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
Дикий Ілля
(*прізвище, ім’я, по батькові*)

Зарахована
“____” “_____” 20___ р.
викладачем
Шкурат Оксаною Сергіївною
(*прізвище, ім’я, по батькові*)

варіант № 3

Київ 2021

Завдання

Тема: Структура файлів формату .bmp. Анімація примітивів за допомогою засобів бібліотеки JavaFX

Мета:

1. вивчення структури та особливостей використання файлів формату .bmp;
2. вивчення стандартних засобів JavaFX для візуалізації зображення;
3. вивчення засобів анімації примітивів в JavaFX.

За допомогою примітивів JavaFX максимально реально зобразити персонажа за варіантом та виконати його 2D анімацію. Для анімації скористатися стандартними засобами бібліотеки JavaFX.

3	

Лістинг коду програми

```
import java.awt.*;
import java.awt.event.ActionEvent;package lab3;

import java.awt.*;
import java.awt.geom.QuadCurve2D;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import  java.io.FileInputStream;
import  java.io.FileOutputStream;

import  javafx.animation.FadeTransition;
import  javafx.animation.ParallelTransition;
import  javafx.animation.PathTransition;
import  javafx.animation.RotateTransition;
import  javafx.animation.ScaleTransition;
import  javafx.animation.TranslateTransition;
import  javafx.application.Application;
import  javafx.scene.Group;
import  javafx.scene.Scene;
import  javafx.scene.paint.Color;
import  javafx.scene.shape.Arc;
import  javafx.scene.shape.Circle;
import  javafx.scene.shape.Line;
import  javafx.scene.shape.QuadCurve;
import  javafx.scene.shape.LineTo;
import  javafx.scene.shape.MoveTo;
import  javafx.scene.shape.Path;
import  javafx.scene.shape.Rectangle;
import javafx.scene.transform.Rotate;
import  javafx.stage.Stage;
import  javafx.util.Duration;

public class PrintingImage extends Application{
    private HeaderBitmapImage image; // приватне поле, яке зберігає об'єкт
зінформацією 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("/home/stilpert/Education/MAOKG/lab3/sou
rces/trajectory1.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);
        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])
            );
        }

        Circle circle = new Circle (0, 0,90, Color.YELLOW);
        circle.setStroke(Color.BLACK);
        circle.setStrokeWidth(1);
```

```
root.getChildren().add(circle);

Group group1 = new Group();
QuadCurve ray1 = new QuadCurve(95, 0, 170, 25, 180, 0);
ray1.setStroke(Color.BLACK);
ray1.setFill(Color.YELLOW);
group1.getChildren().add(ray1);
QuadCurve ray2 = new QuadCurve(95, 0, 170, -25, 180, 0);
ray2.setStroke(Color.BLACK);
ray2.setFill(Color.YELLOW);
group1.getChildren().add(ray2);
QuadCurve ray3 = new QuadCurve(-95, 0, -170, 25, -180, 0);
ray3.setStroke(Color.BLACK);
ray3.setFill(Color.YELLOW);
group1.getChildren().add(ray3);
QuadCurve ray4 = new QuadCurve(-95, 0, -170, -25, -180, 0);
ray4.setStroke(Color.BLACK);
ray4.setFill(Color.YELLOW);
group1.getChildren().add(ray4);
QuadCurve ray5 = new QuadCurve(0, -95, -25, -170, 0, -180);
ray5.setStroke(Color.BLACK);
ray5.setFill(Color.YELLOW);
group1.getChildren().add(ray5);
QuadCurve ray6 = new QuadCurve(0, -95, 25, -170, 0, -180);
ray6.setStroke(Color.BLACK);
ray6.setFill(Color.YELLOW);
group1.getChildren().add(ray6);
QuadCurve ray7 = new QuadCurve(0, 95, -25, 170, 0, 180);
ray7.setStroke(Color.BLACK);
ray7.setFill(Color.YELLOW);
group1.getChildren().add(ray7);
QuadCurve ray8 = new QuadCurve(0, 95, 25, 170, 0, 180);
ray8.setStroke(Color.BLACK);
ray8.setFill(Color.YELLOW);
group1.getChildren().add(ray8);
root.getChildren().add(group1);

Group group2 = new Group();
QuadCurve ray11 = new QuadCurve(95, 0, 170, 25, 180, 0);
ray11.setStroke(Color.BLACK);
ray11.setFill(Color.YELLOW);
```

```
group2.getChildren().add(ray11);
QuadCurve ray21 = new QuadCurve(95, 0, 170, -25, 180, 0);
ray21.setStroke(Color.BLACK);
ray21.setFill(Color.YELLOW);
group2.getChildren().add(ray21);
QuadCurve ray31 = new QuadCurve(-95, 0, -170, 25, -180, 0);
ray31.setStroke(Color.BLACK);
ray31.setFill(Color.YELLOW);
group2.getChildren().add(ray31);
QuadCurve ray41 = new QuadCurve(-95, 0, -170, -25, -180, 0);
ray41.setStroke(Color.BLACK);
ray41.setFill(Color.YELLOW);
group2.getChildren().add(ray41);
QuadCurve ray51 = new QuadCurve(0, -95, -25, -170, 0, -180);
ray51.setStroke(Color.BLACK);
ray51.setFill(Color.YELLOW);
group2.getChildren().add(ray51);
QuadCurve ray61 = new QuadCurve(0, -95, 25, -170, 0, -180);
ray61.setStroke(Color.BLACK);
ray61.setFill(Color.YELLOW);
group2.getChildren().add(ray61);
QuadCurve ray71 = new QuadCurve(0, 95, -25, 170, 0, 180);
ray71.setStroke(Color.BLACK);
ray71.setFill(Color.YELLOW);
group2.getChildren().add(ray71);
QuadCurve ray81 = new QuadCurve(0, 95, 25, 170, 0, 180);
ray81.setStroke(Color.BLACK);
ray81.setFill(Color.YELLOW);
group2.getChildren().add(ray81);

Rotate rotate = new Rotate();
rotate.setPivotX(0);
rotate.setPivotY(0);
rotate.setAngle(30);
group2.getTransforms().add(rotate);
root.getChildren().add(group2);

Group group3 = new Group();
QuadCurve ray12 = new QuadCurve(95, 0, 170, 25, 180, 0);
ray12.setStroke(Color.BLACK);
ray12.setFill(Color.YELLOW);
```

```
group3.getChildren().add(ray12);
QuadCurve ray22 = new QuadCurve(95, 0, 170, -25, 180, 0);
ray22.setStroke(Color.BLACK);
ray22.setFill(Color.YELLOW);
group3.getChildren().add(ray22);
QuadCurve ray32 = new QuadCurve(-95, 0, -170, 25, -180, 0);
ray32.setStroke(Color.BLACK);
ray32.setFill(Color.YELLOW);
group3.getChildren().add(ray32);
QuadCurve ray42 = new QuadCurve(-95, 0, -170, -25, -180, 0);
ray42.setStroke(Color.BLACK);
ray42.setFill(Color.YELLOW);
group3.getChildren().add(ray42);
QuadCurve ray52 = new QuadCurve(0, -95, -25, -170, 0, -180);
ray52.setStroke(Color.BLACK);
ray52.setFill(Color.YELLOW);
group3.getChildren().add(ray52);
QuadCurve ray62 = new QuadCurve(0, -95, 25, -170, 0, -180);
ray62.setStroke(Color.BLACK);
ray62.setFill(Color.YELLOW);
group3.getChildren().add(ray62);
QuadCurve ray72 = new QuadCurve(0, 95, -25, 170, 0, 180);
ray72.setStroke(Color.BLACK);
ray72.setFill(Color.YELLOW);
group3.getChildren().add(ray72);
QuadCurve ray82 = new QuadCurve(0, 95, 25, 170, 0, 180);
ray82.setStroke(Color.BLACK);
ray82.setFill(Color.YELLOW);
group3.getChildren().add(ray82);

Rotate rotate2 = new Rotate();
rotate2.setPivotX(0);
rotate2.setPivotY(0);
rotate2.setAngle(60);
group3.getTransforms().add(rotate2);
root.getChildren().add(group3);

QuadCurve eye1 = new QuadCurve(-45, -15, -33, -37, -22,-22 );
eye1.setStroke(Color.BLACK);
eye1.setFill(Color.BLACK);
root.getChildren().add(eye1);
```

```
        QuadCurve eye2 = new QuadCurve(10, -25, 18, -42, 33,-23 );
        eye2.setStroke(Color.BLACK);
        eye2.setFill(Color.BLACK);
        root.getChildren().add(eye2);

        Circle nose = new Circle (-10, -15,10, Color.YELLOW);
        nose.setStroke(Color.BLACK);
        nose.setStrokeWidth(1);
        root.getChildren().add(nose);

        QuadCurve smile = new QuadCurve(-35, 10, 10, 70, 30, 0);
        smile.setStroke(Color.BLACK);
        smile.setFill(Color.PINK);
        root.getChildren().add(smile);

        RotateTransition rotForRoot = new
RotateTransition(Duration.millis(500), root);
        rotForRoot.setByAngle(20f);
        rotForRoot.setCycleCount(20);
        rotForRoot.setAutoReverse(true);
        rotForRoot.play();

        PathTransition pathTransition = new PathTransition();
        pathTransition.setDuration(Duration.millis(5000));
        pathTransition.setPath(path2);
        pathTransition.setNode(root);
        pathTransition.play();

        ScaleTransition scaleTransition = new
ScaleTransition(Duration.seconds(5.0), root);
        scaleTransition.setToX(0.3f);
        scaleTransition.setToY(0.3f);
        scaleTransition.setAutoReverse(true);
        scaleTransition.play();

    }
    // далі необхідно зробити рух об'єкту по заданій траєторії
    private String returnPixelColor (int color) // метод для
співставленнякольорів 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);
    }
}
import java.awt.event.ActionListener;
```

Результати