

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“

**Тема:** “Побудова та анімація зображень за допомогою Java2D“

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-72  Кузнєцов Ілля Іванович  (*прізвище, ім’я, по батькові*)  варіант № 10 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2020

**Варіант завдання**

**Завдання**:

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

Додатково виконати:

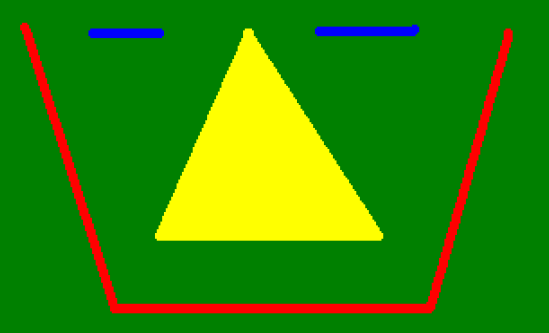
1. Хоча б 1 стандартний примітив, та хоча б 1 фігуру, побудовану по точкам (ламаною).

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:**



**Лістинг коду програми**

**package** sample;

**import** **java.awt.\***;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **java.awt.geom.GeneralPath**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JPanel**;

**import** **javax.swing.Timer**;

**@SuppressWarnings**("serial")

**public** **class** **Skeleton** **extends** JPanel **implements** ActionListener {

**private** **static** **int** maxWidth;

**private** **static** **int** maxHeight;

Timer timer;

**private** **double** angle = **0**;

**private** **double** scale = **0.5**;

**private** **double** delta = **0.01**;

**public** **Skeleton**() {

timer = **new** Timer(**50**, **this**);

timer.start();

}

**public** **void** **paint**(Graphics g) {

**super**.paint(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setBackground(Color.GREEN);

g2d.clearRect(**0**, **0**, maxWidth, maxHeight);

BasicStroke bs1 = **new** BasicStroke(**15**, BasicStroke.CAP\_ROUND,

BasicStroke.JOIN\_BEVEL);

g2d.setStroke(bs1);

g2d.drawRect(**15**, **15**, maxWidth-**30**, maxHeight-**30**);

g2d.translate( **150** , **150** );

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING,

RenderingHints.VALUE\_ANTIALIAS\_ON);

g2d.setRenderingHint(RenderingHints.KEY\_RENDERING,

RenderingHints.VALUE\_RENDER\_QUALITY);

bs1 = **new** BasicStroke(**5**, BasicStroke.CAP\_ROUND,

BasicStroke.JOIN\_BEVEL);

g2d.setStroke(bs1);

**int** xPoints[] = **new** **int**[] {**100**, **130**, **220**, **250**};

**int** yPoints[] = **new** **int**[] {**100**, **200**, **200**, **100**};

GradientPaint gp = **new** GradientPaint(**5**, **25**,

Color.YELLOW, **20**, **2**, Color.BLUE, **true**);

**double** points[][] = {

{ **140**, **180** }, { **210**, **180** }, { **175**, **105** }, { **140**, **180** }

};

GeneralPath triangle = **new** GeneralPath();

triangle.moveTo(points[**0**][**0**], points[**0**][**1**]);

**for** (**int** k = **1**; k < points.length; k++)

triangle.lineTo(points[k][**0**], points[k][**1**]);

triangle.closePath();

g2d.rotate(angle, **220**, **200**);

g2d.scale(scale, scale);

g2d.setColor(Color.RED);

g2d.drawPolyline(xPoints, yPoints, xPoints.length);

g2d.setPaint(gp);

g2d.fill(triangle);

g2d.setColor(Color.BLUE);

g2d.drawLine(**120**, **105**, **150**, **105**);

g2d.drawLine(**200**, **105**, **230**, **105**);

}

**public** **static** **void** **main**(String[] args) {

JFrame frame = **new** JFrame("Lab 2");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(**700**, **700**);

frame.setLocationRelativeTo(**null**); //centre

frame.setResizable(**false**);

frame.add(**new** Skeleton());

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**;

}

**public** **void** **actionPerformed**(ActionEvent e) {

**if** ( scale < **0.1** ) {

delta = -delta;

} **else** **if** (scale > **0.99**) {

delta = -delta;

}

scale += delta;

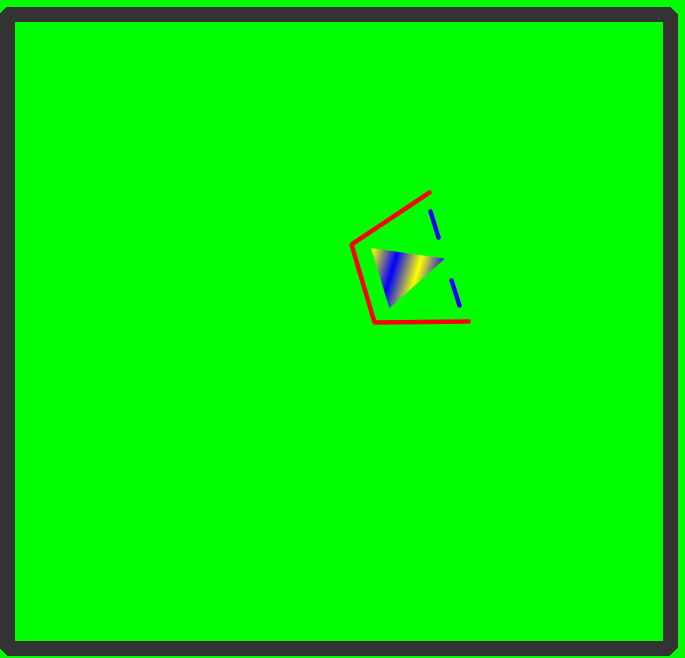
angle += **0.05**;

repaint();

}

}

**Результат**

****