

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

Шапошник Богдан Ігорович

(*прізвище, ім’я, по батькові*)

варіант № 20

Зарахована “ ” “ ” 20 р.

викладачем

Шкурат Оксаною Сергіївною

(*прізвище, ім’я, по батькові*)

Київ 2021

# 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.

# Варіант: 20



package sample;

# Лістинг коду програми

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.shape.\*; import javafx.stage.Stage; 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("D:/maokg/lab3/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(); Group a = new Group();

Scene scene = new Scene (a, 800, 800); 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

{

пікселів

}

else

{

}

}

map[j\*2][height-1-i] = '1';

numberOfPixels++; // збільшуємо кількість чорних

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")

{

}

else

{

}

}

}

}

map[j\*2+1][height-1-i] = '1'; numberOfPixels++;

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=numberOfPixels-1; l > 0; l--)

{

path2.getElements().addAll(

new MoveTo(black[l-1][0],black[l-1][1]),

new LineTo(black[l][0],black[l][1])

);

}

Circle body = new Circle();

body.setCenterX(400); body.setCenterY(400); body.setRadius(100.0f); body.setFill(Color.BLACK); root.getChildren().add(body);

Circle eye1 = new Circle(); eye1.setCenterX(-10); eye1.setCenterY(-30); eye1.setRadius(15); eye1.setFill(Color.WHITE);

eye1.setTranslateX(body.getCenterX()); eye1.setTranslateY(body.getCenterY()); root.getChildren().add(eye1);

Circle pupil1 = new Circle(); pupil1.setCenterX(eye1.getCenterX()+2); pupil1.setCenterY(eye1.getCenterY()); pupil1.setRadius(8); pupil1.setFill(Color.BLACK); pupil1.setTranslateX(eye1.getTranslateX()); pupil1.setTranslateY(eye1.getTranslateY()); root.getChildren().add(pupil1);

Circle eye2 = new Circle(); eye2.setCenterX(50); eye2.setCenterY(-30); eye2.setRadius(15); eye2.setFill(Color.WHITE);

eye2.setTranslateX(body.getCenterX()); eye2.setTranslateY(body.getCenterY()); root.getChildren().add(eye2);

Circle pupil2 = new Circle(); pupil2.setCenterX(eye2.getCenterX()+2); pupil2.setCenterY(eye2.getCenterY()); pupil2.setRadius(8); pupil2.setFill(Color.BLACK); pupil2.setTranslateX(eye2.getTranslateX()); pupil2.setTranslateY(eye2.getTranslateY()); root.getChildren().add(pupil2);

Ellipse ellipse = new Ellipse(); ellipse.setCenterX(body.getCenterX()+25); ellipse.setCenterY(body.getCenterY()+12); ellipse.setRadiusX(20); ellipse.setRadiusY(10); ellipse.setFill(Color.WHITE); root.getChildren().add(ellipse);

Arc arc = new Arc(); arc.setCenterX(body.getCenterX()+25); arc.setCenterY(body.getCenterY()+10); arc.setRadiusX(20); arc.setRadiusY(20); arc.setStartAngle(200); arc.setLength(180); arc.setFill(Color.rgb(250, 172, 3)); arc.setType(ArcType.ROUND); arc.setStroke(Color.BLACK); arc.setStrokeWidth(2); root.getChildren().add(arc);

MoveTo moveTo = new MoveTo(body.getCenterX()+6, body.getCenterY()+8);

QuadCurveTo quadCurveTo = new QuadCurveTo(); quadCurveTo.setX(arc.getCenterX()+arc.getRadiusX()+25); quadCurveTo.setY(arc.getCenterY()+10); quadCurveTo.setControlX(moveTo.getX()+40); quadCurveTo.setControlY(moveTo.getY()-10);

QuadCurveTo quadCurveTo2 = new QuadCurveTo(); quadCurveTo2.setX(moveTo.getX()); quadCurveTo2.setY(moveTo.getY()); quadCurveTo2.setControlX(moveTo.getX()+25); quadCurveTo2.setControlY(moveTo.getY()-30);

Path path = new Path();

path.getElements().addAll(moveTo, quadCurveTo, quadCurveTo2); path.setFill(Color.rgb(250, 172, 3)); path.setStroke(Color.BLACK);

path.setStrokeWidth(2); root.getChildren().add(path);

Polygon eyebrow = new Polygon();

double startX = body.getCenterX()+eye1.getCenterX()+20;

double startY = body.getCenterY()+eye1.getCenterY()-eye1.getRadius()+10; eyebrow.getPoints().addAll(new Double[] {

startX, startY, startX+5, startY-5, startX-50, startY-20, startX-55, startY-10

});

eyebrow.setFill(Color.rgb(196, 71, 2)); root.getChildren().add(eyebrow);

Polygon eyebrow2 = new Polygon();

startX = body.getCenterX()+eye2.getCenterX()-20;

startY = body.getCenterY()+eye2.getCenterY()-eye2.getRadius()+10; eyebrow2.getPoints().addAll(new Double[] {

startX, startY, startX-5, startY-5, startX+50, startY-20, startX+55, startY-10

});

eyebrow2.setFill(Color.rgb(196, 71, 2)); root.getChildren().add(eyebrow2);

MoveTo mt = new MoveTo(body.getCenterX(), body.getCenterY()-body.getRadius()); QuadCurveTo qCT1 = new QuadCurveTo();

qCT1.setX(mt.getX()-40);

qCT1.setY(mt.getY()-20); qCT1.setControlX(mt.getX()); qCT1.setControlY(mt.getY()-10);

ArcTo arcTo = new ArcTo(); arcTo.setX(qCT1.getX()-5); arcTo.setY(qCT1.getY()+10); arcTo.setRadiusX(10); arcTo.setRadiusY(10);

QuadCurveTo qCT2 = new QuadCurveTo(); qCT2.setX(mt.getX()-10); qCT2.setY(mt.getY()+10); qCT2.setControlX(mt.getX()); qCT2.setControlY(mt.getY());

Path tail = new Path(); tail.getElements().addAll(mt, qCT1, arcTo, qCT2); tail.setFill(Color.BLACK); root.getChildren().add(tail);

MoveTo mT2 = new MoveTo(qCT1.getX()+1, qCT1.getY()+2); ArcTo aT2 = new ArcTo();

aT2.setX(mT2.getX()-3);

aT2.setY(mT2.getY()+7); aT2.setRadiusX(10); aT2.setRadiusY(10);

QuadCurveTo qCT3 = new QuadCurveTo();

qCT3.setX(aT2.getX()+11); qCT3.setY(aT2.getY()+3); qCT3.setControlX(aT2.getX()); qCT3.setControlY(aT2.getY());

LineTo lT1 = new LineTo(); lT1.setX(qCT3.getX()-3);

lT1.setY(qCT3.getY()-5);

LineTo lT2 = new LineTo(); lT2.setX(qCT3.getX()+6); lT2.setY(qCT3.getY()-6);

QuadCurveTo qCT4 = new QuadCurveTo(); qCT4.setX(mT2.getX());

qCT4.setY(mT2.getY()); qCT4.setControlX(mT2.getX()+1); qCT4.setControlY(mT2.getY()-1);

Path tail2 = new Path();

tail2.getElements().addAll(mT2, aT2, qCT3, lT1, lT2, qCT4); tail2.setFill(Color.rgb(196, 71, 2)); root.getChildren().add(tail2);

Circle spot = new Circle(); spot.setCenterX(20); spot.setCenterY(-67); spot.setRadius(10); spot.setFill(Color.WHITE);

spot.setTranslateX(body.getCenterX()); spot.setTranslateY(body.getCenterY()); root.getChildren().add(spot);

Arc arc1 = new Arc(); arc1.setCenterX(body.getCenterX()); arc1.setCenterY(body.getCenterY()); arc1.setRadiusX(98); arc1.setRadiusY(98); arc1.setStartAngle(245); arc1.setLength(65); arc1.setFill(Color.rgb(67, 67, 67)); root.getChildren().add(arc1);

QuadCurve curve = new QuadCurve();

curve.setStartX(body.getCenterX()+body.getRadius()\*Math.cos(Math.toRadians(arc1.getSta rtAngle()))+2);

curve.setStartY(body.getCenterY()-body.getRadius()\*Math.sin(Math.toRadians(arc1.getSta rtAngle())));

curve.setEndX(body.getCenterX()+body.getRadius()\*Math.cos(Math.toRadians(311))-2); curve.setEndY(body.getCenterY()-body.getRadius()\*Math.sin(Math.toRadians(311)));

curve.setControlX(curve.getStartX()+40); curve.setControlY(curve.getStartY()-75); curve.setFill(Color.rgb(67, 67, 67)); root.getChildren().add(curve);

a.getChildren().add(root); int cycleCount = 4;

int time = 1500;

PathTransition pathTransition = new PathTransition(); pathTransition.setDuration(Duration.millis(time)); pathTransition.setPath(path2); pathTransition.setNode(root); pathTransition.setCycleCount(cycleCount+1); pathTransition.setAutoReverse(true);

a);

ScaleTransition scaleTransition = new ScaleTransition(Duration.millis(time),

scaleTransition.setToX(2); scaleTransition.setToY(2); scaleTransition.setCycleCount(cycleCount); scaleTransition.setAutoReverse(true);

RotateTransition rotateTransition = new

RotateTransition(Duration.millis(time), a); rotateTransition.setByAngle(360f); rotateTransition.setCycleCount(cycleCount); rotateTransition.setAutoReverse(true);

ParallelTransition parallelTransition = new ParallelTransition(); parallelTransition.getChildren().addAll(

pathTransition, rotateTransition, scaleTransition

);

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);

}

}

# Результат

