МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №2

з дисципліни *“*Математичні та алгоритмічні основи комп’ютерної графіки*”*

Виконав
студент ІІІ курсу
групи КП-83
Коваль Андрій Олександрович
(прізвище, ім’я, по батькові)
варіант № 8

Зарахована
“____” “_____” 20___ р.
викладачем Шкурат Оксаною Сергіївною
(прізвище, ім’я, по батькові)

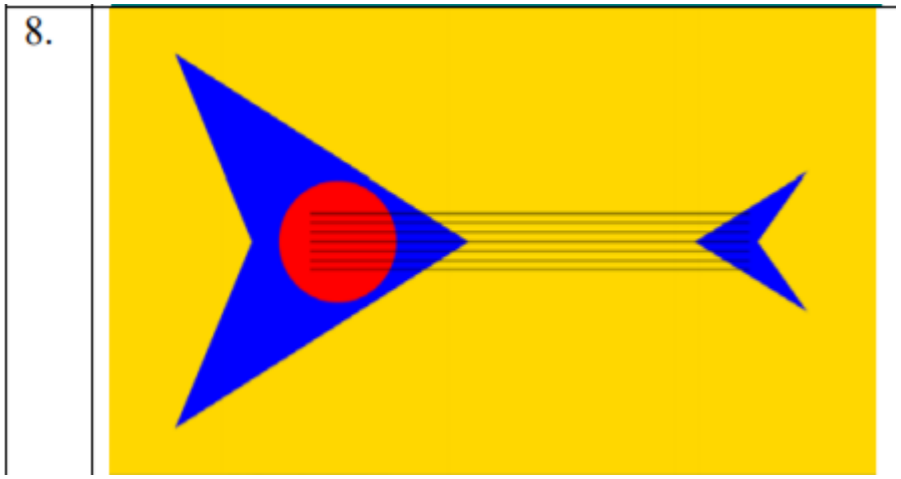
Київ 2021

Завдання

За допомогою Java 2D намалювати картинку з лабораторної роботи №1 (за варіантом). Додатково виконати:

1. Хоча б 1 стандартний примітив, та хоча б 1 фігуру, побудовану по точкам (ламаною).
2. Хоча б 1 фігуру залити градієнтною фарбою за вибором (в цьому випадку колір може не співпадати з варіантом із лабораторної роботи № 1).
3. На достатній відстані від побудованого малюнку намалювати прямокутну рамку, всередині якої відбуватиметься анімація. Тип лінії рамки задано за варіантом. (JOIN_MITER)
4. Виконати анімацію малюнку, за варіантом. При цьому рамка повинна залишатися статичною. Взаємодія з рамкою не обов’язкова, якщо не передбачено варіантом.

Варіант 8.



Лістинг коду програми

Main.java

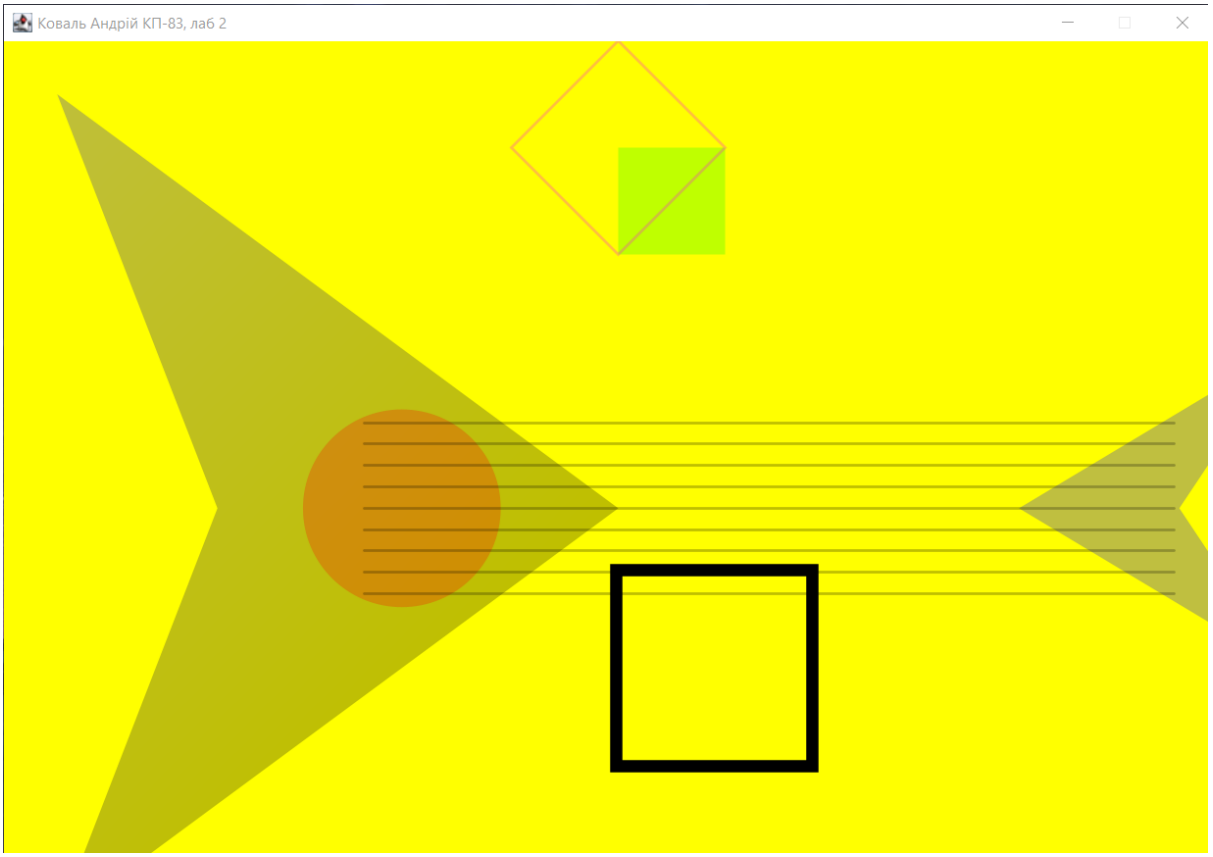
```
package com.lab2;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Ellipse2D;
import java.awt.geom.GeneralPath;

public class Main extends JPanel implements ActionListener {
    private static int MAX_WIDTH = 1000;
    private static int MAX_HEIGHT = 700;

    private static double HALF_HEIGHT = MAX_HEIGHT / 2;

    private static double PADDING = 40;
    private static double HOLE_RADIUS = PADDING * 3.7;

    private static int LINES_COUNT = 9;
    private static int LINES_BLOCK_PADDING_FROM_CIRCLE = (int)(PADDING / 5);
    private static int LINE_PADDING = (int)((HOLE_RADIUS -
LINES_BLOCK_PADDING_FROM_CIRCLE  * 2) / (LINES_COUNT - 1));
    private static int LINE_LENGTH = (int)(MAX_WIDTH / 1.65);

    private static Color BACKGROUND_COLOR = Color.YELLOW;
    private static GradientPaint LEFT_GUITAR_PART_GRADIENT = new GradientPaint(
            0, 0, Color.BLUE, 400, 400, Color.BLACK
    );
    private static Color RIGHT_GUITAR_PART_COLOR = Color.BLUE;
    private static Color GUITAR_HOLE_COLOR = Color.RED;
    private static Color STRING_COLOR = Color.BLACK;
    private static Color PRIMITIVE_COLOR = Color.GREEN;
    private static Color PRIMITIVE_LINE_COLOR = Color.MAGENTA;

    Timer timer;
    private double scale = 1;
    private double minScale = 0.5;
    private double maxScale = 1.5;
    private double dScale = 0.01;

    private double alpha = 1;
    private double minAlpha = 0.1;
    private double maxAlpha = 1;
    private double dAlpha = 0.05;

    public Main() {
        timer = new Timer(10, this);
        timer.start();
    }

    public void paint(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        RenderingHints rh = new RenderingHints(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        rh.put(RenderingHints.KEY_RENDERING, RenderingHints.VALUE_RENDER_QUALITY);
        g2d.setRenderingHints(rh);
        g2d.setBackground(BACKGROUND_COLOR);
        g2d.clearRect(0, 0, MAX_WIDTH, MAX_HEIGHT);
```

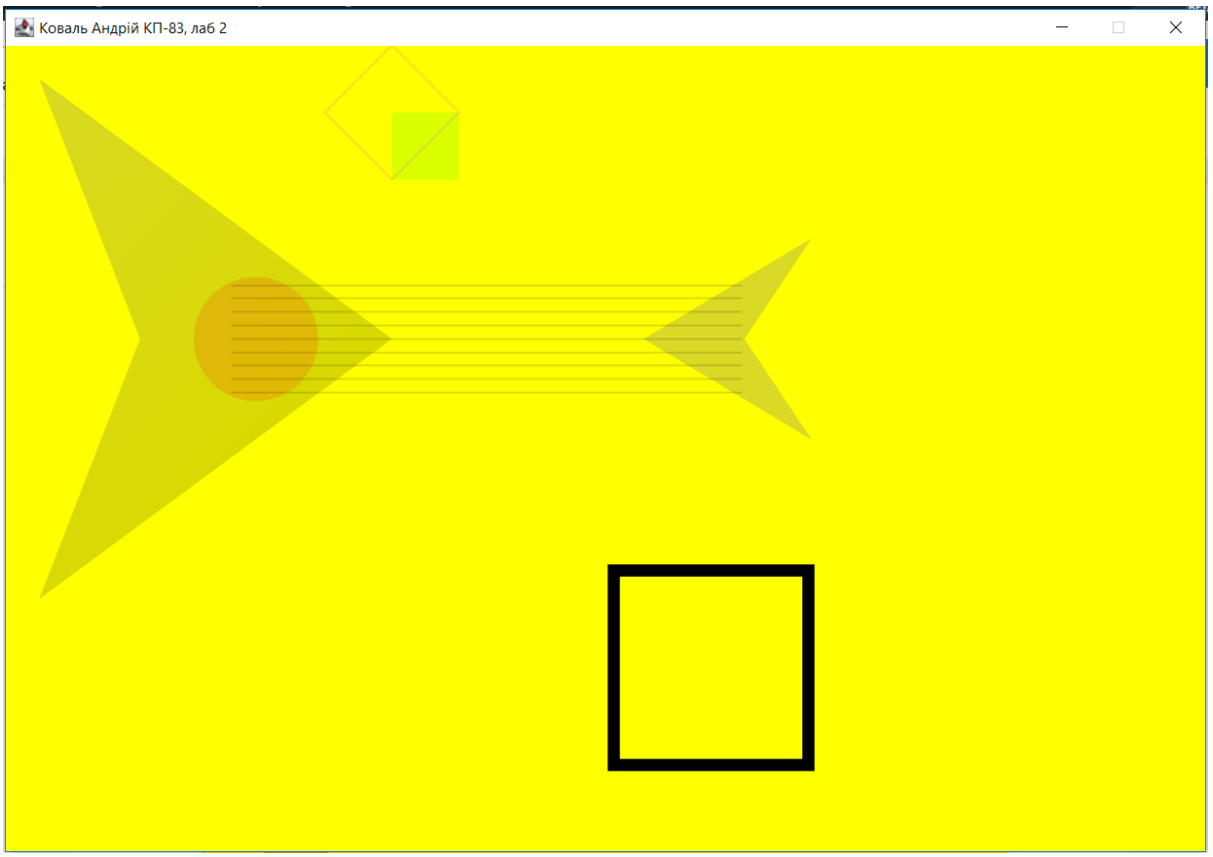
```
        attachAnimationContainer(g2d);
        g2d.scale(scale, scale);
        g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
(float)alpha));

        attachLeftGuitarPart(g2d);
        attachRightGuitarBackground(g2d);
        attachGuitarCircle(g2d);
        attachLines(g2d);
        attachPrimitiveWithLine(g2d);
    }

    private void attachLeftGuitarPart(Graphics2D g2d) {
        g2d.setPaint(LEFT_GUITAR_PART_GRADIENT);
        double [][] points = {
                {PADDING, PADDING},
                {MAX_WIDTH / 2 - PADDING, HALF_HEIGHT},
                {PADDING, MAX_HEIGHT - PADDING},
                {PADDING * 4, HALF_HEIGHT}
        };
        GeneralPath leftPart = new GeneralPath();
        leftPart.moveTo(points[0][0], points[0][1]);

        for (int i = 1; i < points.length; i++) {
            double x = points[i][0];
            double y = points[i][1];
            leftPart.lineTo(x, y);
        }

        leftPart.closePath();
        g2d.fill(leftPart);
    };

    private void attachRightGuitarBackground(Graphics2D g2d) {
        g2d.setPaint(RIGHT_GUITAR_PART_COLOR);
        double [][] points = {
                {MAX_WIDTH - PADDING, HALF_HEIGHT - 3 * PADDING,},
                {MAX_WIDTH - 3 * PADDING, HALF_HEIGHT,},
                {MAX_WIDTH - PADDING, HALF_HEIGHT + 3 * PADDING,},
                {MAX_WIDTH - 6 * PADDING, HALF_HEIGHT}
        };
        GeneralPath rightPart = new GeneralPath();
        rightPart.moveTo(points[0][0], points[0][1]);

        for (int i = 1; i < points.length; i++) {
            double x = points[i][0];
            double y = points[i][1];
            rightPart.lineTo(x, y);
        }
        rightPart.closePath();
        g2d.fill(rightPart);
    }

    private void attachGuitarCircle(Graphics2D g2d) {
        g2d.setPaint(GUITAR_HOLE_COLOR);
        double centerShift = HOLE_RADIUS / 2;

        g2d.fill(new Ellipse2D.Double(
                PADDING * 5.6,
                HALF_HEIGHT - centerShift,
                HOLE_RADIUS,
                HOLE_RADIUS
        ));
    }
```

```
    private void attachLines(Graphics2D g2d) {
        g2d.setPaint(STRING_COLOR);
        g2d.setStroke(new BasicStroke(2, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_MITER));
        int linesBlockHeight = (LINES_COUNT - 1) * LINE_PADDING;
        int firstLineYPos = (int)(HALF_HEIGHT - linesBlockHeight / 2);
        int firstLineXPos = (int)(MAX_WIDTH / 3.7);
        for (int i = 0; i < LINES_COUNT; i++) {
            g2d.drawLine(
                    firstLineXPos,
                    firstLineYPos + LINE_PADDING * i,
                    firstLineXPos + LINE_LENGTH,
                    firstLineYPos + LINE_PADDING * i
            );
        }
    }

    private void attachPrimitiveWithLine(Graphics2D g2d) {
        int squareSize = (int)(PADDING * 2);
        int x = MAX_WIDTH / 2 - squareSize / 2;
        int y = (int)(PADDING * 3) - squareSize / 2;
        g2d.setPaint(PRIMITIVE_COLOR);
        g2d.fillRect(x, y, squareSize, squareSize);
        GeneralPath primitiveLine = new GeneralPath();
        g2d.setPaint(PRIMITIVE_LINE_COLOR);
        primitiveLine.moveTo(x, y - squareSize);
        primitiveLine.lineTo(x + squareSize, y);
        primitiveLine.lineTo(x, y + squareSize);
        primitiveLine.lineTo(x - squareSize, y);
        primitiveLine.lineTo(x, y - squareSize);
        g2d.draw(primitiveLine);
    }

    private void attachAnimationContainer(Graphics2D g2d) {
        BasicStroke bs2 = new BasicStroke(10, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_MITER);
        int cx = MAX_WIDTH / 2;
        int cy = MAX_HEIGHT - (int)(PADDING * 2.7);
        int rectSize = (int)(PADDING * 4);
        g2d.setStroke(bs2);
        g2d.drawRect(cx, cy - rectSize, rectSize, rectSize);
    }

    public static void main (String[] args) {
        JFrame frame = new JFrame("Коваль Андрій КП-83, лаб 2");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(MAX_WIDTH, MAX_HEIGHT);
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.add(new Main());
        frame.setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (scale > maxScale || scale < minScale) {
            dScale *= -1;
        }
        scale += dScale;

        if (alpha >= maxAlpha || alpha <= minAlpha) {
            dAlpha *= -1;
        }
        alpha += dAlpha;
```

```
        repaint();
    }
}
```

Результат




Коваль Андрій КП-83, лаб 2