CS352 Assignment-2

Krishanu Saini
190001029

Write a program to draw a line joining two endpoints given by the user by Implementing the following line drawing algorithms:

- Digital Differential Analyzer (DDA)
- Polynomial Method

Problem 1: DDA

Solution

DDA algorithm is used to draw a line between 2 points.
If m < 1 then increment x by 1, then increment y = y + m.
If m > 1 then increment y by 1, then increment x = x + 1/m.

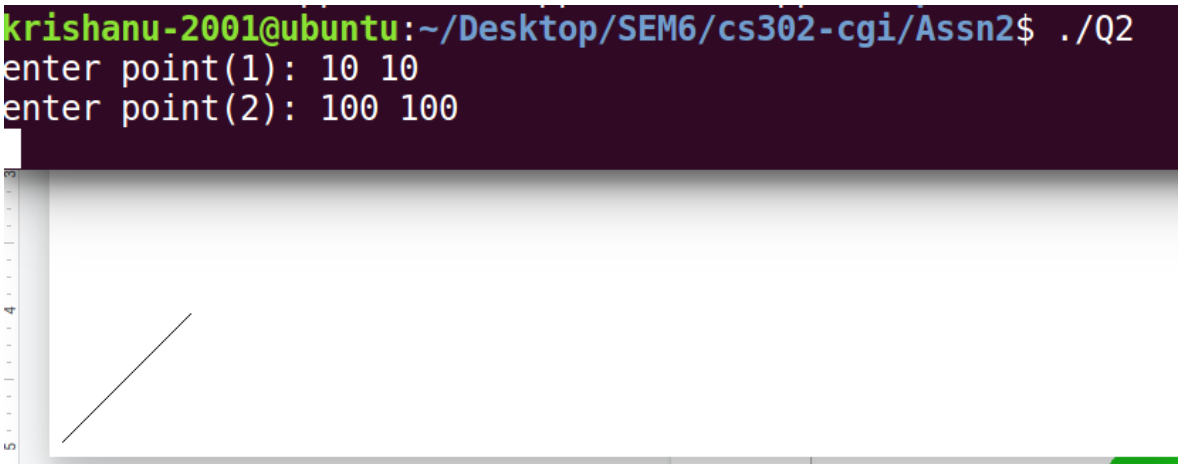
Code

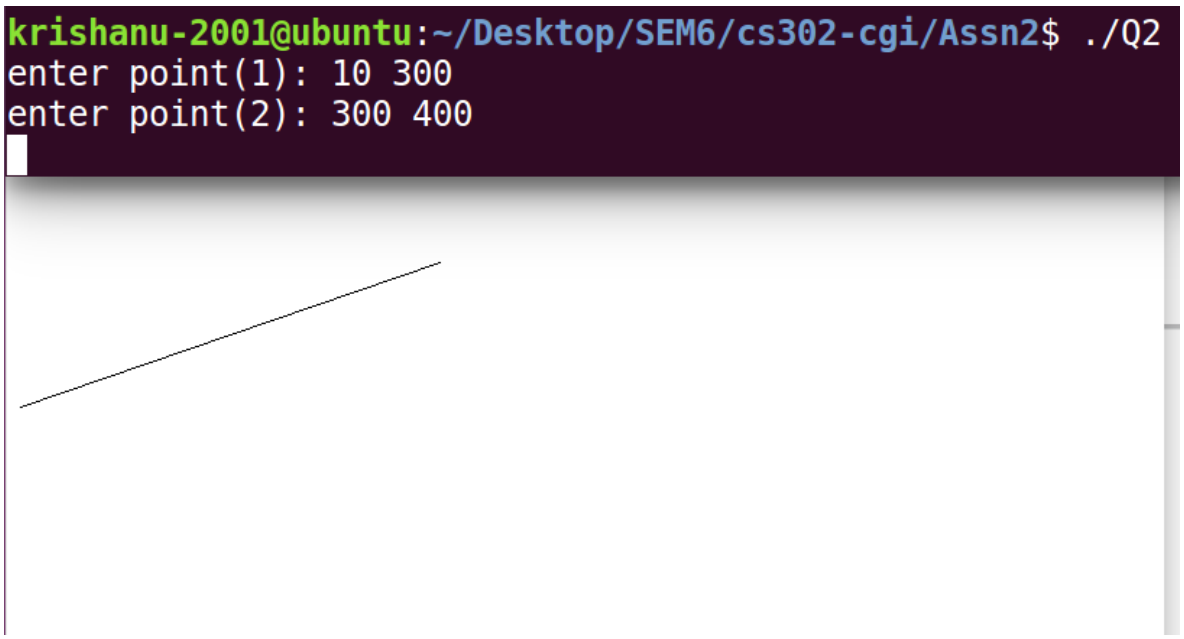
```
#include <math.h>
#include <GL/glut.h>
#include <iostream>
#include <cassert>
#define debug(x,y) cout << #x << " " << x << " " << #y << " " << y << "\n";

using namespace std;

/*
 * Name: Krishanu Saini
 * Roll: 190001029
 * Ques: Problem 2) DDA Method
 * Date: 17/01/22
 */

float xcord1,ycord1;
float xcord2,ycord2;

void DDA(float x1, float y1, float x2, float y2) {
   glColor3f(0,0,0);

   float xi = x1;
   float yi = y1;

   float delx = (x2 - x1);
   float dely = (y2 -y1);

   int totalSteps = 0;
```

```
    totalSteps = delx;
    if(abs(delx) < abs(dely)) {
        totalSteps = dely;
    }

    float dx = delx / totalSteps;
    float dy = dely / totalSteps;

    glBegin(GL_POINTS);
    glVertex2i((int)xi, (int)yi);

    for(int i=1;i<=totalSteps;i++) {
        xi = xi + dx;
        yi = yi + dy;
        glVertex2i((int)round(xi), (int)round(yi));
    }

    glEnd();
}

void Draw() {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(1);

    DDA(xcord1,ycord1,xcord2,ycord2);

    glFlush();
}

int main(int argc, char *argv[]) {
    printf("enter point(1): ");
    cin >> xcord1 >> ycord1;
    printf("enter point(2): ");
    cin >> xcord2 >> ycord2;
    if(xcord1 > xcord2) swap(xcord1, xcord2), swap(ycord1, ycord2);
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(800, 800);
    glutInitDisplayMode(GLUT_RGB);

    glutCreateWindow("DDA Model");
    gluOrtho2D(0, 800, 0, 800);
    glClearColor(1,1,1,0.0);
```

```
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

Output





Problem 2: Polygon

Solution

Used to draw a line. If m < 1 at (xi, yi) then increment x by 1. Calculate y = m*x + b.
m and b are calculated.

Code

```
#include <math.h>
#include <GL/glut.h>
#include <iostream>
#include <cassert>
#define debug(x,y) cout << #x << " " << x << " " << #y << " " << y << "\n";

using namespace std;

/*
 * Name: Krishanu Saini
 * Roll: 190001029
 * Ques: Problem 1) Polynomial Method
 * Date: 17/01/22
 */

float xcord1,ycord1;
float xcord2,ycord2;

void PolynomialMethod(float x1, float y1, float x2, float y2) {
   glColor3f(0,0,0);

   float xi = x1;
   float yi = y1;

   float delx = (x2 - x1);
   float dely = (y2 -y1);

   int totalSteps = 0;

   totalSteps = delx;
```

```
    if(abs(delx) < abs(dely)) {
        totalSteps = dely;
    }

    glBegin(GL_POINTS);
    glVertex2i((int)xi, (int)yi);
    for(int i=1;i<=totalSteps;i++) {
        float xinext, yinext;
        float b = ((y1*x2 - y2*x1) / (x2 - x1));
        if(x2 == x1) {
            int m = 1;
            if(dely < 0) m = -1;
            yinext = yi + m; // can increase or decrease
            xinext = xi;
        } else {
            float m = (y2 - y1) / (x2 - x1);
            xinext = xi + 1;
            yinext = (m * xinext) + b; // can increase or decrease
            if(abs(delx) < abs(dely)) {
                int c = 1;
                if(dely < 0) c = -1;
                yinext = yi + c; // can increase or decrease
                xinext = (yinext - b) / m;
            }
        }
        xi = xinext;
        yi = yinext;
        glVertex2i((int)round(xi), (int)round(yi));
    }

    glEnd();
}

void Draw() {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(1);

    PolynomialMethod(xcord1,ycord1,xcord2,ycord2);

    glFlush();
}

int main(int argc, char *argv[]) {
    printf("enter point(1): ");
```

```
    cin >> xcord1 >> ycord1;
    printf("enter point(2): ");
    cin >> xcord2 >> ycord2;
    if(xcord1 > xcord2) swap(xcord1, xcord2), swap(ycord1, ycord2);
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(800, 800);
    glutInitDisplayMode(GLUT_RGB);

    glutCreateWindow("Polynomial Model");
    gluOrtho2D(0, 800, 0, 800);
    glClearColor(1,1,1,0.0);
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

Output

Write a program for the previous question with the following conditions:

if |m| > 1, Increment x by 1 and compute y

if |m| < 1, Increment y by 1 and compute x

Solution

We write the polygon algorithm with changed formula

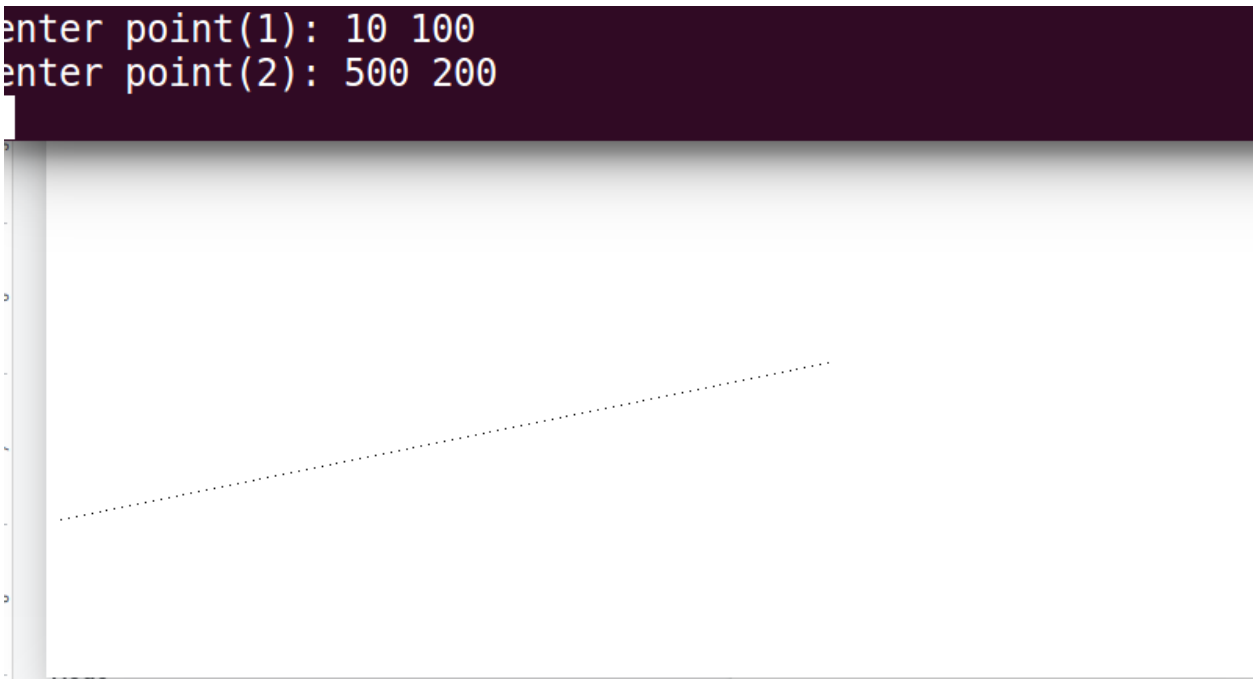
Code

```
#include <math.h>
#include <GL/glut.h>
#include <iostream>
#include <cassert>
#define debug(x,y) cout << #x << " " << x << " " << #y << " " << y << "\n";

using namespace std;

/*
 * Name: Krishanu Saini
 * Roll: 190001029
 * Ques: Problem 3) Polynomial Method Defective
 * Date: 17/01/22
 */

float xcord1,ycord1;
float xcord2,ycord2;

void PolynomialMethod(float x1, float y1, float x2, float y2) {
    glColor3f(0,0,0);

    float xi = x1;
    float yi = y1;

    float delx = (x2 - x1);
    float dely = (y2 -y1);

    int totalSteps = 0;

    totalSteps = delx;
    if(abs(delx) > abs(dely)) { // changes done here <
        totalSteps = dely;
    }

    glBegin(GL_POINTS);
    glVertex2i((int)xi, (int)yi);
    for(int i=1;i<=totalSteps;i++) {
        float xinext, yinext;
        float b = ((y1*x2 - y2*x1) / (x2 - x1));
        if(x2 == x1) {
            int m = 1;
            if(dely < 0) m = -1;
```

```
            yinext = yi + m; // can increase or decrease
            xinext = xi;
        } else {
            float m = (y2 - y1) / (x2 - x1);
            xinext = xi + 1; // if |m| > 1 then increment x (defective)
            yinext = (m * xinext) + b; // can increase or decrease
            if(abs(delx) > abs(dely)) { // if |m| < 1 then increment y
                yinext = yi + 1;
                xinext = (yinext - b) / m;
            }
        }
        xi = xinext;
        yi = yinext;
        glVertex2i((int)round(xi), (int)round(yi));
    }

    glEnd();
}
/*
see example
10 100
100 110
-- the whole graph is broken up --
*/

void Draw() {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(1);

    PolynomialMethod(xcord1,ycord1,xcord2,ycord2);

    glFlush();
}

int main(int argc, char *argv[]) {
    printf("enter point(1): ");
    cin >> xcord1 >> ycord1;
    printf("enter point(2): ");
    cin >> xcord2 >> ycord2;
    if(xcord1 > xcord2) swap(xcord1, xcord2), swap(ycord1, ycord2);
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(800, 800);
    glutInitDisplayMode(GLUT_RGB);
```

```
    glutCreateWindow("Polynomial Model");
    gluOrtho2D(0, 800, 0, 800);
    glClearColor(1,1,1,0.0);
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

Output



We can see dotted lines