UNIT-2

CHAPTER 3

FUNDAMENTAL ALGORITHMS

CONTENTS

3.0 OBJECTIVES:

After reading this chapter you will be able to:

- Learn about the algorithms of basic shapes.
- Draw lines using different algorithms
- Draw circles using different algorithms
- Draw ellipse
- Measure the efficiencies of different algorithms.

3.1 INTRODUCTION:

Graphics objects, though very complex, but at the bottom end, certain fundamental shapes plays central role. These are points, straight lines, circles, ellipses, arcs etc. Fundamentally the definition information is supplied to the graphics system and the system then uses this information to form a set of pixel for the display of the given primitive shape. For example, for a straight line, the two endpoints are given and the task of the graphic system is to generate the internal points of the straight line. The same thing is true for other shapes also (basic or complex). One of the major issues in this pixel geometry is that, the pixels are the smallest possible unit in the display device and we cannot go beyond a pixel. The pixel coordinate is always an integer, so the fractional value for a pixel will always be approximated to its nearest integer. So, the pixel generation should be done in such a way that we can avoid fractional result for pixel as largely as possible. This conversion from mathematical definition to pixel is known as scan conversion. For a scan conversion we need good algorithms which will handle the issues stated above as well as will be efficient with respect to time and space. There are numerous algorithms available starting from very simple shape scan conversion to that of very complex figures. In this chapter we shall discuss a few popular algorithms for basic shapes.

3.2 LINE DRAWING ALGORITHMS:

The first basic shape that we shall discuss is a straight line. In graphics straight line means straight line segment defined by two end points. The equation of straight line can be given by $y = mx + c$ where m is the slope of the line and c is the intercept to the y axis. We can directly use this equation for generating points on the straight line and convert them into pixels. But that requires a lot of calculations. Especially we will not be able to avoid floating point multiplication and round off calculations. There are certain algorithms which can avoid the same. Next we shall discuss two such algorithms. The DDA algorithm and the Bresenham's algorithm.

3.2.1 DDA ALGORITHM:

The Digital Differential Analyzer or DDA algorithm is very simple algorithm for straight line drawing.

We know that the equation for straight line is –

$$y = mx + c \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \; 3.1$$

where m is the slope or gradient of the line and c is the interception to the y-axis. If (X_1, Y_1) and(X_2, Y_2) be the two end points of the line segment then

$$m = \frac{Y_2 - Y_1}{X_2 - X_1} \text{ and} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.2$$

$$c = Y_1 - m \,.X_1 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.3$$

Let $\Delta x = X_2 - X_1$ ……………………………………..3.4

and $\Delta y = Y_2 - Y_1$ ……………………………………..3.5

So, m can be written as

$$m = \frac{\Delta y}{\Delta x} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.6$$

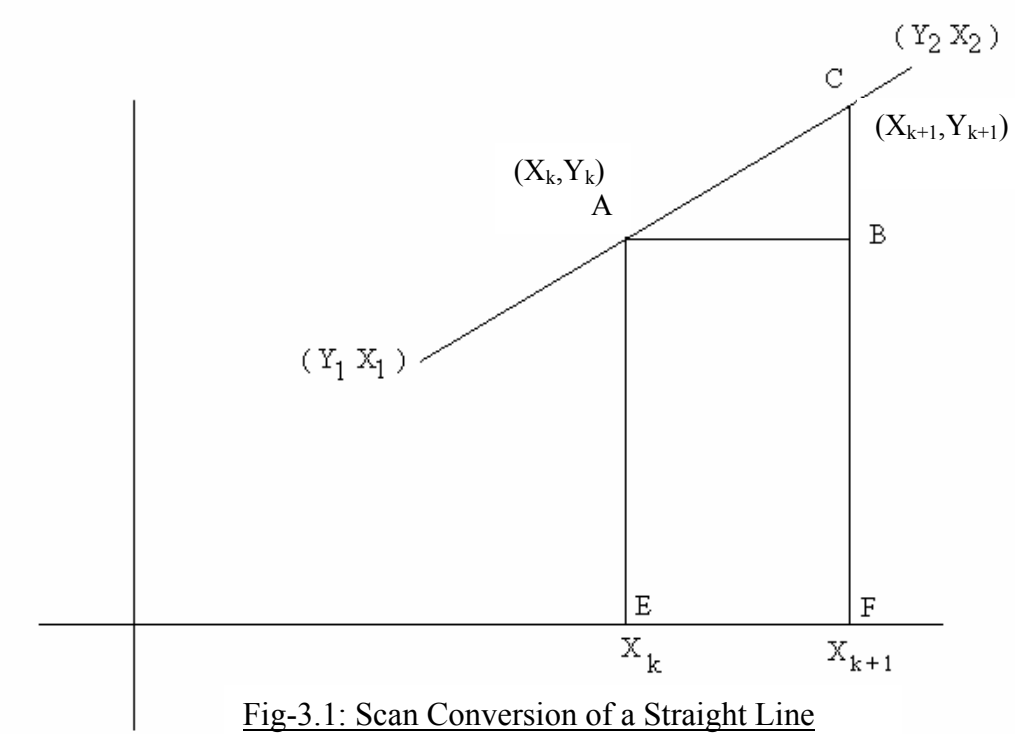

Fig-3.1: Scan Conversion of a Straight Line

Let the slope of the line lies between $0 < m < 1$. In this case the value of X_{k+1} will be X_k. +1 and we have to calculate the value of Y_{k+1} If the slope is greater than 1 i.e., $m > 1$ then Y_{k+1} will be $Y_k + 1$ and we have to calculate X_{k+1}.

Case 1: $0 < m < 1$:

Given any pint (X_k, Y_k) on the straight line, we have to find out the next point. As $0 < m < 1$, then X_{k+1} will be $X_k + 1$ and we have to find what will be value of Y_{k+1}.

From the above figure 3.1, it is clear that

$Y_{k+1} = FC.$

$= FB + BC$

$= Y_k + BC$ …….. 3.7.

Now BC/AB = m (slope of the line)

$\therefore$ BC = AB. m …….. 3.8.

Putting the value of BC in equation 3.7 we get

$Y_{k+1} = Y_k + AB.m.$

but AB = 1 (As $X_{k+1} = X_k + 1$)

So, $Y_{k+1} = Y_k + m.$ …….. 3.9.

Case 2: m>1 :

If m>1, then instead of X_k we shall take Y_k and calculate the value of X_{k+1} as –

$X_{k+1} = X_k + {}^1/_m$ …….. 3.10.

The process should start from (X_1, Y_1) point should continue for $X_2 - X_1$ iterations for 0<m<1 (for m>1 the iterations should be $Y_2 - Y_1$) and should reach to (X_2, Y_2) point. The algorithm is given below.

```
PROCEDURE       DDAline (X1, Y1, X2, Y2)
        Integer Δx, Δy, steps,
        float incremeantx, incrementy, x ,y.
        x = X1, y = Y1
        Δx =  X2- X1
        Δy =  Y2- Y1
        IF ( Δx >Δy)  THEN  // Assume Δx, Δy are positive
                steps = ΔX
        ELSE
                steps = ΔY
        ENDIF
        incrementX = ΔX/(float) steps
        incrementY = ΔY/(float) steps
        putpixel (ROUND (X), ROUND (Y))
        FOR  i = 0; i<steps ; i = i + 1
                x = x+ incrementX
                y = y + incrementY
                Putpixel (ROUND (X), ROUND (Y)
```

END FOR

END PROCEDURE

Though is a faster method, the round off and floating point approximation can generate some errors and the line may shift from the actual path.

3. 2. 2 BRESENHAM'S LINE DRAWING ALGORITHM:

Bresenham's line drawing algorithm is an efficient and accurate line drawing algorithm. It uses only integer calculations.

The method work as follows.

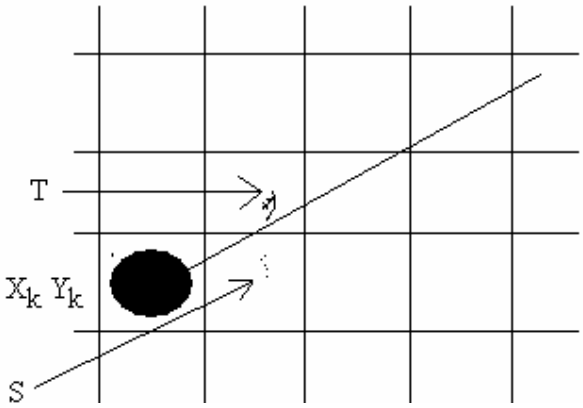

Fig-3.2: Bresenham's Line Drawing Method

Let the slope of the line m is 0 <m<1. assume that we have got the point (X_k, Y_k). We have to determine whether the next point will be (X_k+1, Y_k) or (X_k+1, Y_k+1). i.e., In the figure 3.2, whether the next point will be S or T. Clearly X_{k+1} is X_k+1 i.e, we shall increment the X-value by 1. We have to calculate the value of $Y_{k+1.}$ We shall find out the distance of the line from T and S and whichever distance is less we shall choose that pixel.

We know that

$y = m(X_k + 1) + c$ …….. ……. 3.11

$\therefore$ The distance of the line from S is

$d_1 = y - Y_k$

$= m(X_k + 1) - Y_k + c$

and distance of the line from the point T is

$d_2 = Y_k - y$

$= Y_k + 1 - m(X_k + 1) - c$

So, the difference between the distances is

$d1 - d_2 = m(X_k + 1) - Y_k + c - Y_k - 1 + m(X_k + 1) + c$

$= 2m(X_k + 1) - 2Y_k + 2c - 1$ ……...3.12.

Let us consider $P_k = \Delta X (d_1 - d_2)$ where $\Delta X = X_2 - X_1$ and $\Delta Y = Y_2 - Y_1$

And $m = {}^{\Delta Y}/_{\Delta X}$

$\therefore P_k = \Delta X (d_1 - d_2)$

$= 2\,\Delta Y(X_k + 1) - 2\Delta X Y_k + (2c - 1)\,\Delta X$

$= 2\,\Delta Y X_k - 2\Delta X Y_k + 2\Delta Y + (2c - 1)\,\Delta X$

$= 2\,\Delta Y X_k - 2\Delta X Y_k + K$ …...…….3.13

Where $K = 2\Delta Y + (2c - 1)\,\Delta X$ ……...…….3.14

As $\Delta X > 0$, clearly the sign of P_k is same as the sign of $(d_1 - d_2)$. The constant K is independent of pixel position (X_k, Y_k).

Clearly if $P_k < 0$ i.e., $d_1 < d_2$ i.e., the point S is closer to the line. i.e., the coordinate of the next pixel will be $(X_k + 1, Y_k)$ and if $P_k > 0$, i.e, $d_1 > d_2$, then the point T is closer to the line and we shall choose T and the next coordinate will be $(X_k + 1, Y_k + 1)$.

Putting k=k+1 in equation 3.13 we get

$P_{k+1} = 2\Delta Y X_{k+1} - 2\Delta X Y_{k+1} + K$

$\therefore P_{k+1} - P_k = 2\Delta Y(X_{k+1} - X_k) - 2\Delta X(Y_{k+1} - Y_k)$

Or , $P_{k+1} = P_k + 2\Delta Y - 2\Delta X (Y_{k+1} - Y_k)$, as $X_{k+1} = X_k + 1$ …….……3.15.

As per the above discussion, $Y_{k+1} = Y_k$ or Y_k+1 depending upon whether $P_k<0$ or $P_k \geq 0$.

$$P_{k+1} = \begin{cases} P_k + 2\Delta Y, & \text{If } P_k<0 \\ P_k + 2(\Delta Y - \Delta X), & \text{If } P_k \geq 0 \end{cases}$$

For initial value of P_k consider the equations 3.13 and 3.14

$P_1 = 2\Delta YX_1 - 2\Delta XY_1 + 2\Delta Y + (2C-1)\Delta X.$

Clearly from $Y_1 = mX_1 + c$ We get

$c = Y_1 - {}^{\Delta Y}/_{\Delta X}\, X_1.$

Putting the value of c in the above equation we gets

$P_1 = 2\Delta YX_1 - 2\Delta XY_1 + 2\Delta Y + (2Y_1 - 2\,{}^{\Delta Y}/_{\Delta X}\, X_1).\ \Delta X - \Delta X$

$= 2\Delta YX_1 - 2\Delta XY_1 + 2\Delta Y + 2\Delta XY_1 - 2\Delta YX_1 - \Delta X$

$= 2\Delta Y - \Delta X.$

Now we shall develop the Bresenham's line drawing algorithm.

```
PROCEDURE     BresenhamLine (X1; Y1, X2, Y2)
/*Assume 0 < m < 1 and X2>X1 and Y2> Y1  */
        Integer x =X1, y =Y1
        Integer  ΔX = X2–X1;  ΔY = Y2 –Y1;
        Integer dT = 2(ΔY–ΔX), dS = 2ΔY;
        Integer Pk = 2ΔY –ΔX.
        Putpixel (x, y)
        WHILE (x< X2)
                x = x +1
                IF (Pk <0) THEN
                   Pk = Pk + dS
                ELSE
                        y = y+1;
                        Pk = Pk + dT
                END IF
                Putpixel (x, y)
        END WHILE
END PROCEDURE
```

3.3 CIRCLE DRAWING ALGORITHMS:

In this section we will learn the algorithms and the mathematical foundations of the algorithms of scan-conversion of circles. Whatever may be the algorithm, we shall never calculate all the points of the circle rather we shall use one of the beautiful properties of circle, the symmetric property.

From a given point of view of a given point of a circle, a circle is eight-way symmetric. Let the given point on the circle is $P_1 = (x, y)$. Assuming the centre of the circle is in origin (0, 0); the other seven symmetric points will also be on the circle. These eight points are

$P_1 = (x, y)$, $P_2 = (y, x)$, $P_3 = (y, -x)$, $P_4 = (x, -y)$, $P_5 = (-x, -y)$, $P_6 = (-y, -x)$, $P_7 = (-y, x)$ and $P_8 = (-x, y)$.

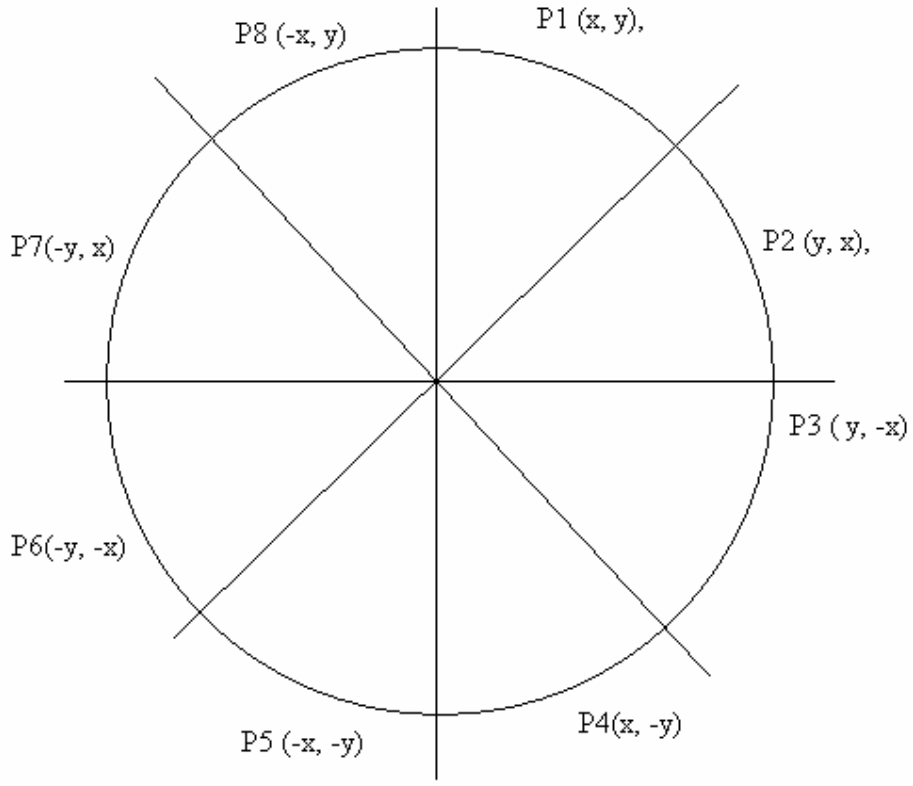

Fig-3.3: The Eight-Symmetric Property of a Circle.

The figure 3.3 will illustrate the matter. The points mentioned on the circle are the symmetric points.

Mathematical Definition of Circle: Assuming the centre of the circle is at origin, the equation of the $x^2 + y^2 = a^2$ when a is the radius of the circle and x and y are x and y coordinate respectively.

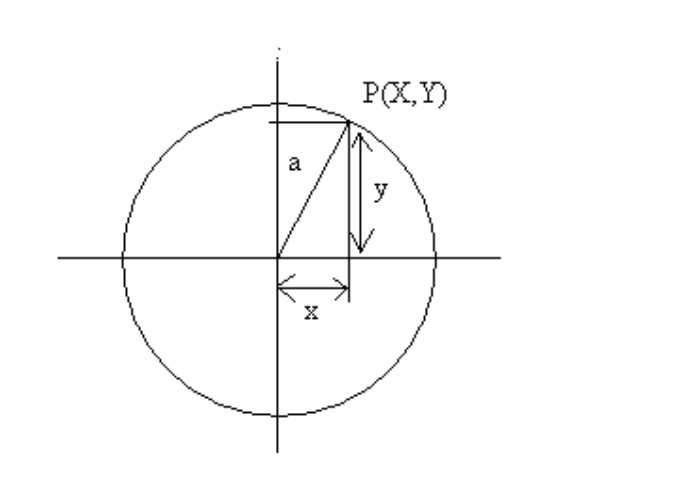

Fig-3.4 Locus of a point P is a circle when it is at fixed distance from a fixed point.

Let us try to find out the x and y coordinates in the 90^0 to 45^0 sector.

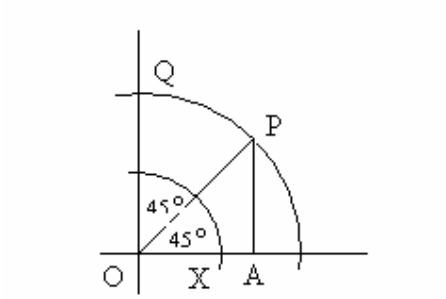

Fig-3.5: We shall find out the points of QP Sector

We shall find out the value of y for each unit increment of x from 90^0 to 45^0. Clearly in the figure 3.5 when the angle is 90^0 i.e., the locus of a point is (0, a) and when the radius forms 45^0 angle i.e., when the point is P then X coordinate is

$OA = OP \cos 45^0 = a / \sqrt{2}$

$\therefore$ We have to find the value of y from $y = \sqrt{(a^2 - x^2)}$ for each $x = 0$ to $x = a/\sqrt{2}$ incremented by 1 to find out one symmetric sector out of eight. This method is

very inefficient because it calculates the square of a and x and then subtraction and then one square root of the result in each iteration to find out y.

Another way to define a circle is to follow the trigonometric approach. In this approach the locus of the circle can be defined by

x = a Cosθ and y = a Sinθ, where θ is the angle in radian of the locus at the centre with the x axis. In this method θ is considered from π/2 radian to π/4 radian and the x and the y values are calculated accordingly. This method is even more time consuming than the earlier method as it involves the calculation of Sine and Cosine of angles at every iterations.

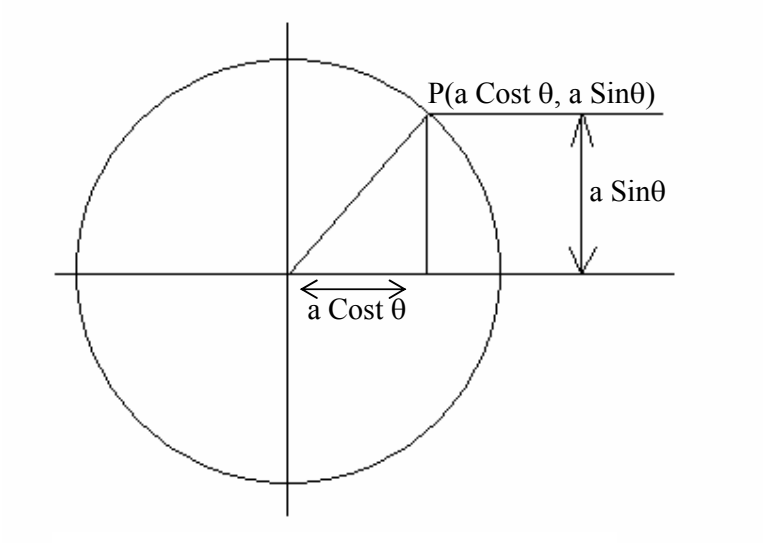

Fig-3.6: Parametric equation of a circle

3.3.1 BRESENHAM'S CIRCLE DRAWING ALGORITHM:

Whatever is the calculation process we have to convert the x values and y values ultimately to pixel coordinate which takes integer values only. So the involvement of squares operation square root operation or sine or cosine function will be of no extra accuracy because ultimately it is to be approximated by the nearest integer value. Considering this fact, Bresenham proposes an algorithm which involves no square operation, square root operation, since or cosine operation. It simply involves addition, subtraction and simple multiplication to

find the scan converted points. This algorithm is known as Bresenham's circle drawing algorithm. Let us first develop the working principle of the algorithm.

If we consider the eight-way symmetricity of a circle we have to calculate only one-eighth of a circle and all other points can be found out directly from this point. So we shall only consider the sector from 90^0 to 45^0 in our algorithm.

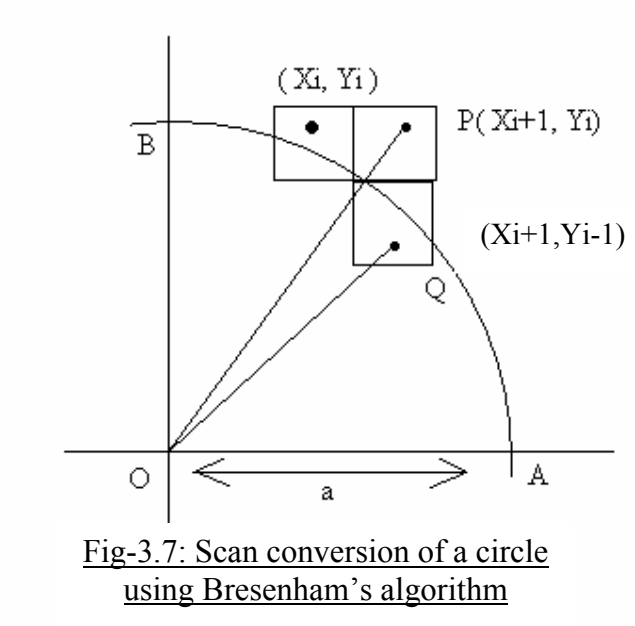

Fig-3.7: Scan conversion of a circle using Bresenham's algorithm

If we take a closure look to the figure-3.7 we will see that to work with the sector 90^0 to 45^0, we have to move in the positive direction of x and in the negative direction of y. i.e. x will increase and y will decrease. The best approximation clearly is the set of those pixels which forms the least distance from the true circle and if we take a more closure look to the figure-3.7 we shall see that, this can be achieved by tow ways. Either (i) increment x by 1 or (ii) increment x by 1 and decrement y by 1. In our figure-3.7, these points are $P(X_i+1,Y_i)$ and $Q(X_i+1,Y_i-1)$ respectively. Bresenham's algorithm finds the closest among them from the circle to find out the next pixel coordinate.

Let us assume that (X_i, Y_i) is the last scan-converted pixel for the step i. Let the squared distance (SD) from a point to the circle is defined by

SD(Point location) = (square of the distance of the point from the origin) –

(square of the distance of the circle from the origin),

if the point P falls outside of the Circle.

Or

SD(Point location) = (square of the distance from the circle to the centre) –

(square of the distance from the point to the centre),

if the point falls inside the circle.

From this we get,

$SD(P) = (X_i + 1)^2 + Y_i^2 - a^2$ since the point p falls outside the circle and

$SD(Q) = a^2 - (X_i + 1)^2 - (Y_i - 1)^2$ as the point falls inside the circle.

Let us define decision variable d_i as follows .

$d_i = SD(P) - SD(Q)$ ...……..……..3.16

$= 2(X_i + 1)^2 + (Y_i)^2 + (Y_i - 1)^2 - 2a^2$ ……..……..3.17

Clearly if P is closure to Q then d_i will be less than zero and of Q is closure to P then d_i is greater than zero and we will choose P or Q according to the sign of d_i.

The discussion so far we have done, no improvement is observed. The calculation of d_i, given X_i and Y_i, involves several square operations.

Let us try to find out d_{i+1} from the given point(X_{i+1},Y_{i+1}).

Clearly,

$d_{i+1} = (X_{i+1}+1)^2 + Y^2_{i+1} + (Y_{i+1} - 1)^2 - 2a^2$ …….. 3.18

Now, $d_{1+1} - d_i = 2(X_{i+1}+1)^2 + Y^2_{i+1} + (Y_{i+1} - 1)^2 - 2a^2 - 2(X_i+1)^2 - Y^2_i$

$-(Y_i - 1)^2 + 2a^2$

$=2[(X_{i+1}+1)^2 - (X_i+1)^2] + (Y^2_{i+1} - Y_i^2) + [(Y_{i+1} - 1)^2 - (Y_i - 1)^2]$ ……...3.19

We know that $X_{i+1} = X_i+1$ and putting this value to equation-3.19 we get,

$d_{i+1} - d_i = 2\,[(X_i+1+1)^2 - (X_i+1)^2\,] + (Y^2_{i+1} - Y^2_i) + (Y^2_{i+1} - 2Y_{i+1} + 1 - Y^2_i + 2Y_i - 1]$

$= 2[X_i^2 + 4X_i + 4 - X_i^2 - 2X_i - 1] + 2(Y^2_{i+1} - Y^2_i) - 2(Y_{i+1} - Y_i)$

$= 4X_i + 6 + 2(Y^2_{i+1} - Y^2_i) - 2(Y_{i+1} - Y_i)$

$\therefore d_{i+1} = d_i + 4X_i + 2(Y^2_{i+1} - Y^2_i) - 2(Y_{i+1} - Y_i) + 6$ ……3.20

Now, if the next co-ordinate is (X_i+1, Y_i) i.e., $Y_{i+1} = Y_i$ i.e,. $d_i < 0$

Then $d_{i+1} = d_i + 4X_i + 6$ …….....3.21

If the next pixel is $(X_i + 1, Y_i - 1)$

i.e., $Y_{i+1} = Y_i - 1$ i.e., $d_i \geq 0$

$d_{i+1} = d_i + 4X_i + 2[(Y_i - 1)^2 - Y_i^2] - 2(Y_i - 1 - Y_i) + 6$

$= d_i + 4X_i + 2[-2Y_i + 1] + 2 + 6$

$= d_i + 4X_i - 4Y_i + 10$

$= d_i + 4(X_i - Y_i) + 10$ …….3.22

Hence d_{i+1} can be written in the form

$$d_{i+1} = \begin{cases} d_i + 4X_i + 6 & \text{If } d_i < 0 \\ d_i + 4(X_i - Y_i) + 10 & \text{If } d_i \geq 0 \end{cases}$$ …….3.23

Now, clearly (0, a) is the starting pixel (actually integral part of a will be considered) and we get the value of d_i from the Equation 3.17 as

$d_i = 2(0 + 1)^2 + a^2 + (a - 1)^2 - 2a^2$

$= 2 + a^2 + a^2 - 2a + 1 - 2a^2$

$= 3 - 2a$ …….3.24

One final point to note that when we are at B of figure-3.7, the x = 0, y = a. At 45^0, $x = a/\sqrt{2}$ and $y = a/\sqrt{2}$ i.e, x and y will be equal. If we go father x will increase and y will decrease i.e., we shall iterate from x = 0 to x <= y. i.e. from 90^0 to 45^0.

Now we are ready to present the algorithm. If the origin is not (0, 0) then the only task we have to do is add the coordinate of the center with the scan-converted pixels.

```
PROCEDURE BresenhamCircle (radius)
    Integer x y, d;
    x = 0
    y = radius
    d = 3 –2 radius
    WHILE (x<= y)
        Putpixel (x , y)
        /*Put other seven points of the circle*/
        IF (d < 0) THEN
            d = d + 4*x + 6
        ELSE
            d = d + 4(X – Y) + 10
            y = y + 1
        END IF
        x = x + 1;
    END WHILE
END PROCEDURE
```

3.3.2 MIDPOINT CIRCLE ALGORITHM:

Now we present another circle drawing algorithm called mid point circle drawing algorithm. It is based on the relationship between arbitrary point (x, y) and a circle of radius a.

We know that

$$f(X,Y) = X^2 + Y^2 - a^2 \begin{cases} <0 & \text{If (X,Y) is inside the circle.} \\ =0 & \text{If (X,Y) is on the circle.} \\ >0 & \text{if (X,Y) is outside the circle.} \end{cases}$$

In the Bresenham algorithm we considered the distance between point P, Q and the radius of the circle. If this cone we consider the middle of pixel P and Q of Fig-3.8.

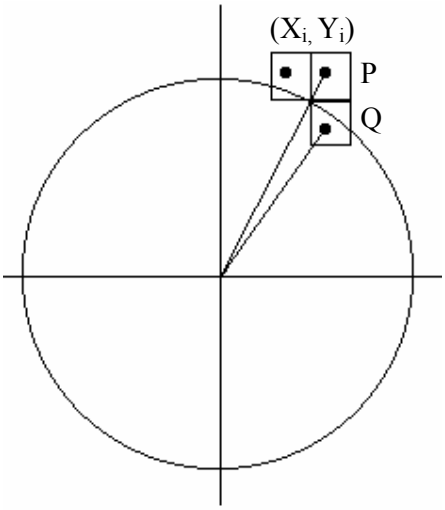

Fig-3.8: Midpoint Circle Drawing

The mid point is ($X_i + 1, Y_i - ½$). This is called the midpoint. Now we define the decision parameter

$MID_i = f (X_i + 1, Y_i - ½) = (X_i + 1)^2 + (Y_i - ½)^2 - a^2$ ..…….3.25

Clearly if MID_i is negative, then mid point is inside the circle i.e., P is closure than Q. So, P is the selected point. On the other hand if MID_i is positive then the mid point is outside the circle. So, Q is closure to the circle than P. So, Q is selected. If MID_i is 0 then we shall brake the conflict by choosing Q.

Now, $MID_{i+1} = (X_{i+1} + 1)^2 + (Y_{i+1} - ½)^2 - a^2$.

Now let us consider $MID_{i+1} - MID_i$

$MID_{i+1} - MID_i = (x_{i+1} + 1)^2 + (Y_{i+1} - ½)^2 - a^2 - (x_i + 1)^2 - (Y_i - ½)^2 + a^2$

Since $X_{i+1} = X_i + 1$ we get,

$MID_{i+1} - MID_i = (X_i + 1 + 1) - (X_i + 1)^2 + (Y_{i+1} - ½)^2 - (Y_i - ½)^2$

$= (X_i^2 + 4X_i + 4 - X^2_i - 2X_i - 1 + Y^2_{i+1} - Y_{i+1} + ¼ - Y^2_i + Y_i - ¼$

$= 2X_i + 3 + (Y^2_{i+1} - Y^2_i) - (Y_{i+1} - Y_i)$ …….. 3.26

Now if P is chosen i.e., $MID_i < 0$

Then $Y_{i+1} = Y_i$

Then the equation-3.26 becomes

$$MID_{i+1} - MID_i = 2X + 3 \quad \text{.....3.27}$$

If Q is chosen, i.e., $MID_i >= 0$ then $Y_{i+1} = Y_i - 1$ and

$$MID_{i+1} - MID_i = 2X_i + 3 + [(Y_i - 1)^2 - Y^2_i] - [(Y_i - 1) - Y_i]$$
$$= 2X_i + 3 + [-2Y_i + 1] + 1$$
$$= 2(X_i - Y_i) + 5 \quad \text{.....3.28}$$

From equation 3.26 and 3.27 we get

$$MID_{i+1} = \begin{cases} MID_i + 2X_{i+1} + 1; & \text{If } MID_i < 0 \\ MID_i + 2(X_{i+1} - Y_{i+1}) + 1; & \text{If } MID_i >= 0 \end{cases} \quad \text{.....3.29}$$

Finally, the initial value can be computed by putting (0, a) in the equation 3.25 we get,

$$MID_1 = (0 + 1)^2 + (a - 1/2)^2 - a^2$$
$$= 1 + a^2 - a + ¼ - a^2$$
$$= 5/4 - a \quad \text{.....3.30}$$

Clearly 5/4 is not an integer but that is not a problem. When a is integer, 5/4 will be approximated to 1 and the computation will be $MID_1 = 1 - a$.

The algorithm of the midpoint circle drawing is given below.

```
PROCEDURE  MIDcircle (int a)   /* Assuming centre at (0,0)*/
    Integer x = 0 , y = a, Mid = 1–a
    WHILE (x <= y)   /* 90o to 45o */
        Putpixel (x, y)
        /* Put other symmetric pixels.*/
        IF (Mid < 0) THEN
            Mid = Mid + 2x + 3
        ELSE
            Mid = Mid + 2 (x-y) +5
            y = y–1
```

END IF

x = x +1

END WHILE

END PROCEDURE

3.3.3 ARBITRARY CENTERED CIRCLE:

The above discussions are all based on the assumption that the centre of the circle is at origin i.e. at (0, 0). If the centre of the circle is not at origin and it is an arbitrary point, then we have to translate the circle with respect to that point. Let us suppose that the centre of the circle is at (h, k), then every point (x,y) on the circle will be translated to (x+h, y+k) point. In case of Putpixel() function, the function will be called with Putpixel(x+h, y+k). other than this all other things will be same.

3.4 ELLIPSE DRAWING ALGORITHM:

The equation of an ellipse can be given by

$$\frac{X^2}{a^2} + \frac{Y^2}{b^2} = 1 \quad \text{where}$$

a is the length of major axis and b is the length of minor axis. In the above equation it is assumed that the centre of the ellipse is (0, 0). If instead of (0, 0) the centre would be arbitrary (h, k) then the equation could be given by

$$\frac{(X-h)^2}{a^2} + \frac{(Y-k)^2}{b^2} = 1$$

Parametric equation of an ellipse:

The parametric equation of an ellipse is

$X = a \cos\theta + h$ and

$Y = b \sin\theta + k$

where (h, k) is the coordinate of the centre of the ellipse and θ is the angle of the locus of the current point at the centre of the ellipse with the x axis.

An ellipse can easily be generated from either of its equations but it performs several computations like sine or cosine in case of parametric equation and square or division in case of general definition.

Now we shall discuss an algorithm which performs only addition and multiplication to scan convert an ellipse.

3.4.1 MIDPOINT ELLIPSE DRAWING ALGORITHM:

An ellipse is a four way symmetric figure. So the entire ellipse will be divided into four parts each belongs to one quadrant. And we shall calculate the points or pixels values of one quadrant only. The other points will be generated from these values.

Let the calculated point is (X, Y)

So the four symmetric points are

	At first quadrant	(X, Y)
	At second quadrant	(-X, Y)
	At third quadrant	(-X, -Y)
And	At fourth quadrant	(-X, -Y)

We know that the equation of an ellipse centered at origin is

$$\frac{X^2}{a^2} + \frac{Y^2}{b^2} = 1$$

Rewriting the equation we get

$$b^2X^2 + a^2 Y^2 - a^2 b^2 = 0 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. \; 3.31$$

Let us define

$$f(X,Y) = b^2X^2 + a^2 Y^2 - a^2 b^2$$

Clearly

$$f(X,Y) = b^2X^2 + a^2 Y^2 - a^2 b^2 \quad \begin{cases} < 0 \text{ if } (X,Y) \text{ is inside the ellipse} \\ = 0 \text{ if } (X,Y) \text{ is on the ellipse} \\ > 0 \text{ if } (X,Y) \text{ is outside the ellipse} \end{cases}$$

We shall calculate the pixel points of the first quadrant by diving it into two parts.

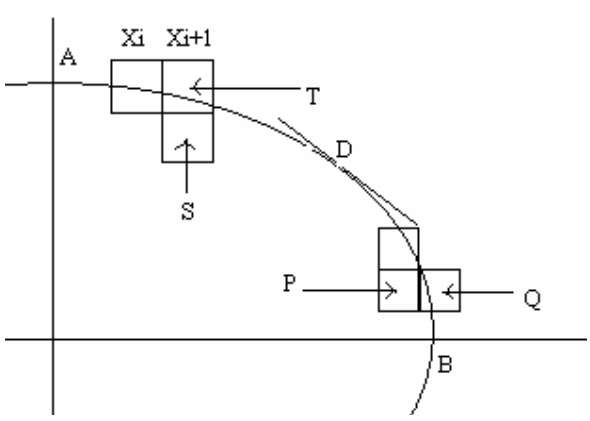

Fig-3.9: Scan conversion of an ellipse

We shall find out the tangent to the ellipse where tangent value is 45^0(or 135^0) to the x-axis i.e., the dY/dX = -1. Let that point on the ellipse is D. then the two parts are above D i.e., AD part and below D i.e, DB part.

The equation of an ellipse is

$$b^2X^2 + a^2 X^2 – a^2 b^2 = 0$$

Taking derivatives with respect to X we get

$$2b^2X + a^2 Y \, dY/dX = 0$$

$$\text{or } dY/ dX = - b^2X / a^2 Y$$

we can take partial derivatives of the above equation with respect to X and Y and we get,

$f_X = 2b^2X$ and

$f_Y = 2a^2Y$ and

Clearly in AD part of the ellipse $f_X < f_Y$.

We shall start our calculation from (0, b) point i.e., from A. We will now discuss the calculation of AD part of the ellipse.

Suppose the last scan converted pixel at i^{th} iteration is (X_i, Y_i). We have to select the next point. The next point will be either T or S. If it is T, the coordinate will be (X_{i+1}, Y_i). If S is selected then the coordinate will be (X_{i+1}, Y_{i-1}). Let us take the mid point of S and T and take the decision parameter P_i as

$$P_i = f(X_i+1, Y_i + ½) = b^2 (X_i+1)^2 + a^2 (Y_i – ½)^2 – a^2b^2 \quad \text{……3.32}$$

Clearly if $P_i > 0$ then the midpoint is outside the ellipse and S is closure to ellipse and so S will be the next point. On the other hand, if $P_i < 0$, that means mid point is inside the ellipse so T will be chosen.

Let us calculate P_{i+1}

$$P_{i+1} = b^2(X_{i+1}+1)^2+a^2 (Y_{i+1} - ½)^2 - a^2b^2 \qquad \text{……3.33}$$

Now $P_{i+1} - P_i$ gives

$$P_{i+1} - P_i = b^2[(X_{i+1}+1)^2 - (X_i+1)^2] + a^2[(Y_{i+1} - ½)^2 - (Y_i - ½)^2]$$

We know that $X_{i+1} = X_i+1$. so putting this value in the above equation we get,

$$P_{i+1} - P_i = b^2[(X^2_{i+1}+2X_{i+1}+1-X_{i+1}^2] + a^2[(Y_{i+1} - ½)^2 - (Y_i - ½)^2]$$

$$P_{i+1} = P_i + 2b^2X_{i+1}+b^2 + a^2[(Y_{i+1} - ½)^2 - (Y_i - ½)^2] \qquad \text{…….3.34}$$

Clearly if $P_i < 0$ i.e., T is chosen, then $Y_{i+1} = Y_i$ and if $P_i > 0$ i.e., S is chosen, then $Y_{i+1} = Y_i - 1$.

So we can write

$$P_{i+1} = \begin{cases} P_i + 2b^2X_{i+1}+b^2 \; ; & \text{If } P_i < 0 \\ P_i + 2b^2X_{i+1}+b^2 - 2\,a^2y_{i+1} \; ; & \text{If } P_i >= 0 \end{cases} \qquad \text{…….3.34}$$

The initial value of P_i i.e., P_1 can be derived from equation 3.32 putting (0, b) point.

$$P_1 = b^2 + a^2 (b - ½)^2 - a^2b^2$$

$$\text{Or } P_1=b^2+a^2 (b^2-2b+ {}^1/_4) - a^2b^2$$

$$\text{Or } P_1=b^2-2a^2b+ a^2/_4 \qquad \text{……..3.35}$$

Now let us derive the calculations for DB part. In the DB part, the current scan converted point is (X_k, Y_k). We have to find out whether the next point will be P or Q. If it is P, then the coordinate will be (X_k, Y_k-1). If it is Q, then the coordinate will be (X_{k+1}, Y_k-1).

Let the decision parameter be q_k where

$$q_k = f(X_k+½, Y_k - 1) = b^2 (X_k+½)^2 + a^2 (Y_k - 1)^2 - a^2b^2 \qquad \text{…….3.36}$$

Clearly, if $q_k <0$, then Q is chosen and if $q_k >=0$, then P is chosen.

Let us define q_{k+1} as

$q_{k+1} = f(X_{k+1}+½, Y_{k+1} -1) = b^2 (X_{k+1}+½)^2 + a^2 (Y_{k+1} -1)^2 - a^2b^2$

since $Y_{k+1} = Y_k - 1$, we have

$$q_{k+1} - q_k = b^2 [(X_{k+1}+½)^2 - (X_k+½)^2] + a^2 [(Y_{k+1} -1)^2 - (Y_k -1)^2]$$
$$= b^2 [(X_{k+1}+½)^2 - (X_k+½)^2] + a^2 [(Y_{k+1} -1)^2 - Y_{k+1}^2]$$
$$= b^2 [(X_{k+1}+½)^2 - (X_k+½)^2] + a^2 [-2Y_{k+1} +1]$$
$$= b^2 [(X_{k+1}+½)^2 - (X_k+½)^2] + -2a^2Y_{k+1} + a^2$$

Clearly if $q_k<0$, then Q is chosen. i.e. $X_{k+1} = X_k+1$ and if $q_k>=0$ then P is chosen and $X_{k+1} = X_k$.

So,

$$q_{k+1} = \begin{cases} q_k + 2b^2X_{k+1} - 2a^2Y_{k+1} + a^2 & \text{If } q_k<0 \\ q_k - 2a^2Y_{k+1} + a^2 & \text{If } q_k>=0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.37$$

The value of q_1 can be obtained from the last pixel generated in the first part. Let that be (X_1, Y_1).

Therefore

$$q_1 = f(X_1+½, Y_1 -1) = b^2 (X_1+½)^2 + a^2 (Y_1 -1)^2 - a^2b^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots 3.38$$

We shall now discuss the midpoint ellipse drawing algorithm. We shall generate the points of the first quadrant only. The points of the other quadrants will be obtained from the symmetric property.

```
PROCEDURE    midpointEllipse (Integer a, Integer b)
        Integer x = 0, y = b
        Integer aa  = a^2, bb = b^2, aa2 = 2a^2, bb2 = 2b^2
        Integer fx = 0, fy= aa2* b  /*initial partial derivatives*/
        Integer pk = bb –aa * b – 0.25 *aa, qk;
        WHILE (fx < fy)       /* i.e., Slope < 1  */
                Putpixel (x,y).
                /* Put other three pixels  */
                x = x +1
                fx = fx + bb2
                IF (Pk < 0 ) THEN
                        Pk = Pk + fx + bb
                ELSE
                        y = y-1
                        fy = fy – aa2
                        Pk = Pk + fx + bb–fy
```

```
            END IF
        END WHILE
        Putpixel (x, y)
        /* put other three pixels  */
        qk = bb(x + 0.5)^2 + aa(y-1)^2 – aa * bb
        WHILE (y>0)
            y = y -1
            fy = fy –aa2
            IF (qk > 0) THEN
                qk = qk – fy + aa
            ELSE
                x = x+1
                fx = fx +bb2
                qk = qk +fx –fy+aa
            END IF
            Putpixel (x,y)
            /* put other three pixels.*/
        END WHILE
    END PROCEDURE
```

In this algorithm we have assumed that the centre of the ellipse is at (0,0). If it is at (h, k) instead of (0,0) then we have to add h with x and k with y to get the arbitrary centered ellipse.

3.5 PROGRAMS OF THE ALGORITHMS:

Following are the programs of the algorithms discussed in this chapter. These programs are written in C and it assumes all the graphics related files are at F:\tc\bgi directory.

3.5.1 PROGRAM FOR DDA ALGORITHM:

```
#include<graphics.h>
#include<conio.h>
int main()
{
```

```
 int gd=DETECT,gm;
 int i,steps;
 int x1=100,y1=100,x2=200,y2=300;
 int dx=x2-x1,dy=y2-y1;
 float x=x1,y=y1,xincr,yincr;
 initgraph(&gd,&gm,"F:\\tc\\bgi");
 if(dx>dy)
  steps=dx;
 else
  steps=dy;
 xincr=dx/(float)steps;
 yincr=dy/(float)steps;
 putpixel(x,y,GREEN);
 for(i=1;i<=steps;i++)
  {
   x=x+xincr;
   y=y+yincr;
   putpixel(x,y,GREEN);
  }

 getch();
 closegraph();
 restorecrtmode();
 return(0);
 }
```

3.5.2 PROGRAM FOR BRESENHAM LINE DRAWING ALGORITHM:

```
#include<graphics.h>
#include<conio.h>
int main()
 {
  int x1=100,y1=100,x2=400,y2=200;
  int dx2=2*(x2-x1),dy2=2*(y2-y1);
  int pk=dy2-(x2-x1);
  int x=x1,y=y1,k,steps=x2-x1,gd=DETECT,gm;
```

```
   initgraph(&gd,&gm,"f:\\tc\\bgi");
   putpixel(x,y,GREEN);
   for(k=1;k<=steps;k++)
    {
        if(pk<0)
           pk=pk+dy2;
        else
          {
           y++;
           pk=pk+dy2-dx2;
          }
        x++;
        putpixel(x,y,GREEN);
    }
   getch();
   closegraph();
   restorecrtmode();
   return(0);
 }
```

3.5.3 PROGRAM FOR BRESENHAM CIRCLE DRAWING ALGORITHM:

```
#include<graphics.h>
#include<conio.h>
void circle_bresen(int x_co,int y_co,int r)
  {
   int p=3-2*r;
   int x=0,y=r,col;
   col=getcolor();
   putpixel(x+x_co,y+y_co,col);
   while(y>=x)
    {
        if(p<0)
          p+=4*x+6;
        else
          {
```

```
            p+=4*(x-y)+10;
            y--;
           }
          x++;
          putpixel(x+x_co,y+y_co,col);
          putpixel(x+x_co,-y+y_co,col);
          putpixel(-x+x_co,y+y_co,col);
          putpixel(-x+x_co,-y+y_co,col);
          putpixel(y+x_co,x+y_co,col);
          putpixel(y+x_co,-x+y_co,col);
          putpixel(-y+x_co,x+y_co,col);
          putpixel(-y+x_co,-x+y_co,col);
     }
  }
int main()
  {
   int x_co=200,y_co=200;
   int r=100;
   int gd=DETECT,gm;
   initgraph(&gd,&gm,"F:\\tc\\bgi");
   setcolor(GREEN);
   circle_bresen(x_co,y_co,r);
   getch();
   closegraph();
   restorecrtmode();
   return(0);
  }
```

3.5.4 PROGRAM FOR MIDPOINT CIRCLE DRAWING ALGORITHM:

```
#include<graphics.h>
#include<conio.h>
void circle_midpoint(int x_co,int y_co,int r)
  {
   int p=1-r;
   int x=0,y=r,col;
   col=getcolor();
```

```
    putpixel(x+x_co,y+y_co,col);
    while(y>=x)
      {
          if(p<0)
            p+=2*x+3;
          else
            {
              p+=2*(x-y)+5;
              y--;
            }
          x++;
          putpixel(x+x_co,y+y_co,col);
          putpixel(x+x_co,-y+y_co,col);
          putpixel(-x+x_co,y+y_co,col);
          putpixel(-x+x_co,-y+y_co,col);
          putpixel(y+x_co,x+y_co,col);
          putpixel(y+x_co,-x+y_co,col);
          putpixel(-y+x_co,x+y_co,col);
          putpixel(-y+x_co,-x+y_co,col);

      }
  }
int main()
  {
    int x_co=200,y_co=200;
    int r=100;
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"F:\\tc\\bgi");
    setcolor(GREEN);
    circle_midpoint(x_co,y_co,r);
    getch();
    closegraph();
    restorecrtmode();
    return(0);
  }
```

3.5.5 PROGRAM FOR MIDPOINT ELLIPSE DRAWING ALGORITHM:

```
#include<graphics.h>
#include<conio.h>

void putpixels(int x_co,int y_co,int x,int y,int color)
 {
  putpixel(x_co+x,y_co+y,color);
  putpixel(x_co-x,y_co+y,color);
  putpixel(x_co+x,y_co-y,color);
  putpixel(x_co-x,y_co-y,color);
 }
void mid_ellipse(int x_co,int y_co,int a,int b)
 {
  int x=0,y=b;
  int aa=a*a,bb=b*b,aa2=2*a*a,bb2=2*b*b;
  int delx=0,dely=aa2*b;
  int pk=0.5+(bb-aa*b+(0.25*aa)),qk;
  while(delx<dely)
    {
        putpixels(x_co,y_co,x,y,GREEN);
        x++;
        delx+=bb2;
        if(pk<0)
          pk+=delx+bb;
        else
          {
           y--;
           dely-=aa2;
           pk=pk+bb+delx-dely;
          }
    }
  putpixels(x_co,y_co,x,y,GREEN);
  qk=bb*(x+0.5)*(x+0.5)+aa*(y-1)*(y-1)-aa*bb;
  while(y>0)
    {
```

```
            y--;
            dely-=aa2;
            if(qk>=0)
              qk=qk-dely+aa;
            else
              {
                x++;
                delx=delx+bb2;
                qk=qk+delx-dely+aa;
              }
            putpixels(x_co,y_co,x,y,GREEN);
        }
    }
int main()
  {
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"F:\\tc\\bgi");
    mid_ellipse(300,200,40,10);
    getch();
    closegraph();
    restorecrtmode();
    return(0);
  }
```

3.6 KEY WORDS:

- DDA
- Decision Parameter
- Scan Conversion
- Symmetric Property

3.7 SAMPLE QUESTIONS:

3.7.1 Discuss the mathematical foundation for DDA algorithm.

3.7.2 Discuss the mathematical foundation for Bresenham line drawing algorithm.

3.7.3 Write down the Bresenham line drawing algorithm.

3.7.4 Develop the midpoint circle drawing algorithm.

3.7.5 Develop the midpoint ellipse drawing algorithm.