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”
Варіант № 4

Виконав:
студент 3-го курсу, групи КП-83,
Дереворіз Назар

Перевірив:
викладач
Шкурат Оксана Сергіївна

Київ - 2021

Завдання: Виконати анімацію тривимірної сцени за варіантом.

4. Анімація гусака goose.obj. Гусак повинен рухати ногами, ходити по екрану, з поворотами .

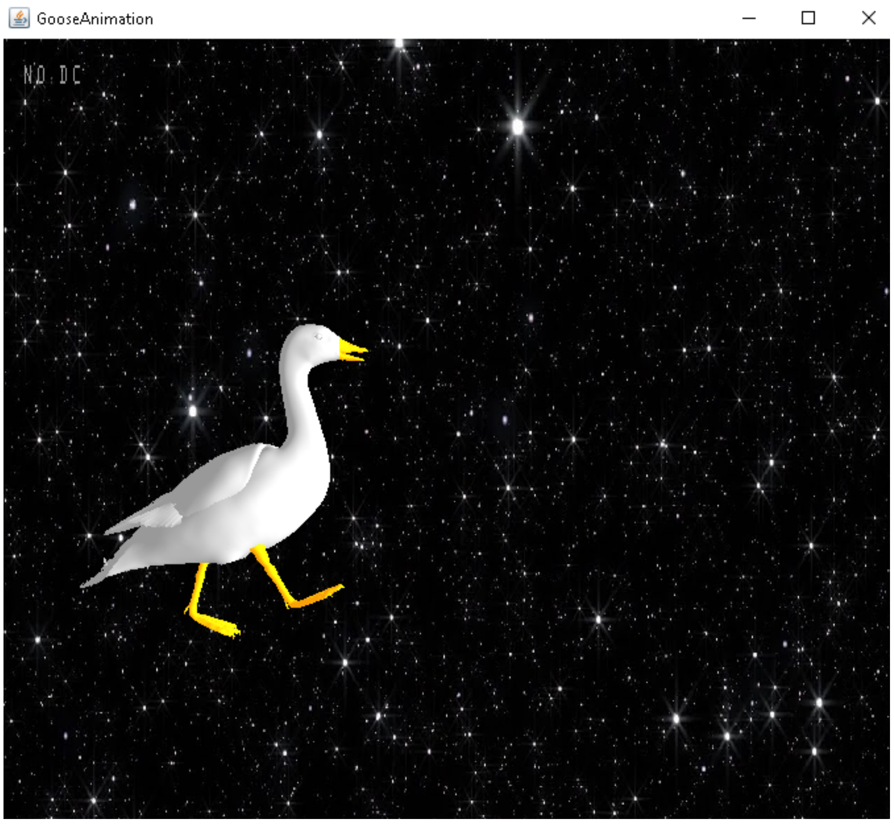
Код програми:

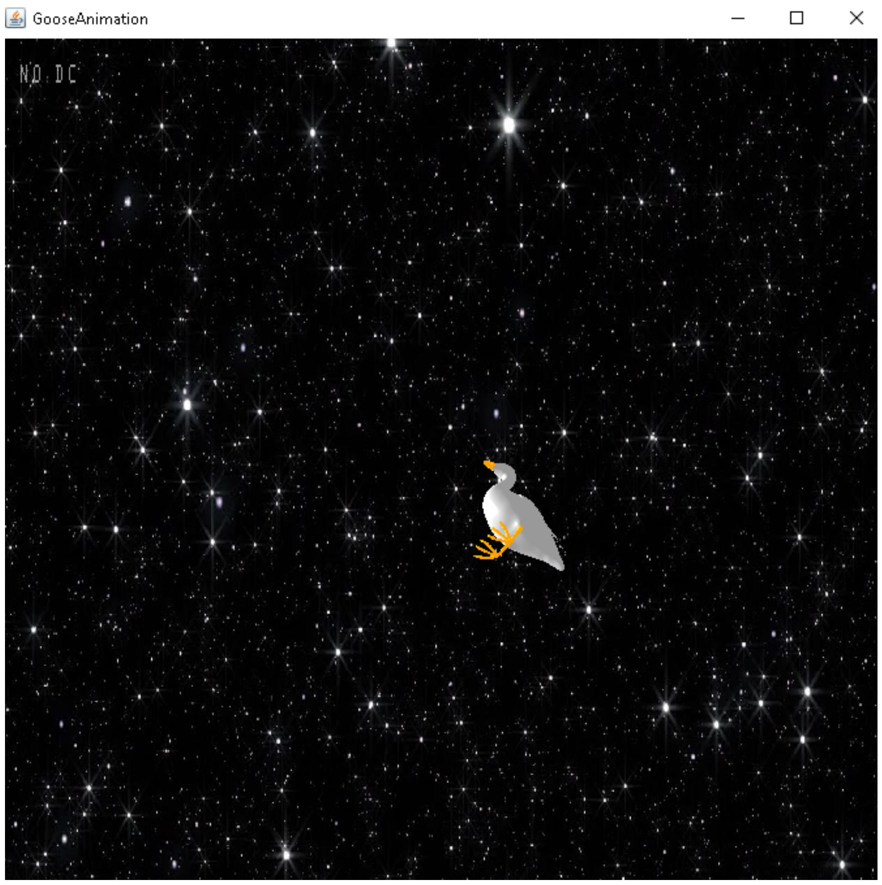
GooseAnimation.java

```java
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.util.Hashtable;
import java.util.Enumeration;

public class GooseAnimation extends JFrame {
   public Canvas3D myCanvas3D;

   public void Run() {
       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("Cosmo-Goose");
       setSize(700, 700);
       getContentPane().add("Center", myCanvas3D);
       setVisible(true);
   }
```

```
    public void createSceneGraph(SimpleUniverse su) {
        ObjectFile f = new ObjectFile(ObjectFile.RESIZE);
        Scene gooseScene = null;
        try {
            gooseScene = f.load("sources/goose.obj");
        } catch (Exception e) {
            System.out.println("File loading failed:" + e);
        }

        Transform3D tfGoose = new Transform3D();
        tfGoose.rotZ(0);
        tfGoose.rotY(Math.PI / 2);
        tfGoose.setScale(1.0 / 3);
        tfGoose.setTranslation(new Vector3d(-0.5f, 0.0f,
0.0f));
        TransformGroup tgGoose = new TransformGroup(tfGoose);

        Hashtable gooseNamedObjects =
gooseScene.getNamedObjects();
        Enumeration enumer = gooseNamedObjects.keys();
        String name;
        while (enumer.hasMoreElements()) {
            name = (String) enumer.nextElement();
            System.out.println("Name: " + name);
        }


        Appearance bodyApp = new Appearance();
        setToMyDefaultAppearance(bodyApp, new Color3f(0.6f,
0.6f, 0.6f));
        Shape3D body = (Shape3D)
gooseNamedObjects.get("body");
        body.setAppearance(bodyApp);


        Appearance orangeApp = new Appearance();
        setToMyDefaultAppearance(orangeApp, new Color3f(250 /
255f, 160 / 255f, 9 / 255f));
        Shape3D beak = (Shape3D)
gooseNamedObjects.get("beak");
        Shape3D legLeft = (Shape3D)
gooseNamedObjects.get("left_leg");
        Shape3D legRight = (Shape3D)
gooseNamedObjects.get("right_leg");
        legLeft.setAppearance(orangeApp);
        legRight.setAppearance(orangeApp);
        beak.setAppearance(orangeApp);

        Shape3D[] goose = new Shape3D[] { body, beak };
        for (Shape3D shape : goose) {
            tgGoose.addChild(shape.cloneTree());
        }

        TransformGroup tgLeftLeg = new TransformGroup();
        tgLeftLeg.addChild(legLeft.cloneTree());

        Transform3D leftLegRotationAxis = new Transform3D();
        leftLegRotationAxis.rotZ(Math.PI / 2);
        int timeStart = 500;
        int timeRotationHour = 500;

        Alpha leftLegRotationAlpha = new Alpha(-1,
Alpha.INCREASING_ENABLE | Alpha.DECREASING_ENABLE, timeStart,
0,
                timeRotationHour, 0, 0, timeRotationHour, 0,
0);

        RotationInterpolator leftLegRotation = new
RotationInterpolator(leftLegRotationAlpha, tgLeftLeg,
                leftLegRotationAxis, (float) Math.PI / 4,
0.0f);
        BoundingSphere bounds = new BoundingSphere(new
Point3d(0.0, 0.0, 0.0), Double.MAX_VALUE);
        leftLegRotation.setSchedulingBounds(bounds);

tgLeftLeg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE)
;
        tgLeftLeg.addChild(leftLegRotation);


        TransformGroup tgRightLeg = new TransformGroup();
        tgRightLeg.addChild(legRight.cloneTree());

        Transform3D rightLegRotationAxis = new Transform3D();
        rightLegRotationAxis.rotZ(Math.PI / 2);

        Alpha rightLegRotationAlpha = new Alpha(-1,
Alpha.INCREASING_ENABLE | Alpha.DECREASING_ENABLE, 0, 0,
                timeRotationHour, 0, 0, timeRotationHour, 0,
0);

        RotationInterpolator rightLegRotation = new
RotationInterpolator(rightLegRotationAlpha, tgRightLeg,
                rightLegRotationAxis, (float) Math.PI / 4,
0.0f);
        rightLegRotation.setSchedulingBounds(bounds);

tgRightLeg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE
);
        tgRightLeg.addChild(rightLegRotation);
        Transform3D tfRotor = new Transform3D();
        Transform3D tfRotor2 = new Transform3D();
        tfRotor.rotZ(-Math.PI / 8);
        tfRotor2.rotY(Math.PI / 2);
        tfRotor.mul(tfRotor2);
        tfRotor.setScale(1.0 / 3);
        tfRotor.setTranslation(new Vector3d(-0.5f, 0.0f,
0.0f));
        TransformGroup tgGooseLegs = new
TransformGroup(tfRotor);
        tgGooseLegs.addChild(tgRightLeg);

tgGooseLegs.setCapability(TransformGroup.ALLOW_TRANSFORM_WRIT
E);
        tgGooseLegs.addChild(tgLeftLeg);

        BranchGroup theScene = new BranchGroup();

        Transform3D gooseRotationAxis = new Transform3D();
        gooseRotationAxis.rotX(Math.PI);
        TransformGroup group = new TransformGroup();
        long crawlTime = 5000;
        Alpha crawlAlpha = new Alpha(-1,
Alpha.INCREASING_ENABLE | Alpha.DECREASING_ENABLE, 0, 0,
crawlTime, 0, 0,
                crawlTime, 0, 0);
        float crawlDistance = 1.0f;

        Alpha gooseRotationAlpha = new Alpha(-1,
Alpha.INCREASING_ENABLE | Alpha.DECREASING_ENABLE, crawlTime,
0, 0, 0,
                crawlTime, 0, 0, crawlTime);
        RotationInterpolator gooseRotation = new
RotationInterpolator(gooseRotationAlpha, group,
gooseRotationAxis,
                0.0f, (float) Math.PI);
        PositionInterpolator posICrawl = new
PositionInterpolator(crawlAlpha, group, gooseRotationAxis,
0.0f,
                crawlDistance);
        posICrawl.setSchedulingBounds(bounds);
        gooseRotation.setSchedulingBounds(bounds);

group.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);


        group.addChild(tgGoose);
        group.addChild(tgGooseLegs);
        theScene.addChild(group);
        TextureLoader t = new
TextureLoader("sources/maxresdefault.jpg",myCanvas3D);
        Background bg = new Background(t.getImage());
        bg.setApplicationBounds(bounds);
        bg.setImageScaleMode(Background.SCALE_FIT_ALL);
        theScene.addChild(bg);
        theScene.compile();

        su.addBranchGraph(theScene);
    }

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

    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);
    }
}
```

Результат роботи програми:


GooseAnimation
NO.DC


GooseAnimation
NO.DC