**Batch: D - 1 Roll No.: 16010122096**

**Experiment No. 03**

|  |
| --- |
| **Title:** Drawing of line using computer graphics. |

**Aim:** Generate the line using computer graphics program

Objectives:

1. Visit the Following link and perform the Vlab Experiment and provide the screenshots.

<https://cse18-iiith.vlabs.ac.in/exp/coordinate-systems/pretest.html>

1. Implement the Digital Differential Analyser (DDA) Line Drawing Algorithm.
2. Implement Bresenham Line Drawing Algorithm

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1: Understand the basic concepts of computer graphics and OpenGL**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

**OpenGL, Google**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Screenshots from VLab**

**Understanding from VLab:**

**Algorithm 1: DDA**

**Step 1:** Start.

**Step 2:** We consider Starting point as (x1 , y1 ), and ending point (x2 , y2 ).

**Step 3:** Now, we have to calculate dx and dy. dx = x2 -x1 dy = y2 -y1 m = dy/dx

**Step 4:** Now, we calculate three cases.

* If m < 1 Then x change in Unit Interval y moves with deviation (xk+1, yk+1) = (xk+1, yk+m).
* If m > 1 Then x moves with deviation y change in Unit Interval (xk+1, yk+1) = (xk+1/m, yk+1).
* If m = 1 Then x moves in Unit Interval y moves in Unit Interval (xk+1, yk+1) = (xk+1, yk+1).

**Step 5:** We will repeat step 4 until we find the ending point of the line.

**Step 6:** Stop.

**Algorithm 2: Bresenhams**

**Step 1:** Start.

**Step 2:** Now, we consider Starting point as (x1 , y1 ) and endingpoint (x2 , y2 ).

**Step 3:** Now, we have to calculate ▲x and ▲y.

▲x = x2 - x1 ▲y = y2 - y1 m = ▲y /▲x

**Step 4:** Now, we will calculate the decision parameter pk with following formula. pk = 2 \* ▲y - ▲x

**Step 5:** Theinitial coordinates of the line are (xk , yk ), and the next coordinatesare (xk + 1, yk+1). Now, we are going to calculate two cases for decision parameter pk.

**Case 1:** If pk < 0 Then pk+1 =pk +2▲y xk+1 = xk +1 yk+1 = yk

**Case 2:** If pk >= 0 Then pk+1 =pk +2▲y-2▲x xk+1 =xk +1 yk+1 =yk +1

**Step 6:** We will repeat step 5 until we found the ending point of the line and the total number of iterations =▲x-1.

**Step 7:** Stop.

**Implementation details (Code can be in C/C++/Java/Python with and without using graphics library functions):**

DDA:

#include <bits/stdc++.h>

#include <GL/gl.h>

#include <GL/glut.h>

using namespace std;

float x\_1, x\_2, y\_1, y\_2;

void display(void) {

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

  dx = x\_2 - x\_1;

  dy = y\_2 - y\_1;

  m = abs(dy) / abs(dx);

  if (abs(dx) > abs(dy)) {

    step = abs(dx);

  } else

    step = abs(dy);

  x = x\_1;

  y = y\_1;

  glBegin(GL\_POINTS);

  glVertex2i(x, y);

  glEnd();

  if(m < 1) {

    Xin = 1;

    Yin = m;

    for (k = 1; k <= step; k++) {

      x = x + Xin;

      y = y + Yin;

      glBegin(GL\_POINTS);

      glVertex2i(x, y);

      glEnd();

    }

  }

  else if(m > 1)

  {

    Xin = m;

    Yin = 1;

    for (k = 1; k <= step; k++) {

      x = x + Xin;

      y = y + Yin;

      glBegin(GL\_POINTS);

      glVertex2i(x, y);

      glEnd();

    }

  }

  else{

    Xin = 1;

    Yin = 1;

    for (k = 1; k <= step; k++) {

      x = x + Xin;

      y = y + Yin;

      glBegin(GL\_POINTS);

      glVertex2i(x, y);

      glEnd();

    }

  }

  glFlush();

}

void myInit() {

    glClearColor(0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0.0, 30.0, 0.0, 30.0);

}

int main(int argc, char\*\* argv) {

    cout << "Value of x1: ";

    cin >> x\_1;

    cout << "Value of y1: ";

    cin >> y\_1;

    cout << "Value of x2: ";

    cin >> x\_2;

    cout << "Value of y2: ";

    cin >> y\_2;

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(640, 480);

    glutInitWindowPosition(100, 150);

    glutCreateWindow("DDA Line Drawing");

    myInit();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

Bresenhams:

#include <iostream>

#include <GL/gl.h>

#include <GL/glut.h>

using namespace std;

int x\_1, y\_1, x\_2, y\_2;

void drawPixel(int x, int y) {

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

}

void bresenhamLine() {

    int dx = abs(x\_2 - x\_1);

    int dy = abs(y\_2 - y\_1);

    int p = 2 \* dy - dx;

    int twoDy = 2 \* dy;

    int twoDyMinusDx = 2 \* (dy - dx);

    int x, y;

    if (x\_1 > x\_2) {

        x = x\_2;

        y = y\_2;

        x\_2 = x\_1;

    } else {

        x = x\_1;

        y = y\_1;

    }

    drawPixel(x, y);

    while (x < x\_2) {

        x++;

        if (p < 0) {

            p += twoDy;

        } else {

            y++;

            p += twoDyMinusDx;

        }

        drawPixel(x, y);

    }

    glFlush();

}

void myInit() {

    glClearColor(0.0, 0.0, 0.0, 0.0);

    glColor3f(1.0, 1.0, 1.0);

    glPointSize(2.0);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0.0, 30.0, 0.0, 30.0);

}

int main(int argc, char\*\* argv) {

    cout << "Enter the value of x1: ";

    cin >> x\_1;

    cout << "Enter the value of y1: ";

    cin >> y\_1;

    cout << "Enter the value of x2: ";

    cin >> x\_2;

    cout << "Enter the value of y2: ";

    cin >> y\_2;

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 150);

    glutCreateWindow("Bresenham's Line Drawing");

    myInit();

    glutDisplayFunc(bresenhamLine);

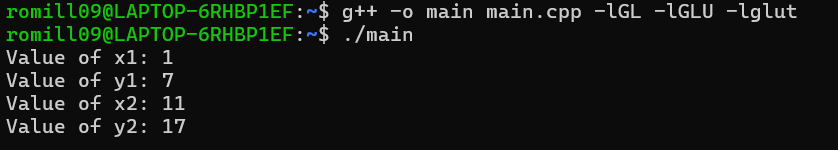
    glutMainLoop();

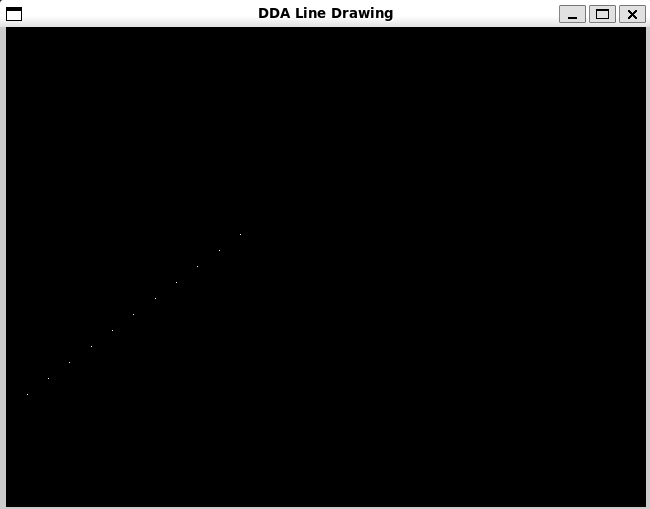
    return 0;

}

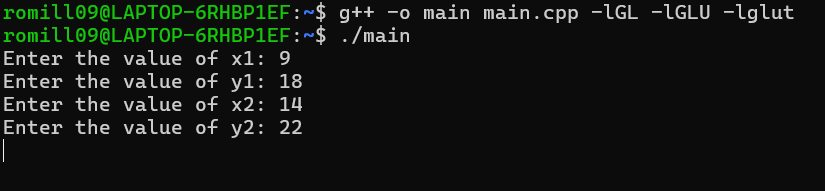
**Output(s) (final edited screen shot):**

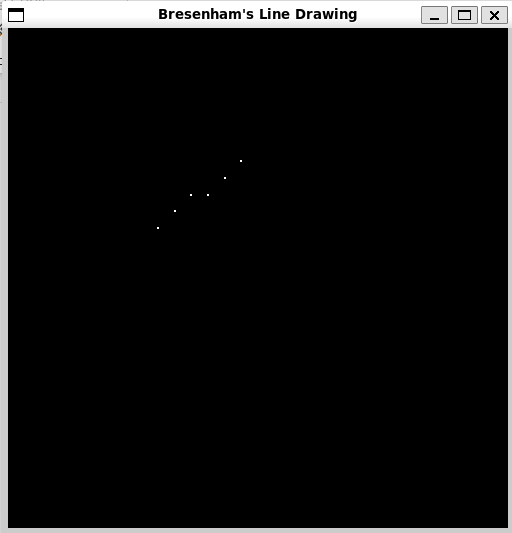
**DDA:**

****

****

**Bresenhams:**

****

****

**Conclusion and discussion (Comparative - compulsory):**

In comparing DD and Bresenham’s line drawing algorithms, DDA is simpler and straightforward but suffers from floating point arithmetic issues, leading to less precision and slower performance. Bresenham’s algorithm, on the other hand, is more efficient and precise, using integer arithmetic to avoid rounding errors. It handles all slopes consistently and is generally preferred for its speed and accuracy in raster graphics.

**Date: 17 / 08 / 2024 Signature of faculty in-charge**

**Explain Mid-point line drawing algorithm and implement it**

The Midpoint Line Drawing algorithm in OpenGL is an efficient way to draw a line between two points. It calculates the midpoint between two potential pixel positions, choosing the one closer to the line path. This decision is made using an error term that is adjusted at each step. The algorithm is efficient as it uses only integer addition, subtraction, and bit shifting for calculations, making it suitable for real-time rendering.

**Algorithm:**

**Given-**

* Starting coordinates = (X0, Y0)
* Ending coordinates = (Xn, Yn)

The points generation using Mid Point Line Drawing Algorithm involves the following steps-

**Step-01:**

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

* ΔX = Xn – X0
* ΔY =Yn – Y0

**Step-02:**

Calculate the value of initial decision parameter and ΔD.

These parameters are calculated as-

* Dinitial = 2ΔY – ΔX
* ΔD = 2(ΔY – ΔX)

**Step-03:**

The decision whether to increment X or Y coordinate depends upon the flowing values of Dinitial.

Follow the below two cases-

****

**Step-04:**

Keep repeating Step-03 until the end point is reached.

For each Dnew value, follow the above cases to find the next coordinates.

**Implementation:**

#include <bits/stdc++.h>

#include <GL/gl.h>

#include <GL/glut.h>

using namespace std;

float x\_1, x\_2, y\_1, y\_2;

void display(void) {

    int dx = x\_2 - x\_1;

    int dy = y\_2 - y\_1;

    int D = 2 \* dy - dx;

    int y = y\_1;

    glBegin(GL\_POINTS);

    glVertex2i(x\_1, y\_1);

    glEnd();

    for (int x = x\_1 + 1; x <= x\_2; x++) {

        if (D >= 0) {

            y++;

            D += 2 \* (dy - dx);

        } else {

            D += 2 \* dy;

        }

        glBegin(GL\_POINTS);

        glVertex2i(x, y);

        glEnd();

    }

    glFlush();

}

void myInit() {

    glClearColor(0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0.0, 40.0, 0.0, 40.0);

}

int main(int argc, char\*\* argv) {

    cout << "Value of x1: ";

    cin >> x\_1;

    cout << "Value of y1: ";

    cin >> y\_1;

    cout << "Value of x2: ";

    cin >> x\_2;

    cout << "Value of y2: ";

    cin >> y\_2;

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(640, 480);

    glutInitWindowPosition(100, 150);

    glutCreateWindow("Midpoint Line Drawing");

    myInit();

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

**Output:**

