

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 №** **4**

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

**Тема:** “Побудова найпростіших тривимірних об'єктів за допомогою бібліотеки Java3D та їх анімація“

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-83  Василець Данило Андрійович  (*прізвище, ім’я, по батькові*)  варіант № 2 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

**Завдання**:

За допомогою засобів, що надає бібліотека Java3D, побудувати тривимірний об’єкт. Для цього скористатися основними примітивами, що буде доцільно використовувати згідно варіанту: сфера, конус, паралелепіпед, циліндр. Об'єкт має складатися з 5-15 примітивів. Задати матеріал кожного примітиву, в разі необхідності накласти текстуру. В сцені має бути мінімум одне джерело освітлення.

Виконати анімацію сцени таким чином, щоб можна було розглянути об'єкт з усіх сторін. За бажанням можна виконати інтерактивні взаємодію з об'єктом за допомогою миші та клавіатури.

**Варіант: 2**

Казковий палац

**Лістинг коду програми**

**Castle.java**

**import** com.sun.j3d.utils.geometry.Box;

**import** com.sun.j3d.utils.geometry.Cone;

**import** com.sun.j3d.utils.geometry.Cylinder;

**import** com.sun.j3d.utils.geometry.Primitive;

**import** com.sun.j3d.utils.image.TextureLoader;

**import** javax.media.j3d.\*;

**import** javax.vecmath.Color3f;

**import** javax.vecmath.Color4f;

**import** javax.vecmath.Point3d;

**import** javax.vecmath.Vector3f;

**import** java.awt.\*;

**public** **class** Castle  {

**private** TransformGroup TransformGroup;

**private** Transform3D TransformGroup3D =  **new** Transform3D();

**private** **float** angle =  0;

**public** BranchGroup createSceneGraph() {

        BranchGroup objRoot =  **new** BranchGroup();

        TransformGroup =  **new** TransformGroup();

        TransformGroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);

        buildCastle();

        objRoot.addChild(TransformGroup);

        BoundingSphere bounds = **new** BoundingSphere(**new** Point3d(0.0, 0.0, 0.0),100.0);

        Color3f light1Color =  **new** Color3f( 1.5f, 1.5f, 1.5f);

        Vector3f light1Direction =  **new** Vector3f( 0.0f, -2.0f, 3.0f);

        DirectionalLight light1 =  **new** DirectionalLight(light1Color, light1Direction);

        light1.setInfluencingBounds(bounds);

        objRoot.addChild(light1);

        Color3f ambientColor = **new** Color3f(1.0f, 1.0f, 1.0f);

        AmbientLight ambientLightNode = **new** AmbientLight(ambientColor);

        ambientLightNode.setInfluencingBounds(bounds);

        objRoot.addChild(ambientLightNode);

**return** objRoot;

    }

**private** **void** buildCastle(){

*//------TOWER-----------------*

        TextureLoader loader = **new** TextureLoader("./source/brick.jpg", "LUMINANCE", **new** Container());

        Texture texture = loader.getTexture();

        texture.setBoundaryModeS(Texture.WRAP);

        texture.setBoundaryModeT(Texture.WRAP);

        texture.setBoundaryColor(**new** Color4f(0.0f, .5f, .5f, 0.0f));

        TextureAttributes texAttr = **new** TextureAttributes();

        texAttr.setTextureMode(TextureAttributes.MODULATE);

        Appearance ap = **new** Appearance();

        ap.setTexture(texture);

        ap.setTextureAttributes(texAttr);

        Color3f emissive = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f ambient = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f diffuse = **new** Color3f(0.2f, 0.2f, 0.2f);

        Color3f specular = **new** Color3f(0.3f, 0.3f, 0.3f);

        ap.setMaterial(**new** Material(emissive, ambient, diffuse, specular,1.0f));

**int** primflags =  Primitive.GENERATE\_NORMALS +  Primitive.GENERATE\_TEXTURE\_COORDS;

        TransformGroup tg = **new** TransformGroup();

        Transform3D transform = **new** Transform3D();

        Cylinder cyl = **new** Cylinder(.38f, .75f, primflags, ap);

        Vector3f vector = **new** Vector3f(0.0f, -0.2f, .0f);

        transform.setTranslation(vector);

        tg.setTransform(transform);

        tg.addChild(cyl);

        TransformGroup.addChild(tg);

*//-----TOWER FOUNDATION----------*

        Appearance apX = **new** Appearance();

        apX.setTexture(texture);

        apX.setTextureAttributes(texAttr);

        Color3f emissiveX = **new** Color3f(0.0f, 0.00f, 0.00f);

        Color3f ambientX = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f diffuseX = **new** Color3f(0.3f, 0.f, .0f);

        Color3f specularX = **new** Color3f(0.1f, 0.0f, 0.0f);

        apX.setMaterial(**new** Material(ambientX, emissiveX, diffuseX, specularX, 1.0f));

**int** primflags2 =  Primitive.GENERATE\_NORMALS +  Primitive.GENERATE\_TEXTURE\_COORDS;

        TransformGroup tgX = **new** TransformGroup();

        Transform3D transformX = **new** Transform3D();

        Box boxX = **new** Box(0.4f, 0.2f,0.4f,primflags2,apX);

        Vector3f vectorX = **new** Vector3f(0.0f, -0.6f, .0f);

        transformX.setTranslation(vectorX);

        tgX.setTransform(transformX);

        tgX.addChild(boxX);

        TransformGroup.addChild(tgX);

*//------ROOF------------------------------*

        TextureLoader loaderR = **new** TextureLoader("./source/roof.jpg", "LUMINANCE", **new** Container());

        Texture textureR = loaderR.getTexture();

        texture.setBoundaryModeS(Texture.WRAP);

        texture.setBoundaryModeT(Texture.WRAP);

        texture.setBoundaryColor(**new** Color4f(0.6f, .1f, .1f, 0.5f));

        TextureAttributes texAttrR = **new** TextureAttributes();

        texAttr.setTextureMode(TextureAttributes.MODULATE);

        Appearance ap2 = **new** Appearance();

        ap2.setTexture(textureR);

        ap2.setTextureAttributes(texAttrR);

        Color3f emissive2 = **new** Color3f(0.0f, 0.00f, 0.00f);

        Color3f ambient2 = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f diffuse2 = **new** Color3f(0.3f, 0.f, .0f);

        Color3f specular2 = **new** Color3f(0.6f, 0.1f, 0.1f);

        ap2.setMaterial(**new** Material(ambient2, emissive2, diffuse2, specular2, .0f));

**int** primflagsR =  Primitive.GENERATE\_NORMALS +  Primitive.GENERATE\_TEXTURE\_COORDS;

        TransformGroup tg2 = **new** TransformGroup();

        Transform3D transform2 = **new** Transform3D();

        Cone con = **new** Cone(.5f, .5f, primflagsR, ap2);

        Vector3f vector2 = **new** Vector3f(0.0f, 0.45f, .0f);

        transform2.setTranslation(vector2);

        tg2.setTransform(transform2);

        tg2.addChild(con);

        TransformGroup.addChild(tg2);

*//--------WINDOWS---------------------------------*

        Appearance ap3 = **new** Appearance();

        Color3f emissive3 = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f ambient3 = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f diffuse3 = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f specular3 = **new** Color3f(0.3f, 0.13f, 0.64f);

        ap3.setMaterial(**new** Material(ambient3, emissive3, diffuse3, specular3, 1.0f));

**int** primflags3 =  Primitive.GENERATE\_NORMALS +  Primitive.GENERATE\_TEXTURE\_COORDS;

        TransformGroup tg3 = **new** TransformGroup();

        Transform3D transform3 = **new** Transform3D();

        Box box =  **new** Box(0.05f, 0.06f, 0.0f, primflags3, ap3);

        Vector3f vector3 = **new** Vector3f(0.0f, 0.0f, .38f);

        transform3.setTranslation(vector3);

        tg3.setTransform(transform3);

        tg3.addChild(box);

        TransformGroup.addChild(tg3);

        TransformGroup tg5 = **new** TransformGroup();

        Transform3D transform5 = **new** Transform3D();

        Box box2 =  **new** Box(0.05f, 0.06f, 0.0f, primflags3, ap3);

        Vector3f vector5 = **new** Vector3f(0.0f, .0f, -0.38f);

        transform5.setTranslation(vector5);

        tg5.setTransform(transform5);

        tg5.addChild(box2);

        TransformGroup.addChild(tg5);

*//--------DOOR----------------------------*

        TextureLoader loader2 = **new** TextureLoader("./source/door.jpg", "LUMINANCE", **new** Container());

        Texture texture2 = loader2.getTexture();

        texture.setBoundaryModeS(Texture.WRAP);

        texture.setBoundaryModeT(Texture.WRAP);

        texture.setBoundaryColor(**new** Color4f(0.0f, .5f, .5f, 0.0f));

        TextureAttributes texAttr2 = **new** TextureAttributes();

        texAttr.setTextureMode(TextureAttributes.MODULATE);

        Appearance ap4 = **new** Appearance();

        ap4.setTexture(texture2);

        ap4.setTextureAttributes(texAttr2);

        Color3f emissive4 = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f ambient4 = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f diffuse4 = **new** Color3f(0.0f, 0.0f, 0.0f);

        Color3f specular4 = **new** Color3f(.1f, 0.1f, 0.1f);

        ap4.setMaterial(**new** Material(ambient4, emissive4, diffuse4, specular4, 1.0f));

**int** primflags4 =  Primitive.GENERATE\_NORMALS +  Primitive.GENERATE\_TEXTURE\_COORDS;

        TransformGroup tg4 = **new** TransformGroup();

        Transform3D transform4 = **new** Transform3D();

        Box box1 =  **new** Box(0.15f, 0.13f, 0.0f, primflags4, ap4);

        Vector3f vector4 = **new** Vector3f(.0f, -0.663f, -.41f);

        transform4.setTranslation(vector4);

        tg4.setTransform(transform4);

        tg4.addChild(box1);

        TransformGroup.addChild(tg4);

    }

**public** **void** rotate() {

        TransformGroup3D.rotY(angle);

        angle += 0.1;

        TransformGroup.setTransform(TransformGroup3D);

    }

}

**Main.java**

**import** com.sun.j3d.utils.universe.SimpleUniverse;

**import** javax.media.j3d.BranchGroup;

**import** javax.media.j3d.Canvas3D;

**import** javax.media.j3d.Transform3D;

**import** javax.swing.\*;

**import** javax.vecmath.Point3d;

**import** javax.vecmath.Vector3d;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.KeyEvent;

**import** java.awt.event.KeyListener;

**public** **class** Main **extends** JFrame **implements** ActionListener, KeyListener {

    Castle Castle;

**boolean** rotate = **false**;

**public** Main() {

*// Frame title*

**super**("Lab4 - Castle");

        Castle = **new** Castle();

        Canvas3D canvas3D = **new** Canvas3D(SimpleUniverse.getPreferredConfiguration());

        add(canvas3D);

        canvas3D.addKeyListener(**this**);

        Timer timer = **new** Timer(50, **this**);

        timer.start();

        BranchGroup scene = Castle.createSceneGraph();

        SimpleUniverse u = **new** SimpleUniverse(canvas3D);

        Transform3D move = lookTowardsOriginFrom(**new** Point3d(0, 0, -3));

        u.getViewingPlatform().getViewPlatformTransform().setTransform(move);

*// u.getViewingPlatform().setNominalViewingTransform();*

        u.addBranchGraph(scene);

        setSize(600, 600);

        setLocationRelativeTo(**null**);

        setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

        setVisible(**true**);

    }

**public** **static** **void** main(String[] args) {

**new** Main();

    }

**public** Transform3D lookTowardsOriginFrom(Point3d point)

    {

        Transform3D move = **new** Transform3D();

        Vector3d up = **new** Vector3d(point.x, point.y + 1, point.z);

        move.lookAt(point, **new** Point3d(0.0d, 0.0d, 0.0d), up);

**return** move;

    }

    @Override

**public** **void** actionPerformed(ActionEvent e) {

**if**(rotate) {

            Castle.rotate();

        }

    }

    @Override

**public** **void** keyTyped(KeyEvent keyEvent) {

    }

    @Override

**public** **void** keyPressed(KeyEvent keyEvent) {

**if**(keyEvent.getKeyCode() == KeyEvent.VK\_RIGHT) {

            rotate = **true**;

        }

    }

    @Override

**public** **void** keyReleased(KeyEvent keyEvent) {

**if**(keyEvent.getKeyCode() == KeyEvent.VK\_SPACE) {

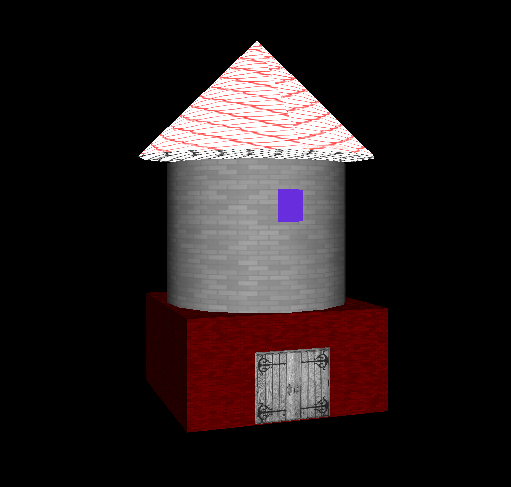
            rotate = **false**;

        }

    }

}

**Результат**

****