SPRAWOZDANIE

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium 3

Data: 27.03.2022

Temat: Modelowanie hierarchiczne w grafice 2D

Jan Głuch

Informatyka I stopień,

zaoczne,

4 semestr,

Gr. 2A

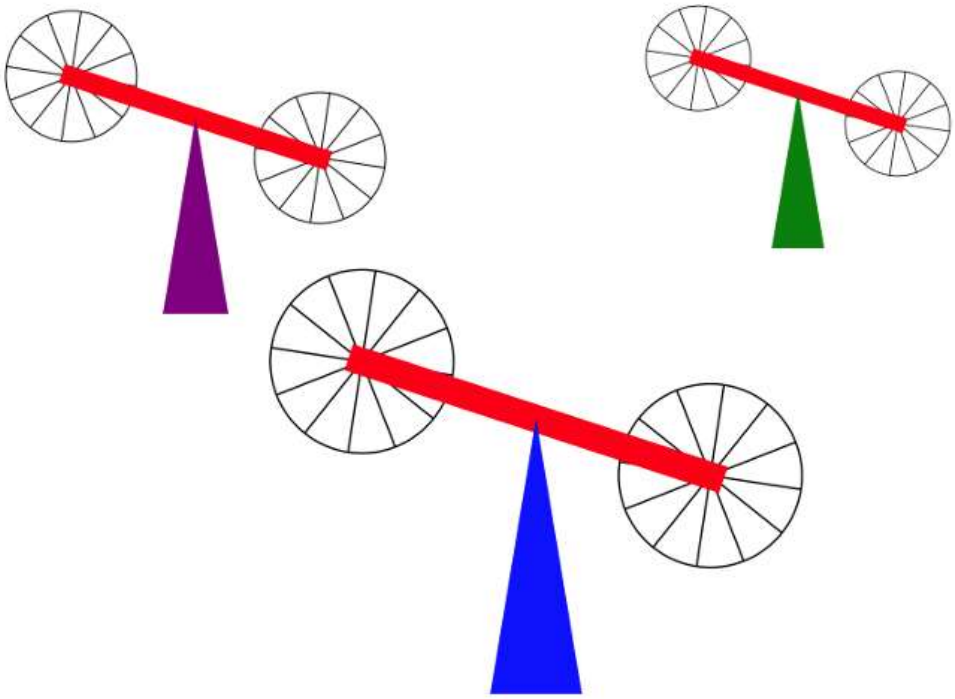
1. Polecenia.

Opracować scenę hierarchiczną zgodnie z obrazem używając zamiast kół wielokąty obracające się (animacja!) według wariantu. Opracowanie powinno być w jednym z języków: Java lub JavaScript,

na dwa sposoby:

(a) używając hierarchiję funkcje (sposób subroutinowy)

(b) tworząc graf sceny (sposób obiektowy). W tym celu proponuję do pobrania odpowiedni pliki



2. Wprowadzane dane.

2.1 Sposób subroutinowy

```
// ------------------------------ Utworzenie wielokata ------------------
private void F1(Graphics2D g2) {
    AffineTransform saveTransform = g2.getTransform();
    Color saveColor = g2.getColor();

    g2.setTransform(saveTransform);
    g2.translate(1, 1);
    int n=9;
    double r = 150,
           t=0,
           k=(Math.PI*2)/n;
    int[] x1 = new int[n];
    int[] y1 = new int[n];
    for (int i=0;i<n;i++)
    {
        x1[i]= (int) (r*Math.sin(t));
        y1[i]= (int) (r*Math.cos(t));
        t+=k;
    }
```

```
        Polygon polygon = new Polygon(x1,y1,n);
        g2.translate(1.9, -2.05);
        g2.setColor( Color.pink );
        g2.rotate( Math.toRadians( frameNumber*0.75 ));
        g2.scale( 0.005, 0.005 );
        g2.fill(polygon);

        g2.setColor(saveColor);
        g2.setTransform(saveTransform);



        g2.translate(-0.4, -0);
        g2.setColor( Color.pink );
        g2.rotate( Math.toRadians( frameNumber*0.75 ));
        g2.scale( 0.005, 0.005 );
        g2.fill(polygon);

        g2.setColor(saveColor);
        g2.setTransform(saveTransform);



        g2.translate(0.7,-2.3);
        g2.setColor(Color.red);

        g2.setStroke(new BasicStroke((float) 0.2));

        g2.draw( new Line2D.Double( -1,2.3, 2.2,1.3) );

        g2.setTransform(saveTransform);


        g2.translate(0.7, -0.8);
        g2.setColor(Color.blue);
        triangle1(g2);

        g2.setTransform(saveTransform);


        g2.setTransform(saveTransform);

        g2.scale(0.7, 0.7);
        g2.translate(-4.4, 3.2);
        g2.setColor( Color.pink );
        g2.rotate( Math.toRadians( frameNumber*0.75 ));
        g2.scale( 0.005, 0.005 );
        g2.fill(polygon);

        g2.setColor(saveColor);
        g2.setTransform(saveTransform);
```

```
        g2.scale(0.7, 0.7);
        g2.translate(-1.4, 2.3);
        g2.setColor( Color.pink );
        g2.rotate( Math.toRadians( frameNumber*0.75 ));
        g2.scale( 0.005, 0.005 );
        g2.fill(polygon);

        g2.setColor(saveColor);


        g2.setTransform(saveTransform);

        g2.scale(0.7, 1);
        g2.translate(-3.5,0.2);
        g2.setColor(Color.red);

        g2.setStroke(new BasicStroke((float) 0.2));

        g2.draw( new Line2D.Double( -0.8,2, 2,1.4) );

        g2.setTransform(saveTransform);

        g2.translate(-2.5, 1.7);
        g2.setColor(new Color(145,17,133));
        g2.scale(0.7, 0.8);
        triangle1(g2);


        g2.setTransform(saveTransform);


        g2.scale(0.6, 0.60);
        g2.translate(4.6, 3);
        g2.setColor( Color.pink );
        g2.rotate( Math.toRadians( frameNumber*0.75 ));
        g2.scale( 0.005, 0.005 );
        g2.fill(polygon);

        g2.setColor(saveColor);
        g2.setTransform(saveTransform);


        g2.scale(0.6, 0.6);
        g2.translate(1.5, 3.9);
        g2.setColor( Color.pink );
        g2.rotate( Math.toRadians( frameNumber*0.75 ));
        g2.scale( 0.005, 0.005 );
        g2.fill(polygon);

        g2.setColor(saveColor);


        g2.setTransform(saveTransform);
```

```
        g2.scale(0.6, 0.8);
        g2.translate(2.5,0.9);
        g2.setColor(Color.red);

        g2.setStroke(new BasicStroke((float) 0.2));

        g2.draw( new Line2D.Double( -0.8,2, 2,1.4) );



        g2.setTransform(saveTransform);

        g2.translate(1.6, 1.9);
        g2.setColor(new Color(42,120,18));
        g2.scale(0.5, 0.6);
        triangle1(g2);

    }

    private void triangle1(Graphics2D g2) {

        g2.translate(0.5, -2);


        Path2D path = new Path2D.Double();
        path.moveTo(-0.5,0);
        path.lineTo(0.5,0);
        path.lineTo(0,2.3);
        path.closePath();
        g2.fill(path);
    }

    private void line1(Graphics2D g2) {
        g2.setColor(Color.red);

        g2.setStroke(new BasicStroke((float) 0.3));

        g2.draw( new Line2D.Double( -1,2.3, 2.2,1.3) );
    }
```

```
//------------------- Some methods for drawing basic shapes. ----------------

private static void line(Graphics2D g2) { // Draws a line from (-0.5,0) to (0.5,0)
    g2.draw( new Line2D.Double( -0.5,0, 0.5,0) );
}

private static void rect(Graphics2D g2) { // Strokes a square, size = 1, center = (0,0)
    g2.draw(new Rectangle2D.Double(-0.5,-0.5,1,1));
}

private static void filledRect(Graphics2D g2) { // Fills a square, size = 1, center = (0,0)
    g2.fill(new Rectangle2D.Double(-0.5,-0.5,1,1));
}
private static void filledPolygon(Graphics2D g2) { // Fills a square, size = 1, center = (0,0)
    g2.fill(new Rectangle2D.Double(-0.5,-0.5,1,1));
}

private static void circle(Graphics2D g2) { // Strokes a circle, diameter = 1, center = (0,0)
    g2.draw(new Ellipse2D.Double(-0.5,-0.5,1,1));
}

private static void filledCircle(Graphics2D g2) { // Fills a circle, diameter = 1, center = (0,0)
    g2.draw(new Ellipse2D.Double(-0.5,-0.5,1,1));
}

private static void filledTriangle(Graphics2D g2) { // width = 1, height = 1, center of base is at (0,0);
    Path2D path = new Path2D.Double();
    path.moveTo(-0.5,0);
    path.lineTo(0.5,0);
    path.lineTo(0,1);
    path.closePath();
    g2.fill(path);
}
```

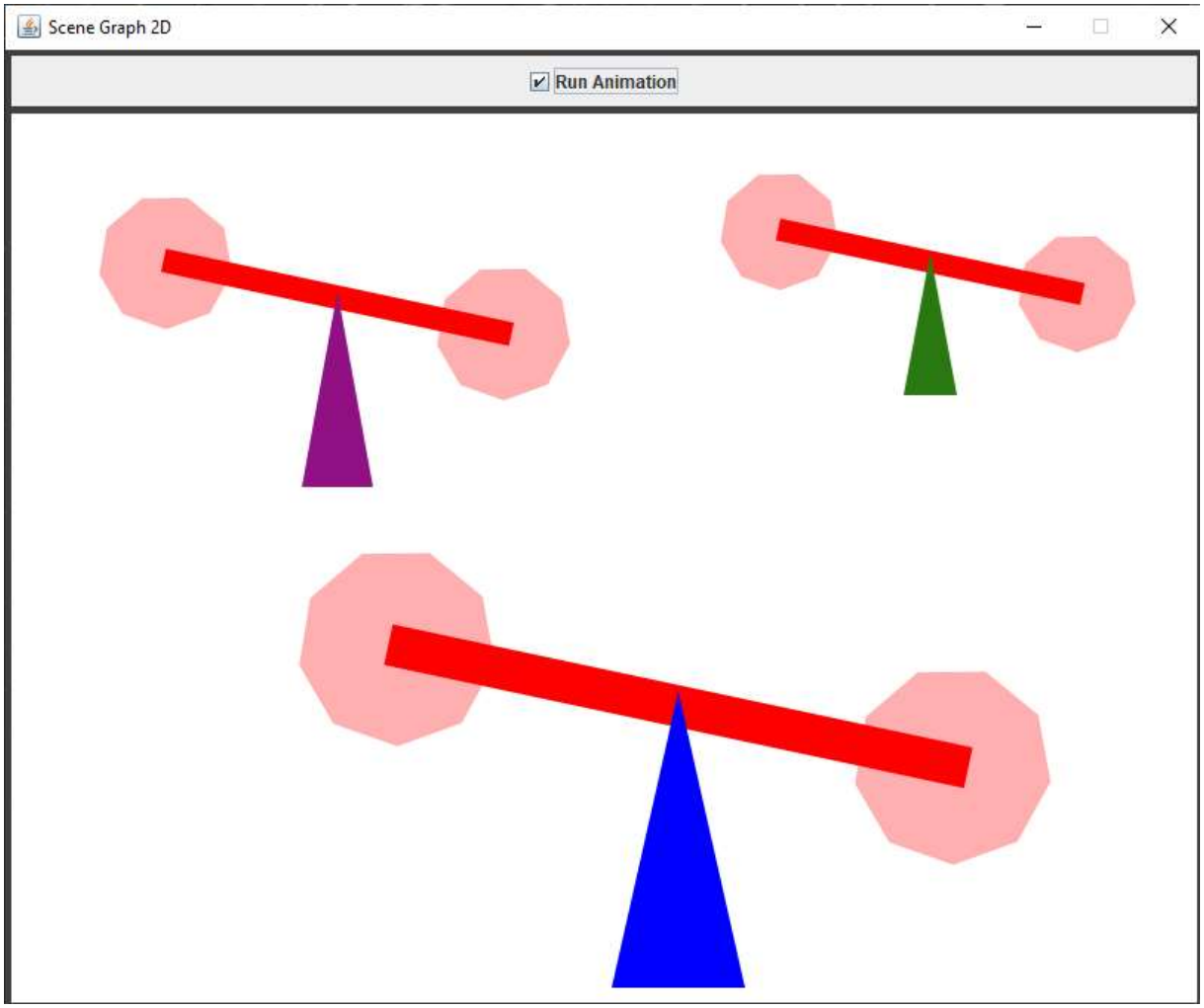
```
//-------------------------------- Implementation -----------------------------------

private JPanel display;  // The JPanel in which the scene is drawn.

/**
 * Constructor creates the scene graph data structure that represents the
 * scene that is to be drawn in this panel, by calling createWorld().
 * It also sets the preferred size of the panel to the constants WIDTH and HEIGHT.
 * And it creates a timer to drive the animation.
 */
public SubroutineHierarchy() {
    display = new JPanel() {
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2 = (Graphics2D)g.create();
            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
            applyLimits(g2, X_LEFT, X_RIGHT, Y_TOP, Y_BOTTOM, false);
            g2.setStroke( new BasicStroke(pixelSize) ); // set default line width to one pixel.
            drawWorld(g2);  // draw the world
        }
    };
    display.setPreferredSize( new Dimension(WIDTH,HEIGHT));
    display.setBackground( BACKGROUND );
    final Timer timer = new Timer(17,new ActionListener() { // about 60 frames per second
        public void actionPerformed(ActionEvent evt) {
            updateFrame();
            repaint();
        }
    });
    final JCheckBox animationCheck = new JCheckBox("Run Animation");
    animationCheck.addActionListener( new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            if (animationCheck.isSelected()) {
                if ( ! timer.isRunning() )
                    timer.start();
            }
            else {
                if ( timer.isRunning() )
                    timer.stop();
            }
        }
    });
    JPanel top = new JPanel();
    top.add(animationCheck);
    setLayout(new BorderLayout(5,5));
    setBackground(Color.DARK_GRAY);
    setBorder( BorderFactory.createLineBorder(Color.DARK_GRAY,4) );
    add(top,BorderLayout.NORTH);
    add(display,BorderLayout.CENTER);
}
```

Otrzymane wyniki dla tej metody:

Kształt bryły wielokąta to dziewięciokąt



2.2 Graf Sceny

```
// ------------------------------ Utworzenie wielokata ------------------
private void F1(Graphics2D g2) {
    AffineTransform saveTransform = g2.getTransform();
    Color saveColor = g2.getColor();

    g2.setTransform(saveTransform);
    g2.translate(1, 1);
    int n=9;
    double r = 150,
           t=0,
           k=(Math.PI*2)/n;
    int[] x1 = new int[n];
    int[] y1 = new int[n];
    for (int i=0;i<n;i++)
    {
        x1[i]= (int) (r*Math.sin(t));
        y1[i]= (int) (r*Math.cos(t));
        t+=k;
    }


    Polygon polygon = new Polygon(x1,y1,n);
    g2.translate(1.9, -2.05);
    g2.setColor( Color.pink );
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale( 0.005, 0.005 );
    g2.fill(polygon);

    g2.setColor(saveColor);
    g2.setTransform(saveTransform);



    g2.translate(-0.4, -0);
    g2.setColor( Color.pink );
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale( 0.005, 0.005 );
    g2.fill(polygon);

    g2.setColor(saveColor);
    g2.setTransform(saveTransform);



    g2.translate(0.7,-2.3);
    g2.setColor(Color.red);

    g2.setStroke(new BasicStroke((float) 0.2));

    g2.draw( new Line2D.Double( -1,2.3, 2.2,1.3) );
```

```
    g2.setTransform(saveTransform);


    g2.translate(0.7, -0.8);
    g2.setColor(Color.blue);
    triangle1(g2);

    g2.setTransform(saveTransform);


    g2.setTransform(saveTransform);

    g2.scale(0.7, 0.7);
    g2.translate(-4.4, 3.2);
    g2.setColor( Color.pink );
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale( 0.005, 0.005 );
    g2.fill(polygon);

    g2.setColor(saveColor);
    g2.setTransform(saveTransform);


    g2.scale(0.7, 0.7);
    g2.translate(-1.4, 2.3);
    g2.setColor( Color.pink );
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale( 0.005, 0.005 );
    g2.fill(polygon);

    g2.setColor(saveColor);


    g2.setTransform(saveTransform);

    g2.scale(0.7, 1);
    g2.translate(-3.5,0.2);
    g2.setColor(Color.red);

    g2.setStroke(new BasicStroke((float) 0.2));

    g2.draw( new Line2D.Double( -0.8,2, 2,1.4) );

    g2.setTransform(saveTransform);

    g2.translate(-2.5, 1.7);
    g2.setColor(new Color(145,17,133));
    g2.scale(0.7, 0.8);
    triangle1(g2);


    g2.setTransform(saveTransform);
```

```
    g2.scale(0.6, 0.60);
    g2.translate(4.6, 3);
    g2.setColor( Color.pink );
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale( 0.005, 0.005 );
    g2.fill(polygon);

    g2.setColor(saveColor);
    g2.setTransform(saveTransform);


    g2.scale(0.6, 0.6);
    g2.translate(1.5, 3.9);
    g2.setColor( Color.pink );
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale( 0.005, 0.005 );
    g2.fill(polygon);

    g2.setColor(saveColor);


    g2.setTransform(saveTransform);

    g2.scale(0.6, 0.8);
    g2.translate(2.5,0.9);
    g2.setColor(Color.red);

    g2.setStroke(new BasicStroke((float) 0.2));

    g2.draw( new Line2D.Double( -0.8,2, 2,1.4) );



    g2.setTransform(saveTransform);

    g2.translate(1.6, 1.9);
    g2.setColor(new Color(42,120,18));
    g2.scale(0.5, 0.6);
    triangle1(g2);

}
```

```
    }

    private void triangle1(Graphics2D g2) {

        g2.translate(0.5, -2);


        Path2D path = new Path2D.Double();
        path.moveTo(-0.5,0);
        path.lineTo(0.5,0);
        path.lineTo(0,2.3);
        path.closePath();
        g2.fill(path);
    }

    private void line1(Graphics2D g2) {
        g2.setColor(Color.red);

        g2.setStroke(new BasicStroke((float) 0.3));

        g2.draw( new Line2D.Double( -1,2.3, 2.2,1.3) );
    }



    //------------------- Some methods for drawing basic shapes. ----------------

    private static void line(Graphics2D g2) { // Draws a line from (-0.5,0) to (0.5,0)
        g2.draw( new Line2D.Double( -0.5,0, 0.5,0) );
    }

    private static void rect(Graphics2D g2) { // Strokes a square, size = 1, center = (0,0)
        g2.draw(new Rectangle2D.Double(-0.5,-0.5,1,1));
    }

    private static void filledRect(Graphics2D g2) { // Fills a square, size = 1, center = (0,0)
        g2.fill(new Rectangle2D.Double(-0.5,-0.5,1,1));
    }
    private static void filledPolygon(Graphics2D g2) { // Fills a square, size = 1, center = (0,0)
        g2.fill(new Rectangle2D.Double(-0.5,-0.5,1,1));
    }

    private static void circle(Graphics2D g2) { // Strokes a circle, diameter = 1, center = (0,0)
        g2.draw(new Ellipse2D.Double(-0.5,-0.5,1,1));
    }

    private static void filledCircle(Graphics2D g2) { // Fills a circle, diameter = 1, center = (0,0)
        g2.draw(new Ellipse2D.Double(-0.5,-0.5,1,1));
    }
```

```
    private static void filledTriangle(Graphics2D g2) { // width = 1, height = 1, center of base is at (0,0);
        Path2D path = new Path2D.Double();
        path.moveTo(-0.5,0);
        path.lineTo(0.5,0);
        path.lineTo(0,1);
        path.closePath();
        g2.fill(path);
    }



    //-------------------------------- Implementation ----------------------------------

    private JPanel display;  // The JPanel in which the scene is drawn.

    /**
     * Constructor creates the scene graph data structure that represents the
     * scene that is to be drawn in this panel, by calling createWorld().
     * It also sets the preferred size of the panel to the constants WIDTH and HEIGHT.
     * And it creates a timer to drive the animation.
     */
    public SubroutineHierarchy() {
        display = new JPanel() {
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2 = (Graphics2D)g.create();
                g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
                applyLimits(g2, X_LEFT, X_RIGHT, Y_TOP, Y_BOTTOM, false);
                g2.setStroke( new BasicStroke(pixelSize) ); // set default line width to one pixel.
                drawWorld(g2);  // draw the world
            }
        };
        display.setPreferredSize( new Dimension(WIDTH,HEIGHT));
        display.setBackground( BACKGROUND );
        final Timer timer = new Timer(17,new ActionListener() { // about 60 frames per second
            public void actionPerformed(ActionEvent evt) {
                updateFrame();
                repaint();
            }
        });
```

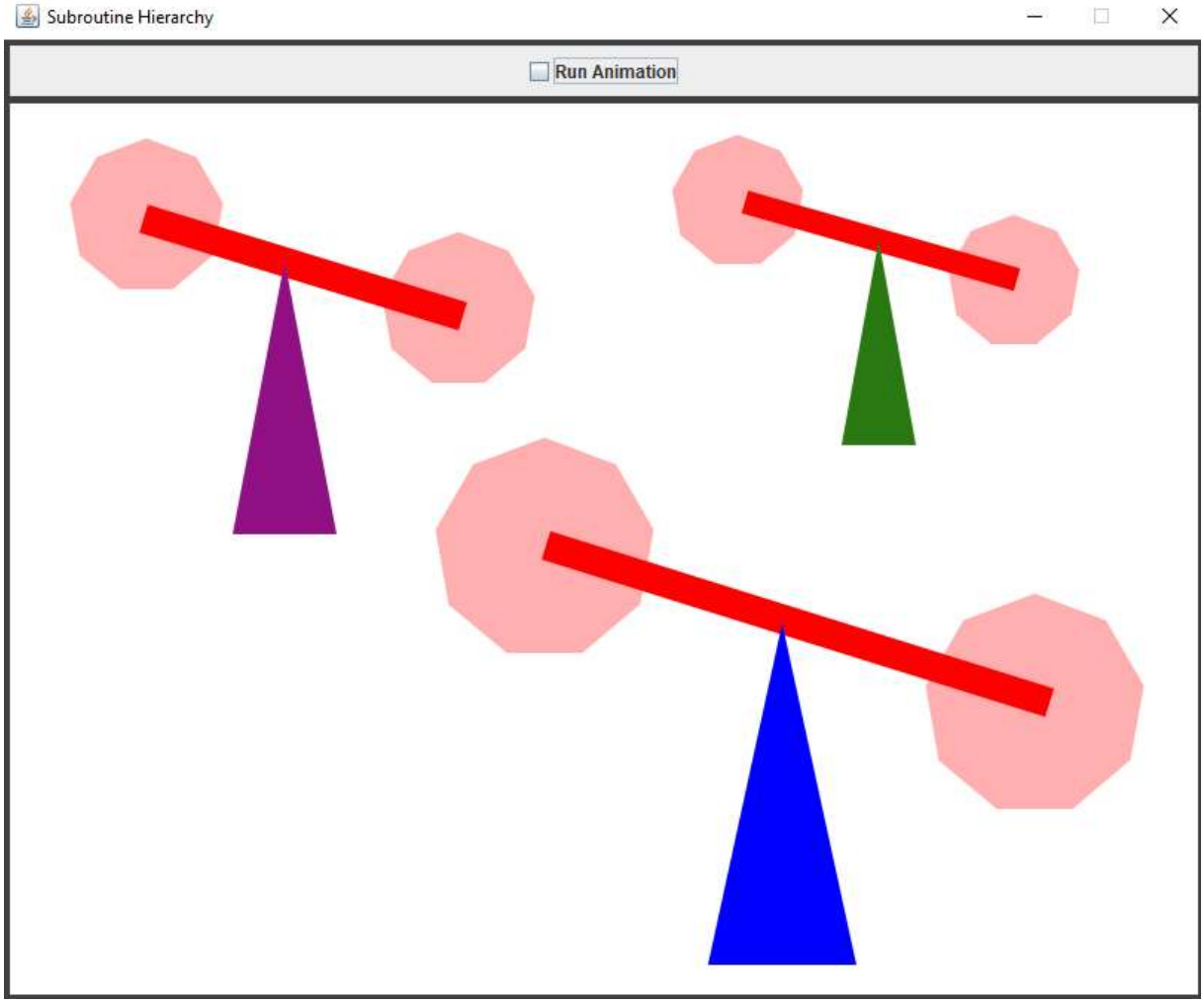
```
        }
    });
    final JCheckBox animationCheck = new JCheckBox("Run Animation");
    animationCheck.addActionListener( new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            if (animationCheck.isSelected()) {
                if ( ! timer.isRunning() )
                    timer.start();
            }
            else {
                if ( timer.isRunning() )
                    timer.stop();
            }
        }
    });
    JPanel top = new JPanel();
    top.add(animationCheck);
    setLayout(new BorderLayout(5,5));
    setBackground(Color.DARK_GRAY);
    setBorder( BorderFactory.createLineBorder(Color.DARK_GRAY,4) );
    add(top,BorderLayout.NORTH);
    add(display,BorderLayout.CENTER);
}
```

```
private void applyLimits(Graphics2D g2, double xleft, double xright,
                         double ytop, double ybottom, boolean preserveAspect) {
    int width = display.getWidth();   // The width of the drawing area, in pixels.
    int height = display.getHeight(); // The height of the drawing area, in pixels.
    if (preserveAspect) {
        // Adjust the limits to match the aspect ratio of the drawing area.
        double displayAspect = Math.abs((double)height / width);
        double requestedAspect = Math.abs(( ybottom-ytop ) / ( xright-xleft ));
        if (displayAspect > requestedAspect) {
            double excess = (ybottom-ytop) * (displayAspect/requestedAspect - 1);
            ybottom += excess/2;
            ytop -= excess/2;
        }
        else if (displayAspect < requestedAspect) {
            double excess = (xright-xleft) * (requestedAspect/displayAspect - 1);
            xright += excess/2;
            xleft -= excess/2;
        }
    }
    double pixelWidth = Math.abs(( xright - xleft ) / width);
    double pixelHeight = Math.abs(( ybottom - ytop ) / height);
    pixelSize = (float)Math.min(pixelWidth,pixelHeight);
    g2.scale( width / (xright-xleft), height / (ybottom-ytop) );
    g2.translate( -xleft, -ytop );
}

}
```

Otrzymane wyniki dla tej metody:

Kształt bryły wielokąta to dziewięciokąt



Wnioski

Obie metody zapewniają podobne takie same możliwości wykonania zadania. W modelu hierarchicznym ilość kodu potrzebnego do wykonania zadania jest zdecydowanie większa.

Link do GitHuba: https://github.com/kenaj83/Grafika.git