Ex. No. 2	DDA Line Drawing Algorithm	Karthik D
September 8, 2022	UCS1712 - Graphics and Multimedia Lab	195001047

Aim

To develop a C++ program using the OpenGL framework to implement the DDA Line Drawing algorithm, and demonstrate all its output cases.

Question

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

1. |m|<= 1
2. |m|>1

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

DDA Algorithm

Procedure plotLineDDA(xa,xb,ya,yb:integer);

```
var
dx, dy, steps, k : integer;
xIncrement, yIncrement, x , y: real;

Begin
        dx:=xb-xa;
        dy:=yb-ya;

        if abs(dx) >abs(dy) then steps:=abs(dx)
        else steps:=abs(dy)

        xIncrement :=dx/steps;
        yIncrement :=dy/steps;

        x:=xa;
        y:=ya;
```

```
            setPixel(round(x),round(y),1);

            for k:=1 to steps do
            Begin
                  x:=x+xIncrement;
                  y:=y+yIncrement;
                  setPixel(round(x), round(y),1);
            End

      End
End {plotLineDDA}
```

Implementation using C++ Program Code

1. main.cpp - Driver and Handler to render the line using DDA for given coorinates
   Function *plotLineDDA()* implements the DDA algorithm

```
#include <GLUT/glut.h>
#include <stdio.h>
#include <math.h>

#define BUFFER_SIZE 100


void plotDivisionLines(){
      glBegin(GL_LINES);
      glVertex2d(-320, 0);
      glVertex2d(320, 0);
      glVertex2d(0, -240);
      glVertex2d(0, 240);
      glEnd();
}


void renderSpacedBitmapString(float x, float y, void *font, char *string) {
      char *c;
      int x1 = x;
      for (c = string; *c != '\0'; c++) {
      glRasterPos2f(x1, y);
      glutBitmapCharacter(font, *c);
      x1 = x1 + glutBitmapWidth(font, *c);
      }
}
```

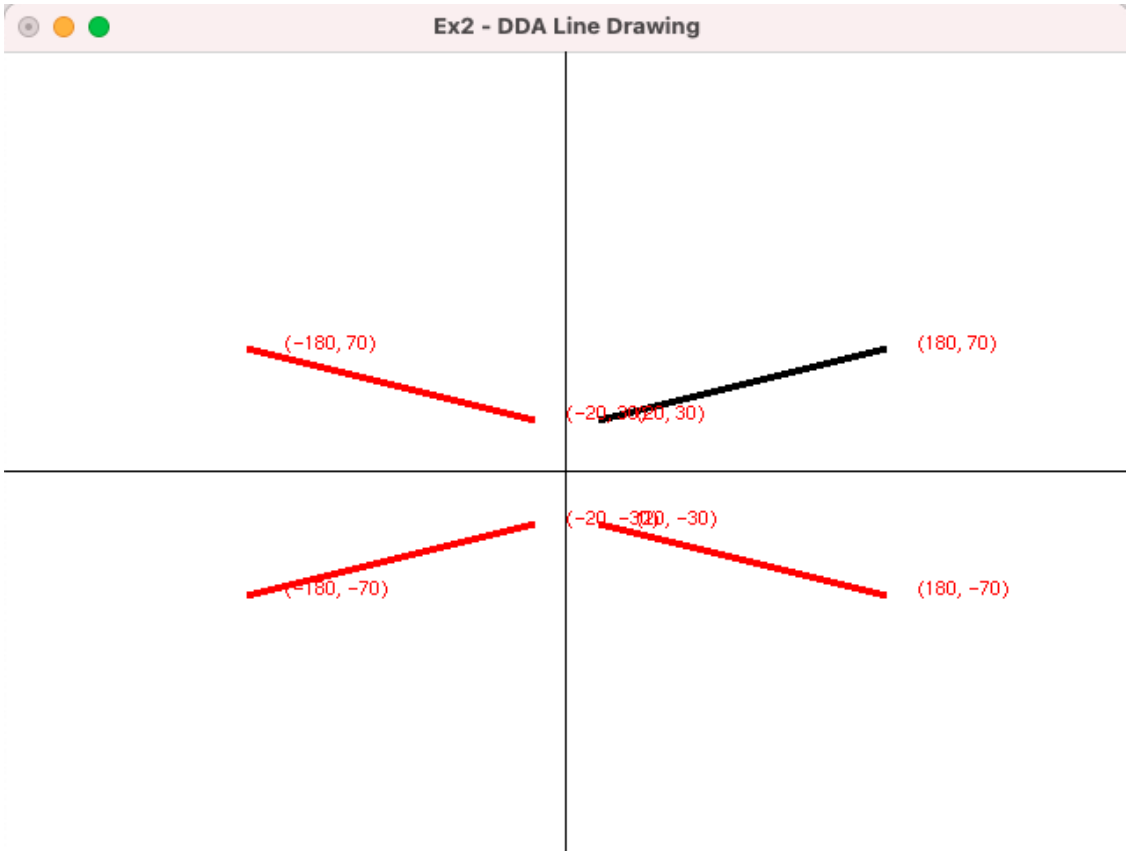
```
void markString(char *string, int x, int y, int x_offset, int y_offset) {
      glColor3f(255.0, 0, 0.0); // red color
      renderSpacedBitmapString(x+x_offset, y+y_offset,
GLUT_BITMAP_HELVETICA_10, string);
      glFlush();
}


void plotPoint(int x, int y)  {
      glBegin(GL_POINTS);
      glVertex2d(x, y);
      glEnd();
}


void plotLineDDA(int start_x, int start_y, int end_x, int end_y)  {

      float slope = (float) (end_y - start_y)/(end_x - start_x);
      float abs_slope = slope;

      short dx_sign = (start_x <= end_x ? 1 : -1);
      short dy_sign = (start_y <= end_y ? 1 : -1);

      if(slope>0) {
      abs_slope = slope;
      }
      else  {
      abs_slope = -1 * slope;
      }

      float dx, dy;
      short check_x;
      if(abs_slope <= 1)  {
      dx = (float) (1/abs_slope) * dx_sign;
      dy = (float) 1 * dy_sign;
      check_x = 0;
      }
      else  {
      dx = (float) 1 * dx_sign;
      dy = (float) abs_slope * dy_sign;
      check_x = 1;
      }

      printf("slope: %f\n", slope);
      printf("dx: %f, dy: %f\n", dx, dy);
      printf("check_x: %d\n\n", check_x);

      int x_ = start_x;
      int y_ = start_y;
      float x_val = start_x;
      float y_val = start_y;
```

```
      plotPoint(x_, y_);
      while((check_x && end_x!=x_) || (!check_x && end_y!=y_))    {
      x_val += dx;
      y_val += dy;
      x_ = (int) round(x_val);
      y_ = (int) round(y_val);
      plotPoint(x_, y_);
      // printf("\nx: %d, y: %d", x_, y_);
      fflush(stdout);
      }
      char *point_label = (char*)malloc(sizeof(char)*BUFFER_SIZE);
      sprintf(point_label, "(%d, %d)", start_x, start_y);
      markString(point_label, start_x, start_y, 20, 0);
      sprintf(point_label, "(%d, %d)", end_x, end_y);
      markString(point_label, end_x, end_y, 20, 0);
}


void display()  {
      glClear(GL_COLOR_BUFFER_BIT);
      plotDivisionLines();

      int start_x, start_y, end_x, end_y;
      printf("\nEnter Start Coordinates (x y): ");
      scanf(" %d %d", &start_x, &start_y);
      printf("Enter End Coordinates (x y): ");
      scanf(" %d %d", &end_x, &end_y);

      // plotLineDDA(10, -20, 100, -80);
      plotLineDDA(start_x, start_y, end_x, end_y);
      plotLineDDA(start_x, -start_y, end_x, -end_y);
      plotLineDDA(-start_x, start_y, -end_x, end_y);
      plotLineDDA(-start_x, -start_y, -end_x, -end_y);

      glFlush();
}


void init() {
      glClearColor(1.0, 1.0, 1.0, 0.0);
      glColor3f(0.0f, 0.0f, 0.0f);
      glPointSize(5);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(-320.0, 320.0, -240.0, 240.0);
}


int main(int argc,char* argv[]) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
```

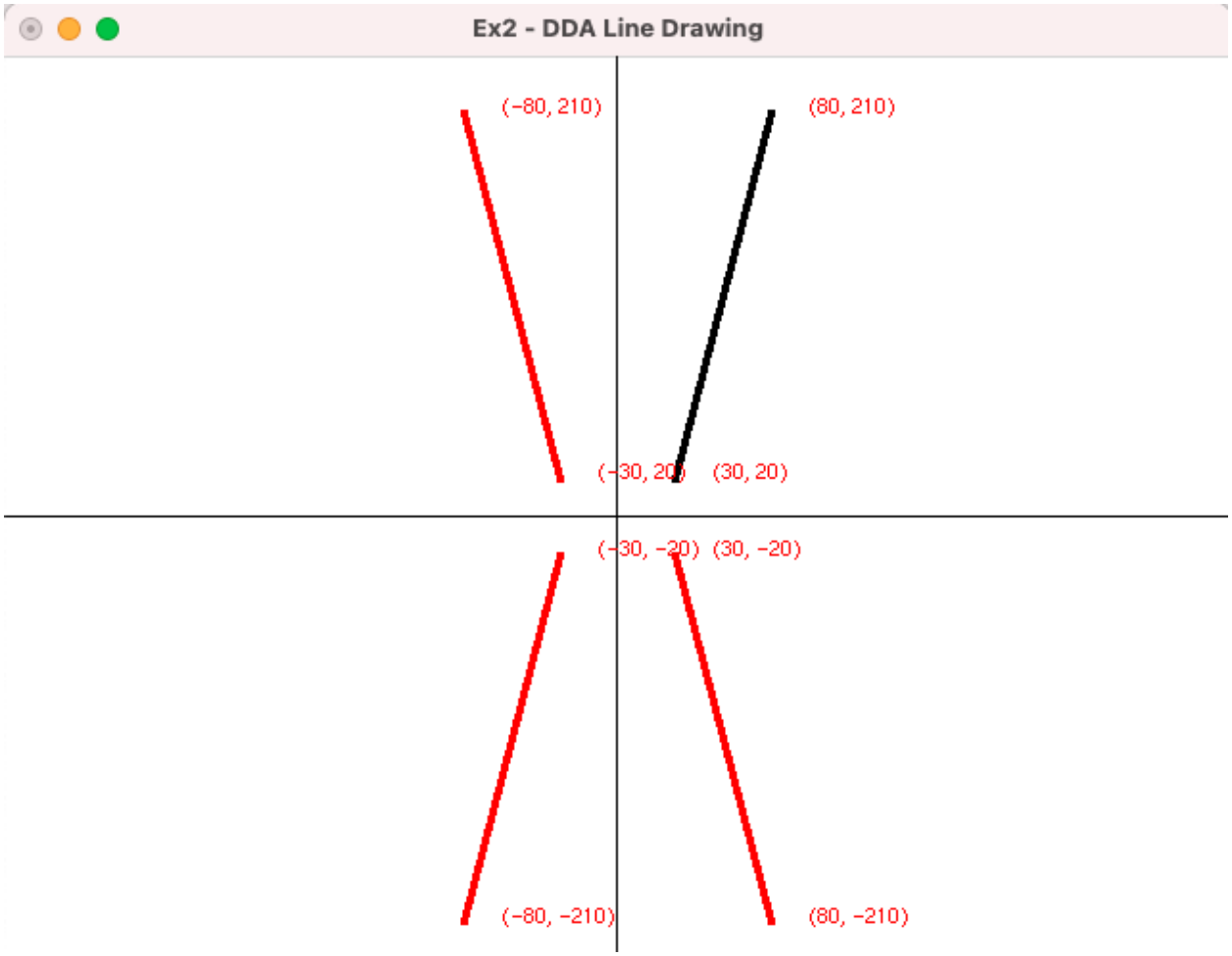
```
        glutInitWindowSize(640, 480);
        glutCreateWindow("Ex2 - DDA Line Drawing");
        glutDisplayFunc(display);
        init();
        glutMainLoop();
        return 1;
}
```

Sample Output

- **All 4 cases for |m| <= 1**



- **All 4 cases for |m| > 1**



Learning Outcomes

Through this implementation of DDA Line Drawing algorithm using the OpenGL framework and C++ programming language, the following concepts were learnt:

1. The working of the DDA algorithm
2. General understanding of the OpenGL framework and its APIs