

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 теми “Імпорт тривімирних моделей у середовище програмування java3D, обробка та маніапуляція цих зображень.”

|  |  |  |  |
| --- | --- | --- | --- |
| Виконала  студентка III курсу  групи КП-82  Лахман Ксенія  (*прізвище, ім’я, по батькові*)  варіант № 11 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Мета:**

1. Здобути навички імпорту моделей, побудованих у тривимірних редакторах, (об’єктів форматів .obj, .lwo, .3ds) до бібліотеки java3D.

2. Навчитися анімувати імпортовані об’єкти.

**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.

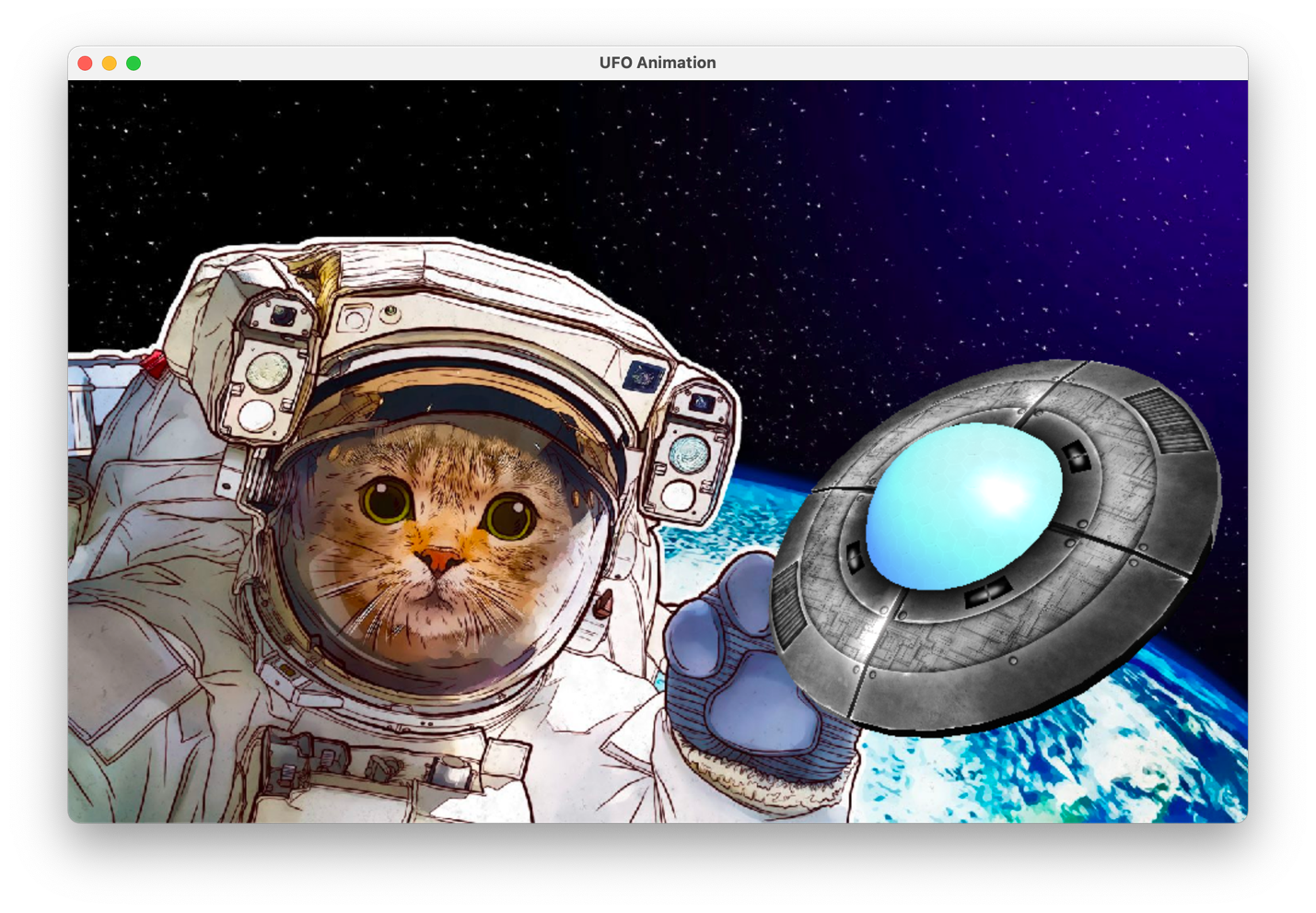
**Лістинг коду програми**

|  |
| --- |
| UFO.java |
| package com.company; 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.\*; import java.awt.\*; import java.io.FileReader; import java.io.IOException; import java.util.Map;  public class UFO extends JFrame {  private Canvas3D canvas;  private SimpleUniverse universe;  private BranchGroup root;   private TransformGroup space = new TransformGroup();  private TransformGroup Ufo;   private Map<String, Shape3D> shapeMap;  private Color lightColor = Color.*white*;  private Color cockpitColor = new Color(108, 164, 248);   public UFO() throws IOException {  configureWindow();  configureCanvas();  configureUniverse();   root = new BranchGroup();  root.setCapability(BranchGroup.*ALLOW\_CHILDREN\_EXTEND*);  addImageBackground("resources/cat.jpg");  addLightToUniverse();  ChangeViewAngle();  Ufo = getUFOGroup();  space.addChild(Ufo);  root.addChild(space);  addAppearance();  Animation UfoAnimation = new Animation(this);  canvas.addKeyListener(UfoAnimation);  root.compile();  universe.addBranchGraph(root);  }   private void configureWindow() {  setTitle("UFO Animation");  setSize(973, 640);  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();  }   private void addImageBackground(String path) {  TextureLoader t = new TextureLoader(path, 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(lightColor),  new Vector3f(0, -0.5f, -0.5f));  directionalLight.setInfluencingBounds(bounds);   AmbientLight ambientLight = new AmbientLight(  new Color3f(lightColor));  ambientLight.setInfluencingBounds(bounds);   root.addChild(directionalLight);  root.addChild(ambientLight);  }   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 TransformGroup getUFOGroup() throws IOException {  Transform3D transform3D = new Transform3D();  transform3D.setTranslation(new Vector3d(1, 0,0));  TransformGroup group = getModelGroup("resources/ufo.obj");  group.setTransform(transform3D);   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);  }  }   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));  }   private void addAppearance() {  Appearance bodyAppearance = new Appearance();  bodyAppearance.setTexture(getTexture("resources/ufo\_diffuse.png"));  TextureAttributes texAttr1 = new TextureAttributes();  texAttr1.setTextureMode(TextureAttributes.*BLEND*);  bodyAppearance.setMaterial(getMaterial(  Color.*BLACK*,  new Color(236, 232, 232)));  shapeMap.get("ufo\_body").setAppearance(bodyAppearance);   Appearance cockpitAppearance = new Appearance();  cockpitAppearance.setTexture(getTexture("resources/ufo\_diffuse\_glow.png"));  TextureAttributes texAttr2 = new TextureAttributes();  texAttr2.setTextureMode(TextureAttributes.*BLEND*);  cockpitAppearance.setTextureAttributes(texAttr2);  cockpitAppearance.setMaterial(getMaterial(Color.*BLUE*, cockpitColor));  Shape3D cockpit = shapeMap.get("ufo\_cockpit");  cockpit.setAppearance(cockpitAppearance);  }   public TransformGroup getUfoTransformGroup() {  return Ufo;  }   Texture getTexture(String path) {  TextureLoader textureLoader = new TextureLoader(path, "LUMINANCE", canvas);  Texture texture = textureLoader.getTexture();  texture.setBoundaryModeS(Texture.*WRAP*);  texture.setBoundaryModeT(Texture.*WRAP*);  texture.setBoundaryColor(new Color4f(0.0f, 1.0f, 1.0f, 0.0f));  return texture;  }   Material getMaterial(Color emissiveColor, Color defaultColor) {  Material material = new Material();  material.setEmissiveColor(new Color3f(emissiveColor));  material.setAmbientColor(new Color3f(defaultColor));  material.setDiffuseColor(new Color3f(defaultColor));  material.setSpecularColor(new Color3f(defaultColor));  material.setShininess(61);  material.setLightingEnable(true);  return material;  } } |

|  |
| --- |
| Animation.java |
| package com.company; import javax.media.j3d.Transform3D; import javax.media.j3d.TransformGroup; import javax.swing.\*; import javax.vecmath.Vector3f; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.awt.event.KeyAdapter; import java.awt.event.KeyEvent;  public class Animation extends KeyAdapter implements ActionListener {  private static final float *DELTA\_DISTANCE* = 0.02f;  private static final float *DELTA\_ANGLE* = 0.05f;   private UFO Ufo;  private TransformGroup UfoTransformGroup;  private Transform3D transform3D = new Transform3D();   private float xLoc = 0;  private float yLoc = 0;   private float xAngle = 0;  private float yAngle = 0;  private float zAngle = 0;    private boolean resetXRotation = false;  private boolean resetYRotation = false;  private boolean resetZRotation = false;   private boolean isRotatedPosY = false;  private boolean isRotatedNegY = false;    private boolean isPressedW = false;  private boolean isPressedS = false;  private boolean isPressedA = false;  private boolean isPressedD = false;  private boolean isPressedRight = false;  private boolean isPressedLeft = false;  private boolean isPressedUp = false;  private boolean isPressedDown = false;   @Override  public void actionPerformed(ActionEvent e) {  Move();  }   public Animation(UFO ufo) {  this.Ufo = ufo;   this.UfoTransformGroup = Ufo.getUfoTransformGroup();  this.UfoTransformGroup.getTransform(this.transform3D);   Timer timer = new Timer(20, this);  timer.start();  }   private void Move() {  if (isPressedW) {  yLoc += *DELTA\_DISTANCE*;  }   if (isPressedS) {  yLoc -= *DELTA\_DISTANCE*;  }   if (isPressedA) {  xLoc -= *DELTA\_DISTANCE*;  }   if (isPressedD) {  xLoc += *DELTA\_DISTANCE*;  }   transform3D.setTranslation(new Vector3f(xLoc, yLoc, 0));   if (isPressedRight) {  Transform3D rotation = new Transform3D();  rotation.rotZ(*DELTA\_ANGLE*);  zAngle += *DELTA\_ANGLE*;  transform3D.mul(rotation);  }   if (isPressedLeft) {  Transform3D rotation = new Transform3D();  rotation.rotZ(-*DELTA\_ANGLE*);  zAngle -= *DELTA\_ANGLE*;  transform3D.mul(rotation);  }   if (isPressedUp) {  Transform3D rotation = new Transform3D();  rotation.rotX(-*DELTA\_ANGLE*);  xAngle -= *DELTA\_ANGLE*;  transform3D.mul(rotation);  }   if (isPressedDown) {  Transform3D rotation = new Transform3D();  rotation.rotX(*DELTA\_ANGLE*);  xAngle += *DELTA\_ANGLE*;  transform3D.mul(rotation);  }   if (isRotatedPosY) {  Transform3D rotation = new Transform3D();  rotation.rotY(degree(20));  transform3D.mul(rotation);   yAngle += degree(20);  isRotatedPosY = false;  }   if (isRotatedNegY) {  Transform3D rotation = new Transform3D();  rotation.rotY(degree(-20));  transform3D.mul(rotation);   yAngle += degree(-20);  isRotatedNegY = false;  }   if (resetYRotation) {  Transform3D rotation = new Transform3D();  rotation.rotY(-yAngle);  transform3D.mul(rotation);   resetYRotation = false;  yAngle = 0;  }   if (resetZRotation) {  Transform3D rotation = new Transform3D();  rotation.rotZ(-zAngle);  transform3D.mul(rotation);   resetZRotation = false;  zAngle = 0;  }   if (resetXRotation) {  Transform3D rotation = new Transform3D();  rotation.rotX(-xAngle);  transform3D.mul(rotation);   resetXRotation = false;  xAngle = 0;  }   UfoTransformGroup.setTransform(transform3D);  }   @Override  public void keyPressed(KeyEvent ev) {  switch (ev.getKeyCode()) {  case 87: *// W* isPressedW = true;  break;  case 83: *// S* isPressedS = true;  break;  case 65: *// A* if (!isPressedA) {  isPressedA = true;  isRotatedNegY = true;  }  break;  case 68: *// D* if (!isPressedD) {  isPressedD = true;  isRotatedPosY = true;  }  break;  case KeyEvent.*VK\_LEFT*:  isPressedLeft = true;  break;  case KeyEvent.*VK\_RIGHT*:  isPressedRight = true;  break;  case KeyEvent.*VK\_UP*:  isPressedUp = true;  break;  case KeyEvent.*VK\_DOWN*:  isPressedDown = true;  break;  case 48: *// 0* resetXRotation = true;  resetYRotation = true;  resetZRotation = true;  break;  }  }   @Override  public void keyReleased(KeyEvent ev) {  switch (ev.getKeyCode()) {  case 87: *// W* isPressedW = false;  break;  case 83: *//S* isPressedS = false;  break;  case 65: *//A* isPressedA = false;  resetYRotation = true;  break;  case 68: *//D* isPressedD = false;  resetYRotation = true;  break;  case KeyEvent.*VK\_RIGHT*:  isPressedRight = false;  break;  case KeyEvent.*VK\_LEFT*:  isPressedLeft = false;  break;  case KeyEvent.*VK\_UP*:  isPressedUp = false;  break;  case KeyEvent.*VK\_DOWN*:  isPressedDown = false;  break;  }  }   private float degree(float degrees) {  return (float) (degrees \* Math.*PI* / 180);  } } |

|  |
| --- |
| Main.java |
| package com.company; import java.io.IOException;  public class Main {  public static void main(String[] args) {  try {  UFO window = new UFO();  window.setVisible(true);  } catch (IOException ex) {  System.*out*.println(ex.getMessage());  }  } } |

**Результат**

****