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”

Виконала
студентка III курсу
групи КП-82
Морщак Каріна Юріївна
(*прізвище, ім’я, по батькові*)

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

варіант № 14

Київ 2021

Варіант завдання

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

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.

Завдання за варіантом:

Лістинг коду програми

Umbrella.java

```
package lab3;

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 java.awt.geom.RoundRectangle2D;
import javafx.util.Duration;



public class Umbrella extends Application {
      @Override
    public void start(Stage primaryStage){
        Group root = new Group();
        Scene scene = new Scene(root, 1000, 500);
        //внутрішня частина парасольки
        Ellipse ellipse1 = new Ellipse(228, 167, 80.5, 80.5);
        ellipse1.setFill(Color.rgb(192,98,133));
        root.getChildren().add(ellipse1);

        Ellipse ellipse2 = new Ellipse(177, 187, 44, 35);
        ellipse2.setFill(Color.WHITE);
        root.getChildren().add(ellipse2);

        Ellipse ellipse3 = new Ellipse(231, 178, 55, 35);
        ellipse3.setFill(Color.WHITE);
        root.getChildren().add(ellipse3);

        Ellipse ellipse4 = new Ellipse(277, 170, 38, 30);
        ellipse4.setFill(Color.WHITE);
        root.getChildren().add(ellipse4);

        Ellipse ellipse5 = new Ellipse(231, 218, 85, 70);
        ellipse5.setFill(Color.WHITE);
        root.getChildren().add(ellipse5);

        //ручка парасольки
```

```
        MoveTo mt3 = new MoveTo(220, 120);
        LineTo lt2 = new LineTo(235, 220);
        QuadCurveTo qt8 = new QuadCurveTo(240, 226, 246, 220);
        LineTo lt3 = new LineTo(245, 212);
        Path handle = new Path();
        handle.setStrokeWidth(2);
        handle.setStroke(Color.rgb(213, 214, 204));
        handle.getElements().addAll(mt3, lt2, qt8, lt3);
        root.getChildren().add(handle);

        //зовнішня частина парасольки
        MoveTo mt1 = new MoveTo(146, 161.5);
        QuadCurveTo qt1 = new QuadCurveTo(160, 100, 211, 85);
        QuadCurveTo qt2 = new QuadCurveTo(277, 119, 250, 135);
        QuadCurveTo qt3 = new QuadCurveTo(217, 117, 190, 143);
        QuadCurveTo qt4 = new QuadCurveTo(170, 130, 146, 163);
        QuadCurveTo qt5 = new QuadCurveTo(280, 75, 307, 150);
        Path top = new Path();
        top.setStrokeWidth(2);
        top.setStroke(Color.rgb(213, 214, 204));
        top.setFill(Color.rgb(121, 30, 62));
        top.getElements().addAll(mt1, qt1, qt5, qt2, qt3, qt4);
        root.getChildren().add(top);

        //верхній "хвостик" парасольки
        MoveTo mt2 = new MoveTo(215, 86);
        LineTo lt1 = new LineTo(215, 70);
        Path pin = new Path();
        pin.setStrokeWidth(2);
        pin.setStroke(Color.rgb(213, 214, 204));
        pin.getElements().addAll(mt2, lt1);
        root.getChildren().add(pin);

        //дуги на зовнішній частині парасольки
        QuadCurveTo qt6 = new QuadCurveTo(185, 110, 190, 142);
        QuadCurveTo qt7 = new QuadCurveTo(250, 100, 250, 134);
        Path lines = new Path();
        lines.setStrokeWidth(2);
        lines.setStroke(Color.rgb(213, 214, 204));
        lines.getElements().addAll(mt2, qt6, mt2, qt7);
        root.getChildren().add(lines);
```

```
        // Анімація
        int cycleCount = 2;
        int time = 3000;

        TranslateTransition translateTransition = new
TranslateTransition(Duration.millis(time), root);
        translateTransition.setFromY(50);
        translateTransition.setToY(200);
        translateTransition.setFromX(100);
        translateTransition.setToX(400);
        translateTransition.setCycleCount(cycleCount + 1);
        translateTransition.setAutoReverse(true);

        ScaleTransition scaleTransition = new
ScaleTransition(Duration.millis(time), root);
        scaleTransition.setToX(3);
        scaleTransition.setToY(3);
        scaleTransition.setAutoReverse(true);

        RotateTransition rotateTransition = new
RotateTransition(Duration.millis(time), root);
        rotateTransition.setByAngle(180f);
        rotateTransition.setCycleCount(cycleCount);
        rotateTransition.setAutoReverse(true);

        ScaleTransition scaleTransition2 = new
ScaleTransition(Duration.millis(time), root);
        scaleTransition2.setToX(0.3);
        scaleTransition2.setToY(0.3);
        scaleTransition2.setCycleCount(cycleCount);
        scaleTransition2.setAutoReverse(true);

        ParallelTransition parallelTransition = new ParallelTransition();
        parallelTransition.getChildren().addAll(
                  translateTransition,
                  scaleTransition,
                  rotateTransition,
                scaleTransition2
        );
        parallelTransition.setCycleCount(Timeline.INDEFINITE);
        parallelTransition.play();
```

```
        primaryStage.setResizable(false);
        primaryStage.setTitle("lab3");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

Transitions.java

```
package lab3;

import javafx.animation.FadeTransition;
import javafx.animation.Interpolator;
import javafx.animation.KeyFrame;
import javafx.animation.KeyValue;
import javafx.animation.ParallelTransition;
import javafx.animation.PathTransition;
import javafx.animation.RotateTransition;
import javafx.animation.ScaleTransition;
import javafx.animation.SequentialTransition;
import javafx.animation.Timeline;
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.CubicCurveTo;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;
import javafx.util.Duration;

public class Transitions extends Application {

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

```
      }

      public void start(Stage primaryStage) throws Exception {
            Group root = new Group();
            Scene scene = new Scene (root, 500, 500);

            //Створення прямокутника червоного кольору
            final Rectangle rect1 = new Rectangle(10, 10, 100, 100);
            rect1.setArcHeight(20);
            rect1.setArcWidth(20);
            rect1.setFill(Color.RED);
            root.getChildren().add(rect1);

            // створення ефекту зникнення
            FadeTransition ft = new FadeTransition(Duration.millis(3000),
rect1);
            ft.setFromValue(1.0); // встановлення початкового значення
прозорості об'єкту
            ft.setToValue(0.1); // встановлення кінцевого значення
прозорості об'єкту
            ft.setCycleCount(Timeline.INDEFINITE);
            ft.setAutoReverse(true);
            ft.play();

            // створення синього прямокутника з круглими кутами
            final Rectangle rectPath = new Rectangle (0, 0, 40, 40);
            rectPath.setArcHeight(10);
            rectPath.setArcWidth(10);
            rectPath.setFill(Color.BLUE);
            root.getChildren().add(rectPath);

            // створення траєкторії з 2 ліній типу CubicCurveTo
            Path path = new Path();
            path.getElements().add(new MoveTo(20,20)); // вказання
початкової позиції, з якої починається траєкторія
            path.getElements().add(new CubicCurveTo(380, 0, 380, 120, 200,
120)); // перша крива
            path.getElements().add(new CubicCurveTo(0, 120, 0, 240, 380,
240));   // друга крива

            // створення анімації руху по траєкторії
            PathTransition pathTransition = new PathTransition();
```

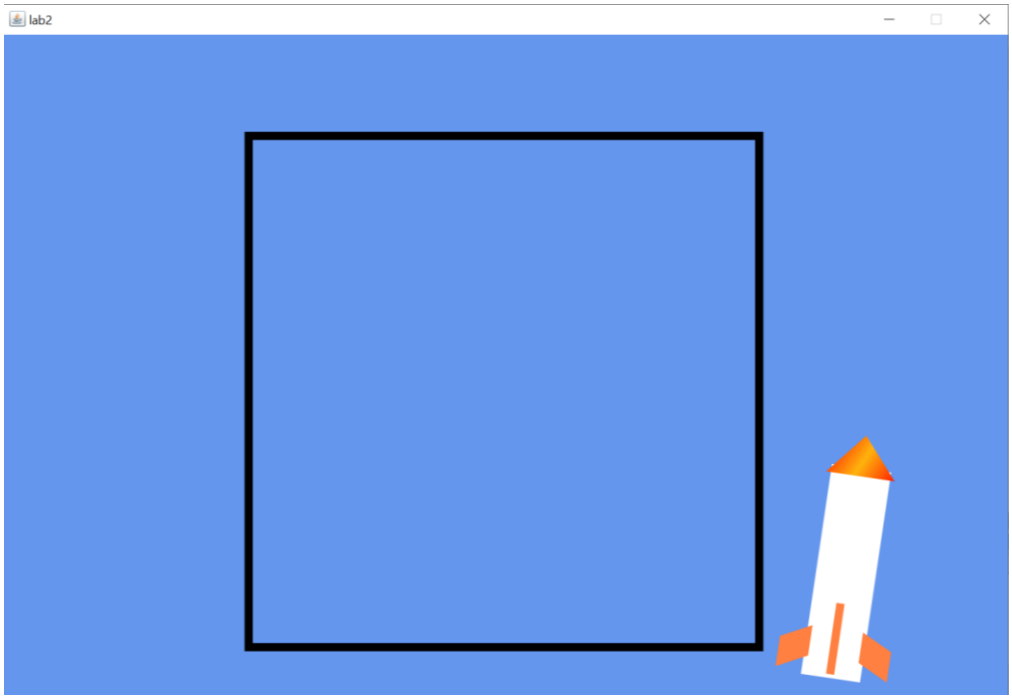
```
            pathTransition.setDuration(Duration.millis(4000)); //
встановлення часу анмації
            pathTransition.setPath(path); // прив'язування траекторії
            pathTransition.setNode(rectPath); // вибір об'єкта, який буде
анімуватися

      pathTransition.setOrientation(PathTransition.OrientationType.ORTHOGONA
L_TO_TANGENT); // вказання орієнтації об'єкта при русі
            pathTransition.setCycleCount(Timeline.INDEFINITE); // циклічна
анімація
            pathTransition.setAutoReverse(true); // можливість руху назад
            pathTransition.play(); // відтворення анімації

            //Створення прямокутника
            Rectangle rectParallel = new Rectangle(10,200,50, 50);
            rectParallel.setArcHeight(15);
            rectParallel.setArcWidth(15);
            rectParallel.setFill(Color.DARKBLUE);
            rectParallel.setTranslateX(50);
            rectParallel.setTranslateY(75);
            root.getChildren().add(rectParallel);

            // створення ефекту зникнення
            FadeTransition fadeTransition =
                        new FadeTransition(Duration.millis(3000),
rectParallel);
            fadeTransition.setFromValue(1.0f);
            fadeTransition.setToValue(0.3f);
            fadeTransition.setCycleCount(2);
            fadeTransition.setAutoReverse(true);

            // Створення ефекту переміщення
            TranslateTransition translateTransition =
                        new TranslateTransition(Duration.millis(2000),
rectParallel);
            translateTransition.setFromX(50);
            translateTransition.setToX(350);
            translateTransition.setCycleCount(2);
            translateTransition.setAutoReverse(true);

            // Створення повороту об'єкту
            RotateTransition rotateTransition =
```

```
                        new RotateTransition(Duration.millis(3000),
rectParallel);
            rotateTransition.setByAngle(180f);
            rotateTransition.setCycleCount(4);
            rotateTransition.setAutoReverse(true);

            // Масштабування об'єкту
            ScaleTransition scaleTransition =
                        new ScaleTransition(Duration.millis(2000),
rectParallel);
            scaleTransition.setToX(2f);
            scaleTransition.setToY(2f);
            scaleTransition.setCycleCount(2);
            scaleTransition.setAutoReverse(true);

            // Створення можливості паралельно виконувати анімацію
            ParallelTransition parallelTransition =
                        new ParallelTransition();
            parallelTransition.getChildren().addAll(
                        fadeTransition,
                        translateTransition,
                        rotateTransition,
                        scaleTransition
            );
            parallelTransition.setCycleCount(Timeline.INDEFINITE);
            parallelTransition.play();

            Rectangle rectSeq = new Rectangle(25,25,50,50);
            rectSeq.setArcHeight(15);
            rectSeq.setArcWidth(15);
            rectSeq.setFill(Color.CRIMSON);
            rectSeq.setTranslateX(50);
            rectSeq.setTranslateY(50);
            root.getChildren().add(rectSeq);

            fadeTransition =
                        new FadeTransition(Duration.millis(1000), rectSeq);
            fadeTransition.setFromValue(1.0f);
            fadeTransition.setToValue(0.3f);
            fadeTransition.setCycleCount(1);
            fadeTransition.setAutoReverse(true);
```

```
            translateTransition =
                        new TranslateTransition(Duration.millis(2000),
rectSeq);
            translateTransition.setFromX(50);
            translateTransition.setFromY(40);
            translateTransition.setToX(375);
            translateTransition.setToY(375);
            translateTransition.setCycleCount(1);
            translateTransition.setAutoReverse(true);

            rotateTransition =
                        new RotateTransition(Duration.millis(2000),
rectSeq);
            rotateTransition.setByAngle(180f);
            rotateTransition.setCycleCount(4);
            rotateTransition.setAutoReverse(true);

            SequentialTransition sequentialTransition = new
SequentialTransition();
            sequentialTransition.getChildren().addAll(
                        fadeTransition,
                        translateTransition
            );

            sequentialTransition.setCycleCount(Timeline.INDEFINITE);
            sequentialTransition.setAutoReverse(true);
            sequentialTransition.play();

            final Rectangle rectBasicTimeline = new Rectangle(100, 60, 150,
50);
            rectBasicTimeline.setFill(Color.RED);
            root.getChildren().add(rectBasicTimeline);

            final Timeline timeline = new Timeline();
            timeline.setCycleCount(Timeline.INDEFINITE);
            timeline.setAutoReverse(true);
            final KeyValue kv = new KeyValue(rectBasicTimeline.xProperty(),
300);
            final KeyFrame kf = new KeyFrame(Duration.millis(1000), kv);
            timeline.getKeyFrames().add(kf);
            timeline.play();
```

```
            final Rectangle rectBasicTimeline1 = new Rectangle(200, 100,
200, 100);
            rectBasicTimeline1.setFill(Color.BROWN);
            root.getChildren().add(rectBasicTimeline1);
            final Timeline timeline1 = new Timeline();
            timeline1.setCycleCount(Timeline.INDEFINITE);
            timeline1.setAutoReverse(true);
            final KeyValue kv1 = new
KeyValue(rectBasicTimeline1.xProperty(), 400,
                        Interpolator.EASE_BOTH);
            final KeyFrame kf1 = new KeyFrame(Duration.millis(700), kv1);
            timeline1.getKeyFrames().add(kf1);
            timeline1.play();

            primaryStage.setScene(scene);
            primaryStage.show();
      }
}
```

Результат


lab2