

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 №** **5**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-83  Линь Дмитро Іванович  варіант № 10 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною |

Київ 2021

**Варіант завдання**

**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.

**Обраний варіант даних:**

3D-модель бджоли на фоні квіткового поля.

**Лістинг коду програми**

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.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.net.URL;  
import java.util.Map;  
  
public class Main extends JFrame implements ActionListener, KeyListener {  
 private final static String *backgroundLocation* = "flowers.jpg";  
 private final static String *modelLocation* = "BEE.OBJ";  
 private final BranchGroup root = new BranchGroup();  
 private final Canvas3D canvas = new Canvas3D(SimpleUniverse.getPreferredConfiguration());  
 private final TransformGroup trexGroup = new TransformGroup();  
 private final Transform3D transform3D = new Transform3D();  
 private final Transform3D rotateTransformX = new Transform3D();  
 private final Transform3D rotateTransformY = new Transform3D();  
 private final Transform3D rotateTransformZ = new Transform3D();  
 private final ClassLoader classLoader = Thread.*currentThread*().getContextClassLoader();  
 private SimpleUniverse universe;  
 private Scene trex;  
 private Background background;  
 private Map<String, Shape3D> nameMap;  
  
 private float x\_location\_current = -3;  
 private float y\_location\_current = 0;  
 private float z\_location\_current = 0;  
 private float scale\_cur = 1;  
  
  
 public static void main(String[] args) {  
 try {  
 Main window = new Main();  
 window.addKeyListener(window);  
 window.setVisible(true);  
 } catch (IOException e) {  
 System.*err*.println(e.getMessage());  
 }  
 }  
  
 public Main() throws IOException {  
 initialize();  
 addTexture();  
 addImageBackground();  
 addLight();  
 setInitialViewAngle();  
 setInitialLocation();  
 root.compile();  
 universe.addBranchGraph(root);  
 }  
  
 private void setInitialLocation() {  
 transform3D.setTranslation(new Vector3f(x\_location\_current, 0, 0));  
 transform3D.setScale(scale\_cur);  
 trexGroup.setTransform(transform3D);  
 }  
  
 private void initialize() throws IOException {  
 // window settings  
 setTitle("Lab #5");  
 setSize(1000, 700);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 canvas.setDoubleBufferEnable(true);  
 getContentPane().add(canvas, BorderLayout.*CENTER*);  
  
 universe = new SimpleUniverse(canvas);  
 universe.getViewingPlatform().setNominalViewingTransform();  
  
 canvas.addKeyListener(this);  
 trex = getSceneFromFile();  
 }  
  
 private void addLight() {  
 DirectionalLight dirLight = new DirectionalLight(  
 new Color3f(Color.*WHITE*),  
 new Vector3f(4.0f, -7.0f, -12.0f)  
 );  
  
 dirLight.setInfluencingBounds(new BoundingSphere(new Point3d(), 1000));  
 root.addChild(dirLight);  
  
 AmbientLight ambientLight = new AmbientLight(new Color3f(Color.*WHITE*));  
 DirectionalLight directionalLight = new DirectionalLight(  
 new Color3f(Color.*BLACK*),  
 new Vector3f(-1F, -1F, -1F)  
 );  
 BoundingSphere influenceRegion = new BoundingSphere(new Point3d(), 1000);  
 ambientLight.setInfluencingBounds(influenceRegion);  
 directionalLight.setInfluencingBounds(influenceRegion);  
 root.addChild(ambientLight);  
 root.addChild(directionalLight);  
 }  
  
 private TextureLoader getTextureLoader(String path) throws IOException {  
 URL textureResource = classLoader.getResource(path);  
 if (textureResource == null) {  
 throw new IOException("Couldn't find texture: " + path);  
 }  
 return new TextureLoader(textureResource.getPath(), canvas);  
 }  
  
 private Material getMaterial() {  
 Material material = new Material();  
 material.setAmbientColor(new Color3f(new Color(243, 242, 221)));  
 material.setDiffuseColor(new Color3f(new Color(255, 233, 207)));  
 material.setSpecularColor(new Color3f(new Color(255, 203, 195)));  
 material.setLightingEnable(true);  
 return material;  
 }  
  
 private void addTexture() throws IOException {  
 nameMap = trex.getNamedObjects();  
 trexGroup.addChild(trex.getSceneGroup());  
 trexGroup.setCapability(TransformGroup.*ALLOW\_TRANSFORM\_WRITE*);  
 root.addChild(trexGroup);  
  
 }  
  
 private void addImageBackground() throws IOException {  
 background = new Background(getTextureLoader(*backgroundLocation*).getImage());  
 background.setImageScaleMode(Background.*SCALE\_FIT\_MAX*);  
 background.setApplicationBounds(new BoundingSphere(new Point3d(),1000));  
 background.setCapability(Background.*ALLOW\_IMAGE\_WRITE*);  
 root.addChild(background);  
 }  
  
 private void setInitialViewAngle() {  
 ViewingPlatform vp = universe.getViewingPlatform();  
 Transform3D transform = new Transform3D();  
 transform.lookAt(  
 new Point3d(-3.985f, -0.07f, 0),  
 new Point3d(-0.7, 0, 0),  
 new Vector3d(0, 1, 0)  
 );  
 transform.invert();  
 vp.getViewPlatformTransform().setTransform(transform);  
 }  
  
 private Scene getSceneFromFile() throws IOException {  
 ObjectFile file = new ObjectFile(ObjectFile.RESIZE);  
 file.setFlags(ObjectFile.RESIZE | ObjectFile.TRIANGULATE | ObjectFile.STRIPIFY);  
 InputStream inputStream = classLoader.getResourceAsStream(*modelLocation*);  
 if (inputStream == null) {  
 throw new IOException("Resource " + *modelLocation* + " not found");  
 }  
 return file.load(new BufferedReader(new InputStreamReader(inputStream)));  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 System.*out*.println(e.getKeyCode());  
 int keyCode = e.getKeyCode();  
 float diff = 0.05f;  
  
 switch (keyCode) {  
 case KeyEvent.*VK\_UP*: {  
 x\_location\_current += 0.05;  
 scale\_cur -= 0.005;  
 transform3D.setTranslation(new Vector3f(x\_location\_current, y\_location\_current, z\_location\_current));  
 transform3D.setScale(scale\_cur);  
 trexGroup.setTransform(transform3D);  
 } break;  
 case KeyEvent.*VK\_DOWN*: {  
 x\_location\_current -= 0.05;  
 scale\_cur += 0.005;  
 transform3D.setTranslation(new Vector3f(x\_location\_current, y\_location\_current, z\_location\_current));  
 transform3D.setScale(scale\_cur);  
 trexGroup.setTransform(transform3D);  
 } break;  
 case KeyEvent.*VK\_LEFT*: {  
 z\_location\_current -= 0.05;  
 transform3D.setTranslation(new Vector3f(x\_location\_current, y\_location\_current, z\_location\_current));  
 trexGroup.setTransform(transform3D);  
 } break;  
 case KeyEvent.*VK\_RIGHT*: {  
 z\_location\_current += 0.05;  
 transform3D.setTranslation(new Vector3f(x\_location\_current, y\_location\_current, z\_location\_current));  
 trexGroup.setTransform(transform3D);  
 } break;  
 case KeyEvent.*VK\_X*: {  
 rotateTransformX.rotX(diff);  
 transform3D.mul(rotateTransformX);  
 trexGroup.setTransform(transform3D);  
 } break;  
 case KeyEvent.*VK\_Y*: {  
 rotateTransformY.rotY(diff);  
 transform3D.mul(rotateTransformY);  
 trexGroup.setTransform(transform3D);  
 } break;  
 case KeyEvent.*VK\_Z*: {  
 rotateTransformZ.rotZ(diff);  
 transform3D.mul(rotateTransformZ);  
 trexGroup.setTransform(transform3D);  
 } break;  
 }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) { }  
  
 @Override  
 public void keyTyped(KeyEvent e) { }  
  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
  
 }  
}

**Результат у директорії разом з цим документом**