Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

Lab Report 2

[Code No:COMP 314]

Submitted by

Jiwan Humagain(19)

Submitted to

Mr. Dhiraj Shrestha
Assistant Professor
Department of Computer Science and Engineering

Submission Date: 2024-05-10

1. Implement Digital Differential Analyzer Line drawing algorithm.

a. Title of lab activity

To Implement the DDA Line drawing algorithm.

b. Algorithm used for drawing line

Here is the algorithm used for drawing the line.

I. Input the two endpoints of the line segment, (x1,y1) and (x2,y2).

II. Calculate the difference between the x-coordinates and y-coordinates of the endpoints as dx and dy respectively.

III. Calculate the slope of the line as m = dy/dx.

IV. Set the initial point of the line as (x1,y1).

V. Loop through the x-coordinates of the line, incrementing by one each time, and calculate the corresponding y-coordinate using the equation y = y1 + m(x – x1).

VI. Plot the pixel at the calculated (x,y) coordinate.

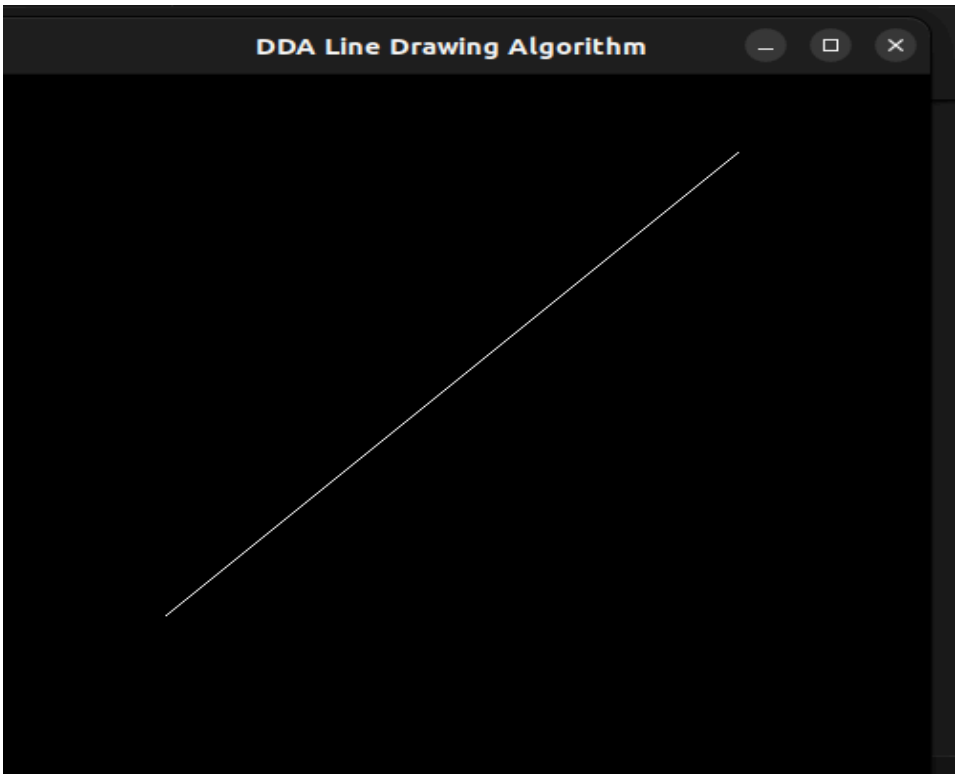
VII. Repeat steps 5 and 6 until the endpoint (x2,y2) is reached.

c. Source Code

```
#include <GL/glut.h>
#include <iostream>
#include <cmath>

void init() {
    glClearColor(0.0, 0.0, 0.0, 0.0); // Set background color to white
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 500, 0, 500);
}

void drawVertex(int x, int y) {
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}

void DDA(int x1, int y1, int x2, int y2)
{
    int dx = x2 - x1;
    int dy = y2 - y1;
    int m = dy / dx;
    int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);

    float Xinc = dx / (float)steps;
    float Yinc = dy / (float)steps;
    float array_x_coordinates[steps];
    float array_y_coordinates[steps];
    array_x_coordinates[0] = x1;
```

```
    array_y_coordinates[0] = y1;
    for (int i = 1; i < steps; i++) {
        array_x_coordinates[i]=round(array_x_coordinates[i-1]+Xinc) ;
        array_y_coordinates[i]=round(array_y_coordinates[i-1]+Yinc);
    }
    glColor3f(1.0, 1.0, 1.0); // Set line color to black
    for (int i = 0; i < steps; i++)
    {
        drawVertex(array_x_coordinates[i],array_y_coordinates[i]);
    }
    glFlush();
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    DDA(100,150,400,450);
    glFlush();
}
int main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("DDA Line Drawing Algorithm");
    init();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

d. OutPut



2. Implement Bresenham Line Drawing algorithm for both slopes(|m|<1 and |m|>=1).

a. Title of lab activity

To Implement the BLA Line drawing algorithm.

b. Algorithm used for drawing line

Step1: Declare variable $x_1,x_2,y_1,y_2,d,i_1,i_2,dx,dy$

Step2: Enter value of x_1,y_1,x_2,y_2

Where $x_1,y_1$are coordinates of starting point

And x_2,y_2 are coordinates of Ending point

Step3: Calculate $dx = x_2-x_1$

Calculate $dy = y_2-y_1$

Calculate $i_1=2*dy$

Calculate $i_2=2*(dy-dx)$

Calculate $d=i_1-dx$

Step4: Consider (x, y) as starting point and x_{end}as maximum possible value of x.

If $dx < 0$

Then $x = x_2$

$y = y_2$

$x_{end}=x_1$

If $dx > 0$

Then $x = x_1$

$y = y_1$

$x_{end}=x_2$

Step5: Generate points at (x,y)coordinates.

Step6: Check if the whole line is generated.

If $x >= x_{end}$

Stop.

Step7: Calculate coordinates of the next pixel

If $d < 0$

Then $d = d + i_1$

If $d \geq 0$

Then $d = d + i_2$

Increment $y = y + 1$

Step8: Increment $x = x + 1$

Step9: Draw a point of latest (x, y) coordinates

Step10: Go to step 6

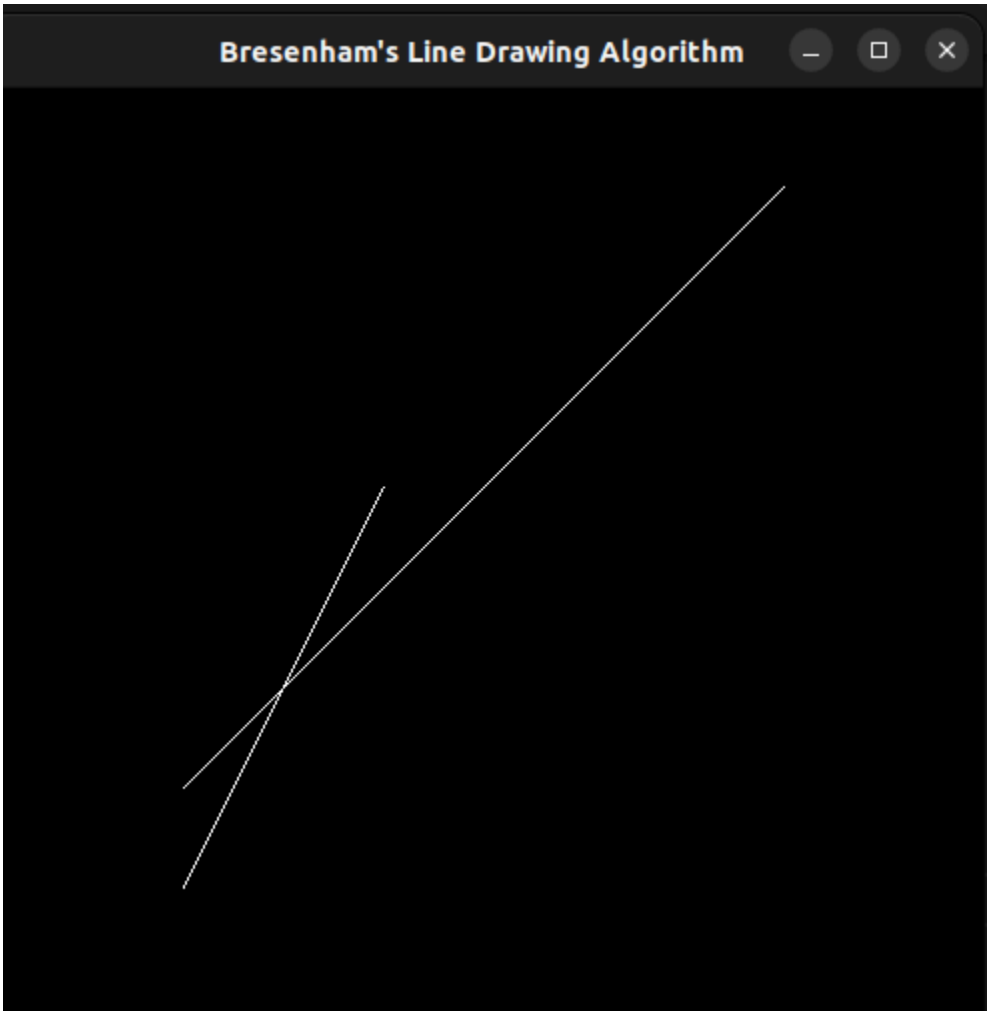
c. Source Code

```
#include <GL/glut.h>
#include <iostream>
#include <cmath>

void init() {
    glClearColor(0.0, 0.0, 0.0, 0.0); // Set background color to black
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 500, 0, 500);
}

void drawVertex(int x, int y) {
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}

void bresenham(int x1, int y1, int x2, int y2) {
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int sx = (x1 < x2) ? 1 : -1;
    int sy = (y1 < y2) ? 1 : -1;

    int err = dx - dy;
    int x = x1;
```

```
    int y = y1;

    while (true) {
        drawVertex(x, y);

        if (x == x2 && y == y2)
            break;

        int e2 = 2 * err;
        if (e2 > -dy) {
            err -= dy;
            x += sx;
        }
        if (e2 < dx) {
            err += dx;
            y += sy;
        }
    }

    glFlush();
}
```

```
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0); // Set line color to white

    // Example line with slope > 1
    bresenham(100, 100, 200, 300);

    // Example line with slope < 1
    bresenham(100, 150, 400, 450);

    glFlush();
}

int main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Bresenham's Line Drawing Algorithm");
    init();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

d. OutPut



3. Implement the given line drawing algorithm to draw a line histogram for any given frequency inputs

a. Title of lab activity

To Draw a line histogram using the BLA Line drawing algorithm.

b. Source Code

```
#include <GL/glut.h>
#include <iostream>
#include <vector>
using namespace std;
// Define the window dimensions
const int WIDTH = 800;
const int HEIGHT = 600;

// Define the maximum frequency value
const int MAX_FREQUENCY = 100;

// Input frequency values (sample data)
std::vector<int> frequencies = {20, 15, 30, 20,20,20};
```

```
15  void init()
16  {
17      glClearColor(1.0, 1.0, 1.0, 1.0); // Set background color to white
18      glMatrixMode(GL_PROJECTION);
19      glLoadIdentity();
20      gluOrtho2D(0, WIDTH, 0, HEIGHT);
21  }
22
23  void drawPixel(int x, int y)
24  {
25      glBegin(GL_POINTS);
26      glVertex2i(x, y);
27      glEnd();
28  }
```

```
30  void bresenhamLine(int x1, int y1, int x2, int y2)
31  {
32      int dx = abs(x2 - x1);
33      int dy = abs(y2 - y1);
34      int sx = (x1 < x2) ? 1 : -1;
35      int sy = (y1 < y2) ? 1 : -1;
36      int err = dx - dy;
37      int x = x1;
38      int y = y1;
39      while (true)
40      {
41          drawPixel(x, y);
42
43          if (x == x2 && y == y2)
44              break;
45          int e2 = 2 * err;
46          if (e2 > -dy)
47          {
48              err -= dy;
49              x += sx;
50          }
51          if (e2 < dx)
52          {
53              err += dx;
54              y += sy;
55          }
56      }
57  }
```

```
59  void drawAxes()
60  {
61      glClear(GL_COLOR_BUFFER_BIT);
62      glColor3f(1.0, 0.5, 0.0);
63
64      // Draw x-axis
65      for (int i = 0; i < 3; ++i)
66          bresenhamLine(0, HEIGHT - 550 - i, WIDTH, HEIGHT - 550 - i);
67
68      // Draw y-axis
69      for (int i = 0; i < 3; ++i)
70          bresenhamLine(WIDTH - 750 + i, 0, WIDTH - 750 + i, HEIGHT);
71  }
```

```
void drawHistogramBar()
{
    int barWidth = WIDTH / frequencies.size();
    int startX = barWidth / 2;

    for (size_t i = 0; i < frequencies.size(); i++)
    {
        int barHeight = frequencies[i] * HEIGHT / MAX_FREQUENCY;
        if (i != 0)
        {
            int prevbarHeight = (HEIGHT / 2) + (frequencies[i - 1] * HEIGHT / MAX_FREQUENCY);

            // cout<<"PrevHeight "<<prevbarHeight<<" nextHeight"<<barHeight;
            if (prevbarHeight > barHeight)
            {
                bresenhamLine(startX, HEIGHT / 2 + barHeight, startX, prevbarHeight);
                // cout<<"Prev height "<<prevbarHeight;
            }
        }

        int tempYCoordinate = HEIGHT / 2 + barHeight;

        // Draw vertical line starting from the x-axis
        bresenhamLine(startX, HEIGHT - 550, startX, HEIGHT / 2 + barHeight);
        if (i != frequencies.size() - 1)
        {
            bresenhamLine(startX, HEIGHT / 2 + barHeight, startX + barWidth, HEIGHT / 2 + barHeight);
        }
```

```
            // cout<<"StartX"<<tempYCoordinate<<" Height"<<HEIGHT/2+barHeight;

            // Move to next bar
            startX += barWidth;
        }
    }

    void drawHistogram()
    {
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.0, 0.0, 0.0); // Set line color to black

        drawAxes(); // Draw x and y axes
        drawHistogramBar();

        glFlush();
    }
```

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(WIDTH, HEIGHT);
    glutCreateWindow("Line Histogram Drawing");
    init();
    glutDisplayFunc(drawHistogram);
    glutMainLoop();

    return 0;
}
```

c. OutPut

Conclusion

In this assignment, we effectively implemented line drawing algorithms, including DDA, Bresenham's Line Algorithm, catering to various slope conditions. While the Digital Differential Analyzer (DDA) Line Drawing Algorithm wasn't directly utilized, Bresenham's method provided was used for making histogram. We make a line histogram by dynamically adjusting line lengths based on given frequency inputs. Through these implementations, we showcased the versatility and effectiveness of Bresenham's approach, ensuring precise line rendering and histogram visualization in raster graphics applications, while demonstrating an understanding of fundamental line drawing techniques.