

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** **2**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-83  Клименко Ярослав  (*прізвище, ім’я, по батькові*)  варіант № 7 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2020

**Варіант завдання**

**Завдання**: Створити малюнок за варіантом користуючись графічними примітивами бібліотеки JavaFX.

**Варіант 7:**

Анімації:

* Рух по квадрату проти годинникової стрілки
* Обертання навколо центру малюнка за годинниковою стрілкою

Тип лінії рамки:

* JOIN\_BEVEL



**Лістинг коду програми**

import java.awt.\*;  
import javax.swing.JFrame;  
import javax.swing.JPanel;  
import javax.swing.Timer;  
  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.geom.GeneralPath;  
  
enum RectSidesState {  
 *DOWN*,  
 *LEFT*,  
 *UP*,  
 *RIGHT*,  
}  
  
public class Bug extends JPanel implements ActionListener {  
  
 private static int *maxWidth*;  
 private static int *maxHeight*;  
  
 private double tx = 0;  
 private double ty = 0;  
  
 private RectSidesState currMoveDirection = RectSidesState.*RIGHT*;  
  
 Timer timer;  
 private double angle = 0;  
  
 public Bug() {  
 timer = new Timer(10, this);  
 timer.start();  
 }  
  
 public void paint(Graphics g) {  
 super.paint(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(new Color(0, 128, 128));  
 g2d.clearRect(0, 0, *maxWidth*, *maxHeight*);  
  
  
 GradientPaint gp = new GradientPaint(5, 25,  
 new Color(255, 255, 0), 20, 2, new Color(0, 255, 0), true);  
 g2d.setPaint(gp);  
  
 // BORDER  
 g2d.setColor(new Color(4, 196, 68));  
 BasicStroke bs1 = new BasicStroke(10, BasicStroke.*CAP\_ROUND*,  
 BasicStroke.*JOIN\_BEVEL*);  
 g2d.setStroke(bs1);  
 int interval = 10;  
 g2d.drawRect(interval, interval, *maxWidth* - interval \* 2, *maxHeight* - interval \* 2);  
  
  
// BUG Central Axis  
 g2d.translate(tx, ty);  
  
 GeneralPath axis = new GeneralPath();  
 axis.moveTo(130, 135);  
 axis.closePath();  
  
 g2d.rotate(angle, axis.getBounds2D().getCenterX(),  
 axis.getBounds2D().getCenterY());  
  
  
  
// BODY 1  
 double pointsBody1[][] = {  
 {60, 125}, {140, 50}, {280, 100}, {200, 145},  
 };  
  
 GeneralPath body1 = new GeneralPath();  
 body1.moveTo(pointsBody1[0][0], pointsBody1[0][1]);  
 for (int k = 1; k < pointsBody1.length; k++)  
 body1.lineTo(pointsBody1[k][0], pointsBody1[k][1]);  
 body1.closePath();  
 g2d.fill(body1);  
  
  
// BODY 2  
 double pointsBody2[][] = {  
 {100, 220}, {60, 125}, {200, 145}, {250, 200},  
 };  
  
 GeneralPath body2 = new GeneralPath();  
 body2.moveTo(pointsBody2[0][0], pointsBody2[0][1]);  
 for (int k = 1; k < pointsBody2.length; k++)  
 body2.lineTo(pointsBody2[k][0], pointsBody2[k][1]);  
  
 body2.closePath();  
  
// ANIMATION  
 g2d.fill(body2);  
  
// TAIL  
 g2d.setPaint(Color.*yellow*);  
  
 double pointsTail[][] = {  
 {215.0, 145.0}, {270.0, 120.0}, {255.0, 183.0},  
 };  
  
 GeneralPath tail = new GeneralPath();  
 tail.moveTo(pointsTail[0][0], pointsTail[0][1]);  
 for (int k = 1; k < pointsTail.length; k++)  
 tail.lineTo(pointsTail[k][0], pointsTail[k][1]);  
 tail.closePath();  
 g2d.fill(tail);  
  
 g2d.setPaint(Color.*black*);  
 g2d.setStroke(new BasicStroke(5));  
  
// ANTENNA 1  
 g2d.drawLine(30, 50, 75, 115);  
// ANTENNA 2  
 g2d.drawLine(30, 190, 75, 145);  
  
// DELIMITER  
 g2d.setStroke(new BasicStroke(2));  
  
 g2d.drawLine(60, 125, 200, 145);  
  
// EYES  
 g2d.setPaint(new Color(0, 128, 0));  
  
 g2d.fillRect(120, 100, 8, 8);  
 g2d.fillRect(110, 150, 8, 8);  
  
  
  
  
 }  
  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("lab2");  
 frame.add(new Bug());  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(900, 900);  
 frame.setLocationRelativeTo(null);  
 frame.setResizable(false);  
 frame.setVisible(true);  
//  
 Dimension size = frame.getSize();  
 Insets insets = frame.getInsets();  
 *maxWidth* = size.width - insets.left - insets.right - 1;  
 *maxHeight* = size.height - insets.top - insets.bottom - 1;  
  
 System.*out*.println(*maxWidth*);  
 System.*out*.println(*maxHeight*);  
 }  
  
 public void actionPerformed(ActionEvent e) {  
  
 int animationStep = 4;  
 int borderPaddingHeightTop = 0;  
 int borderPaddingWidthBottomLeft = 0;  
 int borderPaddingWidthBottomRight = 300;  
 int borderPaddingHeightBottom = 250;  
  
  
 if(tx <= borderPaddingWidthBottomLeft && ty <= borderPaddingHeightTop) {  
 currMoveDirection = RectSidesState.*RIGHT*;  
 } else if (tx >= *maxWidth* - borderPaddingWidthBottomRight && ty <= borderPaddingHeightTop) {  
 currMoveDirection = RectSidesState.*DOWN*;  
 } else if (tx >= *maxWidth* - borderPaddingWidthBottomRight && ty >= *maxHeight* - borderPaddingHeightBottom) {  
 currMoveDirection = RectSidesState.*LEFT*;  
 } else if (tx <= borderPaddingWidthBottomLeft && ty >= *maxHeight* - borderPaddingHeightBottom) {  
 currMoveDirection = RectSidesState.*UP*;  
 }  
  
 switch (currMoveDirection) {  
 case *UP*: {  
 ty-=animationStep;  
 }  
 break;  
 case *DOWN*: {  
 ty+=animationStep;  
  
 }  
 break;  
 case *LEFT*: {  
 tx-=animationStep;  
  
 }  
 break;  
 case *RIGHT*: {  
 tx+=animationStep;  
  
 }  
 break;  
  
 }  
  
 angle += 0.01;  
  
 repaint();  
 }  
}

**Результат**

