MID POINT CIRCLE DRAWING ALGORITHM

Nancy Victor
Assistant Professor
SITE, VIT University,
Vellore

A Simple Circle Drawing Algorithm

- The equation for a circle is:

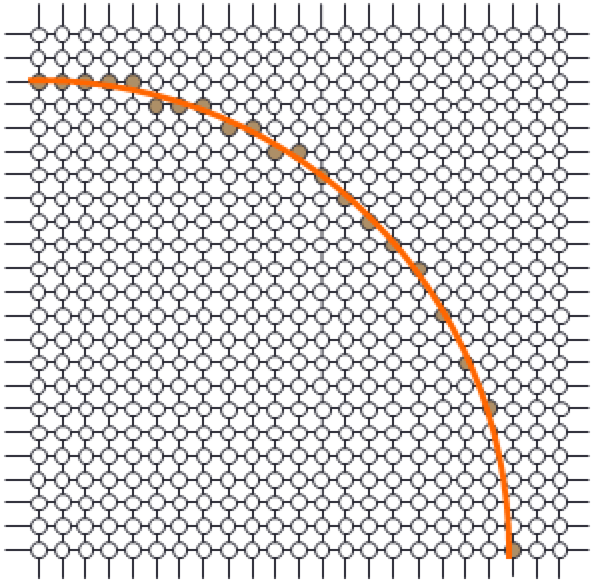
$$x^2 + y^2 = r^2$$

  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…)



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

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

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

$$\vdots$$

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

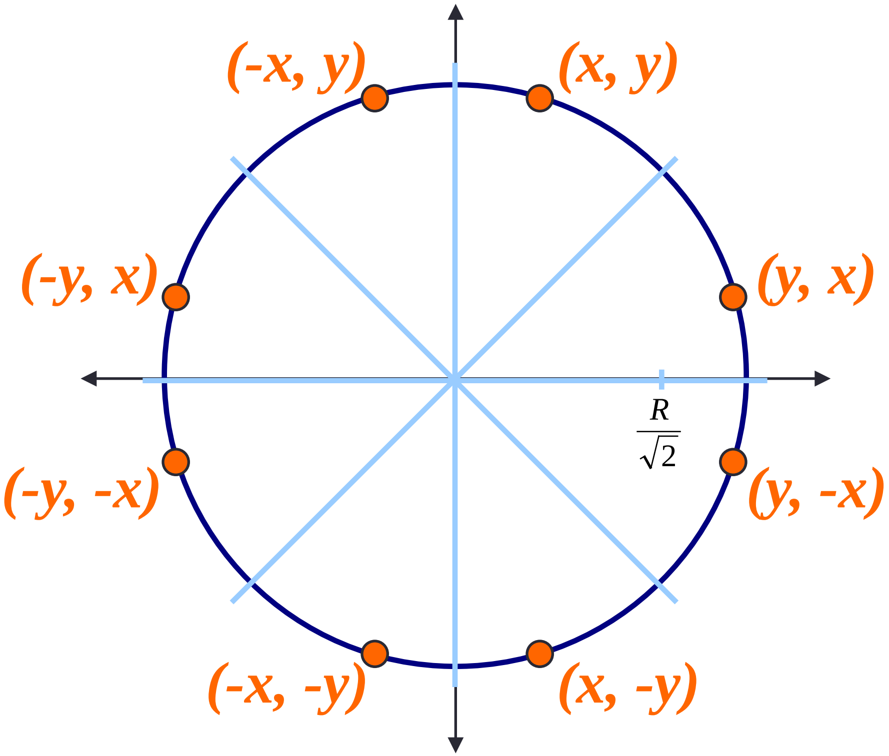
$$y_{20} = \sqrt{20^2 - 20^2} \approx 0$$

A Simple Circle Drawing Algorithm (cont…)

- 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
- 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*



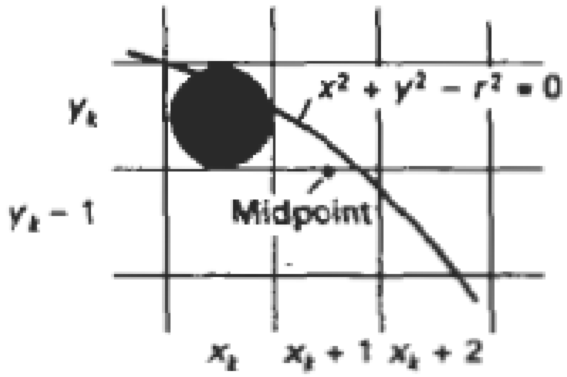
Mid-Point Circle Algorithm

- 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.
- Calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points

- To determine the closest pixel position to the specified circle path at each step.
- For given radius r and screen center position (x_c, y_c), calculate pixel positions around a circle path centered at the co-odinate origin (0,0).
- Then, move each calculated position (x, y) to its proper screen position by adding **x_c to x** and **y_c to y**.

Mid-Point 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.



Mid-Point Circle Algorithm

- The equation of the circle is re-designed as:

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

- The equation evaluates as follows:

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

Mid-Point 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:

$$
\begin{aligned}
p_k &= f_{circ}(x_k+1, y_k - 1/2) \\
&= (x_k+1)^2 + (y_k - 1/2)^2 - r^2
\end{aligned}
$$

- 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

- To ensure things are as efficient as possible we can do all of our calculations incrementally
- First consider:

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

- 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

Mid-Point Circle Algorithm

- The first decision variable is given as:

$$\begin{aligned} p_0 &= f_{circ}(1, r - 1/2) \\ &= 1 + (r - 1/2)^2 - r^2 \\ &= 5/4 - r \end{aligned}$$

- 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:

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

Algorithm

Midpoint Circle Algorithm

1. Input radius r and circle center (x_c, y_c), and obtain the first point on the circumference of a circle centered on the origin as

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

2. Calculate the initial value of the decision parameter as

$$p_0 = \frac{5}{4} - r$$

3. At each x_k position, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point along the circle centered on (0, 0) is (x_{k+1}, y_k) and

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

   Otherwise, 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 $2y_{k+1} = 2y_k - 2$.
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

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

6. Repeat steps 3 through 5 until $x \geq y$.

Mid-Point Circle Algorithm Example

- To see the mid-point circle algorithm in action lets use it to draw a circle centred at (0,0) with radius 10

Midpoint Circle Drawing Algorithm

For the initial point, $(x_0 , y_{0)} = (0, r)$

$$f_0 = f_{circle}(1, r-½\ )$$

$$= 1 + (r-½\ )^2 - r^2$$

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

$$\approx 1 - r$$

Midpoint Circle Drawing Algorithm

Example:

Given a circle radius = 10, determine the circle octant in the first quadrant from x=0 to x=y.

Solution:

$$f_0 = \frac{5}{4} - r$$

$$= \frac{5}{4} - 10$$

$$= -8.75$$

$$\approx -9$$

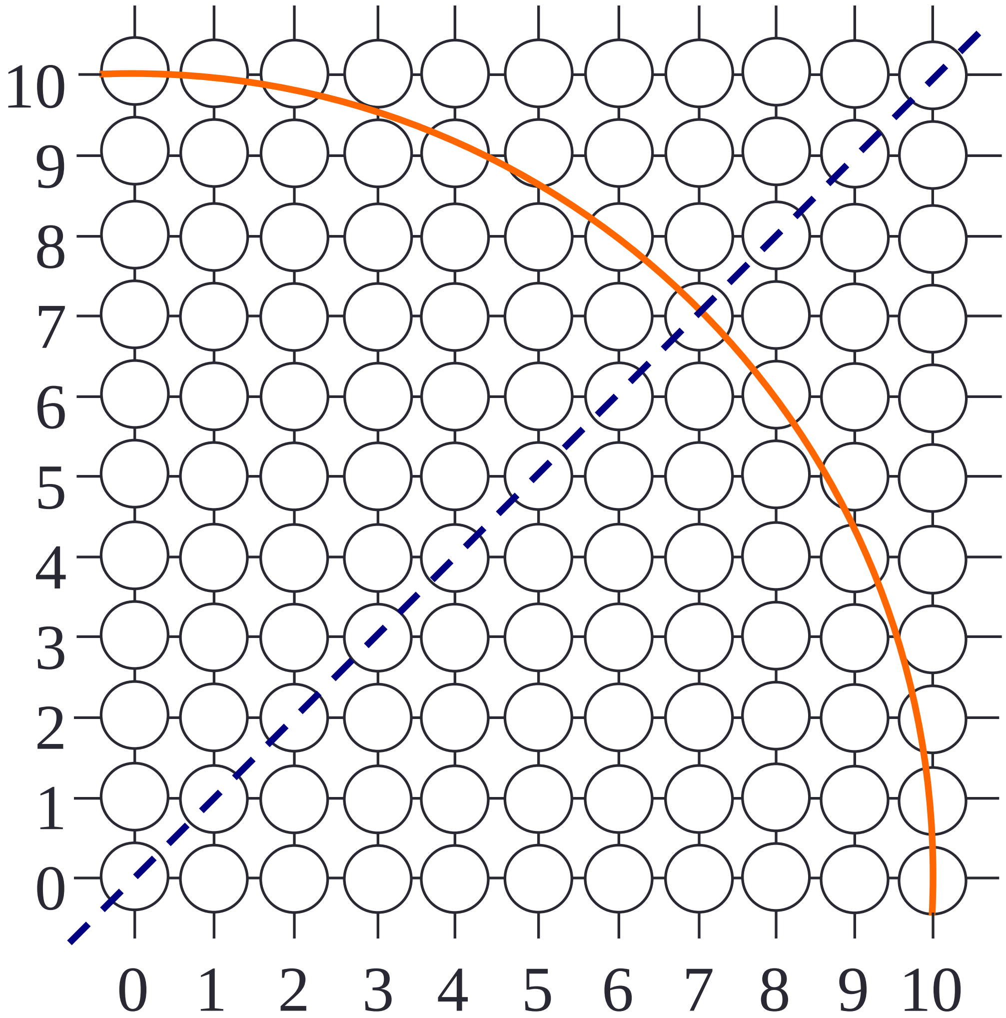
Midpoint Circle Drawing Algorithm

Initial $(x_0, y_0) = (1,10)$

Decision parameters are: $2x_0 = 2$, $2y_0 = 20$

k	F_k	x	y	$2x_{k+1}$	$2y_{k+1}$
0	-9	1	10	2	20
1	-9+2+1=-6	2	10	4	20
2	-6+4+1=-1	3	10	6	20
3	-1+6+1=6	4	9	8	18
4	6+8+1-18=-3	5	9	10	18
5	-3+10+1=8	6	8	12	16
6	8+12+1-16=5	7	7	14	14

Mid-Point Circle Algorithm Example (cont…)



k	p_k	(x_{k+1},y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0				
1				
2				
3				
4				
5				
6				

Mid-Point Circle Algorithm Summary

- The key insights in the mid-point circle algorithm are:
  - Eight-way symmetry can hugely reduce the work in drawing a circle
  - Moving in unit steps along the x axis at each point along the circle's edge we need to choose between two possible y coordinates

Thank You…