PROBLEM STATEMENT

To implement Bresenham's Line Drawing Algorithm in C using OpenGL, and to draw a line between two points on a 2D plane. The program will also draw the coordinate axes to visualize the line in the context of a Cartesian coordinate system.

THEORY

Bresenham's Line Drawing Algorithm is an efficient way to generate points of a line between two given points in a raster display. It uses only integer addition, subtraction, and bit shifting, which makes it faster than other methods like the DDA (Digital Differential Analyzer) algorithm. The algorithm determines the points of an n-dimensional raster that should be selected to form a close approximation to a straight line between two points.

The key idea is to keep track of the error term, which is the difference between the actual and the ideal y-coordinate values. By incrementally updating this error term and the x and y coordinates, the algorithm plots the line pixel by pixel.

ALGORITHM

1. **Initialize**:

   - Start with the initial point (x_0, y_0).
   - Calculate the differences: $dx = |x1 - x0|$ and $dy = |y1 - y0|$.
   - Determine the step direction for x and y: $sx = x0 < x1?1 : -1$ and $sy = y0 < y1?1 : -1$.
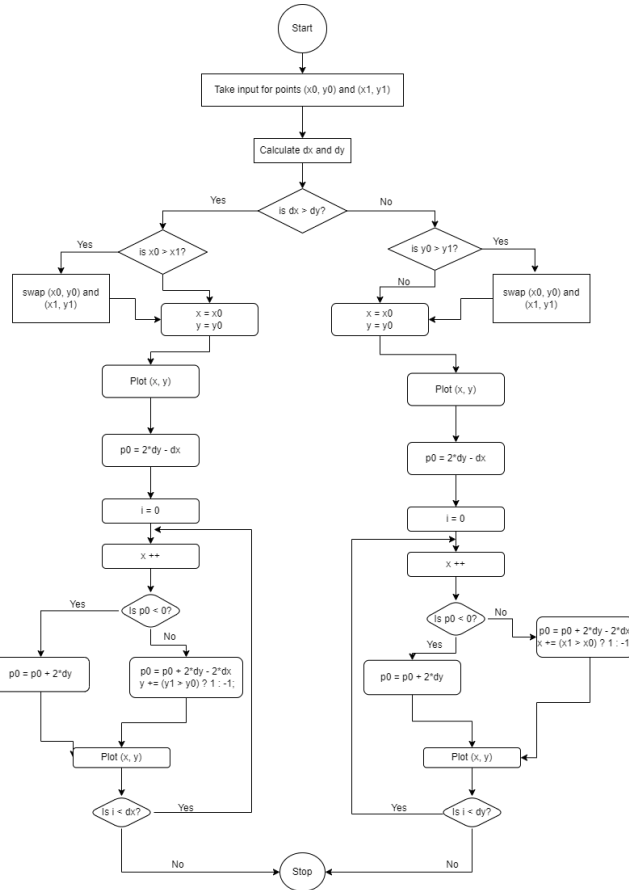   - Initialize the error term: $err = dx - dy$.

2. **Plotting**:

   - While the current point is not the end point:
     - Plot the current point.
     - Calculate the error term for the next point.
     - Update the coordinates based on the error term.

3. **Termination**:

   - The loop terminates when the current point reaches the end point $(x1, y1)$.

FLOWCHART

The flowchart contains the following elements:

- Start
- Take input for points (x0, y0) and (x1, y1)
- Calculate dx and dy
- is dx > dy?
  - Yes branch:
    - is x0 > x1?
      - Yes: swap (x0, y0) and (x1, y1)
      - x = x0, y = y0
      - Plot (x, y)
      - p0 = 2*dy - dx
      - i = 0
      - x ++
      - Is p0 < 0?
        - Yes: p0 = p0 + 2*dy
        - No: p0 = p0 + 2*dy - 2*dx, y += (y1 > y0) ? 1 : -1;
      - Plot (x, y)
      - Is i < dx?
        - Yes: (loop back)
        - No: Stop
  - No branch:
    - is y0 > y1?
      - Yes: swap (x0, y0) and (x1, y1)
      - No: x = x0, y = y0
      - Plot (x, y)
      - p0 = 2*dy - dx
      - i = 0
      - x ++
      - Is p0 < 0?
        - No: p0 = p0 + 2*dy - 2*dx, x += (x1 > x0) ? 1 : -1;
        - Yes: p0 = p0 + 2*dy
      - Plot (x, y)
      - Is i < dy?
        - Yes: (loop back)
        - No: Stop
- Stop

SAMPLE I/O

Input:

```
Enter the coordinates of the first point (x_start, y_start): -50 -50
Enter the coordinates of the second point (x_end, y_end): 50 50
```

Output: A line is drawn from $(-50, -50)$ to $(50, 50)$ on a window with coordinate axes displayed.

DISCUSSIONS

- **Accuracy**: Bresenham's algorithm provides a highly accurate and visually appealing way to draw lines because it minimizes the error and ensures that the line is continuous.

- **Efficiency**: The algorithm is computationally efficient as it uses only integer arithmetic, which is faster than floating-point operations.

- **Applications**: Beyond simple line drawing, Bresenham's algorithm is foundational in computer graphics and is used in rendering shapes, text, and other graphical elements.

CONCLUSION

Bresenham's Line Drawing Algorithm is an essential algorithm in computer graphics for rendering straight lines with high efficiency and accuracy. The implementation using OpenGL effectively demonstrates the algorithm's capability to draw lines between any two points while also integrating the visualization of coordinate axes.