Experiment No. 2

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 –
Step1: Start Algorithm

Step2: Declare variable x1,x2,y1,y2,d,i1,i2,dx,dy

Step3: Enter value of x1,y1,x2,y2
    Where x1,y1are coordinates of starting point
    And x2,y2 are coordinates of Ending point

Step4: Calculate dx = x2-x1
    Calculate dy = y2-y1
    Calculate i1=2*dy
    Calculate i2=2*(dy-dx)
    Calculate d=i1-dx

Step5: Consider (x, y) as starting point and xendas maximum possible value of x.
    If dx < 0
      Then x = x2
      y = y2
       xend=x1
    If dx > 0
      Then x = x1
    y = y1
      xend=x2

Step6: Generate point at (x,y)coordinates.

Step7: Check if whole line is generated.

If x > = xend
Stop.

Step8: Calculate co-ordinates of the next pixel
If d < 0
Then d = d + i1
If d ≥ 0
Then d = d + i2
Increment y = y + 1

Step9: Increment x = x + 1

Step10: Draw a point of latest (x, y) coordinates

Step11: Go to step 7

Step12: End of Algorithm

Program –

```c
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
int main()
{
    int x,y,x1,y1,x2,y2,p,dx,dy;
    int gd=DETECT,gm=0;
    initgraph(&gd,&gm, "");
    printf("\n Enter x1 cordinate: ");
    scanf("%d",&x1);
    printf("\n Enter y1 cordinate: ");
    scanf("%d",&y1);
    printf("\n Enter x2 cordinate: ");
    scanf("%d",&x2);
```
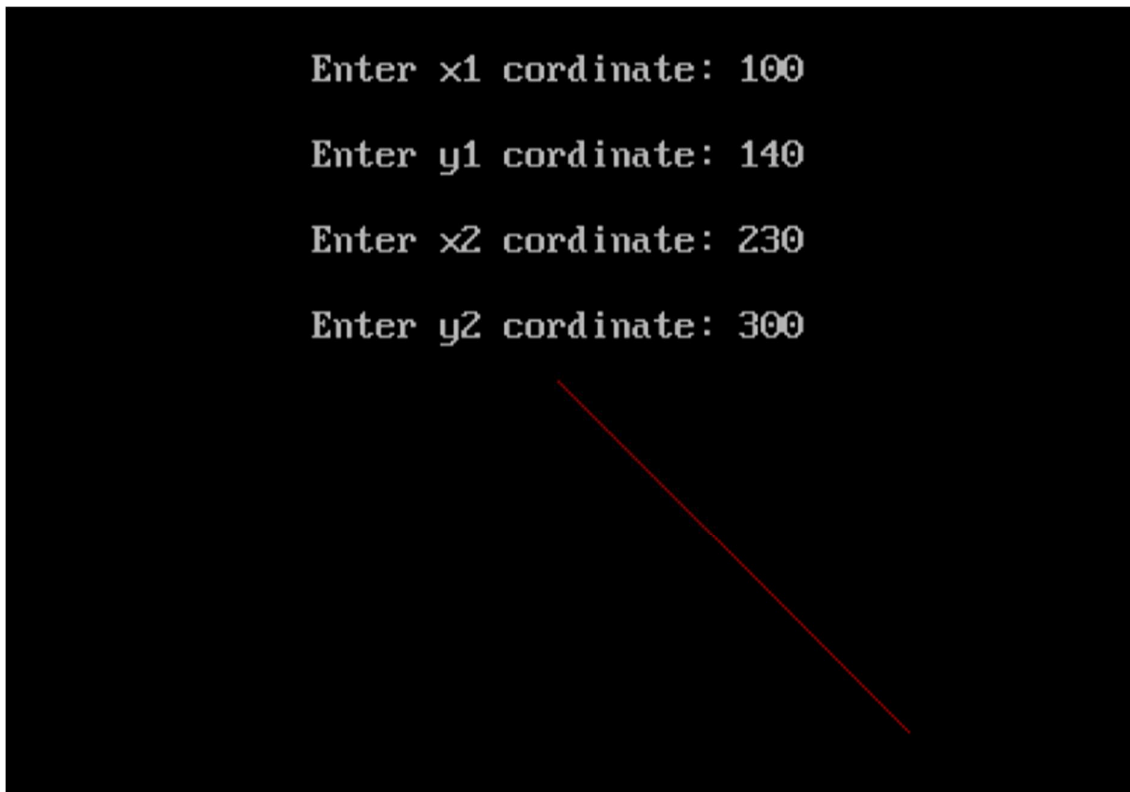
```c
printf("\n Enter y2 cordinate: ");

scanf("%d",&y2);


x=x1;

y=y1;

dx=x2-x1;

dy=y2-y1;


putpixel (x,y, RED);

p = (2 * dy-dx);


while(x <= x2)

{

        if(p<0)

        {

                x = x+1;

                p = p + 2*dy;

        }

        else

        {

                x = x + 1;

                y = y + 1;

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


        }

        putpixel (x,y, RED);


}
```

```
    getch();

    closegraph();

}
```

Output –

```
Enter x1 cordinate: 100

Enter y1 cordinate: 140

Enter x2 cordinate: 230

Enter y2 cordinate: 300
```

Conclusion: Comment on -
1. Pixel
2. Equation for line
3. Need of line drawing algorithm
4. Slow or fast