

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  Ладуди Данила Володимировича  (*прізвище, ім’я, по батькові*)  варіант № 9 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2020

**Варіант завдання**

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

**Варіант: 9.** Анімація білка Скрата (із мультфільму) scrat.obj. Горіх повинен рухатися по екрану, білка – за ним; білка повинна рухати руками або ногами, хвостом

**Лістинг коду програми**

**MyAnimation.java**

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 MyAnimation implements ActionListener, KeyListener{  
  
 private TransformGroup wholePlane;  
 private Transform3D translateTransform;  
 private Transform3D rotateTransformX;  
 private Transform3D rotateTransformY;  
 private Transform3D rotateTransformZ;  
 private Transform3D scaleTransform;  
 private TransformGroup left\_hand;  
 private Transform3D left\_trans;  
 private TransformGroup right\_hand;  
 private Transform3D right\_trans;  
 private TransformGroup tail;  
 private Transform3D tail\_trans;  
  
 private JFrame mainFrame;  
  
 private float rot\_angle = 0.f;  
 private float sign=1.0f;  
 private float zoom=0.5f;  
 private float xloc=0.3f;  
 private float yloc=0.3f;  
 private float zloc=0.0f;  
 private int moveType=1;  
 private Timer timer;  
  
 public MyAnimation(TransformGroup wholePlane,Transform3D trans,  
 TransformGroup left\_hand,Transform3D left\_trans,  
 TransformGroup right\_hand, Transform3D right\_trans,  
 TransformGroup tail, Transform3D tail\_trans,  
 JFrame frame){  
 this.tail = tail;  
 this.tail\_trans = tail\_trans;  
 this.left\_hand = left\_hand;  
 this.left\_trans = left\_trans;  
 this.right\_hand = right\_hand;  
 this.right\_trans = right\_trans;  
 this.wholePlane=wholePlane;  
 this.translateTransform=trans;  
 this.mainFrame=frame;  
  
 rotateTransformX= new Transform3D();  
 rotateTransformY= new Transform3D();  
 rotateTransformZ= new Transform3D();  
  
 timer = new Timer(100, this);  
  
 timer.start();  
 }  
  
  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 // start timer when button is pressed  
  
 Move(moveType);  
 translateTransform.setScale(new Vector3d(zoom,zoom,zoom));  
 translateTransform.setRotation(new AxisAngle4d(0,yloc,0,rot\_angle ));  
 translateTransform.setTranslation(new Vector3f(xloc,yloc,zloc));  
 wholePlane.setTransform(translateTransform);  
  
 float newangle = rot\_angle\*3;  
 while(newangle<-2\*Math.*PI*){  
 newangle += 2\*Math.*PI*;  
 }  
  
 while(Math.*abs*(newangle)>Math.*PI*/3){  
 if (newangle>Math.*PI*/6){  
 newangle = (float) (Math.*PI*/6 - newangle);  
 }  
 if (newangle< - Math.*PI*/6){  
 newangle = (float) (-Math.*PI*/6 - newangle);  
 }  
 }  
 left\_trans.rotX(newangle);  
 left\_hand.setTransform(left\_trans);  
 right\_trans.rotX(-newangle);  
 right\_hand.setTransform(right\_trans);  
 tail\_trans.rotY(newangle);  
 tail.setTransform(tail\_trans);  
 }  
  
 private void Move(int mType) {  
 xloc = (float) Math.*sin*(rot\_angle);  
 yloc = 0.6f \* (float) (Math.*cos*(rot\_angle)-1);  
 zoom = ((float) (-Math.*cos*(rot\_angle)+1))/4f+0.5f;  
 rot\_angle -= 0.1;  
 if (rot\_angle>2\*Math.*PI*){  
 rot\_angle = 0f;  
 }  
 }  
  
 @Override  
 public void keyTyped(KeyEvent e) {  
 //Invoked when a key has been typed.  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 //Invoked when a key has been pressed.  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
 // Invoked when a key has been released.  
 }  
}

**Squirrel.java**

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;  
  
import javax.media.j3d.Material;  
  
import javax.media.j3d.Background;  
import javax.swing.JFrame;  
  
public class Squirrel extends JFrame {  
 static SimpleUniverse *universe*;  
 static Scene *scene*;  
 static Map<String, Shape3D> *nameMap*;  
 static BranchGroup *root*;  
 static Canvas3D *canvas*;  
  
 static TransformGroup *wholeModel*;  
 static Transform3D *transform3D*;  
 static TransformGroup *left\_hand*;  
 static Transform3D *left\_trans*;  
 static TransformGroup *right\_hand*;  
 static Transform3D *right\_trans*;  
 static TransformGroup *tail*;  
 static Transform3D *tail\_trans*;  
  
 public Squirrel() throws IOException {  
 configureWindow();  
 configureCanvas();  
 configureUniverse();  
 addModelToUniverse();  
 setModelElementsList();  
 addAppearance();  
 addImageBackground();  
 addLightToUniverse();  
 *root*.compile();  
 *universe*.addBranchGraph(*root*);  
 ChangeViewAngle();  
 }  
 private void configureWindow() {  
 setTitle("Animation Example");  
 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*("c:\\objects\\scrat.obj");  
 *root* = *scene*.getSceneGroup();  
 }  
  
 private void printModelElementsList(Map<String,Shape3D> nameMap){  
 for (String name : nameMap.keySet()) {  
 System.*out*.printf("Name: %s\n", name);}  
 }  
  
 private void setModelElementsList() {  
 *nameMap*=*scene*.getNamedObjects();  
 printModelElementsList(*nameMap*);  
 *tail* = new TransformGroup();  
 *wholeModel* = new TransformGroup();  
 *left\_hand* = new TransformGroup();  
 *root*.removeChild(*nameMap*.get("left\_hand"));  
 *root*.removeChild(*nameMap*.get("tale"));  
 *left\_hand*.addChild(*nameMap*.get("left\_hand"));  
 *right\_hand*= new TransformGroup();  
 *root*.removeChild(*nameMap*.get("right\_hand"));  
 *right\_hand*.addChild(*nameMap*.get("right\_hand"));  
 *transform3D* = new Transform3D();  
 *left\_trans* = new Transform3D();  
 *tail\_trans* = new Transform3D();  
 *left\_hand*.setTransform(*left\_trans*);  
 *right\_trans* = new Transform3D();  
 *right\_hand*.setTransform(*right\_trans*);  
 *transform3D*.setScale(new Vector3d(0.5,0.5,0.5));  
 *wholeModel*.setTransform(*transform3D*);  
 *tail*.addChild(*nameMap*.get("tale"));  
 for (Map.Entry<String, Shape3D> entry : *nameMap*.entrySet()) {  
 if (!entry.getKey().equals("left\_hand") && !entry.getKey().equals("right\_hand")&& !entry.getKey().equals("tale")){  
 *root*.removeChild(entry.getValue());  
 *wholeModel*.addChild(entry.getValue());  
 }  
 }  
 *wholeModel*.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 *left\_hand*.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 *right\_hand*.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 *tail*.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 *root*.addChild(*wholeModel*);  
 *wholeModel*.addChild(*left\_hand*);  
 *wholeModel*.addChild(*right\_hand*);  
 *wholeModel*.addChild(*tail*);  
 }  
 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, 0.0f, 0.0f ) );  
 return texture;  
 }  
  
 Material getMaterial() {  
 Material material = new Material();  
 material.setAmbientColor ( new Color3f( 3.f, 3.f, 3.f ) );  
 material.setDiffuseColor ( new Color3f( 5f, 4f, 3.f ) );  
 material.setSpecularColor( new Color3f( 5f, 3.f, 3.f ) );  
 material.setSpecularColor( new Color3f( 5f, 3.f, 3.f ) );  
 material.setShininess( 0.3f );  
 material.setLightingEnable(true);  
 return material;  
 }  
  
 Material getNutMaterial(){  
 Material material = new Material();  
 material.setAmbientColor ( new Color3f( 0.33f, 0.26f, 0.23f ) );  
 material.setDiffuseColor ( new Color3f( 0.50f, 0.11f, 0.00f ) );  
 material.setSpecularColor( new Color3f( 0.95f, 0.73f, 0.00f ) );  
 material.setShininess( 0.3f );  
 material.setLightingEnable(true);  
 return material;  
 }  
 private void addAppearance(){  
 Appearance bodyAppearance = new Appearance();  
 Appearance nutAppearence = new Appearance();  
 Texture t = getTexture("c:\\objects\\wood2.jpg");  
 bodyAppearance.setTexture(t);  
 nutAppearence.setTexture(t);  
 TextureAttributes texAttr = new TextureAttributes();  
 texAttr.setTextureMode(TextureAttributes.*COMBINE*);  
 bodyAppearance.setTextureAttributes(texAttr);  
 bodyAppearance.setMaterial(getMaterial());  
 nutAppearence.setTextureAttributes(texAttr);  
 nutAppearence.setMaterial(getNutMaterial());  
 for (Map.Entry<String, Shape3D> entry : *nameMap*.entrySet()) {  
 if (entry.getKey()=="nut"){  
 entry.getValue().setAppearance(nutAppearence);  
 }else{  
 entry.getValue().setAppearance(bodyAppearance);  
 }  
 }  
  
 Shape3D nut = *nameMap*.get("nut");  
 nut.setAppearance(nutAppearence);  
  
 }  
  
 private void addImageBackground(){  
 TextureLoader t = new TextureLoader("c:\\objects\\mountains.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 addLightToUniverse(){  
 Bounds bounds = new BoundingSphere();  
 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);  
 }  
 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 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);  
 }  
 public static void main(String[]args){  
 try {  
 Squirrel window = new Squirrel();  
 MyAnimation planeMovement = new MyAnimation(*wholeModel*, *transform3D*,  
 *left\_hand*, *left\_trans*, *right\_hand*, *right\_trans*,*tail*, *tail\_trans*, window);  
 window.addKeyListener(planeMovement);  
 window.setVisible(true);  
 }  
 catch (IOException ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 }  
}

**Результат**

****