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 “Математичні та алгоритмічні основи комп’ютерної графіки”

тема: “Побудова найпростіших тривимірних об'єктів за допомогою бібліотеки Java3D та їх анімація”

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

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

Київ 2021

Мета:

1) вивчення стандартних засобів Java3D для візуалізація зображення;
2) вивчення засобів анімації примітивів та складених об'єктів в Java3D.

Задання на лабораторну роботу

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:

Піротехнічна ракета

Лістинг коду програми

PyrotechnicRocket.java

```
package sample;

import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.universe.SimpleUniverse;

import javax.media.j3d.*;
import javax.swing.*;
import javax.vecmath.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class PyrotechnicRocket implements ActionListener {
    private float upperEyeLimit = 8.0f;
    private float lowerEyeLimit = 5.0f;
    private float farthestEyeLimit = 10.0f;
    private float nearestEyeLimit = 4.0f;

    private TransformGroup treeTransformGroup;
    private TransformGroup viewingTransformGroup;
    private Transform3D treeTransform3D = new Transform3D();
    private Transform3D viewingTransform = new Transform3D();
    private float angle = 0;
    private float eyeHeight;
    private float eyeDistance;
    private boolean descend = true;
    private boolean approaching = true;

    public static void main(String[] args) {
        new PyrotechnicRocket();
    }

    private PyrotechnicRocket() {
        Timer timer = new Timer(50, this);
        SimpleUniverse universe = new SimpleUniverse();

        viewingTransformGroup = universe.getViewingPlatform().getViewPlatformTransform();
        universe.addBranchGraph(createSceneGraph());

        eyeHeight = upperEyeLimit;
        eyeDistance = farthestEyeLimit;
        timer.start();
    }

    private BranchGroup createSceneGraph() {
        BranchGroup objRoot = new BranchGroup();

        treeTransformGroup = new TransformGroup();
        treeTransformGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        buildPyrotechnicRocket();
        objRoot.addChild(treeTransformGroup);

        Background background = new Background(new Color3f(1, 1, 1)); // white color
        BoundingSphere sphere = new BoundingSphere(new Point3d(0,0,0), 100);
        background.setApplicationBounds(sphere);
        objRoot.addChild(background);

        Color3f light1Color = new Color3f(1.7f, 1.6f, .0f);
        BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
        Vector3f light1Direction = new Vector3f(4.0f, -7.0f, -12.0f);
        DirectionalLight light1 = new DirectionalLight(light1Color, light1Direction);
        light1.setInfluencingBounds(bounds);
        objRoot.addChild(light1);

        Color3f ambientColor = new Color3f(1.5f, 0.5f, 0f);
        AmbientLight ambientLightNode = new AmbientLight(ambientColor);
        ambientLightNode.setInfluencingBounds(bounds);

        return objRoot;
    }

    private void buildPyrotechnicRocket() {
        var body = new Cylinder(0.5f, 3, Utils.getBodyAppearence());
        var bodyT = new Transform3D();
```

```
        bodyT.setTranslation(new Vector3f());
        bodyT.rotX(Math.PI / 2);
        var bodyTG = new TransformGroup();
        bodyTG.setTransform(bodyT);
        bodyTG.addChild(body);

        var ball = new Cone(0.5f, 2, Utils.getConeAppearence());
        var ballT = new Transform3D();
        ballT.setTranslation(new Vector3f(0, 2.5f, 0));
        var ballTG = new TransformGroup();
        ballTG.setTransform(ballT);
        ballTG.addChild(ball);
        bodyTG.addChild(ballTG);

        var ball2 = new Cone(0.3f, 1.5f, Utils.getBodyAppearence());
        var ballT2 = new Transform3D();
        ballT2.setTranslation(new Vector3f(0, 3f, 0));
        var ballTG2 = new TransformGroup();
        ballTG2.setTransform(ballT2);
        ballTG2.addChild(ball2);
        bodyTG.addChild(ballTG2);

        var cyl1 = new Cylinder(0.53f, 0.75f, Utils.getCycAppearence());
        var cylT1 = new Transform3D();
        cylT1.setTranslation(new Vector3f(0, -0.85f, 0));
        var cylTG1 = new TransformGroup();
        cylTG1.setTransform(cylT1);
        cylTG1.addChild(cyl1);
        bodyTG.addChild(cylTG1);

        var rub = new Cylinder(0.51f, 0.75f, Utils.getRubberAppearence());
        var rubT = new Transform3D();
        rubT.setTranslation(new Vector3f(0, -1.2f, 0));
        var rubTG = new TransformGroup();
        rubTG.setTransform(rubT);
        rubTG.addChild(rub);
        bodyTG.addChild(rubTG);

        treeTransformGroup.addChild(bodyTG);
    }

    // ActionListener interface
    @Override
    public void actionPerformed(ActionEvent e) {
        float delta = 0.03f;

        treeTransform3D.rotZ(angle);
        treeTransformGroup.setTransform(treeTransform3D);
        angle += delta;

        if (eyeHeight > upperEyeLimit){
            descend = true;
        }else if(eyeHeight < lowerEyeLimit){
            descend = false;
        }
        if (descend){
            eyeHeight -= delta;
        }else{
            eyeHeight += delta;
        }

        if (eyeDistance > farthestEyeLimit){
            approaching = true;
        }else if(eyeDistance < nearestEyeLimit){
            approaching = false;
        }
        if (approaching){
            eyeDistance -= delta;
        }else{
            eyeDistance += delta;
        }

        Point3d eye = new Point3d(eyeDistance, eyeDistance, eyeHeight); // spectator's eye
        Point3d center = new Point3d(.0f, .0f ,0.1f); // sight target
        Vector3d up = new Vector3d(.0f, .0f, 1.0f);; // the camera frustum
        viewingTransform.lookAt(eye, center, up);
        viewingTransform.invert();
```

```
        viewingTransformGroup.setTransform(viewingTransform);
    }
}
```

Результат