

Lab Assignment No :10

Name: Swapnil Satish Kalshetti

Class: SE A

Roll no: C02062

Subject: OOP&CG

```
import java.awt.*;
import javax.swing.*;

public class TransformTriangleAll extends JPanel {
    double angle; // rotation angle in radians
    double scaleX, scaleY;
    int tx, ty; // translation values
    double shx, shy; // shear factors

    public TransformTriangleAll() {
        this.angle = Math.toRadians(45); // rotate 45 degrees
        this.scaleX = 1.5; // scale 1.5x in X
        this.scaleY = 1.2; // scale 1.2x in Y
        this.tx = 100; // translate +100 in X
        this.ty = 50; // translate +50 in Y
        this.shx = 0.5; // shear factor in X
        this.shy = 0.3; // shear factor in Y
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

        int width = getWidth();
        int height = getHeight();

        // === Draw coordinate axes ===
        g2.setColor(Color.GRAY);
        g2.drawLine(0, height / 2, width, height / 2); // X-axis
        g2.drawLine(width / 2, 0, width / 2, height); // Y-axis

        g2.setFont(new Font("Arial", Font.BOLD, 12));
        g2.drawString("X", width - 15, height / 2 - 5);
        g2.drawString("Y", width / 2 + 5, 15);

        // Tick marks
        for (int i = width / 2; i < width; i += 50) {
            g2.drawLine(i, height / 2 - 5, i, height / 2 + 5);
            g2.drawLine(width - i, height / 2 - 5, width - i, height / 2 + 5);
        }
    }
}
```

```

for (int j = height / 2; j < height; j += 50) {
    g2.drawLine(width / 2 - 5, j, width / 2 + 5, j);
    g2.drawLine(width / 2 - 5, height - j, width / 2 + 5, height - j);
}

// Shift origin to center
g2.translate(width / 2, height / 2);
g2.scale(1, -1);

// Base triangle (small for better visibility)
int[] xPoints = { 0, 40, 80 };
int[] yPoints = { 60, 0, 60 };

// === Quadrant I: Original (Blue) ===
g2.setColor(Color.BLUE);
g2.drawPolygon(xPoints, yPoints, 3);
g2.scale(1, -1); g2.drawString("Original", 20, -70); g2.scale(1, -1);

// === Quadrant II: Translated (Green) ===
int[] xTranslated = new int[3];
int[] yTranslated = new int[3];
for (int i = 0; i < 3; i++) {
    xTranslated[i] = xPoints[i] - 150; // move left
    yTranslated[i] = yPoints[i] + 50; // move up
}
g2.setColor(Color.GREEN);
g2.drawPolygon(xTranslated, yTranslated, 3);
g2.scale(1, -1); g2.drawString("Translated", -220, -70); g2.scale(1, -1);

// === Quadrant III: Rotated (Red) ===
int[] xRotated = new int[3];
int[] yRotated = new int[3];
double cx = (xPoints[0] + xPoints[1] + xPoints[2]) / 3.0;
double cy = (yPoints[0] + yPoints[1] + yPoints[2]) / 3.0;
for (int i = 0; i < 3; i++) {
    double xShifted = xPoints[i] - cx;
    double yShifted = yPoints[i] - cy;
    double xRot = xShifted * Math.cos(angle) - yShifted * Math.sin(angle);
    double yRot = xShifted * Math.sin(angle) + yShifted * Math.cos(angle);
    xRotated[i] = (int) (xRot + cx - 150); // shift to left
    yRotated[i] = (int) (yRot + cy - 150); // shift down
}
g2.setColor(Color.RED);
g2.drawPolygon(xRotated, yRotated, 3);
g2.scale(1, -1); g2.drawString("Rotated", -220, 150); g2.scale(1, -1);

// === Quadrant IV: Scaled (Orange) ===
int[] xScaled = new int[3];
int[] yScaled = new int[3];
for (int i = 0; i < 3; i++) {
    xScaled[i] = (int) (xPoints[i] * scaleX) + 100; // move right
    yScaled[i] = (int) (yPoints[i] * scaleY) - 150; // move down
}
g2.setColor(Color.ORANGE);
g2.drawPolygon(xScaled, yScaled, 3);
g2.scale(1, -1); g2.drawString("Scaled", 120, 150); g2.scale(1, -1);

```

```

// === Center Area: Shear (Magenta) ===
int[] xShear = new int[3];
int[] yShear = new int[3];
for (int i = 0; i < 3; i++) {
    xShear[i] = (int) (xPoints[i] + shx * yPoints[i]);
    yShear[i] = (int) (yPoints[i] + shy * xPoints[i]);
}
g2.setColor(Color.MAGENTA);
g2.drawPolygon(xShear, yShear, 3);
g2.scale(1, -1); g2.drawString("Sheared", 40, -20); g2.scale(1, -1);
}

public static void main(String[] args) {
    JFrame frame = new JFrame("2D Transformations - All Quadrants");
    TransformTriangleAll panel = new TransformTriangleAll();

    frame.add(panel);
    frame.setSize(700, 700);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
}
}

```

Output:

