**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Заместитель руководителя  Департамента  Программной Инженерии  Факультета Компьютерных Наук  профессор  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е. М. Гринкруг  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |  | УТВЕРЖДАЮ  Академический руководитель  образовательной программы  «Программная инженерия»  профессор, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | **RU.17701729.04.01-01 12** | | **JavaFX ПРИЛОЖЕНИЕ «КВАТЕРНИОНЫ В 3D ГРАФИКЕ»**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 12 01-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы 175 ПИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Т. О. Мартиросян /  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | |
|  | | |
|  | |  |

**Москва 2019**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.02.07-01 12 01-1 | | **JavaFX ПРИЛОЖЕНИЕ «КВАТЕРНИОНЫ В 3D ГРАФИКЕ»**  **Текст программы**  **RU.17701729. 02.07-01 12 01-1**  **Листов 28** | | | | |
|  |  | | | |
|  | | | |
|  | | | | |
|  | | |  | |

**2019**

СОДЕРЖАНИЕ

[1. ТЕКСТ ПРОГРАММЫ 2](#_Toc8875258)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 28](#_Toc8875259)

1. **ТЕКСТ ПРОГРАММЫ**

./src/sample/AxisGridPane.java

package sample;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.geometry.Point3D;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.scene.shape.Cylinder;

import java.io.IOException;

import java.net.URL;

import java.util.ResourceBundle;

public class AxisGridPane extends GridPane implements Initializable {

@FXML

public TextField axisX;

@FXML

public TextField axisY;

@FXML

public TextField axisZ;

@FXML

private GridPane gridpane;

private P2PCylinder c;

public TextField getAxisX() {

return axisX;

}

public TextField getAxisY() {

return axisY;

}

public TextField getAxisZ() {

return axisZ;

}

public AxisGridPane() {

super();

try {

FXMLLoader loader = new FXMLLoader(this.getClass().getResource("/AxisGridPane.fxml"));

loader.setController(AxisGridPane.this);

loader.setRoot(AxisGridPane.this);

loader.load();

axisX.textProperty().addListener((observable, oldValue, newValue) -> {

double x = Double.parseDouble(newValue);

double y = c.target.getY();

double z = c.target.getZ();

c.createConnection(new Point3D(0,0,0),new Point3D(x,y,z));

this.setInfo(c.target);

});

axisY.textProperty().addListener((observable, oldValue, newValue) -> {

double x = c.target.getX();

double y = Double.parseDouble(newValue);

double z = c.target.getZ();

c.createConnection(new Point3D(0,0,0),new Point3D(x,y,z));

this.setInfo(c.target);

});

axisZ.textProperty().addListener((observable, oldValue, newValue) -> {

double x = c.target.getX();

double y = c.target.getY();

double z = Double.parseDouble(newValue);

c.createConnection(new Point3D(0,0,0),new Point3D(x,y,z));

this.setInfo(c.target);

});

} catch (IOException ex) {

System.out.println(ex.toString());

}

}

@Override

public void initialize(URL location, ResourceBundle resources) {

}

public void setInfo(Point3D p){

axisX.setText(String.valueOf(p.getX()));

axisY.setText(String.valueOf(p.getY()));

axisZ.setText(String.valueOf(p.getZ()));

}

public void setC(P2PCylinder c){

this.c = c;

setInfo(c.target);

}

}

./src/sample/Main.java

package sample;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class Main extends Application {

MainWindowController mainWindowController;

@Override

public void start(Stage primaryStage) throws Exception {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/MainWindow.fxml"));

loader.setController(new MainWindowController());

VBox mainWindow = loader.load();

mainWindowController = loader.getController();

primaryStage.setTitle("JavaFx 3D Quaternions");

Scene scene = new Scene(mainWindow, 1100, 768);

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

./src/sample/MainWindowController.java

package sample;

import javafx.beans.property.\*;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.geometry.Point3D;

import javafx.geometry.Pos;

import javafx.scene.\*;

import javafx.scene.control.\*;

import javafx.scene.image.Image;

import javafx.scene.input.KeyCode;

import javafx.scene.input.MouseEvent;

import javafx.scene.input.PickResult;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.paint.PhongMaterial;

import javafx.scene.shape.Box;

import javafx.scene.shape.DrawMode;

import javafx.scene.shape.Shape3D;

import javafx.scene.shape.Sphere;

import javafx.scene.transform.Rotate;

import javafx.scene.transform.Translate;

import javafx.util.Duration;

import java.net.URL;

import java.util.ArrayList;

import java.util.ResourceBundle;

import java.util.concurrent.atomic.AtomicInteger;

public class MainWindowController implements Initializable {

@FXML

private MenuBar menuBar;

@FXML

private Accordion accordion;

@FXML

private Pane southEastResize;

@FXML

private VBox MainWindow;

@FXML

private StackPane contentPane;

@FXML

private VBox quaternionVbox;

@FXML

private VBox axisVbox;

@FXML

private MenuItem cubeMenuButton;

@FXML

private MenuItem sphereMenuButton;

@FXML

private HBox sliderHbox;

private static final double ZOOM\_SENSITIVITY = 0.2;

private static final int ZOOM\_MIN = 80;

private static final int ZOOM\_MAX = 2000;

private PerspectiveCamera camera;

Group g;

Shape3D box;

private SubScene axisScene;

private SubScene subScene;

P2PCylinder c;

Point3D end;

QuaternionSphere sphere;

ObjectProperty<Point3D> point3dprop = new SimpleObjectProperty<Point3D>(new Point3D(0, 0, 0));

private double anchorX = 0;

private double anchorY = 0;

private double anchorAngleX = 0;

private double anchorAngleY = 0;

private final DoubleProperty angleX = new SimpleDoubleProperty(0);

private final DoubleProperty angleY = new SimpleDoubleProperty(0);

private final IntegerProperty degreeProperty = new SimpleIntegerProperty(0);

private Rotate rotateX = new Rotate(0, Rotate.X\_AXIS);

private Rotate rotateY = new Rotate(0, Rotate.Y\_AXIS);

private Translate zoom = new Translate(0, 0, -200);

private boolean drawC = false;

QuaternionGridPane quaternionGridPane;

AxisGridPane axisGridPane;

public MenuBar getMenuBar() {

return menuBar;

}

public Accordion getAccordion() {

return accordion;

}

public Pane getSouthEastResize() {

return southEastResize;

}

public VBox getMainWindow() {

return MainWindow;

}

public StackPane getContentPane() {

return contentPane;

}

Matrix4x3 mat = new Matrix4x3();

private static double cos(double i) {

return Math.cos(i \* Math.PI / 180);

}

private static double sin(double i) {

return Math.sin(i \* Math.PI / 180);

}

@Override

public void initialize(URL location, ResourceBundle resources) {

// quaternionGridPane.prefWidthProperty().bind(contentPane.widthProperty());

}

Point3D pointStart;

private void prepareQuaternionSupport(ProjectType type) {

c = new P2PCylinder();

axisGridPane.setC(c);

subScene.addEventHandler(MouseEvent.MOUSE\_PRESSED, event -> {

if (!drawC) {

if (event.isAltDown()) {

System.out.println(event.getPickResult());

pointStart = new Point3D(0, 0, 0);

System.out.println(pointStart);

drawC = true;

}

}

});

subScene.addEventHandler(MouseEvent.MOUSE\_DRAGGED, event -> {

if (event.isAltDown()) {

if (drawC) {

int idx = g.getChildren().indexOf(c);

System.out.println(camera.getRotationAxis());

System.out.println(camera.getTransforms());

Point3D subsceneCenter = new Point3D(subScene.getWidth() / 2, subScene.getHeight() / 2, subScene.getTranslateZ());

end = new Point3D(event.getSceneX() - subsceneCenter.getX(), event.getSceneY() - subsceneCenter.getY(), event.getZ());

System.out.println(end);

end = rotateY.transform(end);

System.out.println(end);

end = rotateX.transform(end);

point3dprop.setValue(end);

System.out.println(end);

c = new P2PCylinder(pointStart, end);

axisGridPane.setC(c);

if (idx != -1)

g.getChildren().set(idx, c);

else g.getChildren().add(c);

}

}

});

subScene.addEventHandler(MouseEvent.MOUSE\_RELEASED, event -> {

drawC = false;

});

if (type == ProjectType.CUBE) {

box.getTransforms().setAll(mat);

// binding degree slider

Slider degreeSlider = new Slider(0, 360, 0);

Label degreeLabel = new Label(null);

sliderHbox.getChildren().removeAll(sliderHbox.getChildren());

sliderHbox.getChildren().addAll(degreeSlider, degreeLabel);

degreeSlider.valueProperty().addListener((obs, oldval, newVal) ->

degreeSlider.setValue(newVal.intValue()));

degreeProperty.bind(degreeSlider.valueProperty());

degreeLabel.textProperty().bind(degreeSlider.valueProperty().asString());

degreeSlider.valueProperty().addListener((observable, oldValue, newValue) -> {

double alpha = newValue.doubleValue();

double cos = cos(alpha / 2);

double sin = sin(alpha / 2);

Point3D norm = point3dprop.get().normalize();

Matrix4x3 mat = new Matrix4x3();

Quaternion qt = new Quaternion((norm.getX() \* sin), (norm.getY() \* sin), (norm.getZ() \* sin), cos);

box.getTransforms().setAll(mat.rotate(qt));

quaternionGridPane.setInfo(qt);

});

}

}

private void prepareSubscene(ProjectType type) {

if (type == ProjectType.CUBE) {

g = new Group();

box = new Box(100, 100, 100);

box.setDrawMode(DrawMode.FILL);

PhongMaterial material = new PhongMaterial();

material.setDiffuseMap(new Image("/diamond.jpg"));

box.setMaterial(material);

box.setPickOnBounds(true);

g.getChildren().add(box);

subScene = new SubScene(g, 400, 400, true, SceneAntialiasing.BALANCED);

subScene.setFill(Color.TRANSPARENT);

prepareCamera();

addCameraRotations();

subScene.setCamera(camera);

}

if (type == ProjectType.SPHERE) {

g = new Group();

sphere = new QuaternionSphere(40);

sphere.setDrawMode(DrawMode.LINE);

Group sphereGroup = new Group();

sphereGroup.getChildren().add(sphere);

g.getChildren().add(sphereGroup);

AtomicInteger clicks = new AtomicInteger();

ArrayList<Node> sphereChildren = new ArrayList<>();

ArrayList<Node> groupChildren = new ArrayList<>();

g.setOnMousePressed(event -> {

clicks.getAndIncrement();

if(clicks.get()>=3){

sphereGroup.getChildren().removeAll(sphereChildren);

g.getChildren().removeAll(groupChildren);

clicks.set(1);

}

Sphere s = new Sphere(2);

Point3D point = event.getPickResult().getIntersectedPoint();

s.setTranslateX(point.getX());

s.setTranslateY(point.getY());

s.setTranslateZ(point.getZ());

if(clicks.get()<2){

groupChildren.add(s);

g.getChildren().add(s);

}else {

sphereChildren.add(s);

sphereGroup.getChildren().add(s);

}

System.out.println("Picked placed");

});

subScene = new SubScene(g, 400, 400, true, SceneAntialiasing.BALANCED);

subScene.setFill(Color.TRANSPARENT);

prepareCamera();

addCameraRotations();

subScene.setCamera(camera);

}

}

private void prepareAxisScene() {

Group axis = buildAxes();

axisScene = new SubScene(axis, 200, 200, true, SceneAntialiasing.BALANCED);

PerspectiveCamera axisCamera = new PerspectiveCamera(true);

axisCamera.setNearClip(0.1);

axisCamera.setFarClip(100000.0);

axisCamera.setTranslateZ(-330);

axisCamera.setVerticalFieldOfView(false);

axisScene.setCamera(axisCamera);

axis.getTransforms().addAll(rotateX, rotateY);

contentPane.getChildren().add(axisScene);

contentPane.setAlignment(axisScene, Pos.TOP\_RIGHT);

}

private void prepareCamera() {

camera = new PerspectiveCamera(true);

camera.setNearClip(0.1);

camera.setFarClip(100000.0);

camera.setVerticalFieldOfView(false);

//camera.setFieldOfView(42);

}

private void addCameraRotations() {

camera.getTransforms().addAll(rotateX, rotateY, zoom);

rotateX.angleProperty().bind(angleX);

rotateY.angleProperty().bind(angleY);

subScene.setOnMousePressed(me -> {

PickResult pr = me.getPickResult();

subScene.requestFocus();

System.out.println(pr);

anchorX = me.getSceneX();

anchorY = me.getSceneY();

anchorAngleX = angleX.get();

anchorAngleY = angleY.get();

});

subScene.setOnMouseDragged(event -> {

if (!event.isAltDown()) {

angleX.set(((anchorAngleX - (anchorY - event.getSceneY())) % 360 + 540) % 360 - 180);

angleY.set(((anchorAngleY + anchorX - event.getSceneX()) % 360 + 540) % 360 - 180);

}

});

subScene.setOnKeyPressed(event -> {

if (event.getCode() == KeyCode.R) {

angleX.set(0);

angleY.set(0);

//g.getChildren().remove(c);

drawC = false;

}

});

subScene.setOnScroll(event -> zoomCameraBy(event.getDeltaY()));

}

private void zoomCameraBy(double diff) {

if (diff == 0)

return;

if (-(zoom.getZ() + diff) >= ZOOM\_MIN && -(zoom.getZ() + diff) <= ZOOM\_MAX)

zoom.setZ(zoom.getZ() + diff \* ZOOM\_SENSITIVITY);

}

private Group buildAxes() {

Group axisGroup = new XYZAxis();

System.out.println("buildAxes()");

return axisGroup;

}

private void prepareTextures() {

}

@FXML

void onCubeMenuButtonClicked(ActionEvent event) {

newProject();

axisGridPane = new AxisGridPane();

quaternionGridPane = new QuaternionGridPane();

prepareSubscene(ProjectType.CUBE);

prepareAxisScene();

prepareQuaternionSupport(ProjectType.CUBE);

contentPane.getChildren().add(subScene);

axisVbox.getChildren().add(axisGridPane);

quaternionVbox.getChildren().add(quaternionGridPane);

subScene.widthProperty().bind(contentPane.widthProperty());

subScene.heightProperty().bind(contentPane.heightProperty());

subScene.setManaged(false);

axisScene.toFront();

}

private void newProject() {

contentPane.getChildren().removeAll(contentPane.getChildren());

axisVbox.getChildren().removeAll(axisVbox.getChildren());

quaternionVbox.getChildren().removeAll(quaternionVbox.getChildren());

angleX.set(0);

angleY.set(0);

}

@FXML

void onSphereMenuButtonClicked(ActionEvent event) {

newProject();

prepareSubscene(ProjectType.SPHERE);

prepareAxisScene();

// prepareQuaternionSupport();

quaternionGridPane = new QuaternionGridPane();

Slider durationSlider = new Slider(1, 15, 3);

Label durationLabel = new Label(null);

sliderHbox.getChildren().removeAll(sliderHbox.getChildren());

sliderHbox.getChildren().addAll(durationSlider, durationLabel);

durationSlider.valueProperty().addListener((obs, oldval, newVal) ->

durationSlider.setValue(newVal.intValue()));

durationLabel.textProperty().bind(durationSlider.valueProperty().asString());

contentPane.getChildren().add(subScene);

Button btn = new Button("Play");

btn.setOnMouseClicked(event1 -> {

sphere.setDuration(Duration.seconds(durationSlider.getValue()));

quaternionGridPane.setInfo(sphere.getState());

sphere.play();

});

axisVbox.getChildren().add(btn);

quaternionVbox.getChildren().add(quaternionGridPane);

subScene.widthProperty().bind(contentPane.widthProperty());

subScene.heightProperty().bind(contentPane.heightProperty());

subScene.setManaged(false);

axisScene.toFront();

}

}

enum ProjectType {

CUBE, SPHERE, IMPORT;

}

./src/sample/Matrix4x3.java

package sample;

import javafx.scene.transform.Affine;

import com.sun.javafx.geom.\*;

public class Matrix4x3 extends Affine {

public Matrix4x3 rotate(Quaternion quat) {

double wp = quat.getW() \* quat.getW(),

xp = quat.getX() \* quat.getX(),

yp = quat.getY() \* quat.getY(),

zp = quat.getZ() \* quat.getZ();

double zw = quat.getZ() \* quat.getW(), dzw = zw + zw;

double xy = quat.getX() \* quat.getY(), dxy = xy + xy;

double xz = quat.getX() \* quat.getZ(), dxz = xz + xz;

double yw = quat.getY() \* quat.getW(), dyw = yw + yw;

double yz = quat.getY() \* quat.getZ(), dyz = yz + yz;

double xw = quat.getX() \* quat.getW(), dxw = xw + xw;

setMxx(wp + xp - zp - yp);

setMxy(dxy + dzw);

setMxz(dxz - dyw);

setMyx(dxy - dzw);

setMyy(yp - zp + wp - xp);

setMyz(dyz + dxw);

setMzx(dyw + dxz);

setMzy(dyz - dxw);

setMzz(zp - yp - xp + wp);

setTx(0.0);

setTy(0.0);

setTz(0.0);

return this;

}

public Matrix4x3 rotateGeneric(Quaternion quat) {

double w2 = quat.getW() \* quat.getW(), x2 = quat.getX() \* quat.getX();

double y2 = quat.getY() \* quat.getY(), z2 = quat.getZ() \* quat.getZ();

double zw = quat.getZ() \* quat.getW(), dzw = zw + zw, xy = quat.getX() \* quat.getY(), dxy = xy + xy;

double xz = quat.getX() \* quat.getZ(), dxz = xz + xz, yw = quat.getY() \* quat.getW(), dyw = yw + yw;

double yz = quat.getY() \* quat.getZ(), dyz = yz + yz, xw = quat.getX() \* quat.getW(), dxw = xw + xw;

double rm00 = w2 + x2 - z2 - y2, rm01 = dxy + dzw, rm02 = dxz - dyw;

double rm10 = dxy - dzw, rm11 = y2 - z2 + w2 - x2, rm12 = dyz + dxw;

double rm20 = dyw + dxz, rm21 = dyz - dxw, rm22 = z2 - y2 - x2 + w2;

double nm00 = this.getMxx() \* rm00 + this.getMyx() \* rm01 + this.getMzx() \* rm02;

double nm01 = this.getMxy() \* rm00 + this.getMyy() \* rm01 + this.getMzy() \* rm02;

double nm02 = this.getMxz() \* rm00 + this.getMyz() \* rm01 + this.getMzz() \* rm02;

double nm10 = this.getMxx() \* rm10 + this.getMyx() \* rm11 + this.getMzx() \* rm12;

double nm11 = this.getMxy() \* rm10 + this.getMyy() \* rm11 + this.getMzy() \* rm12;

double nm12 = this.getMxz() \* rm10 + this.getMyz() \* rm11 + this.getMzz() \* rm12;

setMzx(this.getMxx() \* rm20 + this.getMyx() \* rm21 + this.getMzx() \* rm22);

setMzy(this.getMxy() \* rm20 + this.getMyy() \* rm21 + this.getMzy() \* rm22);

setMzz(this.getMxz() \* rm20 + this.getMyz() \* rm21 + this.getMzz() \* rm22);

setMxx(nm00);

setMxy(nm01);

setMxz(nm02);

setMyx(nm10);

setMyy(nm11);

setMyz(nm12);

return this;

}

}

./src/sample/P2PCylinder.java

package sample;

import javafx.beans.property.DoubleProperty;

import javafx.beans.property.SimpleDoubleProperty;

import javafx.geometry.Point3D;

import javafx.scene.shape.Cylinder;

import javafx.scene.transform.Rotate;

import javafx.scene.transform.Translate;

public class P2PCylinder extends Cylinder {

private double anchorX = 0;

private double anchorY = 0;

private double anchorAngleX = 0;

private double anchorAngleY = 0;

private final DoubleProperty angleX = new SimpleDoubleProperty(0);

private final DoubleProperty angleY = new SimpleDoubleProperty(0);

private Rotate rotateX = new Rotate(0, Rotate.X\_AXIS);

private Rotate rotateY = new Rotate(0, Rotate.Z\_AXIS);

Point3D origin;

Point3D target;

P2PCylinder(){

this(new Point3D(0,0,0),new Point3D(0,0,0));

}

P2PCylinder(Point3D origin, Point3D target) {

super();

this.origin=origin;

this.target=target;

createConnection(origin, target);

}

private void Rotations() {

getTransforms().addAll(rotateX, rotateY);

rotateX.angleProperty().bind(angleX);

rotateY.angleProperty().bind(angleY);

System.out.println("KKFZDD");

setOnMousePressed(event -> {

System.out.println("Cylinder Dragged");

//System.out.println(me);

anchorX = event.getSceneX();

anchorY = event.getSceneY();

anchorAngleX = angleX.get();

anchorAngleY = angleY.get();

});

setOnMouseDragged(event -> {

angleX.set(((anchorAngleX - (anchorY - event.getSceneY())) % 360 + 540) % 360 - 180);

angleY.set(((anchorAngleY + anchorX - event.getSceneX()) % 360 + 540) % 360 - 180);

});

}

public void createConnection(Point3D origin, Point3D target) {

this.getTransforms().removeAll(this.getTransforms());

Point3D yAxis = new Point3D(0, 1, 0);

Point3D diff = target.subtract(origin);

double height = diff.magnitude();

Point3D mid = target.midpoint(origin);

Translate moveToMidpoint = new Translate(mid.getX(), mid.getY(), mid.getZ());

Point3D axisOfRotation = diff.crossProduct(yAxis);

double angle = Math.acos(diff.normalize().dotProduct(yAxis));

Rotate rotateAroundCenter = new Rotate(-Math.toDegrees(angle), axisOfRotation);

//Cylinder line = new Cylinder(1, height);

this.setHeight(height);

this.getTransforms().addAll(moveToMidpoint, rotateAroundCenter);

this.origin=origin;

this.target=target;

}

}

./src/sample/Quaternion.java

package sample;

import javafx.geometry.Point3D;

public class Quaternion {

private static final double EPSILON = 1e-6;

public double getX() {

return x;

}

public double getY() {

return y;

}

public double getZ() {

return z;

}

public double getW() {

return w;

}

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 void setW(double w) {

this.w = w;

}

private double x, y, z, w;

private static double cos(double i) {

return Math.cos(i \* Math.PI / 180);

}

private static double sin(double i) {

return Math.sin(i \* Math.PI / 180);

}

public Quaternion() {

x = 0;

y = 0;

z = 0;

w = 1;

}

public Quaternion(double x, double y, double z, double w) {

this.x = x;

this.y = y;

this.z = z;

this.w = w;

}

public Quaternion set(double x, double y, double z, double w) {

this.x = x;

this.y = y;

this.z = z;

this.w = w;

return this;

}

public Quaternion mul(Quaternion src) {

return this.set(w \* src.x + x \* src.w + y \* src.z - z \* src.y, w \* src.y - x \* src.z + y \* src.w + z \* src.x, w \* src.z + x \* src.y - y \* src.x + z \* src.w, w \* src.w - x \* src.x - y \* src.y - z \* src.z);

}

public double dot(Quaternion src) {

return w \* src.w + x \* src.x + y \* src.y + z \* src.z;

}

public static Quaternion fromRotationBetween(Point3D a, Point3D b) {

a = a.normalize();

b = b.normalize();

double dot = a.dotProduct(b);

double dotError = 1.0 - Math.abs(clamp(dot, -1d, 1f));

Point3D tmp = new Point3D(0, 0, 0);

if (dotError <= EPSILON) {

if (dot < 0.0) {

tmp = new Point3D(1, 0, 0).crossProduct(a);

if (tmp.magnitude() < EPSILON) {

tmp = new Point3D(0, 1, 0).crossProduct(a);

}

tmp.normalize();

if (tmp.magnitude() == 0.0)

return new Quaternion(0, 0, 0, 1);

double angle = 180;

return new Quaternion(tmp.getX() \* sin(angle / 2), tmp.getY() \* sin(angle / 2), tmp.getZ() \* sin(angle / 2), cos(angle / 2));

} else {

return new Quaternion(0, 0, 0, 1);

}

}

tmp = a.crossProduct(b);

return new Quaternion(tmp.getX(), tmp.getY(), tmp.getZ(), 1 + dot);

}

public static double clamp(double value, double min, double max) {

return value < min ? min : value > max ? max : value;

}

public void normalize() {

double norm = (x \* x + y \* y + z \* z + w \* w);

if (norm > 0.0) {

norm = 1.0 / Math.sqrt(norm);

this.x \*= norm;

this.y \*= norm;

this.z \*= norm;

this.w \*= norm;

} else {

this.x = 0.0;

this.y = 0.0;

this.z = 0.0;

this.w = 0.0;

}

}

}

./src/sample/QuaternionGridPane.java

package sample;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import java.io.IOException;

import java.net.URL;

import java.util.ResourceBundle;

public class QuaternionGridPane extends GridPane implements Initializable {

@FXML

private TextField qtX;

@FXML

private TextField qtW;

@FXML

private TextField qtZ;

@FXML

private TextField qtY;

@FXML

private GridPane gridpane;

public QuaternionGridPane() {

super();

try {

FXMLLoader loader = new FXMLLoader(this.getClass().getResource("/QuaternionGridPane.fxml"));

loader.setController(QuaternionGridPane.this);

loader.setRoot(QuaternionGridPane.this);

loader.load();

} catch (IOException ex) {

System.out.println(ex.toString());

}

}

///////////////////////////////////////////////////

////// //////

////// \*Getters\* //////

////// //////

///////////////////////////////////////////////////

public TextField getQtX() {

return qtX;

}

public TextField getQtW() {

return qtW;

}

public TextField getQtZ() {

return qtZ;

}

public TextField getQtY() {

return qtY;

}

@Override

public void initialize(URL location, ResourceBundle resources) {

}

public void setInfo(Quaternion qt) {

qtX.setText(String.valueOf(qt.getX()));

qtY.setText(String.valueOf(qt.getY()));

qtZ.setText(String.valueOf(qt.getZ()));

qtW.setText(String.valueOf(qt.getW()));

}

}

./src/sample/QuaternionSphere.java

package sample;

import javafx.geometry.Point3D;

import javafx.scene.image.Image;

import javafx.scene.paint.PhongMaterial;

import javafx.scene.shape.Sphere;

import javafx.util.Duration;

import java.util.ArrayList;

import java.util.concurrent.atomic.AtomicInteger;

public class QuaternionSphere extends Sphere {

private static final double EPSILON = 1e-6;

private ArrayList<Point3D> points = new ArrayList<>();

private QuaternionTransition qrt;

public Quaternion getState() {

return state;

}

private Quaternion state = new Quaternion();

public QuaternionSphere(double radius) {

super(radius);

PhongMaterial material = new PhongMaterial();

material.setDiffuseMap(new Image("/images/earth.jpeg"));

setMaterial(material);

AtomicInteger clicks = new AtomicInteger();

setOnMousePressed(event -> {

clicks.getAndIncrement();

if(clicks.get()>=3){

points.clear();;

clicks.set(1);

}

points.add(event.getPickResult().getIntersectedPoint());

});

qrt = new QuaternionTransition(this);

//qrt.setInterpolator(Interpolator.TANGENT());

qrt.setOnFinished(event -> {

// Quaternion newState = new Quaternion();

// Point3D startPoint = new Point3D(0, 0, 0);

// if (!points.isEmpty()) {

// startPoint = points.get(0);

// points.remove(0);

// }

// if (points.size() >= 2) {

// qrt.setStart(state);

// newState = Quaternion.fromRotationBetween(points.get(0), points.get(1));

// qrt.setEnd(newState);

// qrt.play();

// }

// state = newState;

});

}

public void setDuration(Duration duration) {

qrt.durationProperty().setValue(duration);

}

public void play() {

qrt.setStart(state);

state = Quaternion.fromRotationBetween(points.get(0),points.get(1));

state.normalize();

//rotateByQuaternion(state);

qrt.setEnd(state);

qrt.play();

}

public void rotateByQuaternion(Quaternion q){

Matrix4x3 mat = new Matrix4x3();

getParent().getTransforms().setAll(mat.rotateGeneric(q));

}

}

./src/sample/QuaternionTransition.java

package sample;

import javafx.animation.Transition;

import javafx.beans.property.ObjectProperty;

import javafx.beans.property.ObjectPropertyBase;

import javafx.util.Duration;

public class QuaternionTransition extends Transition {

QuaternionSphere sphere;

private ObjectProperty<Duration> duration;

private static final Duration DEFAULT\_DURATION = Duration.millis(400);

Quaternion start;

Quaternion end;

QuaternionTransition(QuaternionSphere sphere) {

this.sphere = sphere;

}

@Override

protected void interpolate(double frac) {

double result = start.dot(end);

if (result < 0.0f) {

end.setX(-end.getX());

end.setY(-end.getY());

end.setZ(-end.getZ());

end.setW(-end.getW());

result = -result;

}

double scale0 = 1 - frac;

double scale1 = frac;

if ( (1 - result) >0.0) {

double theta = Math.acos(result);

double invSinTheta = 1 / Math.sin(theta);

scale0 = Math.sin((1 - frac) \* theta) \* invSinTheta;

scale1 = Math.sin((frac \* theta)) \* invSinTheta;

}

double x = (scale0 \* start.getX()) + (scale1 \* end.getX());

double y = (scale0 \* start.getY()) + (scale1 \* end.getY());

double z = (scale0 \* start.getZ()) + (scale1 \* end.getZ());

double w = (scale0 \* start.getW()) + (scale1 \* end.getW());

Quaternion q = new Quaternion(x,y,z,w);

q.normalize();

sphere.rotateByQuaternion(q);

}

public final Duration getDuration() {

return (duration == null) ? DEFAULT\_DURATION : duration.get();

}

public final ObjectProperty<Duration> durationProperty() {

if (duration == null) {

duration = new ObjectPropertyBase<Duration>(DEFAULT\_DURATION) {

@Override

public void invalidated() {

try {

setCycleDuration(getDuration());

} catch (IllegalArgumentException e) {

if (isBound()) {

unbind();

}

set(getCycleDuration());

throw e;

}

}

@Override

public Object getBean() {

return QuaternionTransition.this;

}

@Override

public String getName() {

return "duration";

}

};

}

return duration;

}

public void setStart(Quaternion startQuaternion){

start=startQuaternion;

}

public void setEnd(Quaternion endQuaternion){

end=endQuaternion;

}

}

./src/sample/XYZAxis.java

package sample;

import javafx.scene.Group;

import javafx.scene.paint.Color;

import javafx.scene.paint.Material;

import javafx.scene.paint.PhongMaterial;

import javafx.scene.shape.\*;

import javafx.scene.transform.Rotate;

public class XYZAxis extends Group {

private Cylinder CX = new Cylinder(2, 25);

private Cylinder CY = new Cylinder(2, 25);

private Cylinder CZ = new Cylinder(2, 25);

private Sphere S = new Sphere(4);

private MeshView yCone;

private MeshView xCone;

private MeshView zCone;

public XYZAxis() {

super();

setMaterials();

positioning();

getChildren().addAll(CX, CY, CZ, S);

getChildren().addAll(xCone, yCone, zCone);

}

private void setMaterials() {

Material mat = new PhongMaterial(Color.WHITE);

PhongMaterial Xmat = new PhongMaterial();

Xmat.setDiffuseColor(Color.RED);

PhongMaterial Ymat = new PhongMaterial();

Ymat.setDiffuseColor(Color.GREEN);

PhongMaterial Zmat = new PhongMaterial();

Zmat.setDiffuseColor(Color.BLUE);

S.setMaterial(Zmat);

CY.setMaterial(mat);

CX.setMaterial(mat);

CZ.setMaterial(mat);

TriangleMesh coneMeshY = createCone(3.5f, 7.5f);

TriangleMesh coneMeshX = createCone(3.5f, 7.5f);

TriangleMesh coneMeshZ = createCone(3.5f, 7.5f);

yCone = new MeshView(coneMeshY);

xCone = new MeshView(coneMeshX);

zCone = new MeshView(coneMeshZ);

yCone.setMaterial(Ymat);

xCone.setMaterial(Xmat);

zCone.setMaterial(Zmat);

yCone.setDrawMode(DrawMode.FILL);

xCone.setDrawMode(DrawMode.FILL);

zCone.setDrawMode(DrawMode.FILL);

}

private void positioning() {

CY.setTranslateY(-12.5);

CX.setTranslateX(15);

CX.setRotationAxis(Rotate.Z\_AXIS);

CX.setRotate(90);

CZ.setRotationAxis(Rotate.X\_AXIS);

CZ.setRotate(90);

CZ.setTranslateZ(-12.5);

yCone.setTranslateY(-32.5);

xCone.setTranslateY(-3.75);

xCone.setRotationAxis(Rotate.Z\_AXIS);

xCone.setRotate(90);

xCone.setTranslateX(28.5);

zCone.setRotationAxis(Rotate.X\_AXIS);

zCone.setTranslateY(-3.75);

zCone.setRotate(90);

zCone.setTranslateZ(-28.5);

}

private TriangleMesh createCone(float radius, float height) {

int divisions = 500;

TriangleMesh mesh = new TriangleMesh();

mesh.getPoints().addAll(0, 0, 0);

double segment\_angle = 2.0 \* Math.PI / divisions;

float x, z;

double angle;

double halfCount = (Math.PI / 2 - Math.PI / (divisions / 2));

for (int i = divisions + 1; --i >= 0; ) {

angle = segment\_angle \* i;

x = (float) (radius \* Math.cos(angle - halfCount));

z = (float) (radius \* Math.sin(angle - halfCount));

mesh.getPoints().addAll(x, height, z);

}

mesh.getPoints().addAll(0, height, 0);

mesh.getTexCoords().addAll(0, 0);

for (int i = 1; i <= divisions; i++) {

mesh.getFaces().addAll(

0, 0, i + 1, 0, i, 0, //COunter clock wise

divisions + 2, 0, i, 0, i + 1, 0 // Clock wise

);

}

return mesh;

}

}

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |