Maharaja Agrasen Institute of Technology
ETCS 211

Computer Graphics & Multimedia
UNIT 1

BRESENHAM'S AND MID-POINT CIRCLE DRAWING ALGORITHMS

Scan Conversion

It is the responsibility of the graphics system or the application program to convert each primitive from its geometric definition into a set of pixels that make up the primitive in the image space.

This conversion task is generally referred to as scan conversion or rasterisation.

Scan Converting a Circle

A circle is a symmetric figure. Any circle generating algorithm can take advantage of the circle’s symmetry to plot eight points for each value that the algorithm calculates.

Eight way symmetry is used by reflecting each calculated point around each 45 º axis.

A Simple Circle Drawing Algorithm

The equation for a circle is:

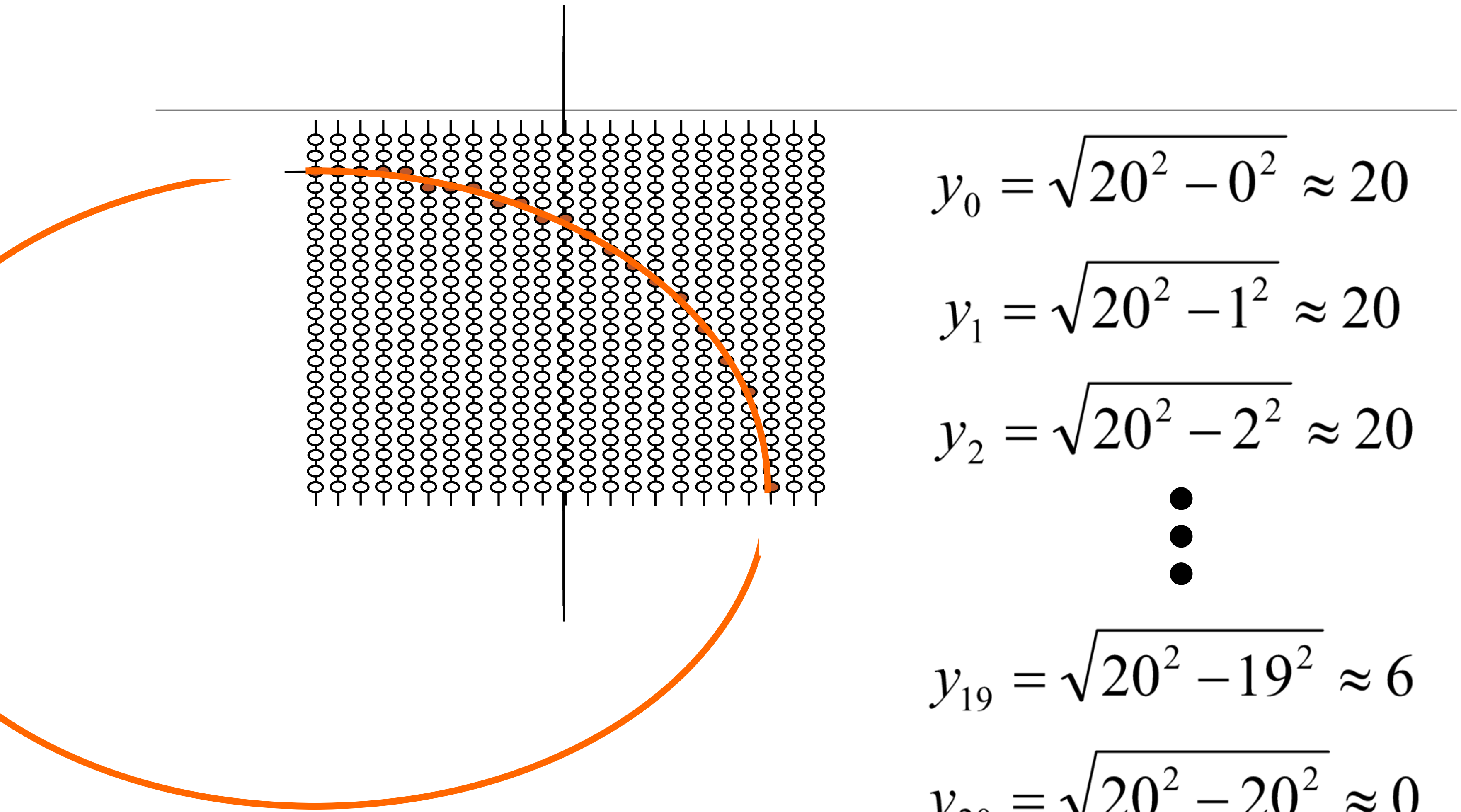
$$x^2 + y^2 = r^2$$

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

Polar coordinates

$X=r*\cos\theta+x_c$

$Y=r*\sin\theta+y_c$

$0º\leq\theta\leq360º$

Or

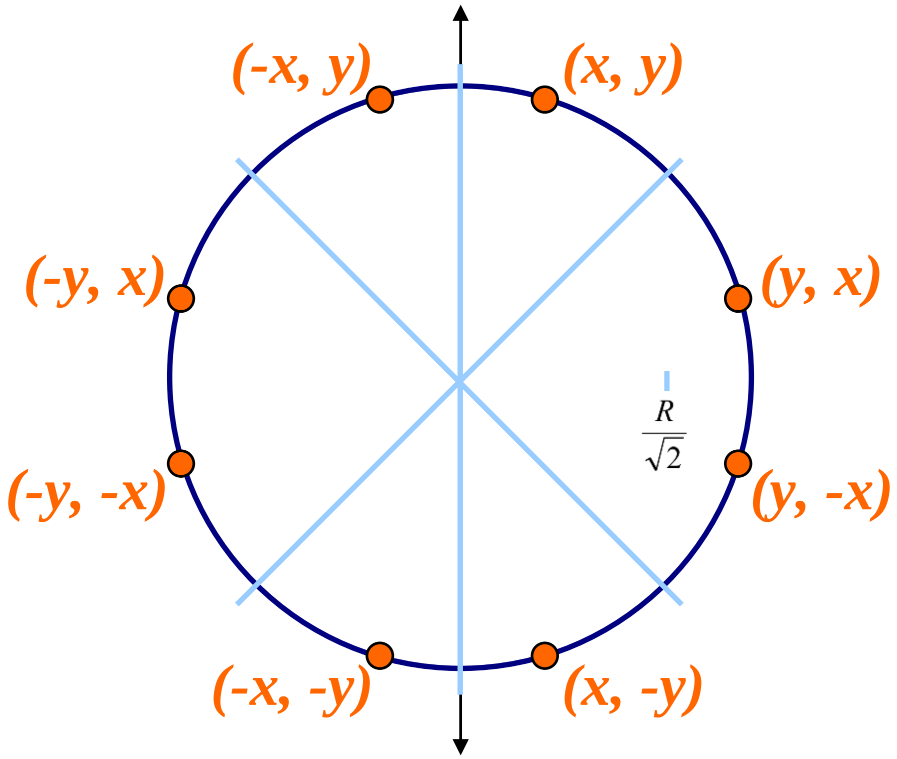
$0 \leq \theta \leq 6.28(2*\pi)$

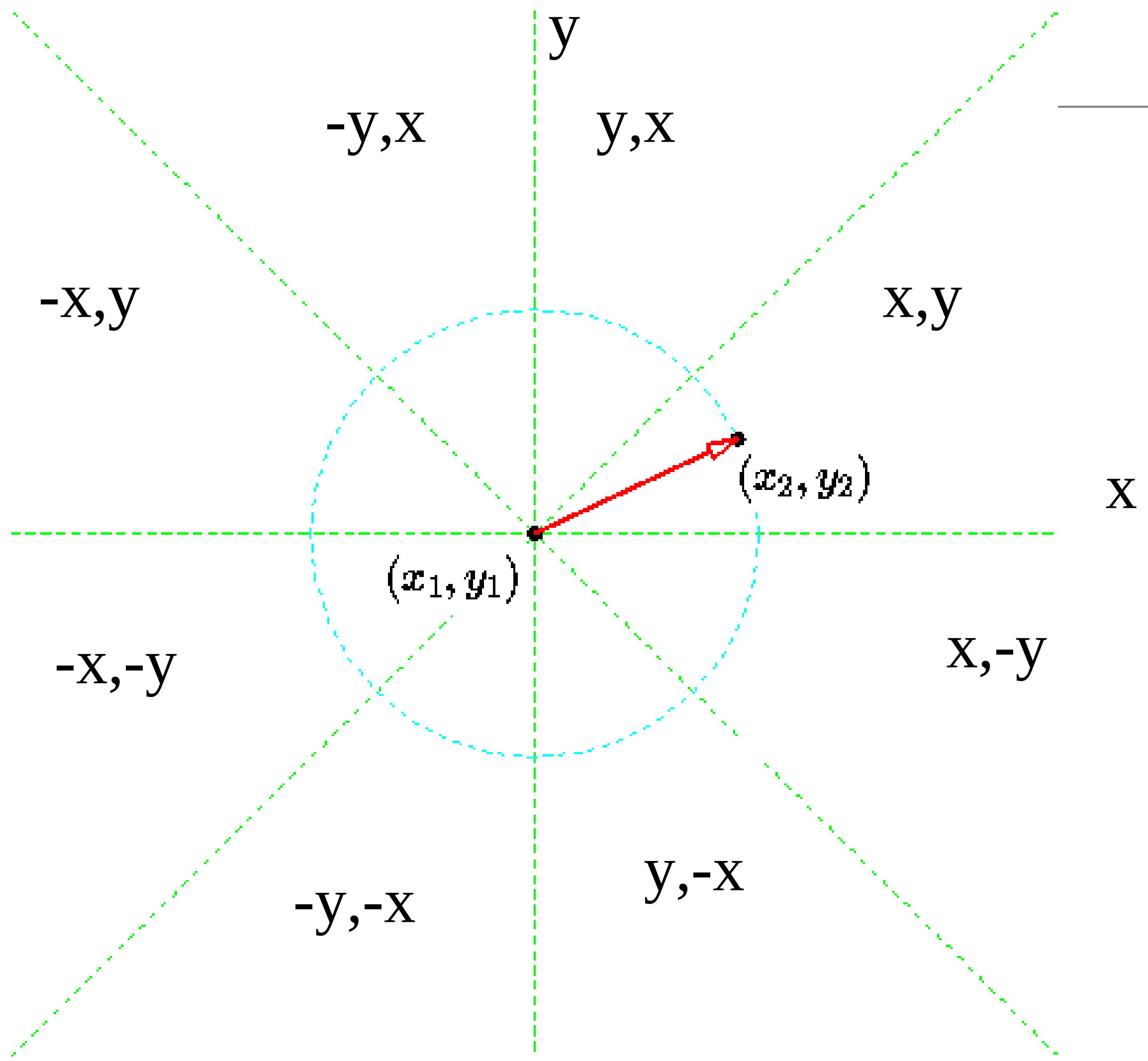
Problem:

- Deciding the increment in θ
- Cos, sin calculations

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*




y
-y,x
y,x
-x,y
x,y
(x_2, y_2)
x
(x_1, y_1)
-x,-y
x,-y
-y,-x
y,-x

Defining a Circle using polynomial method:-

This method defines a circle with second order polynomial equation

$$y^2 = r^2 - x^2$$

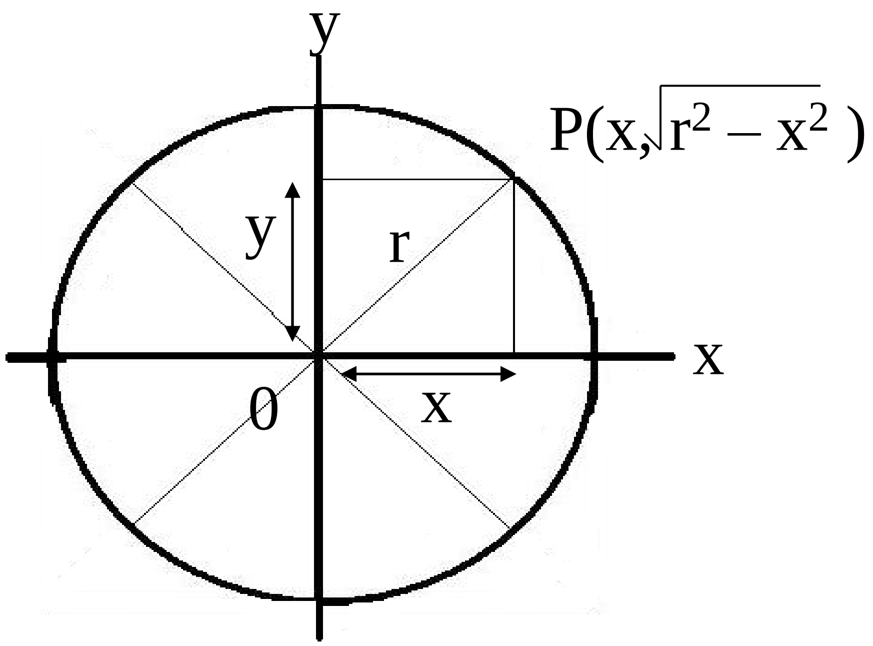
where **x, y** = x & y coordinates, **r** = radius

With this method, each x coordinate in the sector, from 90º to 45º, is found by stepping x from 0 to $r/\sqrt{2}$,

Each y coordinate is found by evaluating $\sqrt{r^2 - x^2}$ for each step of x.

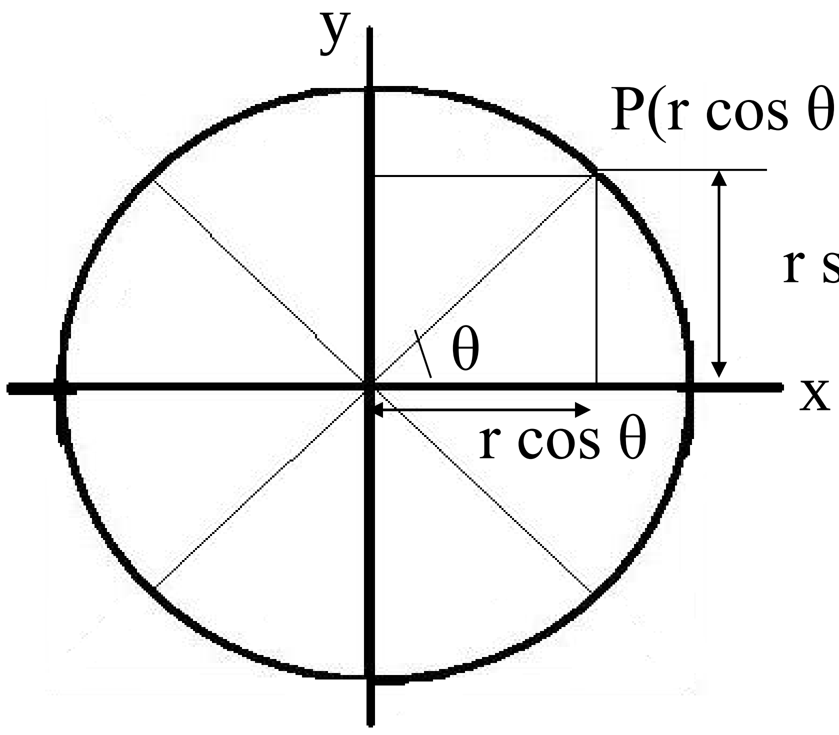
This is a very inefficient method, however, because for each point both x & r must be squared and subtracted from each other.

Then the square root of the result must be found.



Defining a Circle using trigonometric method:-

This second method of defining a circle uses functions : **x = r cos θ, y = r sin θ**



θ = current angle, r = circle radius

By this method θ is stepped from θ to π /4

& each value of x & y is calculated.

Computation of values of sin θ & cos θ are

very time consuming as compared to the first method.

Bresenham’s Circle Algorithm

Working:

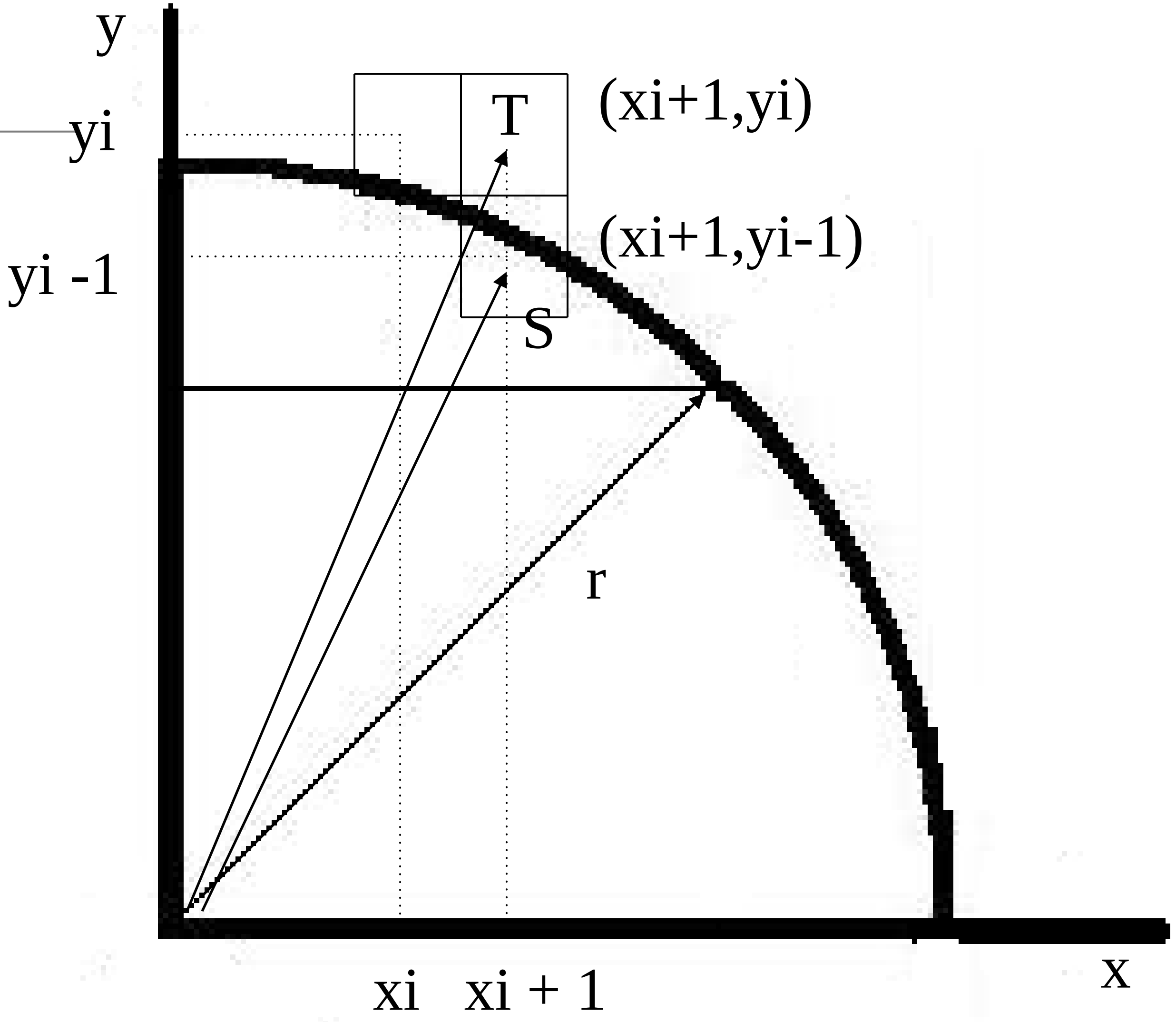
If the eight-way symmetry of a circle is used to generate a circle, points will have to be generated through 45º angle.

If points are generated from 90º and 45º ,each new point closest to the true circle can be found by taking either of two actions:

(1) Move in the x direction one unit or

(2) Move in the x direction 1 unit & -ve y direction 1 unit.

Bresenham's Circle Algorithm



Bresenham's Circle Algorithm

Assume that (xi, yi) are the coordinates of the last scan-converted pixel upon entering step i.

Let the distance from the origin to pixel T squared minus the distance to the true circle squared = D(T) = $t^2 - r^2$.

And let the distance from the origin to pixel S squared minus the distance to the true circle squared = D(S) = $s^2 - r^2$.

Coordinates of T are (xi +1, yi)

S are (xi +1, yi – 1)

Bresenham's Circle Algorithm

Following expressions can be developed:

1. $D(T) = (x_i + 1)^2 + y_i^2 - r^2$
2. $D(S) = (x_i + 1)^2 + (y_i - 1)^2 - r^2$

This function D provides a relative

measurement of the distance from the center

of a pixel to the true circle.

Since D(T) is always +ve (T is outside the circle) & D(S) will always be –ve (S is inside the true circle),

a decision variable di can be defined as:

$$d_i = D(T) + D(S)$$

Bresenham’s Circle Algorithm

Therefore,

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

When $d_i<0$, we have $|D(T)|<|D(S)|$ and pixel T is chosen.

When $d_i>=0$, we have $|D(T)|>|D(S)|$ and pixel S is selected. Decision variable for next step:

$d_{i+1} = 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2$

Bresenham's Circle Algorithm

Hence,

$d_{i+1} – di = 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} – 1)^2 -$

$2(xi + 1)^2 - yi^2 - (yi – 1)^2$

Since $x_{i+1} = xi + 1$, we have

$d_{i+1} = di + 4xi + 2(y_{i+1}^2 - yi^2) – 2(y_{i+1} – yi) + 6$

- If T is chosen pixel (meaning that **$di < 0$**) then $y_{i+1} = yi$ and so **$d_{i+1} = di + 4xi + 6$**
- If S is chosen (meaning that **$di > 0$**) then $y_{i+1} = yi –1$ and so **$d_{i+1} = di + 4(xi-yi) + 10$**

Bresenham’s Circle Algorithm

Hence we have

d_{i+1} = $d_i + 4x_i + 6$ if $d_i < 0$

$d_i + 4(x_i-y_i) + 10$ if $d_i > 0$

Finally, we set(0,r) to the starting pixel coordinates and compute d1 from the original definition of di:

$x = 0, y = r$

$$d_1 = 2(0 + 1)^2 + r^2 + (r-1)^2 - 2r^2$$

$$= 3 - 2r$$

Algorithm:

For generating all the pixel coordinates in the 90º to 45º octant that are needed when scan-converting a circle of radius r.

```
(i)    int x = 0, y = r, d = 3-2r
(ii)   while (x <= y)
       {
       setpixel(x,y)
        if (d < 0)
        d = d + 4x + 6
```

```
   else
  {
   d = d + 4(x - y) + 10
   y--
   }
  x++
}
```

Midpoint Circle Algorithm

Circle equation : $f(x,y) = x^2 + y^2 - r^2$

Any point on the circumference of the circle will satisfy the above equation.

Hence :

>0 ⍰ Point is outside the boundary of the circle

$f(x,y)$ $=0$ ⍰ Point is on the boundary of the circle

$<=$ ⍰ Point is inside the boundary of the circle

Midpoint Circle Algorithm

Now consider the coordinates of the point halfway between pixel T & S $(x_i + 1, y_i - \frac{1}{2})$ This is called the midpoint and we use it to define a decision parameter:

$$p_i = f(x_i + 1, y_i - \tfrac{1}{2}) = (x_i + 1)2 + (y_i - \tfrac{1}{2})2 - r2$$

If p_i is –ve , the midpoint is inside the circle, then we choose pixel T.

If p_i is +ve, the midpoint is outside the circle, & we choose S.

Similarly,

$$p_{i+1} = (x_{i+1} + 1)2 + (y_{i+1} - \tfrac{1}{2})2 - r2$$

Midpoint Circle Algorithm

pi+1 = pi + 2(xi+ 1) + 1 + (y i+1 2 – yi2) – (y i+1 – yi)

If pixel T is chosen (meaning pi < 0), we

have yi+1 = yi.

If pixel S is chosen (meaning pi > 0), we have yi+1 = yi – 1. Thus,

In terms of (xi, yi), we have

pi+1= pi + 2xi + 3 if pi<0

pi + 2(xi - yi) + 5 if pi>=0

Midpoint Circle Algorithm

Finally, we compute the initial value for the decision parameter using the original definition of pi and (0,r):

$pi = (0 + 1)2 + (r - ½)2 - r2 = 5/4 - r$

One can see that this is not really integer computation. However, when r is an integer we can simply set **$p1 = 1 - r$**

The Mid-Point 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:

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

Starting with $k = 0$ at each position x_k, perform the following test. If $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and:

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

The Mid-Point Circle Algorithm (cont…)

Otherwise the next point along the circle is (x_k+1, y_k-1) and:

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

Determine symmetry points in the other seven octants

Move each calculated pixel position (x, y) onto the circular path centred at (x_c, y_c) to plot the coordinate values:

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

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

Algorithm

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

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

2. Calculate the initial value of the decision parameter as

$$P_0 = 1 - r$$

where r is the radius of the circle

Midpoint Circle Algorithm

3. At each x_k position, starting at k= 0, perform the following test:

If **($P_k < 0$)**, then the **NEXT POINT** along the circle centred on (0, 0) is

(x_{k+1}, y_k) and $P_{k+1} = P_k + 2x_{k+1} + 1$

$\Rightarrow \quad P_{k+1} = P_k + 2x_k + 3$

Otherwise, the **NEXT POINT** along the circle is

(x_{k+1}, y_{k-1}) and $P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$

$\Rightarrow \quad P_{k+1} = P_k + 2x_k - 2y_k + 5$

Midpoint Circle Algorithm

4. Determine the Symmetry Points on the other 7 Octants

5. Move each calculated pixel position (x,y) onto the circular path centered at (x_c,y_c) and plot the coordinate values as

$$\mathbf{x = x + x_c \text{ and } y = y + y_c}$$

6. Repeat steps 3 to 5 until x >= y

Arbitrarily Centered Circles

To scan-convert a circle centered at (x1,y1), we can simply replace the setPixel(x,y) statement in the algorithm description with setPixel(x+x1,y+y1). This is because a circle centered at(x1,y1) can be viewed as a circle centered at origin that is moved by x1 and y1 in the x and the y direction.

Q & A

1.The midpoint circle drawing algorithm also uses the __of the circle to generate?

a) two-way symmetry
b) four-way symmetry
c) eight-way symmetry
d) both a & b

2. A circle, if scaled only in one direction becomes a ?

a) parabola
b) hyperbola
c) ellipse
d) remains a circle

3.Let R be the radius of a circle. The angle subtended by an arc of length R at the center of the circle is ?

a) 1 degree
b) 1 radian
c) 45 degree
d) impossible to determine