**UCS1712 – GRAPHICS AND MULTIMEDIA LAB**

**Gokhulnath T**

**185001051**

------------------------------------------------------------------------------------------------------------

Lab Exercise 2 : DDA Line Drawing Algorithm in C++ using OpenGL

2) 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

* |m|<= 1

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<cstdlib>

void display(void)

{

    float dy, dx, step, x, y, k, Xin, Yin;

    float x1 = 0;

    float y1 = 0;

    float x2 = 60;

    float y2 = 40;

    dx = x2 - x1;

    dy = y2 - y1;

    if (abs(dx) > abs(dy))

    {

        step = abs(dx);

    }

    else

        step = abs(dy);

    Xin = dx / step;

    Yin = dy / step;

    x = x1;

    y = y1;

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

    for (k = 1; k <= step; k++)

    {

        x = x + Xin;

        y = y + Yin;

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void init(void)

{

    glClearColor(0.7, 0.7, 0.7, 0.7);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-100, 100, -100, 100);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("EX2 DDA");

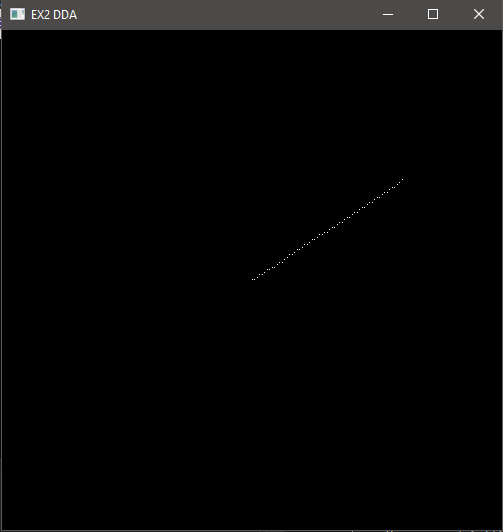
    init();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}



* |m|>1

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<cstdlib>

void display(void)

{

    float dy, dx, step, x, y, k, Xin, Yin;

    float x1 = 0;

    float y1 = 0;

    float x2 = 40;

    float y2 = 60;

    dx = x2 - x1;

    dy = y2 - y1;

    if (abs(dx) > abs(dy))

    {

        step = abs(dx);

    }

    else

        step = abs(dy);

    Xin = dx / step;

    Yin = dy / step;

    x = x1;

    y = y1;

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

    for (k = 1; k <= step; k++)

    {

        x = x + Xin;

        y = y + Yin;

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void init(void)

{

    glClearColor(0.7, 0.7, 0.7, 0.7);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-100, 100, -100, 100);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

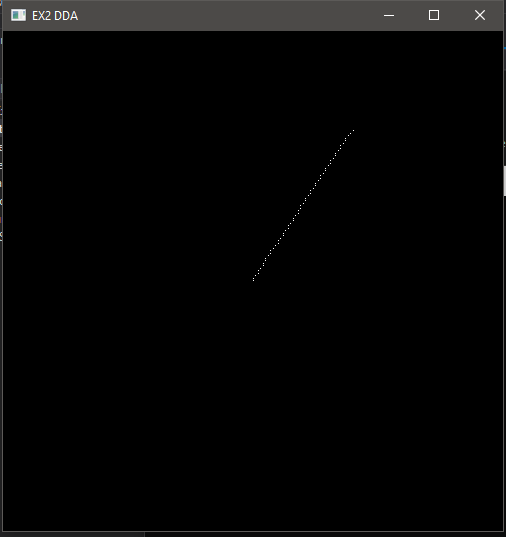
    glutCreateWindow("EX2 DDA");

    init();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

Case 2: +ve slope Right to Left line

* |m|<= 1

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<cstdlib>

void display(void)

{

    float dy, dx, step, x, y, k, Xin, Yin;

    float x1 = 60;

    float y1 = 20;

    float x2 = 0;

    float y2 = 0;

    dx = x2 - x1;

    dy = y2 - y1;

    if (abs(dx) > abs(dy))

    {

        step = abs(dx);

    }

    else

        step = abs(dy);

    Xin = dx / step;

    Yin = dy / step;

    x = x1;

    y = y1;

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

    for (k = 1; k <= step; k++)

    {

        x = x + Xin;

        y = y + Yin;

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void init(void)

{

    glClearColor(0.7, 0.7, 0.7, 0.7);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-100, 100, -100, 100);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("EX2 DDA");

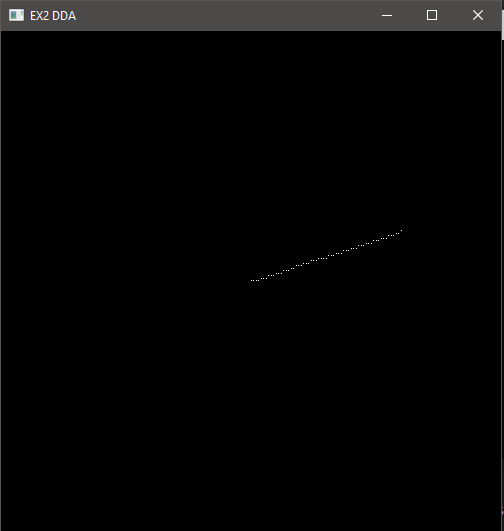
    init();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}



* |m|>1

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<cstdlib>

void display(void)

{

    float dy, dx, step, x, y, k, Xin, Yin;

    float x1 = 20;

    float y1 = 60;

    float x2 = 0;

    float y2 = 0;

    dx = x2 - x1;

    dy = y2 - y1;

    if (abs(dx) > abs(dy))

    {

        step = abs(dx);

    }

    else

        step = abs(dy);

    Xin = dx / step;

    Yin = dy / step;

    x = x1;

    y = y1;

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

    for (k = 1; k <= step; k++)

    {

        x = x + Xin;

        y = y + Yin;

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void init(void)

{

    glClearColor(0.7, 0.7, 0.7, 0.7);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-100, 100, -100, 100);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("EX2 DDA");

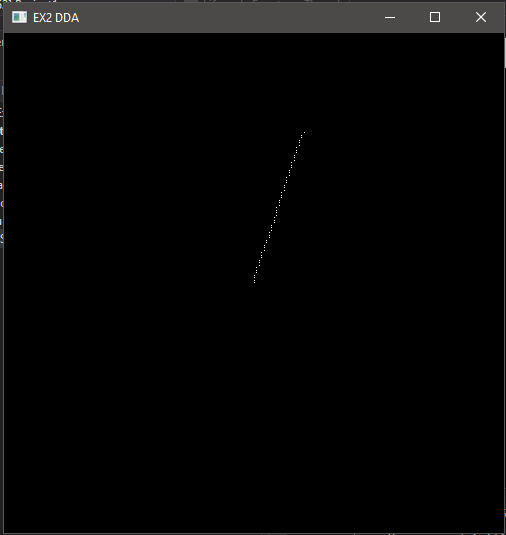
    init();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}



Case 3: -ve slope Left to Right line

* |m|<= 1

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<cstdlib>

void display(void)

{

    float dy, dx, step, x, y, k, Xin, Yin;

    float x1 = -60;

    float y1 = 40;

    float x2 = 0;

    float y2 = 0;

    dx = x2 - x1;

    dy = y2 - y1;

    if (abs(dx) > abs(dy))

    {

        step = abs(dx);

    }

    else

        step = abs(dy);

    Xin = dx / step;

    Yin = dy / step;

    x = x1;

    y = y1;

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

    for (k = 1; k <= step; k++)

    {

        x = x + Xin;

        y = y + Yin;

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void init(void)

{

    glClearColor(0.7, 0.7, 0.7, 0.7);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-100, 100, -100, 100);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("EX2 DDA");

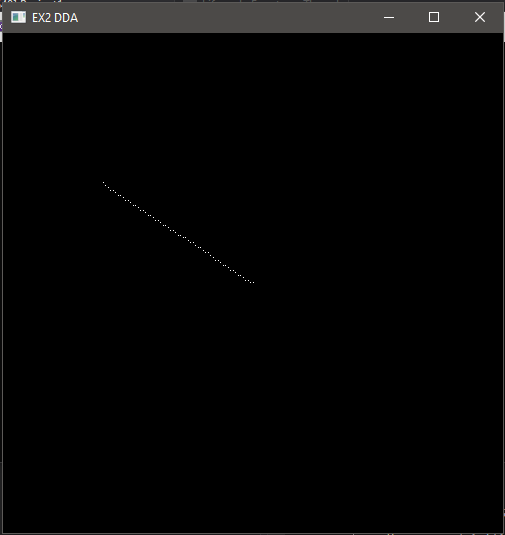
    init();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}



* |m|>1

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<cstdlib>

void display(void)

{

    float dy, dx, step, x, y, k, Xin, Yin;

    float x1 = -40;

    float y1 = 60;

    float x2 = 0;

    float y2 = 0;

    dx = x2 - x1;

    dy = y2 - y1;

    if (abs(dx) > abs(dy))

    {

        step = abs(dx);

    }

    else

        step = abs(dy);

    Xin = dx / step;

    Yin = dy / step;

    x = x1;

    y = y1;

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

    for (k = 1; k <= step; k++)

    {

        x = x + Xin;

        y = y + Yin;

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void init(void)

{

    glClearColor(0.7, 0.7, 0.7, 0.7);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-100, 100, -100, 100);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("EX2 DDA");

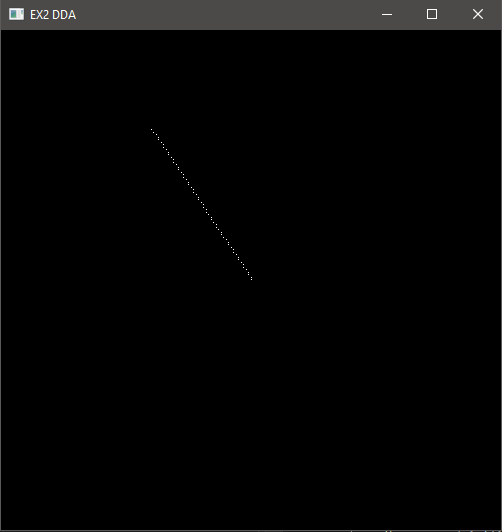
    init();

    glutDisplayFunc(display);

    glutMainLoop();

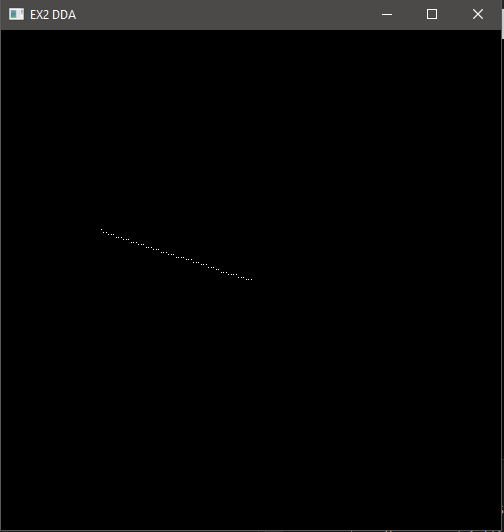
    return 0;

}



Case 4: -ve slope Right to Left line

* |m|<= 1
* #include<GL/glut.h>
* #include<stdlib.h>
* #include<stdio.h>
* #include<cstdlib>
* void display(void)
* {
* float dy, dx, step, x, y, k, Xin, Yin;
* float x1 = 0;
* float y1 = 0;
* float x2 = -60;
* float y2 = 20;
* dx = x2 - x1;
* dy = y2 - y1;
* if (abs(dx) > abs(dy))
* {
* step = abs(dx);
* }
* else
* step = abs(dy);
* Xin = dx / step;
* Yin = dy / step;
* x = x1;
* y = y1;
* glBegin(GL\_POINTS);
* glVertex2i(x, y);
* glEnd();
* for (k = 1; k <= step; k++)
* {
* x = x + Xin;
* y = y + Yin;
* glBegin(GL\_POINTS);
* glVertex2i(x, y);
* glEnd();
* }
* glFlush();
* }
* void init(void)
* {
* glClearColor(0.7, 0.7, 0.7, 0.7);
* glMatrixMode(GL\_PROJECTION);
* glLoadIdentity();
* gluOrtho2D(-100, 100, -100, 100);
* }
* int main(int argc, char\*\* argv) {
* glutInit(&argc, argv);
* glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);
* glutInitWindowSize(500, 500);
* glutInitWindowPosition(100, 100);
* glutCreateWindow("EX2 DDA");
* init();
* glutDisplayFunc(display);
* glutMainLoop();
* return 0;
* }



* |m|>1

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<cstdlib>

void display(void)

{

    float dy, dx, step, x, y, k, Xin, Yin;

    float x1 = 0;

    float y1 = 0;

    float x2 = -20;

    float y2 = 60;

    dx = x2 - x1;

    dy = y2 - y1;

    if (abs(dx) > abs(dy))

    {

        step = abs(dx);

    }

    else

        step = abs(dy);

    Xin = dx / step;

    Yin = dy / step;

    x = x1;

    y = y1;

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

    for (k = 1; k <= step; k++)

    {

        x = x + Xin;

        y = y + Yin;

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void init(void)

{

    glClearColor(0.7, 0.7, 0.7, 0.7);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-100, 100, -100, 100);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("EX2 DDA");

    init();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

