

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 6**

з дисципліни “ Математичні та алгоритмічні основи комп’ютерної графіки”

|  |  |
| --- | --- |
| Виконав  студент III курсу  групи КП-73  Булаєвський Ігор Олегович  (*прізвище, ім’я, по батькові*)  Варіант №3 |  |

Київ - 2020Лістинг програми

**Animation.java**

**import java.applet.Applet;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.util.Hashtable;**

**import javax.media.j3d.\*;**

**import javax.swing.\*;**

**import javax.vecmath.\*;**

**import com.sun.j3d.loaders.Scene;**

**import com.sun.j3d.loaders.objectfile.ObjectFile;**

**import com.sun.j3d.utils.applet.MainFrame;**

**import com.sun.j3d.utils.behaviors.vp.OrbitBehavior;**

**import com.sun.j3d.utils.image.TextureLoader;**

**import com.sun.j3d.utils.universe.SimpleUniverse;**

**public class Animation extends Applet implements ActionListener {**

**Transform3D trans3d;**

**TransformGroup transGroup, spiderGroup = new TransformGroup();**

**double angle = 0;**

**public BranchGroup createSceneGraph() {**

**BranchGroup group = new BranchGroup();**

**transGroup = new TransformGroup();**

**transGroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);**

**trans3d = new Transform3D();**

**transGroup.setTransform(trans3d);**

**group.addChild(transGroup);**

**BoundingSphere bound = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);**

**TextureLoader t = new TextureLoader("textures/web.jpg", this);**

**Background bg = new Background(t.getImage());**

**bg.setImageScaleMode(Background.SCALE\_FIT\_ALL);**

**bg.setApplicationBounds(bound);**

**group.addChild(bg);**

**Color3f lightColor = new Color3f(1.0f, 1.0f, 0.9f);**

**Vector3f lightDirection = new Vector3f(4.0f, -7.0f, -12.0f);**

**DirectionalLight light = new DirectionalLight(lightColor, lightDirection);**

**light.setInfluencingBounds(bound);**

**group.addChild(light);**

**spiderGroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);**

**AmbientLight ambientLight = new AmbientLight(lightColor);**

**ambientLight.setInfluencingBounds(bound);**

**group.addChild(ambientLight);**

**Hashtable<String, Shape3D> map = getHashMap();**

**Appearance legApp = loadTexture("textures/bug.jpg");**

**for (int i = 1; i <= 8; i++) {**

**float start = i % 2 == 0 ? 0f : (float) Math.PI / 4;**

**float end = i % 2 != 0 ? 0f : (float) Math.PI / 4;**

**Shape3D leg = map.get("leg" + i);**

**leg.setAppearance(legApp);**

**TransformGroup tg = createLegTransformation(leg, start, end);**

**spiderGroup.addChild(tg);**

**}**

**Shape3D body = map.get("blkw\_body");**

**body.setAppearance(loadTexture("textures/skin.jpg"));**

**spiderGroup.addChild(body);**

**transGroup.addChild(spiderGroup);**

**group.compile();**

**return group;**

**}**

**TransformGroup createLegTransformation(Shape3D leg, float start, float end) {**

**TransformGroup transformGroup = new TransformGroup();**

**transformGroup.addChild(leg);**

**Transform3D rotationAxis = new Transform3D();**

**rotationAxis.rotZ(Math.PI / 2);**

**int timeRotation = 500;**

**Alpha rotationAlpha = new Alpha();**

**rotationAlpha.setLoopCount(Integer.MAX\_VALUE);**

**rotationAlpha.setMode(Alpha.DECREASING\_ENABLE | Alpha.INCREASING\_ENABLE);**

**rotationAlpha.setIncreasingAlphaDuration(timeRotation);**

**rotationAlpha.setDecreasingAlphaDuration(timeRotation);**

**RotationInterpolator rotation = new RotationInterpolator(**

**rotationAlpha, transformGroup,**

**rotationAxis, start, end); //опис руху стрілки**

**rotation.setSchedulingBounds(new BoundingSphere(new Point3d(0.0, 0.0, 0.0), Double.MAX\_VALUE));**

**//дозволяємо трансформації об’єкту у групі**

**transformGroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);**

**transformGroup.addChild(rotation);**

**return transformGroup;**

**}**

**public Animation() {**

**setLayout(new BorderLayout());**

**GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();**

**Canvas3D canvas = new Canvas3D(config);**

**add("Center", canvas);**

**BranchGroup scene = createSceneGraph();**

**SimpleUniverse universe = new SimpleUniverse(canvas);**

**universe.getViewingPlatform().setNominalViewingTransform();**

**universe.addBranchGraph(scene);**

**OrbitBehavior ob = new OrbitBehavior(canvas);**

**ob.setRotFactors(-1, -1);**

**ob.setSchedulingBounds(new BoundingSphere(new Point3d(0.0, 0.0, 0.0), Double.MAX\_VALUE));**

**universe.getViewingPlatform().setViewPlatformBehavior(ob);**

**Timer timer = new Timer(10, this);**

**timer.start();**

**}**

**public static void main(String[] args) {**

**new MainFrame(new Animation(), 800, 600);**

**}**

**Appearance loadTexture(String path) {**

**Texture tex = new TextureLoader(path, this).getTexture();**

**tex.setBoundaryModeS(Texture.WRAP);**

**tex.setBoundaryModeT(Texture.WRAP);**

**TextureAttributes texAttr = new TextureAttributes();**

**texAttr.setTextureMode(TextureAttributes.COMBINE);**

**Appearance ap = new Appearance();**

**ap.setTexture(tex);**

**ap.setTextureAttributes(texAttr);**

**Material material = new Material();**

**material.setSpecularColor(new Color3f(Color.WHITE));**

**material.setDiffuseColor(new Color3f(Color.WHITE));**

**ap.setMaterial(material);**

**return ap;**

**}**

**Hashtable<String, Shape3D> getHashMap() {**

**int flags = ObjectFile.RESIZE;**

**double creaseAngle = 60.0;**

**ObjectFile objFile = new ObjectFile(flags, (float) (creaseAngle \* Math.PI) / 180);**

**Scene scene = null;**

**try {**

**scene = objFile.load("models/black\_widow.obj");**

**} catch (Exception e) {**

**e.printStackTrace();**

**System.out.println("OBJ load Err：" + e.getMessage());**

**}**

**Hashtable<String, Shape3D> map = scene.getNamedObjects();**

**map.replaceAll((k, v) -> (Shape3D) map.get(k).cloneTree());**

**return map;**

**}**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**Transform3D transform3D = new Transform3D();**

**transform3D.setScale(0.5);**

**transform3D.setTranslation(new Vector3f(0.5f, 0f, 0f));**

**Transform3D rotation = new Transform3D();**

**rotation.rotZ(angle);**

**angle += 0.01;**

**Transform3D back = new Transform3D();**

**back.setTranslation(new Vector3f(-0.5f, 0f, 0f));**

**Transform3D flipY = new Transform3D();**

**flipY.rotY(Math.PI / 2);**

**Transform3D flipX = new Transform3D();**

**flipX.rotX(Math.PI / 2);**

**transform3D.mul(rotation);**

**transform3D.mul(flipX);**

**transform3D.mul(flipY);**

**transform3D.mul(back);**

**spiderGroup.setTransform(transform3D);**

**}**

**}**

Скріншоти результатів

