Лабораторна робота №4

Тема: «Побудова найпростіших тривимірних об'єктів за допомогою бібліотеки Java3D та їх анімація»
Мета:
1) вивчення стандартних засобів Java3D для візуалізація зображення;
2) вивчення засобів анімації примітивів та складених об'єктів в Java3D.

Теоретичні відомості

Java 3D – це доповнення до мови програмування Java для відображення тривимірної графіки. Програми, написані за допомогою Java 3D можуть бути запущені на різних платформах персональних комп'ютерів, а також доступні для перегляду в мережі Інтернет за допомогою різних браузерів.

Java 3D – це бібліотека, що надає інтерфейс для роботи з комп'ютерною графікою. Вона є простішою за більшість існуючих графічних бібліотек, але має достатньо можливостей для створення високоякісних ігор та анімації. Java 3D побудована на існуючих технологіях, таких як DirectX та OpenGL, тому є досить швидкою. Також Java 3D має можливості для роботи з об'єктами, що створені в таких пакетах для 3D моделювання як TrueSpace, VRML та інших.

Для роботи з бібліотекою необхідно встановити її з сайту розробника:

http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-client-419417.html#java3d-1.5.1-oth-JPR

Після цього необхідно підключити її в середовище розробки (наприклад NetBeans або Eclipse) як зовнішню бібліотеку користувача.

Детальна інструкція:

http://www.cs.utexas.edu/~scottm/cs324e/handouts/setUpJava3dEclipse.htm

Після цього можна переходити до створення програм.

Основи Java3D

Будь-яка тривимірна сцена складається з таких основних складових: власне простору, об'єктів, що розташовані у сцені, джерела освітлення та точки спостереження. Відповідно до цього в кожній графічній бібліотеці є відповідний інструментарій для створення сцен.

Оскільки Java – об’єктно-орієнтована мова, то всі компоненти сцени є об’єктами. Промальовка сцени відбувається в конструкторі головного класу, що викликається в методі `main`. Для створення простого простору використовується клас `SimpleUniverse`. В Java 3D сцена може містити декілька груп об’єктів, скомпонованих програмістом за його бажанням. Для створення групи використовується клас `BranchGroup`. До групи додаються об’єкти, що беруть участь у сцені. Усі графічні примітиви в Java 3D спадкуються від класу Primitive, зокрема використаний у наступному прикладі клас `ColorCube`.

Для того, щоб встановити точку перегляду, треба отримати об’єкт класу `ViewingPlatform`, що належить об’єкту простору, а потім встановити йому бажану точку перегляду, в даному випадку `NominalViewingTransform`, яка значить перегляд фронтальної проекції в такому обсязі, щоб повністю охопити координати X від -1.0 до 1.0.

Найпростіша програма, що додає у сцену куб:

```
import javax.media.j3d.BranchGroup;
import com.sun.j3d.utils.geometry.ColorCube;
import com.sun.j3d.utils.universe.SimpleUniverse;

public class RedCube {

	public static void main(String[] args) {
		new RedCube();
	}

	public RedCube()

	{
		// створюємо простір, в якому будемо працювати
		SimpleUniverse universe = new SimpleUniverse();
		// створюємо групу, в яку додаємо об'єкти для відображення
		BranchGroup group = new BranchGroup();
		// додаємо в групу куб зі стороною, що дорівнює 0.3 від штрини вікна
		group.addChild(new ColorCube(0.3));
		// встановлюємо точку перегляду за замовченням
		universe.getViewingPlatform().setNominalViewingTransform();
		// додаємо створену групу у простір
		universe.addBranchGraph(group);
	}
}
```

Результат роботи програми (рис. 1):

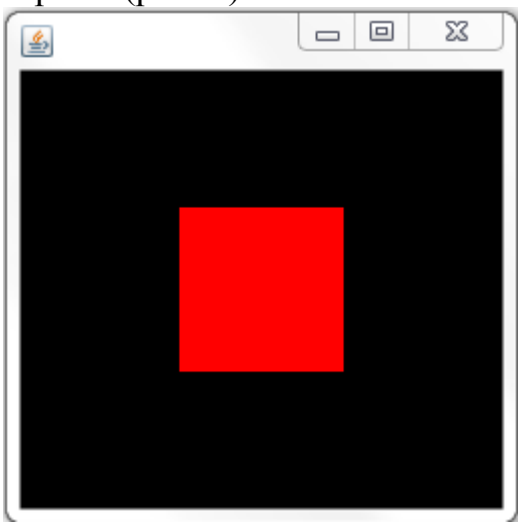

Рис. 1

Оскільки було встановлено точку перегляду за замовченням, то ми бачимо куб у фронтальній проекції, фактично двовимірним.

Робота зі світлом

Для того щоб сцена виглядала реалістично, а об’єкти мали властивості, що притаманні об’єктам реального світу – об’єм, колір, матеріал – в сцену додається освітлення.

Джерело освітлення має колір, що задається за допомогою об’єкту класу `Color3f`, має сферу поширення, тобто простір, в середині якого при розрахунку сцени воно буде враховуватись, задається за допомогою об’єкту класу `BoundingSphere`, а також в залежності від типу джерела освітлення певні інші характеристики, такі як розмір і напрям. Напрям поширення світла задається за допомогою об’єкта класу `Vector3f`. Після визначення вказаних характеристик створюється об’єкт класу `DirectionalLight`, що представляє собою джерело направленого світла.

```
import javax.media.j3d.BoundingSphere;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.DirectionalLight;
import javax.vecmath.Color3f;
import javax.vecmath.Point3d;
import javax.vecmath.Vector3f;

import com.sun.j3d.utils.geometry.Sphere;
import com.sun.j3d.utils.universe.SimpleUniverse;
```

```
public class Lighting {

	public static void main(String[] args) {
		new Lighting();
	}

	public Lighting()

	{
		SimpleUniverse universe = new SimpleUniverse();
		BranchGroup group = new BranchGroup();
		Sphere sphere = new Sphere(0.5f);
		group.addChild(sphere);

		// Створюємо зелене світло, яке  світить з відстані 100м від об'єкту
		Color3f light1Color = new Color3f(0.8f, 1.1f, 0.1f); // параметри конструктору
- це відповідно червона, зелена та синя компоненти кольору
		BoundingSphere bounds =   new BoundingSphere(new Point3d(0.0, 0.0, 0.0),
100.0); // вказуємо сферу, внутрішній простір якої буде освітлено
		Vector3f light1Direction = new Vector3f(4.0f, -7.0f, -12.0f); //встановлюємо
вектор, що задає напрям освітлення
		DirectionalLight light1 = new DirectionalLight(light1Color,
	light1Direction); //створюмо власне об'єкт освітлення
		light1.setInfluencingBounds(bounds); //вказуємо, яка частина сцени має бути
освітлена
		group.addChild(light1); //додаємо освітлення ло сцени

		universe.getViewingPlatform().setNominalViewingTransform();
		universe.addBranchGraph(group);
	}
}
```

Результат роботи програми (рис. 2) :

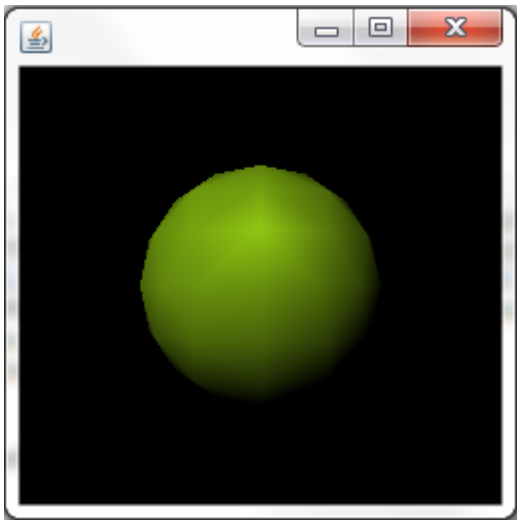

Рис. 2

Трансформації

Кожен об’єкт у просторі має своє положення. За замовчанням об’єкти додаються у початок координат, тобто мають позицію (0,0,0). З будь-яким об’єктом, що знаходиться у просторі, можна виконувати такі основні трансформації: переміщення на заданий вектор, поворот на заданий кут відносно точки або осі, та масштабування.

В Java3D кожна трансформація виконується над окремою групою об’єктів. Група може містити від одного об’єкту, причому об’єкти, що містяться в групу теж можуть бути групою. Для об’єднання об’єктів у групи існує клас `TransformGroup`. До кожної групи можна застосувати 3Д трансформацію, що представлена об’єктом `Transform3D`. Для нього визначені методи `setTranslation`, `setRotation`, `setScale` з різними параметрами.

```
import javax.media.j3d.BoundingSphere;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.DirectionalLight;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import javax.vecmath.Color3f;
import javax.vecmath.Point3d;
import javax.vecmath.Vector3f;

import com.sun.j3d.utils.geometry.Cone;
import com.sun.j3d.utils.geometry.Cylinder;
import com.sun.j3d.utils.geometry.Sphere;
import com.sun.j3d.utils.universe.SimpleUniverse;

public class Position {

	public static void main(String[] args) {
		new Position();
	}

	public Position()

	{
		SimpleUniverse universe = new SimpleUniverse();
		BranchGroup group = new BranchGroup();

		// створюємо вісь X за допомогою сфер
		for (float x = -1.0f; x <= 1.0f; x = x + 0.1f)
		{
			// створюємо 3д об'єкт (сферу)
			Sphere sphere = new Sphere(0.05f);
			// створюємо групу трансформації
			TransformGroup tg = new TransformGroup();
			// створюємо 3д трансформацію
			Transform3D transform = new Transform3D();
```

```
                // створюємо вектор, на яких буде здійснене перенесення
                Vector3f vector = new Vector3f(x, .0f, .0f);
                // задаємо трансформації властивість перенесення на вказаний вектор
                transform.setTranslation(vector);
                // вказуємо створену трансформацію заданій групі трансформації
                tg.setTransform(transform);
                // вказуємо об'єкт, що потрібно трансформувати
                tg.addChild(sphere);
                // додаємо групу трансформації до сцени (сам об'єкт додавати уже не
потрібно!)
                group.addChild(tg);
            }

            // вісь У за допомогою конусів
            for (float y = -1.0f; y <= 1.0f; y = y + 0.1f)
            {
                TransformGroup tg = new TransformGroup();
                Transform3D transform = new Transform3D();
                Cone cone = new Cone(0.05f, 0.1f);
                Vector3f vector = new Vector3f(.0f, y, .0f);
                transform.setTranslation(vector);
                tg.setTransform(transform);
                tg.addChild(cone);
                group.addChild(tg);
            }

            // вісь Z за допомогою циліндрів
            for (float z = -1.0f; z <= 1.0f; z = z + 0.1f)
            {
                TransformGroup tg = new TransformGroup();
                Transform3D transform = new Transform3D();
                Cylinder cylinder = new Cylinder(0.05f, 0.1f);
                Vector3f vector = new Vector3f(.0f, .0f, z);
                transform.setTranslation(vector);
                tg.setTransform(transform);
                tg.addChild(cylinder);
                group.addChild(tg);
            }

            // встановлюємо жовте освітлення
            Color3f light1Color = new Color3f(1.7f, 1.6f, .0f);
            BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
            Vector3f light1Direction = new Vector3f(4.0f, -7.0f, -12.0f);
            DirectionalLight light1 = new DirectionalLight(light1Color, light1Direction);
            light1.setInfluencingBounds(bounds);
            group.addChild(light1);

            // встановлюємо точку перегляду та вставляємо групу об'єктів у сцену
            universe.getViewingPlatform().setNominalViewingTransform();
            universe.addBranchGraph(group);
        }
}
```

Результат роботи (рис. 3):

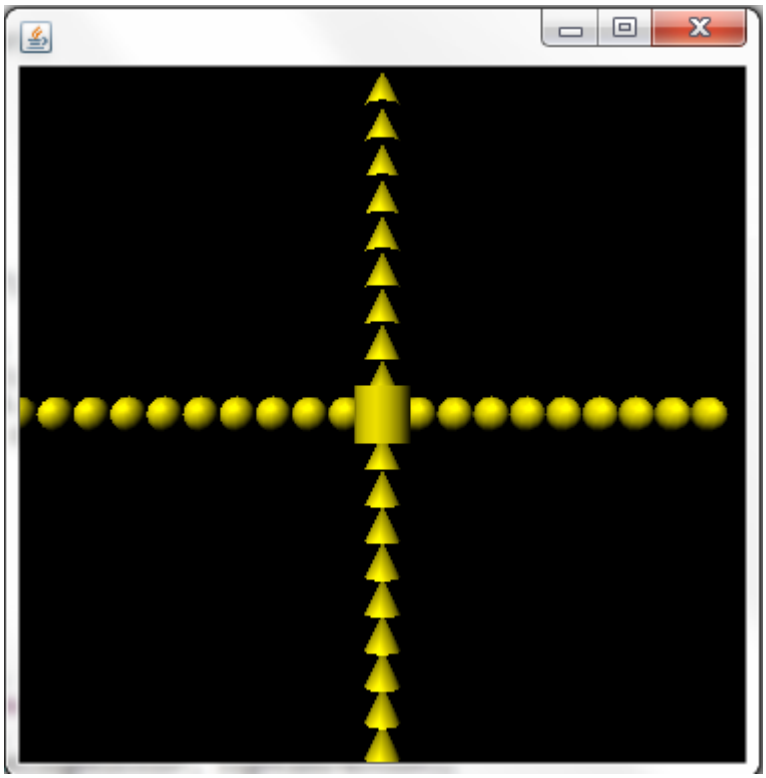

Рис. 3

Матеріал, колір, текстури

Кожен об’єкт реального світу має колір, створений з певного матеріалу та часто має певне забарвлення – текстуру. Для забезпечення цих властивостей використовується об’єкт класу `Appearance`. Матеріал об’єкту визначає, як о’єкт взаємодіє зі світлом – які компоненти відбиваються, які поглинаються, які випромінюються. Тому для визначення матеріалу потрібно задати такі компоненти світла відповідно по червоному, зеленому та синьому кольорам:

- Emissive – світло, що випромінюється
- Ambient – навколишнє світло
- Diffuse – світло, що розсіюється
- Specular – світло, що відбивається

Ці компоненти задаються об’єкту класу `Material`.

Для накладання текстури використовується клас `Texture` та допоміжний `TextureLoader`. Для коректного накладання текстури треба визначити режим

накладання, а також вказати Java, що треба побудувати нормалі та згенерувати текстурні координати.

```
import java.awt.Container;

import javax.media.j3d.AmbientLight;
import javax.media.j3d.Appearance;
import javax.media.j3d.BoundingSphere;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.DirectionalLight;
import javax.media.j3d.Material;
import javax.media.j3d.Texture;
import javax.media.j3d.TextureAttributes;
import javax.vecmath.Color3f;
import javax.vecmath.Color4f;
import javax.vecmath.Point3d;
import javax.vecmath.Vector3f;

import com.sun.j3d.utils.geometry.Primitive;
import com.sun.j3d.utils.geometry.Sphere;
import com.sun.j3d.utils.image.TextureLoader;
import com.sun.j3d.utils.universe.SimpleUniverse;

public class AppearenceSphere {

	public static void main(String[] args) {
		new AppearenceSphere();
	}

	public AppearenceSphere()

	{
		SimpleUniverse universe = new SimpleUniverse();
		BranchGroup group = new BranchGroup();

		Color3f emissive = new Color3f(0.0f, 0.0f, 0.0f);
		Color3f ambient = new Color3f(0.9f, .15f, .15f);
		Color3f diffuse = new Color3f(0.7f, .15f, .15f);
		Color3f specular = new Color3f(0.0f, 0.0f, 0.0f);

		// завантажуємо текстуру
		TextureLoader loader = new TextureLoader("C:\\2.jpg", "LUMINANCE", new
Container());
		Texture texture = loader.getTexture();
		// задаємо властивості границі текстури
		texture.setBoundaryModeS(Texture.WRAP);
		texture.setBoundaryModeT(Texture.WRAP);
		texture.setBoundaryColor(new Color4f(0.0f, 1.0f, 1.0f, 0.0f));

		// встановлюємо атрибути текстури
		// може бути REPLACE, BLEND або DECAL замість MODULATE
		TextureAttributes texAttr = new TextureAttributes();
		texAttr.setTextureMode(TextureAttributes.MODULATE);

		// створюємо новий вигляд
		Appearance ap = new Appearance();
```

```
            // додаємо до вигляду текстуру
            ap.setTexture(texture);
            ap.setTextureAttributes(texAttr);

            // встановлюємо матеріал
            ap.setMaterial(new Material(ambient, emissive, diffuse, specular, 1.0f));

            // створюємо флаги, що вказують, що для об'єкту треба згенерувати нормалі та
текстурні кординати
            int primflags = Primitive.GENERATE_NORMALS + Primitive.GENERATE_TEXTURE_COORDS;

            // створюємо кулю, на яку будемо накладати текстуру та матеріал
            Sphere sphere = new Sphere(0.5f, primflags, ap);
            group.addChild(sphere);

            // створюємо біле світло
            Color3f light1Color = new Color3f(1f, 1f, 1f);
            BoundingSphere bounds =    new BoundingSphere(new Point3d(0.0, 0.0, 0.0),
100.0);
            Vector3f light1Direction = new Vector3f(4.0f, -7.0f, -12.0f);
            DirectionalLight light1    = new DirectionalLight(light1Color, light1Direction);
            light1.setInfluencingBounds(bounds);
            group.addChild(light1);

            // створюємо ненаправлене світло
            AmbientLight ambientLight = new AmbientLight(new Color3f(.5f, .5f, .5f));
            ambientLight.setInfluencingBounds(bounds);
            group.addChild(ambientLight);

            universe.getViewingPlatform().setNominalViewingTransform();
            universe.addBranchGraph(group);
        }
}
```

Результат роботи програми (рис. 4):

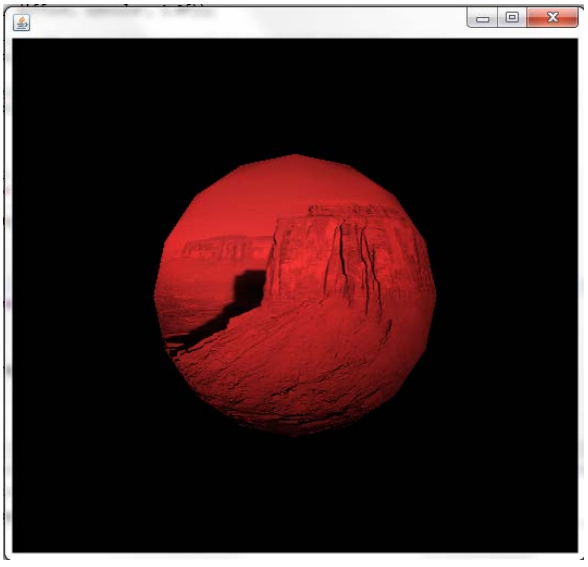

Рис. 4

Анімація та взаємодія з інтерфейсом користувача

Анімація будь-якої сцени відбувається шляхом зміни певних властивостей об’єктів за певний проміжок часу. Це може бути зміна кольору, розмірів, позиції об’єктів, як абсолютної, так і відносно інших об’єктів.

Крім того, за допомогою властивостей мови Java можна керувати об’єктами, що знаходяться в сцені. Для цого використовується бібліотека `java.awt або javax.swing`.

Наступний приклад показує як запустити анімацію кульки за допомогою кнопки миші, та як з клавіатури можна змінювати її позицію на екрані.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.awt.event.WindowAdapter;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import javax.vecmath.*;
import com.sun.j3d.utils.geometry.Sphere;
import javax.swing.Timer;

public class BouncingBall extends Applet implements ActionListener, KeyListener {

	private Button go = new Button("Go");
	private TransformGroup objTrans;
	private Transform3D trans = new Transform3D();
	private float height = 0.0f;
	private float sign = 1.0f; // going up or down
	private Timer timer;
	private float xloc = 0.0f;

	public static void main(String[] args) {
		System.out.println("Program Started");
		BouncingBall bb = new BouncingBall();
		bb.addKeyListener(bb);
		//MainFrame mf = new MainFrame(bb, 256, 256);
	}

	public BranchGroup createSceneGraph() {

		// створюємо групу об'єктів
		BranchGroup objRoot = new BranchGroup();

		// створюємо об'єкт, що будемо додавати до групи
		Sphere sphere = new Sphere(0.25f);
		objTrans = new TransformGroup();
		objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
		Transform3D pos1 = new Transform3D();
		pos1.setTranslation(new Vector3f(0.0f, 0.0f, 0.0f));
		objTrans.setTransform(pos1);
```

```
            objTrans.addChild(sphere);
            objRoot.addChild(objTrans);

            //налаштовуємо освітлення
            BoundingSphere bounds =   new BoundingSphere(new Point3d(0.0, 0.0, 0.0),
100.0);
            Color3f light1Color = new Color3f(1.0f, 0.0f, 0.2f);
            Vector3f light1Direction = new Vector3f(4.0f, -7.0f, -12.0f);
            DirectionalLight light1   = new DirectionalLight(light1Color, light1Direction);
            light1.setInfluencingBounds(bounds);
            objRoot.addChild(light1);

            // встановлюємо навколишнє освітлення
            Color3f ambientColor = new Color3f(1.0f, 1.0f, 1.0f);
            AmbientLight ambientLightNode = new AmbientLight(ambientColor);
            ambientLightNode.setInfluencingBounds(bounds);
            objRoot.addChild(ambientLightNode);
            return objRoot;

      }

      public BouncingBall() {

            //налаштовуємо вікно
            setLayout(new BorderLayout());
            GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
            Canvas3D c = new Canvas3D(config);
            // розміщуємо сцену в центрі фрейму
            add("Center", c);
            // підписуємо об'єкт поточного класу на подію натиснення кнопки клавіатури
            c.addKeyListener(this);
            // створюємо об'єкт таймеру з інтервалом 100 мілісекунд та підписуємо
            timer = new Timer(100, this);
            Panel p = new Panel();
            p.add(go);
            add("North", p);
            go.addActionListener(this);
            go.addKeyListener(this);

            // створюємо просту сцену та додаємо її до простору
            BranchGroup scene = createSceneGraph();
            SimpleUniverse u = new SimpleUniverse(c);
            u.getViewingPlatform().setNominalViewingTransform();
            u.addBranchGraph(scene);

      }

      public void keyPressed(KeyEvent e) {

            // дія по натисненню на клавішу
            if (e.getKeyChar() == 's') {
                  xloc = xloc + .1f;
            }
            if (e.getKeyChar() == 'a') {
                  xloc = xloc - .1f;
            }
      }
```

```
	public void actionPerformed(ActionEvent e) {

		// запустити таймер по натисненню на клавішу
		if (e.getSource() == go) {
			if (!timer.isRunning()) {
				timer.start();
			}
		}
		else {
			height += .1 * sign;
			if (Math.abs(height * 2) >= 1)
				sign = -1.0f * sign;
			if (height < -0.4f) {
				trans.setScale(new Vector3d(1.0, .8, 1.0));
			}
			else {
				trans.setScale(new Vector3d(1.0, 1.0, 1.0));
			}
			trans.setTranslation(new Vector3f(xloc, height, 0.0f));
			objTrans.setTransform(trans);
		}
	}

	@Override
	public void keyReleased(KeyEvent arg0) {
		// TODO Auto-generated method stub

	}

	@Override
	public void keyTyped(KeyEvent arg0) {
		// TODO Auto-generated method stub

	}
}
```

Результат роботи програми (рис. 5):

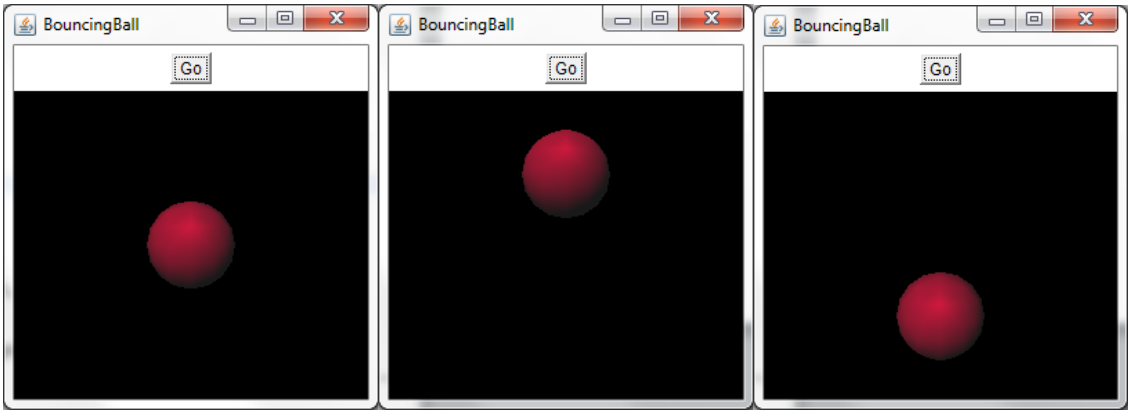

Рис. 5

Створення складного об’єкту з примітивів

В сцені дуже рідко знаходиться один простий об’єкт, який являє собою певний примітив. Більшість об’єктів реального світу є складними, але велика частина їх може бути створена за допомогою примітивів. Для створення складного об’єкту за допомогою примітивів треба виконати його декомпозицію за такими параметрами:

- виділити геометричні примітиви – сферу, циліндр, паралелепіпид, конус тощо
- з’ясувати з якого матеріалу створений той чи інший примітив
- з’ясувати, чи є якісь примітиви текстурованими
- об’єднати примітиви логічні групи – наприклад доцільним є об’єднувати у групу об’єкти, що є статичними відносно один одного
- за необхідності утворені групи також об’єднати

За допомогою трансформацій примітиви позиціонуються в потрібному місці, анімація застосовується уже не до окремих примітивів, а до логічних груп або всього об’єкту.

Далі наведено приклад створення складеного об’єкту – новорічної ялинки, що анімовано як обертання навколо вертикальної осі.

```
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.image.TextureLoader;
import com.sun.j3d.utils.universe.SimpleUniverse;

import javax.media.j3d.*;
import javax.swing.Timer;
import javax.vecmath.*;

public class ComplexObject implements ActionListener {

	private TransformGroup treeTransformGroup;
	private Transform3D treeTransform3D = new Transform3D();
	private Timer timer;
	private float angle = 0;

	public static void main(String[] args) {
		new ComplexObject();
	}
```

```
public ComplexObject() {
	timer = new Timer(50, this);
	timer.start();
	BranchGroup scene = createSceneGraph();
	SimpleUniverse u = new SimpleUniverse();
	u.getViewingPlatform().setNominalViewingTransform();
	u.addBranchGraph(scene);
}

public BranchGroup createSceneGraph() {

	// створюємо групу об'єктів
	BranchGroup objRoot = new BranchGroup();

	// створюємо об'єкт, що будемо додавати до групи
	treeTransformGroup = new TransformGroup();
	treeTransformGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
	buildTree();
	objRoot.addChild(treeTransformGroup);

	// налаштовуємо освітлення
	BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0),100.0);
	Color3f light1Color = new Color3f(1.0f, 0.5f, 0.4f);
	Vector3f light1Direction = new Vector3f(4.0f, -7.0f, -12.0f);
	DirectionalLight light1 = new DirectionalLight(light1Color,
light1Direction);
	light1.setInfluencingBounds(bounds);
	objRoot.addChild(light1);

	// встановлюємо навколишнє освітлення
	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 buildTree() {
	// створюємо гілки ялинки
	TransformGroup tgTop = new TransformGroup();
	Transform3D transformTop = new Transform3D();
	Cone coneTop = XMassTree.getCone(0.3f, 0.2f);
	Vector3f vectorTop = new Vector3f(.0f, 0.2f, .0f);
	transformTop.setTranslation(vectorTop);
	tgTop.setTransform(transformTop);
	tgTop.addChild(coneTop);
	treeTransformGroup.addChild(tgTop);

	TransformGroup tgMiddle = new TransformGroup();
	Transform3D transformMiddle = new Transform3D();
	Cone coneMiddle = XMassTree.getCone(0.3f, 0.25f);
	Vector3f vectorMiddle = new Vector3f(.0f, .0f, .0f);
	transformMiddle.setTranslation(vectorMiddle);
	tgMiddle.setTransform(transformMiddle);
	tgMiddle.addChild(coneMiddle);
	treeTransformGroup.addChild(tgMiddle);
```

```
TransformGroup tgBottom = new TransformGroup();
Transform3D transformBottom = new Transform3D();
Cone coneBottom = XMassTree.getCone(0.3f, 0.3f);
Vector3f vectorBottom = new Vector3f(.0f, -0.2f, .0f);
transformBottom.setTranslation(vectorBottom);
tgBottom.setTransform(transformBottom);
tgBottom.addChild(coneBottom);
treeTransformGroup.addChild(tgBottom);

// прикрашаємо її кульками
createBall(0.03f, .0f, 0.35f, .0f, "", new Color3f(0.5f, 0.5f, 0.0f));

// верхній ярус
createBall(0.02f, 0.07f, 0.15f, 0.12f, "", new Color3f(0.0f, 0.0f, 1.0f));
createBall(0.02f, -0.12f, 0.13f, 0.1f, "", new Color3f(0.2f, 0.5f, 0.5f));
createBall(0.02f, 0.13f, 0.12f, 0.1f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, 0.11f, 0.2f, 0.0f, "", new Color3f(0.2f, 1.2f, 1.5f));

createBall(0.02f, -0.07f, 0.15f, -0.12f, "", new Color3f(0.0f, 0.0f, 1.0f));
createBall(0.02f, 0.12f, 0.13f, -0.1f, "", new Color3f(0.2f, 0.5f, 0.5f));
createBall(0.02f, -0.13f, 0.12f, -0.1f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, -0.11f, 0.2f, -0.0f, "", new Color3f(0.2f, 1.2f, 1.5f));


// середній ярус
createBall(0.02f, -0.02f, 0.2f, 0.1f, "", new Color3f(0.5f, 0.0f, 0.5f));
createBall(0.02f, 0.12f, -0.05f, 0.12f, "", new Color3f(0.0f, 0.5f, 0.5f));
createBall(0.02f, -0.12f, -0.03f, 0.11f, "", new Color3f(0.2f, 0.2f, 0.5f));
createBall(0.02f, -0.05f, -0.1f, 0.2f, "", new Color3f(0.2f, 0.2f, 0.5f));
createBall(0.02f, 0.05f, -0.02f, 0.13f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, 0.22f, -0.1f, 0.0f, "", new Color3f(1.2f, 1.2f, 0.5f));
createBall(0.02f, 0.12f, .0f, 0.05f, "", new Color3f(0.2f, 0.2f, 1.5f));

createBall(0.02f, 0.02f, 0.2f, -0.1f, "", new Color3f(0.5f, 1.0f, 0.5f));
createBall(0.02f, -0.12f, -0.05f, -0.12f, "", new Color3f(0.0f, 0.5f, 0.5f));
createBall(0.02f, 0.12f, -0.03f, -0.11f, "", new Color3f(0.2f, 0.2f, 0.5f));
createBall(0.025f, 0.05f, -0.1f, -0.2f, "", new Color3f(0.2f, 0.2f, 0.5f));
createBall(0.02f, -0.05f, -0.02f, -0.13f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, -0.22f, -0.1f, -0.0f, "", new Color3f(1.2f, 1.2f, 0.5f));
createBall(0.02f, -0.12f, .0f, -0.05f, "", new Color3f(0.2f, 0.2f, 1.5f));


//нижній ярус
createBall(0.02f, -0.02f, -0.3f, 0.25f, "", new Color3f(0.5f, 0.0f, 0.5f));
createBall(0.02f, -0.12f, -0.23f, 0.14f, "", new Color3f(0.2f, 0.2f, 0.5f));
createBall(0.02f, 0.05f, -0.22f, 0.16f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, 0.23f, -0.3f, 0.1f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, -0.12f, -0.28f, 0.2f, "", new Color3f(0.0f, 0.5f, 0.5f));
createBall(0.02f, 0.12f, -0.25f, 0.16f, "", new Color3f(1.2f, 1.2f, 0.5f));
createBall(0.02f, -0.2f, -0.28f, 0.13f, "", new Color3f(0.2f, 0.2f, 1.5f));
createBall(0.02f, 0.15f, -0.2f, 0.05f, "", new Color3f(0.2f, 1.2f, 0.0f));

createBall(0.02f, 0.02f, -0.3f, -0.25f, "", new Color3f(0.5f, 0.0f, 0.5f));
createBall(0.02f, 0.12f, -0.23f, -0.14f, "", new Color3f(0.2f, 0.2f, 0.5f));
createBall(0.02f, -0.05f, -0.22f, -0.16f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, -0.23f, -0.3f, -0.1f, "", new Color3f(1.2f, 0.2f, 0.5f));
createBall(0.02f, 0.12f, -0.28f, -0.2f, "", new Color3f(0.0f, 0.5f, 0.5f));
```

```
		createBall(0.02f, -0.12f, -0.25f, -0.16f, "", new Color3f(1.2f, 1.2f, 0.5f));
		createBall(0.02f, 0.2f, -0.28f, -0.13f, "", new Color3f(0.2f, 0.2f, 1.5f));
		createBall(0.02f, -0.15f, -0.2f, -0.05f, "", new Color3f(0.6f, 1.2f, 0.0f));
	}

	private void createBall(float radius, float x, float y, float z, String picture,
Color3f emissive) {
		TransformGroup tg = new TransformGroup();
		Transform3D transform = new Transform3D();
		Sphere cone = XMassBall.getSphere(radius, picture, emissive);
		Vector3f vector = new Vector3f(x, y, z);
		transform.setTranslation(vector);
		tg.setTransform(transform);
		tg.addChild(cone);
		treeTransformGroup.addChild(tg);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		treeTransform3D.rotY(angle);
		treeTransformGroup.setTransform(treeTransform3D);
		angle += 0.05;
	}
}


import javax.media.j3d.Appearance;
import javax.media.j3d.Material;
import javax.vecmath.Color3f;

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

public class XMassTree {
	public static Cone getCone(float height, float radius) {
		int primflags = Primitive.GENERATE_NORMALS + Primitive.GENERATE_TEXTURE_COORDS;
		return new Cone(radius, height, primflags, getXMassTreeAppearence());
	}

	private static Appearance getXMassTreeAppearence() {
		Appearance ap = new Appearance();
		Color3f emissive = new Color3f(0.0f, 0.05f, 0.0f);
		Color3f ambient = new Color3f(0.2f, 0.5f, 0.15f);
		Color3f diffuse = new Color3f(0.2f, 0.15f, .15f);
		Color3f specular = new Color3f(0.0f, 0.8f, 0.0f);
		ap.setMaterial(new Material(ambient, emissive, diffuse, specular, 1.0f));
		return ap;
	}
}

import java.awt.Container;

import javax.media.j3d.Appearance;
import javax.media.j3d.Material;
import javax.media.j3d.Texture;
import javax.media.j3d.TextureAttributes;
import javax.vecmath.Color3f;
```

```
import javax.vecmath.Color4f;

import com.sun.j3d.utils.geometry.Sphere;
import com.sun.j3d.utils.geometry.Primitive;
import com.sun.j3d.utils.image.TextureLoader;

public class XMassBall {

	public static Sphere getSphere(float radius, String picture, Color3f emissiveColor) {
		int primflags = Primitive.GENERATE_NORMALS + Primitive.GENERATE_TEXTURE_COORDS;
		return new Sphere(radius, primflags, getXMassBallsAppearence(picture,
emissiveColor));
	}

	private static Appearance getXMassBallsAppearence(String picture, Color3f emissive) {
		Appearance ap = new Appearance();
		Color3f ambient = new Color3f(0.2f, 0.15f, .15f);
		Color3f diffuse = new Color3f(1.2f, 1.15f, .15f);
		Color3f specular = new Color3f(0.0f, 0.0f, 0.0f);
		ap.setMaterial(new Material(ambient, emissive, diffuse, specular, 1.0f));

		if (picture != "") {
			// завантажуємо текстуру
			TextureLoader loader = new TextureLoader(picture, "LUMINANCE", new
Container());
			Texture texture = loader.getTexture();
			// задаємо властивості границі
			texture.setBoundaryModeS(Texture.WRAP);
			texture.setBoundaryModeT(Texture.WRAP);
			texture.setBoundaryColor(new Color4f(0.0f, 1.0f, 1.0f, 0.0f));
			TextureAttributes texAttr = new TextureAttributes();
			texAttr.setTextureMode(TextureAttributes.MODULATE);
			ap.setTexture(texture);
			ap.setTextureAttributes(texAttr);
		}
		return ap;
	}
}
```

Результат роботи програми (рис. 6):

Рис. 6

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.

Варіанти

1. Сільський будинок
2. Казковий палац
3. Стіл для пінг-понгу
4. Олівець
5. Кубік Рубіка
6. Сонячна система
7. Морозиво
8. Сніговик
9. Автомобіль
10. Паровоз
11. Гантелі
12. Мухомор
13. Книга
14. Годинник
15. Качелі
16. Жолудь
17. Телефон
18. Людина-робот
19. Персональний комп'ютер
20. Літак