Лабораторна робота № 5

Тема: Імпорт тривімирних моделей у середовище програмування java3D, обробка та маніапуляція цих зображень.

Мета:

1. Здобути навички імпорту моделей, побудованих у тривимірних редакторах, (об'єктів форматів .obj, .lwo, .3ds) до бібліотеки java3D.
2. Навчитися анімувати імпортовані об'єкти.

Використані програмні засоби:

1. Середовище програмування NetBeans IDE 7.2.1.
2. Бібліотеки *j3dcore.jar*, *j3dvecmath.jar*, *j3dutils.jar* з пакету ***j3d-1_5_2-windows-i586***.
3. Програма для перегляду та редагування тривимірних моделей PoseRay 3.13.16 beta.

Теоретичні відомості

Моделювання комплексних тривимірних об'єктів засобами бібліотеки java3D потребує багато часу та розрахунків координат, нормалей до частин моделей нетривіальної форми і т.д. В той же час існує програмне забезпечення, яке призначено саме для моделювання тривимірних моделей. Серед подібних пакетів можна назвати: 3ds Max, Blender, LightWave, Maya, тощо. Головною перевагою для використання зазначеного програмного забезпечення є можливість перегляду результатів розробки моделі в будь-який момент часу, наявність різноманітних режимів перегляду моделі з точки зору взаємного розташування об'єктів у просторі сцени, властивостей текстур, що накладені на об'єкти, матеріалів, встановлення та візуальне відображення джерел освітлення, тощо. Ще одна перевага розробки тривимірної моделі у даних редакторах є можливість розподілу роботи над проектом між моделювальником та розробником: використання зазначених програмних засобів не потребує навичок програмування та знань певних мов програмування. Отож, важливим є питання про використання тривимірних моделей різного генезису у бібліотеках стандартних мов програмування, зокрема у java3D.

Формати 3d моделей

Кожен тривимірний графічний редактор має можливість збереження своїх моделей у файлах кількох форматів. Детальна інформація у таблиці 1.

Таблиця 1

Програмний продукт	Формати, що підтримує
3ds Max	obj, max, 3ds, dae, dxf, lai, тощо.
Blender	obj, blend, dxf, stl, 3ds, тощо.
LightWave	obj, lightwave 5, 3ds, dxf, throw, тощо.
Maya	obj, mi, fbx, тощо.

З таблиці видно, що формат obj є широко розповсюдженим, в той час як інші формати прив’язані до конкретного програмного продукту.

Формат .obj

OBJ – це формат файлів опису геометрії, розроблений компанією Wavefront Technologies для їх анімаційного пакету Advanced Visualizer. Даний формат є відкритим. Був прийнятий як один з форматів файлів розробниками інших систем 3D графіки, наприклад e-Frontier's Poser, Maya, XSI, Blender, MeshLab, Misfit Model 3D, 3D Studio Max и Rhinoceros 3D, Hexagon, CATIA, Newtek LightWave, Art of Illusion, Modo, Cinema 4D, Zanoza Modeller. Файли даного формату можуть бути імпортованими/експортованими до зазначених програм.

Формат даних простий для читання та обробки, зберігає лише 3D геометрію, а саме: позицію вершин, зв’язок координати текстури з вершиною, нормаль для кожної вершини, а також параметри для створення полігонів.

Кожен рядок файлу містить дескриптор рядка, який визначає вид даних, що записані в рядку. Значення дескрипторів подані у таблиці 2.

Таблиця 2

Дескриптор	Позначення	Приклад рядка
#	Коментар	`# some comment`
v	Список вершин, з координатами (x,y,z[,w]), w є необов'зковим, значення по замовченню 1.0.	`v 0.123 0.234 0.345 1.0`
vt	Текстурні координати (u[,v][,w]), v та w – необов'язкові елементи, по замовченню мають значення 0.	`vt 0.500 -1.352 [0.234]`
vn	Нормалі (x,y,z), можуть бути відсутніми.	`vn 0.707 0.000 0.707`
vp	Параметри вершин у просторі (u [,v] [,w]), вільна форма геометричного стану.	`vp 0.310000 3.210000 2.100000`

f	Визначення поверхні або сторони.
	Поверхня визначається через список вершин, текстурних координат чи нормалей. Полігони, наприклад квадрати, можуть бути визначені за допомогою 3-х чи більше вершин/текстурних координат/нормалей.
		`Визначення сторін:`
		`f 1 2 3`
		`f 3/1 4/2 5/3`
		`f 6/4/1 3/5/3 7/6/5`
		`Визначення поверхні:`
		`f v1 v2 v3 v4`

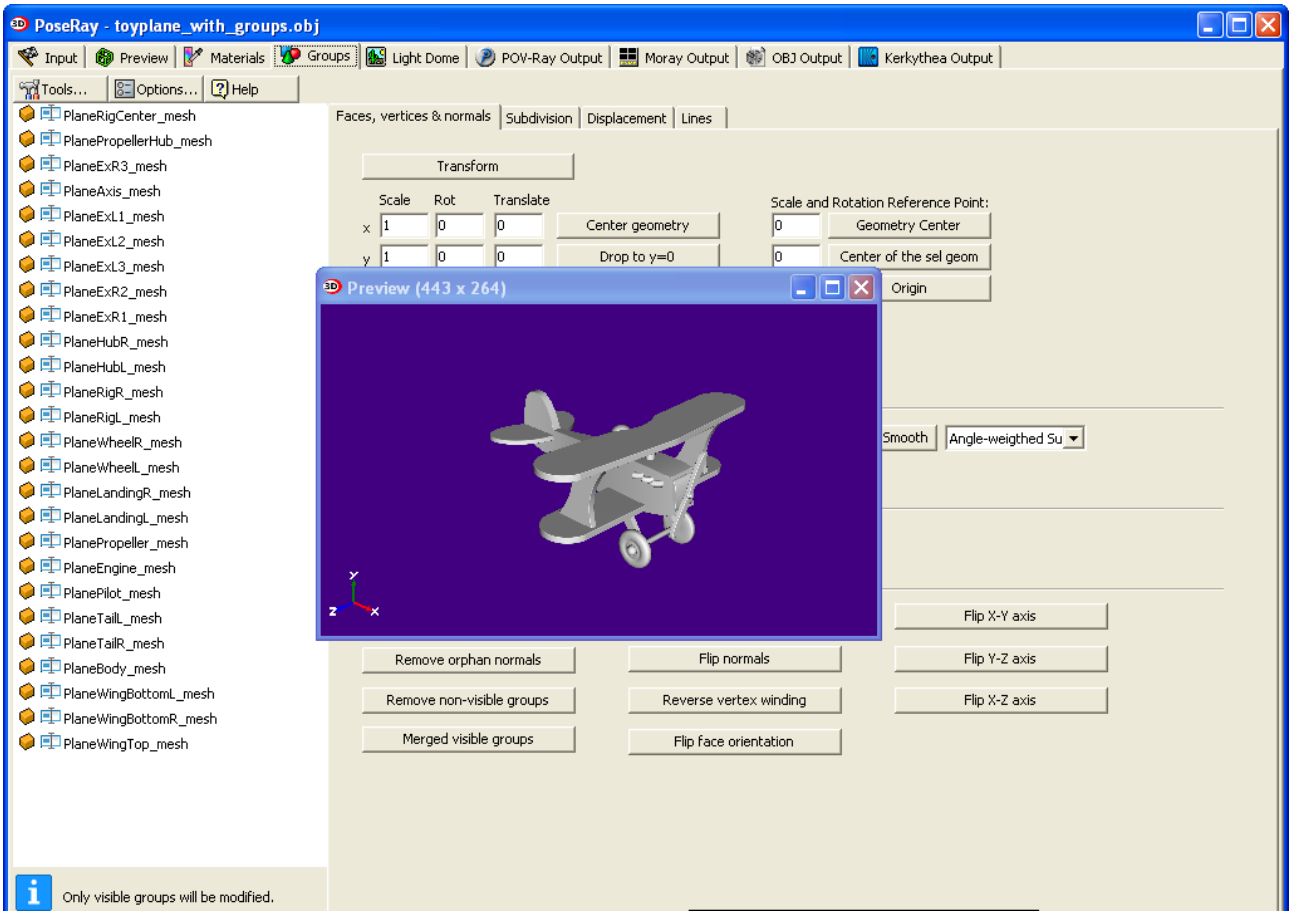
Приклад імпорту та анімації тривимірної моделі засобами Java 3D

У прикладі обрано модель toyplane.obj іграшкового літака, зробленого з дерева, малюнок для фону Mountains.jpg та текстура дерева з файлу wood1.jpg.

1. Попередня робота з моделлю.

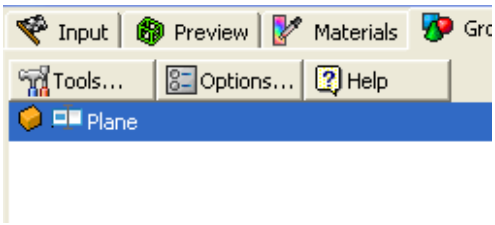
Як правило, моделі об'єктів, що не є тривимірними примітивами є складеними з деякої кількості примітивів. Кожна частина моделі має свою назву та може мати окремі властивості. Для програмної маніпуляції імпортованої моделі необхідно знати назви її частин. Переглянути структуру моделі можливо у будь-якому редакторі тривимірних зображень, що підтримує її формат. В даному прикладі скористаємось програмою PoseRay, що не потребує інсталяції, займає всього кілька Мбайт, має інтуїтивний інтерфейс та дозволяє переглядати та редагувати елементи моделей.

Інтерфейс програми з завантаженою моделлю на малюнку 1.



Малюнок 1

Як видно зі списку на вкладці «Групи» модель містить багато елементів: крила, гвинти, колеса, тощо. Оскільки для прикладу достатньо працювати з літаком вцілому, об’єднаємо їх в одну группу: plane.



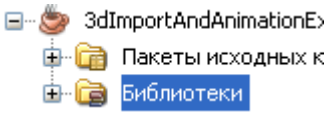
Малюнок 2

Зверніть увагу: Java 3D сприйматиме назви груп лише з малої букви, незважаючи на їх назву, наприклад, “Plane”. Збережемо змінено модель замість вихідної.

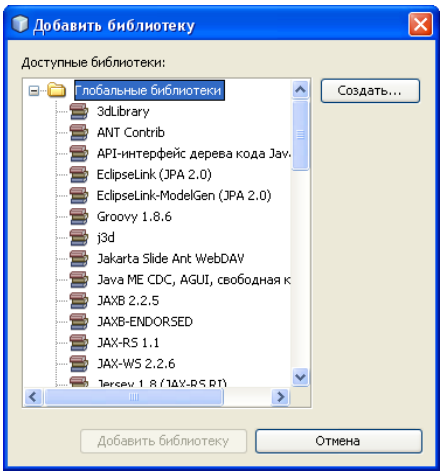
2. Створення середовища перегляду та імпорт моделі.

Створюємо проект з меню Файл-Створити проект. Вказуємо ім’я проекту, ім’я класу, що містить метод main. Для додавання бібліотек у дереві

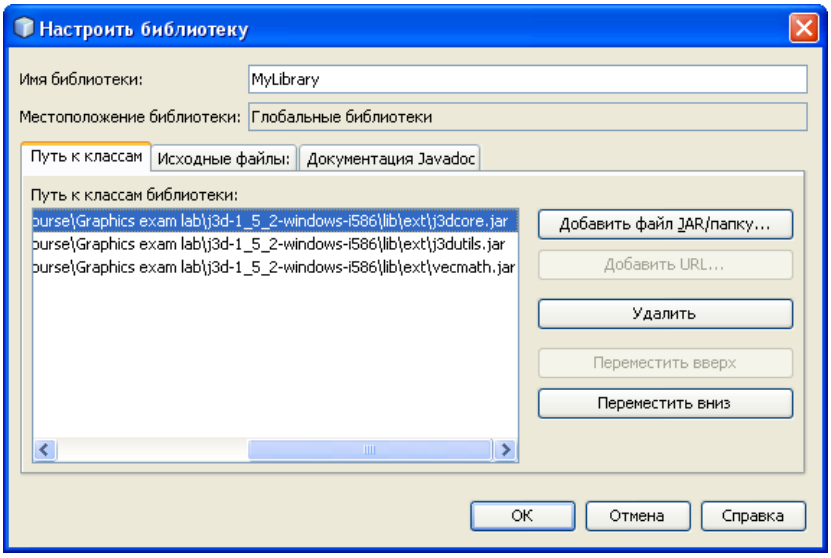
проекту (мал.1правою кнопкою миші клікнути на гілці «Бібліотеки», обрати «Додати бібліотеку», створити власну бібліотеку, додати до неї файли j3dcore.jar, j3dvecmath.jar, j3dutils.jar.



Малюнок 3



Малюнок 4



Малюнок 5

Для роботи знадобляться методи та об’єкти зовнішніх пакетів та бібліотек, імпортуємо у відповідному розділі наступні класи та пакети:

```
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.universe.*;
import java.awt.Color;
```

```
import javax.media.j3d.*;
import javax.media.j3d.Material;
import javax.vecmath.*;
import javax.media.j3d.Background;
import com.sun.j3d.loaders.*;
import com.sun.j3d.loaders.objectfile.ObjectFile;
import com.sun.j3d.loaders.lw3d.Lw3dLoader;
import com.sun.j3d.utils.image.TextureLoader;
import java.awt.*;
import java.io.FileReader;
import java.io.IOException;
import java.util.Map;
import javax.swing.Timer;
import javax.swing.JFrame;
```

Для того, аби в подальшому спростити взаємодію з користувачем, наслідуємо головний клас програми від компонента JFrame.

Опціонально можливо задати початкові налаштування для вікна відображення програми: розмір, назву вікна, умову закриття, режим зміни розмірів вікна, тощо.

Для створення тривимірної сцени створимо та задамо налаштування для полотна для зображення на ньому сцени, об‘єкти-контейнери, в яких вона буде міститися модель, джерела освітлення – щоб побачити об‘єкти сцени.

Оскільки вершиною графу об‘єктів сцени є всесвіт, то створюємо всесвіт, сцену та группу для об’єктів всесвіту.

```
    static SimpleUniverse universe;
    static Scene scene;
    static BranchGroup root;
    static Canvas3D canvas;
```

Елемент типу Canvas3D необхідний для відображення об’єктів сцени. Початкові налаштування для оголошених об’єктів:

```
   private void configureUniverse(){
       root= new BranchGroup();
       universe= new SimpleUniverse(canvas);
       universe.getViewingPlatform().setNominalViewingTransform();
    }
```

Останній оператор задає кут огляду сцени – в даному випадку ми дивимося прямо на екран.

Додамо до сцени напрямлене світло:

```
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);
    }
```

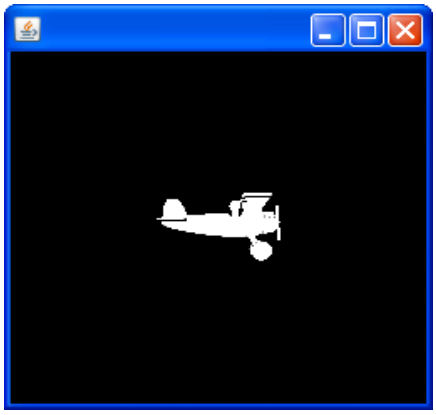
Об’єкти типу світло є частинами графу сцени, тому повинні бути доданими до структури графу – команда addChild. Екземпляри типу Bounds визначають межі проведення розрахунків для об’єкту, екземпляри типів Color3f, Vector3f визначають відповідно тип та напрямний вектор світла.

Для завантаження моделей типу obj та lwo у Java 3D існують вбудовані пакети com.sun.j3d.loaders.objectfile.ObjectFile та com.sun.j3d.loaders.lw3d.Lw3dLoader. Для завантаження сцени необхідно створити об’єкт типу ObjectFile чи Lw3dLoader та викликати метод load, що повертає сцену з вказаного при завантаженні файлу. Надалі необхідно додати об’єкти сцени до всесвіту:

```
root=scene.getSceneGroup();
```

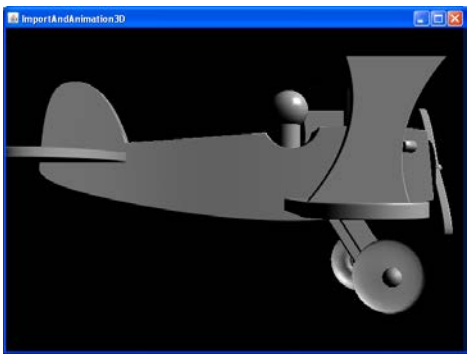
де root – об’єкт типу BranchGroup та містить усі компоненти всесвіту.

На малюнку 6 зображена модель без використання компоненту Canvas:



Малюнок 6

На малюнку 7 – модель імпортована у елемент Canvas:



Малюнок 7

Для взаємодії з завантаженною моделлю не як з об’єктом сцени, а з 3DShape – тобто з тривимірним об’єктом, що може мати свою поведінку, матеріал, текстуру, необхідно занести елементи сцени у структуру Map<String, Shape3D>. Можливо це зробити наступним чином:

```
private void setPlaneElementsList() {
nameMap=scene.getNamedObjects();

wholePlane = new TransformGroup();
transform3D = new Transform3D();
transform3D.setScale(new Vector3d(0.2,0.2,0.2));
wholePlane.setTransform(transform3D);
root.removeChild(nameMap.get("plane"));
wholePlane.addChild(nameMap.get("plane"));
wholePlane.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
root.addChild(wholePlane);
```

Структура nameMap містить усі елементи моделі, до яких можна звертатися за ім’ям. Група TransformGroup надає можливість застосовувати трансформації до об’єктів: масштабування, зсув, поворот. У прикладі модель виведеться на екран у зменшеному масштабі. Елемент типу Transform3D визначає вид трансформації, якщо до об’єкту необхідно застосувати кілька трансформацій, обов’язково поєднайте трансформаційні матриці оператором:

```
someTransform.mul(Transform3D otherTransform);
```

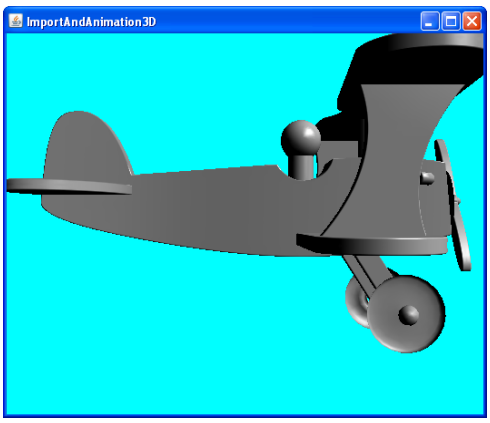
Зауважте: У одного об’єкта сцени не може бути кілька «батьків», тому додавши його до однієї групи, видаліть із попередньої.

Додамо до сцени фон. За бажанням фон може бути залитим обраним кольором, малюнком, або побудований геометричний.

Для побудови кольорового фону виконайте наступні дії:

```
        Background background = new Background(new Color3f(Color.CYAN));
        BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0,
0.0),100.0);
        background.setApplicationBounds(bounds);
        root.addChild(background);
```

Результат на малюнку 8.



Малюнок 8

Для завантаження фону з файлу необхідно скористатися TextureLoader.

```
        TextureLoader t = new TextureLoader("d://3dModels//Mountains.jpg",
canvas);
        Background background = new Background(t.getImage());
        background.setImageScaleMode(Background.SCALE_FIT_ALL);
```

Ми зазначили режим «Розтягнути зображення на все вікно». Доступні інші стандартні режими виводу зображень: розтягнути по горизонталі, по вертикалі, плитка, тощо.

В оригіналі модель створювалася дерев'яною, отож додамо до неї текстуру та задамо властивості матеріалу, що будуть схожими на дерево.

```
        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);
```

Для матеріалу можливо задати його властивості поглинання, розсіювання світла та яскравість поверхні. Вбудованих констант для матеріалів не існує, параметри підбираються емпірично.

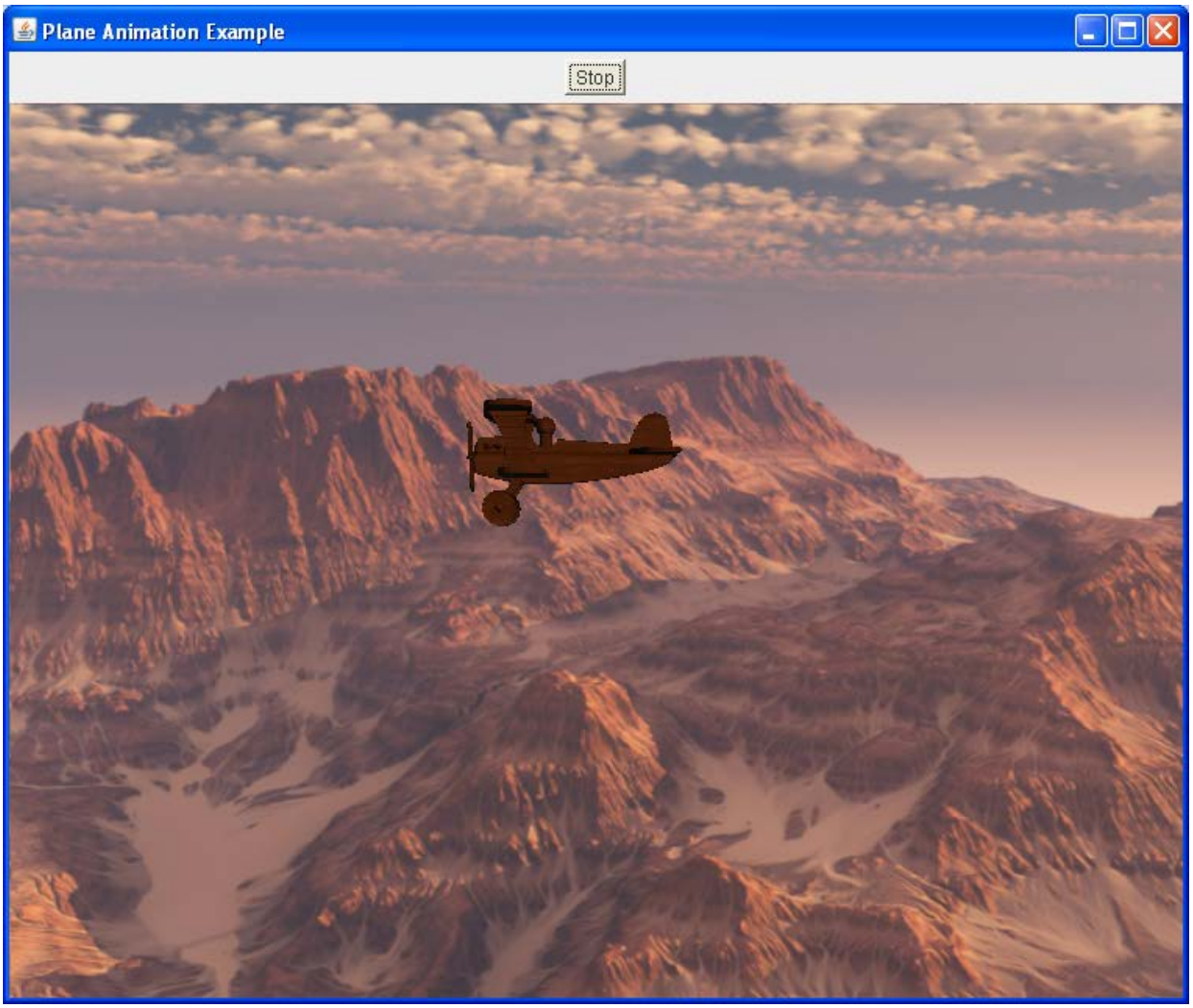
При завантаженні текстури можна вказати, чи вона випромінююча, чи ні:

```
TextureLoader textureLoader = new TextureLoader(path,"LUMINANCE",canvas);
```

Для надання вигляду екземпляру потрібно створити об’єкт типу Appearance, додати до неї потрібну текстуру, встановити режим суміщення текстури з матеріалом: комбінування, заміщення, віднімання.

```
TextureAttributes texAttr = new TextureAttributes();
texAttr.setTextureMode(TextureAttributes.COMBINE);
```

Результат:



Малюнок 9

3. Анімація об’єктів, взаємодія з користувачем програми.

   Створимо окремий клас, який міститиме логіку з анімації моделі. Для реакцію на дії користувача клас повинен реалізовувати інтерфейси ActionListener, KeyListener.

Нехай літак повинен рухатись вліво та вправо після натискання кнопки “Go”, зупинятись при натисканні “Stop”, та рухатись вліво-вправо-вверх-вниз за натисканням клавіш a, s, w, z відповідно. При натисканні клавіш 1, 2, 3 літак може обертатись у площинах x, y, z на 90 градусів. Якщо глядачеві набридло спостерігати за рухом літака, клавішею 0 він може відправити його «за горизонт».

Додамо кнопку Button go до класу з анімації. Створимо кілька об’єктів типу Transform3D для переміщення та повороту літака: translateTransform, rotateTransformX, rotateTransformY, rotateTransformZ. Додамо змінні, що відповідають за поточні координати літака у сцені, його масштаб та напрямок руху, тип руху:

```
private float sign=1.0f;
private float zoom=0.5f;
private float xloc=0.0f;
private float yloc=0.0f;
private float zloc=0.0f;
private int moveType=1;
private Timer timer;
```

Додамо таймер типу javax.swing.Timer, який буде змінювати приріст координат літака. Опишемо реакції на події натискання на кнопку та на клавіші клавіатури в реалізувавши методи наслідуваних інтерфейсів:

```
@Override
    public void actionPerformed(ActionEvent e) {
        // start timer when button is pressed
       if (e.getSource()==go){
          if (!timer.isRunning()) {
             timer.start();
             go.setLabel("Stop");
          }
          else {
              timer.stop();
              go.setLabel("Go");
          }
       }
       else {
           Move(moveType);
           translateTransform.setScale(new Vector3d(zoom,zoom,zoom));
           translateTransform.setTranslation(new Vector3f(xloc,yloc,zloc));
```

```
            wholePlane.setTransform(translateTransform);
        }
    }
```

Крім цих методів, необхідно також реалізувати метод keyPressed, для реакції на натиснені клавіші. А методи keyTyped та keyReleased можна залишити порожніми – ми не потребуємо реакції окремо на ці події.

```
@Override
    public void keyPressed(KeyEvent e) {
         //Invoked when a key has been pressed.
        if (e.getKeyChar()=='s') {xloc = xloc + .05f;}
        if (e.getKeyChar()=='a') {xloc = xloc - .05f;}
        if (e.getKeyChar()=='w') {yloc = yloc + .05f;}
        if (e.getKeyChar()=='z') {yloc = yloc - .05f;}

        if (e.getKeyChar()=='1') {
            rotateTransformX.rotX(Math.PI/2);
            translateTransform.mul(rotateTransformX);
        }
        if (e.getKeyChar()=='2') {
            rotateTransformY.rotY(Math.PI/2);
            translateTransform.mul(rotateTransformY);
        }
        if (e.getKeyChar()=='3') {
            rotateTransformZ.rotZ(Math.PI/2);
            translateTransform.mul(rotateTransformZ);
        }
        if (e.getKeyChar()=='0'){
            rotateTransformY.rotY(Math.PI/2.8);
            translateTransform.mul(rotateTransformY);
            moveType=2;
        }
    }
```

Тепер літак відповідним чином реагуватиме на події користувача. Аби пов’язати модель та реакцію на події необхідно додати «слухача» події. В коді методу main:

```
FirstMainClass window = new FirstMainClass();
AnimationPlane planeMovement = new
AnimationPlane(wholePlane,transform3D, window);
window.addKeyListener(planeMovement);
window.setVisible(true);
```

Анімація моделі у прикладі завершена. Для надання реалістичності можливо визначити кілька траєкторій руху, вказати математичну модель руху, що можна буде зробити при реалізації лабораторної роботи.

Завдання на лабораторну роботу

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.

Приклади сцен для анімації

1. Імпортувати модель доміно. Анімувати почергове падіння кількох доміно.
2. Імпортувати модель м'яча, сходів. Анімувати стрибки кульки по сходах.
3. Імпортувати модель паперового кораблика. Анімувати рух кораблика по весняній калюжі.
4. Імпортувати модель iPad. Анімувати відео презентацію телефону: повороти, переміщення, порівняння розмірів з іншими об'єктами, демонстрація товщини, висоти, тощо.
5. Анімувати рух автомобіля по трасі, повороти, збільшення, зменшення швидкості, зупинку.

Корисні посилання

1. Офіційний тюторіал з Java3D: http://www.java3d.org/tutorial.html
2. Офіційний сайт PoseRay – переглядач і простий редактор для 3Д-моделей: https://sites.google.com/site/poseray/
3. Коротка стаття про завантаження obj та lwo файлів: http://www.vrupl.evl.uic.edu/LabAccidents/java3d/lesson08/indexa.html

4. Детальна стаття про імпорт та подальшу обробку obj-файлів:

   http://www.daltonfilho.com/articles/java3d/

5. Ресурси для скачування 3D моделей:

   http://www.visual-form.ru/zakachka/free/free_files.html

   http://www.hongkiat.com/blog/60-excellent-free-3d-model-websites/

   http://www.turbosquid.com/Search/3D-Models/Animals

6. Про основи роботи з примітивами у 3ds max:

   http://soohar.ru/primitivy-v-3d-max/

7. Адреси для скачування лоадерів:

   http://sourceforge.net/projects/java3dsloader/ – лоадер 3ds файлів. Зчитує та конвертує у формат, що сприймається java
   http://www.starfireresearch.com/services/java3d/java3d.html – нестабільний сайт, може бути не доступним; популярний лоадер
   http://www.cybergarage.org/twiki/bin/view/Main/CyberVRML97ForJava.