Higher Technological Institute
Computer Science Department

Computer Graphics

Dr Osama Farouk
Dr Ayman Soliman
Dr Adel Khaled

Lecture Three
Graphics Output Primitives

Circle drawing algorithms

Properties of Circles

A circle (Figure) is defined as the set of points that are all at a given distance r from a center position (x_c, y_c) .

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad (1)$$

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2} \quad (2)$$

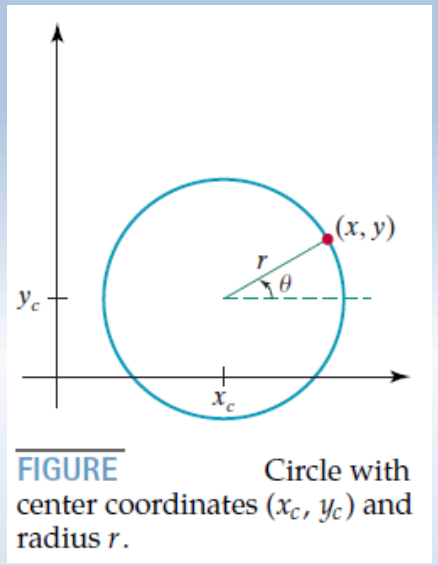


FIGURE Circle with center coordinates (x_c, y_c) and radius r.

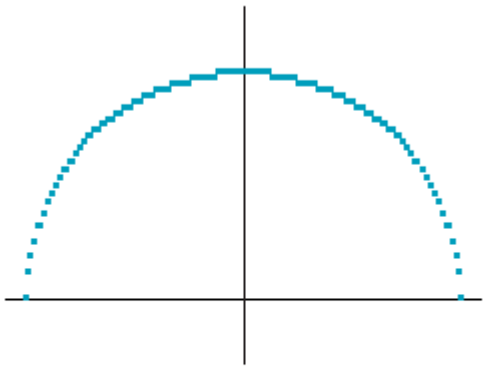


FIGURE Upper half of a circle plotted with Eq. 2 and with $(x_c, y_c) = (0, 0)$.

But this is not the best method for generating a circle.
One problem with this approach is that it involves considerable computation at each step. Moreover, the spacing between plotted pixel positions is not uniform, as demonstrated in Figure

Expressing the circle equation in parametric polar form yields the pair of equations

$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta \qquad (3)$$

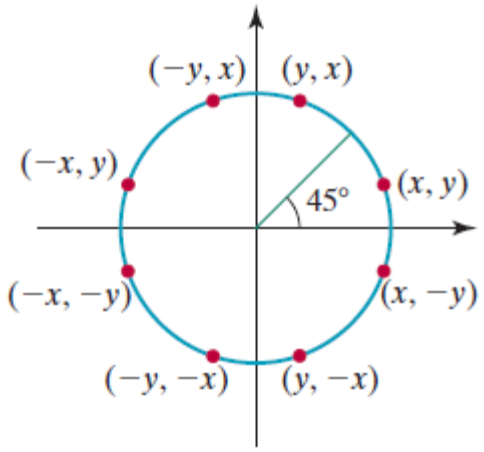


FIGURE Symmetry of a circle. Calculation of a circle point (x, y) in one octant yields the circle points shown for the other seven octants.

When a display is generated with these equations using a fixed angular step size, a circle is plotted with equally spaced points along the circumference

The shape of the circle is similar in each quadrant. Therefore, if we determine the curve positions in the first quadrant, we can generate the circle section in the second quadrant of the *xy* plane by noting that the two circle sections are symmetric with respect to the y axis.

And circle sections in the third and fourth quadrants can be obtained from sections in the first and second quadrants by considering symmetry about the *x* axis.

Midpoint Circle Algorithm

The basic idea in this approach is to test the halfway position between two pixels to determine if this midpoint is inside or outside the circle boundary.

To apply the midpoint method, we define a circle function as

$$f_{\text{circ}}(x, y) = x^2 + y^2 - r^2 \qquad (4)$$

$$f_{\text{circ}}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x, y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases} \qquad (5)$$

The tests in (5) are performed for the mid positions between pixels near the circle path at each sampling step.

Assuming that we have just plotted the pixel at (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. Our decision parameter is the circle function (4) evaluated at the midpoint between these two pixels:

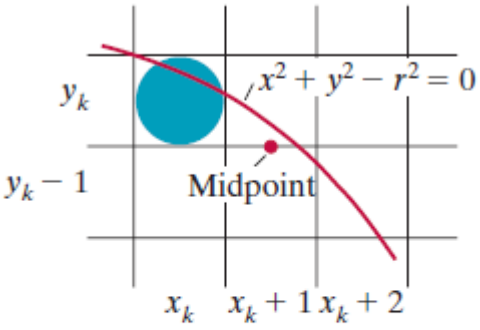


FIGURE Midpoint between candidate pixels at sampling position $x_k + 1$ along a circular path.

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

$$= (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \qquad (6)$$

If $p_k < 0$, this midpoint is inside the circle and the pixel on scan line y_k is closer to the circle boundary. Otherwise, the midposition is outside or on the circle boundary, and we select the pixel on scan line $y_k - 1$.

Successive decision parameters are obtained using incremental calculations.

We obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position $x_{k+1} + 1 = x_k + 2$

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

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

or

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \qquad (7)$$

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

Increments for obtaining p_{k+1} are either $2x_{k+1}+1$ (if p_k is negative) or $2x_{k+1}+1-2y_{k+1}$.
Evaluation of the terms $2x_{k+1}$ and $2y_{k+1}$ can also be done incrementally as

$$\begin{aligned} 2x_{k+1} &= 2x_k + 2 \\ 2y_{k+1} &= 2y_k - 2 \end{aligned} \qquad (8)$$

The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$ *from eq.(6)*:

$$\begin{aligned} p_0 &= f_{\text{circ}}\left(1, r - \frac{1}{2}\right) \\ &= 1 + \left(r - \frac{1}{2}\right)^2 - r^2 \end{aligned}$$

$$p_0 = \frac{5}{4} - r \qquad (9)$$

If the radius r is specified as an integer, we can simply round p_0 to

$$p_0 = 1 - r \qquad \text{(for } r \text{ an integer)}$$

Midpoint Circle Algorithm

Midpoint Circle Algorithm

1. Input radius r and circle center (x_c, y_c), then set the coordinates for 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 at (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 $x \geq y$.

Example: Midpoint Circle Drawing

Given a circle radius r = 10, we demonstrate the midpoint circle algorithm by determining positions along the circle octant in the first quadrant from x = 0 to x = y. The initial value of the decision parameter is

$$p_0 = 1 - r = -9$$

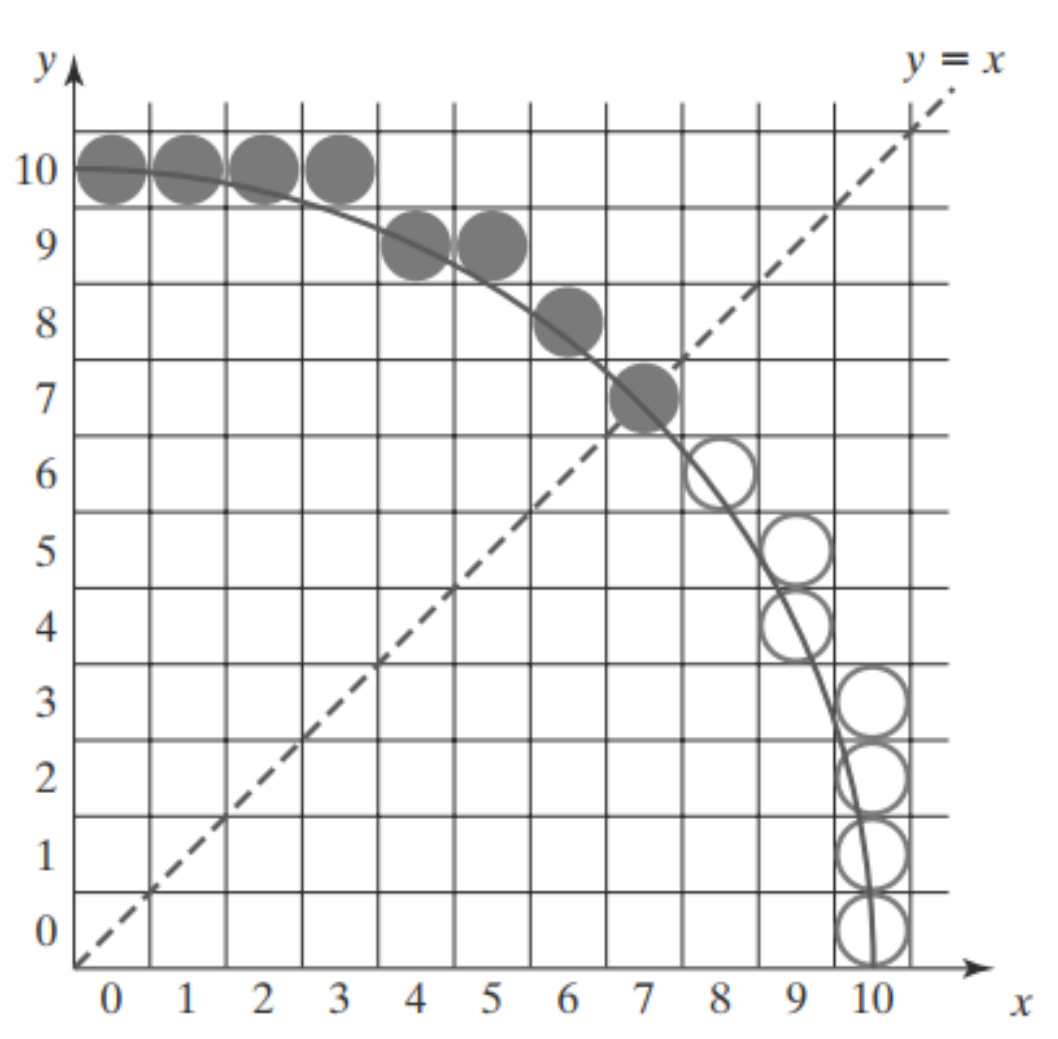
For the circle centered on the coordinate origin, the initial point is $(x_0, y_0) = (0, 10)$, and initial increment terms for calculating the decision parameters are

$$2x_0 = 0, \qquad 2y_0 = 20$$

Successive midpoint decision parameter values and the corresponding coordinate positions along the circle path are listed in the following table:

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

A plot of the generated pixel positions in the first quadrant is shown



```
#include <GL/glut.h>

class screenPt
{
    private:
       GLint x, y;

    public:
       /*  Default Constructor: initializes coordinate position to (0, 0).  */
       screenPt ( )  {
           x = y = 0;
       }
       void setCoords (GLint xCoordValue, GLint yCoordValue)  {
           x = xCoordValue;
           y = yCoordValue;
       }

       GLint getx ( ) const  {
           return x;
       }

       GLint gety ( ) const  {
           return y;
       }
       void incrementx ( )  {
           x++;
       }
       void decrementy ( )  {
           y--;
       }
};
```

```
void setPixel (GLint xCoord, GLint yCoord)
{
    glBegin (GL_POINTS);
        glVertex2i (xCoord, yCoord);
    glEnd ( );
}

void circleMidpoint (GLint xc, GLint yc, GLint radius)
{
    screenPt circPt;

    GLint p = 1 - radius;          //  Initial value for midpoint parameter.

    circPt.setCoords (0, radius); //  Set coordinates for top point of circle.

    void circlePlotPoints (GLint, GLint, screenPt);
    /*  Plot the initial point in each circle quadrant.  */
    circlePlotPoints (xc, yc, circPt);
    /*  Calculate next point and plot in each octant.  */
```

```
    while (circPt.getx ( ) < circPt.gety ( )) {
        circPt.incrementx ( );
        if (p < 0)
            p += 2 * circPt.getx ( ) + 1;
        else {
            circPt.decrementy ( );
            p += 2 * (circPt.getx ( ) - circPt.gety ( )) + 1;
        }
        circlePlotPoints (xc, yc, circPt);
    }
}

void circlePlotPoints (GLint xc, GLint yc, screenPt circPt)
{
    setPixel (xc + circPt.getx ( ), yc + circPt.gety ( ));
    setPixel (xc - circPt.getx ( ), yc + circPt.gety ( ));
    setPixel (xc + circPt.getx ( ), yc - circPt.gety ( ));
    setPixel (xc - circPt.getx ( ), yc - circPt.gety ( ));
    setPixel (xc + circPt.gety ( ), yc + circPt.getx ( ));
    setPixel (xc - circPt.gety ( ), yc + circPt.getx ( ));
    setPixel (xc + circPt.gety ( ), yc - circPt.getx ( ));
    setPixel (xc - circPt.gety ( ), yc - circPt.getx ( ));
}
```

QUIZ

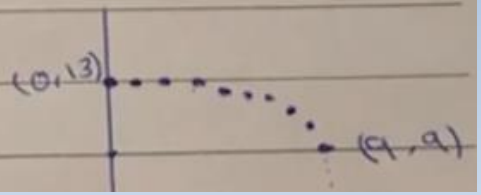
Question: given a circle centred at origin and radius = 13, what positions should be generated using mid point algorithm ?

X	Y	P
0	13	-11.75
1	13	-8.75
2	13	-3.75
3	13	3.25
4	12	-11.75
5	12	-0.75
6	12	12.25
7	11	5.25
8	10	2.25
9	9	

$r = 13$

$y = r$

$x = 0$



End of Lecture
Good Luck!

See you

in next lecture...