Assignment No: 02

Title/ Problem Statement:

Write C++/Java program to draw circle using Bresenham‘s algorithm. Inherit pixel class.

Prerequisites:

OBJECTIVE:

1. To understand Bresenham‘s circle drawing algorithms used for computer graphics.

THEORY:

In computer graphics, the midpoint circle algorithm is an algorithm used to determine the points needed for drawing a circle. The algorithm is a variant of Bresenham's line algorithm, and is thus sometimes known as Bresenham's circle algorithm, although not actually invented by Jack E. Bresenham. The Midpoint circle drawing algorithm works on the same midpoint concept as the Bresenham’s line algorithm. In the Midpoint circle drawing algorithm, you determine the next pixel to be plotted based on the position of the midpoint between the current and next consecutive pixel.

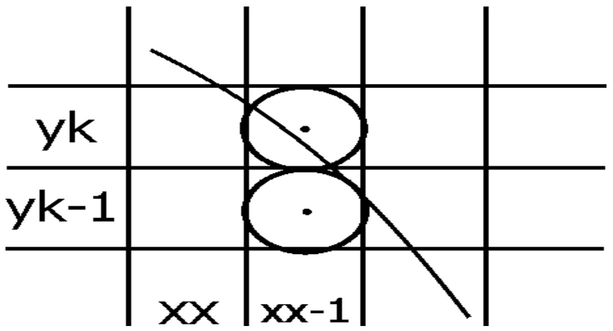

Fig 6.1 Midpoint circle Algorithm

PLATFORM REQUIRED:

- Microsoft Windows 7/ Windows 8 Operating System onwards or 64-bit Open source Linux or its derivative.

Algorithm :

BRESENHAM’S ALGORITHM TO DRAW A CIRCLE.

1. Start.
2. Declare variables x,y,p and also declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.

4. Input the radius of the circle r.
5. Load x-0,y=r,initial decision parameter p=1-r.so the first point is (0,r).
6. Repeat Step 7 while (x<y) and increment x-value simultaneously.
7. If (p>0),

   do p=p+2*(x-y)+1.
8. Otherwise
   p=p+2*x+1 and
   y is decremented simultaneously.
9. Then calculate the value of the function circlepoints() with p.arameters (x,y).
10. Place pixels using putpixel at points (x+300,y+300) in specified colour in

    circlepoints() function shifting the origin to 300 on both x-axis and y-axis.
11. Close Graph.
12. Stop.

BRESENHAM'S CIRCLE ALGORITHM
Bresenham Circle (X_c, Y_c, R):

Description: Here $\mathbf{X_c}$ and $\mathbf{Y_c}$ denote the x – coordinate and y – coordinate of the center of the circle. **R** is the radius.

1. Input radius r and circle center (x_c,y_c) , and obtain the first point on the circumference of the circle centered on the origin as (x_0,y_0) = (0,r)

2. Caculate the initial value of the decision parameter as

   $P_0 = 3 – 2R$
3. If $p_k < 0$, the next point along the circle is ($x_k + 1$,y_k) and

   P = P + 4X + 6

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

   P = P + 4(X – Y) + 10
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x,y) onto the circular path centered on (x_c,y_c) and plot the coordinate values :

   $x = x + x_c$

   $y = y + y_c$

6. Repeat steps 3 through 5 while x<y.

Draw Symmetry points (X_c, Y_c, X, Y):

1. Call PutPixel($X_c + X$, Y_c, $+ Y$)
2. Call PutPixel($X_c - X$, Y_c, $+ Y$)
3. Call PutPixel($X_c + X$, Y_c, $- Y$)
4. Call PutPixel($X_c - X$, Y_c, $- Y$)
5. Call PutPixel($X_c + Y$, Y_c, $+ X$)
6. Call PutPixel($X_c - Y$, Y_c, $+ X$)
7. Call PutPixel($X_c + Y$, Y_c, $- X$)
8. Call PutPixel($X_c - Y$, Y_c, $- X$)

WAP TO DRAW A CIRCLE USING BRESENHAM'S ALGORITHM.

```
#include<iostream>
#include<graphics.h>
using namespace std;

class Pixel
{
  public:
      void plotPixel(int x,int y)
      {
            putpixel(x,y,WHITE);
      }
};


class MyCircle : public Pixel
{
public:
      void drawCircle(int xc,int yc,int r)
      {
            int x,y;
            float s;
            x = 0;
            y = r;
            s = 3-2*r;
            while(x < y)
            {
                  if(s<=0)
                  {
                        s = s + 4 * x + 6;
                        x = x + 1;
```

```
                }
                else
                {
                        s = s + 4 * (x-y) + 10;
                        x = x + 1;
                        y = y - 1;
                }
                display(xc,yc,x,y);
                delay(100);
        }

    }
    void display(int xc,int yc,int x,int y)
    {
        plotPixel(xc+x,yc+y);
        plotPixel(xc+y,yc+x);
        plotPixel(xc+y,yc-x);
        plotPixel(xc+x,yc-y);
        plotPixel(xc-x,yc-y);
        plotPixel(xc-y,yc-x);
        plotPixel(xc-y,yc+x);
        plotPixel(xc-x,yc+y);
    }
};

int main()
{
    int gd = DETECT, gm,xc,yc,r;
    initgraph(&gd,&gm,NULL);
    MyCircle c;
    cout<<"Enter center coordinates and radius";
    cin>>xc>>yc>>r;
    c.drawCircle(xc,yc,r);
      getch();
      closegraph();
    return 0;

}
```

Input : Coordiates of Center and radius of Circle

Output : Circle plotted

Conclusion: Thus we have implemented program to draw circle using Bresenham's circle generation algorithms.