Министерство науки и высшего образования Российской Федерации   
Федеральное государственное бюджетное образовательное   
учреждение высшего образования   
«Алтайский государственный технический   
университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Андреева А.Ю.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчёт

по лабораторной работе № 1

«Аффинные преобразования»

по дисциплине «Компьютерная графика»

Студент группы ПИ-92 Шульпов В.М.

Преподаватель доцент, к.ф.-м.н. Андреева А.Ю.

Барнаул 2022

**Задание:** Разработать программу аффинных преобразований и проецирования   
3d проволочного объекта (например, первая буква фамилии в 3d). Интерфейс должен позволять управлять текущим преобразованием объекта мышью или клавиатурой.

**Необходимый минимум:** все элементарные преобразования (перемещения, вращения и масштабирование). Кроме того, реализовать дополнительное динамическое преобразование (анимацию) по варианту.

**Вариант 16 (1).** Перемещение (анимированное) вдоль произвольной прямойна заданное расстояние с замедлением перед остановкой.

**Код программы**

package com.example.cg1.models;  
  
import static java.lang.Math.\*;  
  
//Аффинная матрица преобразования для трехмерного случая  
abstract public class AffineTransformationMatrix3D {  
  
 //вывести матрицу в консоль (тестовый метод)  
 private static void printMatrix(double[][] matrix) {  
 for (double[] line: matrix  
 ) {  
 for (double elem: line  
 ) {  
 System.*out*.print(elem + " ");  
 }  
 System.*out*.println();  
 }  
 }  
  
 //получить нулевую матрицу 4x4 (тестовый метод)  
 public static double[][] getZeroMatrix()  
 {  
 return new double[][]{  
 {0.0, 0.0, 0.0, 0.0},  
 {0.0, 0.0, 0.0, 0.0},  
 {0.0, 0.0, 0.0, 0.0},  
 {0.0, 0.0, 0.0, 0.0}};  
 }  
  
 //получить матрицу перемещения 4x4  
 public static double[][] getMovingMatrix(double toX, double toY, double toZ) {  
 return new double[][]{  
 {1.0, 0.0, 0.0, 0.0},  
 {0.0, 1.0, 0.0, 0.0},  
 {0.0, 0.0, 1.0, 0.0},  
 {toX, toY, toZ, 1.0}};  
 }  
  
 //получить матрицу поворота 4x4 вокруг оси X  
 public static double[][] getXRotationMatrix(double angle) {  
 return new double[][]{  
 {1.0, 0.0, 0.0, 0.0},  
 {0.0, *cos*(angle), *sin*(angle), 0.0},  
 {0.0, -*sin*(angle), *cos*(angle), 0.0},  
 {0.0, 0.0, 0.0, 1.0}};  
 }  
  
 //получить матрицу поворота 4x4 вокруг оси Y  
 public static double[][] getYRotationMatrix(double angle) {  
 return new double[][]{  
 {*cos*(angle), 0.0, -*sin*(angle), 0.0},  
 {0.0, 1.0, 0.0, 0.0},  
 {*sin*(angle), 0.0, *cos*(angle), 0.0},  
 {0.0, 0.0, 0.0, 1.0}};  
 }  
  
 //получить матрицу поворота 4x4 вокруг оси Z  
 public static double[][] getZRotationMatrix(double angle) {  
 return new double[][]{  
 {*cos*(angle), *sin*(angle), 0.0, 0.0},  
 {-*sin*(angle), *cos*(angle), 0.0, 0.0},  
 {0.0, 0.0, 1.0, 0.0},  
 {0.0, 0.0, 0.0, 1.0}};  
 }  
  
 //получить матрицу масштабирования 4x4  
 public static double[][] getScalingMatrix(double forX, double forY, double forZ) throws Exception {  
 if (forX <= 0 || forY <=0 || forZ <= 0) {  
 throw new Exception("scale factor cannot be less than zero");  
 }  
 return new double[][]{  
 {forX, 0.0, 0.0, 0.0},  
 {0.0, forY, 0.0, 0.0},  
 {0.0, 0.0, forZ, 0.0},  
 {0.0, 0.0, 0.0, 1.0}};  
 }  
  
 //получить матрицу отзеркаливания 4x4  
 public static double[][] getReflectionMatrix(int yoz, int zox, int xoy) throws Exception {  
 if (*abs*(yoz) != 1 || *abs*(zox) != 1 || *abs*(xoy) != 1) {  
 throw new Exception("method parameter is wrong");  
 }  
 return new double[][]{  
 {yoz, 0.0, 0.0, 0.0},  
 {0.0, zox, 0.0, 0.0},  
 {0.0, 0.0, xoy, 0.0},  
 {0.0, 0.0, 0.0, 1.0}};  
 }  
}

package com.example.cg1.models;  
  
import javafx.animation.KeyFrame;  
import javafx.animation.KeyValue;  
import javafx.animation.Timeline;  
import javafx.beans.value.WritableValue;  
import javafx.geometry.Bounds;  
import javafx.geometry.Point3D;  
import javafx.scene.canvas.Canvas;  
import javafx.util.Duration;  
  
public class CustomPoint3D {  
 private double x;  
 private double y;  
 private double z;  
  
 public double getX() {  
 return x;  
 }  
 public double getY() {  
 return y;  
 }  
 public double getZ() {  
 return z;  
 }  
  
 public void setX(double x) {  
 this.x = x;  
 }  
  
 public void setY(double y) {  
 this.y = y;  
 }  
  
 public void setZ(double z) {  
 this.z = z;  
 }  
  
 public CustomPoint3D(double x, double y, double z) {  
 this.x = x;  
 this.y = y;  
 this.z = z;  
 }  
  
 public static final int *n* = 4; //размер матрицы Аффинного преобразования (3D)  
  
 //получить вектор координат точки для Аффинного преобразования (3D)  
 private double[] getVectorForAffineTransformation() {  
 double[] vector = new double[4];  
 vector[0] = getX();  
 vector[1] = getY();  
 vector[2] = getZ();  
 vector[3] = 1.0;  
 return vector;  
 }  
  
 //сделать Аффинное преобразование точки (3D)  
 public void doAffineTransformation(double[][] affineTransformationMatrix) {  
 double[] vector = getVectorForAffineTransformation();  
 double[] res = new double[*n*];  
 for (int i = 0; i < *n*; i++)  
 {  
 res[i] = 0;  
 for (int j = 0; j < *n*; j++)  
 res[i] += vector[j] \* affineTransformationMatrix[j][i];  
 }  
 this.x = res[0];  
 this.y = res[1];  
 this.z = res[2];  
 }  
  
}

package com.example.cg1.models;  
  
  
import javafx.geometry.Point3D;  
import javafx.scene.canvas.Canvas;  
  
public class Line3D {  
 private final CustomPoint3D startPoint;  
 private final CustomPoint3D endPoint;  
  
 @Override  
 public String toString() {  
 return "Line3D{" +  
 "startPoint= (" +  
 startPoint.getX() + ", " +  
 startPoint.getY() + ", " +  
 startPoint.getZ() + ")" +  
 ", endPoint= (" +  
 endPoint.getX() + ", " +  
 endPoint.getY() + ", " +  
 endPoint.getZ() + ")" +  
 "}\n";  
 }  
  
 public Line3D(CustomPoint3D startPoint, CustomPoint3D endPoint) {  
 this.startPoint = startPoint;  
 this.endPoint = endPoint;  
 }  
  
 public Line3D(double startX, double startY, double startZ, double endX, double endY, double endZ) {  
 this.startPoint = new CustomPoint3D(startX, startY, startZ);  
 this.endPoint = new CustomPoint3D(endX, endY, endZ);  
 }  
  
 public CustomPoint3D getStartPoint() {  
 return startPoint;  
 }  
  
 public CustomPoint3D getEndPoint() {  
 return endPoint;  
 }  
  
 //сделать Аффинное преобразование отрезка (его точек)  
 public void doAffineTransformation(double[][] affineTransformationMatrix) {  
 getStartPoint().doAffineTransformation(affineTransformationMatrix);  
 getEndPoint().doAffineTransformation(affineTransformationMatrix);  
 }  
}

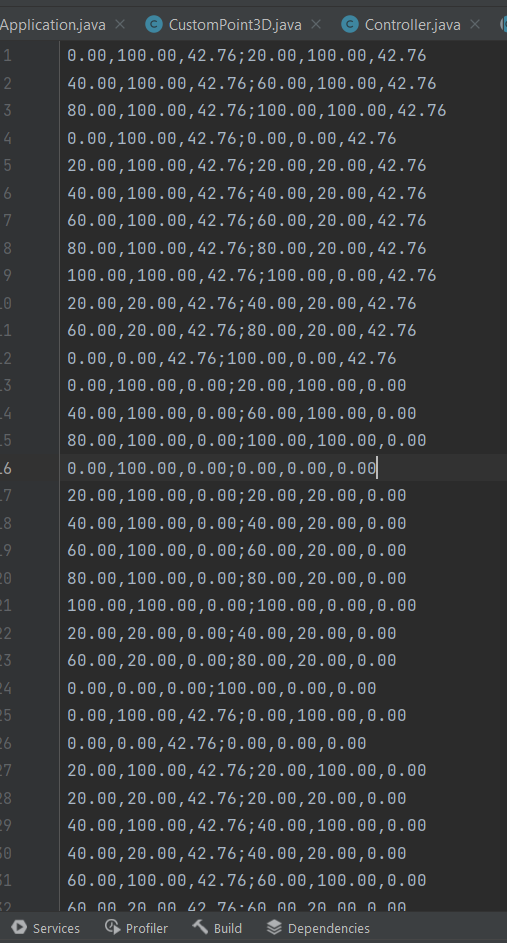
package com.example.cg1.models;  
  
import javafx.animation.\*;  
import javafx.beans.value.WritableValue;  
import javafx.geometry.Bounds;  
import javafx.scene.canvas.Canvas;  
import javafx.util.Duration;  
  
import java.io.\*;  
import java.text.DecimalFormat;  
import java.text.DecimalFormatSymbols;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Locale;  
  
import static java.lang.Double.*parseDouble*;  
  
public class Object3D {  
 private final List<Line3D> lineList = new ArrayList<>();  
  
 public void addLineToList(Line3D line) {  
 lineList.add(line);  
 }  
  
 //прочитать строки с данными о координатах из файла  
 public void readLinesFromFile(String filePath) {  
 try {  
 File file = new File(filePath);  
 FileReader fr = new FileReader(file);  
 BufferedReader reader = new BufferedReader(fr); //буфер для построчного считывания  
 String line = reader.readLine(); //считывание 1ой строки  
 lineList.clear();  
 while (line != null) {  
 this.addLineToList(convertFileStringToLine3D(line)); //добавление нового объекта-линии в List  
 line = reader.readLine();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 //записать новые значения координат проволочной фигуры в файл  
 public void writeLinesToFile(String filePath) {  
 try (FileWriter writer = new FileWriter(filePath, false)) {  
 String stringDataLine;  
 for (Line3D line3D : lineList  
 ) {  
 DecimalFormat dF = new DecimalFormat("#################0.00");  
 //для "." в double (по дефолту запятая)  
 dF.setDecimalFormatSymbols(new DecimalFormatSymbols(Locale.*CANADA*));  
 stringDataLine = String.*format*("%s,%s,%s;%s,%s,%s\n",  
 dF.format(line3D.getStartPoint().getX()), //X1  
 dF.format(line3D.getStartPoint().getY()), //Y1  
 dF.format(line3D.getStartPoint().getZ()), //Z1  
 dF.format(line3D.getEndPoint().getX()), //X2  
 dF.format(line3D.getEndPoint().getY()), //Y2  
 dF.format(line3D.getEndPoint().getZ())); //Z2  
  
 assert stringDataLine != null;  
 System.*out*.println(stringDataLine);  
 writer.write(stringDataLine);  
 writer.flush();  
 }  
  
 } catch (IOException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
  
 //создать объект отрезка из данных полученных из строки файла  
 private Line3D convertFileStringToLine3D(String string) {  
 String[] points = string.split(";");  
 String[] startCoords = points[0].split(",");  
 String[] endCoords = points[1].split(",");  
 CustomPoint3D startPoint = new CustomPoint3D(*parseDouble*(startCoords[0]),  
 *parseDouble*(startCoords[1]),  
 *parseDouble*(startCoords[2]));  
 CustomPoint3D endPoint = new CustomPoint3D(*parseDouble*(endCoords[0]),  
 *parseDouble*(endCoords[1]),  
 *parseDouble*(endCoords[2]));  
 return new Line3D(startPoint, endPoint);  
 }  
  
 public List<Line3D> getLineList() {  
 return lineList;  
 }  
  
 //сделать Аффинное преобразование всех отрезков объекта (их точек)  
 public void doAffineTransformation(double[][] affineTransformationMatrix) {  
 for (Line3D line : lineList  
 ) {  
 line.doAffineTransformation(affineTransformationMatrix);  
 }  
 }  
  
}

package com.example.cg1;  
  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class Application extends javafx.application.Application {  
  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("view.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 stage.setTitle("Lab 1");  
 stage.setScene(scene);  
 stage.show();  
 }  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

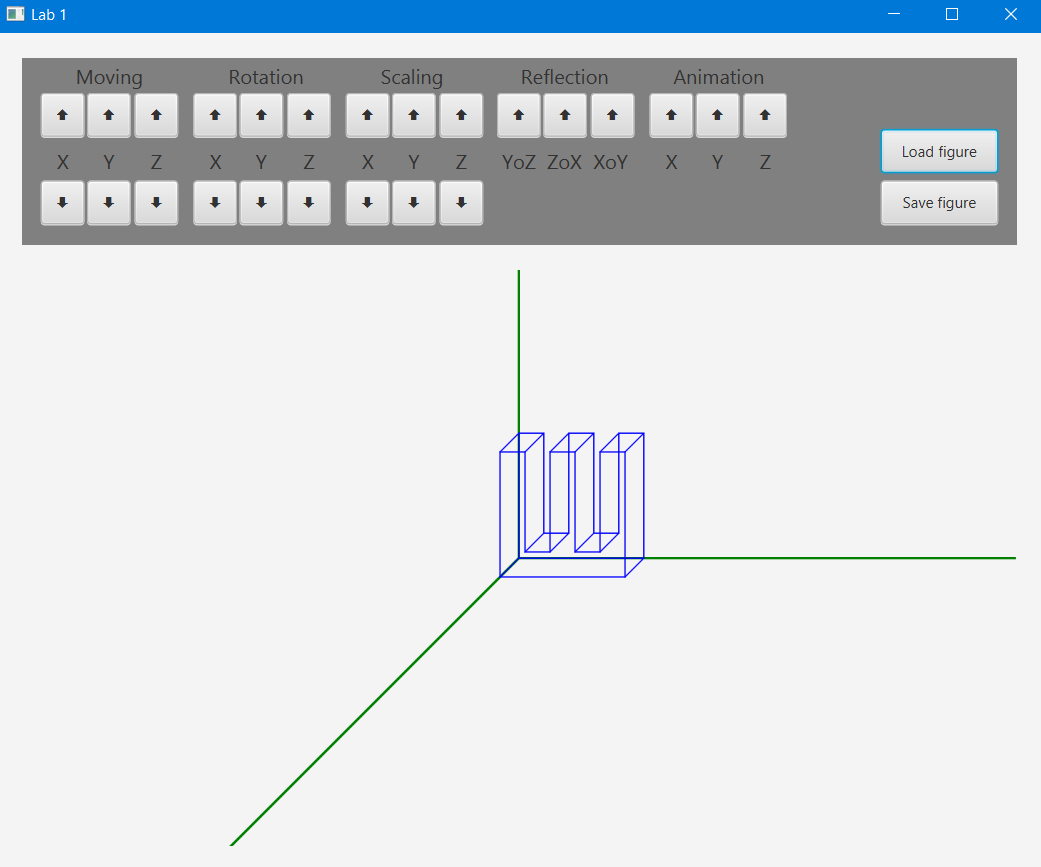
package com.example.cg1;  
  
import com.example.cg1.models.AffineTransformationMatrix3D;  
import com.example.cg1.models.Line3D;  
import com.example.cg1.models.Object3D;  
import javafx.animation.AnimationTimer;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.canvas.Canvas;  
import javafx.scene.canvas.GraphicsContext;  
import javafx.scene.paint.Color;  
import java.net.URISyntaxException;  
  
import static java.lang.Math.\*;  
  
public class Controller {  
 @FXML  
 private Canvas canvasPlot;  
  
 private final Object3D object3D = new Object3D(); //проволочная фигура  
 private GraphicsContext gc;  
  
 public Controller() throws URISyntaxException {  
 object3D.readLinesFromFile("lines.txt");  
 }  
  
 //FMXL конструктор  
 @FXML  
 public void initialize() {  
 gc = canvasPlot.getGraphicsContext2D();  
 gc.translate(canvasPlot.getWidth() / 2, canvasPlot.getHeight() / 2);  
  
 }  
 //нарисовать линию  
 private void drawLine(GraphicsContext gc, double xStart, double yStart, double xEnd, double yEnd) {  
 gc.setLineWidth(1.0);  
 gc.setStroke(Color.*BLUE*);  
 gc.strokeLine(xStart, yStart, xEnd, yEnd);  
 }  
  
 //очистить холст  
 public void clearCanvas(GraphicsContext gc) {  
  
 gc.clearRect(-canvasPlot.getWidth() / 2,  
 -canvasPlot.getHeight() / 2,  
 canvasPlot.getWidth(),  
 canvasPlot.getHeight());  
 }  
  
 //отрисовать оси координат  
 public void drawAxes(GraphicsContext gc) {  
 gc.setLineWidth(2.0);  
 gc.setStroke(Color.*GREEN*);  
  
 gc.strokeLine(0, 0, 500, 0);// Ox  
 gc.strokeLine(0, 0, 0, -500);// Oy  
 gc.strokeLine(0, 0, -500, 500);// Oz  
  
 }  
  
 //отрисовать все элементы (очистка, оси, фигура)  
 public void drawAll() {  
 clearCanvas(gc);  
 drawAxes(gc);  
 drawFigure(gc);  
 }  
  
 //отрисовать проволочную фигуру  
 public void drawFigure(GraphicsContext gc) {  
 for (Line3D line: object3D.getLineList()  
 ) {  
 //проецирование (комнатая проекция (45 градусов))  
 double startX = line.getStartPoint().getX() - line.getStartPoint().getZ() \* 0.5 \* Math.*cos*(Math.*PI* / 4);  
 double startY = -line.getStartPoint().getY() + line.getStartPoint().getZ() \* 0.5 \* Math.*cos*(Math.*PI* / 4);  
 double endX = line.getEndPoint().getX() - line.getEndPoint().getZ() \* 0.5 \* Math.*cos*(Math.*PI* / 4);  
 double endY = -line.getEndPoint().getY() + line.getEndPoint().getZ() \* 0.5 \* Math.*cos*(Math.*PI* / 4);  
  
 drawLine(gc, startX, startY, endX, endY);  
 }  
 }  
  
 //обработчик кнопки смещения по oX (увеличение)  
 public void onBtnMovingIncreaseXClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getMovingMatrix*(10.0, 0.0, 0.0));  
 drawAll();  
 }  
  
 //обработчик кнопки смещения по oX (уменьшение)  
 public void onBtnMovingReduceXClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getMovingMatrix*(-10.0, 0.0, 0.0));  
 drawAll();  
 }  
  
 //обработчик кнопки смещения по oY (увеличение)  
 public void onBtnMovingIncreaseYClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getMovingMatrix*(0.0, 10.0, 0.0));  
 drawAll();  
 }  
  
 //обработчик кнопки смещения по oY (уменьшение)  
 public void onBtnMovingReduceYClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getMovingMatrix*(0.0, -10.0, 0.0));  
 drawAll();  
 }  
  
 //обработчик кнопки смещения по oZ (увеличение)  
 public void onBtnMovingIncreaseZClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getMovingMatrix*(0.0, 0.0, 10.0));  
 drawAll();  
 }  
  
 //обработчик кнопки смещения по oZ (уменьшение)  
 public void onBtnMovingReduceZClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getMovingMatrix*(0.0, 0.0, -10.0));  
 drawAll();  
 }  
  
 //обработчик кнопки поворота вокруг oX (увеличение)  
 public void onBtnRotationIncreaseXClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getXRotationMatrix*(Math.*toRadians*(10)));  
 drawAll();  
 }  
  
 //обработчик кнопки поворота вокруг oX (уменьшение)  
 public void onBtnRotationReduceXClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getXRotationMatrix*(Math.*toRadians*(-10)));  
 drawAll();  
 }  
  
 //обработчик кнопки поворота вокруг oY (увеличение)  
 public void onBtnRotationIncreaseYClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getYRotationMatrix*(Math.*toRadians*(10)));  
 drawAll();  
 }  
  
 //обработчик кнопки поворота вокруг oY (уменьшение)  
 public void onBtnRotationReduceYClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getYRotationMatrix*(Math.*toRadians*(-10)));  
 drawAll();  
 }  
  
 //обработчик кнопки поворота вокруг oZ (увеличение)  
 public void onBtnRotationIncreaseZClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getZRotationMatrix*(Math.*toRadians*(10)));  
 drawAll();  
 }  
  
 //обработчик кнопки поворота вокруг oZ (уменьшение)  
 public void onBtnRotationReduceZClick(ActionEvent actionEvent) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getZRotationMatrix*(Math.*toRadians*(-10)));  
 drawAll();  
 }  
  
 //обработчик кнопки масштабирования относительно oX (увеличение в 1.5 раза)  
 public void onBtnScalingIncreaseXClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getScalingMatrix*(  
 (double) 3/2, 1, 1));  
 drawAll();  
 }  
  
 //обработчик кнопки масштабирования относительно oX (уменьшение в 1.5 раза)  
 public void onBtnScalingReduceXClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getScalingMatrix*(  
 (double) 2/3, 1, 1));  
 drawAll();  
 }  
  
 //обработчик кнопки масштабирования относительно oY (увеличение в 1.5 раза)  
 public void onBtnScalingIncreaseYClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getScalingMatrix*(  
 1, (double) 3/2, 1));  
 drawAll();  
 }  
  
 //обработчик кнопки масштабирования относительно oY (уменьшение в 1.5 раза)  
 public void onBtnScalingReduceYClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getScalingMatrix*(  
 1, (double) 2/3, 1));  
 drawAll();  
 }  
  
 //обработчик кнопки масштабирования относительно oZ (увеличение в 1.5 раза)  
 public void onBtnScalingIncreaseZClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getScalingMatrix*(  
 1, 1, (double) 3/2));  
 drawAll();  
 }  
  
 //обработчик кнопки масштабирования относительно oZ (уменьшение в 1.5 раза)  
 public void onBtnScalingReduceZClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getScalingMatrix*(  
 1, 1, (double) 2/3));  
 drawAll();  
 }  
  
 //обработчик кнопки отзеркаливания относительно YoZ  
 public void onBtnReflectionYoZClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getReflectionMatrix*(-1, 1, 1));  
 drawAll();  
 }  
  
 //обработчик кнопки отзеркаливания относительно ZoX  
 public void onBtnReflectionZoXClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getReflectionMatrix*(1, -1, 1));  
 drawAll();  
 }  
  
 //обработчик кнопки отзеркаливания относительно XoZ  
 public void onBtnReflectionXoYClick(ActionEvent actionEvent) throws Exception {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getReflectionMatrix*(1, 1, -1));  
 drawAll();  
 }  
  
 //обработчик кнопки сохранения координат фигуры  
 public void onBtnSavingFigureClick(ActionEvent actionEvent) throws URISyntaxException {  
 object3D.writeLinesToFile("lines.txt");  
 }  
  
 public void onBtnLoadingFigureClick(ActionEvent actionEvent) {  
 object3D.readLinesFromFile("lines.txt");  
 drawAll();  
 }  
  
 //*TODO можно найти более удачный таймер, в который можно передавать параметры (toX, toY, toZ, timingFunction())...* public void onBtnFigureTranslatingXAnimationClick(ActionEvent actionEvent) {  
 timer = new AnimationTimer(){  
 private int counter = 0;  
 private double toX = 10.0;  
 private int steps = 20; //количество формальных сдвигов  
  
 @Override  
 public void handle(long now) {  
 counter = translateAnimationProcess(toX, 0.0, 0.0, steps, counter);  
 drawAll();  
 }  
 };  
 timer.start();  
 }  
  
 public void onBtnFigureTranslatingYAnimationClick(ActionEvent actionEvent) {  
 timer = new AnimationTimer(){  
 private int counter = 0;  
 private double toY = 10.0;  
 private int steps = 20; //количество формальных сдвигов  
 @Override  
 public void handle(long now) {  
 counter = translateAnimationProcess(0.0, toY, 0.0, steps, counter);  
 drawAll();  
 }  
 };  
 timer.start();  
 }  
  
 //тайминг функция  
 private double timingFunction(int num, int steps) {  
 // первая половина времени - 10, вторая - плавное уменьшение по гиперболе  
 int center = steps/2;  
 if (num < center) {  
 System.*out*.println("num=" + num + " res=10");  
 return 10.0;  
 } else {  
 System.*out*.println("num =" + num + " res=" + 5 \* ((double)steps / *abs*(num)));  
 return 5 \* ((double)steps / num);  
  
 }  
 }  
 public void onBtnFigureTranslatingZAnimationClick(ActionEvent actionEvent) {  
 timer = new AnimationTimer(){  
 private int counter = 0;  
 private double toZ = 10.0;  
 private int steps = 20; //количество формальных сдвигов  
  
 @Override  
 public void handle(long now) {  
 toZ = timingFunction(counter, steps);  
 counter = translateAnimationProcess(0.0, 0.0, toZ, steps, counter);  
 drawAll();  
 }  
 };  
 timer.start();  
  
 }  
  
  
  
 //условное перемещение фигуры (с проверкой состояния)  
 private int translateAnimationProcess(double toX, double toY, double toZ, int steps, int counter) {  
 object3D.doAffineTransformation(AffineTransformationMatrix3D.*getMovingMatrix*(toX, toY, toZ));  
 if(counter == steps) {  
 timer.stop();  
 }  
 return ++counter;  
 }  
  
 private AnimationTimer timer;  
}

**Тест программы**

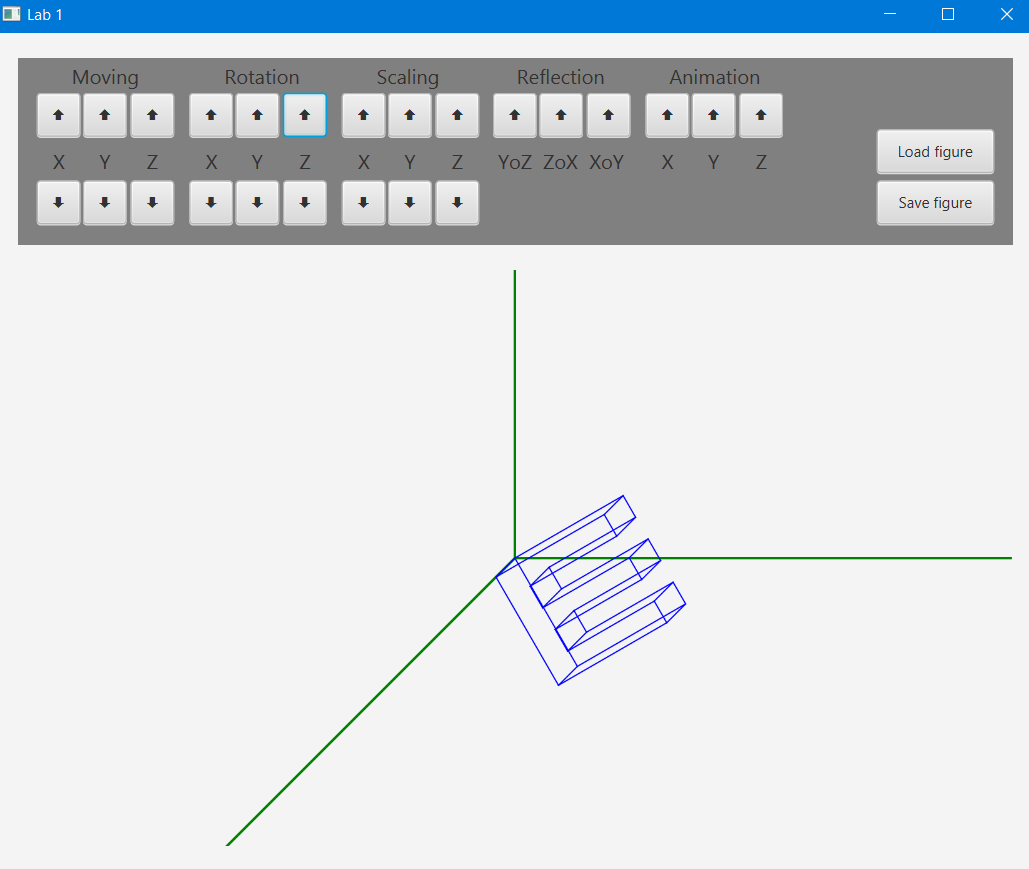
Информация в файле lines.txt (координаты вершин и рёбра объекта):



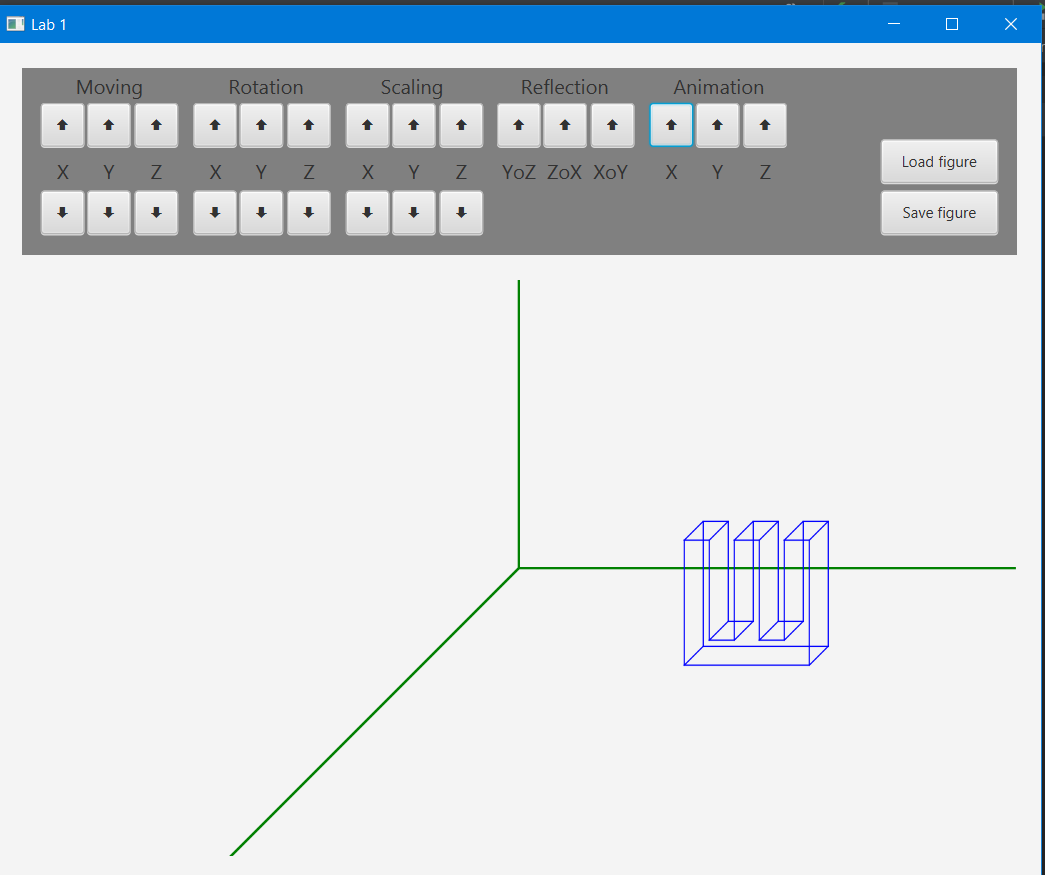
кабинетная проекция объекта



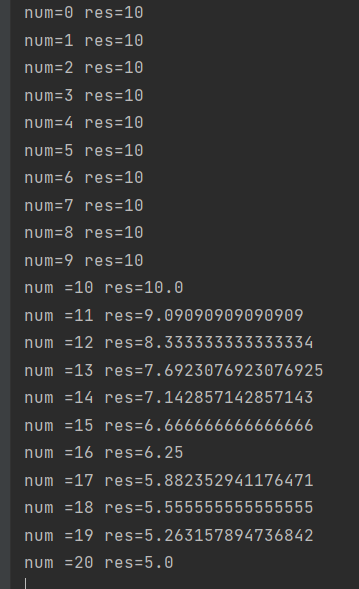
Вращение объекта



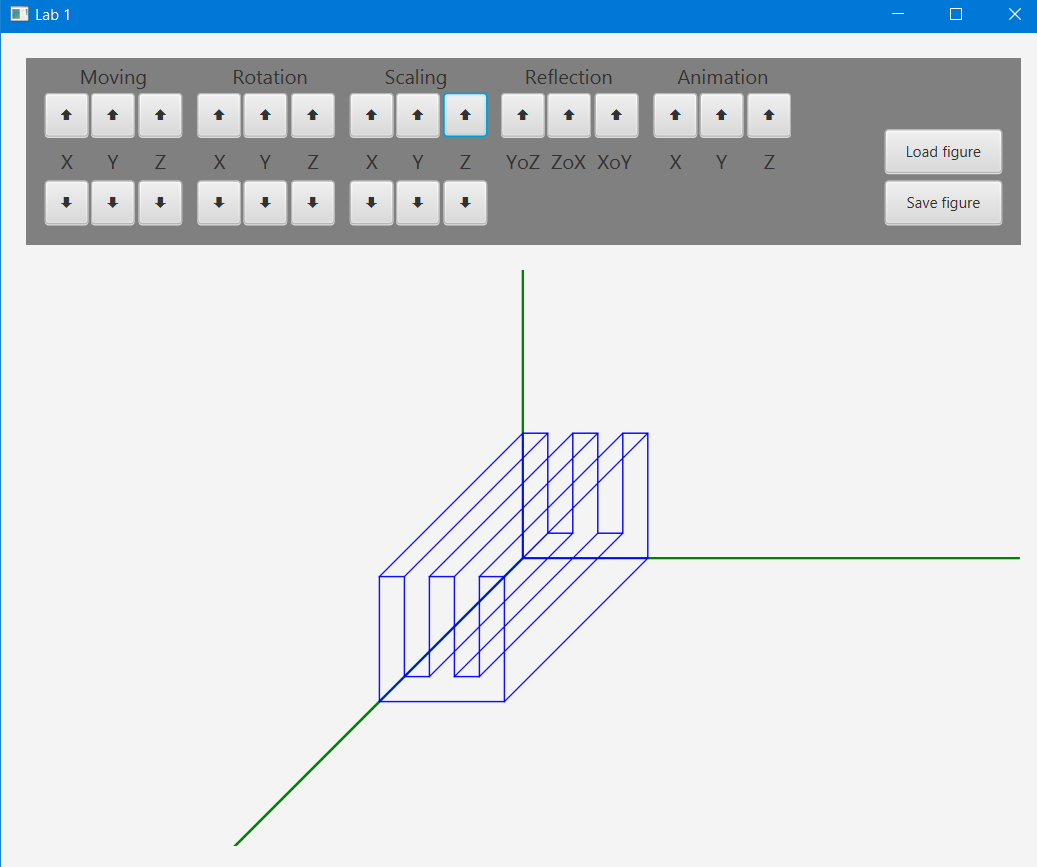
Перемещение



Тайминг функция, обеспечивающая плавную остановку фигуры:



Растяжение объекта



Отражение

