

## **PREMIER UNIVERSITY CHITTAGONG**



### **Department of Computer Science & Engineering**

Course Code : CSE-451

Course Title : Computer Graphics and Image Processing Laboratory.

Report No : 02

Report Name: Implementation of Bresenham's Line drawing algorithm.

Date of Submission : 05/02/2023

### **SUBMITTED BY**

MD: Mohiuddin Faysal
ID:1903610201765
Department : CSE
Semester: 7 <sup>th</sup>
Section: C2

### **SUBMITTED TO**

MD. NEAMUL HOQUE
Lecturer
Department of CSE

**Abstract :**

The Bresenham line drawing algorithm is a widely used technique for generating a straight line between two points on a computer screen. The algorithm works by incrementally determining the pixels that are closest to the true line using integer arithmetic operations. This approach results in faster and more accurate line drawing than the Digital Differential Analyzer (DDA) algorithm. The Bresenham algorithm can draw lines with any slope and is particularly efficient for drawing lines with a slope between 0 and 1. It is widely used in computer graphics and image processing applications due to its speed and accuracy. In this article, we will discuss the Bresenham line drawing algorithm in detail, including its principles, advantages, limitations, and applications.

**Introduction :**

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.

**Software & hardware required:**

1. Codeblocks
2. Microsoft word
3. Snipping Tool

**Algorithms:**

1. Start
2. Declare variables  $x_1, x_2, y_1, y_2, dx, dy, p, x, y$
3. Enter the value of  $x_1, y_1, x_2, y_2$
4. Calculate  $dx = x_2 - x_1, dy = y_2 - y_1, p = 2(dy - dx)$
5. Consider  $(x_1, y_1)$  as starting point and  $x_2$  as maximum possible value of  $x$

If  $dx < 0$

Then  $x = x_2$

$y = y_2$

$x_{end} = x_1$

If  $dx > 0$

Then  $x = x_1$

$y = y_1$

$x_{end} = x_2$

6. Generate point at  $(x, y)$  coordinates

7. Calculate co-ordinates of the next pixel

8. End of algorithm

**Pseudocode:**

Input:  $(x_1, y_1), (x_2, y_2)$

$dx = \text{abs}(x_2 - x_1)$

$dy = \text{abs}(y_2 - y_1)$

$sx = x_1 < x_2 ? 1 : -1$

$sy = y_1 < y_2 ? 1 : -1$

$err = dx - dy$

```

while (x1 != x2 || y1 != y2) do
Plot (x1, y1)
e2 = err * 2
if e2 > -dy then
err = err - dy
x1 = x1 + sx
end if
if e2 < dx then
err = err + dx
y1 = y1 + sy
end if
end while

```

### Flow Chart:

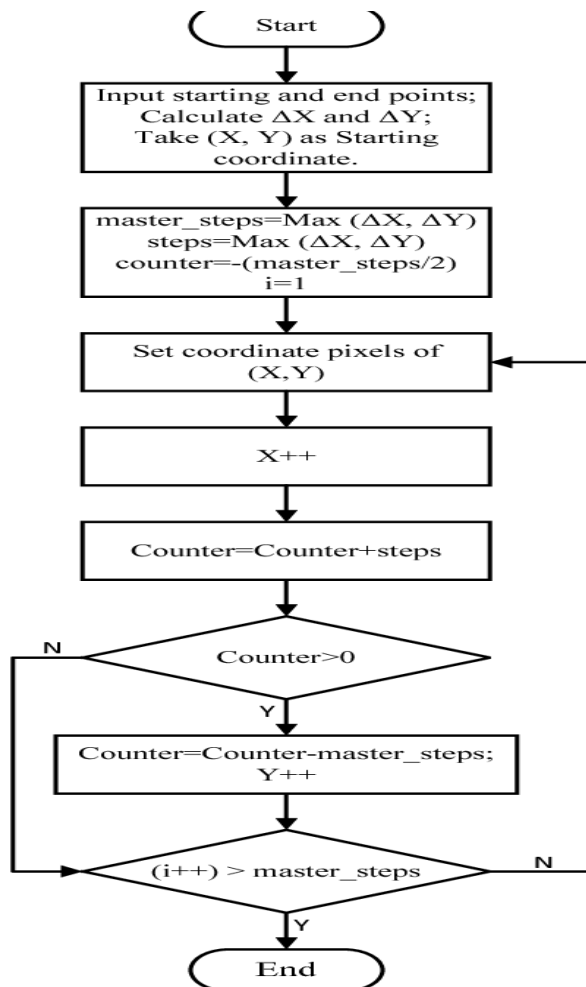


Fig. 2. Bresenham's algorithm flow chart

### Code:

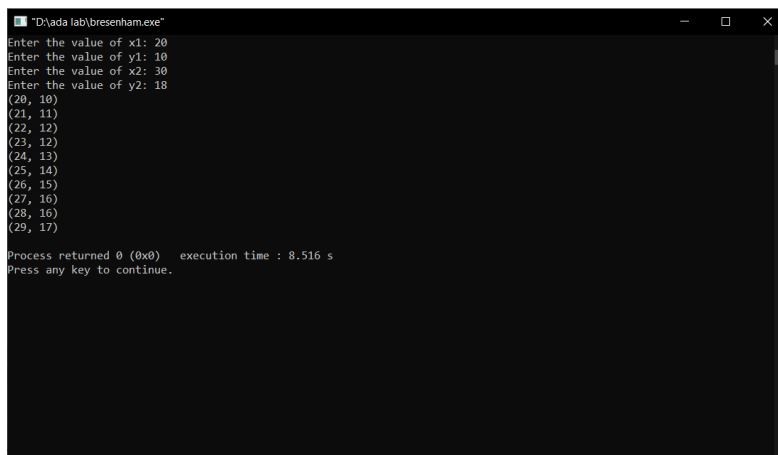
```

#include <stdio.h>
void bresenham_line(int x1, int y1, int x2, int y2) {
int dx, dy, p, x, y;

```

```
dx = x2 - x1;
dy = y2 - y1;
x = x1;
y = y1;
p = 2 * dy - dx;
while (x < x2) {
if (p >= 0) {
printf("(%d, %d)\n", x, y);
y++;
p = p + 2 * dy - 2 * dx;
} else {
printf("(%d, %d)\n", x, y);
p = p + 2 * dy;
}
x++;
}
}
int main() {
int x1, y1, x2, y2;
printf("Enter the value of x1: ");
scanf("%d", &x1);
printf("Enter the value of y1: ");
scanf("%d", &y1);
printf("Enter the value of x2: ");
scanf("%d", &x2);
printf("Enter the value of y2: ");
scanf("%d", &y2);
bresenham_line(x1, y1, x2, y2);
return 0;
}
```

### Output:



```
"D:\ada lab\bresenham.exe"
Enter the value of x1: 20
Enter the value of y1: 10
Enter the value of x2: 30
Enter the value of y2: 18
(20, 10)
(21, 11)
(22, 12)
(23, 12)
(24, 13)
(25, 14)
(26, 15)
(27, 16)
(28, 16)
(29, 17)
Process returned 0 (0x0)   execution time : 8.516 s
Press any key to continue.
```

**Limitation:**

Despite its many advantages, the Bresenham line drawing algorithm has some limitations that can affect its performance and accuracy. Here are some of the most significant limitations of the Bresenham algorithm:

1. Limited to straight lines: Like the DDA algorithm, the Bresenham algorithm can only produce straight lines and cannot draw curves or other complex shapes.
2. Difficulty with vertical lines: The Bresenham algorithm performs less efficiently when drawing vertical lines since it requires more computation than horizontal lines.
3. Limited accuracy: While the Bresenham algorithm is more accurate than the DDA algorithm, it can still produce rounding errors that can affect the final output.
4. Difficulty with thick lines: The Bresenham algorithm only plots pixels on a single line, which makes it difficult to create thick lines or to fill shapes efficiently.
5. Not suitable for anti-aliasing: The Bresenham algorithm does not produce smooth lines, which makes it unsuitable for anti-aliasing, a technique used to reduce the jagged appearance of diagonal lines.

Despite these limitations, the Bresenham line drawing algorithm remains a popular choice for line drawing in computer graphics and image processing applications. Researchers and practitioners have developed several modifications and variations of the Bresenham algorithm that address some of its limitations and improve its performance and accuracy.

**Discussion :** In this lab class we learned about Bresenham line drawing algorithm. We learned about the implementation of bresenham line drawing algorithm.

**References:**

No reference used.