

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“

|  |  |  |  |
| --- | --- | --- | --- |
| Виконала  студентка III курсу  групи КП-83  Дзьоба Уляна Олегівна  (*прізвище, ім’я, по батькові*)  варіант № 5 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

**Завдання**:

Виконати анімацію тривимірної сцени за варіантом.

**Варіант:**

Анімація вертольоту helicopter.obj. У вертольота повинні рухатися обидва гвинти, вертоліт повинен пересуватися по екрану.

**Лістинг коду програми**

**Main.java**

package lab6;

import javax.vecmath.\*;

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

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

import javax.media.j3d.\*;

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

import javax.swing.JFrame;

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

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.net.URL;

import java.util.Hashtable;

import java.util.Enumeration;

public class Chopper extends JFrame{

public Canvas3D myCanvas3D;

public Chopper() throws IOException {

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

myCanvas3D = new Canvas3D(SimpleUniverse.getPreferredConfiguration());

SimpleUniverse simpUniv = new SimpleUniverse(myCanvas3D);

simpUniv.getViewingPlatform().setNominalViewingTransform();

createSceneGraph(simpUniv);

addLight(simpUniv);

OrbitBehavior ob = new OrbitBehavior(myCanvas3D);

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

simpUniv.getViewingPlatform().setViewPlatformBehavior(ob);

setTitle("chopper");

setSize(700,700);

getContentPane().add("Center", myCanvas3D);

setVisible(true);

}

public void createSceneGraph(SimpleUniverse su) throws IOException {

ObjectFile f = new ObjectFile(ObjectFile.RESIZE);

BoundingSphere bs = new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.MAX\_VALUE);

String name;

BranchGroup chopperBranchGroup = new BranchGroup();

ClassLoader classLoader = Thread.currentThread().getContextClassLoader();

InputStream inputStream = classLoader.getResourceAsStream("helicopter.obj");

Scene chopperScene = f.load(new BufferedReader(new InputStreamReader(inputStream)));

Hashtable roachNamedObjects = chopperScene.getNamedObjects();

Enumeration enumer = roachNamedObjects.keys();

while (enumer.hasMoreElements()){

name = (String) enumer.nextElement();

System.out.println("Name: " + name);

}

// start animation

Transform3D startTransformation = new Transform3D();

startTransformation.setScale(2.0/6);

Transform3D combinedStartTransformation = new Transform3D();

combinedStartTransformation.mul(startTransformation);

TransformGroup chopperStartTransformGroup = new TransformGroup(combinedStartTransformation);

//

Shape3D bp = (Shape3D) roachNamedObjects.get("big\_propeller");

Appearance bpApp = new Appearance();

setToMyDefaultAppearance(bpApp, new Color3f(0.0f, 0.0f, 0.0f));

bp.setAppearance(bpApp);

TransformGroup bpGroup = new TransformGroup();

bpGroup.addChild(bp.cloneTree());

Transform3D bpRotAxis = new Transform3D();

bpRotAxis.set(new Vector3d(0.0, 0.0, -0.2));

RotationInterpolator bpRot = new RotationInterpolator(new Alpha(-1,1000), bpGroup, bpRotAxis, 0.0f,(float) Math.PI\*2);

bpRot.setSchedulingBounds(bs);

bpGroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

bpGroup.addChild(bpRot);

//

Shape3D sp = (Shape3D) roachNamedObjects.get("small\_propeller");

sp.setAppearance(bpApp);

TransformGroup spGroup = new TransformGroup();

spGroup.addChild(sp.cloneTree());

Transform3D spRotAxis1 = new Transform3D();

Transform3D spRotAxis2 = new Transform3D();

spRotAxis1.rotZ(Math.PI / 2);

spRotAxis2.set(new Vector3d(0.05, 0.1, 0.83));

spRotAxis1.mul(spRotAxis2);

RotationInterpolator spRot = new RotationInterpolator(new Alpha(-1,1000), spGroup, spRotAxis1, 0.0f,(float) Math.PI\*2);

spRot.setSchedulingBounds(bs);

spGroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

spGroup.addChild(spRot);

TransformGroup sceneGroup = new TransformGroup();

sceneGroup.addChild(bpGroup);

sceneGroup.addChild(spGroup);

Shape3D d = (Shape3D) roachNamedObjects.get("decal");

sceneGroup.addChild(d.cloneTree());

Shape3D a = (Shape3D) roachNamedObjects.get("alpha");

sceneGroup.addChild(a.cloneTree());

Shape3D m1 = (Shape3D) roachNamedObjects.get("missile\_1");

m1.setAppearance(bpApp);

sceneGroup.addChild(m1.cloneTree());

Shape3D mg = (Shape3D) roachNamedObjects.get("missile\_gl");

sceneGroup.addChild(mg.cloneTree());

Shape3D m = (Shape3D) roachNamedObjects.get("main\_");

sceneGroup.addChild(m.cloneTree());

Appearance mbApp = new Appearance();

setToMyDefaultAppearance(mbApp, new Color3f(0.5f, 0.5f, 0.0f));

Shape3D mb = (Shape3D) roachNamedObjects.get("main\_body\_");

mb.setAppearance(mbApp);

sceneGroup.addChild(mb.cloneTree());

Appearance gApp = new Appearance();

setToMyDefaultAppearance(gApp, new Color3f(0.4f, 0.7f, 0.8f));

Shape3D g = (Shape3D) roachNamedObjects.get("glass");

g.setAppearance(gApp);

sceneGroup.addChild(g.cloneTree());

TransformGroup setRadius = translate(

chopperStartTransformGroup,

new Vector3f(0.7f,0.0f,0.0f));

TransformGroup rotateGroup = rotate(setRadius, new Alpha(-1,5000));

chopperBranchGroup.addChild(rotateGroup);

chopperStartTransformGroup.addChild(sceneGroup);

Background background = new Background(getTextureLoader("bg.jpg").getImage());

background.setImageScaleMode(Background.SCALE\_FIT\_MAX);

background.setApplicationBounds(new BoundingSphere(new Point3d(),1000));

background.setCapability(Background.ALLOW\_IMAGE\_WRITE);

BoundingSphere bounds = new BoundingSphere(new Point3d(120.0,250.0,100.0),Double.MAX\_VALUE);

chopperBranchGroup.addChild(background);

chopperBranchGroup.compile();

su.addBranchGraph(chopperBranchGroup);

}

private TextureLoader getTextureLoader(String path) throws IOException {

ClassLoader classLoader = Thread.currentThread().getContextClassLoader();

URL textureResource = classLoader.getResource(path);

if (textureResource == null) {

throw new IOException("Couldn't find texture: " + path);

}

return new TextureLoader(textureResource.getPath(), myCanvas3D);

}

public void addLight(SimpleUniverse su){

BranchGroup bgLight = new BranchGroup();

BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);

Color3f lightColour1 = new Color3f(1.0f,1.0f,1.0f);

Vector3f lightDir1 = new Vector3f(-1.0f,0.0f,-0.5f);

DirectionalLight light1 = new DirectionalLight(lightColour1, lightDir1);

light1.setInfluencingBounds(bounds);

bgLight.addChild(light1);

su.addBranchGraph(bgLight);

}

private TransformGroup translate(Node node, Vector3f vector){

Transform3D transform3D = new Transform3D();

transform3D.setTranslation(vector);

TransformGroup transformGroup =

new TransformGroup();

transformGroup.setTransform(transform3D);

transformGroup.addChild(node);

return transformGroup;

}

private TransformGroup rotate(Node node, Alpha alpha){

TransformGroup xformGroup = new TransformGroup();

xformGroup.setCapability(

TransformGroup.ALLOW\_TRANSFORM\_WRITE);

RotationInterpolator interpolator =

new RotationInterpolator(alpha,xformGroup);

interpolator.setSchedulingBounds(new BoundingSphere(

new Point3d(0.0,0.0,0.0),1.0));

xformGroup.addChild(interpolator);

xformGroup.addChild(node);

return xformGroup;

}

public static void setToMyDefaultAppearance(Appearance app, Color3f col) {

app.setMaterial(new Material(col, col, col, col, 150.0f));

}

public static void main(String[] args) throws IOException {

Chopper start = new Chopper();

}

}

**Результат**

****