

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-82  Мельничук Олексій Геннадійович  (*прізвище, ім’я, по батькові*)  варіант № 12 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**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.

**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

Студентам пропонується скористатися розглянутими класами для читання, обробки та збереження зображень формату .bmp з метою використання рисунку для створення траєкторії руху або меж, в яких дозволений рух об’єктів. В даному випадку рекомендується використовувати кольори великої контрастності для різних призначень (наприклад, чорний колір відповідатиме за траєкторію руху, а інші кольори – заборонятимуть рух).

**Варіант: 12**

**Код програми**

|  |
| --- |
| **Penguin.java** |
| package lab3;  import java.io.IOException;  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.scene.transform.Rotate;  import javafx.stage.Stage;  import javafx.util.Duration;  public class Penguin extends Application {  public static void main (String args[]) {  launch(args);  }  double WIDTH = 640;  double HEIGHT = 480;    private static Color beakColor = Color.rgb(247,189,33);  private static Color bodyColor = Color.rgb(23,151,229);  private static Color bodyAccentColor = Color.rgb(207,228,249);  private static Color wingColor = Color.rgb(4,75,178);  private static Color eyeColor = Color.WHITE;  private static Color irisColor = Color.BLACK;    @Override  public void start(Stage primaryStage) throws IOException {  Group root = new Group();  // Scene (root, x, y)  Scene scene = new Scene (root, WIDTH, HEIGHT);  double centerx = WIDTH / 2;  double centery = HEIGHT / 2;    //head  Ellipse head = new Ellipse(centerx, centery - 27 - 49, 85, 99);  head.setFill(bodyColor);  root.getChildren().add(head);      Ellipse body = new Ellipse(centerx, centery, 103, 103);  body.setFill(bodyColor);  root.getChildren().add(body);    Ellipse body2 = new Ellipse(centerx, centery + 10, 73, 93);  body2.setFill(bodyAccentColor);  root.getChildren().add(body2);      // beak  Polygon beak = new Polygon(centerx, centery - 34,  centerx - 46, centery - 82,  centerx + 46, centery - 82);  beak.setFill(beakColor);  root.getChildren().add(beak);    //eyes    Ellipse white1 = new Ellipse(centerx - 24, centery - 102, 25, 37);  white1.setFill(eyeColor);  root.getChildren().add(white1);    Ellipse white2 = new Ellipse(centerx + 32, centery - 100, 31, 25);  white2.setFill(eyeColor);  root.getChildren().add(white2);    Ellipse iris1 = new Ellipse(centerx - 10, centery - 95, 8, 10);  iris1.setFill(irisColor);  root.getChildren().add(iris1);    Ellipse iris2 = new Ellipse(centerx + 10, centery - 95, 8, 10);  iris2.setFill(irisColor);  root.getChildren().add(iris2);    // wings    //rotating the 2nd rectangle.    Ellipse wing1 = new Ellipse(centerx - 112, centery - 20, 17, 50);  wing1.setFill(wingColor);  wing1.getTransforms().add(new Rotate(45, centerx - 112, centery - 20));  root.getChildren().add(wing1);    Ellipse wing2 = new Ellipse(centerx + 112, centery - 20, 17, 50);  wing2.setFill(wingColor);  wing2.getTransforms().add(new Rotate(-45, centerx + 112, centery - 20));  root.getChildren().add(wing2);        //legs    Rectangle leg12 = new Rectangle(centerx + 8, centery + 82, 96, 37);  leg12.setFill(beakColor);  leg12.setArcWidth(30);  leg12.setArcHeight(20);  root.getChildren().add(leg12);    Ellipse leg11 = new Ellipse(centerx + 56, centery + 92, 48, 27);  leg11.setFill(beakColor);  root.getChildren().add(leg11);  Rectangle leg22 = new Rectangle(centerx + 8 - 56 - 56, centery + 82, 96, 37);  leg22.setFill(beakColor);  leg22.setArcWidth(30);  leg22.setArcHeight(20);  root.getChildren().add(leg22);    Ellipse leg21 = new Ellipse(centerx - 56, centery + 92, 48, 27);  leg21.setFill(beakColor);  root.getChildren().add(leg21);        // Animation    int cycleCount = 1;  int time = 5000;    Path penguinPath = new GetPathFromBMP().getPath("D:/eclipse\_labs/ffs/sources/trajectory2.bmp");  PathTransition pengTransition = new PathTransition();  pengTransition.setDuration(Duration.millis(time));  pengTransition.setPath(penguinPath);  pengTransition.setNode(root);  pengTransition.setCycleCount(cycleCount);  pengTransition.setAutoReverse(true);    TranslateTransition translateTransition1 = new TranslateTransition(Duration.millis(1000), root);  translateTransition1.setToX(0);  translateTransition1.setToY(0);  translateTransition1.setCycleCount(cycleCount);  translateTransition1.setAutoReverse(true);        ScaleTransition scaleTransition = new ScaleTransition(Duration.millis(1000), root);  scaleTransition.setToX(0.25);  scaleTransition.setToY(0.25);  scaleTransition.setAutoReverse(true);    RotateTransition rotateTransition = new RotateTransition(Duration.millis(1), root);  rotateTransition.setByAngle(360f);  rotateTransition.setCycleCount(cycleCount \* time);  rotateTransition.setAutoReverse(false);  SequentialTransition parallelTransition = new SequentialTransition();  parallelTransition.getChildren().addAll(  scaleTransition,  pengTransition,  translateTransition1,  rotateTransition    );  parallelTransition.setCycleCount(Timeline.INDEFINITE);  parallelTransition.play();  // End of animation  primaryStage.setTitle("penguin");  primaryStage.setScene(scene);  primaryStage.show();  }  } |

# 

|  |
| --- |
| **GetPathFromBMP.java** |
| package lab3;  import java.io.BufferedInputStream;  import java.io.BufferedOutputStream;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  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.LineTo;  import javafx.scene.shape.MoveTo;  import javafx.scene.shape.Path;  import javafx.scene.shape.Rectangle;  import javafx.stage.Stage;  import javafx.util.Duration;  import javafx.scene.shape.Path;  public class GetPathFromBMP {  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 GetPathFromBMP(HeaderBitmapImage image) // перевизначений стандартний конструктор  {  this.image = image;  }    public GetPathFromBMP() {  // TODO Auto-generated constructor stub  }  public Path getPath(String source) throws IOException  {  ReadingImageFromFile.loadBitmapImage(source);  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.BLACK);  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';  }  }  }  }    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])  );  }    return path2;  }    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;  }  } |

# **Результати роботи програми**

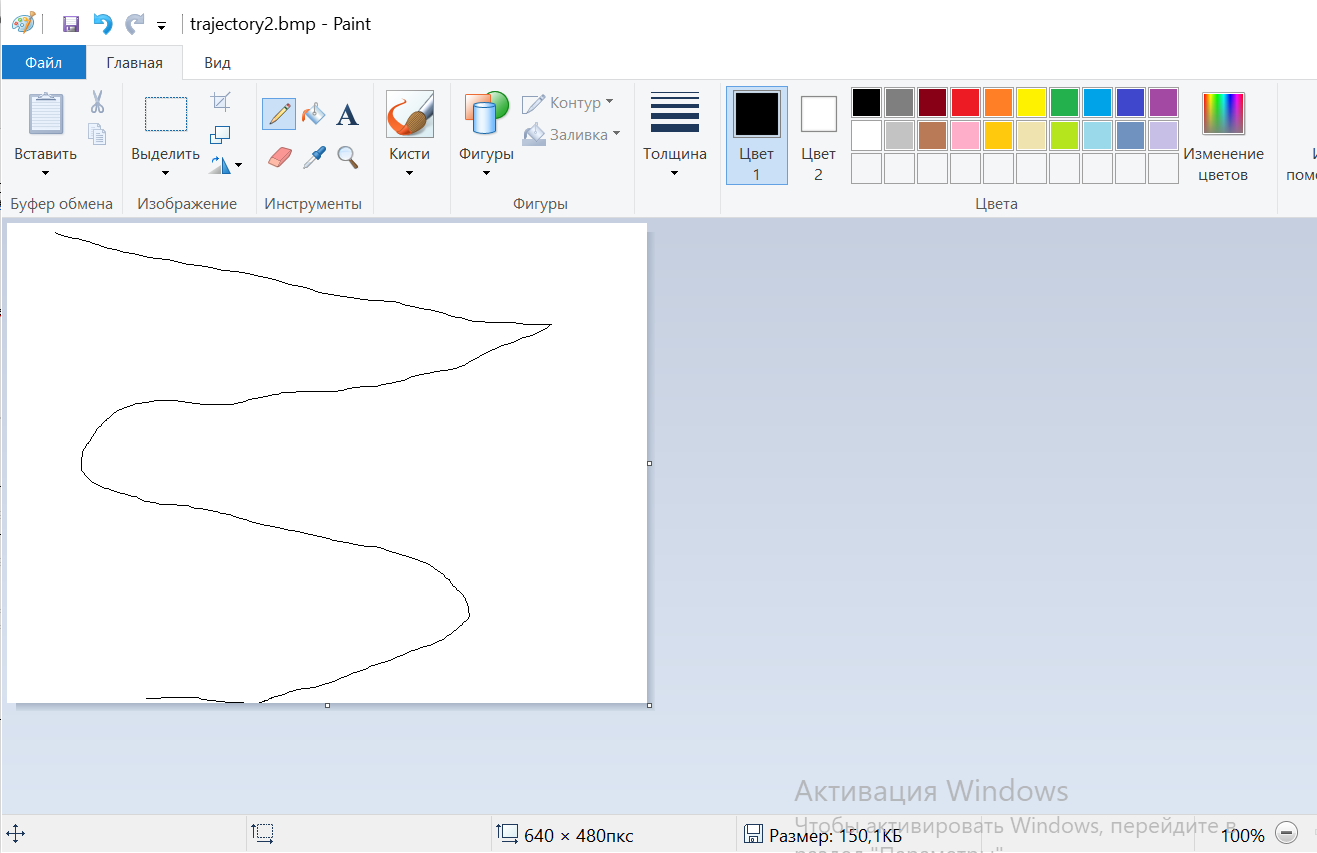
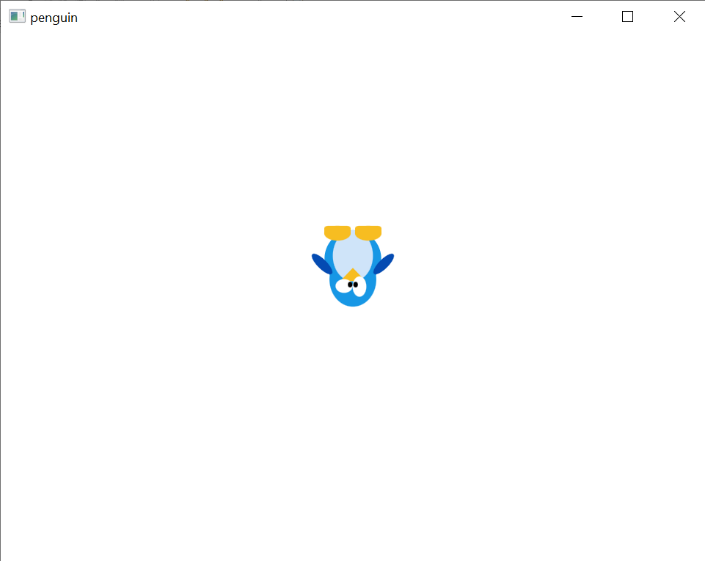
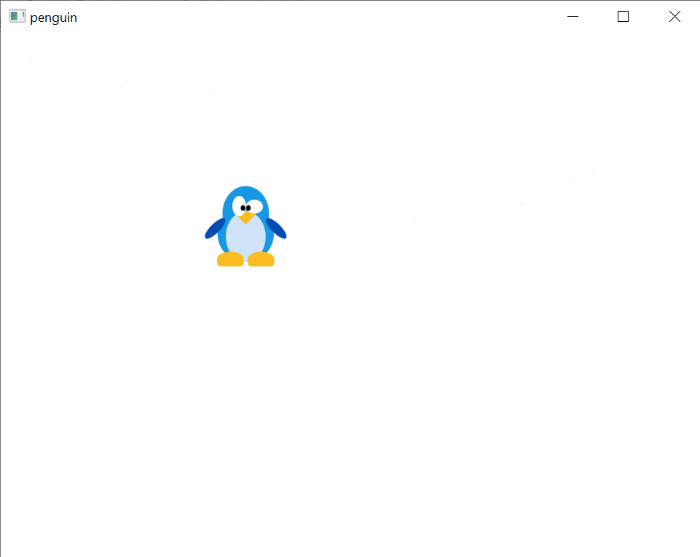


Рис.1. Шлях в форматі .bpm



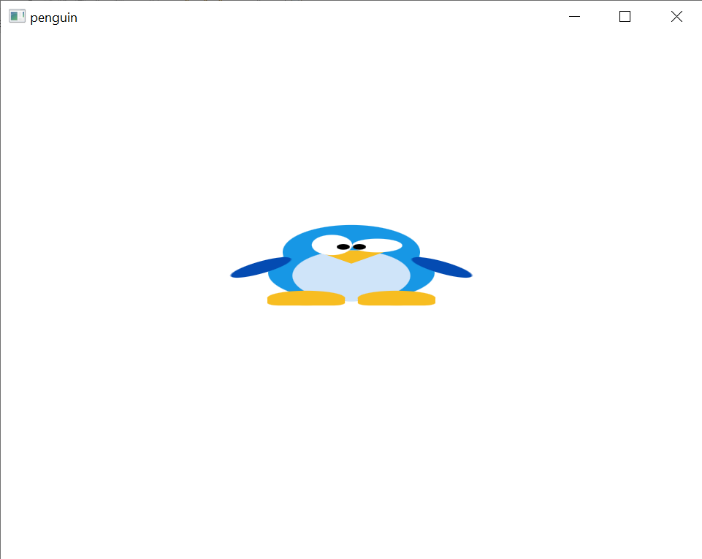
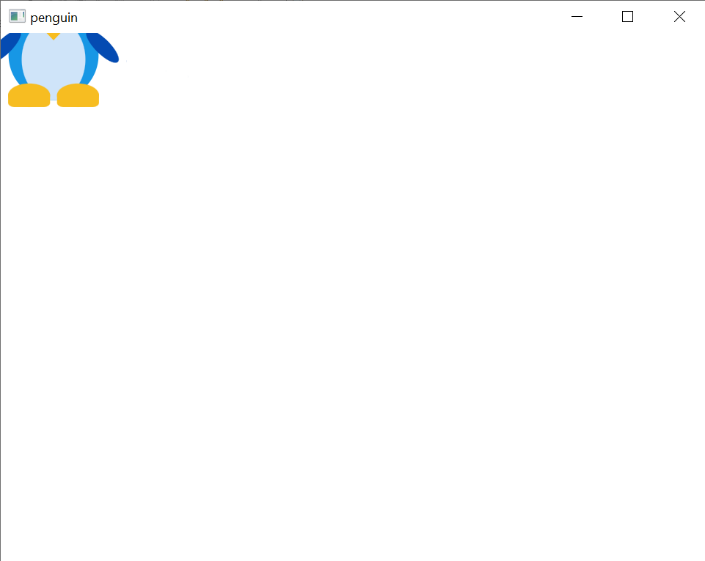


Рис.2-4. Перетворення

**Висновки**

Виконавши дану лабораторну роботу я розібрався зі структурою .bmp та особливостями використання файлів формату, а також використав файл цього формату разом зі стандартними функціями JavaFX для створення анімації примітивів.