МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №6

з дисципліни “Математичні та алгоритмічні основи комп’ютерної графіки”
тема: “Анімація тривимірних об’єктів”

Виконав
студент III курсу
групи КП-83
Ландо Максим Юрійович
варіант № 9

Зарахована
“____” “___________” 20___ р.
викладачем
Шкурат Оксаною Сергіївною

Київ 2021

Мета: Навчитися анімувати складні об’єкти тривимірної сцени.

Задання на лабораторну роботу

Виконати анімацію тривимірної сцени за варіантом.

Варіант: Клещ

Лістинг коду програми

Mite.java

```
import com.sun.j3d.loaders.Scene;
import com.sun.j3d.loaders.objectfile.ObjectFile;
import com.sun.j3d.utils.geometry.Box;
import com.sun.j3d.utils.image.TextureLoader;

import javax.media.j3d.*;
import javax.vecmath.Color3f;
import java.awt.*;
import java.io.FileNotFoundException;
import java.util.Map;

public class Mite {
    private TransformUtility body;

    private TransformUtility leftLeg1;
    private TransformUtility leftLeg2;
    private TransformUtility leftLeg3;

    private TransformUtility rightLeg1;
    private TransformUtility rightLeg2;
    private TransformUtility rightLeg3;
    private TransformUtility ant1;
    private TransformUtility ant2;
    private TransformUtility mainMove1;
    private TransformUtility mainMove2;
    private TransformUtility mainModel1;
    private TransformUtility mainModel2;

    public Mite(Canvas canvas) throws FileNotFoundException {
        TransformUtility[] transfoms = loadObject("mite", "leg6", "leg5", "leg4", "leg3", "leg2", "leg1
", "antenna2", "antenna");

        body = transfoms[0];
        leftLeg1 = transfoms[1];
        leftLeg2 = transfoms[2];
        leftLeg3 = transfoms[3];
        rightLeg1 = transfoms[4];
        rightLeg2 = transfoms[5];
        rightLeg3 = transfoms[6];
        ant1 = transfoms[7];
        ant2 = transfoms[8];
        Material material = new Material();
        material.setAmbientColor (new Color3f(1, 1, 1));
        material.setDiffuseColor (new Color3f(1f, 1f, 1f));
        material.setSpecularColor(new Color3f(0.1f, 0.1f, 0.1f));
        material.setShininess(1f);
        material.setLightingEnable(true);

        TextureAttributes texAttr = new TextureAttributes();
        texAttr.setTextureMode(TextureAttributes.COMBINE);

        TextureLoader textureLoader = new TextureLoader("ground.jpg", "RGB", canvas);
        Appearance ap = new Appearance();
        ap.setTexCoordGeneration(new TexCoordGeneration(
                TexCoordGeneration.OBJECT_LINEAR, TexCoordGeneration.TEXTURE_COORDINATE_2));
        ap.setMaterial(material);
        ap.setTextureAttributes(texAttr);
        ap.setTexture(textureLoader.getTexture());

        TransformUtility ground = new TransformUtility(new Box(1000, 1000, 0.1f, ap));
        ground.translate(0, 0, -0.1);

        mainMove1 = new TransformUtility(body.asNode(), leftLeg1.asNode(), leftLeg2.asNode(), leftLeg3.
asNode(),
                rightLeg1.asNode(), rightLeg2.asNode(), rightLeg3.asNode(), ant1.asNode(), ant2.asNode(
));
        mainMove2 = new TransformUtility(mainMove1.asNode());
        mainModel1 = new TransformUtility(mainMove2.asNode(), ground.asNode());
        mainModel2 = new TransformUtility(mainModel1.asNode());
        mainMove1.rotate(Math.PI/2.1,0,0);
        rotateModel(-Math.PI/1.2, Math.PI, 0);
    }
    private static TransformUtility[] loadObject(String... groupNames) throws FileNotFoundException {
        Scene scene = new ObjectFile(ObjectFile.RESIZE/2).load("Mite.obj");
        BranchGroup root = scene.getSceneGroup();

        Map<String, Shape3D> nameMap = scene.getNamedObjects();

        root.removeAllChildren();

        TransformUtility[] ret = new TransformUtility[groupNames.length];

        for (int i = 0; i < groupNames.length; ++i) {
            ret[i] = new TransformUtility(nameMap.get(groupNames[i]));
        }

        return ret;
    }

    double legRotateDX = 0.02, bodyRotateDy = 0.005;

    public void update(boolean isUp) {
        leftLeg1.rotate(legRotateDX, legRotateDX, 0);
        rightLeg2.rotate(legRotateDX, legRotateDX, 0);
        leftLeg3.rotate(legRotateDX, legRotateDX, 0);
        rightLeg1.rotate(-legRotateDX, -legRotateDX, 0);
        leftLeg2.rotate(-legRotateDX, -legRotateDX, 0);
        rightLeg3.rotate(-legRotateDX, -legRotateDX, 0);
        if(Math.abs(leftLeg1.rotX) > 0.1) {
            legRotateDX *= -1;
        }

        body.rotate(0, bodyRotateDy, 0);

        if(Math.abs(body.rotY) > 0.05) {
            bodyRotateDy *= -1;
        }

        double speed = 0.01;
        if(isUp){
            mainMove2.translate(speed * Math.sin(mainMove1.rotZ), -
speed * Math.cos(mainMove1.rotZ), 0);
        } else {
            mainMove2.translate(-
speed * Math.sin(mainMove1.rotZ), speed * Math.cos(mainMove1.rotZ), 0);
        }

    }

    public void rotateModel(double rotX, double rotY, double rotZ) {
        mainModel1.rotate(rotX, 0, 0);
        mainModel2.rotate(0, rotY, 0);
    }

    public Node asNode() {
        return mainModel2.asNode();
    }
}
```

SceneControl.java

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

import javax.media.j3d.*;
import javax.swing.*;
import javax.vecmath.Color3f;
import javax.vecmath.Vector3f;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.IOException;

public class SceneControl implements KeyListener, ActionListener {
    private SimpleUniverse universe;
    private BranchGroup root;

    private Canvas3D canvas;

    private Mite object;

    public SceneControl() throws IOException {
        initCanvas();
        initUniverse();
        Bounds influenceRegion = new BoundingSphere();

        object = new Mite(canvas);
        root = new BranchGroup();

        root.addChild(object.asNode());

        addLightsToUniverse(influenceRegion);
        addBackground(influenceRegion);
        root.compile();
        universe.addBranchGraph(root);

        new Timer(10, this).start();
    }

    private void initCanvas() {
        canvas = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
        canvas.setDoubleBufferEnable(true);
        canvas.setFocusable(true);

        canvas.addKeyListener(this);
    }

    private void initUniverse() {
        universe = new SimpleUniverse(canvas);
        universe.getViewingPlatform().setNominalViewingTransform();
    }

    private void addLightsToUniverse(Bounds influenceRegion) {
        Color3f lightColor = new Color3f(Color.WHITE);
        Vector3f lightDirection = new Vector3f(0F, -1F, -1F);
        DirectionalLight light = new DirectionalLight(lightColor, lightDirection);
        light.setInfluencingBounds(influenceRegion);
        root.addChild(light);
    }

    private void addBackground(Bounds influenceRegion) {
        Background background = new Background(new Color3f(Color.CYAN));
        background.setApplicationBounds(influenceRegion);
        root.addChild(background);
    }

    public Canvas3D getCanvas() {
        return canvas;
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(keyForward) {
            object.update(true);
        }
        if(keyDown){
            object.update(false);
        }
        double rotateX = (keyViewUp ? 1 : 0) - (keyViewDown ? 1 : 0);
        double rotateY = (keyViewLeft ? 1 : 0) - (keyViewRight ? 1 : 0);

        object.rotateModel(rotateX*0.05, rotateY*0.05, 0);
    }

    private boolean keyDown, keyForward, keyViewLeft, keyViewRight, keyViewUp, keyViewDown;

    @Override
    public void keyTyped(KeyEvent e) {}

    @Override
    public void keyPressed(KeyEvent e) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_W: keyForward = true; break;
            case KeyEvent.VK_S: keyDown = true; break;

            case KeyEvent.VK_LEFT: keyViewLeft = true; break;
            case KeyEvent.VK_RIGHT: keyViewRight = true; break;
            case KeyEvent.VK_UP: keyViewUp = true; break;
            case KeyEvent.VK_DOWN: keyViewDown = true; break;
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_W: keyForward = false; break;
            case KeyEvent.VK_S: keyDown = false; break;

            case KeyEvent.VK_LEFT: keyViewLeft = false; break;
            case KeyEvent.VK_RIGHT: keyViewRight = false; break;
            case KeyEvent.VK_UP: keyViewUp = false; break;
            case KeyEvent.VK_DOWN: keyViewDown = false; break;
        }
    }
}
```

TransformUtility.java

```
import javax.media.j3d.Node;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import javax.vecmath.Vector3d;

public class TransformUtility {
    private TransformGroup translationGroup = new TransformGroup();
    private TransformGroup rotationGroup = new TransformGroup();

    private Transform3D translationTransform = new Transform3D();
    private Transform3D rotationTransform = new Transform3D();

    double x, y, z, rotX, rotY, rotZ;

    public TransformUtility(Node... objects) {
        for(Node n: objects) {
            translationGroup.addChild(n);
        }

        translationGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        translationGroup.setTransform(translationTransform);

        rotationGroup.addChild(translationGroup);
        rotationGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        rotationGroup.setTransform(rotationTransform);
    }

    public void translate(double x, double y, double z) {
        this.x += x;
        this.y += y;
        this.z += z;

        translationTransform.setTranslation(new Vector3d(this.x, this.y, this.z));
        translationGroup.setTransform(translationTransform);
    }

    public void rotate(double rotX, double rotY, double rotZ) {
        this.rotX += rotX;
        this.rotY += rotY;
        this.rotZ += rotZ;

        if(this.rotX != 0) rotationTransform.rotX(this.rotX);
        if(this.rotY != 0) rotationTransform.rotY(this.rotY);
        if(this.rotZ != 0) rotationTransform.rotZ(this.rotZ);

        rotationGroup.setTransform(rotationTransform);
    }

    public Node asNode() {
        return rotationGroup;
    }
}
```

Результат