МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №6

з дисципліни “Побудова та анімація зображень за допомогою Java2D”

Виконав
студент III курсу
групи КП-83
Коваль Андрій Олександрович
(прізвище, ім’я, по батькові)
варіант № 8

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

Київ 2021

Завдання

Виконати анімацію 3D сцени за варіантом

Варіант: Анімація тренеру покемонів (із мультфільму) pokemon_trainer.obj. Повинен рухати руками, покемони повинні вилітати з рук, рухатися по екрану та повертатися в руки.

Лістинг коду програми

Main.java

```
package com.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.awt.*;
import java.io.IOException;
import java.util.Hashtable;

public class Main extends JFrame {
      public Canvas3D canvas;
      private Color3f BODY_COLOR = new Color3f(Color.RED);
      private Color3f HAT_COLOR = new Color3f(Color.ORANGE);
      private Color3f HEAD_COLOR = new Color3f(Color.YELLOW);
      private Color3f FACE_COLOR = new Color3f(Color.PINK);
      private Color3f BACKPACK_COLOR = new Color3f(new Color(150, 150, 0));
      private Color3f BALL_COLOR = new Color3f(new Color(205, 235, 190));

      private int ANIMATION_START_TIME = 1000;
      private int ANIMATION_NUM_ROTATIONS = 10000;
      private int ANIMATION_ROTATION_TIME = 3600;

      public Main() throws IOException {
      // canvas & universe
      canvas = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
      SimpleUniverse universe = new SimpleUniverse(canvas);
      universe.getViewingPlatform().setNominalViewingTransform();
      createSceneGraph(universe);

      // window
      setTitle("Коваль Андрій КП-83 лаб6");
      setSize(800, 600);
      getContentPane().add("Center", canvas);
```

```
	setVisible(true);
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

	// mouse navigation
	OrbitBehavior ob = new OrbitBehavior(canvas);
	ob.setSchedulingBounds(new BoundingSphere(new 
Point3d(0.0,0.0,0.0),Double.MAX_VALUE));
	universe.getViewingPlatform().setViewPlatformBehavior(ob);

	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);
	universe.addBranchGraph(bgLight);
	}

	public static void main(String[] args) throws IOException {
	new Main();
	}

	public void createSceneGraph(SimpleUniverse universe) throws IOException 
{
	BoundingSphere bounds = new BoundingSphere(new 
Point3d(0.0,0.0,0.0),Double.MAX_VALUE);
	TextureLoader tl = new TextureLoader("assets/back.jpg", canvas);
	Background background =  new Background(tl.getImage());
	BranchGroup mainBranchGroup = new BranchGroup();

	Scene mainScene = null;
	try {
		mainScene = new 
ObjectFile(ObjectFile.RESIZE).load("assets/pokemon_trainer.obj");
	} catch (Exception e) {
		System.out.println("File loading failed ->" + e);
		throw e;
	}

	Transform3D mainTransform3d = new Transform3D();
	mainTransform3d.rotZ(0);
	mainTransform3d.rotY(Math.PI/3);
	mainTransform3d.setScale(1.0/4);
	TransformGroup mainTransformGroup = new TransformGroup(mainTransform3d);
	Hashtable namedObjects = mainScene.getNamedObjects();
	Shape3D body = (Shape3D) namedObjects.get("body");
	setColorToShape(body, BODY_COLOR);
	Shape3D hat = (Shape3D) namedObjects.get("hat");
	setColorToShape(hat, HAT_COLOR);
	Shape3D head = (Shape3D) namedObjects.get("head");
	setColorToShape(head, HEAD_COLOR);
	Shape3D faceLeft = (Shape3D) namedObjects.get("face-left");
	Shape3D faceRight = (Shape3D) namedObjects.get("face-right");
	setColorToShape(faceLeft, FACE_COLOR);
	setColorToShape(faceRight, FACE_COLOR);
	Shape3D backpack = (Shape3D) namedObjects.get("backpack");
```

```
      setColorToShape(backpack, BACKPACK_COLOR);
      Shape3D ball1 = (Shape3D) namedObjects.get("ball1");
      Shape3D ball2 = (Shape3D) namedObjects.get("ball2");

      setColorToShape(ball1, BALL_COLOR);
      setColorToShape(ball2, BALL_COLOR);

      Shape3D[] fullBody = new Shape3D[] { body, hat, head, faceLeft,
faceRight, backpack };

      for (Shape3D shape:fullBody) {
            mainTransformGroup.addChild(shape.cloneTree());
      }

      Transform3D startTransformation = new Transform3D();
      Transform3D combinedStartTransformation = new Transform3D();
      combinedStartTransformation.mul(startTransformation);
      TransformGroup initialTransformGroup = new
TransformGroup(combinedStartTransformation);

      Transform3D wheel2RotAxis = new Transform3D();
      wheel2RotAxis.set(new Vector3d(0, -0.095, 0.5));
      wheel2RotAxis.setRotation(new AxisAngle4d(0, 0, -0.1, Math.PI / 2));
      TransformGroup tgWheel2 = new TransformGroup();
      tgWheel2.addChild(ball1.cloneTree());

      Transform3D wheel1RotAxis = new Transform3D();
      wheel1RotAxis.set(new Vector3d(0, -0.095, -0.65));
      wheel1RotAxis.setRotation(new AxisAngle4d(0, 0, -0.1, Math.PI / 2));
      TransformGroup tgWheel1 = new TransformGroup();
      tgWheel1.addChild(ball2.cloneTree());

      Alpha wheelRotAlpha = new Alpha(ANIMATION_NUM_ROTATIONS,
Alpha.INCREASING_ENABLE, ANIMATION_START_TIME, 0, ANIMATION_ROTATION_TIME
,0,0,0,0,0);

      RotationInterpolator ball1Rot = new RotationInterpolator(wheelRotAlpha,
tgWheel1, wheel1RotAxis, 0.0f, (float) Math.PI * 2);
      ball1Rot.setSchedulingBounds(bounds);
      tgWheel1.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
      tgWheel1.addChild(ball1Rot);

      RotationInterpolator ball2Rot = new RotationInterpolator(wheelRotAlpha,
tgWheel2, wheel2RotAxis, 0.0f, (float) Math.PI * 2);
      ball2Rot.setSchedulingBounds(bounds);
      tgWheel2.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
      tgWheel2.addChild(ball2Rot);

      Transform3D tfWheel = new Transform3D();
      tfWheel.rotY(Math.PI/3);
      tfWheel.setScale(1.0/4);

      TransformGroup tgCarWheel2 = new TransformGroup(tfWheel);
      tgCarWheel2.addChild(tgWheel2);

      TransformGroup tgCarWheel1 = new TransformGroup(tfWheel);
      tgCarWheel1.addChild(tgWheel1);
```

```
        BranchGroup scene = new BranchGroup();
        scene.addChild(mainTransformGroup);;
        scene.addChild(tgCarWheel2);
        scene.addChild(tgCarWheel1);

        TransformGroup translateXGroup = translate(initialTransformGroup, new
Vector3f(0.0f,0.0f,0.5f));
        TransformGroup rotateXGroup = rotate(translateXGroup, new
Alpha(10,10000));
        mainBranchGroup.addChild(rotateXGroup);
        initialTransformGroup.addChild(scene);


        addBackground(bounds, background, scene);

        mainBranchGroup.compile();
        universe.addBranchGraph(mainBranchGroup);
        }

        private void addBackground(BoundingSphere bounds, Background background,
BranchGroup scene) {
        background.setImageScaleMode(Background.SCALE_FIT_MAX);
        background.setApplicationBounds(bounds);
        background.setCapability(Background.ALLOW_IMAGE_WRITE);
        scene.addChild(background);
        }

        public static void setColorToShape(Shape3D shape, Color3f color) {
        Appearance appearance = new Appearance();
        appearance.setMaterial(new Material(color, color, color, color, 150.0f));
        shape.setAppearance(appearance);
        }


        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 transformGroup = new TransformGroup();
        transformGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        RotationInterpolator interpolator = new RotationInterpolator(alpha,
transformGroup);
        interpolator.setSchedulingBounds(new BoundingSphere(new
Point3d(0.0,0.0,0.0),1.0));
        transformGroup.addChild(interpolator);
        transformGroup.addChild(node);
        return transformGroup;
        }
}
```

Результат роботи