Lab Report No: 02

Problem Name:

Implementation of Bresenham's Line Drawing Algorithm.

Description:

Bresenham's Line Drawing Algorithm is an efficient algorithm used in computer graphics to draw a straight line between two points using only integer calculations. It eliminates the need for floating-point arithmetic, making it faster and more optimized for raster displays. The algorithm works by iteratively determining the nearest pixel to the ideal line using a decision parameter. By incrementing one coordinate at a time and adjusting the other based on the error term, Bresenham's algorithm ensures smooth and accurate line rendering with minimal computation.

Algorithm:

Step 1:

Calculate ΔX and ΔY

From the given input coordinates:

Starting coordinates: (X0, Y0)

Ending coordinates: (Xn, Yn)

Calculate:

(ΔX = Xn - X0)

(ΔY = Yn - Y0)

Step 2:

Compute the Initial Decision Parameter

The decision parameter (Pk) is calculated as: [Pk = 2*ΔY - ΔX]

Step 3:

Determine the Next Point

For each step, the next point ((X{k+1}, Y{k+1})) is determined based on the value of (Pk):

Case-1:

If (Pk < **0):**

The next point is ((X{k + 1}, Yk))

Update (Pk) as: [P{k+1} = P{k + 2}\ΔY]

X{k+1}=Xk+1

Case-2: >= 0):

The next point is ((X{k + 1}, Y{k + 1})

Update (Pk) as: [P{k+1} = Pk + 2ΔX]

X{k+1}=Xk+1

Step 4:

Repeat until the End Point is reached

Keep repeating Step 3 until the endpoint ((Xn, Yn)) is reached or the number of iterations equals ((ΔX - 1)).

Code:

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

using namespace std;

void drawline(int x0, int y0, int x1, int y1)
{
   int dx, dy, p, x, y;
   dx = abs(x1 - x0);
   dy = abs(y1 - y0);
   x = x0;
   y = y0;
   p = 2 * dy - dx;

   while(x <= x1)  // Ensures it works for all cases
   {
      putpixel(x, y, BLACK);  // Use BLACK instead of WHITE
      if(p >= 0)
      {
         y = y + 1;
```

```
            p = p + 2 * dy - 2 * dx;
        }
        else
        {
            p = p + 2 * dy;
        }
        x = x + 1;
    }
}

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    setbkcolor(WHITE); // Set background color
    cleardevice();  // Clear screen with white background

    int x0 = 200, y0 = 200, x1 = 300, y1 = 400; // Define only once

    drawline(x0, y0, x1, y1);

    delay(5000); // Delay to view the result
    closegraph(); // Close graphics mode

    return 0;
}
```

Output:

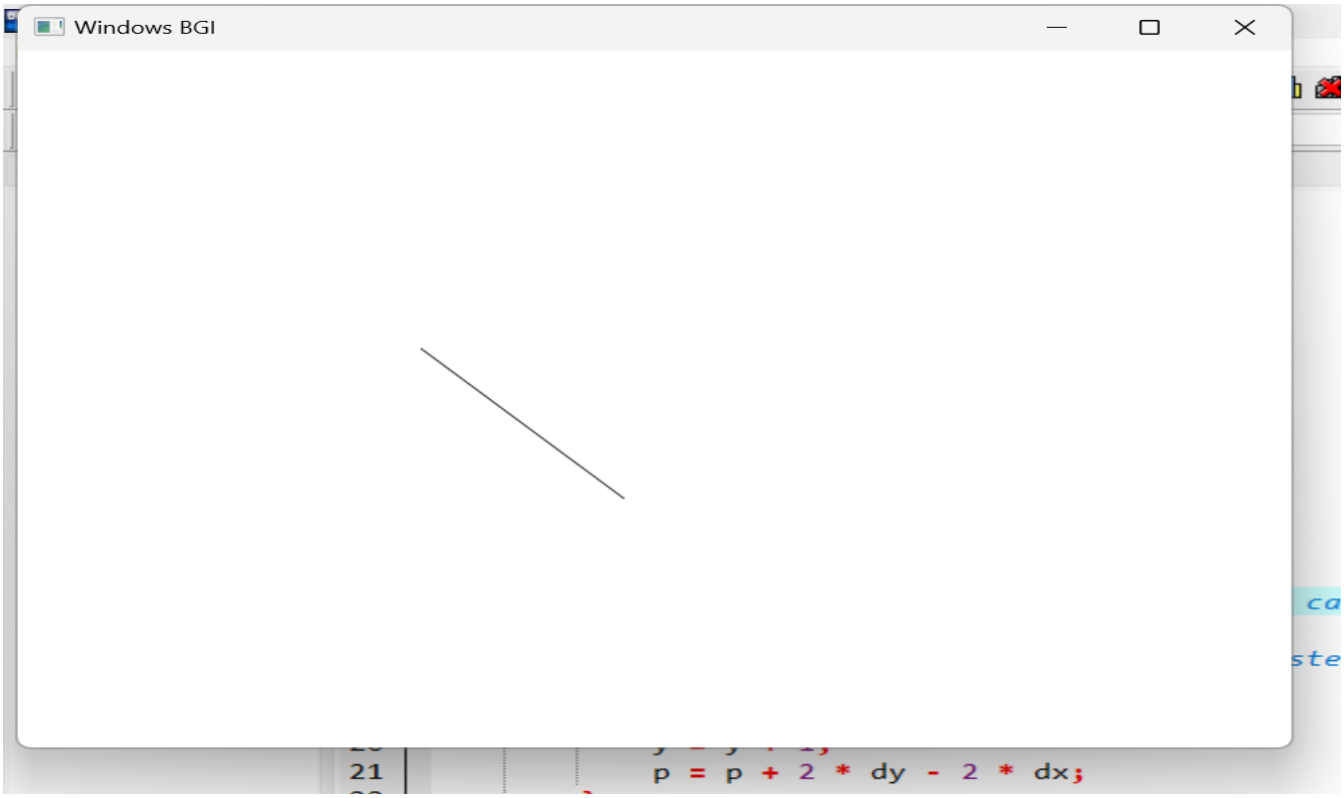

Figure Name: System Output.

Conclusion:

The **Bresenham's Line Drawing Algorithm** is an efficient and precise method for drawing a straight line by determining pixel positions using integer calculations. In this lab, we observed that Bresenham's algorithm produces accurate and visually smooth lines while minimizing computational complexity. By using a decision parameter to choose the next pixel, it efficiently handles different slopes without requiring floating-point arithmetic. This experiment helped us understand an essential technique in computer graphics for optimized line rendering on raster displays.

Remarks: Using App:

DEV C++