BUBT | BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Committed to Academic Excellence

Lab Report

Title	Marks
Course Code: CSE-342 **Course Title**: Computer Graphics **Lab no**:	

Submitted By	Submitted To
Name: Junaeid Hasan Bijoy **ID**: 21225103299 **Intake**: 49 **Section**: 05 Department of CSE Bangladesh University of Business & Technology	**Name**: Md. Khairul Islam **Designation**: Lecturer Department of CSE Bangladesh University of Business & Technology

Submission Date *Teacher's Signature*

1. **Title**: Line Drawing Algorithm

2. **Objective**:

The objective of this report is to understand and implement the line drawing algorithm(DDA), analyze the efficiency, and evaluate the effectiveness in computer graphics for generating straight lines between two points.

3. **Environment**:

- Operating System: Windows
- Programming Language: Python
- Editor: Jupyter Notebook
- Graphics Library: matplotlib
- Hardware: Standard computer system with basic graphics capabilities

4. **Introduction**:

In computer graphics, a line is a fundamental element used to create shapes, curves, and complex images. Since computer screens are made up of discrete pixels, drawing a straight line requires approximating the ideal line by selecting the nearest pixels. The Digital Differential Analyzer (DDA) algorithm is one of the simplest methods used for this purpose. It incrementally plots points between two endpoints of a line by calculating intermediate positions.

This lab report focuses on implementing and analyzing the DDA algorithm to understand its working principle and performance.

5. **Algorithm**:

Digital Differential Analyzer (DDA) Algorithm

Steps:

1. Input Coordinates:

Accept the starting point (x0,y0) and the ending point (x1,y1) of the line.

2. Calculate Differences:

Compute the absolute differences:

$dx = |x1 - x0|$

$dy = |y1 - y0|$

3. Determine Steps:

Set the number of steps as the maximum of dx and dy:

max_steps=max(dx,dy)

4. Compute Increments:

Calculate the incremental changes in x and y for each step:

xincrement=(dx/max_steps)

yincrement=(dy/max_steps)

5. Initialize Starting Point:

Set x=x0and y=y0

6. Iterate Over Steps:

For each step:

1. Append the rounded x and y values to the list of coordinates.
2. Update x and y by adding xincrementand yincrement, respectively.

7. Plot the Line:

Use the list of computed points to plot the line on a graph.

8. Display Results:
   Show the generated line on a grid with appropriate markers.

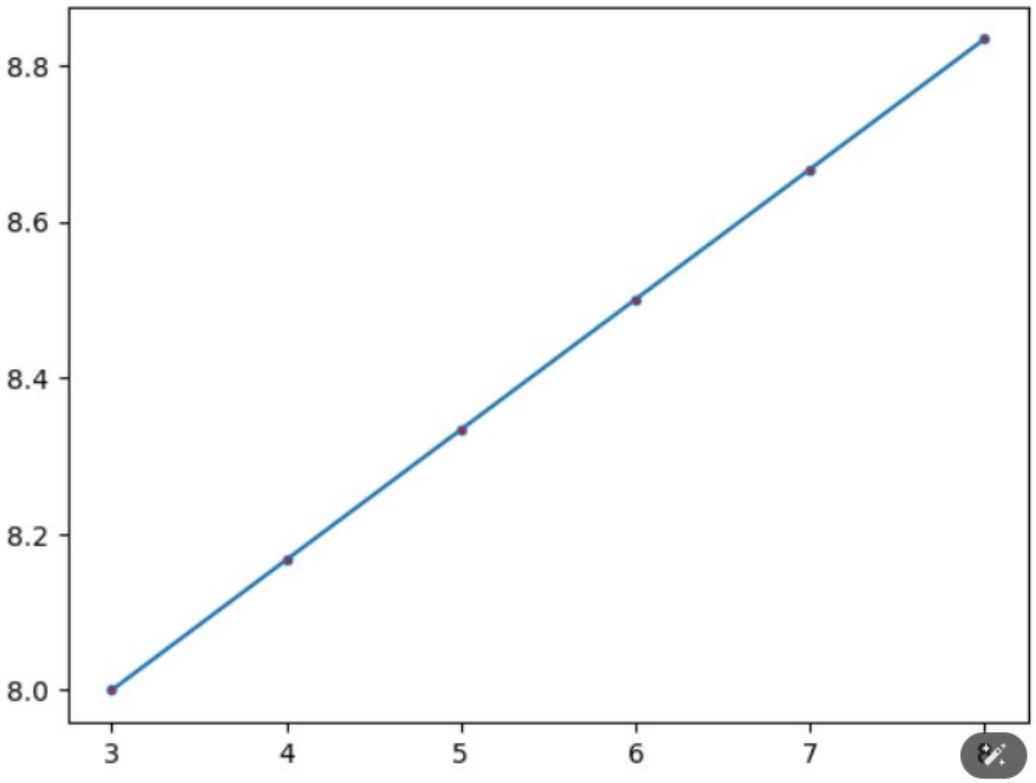
6. **Code**:

```
import matplotlib.pyplot as plt
```

```
def DDA(x0,y0,x1,y1):
    dx=abs(x0-x1)
    dy=abs(y0-y1)
    max_steps=max(dx,dy)
    xincrement=dx/max_steps
    yincrement=dy/max_steps
    x=float(x0)
    y=float(y0)
    xcoordinate=[]
    ycoordinate=[]
    for i in range (max_steps):
        xcoordinate.append(x)
        ycoordinate.append(y)
        print("x:", x, end=" ")
        print("y:", y, end="\n")
        x=x+xincrement
        y=y+yincrement
    plt.plot(xcoordinate,ycoordinate,marker='o',markersize=3,markerfacecolor="red")
    plt.show()
print("Input first point coordinates:")
x0=int(input())
y0=int(input())
print("Input second point coordinates:")
x1=int(input())
y1=int(input())
DDA(x0,y0,x1,y1)
```

7.**Snapshot(Input & Output)**:

```
Input first point coordinates:
 3
 8
Input second point coordinates:
 9
 9
x: 3.0 y: 8.0
x: 4.0 y: 8.166666666666666
x: 5.0 y: 8.333333333333332
x: 6.0 y: 8.499999999999998
x: 7.0 y: 8.666666666666664
x: 8.0 y: 8.83333333333333
```



8. **Discussion & conclusion**:

From the implementation and analysis of the DDA algorithm, we observed the following:

1. The algorithm uses floating-point arithmetic, which can be computationally expensive compared to integer-based methods.
2. It produces smooth lines but requires rounding operations, which may introduce minor inaccuracies.
3. Despite its limitations, the DDA algorithm is simple to implement and serves as a foundation for understanding line-drawing techniques in computer graphics.

In conclusion, the DDA algorithm is a straightforward method suitable for educational purposes and applications where simplicity is preferred over computational efficiency.