

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 №** **6**

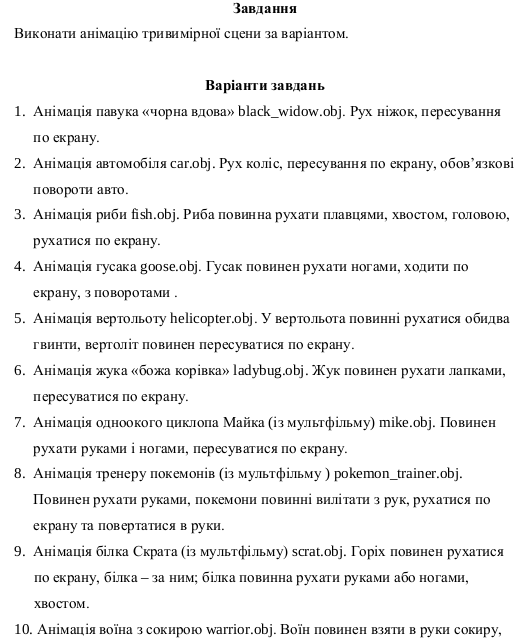
з дисципліни “Математичні та алгоритмічні основи комп’ютерної графіки”

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-81  Черняєв Іван  (*прізвище, ім’я, по батькові*)  варіант № 19 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

**Варіант 19**



**Лістинг коду програми**

package com.company;  
  
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.awt.\*;  
import java.util.Hashtable;  
import java.util.Enumeration;  
  
public class Main extends JFrame {  
  
 public Canvas3D myCanvas3D;  
  
 public Main() {  
 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("Scrat");  
 setSize(1000, 900);  
 getContentPane().add("Center", myCanvas3D);  
 setVisible(true);  
 }  
  
 Material getMaterialForBody() {  
 Material material = new Material();  
 material.setAmbientColor(new Color3f(new Color(0xC9DF983F, true)));  
 material.setDiffuseColor(new Color3f(new Color(0xC9DF983F, true)));  
 material.setSpecularColor(new Color3f(new Color(0xC9DF983F, true)));  
 material.setShininess(0.3f);  
 material.setLightingEnable(true);  
 return material;  
 }  
  
 Material getMaterialForEyes() {  
 Material material = new Material();  
 material.setAmbientColor(new Color3f(new Color(0x0FFFFFF, true)));  
 material.setDiffuseColor(new Color3f(new Color(0x0FFFFFF, true)));  
 material.setSpecularColor(new Color3f(new Color(0x0FFFFFF, true)));  
 material.setShininess(0.3f);  
 material.setLightingEnable(true);  
 return material;  
 }  
  
 Material getMaterialForBlack() {  
 Material material = new Material();  
 material.setAmbientColor(new Color3f(new Color(0xFF000000, true)));  
 material.setDiffuseColor(new Color3f(new Color(0xFF000000, true)));  
 material.setSpecularColor(new Color3f(new Color(0xFF000000, true)));  
 material.setShininess(0.3f);  
 material.setLightingEnable(true);  
 return material;  
 }  
  
 Material getMaterialForTail() {  
 Material material = new Material();  
 material.setAmbientColor(new Color3f(new Color(0xCC78776F, true)));  
 material.setDiffuseColor(new Color3f(new Color(0xCC78776F, true)));  
 material.setSpecularColor(new Color3f(new Color(0xCC78776F, true)));  
 material.setShininess(0.3f);  
 material.setLightingEnable(true);  
 return material;  
 }  
  
 Material getMaterialForNut() {  
 Material material = new Material();  
 material.setAmbientColor(new Color3f(new Color(0xCA9F4600, true)));  
 material.setDiffuseColor(new Color3f(new Color(0xCA9F4600, true)));  
 material.setSpecularColor(new Color3f(new Color(0xCA9F4600, true)));  
 material.setShininess(0.3f);  
 material.setLightingEnable(true);  
 return material;  
 }  
  
 Texture getTexture(String path) {  
 TextureLoader textureLoader = new TextureLoader(path, "LUMINANCE", myCanvas3D);  
 Texture texture = textureLoader.getTexture();  
 return texture;  
 }  
  
 private Appearance getForBody() {  
 Appearance appearance = new Appearance();  
 appearance.setTexture(getTexture("resources/body.jpg"));  
 TextureAttributes texAttr = new TextureAttributes();  
 texAttr.setTextureMode(TextureAttributes.*COMBINE*);  
 appearance.setTextureAttributes(texAttr);  
 appearance.setMaterial(getMaterialForBody());  
 return appearance;  
 }  
  
 private Appearance getForEyes() {  
 Appearance appearance = new Appearance();  
 appearance.setTexture(getTexture("resources/eyes.jpg"));  
 TextureAttributes texAttr = new TextureAttributes();  
 texAttr.setTextureMode(TextureAttributes.*COMBINE*);  
 appearance.setTextureAttributes(texAttr);  
 appearance.setMaterial(getMaterialForEyes());  
 return appearance;  
 }  
  
 private Appearance getForBlack() {  
 Appearance appearance = new Appearance();  
 appearance.setTexture(getTexture("resources/black.jpg"));  
 TextureAttributes texAttr = new TextureAttributes();  
 texAttr.setTextureMode(TextureAttributes.*COMBINE*);  
 appearance.setTextureAttributes(texAttr);  
 appearance.setMaterial(getMaterialForBlack());  
 return appearance;  
 }  
  
 private Appearance getForTail() {  
 Appearance appearance = new Appearance();  
 appearance.setTexture(getTexture("resources/tail.jpg"));  
 TextureAttributes texAttr = new TextureAttributes();  
 texAttr.setTextureMode(TextureAttributes.*COMBINE*);  
 appearance.setTextureAttributes(texAttr);  
 appearance.setMaterial(getMaterialForTail());  
 return appearance;  
 }  
  
 private Appearance getForNut() {  
 Appearance appearance = new Appearance();  
 appearance.setTexture(getTexture("resources/nut.jpg"));  
 TextureAttributes texAttr = new TextureAttributes();  
 texAttr.setTextureMode(TextureAttributes.*COMBINE*);  
 appearance.setTextureAttributes(texAttr);  
 appearance.setMaterial(getMaterialForNut());  
 return appearance;  
 }  
  
  
 public void createSceneGraph(SimpleUniverse su) {  
 ObjectFile f = new ObjectFile(ObjectFile.*RESIZE*);  
 BoundingSphere bs = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), Double.*MAX\_VALUE*);  
 String name;  
 BranchGroup trainerBranchGroup = new BranchGroup();  
 TextureLoader t = new TextureLoader("resources/bg.jpg", myCanvas3D);  
 Background trainerBackground = new Background(t.getImage());  
 trainerBackground.setImageScaleMode(Background.*SCALE\_FIT\_ALL*);  
  
 Scene trainerScene = null;  
 try {  
 trainerScene = f.load("resources/scrat.obj");  
 } catch (Exception e) {  
 System.*out*.println("File loading failed:" + e);  
 }  
 Hashtable roachNamedObjects = trainerScene.getNamedObjects();  
 Enumeration enumer = roachNamedObjects.keys();  
 while (enumer.hasMoreElements()) {  
 name = (String) enumer.nextElement();  
 System.*out*.println("Name: " + name);  
 }  
  
 // start animation  
 Transform3D startTransformation = new Transform3D();  
 startTransformation.setScale(2.0 / 6);  
 Transform3D combinedStartTransformation = new Transform3D();  
 combinedStartTransformation.mul(startTransformation);  
  
 TransformGroup scratStartTransformGroup = new TransformGroup(combinedStartTransformation);  
  
 // moves  
 int movesCount = 100; // moves count  
 int movesDuration = 700; // moves for 0,3 seconds  
 int startTime = 0; // launch animation after timeStart seconds  
  
 // head  
 Shape3D nose = (Shape3D) roachNamedObjects.get("nose");  
 nose.setAppearance(getForBlack());  
 TransformGroup headTG = new TransformGroup();  
  
 Shape3D noseCircles = (Shape3D) roachNamedObjects.get("objsphere12");  
 noseCircles.setAppearance(getForBlack());  
  
 Shape3D mouth = (Shape3D) roachNamedObjects.get("objobject07");  
 mouth.setAppearance(getForBody());  
  
 Shape3D eyeLeft = (Shape3D) roachNamedObjects.get("left\_eye");  
 eyeLeft.setAppearance(getForEyes());  
  
 Shape3D eyeRight = (Shape3D) roachNamedObjects.get("right\_eye");  
 eyeRight.setAppearance(getForEyes());  
  
 Shape3D eyesPoints = (Shape3D) roachNamedObjects.get("objsphere09");  
 eyesPoints.setAppearance(getForBlack());  
  
 Shape3D tongue = (Shape3D) roachNamedObjects.get("objobject06");  
 tongue.setAppearance(getForBody());  
  
 Shape3D mouth2 = (Shape3D) roachNamedObjects.get("objobject05");  
 mouth2.setAppearance(getForBody());  
  
 Shape3D body = (Shape3D) roachNamedObjects.get("body");  
 body.setAppearance(getForBody());  
  
  
 headTG.addChild(nose.cloneTree());  
 headTG.addChild(tongue.cloneTree());  
 headTG.addChild(mouth.cloneTree());  
 headTG.addChild(eyesPoints.cloneTree());  
 headTG.addChild(eyeLeft.cloneTree());  
 headTG.addChild(eyeRight.cloneTree());  
 headTG.addChild(noseCircles.cloneTree());  
 headTG.addChild(mouth2.cloneTree());  
 headTG.addChild(body.cloneTree());  
  
 Alpha bodyAlpha = new Alpha(movesCount, Alpha.*INCREASING\_ENABLE*, startTime, 0, movesDuration, 0, 0, 0, 0, 0);  
 Transform3D bodyRotAxis = new Transform3D();  
 bodyRotAxis.rotX(Math.*PI*);  
 RotationInterpolator bodyrot = new RotationInterpolator(bodyAlpha, headTG, bodyRotAxis, (float) -Math.*PI* / 6, (float) Math.*PI* / 6); // Math.PI\*2  
 bodyrot.setSchedulingBounds(bs);  
 headTG.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 headTG.addChild(bodyrot);  
 TransformGroup sceneGroup = new TransformGroup();  
 sceneGroup.addChild(headTG);  
  
  
 TransformGroup TailTG = new TransformGroup();  
 Shape3D tail = (Shape3D) roachNamedObjects.get("tale");  
 tail.setAppearance(getForTail());  
 TailTG.addChild(tail.cloneTree());  
  
 Alpha tailAlpha = new Alpha(movesCount, Alpha.*INCREASING\_ENABLE*, startTime, 0, 1000, 0, 0, 0, 0, 0);  
 Transform3D tailRotAxis = new Transform3D();  
 tailRotAxis.rotZ(Math.*PI* / 2);  
 RotationInterpolator tailrot = new RotationInterpolator(tailAlpha, TailTG, tailRotAxis, (float) -Math.*PI*, (float) Math.*PI*); // Math.PI\*2  
 tailrot.setSchedulingBounds(bs);  
 TailTG.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 TailTG.addChild(tailrot);  
 sceneGroup.addChild(TailTG);  
  
 TransformGroup LeftHandTG = new TransformGroup();  
 Shape3D leftHand = (Shape3D) roachNamedObjects.get("left\_hand");  
 leftHand.setAppearance(getForBody());  
 LeftHandTG.addChild(leftHand.cloneTree());  
  
 Alpha leftHandAlpha = new Alpha(movesCount, Alpha.*INCREASING\_ENABLE*, startTime, 0, 300, 0, 0, 0, 0, 0);  
 Transform3D leftHandRotAxis = new Transform3D();  
 leftHandRotAxis.rotX(Math.*PI* / 4);  
 RotationInterpolator leftHandrot = new RotationInterpolator(leftHandAlpha, LeftHandTG, leftHandRotAxis, (float) -Math.*PI* / 2f, (float) Math.*PI* / 2f); // Math.PI\*2  
 leftHandrot.setSchedulingBounds(bs);  
 LeftHandTG.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 LeftHandTG.addChild(leftHandrot);  
 sceneGroup.addChild(LeftHandTG);  
  
  
 TransformGroup RightHandTG = new TransformGroup();  
 Shape3D rightHand = (Shape3D) roachNamedObjects.get("right\_hand");  
 rightHand.setAppearance(getForBody());  
 RightHandTG.addChild(rightHand.cloneTree());  
  
 Alpha RightHandAlpha = new Alpha(movesCount, Alpha.*INCREASING\_ENABLE*, startTime, 0, 300, 0, 0, 0, 0, 0);  
 Transform3D RightHandRotAxis = new Transform3D();  
 RightHandRotAxis.rotX(-Math.*PI* / 4);  
 RotationInterpolator RightHandrot = new RotationInterpolator(RightHandAlpha, RightHandTG, RightHandRotAxis, (float) Math.*PI* / 2f, (float) -Math.*PI* / 2f); // Math.PI\*2  
 RightHandrot.setSchedulingBounds(bs);  
 RightHandTG.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 RightHandTG.addChild(RightHandrot);  
 sceneGroup.addChild(RightHandTG);  
  
  
 TransformGroup NutTG = new TransformGroup();  
 Shape3D Nut = (Shape3D) roachNamedObjects.get("nut");  
 Nut.setAppearance(getForNut());  
 NutTG.addChild(Nut.cloneTree());  
  
 Transform3D x = new Transform3D();  
 x.rotX(Math.*PI* / 2);  
 NutTG.setTransform(x);  
 TransformGroup NutRotGroup = translate(NutTG, new Vector3f(-0.5f, -0.3f, 0.0f));  
  
  
 TransformGroup whiteTransXformGroup = translate(  
 scratStartTransformGroup,  
 new Vector3f(0.0f, 0.0f, -0.7f));  
  
 TransformGroup whiteRotXformGroup = rotate(whiteTransXformGroup, new Alpha(10, 5000));  
 scratStartTransformGroup.addChild(sceneGroup);  
 scratStartTransformGroup.addChild(NutRotGroup);  
 trainerBranchGroup.addChild(whiteRotXformGroup);  
  
  
 BoundingSphere bounds = new BoundingSphere(new Point3d(120.0, 250.0, 100.0), Double.*MAX\_VALUE*);  
 trainerBackground.setApplicationBounds(bounds);  
 trainerBranchGroup.addChild(trainerBackground);  
  
 trainerBranchGroup.compile();  
 su.addBranchGraph(trainerBranchGroup);  
 }  
  
 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);  
 }  
  
 private TransformGroup translate(Node node, Vector3f vector) {  
  
 Transform3D transform3D = new Transform3D();  
 transform3D.setTranslation(vector);  
 TransformGroup transformGroup =  
 new TransformGroup();  
 transformGroup.setTransform(transform3D);  
  
 transformGroup.addChild(node);  
 return transformGroup;  
 }  
  
 private TransformGroup rotate(Node node, Alpha alpha) {  
 TransformGroup xformGroup = new TransformGroup();  
 xformGroup.setCapability(  
 TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
  
 RotationInterpolator interpolator =  
 new RotationInterpolator(alpha, xformGroup);  
  
 interpolator.setSchedulingBounds(new BoundingSphere(  
 new Point3d(0.0, 0.0, 0.0), 1.0));  
  
 xformGroup.addChild(interpolator);  
 xformGroup.addChild(node);  
  
 return xformGroup;  
 }  
  
  
 public static void main(String[] args) {  
 new Main();  
 }  
  
}

**Результат**

