BUBT | BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY
Committed to Academic Excellence

Lab Report-02

Course Code: CSE-342
Course Title: Computer Graphics Lab

Submitted By :

Name: Ekramul Asfah Abeer
ID: 21225103134
Intake: 49
Section: 03

Submitted To:

Md. Khairul Islam
Lecturer,
Dept. of CSE
Bangladesh University of Business and Technology

Title: Implementation of Bresenham's Line Drawing Algorithm

Objective: Implement Bresenham's line drawing algorithm, which efficiently determines and plots the pixels that form a straight line between two given points on a 2D grid, minimizing computational complexity.

Introduction:

Bresenham's algorithm - a fundamental method in Computer Graphics - is a clever way of approximating a continuous straight line with discrete pixels, ensuring that the line appears straight and smooth on a pixel-based display.

The algorithm calculates which pixels to color in order to create a straight line between two points. Since it works exclusively with integer arithmetic, it avoids the need for costly floating-point calculations and is better suited for hardware-constrained environments. Not to mention, it's also more accurate than simply rounding coordinates to the nearest pixel.

Algorithm:

Given-
Starting coordinates = (X0, Y0)
Ending coordinates = (Xn, Yn)

The points generation using Bresenham Line Drawing Algorithm involves the following steps-

Step-01:

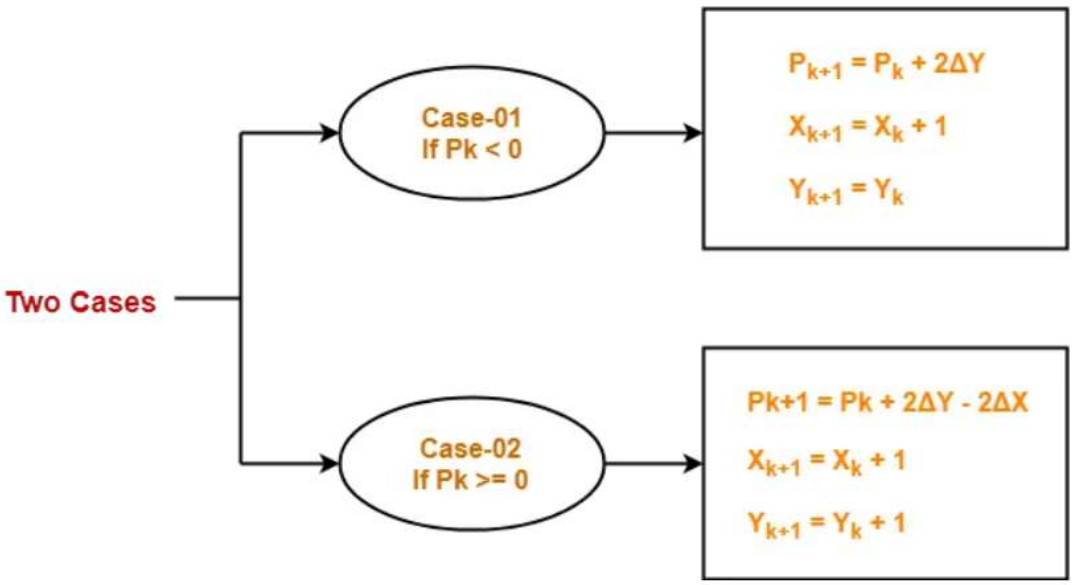
Calculate ΔX and ΔY from the given input.
These parameters are calculated as-
$\Delta X = Xn - X0$
$\Delta Y = Yn - Y0$

Step-02:

Calculate the decision parameter Pk.
It is calculated as-
$Pk = 2\Delta Y - \Delta X$

Step-03:

Suppose the current point is (Xk, Yk) and the next point is (Xk+1, Yk+1).
Find the next point depending on the value of decision parameter Pk.
Follow the below two cases-



Step-04:

Keep repeating Step-03 until the end point is reached or number of iterations equals to (ΔX-1) times**.**

Code:

```
import matplotlib.pyplot as plt

def bresenham_line(x0, y0, x1, y1):

    xcoordinates = []
    ycoordinates = []

    dx = abs(x1 - x0)
    dy = abs(y1 - y0)
    x = x0
    y = y0

    p = 2*dy-dx

    while x<x1:
        if p>=0:
            y=y+1
            p=p+2*(dy-dx)
        else:
            p=p+2*dy
      Z  x =x+1
```

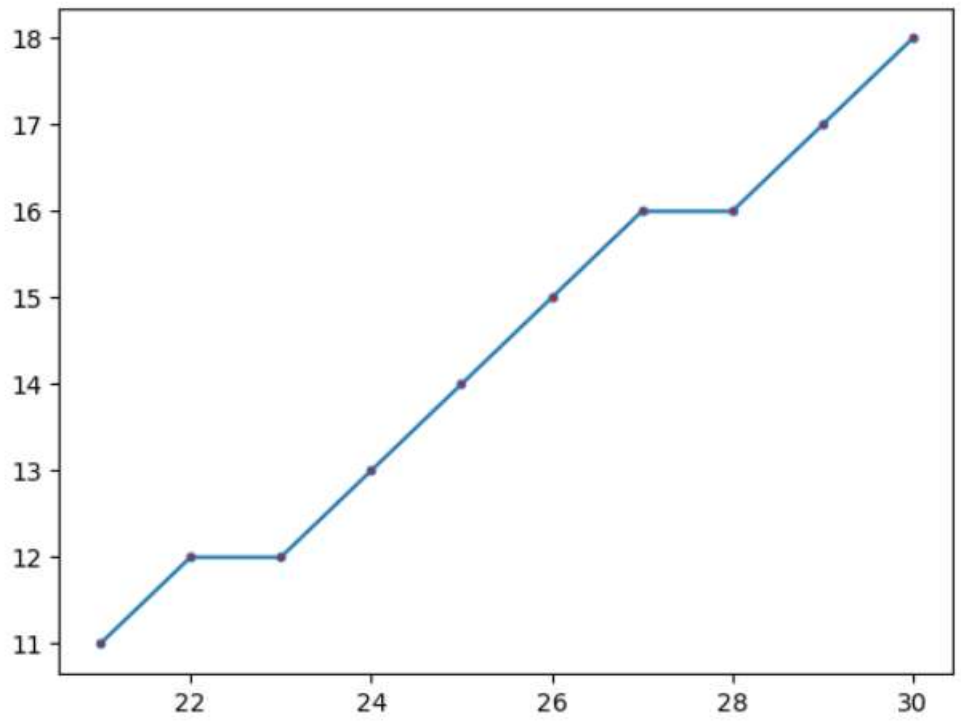
```
        print("x:", x , end=" ")
        print("y", y,end="\n")

        xcoordinates.append(x)
        ycoordinates.append(y)
    plt.plot(xcoordinates,ycoordinates, marker='o' ,markersize=3,markerfacecolor='red')
    plt.show()

print("Enter Co-ordinates of Line")
x0 = int(input("Enter x0:"))
y0 = int(input("Enter y0:"))
x1 = int(input("Enter x1:"))
y1 = int(input("Enter y1:"))
bresenham_line(x0,y0,x1,y1)
```

Output:

```
Enter Co-ordinates of Line
Enter x0:20
Enter y0:10
Enter x1:30
Enter y1:18
```



Conclusion:

Bresenham's Line Drawing Algorithm is an efficient and accurate rasterization algorithm used to draw straight lines on a pixel-based display. It eliminates the need for floating-point arithmetic by using only integer operations, making it computationally faster than traditional methods like the Digital Differential Analyzer (DDA).

The algorithm is particularly useful in computer graphics for rendering lines with high performance, especially in embedded systems and low-power devices. It ensures that the plotted pixels are as close as possible to the actual theoretical line, maintaining visual accuracy.

Overall, Bresenham's algorithm is widely used in computer graphics applications, including CAD software, gaming, and image processing, due to its simplicity, speed, and efficiency.