

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“

|  |  |  |  |
| --- | --- | --- | --- |
| Виконала  студентка III курсу  групи КП-83  Дзьоба Уляна Олегівна  (*прізвище, ім’я, по батькові*)  варіант № 5 |  |  | 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:**

Кубік Рубіка

**Лістинг коду програми**

**Cube.java**

package lab4;

import com.sun.j3d.utils.applet.MainFrame;

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

import javax.media.j3d.\*;

import javax.swing.\*;

import javax.vecmath.\*;

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Cube extends Applet implements ActionListener {

private final TransformGroup cubeTransformGroup = new TransformGroup();

private final Transform3D cubeTransform3dx = new Transform3D();

private final Transform3D cubeTransform3dy = new Transform3D();

private final Transform3D cubeTransform3dz = new Transform3D();

private final Transform3D cubeTransform3d = new Transform3D();

private final Timer timer = new Timer(50, this);

private float angle = 0;

private float angleX = 0;

private float angleY = 0;

private double scale = 0;

private boolean rotateY = true;

private boolean isDecreasing = false;

public static void main(String[] args) {

Cube obj = new Cube();

MainFrame mf = new MainFrame(obj, 700, 700);

mf.run();

}

private Cube() {

setLayout(new BorderLayout());

GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();

Canvas3D c = new Canvas3D(config);

add("Center", c);

SimpleUniverse universe = new SimpleUniverse(c);

timer.start();

universe.getViewingPlatform().setNominalViewingTransform();

universe.addBranchGraph(createSceneGraph());

}

private BranchGroup createSceneGraph() {

BranchGroup root = new BranchGroup();

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

root.addChild(cubeTransformGroup);

Color3f colorRed = new Color3f(new Color(100, 0, 15));

Color3f colorOrange = new Color3f(new Color(100, 20, 0));

Color3f colorYellow = new Color3f(new Color(100, 60, 0));

Color3f colorGreen = new Color3f(new Color(0,45,30));

Color3f colorBlue = new Color3f(new Color(0,0,80));

Color3f colorWhite = new Color3f(new Color(105,105,105));

Color3f colorGray = new Color3f(new Color(10,10,10));

float indent = 0.307f;

float firstX = -0.3f;

float firstY = -0.3f;

float firstZ = 0.3f;

createElement(firstX, firstY, firstZ, colorGreen, colorGray, colorGray, colorOrange, colorGray, colorBlue);

createElement(firstX+indent, firstY, firstZ, colorGreen, colorGray, colorGray, colorGray, colorGray, colorBlue);

createElement(firstX+indent, firstY+indent, firstZ-2\*indent, colorGray, colorYellow, colorGray, colorGray, colorGray, colorGray);

createElement(firstX+indent, firstY+2\*indent, firstZ-2\*indent, colorGray, colorYellow, colorGray, colorGray, colorWhite, colorGray);

createElement(firstX+indent, firstY+2\*indent, firstZ-indent, colorGray, colorGray, colorGray, colorGray, colorWhite, colorGray);

createElement(firstX+indent, firstY+2\*indent, firstZ, colorGreen, colorGray, colorGray, colorGray, colorWhite, colorGray);//

createElement(firstX+indent, firstY+indent, firstZ, colorGreen, colorGray, colorGray, colorGray, colorGray, colorGray);//

createElement(firstX+2\*indent, firstY, firstZ, colorGreen, colorGray, colorRed, colorGray, colorGray, colorBlue);

createElement(firstX+2\*indent, firstY+indent, firstZ, colorGreen, colorGray, colorRed, colorGray, colorGray, colorGray);

createElement(firstX+2\*indent, firstY+2\*indent, firstZ, colorGreen, colorGray, colorRed, colorGray, colorWhite, colorGray);

createElement(firstX+2\*indent, firstY+indent, firstZ-indent, colorGray, colorGray, colorRed, colorGray, colorGray, colorGray);

createElement(firstX+2\*indent, firstY+2\*indent, firstZ-indent, colorGray, colorGray, colorRed, colorGray, colorWhite, colorGray);

createElement(firstX+2\*indent, firstY+indent, firstZ-2\*indent, colorGray, colorYellow, colorRed, colorGray, colorGray, colorGray);

createElement(firstX+2\*indent, firstY+2\*indent, firstZ-2\*indent, colorGray, colorYellow, colorRed, colorGray, colorWhite, colorGray);

createElement(firstX, firstY, firstZ-indent, colorGray, colorGray, colorGray, colorOrange, colorGray, colorBlue);

createElement(firstX+indent, firstY, firstZ-indent, colorGray, colorGray, colorGray, colorGray, colorGray, colorBlue);

createElement(firstX+2\*indent, firstY, firstZ-indent, colorGray, colorGray, colorRed, colorGray, colorGray, colorBlue);

createElement(firstX, firstY, firstZ-2\*indent, colorGray, colorYellow, colorGray, colorOrange, colorGray, colorBlue);

createElement(firstX+indent, firstY, firstZ-2\*indent, colorGray, colorYellow, colorGray, colorGray, colorGray, colorBlue);

createElement(firstX+2\*indent, firstY, firstZ-2\*indent, colorGray, colorYellow, colorRed, colorGray, colorGray, colorBlue);

createElement(firstX, firstY+indent, firstZ, colorGreen, colorGray, colorGray, colorOrange, colorGray, colorGray);

createElement(firstX, firstY+2\*indent, firstZ, colorGreen, colorGray, colorGray, colorOrange, colorWhite, colorGray);

createElement(firstX, firstY+indent, firstZ-indent, colorGray, colorGray, colorGray, colorOrange, colorGray, colorGray);

createElement(firstX, firstY+2\*indent, firstZ-indent, colorGray, colorGray, colorGray, colorOrange, colorWhite, colorGray);

createElement(firstX, firstY+indent, firstZ-2\*indent, colorGray, colorYellow, colorGray, colorOrange, colorGray, colorGray);

createElement(firstX, firstY+2\*indent, firstZ-2\*indent, colorGray, colorYellow, colorGray, colorOrange, colorWhite, colorGray);

// light section start

BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0),100);

//Color sunLightColor = new Color(242, 255, 0);

Color sunLightColor = new Color(255, 255, 255);

DirectionalLight lightDirect = new DirectionalLight(new Color3f(sunLightColor), new Vector3f(4.0f, -7.0f, -12.0f));

lightDirect.setInfluencingBounds(bounds);

root.addChild(lightDirect);

AmbientLight ambientLightNode = new AmbientLight(new Color3f(new Color(255, 255, 255)));

ambientLightNode.setInfluencingBounds(bounds);

root.addChild(ambientLightNode);

// light section end

return root;

}

private void createElement(float x, float y, float z, Color3f clrFront, Color3f clrBack, Color3f clrRight, Color3f clrLeft, Color3f clrTop, Color3f clrBottom) {

TransformGroup tg = new TransformGroup();

Transform3D transform = new Transform3D();

Node cube = CubeParts.getElement(0.15f, clrFront, clrBack, clrRight, clrLeft, clrTop, clrBottom);

Vector3f vector = new Vector3f(x, y, z);

transform.setTranslation(vector);

tg.setTransform(transform);

tg.addChild(cube);

cubeTransformGroup.addChild(tg);

}

@Override

public void actionPerformed(ActionEvent e) {

cubeTransform3dx.rotX(angleX);

cubeTransform3dy.rotY(angleY);

cubeTransform3d.mul(cubeTransform3dx, cubeTransform3dy);

angleX += 0.05;

angleY += 0.05;

angle += 0.05;

if (angle < 6.283) {

angleX = 0;

}

if (angle > 6.283 && angle < 12.566) {

angleY = 0;

}

cubeTransformGroup.setTransform(cubeTransform3d);

}

}

**CubeParts.java**

package lab4;

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

import javax.media.j3d.\*;

import javax.vecmath.\*;

import java.awt.\*;

public class CubeParts {

private static final float[] verts = {

// front face

1.0f, -1.0f, 1.0f,

1.0f, 1.0f, 1.0f,

-1.0f, 1.0f, 1.0f,

-1.0f, -1.0f, 1.0f,

// back face

-1.0f, -1.0f, -1.0f,

-1.0f, 1.0f, -1.0f,

1.0f, 1.0f, -1.0f,

1.0f, -1.0f, -1.0f,

// right face

1.0f, -1.0f, -1.0f,

1.0f, 1.0f, -1.0f,

1.0f, 1.0f, 1.0f,

1.0f, -1.0f, 1.0f,

// left face

-1.0f, -1.0f, 1.0f,

-1.0f, 1.0f, 1.0f,

-1.0f, 1.0f, -1.0f,

-1.0f, -1.0f, -1.0f,

// top face

1.0f, 1.0f, 1.0f,

1.0f, 1.0f, -1.0f,

-1.0f, 1.0f, -1.0f,

-1.0f, 1.0f, 1.0f,

// bottom face

-1.0f, -1.0f, 1.0f,

-1.0f, -1.0f, -1.0f,

1.0f, -1.0f, -1.0f,

1.0f, -1.0f, 1.0f,

};

private static Shape3D createShape(double scale, Color3f clrFront, Color3f clrBack, Color3f clrRight, Color3f clrLeft, Color3f clrTop, Color3f clrBottom) {

final Color3f[] colors = new Color3f[]{clrFront, clrFront, clrFront, clrFront,

clrBack, clrBack, clrBack, clrBack,

clrRight, clrRight, clrRight, clrRight,

clrLeft, clrLeft, clrLeft, clrLeft,

clrTop, clrTop, clrTop, clrTop,

clrBottom, clrBottom, clrBottom, clrBottom};

QuadArray cube = new QuadArray(24, QuadArray.COORDINATES |

QuadArray.COLOR\_3);

float scaledVerts[] = new float[verts.length];

for (int i = 0; i < verts.length; i++)

scaledVerts[i] = verts[i] \* (float)scale;

cube.setCoordinates(0, scaledVerts);

cube.setColors(0, colors);

final GeometryInfo gi = new GeometryInfo(cube);

//also generate normal vectors so that the surface can be light

final NormalGenerator normalGenerator = new NormalGenerator();

normalGenerator.generateNormals(gi);

final GeometryArray geometryArray = gi.getGeometryArray();

return new Shape3D(geometryArray);

}

public static Node getElement(float size, Color3f clrFront, Color3f clrBack, Color3f clrRight, Color3f clrLeft, Color3f clrTop, Color3f clrBottom) {

final Shape3D child = createShape(size, clrFront, clrBack, clrRight, clrLeft, clrTop, clrBottom);

final Appearance fillAppNode = getElementAppearence();

final PolygonAttributes pAtt = new PolygonAttributes();

// avoid trouble with clockwise faces

pAtt.setCullFace(PolygonAttributes.CULL\_NONE);

pAtt.setBackFaceNormalFlip(true);

fillAppNode.setPolygonAttributes(pAtt);

child.setAppearance(fillAppNode);

return child;

}

public static Appearance getElementAppearence() {

Appearance ap = new Appearance();

Color3f specular = new Color3f(new Color(0,0, 0));

Material material = new Material();

material.setSpecularColor(specular);

material.setShininess(1.0f);

material.setColorTarget(Material.AMBIENT\_AND\_DIFFUSE);

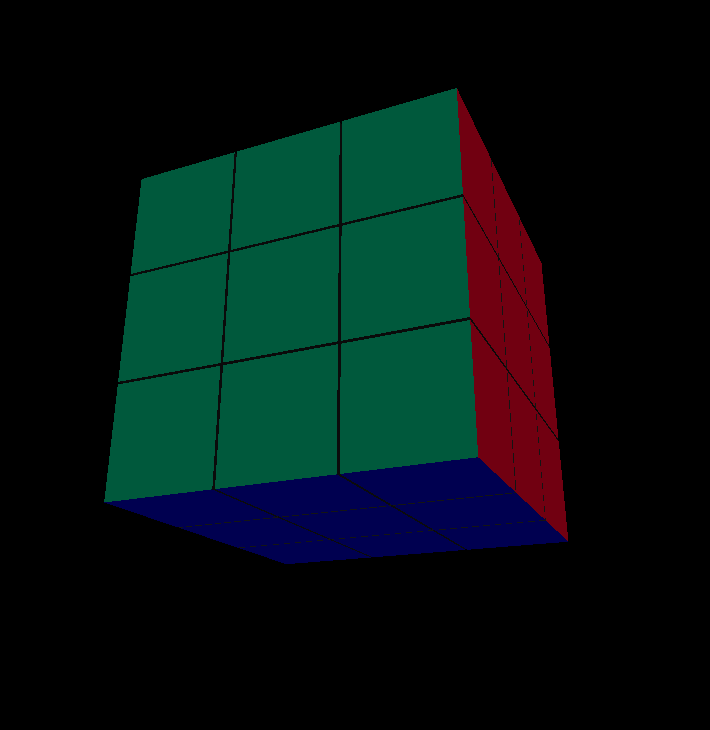
ap.setMaterial(material);

return ap;

}

}

**Результат**

****