Name:	TANVI MANGESH SURVE
Roll No:	
Class/Sem:	DSE/III
Experiment No.:	2
Title:	Bresenham's Line Drawing Algorithm
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Experiment No. 2

Aim :Write a program to implement Bresenham Line Drawing Algorithm in C.

Objective :To implement Bresenham line drawing algorithm for drawing a line segment between two points A (x1, y1) and B (x2, y2)

Theory :

Bresenham Algorithm

This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

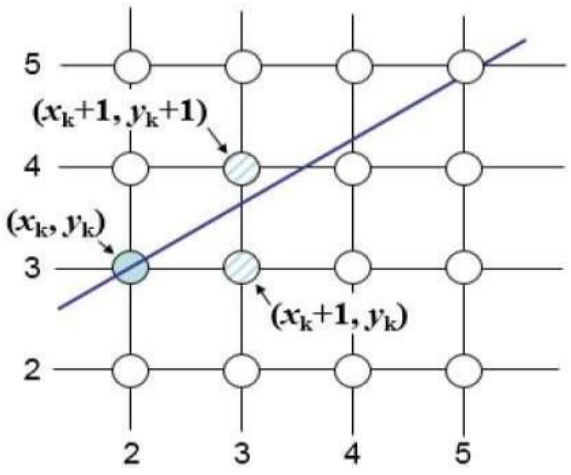
In this method, next pixel selected is that one who has the least distance from true line

Basic Concept:

Move across the x axis in unit intervals and at each step choose between two different y coordinates.

For example, from position (2, 3) we have to choose between (3, 3) and (3, 4). We would like the point that is closer to the original line.

So we have to take decision to choose next point. So next pixels are selected based on the value of decision parameter p. The equations are given in below algorithm



Algorithm :

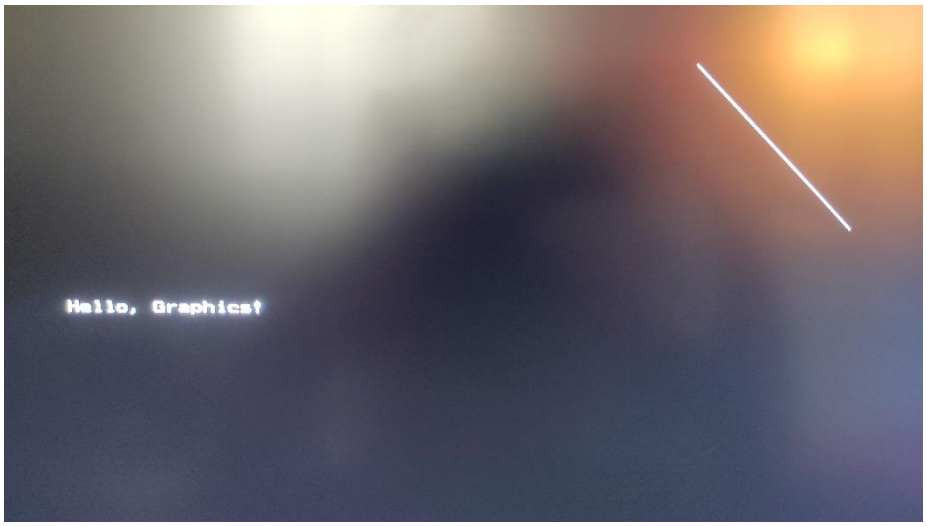
Step1: StartAlgorithm
Step2: Declare variable x1,x2,y1,y2,d,i1,i2,dx,dy
Step3: Enter value of x1,y1,x2,y2
Where x1,y1are coordinates of starting point And x2,y2 are coordinates of Ending point
Step4: Calculate dx = x2-x1 Calculate dy = y2-y1 Calculate i1=2*dy Calculate i2=2*(dy-dx) Calculate d=i1-dx
Step5: Consider (x, y) as starting point and xend as 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

Code :

```
#include <stdio.h>
#include <graphics.h>
void drawLineBresenham(int x1, int y1, int x2, int y2) {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int slope_gt_one = 0;
    if (dy > dx) {
        slope_gt_one = 1;
        int temp = x1;
        x1 = y1;
        y1 = temp;
        temp = x2;
        x2 = y2;
        y2 = temp;
        dx = abs(x2 - x1);
```

```
        dy = abs(y2 - y1);
    }
    int p = 2 * dy - dx;
    int twoDy = 2 * dy;
    int twoDyMinusDx = 2 * (dy - dx);
    int x, y, xEnd;
    if (x1 > x2) {
        x = x2;
        y = y2;
        xEnd = x1;
    } else {
        x = x1;
        y = y1;
        xEnd = x2;
    }
    putpixel(x, y, WHITE);
    while (x < xEnd) {
        x++;
        if (p < 0)
            p += twoDy;
        else {
            y++;
            p += twoDyMinusDx;
        }
        if (slope_gt_one)
            putpixel(y, x, WHITE);
        else
            putpixel(x, y, WHITE);
    }
    closegraph();
}
int main() {
    int x1, y1, x2, y2;
    printf("Enter the starting point (x1, y1): ");
    scanf("%d %d", &x1, &y1);
    printf("Enter the ending point (x2, y2): ");
    scanf("%d %d", &x2, &y2);
    drawLineBresenham(x1, y1, x2, y2);
    return 0;
}
```

Output



Conclusion :

The Bresenham Line Drawing Algorithm is a powerful and efficient method for drawing lines on a computer screen using integer arithmetic. It avoids the need for floating-point calculations, making it faster and more suitable for systems with limited computational resources.

One key advantage of the Bresenham algorithm is its ability to make decisions based on integer operations and avoid the overhead associated with floating-point arithmetic. This leads to a faster execution and is particularly beneficial in resource-constrained environments.

The algorithm is also versatile and can be extended to draw lines at any angle efficiently. However, it is specific to drawing lines and doesn't handle other shapes. In contemporary graphics programming, hardware acceleration and more advanced algorithms are often used for efficiency, but the Bresenham algorithm remains a fundamental concept and is widely used in various applications.