

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 №** **5**

з дисципліни “Математичні та алгоритмічні основи комп’ютерної графіки“

**Тема:** “Імпорт тривімирних моделей у середовище програмування java3D, обробка та маніапуляція цих зображень. “

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-83  Василець Данило Андрійович  (*прізвище, ім’я, по батькові*)  варіант № 2 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

**Завдання**:

Імпортувати моделі тривимірних об’єктів форматів, що визначені варіантом. Створити реалістичну анімацію об’єкту. Додати до сцени фон, інші об’єкти для надання сцені реалістичного вигляду. Для цього використати текстури, матеріали, імпортувати додаткові об’єкти з відкритих бібліотек, за бажанням створити прості об’єкти у графічному редакторі.

**Варіант: 2**

Бджола на квітах

**Лістинг коду програми**

**Bee.java**

**import** com.sun.j3d.loaders.Scene;

**import** com.sun.j3d.loaders.objectfile.ObjectFile;

**import** com.sun.j3d.utils.image.TextureLoader;

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

**import** com.sun.j3d.utils.universe.ViewingPlatform;

**import** javax.media.j3d.\*;

**import** javax.swing.\*;

**import** javax.vecmath.Color3f;

**import** javax.vecmath.Point3d;

**import** javax.vecmath.Vector3d;

**import** javax.vecmath.Vector3f;

**import** java.awt.\*;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.Map;

**class** Bee **extends** JFrame {

**private** Canvas3D canvas;

**private** SimpleUniverse universe;

**private** BranchGroup root;

**private** TransformGroup bee;

**private** Map<String, Shape3D> shapeMap;

    Bee() **throws** IOException {

        configureWindow();

        configureCanvas();

        configureUniverse();

        root = **new** BranchGroup();

        root.setCapability(BranchGroup.ALLOW\_CHILDREN\_EXTEND);

        addImageBackground("sources/stage.jpg");

        addLightToUniverse();

        changeViewAngle();

        bee = getBeeGroup();

        TransformGroup room = **new** TransformGroup();

        room.addChild(bee);

        root.addChild(room);

        addAppearance();

        BeeAnimation bee = **new** BeeAnimation(**this**);

        canvas.addKeyListener(bee);

        root.compile();

        universe.addBranchGraph(root);

    }

**private** **void** configureWindow() {

        setTitle("Bee Animation");

        setSize(960, 540);

        setResizable(**true**);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

    }

**private** **void** configureCanvas() {

        canvas = **new** Canvas3D(SimpleUniverse.getPreferredConfiguration());

        canvas.setDoubleBufferEnable(**true**);

        canvas.setFocusable(**true**);

        add(canvas, BorderLayout.CENTER);

    }

**private** **void** configureUniverse() {

        universe = **new** SimpleUniverse(canvas);

        universe.getViewingPlatform().setNominalViewingTransform();

    }

    TransformGroup getBeeTransformGroup() {

**return** bee;

    }

**private** TransformGroup getBeeGroup() **throws** IOException {

        Transform3D scale = **new** Transform3D();

        scale.setScale(**new** Vector3d(1, 1, 1));

        Transform3D RotationX = **new** Transform3D();

        RotationX.rotX(-.6);

        Transform3D RotationY = **new** Transform3D();

        RotationY.rotY(13);

        Transform3D RotationZ = **new** Transform3D();

        RotationZ.rotZ(.3);

        RotationZ.mul(RotationX);

        RotationY.mul(RotationZ);

        scale.mul(RotationY);

        TransformGroup group = getModelGroup("sources**\\**bee.obj");

        group.setTransform(scale);

**return** group;

    }

**private** TransformGroup getModelGroup(String path) **throws** IOException {

        Scene scene = getSceneFromFile(path);

        shapeMap = scene.getNamedObjects();

        printModelElementsList(shapeMap);

        TransformGroup group = **new** TransformGroup();

**for** (String shapeName : shapeMap.keySet()) {

            Shape3D shape = shapeMap.get(shapeName);

            scene.getSceneGroup().removeChild(shape);

            group.addChild(shape);

        }

        group.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

**return** group;

    }

**private** **void** printModelElementsList(Map<String, Shape3D> shapeMap) {

**for** (String name : shapeMap.keySet()) {

            System.out.printf("Name: %s**\n**", name);

        }

    }

**private** **void** addAppearance() {

        shapeMap.get("merged\_group").setAppearance(getAppearance(**new** Color(140, 110, 17)));

    }

**private** Appearance getAppearance(Color materialColor) {

        Appearance appearance = **new** Appearance();

        appearance.setMaterial(getMaterial(materialColor));

**return** appearance;

    }

**private** Material getMaterial(Color defaultColor) {

        Material material = **new** Material();

        material.setEmissiveColor(**new** Color3f(Color.BLACK));

        material.setAmbientColor(**new** Color3f(defaultColor));

        material.setDiffuseColor(**new** Color3f(defaultColor));

        material.setSpecularColor(**new** Color3f(defaultColor));

        material.setShininess(80);

        material.setLightingEnable(**true**);

**return** material;

    }

**private** **void** changeViewAngle() {

        ViewingPlatform vp = universe.getViewingPlatform();

        TransformGroup vpGroup = vp.getMultiTransformGroup().getTransformGroup(0);

        Transform3D vpTranslation = **new** Transform3D();

        vpTranslation.setTranslation(**new** Vector3f(0, 0, 6));

        vpGroup.setTransform(vpTranslation);

    }

**private** **static** Scene getSceneFromFile(String location) **throws** IOException {

        ObjectFile file = **new** ObjectFile(ObjectFile.RESIZE);

        file.setFlags(ObjectFile.RESIZE | ObjectFile.TRIANGULATE | ObjectFile.STRIPIFY);

**return** file.load(**new** FileReader(location));

    }

**private** **void** addImageBackground(String imagePath) {

        TextureLoader t = **new** TextureLoader(imagePath, 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);

    }

**private** **void** addLightToUniverse() {

        BoundingSphere bounds = **new** BoundingSphere();

        bounds.setRadius(1000);

        DirectionalLight directionalLight = **new** DirectionalLight(

**new** Color3f(**new** Color(255, 255, 255)),

**new** Vector3f(0, -0.5f, -0.5f));

        directionalLight.setInfluencingBounds(bounds);

        AmbientLight ambientLight = **new** AmbientLight(

**new** Color3f(**new** Color(255, 255, 245)));

        ambientLight.setInfluencingBounds(bounds);

        root.addChild(directionalLight);

        root.addChild(ambientLight);

    }

}

**BeeAnimation.java**

**import** javax.media.j3d.Transform3D;

**import** javax.media.j3d.TransformGroup;

**import** javax.swing.\*;

**import** javax.vecmath.Vector3f;

**import** java.awt.event.\*;

**public** **class** BeeAnimation **extends** KeyAdapter **implements** ActionListener {

**private** **static** **final** **float** DELTA\_DISTANCE = 0.02f;

**private** **static** **final** **float** DELTA\_ANGLE = 0.05f;

**private** TransformGroup bTransformGroup;

**private** Transform3D transform3D = **new** Transform3D();

**private** **float** LocationX = 0;

**private** **float** LocationY = 0;

**private** **boolean** pressedZ = **false**;

**private** **boolean** pressedX = **false**;

**private** **boolean** pressedW = **false**;

**private** **boolean** pressedS = **false**;

**private** **boolean** pressedA = **false**;

**private** **boolean** pressedD = **false**;

**private** **boolean** pressedVKRight = **false**;

**private** **boolean** pressedVKLeft = **false**;

**private** **boolean** pressedVKUp = **false**;

**private** **boolean** pressedVKDown = **false**;

    BeeAnimation(Bee bee) {

**this**.bTransformGroup = bee.getBeeTransformGroup();

**this**.bTransformGroup.getTransform(**this**.transform3D);

        Timer timer = **new** Timer(20, **this**);

        timer.start();

    }

**private** **void** Move() {

**if** (pressedW)

            LocationY += DELTA\_DISTANCE;

**if** (pressedS)

            LocationY -= DELTA\_DISTANCE;

**if** (pressedA)

            LocationX -= DELTA\_DISTANCE;

**if** (pressedD)

            LocationX += DELTA\_DISTANCE;

        transform3D.setTranslation(**new** Vector3f(LocationX, LocationY, 0));

**if** (pressedVKRight) {

            Transform3D rotation = **new** Transform3D();

            rotation.rotZ(DELTA\_ANGLE);

            transform3D.mul(rotation);

        }

**if** (pressedVKLeft) {

            Transform3D rotation = **new** Transform3D();

            rotation.rotZ(-DELTA\_ANGLE);

            transform3D.mul(rotation);

        }

**if** (pressedVKUp) {

            Transform3D rotation = **new** Transform3D();

            rotation.rotX(-DELTA\_ANGLE);

            transform3D.mul(rotation);

        }

**if** (pressedVKDown) {

            Transform3D rotation = **new** Transform3D();

            rotation.rotX(DELTA\_ANGLE);

            transform3D.mul(rotation);

        }

**if** (pressedZ) {

            Transform3D rotation = **new** Transform3D();

            rotation.rotY(-DELTA\_ANGLE);

            transform3D.mul(rotation);

            pressedZ = **false**;

        }

**if** (pressedX) {

            Transform3D rotation = **new** Transform3D();

            rotation.rotY(DELTA\_ANGLE);

            transform3D.mul(rotation);

            pressedX = **false**;

        }

        bTransformGroup.setTransform(transform3D);

    }

    @Override

**public** **void** actionPerformed(ActionEvent e) {

        Move();

    }

    @Override

**public** **void** keyPressed(KeyEvent ev) {

**switch** (ev.getKeyCode()) {

**case** 87: *// W*

                pressedW = **true**;

**break**;

**case** 83: *// S*

                pressedS = **true**;

**break**;

**case** 65: *// A*

                pressedA = **true**;

**break**;

**case** 68: *// D*

                pressedD = **true**;

**break**;

**case** KeyEvent.VK\_LEFT:

                pressedVKLeft = **true**;

**break**;

**case** KeyEvent.VK\_RIGHT:

                pressedVKRight = **true**;

**break**;

**case** KeyEvent.VK\_UP:

                pressedVKUp = **true**;

**break**;

**case** KeyEvent.VK\_DOWN:

                pressedVKDown = **true**;

**break**;

**case** 90:

                pressedZ = **true**;

**break**;

**case** 88:

                pressedX = **true**;

**break**;

        }

    }

    @Override

**public** **void** keyReleased(KeyEvent ev) {

**switch** (ev.getKeyCode()) {

**case** 87: *// W*

                pressedW = **false**;

**break**;

**case** 83: *// S*

                pressedS = **false**;

**break**;

**case** 65: *// A*

                pressedA = **false**;

**break**;

**case** 68: *// D*

                pressedD = **false**;

**break**;

**case** KeyEvent.VK\_RIGHT:

                pressedVKRight = **false**;

**break**;

**case** KeyEvent.VK\_LEFT:

                pressedVKLeft = **false**;

**break**;

**case** KeyEvent.VK\_UP:

                pressedVKUp = **false**;

**break**;

**case** KeyEvent.VK\_DOWN:

                pressedVKDown = **false**;

**break**;

**case** 90:

                pressedZ = **false**;

**break**;

**case** 88:

                pressedX = **false**;

**break**;

        }

    }

**private** **float** degree(**float** degrees) {

**return** (**float**) (degrees \* Math.PI / 180);

    }

}

**Результат**

