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 №5

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

на тему *“*Імпорт тривімирних моделей у середовище програмування java3D, обробка та маніапуляція цих зображень*”*

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

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. Студенти, які мають непарний номер варіанту у списку групи імпортують моделі формату .obj, парний варіант – .lwo.

Варіант 8, необхідно обрати моделі формату .lwo.
Нажаль, під час виконання роботи з’явилися труднощі й не вдалося використати формат .lwo й було обрано .obj.

Релізація

Основні методи для відображення та маніпуляції 3д обьектами було розглянуто у попередній лабораторній роботі. Наразі завданням було імпортуати готову 3д модель та анімувати її будь-яким логічним чином.

Нажаль не вишло нанести текстуру на модель.

У даному випадку було створено класс PathInterpolator для плавної анімації між точками перенесення (Translation3D)

PathInterpolator.java

```
package ua.kpi.fpm.pzks.maokg;

import java.security.InvalidAlgorithmParameterException;

public class PathInterpolator {
    public static class Coordinats3D {
        public float currentX = 0,
        currentY = 0,
```

```
        currentZ = 0,
        currentScale = 0,
        currentRotateX = 0,
        currentRotateY = 0,
        currentRotateZ = 0;
        public Coordinats3D(float x, float y, float z, float scale, float rotX,
float rotY, float rotZ) {
            this.currentX = x;
            this.currentY = y;
            this.currentZ = z;
            this.currentScale = scale;
            this.currentRotateX = rotX;
            this.currentRotateY = rotY;
            this.currentRotateZ = rotZ;
        }
    }

    Coordinats3D[] initialPath;
    Coordinats3D[] interpolatedPath;
    private int iterationsCount;
    private int maxIterationNumber;

    public PathInterpolator(Coordinats3D[] path, int iterationsCount) {
        this.initialPath = path;
        this.iterationsCount = iterationsCount;
        updateInterpolatedPath(path, iterationsCount);
    }

    public int getMaxIterationNumber() {
        return maxIterationNumber;
    }

    public Coordinats3D[] getInterpolatedPath() {
        return this.interpolatedPath;
    }

    private void updateInterpolatedPath(Coordinats3D[] notInterpolatedPath, int
iterationsCount) {
        this.maxIterationNumber = iterationsCount - (iterationsCount %
(notInterpolatedPath.length - 1));
        System.out.println(maxIterationNumber);
        this.interpolatedPath = new Coordinats3D[this.maxIterationNumber];
        int onePathLength = this.maxIterationNumber /
(notInterpolatedPath.length - 1);
        for (int i = 0; i <= maxIterationNumber - onePathLength; i +=
onePathLength) {
            Coordinats3D currentPathPoint = this.initialPath[i / onePathLength];
            Coordinats3D nextPathPoint = this.initialPath[i / onePathLength +
1];
            float xDiff = (nextPathPoint.currentX - currentPathPoint.currentX) /
onePathLength;
            float yDiff = (nextPathPoint.currentY - currentPathPoint.currentY) /
onePathLength;
            float zDiff = (nextPathPoint.currentZ - currentPathPoint.currentZ) /
onePathLength;
            float scaleDiff = (nextPathPoint.currentScale -
currentPathPoint.currentScale) / onePathLength;
            float rotXDiff = (nextPathPoint.currentRotateX -
```

```
currentPathPoint.currentRotateX) / onePathLength;
            float rotYDiff = (nextPathPoint.currentRotateY -
currentPathPoint.currentRotateY) / onePathLength;
            float rotZDiff = (nextPathPoint.currentRotateZ -
currentPathPoint.currentRotateZ) / onePathLength;
            for (int j = 0; j < onePathLength; j++) {
                Coordinats3D interpolatedCoords = new Coordinats3D(
                        currentPathPoint.currentX + xDiff * j,
                        currentPathPoint.currentY + yDiff * j,
                        currentPathPoint.currentZ + zDiff * j,
                        currentPathPoint.currentScale + scaleDiff * j,
                        currentPathPoint.currentRotateX + rotXDiff * j,
                        currentPathPoint.currentRotateY + rotYDiff * j,
                        currentPathPoint.currentRotateZ + rotZDiff * j
                );

                this.interpolatedPath[i + j] = interpolatedCoords;
            }
        }
    }
}
```

Завдяки ньому анімація є достатньо плавною.

Повний програмний код рішення

Main.java

```
package ua.kpi.fpm.pzks.maokg;

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 jdk.jfr.TransitionTo;

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.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.Objects;
```

```
public class Main extends JFrame implements ActionListener, KeyListener {
    private final static String crabLocation = "crab.obj";
    private final static String backgroundLocation = "background.png";
    private final BranchGroup root = new BranchGroup();
    private final Canvas3D canvas = new
Canvas3D(SimpleUniverse.getPreferredConfiguration());
    private final TransformGroup mainTransformGroup = new TransformGroup();
    private final Transform3D transform3D = new Transform3D();
    private Transform3D rotateTransformX = new Transform3D();
    private final Transform3D rotateTransformY = new Transform3D();
    private final Transform3D rotateTransformZ = new Transform3D();
    private final ClassLoader currentClassLoader =
Thread.currentThread().getContextClassLoader();
    private SimpleUniverse universe;
    private Scene scene;
    private Background background;
    private Timer timer;

    private int maxCoordinatsCounter = 200;
    private PathInterpolator.Coordinats3D[] path = new
PathInterpolator.Coordinats3D[] {
            new PathInterpolator.Coordinats3D(0f, 0f, 0.3f, 0.2f, 0.5f, -2.05f,
0.2f),
            new PathInterpolator.Coordinats3D(0f, -0.15f, 0.15f, 0.3f, 0, 3, 0),
            new PathInterpolator.Coordinats3D(0f, -0.35f, 0.35f, 0.4f, 0, 0, 0),
            new PathInterpolator.Coordinats3D(0f, -0.9f, -0.2f, 0.5f, 0, 4, 0),
            new PathInterpolator.Coordinats3D(0f, -0.05f, 0.3f, 3.5f, 0, 0, 5f),
            new PathInterpolator.Coordinats3D(0f, -0.05f, 0.3f, 0.2f, 0, 1, 0),
    };

    private PathInterpolator pathInterpolator = new PathInterpolator(path,
maxCoordinatsCounter);

    private PathInterpolator.Coordinats3D[] interpolatedPath =
pathInterpolator.getInterpolatedPath();
    private int coordinatsCounter = 0;

    private float currentX = path[0].currentX,
                currentY = path[0].currentY,
                currentZ = path[0].currentZ,
                currentScale = path[0].currentScale,
                currentRotateX = path[0].currentRotateX,
                currentRotateY = path[0].currentRotateY,
                currentRotateZ = path[0].currentRotateZ;

    public static void main(String[] args) {
        try {
            Main window = new Main();
            window.setVisible(true);

        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    }

    public Main() throws IOException {
        init();
```

```
        addBackground();
        addLight();
        setInitialViewAngle();
        setInitialLocation();
        compile();
    }

    private void compile() {
        root.compile();
        universe.addBranchGraph(root);
    }

    private void updateLocation(PathInterpolator.Coordinats3D nextCoords,
PathInterpolator.Coordinats3D prevCoords) {
        if (prevCoords != null) {
            rotateTransformX.rotX(nextCoords.currentRotateX -
prevCoords.currentRotateX);
            transform3D.mul(rotateTransformX);
            rotateTransformY.rotY(nextCoords.currentRotateY -
prevCoords.currentRotateY);
            transform3D.mul(rotateTransformY);
            rotateTransformZ.rotZ(nextCoords.currentRotateZ -
prevCoords.currentRotateZ);
            transform3D.mul(rotateTransformZ);
        }
        transform3D.setTranslation(new Vector3f(nextCoords.currentX,
nextCoords.currentY, nextCoords.currentZ));
        transform3D.setScale(nextCoords.currentScale);
        mainTransformGroup.setTransform(transform3D);
    }

    private void setInitialLocation() {
        transform3D.setTranslation(new Vector3f(currentX, currentY, currentZ));
        transform3D.setScale(currentScale);
        mainTransformGroup.setTransform(transform3D);
    }

    private void init() throws IOException {
        setTitle("Андрій Коваль КП-83 ЛАБ 5");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        canvas.setDoubleBufferEnable(true);
        canvas.addKeyListener(this);
        getContentPane().add(canvas, BorderLayout.CENTER);

        universe = new SimpleUniverse(canvas);
        universe.getViewingPlatform().setNominalViewingTransform();

        scene = getSceneFromFile();

        mainTransformGroup.addChild(scene.getSceneGroup());
        mainTransformGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        root.addChild(mainTransformGroup);

        timer = new Timer(10, this);
        timer.start();
    }
```

```
    private void addLight() {
        DirectionalLight dirLight = new DirectionalLight(
                new Color3f(Color.WHITE),
                new Vector3f(4.0f, -7.0f, -12.0f)
        );

        dirLight.setInfluencingBounds(new BoundingSphere(new Point3d(), 1000));
        root.addChild(dirLight);

        AmbientLight ambientLight = new AmbientLight(new Color3f(Color.WHITE));
        DirectionalLight directionalLight = new DirectionalLight(
                new Color3f(Color.BLACK),
                new Vector3f(-1F, -1F, -1F)
        );
        BoundingSphere influenceRegion = new BoundingSphere(new Point3d(),
1000);
        ambientLight.setInfluencingBounds(influenceRegion);
        directionalLight.setInfluencingBounds(influenceRegion);
        root.addChild(ambientLight);
        root.addChild(directionalLight);
    }

    private TextureLoader getTextureLoader(String path) throws IOException {
        URL textureResource = currentClassLoader.getResource(path);
        if (textureResource == null) {
            throw new IOException("Couldn't find texture: " + path);
        }
        return new TextureLoader(textureResource.getPath(), canvas);
    }

    private void addBackground() throws IOException {
        background = new
Background(getTextureLoader(backgroundLocation).getImage());
        background.setImageScaleMode(Background.SCALE_FIT_MAX);
        background.setApplicationBounds(new BoundingSphere(new Point3d(),1000));
        background.setCapability(Background.ALLOW_IMAGE_WRITE);
        root.addChild(background);
    }

    private void setInitialViewAngle() {
        ViewingPlatform vp = universe.getViewingPlatform();
        Transform3D transform = new Transform3D();
        transform.lookAt(
                new Point3d(-4, 0, 0),
                new Point3d(-1, 0, 0),
                new Vector3d(0, 1, 0)
        );
        transform.invert();
        vp.getViewPlatformTransform().setTransform(transform);
    }

    private Scene getSceneFromFile() throws IOException {
        ObjectFile file = new ObjectFile(ObjectFile.RESIZE);
        file.setFlags(ObjectFile.RESIZE | ObjectFile.TRIANGULATE |
ObjectFile.STRIPIFY);
        InputStream inputStream =
currentClassLoader.getResourceAsStream(crabLocation);
```

```
        if (inputStream == null) {
            throw new IOException("Resource " + crabLocation + " not found");
        }
        return file.load(new BufferedReader(new
InputStreamReader(inputStream)));
    }

    @Override
    public void keyPressed(KeyEvent e) {
        float diff = (e.isShiftDown() ? -1 : 1) * 0.05f;
        switch (e.getKeyCode()) {
            case KeyEvent.VK_X: {
                currentX += diff;
            } break;
            case KeyEvent.VK_Y: {
                currentY += diff;
            } break;
            case KeyEvent.VK_Z: {
                currentZ += diff;
            } break;
            case KeyEvent.VK_S: {
                currentScale += diff;
            } break;
            case KeyEvent.VK_UP: {
                currentRotateX += diff;
                rotateTransformX.rotX(currentRotateX);
                transform3D.mul(rotateTransformX);
            } break;
            case KeyEvent.VK_DOWN: {
                currentRotateY += diff;
                rotateTransformY.rotY(diff);
                transform3D.mul(rotateTransformY);
            } break;
            case KeyEvent.VK_LEFT: {
                currentRotateZ += diff;
                rotateTransformZ.rotZ(diff);
                transform3D.mul(rotateTransformZ);
            } break;
        }
        String debugMessage = String.format(
                "\nX: %f, Y: %f, Z: %f, Scale: %f\n" +
                "RotX: %f, RotY: %f, RotZ: %f",
                currentX, currentY, currentZ, currentScale, currentRotateX,
currentRotateY, currentRotateZ
        );
        System.out.println(debugMessage);
        updateLocation(new PathInterpolator.Coordinats3D(
                currentX, currentY, currentZ, currentScale, currentRotateX,
currentRotateY, currentRotateZ
        ), null);
    }

    @Override
    public void keyReleased(KeyEvent e) { }

    @Override
    public void keyTyped(KeyEvent e) { }
```

```
    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        if (coordinatsCounter >= interpolatedPath.length) {
            coordinatsCounter = 0;
        }
        PathInterpolator.Coordinats3D currentCoords =
interpolatedPath[coordinatsCounter];
        PathInterpolator.Coordinats3D prevCoords = coordinatsCounter == 0
                ? null
                : interpolatedPath[coordinatsCounter - 1];
        coordinatsCounter += 1;
        updateLocation(currentCoords, prevCoords);
    }
}
```

Приклади результату


BiKiNi