

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-82  Кривчук Денис  (*прізвище, ім’я, по батькові*)  варіант № 9 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2020

**Варіант завдання**

**Завдання**: За допомогою примітивів JavaFX максимально реально зобразити персонажа за варіантом та виконати його 2D анімацію. Для анімації скористатися стандартними засобами бібліотеки JavaFX. Обов’язковою є реалізація таких видів анімації:

1) переміщення;

2) поворот;

3) масштабування.

Студентам пропонується скористатися розглянутими класами для читання, обробки та збереження зображень формату .bmp з метою використання рисунку для створення траєкторії руху або меж, в яких дозволений рух об’єктів. В даному випадку рекомендується використовувати кольори великої контрастності для різних призначень (наприклад, чорний колір відповідатиме за траєкторію руху, а інші кольори – заборонятимуть рух).

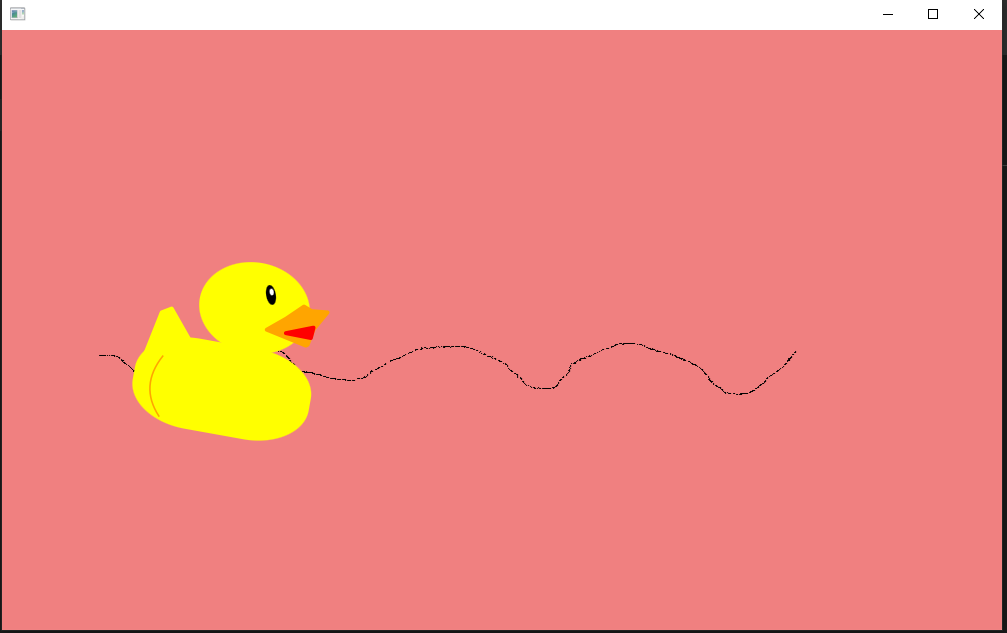
**Варіант:**



**Лістинг коду програми**

package sample;  
  
import java.io.BufferedInputStream;  
import java.io.BufferedOutputStream;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
  
import javafx.animation.FadeTransition;  
import javafx.animation.RotateTransition;  
import javafx.animation.ScaleTransition;  
import javafx.application.Application;  
import javafx.scene.Group;  
import javafx.scene.Scene;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Circle;  
import javafx.scene.transform.Rotate;  
import javafx.stage.Stage;  
import javafx.scene.shape.\*;  
import javafx.animation.PathTransition;  
import javafx.util.Duration;  
  
public class PrintingImage extends Application{  
 private HeaderBitmapImage image; // приватне поле, яке зберігає об'єкт з інформацією про заголовок зображення  
 private int numberOfPixels; // приватне поле для збереження кількості пікселів з чорним кольором  
  
 public PrintingImage(){}  
  
 public PrintingImage(HeaderBitmapImage image) // перевизначений стандартний конструктор  
 {  
 this.image = image;  
 }  
  
 @Override  
 public void start(Stage primaryStage) throws Exception {  
 ReadingImageFromFile.*loadBitmapImage*("C:/Users/Dionis/Desktop/3\_2 docs/java/lab3/path.bmp");  
 this.image = ReadingImageFromFile.*pr*.image;  
 int width = (int)this.image.getWidth();  
 int height = (int)this.image.getHeight();  
 int half = (int)image.getHalfOfWidth();  
 Group root = new Group();  
 Scene scene = new Scene (root, width, height);  
 Circle cir;  
 int let = 0;  
 int let1 = 0;  
 int let2 = 0;  
 char[][] map = new char[width][height];  
  
 // виконуємо зчитування даних про пікселі  
 BufferedInputStream reader = new BufferedInputStream (new FileInputStream("pixels.txt"));  
  
 for(int i=0;i<height;i++) // поки не кінець зображення по висоті  
 {  
 for(int j=0;j<half;j++) // поки не кінець зображення по довжині  
 {  
 let = reader.read(); // зчитуємо один символ з файлу  
 let1 = let;  
 let2 = let;  
 let1 = let1&(0xf0); // старший байт - перший піксель  
 let1 = let1>>4; // зсув на 4 розряди  
 let2 = let2&(0x0f); // молодший байт - другий піксель  
 if(j\*2<width) // так як 1 символ кодує 2 пікселі нам необхідно пройти до середини ширини зображення  
 {  
 cir = new Circle ((j)\*2,(height-1-i),1,Color.*valueOf*((returnPixelColor(let1)))); // за допомогою стандартного  
 // примітива Коло радіусом в 1 піксель та кольором визначеним за допомогою методу returnPixelColor малюємо піксель  
 root.getChildren().add(cir); //додаємо об'єкт в сцену  
 if (returnPixelColor(let1) == "BLACK") // якщо колір пікселя чорний, то ставимо в масиві 1  
 {  
 map[j\*2][height-1-i] = '1';  
 numberOfPixels++; // збільшуємо кількість чорних пікселів  
 }  
 else  
 {  
 map[j\*2][height-1-i] = '0';  
 }  
 }  
  
 if(j\*2+1<width) // для другого пікселя  
 {  
 cir = new Circle ((j)\*2+1,(height-1-i),1,Color.*valueOf*((returnPixelColor(let2))));  
 root.getChildren().add(cir);  
 if (returnPixelColor(let2) == "BLACK")  
 {  
 map[j\*2+1][height-1-i] = '1';  
 numberOfPixels++;  
 }  
 else  
 {  
 map[j\*2+1][height-1-i] = '0';  
 }  
 }  
 }  
 }  
 primaryStage.setScene(scene); // ініціалізуємо сцену  
 primaryStage.show(); // візуалізуємо сцену  
 reader.close();  
 int[][] black;  
 black = new int[numberOfPixels][2];  
 int lich = 0;  
 // writing  
 BufferedOutputStream writer = new BufferedOutputStream (new FileOutputStream("map.txt")); // записуємо карту для руху по траекторії в файл  
 for(int i=0;i<width;i++) // поки не кінець зображення по висоті  
 {  
 for(int j=0;j<height;j++) // поки не кінець зображення по довжині  
 {  
 if (map[i][j] == '1')  
 {  
 black[lich][0] = i;  
 black[lich][1] = j;  
 lich++;  
 }  
 writer.write(map[i][j]);  
 }  
 writer.write(10);  
 }  
 writer.close();  
 System.*out*.println("number of black color pixels = " + numberOfPixels);  
 Path path2 = new Path();  
 for (int l=0; l<numberOfPixels-1; l++)  
 {  
 path2.getElements().addAll(  
 new MoveTo(black[l][0],black[l][1]),  
 new LineTo (black[l+1][0],black[l+1][1])  
 );  
 }  
 Group gr = new Group();  
 root.getChildren().add(gr);  
 Ellipse ellipse = new Ellipse(350,170,140, 115);  
 ellipse.setFill(Color.*YELLOW*);  
 gr.getChildren().add(ellipse);  
  
 Ellipse ellipse1 = new Ellipse(385,130,12, 25);  
 ellipse1.setFill(Color.*BLACK*);  
 ellipse1.getTransforms().add(new Rotate(-20, 385,130));  
 gr.getChildren().add(ellipse1);  
  
 Ellipse ellipse2 = new Ellipse(388,123,5, 8);  
 ellipse2.setFill(Color.*WHITE*);  
 ellipse2.getTransforms().add(new Rotate(-20, 385,130));  
 gr.getChildren().add(ellipse2);  
  
 Rectangle rect = new Rectangle(  
 80,270,  
 450,230  
 );  
 rect.setArcWidth(300);  
 rect.setArcHeight(200);  
 rect.setFill(Color.*YELLOW*);  
 gr.getChildren().add(rect);  
  
 Polygon polygon = new Polygon();  
 polygon.getPoints().addAll(new Double[]{  
 100.0, 330.0,  
 122.0, 222.0,  
 145.0, 208.0,  
 205.0, 280.0});  
 polygon.setFill(Color.*YELLOW*);  
 polygon.setStroke(Color.*YELLOW*);  
 polygon.setStrokeWidth(10);  
 polygon.setStrokeLineCap(StrokeLineCap.*ROUND*);  
 polygon.setStrokeLineJoin(StrokeLineJoin.*ROUND*);  
 gr.getChildren().add(polygon);  
  
 Polygon polygon1 = new Polygon();  
 polygon1.getPoints().addAll(new Double[]{  
 390.0, 218.0,  
 433.0, 182.0,  
 472.0, 144.0,  
 493.0, 152.0,  
 533.0, 148.0,  
 506.0, 196.0,  
 498.0, 237.0,  
 494.0, 240.0});  
 polygon1.setFill(Color.*ORANGE*);  
 polygon1.setStroke(Color.*ORANGE*);  
 polygon1.setStrokeWidth(10);  
 polygon1.setStrokeLineCap(StrokeLineCap.*ROUND*);  
 polygon1.setStrokeLineJoin(StrokeLineJoin.*ROUND*);  
 gr.getChildren().add(polygon1);  
  
 Polygon polygon2 = new Polygon();  
 polygon2.getPoints().addAll(new Double[]{  
 439.0, 218.0,  
 505.0, 192.0,  
 503.0, 219.0});  
 polygon2.setFill(Color.*RED*);  
 polygon2.setStroke(Color.*RED*);  
 polygon2.setStrokeWidth(10);  
 polygon2.setStrokeLineCap(StrokeLineCap.*ROUND*);  
 polygon2.setStrokeLineJoin(StrokeLineJoin.*ROUND*);  
 gr.getChildren().add(polygon2);  
  
 Path path = new Path();  
  
 MoveTo moveTo = new MoveTo();  
 moveTo.setX(144);  
 moveTo.setY(330);  
  
 QuadCurveTo quadTo = new QuadCurveTo();  
 quadTo.setControlX(99);  
 quadTo.setControlY(414);  
 quadTo.setX(161);  
 quadTo.setY(480);  
  
 path.getElements().add(moveTo);  
 path.getElements().add(quadTo);  
 path.setStroke(Color.*ORANGE*);  
 path.setStrokeWidth(4);  
 gr.getChildren().add(path);  
  
 FadeTransition ft = new FadeTransition(Duration.*millis*(3000), gr);  
 ft.setFromValue(1.0);  
 ft.setToValue(0.1);  
 ft.setCycleCount(5);  
 ft.setAutoReverse(true);  
// ft.play();  
  
 ScaleTransition scaleTransition = new ScaleTransition(Duration.*millis*(2000), gr);  
 scaleTransition.setToX(320);  
 scaleTransition.setToY(320);  
 scaleTransition.setFromX(0.2);  
 scaleTransition.setFromY(0.2);  
 scaleTransition.setToX(0.5);  
 scaleTransition.setToY(0.5);  
 scaleTransition.setCycleCount(100);  
 scaleTransition.setAutoReverse(true);  
 scaleTransition.play();  
  
 RotateTransition rotForArc1 =  
 new RotateTransition(Duration.*millis*(500), gr);  
 rotForArc1.setByAngle(20f);  
 rotForArc1.setCycleCount(200);  
 rotForArc1.setAutoReverse(true);  
 rotForArc1.play();  
 primaryStage.show();  
  
 PathTransition pathTransition = new PathTransition();  
 pathTransition.setDuration(Duration.*millis*(5000));  
 pathTransition.setCycleCount(20);  
 pathTransition.setAutoReverse(true);  
 pathTransition.setPath(path2);  
 pathTransition.setNode(gr);  
 pathTransition.play();  
 }  
  
 // далі необхідно зробити рух об'єкту по заданій траеторії  
 private String returnPixelColor (int color) // метод для співставлення кольорів 16-бітного зображення  
 {  
 String col = "BLACK";  
 switch(color)  
 {  
 case 0: return "BLACK";  
 case 1: return "LIGHTCORAL";  
 case 2: return "GREEN";  
 case 3: return "BROWN";  
 case 4: return "BLUE";  
 case 5: return "MAGENTA";  
 case 6: return "CYAN";  
 case 7: return "LIGHTGRAY";  
 case 8: return "DARKGRAY";  
 case 9: return "RED";  
 case 10:return "LIGHTGREEN";  
 case 11:return "YELLOW";  
 case 12:return "LIGHTBLUE";  
 case 13:return "LIGHTPINK";  
 case 14:return "LIGHTCYAN";  
 case 15:return "WHITE";  
 }  
 return col;  
 }  
  
 public static void main (String args[])  
 {  
 *launch*(args);  
 }  
}

**Результат**

****