

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”

|  |  |
| --- | --- |
| Виконав  студент III курсу групи КП-02  Щербатюк Ярослав Сергійович  (*прізвище, ім’я, по батькові*)  варіант № 17 | Зарахована “ ” “ ” 20 р.  викладачем  Шкурат Оксаною Сергіївною  (*прізвище, ім’я, по батькові*) |

Київ 2023

# Варіант завдання

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

**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

# Лістинг коду програми

|  |
| --- |
| **AnimatiomRocket.java** |
| package sample;  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.util.Random;  import javax.media.j3d.\*;  import javax.swing.JFrame;  import javax.swing.Timer;  import javax.vecmath.\*;  public class AnimationRocket implements ActionListener, KeyListener {      private TransformGroup wholerocket;      private Transform3D translateTransform;      private Transform3D rotateTransformX;      private Transform3D rotateTransformZ;      private JFrame mainFrame;      private float zoom = 0.2f;      private float xloc = -1.5499996f;      private float yloc = -1.6999996f;      private float zloc = 0.0f;      private Timer timer;      private double angle = 0, step = 0;      public AnimationRocket(TransformGroup wholerocket, Transform3D trans, JFrame frame) {          this.wholerocket = wholerocket;          this.translateTransform = trans;          this.mainFrame = frame;          rotateTransformX = new Transform3D();          rotateTransformZ = new Transform3D();          rotateTransformX.rotX(3 \* Math.PI / 2);          translateTransform.mul(rotateTransformX);          Main.canvas.addKeyListener(this);          timer = new Timer(100, this);          Panel p = new Panel();          mainFrame.add("North", p);          timer.start();      }      @Override      public void actionPerformed(ActionEvent e) {          step += 0.01;          xloc += ImproveNoise.noise(angle, 0, 0) / 10;          yloc += ImproveNoise.noise(0, angle, 0) / 10;          zloc += ImproveNoise.noise(0, angle, 0) / 10;          Transform3D main = new Transform3D();          main.setTranslation(new Vector3f(xloc, yloc, zloc));          angle += 0.02;          Transform3D rotX = new Transform3D();          rotX.rotX(angle);          Transform3D rotY = new Transform3D();          rotY.rotY(angle);          Transform3D rotZ = new Transform3D();          rotZ.rotZ(angle);          main.mul(rotX);          main.mul(rotY);          main.mul(rotZ);          wholerocket.setTransform(main);      }      @Override      public void keyTyped(KeyEvent e) {          //Invoked when a key has been typed.      }      @Override      public void keyPressed(KeyEvent e) {      }      @Override      public void keyReleased(KeyEvent e) {          // Invoked when a key has been released.      }  } |

|  |
| --- |
| Main.java |
| package sample;  import com.sun.j3d.utils.universe.\*;  import java.awt.Color;  import javax.media.j3d.\*;  import javax.media.j3d.Material;  import javax.vecmath.\*;  import javax.media.j3d.Background;  import com.sun.j3d.loaders.\*;  import com.sun.j3d.loaders.objectfile.ObjectFile;  import com.sun.j3d.loaders.lw3d.Lw3dLoader;  import com.sun.j3d.utils.image.TextureLoader;  import java.awt.\*;  import java.io.FileReader;  import java.io.IOException;  import java.util.Map;  import javax.swing.JFrame;  public class Main extends JFrame {      static SimpleUniverse universe;      static Scene scene;      static Map<String, Shape3D> nameMap;      static BranchGroup root;      static Canvas3D canvas;      static TransformGroup wholeRocket;      static Transform3D transform3D;      public Main() throws IOException {          configureWindow();          configureCanvas();          configureUniverse();          addModelToUniverse();          setRocketElementsList();          addAppearance();          addImageBackground();          addLightToUniverse();          addOtherLight();          ChangeViewAngle();          root.compile();          universe.addBranchGraph(root);      }      private void configureWindow() {          setTitle("Rocket Animation");          setSize(760, 640);          setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);      }      private void configureCanvas() {          canvas = new Canvas3D(SimpleUniverse.getPreferredConfiguration());          canvas.setDoubleBufferEnable(true);          getContentPane().add(canvas, BorderLayout.CENTER);      }      private void configureUniverse() {          root = new BranchGroup();          universe = new SimpleUniverse(canvas);          universe.getViewingPlatform().setNominalViewingTransform();      }      private void addModelToUniverse() throws IOException {          scene = getSceneFromFile("images/rocket.obj");          // scene=getSceneFromLwoFile("d://3dModels//Aspen.lwo");          root = scene.getSceneGroup();      }      private void addLightToUniverse() {          Bounds bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);          Color3f color = new Color3f(65 / 255f, 30 / 255f, 25 / 255f);          Vector3f lightdirection = new Vector3f(-1f, -1f, -1f);          DirectionalLight dirlight = new DirectionalLight(color, lightdirection);          dirlight.setInfluencingBounds(bounds);          root.addChild(dirlight);      }      private void printModelElementsList(Map<String, Shape3D> nameMap) {          for (String name : nameMap.keySet()) {              System.out.printf("Name: %s\n", name);          }      }      private void setRocketElementsList() {          nameMap = scene.getNamedObjects();          //Print elements of your model:          printModelElementsList(nameMap);          wholeRocket = new TransformGroup();          transform3D = new Transform3D();          transform3D.setScale(new Vector3d(0.5, 0.5, 0.5));          wholeRocket.setTransform(transform3D);          root.removeChild(nameMap.get("10475\_rocket\_ship\_v1\_sg"));          wholeRocket.addChild(nameMap.get("10475\_rocket\_ship\_v1\_sg"));          wholeRocket.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);          root.addChild(wholeRocket);      }      Texture getTexture(String path) {          TextureLoader textureLoader = new TextureLoader(path, "RGP", new Container());          Texture texture = textureLoader.getTexture();          texture.setBoundaryModeS(Texture.WRAP);          texture.setBoundaryModeT(Texture.WRAP);          texture.setBoundaryColor(new Color4f(Color.BLUE));          return texture;      }      Material getMaterial() {          Material material = new Material();          material.setAmbientColor(new Color3f(Color.WHITE));          material.setDiffuseColor(new Color3f(Color.CYAN));          material.setSpecularColor(new Color3f(Color.WHITE));          material.setShininess(0.5f);          material.setLightingEnable(true);          return material;      }      private void addAppearance() {          Appearance RocketAppearance = new Appearance();          RocketAppearance.setTexture(getTexture("images/rocket.jpg"));          TextureAttributes texAttr = new TextureAttributes();          texAttr.setTextureMode(TextureAttributes.COMBINE);          RocketAppearance.setTextureAttributes(texAttr);          RocketAppearance.setMaterial(getMaterial());          Shape3D Rocket = nameMap.get("10475\_rocket\_ship\_v1\_sg");          Rocket.setAppearance(RocketAppearance);      }      private void addColorBackground() {          Background background = new Background(new Color3f(Color.CYAN));          BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);          background.setApplicationBounds(bounds);          root.addChild(background);      }      private void addImageBackground() {          TextureLoader t = new TextureLoader("images/universe.jpg", 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 ChangeViewAngle() {          ViewingPlatform vp = universe.getViewingPlatform();          TransformGroup vpGroup = vp.getMultiTransformGroup().getTransformGroup(0);          Transform3D vpTranslation = new Transform3D();          Vector3f translationVector = new Vector3f(0.0F, -1.2F, 6F);          vpTranslation.setTranslation(translationVector);          vpGroup.setTransform(vpTranslation);      }      private void addOtherLight() {          Color3f directionalLightColor = new Color3f(Color.BLACK);          Color3f ambientLightColor = new Color3f(Color.WHITE);          Vector3f lightDirection = new Vector3f(-1F, -1F, -1F);          AmbientLight ambientLight = new AmbientLight(ambientLightColor);          DirectionalLight directionalLight = new DirectionalLight(directionalLightColor, lightDirection);          Bounds influenceRegion = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);          ambientLight.setInfluencingBounds(influenceRegion);          directionalLight.setInfluencingBounds(influenceRegion);          root.addChild(ambientLight);          root.addChild(directionalLight);      }      public 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));      }      //Not always works      public static Scene getSceneFromLwoFile(String location) throws IOException {          Lw3dLoader loader = new Lw3dLoader();          return loader.load(new FileReader(location));      }      public static void main(String[] args) {          try {              Main window = new Main();              AnimationRocket RocketMovement = new AnimationRocket(wholeRocket, transform3D, window);              window.addKeyListener(RocketMovement);              window.setVisible(true);          } catch (IOException ex) {              System.out.println(ex.getMessage());          }      }  } |

|  |
| --- |
| ImproveNoise.java |
| package sample;  public final class ImproveNoise {      static public double noise(double x, double y, double z) {          int X = (int) Math.floor(x) & 255,                  // FIND UNIT CUBE THAT                  Y = (int) Math.floor(y) & 255,                  // CONTAINS POINT.                  Z = (int) Math.floor(z) & 255;          x -= Math.floor(x);                                // FIND RELATIVE X,Y,Z          y -= Math.floor(y);                                // OF POINT IN CUBE.          z -= Math.floor(z);          double u = fade(x),                                // COMPUTE FADE CURVES                  v = fade(y),                                // FOR EACH OF X,Y,Z.                  w = fade(z);          int A = p[X] + Y, AA = p[A] + Z, AB = p[A + 1] + Z,      // HASH COORDINATES OF                  B = p[X + 1] + Y, BA = p[B] + Z, BB = p[B + 1] + Z;      // THE 8 CUBE CORNERS,          return lerp(w, lerp(v, lerp(u, grad(p[AA], x, y, z),  // AND ADD                  grad(p[BA], x - 1, y, z)), // BLENDED                  lerp(u, grad(p[AB], x, y - 1, z),  // RESULTS                          grad(p[BB], x - 1, y - 1, z))),// FROM  8                  lerp(v, lerp(u, grad(p[AA + 1], x, y, z - 1),  // CORNERS                          grad(p[BA + 1], x - 1, y, z - 1)), // OF CUBE                          lerp(u, grad(p[AB + 1], x, y - 1, z - 1),                                  grad(p[BB + 1], x - 1, y - 1, z - 1))));      }      static double fade(double t) {          return t \* t \* t \* (t \* (t \* 6 - 15) + 10);      }      static double lerp(double t, double a, double b) {          return a + t \* (b - a);      }      static double grad(int hash, double x, double y, double z) {          int h = hash & 15;                      // CONVERT LO 4 BITS OF HASH CODE          double u = h < 8 ? x : y,                 // INTO 12 GRADIENT DIRECTIONS.                  v = h < 4 ? y : h == 12 || h == 14 ? x : z;          return ((h & 1) == 0 ? u : -u) + ((h & 2) == 0 ? v : -v);      }      static final int p[] = new int[512], permutation[] = {151, 160, 137, 91, 90, 15,              131, 13, 201, 95, 96, 53, 194, 233, 7, 225, 140, 36, 103, 30, 69, 142, 8, 99, 37, 240, 21, 10, 23,              190, 6, 148, 247, 120, 234, 75, 0, 26, 197, 62, 94, 252, 219, 203, 117, 35, 11, 32, 57, 177, 33,              88, 237, 149, 56, 87, 174, 20, 125, 136, 171, 168, 68, 175, 74, 165, 71, 134, 139, 48, 27, 166,              77, 146, 158, 231, 83, 111, 229, 122, 60, 211, 133, 230, 220, 105, 92, 41, 55, 46, 245, 40, 244,              102, 143, 54, 65, 25, 63, 161, 1, 216, 80, 73, 209, 76, 132, 187, 208, 89, 18, 169, 200, 196,              135, 130, 116, 188, 159, 86, 164, 100, 109, 198, 173, 186, 3, 64, 52, 217, 226, 250, 124, 123,              5, 202, 38, 147, 118, 126, 255, 82, 85, 212, 207, 206, 59, 227, 47, 16, 58, 17, 182, 189, 28, 42,              223, 183, 170, 213, 119, 248, 152, 2, 44, 154, 163, 70, 221, 153, 101, 155, 167, 43, 172, 9,              129, 22, 39, 253, 19, 98, 108, 110, 79, 113, 224, 232, 178, 185, 112, 104, 218, 246, 97, 228,              251, 34, 242, 193, 238, 210, 144, 12, 191, 179, 162, 241, 81, 51, 145, 235, 249, 14, 239, 107,              49, 192, 214, 31, 181, 199, 106, 157, 184, 84, 204, 176, 115, 121, 50, 45, 127, 4, 150, 254,              138, 236, 205, 93, 222, 114, 67, 29, 24, 72, 243, 141, 128, 195, 78, 66, 215, 61, 156, 180      };      static {          for (int i = 0; i < 256; i++) p[256 + i] = p[i] = permutation[i];      }  } |

# Результат



