Department of CSE
SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VII

31 July 2021

UCS 1712 - Graphics And Multimedia Lab

Exercise 2: Line Drawing Using Digital Differential Analyzer Algorithm

Aim:

To plot points that make up the line with endpoints (x0, y0) and (xn, yn) using DDA line drawing algorithm.

- Case 1: +ve slope Left to Right line
- Case 2: +ve slope Right to Left line
- Case 3: -ve slope Left to Right line
- Case 4: -ve slope Right to Left line

Each case has two subdivisions (i) $|m| <= 1$ (ii) $|m| > 1$

Note that all four cases of line drawing must be given as test cases.

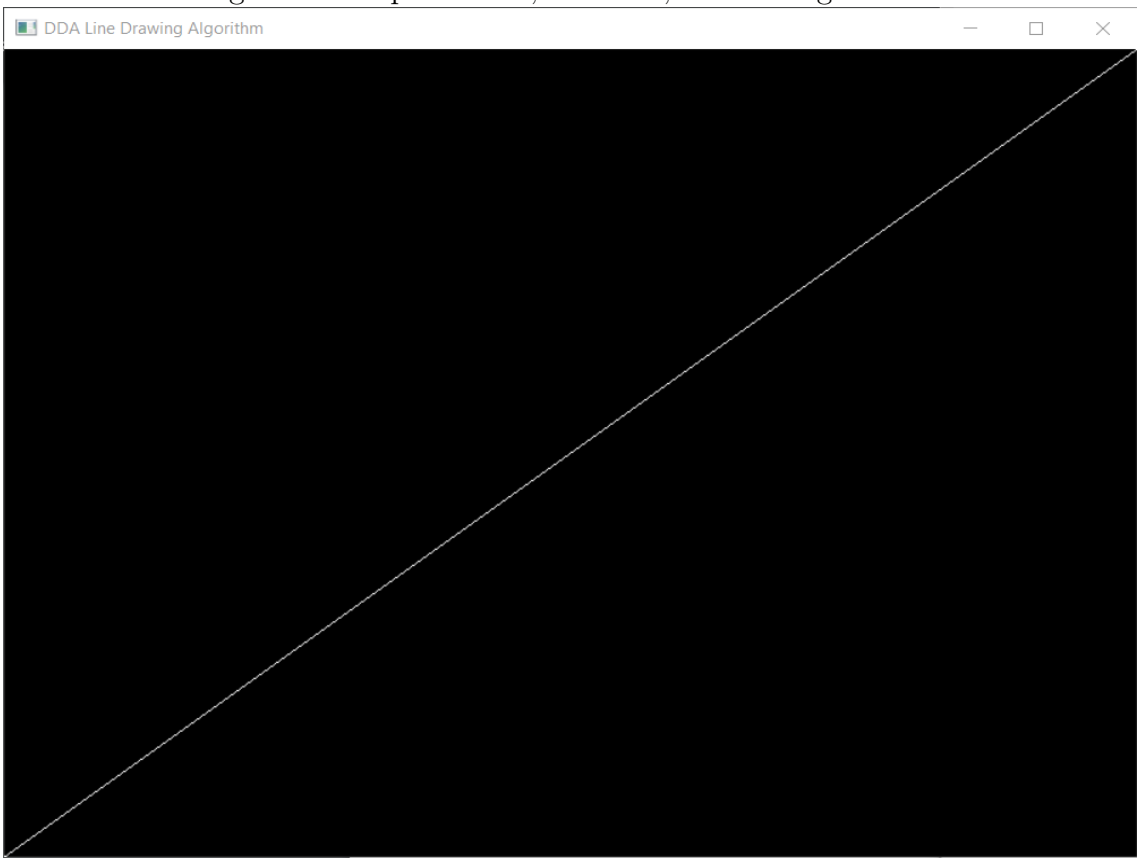
Code: DDA:

```
//To implement the DDA Line Drawing Algorithm
// DDA: Digital Differential Algorithm

#include <windows.h>
#include <stdio.h>
#include <GL/glut.h>

GLfloat x1, y1, x2, y2;

const int WINDOW_WIDTH = 800;
const int WINDOW_HEIGHT = 600;

void initializeDisplay();
void drawLine();
GLint round(GLfloat num);

int main(int argc, char **argv){


    printf("\nEnter the value of X1: ");
    scanf("%f", &x1);

    printf("\nEnter the value of Y1: ");
    scanf("%f", &y1);

    printf("\nEnter the value of X2: ");
    scanf("%f", &x2);

    printf("\nEnter the value of Y2: ");
    scanf("%f", &y2);

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    glutCreateWindow("DDA Line Drawing Algorithm");

    initializeDisplay();
    glutDisplayFunc(drawLine);
    glutMainLoop();

    return 1;
}

void initializeDisplay(){
    //Initialize the display parameters

```

```
48      glClearColor(0, 1, 1, 0);       //Display window color
49      glMatrixMode(GL_PROJECTION);    //Choose projection
50      gluOrtho2D(0, 800, 0, 600);     //Set transformation
51  }
52
53  void drawLine(){
54      GLfloat dx, dy, k, x_inc, y_inc, x, y, slope_max;
55
56      dx = x2 - x1;
57      dy = y2 - y1;
58
59      if(abs(dx) > abs(dy)){
60          // |slope| < 0
61          slope_max = abs(dx);
62      } else {
63          // |slope| >= 0
64          slope_max = abs(dy);
65      }
66
67      // if dx/dy >= slope_max, then x_inc/y_inc will be >= 1
68      // respectively, and the other will be < 1.
69      // increments will be calculated on this basis
70      // according to the DDA Algorithm
71      x_inc = dx/slope_max;
72      y_inc = dy/slope_max;
73
74      //Initial point
75      x = x1;
76      y = y1;
77
78      glBegin(GL_POINTS);
79      glVertex2i(x, y);
80
81      //Plot for all points from 1 to slope_max
82      for(k = 1; k <= slope_max; k++){
83          x += x_inc;
84          y += y_inc;
85
86          glVertex2i(x, y);
87      }
88
89      glEnd();
90      glFlush();
91  }
```

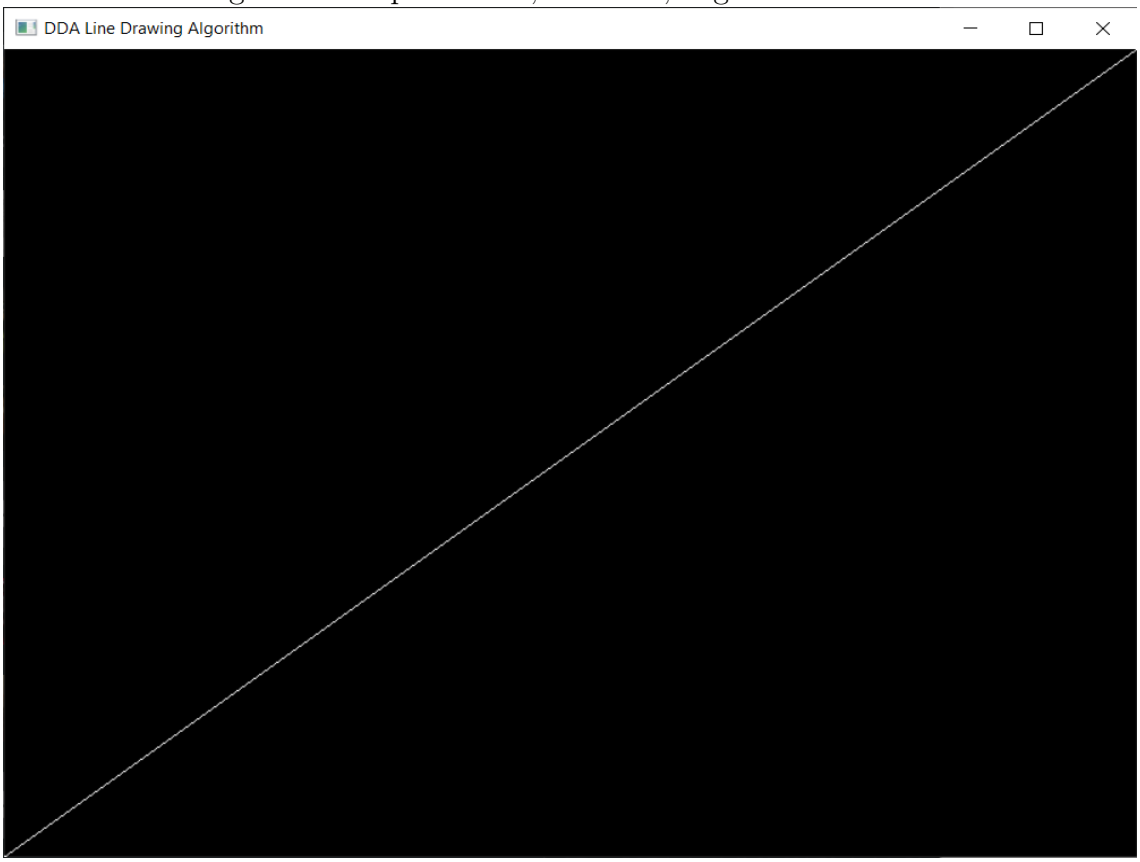
Output: DDA Case 1 - (0, 0) to (800, 600):

Figure 1: Output: DDA, M <= 1, Left to Right Line.



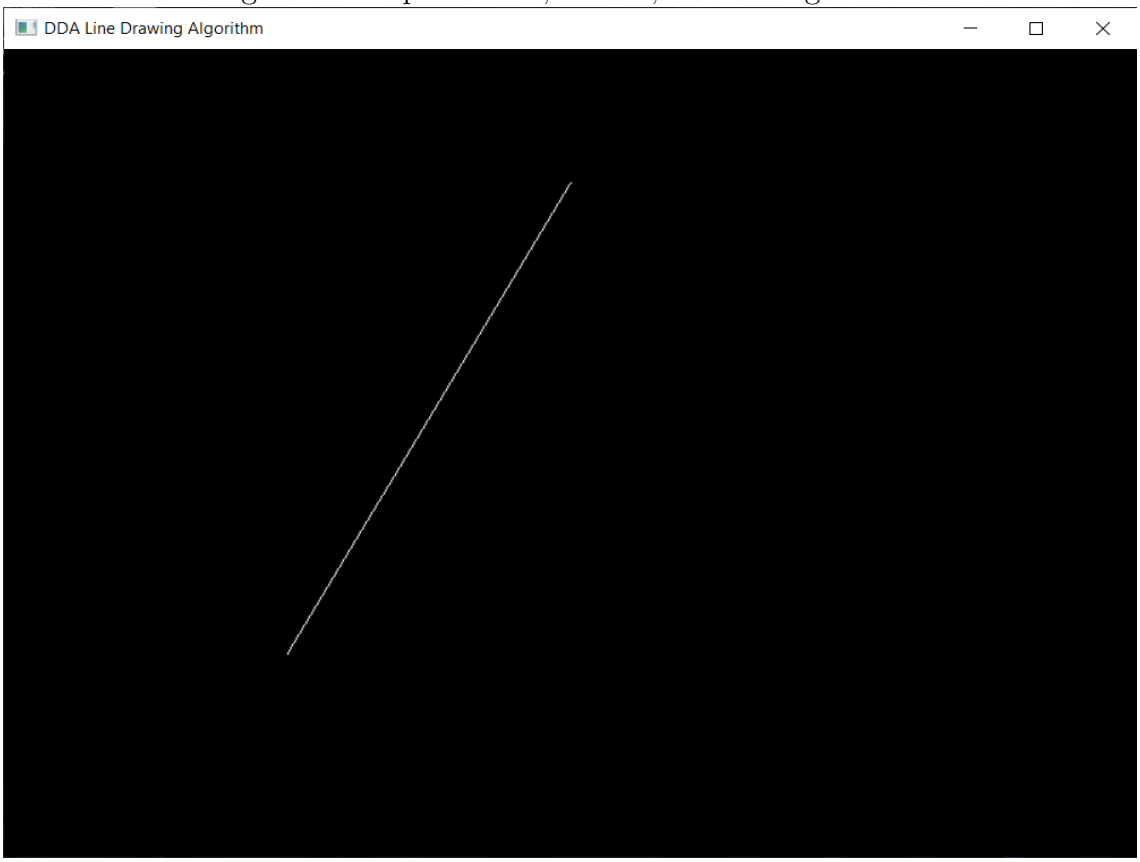
Output: DDA Case 2 - (800, 600) to (0, 0):

Figure 2: Output: DDA, M <= 1, Right to Left Line.



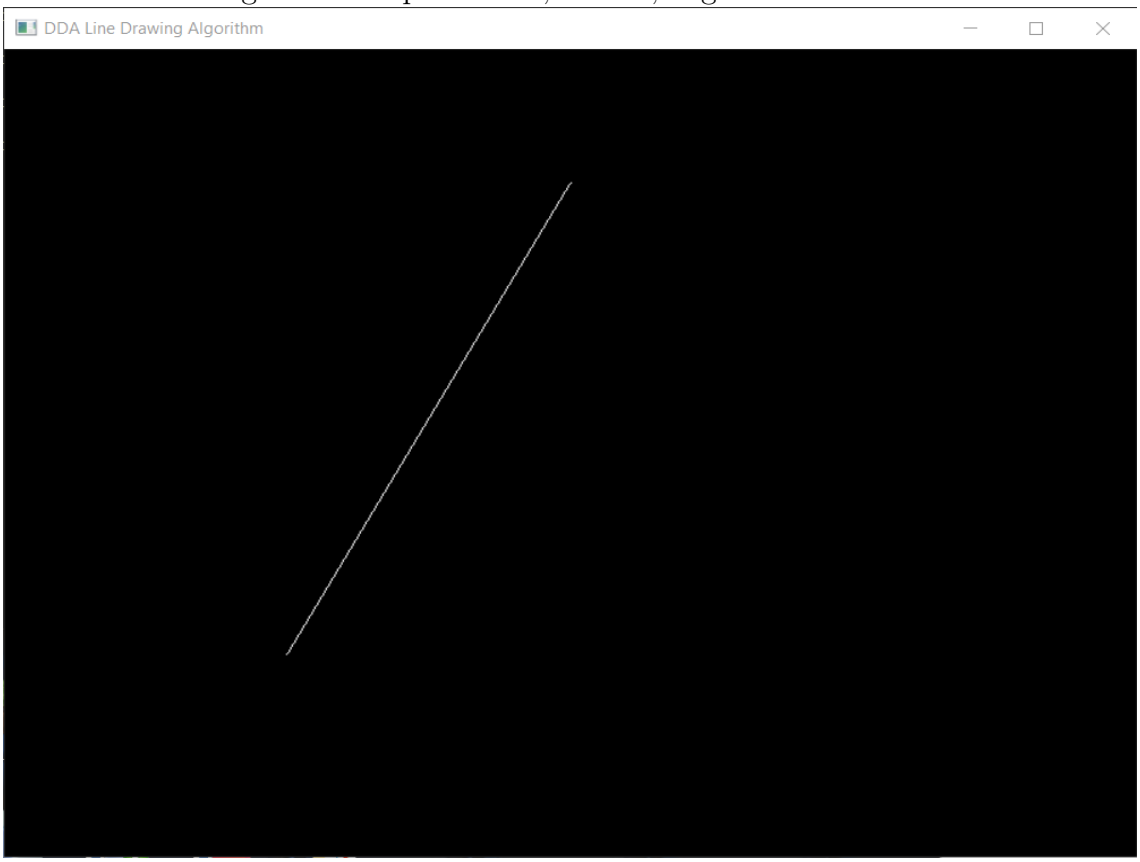
Output: DDA Case 3 - (200, 150) to (400, 500):

Figure 3: Output: DDA, M > 1, Left to Right Line.



Output: DDA Case 4 - (400, 500) to (200, 150):

Figure 4: Output: DDA, M > 1, Right to Left Line.



Learning Outcome:

- I understood the **Digital Differential Analyzer Algorithm**'s working.
- I implemented the DDA Algorithm using an OpenGL program.
- I understood how points are plotted and how increments are calculated based on slope & Δx *and* Δy values in DDA Algorithm.
- I understood that DDA Algorithm makes fine approximations and rounding off's which might be a pitfall when accurate diagrams are required.
- I understood that DDA Algorithm is less efficient and less precise than the Bresenham's Line Algorithm.
- I was able to output all different test cases appropriately to verify the correctness of my program to implement DDA Algorithm.