Department of CSE
SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VII

31 July 2021

UCS 1712 - Graphics And Multimedia Lab

Exercise 3: Line Drawing Using Bresenham's Algorithm

Aim:

To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using Bresenham's 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: Bresenham's Line Drawing Algorithm:

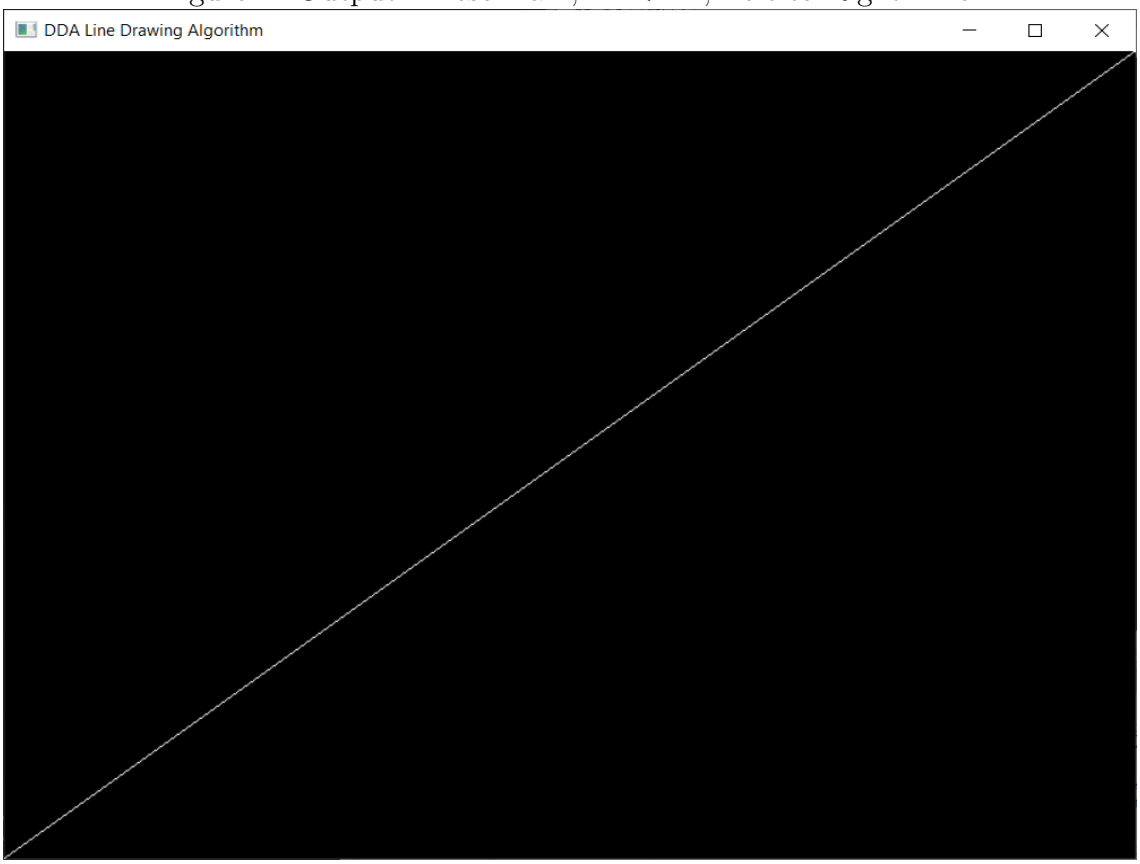
```
//To implement the Bresenham Line Drawing Algorithm

#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

    glClearColor(0, 1, 1, 0);       //Display window color
    glMatrixMode(GL_PROJECTION);    //Choose projection
```

```
48      gluOrtho2D(0, 800, 0, 600);      //Set transformation
49  }
50
51  void drawLine(){
52      GLint dx, dy, x, y, p, inc_x1, inc_y1, inc_p1, inc_p2;
53      GLint x_sign, y_sign;
54
55      dx = x2 - x1;
56      dy = y2 - y1;
57
58      //note the sign for directionality
59      x_sign = dx/abs(dx);
60      y_sign = dy/abs(dy);
61
62      //increment is by 1, and in the direction of +/-
63      inc_x1 = 1 * x_sign;
64      inc_y1 = 1 * y_sign;
65
66      //change the differences to absolute values (crucial step)
67      dx = abs(dx);
68      dy = abs(dy);
69
70      //initial coordinates
71      x = x1;
72      y = y1;
73
74      glBegin(GL_POINTS);
75
76      if(abs(dx) > abs(dy)){
77          //X difference > Y difference
78          glVertex2i(x, y);
79
80          p = (2 * dy) - dx;
81          inc_p1 = 2 * (dy - dx);
82          inc_p2 = 2 * dy;
83
84          //plot for dx number of points
85          for(GLint i = 0; i < dx; i++){
86              if(p >= 0){
87                  y += inc_y1;
88                  p += inc_p1;
89              }
90              else{
91                  p += inc_p2;
92              }
93
94              x += inc_x1;
95
96
97              glVertex2i(x, y);
98          }
```

```
99      }
100     else{
101         //X difference <= Y difference
102         glVertex2i(x, y);
103 
104         p = (2 * dx) - dy;
105         inc_p1 = 2 * (dx - dy);
106         inc_p2 = 2 * dx;
107 
108         //plot for dy number of points
109         for(GLint i = 0; i < dy; i++){
110             if(p >= 0){
111                 x += inc_x1;
112                 p += inc_p1;
113             }
114             else{
115                 p += inc_p2;
116             }
117 
118             y += inc_y1;
119 
120             glVertex2i(x, y);
121         }
122     }
123 
124     glEnd();
125     glFlush();
126 }
```

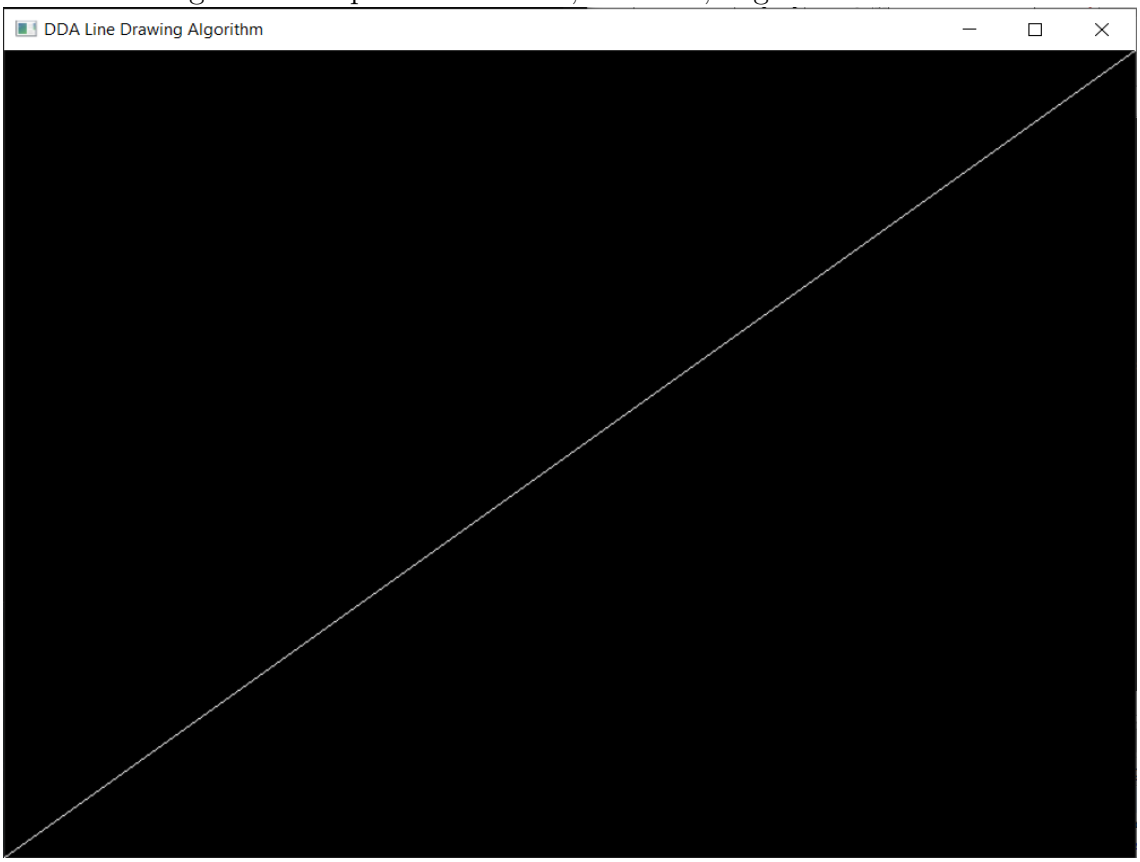
Output: Bresenham Case 1 - (0, 0) to (800, 600):

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



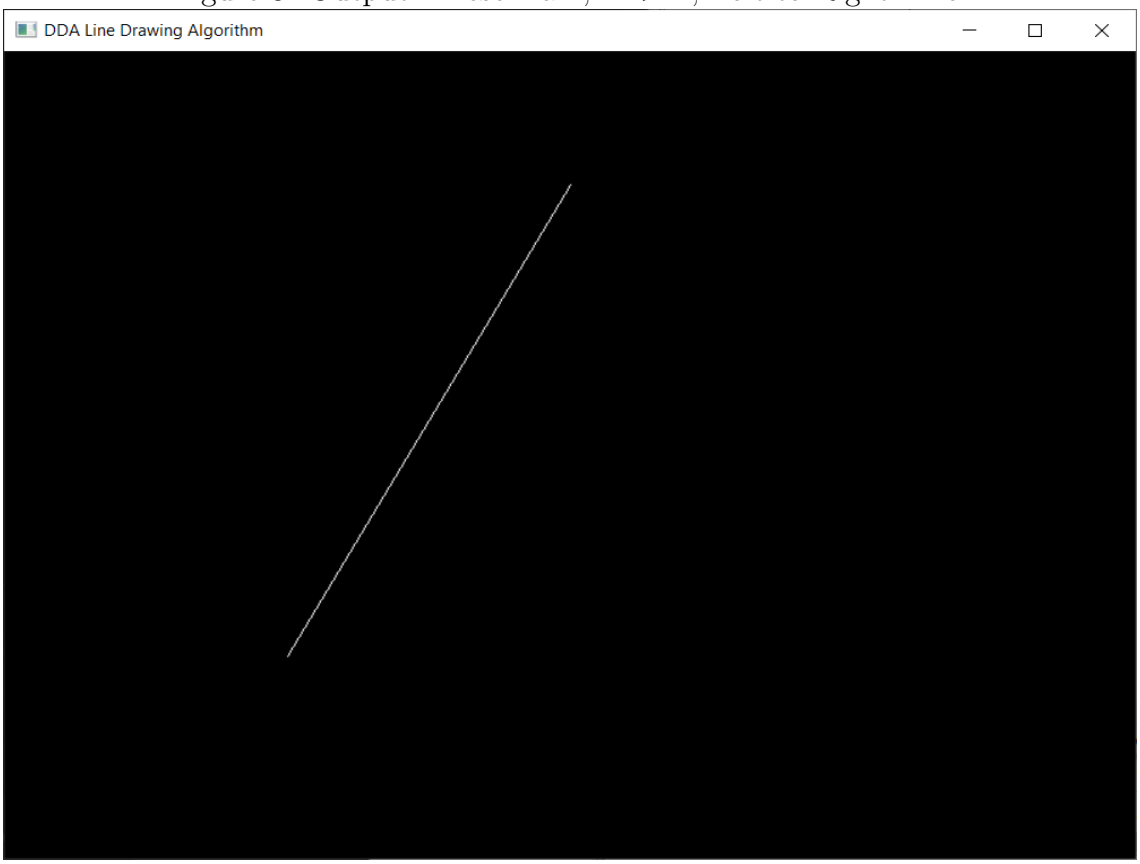
Output: Bresenham Case 2 - (800, 600) to (0, 0):

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



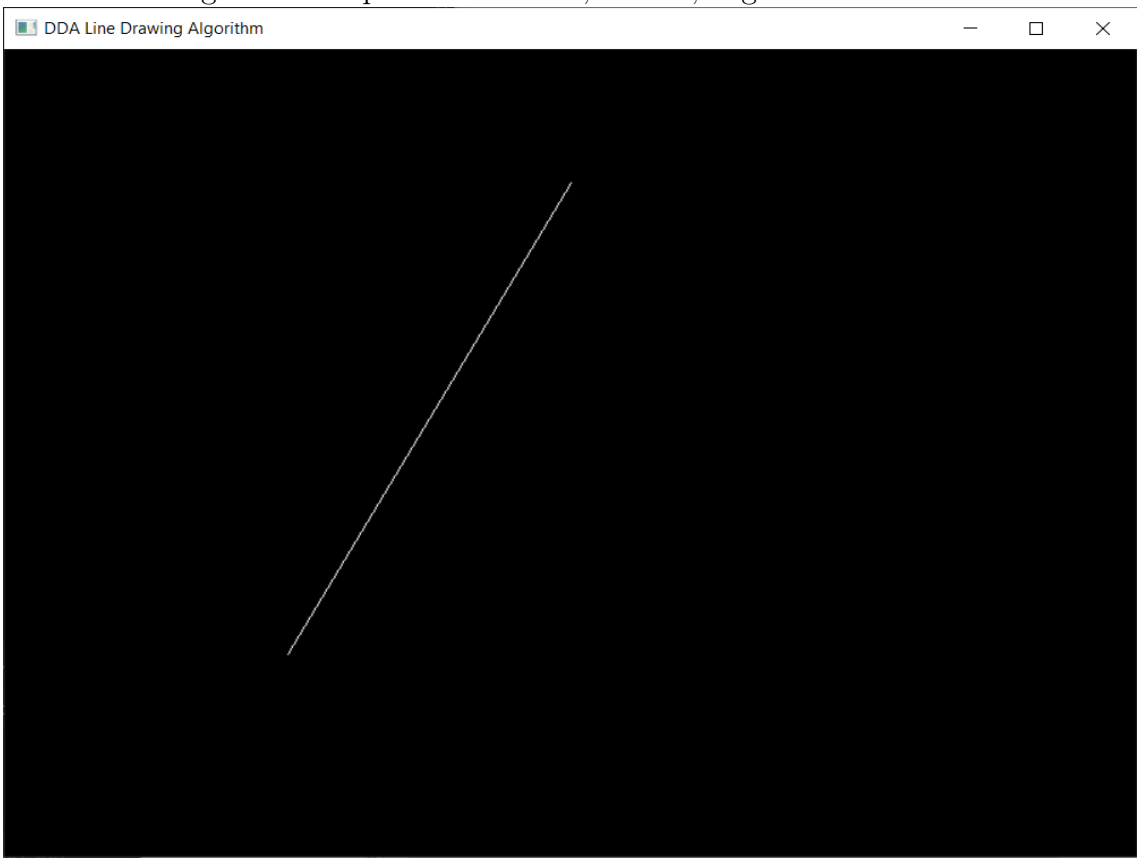
Output: Bresenham Case 3 - (200, 150) to (400, 500):

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



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

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



Learning Outcome:

- I understood the **Bresenham's Line Drawing Algorithm**'s working.
- I implemented the Bresenham Line algorithm using an OpenGL program.
- I understood how points are plotted and how increments are calculated based on the Δx *and* Δy values.
- I understood that there are two different calculations to be followed in the Bresenham's algorithm, based on the difference between the Δx *and* Δy values.
- I understood how each iteration increments x and y based on a parameter value p and how p changes it's value in each iteration.
- I understood that Bresenham's Algorithm is faster than DDA Algorithm due to purely integral calculations, avoiding the necessity for rounding off.
- I understood that Bresenham's Algorithm is also more optimal and precise, compared to DDA Algorithm.
- I was able to output all different test cases appropriately to verify the correctness of my program to implement Bresenham's Line Drawing Algorithm.