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”

Виконав
студент III курсу
групи КП-83
Мричко Богдан
(*прізвище, ім’я, по батькові*)

Зарахована
“____” “_____” 20___ р.
викладачем
Шкурат Оксаною Сергіївною
(*прізвище, ім’я, по батькові*)

варіант № 15

Київ 2020

Варіант завдання

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

Код програми

Main.java

```
package sample;

import javax.vecmath.*;

import com.sun.j3d.utils.behaviors.vp.OrbitBehavior;
import com.sun.j3d.utils.image.TextureLoader;
import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import javax.swing.JFrame;
import com.sun.j3d.loaders.*;
import com.sun.j3d.loaders.objectfile.*;

import java.awt.*;
import java.util.Enumeration;
import java.util.Hashtable;

public class Main extends JFrame{
   private static Canvas3D canvas;
   private static SimpleUniverse universe;
   private static BranchGroup root;
   private static TransformGroup helicopterTransformGroup;
   private static Scene helicopterScene;

   private static String assetsDir = System.getProperty("user.dir") +
"\\src\\resources\\";

   // moves
   private static int movesCount = 999;
   private static int movesDuration = 700;
   private static int startTime = 0;

   public Main(){
       configureWindow();
       configureCanvas();
       configureUniverse();
       root = new BranchGroup();
       addImageBackground();
       addLight();
       helicopterTransformGroup = getHelicopterGroup();
       root.addChild(helicopterTransformGroup);
       root.compile();
       universe.addBranchGraph(root);
   }

   private void configureWindow() {
       setTitle("Lab6");
       setExtendedState(JFrame.MAXIMIZED_BOTH);
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }

   private void configureCanvas() {
       canvas = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
       canvas.setDoubleBufferEnable(true);
       getContentPane().add(canvas, BorderLayout.CENTER);
   }
```

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

    private void addImageBackground() {
        TextureLoader t = new TextureLoader(assetsDir + "space.jpg", canvas);
        Background background = new Background(t.getImage());
        background.setImageScaleMode(Background.SCALE_FIT_ALL);
        BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0,
0.0), 100.0);
        background.setApplicationBounds(bounds);
        root.addChild(background);
    }

    public void addLight(){
        BoundingSphere bounds = new BoundingSphere();
        bounds.setRadius(100);
        DirectionalLight directionalLight = new DirectionalLight(new
Color3f(1, 1, 1), new Vector3f(-1, -1, -1));
        directionalLight.setInfluencingBounds(bounds);
        root.addChild(directionalLight);

        AmbientLight ambientLight = new AmbientLight(new Color3f(1, 1, 1));
        ambientLight.setInfluencingBounds(new BoundingSphere());
        root.addChild(ambientLight);
    }

    private void addAppearance(Shape3D shape, String path) {
        TextureLoader loader = new TextureLoader(path, "RGP", new
Container());
        Texture texture = loader.getTexture();
        Appearance appearance = new Appearance();
        appearance.setTexture(texture);
        shape.setAppearance(appearance);
    }

    public TransformGroup getHelicopterGroup(){
        BoundingSphere boundingSphere = new BoundingSphere(new
Point3d(0.0,0.0,0.0),Double.MAX_VALUE);
        ObjectFile loader = new ObjectFile(ObjectFile.RESIZE);
        try {
            helicopterScene = loader.load(assetsDir + "helicopter.obj");
        }
        catch (Exception e){
            System.out.println("File loading failed:" + e);
        }

        TransformGroup bodyTG = new TransformGroup();
        TransformGroup smallPropellerTG = new TransformGroup();
        smallPropellerTG.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        TransformGroup detailsTG = new TransformGroup();
        TransformGroup bigPropellerTG = new TransformGroup();
        bigPropellerTG.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

        Appearance bidPropellerAppearance = new Appearance();
        combineWithGlobalAppearance(bidPropellerAppearance, new Color3f(.99f,
.0f, .0f));
        Appearance smallPropellerAppearance = new Appearance();
        combineWithGlobalAppearance(smallPropellerAppearance, new
Color3f(.0f, .0f, .99f));
        Appearance detailsAppearance = new Appearance();
        combineWithGlobalAppearance(detailsAppearance, new Color3f(.43f,
```

```
.29f, .12f));

        Hashtable helicopter = helicopterScene.getNamedObjects();

        // main_body_
        Shape3D body = (Shape3D) helicopter.get("main_body_");
        addAppearance(body, assetsDir+"body.jpg");
        bodyTG.addChild(body.cloneTree());

        // small_propeller
        Alpha smallPropellerAlpha = new Alpha(movesCount,
Alpha.INCREASING_ENABLE, startTime, 0, movesDuration,0,0,0,0,0);
        Transform3D inclineY = new Transform3D();
        inclineY.set(new Vector3d(0, 0.02, 0.86));
        inclineY.setRotation(new AxisAngle4d(0, 0, 1, Math.PI/2));
        //RotationInterpolator ballInterpolator = new
RotationInterpolator(smallPropellerAlpha, smallPropellerTG, new
Transform3D(), 0.0f, (float) Math.PI * 2);
        RotationInterpolator ballInterpolator = new
RotationInterpolator(smallPropellerAlpha, smallPropellerTG, inclineY, 0.0f,
(float) Math.PI * 2);
        ballInterpolator.setSchedulingBounds(boundingSphere);

        Shape3D smallPropeller = (Shape3D) helicopter.get("small_propeller");
        smallPropeller.setAppearance(smallPropellerAppearance);
        smallPropellerTG.addChild(smallPropeller.cloneTree());
        smallPropellerTG.addChild(ballInterpolator);

        // big_propeller
        Alpha bigPropellerAlpha = new Alpha(movesCount,
Alpha.INCREASING_ENABLE, startTime, 0, movesDuration,0,0,0,0,0);
        Transform3D inclineX = new Transform3D();
        inclineX.set(new Vector3d(0, 1, -0.21));
        inclineX.setRotation(new AxisAngle4d(0, 1, 0, Math.PI/2));
        RotationInterpolator bigPropellerInterpolator = new
RotationInterpolator(bigPropellerAlpha, bigPropellerTG, inclineX, 0.0f,
(float) Math.PI*2);
        bigPropellerInterpolator.setSchedulingBounds(boundingSphere);

        Shape3D bigPropeller = (Shape3D) helicopter.get("big_propeller");
        bigPropeller.setAppearance(bidPropellerAppearance);
        bigPropellerTG.addChild(bigPropeller.cloneTree());
        bigPropellerTG.addChild(bigPropellerInterpolator);

        // details
        Shape3D details = (Shape3D) helicopter.get("main_");
        details.setAppearance(detailsAppearance);
        detailsTG.addChild(details.cloneTree());

        Transform3D startTransformation = new Transform3D();
        startTransformation.setScale(0.3);

        TransformGroup transformGroup = new
TransformGroup(startTransformation);
        TransformGroup sceneGroup = new TransformGroup();

        sceneGroup.addChild(detailsTG);
        sceneGroup.addChild(bodyTG);
        sceneGroup.addChild(bigPropellerTG);
        sceneGroup.addChild(smallPropellerTG);

        transformGroup.addChild(sceneGroup);
        return rotate(
```

```
                translate(transformGroup, new Vector3f(0.0f,0.0f,0.06f)),
                new Alpha(3,3000)
        );
    }

    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(node);
        return xformGroup;
    }

    public static void combineWithGlobalAppearance(Appearance app, Color3f
col) {
        app.setMaterial(new Material(col, col, col, col, 150.0f));
    }

    public static void main(String[] args) {
        try {
            Main window = new Main();
            sample.Controller helicopterMovement = new
sample.Controller(helicopterTransformGroup);
            canvas.addKeyListener(helicopterMovement);
            window.setVisible(true);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

}
```

Animation.java

```
package sample;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.media.j3d.*;
import javax.swing.Timer;
import javax.vecmath.*;

public class Controller implements ActionListener, KeyListener {
    private TransformGroup bear;
    private Transform3D transform3D = new Transform3D();
```

```
private float x = 0;
private float y = 0;

private boolean w = false;
private boolean s = false;
private boolean a = false;
private boolean d = false;
private boolean e = false;
private boolean q = false;

private boolean z = false;
private boolean _x = false;

Controller(TransformGroup bear) {
    this.bear = bear;
    this.bear.getTransform(this.transform3D);
    Timer timer = new Timer(20, this);
    timer.start();
}

private void Move() {
    if (w) {
        y += 0.02f;
        if (y > 0.2f) y = 0.2f;
    }
    if (s) {
        y -= 0.02f;
        if (y < -0.3f) y = -0.3f;
    }
    if (a) {
        x -= 0.02f;
        if (x < -0.8f) x = -0.8f;
    }
    if (d) {
        x += 0.02f;
        if (x > 0.8f) x = 0.8f;
    }
    transform3D.setTranslation(new Vector3f(x, y, 0));
    if (e) {
        Transform3D rotation = new Transform3D();
        rotation.rotY(0.05f);
        transform3D.mul(rotation);
    }
    if (q) {

        Transform3D rotation = new Transform3D();
        rotation.rotY(-0.05f);
        transform3D.mul(rotation);
    }
    if (z) {
        Transform3D rotation = new Transform3D();
        rotation.rotX(0.05f);
        transform3D.mul(rotation);
    }
    if (_x) {

        Transform3D rotation = new Transform3D();
        rotation.rotX(-0.05f);
        transform3D.mul(rotation);
    }
    bear.setTransform(transform3D);
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
    Move();
}

@Override
public void keyPressed(KeyEvent ev) {
    switch (ev.getKeyChar()) {
        case 'w':
            w = true;
            break;
        case 's':
            s = true;
            break;
        case 'a':
            a = true;
            break;
        case 'd':
            d = true;
            break;
        case 'e':
            e = true;
            break;
        case 'q':
            q = true;
            break;
        case 'z':
            z = true;
            break;
        case 'x':
            _x = true;
            break;
    }
}

@Override
public void keyTyped(KeyEvent e) {}

@Override
public void keyReleased(KeyEvent ev) {
    switch (ev.getKeyChar()) {
        case 'w':
            w = false;
            break;
        case 's':
            s = false;
            break;
        case 'a':
            a = false;
            break;
        case 'd':
            d = false;
            break;
        case 'e':
            e = false;
            break;
        case 'q':
            q = false;
            break;
        case 'z':
            z = false;
            break;
```

```
            case 'x':
                _x = false;
                break;
        }
    }
}
```

Результат