KATHMANDU UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB 2

"Line Drawing Algorithm"

A **Third year/ Second Semester** Computer Graphics [COMP 342]
Lab Work submitted in partial fulfillment of the requirements
for the degree of Bachelor of Engineering.

Submitted by:
Ashish Pokhrel
Faculty: C.E.
Roll: 38
Registration No: 022446-17

Submitted to:
Mr. Dhiraj Shrestha
Computer Graphics [COMP 342]
Department of Computer Science and Engineering

Aug 19, 2020

Objectives:

1. To Implement Digital Differential Analyzer Line drawing algorithm.
2. To Implement Bresenham Line Drawing algorithm for both slopes(|m|<1 and |m|>=1).
3. To Implement Midpoint algorithm Line drawing for both slopes(|m|<1 and |m|>=1).

A. To Implement Digital Differential Analyzer Line drawing algorithm.

Theory:

Line Drawing Algorithm

In computer graphics, a line drawing algorithm is an algorithm for approximating a line segment on discrete graphical media, such as pixel-based displays and printers. On such media, line drawing requires an approximation (in nontrivial cases). Basic algorithms rasterize lines in one color.

Digital Differential Analyzer

The digital differential analyzer (DDA) is a scan conversion method for line drawing which is based on incremental approach. The DDA method can be implemented using floating-point or integer arithmetic. The native floating-point implementation requires one addition and one rounding operation per interpolated value (e.g. coordinate x, y, depth, color component etc.) and output result. This process is only efficient when an FPU with fast add and rounding operation will be available.

The fixed-point integer operation requires two additions per output cycle, and in case of fractional part overflow, one additional increment and subtraction. The probability of fractional part overflows is proportional to the ratio m of the interpolated start/end values.

DDAs are well suited for hardware implementation and can be pipelined for maximized throughput.

Algorithm:

Step 1: Enter the end points of the line. Let it be (x1,y1) and (x2,y2)

Step 2: Calculate slope (m) of the line from given two points.

Step 3: if | m | <=1,

x-coordinate is incremented by 1 and y-coordinate is incremented by m.

else

x-coordinate is incremented by 1/m and x-coordinate is incremented by 1.

Step 4: Plot the points until end point is reached.

Source Code:

```
// To Implement Digital Differential Analyzer Line drawing algorithm.

function setup() {
    createCanvas(700, 750);

}
function draw() {
    background("#9b59b6");

    let x1 = 100, y1 = 100, x2 = 400, y2 = 400;

    const m = (y2 - y1) / (x2 - x1)

    fill(0, 0, 0)

    if (Math.abs(m) <= 1) {
        let i;
        beginShape();
        vertex(x1, y1);
        for (i = x1; i < x2; i++) {
            if (x1 < x2) {
                x1 = x1 + 1;
                y1 = y1 + m;
                vertex(x1, y1)
            }
        }
        vertex(x2, y2);
        endShape();
    } else if (Math.abs(m) > 1) {
        let i;
        beginShape();
        vertex(x1, y1);
        for (i = y1; i < y2; i++) {
            if (y1 < y2) {
                y1 = y1 + 1;
                x1 = x1 + 1 / m;
            }
        }
        vertex(x2, y2);
        endShape();
    }
}
```

Output:

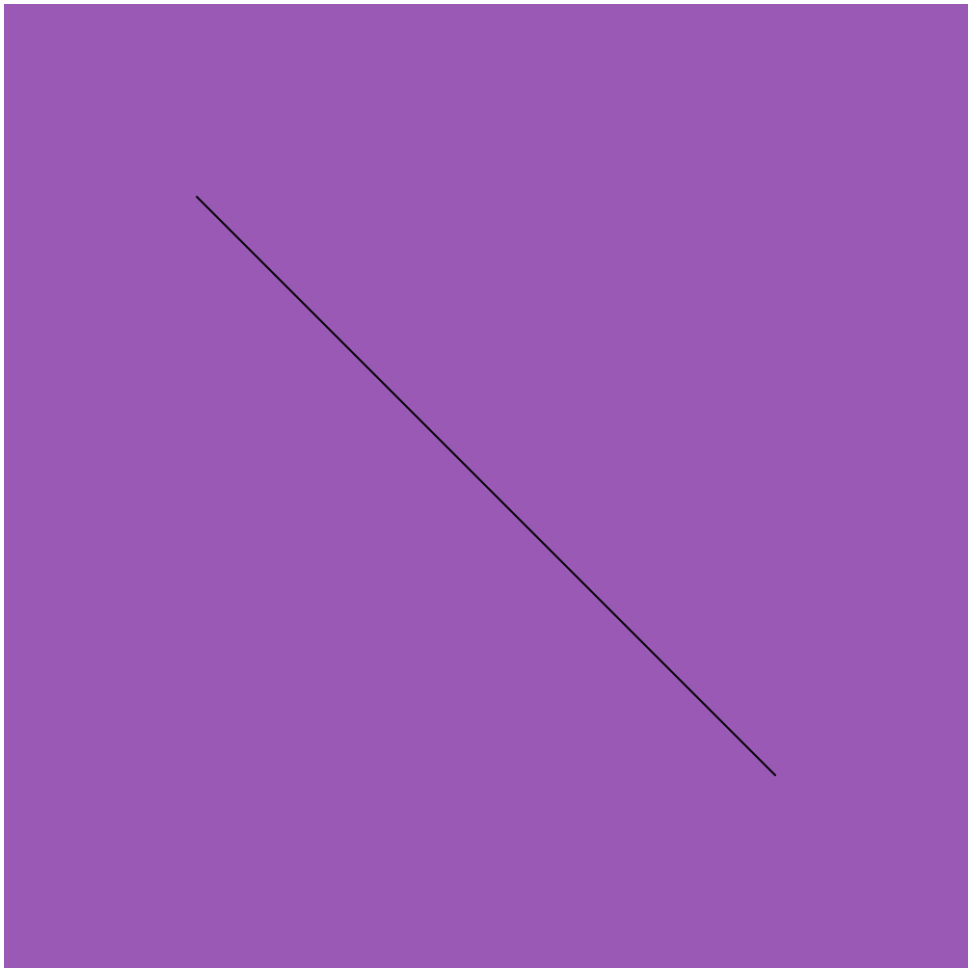

Fig: Digital Differential Analyzer Line drawing algorithm

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

Theory:

Bresenham's line algorithm is a line drawing algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

In this method, next pixel selected is that one who has the least distance from true line.

Algorithm:

Step 1: Input the two-line endpoints and store the left endpoint in (x0,y0)

Step 2: Load (x0, y0) into the frame buffer; that is, plot the first point.

If(dy/dx >=1) then change y to x and x to y.

Step 3: Calculate constants dx, dy, 2dy, and 2dy - 2dx, and obtain the starting value for the decision parameter as $p0 = 2dy – dx$

Step 4: At each xk along the line, starting at k = 0, perform the following test:

If pk < 0, the next point to plot is (xk + 1, yk) and $p_{k+1} = pk + 2dy$

Otherwise, the next point to plot is (xk + 1, yk + 1) and

$p_{k+1} = pk + 2dy – 2dx$

Step 5: Repeat step 4 dx times.

Source Code:

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

function setup() {
    createCanvas(700, 750);
}

function draw() {
    background("#9b59b6");

    let x1 = 100, y1 = 100, x2 = 400, y2 = 400;
    const dx = x2 - x1;
    const dy = y2 - y1;
    const m = dy / dx;

    fill(0, 0, 0)

    if (Math.abs(m) < 1) {
        let i;
        let p0 = 2 * dy - dx;
        beginShape();
        vertex(x1, y1);
        for (i = x1; i < x2; i++) {
            if (p0 < 0) {
                x1 += 1;
                p0 = p0 + 2 * dy
                vertex(x1, y1)
            } else {
                x1 += 1;
                y1 += 1;
                p0 = p0 + 2 * dy - 2 * dx
                vertex(x1, y1)
            }
        }
        vertex(x2, y2);
        endShape();
    } else if (Math.abs(m) >= 1) {
        let i;
        let p0 = 2 * dx - dy;
        beginShape();
        vertex(x1, y1);
        for (i = y1; i < y2; i++) {
```

```
            if (p0 < 0) {
                y1 += 1;
                p0 = p0 + 2 * dx
                vertex(x1, y1)
            } else {
                x1 += 1;
                y1 += 1;
                vertex(x1, y1)
                p0 = p0 + 2 * (dx - dy);
            }
        }
        vertex(x2, y2);
        endShape();
    }
}
```

Output:

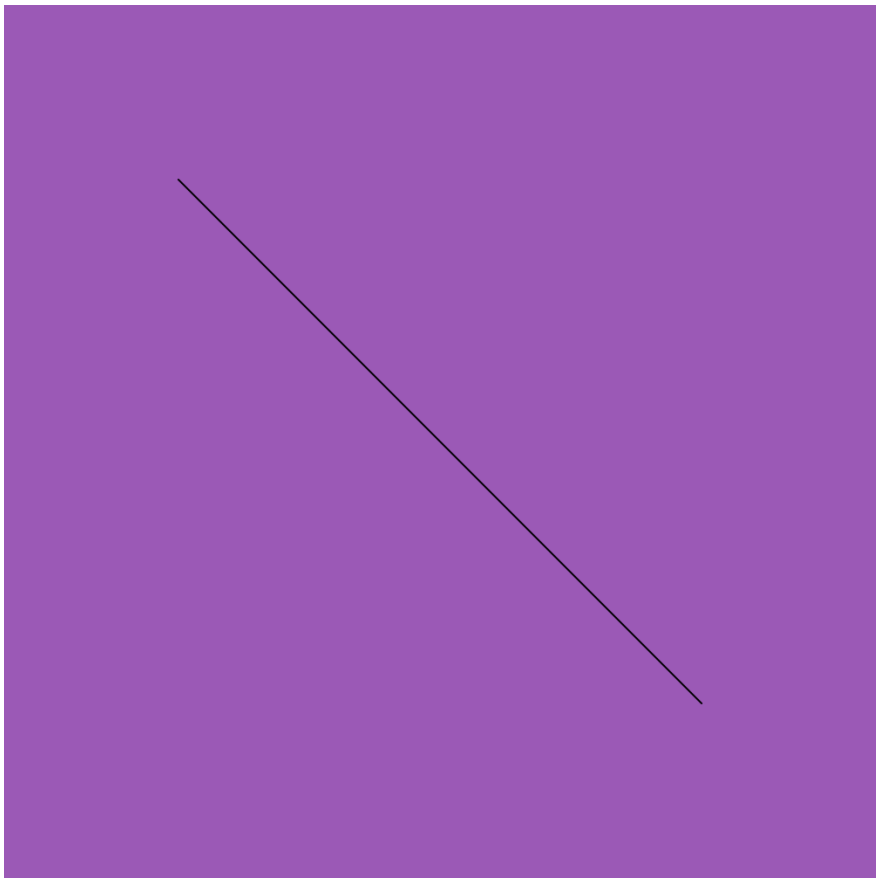

Fig: Bresenham Line Drawing Algorithm

C. To Implement Midpoint algorithm Line drawing for both slopes(|m|<1 and |m|>=1).

Theory:

Accuracy of finding points is a key feature of this Midpoint algorithm. It is simple to implement. It uses basic arithmetic operations. It takes less time for computation. The resulted line is smooth as compared to other line drawing algorithms. This algorithm may not be an ideal choice for complex graphics and images. In terms of accuracy of finding points, improvement is still needed. There is no any remarkable improvement made by this algorithm.

Algorithm:

Step 1: Find middle of two possible next points. Middle of E(Xp+1, Yp) and NE(Xp+1, Yp+1) is M(Xp+1, Yp+1/2).

Step 2: If M is above the line, then choose E as next point.

Step 3: If M is below the line, then choose NE as next point.

For this, we need decision parameter, d such that d=dy-dx/2

if (d<0) => d=d+dy, x++ and same y-coordinate

else d=d+(dy-dx), x++,y++,

Also if | m |>=1 swipe x and y from above step.

Source Code:

```
// To Implement Midpoint algorithm Line drawing for both slopes(|m|<1 and |m|>=1)

function setup() {
    createCanvas(700, 750);
}

function draw() {
    background("#9b59b6");

    let x1 = 100, y1 = 100, x2 = 400, y2 = 400;

    const dx = x2 - x1;
    const dy = y2 - y1;
    const m = dy / dx;

    let d = dy - (dx / 2);
    let x = x1, y = y1;

    beginShape();
    vertex(x, y);
    if (Math.abs(m) < 1) {
        while (x < x2) {
            x++;
            if (d < 0) {
                d = d + dy;
            } else {
                d += (dy - dx);
                y++;
            }
            vertex(x, y);

        }
    } else {
        while (y < y2) {
            y++;
            if (d < 0) {
                d = d + dx;
            } else {
                d += (dx - dy);
                x++;
            }
            vertex(x, y);
```

```
        }
    }

    endShape();
}
```

Output:

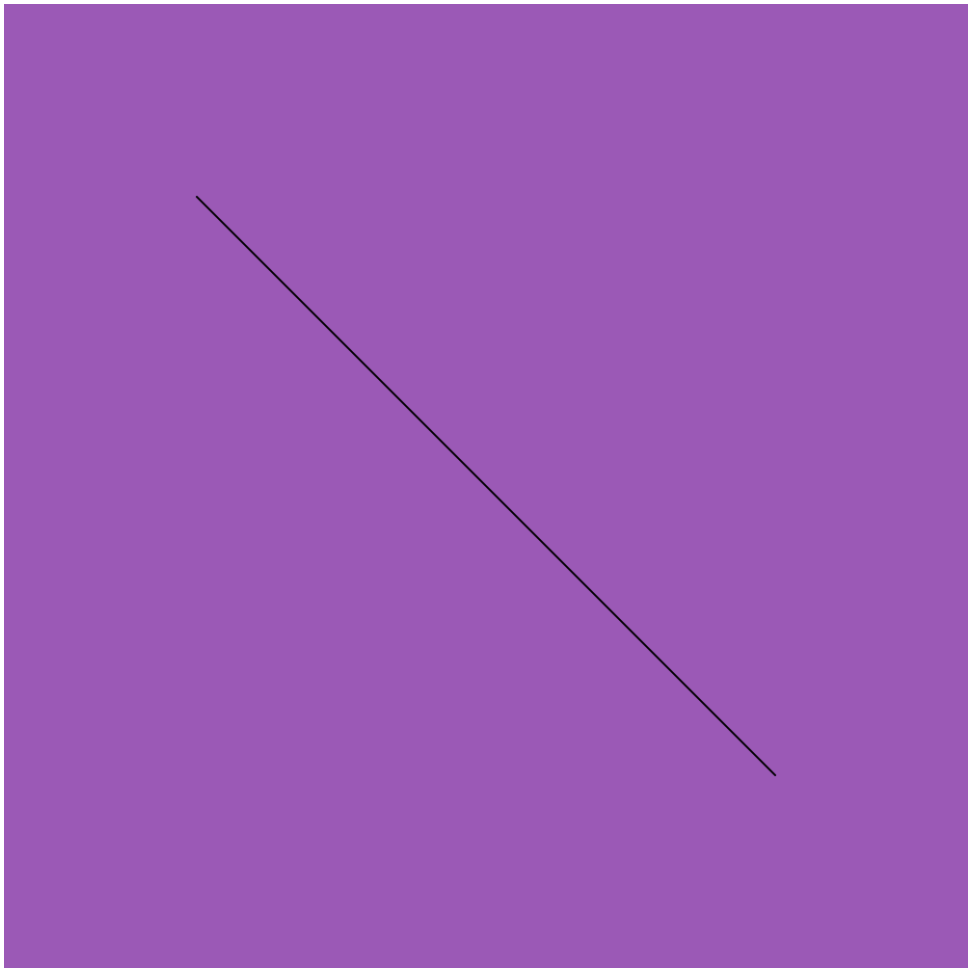

Fig: Midpoint Line drawing algorithm

Conclusion:

In this way, we used three Line Drawing Algorithms to draw a line segment from the given two points using p5 library. We obtained a line segment each form DDA Line Drawing Algorithm, Bresenham Line Drawing Algorithm and Mid Point Line Drawing Algorithm.

.