**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

СОГЛАСОВАНО  
Профессор департамента программной инженерии, кандидат технических наук  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.М. Гринкруг  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. УТВЕРЖДАЮ  
Академический руководитель образовательной программы «Программная инженерия»  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**ПРОГРАММНЫЙ КОМПОНЕНТ OPENGLVIEWER БИБЛИОТЕКИ JAVABEANS-КОМПОНЕНТ ДЛЯ 3D-ГРАФИКИ**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729. 04.01-01 12 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ163

\_\_\_\_\_\_\_\_\_\_\_\_ / Д.Е. Крайнов /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**Москва 2019**

**УТВЕРЖДЕН**

**RU.17701729.04.01-01 12 01-1-ЛУ**

**ПРОГРАММНЫЙ КОМПОНЕНТ OPENGLVIEWER БИБЛИОТЕКИ JAVABEANS-КОМПОНЕНТ ДЛЯ 3D-ГРАФИКИ**

**Текст программы**

**RU.17701729.04.01-01 12 01-1**

**Листов 54**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Москва 2019**

**СОДЕРЖАНИЕ**

1. [**Пакет ru.dansstuff.simpleopengl** 5](#_Toc6468130)

[1.1. Файл Main.java 5](#_Toc6468131)

1. [**Пакет ru.dansstuff.simpleopengl.math** 6](#_Toc6468132)

[2.1. Файл Vec3.java 6](#_Toc6468133)

[2.2. Файл Vec4.java 6](#_Toc6468134)

1. [**Пакет ru.dansstuff.simpleopengl.misc.helpers** 7](#_Toc6468135)

[3.1. Файл ObjectCreationFrameFactory.java 7](#_Toc6468136)

[3.2. Файл SceneFileHelper.java 8](#_Toc6468137)

1. [**Пакет ru.dansstuff.simpleopengl.objects** 10](#_Toc6468138)

[4.1. Файл Box.java 10](#_Toc6468139)

[4.2. Файл Cylinder.java 11](#_Toc6468140)

[4.3. Файл DirectionalLight.java 12](#_Toc6468141)

[4.4. Файл EmptyObject.java 13](#_Toc6468142)

[4.5. Файл GLObject.java 14](#_Toc6468143)

[4.6. Файл Line.java 17](#_Toc6468144)

[4.7. Файл OpenGLColor.java 18](#_Toc6468145)

[4.8. Файл Sphere.java 19](#_Toc6468146)

[4.9. Файл Triangle.java 20](#_Toc6468147)

1. [**Пакет ru.dansstuff.simpleopengl.objects.windows** 22](#_Toc6468148)

[5.1. Файл BoxFrame.java 22](#_Toc6468149)

[5.2. Файл CylinderFrame.java 23](#_Toc6468150)

[5.3. Файл DirectionalLightFrame.java 25](#_Toc6468151)

[5.4. Файл EmptyObjectFrame.java 27](#_Toc6468152)

[5.5. Файл LineFrame.java 28](#_Toc6468153)

[5.6. Файл SphereFrame.java 30](#_Toc6468154)

[5.7. Файл TriangleFrame.java 32](#_Toc6468155)

[5.8. Файл TypeBaseFrame.java 34](#_Toc6468156)

1. [**Пакет ru.dansstuff.simpleopengl.operations** 35](#_Toc6468157)

[6.1. Файл OpenGLOperation.java 35](#_Toc6468158)

[6.2. Файл Rotation.java 35](#_Toc6468159)

[6.3. Файл Translation.java 36](#_Toc6468160)

1. [**Пакет ru.dansstuff.simpleopengl.viewer** 37](#_Toc6468161)

[7.1. Файл GLViewerCanvas.java 37](#_Toc6468162)

[7.2. Файл OpenGLViewer.java 38](#_Toc6468163)

1. [**Пакет ru.dansstuff.simpleopengl.viewer.listeners** 45](#_Toc6468164)

[8.1. Файл OpenGLViewerPopupMenu.java 45](#_Toc6468165)

[8.2. Файл OpenGLViewerKeyListener.java 46](#_Toc6468166)

[8.3. Файл OpenGLViewerMouseListener.java 47](#_Toc6468167)

1. [**Пакет ru.dansstuff.simpleopengl.window** 49](#_Toc6468168)

[9.1. Файл OpenGLViewerMouseListener.java 49](#_Toc6468169)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 54](#_Toc6468170)

# **Пакет ru.dansstuff.simpleopengl**

## Файл Main.java

**Модуль, запускающий приложение**

package ru.dansstuff.simpleopengl;

import ru.dansstuff.simpleopengl.window.OpenGLTestFrame;

public class Main {

private static int width = 800;

private static int height = 600;

public static void main(String[] args) {

if (args.length == 2) {

try {

width = Integer.parseInt(args[0]);

height = Integer.parseInt(args[1]);

}

catch (Exception ex) {

System.out.println("Invalid width or height parameter");

return;

}

}

OpenGLTestFrame frame = new OpenGLTestFrame(width, height);

}

}

# **Пакет ru.dansstuff.simpleopengl.math**

## 2.1. Файл Vec3.java

**Класс, представляющий трехмерный числовой вектор.**

package ru.dansstuff.simpleopengl.math;

import lombok.\*;

import lombok.experimental.Accessors;

@NoArgsConstructor

@EqualsAndHashCode(callSuper = false)

public class Vec3 {

@Getter @Setter

private float x, y, z;

public Vec3(float x, float y, float z) {

this.x = x;

this.y = y;

this.z = z;

}

}

## 2.2. Файл Vec4.java

**Класс, представляющий четырехмерный числовой вектор.**

package ru.dansstuff.simpleopengl.math;

import lombok.\*;

import lombok.experimental.Accessors;

@NoArgsConstructor

@EqualsAndHashCode(callSuper = false)

public class Vec4 {

@Getter @Setter

private float x, y, z, w;

public Vec4(float x, float y, float z, float w) {

this.x = x;

this.y = y;

this.z = z;

this.w = w;

}

}

# **Пакет ru.dansstuff.simpleopengl.misc.helpers**

## 3.1. Файл ObjectCreationFrameFactory.java

**Класс для создания оконных объектов конфигурации новых примитивов.**

package ru.dansstuff.simpleopengl.misc.helpers;

import ru.dansstuff.simpleopengl.objects.\*;

import ru.dansstuff.simpleopengl.objects.windows.\*;

import ru.dansstuff.simpleopengl.window.OpenGLTestFrame;

import javax.swing.\*;

import java.awt.\*;

import java.util.HashMap;

import java.util.Map;

public class ObjectCreationFrameFactory {

private static Map<Class, Class> frameMap;

static {

frameMap = new HashMap<>();

try {

for (Class clazz : GLObject.getObjectTypes()) {

frameMap.put(clazz, ((GLObject)clazz.getConstructor().newInstance()).getFrameClass());

}

}

catch (Exception ex) {

ex.printStackTrace();

}

}

public static JFrame getFrame(Class clazz, OpenGLTestFrame parent) {

TypeBaseFrame frame;

try {

frame = (TypeBaseFrame)frameMap.get(clazz).getConstructor().newInstance();

}

catch (Exception ex) {

frame = new TypeBaseFrame() {

@Override

protected void createObject() {

}

};

}

frame.setParent(parent);

frame.setTitle(clazz.getSimpleName());

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

int width = parent.getWidth() / 4;

int height = parent.getHeight() / 3;

frame.setBounds((screenSize.width - width) / 2,

(screenSize.height - height) / 2,

width, height);

frame.setVisible(true);

return frame;

}

}

## 3.2. Файл SceneFileHelper.java

**Статический класс для сериализации графов сцены.**

package ru.dansstuff.simpleopengl.misc.helpers;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import com.google.gson.stream.JsonReader;

import ru.dansstuff.simpleopengl.misc.gson.RuntimeTypeAdapterFactory;

import ru.dansstuff.simpleopengl.objects.\*;

import java.io.\*;

public final class SceneFileHelper {

private static RuntimeTypeAdapterFactory<GLObject> rFactory = RuntimeTypeAdapterFactory

.of(GLObject.class, "type");

private static Gson gson;

static {

for (Class objectType : GLObject.getObjectTypes()) {

rFactory.registerSubtype(objectType, objectType.getSimpleName());

}

gson = new GsonBuilder().registerTypeAdapterFactory(rFactory).create();

}

public static GLObject readScene(File file) throws FileNotFoundException, IllegalStateException {

return gson.fromJson(new JsonReader(new FileReader(file)), GLObject.class);

}

public static String getSceneJson(GLObject root) {

return gson.toJson(root, GLObject.class);

}

public static void writeScene(GLObject root, File file) throws IOException {

BufferedWriter bw = new BufferedWriter(new FileWriter(file.getAbsolutePath()));

bw.write(getSceneJson(root));

bw.close();

}

}

# **Пакет ru.dansstuff.simpleopengl.objects**

## 4.1. Файл Box.java

**Класс, представляющий примитив «Параллелепипед».**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import com.jogamp.opengl.util.gl2.GLUT;

import lombok.\*;

import lombok.experimental.Accessors;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.windows.BoxFrame;

@Builder

@EqualsAndHashCode(callSuper = false)

public class Box extends GLObject {

@Getter @Setter

private Vec3 center;

@Getter

private float length;

@Getter @Setter

private OpenGLColor color;

public void setLength(float length) {

if (length < 0) {

throw new IllegalArgumentException("Invalid length: " + length);

}

this.length = length;

}

public Box() {

this(new Vec3(0, 0, 0), 1, OpenGLColor.WHITE);

}

public Box(Vec3 center, float length, OpenGLColor color) {

this.center = center;

this.length = length;

this.color = color;

frameClass = BoxFrame.class;

}

public void draw (GL2 gl) {

GLUT glut = new GLUT();

gl.glTranslatef(center.getX(), center.getY(), center.getZ());

gl.glColor3f(color.getR(), color.getG(), color.getB());

glut.glutSolidCube(length);

}

public void update() {

}

}

## 4.2. Файл Cylinder.java

**Класс, представляющий примитив «Цилиндр».**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import com.jogamp.opengl.glu.GLU;

import com.jogamp.opengl.glu.GLUquadric;

import lombok.\*;

import lombok.experimental.Accessors;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.windows.CylinderFrame;

@Builder

@EqualsAndHashCode(callSuper = false)

public class Cylinder extends GLObject {

@Getter @Setter

private Vec3 center;

@Getter

private float radius;

@Getter

private float height;

@Getter @Setter

private OpenGLColor color;

public void setHeight(float height) {

if (height < 0) {

throw new IllegalArgumentException("Invalid height: " + height);

}

this.height = height;

}

public void setRadius(float radius) {

if (radius < 0) {

throw new IllegalArgumentException("Invalid height: " + radius);

}

this.radius = radius;

}

public Cylinder() {

this(new Vec3(0, 0, 0), 1, 1, OpenGLColor.WHITE);

}

public Cylinder(Vec3 center, float radius, float height, OpenGLColor color) {

this.center = center;

this.radius = radius;

this.color = color;

this.height = height;

frameClass = CylinderFrame.class;

}

@Override

public void draw(GL2 gl) {

GLU glu = new GLU();

gl.glTranslatef(center.getX(), center.getY(), center.getZ());

gl.glColor3f(color.getR(), color.getG(), color.getB());

GLUquadric q = glu.gluNewQuadric();

glu.gluCylinder(q, radius, radius, height, 100, 100);

}

public void update() {

}

}

## 4.3. Файл DirectionalLight.java

**Класс, представляющий источник света.**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import lombok.\*;

import lombok.experimental.Accessors;

import ru.dansstuff.simpleopengl.math.\*;

import ru.dansstuff.simpleopengl.objects.windows.DirectionalLightFrame;

@Builder

@EqualsAndHashCode(callSuper = false)

public class DirectionalLight extends GLObject {

@Getter

private Vec4 color;

@Getter @Setter

private Vec4 pos;

@Getter @Setter

private int index;

public void setColor(Vec4 color) {

if (color.getX() < 0 || color.getY() < 0 || color.getZ() < 0) {

throw new IllegalArgumentException("Invalid color: " + color);

}

this.color = color;

}

public void setColor(OpenGLColor color) {

if (color.getR() < 0 || color.getG() < 0 || color.getB() < 0) {

throw new IllegalArgumentException("Invalid color: " + color);

}

this.color = new Vec4(color.getR(), color.getG(), color.getB(), this.color.getW());

}

public DirectionalLight() {

this(new Vec4(1, 1, 1, 1), new Vec4(0, 0, -5, 0), 0);

}

public DirectionalLight(Vec4 color, Vec4 pos, int index) {

this.color = color;

this.pos = pos;

this.index = index;

frameClass = DirectionalLightFrame.class;

}

@Override

public void draw(GL2 gl) {

gl.glLightfv(gl.GL\_LIGHT0 + index, gl.GL\_DIFFUSE, new float[] { color.getX(), color.getY(), color.getZ(), color.getW() }, index);

gl.glLightfv(gl.GL\_LIGHT0 + index, gl.GL\_POSITION, new float[] { pos.getX(), pos.getY(), pos.getZ(), pos.getW() }, index);

}

public void update() {

}

}

## 4.4. Файл EmptyObject.java

**Класс, представляющий пустой объект.**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import lombok.\*;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.windows.EmptyObjectFrame;

public class EmptyObject extends GLObject {

@Getter @Setter

public Vec3 center;

public EmptyObject() {

this(new Vec3(0, 0, 0));

}

public EmptyObject(Vec3 center) {

this.center = center;

frameClass = EmptyObjectFrame.class;

}

@Override

public void draw(GL2 gl) {

}

public void update() {

}

}

## 4.5. Файл GLObject.java

**Базовый класс примитивов и вершин графа, от которого наследуются все классы-примитивы**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import com.jogamp.opengl.util.texture.Texture;

import com.jogamp.opengl.util.texture.TextureIO;

import lombok.\*;

import lombok.experimental.Accessors;

import java.io.File;

import java.io.IOException;

import java.io.Serializable;

import java.util.\*;

public abstract class GLObject implements Serializable {

public abstract void draw(GL2 gl);

public abstract void update();

@Getter @Setter

protected String id = "";

@Getter @Setter

protected List<GLObject> children = new ArrayList<>();

@Getter

protected String type = getClass().getSimpleName();

@Getter @Setter

protected String textureFile;

@Getter @Setter

protected Texture texture;

@Getter

protected static Class frameClass;

public GLObject getChildren(int index) {

return children.get(index);

}

public void setChildren(int index, GLObject child) {

children.add(index, child);

}

public void resolveTexturesForTree(Map<String, Texture> textureMap) throws IOException {

resolveTexture(textureMap);

for (GLObject child : children) {

child.resolveTexturesForTree(textureMap);

}

}

private Texture getTextureFromFile(File file) throws IOException {

if (file.exists()) {

return TextureIO.newTexture(file, true);

}

return null;

}

public void resolveTexture(Map<String, Texture> textureMap) throws IOException {

if (textureFile == null) {

return;

}

if (textureMap.containsKey(textureFile)) {

texture = textureMap.get(textureFile);

return;

}

texture = getTextureFromFile(new File(textureFile));

textureMap.put(textureFile, texture);

}

public void addChild(GLObject child) {

children.add(child);

}

public void drawTree(GL2 gl) {

this.draw(gl);

this.update();

for (GLObject child : children) {

child.drawTree(gl);

}

}

public int getObjectsCount() {

int count = 1;

for (GLObject child : children) {

count += child.getObjectsCount();

}

return count;

}

public List<GLObject> getTreeAsList() {

List<GLObject> childrenList = new ArrayList<>();

childrenList.add(this);

for (GLObject child : children) {

childrenList.addAll(child.getTreeAsList());

}

return childrenList;

}

public void clear() {

for (GLObject child : children) {

child.clear();

}

children.clear();

}

public static Set<Class> getObjectTypes() {

Set<Class> types = new TreeSet<>(Comparator.comparing(Class::getSimpleName));

types.addAll(Arrays.asList(Box.class, Cylinder.class, DirectionalLight.class,

Line.class, Sphere.class, Triangle.class, EmptyObject.class));

return types;

}

}

## 4.6. Файл Line.java

**Класс, представляющий примитив «Линия».**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import lombok.\*;

import lombok.experimental.Accessors;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.windows.LineFrame;

@Builder

@EqualsAndHashCode(callSuper = false)

public class Line extends GLObject {

@Getter @Setter

private Vec3 p1, p2;

@Getter @Setter

private OpenGLColor color;

public Line() {

this(new Vec3(0, 0, 0), new Vec3(1, 0, 0), OpenGLColor.WHITE);

}

public Line(Vec3 p1, Vec3 p2, OpenGLColor color) {

this.p1 = p1;

this.p2 = p2;

this.color = color;

frameClass = LineFrame.class;

}

public void draw(GL2 gl) {

gl.glBegin(GL2.GL\_LINES);

gl.glColor3f(color.getR(), color.getG(), color.getB());

gl.glVertex3f(p1.getX(), p1.getY(), p1.getZ());

gl.glVertex3f(p2.getX(), p2.getY(), p2.getZ());

gl.glEnd();

}

public void update() {

}

}

## 4.7. Файл OpenGLColor.java

**Класс, представляющий цвет в палитре RGB.**

package ru.dansstuff.simpleopengl.objects;

import lombok.\*;

import lombok.experimental.Accessors;

import ru.dansstuff.simpleopengl.math.Vec3;

import java.util.Random;

@Builder

@EqualsAndHashCode

public class OpenGLColor {

@Getter

private float r, g, b;

private transient static Random random = new Random();

public void setB(float b) {

if (b < 0) {

throw new IllegalArgumentException("Invalid b: " + b);

}

this.b = b;

}

public void setG(float g) {

if (g < 0) {

throw new IllegalArgumentException("Invalid g: " + g);

}

this.g = g;

}

public void setR(float r) {

if (r < 0) {

throw new IllegalArgumentException("Invalid r: " + r);

}

this.r = r;

}

public OpenGLColor() {

this(1, 1, 1);

}

public OpenGLColor(Vec3 color) {

this(color.getX(), color.getY(), color.getZ());

}

public OpenGLColor(float r, float g, float b) {

this.r = r;

this.g = g;

this.b = b;

}

public static final OpenGLColor RED = new OpenGLColor(1, 0, 0);

public static final OpenGLColor GREEN = new OpenGLColor(0, 1, 0);

public static final OpenGLColor BLUE = new OpenGLColor(0, 0, 1);

public static final OpenGLColor WHITE = new OpenGLColor(1, 1, 1);

public static final OpenGLColor[] COLORS = {

RED, GREEN, BLUE, WHITE

};

public String toString() {

return "R " + r + " G " + g + " B " + b;

}

public static OpenGLColor getRandomColor() {

return COLORS[random.nextInt(COLORS.length)];

}

}

## 4.8. Файл Sphere.java

**Класс, представляющий примитив «Сфера».**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import com.jogamp.opengl.glu.GLU;

import com.jogamp.opengl.glu.GLUquadric;

import lombok.\*;

import lombok.experimental.Accessors;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.windows.SphereFrame;

@Builder

@EqualsAndHashCode(callSuper = false)

public class Sphere extends GLObject {

@Getter @Setter

private Vec3 center;

@Getter

private float radius;

@Getter @Setter

private OpenGLColor color;

public void setRadius(float radius) {

if (radius < 0) {

throw new IllegalArgumentException("Invalid radius: " + radius);

}

}

public Sphere() {

this(new Vec3(0, 0, 0), 1, OpenGLColor.WHITE);

}

public Sphere(Vec3 center, float radius, OpenGLColor color) {

this.center = center;

this.radius = radius;

this.color = color;

frameClass = SphereFrame.class;

}

@Override

public void draw(GL2 gl) {

GLU glu = new GLU();

gl.glTranslatef(center.getX(), center.getY(), center.getZ());

gl.glColor3f(color.getR(), color.getG(), color.getB());

GLUquadric q = glu.gluNewQuadric();

if (texture != null) {

glu.gluQuadricTexture(q, true);

gl.glBindTexture(gl.GL\_TEXTURE\_2D, texture.getTextureObject(gl));

}

glu.gluSphere(q, radius, 100, 100);

glu.gluDeleteQuadric(q);

}

public void update() {

}

}

## 4.9. Файл Triangle.java

**Класс, представляющий примитив «Треугольник».**

package ru.dansstuff.simpleopengl.objects;

import com.jogamp.opengl.GL2;

import lombok.\*;

import lombok.experimental.Accessors;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.windows.TriangleFrame;

@Builder

@EqualsAndHashCode(callSuper = false)

public class Triangle extends GLObject {

@Getter @Setter

private Vec3 p1, p2, p3;

@Getter @Setter

private OpenGLColor color;

public Triangle() {

this(new Vec3(0, 0, 0), new Vec3(1, 0, 0), new Vec3(0, 1, 0), OpenGLColor.WHITE);

}

public Triangle(Vec3 p1, Vec3 p2, Vec3 p3, OpenGLColor color) {

this.p1 = p1;

this.p2 = p2;

this.p3 = p3;

this.color = OpenGLColor.WHITE;

frameClass = TriangleFrame.class;

}

public void draw(GL2 gl) {

gl.glBegin(GL2.GL\_TRIANGLES);

gl.glColor3f(color.getR(), color.getG(), color.getB());

gl.glVertex3f(p1.getX(), p1.getY(), p1.getZ());

gl.glVertex3f(p2.getX(), p2.getY(), p2.getZ());

gl.glVertex3f(p3.getX(), p3.getY(), p3.getZ());

gl.glEnd();

}

public void update() {

}

}

# **Пакет ru.dansstuff.simpleopengl.objects.windows**

## 5.1. Файл BoxFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.Box;

import ru.dansstuff.simpleopengl.objects.GLObject;

import ru.dansstuff.simpleopengl.objects.OpenGLColor;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowEvent;

import java.text.NumberFormat;

public class BoxFrame extends TypeBaseFrame {

private JLabel xLabel = new JLabel("X");

private JTextField xField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel yLabel = new JLabel("Y");

private JTextField yField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel zLabel = new JLabel("Z");

private JTextField zField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel lenLabel = new JLabel("Length");

private JTextField lenField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel rLabel = new JLabel("R");

private JTextField rField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel gLabel = new JLabel("G");

private JTextField gField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel bLabel = new JLabel("B");

private JTextField bField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JButton okButton = new JButton("OK");

public BoxFrame() {

setLayout(new FlowLayout());

xField.setColumns(5); yField.setColumns(5); zField.setColumns(5);

rField.setColumns(5); gField.setColumns(5); bField.setColumns(5);

lenField.setColumns(5);

add(xLabel); add(xField); add(yLabel); add(yField); add(zLabel); add(zField);

add(lenLabel); add(lenField);

add(rLabel); add(rField); add(gLabel); add(gField); add(bLabel); add(bField);

okButton.addActionListener(e -> {

try {

createObject();

dispatchEvent(new WindowEvent(this, WindowEvent.WINDOW\_CLOSING));

}

catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Invalid input");

}

});

add(okButton);

}

protected void createObject() {

Box box = new Box();

box.setCenter(new Vec3(getNum(xField), getNum(yField), getNum(zField)));

box.setColor(new OpenGLColor(getNum(rField), getNum(gField), getNum(bField)));

box.setLength(getNum(lenField));

getParent().getCurrentObject().addChild(box);

}

}

## 5.2. Файл CylinderFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.Cylinder;

import ru.dansstuff.simpleopengl.objects.OpenGLColor;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowEvent;

import java.text.NumberFormat;

public class CylinderFrame extends TypeBaseFrame {

private JLabel xLabel = new JLabel("X");

private JTextField xField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel yLabel = new JLabel("Y");

private JTextField yField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel zLabel = new JLabel("Z");

private JTextField zField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel radiusLabel = new JLabel("Radius");

private JTextField radiusField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel heightLabel = new JLabel("Height");

private JTextField heightField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel rLabel = new JLabel("R");

private JTextField rField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel gLabel = new JLabel("G");

private JTextField gField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel bLabel = new JLabel("B");

private JTextField bField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JButton okButton = new JButton("OK");

public CylinderFrame() {

setLayout(new FlowLayout());

xField.setColumns(5); yField.setColumns(5); zField.setColumns(5);

rField.setColumns(5); gField.setColumns(5); bField.setColumns(5);

radiusField.setColumns(5); heightField.setColumns(5);

add(xLabel); add(xField); add(yLabel); add(yField); add(zLabel); add(zField);

add(radiusLabel); add(radiusField);

add(heightLabel); add(heightField);

add(rLabel); add(rField); add(gLabel); add(gField); add(bLabel); add(bField);

okButton.addActionListener(e -> {

try {

createObject();

dispatchEvent(new WindowEvent(this, WindowEvent.WINDOW\_CLOSING));

}

catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Invalid input");

}

});

add(okButton);

}

protected void createObject() {

Cylinder cylinder = new Cylinder();

cylinder.setCenter(new Vec3(getNum(xField), getNum(yField), getNum(zField)));

cylinder.setColor(new OpenGLColor(getNum(rField), getNum(gField), getNum(bField)));

cylinder.setRadius(getNum(radiusField));

cylinder.setHeight(getNum(heightField));

getParent().getCurrentObject().addChild(cylinder);

}

}

## 5.3. Файл DirectionalLightFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import ru.dansstuff.simpleopengl.math.Vec4;

import ru.dansstuff.simpleopengl.objects.DirectionalLight;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowEvent;

import java.text.NumberFormat;

public class DirectionalLightFrame extends TypeBaseFrame {

private JLabel xLabel = new JLabel("X");

private JTextField xField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel yLabel = new JLabel("Y");

private JTextField yField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel zLabel = new JLabel("Z");

private JTextField zField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel wLabel = new JLabel("W");

private JTextField wField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel rLabel = new JLabel("R");

private JTextField rField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel gLabel = new JLabel("G");

private JTextField gField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel bLabel = new JLabel("B");

private JTextField bField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel aLabel = new JLabel("A");

private JTextField aField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JButton okButton = new JButton("OK");

public DirectionalLightFrame() {

setLayout(new FlowLayout());

xField.setColumns(5); yField.setColumns(5); zField.setColumns(5); wField.setColumns(5);

rField.setColumns(5); gField.setColumns(5); bField.setColumns(5); aField.setColumns(5);

add(xLabel); add(xField); add(yLabel); add(yField); add(zLabel); add(zField); add(wLabel); add(wField);

add(rLabel); add(rField); add(gLabel); add(gField); add(bLabel); add(bField); add(aLabel); add(aField);

okButton.addActionListener(e -> {

try {

createObject();

dispatchEvent(new WindowEvent(this, WindowEvent.WINDOW\_CLOSING));

}

catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Invalid input");

}

});

add(okButton);

}

protected void createObject() {

DirectionalLight light = new DirectionalLight();

light.setColor(new Vec4(getNum(rField), getNum(gField), getNum(bField), getNum(aField)));

light.setPos(new Vec4(getNum(xField), getNum(yField), getNum(zField), getNum(wField)));

getParent().getCurrentObject().addChild(light);

}

}

## 5.4. Файл EmptyObjectFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.EmptyObject;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowEvent;

import java.text.NumberFormat;

public class EmptyObjectFrame extends TypeBaseFrame {

private JLabel xLabel = new JLabel("X");

private JTextField xField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel yLabel = new JLabel("Y");

private JTextField yField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel zLabel = new JLabel("Z");

private JTextField zField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JButton okButton = new JButton("OK");

public EmptyObjectFrame() {

setLayout(new FlowLayout());

xField.setColumns(5); yField.setColumns(5); zField.setColumns(5);

add(xLabel); add(xField); add(yLabel); add(yField); add(zLabel); add(zField);

okButton.addActionListener(e -> {

try {

createObject();

dispatchEvent(new WindowEvent(this, WindowEvent.WINDOW\_CLOSING));

}

catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Invalid input");

}

});

add(okButton);

}

protected void createObject() {

EmptyObject emptyObject = new EmptyObject();

emptyObject.setCenter(new Vec3(getNum(xField), getNum(yField), getNum(zField)));

getParent().getCurrentObject().addChild(emptyObject);

}

}

## 5.5. Файл LineFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.OpenGLColor;

import ru.dansstuff.simpleopengl.objects.Line;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowEvent;

import java.text.NumberFormat;

public class LineFrame extends TypeBaseFrame {

private JLabel x1Label = new JLabel("X1");

private JTextField x1Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel y1Label = new JLabel("Y1");

private JTextField y1Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel z1Label = new JLabel("Z1");

private JTextField z1Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel x2Label = new JLabel("X2");

private JTextField x2Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel y2Label = new JLabel("Y2");

private JTextField y2Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel z2Label = new JLabel("Z2");

private JTextField z2Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel rLabel = new JLabel("R");

private JTextField rField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel gLabel = new JLabel("G");

private JTextField gField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel bLabel = new JLabel("B");

private JTextField bField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JButton okButton = new JButton("OK");

public LineFrame() {

setLayout(new FlowLayout());

x1Field.setColumns(5); y1Field.setColumns(5); z1Field.setColumns(5);

x2Field.setColumns(5); y2Field.setColumns(5); z2Field.setColumns(5);

rField.setColumns(5); gField.setColumns(5); bField.setColumns(5);

add(x1Label); add(x1Field); add(y1Label); add(y1Field); add(z1Label); add(z1Field);

add(x2Label); add(x2Field); add(y2Label); add(y2Field); add(z2Label); add(z2Field);

add(rLabel); add(rField); add(gLabel); add(gField); add(bLabel); add(bField);

okButton.addActionListener(e -> {

try {

createObject();

dispatchEvent(new WindowEvent(this, WindowEvent.WINDOW\_CLOSING));

}

catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Invalid input");

}

});

add(okButton);

}

protected void createObject() {

Line line = new Line();

line.setP1(new Vec3(getNum(x1Field), getNum(y1Field), getNum(z1Field)));

line.setP2(new Vec3(getNum(x2Field), getNum(y2Field), getNum(z2Field)));

line.setColor(new OpenGLColor(getNum(rField), getNum(gField), getNum(bField)));

getParent().getCurrentObject().addChild(line);

}

}

## 5.6. Файл SphereFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.OpenGLColor;

import ru.dansstuff.simpleopengl.objects.Sphere;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowEvent;

import java.text.NumberFormat;

public class SphereFrame extends TypeBaseFrame {

private JLabel xLabel = new JLabel("X");

private JTextField xField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel yLabel = new JLabel("Y");

private JTextField yField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel zLabel = new JLabel("Z");

private JTextField zField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel radiusLabel = new JLabel("Radius");

private JTextField radiusField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel rLabel = new JLabel("R");

private JTextField rField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel gLabel = new JLabel("G");

private JTextField gField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel bLabel = new JLabel("B");

private JTextField bField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JButton okButton = new JButton("OK");

public SphereFrame() {

setLayout(new FlowLayout());

xField.setColumns(5); yField.setColumns(5); zField.setColumns(5);

radiusField.setColumns(5);

rField.setColumns(5); gField.setColumns(5); bField.setColumns(5);

add(xLabel); add(xField); add(yLabel); add(yField); add(zLabel); add(zField);

add(radiusLabel); add(radiusField);

add(rLabel); add(rField); add(gLabel); add(gField); add(bLabel); add(bField);

okButton.addActionListener(e -> {

try {

createObject();

dispatchEvent(new WindowEvent(this, WindowEvent.WINDOW\_CLOSING));

}

catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Invalid input");

}

});

add(okButton);

}

protected void createObject() {

Sphere sphere = new Sphere();

sphere.setCenter(new Vec3(getNum(xField), getNum(yField), getNum(zField)));

sphere.setColor(new OpenGLColor(getNum(rField), getNum(gField), getNum(bField)));

sphere.setRadius(getNum(radiusField));

getParent().getCurrentObject().addChild(sphere);

}

}

## 5.7. Файл TriangleFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.Triangle;

import ru.dansstuff.simpleopengl.objects.OpenGLColor;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowEvent;

import java.text.NumberFormat;

public class TriangleFrame extends TypeBaseFrame {

private JLabel x1Label = new JLabel("X1");

private JTextField x1Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel y1Label = new JLabel("Y1");

private JTextField y1Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel z1Label = new JLabel("Z1");

private JTextField z1Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel x2Label = new JLabel("X2");

private JTextField x2Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel y2Label = new JLabel("Y2");

private JTextField y2Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel z2Label = new JLabel("Z2");

private JTextField z2Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel x3Label = new JLabel("X3");

private JTextField x3Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel y3Label = new JLabel("Y3");

private JTextField y3Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel z3Label = new JLabel("Z3");

private JTextField z3Field = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel rLabel = new JLabel("R");

private JTextField rField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel gLabel = new JLabel("G");

private JTextField gField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JLabel bLabel = new JLabel("B");

private JTextField bField = new JFormattedTextField(NumberFormat.getNumberInstance());

private JButton okButton = new JButton("OK");

public TriangleFrame() {

setLayout(new FlowLayout());

x1Field.setColumns(5); y1Field.setColumns(5); z1Field.setColumns(5);

x2Field.setColumns(5); y2Field.setColumns(5); z2Field.setColumns(5);

x3Field.setColumns(5); y3Field.setColumns(5); z3Field.setColumns(5);

rField.setColumns(5); gField.setColumns(5); bField.setColumns(5);

add(x1Label); add(x1Field); add(y1Label); add(y1Field); add(z1Label); add(z1Field);

add(x2Label); add(x2Field); add(y2Label); add(y2Field); add(z2Label); add(z2Field);

add(x3Label); add(x3Field); add(y3Label); add(y3Field); add(z3Label); add(z3Field);

add(rLabel); add(rField); add(gLabel); add(gField); add(bLabel); add(bField);

okButton.addActionListener(e -> {

try {

createObject();

dispatchEvent(new WindowEvent(this, WindowEvent.WINDOW\_CLOSING));

}

catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Invalid input");

}

});

add(okButton);

}

protected void createObject() {

Triangle triangle = new Triangle();

triangle.setP1(new Vec3(getNum(x1Field), getNum(y1Field), getNum(z1Field)));

triangle.setP2(new Vec3(getNum(x2Field), getNum(y2Field), getNum(z2Field)));

triangle.setP3(new Vec3(getNum(x3Field), getNum(y3Field), getNum(z3Field)));

triangle.setColor(new OpenGLColor(getNum(rField), getNum(gField), getNum(bField)));

getParent().getCurrentObject().addChild(triangle);

}

}

## 5.8. Файл TypeBaseFrame.java

package ru.dansstuff.simpleopengl.objects.windows;

import lombok.Getter;

import lombok.Setter;

import ru.dansstuff.simpleopengl.window.OpenGLTestFrame;

import javax.swing.\*;

public abstract class TypeBaseFrame extends JFrame {

@Getter @Setter

protected OpenGLTestFrame parent;

abstract protected void createObject();

protected int getNum(JTextField field) {

return Integer.parseInt(field.getText());

}

}

# **Пакет ru.dansstuff.simpleopengl.operations**

## 6.1. Файл OpenGLOperation.java

**Интерфейс, представляющий трансформацию сцены или объекта**

package ru.dansstuff.simpleopengl.operations;

import com.jogamp.opengl.GLAutoDrawable;

public interface OpenGLOperation {

void doOperation(GLAutoDrawable glAutoDrawable);

}

## 6.2. Файл Rotation.java

**Класс, представляющий трансформацию поворота**

package ru.dansstuff.simpleopengl.operations;

import com.jogamp.opengl.GLAutoDrawable;

import lombok.Getter;

import lombok.Setter;

public class Rotation implements OpenGLOperation {

@Getter @Setter

public float x;

@Getter @Setter

public float y;

@Getter @Setter

public float z;

@Getter @Setter

public float angle;

public Rotation() {

}

public Rotation(float angle, float x, float y, float z) {

this.angle = angle;

this.x = x;

this.y = y;

this.z = z;

}

@Override

public void doOperation(GLAutoDrawable drawable) {

drawable.getGL().getGL2().glRotatef(angle, x, y, z);

}

}

## 6.3. Файл Translation.java

**Класс, представляющий трансформацию сдвига**

package ru.dansstuff.simpleopengl.operations;

import com.jogamp.opengl.GLAutoDrawable;

import lombok.Getter;

import lombok.Setter;

public class Translation implements OpenGLOperation {

@Getter @Setter

public float x;

@Getter @Setter

public float y;

@Getter @Setter

public float z;

public Translation() {

}

public Translation(float x, float y, float z) {

this.x = x;

this.y = y;

this.z = z;

}

@Override

public void doOperation(GLAutoDrawable drawable) {

drawable.getGL().getGL2().glTranslatef(x, y, z);

}

}

# **Пакет ru.dansstuff.simpleopengl.viewer**

## 7.1. Файл GLViewerCanvas.java

**Класс-обёртка над OpenGLViewer для привязки обработчиков событий**

package ru.dansstuff.simpleopengl.viewer;

import com.jogamp.opengl.awt.GLCanvas;

import lombok.Getter;

import lombok.Setter;

import ru.dansstuff.simpleopengl.viewer.OpenGLViewer;

import ru.dansstuff.simpleopengl.viewer.listeners.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.EventListener;

public class GLViewerCanvas extends GLCanvas {

@Getter @Setter

private Point curPos;

@Getter

private OpenGLViewer viewer;

public void setViewer(OpenGLViewer viewer) {

if (viewer == null) {

return;

}

removeGLEventListener(this.viewer);

this.viewer = viewer;

addGLEventListener(this.viewer);

}

private void setListeners() {

MouseAdapter listener = new OpenGLViewerMouseListener(viewer);

addMouseListener(listener);

addMouseWheelListener(listener);

addMouseMotionListener(listener);

addKeyListener(new OpenGLViewerKeyListener(viewer));

}

public GLViewerCanvas() {

this(new OpenGLViewer());

}

public GLViewerCanvas(OpenGLViewer viewer) {

setViewer(viewer);

setListeners();

curPos = new Point(-1, -1);

}

}

## 7.2. Файл OpenGLViewer.java

**Главный компонент рендеринга сцены**

package ru.dansstuff.simpleopengl.viewer;

import com.jogamp.opengl.GL2;

import com.jogamp.opengl.GLEventListener;

import com.jogamp.opengl.awt.GLCanvas;

import com.jogamp.opengl.glu.GLU;

import com.jogamp.opengl.GLAutoDrawable;

import com.jogamp.opengl.util.awt.TextRenderer;

import com.jogamp.opengl.util.gl2.GLUT;

import lombok.\*;

import ru.dansstuff.simpleopengl.math.Vec3;

import ru.dansstuff.simpleopengl.objects.\*;

import ru.dansstuff.simpleopengl.operations.OpenGLOperation;

import ru.dansstuff.simpleopengl.operations.Translation;

import java.awt.\*;

import java.io.IOException;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Queue;

import java.util.concurrent.ConcurrentLinkedQueue;

public class OpenGLViewer implements GLEventListener, Serializable {

@Getter @Setter

private GL2 gl;

@Getter @Setter

private GLU glu;

private TextRenderer textRenderer;

//@Getter @Setter

//private Vec3 cam;

@Getter @Setter

private Point curMousePos;

@Getter @Setter

private Vec3 rotn;

@Getter @Setter

private Vec3 center;

@Getter @Setter

private boolean drawAxis = true;

@Getter @Setter

private boolean enabled = true;

@Getter @Setter

private boolean drawDebugText = true;

@Getter @Setter

private boolean needTextureResolution = false;

private List<GLObject> axis;

private Queue<OpenGLOperation> pendingOperations;

@Getter @Setter

private GLObject root;

public OpenGLViewer(GLObject root) {

this.root = root;

textRenderer = new TextRenderer(new Font("Monospaced", Font.PLAIN, 12));

//cam = new Vec3(0, 0, -1);

curMousePos = new Point(-1, -1);

rotn = new Vec3(15, 45, 0);

center = new Vec3(0 ,0, -6);

axis = getAxis();

pendingOperations = new ConcurrentLinkedQueue<>();

}

public OpenGLViewer() {

this(new EmptyObject());

}

@Override

public void init(GLAutoDrawable glAutoDrawable) {

gl = glAutoDrawable.getGL().getGL2();

glu = GLU.createGLU(gl);

gl.glClearColor(0, 0, 0, 0);

gl.glShadeModel(gl.GL\_FLAT);

// Lighting setup

gl.glEnable(gl.GL\_LIGHTING);

gl.glEnable(gl.GL\_LIGHT0);

gl.glEnable(gl.GL\_COLOR\_MATERIAL);

gl.glEnable(gl.GL\_NORMALIZE);

gl.glColorMaterial(gl.GL\_FRONT\_AND\_BACK, gl.GL\_AMBIENT\_AND\_DIFFUSE);

float[] ambColor = new float[] {0.5f, 0.5f, 0.5f, 1};

gl.glLightModelfv(gl.GL\_LIGHT\_MODEL\_AMBIENT, ambColor, 0);

gl.glEnable(gl.GL\_DEPTH\_TEST);

gl.glEnable(gl.GL\_TEXTURE\_2D);

gl.glMatrixMode(gl.GL\_PROJECTION);

glu.gluPerspective(45.0f, (double)glAutoDrawable.getSurfaceWidth() / (double)glAutoDrawable.getSurfaceHeight(), 0.1f, 1000f);

gl.glMatrixMode(gl.GL\_MODELVIEW);

moveBackward(5);

}

@Override

public void dispose(GLAutoDrawable glAutoDrawable) {

}

@Override

public void display(GLAutoDrawable drawable) {

if (!enabled) {

return;

}

final GL2 gl = drawable.getGL().getGL2();

gl.glClear (gl.GL\_COLOR\_BUFFER\_BIT | gl.GL\_DEPTH\_BUFFER\_BIT );

if (needTextureResolution && root != null) {

try {

root.resolveTexturesForTree(new HashMap<>());

}

catch (IOException ex) {

ex.printStackTrace();

}

needTextureResolution = false;

}

// ---scene render---

gl.glPushMatrix();

gl.glTranslatef(center.getX(), center.getY(), center.getZ());

gl.glRotatef(rotn.getX(), 1, 0, 0);

gl.glRotatef(rotn.getY(), 0, 1, 0);

gl.glRotatef(rotn.getZ(), 0, 0, 1);

if (drawAxis) {

for (GLObject ax : axis) {

ax.draw(gl);

}

textRenderer.begin3DRendering();

textRenderer.draw3D("X", 3.2f, 0, 0, 0.05f);

textRenderer.draw3D("Y", 0, 3.2f, 0, 0.05f);

textRenderer.draw3D("Z", 0, 0, 3.2f, 0.05f);

textRenderer.end3DRendering();

}

if (root != null) {

gl.glPushMatrix();

root.drawTree(gl);

gl.glPopMatrix();

}

gl.glPopMatrix();

// ---scene render end---

// cam transformations

while (!pendingOperations.isEmpty()) {

pendingOperations.remove().doOperation(drawable);

}

if (drawDebugText) {

drawDebugText(drawable);

}

}

public List<GLObject> getAxis() {

List<GLObject> axis = new ArrayList<>();

// X axis

axis.add(new Line(new Vec3(-3, 0, 0), new Vec3(3, 0, 0), OpenGLColor.RED));

axis.add(new Line(new Vec3(3, 0, 0), new Vec3(2.9f, 0, -0.1f), OpenGLColor.RED));

axis.add(new Line(new Vec3(3, 0, 0), new Vec3(2.9f, 0, 0.1f), OpenGLColor.RED));

// Y axis

axis.add(new Line(new Vec3(0, -3, 0), new Vec3(0, 3, 0), OpenGLColor.GREEN));

axis.add(new Line(new Vec3(0, 3, 0), new Vec3(-0.1f, 2.9f, 0), OpenGLColor.GREEN));

axis.add(new Line(new Vec3(0, 3, 0), new Vec3(0.1f, 2.9f, 0), OpenGLColor.GREEN));

// Z axis

axis.add(new Line(new Vec3(0, 0, -3), new Vec3(0, 0, 3), OpenGLColor.BLUE));

axis.add(new Line(new Vec3(0, 0, 3), new Vec3(0, -0.1f, 2.9f), OpenGLColor.BLUE));

axis.add(new Line(new Vec3(0, 0, 3), new Vec3(0, 0.1f, 2.9f), OpenGLColor.BLUE));

return axis;

}

public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {

final GL2 gl = drawable.getGL().getGL2();

if (height == 0) height = 1;

gl.glViewport(0, 0, width, height);

gl.glMatrixMode(gl.GL\_PROJECTION);

gl.glLoadIdentity();

glu.gluPerspective(45.0f, (double)width / height, 0.1f, 1000f);

gl.glMatrixMode(gl.GL\_MODELVIEW);

gl.glLoadIdentity();

}

public void drawDebugText(GLAutoDrawable drawable) {

if (root != null) {

textRenderer.beginRendering(drawable.getSurfaceWidth(), drawable.getSurfaceHeight());

textRenderer.setColor(Color.WHITE);

textRenderer.draw(String.format("rotn x %.02f y %.02f z %.02f ", rotn.getX(), rotn.getY(), rotn.getZ()), 10, drawable.getSurfaceHeight() - 15);

textRenderer.draw(String.format("objects count: %d", root.getObjectsCount()), 10, drawable.getSurfaceHeight() - 30);

textRenderer.draw(String.format("mouse pos: %d %d", curMousePos.x, curMousePos.y), 10, drawable.getSurfaceHeight() - 45);

textRenderer.draw("controls: WASD/arrows + shift/ctrl for Y axis", 10, drawable.getSurfaceHeight() - 60);

textRenderer.endRendering();

}

}

public void rotLeft(float deg) {

rotn.setY(rotn.getY() + deg);

if (rotn.getY() >= 360 || rotn.getY() <= -360) rotn.setY(0);

//pendingOperations.add(new Rotation(-deg, 0, 1, 0));

}

public void rotRight(float deg) {

rotLeft(-deg);

}

public void rotUp(float deg) {

rotn.setX(rotn.getX() + deg);

if (rotn.getX() >= 360 || rotn.getX() <= -360) rotn.setX(0);

}

public void rotDown(float deg) {

rotUp(-deg);

}

public void scale(int direction) {

moveBackward(direction);

}

public void moveForward(float dist) {

//cam.setZ(cam.getZ() + dist);

pendingOperations.add(new Translation(0, 0, dist));

}

public void moveBackward(float dist) {

moveForward(-dist);

}

public void moveLeft(float dist) {

// center.x += dist;

//cam.setX(cam.getX() + dist);

pendingOperations.add(new Translation(dist, 0, 0));

}

public void moveRight(float dist) {

//center.x -= dist;

moveLeft(-dist);

}

public void moveUp(float dist) {

// center.x += dist;

//cam.setY(cam.getY() + dist);

pendingOperations.add(new Translation(0, -dist, 0));

}

public void moveDown(float dist) {

//center.x -= dist;

moveUp(-dist);

}

public void clear() {

root.clear();

root = null;

//cam = new Vec3(0, 0, -1);

center = new Vec3(0 ,0, -6);

rotn = new Vec3(15, 45, 0);

}

}

# **Пакет ru.dansstuff.simpleopengl.viewer.listeners**

## 8.1. Файл OpenGLViewerPopupMenu.java

**Контекстное меню, отображаемое при нажатии на окно OpenGLViewer правой клавишей мыши**

package ru.dansstuff.simpleopengl.viewer.listeners;

import ru.dansstuff.simpleopengl.misc.helpers.SceneFileHelper;

import ru.dansstuff.simpleopengl.viewer.OpenGLViewer;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.FileNotFoundException;

public class OpenGLViewerPopupMenu extends JPopupMenu {

public OpenGLViewerPopupMenu(OpenGLViewer viewer) {

JMenuItem loadSceneItem = new JMenuItem("Load scene...");

loadSceneItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

final JFileChooser fc = new JFileChooser();

fc.setFileFilter(new FileNameExtensionFilter("Model files", ".json"));

if (fc.showOpenDialog(OpenGLViewerPopupMenu.this) == JFileChooser.APPROVE\_OPTION) {

try {

viewer.setRoot(SceneFileHelper.readScene(fc.getSelectedFile()));

viewer.setNeedTextureResolution(true);

}

catch (FileNotFoundException | IllegalStateException ex) {

ex.printStackTrace();

}

}

}

});

add(loadSceneItem);

JMenuItem enableViewerItem = new JMenuItem("Start/stop rendering");

enableViewerItem.addActionListener(e -> viewer.setEnabled(!viewer.isEnabled()));

add(enableViewerItem);

JMenuItem showAxisItem = new JMenuItem("Show axis");

showAxisItem.addActionListener(e -> viewer.setDrawAxis(!viewer.isDrawAxis()));

add(showAxisItem);

JMenuItem resolveTexturesItem = new JMenuItem("Resolve textures");

resolveTexturesItem.addActionListener(e -> viewer.setNeedTextureResolution(true));

add(resolveTexturesItem);

}

}

## 8.2. Файл OpenGLViewerKeyListener.java

**Обработчик события нажатия на клавишу клавиатуры**

package ru.dansstuff.simpleopengl.viewer.listeners;

import ru.dansstuff.simpleopengl.viewer.OpenGLViewer;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

public class OpenGLViewerKeyListener extends KeyAdapter {

private OpenGLViewer viewer;

public OpenGLViewerKeyListener(OpenGLViewer viewer) {

this.viewer = viewer;

}

@Override

public void keyPressed(KeyEvent e) {

if (viewer == null) {

return;

}

int key = e.getKeyCode();

switch (key) {

case KeyEvent.VK\_W:

case KeyEvent.VK\_UP:

viewer.moveForward(0.1f);

break;

case KeyEvent.VK\_S:

case KeyEvent.VK\_DOWN:

viewer.moveBackward(0.1f);

break;

case KeyEvent.VK\_A:

case KeyEvent.VK\_LEFT:

viewer.moveLeft(0.1f);

break;

case KeyEvent.VK\_D:

case KeyEvent.VK\_RIGHT:

viewer.moveRight(0.1f);

break;

case KeyEvent.VK\_SHIFT:

viewer.moveUp(0.1f);

break;

case KeyEvent.VK\_CONTROL:

viewer.moveDown(0.1f);

break;

}

}

}

## 8.3. Файл OpenGLViewerMouseListener.java

**Обработчик событий, связанных с движением или нажатием на различные клавиши мыши**

package ru.dansstuff.simpleopengl.viewer.listeners;

import ru.dansstuff.simpleopengl.viewer.OpenGLViewer;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseWheelEvent;

public class OpenGLViewerMouseListener extends MouseAdapter {

private OpenGLViewer viewer;

public OpenGLViewerMouseListener(OpenGLViewer viewer) {

this.viewer = viewer;

}

@Override

public void mousePressed(MouseEvent e) {

viewer.getCurMousePos().x = -1;

viewer.getCurMousePos().y = -1;

}

@Override

public void mouseReleased(MouseEvent e) {

viewer.getCurMousePos().x = -1;

viewer.getCurMousePos().y = -1;

if (e.isPopupTrigger()) {

OpenGLViewerPopupMenu menu = new OpenGLViewerPopupMenu(viewer);

menu.show(e.getComponent(), e.getX(), e.getY());

}

}

@Override

public void mouseDragged(MouseEvent e) {

if (viewer == null) {

return;

}

if (e.getX() != viewer.getCurMousePos().x) {

if (viewer.getCurMousePos().x != -1)

viewer.rotLeft((e.getX() - viewer.getCurMousePos().x) \* 0.1f);

}

if (e.getY() != viewer.getCurMousePos().y) {

if (viewer.getCurMousePos().y != -1)

viewer.rotUp((e.getY() - viewer.getCurMousePos().y) \* 0.1f);

}

viewer.getCurMousePos().x = e.getX();

viewer.getCurMousePos().y = e.getY();

}

@Override

public void mouseWheelMoved(MouseWheelEvent e) {

viewer.scale(e.getWheelRotation());

}

}

# **Пакет ru.dansstuff.simpleopengl.window**

## 9.1. Файл OpenGLViewerMouseListener.java

**Окно для работы набора компонентов в режиме отдельного приложения**

package ru.dansstuff.simpleopengl.window;

import com.jogamp.opengl.util.Animator;

import lombok.Getter;

import lombok.Setter;

import ru.dansstuff.simpleopengl.misc.helpers.ObjectCreationFrameFactory;

import ru.dansstuff.simpleopengl.misc.helpers.SceneFileHelper;

import ru.dansstuff.simpleopengl.objects.GLObject;

import ru.dansstuff.simpleopengl.viewer.GLViewerCanvas;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.\*;

import java.io.FileNotFoundException;

import java.io.IOException;

public class OpenGLTestFrame

extends JFrame {

@Getter

private int width;

@Getter

private int height;

@Getter @Setter

private GLViewerCanvas canvas;

@Getter @Setter

private GLObject currentObject;

public void setWidth(int width) {

this.width = width;

canvas.setSize(this.width, this.height);

}

public void setHeight(int height) {

this.height = height;

canvas.setSize(this.width, this.height);

}

public OpenGLTestFrame(int width, int height) {

this(width, height, new GLViewerCanvas());

}

public OpenGLTestFrame(int width, int height, GLViewerCanvas canvas) {

this.canvas = canvas;

currentObject = canvas.getViewer().getRoot();

setWidth(width);

setHeight(height);

setSize(width, height);

setResizable(true);

add(canvas);

initWindow();

Animator animator = new Animator(canvas);

animator.start();

}

private void initWindow() {

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

setBounds((screenSize.width - width) / 2, (screenSize.height - height) / 2, width, height);

setTitle(this.getClass().getSimpleName());

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBackground(Color.BLACK);

initMenuBar();

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

}

catch (Exception ex) {

ex.printStackTrace();

}

setVisible(true);

}

private void initMenuBar() {

JMenuBar mainBar = new JMenuBar();

mainBar.add(getLoadSaveMenu());

mainBar.add(getSceneHandlingMenu());

mainBar.add(getObjectHandlingMenu());

setJMenuBar(mainBar);

}

private JMenu getLoadSaveMenu() {

JMenu loadSaveMenu = new JMenu("File");

JMenuItem loadMenuItem = new JMenuItem("Open scene...");

loadMenuItem.addActionListener(e -> {

final JFileChooser fc = new JFileChooser();

fc.setFileFilter(new FileNameExtensionFilter("Model files (\*.json)", "json"));

if (fc.showOpenDialog(OpenGLTestFrame.this) == JFileChooser.APPROVE\_OPTION) {

try {

canvas.getViewer().setRoot(SceneFileHelper.readScene(fc.getSelectedFile()));

canvas.getViewer().setNeedTextureResolution(true);

}

catch (FileNotFoundException ex) {

ex.printStackTrace();

}

catch (IllegalStateException ex) {

JOptionPane.showMessageDialog(this, "Could not parse scene from file");

}

}

});

JMenuItem saveMenuItem = new JMenuItem("Save scene...");

saveMenuItem.addActionListener(e -> {

final JFileChooser fc = new JFileChooser();

fc.setFileFilter(new FileNameExtensionFilter("Model files (\*.json)", "json"));

if(fc.showSaveDialog(OpenGLTestFrame.this) == JFileChooser.APPROVE\_OPTION) {

try {

SceneFileHelper.writeScene(canvas.getViewer().getRoot(), fc.getSelectedFile());

}

catch (IOException ex) {

ex.printStackTrace();

}

}

});

JMenuItem exitMenuItem = new JMenuItem("Exit");

exitMenuItem.addActionListener(e -> {

System.exit(0);

});

loadSaveMenu.add(loadMenuItem);

loadSaveMenu.add(saveMenuItem);

loadSaveMenu.addSeparator();

loadSaveMenu.add(exitMenuItem);

return loadSaveMenu;

}

private JMenu getSceneHandlingMenu() {

JMenu sceneHandlingMenu = new JMenu("Scene");

JMenuItem enableSceneItem = new JMenuItem("Enable/disable rendering");

enableSceneItem.addActionListener(e -> {

canvas.getViewer().setEnabled(!canvas.getViewer().isEnabled());

});

JMenuItem enableAxisItem = new JMenuItem("Enable/disable axis");

enableAxisItem.addActionListener(e -> {

canvas.getViewer().setDrawAxis(!canvas.getViewer().isDrawAxis());

});

JMenuItem enableDebugTextItem = new JMenuItem("Enable/disable debug text");

enableDebugTextItem.addActionListener(e -> {

canvas.getViewer().setDrawDebugText(!canvas.getViewer().isDrawDebugText());

});

sceneHandlingMenu.add(enableSceneItem);

sceneHandlingMenu.add(enableAxisItem);

sceneHandlingMenu.add(enableDebugTextItem);

return sceneHandlingMenu;

}

private JMenu getObjectHandlingMenu() {

JMenu objectHandlingMenu = new JMenu("Objects");

JMenu objectAddingMenu = new JMenu("Add child object");

for (Class clazz : GLObject.getObjectTypes()) {

JMenuItem typeItem = new JMenuItem(clazz.getSimpleName());

typeItem.addActionListener(e -> {

JFrame typeCreationFrame = ObjectCreationFrameFactory.getFrame(clazz, this);

});

objectAddingMenu.add(typeItem);

}

JMenuItem currentObjectSelectionItem = new JMenuItem("Select current object");

JMenuItem objectEditingItem = new JMenuItem("Edit object...");

objectHandlingMenu.add(objectAddingMenu);

objectHandlingMenu.addSeparator();

objectHandlingMenu.add(currentObjectSelectionItem);

objectHandlingMenu.add(objectEditingItem);

return objectHandlingMenu;

}

}

## ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |