Ex. No. 3 **Bresenham's 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 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

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

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

Bresenham's Line Drawing Algorithm

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

```
    var
    dx, dy, x, y, xEnd, p: integer;

    Begin

        dx := abs(xa-xb);
        dy := abs(ya-yb);
        p := 2 * dy - dx;

        If xa > xb T
        Then
            x := xb;
            y := yb;
            xEnd := xa;
        Else
            x := xa;
            y := ya;
            xEnd := xb;
        End

        setPixel (x, y, 1);

        While x < xEnd
        Begin
            x:= x+1;
            If p < 0 then
            Begin
                p := p + 2*dy
            Else
                y := y + 1;
                p: = p + 2*(dy-dx)
            End
            setPixel (x, y, 1);
        End

    End {plotLineBresenham}
```

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 <GL/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();
}
```

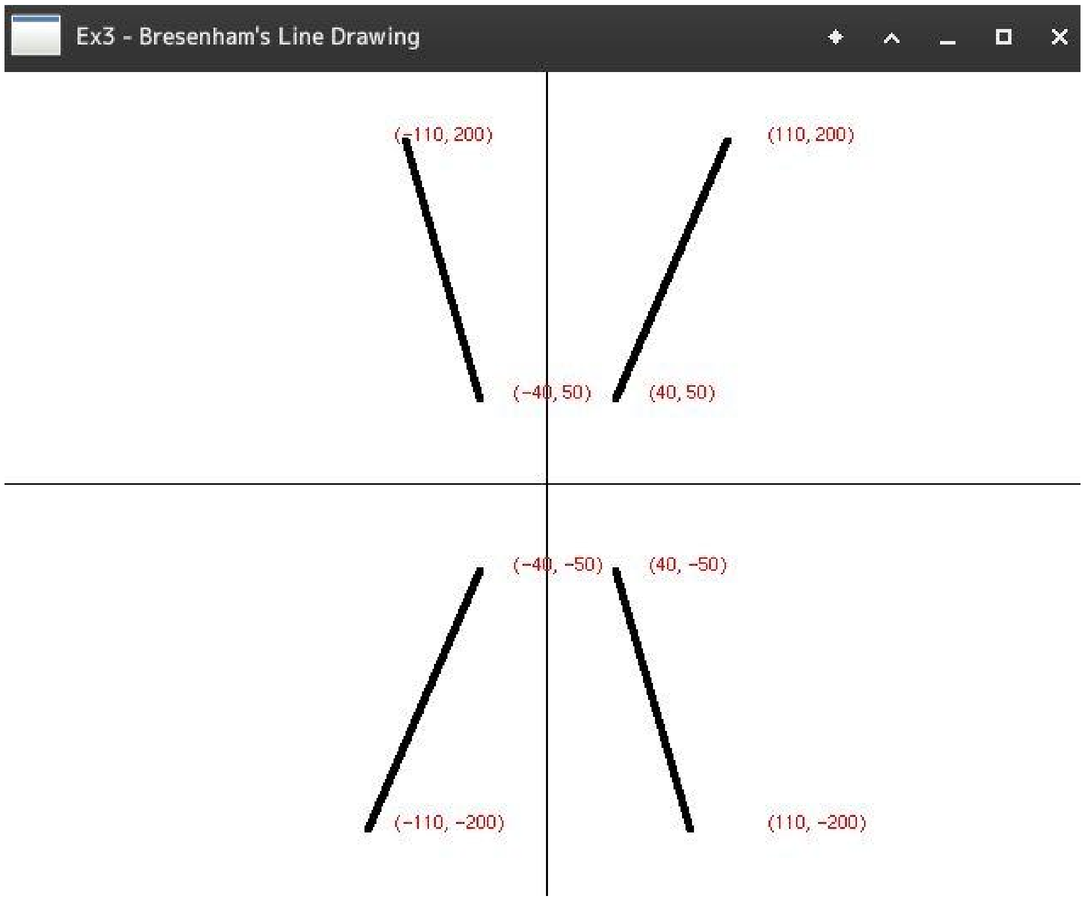
```c
void renderSpacedBitmapString(float x, float y, void *font, char
*string) {
   char *c;
   int x1 = x;
   for (c = string; *c != '\0'; c++) {
       glRasterPos2f(x1, y);
       glutBitmapCharacter(font, *c);
       x1 = x1 + glutBitmapWidth(font, *c);
   }
}


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)    {
   glColor3f(0.0, 0, 0.0); // black color
   glBegin(GL_POINTS);
   glVertex2d(x, y);
   glEnd();
}


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

   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);

int dx = end_x - start_x;
int dy = end_y - start_x;
float slope = dy / dx;

short exchange_xy = 0;
if(slope>1 || slope<-1) {
    // interchange x and y for computations,
    // and change back when plotting
    exchange_xy = 1;
    int temp;
    // exchange xs
    temp = start_x;
    start_x = start_y;
    start_y = temp;
    // exchange ys
    temp = end_y;
    end_y = end_x;
    end_x = temp;
    // exchange ds
    temp = dx;
    dx = dy;
    dy = temp;
}

int y_delta = 1;
if(dy < 0)  {
    // bottom to top
    y_delta = -1;
    dy *= -1;
}

int x_delta = 1;
if(dx < 0)  {
    // right to left
    x_delta = -1;
    dx *= -1;
    }

    int x_ = start_x;
    int y_ = start_y;

    int p_k = (2*dy) - dx;
    exchange_xy ? plotPoint(y_, x_) : plotPoint(x_, y_);
    while(x_!=end_x)  {
        x_ += x_delta;
        if(p_k<0)   {
            exchange_xy ? plotPoint(y_, x_) : plotPoint(x_, y_);
            p_k += 2 * dy;
        }
        else    {
            y_ += y_delta;
            exchange_xy ? plotPoint(y_, x_) : plotPoint(x_, y_);
            p_k += 2*(dy-dx);
        }
    }
}


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);

    // plotLineBresenham(40, 50, 110, 200);
    plotLineBresenham(start_x, start_y, end_x, end_y);
    plotLineBresenham(start_x, -start_y, end_x, -end_y);
    plotLineBresenham(-start_x, start_y, -end_x, end_y);
    plotLineBresenham(-start_x, -start_y, -end_x, -end_y);
```

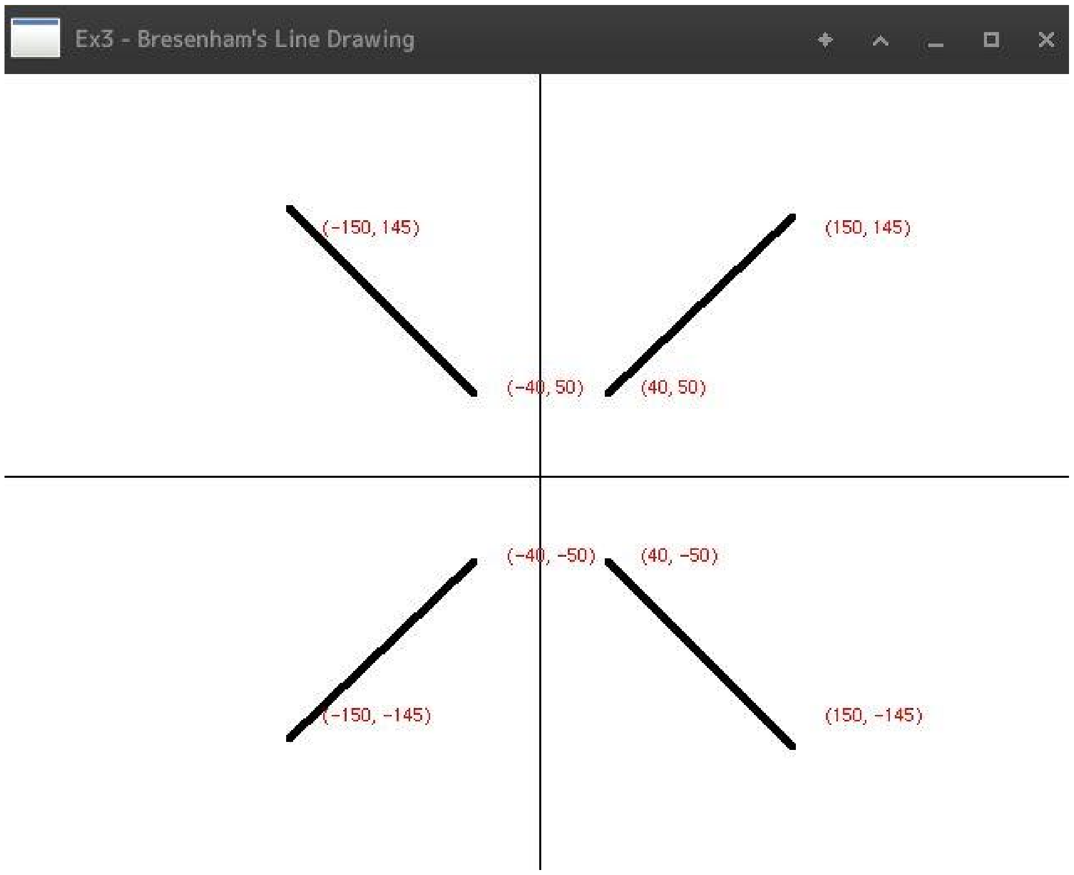
```
   glFlush();
}

void init() {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0f, 0.0f, 0.0f);
   glPointSize(4);
   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("Ex3 - Bresenham's Line Drawing");
   glutDisplayFunc(display);
   init();
   glutMainLoop();
   return 1;
}
```

Sample Output

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



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



<u>Learning Outcomes</u>

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 Bresenham's line drawing algorithm
2. General understanding of the OpenGL framework and its APIs