
  - $(x_c + x , y_c - y)$
  - $(x_c - x , y_c - y)$
  - $(x_c + y , y_c + x)$
  - $(x_c - y , y_c + x)$
  - $(x_c + y , y_c - x)$
  - $(x_c - y , y_c - x)$

Mid-point circle algorithm (Example)

- Given a circle radius $r = 10$, demonstrate the midpoint circle algorithm by determining positions along the circle octant in the first quadrant from $x = 0$ to $x = y$.

<u>Solution:</u>

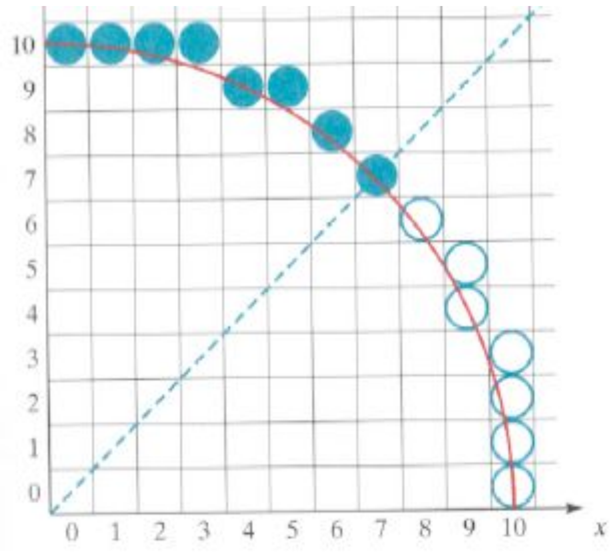
- $p_0 = 1 - r = -9$
- Plot the initial point $(x_0, y_0) = (0, 10)$,

- $2x_0 = 0$ and $2y_0 = 20$.
- Successive decision parameter values and positions along the circle path are calculated using the midpoint method as appear in the next table:

Mid-point circle algorithm (Example)

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 –	(10 ,3)	6	20
3	6	(9 ,4)	8	18
4	3 –	(9 ,5)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14

Mid-point circle algorithm (Example)



Mid-point Circle Algorithm – Example (2)

- Given a circle radius $r = 15$, demonstrate the midpoint circle algorithm by determining positions along the circle octant in the first quadrant from $x = 0$ to $x = y$.

<u>Solution</u>:

- $p_0 = 1 – r = – 14$
- plot the initial point $(x_0 , y_0) = (0, 15)$,
- $2x_0 = 0$ and $2y_0 = 30$.
- Successive decision parameter values and positions along the circle path are calculated using the midpoint method as:

Mid-point Circle Algorithm – Example (2)

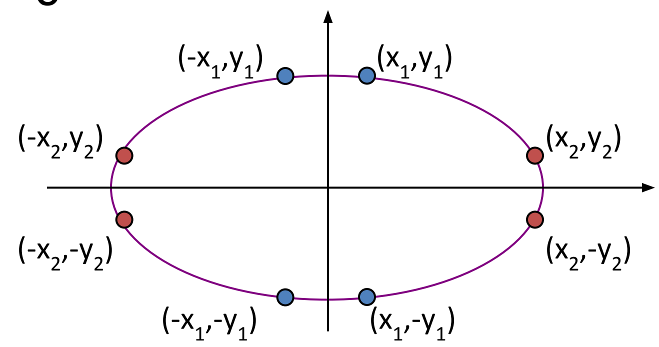
K	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	– 14	(1, 15)	2	30
1	– 11	(2, 15)	4	30
2	– 6	(3, 15)	6	30
3	1	(4, 14)	8	28
4	– 18	(5, 14)	10	28

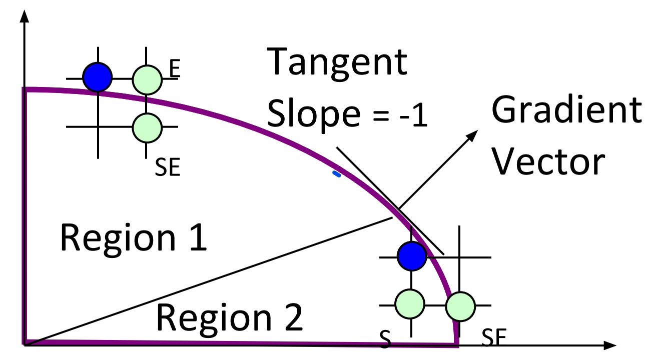
Mid-point Circle Algorithm – Example (2)

K	P_k	(x_{k+1}, y_{k+1})	x_{k+1} 2	y_{k+1} 2
5	7 –	(6,14)	12	28
6	6	(7,13)	14	26
7	5 –	(8,13)	16	26
8	12	(9,12)	18	24
9	7	(10,11)	20	22
10	6	(11,10)	22	20

Midpoint Ellipse Algorithm

- Implicit equation is: $F(x,y) = b^2x^2 + a^2y^2 - a^2b^2 = 0$
- We have only 4-way symmetry
- There exists two regions
  - In **Region 1** dx > dy
    - Increase x at each step
    - y may decrease
  - In **Region 2** dx < dy
    - Decrease y at each step
    - x may increase
- At region boundary:





$$2 \cdot x \cdot b^2 + 2 \cdot y \cdot a^2 \cdot \frac{dy}{dx} = 0$$

$$\therefore \frac{dy}{dx} = -\frac{b^2 \cdot x}{a^2 \cdot y}$$

$$\textit{In} \quad \textit{Region} \quad 1 \quad \left|\frac{dy}{dx}\right| < 1$$

$$\therefore b^2 \cdot x < a^2 \cdot y$$

Midpoint Ellipse Algorithm

- **In region 1**

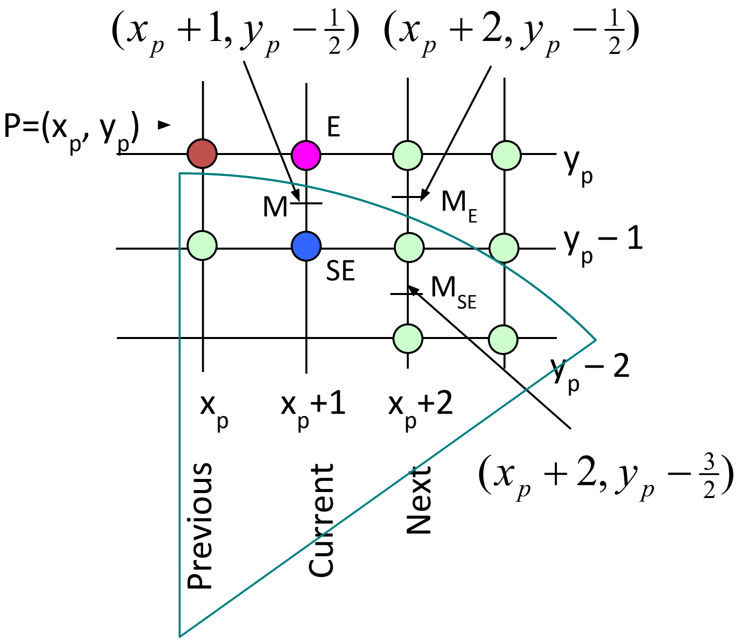
$$d = F(x_p + 1, y_p - \tfrac{1}{2})$$
$$= b^2(x_p+1)^2 + a^2(y_p - \tfrac{1}{2})^2 - a^2b^2$$

if $d < 0$ then move to E

$$d_{new} = F(x_p + 1, y_p)$$
$$= b^2(x_p+1)^2 + a^2(y_p)^2 - a^2b^2$$

if $d > 0$ then move to SE

$$d_{new} = F(x_p + 1, y_p - 1)$$
$$= b^2(x_p+1)^2 + a^2(y_p - 1)^2 - a^2b^2$$



Midpoint Ellipse Algorithm

- **In region 2**

$$d = F(x_p + \tfrac{1}{2}, y_p - 1)$$
$$= b^2(x_p + \tfrac{1}{2})^2 + a^2(y_p - 1)^2 - a^2b^2$$

if $d < 0$ then move to S

$$d_{new} = F(x_p, y_p - 1)$$
$$= b^2(x_p)^2 + a^2(y_p - 1)^2 - a^2b^2$$

if $d > 0$ then move to SE

$$d_{new} = F(x_p + 1, y_p - 1)$$
$$= b^2(x_p + 1)^2 + a^2(y_p - 1)^2 - a^2b^2$$