

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 5**

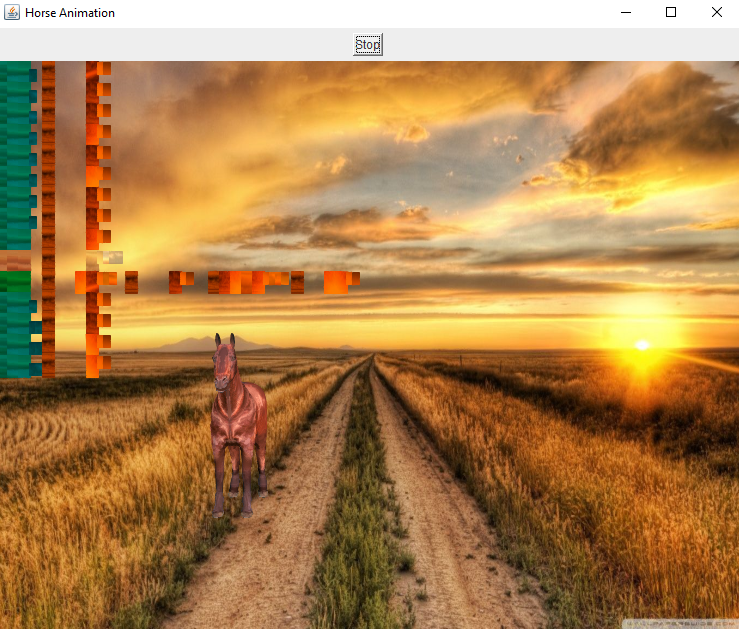
з дисципліни “ МАОКГ”

|  |  |  |
| --- | --- | --- |
| Виконав  студент III курсу  групи КП-82  Залізний Захар Юрійович  (*прізвище, ім’я, по батькові*)  Варіант № 7 |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2021 р.  викладачем  Шкурат О. С.  (*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.

**Результат виконання програми :**



**Код програми:**

|  |
| --- |
| **FirstMainClass.java** |
| package Lab5.src.com.company;  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 FirstMainClass extends JFrame {  static SimpleUniverse *universe*;  static Scene *scene*;  static Map<String, Shape3D> *nameMap*;  static BranchGroup *root*;  static Canvas3D *canvas*;    static TransformGroup *wholeHorse*;  static Transform3D *transform3D*;    public FirstMainClass () throws IOException{  configureWindow();  configureCanvas();  configureUniverse();  addModelToUniverse();  setHorseElementsList();  addAppearance();  addImageBackground();  addLightToUniverse();  addOtherLight();  ChangeViewAngle();  *root*.compile();  *universe*.addBranchGraph(*root*);  }    private void configureWindow() {  setTitle("Horse 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*("G:\\Maokg\\src\\main\\java\\Lab5\\src\\com\\company\\horse.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 setHorseElementsList() {  *nameMap*=*scene*.getNamedObjects();  //Print elements of your model:  printModelElementsList(*nameMap*);    *wholeHorse* = new TransformGroup();  *transform3D* = new Transform3D();  *transform3D*.setScale(new Vector3d(0.5,0.5,0.5));  *wholeHorse*.setTransform(*transform3D*);  *root*.removeChild(*nameMap*.get("10026\_horse\_v01"));  *wholeHorse*.addChild(*nameMap*.get("10026\_horse\_v01"));  *wholeHorse*.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  *root*.addChild(*wholeHorse*);  }    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 HorseAppearance = new Appearance();  HorseAppearance.setTexture(getTexture("G:\\Maokg\\src\\main\\java\\Lab5\\src\\com\\company\\fur1.jpg"));  TextureAttributes texAttr = new TextureAttributes();  texAttr.setTextureMode(TextureAttributes.*COMBINE*);  HorseAppearance.setTextureAttributes(texAttr);  HorseAppearance.setMaterial(getMaterial());  Shape3D Horse = *nameMap*.get("10026\_horse\_v01");  Horse.setAppearance(HorseAppearance);  }    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("G:\\Maokg\\src\\main\\java\\Lab5\\src\\com\\company\\background.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 {  FirstMainClass window = new FirstMainClass();  AnimationHorse HorseMovement = new AnimationHorse(*wholeHorse*, *transform3D*, window);  window.addKeyListener(HorseMovement);  window.setVisible(true);  }  catch (IOException ex) {  System.*out*.println(ex.getMessage());  }  }  } |

|  |
| --- |
| **AnimationHorse.java** |
| package Lab5.src.com.company;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.KeyEvent;  import java.awt.event.KeyListener;  import javax.media.j3d.\*;  import javax.swing.JFrame;  import javax.swing.Timer;  import javax.vecmath.\*;  public class AnimationHorse implements ActionListener, KeyListener {  private Button go;  private TransformGroup wholeHorse;  private Transform3D translateTransform;  private Transform3D rotateTransformX;  private Transform3D rotateTransformY;  private Transform3D rotateTransformZ;  private JFrame mainFrame;  private float signX = 1.0f;  private float signY = 1.0f;  private float signZoom = 1.0f;  private float zoom = 0.2f;  private float xloc = -0.1f;  private float yloc = -1.2f;  private float zloc = 0.0f;  private int moveType = 1;  private int sideSign = 1;  private Timer timer;  int side = 0;  public AnimationHorse(TransformGroup wholeHorse, Transform3D trans, JFrame frame) {  go = new Button("Go");  this.wholeHorse = wholeHorse;  this.translateTransform = trans;  this.mainFrame = frame;  rotateTransformX = new Transform3D();  rotateTransformY = new Transform3D();  rotateTransformZ = new Transform3D();  rotateTransformX.rotX(3 \* Math.*PI* / 2);  translateTransform.mul(rotateTransformX);  FirstMainClass.*canvas*.addKeyListener(this);  timer = new Timer(100, this);  Panel p = new Panel();  p.add(go);  mainFrame.add("North", p);  go.addActionListener(this);  go.addKeyListener(this);  }  @Override  public void actionPerformed(ActionEvent e) {  // start timer when button is pressed  if (e.getSource() == go) {  if (!timer.isRunning()) {  timer.start();  go.setLabel("Stop");  } else {  timer.stop();  go.setLabel("Go");  }  } else {  Move(moveType);  translateTransform.setScale(new Vector3d(zoom, zoom, zoom));  translateTransform.setTranslation(new Vector3f(xloc, yloc, zloc));  wholeHorse.setTransform(translateTransform);  }  }  private void Move(int mType) {  if (side == 0) {  rotateTransformX.rotX(Math.*PI* / 16);  translateTransform.mul(rotateTransformX);  side = 1;  sideSign = -1;  } else if (side == 2) {  rotateTransformX.rotX(-Math.*PI* / 16);  translateTransform.mul(rotateTransformX);  side = 1;  sideSign = 1;  } else {  rotateTransformX.rotX(sideSign \* Math.*PI* / 16);  translateTransform.mul(rotateTransformX);  side = 2;  }  if (xloc <= -2.199999 && yloc <= -2.249999 && signX > 0) {  rotateTransformZ.rotZ(Math.*PI* / 4);  translateTransform.mul(rotateTransformZ);  signX = -1;  } else if (xloc >= -2.0499992 && yloc <= -2.4999988 && signX < 0 && signY != 0) {  rotateTransformZ.rotZ(Math.*PI* / 4);  translateTransform.mul(rotateTransformZ);  signY = 0;  signZoom = 0;  signX = -2;  } else if (xloc >= 1.6999993 && yloc <= -2.249999 && signX < 0 && signY == 0 && signZoom == 0) {  rotateTransformZ.rotZ(Math.*PI* / 16);  translateTransform.mul(rotateTransformZ);  signY = -1;  signZoom = -1;  signX = -1;  } else if (xloc >= 1.9499991 && yloc >= -2.349999 && signX < 0 && signY < 0 && signZoom < 0) {  rotateTransformZ.rotZ(Math.*PI* / 4);  translateTransform.mul(rotateTransformZ);  signX = 1;  signY = -0.73f;  } else if (xloc <= 1.6499994 && yloc >= -2.0444992 && signX > 0 && signY < 0 && signZoom < 0) {  xloc = -1.5499996f;  yloc = -1.6999996f;  signX = 1.0f;  signY = 1.0f;  signZoom = 1.0f;  Transform3D vpTranslation = new Transform3D();  Vector3f translationVector = new Vector3f(0.0F, -1.2F, 6F);  vpTranslation.setTranslation(translationVector);  translateTransform = vpTranslation;  rotateTransformX.rotX(3 \* Math.*PI* / 2);  translateTransform.mul(rotateTransformX);  }  xloc = xloc - signX \* .03f;  yloc = yloc - signY \* .02f;  zoom = zoom + signZoom \* 0.018f;  }  @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.  }  } |