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”

на тему “Структура файлів формату .bmp. Анімація примітивів за допомогою засобів бібліотеки JavaFX”

Виконав
студент ІІІ курсу
групи КП-82
Новохацький Владислав
(*прізвище, ім’я, по батькові*)

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

варіант № 15

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:

Лістинг коду програми

```
package lab3;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import javafx.animation.FadeTransition;
import javafx.animation.ParallelTransition;
import javafx.animation.PathTransition;
import javafx.animation.RotateTransition;
```

```
import javafx.animation.ScaleTransition;
import javafx.animation.Transition;
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.LineTo;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Ellipse;
import javafx.scene.shape.*;
import javafx.scene.shape.Path;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;
import javafx.util.Duration;


public class House extends Application {

	Color grass_c = Color.rgb(108, 165, 64);
	Color brown_c = Color.rgb(146,95,51);

	HeaderBitmapImage image;

	public House(){}

	public House(HeaderBitmapImage image) {
		this.image = image;
	}

	public static void main (String[] args) {
		launch(args);
	}

	@Override
	public void start(Stage stage) throws Exception {
		Group rootroot = new Group();
		Group root = new Group();

		Scene scene = new Scene(rootroot,1000,562);
		stage.setResizable(false);
		stage.setScene(scene);

		ReadingImageFromFile.loadBitmapImage("./bmp/trajectory.bmp");
		this.image = ReadingImageFromFile.pr.image;
```

```
        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];
        int numberOfPixels = 0;

        BufferedInputStream reader = new BufferedInputStream (new
FileInputStream("pixels.txt"));

        for(int i=0;i<height;i++)
        {
            for(int j=0;j<half;j++)
            {
                Circle cir;
                let = reader.read();
                let1=let;
                let2=let;
                let1=let1&(0xf0);
                let1=let1>>4;
                let2=let2&(0x0f);
                if(j*2<width)
                {
                    cir = new Circle ((j)*2,(height-1-i),1,
Color.valueOf((returnPixelColor(let1))));
                    rootroot.getChildren().add(cir);
                    if (returnPixelColor(let1) == "BLACK")
                    {
                        map[j*2][height-1-i] = '1';
                        numberOfPixels++;
                    }
                    else
                    {
                        map[j*2][height-1-i] = '0';
                    }
                }
                if(j*2+1<width)
                {
                    cir = new Circle
((j)*2+1,(height-1-i),1,Color.valueOf((returnPixelColor(let2))));
                    rootroot.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';
                }
            }
        }
    }

    reader.close();

    int[][] black;
    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);

     Ellipse grass = new Ellipse(105,132,90,35);
     grass.setFill(grass_c);
     root.getChildren().add(grass);

     Polygon tree1 = new Polygon(19.0,102.0,
                 25.0,103.0,
                 32.0,129.0,
                 25.0,128.0);
     tree1.setFill(Color.rgb(159, 117, 46));
     root.getChildren().add(tree1);
```

```
            List<Circle> treesc = Arrays.asList(new Circle(15,90,10), new
Circle(23,78,7), new Circle(32,90,8),new Circle (30,92,12), new Circle(19,99,11));
            for (Circle item : treesc) {
                  item.setFill(grass_c);
                  root.getChildren().add(item);
            }

            Polygon house1 = new Polygon (53,120,
                        46,68,
                        113,81,
                        110,139);
            house1.setFill(Color.rgb(242,227,184));
            root.getChildren().add(house1);
            Polygon house2 = new Polygon (113,81,
                        110,139,
                        155,112,
                        160,65);
            house2.setFill(Color.rgb(246,245,223));
            root.getChildren().add(house2);

            Polygon roof1 = new Polygon(30,63,
                        65,13,
                        101,23,
                        113,84);
            QuadCurve roof1_c = new QuadCurve(29,62,70,90,114,83);
            roof1_c.setFill(Color.rgb(188, 137, 189));
            roof1.setFill(Color.rgb(188, 137, 189));
            Polygon roof2 = new Polygon(101,23,
                        113,84,
                        176,54,
                        133,7);
            QuadCurve roof2_c = new QuadCurve(113,83,140,80,177,53);
            roof2_c.setFill(Color.rgb(145,80,160));
            roof2.setFill(Color.rgb(145,80,160));
            Polygon roof3 = new Polygon(65,13,
                        101,-2,
                        133,7,
                        101,23);
            roof3.setFill(Color.rgb(212, 177, 212));

            root.getChildren().add(roof1);
            root.getChildren().add(roof1_c);
            root.getChildren().add(roof2);
            root.getChildren().add(roof2_c);
            root.getChildren().add(roof3);

            Polygon window1 = new Polygon(68,91,
```

```
                97,96,
                96,115,
                70,110);
        window1.setFill(Color.rgb(197,224,239));
        window1.setStroke(brown_c);
        window1.setStrokeType(StrokeType.OUTSIDE);
        window1.setStrokeWidth(4);
        root.getChildren().add(window1);

        List<Line> windows= Arrays.asList(new Line(82,92,82,112), new
Line(68,101,97,105));
        for (Line item : windows) {
               item.setStrokeWidth(4);
               item.setStroke(brown_c);
               root.getChildren().add(item);
        }

        Ellipse door1 = new Ellipse (137,108,9,20);
        door1.setFill(Color.rgb(215,161,93));
        door1.setStroke(brown_c);
        door1.setStrokeWidth(4);
        door1.setRotate(4);
        root.getChildren().add(door1);

        Circle door2 = new Circle(141,106,2);
        door2.setFill(Color.rgb(252, 211, 8));
        root.getChildren().add(door2);

        Polygon grass2 = new Polygon(110,139,
                155,112,
                169,110,
                132,158);
        grass2.setFill(grass_c);
        root.getChildren().add(grass2);

        Group road = new Group();

        QuadCurve road1 = new QuadCurve(143,119,150,166,92,148);
        road1.setFill(Color.YELLOW);
        QuadCurve road2 = new QuadCurve(138,121,134,141,98,144);
        road2.setFill(grass_c);
        QuadCurve road3 = new QuadCurve(59,163,101,132,138,147);
        road3.setFill(Color.YELLOW);
        QuadCurve road4 = new QuadCurve(59,161,80,170,100,150);
        road4.setFill(Color.YELLOW);

        road.getChildren().add(road1);
```

```
            road.getChildren().add(road2);
            road.getChildren().add(road3);
            road.getChildren().add(road4);

            root.getChildren().add(road);

            Polygon pipe1 = new Polygon(122,38,
                        122,22,
                        134,22,
                        134,47);
            pipe1.setFill(Color.rgb(249,180,22));
            Polygon pipe2 = new Polygon(134,47,
                        134,22,
                        144,17,
                        144,40);
            pipe2.setFill(Color.rgb(249,228,12));
            Polygon pipe3 = new Polygon(120,23,
                        120,16,
                        135,17,
                        135,25);
            pipe3.setFill(Color.rgb(148, 184, 56));
            Polygon pipe4 = new Polygon(135,17,
                        135,25,
                        145,18,
                        145,11);
            pipe4.setFill(Color.rgb(218,228,72));
            Polygon pipe5 = new Polygon(120,16,
                        135,17,
                        145,11,
                        130,10);
            pipe5.setFill(Color.rgb(117, 158, 58));

            root.getChildren().addAll(Arrays.asList(pipe1,pipe2,pipe3,pipe4,pipe5));

            rootroot.getChildren().add(root);

            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])
            );
        }

        PathTransition pathTransition = new PathTransition(Duration.millis(8000),
path2, root);
```

```
        pathTransition.setCycleCount(Transition.INDEFINITE);
        pathTransition.setAutoReverse(true);

        ScaleTransition scaleTransition = new ScaleTransition(Duration.millis(4000),
root);
        scaleTransition.setByX(0.3);
        scaleTransition.setByY(0.3);
        scaleTransition.setCycleCount(Transition.INDEFINITE);
        scaleTransition.setAutoReverse(true);

        RotateTransition rotateTransition = new
RotateTransition(Duration.millis(8000), root);
        rotateTransition.setByAngle(360f);
        rotateTransition.setCycleCount(Transition.INDEFINITE);

        ParallelTransition parallelTransition = new ParallelTransition();
        parallelTransition.getChildren().addAll(pathTransition, scaleTransition,
rotateTransition);

        parallelTransition.play();

        stage.show();
       }

       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