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 № 4

з дисципліни “Побудова найпростіших тривимірних об'єктів за допомогою бібліотеки Java3D та їх анімація”

Виконав
студент III курсу
групи КП-82
Жиров Даниїл
(*прізвище, ім’я, по батькові*)

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

варіант № 6

Київ 2021

Варіант завдання

Завдання:

За допомогою засобів, що надає бібліотека Java3D, побудувати тривимірний об’єкт. Для цього скористатися основними примітивами, що буде доцільно використовувати згідно варіанту: сфера, конус, паралелепіпед, циліндр. Об'єкт має складатися з 5-15 примітивів. Задати матеріал кожного примітиву, в разі необхідності накласти текстуру. В сцені має бути мінімум одне джерело освітлення.

Виконати анімацію сцени таким чином, щоб можна було розглянути об'єкт з усіх сторін. За бажанням можна виконати інтерактивні взаємодію з об'єктом за допомогою миші та клавіатури.

Варіант: Сонячна система

Лістинг коду програми

```
package org.example;

import com.sun.j3d.utils.geometry.Primitive;
import com.sun.j3d.utils.geometry.Sphere;
import com.sun.j3d.utils.image.TextureLoader;
import com.sun.j3d.utils.universe.SimpleUniverse;

import javax.media.j3d.*;
import javax.swing.*;
import javax.vecmath.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.util.Objects;

public class SolarSystem implements ActionListener {

    static int primflags = Primitive.GENERATE_NORMALS + Primitive.GENERATE_TEXTURE_COORDS;
    static TextureAttributes texAttr = new TextureAttributes();
    static Color3f emissive = new Color3f(0f, 0f, 0);
    static Color3f ambient = new Color3f(.15f, .15f, .15f);
    static Color3f diffuse = new Color3f(1f, 1f, 1f);
    static Color3f specular = new Color3f(0f, 0f, 0);

    double angle = 3 * Math.PI / 4;

    private TransformGroup rotationY = new TransformGroup();

    private TransformGroup mercuryTransformation = new TransformGroup();
    private TransformGroup venusTransformation = new TransformGroup();
    private TransformGroup earthTransformation = new TransformGroup();
    private TransformGroup marsTransformation = new TransformGroup();
    private TransformGroup jupiterTransformation = new TransformGroup();
    private TransformGroup saturnTransformation = new TransformGroup();
    private TransformGroup uranusTransformation = new TransformGroup();
    private TransformGroup neptuneTransformation = new TransformGroup();

    static {
        texAttr.setPerspectiveCorrectionMode(TextureAttributes.NICEST);
        texAttr.setTextureMode(TextureAttributes.MODULATE);
    }

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

    public SolarSystem() throws IOException {
        SimpleUniverse universe = new SimpleUniverse();
        BranchGroup group = new BranchGroup();

        Background back = new Background();
        BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
        back.setApplicationBounds(bounds);
        BranchGroup bgGeometry = new BranchGroup();
        Appearance app = new Appearance();
        Texture tex = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("stars.jpg"))
, "RGB", new Container()).getTexture();
        app.setTexture( tex );
        Sphere sphere = new Sphere( 1.0f, Primitive.GENERATE_TEXTURE_COORDS |
Primitive.GENERATE_NORMALS_INWARD, app );
        bgGeometry.addChild( sphere );
        back.setGeometry( bgGeometry );

        rotationY.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        mercuryTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        venusTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        earthTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        marsTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
```

```
        jupiterTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        saturnTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        uranusTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        neptuneTransformation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

        rotationY.setTransform(new Transform3D());
        mercuryTransformation.setTransform(new Transform3D());
        venusTransformation.setTransform(new Transform3D());
        earthTransformation.setTransform(new Transform3D());
        marsTransformation.setTransform(new Transform3D());
        jupiterTransformation.setTransform(new Transform3D());
        saturnTransformation.setTransform(new Transform3D());
        uranusTransformation.setTransform(new Transform3D());
        neptuneTransformation.setTransform(new Transform3D());

        rotationY.addChild(createSun());
        rotationY.addChild(createMercury());
        rotationY.addChild(createVenus());
        rotationY.addChild(createEarth());
        rotationY.addChild(createMars());
        rotationY.addChild(createJupiter());
        rotationY.addChild(createSaturn());
        rotationY.addChild(createUranus());
        rotationY.addChild(createNeptune());

        TransformGroup rotationX = new TransformGroup();
        Transform3D transform = new Transform3D();
        transform.rotX(.6);
        rotationX.setTransform(transform);
        rotationX.addChild(rotationY);

        TransformGroup scale = new TransformGroup();
        transform = new Transform3D();
        transform.setScale(0.88);
        scale.setTransform(transform);
        scale.addChild(rotationX);

        group.addChild(scale);

        Color3f light1Color = new Color3f(1f, 1f, 1f);
        PointLight light1 = new PointLight(light1Color, new Point3f(0f, 0f, 0f), new
Point3f(0.1f, 0.1f, 0.1f));
        light1.setInfluencingBounds(bounds);
        group.addChild(light1);

        Color3f light2Color = new Color3f(1f, 1f, 1f);
        AmbientLight light2 = new AmbientLight(light2Color);
        light2.setInfluencingBounds(bounds);
        group.addChild(light2);

        group.addChild(back);


        ViewPlatform vp = new ViewPlatform();
        vp.setActivationRadius(88);
        universe.getViewingPlatform().setNominalViewingTransform();
        universe.addBranchGraph(group);

        Timer timer = new Timer(50, this);
        timer.start();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        Transform3D transform = new Transform3D();
        angle += 0.01;
        angle %= 256 * Math.PI;
        transform.rotY(-0.08 * angle);
        rotationY.setTransform(transform);

        transformMercury();
        transformVenus();
```

```
        transformEarth();
        transformMars();
        transformJupiter();
        transformSaturn();
        transformUranus();
        transformNeptune();
    }

    public Group createSun() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("sun.jpg")),
"RGB", new Container());
        Texture mercuryTex = loader.getTexture();
        mercuryTex.setBoundaryModeS(Texture.WRAP);
        mercuryTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(mercuryTex);
        app.setTextureAttributes(texAttr);
        Sphere sun = new Sphere(0.15f, primflags, app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(0, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(sun);
        return tg;
    }

    public Group createMercury() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("mercury.jpg"
)), "RGB", new Container());
        Texture mercuryTex = loader.getTexture();
        mercuryTex.setBoundaryModeS(Texture.WRAP);
        mercuryTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(mercuryTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere mercury = new Sphere(0.01f, primflags, app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(.2f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(mercury);
        mercuryTransformation.addChild(tg);
        return mercuryTransformation;
    }

    public void transformMercury() {
        Transform3D transform = new Transform3D();
        transform.rotY(33.2 * angle);
        mercuryTransformation.setTransform(transform);
    }

    public Group createVenus() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("venus.jpg"))
, "RGB", new Container());
        Texture venusTex = loader.getTexture();
        venusTex.setBoundaryModeS(Texture.WRAP);
        venusTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(venusTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere venus = new Sphere(0.02f, primflags, app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
```

```
        Vector3f vector = new Vector3f(.27f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(venus);
        venusTransformation.addChild(tg);
        return venusTransformation;
    }

    public void transformVenus() {
        Transform3D transform = new Transform3D();
        transform.rotY(12.96 * angle);
        venusTransformation.setTransform(transform);
    }

    public Group createEarth() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("earth.jpg"))
, "RGB", new Container());
        Texture earthTex = loader.getTexture();
        earthTex.setBoundaryModeS(Texture.WRAP);
        earthTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(earthTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere earth = new Sphere(0.021f, primflags, app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(.34f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(earth);
        earthTransformation.addChild(tg);
        return earthTransformation;
    }

    public void transformEarth() {
        Transform3D transform = new Transform3D();
        transform.rotY(8 * angle);
        earthTransformation.setTransform(transform);
    }

    public Group createMars() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("mars.jpg")),
"RGB", new Container());
        Texture marsTex = loader.getTexture();
        marsTex.setBoundaryModeS(Texture.WRAP);
        marsTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(marsTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere mars = new Sphere(0.016f, primflags, app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(.41f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(mars);
        marsTransformation.addChild(tg);
        return marsTransformation;
    }

    public void transformMars() {
        Transform3D transform = new Transform3D();
        transform.rotY(4.24 * angle);
        marsTransformation.setTransform(transform);
    }
```

```
    public Group createJupiter() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("jupiter.jpg"
)), "RGB", new Container());
        Texture jupiterTex = loader.getTexture();
        jupiterTex.setBoundaryModeS(Texture.WRAP);
        jupiterTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(jupiterTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere jupiter = new Sphere(0.07f, primflags, app);
        jupiter.setAppearance(app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(.55f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(jupiter);
        jupiterTransformation.addChild(tg);
        return jupiterTransformation;
    }

    public void transformJupiter() {
        Transform3D transform = new Transform3D();
        transform.rotY(0.66 * angle);
        jupiterTransformation.setTransform(transform);
    }

    public Group createSaturn() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("saturn.jpg")
), "RGB", new Container());
        Texture saturnTex = loader.getTexture();
        saturnTex.setBoundaryModeS(Texture.WRAP);
        saturnTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(saturnTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere saturn = new Sphere(0.05f, primflags, app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(.72f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(saturn);
        saturnTransformation.addChild(tg);
        return saturnTransformation;
    }

    public void transformSaturn() {
        Transform3D transform = new Transform3D();
        transform.rotY(.28 * angle);
        saturnTransformation.setTransform(transform);
    }

    public Group createUranus() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("uranus.jpg")
), "RGB", new Container());
        Texture uranusTex = loader.getTexture();
        uranusTex.setBoundaryModeS(Texture.WRAP);
        uranusTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(uranusTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere uranus = new Sphere(0.04f, primflags, app);
```

```
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(.86f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(uranus);
        uranusTransformation.addChild(tg);
        return uranusTransformation;
    }

    public void transformUranus() {
        Transform3D transform = new Transform3D();
        transform.rotY(0.1 * angle);
        uranusTransformation.setTransform(transform);
    }

    public Group createNeptune() {
        TextureLoader loader = new
TextureLoader(Objects.requireNonNull(SolarSystem.class.getClassLoader().getResource("neptune.jpg"
)), "RGB", new Container());
        Texture neptuneTex = loader.getTexture();
        neptuneTex.setBoundaryModeS(Texture.WRAP);
        neptuneTex.setBoundaryModeT(Texture.WRAP);
        Appearance app = new Appearance();
        app.setTexture(neptuneTex);
        app.setTextureAttributes(texAttr);
        Material mat = new Material(ambient, emissive, diffuse, specular, 2.0f);
        app.setMaterial(mat);
        Sphere neptune = new Sphere(0.04f, primflags, app);
        TransformGroup tg = new TransformGroup();
        Transform3D transform = new Transform3D();
        Vector3f vector = new Vector3f(.99f, 0, 0f);
        transform.setTranslation(vector);
        tg.setTransform(transform);
        tg.addChild(neptune);
        neptuneTransformation.addChild(tg);
        return neptuneTransformation;
    }

    public void transformNeptune() {
        Transform3D transform = new Transform3D();
        transform.rotY(0.05 * angle);
        neptuneTransformation.setTransform(transform);
    }
}
```

Результат