**Aim:** To implement DDA algorithms for drawing a line segment between two given end points.

**Objective:** Draw the line using (vector) generation algorithms which determine the pixels that should be turned ON are called as digital differential analyzer (DDA).It is one of the techniques for obtaining a rasterized straight line. This algorithm can be used to draw the line in all the quadrants.

**Theory:**

DDA algorithm is an incremental scan conversion method. Here we perform calculations at each step using the results from the preceding step. The characteristic of the DDA algorithm is to take unit steps along one coordinate and compute the corresponding values along the other coordinate. Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.

**Algorithm:**

start

(x1,y1,x2,y2)

dx=x2-x1

dy=y2-y1

x=x1

y=y1

m=dy/dx

if abs(m)<1 then

num\_of\_pixels=abs(dx)

else

num\_of\_pixels=abs(dy)

end

xi=dx/num\_of\_pixels

yi=dy/num\_of\_pixels

putpixel(x,y)

for x=x1 to x2 do

x=x+xi

y=y+yi

end

**Program:**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

void main() {

int gd = DETECT, gm;

int dx, dy, steps;

float xinc, yinc, x, y;

int x1, y1, x2, y2;

printf("Enter the coordinates of the first point (x1, y1):\n");

scanf("%d %d", &x1, &y1);

printf("Enter the coordinates of the second point (x2, y2):\n");

scanf("%d %d", &x2, &y2);

dx = x2 - x1;

dy = y2 - y1;

if (abs(dx) > abs(dy))

steps = abs(dx);

else

steps = abs(dy);

xinc = dx / (float) steps;

yinc = dy / (float) steps;

x = x1;

y = y1;

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

for (int k = 1; k <= steps; k++) {

putpixel(round(x), round(y), RED);

x = x + xinc;

y = y + yinc;

}

getch();

closegraph();

}

**Output:**

****

**Conclusion:** Comment on -

1. Pixel-operation on floating point
2. Equation for line-y=mx+c
3. Need of line drawing algorithm-no special requirements
4. Slow or fast-faster

**Aim:** To implement Bresenham’s algorithms for drawing a line segment between two given end points.

**Objective:**

Draw a line using Bresenham's line algorithm that determines the points of an n-dimensional raster that should be selected to form a close approximation to a straight line between two points

**Theory:**

In Bresenham’s line algorithm pixel positions along the line path are obtained by determining the pixels i.e. nearer the line path at each step.

**Algorithm -**

(x1,y1,x0,y0)

dx=x1-x0

dy=y1-y0

p0=2dy-dx

for k=0 to dx do

if pk<0 then

putpixel(xi+1,yi)

pn=pk+2dy

else

putpixel(xi+1,yi+1)

pn=pk+(2dy-2dx)

end

**Program -**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

void Bresenham(int x1, int y1, int x2, int y2) {

int dx, dy, x, y, p, end;

dx = abs(x1 - x2);

dy = abs(y1 - y2);

p = 2 \* dy - dx;

if (x1 > x2) {

x = x2;

y = y2;

end = x1;

}

else {

x = x1;

y = y1;

end = x2;

}

putpixel(x, y, RED);

while (x < end) {

x = x + 1;

if (p < 0) {

p = p + 2 \* dy;

} else {

y = y + 1;

p = p + 2 \* (dy - dx);

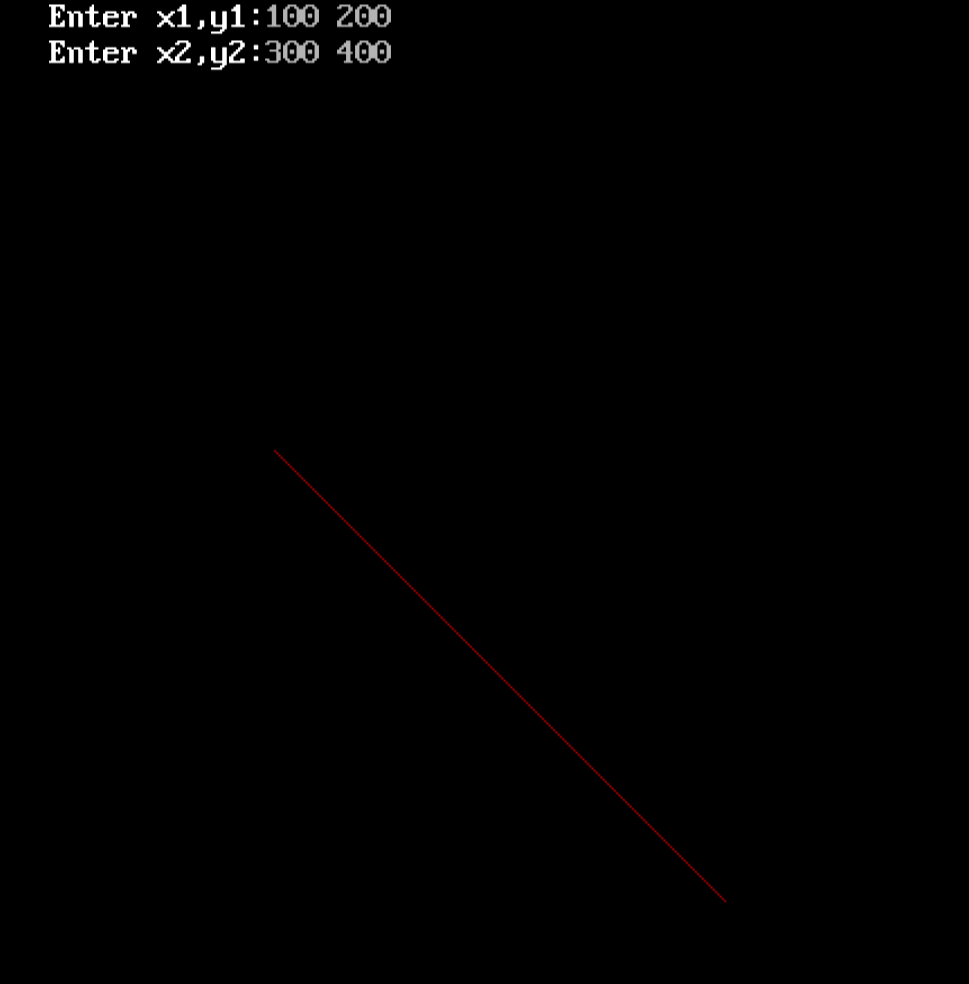
}

putpixel(x, y, RED);

}

}

**Output –**

****

**Conclusion:** Comment on -

1. Pixel-Bresenham’s algorithm does not perform any rounding operation
2. Equation for line-y=mx+c
3. Need of line drawing algorithm-Involves cheaper operation like addition and subtraction
4. Slow or fast-FASTER THAN DDA

**Aim**: To implement midpoint circle algorithm.

**Objective:**

Draw a circle using mid-point circle drawing algorithm by determining the points needed for

rasterizing a circle. The mid-point algorithm to calculate all the perimeter points of the circle

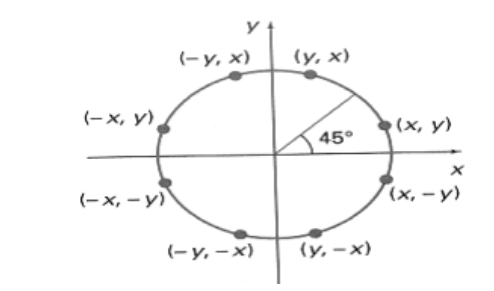
in the first octant and then print them along with their mirror points in the other octants.

**Theory:**

The shape of the circle is similar in each quadrant. We can generate the points in one section

and the points in other sections can be obtained by considering the symmetry about x-axis

and y-axis.



The equation of circle with center at origin is x 2 + y 2 = r 2

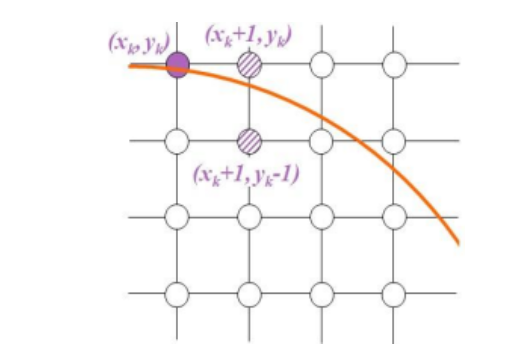
Let the circle function is f circle (x, y) -

 is &lt; 0, if (x, y) is inside circle boundary,

 is = 0, if (x, y) is on circle boundary,

 is &gt; 0, if (x, y) is outside circle boundary.

Consider the pixel at (xk, yk) is plotted,



Now the next pixel along the circumference of the circle will be either (xk + 1, yk) or (xk + 1,

yk – 1) whichever is closer the circle boundary.

Let the decision parameter pk is equal to the circle function evaluate at the mid-point between

two pixels.

If pk &lt; 0, the midpoint is inside the circle and the pixel at yk is closer to the circle boundary.

Otherwise, the midpoint is outside or on the circle boundary and the pixel at yk – 1 is closer

to the circle boundary.

**Algorithm –**

1.enter center co-ordinates (xc,yc) and radius r.

2.initialise center x=0,y=0.

3.calculate initial value of decision parameter p=1-r

4.Check if p is less than 0. If it is update p using p = p + (2 \* x) + 3.

5.Check if p is less than 0. If it is update p using p = p + (2 \* x) + 3.

6.Repeat steps 7 to 10 until y is greater than or equal to x.

7.. Exit the do-while loop.

**Program –**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main(){

int x,y, r,p,xc, yc ;

int gd=DETECT , gm ;

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

printf("Enter xc, yc:");

scanf("%d %d",&xc, &yc);

printf("Enter r :");

scanf("%d", &r);

x=0;

y=r;

p=1-r;

do{

putpixel(xc+x,yc+y,RED);

putpixel(xc+y,yc+x,RED);

putpixel(xc+x,yc-y,RED);

putpixel(xc+y, yc-x,RED);

putpixel(xc-y, yc-x,RED);

putpixel(xc-x, yc-y,RED);

putpixel(xc-x, yc+y,RED);

putpixel(xc-y, yc+x,RED);

if(p <0){

p = p + (2\*x)+3;

}

else

{

y= y-1;

p=p + (2\*x) - (2\*y) + 5;

}

x = x + 1 ;

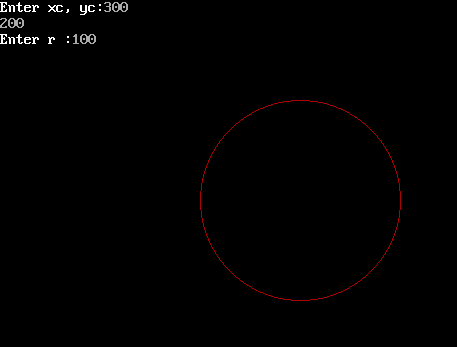
}while(y>=x);

getch();

closegraph();

}

**output –**

****

**Conclusion: Comment on**

**1. Fast or slow-**Midpoint theorem to draw a circle is faster.

**2. Draw one arc only and repeat the process in 8 quadrants-**it draws only one of the 8 octants and then copies it using difference in sign of the co-ordinates.

**3. Difference with line drawing method–**unlike line drawing method using bresonhams and DDA it directly draws a circle and then mirrors it.