Présentation Java 3D

Jean-Marc Farinone
farinone@cnam.fr
(Maître de Conférences CNAM)

Jean-Marc Le Gallic
legallic@ensg.ign.fr
Institut Géographique National
École Nationale des Sciences Géographiques

Alexandre Topol
topol@cnam.fr
(Maître de Conférences CNAM)

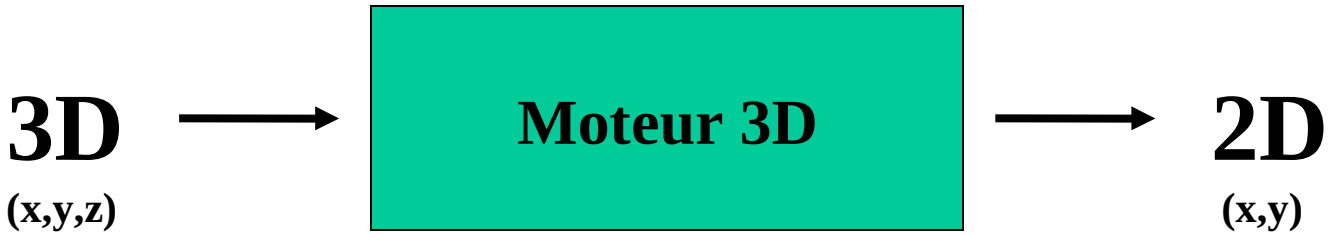
Plan de l'exposé

- La 3D qu'est ce ?
- Java 3D
  - Présentation, bibliographie, installation
  - Architecture de Java 3D
  - Ecrire des programmes 3D en Java 3D
  - Conseils et optimisations
  - Récupérer dans Java 3D des mondes 3D
  - Animer et interagir en Java 3D

La 3D qu'est ce ?

Moteur 3D

La programmation d'un univers 3D implique sa transformation en univers 2D (écran)!



Des exemples de programmes 3D

Voir de bons cours faits par mes collègues ;-) mais ...

La 3D utilise des mathématiques (géométrie 3D, quaternion, etc.) interessantes et non triviales.
Ainsi que des notions informatiques (double buffering, pipe line 3D, etc.) de même qualité ;-)

Il existe des demos 3D en Java avec leurs sources données avec Java SE sous :
`REP_INSTAL_JavaSE/demo/applets/`
voir les répertoires `WireFrame`, `MoleculeViewer`
Voir les sources (et les mathématiques associées) dans le code des applets.

Des demos ? Bon d'accord (filaires et molécules dans navigateur)

Présentation, bibliographie, installation de Java 3D

Qu’est-ce-que Java 3D ?

Java 3D est une bibliothèque de classes destinée à créer des scènes 3D (utilisation de formes complexes, d'éclairages, de textures, d'animations, de sons ...)

Le **package J3D** est une extension du langage Java

J3D utilise les bibliothèques graphiques 3D existantes (les standards **OpenGL**, **DirectX** …) et peut importer des graphes de scènes **VRML**

Indépendant du matériel

Démarrer en Java 3D

1. Prérequis
2. Téléchargement
3. Installation
4. Architecture des packages
5. Performances

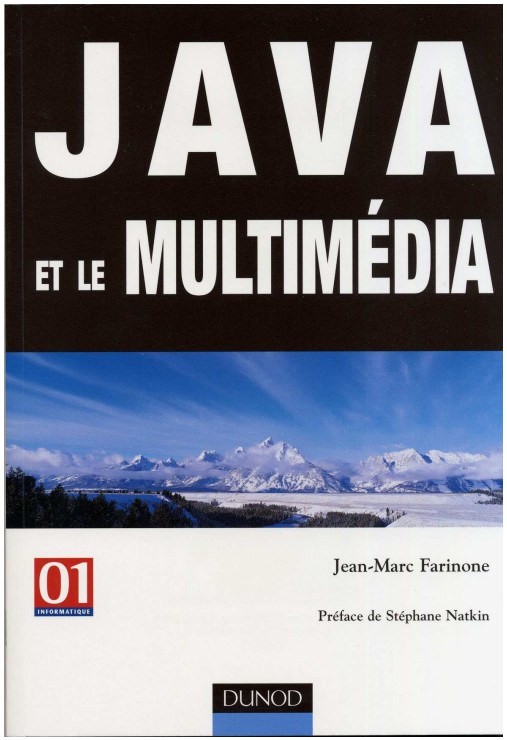
Prérequis

Connaître la programmation en Java

Avoir des connaissances en programmation 3D n’est pas obligatoire mais des notions en géométrie 3D facilitent l’apprentissage

Une bibliographie

Désolé pour la (fausse ?) modestie



Chapitres 2 et 3

Bibliographie

`http://java.sun.com/javase/technologies/desktop/java3d/forDevelopers/J3D_1_2_API/j3dguide/index.html`
: les spécifications de Java 3D

`http://wiki.java.net/bin/view/Javadesktop/Java3DFAQ`
FAQ Java 3D

`http://java.sun.com/javase/technologies/desktop/java3d/forDevelopers/J3D_1_2_API/j3dapi/index.html`
documentation Java 3D en ligne

`http://java.sun.com/developer/onlineTraining/java3d/index.html`
le tutorial Java 3D

Certaines parties du site ne sont accessibles que si on est inscrit : le faire, c'est gratuit.

Téléchargement

“The Java 3D API is now a community source project developed on java.net”. Voir à
`https://java3d.dev.java.net/`

Java 3D est actuellement disponible sous Windows, Solaris, Mac OS X et Linux.

Version actuelle (mars 2010) de Java 3D : 1.6

URL de chargement :
`https://java3d.dev.java.net/binary-builds.html`

Disponible pour Linux, MacOSX, Solaris, Windows

L'installation détecte le Java SE (1.5 au moins) et range les diverses parties dans des répertoires fixes.

Installation

Comme Java 3D est une extension de Java SE, il faut avoir installé Java SE.

Des DLL sont installées dans le répertoire **bin** du JRE

Les fichiers **vecmath.jar, j3dcore.jar, j3daudio.jar et j3dutils.jar** dans le répertoire **lib/ext** du JRE

Pour récupérer les exemples amenés par l'installation, aller à `https://j3d-examples.dev.java.net/` (s'inscrire à la communauté java.net)

il faut utiliser CVS. Voir à

`https://j3d-examples.dev.java.net/servlets/ProjectSource`

Après un peu de bidouille CVS, on récupère les exemples Java 3D sur son disque

Demos Java 3D

Après installation Java 3D amène plusieurs démonstrations rangées dans `%JAVA_HOME/demo/java3d/`

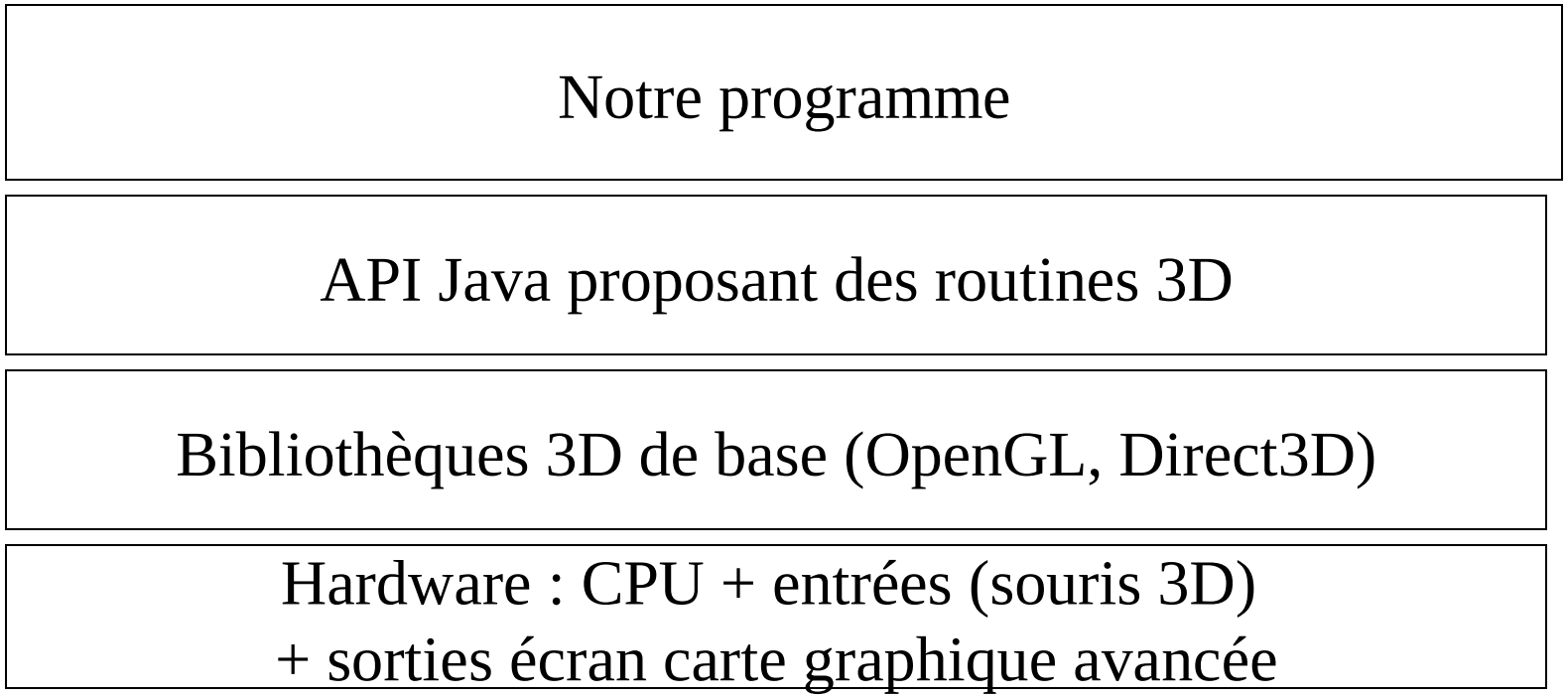
Exemple :

- les engrenages, morphing, TicTacToe, le gallion

Demos Java 3D sous forme d'applets

Architecture de Java 3D

Architecture logicielle



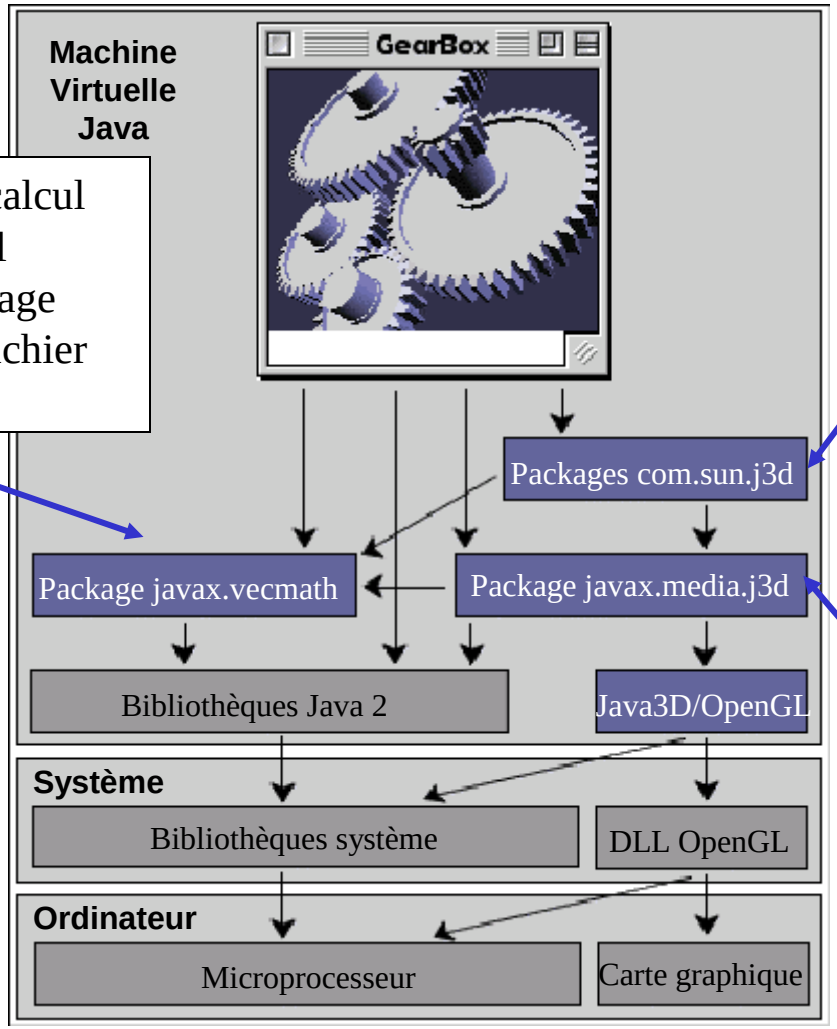
Performances

La 3D est grande consommatrice de calculs. Pour accélérer ses traitements, Java 3D utilise les optimisations offertes par le système

Pour **OpenGL :** appels à des méthodes natives de la **DLL Java 3D/OpenGL**
Ces méthodes appellent des fonctions de la **DLL OpenGL** du système qui utilise l'accélérateur graphique disponible sur la carte graphique

La puissance de traitement de Java 3D dépend donc de la puissance du microprocesseur mais aussi de celle de la carte graphique utilisée

Architecture des packages



Classes dédiées au calcul vectoriel et matriciel appartenant au package **javax.vecmath** (fichier **vecmath.jar**)

Classes utilitaires facilitant la programmation, fournies dans les packages commençant par **com.sun.j3d** ... (fichiers **j3dutils.jar** et **j3daudio.jar**)

Classes de l'API Java 3D appartenant au package **javax.media.j3d** (fichier **j3dcore.jar**)

3D fundamental notions

- The 3D = … a theater

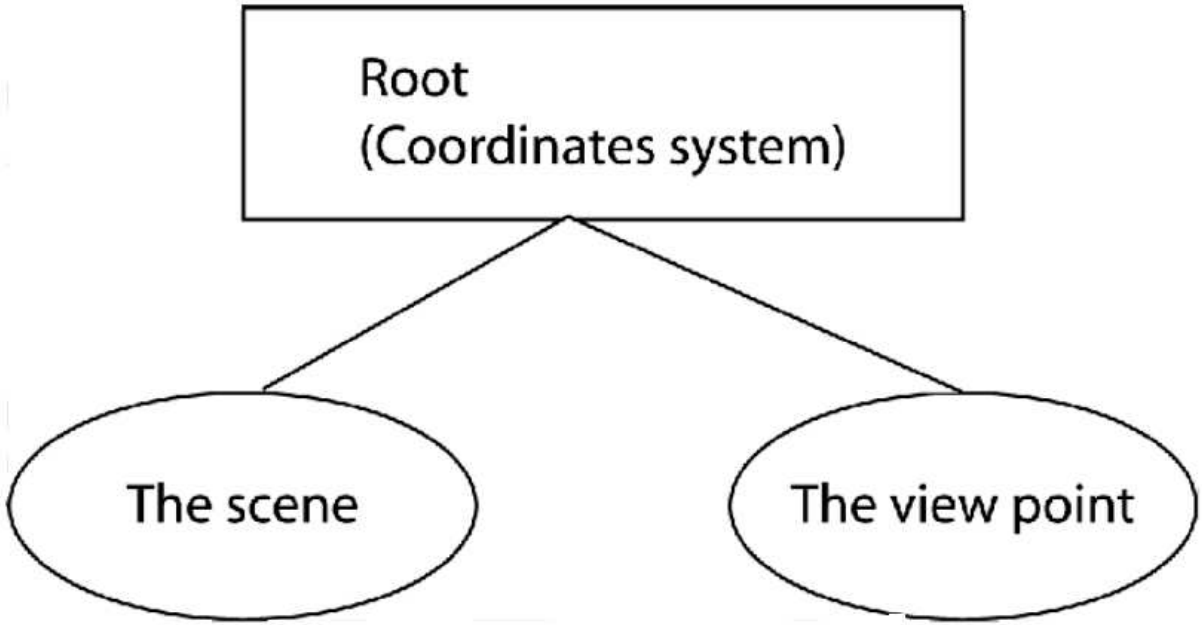
The scene

A point of view

Another point of view

3D Programming fundamental notions

- The 3D = … the previous notions in a tree



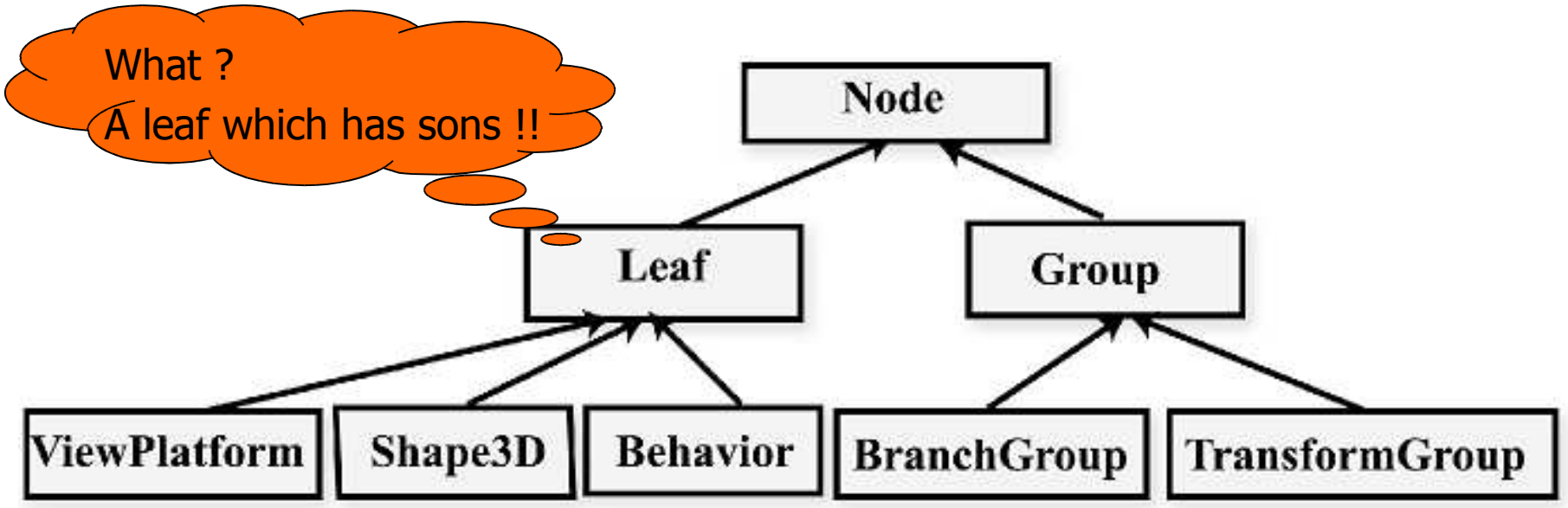
- A placement of 3D components tree

The 3D components tree

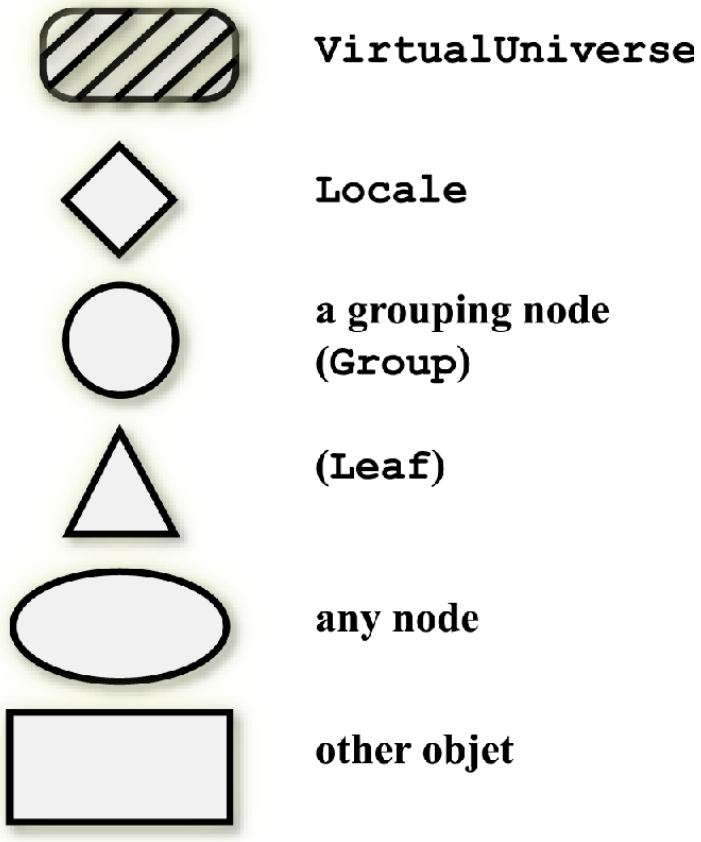
- 3D components correspond to Java objects
- These are obtained as instances of Java classes
- They are, in the program, attached to each other and so, build this tree.

The classes which give 3D components

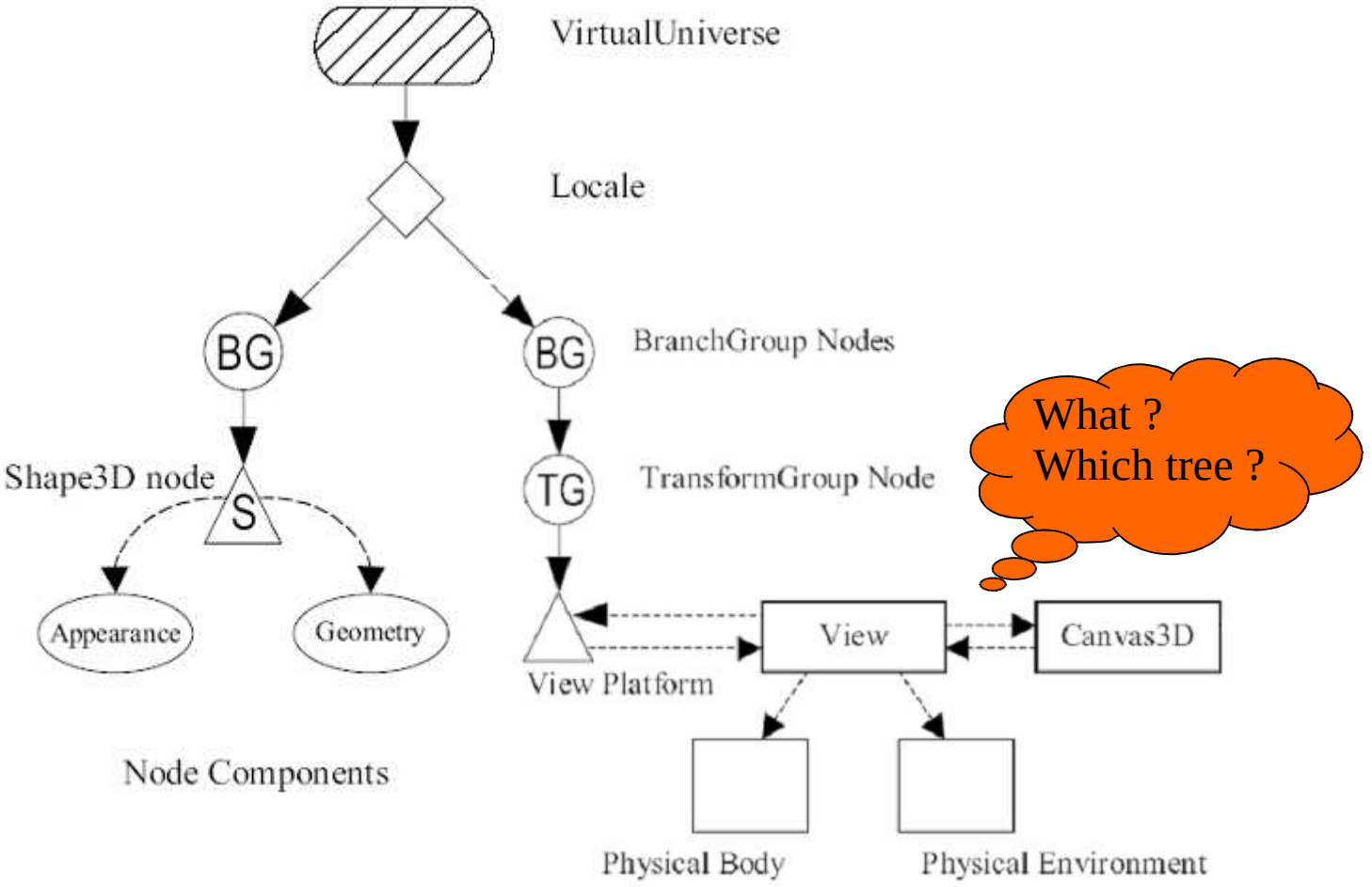
- The main classes:



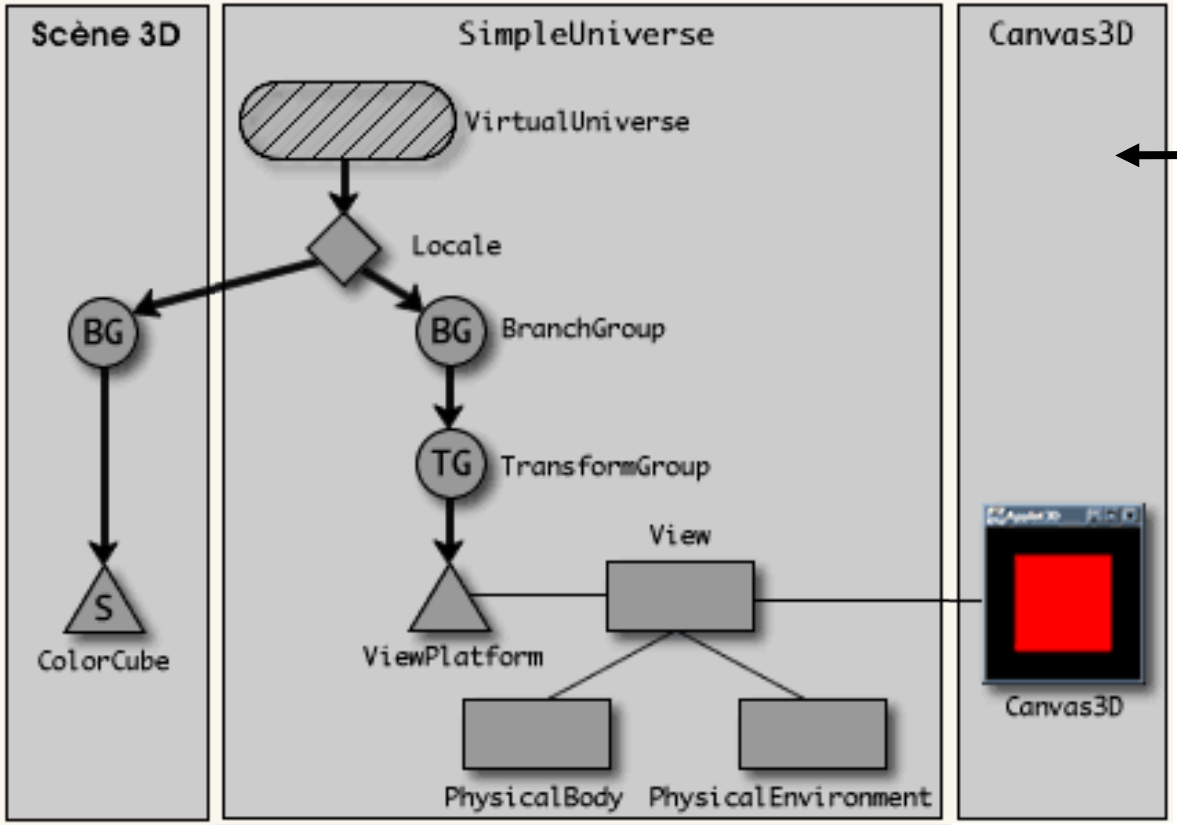
Some graphic definitions for 3D components



The 3D components tree (the return)



The 3D components tree simplified



The **Canvas3D** filled by the APIs Java 3D

Ecrire des programmes 3D en Java 3D

LES PRINCIPES 3D

1. Construction d’un « univers » 3D
2. Le repère 3D
3. Les transformations 3D
4. Conseils et optimisations

De quoi a-t-on besoin ?

Un ensemble de formes **géométriques** (simples ou complexes) et un ensemble de **transformations** appliquées aux formes (translation, rotation …)

⟶ **La scène 3D**

Une zone de l'écran qui permet de visualiser la scène 3D

⟶ **Canvas3D** (**`javax.media.j3d.Canvas3D`**)

Un objet qui permet de relier l'instance de Canvas3D à la scène 3D

⟶ **SimpleUniverse**

La classe **SimpleUniverse** est une classe qui permet de créer des instances par défaut des classes VirtualUniverse, Locale, ViewPlatform, View, PhysicalBody et PhysicalEnvironment, nécessaires à Java 3D

Représentation d’une scène 3D : arbre

Une scène est représentée par un graphe de type arbre comprenant des nœuds et des feuilles

Deux types de nœuds

BranchGroup

Nœud représentant la racine d'un sous-arbre comprenant d’autres nœuds (la racine d ’une scène est une instance de BranchGroup)

TransformGroup

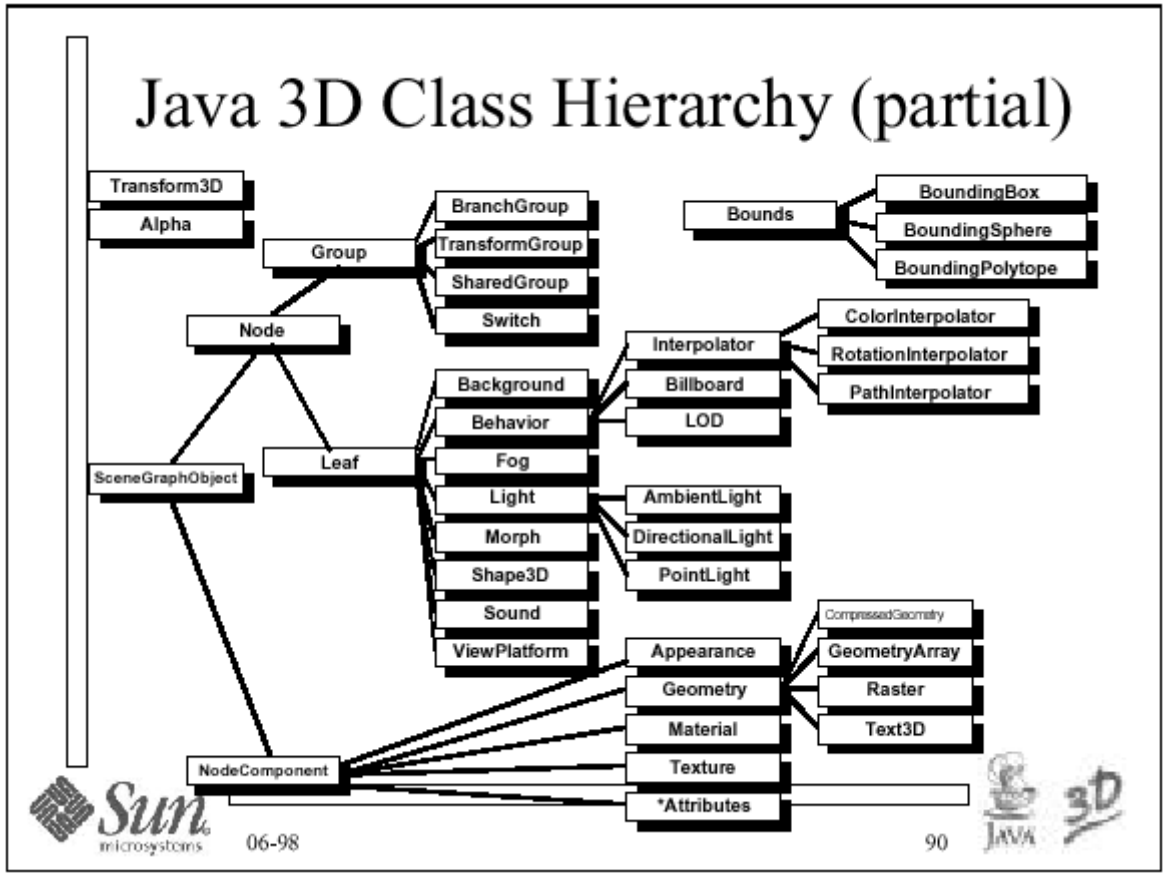
Nœud permettant de gérer un ensemble de transformations appliquées à la racine d'un arbre donc aux objets 3D composant cet arbre.

Les feuilles

représentent les objets : objets graphiques, sons, lumières ...

Pour les objets graphiques il faut décrire la forme (**Geometry**) et l’apparence (**Appearance** - couleur, texture, etc ...)

Les classes de Java 3D



Premier programme Java 3D

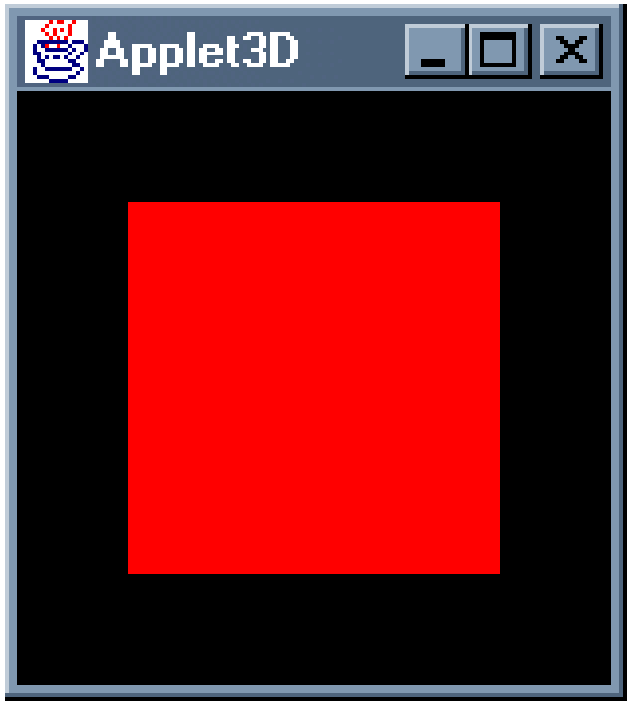
On va construire une applet présentant une scène 3D : le cube fondamental de Java 3D. Cette applet construit une instance de la classe `com.sun.j3d.utils.universe.SimpleUniverse`

Un objet de classe **SimpleUniverse** est constitué :

- d'un objet **VirtualUniverse** qui représente la porte d'entrée dans la scène (c'est la racine de l'arbre de l'application)
- d'un objet **Locale** qui est le descendant direct de la racine et qui permet de gérer le système de coordonnées de l'application
- d'un ou plusieurs nœuds de type **BranchGroup** et **TransformGroup**

Il peut y avoir plusieurs objets **VirtualUniverse** et plusieurs objets **Locale** rattachés à un **Simple Universe** un mais très souvent un seul suffit.

Un exemple : le cube



Démonstration (`1HelloCubeImmobile.bat`)

Procédure de création d’un « univers 3D »

1. Créer un objet **Canvas3D**
2. Créer un objet **VirtualUniverse**
3. Créer un objet **Locale** rattaché au **VirtualUniverse**
4. Créer un objet **BranchGroup**
5. Créer des branches pour les différentes formes 3D
6. Insérer les branches dans l’objet **Locale**

Les étapes 2, 3 et 4 sont réalisées automatiquement

Un exemple complet : le cube

```
import java.applet.Applet;
import java.awt.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.universe.SimpleUniverse;
import com.sun.j3d.utils.geometry.ColorCube;

public class Applet3D extends Applet {
    public void init ( ) {
        Canvas3D canvas = new Canvas3D (SimpleUniverse.getPreferredConfiguration ( ));
        setLayout (new BorderLayout ( ));
        add (canvas, BorderLayout.CENTER);
        BranchGroup scene = createSceneTree ( );
        SimpleUniverse universe = new SimpleUniverse (canvas);
        universe.addBranchGraph (scene);
        universe.getViewingPlatform ( ).setNominalViewingTransform ( );
    }

    public BranchGroup createSceneTree ( ) {
        BranchGroup root = new BranchGroup ( );
        ColorCube cube = new ColorCube (0.5);
        root.addChild (cube);
        return root;
    }
}
```

Création d'un composant de classe Canvas3D permettant de visualiser une scène 3D

Création de la scène 3D à visualiser

Rattachement de la scène 3D à l'univers

Création d'un univers 3D rattaché au Canvas 3D

Positionnement pour avoir une vue correcte sur la scène 3D (permet de voir une scène 3D contenue dans un cube d'1 unité de côté et centré sur le centre du repère i.e. recule l'oeil)

Racine de l'arbre des objets représentés dans la scène 3D

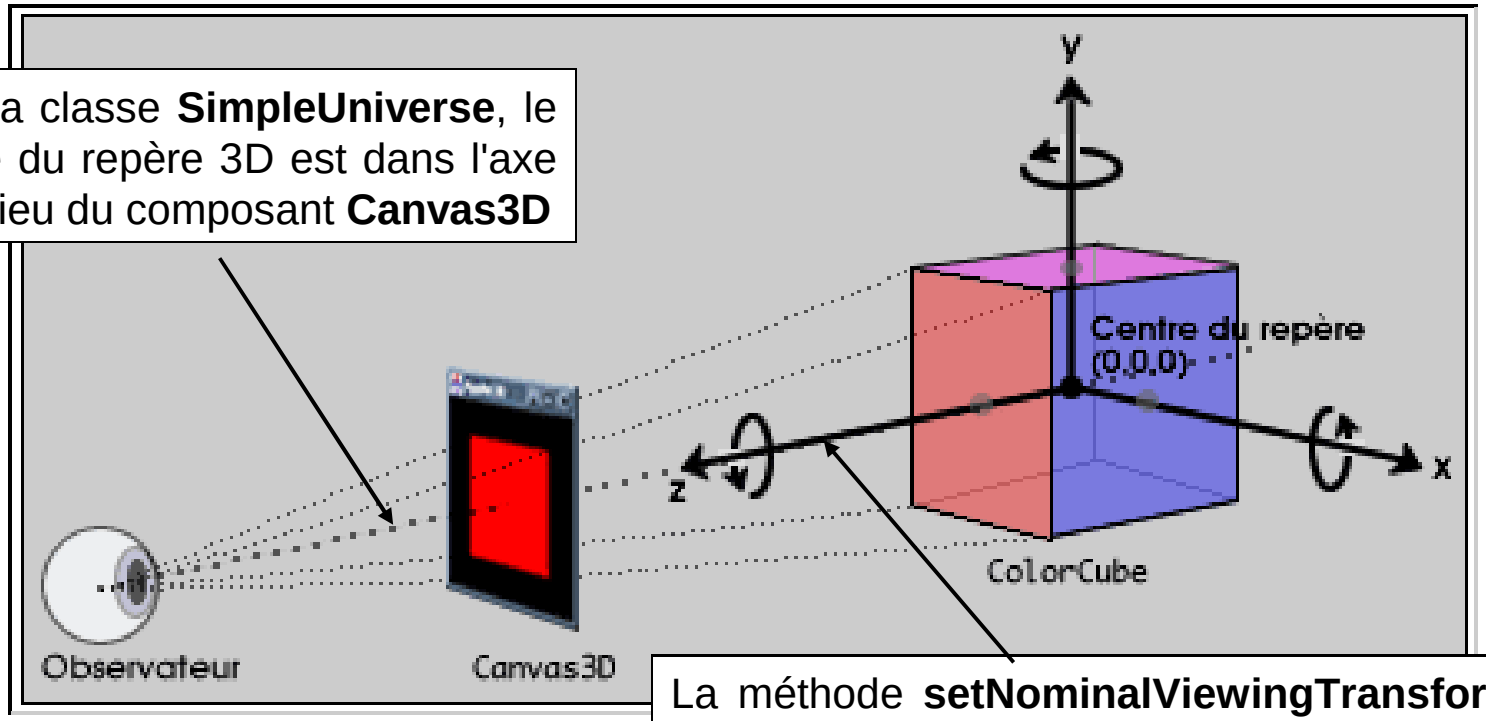
Création d'un cube coloré dont deux des sommets opposés sont situés en (-0.5,-0.5,-0.5) et (0.5,0.5,0.5)

Ajout du cube à la racine de l'arbre

Le repère 3D

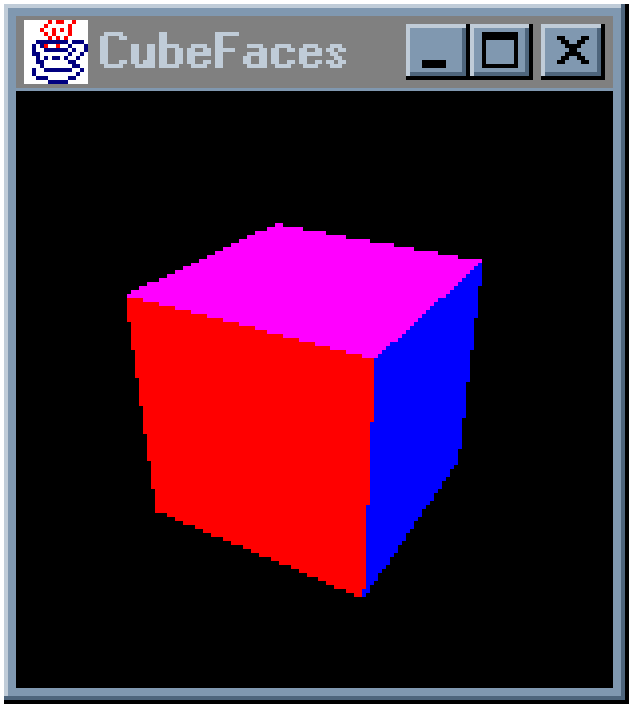
Une scène 3D est composée d'un ensemble de formes 3D. Ces formes sont décrites grâce aux coordonnées (x,y,z) de leurs sommets.

Pour la classe **SimpleUniverse**, le centre du repère 3D est dans l'axe du milieu du composant **Canvas3D**



La méthode **setNominalViewingTransform ()** permet de positionner l'instance de **Canvas3D** à une certaine distance sur **l'axe z** entre l'observateur et l'origine du repère

Les transformations 3D



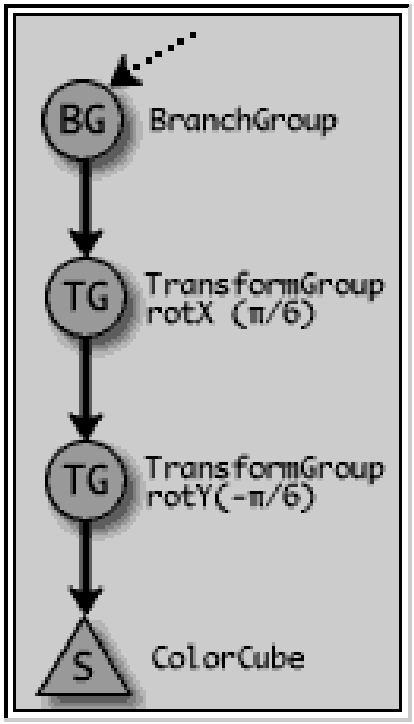
Les transformations 3D

Les **transformations** sont utilisées en 3D pour positionner et animer une forme 3D dans l'espace

Java 3D utilise la classe **Transform3D** pour décrire une opération de translation, de rotation ou d'homothétie

Cette opération est associée à une instance de la classe **TransformGroup** et ce TransformGroup est ajouté à l'arbre de la scène pour l'appliquer sur une forme 3D

Arbre représentant le cube après rotations



L’arbre représentant la scène 3D du cube après une rotation de PI/6 autour de l’axe des X puis une rotation de -PI/6 autour de l’axe des Y

Un exemple : le cube après rotations

```
import javax.media.j3d.*;
import com.sun.j3d.utils.geometry.ColorCube;

public class CubeFaces extends Applet3D {

   public BranchGroup createSceneTree () {

      BranchGroup root = new BranchGroup ();

     Transform3D rotationXAxis = new Transform3D ();
      rotationXAxis.rotX (Math.PI/6);
      TransformGroup rotationXAxisGroup = new TransformGroup (rotationXAxis);

      Transform3D rotationYAxis = new Transform3D ();
      rotationYAxis.rotY (-Math.PI/6);
      TransformGroup rotationYAxisGroup = new TransformGroup (rotationYAxis);

      ColorCube cube = new ColorCube (0.5);

      rotationYAxisGroup.addChild (cube);
      rotationXAxisGroup.addChild (rotationYAxisGroup);
      root.addChild (rotationXAxisGroup);

      return root;
   }
}
```

Méthode de la classe Applet3D surchargée

Racine de l'arbre des objets représentés dans la scène 3D

Création d'une rotation de PI/6 autour de l'axe X

Création d'une rotation de -PI/6 autour de l'axe Y

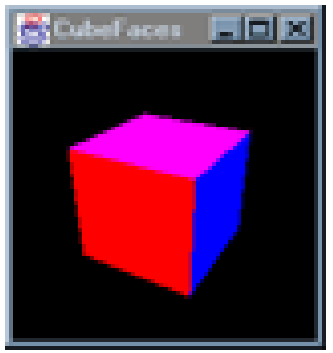
Création d'un cube coloré

Construction de la branche de l'arbre de la scène

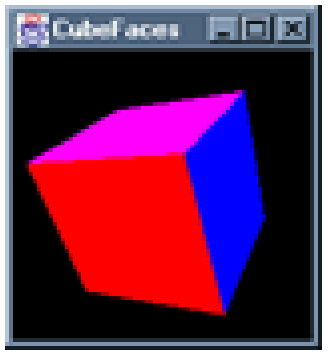
Ordre des transformations

L'ordre dans lequel sont effectuées les transformations a une importance.
Si, dans la méthode **createSceneTree()** de la classe **CubeFaces**, l'enchaînement des rotations est différent, on obtient un résultat différent

```
// Construction de la branche de l'arbre de la scène
// Rotation autour de l'axe y puis de l'axe x
rotationYAxisGroup.addChild (cube);
rotationXAxisGroup.addChild (rotationYAxisGroup);
root.addChild (rotationXAxisGroup);
```




```
// Construction de la branche de l'arbre de la scène
// Rotation autour de l'axe x puis de l'axe y
rotationXAxisGroup.addChild (cube);
rotationYAxisGroup.addChild (rotationXAxisGroup);
root.addChild (rotationYAxisGroup);
```



Démonstration (`2bisCubeFaces2.bat` et `2CubeFaces.bat`)

Conseils et optimisations

Quelques conseils

Dessiner un arbre quand vous créez vos scènes 3D

- permet d'effectuer mentalement l'assemblage des différentes formes d'une scène
- documentation plus lisible que le code

Attention à ne pas affecter **deux parents à un nœud**

- Java 3D déclenchera une exception **MultipleParentException**

Attention à ne pas créer de **branches inutiles**

- Java 3D ne signalera aucune erreur à l'exécution mais le résultat ne correspondra pas à vos attentes

Optimisation

Java 3D propose une notion pour améliorer les performances des calculs d'affichage 3D :

La compilation des nœuds d'une scène

Compilation des nœuds

Compilation => ? représentation interne que Java 3D manipule de manière optimum

En fait peu précisé dans les spécifs Java 3D.

La méthode **compile()** de la classe **BranchGroup** optimise tout le sous-arbre de racine ce **BranchGroup**.

La méthode **isCompiled()** de la classe **SceneGraphObject** de savoir si un nœud a été compilé (= fils d'un **BranchGroup** compilé**)**

La classe **Applet3D** peut être optimisée en ajoutant l'instruction **scene.compile();** dans la méthode **init()**

Nœud vivant, aptitude (capacity) d’un nœud

Ajouter un point de branchement (**BranchGroup**) à un **Locale** le rend *vivant*, ainsi que tous les objets de sa sous-arborescence.

Pour **modifier/lire les caractéristiques d'un objet vivant ou compilé** d ’un nœud (changer des valeurs d'une transformation pour créer une animation), il faut le préciser **avant** son ajout dans le **Locale** ou **avant** la **compilation**.

setCapability() de la classe SceneGraphObject autorise la lecture ou la modification de la propriété correspondante d'un nœud compilé ou vivant

Capacité d ’un nœud TransformGroup **: exemple**

setCapability (TransformGroup.ALLOW_TRANSFORM_WRITE)

Lancée sur un TransformGroup, cette méthode permet de changer les valeurs de ce TransformGroup. Par exemple, permet d’appeler la méthode **setTransform()** pour changer la transformation

Remarque

Quand la racine de l'arbre d'une scène 3D est rattaché à un univers avec la méthode **addBranchGraph**(), Java 3D vérifie les différentes aptitudes (capacity) de chacun des nœuds en utilisant la méthode **getCapability ()** et optimise la représentation interne des objets 3D.

Capacité d ’un nœud : exemple

```
public BranchGroup createSceneGraph() {
      BranchGroup objRoot = new BranchGroup ( );

      objTrans = new TransformGroup ( );
      objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
      objRoot.addChild(objTrans);

      objTrans.addChild(new ColorCube(0.4));

      return objRoot;
   }


 public void actionPerformed(ActionEvent e ) {
       if (e.getSource()==rotateB){
           angle += Math.toRadians( 10 );
           trans.rotY( angle );
           objTrans.setTransform(trans);
       }
    }
```

Récupérer dans Java 3D des mondes 3D

Construire des mondes 3D

- … à l'aide de modeleurs 3D
- Exemples:
  - 3D Studio Max
  - Blender
  - Maya
  - Lightwave 3D
    qui génèrent des formats 3D comme Wavefront obj, VRML, X3D, ...

VRML

```
#VRML V2.0 utf8
DEF T  Transform {
children [
  Transform {   # top
    translation 0.0 0.4 0.0
    children
    Shape {
      appearance Appearance {
        material Material { diffuseColor 0.8 0.8 0.8 }
      }
      geometry Box { size 1.3 0.15 1.3 }
    }
  }
  Transform {   # leg 1
    translation -.5 0.0  -0.4
    children
    DEF Leg Shape {
      appearance Appearance {
        material Material { diffuseColor 1.0 1.0 0.0 }
      }
      geometry Cylinder { height 0.9 radius .075 }
    }
  }
  Transform {   # leg 2
    translation .5  0.0 -0.4
    children USE Leg
  }
  Transform {   # leg 3
    translation  -.5 0.0 0.4
    children USE Leg
  }
  Transform {   # leg 4
    translation .5 0.0 0.4
    children USE Leg
  }
]
}
```

Une table en VRML

Charger des mondes 3D avec Java 3D

- Use a scene loader
- From 3D world file to a `Scene` object: method `Scene load(String filename)` of the `Loader` interface
- `com.sun.j3d.loaders.Scene` is a Java interface
- Use the right class which implements this interface
- Pattern Bridge

De la Scene aux objets Java 3D

- `public BranchGroup getSceneGroup()`
  - returns the `BranchGroup` containing the overall scene loaded by the loader. All enabled items will be loaded into this `Scene`
- `public Hashtable getNamedObjects()`
  - returns a `Hashtable` which contains a list of all named objects in the file and their associated scene graph objects.
  - For example, DEF names of VRML or filenames of objects in Lightwave 3D.

Charger un monde 3D VRML dans Java 3D

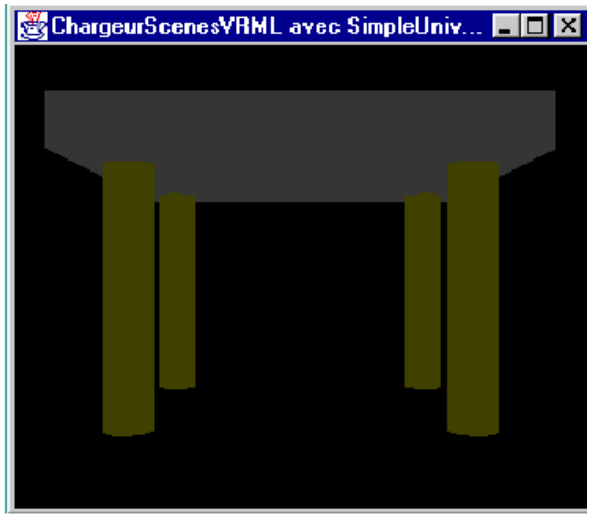
```
// ...
import com.sun.j3d.loaders.Scene;
import org.web3d.j3d.loaders.VRML97Loader;
public class MyLoaderVRML extends JFrame {
  public MyLoaderVRML ( ) {
    // …
    // change BranchGroup scene = buildScene(); by
    BranchGroup scene = loadScene();
    // …
}

 public BranchGroup loadScene ( ) {
    BranchGroup rootBG = new BranchGroup ( );

    VRML97Loader loaderVRML = new VRML97Loader();
    Scene theScene = null;
    // ...
    theScene = loaderVRML.load(nomFichierDeScene);
    // ...
    Node forme3D = theScene.getSceneGroup();
    rootBG.addChild(forme3D);
    return rootBG;
  }

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

- Download Xj3D at `http://www.web3d.org/TaskGroups/source/xj3d.html` to obtain a VRML loader



Bibliography Java 3D

- Java 3D in the real world `http://java.sun.com/products/java-media/3D/in_action/`
- Java 3D et les jeux : `"Killer Game Programming in Java",` Andrew Davison, ed O'Reilly. The main emphasis of my book (over 17 chapters) is on 3D gaming using Java 3D : `http://fivedots.coe.psu.ac.th/~ad/jg/`

Bibliographie Java et le multimédia

`http://java.sun.com/javase/technologies/desktop/techoverview.jsp` : Un site sur les technologies Java additionnelles au Java SE entre autre le multimédia (Java 3D, JOGL = Java Bindings for OpenGL, Java Sound, Java Media Framework, Java Speech)