

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 3**

з дисципліни “ МАОКГ”

|  |  |  |
| --- | --- | --- |
| Виконав  студент III курсу  групи КП-72  Демченко Владислав Миколайович  (*прізвище, ім’я, по батькові*)  Варіант № 7 |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2020 р.  викладачем  Шкурат О. С.  (*прізвище, ім’я, по батькові*) |

Київ-2020

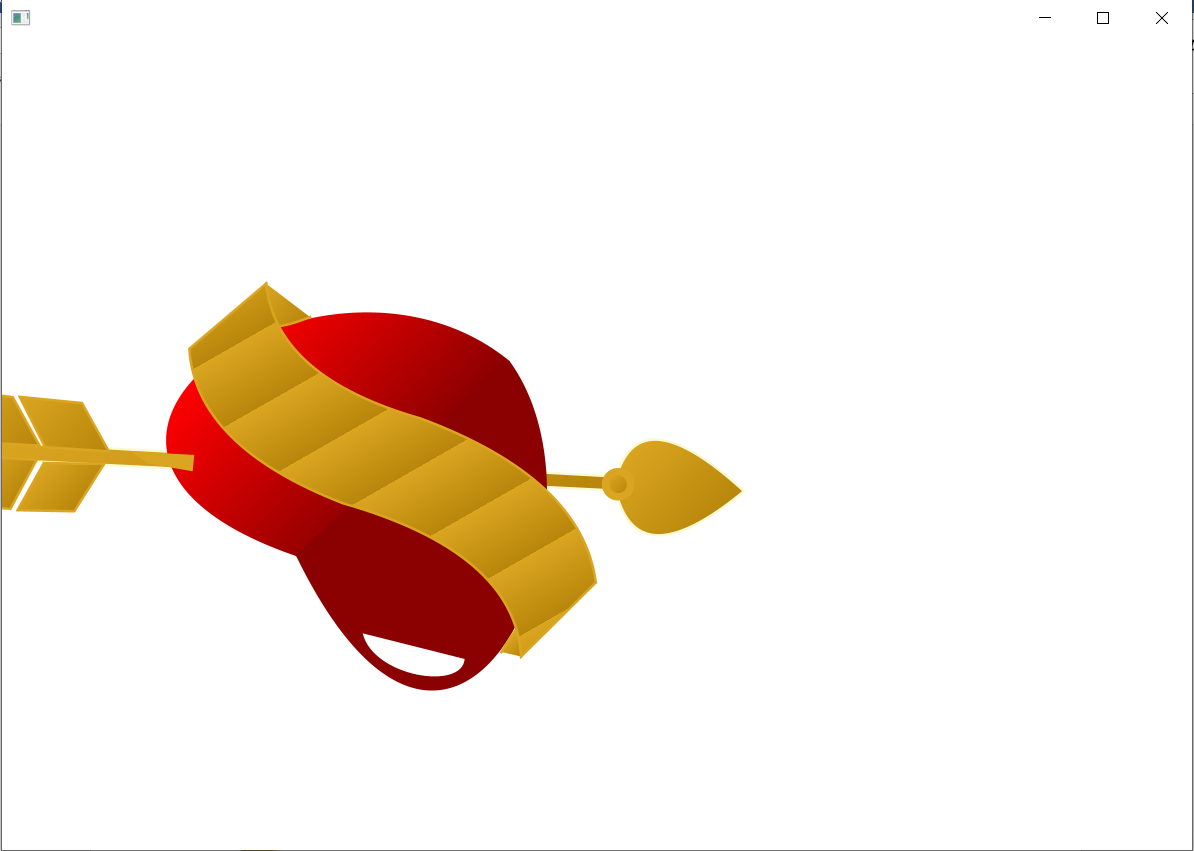
За умовою мають бути анімації:

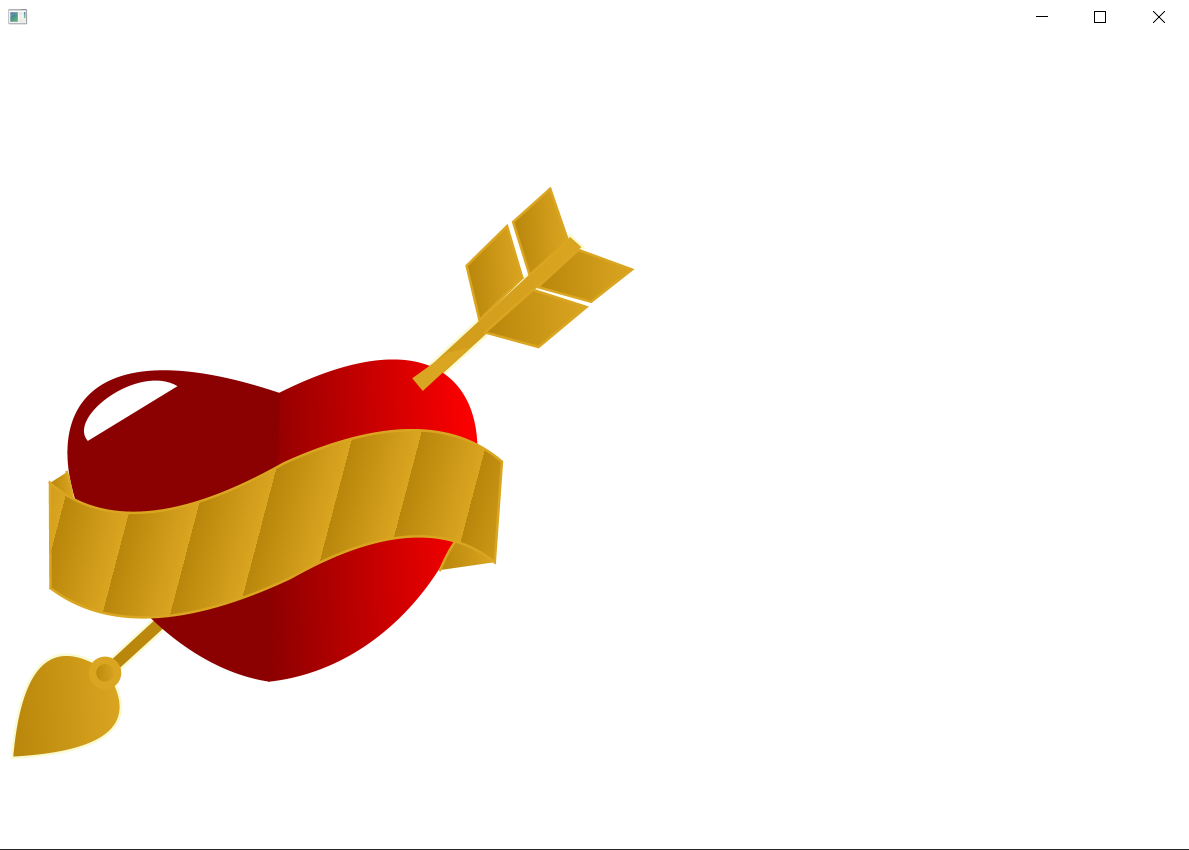
* Переміщення
* Повороту
* Масштабування

Результат(скріншоти) виконання програми:









За роботу програми відповідає головний клас – PrintingImage (код розміщений нижче):

В цьому класі відбувається зчитування траєкторії руху зображення, формування самого зображення та додавання анімацій за умовою.

package lab3;  
  
import java.io.BufferedInputStream;  
import java.io.BufferedOutputStream;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
  
import javafx.animation.\*;  
import javafx.application.Application;  
import javafx.scene.Group;  
import javafx.scene.Scene;  
import javafx.scene.paint.Color;  
import javafx.scene.paint.CycleMethod;  
import javafx.scene.paint.Stop;  
import javafx.scene.shape.\*;  
import javafx.stage.Stage;  
import javafx.util.Duration;  
import javafx.scene.paint.LinearGradient;  
  
import javax.swing.plaf.ColorUIResource;  
  
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*("D:/maokg/lab3/sources/my-trajectory.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 + 100, 250 + height);  
 scene.setFill(Color.*WHITE*);  
 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;  
   
 BufferedOutputStream writer = new BufferedOutputStream (new FileOutputStream("map.txt")); // записуємо карту для руху по траекторії в файл  
 for(int i=0;i<height;i++) // поки не кінець зображення по висоті  
 {   
 for(int j=0;j<width;j++) // поки не кінець зображення по довжині  
 {  
 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 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])  
 );  
 }  
  
  
 //animation  
 PathTransition pathTransition = new PathTransition();  
 pathTransition.setDuration(Duration.*millis*(5000));  
 pathTransition.setPath(path2);  
  
 //arrow's body  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(280, 23),  
 new LineTo(285, 28),  
 new LineTo(91, 187),  
 new LineTo(87, 183),  
 new LineTo(280, 23)  
 );  
  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*LIGHTGOLDENRODYELLOW*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 //the arrowhead  
 {  
 Path p = new Path();  
 MoveTo mv = new MoveTo(86, 187);  
 QuadCurveTo qt1 = new QuadCurveTo(54, 163, 46, 220);  
 QuadCurveTo qt2 = new QuadCurveTo(105, 220, 86, 187);  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*LIGHTGOLDENRODYELLOW*);  
 p.setFill(lg1);  
 p.getElements().addAll(mv, qt1, qt2);  
 root.getChildren().add(p);  
 }  
  
 //circle of an arrow  
 {  
 Ellipse ellipse = new Ellipse();  
 ellipse.setCenterX(85);  
 ellipse.setCenterY(188);  
 ellipse.setRadiusX(5);  
 ellipse.setRadiusY(5);  
 ellipse.setStrokeWidth(3);  
 ellipse.setStroke(Color.*GOLDENROD*);  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 ellipse.setFill(lg1);  
 root.getChildren().add(ellipse);  
 }  
  
 //The ends of an arrow  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(273, 4),  
 new LineTo(279, 25),  
 new LineTo(263, 38),  
 new LineTo(257.5, 16.5),  
 new LineTo(273, 4)  
 );  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(283, 29),  
 new LineTo(266, 43),  
 new LineTo(287, 50),  
 new LineTo(304, 38),  
 new LineTo(283, 29)  
 );  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(264, 44),  
 new LineTo(244, 60),  
 new LineTo(265, 67),  
 new LineTo(285, 52),  
 new LineTo(264, 44)  
 );  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(260, 39),  
 new LineTo(242, 55),  
 new LineTo(238, 33),  
 new LineTo(255, 18),  
 new LineTo(260, 39)  
 );  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 //Parts Of the heart  
 {  
 CubicCurve qt = new CubicCurve(160, 80, 285, 25, 245, 190, 150, 195);  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKRED*), new Stop(1, Color.*RED*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 qt.setStrokeWidth(0);  
 qt.setStroke(Color.*DARKRED*);  
 qt.setFill(lg1);  
 root.getChildren().addAll(qt);  
 }  
  
 {  
 CubicCurve qt = new CubicCurve(161, 80, 22, 25, 75, 180, 150.5, 195);  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKRED*), new Stop(1, Color.*RED*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 0, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 qt.setStrokeWidth(0);  
 qt.setStroke(Color.*RED*);  
 qt.setFill(lg1);  
 root.getChildren().addAll(qt);  
 }  
  
 {  
 CubicCurve qt = new CubicCurve(83, 95, 75, 85, 105, 65, 120, 75);  
 qt.setStrokeWidth(0);  
 qt.setStroke(Color.*WHITE*);  
 qt.setFill(Color.*WHITE*);  
 root.getChildren().addAll(qt);  
 }  
  
 // the part of arrow on the heart  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(229, 68),  
 new LineTo(215, 77),  
 new LineTo(218, 81),  
 new LineTo(235, 67),  
 new LineTo(229, 68)  
 );  
 p.setStrokeWidth(1.5);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(Color.*GOLDENROD*);  
 root.getChildren().add(p);  
 }  
  
 //The ends of the ribbon  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(243, 152),  
 new QuadCurveTo(235, 125, 221, 154)  
 );  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(67, 111),  
 new QuadCurveTo(80, 135, 74, 107)  
 );  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 1, 0, true, CycleMethod.*NO\_CYCLE*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 //The main part of the ribbon  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(65, 153),  
 new LineTo(67, 111),  
 new QuadCurveTo(97, 140, 161, 108),  
 new QuadCurveTo(220, 85, 248, 112),  
 new LineTo(243, 152),  
 new QuadCurveTo(215, 127, 161, 154),  
 new QuadCurveTo(97, 180, 65, 153)  
 );  
 Stop[] stops = new Stop[] { new Stop(0, Color.*DARKGOLDENROD*), new Stop(1, Color.*GOLDENROD*)};  
 LinearGradient lg1 = new LinearGradient(0, 0, 25, 8, false, CycleMethod.*REPEAT*, stops);  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GOLDENROD*);  
 p.setFill(lg1);  
 root.getChildren().add(p);  
 }  
  
 pathTransition.setNode(root);  
  
 //animation part  
 int cycleCount = 5;  
 int time = 2000;  
  
 ScaleTransition scaleTransition = new ScaleTransition(Duration.*millis*(time), root);  
 scaleTransition.setToX(-2);  
 scaleTransition.setToY(-2);  
 scaleTransition.setAutoReverse(true);  
  
 RotateTransition rotateTransition = new RotateTransition(Duration.*millis*(time), root);  
 rotateTransition.setByAngle(180f);  
 rotateTransition.setCycleCount(cycleCount);  
 rotateTransition.setAutoReverse(true);  
  
 ParallelTransition parallelTransition = new ParallelTransition();  
 parallelTransition.getChildren().addAll(  
 rotateTransition,  
 scaleTransition,  
 pathTransition  
 );  
 parallelTransition.setCycleCount(Timeline.*INDEFINITE*);  
 parallelTransition.play();  
  
  
 }  
   
 private String returnPixelColor (int color) // метод для співставлення кольорів 16-бітного зображення  
 {  
 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;  
 }  
   
 public static void main (String args[])   
 {  
 *launch*(args);  
 }  
  
}