Experiment No. 2
Implement Bresenham's Line Drawing algorithm.
Name: Jaffari Mohammed Ali Sayyed Naqi Ali
Roll Number: 16
Date of Performance:
Date of Submission:

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 -

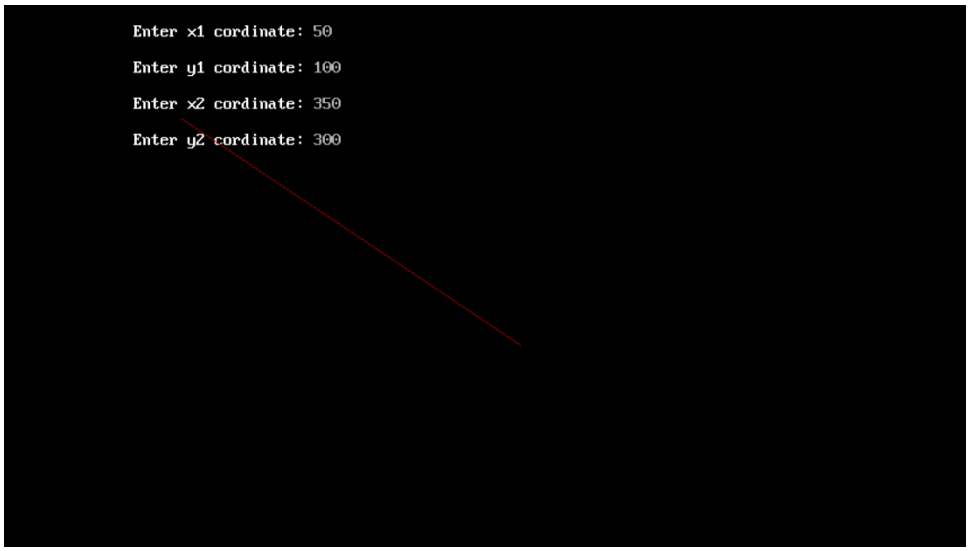
1. Input two endpoints: (x1, y1) and (x2, y2).
2. Calculate the differences in the x and y coordinates:
3. dx = x2 - x1 dy = y2 - y1
4. Initialize variables for tracking the current position, decision parameter, and steps:
5. x = x1 y = y1 d = 2 * dy - dx x_increment = 1 y_increment = 1
6. If dx < 0, set x_increment to -1.
7. If dy < 0, set y_increment to -1.
8. Start a loop that runs from 1 to dx (or -dx if dx is negative):
9. a. Plot the pixel at the current position (x, y).
10. b. If the decision parameter is greater than or equal to 0, increment y by y_increment and update the decision parameter:
11. if d >= 0: y = y + y_increment d = d - 2 * dx
12. c. Increment x by x_increment.
13. d. Update the decision parameter:
14. d = d + 2 * dy
15. Repeat the loop until you have plotted all the necessary pixels to draw the line segment.

Program -

```
#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);
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 –



Conclusion: Comment on -

1. **Pixel Handling** - The "pixel" is managed through the use of the `putpixel` function, which allows you to specify the color of individual screen pixels.
2. **Equation for Drawing Lines** - To sketch a line, an algorithm is employed that calculates and utilizes the differences in x and y coordinates (dx and dy) to decide which pixels should be colored, thus approximating the line's path.
3. **Importance of Line Drawing Algorithms** - Line drawing algorithms are essential due to the digital screen's inherent discrete nature, where images are represented using pixels arranged on a grid. To render a continuous-looking line on such a grid, a method like Bresenham's is necessary to determine which pixels should be filled in order to create the illusion of a smooth line.
4. **Speed and efficiency** - Bresenham's algorithm is notably swift and resource-efficient, particularly when drawing lines with integer coordinates. It relies on integer arithmetic and eliminates the need for time-consuming floating-point calculations.