

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 №** **3**

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

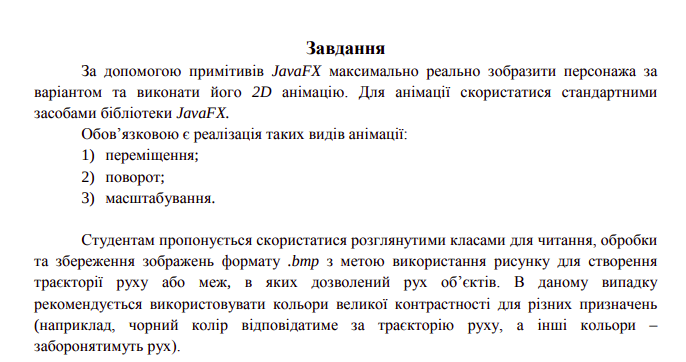
|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-81  Черняєв Іван  (*прізвище, ім’я, по батькові*)  варіант № 19 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

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

**Варіант 19**



**Лістинг коду програми**

package sample;import sample.bmp.HeaderBitmapImage;import sample.bmp.ReadingHeaderFromBitmapImage;import javafx.animation.\*;import javafx.application.Application;import javafx.scene.Group;import javafx.scene.Scene;import javafx.scene.paint.Color;import javafx.scene.shape.\*;import javafx.stage.Stage;import javafx.util.Duration;import java.io.\*;public class Main extends Application { private int x = 2; public static void main(String args[]) { *launch*(args); } @Override public void start(Stage primaryStage) throws IOException { Group root = new Group(); Scene scene = new Scene(root, 1000, 600); { Path p = new Path(); p.setStrokeWidth(3 \* x); p.setStroke(Color.*BLACK*); p.setFill(Color.*rgb*(213, 33, 45)); p.getElements().add(new MoveTo(19 \* x, 67 \* x)); p.getElements().add(new ArcTo(50 \* x, 65 \* x, 0, 60 \* x, 28 \* x, false, true)); p.getElements().add(new ArcTo(22 \* x, 30 \* x, 30, 43 \* x, 26 \* x, false, true)); p.getElements().add(new ArcTo(7 \* x, 7 \* x, 0, 44 \* x, 14 \* x, false, true)); p.getElements().add(new ArcTo(25 \* x, 20 \* x, 15, 74 \* x, 25 \* x, false, true)); p.getElements().add(new QuadCurveTo(64 \* x, 14.5 \* x, 58 \* x, 14.5 \* x)); p.getElements().add(new ArcTo(10 \* x, 10 \* x, 35, 71 \* x, 8 \* x, false, true)); p.getElements().add(new QuadCurveTo(80 \* x, 11 \* x, 90 \* x, 28 \* x)); p.getElements().add(new ArcTo(45 \* x, 55 \* x, 0, 123 \* x, 68 \* x, false, true)); p.getElements().add(new ArcTo(45 \* x, 55 \* x, 0, 122 \* x, 88 \* x, false, true)); p.getElements().add(new ArcTo(53 \* x, 40 \* x, 0, 18 \* x, 84 \* x, false, true)); p.getElements().add(new ArcTo(50 \* x, 65 \* x, 0, 19 \* x, 67 \* x, false, true)); root.getChildren().add(p); Path tail = new Path(); tail.setFill(Color.*BLACK*); tail.setStrokeWidth(x); tail.getElements().addAll( new MoveTo(17 \* x, 75 \* x), new LineTo(11 \* x, 77 \* x), new LineTo(10 \* x, 75 \* x), new LineTo(16 \* x, 72 \* x), new LineTo(16 \* x, 70 \* x), new LineTo(2 \* x, 68 \* x), new LineTo(5 \* x, 61 \* x), new LineTo(18 \* x, 67 \* x), new LineTo(18 \* x, 65 \* x), new LineTo(12 \* x, 57 \* x), new LineTo(16 \* x, 54 \* x), new LineTo(20 \* x, 61 \* x) ); root.getChildren().add(tail); Path t1 = new Path(); t1.setFill(Color.*rgb*(163, 23, 34)); t1.setStrokeWidth(0); t1.getElements().addAll( new MoveTo(43 \* x, 82 \* x), new QuadCurveTo(47 \* x, 86 \* x, 46 \* x, 90 \* x), new ArcTo(5 \* x, 10 \* x, -15, 41 \* x, 92 \* x, false, true), new QuadCurveTo(37 \* x, 89 \* x, 38 \* x, 84 \* x), new ArcTo(5 \* x, 10 \* x, -15, 43 \* x, 82 \* x, false, true) ); root.getChildren().add(t1); Path t2 = new Path(); t2.setFill(Color.*rgb*(163, 23, 34)); t2.setStrokeWidth(0); t2.getElements().addAll( new MoveTo(51 \* x, 74 \* x), new ArcTo(5 \* x, 10 \* x, -15, 58 \* x, 75 \* x, false, true), new QuadCurveTo(62 \* x, 81 \* x, 59 \* x, 89 \* x), new ArcTo(5 \* x, 10 \* x, -15, 51 \* x, 88 \* x, false, true), new QuadCurveTo(49 \* x, 82 \* x, 51 \* x, 74 \* x) ); root.getChildren().add(t2); Path t3 = new Path(); t3.setFill(Color.*rgb*(163, 23, 34)); t3.setStrokeWidth(0); t3.getElements().addAll( new MoveTo(74 \* x, 70 \* x), new CubicCurveTo(63 \* x, 79 \* x, 63 \* x, 88 \* x, 74 \* x, 97 \* x), new LineTo(94 \* x, 86 \* x), new LineTo(74 \* x, 70 \* x) ); root.getChildren().add(t3); Path t4 = new Path(); t4.setFill(Color.*rgb*(163, 23, 34)); t4.setStrokeWidth(0); t4.getElements().addAll( new MoveTo(107 \* x, 93 \* x), new CubicCurveTo(116 \* x, 85 \* x, 117 \* x, 78 \* x, 108 \* x, 70 \* x), new LineTo(95 \* x, 82 \* x), new LineTo(107 \* x, 93 \* x) ); root.getChildren().add(t4); Path eye\_right = new Path(); eye\_right.setFill(Color.*WHITE*); eye\_right.setStrokeWidth(x); eye\_right.getElements().addAll( new MoveTo(104 \* x, 85 \* x), new CubicCurveTo(114 \* x, 80 \* x, 114 \* x, 72 \* x, 109 \* x, 67 \* x), new QuadCurveTo(85 \* x, 67 \* x, 94 \* x, 71 \* x), new QuadCurveTo(81 \* x, 82 \* x, 94 \* x, 78 \* x), new QuadCurveTo(97 \* x, 85 \* x, 104 \* x, 85 \* x) ); root.getChildren().add(eye\_right); Path eye\_left = new Path(); eye\_left.setFill(Color.*WHITE*); eye\_left.setStrokeWidth(x); eye\_left.getElements().addAll( new MoveTo(94 \* x, 71 \* x), new QuadCurveTo(85 \* x, 67 \* x, 76 \* x, 66 \* x), new CubicCurveTo(69 \* x, 74 \* x, 71 \* x, 83 \* x, 84 \* x, 84 \* x), new QuadCurveTo(89 \* x, 82 \* x, 94 \* x, 78 \* x), new VLineTo(71 \* x) ); root.getChildren().add(eye\_left); root.getChildren().add(new Circle(100 \* x, 75 \* x, 3.2 \* x, Color.*BLACK*)); root.getChildren().add(new Circle(85 \* x, 75 \* x, 3.5 \* x, Color.*BLACK*)); Path eyebrows = new Path(); eyebrows.setFill(Color.*BLACK*); eyebrows.setStrokeWidth(x); eyebrows.getElements().addAll( new MoveTo(120 \* x, 57 \* x), new LineTo(94 \* x, 65 \* x), new LineTo(66 \* x, 56 \* x), new LineTo(64 \* x, 63 \* x), new LineTo(94 \* x, 71 \* x), new LineTo(122 \* x, 64 \* x) ); root.getChildren().add(eyebrows); Path white\_spot = new Path(); white\_spot.setFill(Color.*rgb*(225, 195, 171)); white\_spot.setStrokeWidth(0); white\_spot.getElements().addAll( new MoveTo(36 \* x, 107 \* x), new CubicCurveTo(63 \* x, 86 \* x, 82 \* x, 86 \* x, 105 \* x, 107 \* x), new ArcTo(53 \* x, 41.5 \* x, 0, 36 \* x, 107 \* x, false, true) ); root.getChildren().add(white\_spot); Path beak\_bot = new Path(); beak\_bot.setFill(Color.*YELLOW*); beak\_bot.setStrokeWidth(x); beak\_bot.getElements().addAll( new MoveTo(76 \* x, 92 \* x), new CubicCurveTo(90 \* x, 110 \* x, 94 \* x, 110 \* x, 108 \* x, 98 \* x), new QuadCurveTo(110 \* x, 89 \* x, 94 \* x, 78 \* x), new QuadCurveTo(110 \* x, 89 \* x, 75 \* x, 92 \* x) ); root.getChildren().add(beak\_bot); Path beak\_top = new Path(); beak\_top.setFill(Color.*rgb*(254, 183, 31)); beak\_top.setStrokeWidth(x); beak\_top.getElements().addAll( new MoveTo(75 \* x, 92 \* x), new QuadCurveTo(88 \* x, 97 \* x, 115 \* x, 97 \* x), new QuadCurveTo(110 \* x, 89 \* x, 94 \* x, 78 \* x), new QuadCurveTo(87 \* x, 81 \* x, 75 \* x, 92 \* x) ); root.getChildren().add(beak\_top); } int transitionTimeSeconds = 5; FadeTransition fadeTransition = new FadeTransition(Duration.*seconds*(transitionTimeSeconds)); fadeTransition.setFromValue(1); fadeTransition.setToValue(0.5); RotateTransition rotateTransition = new RotateTransition(Duration.*seconds*(transitionTimeSeconds)); rotateTransition.setByAngle(360); ScaleTransition scaleTransition = new ScaleTransition(Duration.*seconds*(transitionTimeSeconds), root); scaleTransition.setToX(2); scaleTransition.setToY(-2); PathTransition pathTransition = new PathTransition(Duration.*seconds*(transitionTimeSeconds), getTrajectoryPath(), root); ParallelTransition parallelTransition = new ParallelTransition(root); parallelTransition.getChildren().addAll(fadeTransition, scaleTransition, rotateTransition, pathTransition); parallelTransition.setCycleCount(Animation.*INDEFINITE*); parallelTransition.setAutoReverse(true); parallelTransition.play(); primaryStage.setResizable(false); primaryStage.setTitle("Lab3"); primaryStage.setScene(scene); primaryStage.show(); } private Path getTrajectoryPath() throws IOException { int numberOfPixels = 0; FileInputStream fileInputStream = new FileInputStream("resources/trajectory2.bmp"); BufferedInputStream bufferedInputStream = new BufferedInputStream(fileInputStream); HeaderBitmapImage image = new ReadingHeaderFromBitmapImage().Reading(bufferedInputStream); int width = (int) image.getWidth(); int height = (int) image.getHeight(); int half = (int) image.getHalfOfWidth(); int let, let1, let2; char[][] map = new char[width][height]; BufferedInputStream reader = new BufferedInputStream(new FileInputStream("pixels.txt")); for (int i = 0; i < height; i++) { for (int j = 0; j < half; j++) { let = reader.read(); let1 = (let & (0xf0)) >> 4; let2 = let & (0x0f); if (j \* 2 < width) { if (returnPixelColor(let1).equals("BLACK")) { map[j \* 2][height - 1 - i] = '1'; numberOfPixels++; } else { map[j \* 2][height - 1 - i] = '0'; } } if (j \* 2 + 1 < width) { if (returnPixelColor(let2).equals("BLACK")) { map[j \* 2 + 1][height - 1 - i] = '1'; numberOfPixels++; } else { map[j \* 2 + 1][height - 1 - i] = '0'; } } } } reader.close(); int[][] black = new int[numberOfPixels][2]; int lich = 0; BufferedOutputStream writer = new BufferedOutputStream(new FileOutputStream("map.txt")); for (int i = 0; i < height; i++) { for (int j = 0; j < width; j++) { if (map[j][i] == '1') { black[lich][0] = j; black[lich][1] = i; lich++; } writer.write(map[j][i]); } writer.write(10); } writer.close(); System.*out*.println("number of black color pixels = " + numberOfPixels); Path path = new Path(); for (int l = 0; l < numberOfPixels - 1; l++) { path.getElements().addAll(new MoveTo(black[l][0], black[l][1]), new LineTo(black[l + 1][0], black[l + 1][1]) ); } return path; } private String returnPixelColor(int color) { String col = "BLACK"; switch (color) { case 0: return "BLACK"; //BLACK; case 1: return "LIGHTCORAL"; //LIGHTCORAL; case 2: return "GREEN"; //GREEN case 3: return "BROWN"; //BROWN case 4: return "BLUE"; //BLUE; case 5: return "MAGENTA"; //MAGENTA; case 6: return "CYAN"; //CYAN; case 7: return "LIGHTGRAY"; //LIGHTGRAY; case 8: return "DARKGRAY"; //DARKGRAY; case 9: return "RED"; //RED; case 10: return "LIGHTGREEN";//LIGHTGREEN case 11: return "YELLOW"; //YELLOW; case 12: return "LIGHTBLUE"; //LIGHTBLUE; case 13: return "LIGHTPINK"; //LIGHTMAGENTA case 14: return "LIGHTCYAN"; //LIGHTCYAN; case 15: return "WHITE"; //WHITE; } return col; }}

**Результат**

