2024.08.04

Practical Implementation of DDA and Bresenham Line Drawing Algorithms with JAVA

(Self-learning Task sheet)

☺ Refer the given samples codes on above mentioned topic.

1. DDA Algorithm

```java
import java.awt.*; // Importing AWT package for Graphics
import javax.swing.*; // Importing Swing package for JFrame and JPanel

public class SimpleDDALine extends JPanel {

    // Starting and ending coordinates for the line
    private int x1, y1, x2, y2;

    // Constructor to initialize the line coordinates
    public SimpleDDALine(int x1, int y1, int x2, int y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
    }

    // Method to draw the line
    public void paintComponent(Graphics g) {
        super.paintComponent(g); // Call to the parent class's method

        // Calculate the differences between the starting and ending points
        int dx = x2 - x1;
        int dy = y2 - y1;

        // Calculate the number of steps required to draw the line
        int steps = Math.max(Math.abs(dx), Math.abs(dy));

        // Calculate the increment in x and y for each step
        float xIncrement = dx / (float) steps;
        float yIncrement = dy / (float) steps;
```

```java
        int dy = y2 - y1;

        // Calculate the number of steps required to draw the line
        int steps = Math.max(Math.abs(dx), Math.abs(dy));

        // Calculate the increment in x and y for each step
        float xIncrement = dx / (float) steps;
        float yIncrement = dy / (float) steps;

        // Initialize starting points
        float x = x1;
        float y = y1;

        // Draw the line by incrementing x and y at each step
        for (int i = 0; i <= steps; i++) {
            g.drawRect(Math.round(x), Math.round(y), 1, 1); // Draw a pixel at the current (x, y) position
            x += xIncrement; // Increment x by xIncrement
            y += yIncrement; // Increment y by yIncrement
        }
    }

    // Main method to create a frame and display the line
    public static void main(String[] args) {
        JFrame frame = new JFrame("Simple DDA Line Drawing"); // Create a frame
        frame.add(new SimpleDDALine(50, 50, 200, 150)); // Add the panel to the frame with example coordinates
        frame.setSize(400, 400); // Set the frame size
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Set the default close operation
        frame.setVisible(true); // Make the frame visible
    }
}
```

2. Bresenham's Line Drawing Algorithhm

```java
import java.awt.*; // Importing AWT package for Graphics
import javax.swing.*; // Importing Swing package for JFrame and JPanel

public class SimpleBresenhamLine extends JPanel {

    // Starting and ending coordinates for the line
    private int x1, y1, x2, y2;

    // Constructor to initialize the line coordinates
    public SimpleBresenhamLine(int x1, int y1, int x2, int y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
    }

    // Method to draw the line
    public void paintComponent(Graphics g) {
        super.paintComponent(g); // Call to the parent class's method

        // Calculate the absolute differences between the starting and ending points
        int dx = Math.abs(x2 - x1);
        int dy = Math.abs(y2 - y1);

        // Determine the direction of the increment for x and y
        int sx = x1 < x2 ? 1 : -1;
        int sy = y1 < y2 ? 1 : -1;

        // Initialize the error term
        int err = dx - dy;

```

```java
        // Initialize the error term
        int err = dx - dy;

        // Initialize starting points
        int x = x1;
        int y = y1;

        // Draw the line
        while (true) {
            g.drawRect(x, y, 1, 1); // Draw a pixel at the current (x, y) position

            if (x == x2 && y == y2) break; // If we've reached the end point, break the loop

            int e2 = 2 * err; // Double the error term for the next step

            if (e2 > -dy) { // If the error term is greater than -dy
                err -= dy; // Decrease the error term by dy
                x += sx; // Increment x by the step size in the x direction
            }

            if (e2 < dx) { // If the error term is less than dx
                err += dx; // Increase the error term by dx
                y += sy; // Increment y by the step size in the y direction
            }
        }
    }

    // Main method to create a frame and display the line
    public static void main(String[] args) {
        JFrame frame = new JFrame("Simple Bresenham Line Drawing"); // Create a frame
```

```java
        while (true) {
            g.drawRect(x, y, 1, 1); // Draw a pixel at the current (x, y) position

            if (x == x2 && y == y2) break; // If we've reached the end point, break the loop

            int e2 = 2 * err; // Double the error term for the next step

            if (e2 > -dy) { // If the error term is greater than -dy
                err -= dy; // Decrease the error term by dy
                x += sx; // Increment x by the step size in the x direction
            }

            if (e2 < dx) { // If the error term is less than dx
                err += dx; // Increase the error term by dx
                y += sy; // Increment y by the step size in the y direction
            }
        }
    }

    // Main method to create a frame and display the line
    public static void main(String[] args) {
        JFrame frame = new JFrame("Simple Bresenham Line Drawing"); // Create a frame
        frame.add(new SimpleBresenhamLine(50, 50, 200, 150)); // Add the panel to the frame with example coordinates
        frame.setSize(400, 400); // Set the frame size
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Set the default close operation
        frame.setVisible(true); // Make the frame visible
    }
}
```

☺ Do the following activities and submit a **zip file** of your src folder to the available link.

1. Create a NetBeans project named "CG_P".
2. Inside that create 4 classes named "H_DDA", "H_BA", "V_DDA" and "V_BA".
3. Inside of H_DDA class,

   Implement the codes for the lines which connect the (50, 100) and (250, 100) using DDA Algorithm.
4. Inside of H_BA class,

   Implement the codes for the lines which connect the (50, 100) and (250, 100) using Bresenham Line Drawing Algorithm.
5. Inside of V_DDA class,

   Implement the codes for the lines which connect the (150, 50) and (150, 300) using DDA algorithm.
6. Inside of H_BA class,

   Implement the codes for the lines which connect the (150, 50) and (150, 300) using Bresenham Line Drawing Algorithm.