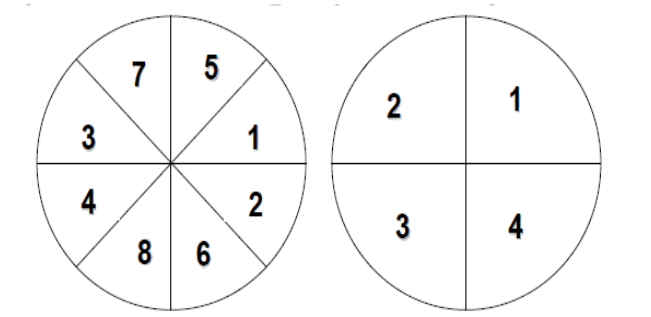
Chapter Five

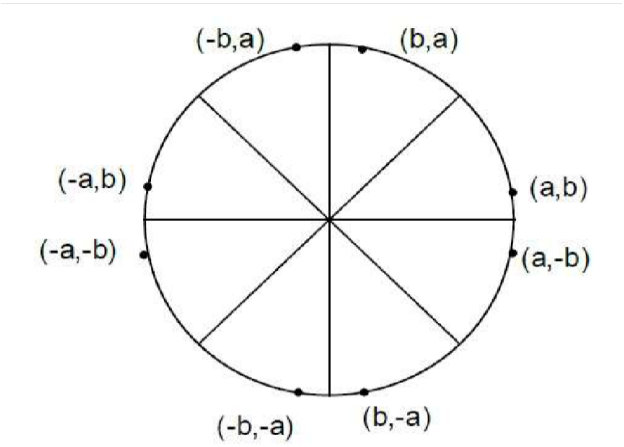
5.1 Symmetric (incremental) algorithm

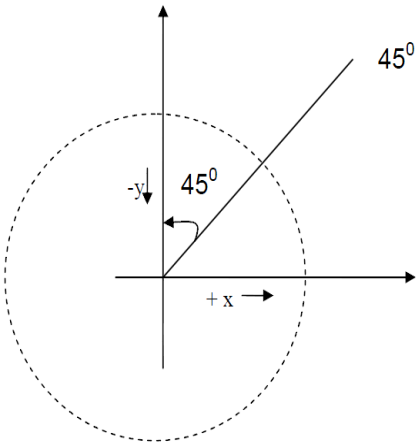
• Computation can be reduced by considering the symmetry of circle.
• The shape of the circle is similar in each quadrant
• We can generate the circle section in the second quadrant by noting that the two circle sections are symmetric with respect to the y axis And circle sections in the third and fourth quadrants can be obtained from sections in the first and second quadrants by considering symmetry about the x axis.



Symmetry in circle

Circle sections in adjacent octants within one quadrant are symmetric with respect to the 45' line dividing the two octants.





Consider a circle center at the origin(0,0),if the point (x,y) is on the circle then we can trivially compute seven other points on the circle.

- This method proposed the center of circle at origin point(0,0),so the first pixel in the circle is (r,0).
- The other pixels are computed depend on polar equation as follow:-

$x = xc + r \cos \theta$

$y = yc + r \sin \theta$

$x = r \cos \theta$

$y = r \sin \theta$

$Dx = -r \sin\theta \, d\theta$

$Dy = r \cos \theta \, d\theta$

$Dx = -y \, d\theta$

$Dy = x \, d\theta$

as we know

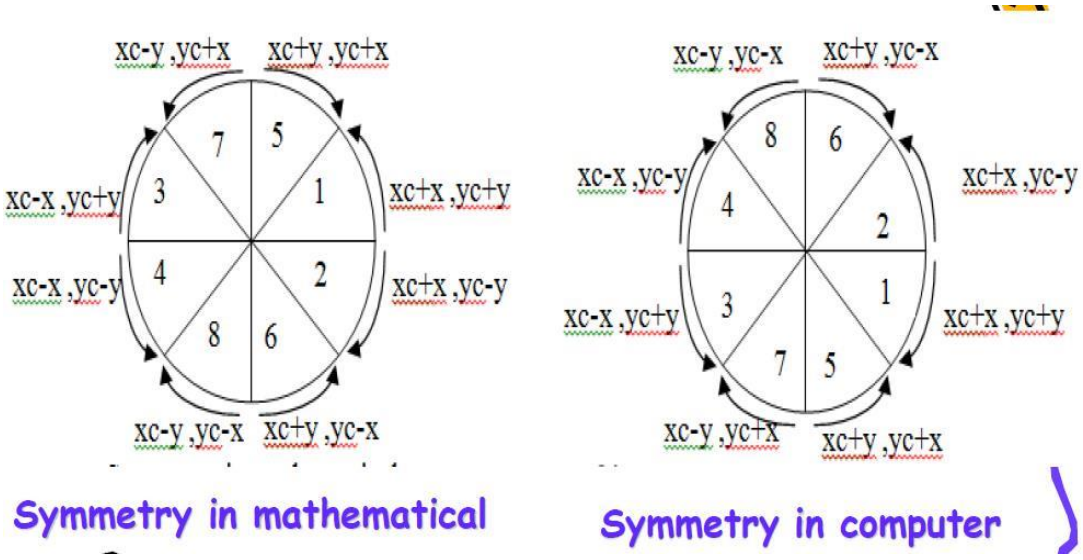
$x = x + Dx$

$y = y + Dy$ ⟶

$$x = x - y^* \, d\emptyset$$
$$y = y + x^* \, d\emptyset$$

symmetry equation

if we add center (xc,yc) we obtain the values of 8 pixels of the circle as figure bellow.



Incremental (Symmetric) algorithm

```
Input : xc,yc,r.
Output : circle
{
  th=0 , pi=3.141593 , dth=1/r , x=r , y=0;
  while th<=pi/4
    {
      plot (integer (xc+x) , integer (yc+y) ,color )
      plot (integer (xc+x) , integer (yc-y) ,color)
      plot (integer (xc-x) , integer (yc+y) ,color )
      plot (integer (xc-x) , integer (yc-y) ,color)
      plot (integer (xc+y) , integer (yc+x) ,color )
      plot (integer (xc+y) , integer (yc-x) ,color )
      plot (integer (xc-y) , integer (yc+x) ,color )
      plot (integer (xc-y) , integer (yc-x) ,color )
      th = th + dth ;
      x = x - y * dth ;
      y = y + x * dth ;
    }
}
```

5.2 Bresenham circle algorithm:

The values of a circle centered at the origin are computed in a 45 sector from x=0 to x=y the remaining seven sectors are obtained from the eight point symmetry of the circle.
They values for this sector decrease as the x values increase if (0, r) is the starting point of the algorithm, then as x increase by one unit the y value either remains the same or is decrease by one unit. If(x, y) is a pixel on the circle, the next pixel is either A or B.

A :(x+1, y); to the right of previous point

B: (x+1, y-1); down and to the right of the previous point.

The algorithm proceeds to choose Pixel A or B by finding and comparing the distance from each pixel to the point on the circle that has x value of (x+1) these distances measure how far from the circle each pixel is the pixel with the smallest distances the best approximation on the circle. The square of the distance of pixel A from the center of the circle is:

(x+1) ^2 + y^2.

The difference between this squared distance and the squared distance to the closest point on the circle is: ***d(A)= (x+1)^2 +y^2 -r^2***
for pixel B the distance is : ***d(B)= (x+1)^2 + (y-1)^2 – r^2.***

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

A point lies on the circle if its d value equals 0, in order to determine which pixel A or B has the smallest d value we examine the sum

S=d(A)+d(B);

- There are three cases to be consider:

1. If the circle goes between A and B then

d(A) > 0 and d(B) <0

If abs (d(A)) > abs (d(B)) → choose B , such that S> 0

If abs (d(A)) <= abs(d(B)) → choose A , such that S<= 0

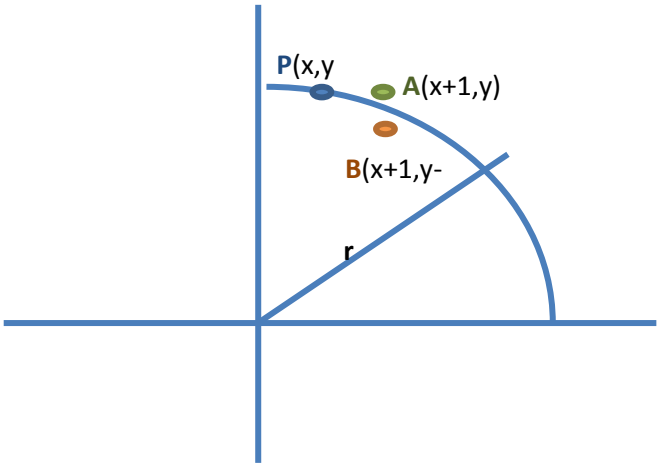
2. If the circle goes through or above A then

d(A)<=0 and d(B)<0 → choose A, such that S<0

3. If the circle goes through B or below then

d(A) > 0 and d(B) >=0 → choose B , such that S>0

Rule: if S>0 we choose pixel B otherwise we choose pixel A.



Bresenham circle algorithm:

```
Input : xc,yc,r.
Output : circle

{
  x=0; y=r;
 While(x<=y)
 {
       plot (integer (xc+x) , integer (yc+y) ,color)
      plot (integer (xc+x) , integer (yc-y) ,color)
      plot (integer (xc-x) , integer (yc+y) ,color )
      plot (integer (xc-x) , integer (yc-y) ,color)
      plot (integer (xc+y) , integer (yc+x) ,color )
      plot (integer (xc+y) , integer (yc-x) ,color )
      plot (integer (xc-y) , integer (yc+x) ,color )
      plot (integer (xc-y) , integer (yc-x) ,color )

      da= (x+1)^2+(y)^2-(r)^2
      db= (x+1)^2+ (y-1)^2 -(r)^2
      s= da + db
      if (s>0)
        { y=y-1  ; }

      x=x+1;
  }
}
```

Ex. /trace the Bresenham algorithm to generate six points of the circle centered at (300,150) with a radius equal to 9 unit.

Sol. / da = (x+1) ^2 + y^2 – R^2 , db= (x+1) ^2 + (y-1) ^2 – R^2,

R^2=81

x	y	x+xc	y+yc	d(A)	d(B)	s
0	9	300	159	1	-16	-15
1	9	301	159	4	-13	-9
2	9	302	159	9	-8	1
3	8	303	158	-1	-16	-17
4	8	304	158	8	-7	1
5	7	305	157	4	-9	-5

5.3 Midpoint circle algorithm:

• Bresenham's line algorithm for raster displays is adapted to circle generation by setting up decision parameters (P) for finding the closest pixel to the circumference at each step.

• For a given radius r and screen center position (xc,yc), we can first set up our algorithm to calculate pixel positions around a circle path centered at the coordinate origin (0,0). So first pixel is (0,r)

• each calculated position (x,y) is moved to its proper screen position by adding xc to x and yc to y.

• we compute the first octant pixels from x=0 to x=y.

• Positions for the other seven octants are then obtained by symmetry

•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 .

• we can take unit steps in the positive x direction over octant and use a decision parameter to determine which of the two possible y positions is closer to the circle path at each step.

$$r^2 = x^2 + y^2$$
$$P = (x+1)^2 + (y-1)^2 - r^2$$
$$p = 2(1-r)$$

• Any point (x , y) on the boundary of the circle with radius r satisfies the equation p = 0.

• If the point is in the interior of the circle, P is negative value.

• if the point is outside the circle, P is positive.

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

• We need to determine whether the pixel at position (x+ 1, y) or the one at position (x+1,y-1) is closer to the circle. So If p<0,the point is inside the circle and the pixel (x+1,y) is closer to the circle boundary. Otherwise, the point is outside or on the circle boundary and we select the pixel (x+1,y-1).

$$P\begin{cases} < 0 & (x+1, y) \\ & p = p + 2x + 1 \\ \geq 0 & (x+1, y-1) \\ & p = p + 2(x-y) + 1 \end{cases}$$

midpoint circle algorithm

Input : xc,yc,r.
Output : circle

```
{
x = 0 ; y = r ; p = 2*(1 – r)
While x <= y
    {
      plot (integer (xc+x) , integer (yc+y) ,color)
      plot (integer (xc+x) , integer (yc-y) ,color)
      plot (integer (xc-x) , integer (yc+y) ,color )
      plot (integer (xc-x) , integer (yc-y) ,color)
      plot (integer (xc+y) , integer (yc+x) ,color )
      plot (integer (xc+y) , integer (yc-x) ,color )
      plot (integer (xc-y) , integer (yc+x) ,color )
      plot (integer (xc-y) , integer (yc-x) ,color )
      x = x + 1;
      If   p < 0 Then
             p = p + 2 * x + 1;
      Else
             y = y - 1 ;
             p = p + 2 * (x - y) + 1;
      End If
    }
}
```

Ex. /trace the **Midpoint algorithm** to generate six points of the circle centered at (120,130) with a radius equal to 7 unit.
Sol. /p=2(1-r)=2(1-7)=-12

If p<0 p=p+2*x+1

If p>=0 p=p+2(x-y)+1

x	y	P	x+xc	y+yc
0	7	-12	120	137
1	7	-9	121	137
2	7	-4	122	137
3	7	3	123	137
4	6	0	124	136
5	5	1	125	135

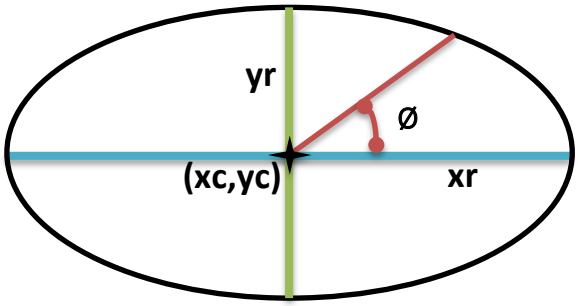
5.4 Ellipses Drawing

- It is a variation of a circle.
- Stretching a circle in one direction produce an ellipse.
- Following algorithm for ellipse drawing which stretched in the x or y direction.

The polar equation for this type of ellipse centered at (x_c , y_c) are:

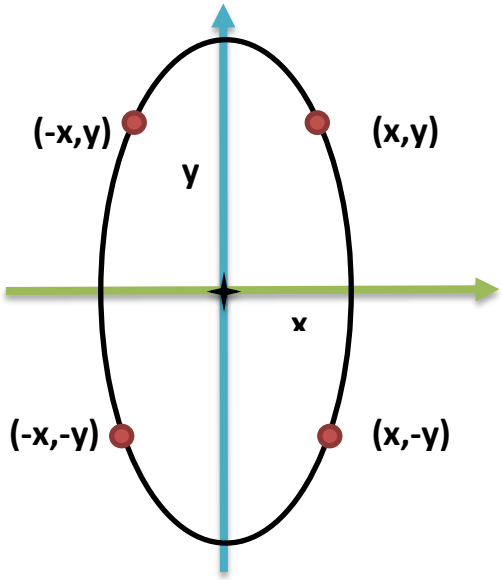
$$\left.\begin{aligned} x &= x_c + xr*\cos(\theta) \\ y &= y_c + yr*\sin(\theta) \end{aligned}\right\} \quad \ldots\ldots\ldots (1)$$

θ values between 0 and 2π radius



The values of (xr) and (yr) effect the shape of the ellipse.

- If yr > xr the ellipse is longer in the y direction
- If yr < xr the ellipse is longer in the x direction.
- If xr = yr the equation produces a circle.
- The ellipse can be drawn using four-points symmetry.
  If (x,y) lies on the ellipse so do the points (-x,y), (x,-y) and (-x,-y).



A)The polar representation of an ellipse:

The polar equation for an ellipse centered at (xc, yc) and xr is the radius on the x-axis and yr is the radius on the y-axis.
Then algorithm is:

```
Input : xc,yc,xr,yr.
Output: Ellipse .
 {
      dth=1/ ((xr +yr)/2) ;
       th=0 ;
       pi= 3.1416;
       While (th<=2*pi)
          {
           x= xr*cos(th) ;
           y= yr*sin(th) ;
           plot (integer (xc+x), integer(yc +y),color);
           plot (integer (xc-x), integer (yc+y), color);
           plot (integer (xc+x), integer (yc-y), color);
           plot (integer (xc-x), integer (yc-y), color) ;
           th=th+dth ;
          }
 }
```

EX /trace the algorithm that use the polar representation to generate six points of the ellipse centered at (300, 150) with xr=10, yr=5.

Sol:\ dth=1/((xr+yr)/2) ; 2*pi=6.2832
=1/((10+5)/2)
= 0.133

x=xr*cos(th)	Plot(x)	y=yr*sin(th)	Plot(y)	x+xc	y+yc	th
10	10	0	0	310	150	0
9.9	10	0.6	1	310	151	0.1333
9.6	10	1.3	1	310	151	0.2667
9.2	9	1.9	2	309	152	0.4000
8.6	9	2.5	3	309	153	0.5333
7.8	8	3.0	3	308	153	0.6667

B)Incremental method to drawing of ellipse: similar of discuss in circle, but differ for equations :

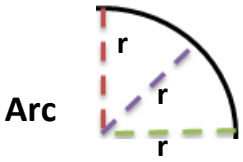
- The following incremental equations for an ellipse are derived from equations (1)

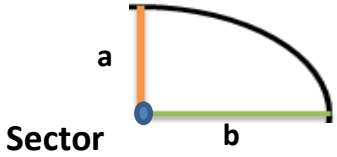
$$Xn+1 = Xn \cos(\Delta\theta) - (Xr/Yr) * Yn \sin(\Delta\theta) \ldots\ldots (2)$$
$$Yn+1 = Yn \cos(\Delta\theta) + (Yr/Xr) * Xn \sin(\Delta\theta) \ldots\ldots (3)$$

This algorithm is modify the polar representation (leave that of students to write this algorithm)

Arc: is part of circle but is needed start angle & end angle and distance of center of arc to surrounding is same as.

Sector: is part of ellipse but contains two line; line1 from center to start angle & other line from center to end angle & distance between center of sector to surrounding is different as.





Scan-converting ARCs and Sectors

- **ARC** can be generated by:

1- Polynomial method

Value of x is varied from x_1 to x_2 and the values of y are found be evaluating the expression $\sqrt{r_2}+\sqrt{x_2}$.
An arc would be appear to be nothing more than portions of circles.

2- Trigonometric method

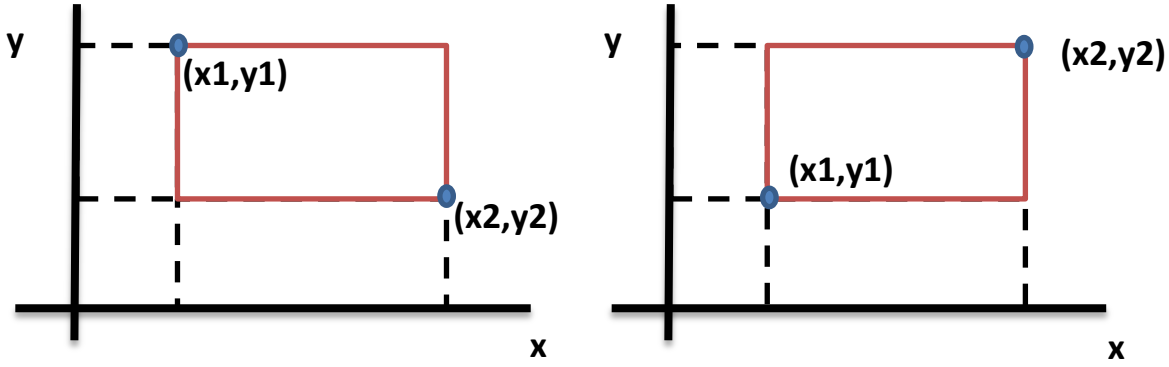
The starting value is set equal to θ_1 and the ending value is set equal to θ_2. The rest of the steps are similar to those used when scan-converting a circle except that symmetry is not used.

- **Sectors**

A sector is scan-converting an arc and then scan-converting two-lines from the center of the Arc to the endpoints of the Arc.
Ex./ Assume that a sector whose center is at point (h,k) is to be scan-converted.
- First scan-convert an Arc from θ_1 to θ_2.
- Next a line would be scan-converted from (h,k) to (r $\cos(\theta_1)$+h, r $\sin(\theta_1)$+k).
- A second line would be scan-converted from (h,k) to (r $\cos(\theta_2)$+h, r $\sin(\theta_2)$+k).

Scan-converting a rectangle

Rectangle sides are parallel to the coordinates axes may be constructed if the locations of two vertices are known. The remaining corner points are then derived. If we have the following rectangle.



The lines would be drawn as follows:

plot ([$x1$ $y1$] , [$x1$ $y2$] ,color)
plot ([$x1$ $y2$] , [$x2$ $y2$] ,color)
plot ([$x2$ $y2$] , [$x2$ $y1$] ,color)
Plot ([$x2$ $y1$] , [$x1$ $y1$] ,color)