Else

p = p + 2(x-y)+5

y =y - 1 (end if)

x =x+1 (end loop)

Step3: End

Program –

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void pixel(int x,int y,int xc,int yc)
{
        putpixel(x+xc,y+yc,BLUE);
        putpixel(x+xc,-y+yc,BLUE);
        putpixel(-x+xc,y+yc,BLUE);
        putpixel(-x+xc,-y+yc,BLUE);
        putpixel(y+xc,x+yc,BLUE);
        putpixel(y+xc,-x+yc,BLUE);
        putpixel(-y+xc,x+yc,BLUE);
        putpixel(-y+xc,-x+yc,BLUE);
}
main()
{
        int gd=DETECT,gm=0,r,xc,yc,x,y;
        float p;
        //detectgraph(&gd,&gm);
        initgraph(&gd,&gm," ");
```

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

```
        printf("\n Enter the radius of the circle:");
        scanf("%d",&r);
        printf("\n Enter the center of the circle:");
        scanf("%d %d",&xc,&yc);
        y=r;
        x=0;
        p=(5/4)-r;
        while(x<y)
        {
                if(p<0)
                {
```

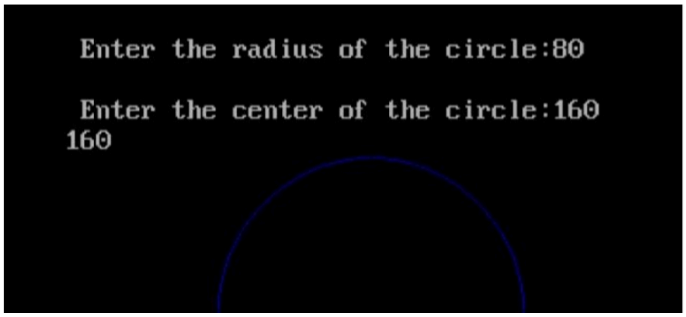
```
int gd=DETECT,gm=0,r,xc,yc,x,y;
float p;
//detectgraph(&gd,&gm);
initgraph(&gd,&gm," ");
```



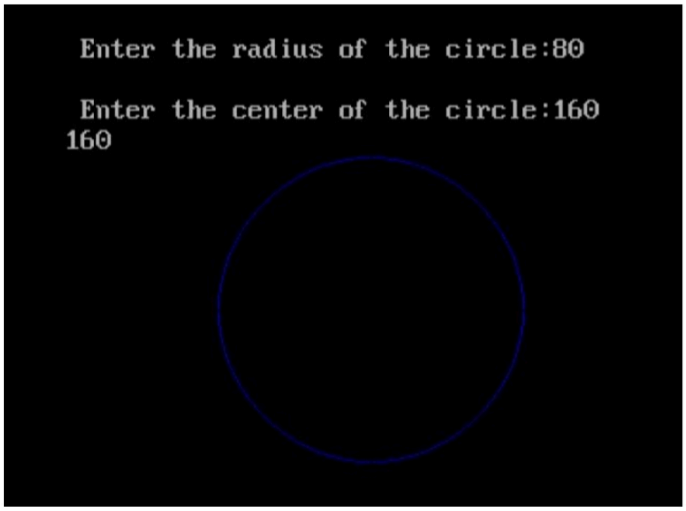
```
printf("\n Enter the radius of the circle:");
scanf("%d",&r);
printf("\n Enter the center of the circle:");
scanf("%d %d",&xc,&yc);
y=r;
x=0;
p=(5/4)-r;
while(x<y)
{
    if(p<0)
    {
        x=x+1;
        y=y;
        p=p+2*x+3;
    }
    else
    {
        x=x+1;
        y=y-1;
        p=p+2*x-2*y+5;
    }
    pixel(x,y,xc,yc);
}
getch();
closegraph();
}
```

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

output –



output –



Conclusion: Comment on

1. Fast or slow

2. Draw one arc only and repeat the process in 8 quadrants

3. Difference with line drawing method

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

Experiment No. 4

Aim-To implement midpoint Ellipse algorithm

Objective:

Draw the ellipse using Mid-point Ellipse algorithm in computer graphics. Midpoint ellipse algorithm plots (finds) points of an ellipse on the first quadrant by dividing the quadrant into two regions.

Theory:

Midpoint ellipse algorithm uses four way symmetry of the ellipse to generate it. Figure shows the 4-way symmetry of the ellipse.

2. Draw one arc only and repeat the process in 8 quadrants

3. Difference with line drawing method

Experiment No. 4

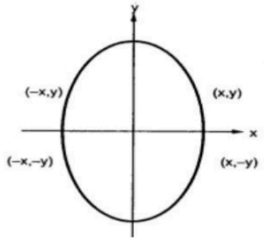
Aim-To implement midpoint Ellipse algorithm

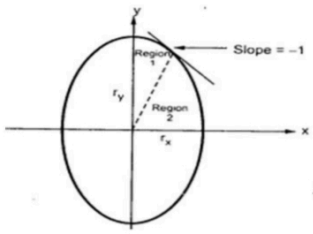
Objective:

Draw the ellipse using Mid-point Ellipse algorithm in computer graphics. Midpoint ellipse algorithm plots (finds) points of an ellipse on the first quadrant by dividing the quadrant into two regions.

Theory:

Midpoint ellipse algorithm uses four way symmetry of the ellipse to generate it. Figure shows the 4-way symmetry of the ellipse.





Here the quadrant of the ellipse is divided into two regions as shown in the fig. Fig. shows the division of first quadrant according to the slope of an ellipse with rx < ry. As ellipse is drawn from 90 0 to 0 0 , x moves in positive direction and y moves in negative direction and ellipse passes through two regions 1 and 2.

The equation of ellipse with center at (xc, yc) is given as -

[(x – xc) / rx] 2 + [(y – yc) / ry] 2 = 1

Therefore, the equation of ellipse with center at origin is given as -

[x / rx] 2 + [y / ry] 2 = 1

i.e. x 2 ry 2 + y 2 rx 2 = rx 2 ry 2

Let, f ellipse (x, y) = x2 ry2 + y2 rx2 - rx2 ry2

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

Algorithm:

Step 1: Start

Step 2: Declare r_x , r_y, x , y , m , dx , dy , P , P2.

Step 3: Initialize initial point of region1 as

x=0 , $y = r_y$

Step 4: Calculate $P= r_y^2 + r_x^2/4 - r_y r_x^2$

dx = 2 r_y^2 x

Let, f ellipse (x, y) = x2 ry2 + y2 rx2 - rx2 ry2

Algorithm:

Step 1: Start

Step 2: Declare r_x , r_y, x , y , m , dx , dy , P , P2.

Step 3: Initialize initial point of region1 as

x=0 , y = r_y

Step 4: Calculate P= $r_y^2 + r_x^2 / 4 - r_y r_x^2$

dx = $2 r_y^2 x$

dy = $2 r_x^2 y$

Step 5: Update values of dx and dy after each iteration.

Step 6: Repeat steps while (dx < dy):

Plot (x,y)

if(P < 0)

Update x = x+1 ;

P += r_y^2[2x + 3]

Else

Update x = x + 1

y= y - 1

Step 7: When dx ≥ dy, plot region 2:

Step 8: Calculate P2 = $r_y^2 (x+1 / 2)^2 + r_x^2 (y -1)^2 - r_x^2 r_y^2$

Step 9: Repeat till (y > 0)

If (P2 > 0)

Update y = y-1 (x will remain same)

P2 =P2 -2 y $r_x^2 + r_x^2$

else

x = x+1

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

y = y-1

P2=P2+ 2 r_y^2[2x] -2 y $r_x^2 + r_x^2$

Step 10: End

y = y-1

P2=P2+ 2 r_y^2 [2x] -2 y r_x^2 + r_x^2

Step 10: End

Program:

```
#include<stdio.h>
#include<graphics.h>
int main()
{
        long x,y,x_center,y_center;
        long a_sqr,b_sqr,fx,fy,d,a,b,tmp1,tmp2;
        int g_driver=DETECT,g_mode;

        initgraph(&g_driver,&g_mode,"");
        printf("*MID POINT ELLIPSE*");
        printf("\n Enter coordinate x = ");
        scanf("%ld",&x_center);
        printf(" Enter coordinate y = ");
        scanf("%ld",&y_center);
        printf("\n Now Enter constants a =");
        scanf("%ld",&a,&b);
        printf(" Now Enter constants b =");
        scanf("%ld",&b);
        x=0;
        y=b;
        a_sqr=a*a;
```

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

```
        b_sqr=b*b;
        fx=2*b_sqr*x;
        fy=2*a_sqr*y;
        d=b_sqr-(a_sqr*b) + (a_sqr*0.25);
        do
```

Experiment No. 8

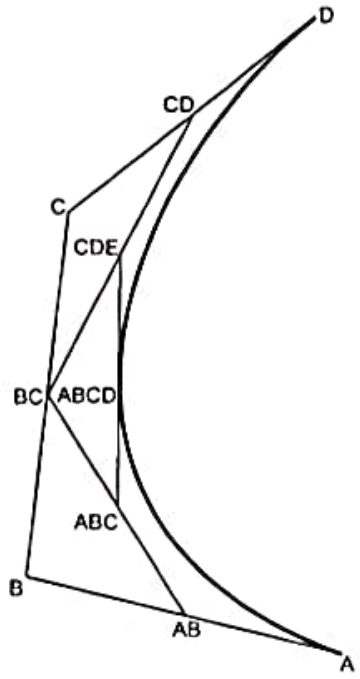
Aim: To implement Bezier curve for n control points. (Midpoint approach)

Objective:

Draw a Bezier curves and surfaces written in Bernstein basis form. The goal of interpolation is to create a smooth curve that passes through an ordered group of points. When used in this fashion, these points are called the control points.

Theory:

In midpoint approach Bezier curve can be constructed simply by taking the midpoints. In this approach midpoints of the line connecting four control points (A, B, C, D) are determined (AB, BC, CD, DA). These midpoints are connected by line segment and their midpoints are ABC and BCD are determined. Finally, these midpoints are connected by line segments and its midpoint ABCD is determined as shown in the figure –



The point ABCD on the Bezier curve divides the original curve in two sections. The original curve gets divided in four different curves. This process can be repeated to split the curve into smaller sections until we have sections so short that they can be replaced by straight lines.

Algorithm:

1) Get four control points say A(xa, ya), B(xb, yb), C(xc, yc), D(xd, yd).

2) Divide the curve represented by points A, B, C, and D in two sections.

xab = (xa + xb) / 2

yab = (ya + yb) / 2

xbc = (xb + xc) / 2

ybc = (yb + yc) / 2

xcd = (xc + xd) / 2

ycd = (yc + yd) / 2

xabc = (xab + xbc) / 2

yabc = (yab + ybc) / 2

xbcd = (xbc + xcd) / 2

ybcd = (ybc + ycd) / 2

xabcd = (xabc + xbcd) / 2

yabcd = (yabc + ybcd) / 2

3) Repeat the step 2 for section A, AB, ABC, ABCD and section ABCD, BCD, CD, D.

4) Repeat step 3 until we have sections so that they can be replaced by straight lines.

5) Repeat small sections by straight lines.

6) Stop.

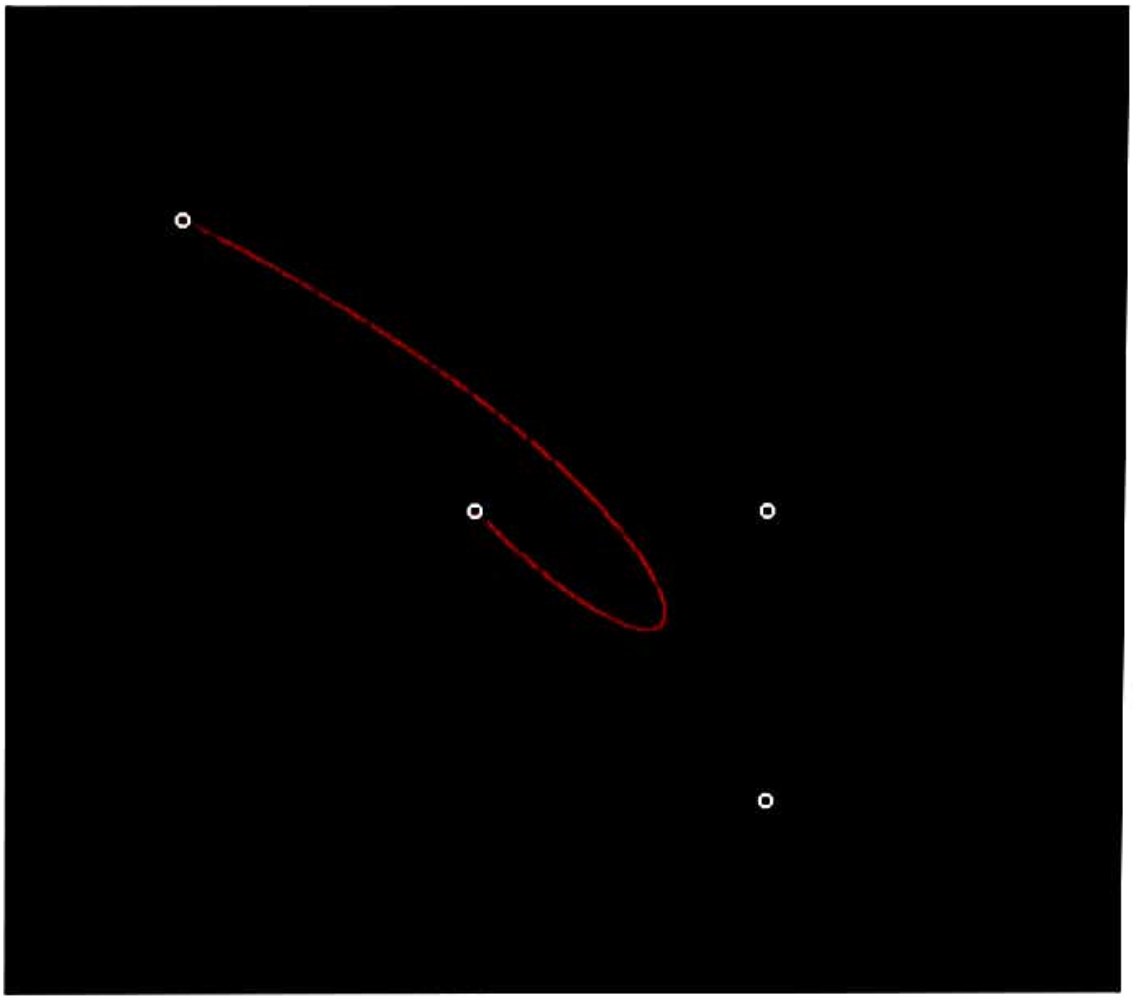
Program:

```
#include<graphics.h>
#include<math.h>
int x[4],y[4];
void bezier(int x[4],int y[4])
{
```

```
int gd=DETECT,gm,i;

double t,xt,yt;
initgraph(&gd,&gm," ");
for(t=0.0;t<1.0;t+=0.0005)
{
xt=pow((1.0-t),3)*x[0]+3*t*pow((1.0-t),2)*x[1]+3*pow(t,2)*(1.0-t)*x[2]+pow(t,3)*x[3];
yt=pow((1.0)-t,3)*y[0]+3*t*pow((1.0)-t,2)*y[1]+3*pow(t,2)*(1.0-t)*y[2]+pow(t,3)*y[3];
putpixel(xt,yt,4);
delay(5);
}
for(i=0;i<4;i++)
{
putpixel(x[i],y[i],5);
circle(x[i],y[i],2);
delay(2);
}
getch();
closegraph();
}
int main()
{
int i,x[4],y[4];
printf("Enter the four control points : ");
for(i=0;i<4;i++)
{
scanf("%d %d",&x[i],&y[i]);
}
```

```
bezier(x,y);
}
```

Output:



Conclusion – Comment on

1. Difference from arc and line
2. Importance of control point
3. Applications

Experiment No. 9

Aim: To implement Character Generation: Bit Map Method

Objective:
Identify the different Methods for Character Generation and generate the character using Stroke

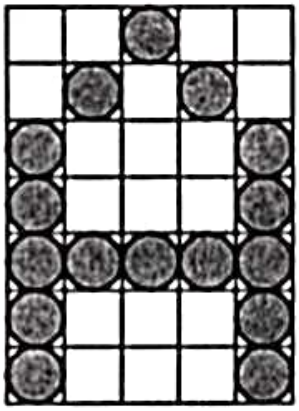
Theory:

Bit map method –

Bitmap method is a called dot-matrix method as the name suggests this method use array of bits for generating a character. These dots are the points for array whose size is fixed.

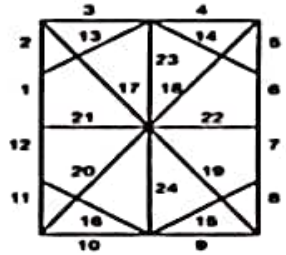
- In bit matrix method when the dots are stored in the form of array the value 1 in array represent the characters i.e. where the dots appear we represent that position with numerical value 1 and the value where dots are not present is represented by 0 in array.
- It is also called dot matrix because in this method characters are represented by an array of dots in the matrix form. It is a two-dimensional array having columns and rows.

A 5x7 array is commonly used to represent characters. However, 7x9 and 9x13 arrays are also used. Higher resolution devices such as inkjet printer or laser printer may use character arrays that are over 100x100.

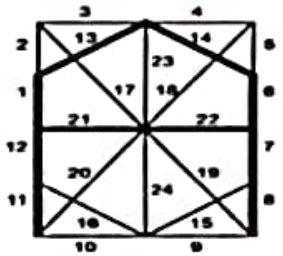


Starburst method –

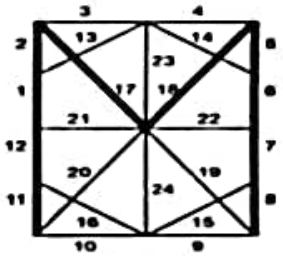
In this method a fix pattern of line segments is used to generate characters. Out of these 24-line segments, segments required to display for particular character are highlighted. This method of character generation is called starburst method because of its characteristic appearance. The starburst patterns for characters A and M. the patterns for particular characters are stored in the form of 24 bit code, each bit representing one line segment. The bit is set to one to highlight the line segment; otherwise, it is set to zero. For example, 24-bit code for Character A is 0011 0000 0011 1100 1110 0001 and for character M is 0000 0011 0000 1100 1111 0011.



a) Star burst pattern of 24 line segments



b) Star burst pattern for character A



c) Star burst pattern for character M

Program:

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>

int main()

{

   int i,j,k,x,y;

   int gd=DETECT,gm;//DETECT is macro defined in graphics.h


   int ch1[][10]={ {1,1,1,1,1,1,1,1,1,1},

                   {1,1,1,1,1,1,1,1,1,1},

                   {0,0,0,0,1,1,0,0,0,0},

                   {0,0,0,0,1,1,0,0,0,0},

                   {0,0,0,0,1,1,0,0,0,0},

                   {0,0,0,0,1,1,0,0,0,0},

                   {0,0,0,0,1,1,0,0,0,0},

                   {0,1,1,0,1,1,0,0,0,0},

                   {0,1,1,0,1,1,0,0,0,0},

                   {0,0,1,1,1,0,0,0,0,0}};

   int ch2[][10]={ {0,0,0,1,1,1,1,0,0,0},

                   {0,0,1,1,1,1,1,1,0,0},

                   {1,1,0,0,0,0,0,0,1,1},

                   {1,1,0,0,0,0,0,0,1,1},
```

```
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {0,0,1,1,1,1,1,1,0,0},
            {0,0,0,1,1,1,1,0,0,0}};
int ch3[][10]={ {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,1,1,1,1,1,1,1,1},
            {1,1,1,1,1,1,1,1,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1}};
int ch4[][10]={ {1,1,0,0,0,0,0,0,1,1},
            {1,1,1,1,0,0,0,0,1,1},
            {1,1,0,1,1,0,0,0,1,1},
            {1,1,0,1,1,0,0,0,1,1},
            {1,1,0,0,1,1,0,0,1,1},
            {1,1,0,0,1,1,0,0,1,1},
            {1,1,0,0,0,1,1,0,1,1},
            {1,1,0,0,0,1,1,0,1,1},
            {1,1,0,0,0,0,1,1,1,1},
            {1,1,0,0,0,0,0,0,1,1}};
initgraph(&gd,&gm," ");//initialize graphic mode
setbkcolor(LIGHTGRAY);//set color of background to darkgray
```

```
for(k=0;k<4;k++)
{
  for(i=0;i<10;i++)
  {
    for(j=0;j<10;j++)
    {
      if(k==0)
      {
        if(ch1[i][j]==1)
        putpixel(j+250,i+230,RED);
      }
      if(k==1)
      {
        if(ch2[i][j]==1)
        putpixel(j+300,i+230,RED);
      }
      if(k==2)
      {
        if(ch3[i][j]==1)
        putpixel(j+350,i+230,RED);
      }
      if(k==3)
      {
        if(ch4[i][j]==1)
        putpixel(j+400,i+230,RED);
      }
    }
    delay(200);
```

```
    }
  }
  getch();
  closegraph();
}
```

Output -

Conclusion: Comment on

1. different methods
2. advantage of stroke method
3. one limitation

Experiment No. 10

Aim: To develop programs for making animations such as

Objective:
Draw an object and apply various transformation techniques to this object. Translation, scaling and rotation is applied to object to perform animation.

Theory:

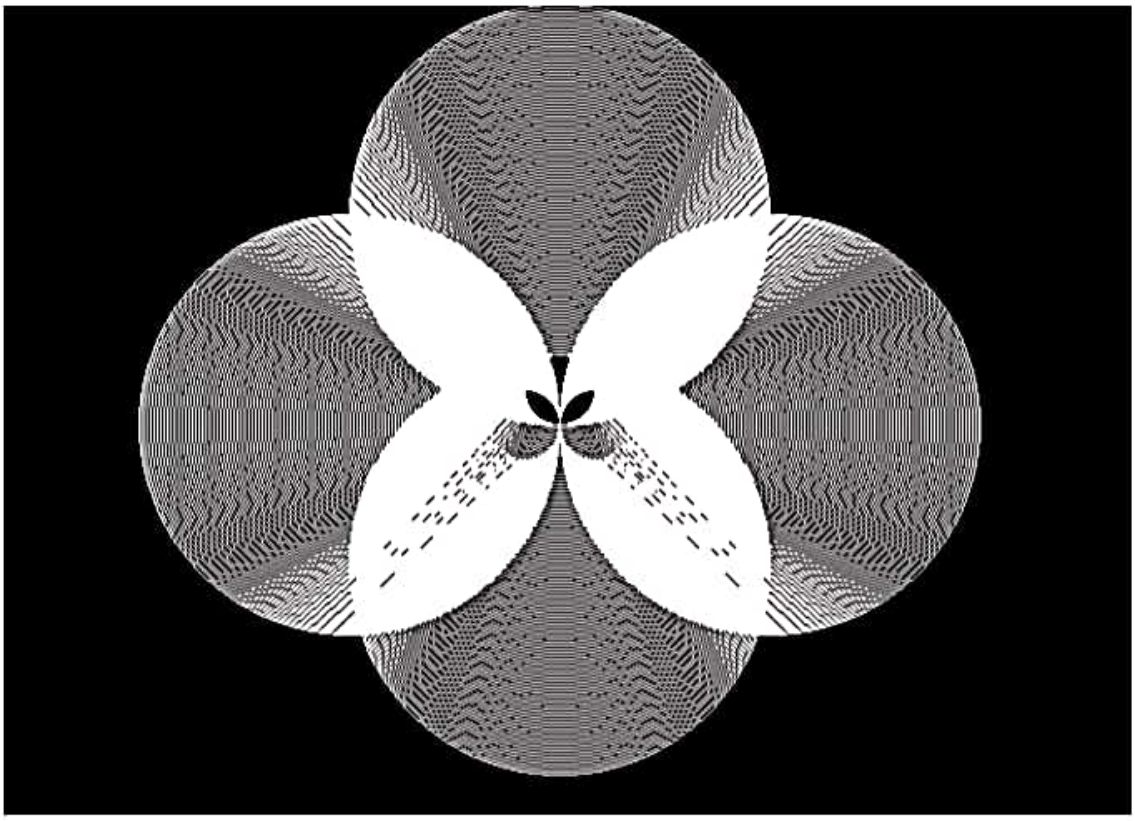
- For moving any object, we incrementally calculate the object coordinates and redraw the picture to give a feel of animation by using for loop.
- Suppose if we want to move a circle from left to right means, we have to shift the position of circle along x-direction continuously in regular intervals.
- The below programs illustrate the movement of objects by using for loop and also using transformations like rotation, translation etc.
- For windmill rotation, we use 2D rotation concept and formulas.

Program:

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <dos.h>

void main()

{

clrscr

{

int gd=DETECT,gm,i;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

for(i=0;i<=100;i++)

{

circle(319,219-i,20+i);
```

```
circle(319,219+i,20+i);
circle(299-i,239,20+i);
circle(339+i,239,20+i);
delay(100);
}
getch();
}
}
```

Output:



Conclusion - Comment on :

1. Importance of story building
2. Defining the basic character of story

3. Apply techniques to these characters