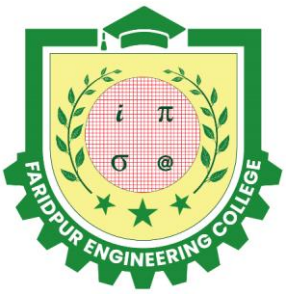
FARIDPUR ENGINEERING COLLEGE



Dept. of Computer Science & Engineering

Lab Report

Subject Name	: Computer Graphics (Sessional)
Subject Code	: CSE 802

Lab Report On : Implement a Line using Bresenham's Line Drawing Algorithm with OpenGL and Codeblocks IDE.

Submitted By:

Name : Md. Rakibul Islam Nahid

ID No. : 2537

Reg No. : 1797

Semester : 8th

Session : 2017-18

Submitted To:

Md. Rony Ahmed

Lecturer, Dept of CSE

Faridpur Engineering College, Faridpur

REMARKS

Date of submission

Experiment No: 02

Experiment Name: Implement a Line using Bresenham's Line Drawing Algorithm with OpenGL and Codeblocks IDE.

Aim: The aim of the lab report on Implementing a Line using Bresenham's Line Drawing Algorithm with OpenGL and Codeblocks IDE is to understand the concept of the Bresenham's line drawing algorithm and to implement it using the OpenGL graphics library and Codeblocks IDE. The main focus of the lab is to understand the differences between Bresenham's algorithm and the DDA algorithm, and to learn about the advantages and limitations of each algorithm. The lab also aims to develop programming skills in implementing the Bresenham's algorithm using C++ programming language and OpenGL graphics library, and to apply mathematical concepts to implement the algorithm. Overall, the lab aims to provide a comprehensive understanding of the Bresenham's line drawing algorithm and its applications in computer graphics.

Description: Bresenham's line drawing algorithm is a popular method used for drawing lines in computer graphics. The algorithm works by selecting the closest pixel to the line path at each step, making it efficient and fast. The Bresenham's algorithm is preferred over other line drawing algorithms because it uses only integer arithmetic, which makes it ideal for applications where speed and efficiency are critical. In addition, it produces accurate and smooth lines with no jagged edges or pixelation. The algorithm is widely used in various fields, including computer graphics, computer-aided design, and digital signal processing. This lab report discusses the implementation of a line using the Bresenham's line drawing algorithm with OpenGL and Codeblocks IDE, providing a detailed overview of the algorithm and its implementation using OpenGL libraries and Codeblocks IDE.

Requirements:

1. Codeblocks
2. OpenGL
3. Glut

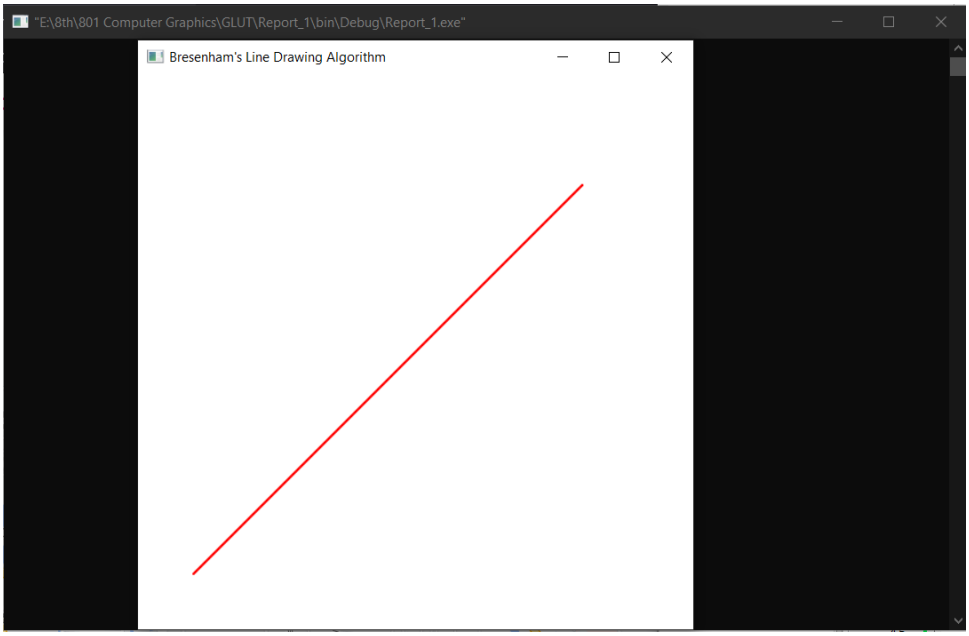
Some of the functions:

glBegin, glEnd: Delimit the vertices of a primitive or a group of like primitives.
glClear:Clears buffers to preset values.
glClearColor: Specifies clear values for the color buffers.
glFlush: Forces execution of OpenGL functions in finite time.
glMatrixMode: Specifies which matrix is the current matrix.
glutInit: Initialize GLUT
glutCreateWindow: Create window with the given title
glutInitWindowSize: Set the window's initial width & height
glutInitWindowPosition: Position the window's initial top-left corner
glutDisplayFunc(display): Register callback handler for window repaint event
initGL(): Our own OpenGL initialization

Algorithm:

1. Initialize the coordinates of the two endpoints of the line (x1, y1) and (x2, y2).
2. Determine the slope of the line m = (y2 - y1) / (x2 - x1).
3. Determine the octant in which the line falls based on the slope.
4. Initialize the error term e = 0.
5. For each pixel on the line, set the current pixel to (x, y) and draw it:
    - If the slope is between 0 and 1 (octants 1 and 8), increment x and update the error term: e = e + (y2 - y1) - (x2 - x1)
        - If e is positive, increment y: y = y + 1 and subtract (x2 - x1) from e.
    - If the slope is greater than 1 (octants 2 and 7), increment y and update the error term: e = e + (x2 - x1) - (y2 - y1)
        - If e is positive, increment x: x = x + 1 and subtract (y2 - y1) from e.
    - If the slope is less than -1 (octants 3 and 6), decrement y and update the error term: e = e + (x2 - x1) + (y2 - y1)
        - If e is positive, increment x: x = x + 1 and add (y2 - y1) to e.
    - If the slope is between -1 and 0 (octants 4 and 5), decrement x and update the error term: e = e - (y2 - y1) - (x2 - x1)
        - If e is positive, decrement y: y = y - 1 and add (x2 - x1) to e.
6. Repeat step 5 until the line is drawn completely.
7. Display the line on the screen.

Output:



Discussion:

1. The first step in the algorithm is to initialize the coordinates of the two endpoints of the line. These coordinates represent the starting and ending points of the line.
2. The slope of the line is then calculated as the difference in y-coordinates divided by the difference in x-coordinates. This slope is used to determine the direction of the line and the rate at which the coordinates of the pixels are incremented.
3. The error term and the decision parameter are then initialized to zero.