SSN COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UCS1712 – GRAPHICS AND MULTIMEDIA LAB

EX NO: 2 – Drawing 2D Primitives –Line – DDA Algorithm

Name : Likhitha Verma A
Reg : 185001084
Date : 03/08/2021

1. **AIM :**

To plot points that make up the line with endpoints (X_0 , Y_0) and (X_n , Y_n) using the DDA line drawing algorithm for four different cases with +ve and -ve slopes, left to right and vice versa with slopes (i) |m| <= 1 (ii) |m|>1

ALGORITHM :

1. Read two points **(X_1 , Y_1)** and **(X_2 , Y_2)** and assign **(X_1 , Y_1)** to **(X, Y)**
2. Compute the difference between **X** and **Y** coordinates as **dx** and **dy**.
3. Calculate the steps required to draw the line (**steps**) as max of **dx** and **dy**.
4. Compute the step size for **X** and **Y** coordinates as **X_i = dx/steps** and **Y_i = dy/steps** respectively.
5. Draw a point using glVertex2d(), with the rounded values of **X** and **Y**.
6. Increment the values of **X** and **Y** by **X_i** and **Y_i** .
7. Repeat steps 5 and 6 for **steps** time.
8. Repeat the same procedure for different cases with different slopes.

CODE :

```
#include<glut.h>
#include<math.h>
#include<stdio.h>
void myInit() {
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glColor3f(0.53, 0.06, 0.74);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glPointSize(3);
```

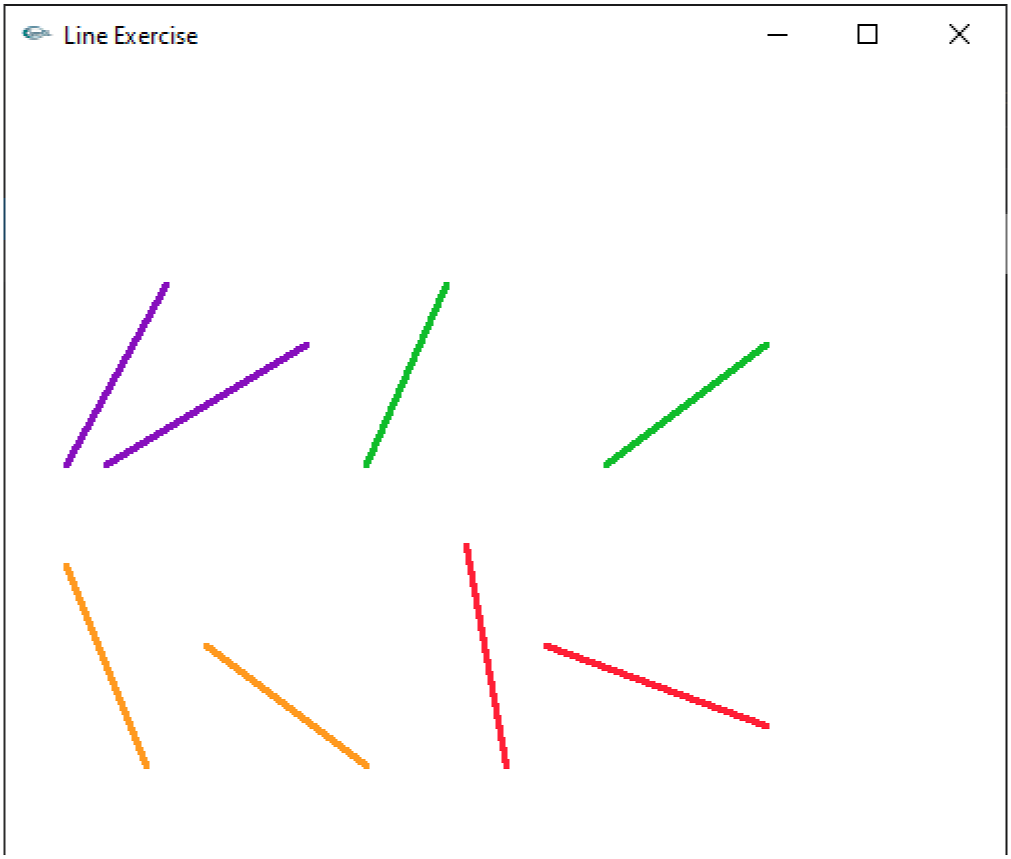
```
      gluOrtho2D(0.0, 500.0, 0.0, 500.0);
}
int round(float num) {
      return floor(num + 0.5);
}
void lineDDA(int x1, int y1, int x2, int y2) {
      float x = x1, y = y1;
      int dx = x2 - x1;
      int dy = y2 - y1;

      int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
      float xi = dx / (float)steps;
      float yi = dy / (float)steps;

      glBegin(GL_POINTS);
      for (int i = 0; i <= steps; ++i) {
            glVertex2d(round(x), round(y));
            x += xi;
            y += yi;
      }
      glEnd();
}
void myDisplay() {
      glClear(GL_COLOR_BUFFER_BIT);
      //i) + slope left to right, m>1
      lineDDA(30, 300, 80, 390);
      //ii) + slope left to right, m<=1
      lineDDA(50, 300, 150, 360);
      glColor3f(0.06, 0.74, 0.17);
      //iii) + slope right to left, m>1
      lineDDA(220, 390, 180, 300);
      //iv) + slope right to left, m<=1
      lineDDA(380,360, 300, 300);
      glColor3f(1, 0.6, 0.12);
      //v) + slope left to right, m>1
      lineDDA(30,250,70,150);
      //vi) + slope left to right, m<=1
      lineDDA(100, 210, 180, 150);
      glColor3f(1, 0.12, 0.21);
```

```
    //vii) + slope right to left, m>1
    lineDDA(230, 260, 250, 150);
    //viii) + slope right to left, m<=1
    lineDDA(270, 210, 380, 170);
    glFlush();
}
int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Line Exercise");
    glutDisplayFunc(myDisplay);
    myInit();
    glutMainLoop();
    return 1;
}
```

OUTPUT :

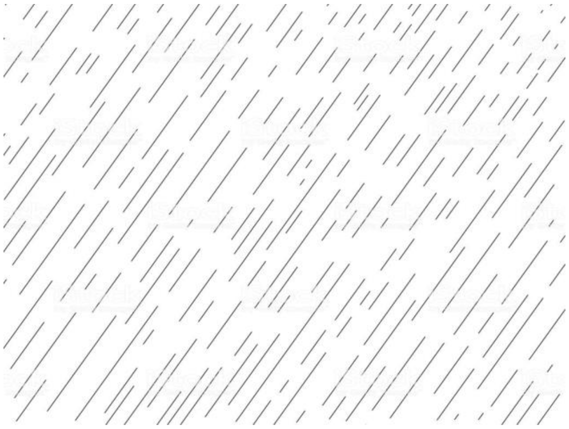


RESULT :

Thus implemented the DDA line drawing algorithm for different cases successfully.

2. AIM :

To implement the given pattern using the DDA algorithm with openGL.



ALGORITHM :

1. Since the lines in the pattern are parallel to each other, the slop remains the same for all lines. Assume the slope to be m = 1.
2. The starting and ending points or length of the varies for each line
3. Randomly fix the starting **(X_1 , Y_1)** and the length of the line.
4. Using the euclidean distance formula, Calculate the ending points **(X_2 , Y_2)** points of the lines.
5. Given **(X_1 , Y_1)** and **(X_1 , Y_1)** draw lines using the DDA algorithm.
6. Repeat the steps for a fixed number of times, say 200.

CODE :

```
#include<stdlib.h>
#include<glut.h>
#include<math.h>
#include<stdio.h>
#include<time.h>
void myInit() {
      glClearColor(1.0, 1.0, 1.0, 0.0);
      glColor3f(0.53, 0.06, 0.74);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
```

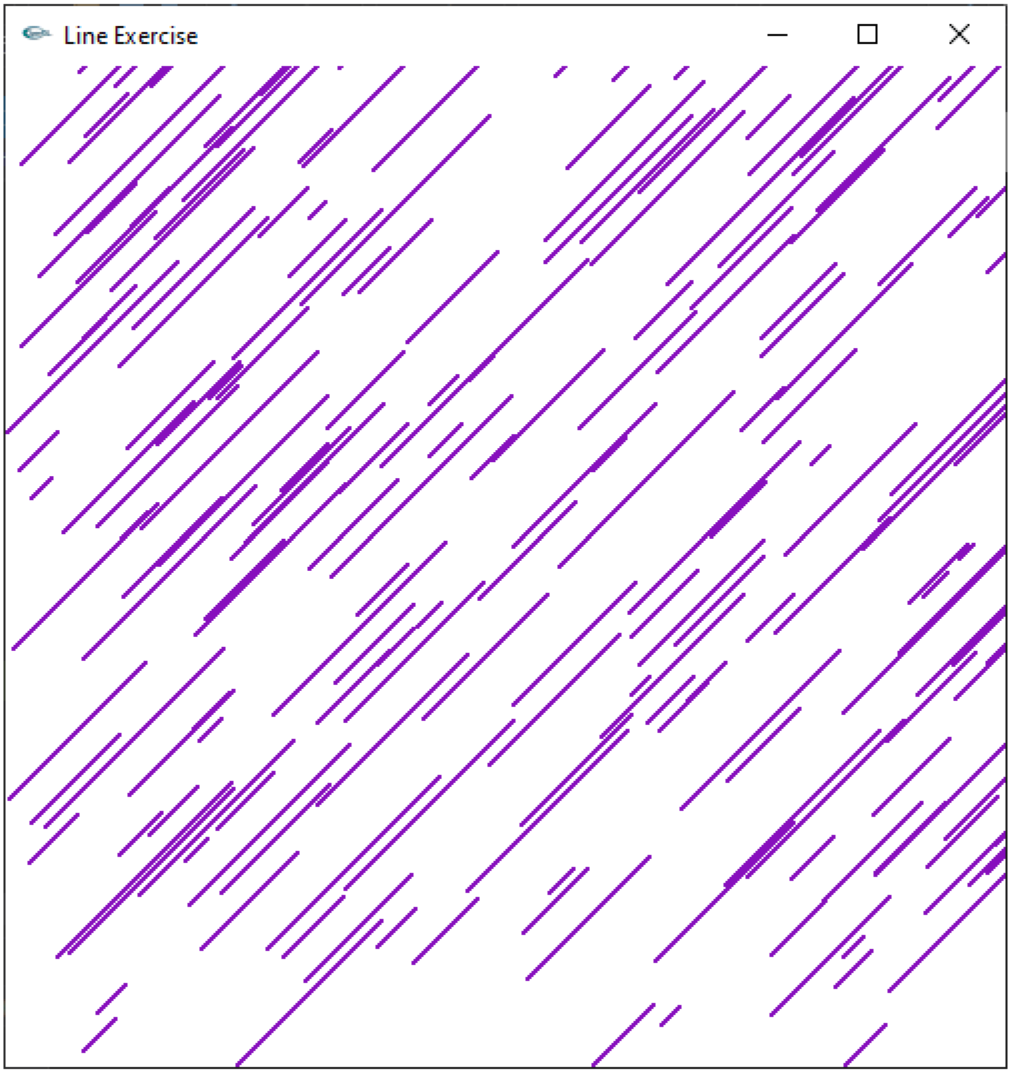
```
	glPointSize(2);
	gluOrtho2D(0.0, 500.0, 0.0, 500.0);
}
int round(float num) {
	return floor(num + 0.5);
}
void lineDDA(int x1, int y1, int x2, int y2) {
	float x = x1, y = y1;
	int dx = x2 - x1;
	int dy = y2 - y1;

	int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
	float xi = dx / (float)steps;
	float yi = dy / (float)steps;

	glBegin(GL_POINTS);
	for (int i = 0; i <= steps; ++i) {
		glVertex2d(round(x), round(y));
		x += xi;
		y += yi;
	}
	glEnd();
}
void myDisplay() {
	glClear(GL_COLOR_BUFFER_BIT);
	srand((unsigned)time(NULL));
	int x1, y1, x2, y2, dis;
	for (int i = 0; i < 200; ++i) {
		x1 = abs(rand() % 500 + 1);
		y1 = abs(rand() % 500 + 1);
		dis = abs(rand() % 150 + 10);
		x2 = int(sqrt((dis * dis) / 2)) + x1;
		y2 = int(sqrt((dis * dis) / 2)) + y1;
		lineDDA(x1, y1, x2, y2);
	}
	glFlush();
}
int main(int argc, char* argv[]) {
	glutInit(&argc, argv);
```

```
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Line Exercise");
    glutDisplayFunc(myDisplay);
    myInit();
    glutMainLoop();
    return 1;
}
```

OUTPUT :



RESULT :

Thus replicated the given pattern successfully using openGL.