

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

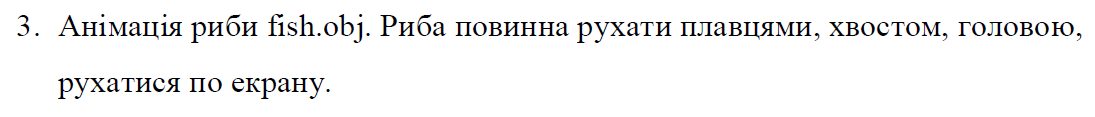
**Лабораторна робота № 6**

з дисципліни “ МАОКГ”

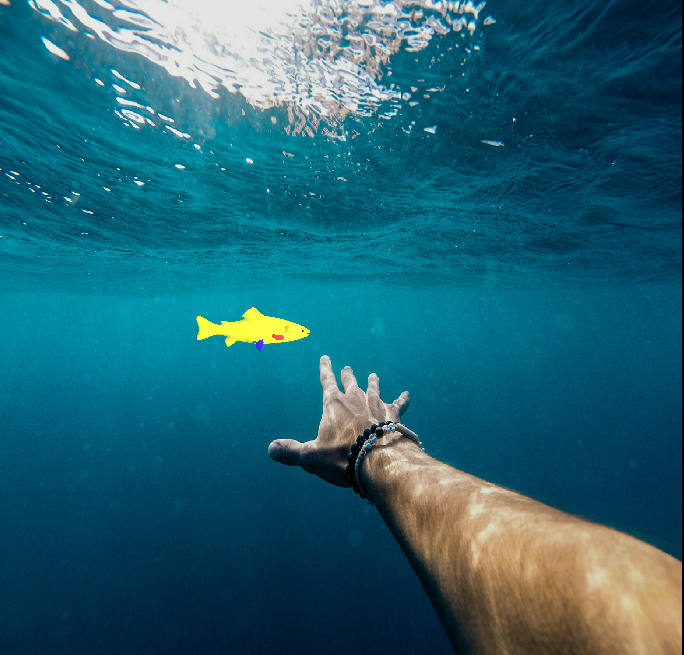
|  |  |  |
| --- | --- | --- |
| Виконав  студент III курсу  групи КП-72  Божко Володимир Сергійович  (*прізвище, ім’я, по батькові*)  Варіант № 3 |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2020  р.  викладачем  Шкурат О. С.  (*прізвище, ім’я, по батькові*) |

Київ-2020

Завдання:



Результат: :



|  |  |  |
| --- | --- | --- |
| |  | | --- | | Лістинг коду програми (class Fish): | | import javax.vecmath.\*;  import com.sun.j3d.utils.image.TextureLoader; import com.sun.j3d.utils.universe.\*; import javax.media.j3d.\*; import com.sun.j3d.utils.behaviors.vp.\*; import javax.swing.JFrame; import com.sun.j3d.loaders.\*; import com.sun.j3d.loaders.objectfile.\*; import java.util.Hashtable;   public class Fish extends JFrame{  public Canvas3D myCanvas3D;   public Fish(){  this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);   myCanvas3D = new Canvas3D(SimpleUniverse.*getPreferredConfiguration*());   SimpleUniverse simpUniv = new SimpleUniverse(myCanvas3D);   simpUniv.getViewingPlatform().setNominalViewingTransform();   createSceneGraph(simpUniv);   addLight(simpUniv);   OrbitBehavior ob = new OrbitBehavior(myCanvas3D);  ob.setSchedulingBounds(new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.*MAX\_VALUE*));  simpUniv.getViewingPlatform().setViewPlatformBehavior(ob);   setTitle("Fishing ukrainian fish");  setSize(700,700);  getContentPane().add("Center", myCanvas3D);  setVisible(true);  }   public void createSceneGraph(SimpleUniverse su){  ObjectFile f = new ObjectFile(ObjectFile.*RESIZE*);  Scene fishScene = null;  try {  fishScene = f.load("3dModels//fish.obj");  } catch (Exception e){  System.*out*.println("File loading failed:" + e);  }   Transform3D scaling = new Transform3D();  scaling.setScale(1.0/6);  Transform3D tf\_Roach = new Transform3D();  tf\_Roach.rotY(5\*Math.*PI*/2);  tf\_Roach.mul(scaling);  TransformGroup tg\_Roach = new TransformGroup(tf\_Roach);  TransformGroup sceneGroup = new TransformGroup();    Hashtable namedObjects = fishScene.getNamedObjects();     BoundingSphere bounds = new BoundingSphere(new Point3d(120.0,250.0,100.0),Double.*MAX\_VALUE*);  BranchGroup theScene = new BranchGroup();   TransformGroup tg\_Body = new TransformGroup();   Appearance bodyApp = new Appearance();  *setToMyDefaultAppearance*(bodyApp, new Color3f(0.8f, 0.8f, 0.2f));   Shape3D fishBody = (Shape3D) namedObjects.get("rt\_body");  fishBody.setAppearance(bodyApp);  tg\_Body.addChild(fishBody.cloneTree());   int noRotTime = 25;  int timeForRotation = 500;   BoundingSphere bs = new BoundingSphere(new Point3d(0.0,0.0,0.0),Double.*MAX\_VALUE*);   int timeStart = 0;   Alpha ventral\_fin1RotAlpha = new Alpha(noRotTime,Alpha.*INCREASING\_ENABLE*,timeStart,0,timeForRotation,  0,0,0,0,0);   Shape3D ventral\_fin1 = (Shape3D) namedObjects.get("ventral\_fin2");   Appearance ventral\_fin1App = new Appearance();  *setToMyDefaultAppearance*(ventral\_fin1App, new Color3f(0.2f, 0.2f, 0.8f));  ventral\_fin1.setAppearance(ventral\_fin1App);   TransformGroup tg\_ventral\_fin1 = new TransformGroup();  tg\_ventral\_fin1.addChild(ventral\_fin1.cloneTree());   Transform3D ventral\_finRotAxis = new Transform3D();  ventral\_finRotAxis.rotZ(Math.*PI*/2);   RotationInterpolator ventral\_fin1Rotation = new RotationInterpolator(ventral\_fin1RotAlpha,tg\_ventral\_fin1,ventral\_finRotAxis,(float) Math.*PI*/2,0.0f);  ventral\_fin1Rotation.setSchedulingBounds(bs);  tg\_ventral\_fin1.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  tg\_ventral\_fin1.addChild(ventral\_fin1Rotation);  Alpha ventral\_fin2RotAlpha = new Alpha(noRotTime,Alpha.*INCREASING\_ENABLE*,timeStart,0,timeForRotation,  0,0,0,0,0);   Shape3D ventral\_fin2 = (Shape3D) namedObjects.get("ventral\_finq");   Appearance ventral\_fin2App = new Appearance();  *setToMyDefaultAppearance*(ventral\_fin2App, new Color3f(0.2f, 0.2f, 0.8f));  ventral\_fin2.setAppearance(ventral\_fin2App);   TransformGroup tg\_ventral\_fin2 = new TransformGroup();  tg\_ventral\_fin2.addChild(ventral\_fin2.cloneTree());   Transform3D ventral\_fin2RotAxis = new Transform3D();  ventral\_fin2RotAxis.rotZ(Math.*PI*/2);   RotationInterpolator ventral\_fin2Rotation = new RotationInterpolator(ventral\_fin2RotAlpha,tg\_ventral\_fin2,ventral\_fin2RotAxis,(float) Math.*PI*/2,0.0f);  ventral\_fin2Rotation.setSchedulingBounds(bs);  tg\_ventral\_fin2.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  tg\_ventral\_fin2.addChild(ventral\_fin2Rotation);   Alpha tailRotAlpha = new Alpha(noRotTime,Alpha.*INCREASING\_ENABLE*,timeStart,0,timeForRotation,  0,0,0,0,0);   Shape3D tail = (Shape3D) namedObjects.get("tail");  TransformGroup tg\_tail = new TransformGroup();   Appearance tailApp = new Appearance();  *setToMyDefaultAppearance*(tailApp, new Color3f(0.8f, 0.8f, 0.2f));  tail.setAppearance(tailApp);   Transform3D transform3D = new Transform3D();  transform3D.setTranslation(new Vector3d(0, 0, 0));   tg\_tail.setTransform(transform3D);   tg\_tail.addChild(tail.cloneTree());   Transform3D tailRotAxis = new Transform3D();  tailRotAxis.set(new Vector3d(0.1, 0.1, 0));  tailRotAxis.setRotation(new AxisAngle4d(0, 0.3, 0, Math.*PI*));   RotationInterpolator tailRotation = new RotationInterpolator(tailRotAlpha,tg\_tail,tailRotAxis,(float) Math.*PI*/24,-(float) Math.*PI*/24);  tailRotation.setSchedulingBounds(bs);  tg\_tail.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  tg\_tail.addChild(tailRotation);    Shape3D rt\_eye = (Shape3D) namedObjects.get("rt\_eye");  Shape3D rt\_mouth = (Shape3D) namedObjects.get("rt\_mouth");    Alpha headRotAlpha = new Alpha(noRotTime,Alpha.*INCREASING\_ENABLE*,timeStart,0,timeForRotation,  0,0,0,0,0);   Shape3D head = (Shape3D) namedObjects.get("head");  TransformGroup tg\_head = new TransformGroup();   Appearance headApp = new Appearance();  *setToMyDefaultAppearance*(headApp, new Color3f(0.8f, 0.8f, 0.2f));  head.setAppearance(headApp);   tg\_head.setTransform(transform3D);   tg\_head.addChild(head.cloneTree());   Transform3D headRotAxis = new Transform3D();  headRotAxis.set(new Vector3d(0.3, 0.3, 0));  headRotAxis.setRotation(new AxisAngle4d(0, 0.9, 0, Math.*PI*));   RotationInterpolator headRotation = new RotationInterpolator(headRotAlpha,tg\_head,headRotAxis,(float) Math.*PI*/24,-(float) Math.*PI*/24);  headRotation.setSchedulingBounds(bs);  tg\_head.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  tg\_head.addChild(headRotation);   Shape3D fin1 = (Shape3D) namedObjects.get("fin1");   Appearance fin1App = new Appearance();  *setToMyDefaultAppearance*(fin1App, new Color3f(0.8f, 0.2f, 0.2f));  fin1.setAppearance(fin1App);   Shape3D fin2 = (Shape3D) namedObjects.get("fin2");   Appearance fin2App = new Appearance();  *setToMyDefaultAppearance*(fin2App, new Color3f(0.8f, 0.2f, 0.2f));  fin2.setAppearance(fin2App);   sceneGroup.addChild(tg\_ventral\_fin1);  sceneGroup.addChild(tg\_ventral\_fin2);  sceneGroup.addChild(rt\_eye.cloneTree());  sceneGroup.addChild(tg\_tail);  sceneGroup.addChild(rt\_mouth.cloneTree());   sceneGroup.addChild(head.cloneTree());  sceneGroup.addChild(fin1.cloneTree());  sceneGroup.addChild(fin2.cloneTree());     sceneGroup.addChild(tg\_Body.cloneTree());    Transform3D tCrawl = new Transform3D();  tCrawl.rotY(-Math.*PI*/2);   long crawlTime = 10000;  Alpha crawlAlpha = new Alpha(1,  Alpha.*INCREASING\_ENABLE*,  0,  0, crawlTime,0,0,0,0,0);   float crawlDistance = 5.0f;  PositionInterpolator posICrawl = new PositionInterpolator(crawlAlpha,  sceneGroup,tCrawl, -9.0f, crawlDistance);   posICrawl.setSchedulingBounds(bs);  sceneGroup.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  sceneGroup.addChild(posICrawl);   Transform3D tCrawl2 = new Transform3D();  tCrawl2.rotX(Math.*PI*/2);  RotationInterpolator testRot = new RotationInterpolator(crawlAlpha,sceneGroup,tCrawl2,(float) Math.*PI*/2,0.0f);  testRot.setSchedulingBounds(bs);  sceneGroup.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  sceneGroup.addChild(testRot);   tg\_Roach.addChild(sceneGroup);  theScene.addChild(tg\_Roach);    Canvas3D canvas;  canvas = new Canvas3D(SimpleUniverse.*getPreferredConfiguration*());  TextureLoader t = new TextureLoader("3dModels//sea.jpg",  canvas);  Background bg = new Background(t.getImage());  bg.setImageScaleMode(Background.*SCALE\_FIT\_ALL*);  bg.setApplicationBounds(bounds);  theScene.addChild(bg);  theScene.compile();   su.addBranchGraph(theScene);  }   public static void setToMyDefaultAppearance(Appearance app, Color3f col){  app.setMaterial(new Material(col,col,col,col,150.0f));  }     public void addLight(SimpleUniverse su){  BranchGroup bgLight = new BranchGroup();  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);  Color3f lightColour1 = new Color3f(1.0f,1.0f,1.0f);  Vector3f lightDir1 = new Vector3f(-1.0f,0.0f,-0.5f);  DirectionalLight light1 = new DirectionalLight(lightColour1, lightDir1);  light1.setInfluencingBounds(bounds);  bgLight.addChild(light1);  su.addBranchGraph(bgLight);  }  public static void main(String[] args) {  Fish fish = new Fish();  } } | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |