

## Lab Report 2

**Title:** To use Bresenham's line algorithm to draw a line between given points.

**Theory:** Bresenham's Line Algorithm is a highly efficient method in computer graphics for drawing lines between two specified points. Unlike the Digital Differential Analyzer (DDA), which employs floating-point arithmetic, Bresenham's algorithm operates solely with integer arithmetic, enhancing its speed and suitability for implementation on devices with limited computational resources. By intelligently selecting which pixels to plot along the line path, Bresenham's algorithm minimizes error and produces accurate line representations, making it a preferred choice for generating straight lines efficiently.

Although primarily suited for drawing straight lines, Bresenham's algorithm may require modifications to handle more complex curves effectively. Nonetheless, its simplicity, computational efficiency, and precision advantages over DDA make it a widely used method in computer graphics applications, particularly those operating under resource constraints.

### Bresenham's Algorithm:

Step 1: Start

Step 2: Take the coordinates of the initial point ( $x_1, y_1$ ) and the final point ( $x_2, y_2$ ).

Step 3: Find the differences between the points by using the formula:

$dx = x_2 - x_1$ ;

$dy = y_2 - y_1$ ;

Step 4: Initialize the decision parameter:

$p = 2 * dy - dx$ ;

Step 5: Begin drawing the line using the following logic:

```
while (x <= x2) {  
    putpixel(x, y, WHITE); // WHITE Line  
    if (p < 0) {  
        p = p + 2 * dy;  
    } else {  
        p = p + (2 * dy) - (2 * dx);  
        y++;  
    }  
    x++;  
}
```

Step 6: Stop

## The code:

The code for the asked program is given below:

//Bresenham's Line Drawing Algorithm

```
#include <iostream>
#include <conio.h>
#include <graphics.h>

class bresenhams_algorithm{
private:
    int x, y, dx, dy;

public:
    void draw_line(int x1, int y1, int x2, int y2)
    {
        x = x1 ;
        y = y1;
        dx = x2 - x1;
        dy = y2 - y1;
        int p = 2 * dy - dx;
        while( x <= x2)
        {
            putpixel(x,y, WHITE); // Plotting the pixel at (x,y) with WHITE color
            if ( p < 0)
            {
                p = p + 2 * dy;
            }
            else{
                p = p + (2 * dy) - (2 * dx);
                y++;
            }
            x++;
        }
    }
};

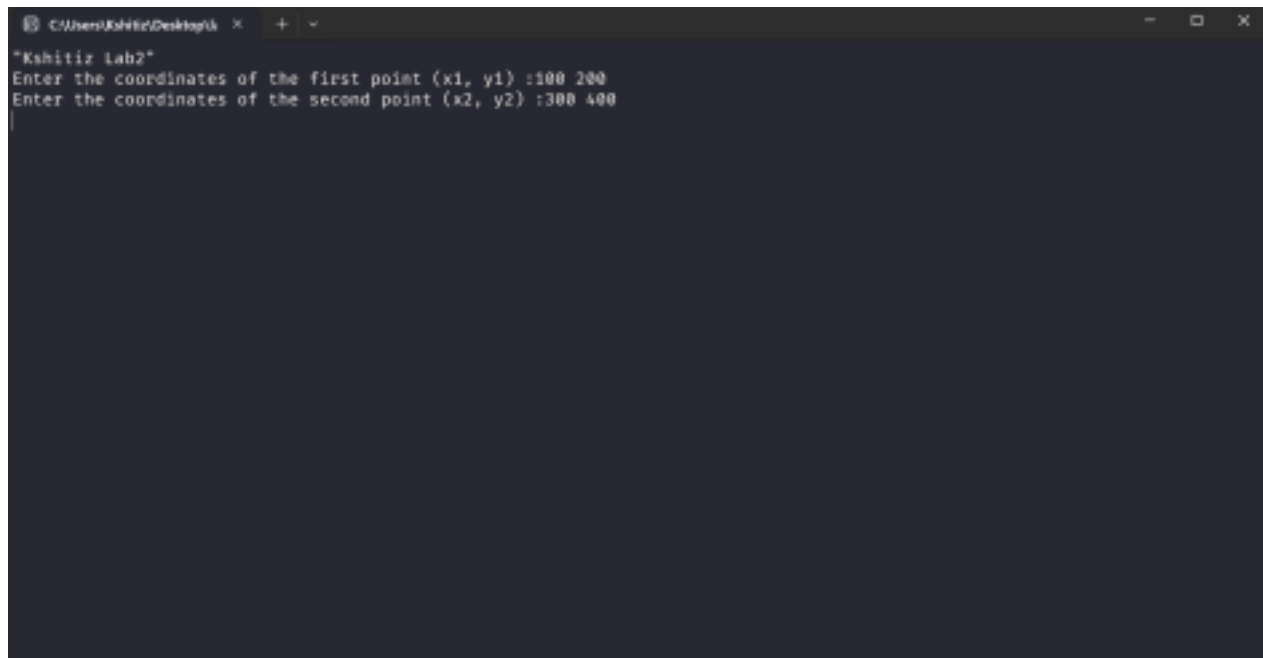
using namespace std;

int main()
{
    int gd = DETECT, gm;
    initwindow(800, 600, "Bresenham's Line Drawing Algorithm"); // Initializing the graphics window
```

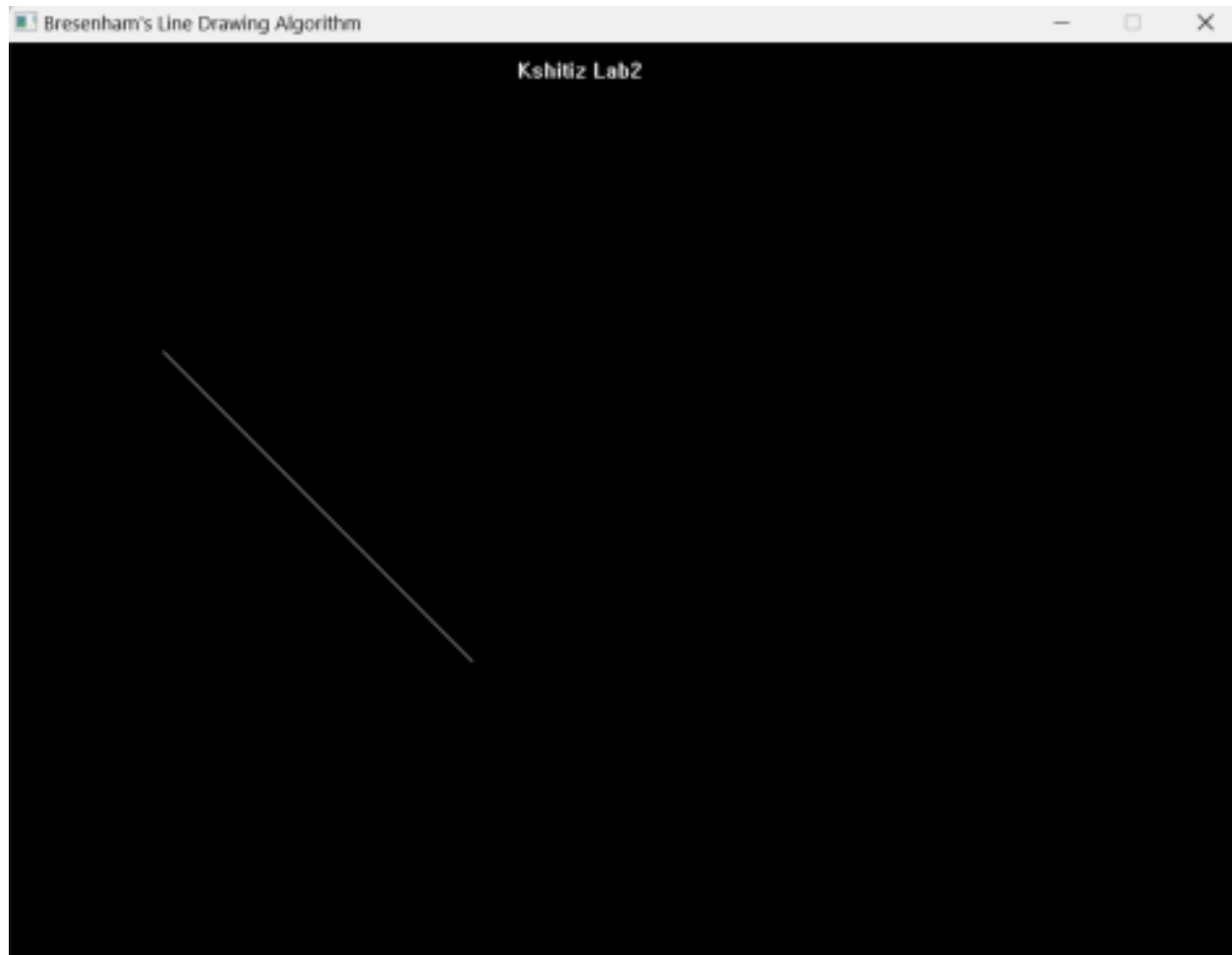
```
outtextxy(330,10, "Kshitiz Lab2"); // Displaying text at (330, 10)
bresenhams_algorithm linedrawing;
int x1, y1, x2, y2;
cout << "\\Kshitiz Lab2\\\\";
cout << "Enter the coordinates of the first point (x1, y1) :";
cin >> x1>> y1;
cout << "Enter the coordinates of the second point (x2, y2) :";
cin >> x2>>y2;
linedrawing.draw_line (x1, y1, x2, y2); // Calling the line drawing function
getch();
closegraph();
return 0;
}
```

## The output:

The output of the asked program is given below:

A screenshot of a terminal window with a dark background. The window title bar shows the file path "C:\Users\Kshitiz\Desktop\k...". The terminal output displays the program's execution: it prints "Kshitiz Lab2", prompts for the first point coordinates, receives "100 200", prompts for the second point coordinates, and receives "300 400".

```
C:\Users\Kshitiz\Desktop\k... x + -
Kshitiz Lab2
Enter the coordinates of the first point (x1, y1) :100 200
Enter the coordinates of the second point (x2, y2) :300 400
```



*Figure : Line drawn using Bresenham's Line algorithm*

## **Conclusion:**

The Bresenham's Line Drawing Algorithm provides an efficient method for drawing lines between specified points. It utilizes an ingenious decision parameter to determine the appropriate pixels to plot along the line. By incorporating simple arithmetic operations, it achieves accurate line rendering while minimizing computational overhead. This algorithm enables the generation of smooth and precise lines, making it suitable for various applications in computer graphics. The Bresenham's Line Drawing Algorithm showcases the effectiveness of optimized algorithms in generating graphical output with minimal computational resources, highlighting its significance in practical implementations within computer graphics.