

МIНIСТЕРСТВО ОСВIТИ І НАУКИ 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 компютерної графіки”

тема “ Побудова та анімація зображень за допомогою Java2D”

|  |  |  |
| --- | --- | --- |
| Виконав:  студент III курсу  групи КП-72  Матюха Микита Володимирович  варіант № 13  Сумарний бал: |  | Перевірила  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2020р.  викладач  Шкурат Оксана Сергіївна |

|  |  |
| --- | --- |
|  |  |

Київ 2020

**Мета:** Ознайомитися з можливостями побудови зображень та їх анімації у Java2D

**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.
4. Виконати анімацію малюнку, за варіантом. При цьому рамка повинна залишатися статичною. Взаємодія з рамкою не обов’язкова, якщо не передбачено варіантом.

**Варіант 13:**

Тип анімації:

Рух по квадрату проти годинникової стрілки та зміна прозорості

**Текст коду програм**

|  |
| --- |
| App.java |
| @SuppressWarnings("serial")  public class App extends JPanel implements ActionListener {  Timer timer;  private static int maxWidth;  private static int maxHeight;  private static int alpha = 0;  private static boolean alphaInc = true;  private static double angle = 0;  public App() {  timer = new Timer(10, this);  timer.start();  }  public void paint(Graphics g) {  Graphics2D graphics2d = (Graphics2D) g;  RenderingHints hints = new RenderingHints(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);  hints.put(RenderingHints.KEY\_RENDERING, RenderingHints.VALUE\_RENDER\_QUALITY);  graphics2d.setRenderingHints(hints);  fillBlackBackground(graphics2d);  drawPicture(graphics2d);  drawPrimitive(graphics2d);  drawComplex(graphics2d);  drawAnimation(graphics2d);  }  private void drawPicture(Graphics2D g) {  drawPicture(g, 255);  }  private void drawPicture(Graphics2D g, int alpha) {  drawBlueCircle(g, alpha);  drawRedCircle(g, alpha);  drawYellowCircle(g, alpha);  drawBlackCircles(g, alpha);  drawTarget(g, alpha);  }  private void drawBlueCircle(Graphics2D g, int alpha) {  g.setColor(new Color(Color.BLUE.getRed(), Color.BLUE.getGreen(), Color.BLUE.getBlue(), alpha));  g.fillOval(100, 75, 100, 100);  }  private void drawRedCircle(Graphics2D g, int alpha) {  g.setColor(new Color(Color.RED.getRed(), Color.RED.getGreen(), Color.RED.getBlue(), alpha));  g.fillOval(120, 95, 60, 60);  }  private void drawYellowCircle(Graphics2D g, int alpha) {  g.setColor(new Color(Color.YELLOW.getRed(), Color.YELLOW.getGreen(), Color.YELLOW.getBlue(), alpha));  g.fillOval(Math.round(150 - 12.5f), Math.round(125 - 12.5f), 25, 25);  }  private void drawBlackCircles(Graphics2D g, int alpha) {  int[] radiuses = new int[] { 80, Math.round(25 + 17.5f), Math.round(12.5f) };  for (int i = 0; i < radiuses.length; i++) {  g.setColor(new Color(Color.BLACK.getRed(), Color.BLACK.getGreen(), Color.BLACK.getBlue(), alpha));  g.drawOval(150 - radiuses[i] / 2, 125 - radiuses[i] / 2, radiuses[i], radiuses[i]);  }  }  private void drawTarget(Graphics2D g, int alpha) {  g.setColor(new Color(Color.BLACK.getRed(), Color.BLACK.getGreen(), Color.BLACK.getBlue(), alpha));  g.drawLine(150 - 2, 125 + 1, 150 + 4, 125 + 1);  g.drawLine(150 + 1, 125 - 2, 150 + 1, 125 + 4);  }  private void drawPrimitive(Graphics2D g) {  g.setColor(Color.WHITE);  g.fillOval(300, 50, 100, 100);  }  private void drawComplex(Graphics2D g) {  g.translate(20, 300);  GradientPaint gp = new GradientPaint(0, 0, Color.GREEN, 3, 12, Color.YELLOW, true);  g.setPaint(gp);  int[][] points = { { 0, 60 }, { 20, 0 }, { 40, 60 }, { 30, 60 }, { 30, 120 }, { 10, 120 }, { 10, 60 },  { 0, 60 } };  GeneralPath arrow = new GeneralPath();  arrow.moveTo(points[0][0], points[0][1]);  for (int i = 1; i < points.length; i++) {  arrow.lineTo(points[i][0], points[i][1]);  }  arrow.closePath();  g.fill(arrow);  g.translate(-20, -300);  }  private void drawAnimation(Graphics2D g) {  int offsetX = 400;  int offsetY = 400;  g.setColor(Color.WHITE);  g.translate(offsetX, offsetY);  g.drawRect(0, 0, maxWidth - offsetX - 1, maxHeight - offsetY - 1);  g.rotate(angle, (maxWidth - offsetX - 1) / 2, (maxHeight - offsetY - 1) / 2);  drawPicture(g, alpha);  }  private void fillBlackBackground(Graphics2D g) {  g.setBackground(Color.BLACK);  g.clearRect(0, 0, maxWidth, maxHeight);  }  private static void setSizes(JFrame frame) {  Dimension size = frame.getSize();  Insets insets = frame.getInsets();  maxWidth = size.width - insets.left - insets.right;  maxHeight = size.height - insets.top - insets.bottom;  }  public void actionPerformed(ActionEvent e) {  angle += 0.01;  if (alpha >= 255) {  alphaInc = false;  } else if (alpha <= 0) {  alphaInc = true;  }  if (alphaInc) {  alpha++;  } else {  alpha--;  }  repaint();  }  public static void main(String[] args) {  JFrame frame = new JFrame("Lab 2 by Vladimir Mikulin");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(1000, 1000);  frame.setLocationRelativeTo(null);  frame.setResizable(false);  frame.add(new App());  frame.setVisible(true);  setSizes(frame);  }  } |

**Висновки**

Виконавши дану лабораторну роботу я навчилась працювати з бібліотекою JavaFX та ознайомилась з можливостями побудови зображень та їх анімації у Java2D.