

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-83  Палій Дмитро Володимирович  (*прізвище, ім’я, по батькові*)  варіант № 16 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

**Завдання**: Виконати анімацію тривимірної сцени за варіантом.

**Варіант:** 7. Анімація одноокого циклопа Майка (із мультфільму) mike.obj. Повинен рухати руками і ногами, пересуватися по екрану.

**Лістинг коду програми**

package sample;  
  
import javax.vecmath.Color3f;  
  
import com.sun.j3d.utils.behaviors.vp.OrbitBehavior;  
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.utils.image.TextureLoader;  
  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.util.Map;  
import javax.swing.Timer;  
import javax.swing.JFrame;  
  
class Main extends JFrame implements ActionListener {  
 static final String *mikeLocation* = "C:\\Users\\dpali\\Documents\\маокг\\lab6\\res\\mike.obj";  
 static final String *bgLocation* = "C:\\Users\\dpali\\Documents\\маокг\\lab6\\res\\bg.jpg";  
  
  
 private static final Color3f *SPECULAR\_LIGHT\_COLOR* = new Color3f(Color.*white*);  
 private static final Color3f *AMBIENT\_LIGHT\_COLOR* = new Color3f(Color.*white*);  
 private static final Color3f *EMISSIVE\_LIGHT\_COLOR* = new Color3f(Color.*black*);  
  
 private Transform3D translateTransform = new Transform3D();  
 private SimpleUniverse universe;  
 private Scene scene;  
 private BranchGroup root;  
 private TransformGroup mike = new TransformGroup();  
 private TransformGroup lLegTg = new TransformGroup();  
 private TransformGroup rLegTg = new TransformGroup();  
 private TransformGroup rHandTg = new TransformGroup();  
 private TransformGroup lHandTg = new TransformGroup();  
 private Canvas3D canvas;  
 private float xvel = 0.005f;  
 private float zvel = 0.01f;  
 private float xloc=0.0f;  
 private float yloc=0.0f;  
 private float zloc=-0.0f;  
  
 private float legvel = 0.1f;  
 private float handvel = 0.1f;  
  
 private float lhandpos = 0f;  
 private float rhandpos = 0f;  
 private float llegpos = 0f;  
 private float rlegpos = 0f;  
  
 public static void main(String[] args) {  
 Main window = new Main();  
 window.setVisible(true);  
 }  
  
 public Main() {  
 configureUniverse();  
 addLightToUniverse();  
 setBackground();  
 try {  
 loadScene();  
 }  
 catch (FileNotFoundException exception) {  
 System.*out*.println("file not found" + *mikeLocation*);  
 }  
 setInitPosition();  
  
 Timer timer = new Timer(20, this);  
 timer.start();  
  
 universe.addBranchGraph(root);  
 }  
  
 private void setInitPosition() {  
 var initPosition = new Vector3d(xloc, yloc, zloc);  
 translateTransform.setTranslation(initPosition);  
 translateTransform.setScale(0.1);  
 var rotation = new Transform3D();  
 var oldVector = new Vector3f(0f, 0f, 1f);  
 var newVector = new Vector3f(xvel, 0f, zvel);  
 rotation.rotY(newVector.angle(oldVector));  
 }  
  
 private void configureUniverse() {  
 canvas = new Canvas3D(SimpleUniverse.*getPreferredConfiguration*());  
 canvas.setDoubleBufferEnable(true);  
 canvas.setFocusable(true);  
 add(canvas, BorderLayout.*CENTER*);  
  
 setTitle("lab 6");  
 setSize(960, 540);  
 setResizable(true);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setExtendedState(JFrame.*MAXIMIZED\_BOTH*);  
  
 root = new BranchGroup();  
 universe = new SimpleUniverse(canvas);  
 universe.getViewingPlatform().setNominalViewingTransform();  
  
 OrbitBehavior ob = new OrbitBehavior(canvas);  
 ob.setSchedulingBounds(new BoundingSphere(new Point3d(0.0,0.0,0.0), Double.*MAX\_VALUE*));  
 ob.setRotYFactor(0);  
 universe.getViewingPlatform().setViewPlatformBehavior(ob);  
 }  
  
 private void addLightToUniverse(){  
 Bounds bounds = new BoundingSphere();  
 Color3f color = new Color3f(1f, 1f, 1f);  
 Vector3f lightDirection = new Vector3f(-1f,-1f,-1f);  
 DirectionalLight directionalLight = new DirectionalLight(color,lightDirection);  
 directionalLight.setInfluencingBounds(bounds);  
 root.addChild(directionalLight);  
 }  
  
 private void loadScene() throws FileNotFoundException {  
 ObjectFile loader = new ObjectFile(ObjectFile.*RESIZE*);  
 scene = loader.load(new FileReader(*mikeLocation*));  
  
 Map<String, Shape3D> nameMap = scene.getNamedObjects();  
  
 var sceneGroup = scene.getSceneGroup();  
  
 Appearance appearance = getAppearance(Color.*GREEN*);  
  
 var body = nameMap.get("monstr");  
 var rhand = nameMap.get("right\_hand");  
 var lhand = nameMap.get("left\_hand");  
 var rleg = nameMap.get("right\_leg");  
 var lleg = nameMap.get("left\_leg");  
  
 sceneGroup.removeChild(body);  
 sceneGroup.removeChild(rhand);  
 sceneGroup.removeChild(lhand);  
 sceneGroup.removeChild(rleg);  
 sceneGroup.removeChild(lleg);  
  
 body.setAppearance(appearance);  
 rhand.setAppearance(appearance);  
 lhand.setAppearance(appearance);  
 rleg.setAppearance(appearance);  
 lleg.setAppearance(appearance);  
  
 lLegTg.addChild(lleg);  
 lLegTg.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
  
 rLegTg.addChild(rleg);  
 rLegTg.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
  
 lHandTg.addChild(lhand);  
 lHandTg.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
  
 rHandTg.addChild(rhand);  
 rHandTg.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
  
 mike.addChild(body);  
 mike.addChild(lLegTg);  
 mike.addChild(rLegTg);  
 mike.addChild(lHandTg);  
 mike.addChild(rHandTg);  
 mike.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 root.addChild(mike);  
 }  
  
 private void setBackground() {  
 TextureLoader t = new TextureLoader(*bgLocation*, 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);  
 }  
  
 Appearance getAppearance(Color color) {  
 Appearance app = new Appearance();  
 app.setMaterial(getMaterial(color));  
 return app;  
 }  
  
 Material getMaterial(Color color) {  
 return new Material(*AMBIENT\_LIGHT\_COLOR*,  
 *EMISSIVE\_LIGHT\_COLOR*,  
 new Color3f(color),  
 *SPECULAR\_LIGHT\_COLOR*,  
 100f);  
 }  
  
 void listSceneNamedObjects(Scene scene) {  
 Map<String, Shape3D> nameMap = scene.getNamedObjects();  
  
 for (String name : nameMap.keySet()) {  
 System.*out*.printf("Name: %s\n", name);  
 }  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 var rotation = new Transform3D();  
 var oldVector = new Vector3f(xvel, 0, zvel);  
 var angleSign = 1;  
  
 if (Math.*abs*(zloc + zvel) >= 0.5) {  
 if (xvel \* zvel < 0) {  
 angleSign = -1;  
 } else {  
 angleSign = 1;  
 }  
  
 zvel \*= -1;  
 }  
 if (Math.*abs*(xloc + xvel) >= 0.5) {  
 if (xvel \* zvel > 0) {  
 angleSign = -1;  
 } else {  
 angleSign = 1;  
 }  
  
 xvel \*= -1;  
 }  
 xloc += xvel;  
 zloc += zvel;  
  
 var newVector = new Vector3f(xvel, 0, zvel);  
 var angle = angleSign \* newVector.angle(oldVector);  
  
 translateTransform.setTranslation(new Vector3f(xloc, yloc, zloc));  
 rotation.rotY(angle);  
 translateTransform.mul(rotation);  
 mike.setTransform(translateTransform);  
  
 if (Math.*abs*(lhandpos + handvel) >= Math.*PI* / 6) {  
 handvel \*= -1;  
 }  
 if (Math.*abs*(rlegpos + legvel) >= Math.*PI* / 6) {  
 legvel \*= -1;  
 }  
  
 lhandpos += handvel;  
 rhandpos -= handvel;  
 llegpos -= legvel;  
 rlegpos +=legvel;  
  
 var lhandRotation = new Transform3D();  
 lhandRotation.rotX(lhandpos);  
 lHandTg.setTransform(lhandRotation);  
  
 var rhandRotation = new Transform3D();  
 rhandRotation.rotX(rhandpos);  
 rHandTg.setTransform(rhandRotation);  
  
 var llegRotation = new Transform3D();  
 llegRotation.rotX(llegpos);  
 lLegTg.setTransform(llegRotation);  
  
 var rlegRotation = new Transform3D();  
 rlegRotation.rotX(rlegpos);  
 rLegTg.setTransform(rlegRotation);  
  
 }  
}

**Результат**





