Java 3D
Concepts de base

Licence Professionnelle : Métiers de l'Informatique

Image et Vidéo

Patrick Reignier
Université Joseph Fourier

2000 – 2001

Plan

- Généralités
- Présentation de Java3D
- Graphe de scènes
- Exemple

A

Généralités sur la synthèse d'images temps réel

Synthèse d'images temps réel

- *Aspect matériel* : facultatif mais **fortement** conseillé.
  - Stations spécialisées :

    SGI, Sun, HP, IBM
  - Cartes accélératrices :

    PC, Mac dans une moindre mesure.

- *Aspect logiciel* :
  - APIs

APIs

- Grande variété.
- Deux grandes catégories :
  - Les APIs bas niveau
  - Les APIs haut niveau

APIs bas niveau

- Interface avec le matériel.
- Emule le matériel inexistant.
- Très bas niveau d'abstraction :
  - Sommets, faces
  - Processus de rendu
- Exemples :
  - OpenGL : Standard multi OS, multi machines.
  - Direct3D IM : Windows.

APIs haut niveau

- S'appuient sur les APIs bas niveau.
- Haut niveau d'abstraction :
  - Objets par opposition à *sommets*
  - Contenu par opposition à *processus de rendu*x
- Graphe de scènes (voir transparents suivants)
- Exemples
  - Open Inventor, Performer, OpenGL Optimizer, Java3D ...
  - Direct3D RM

Graphe de scènes

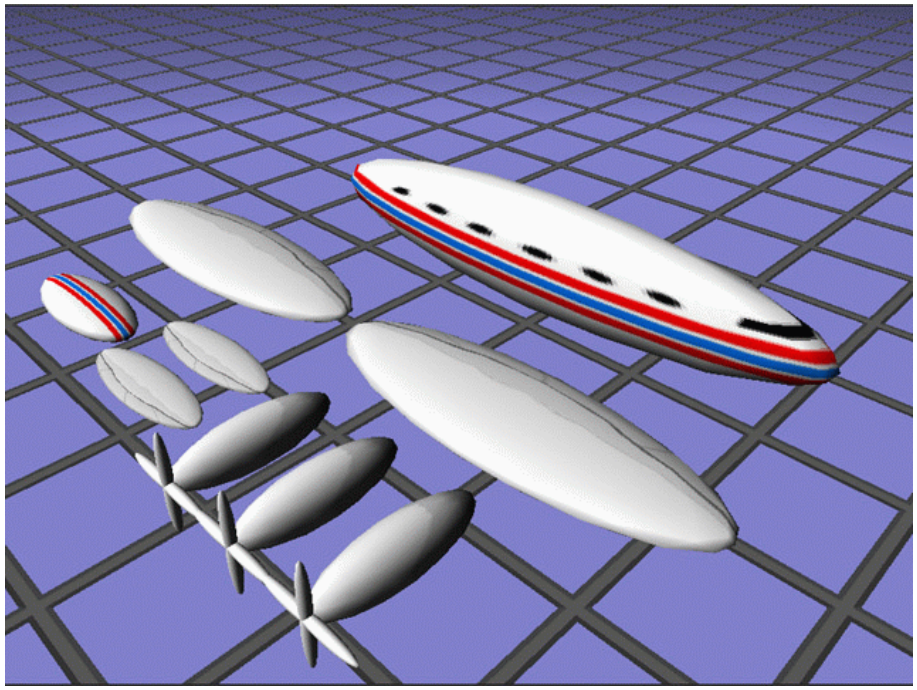
- Univers 3D ;
  - Un ensembles d'objets 3D.
  - Une ou plusieurs caméras.

- Objet 3D :
  - Formés eux-mêmes d'un ensemble d'objets 3D.
  - Exemple :

    Un Avion = un fuselage + des ailes.

Graphe de scènes

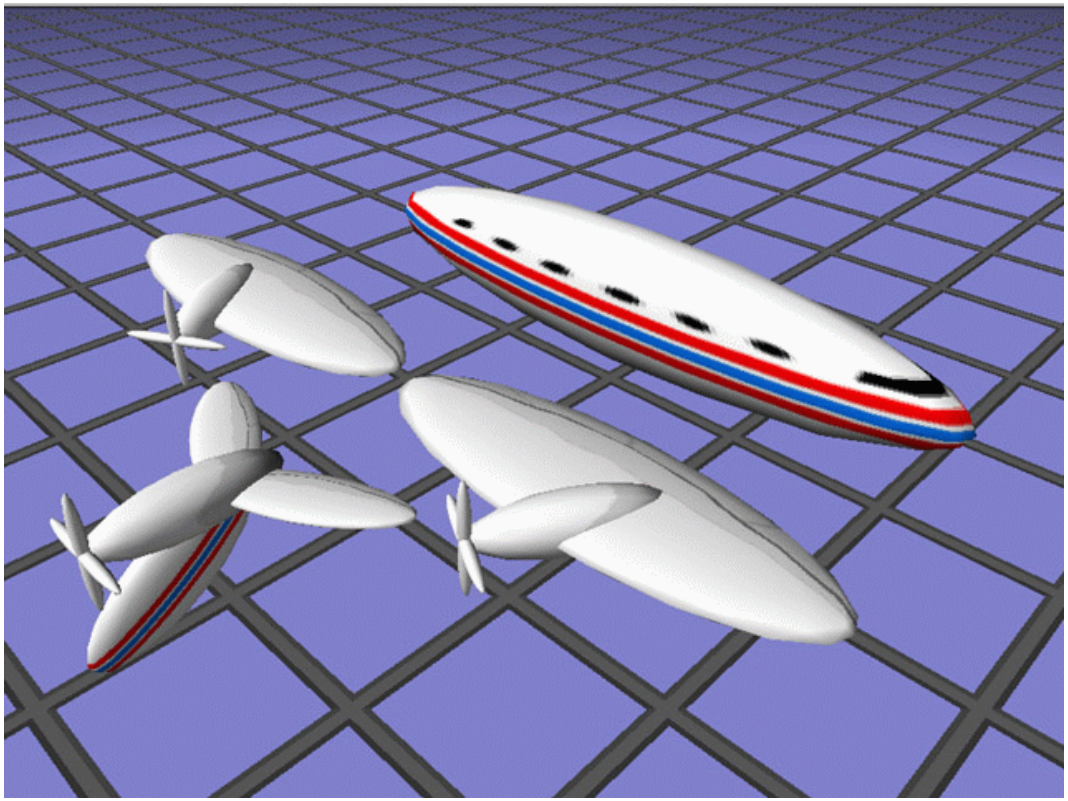
- Scène 3D = hierarchie d'objets
  $\Rightarrow$ notion de graphe de scènes
- Exemple : éléments constituants[a]



[a]extrait de JavaOne 99, Sun

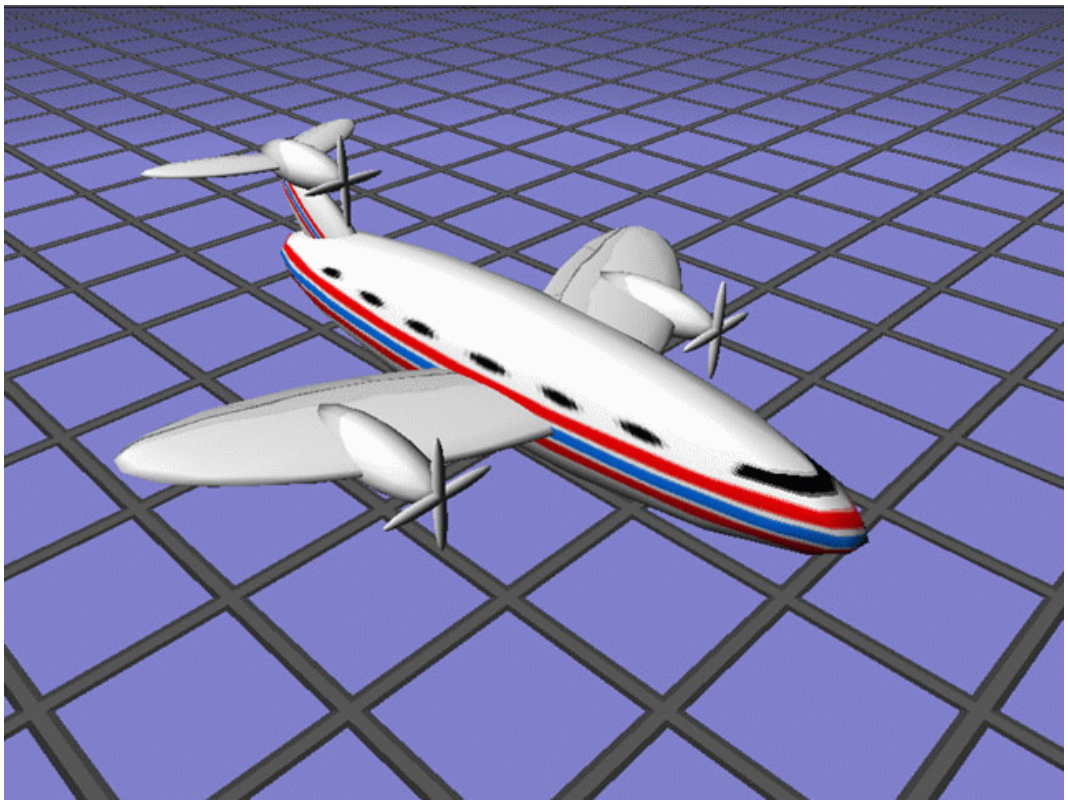
Graphe de scènes

- On commence à grouper les éléments de base

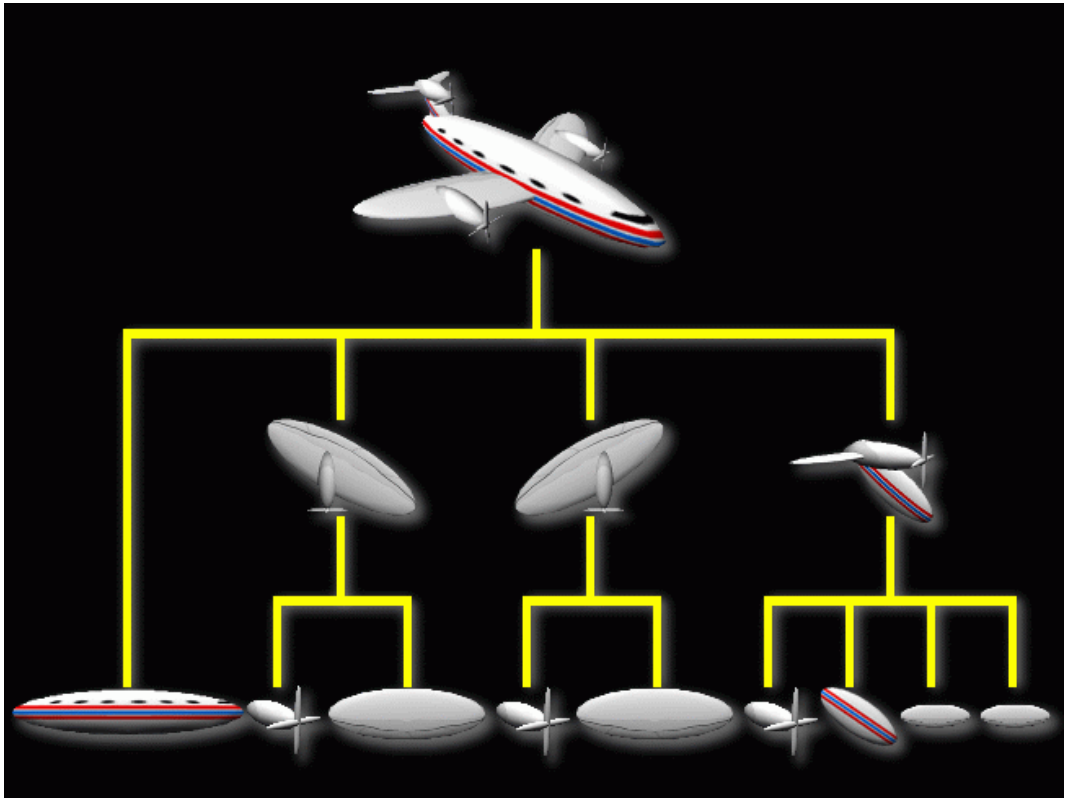


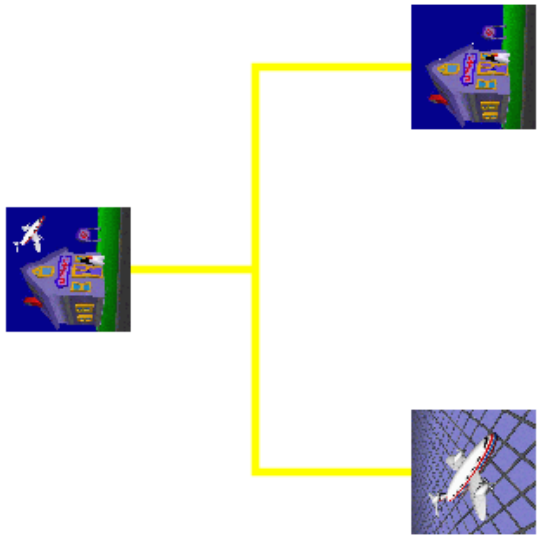
Graphe de scènes

- On finit l'avion.



Graphe de scènes





B

Présentation

Java 3D

- *API* 3D développée par SUN en 1998
- Basé sur la notion de graphe de scènes.
- Classes Java pour :
  - la création
  - la manipulation des scènes 3D
- Accès à tous les packages Java

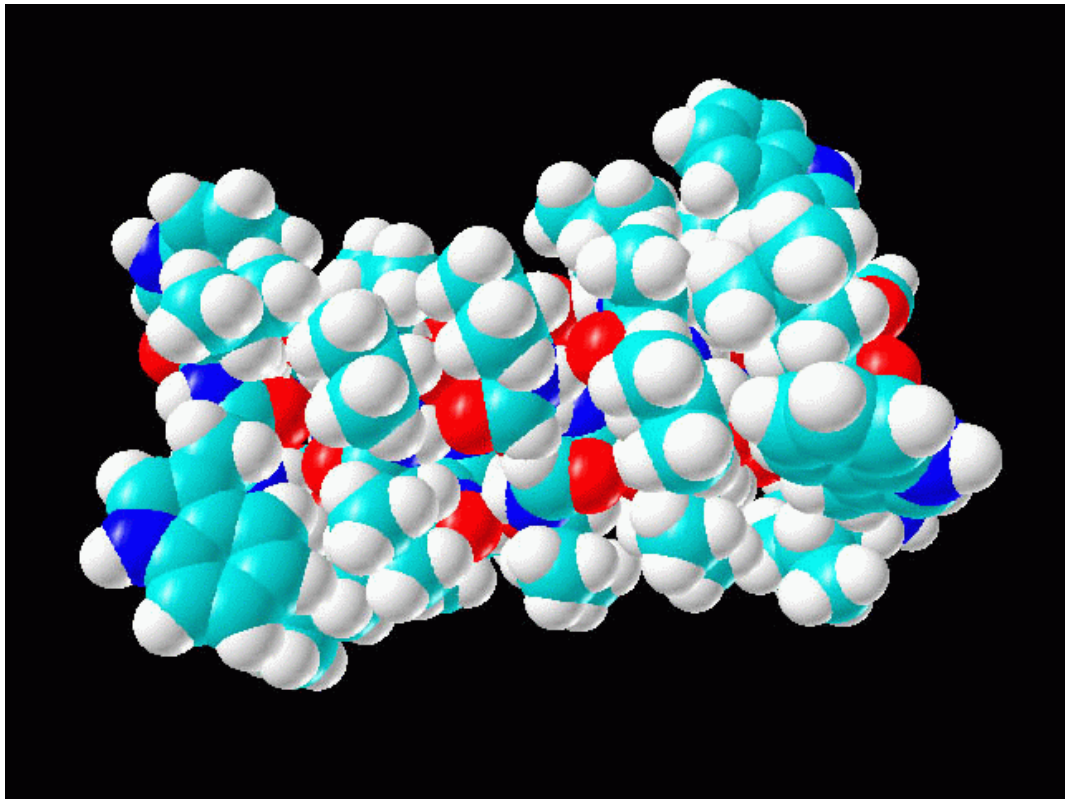
Rendu 3D

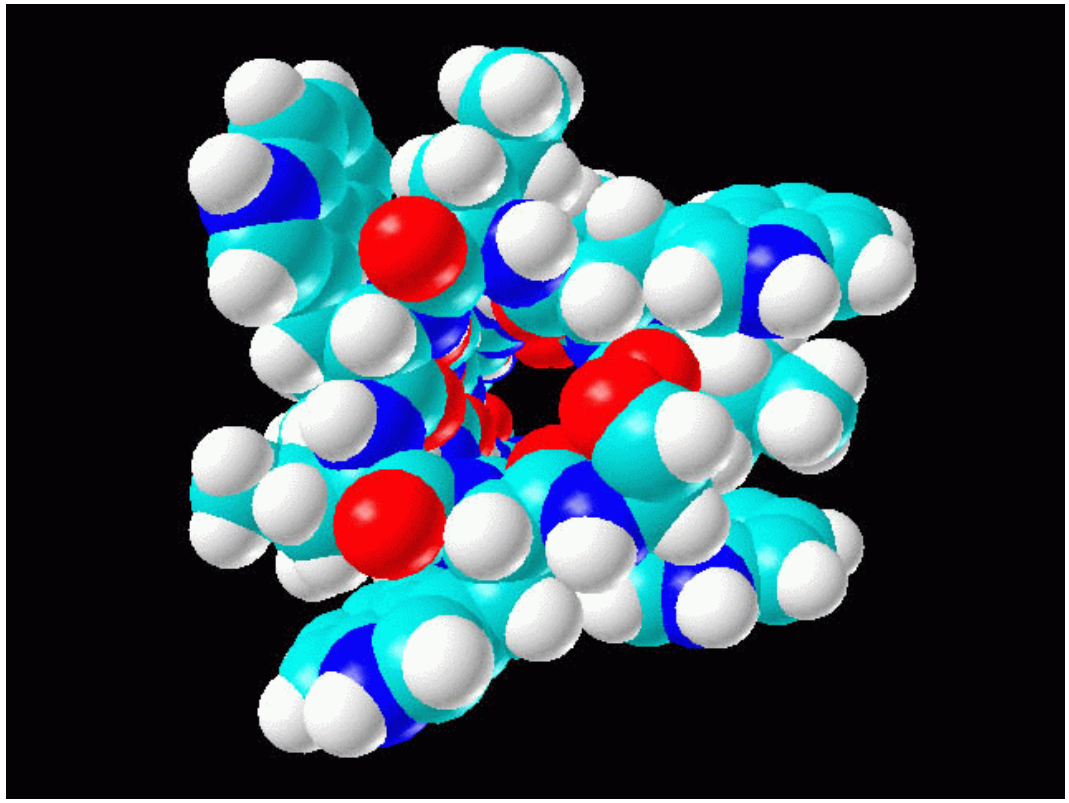
- Rendu haute performance.
- Utilisation de l'accélération 3D matérielle si disponible.
- Basé sur :
  - OpenGL.
  - Direct3D.

Applications

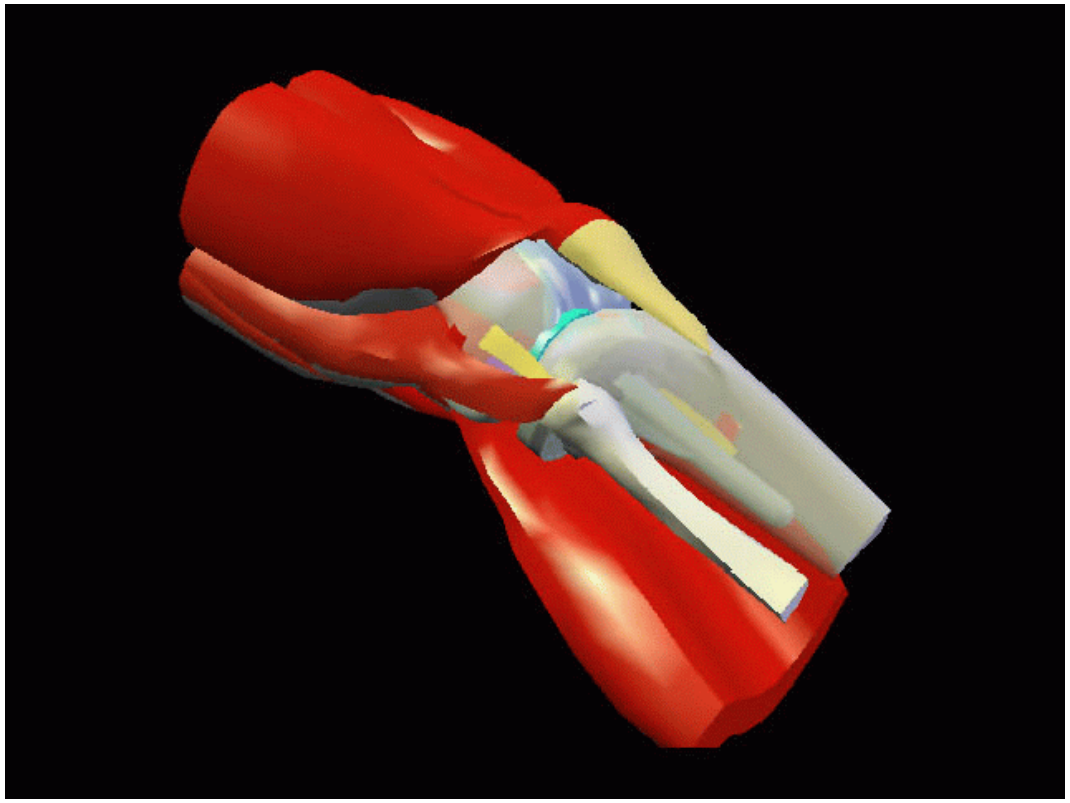
- Visualisation scientifique.
- Visualisation d'informations.
- Entraînement médical.
- Système d'information géographique.
- CAO.
- etc. . .

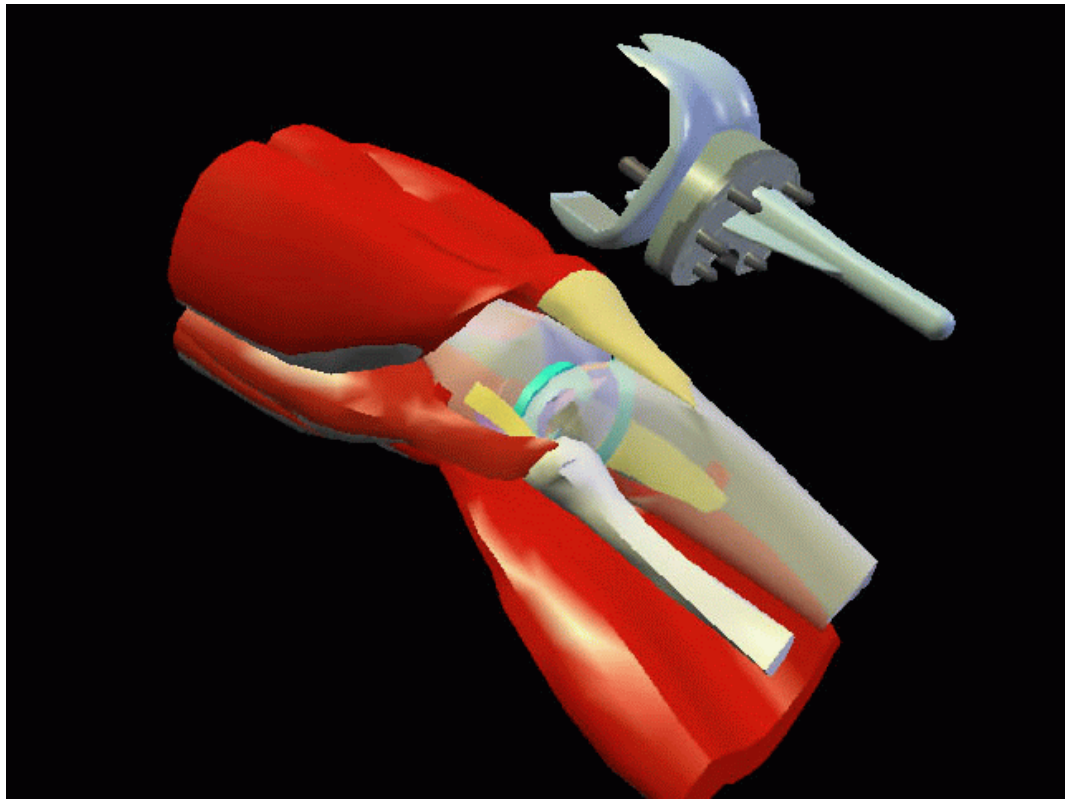
Visualisation scientifique



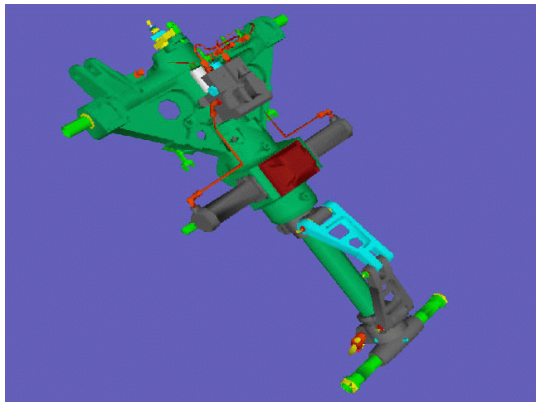


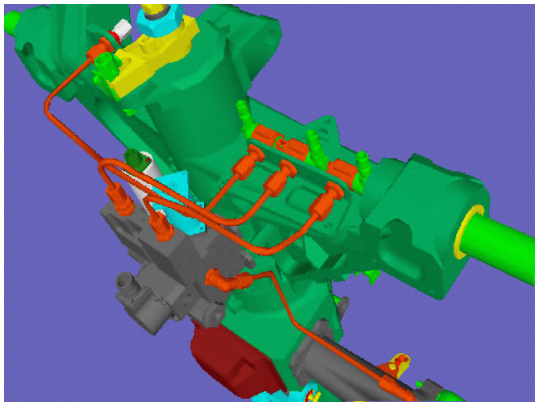
Visualisation médicale

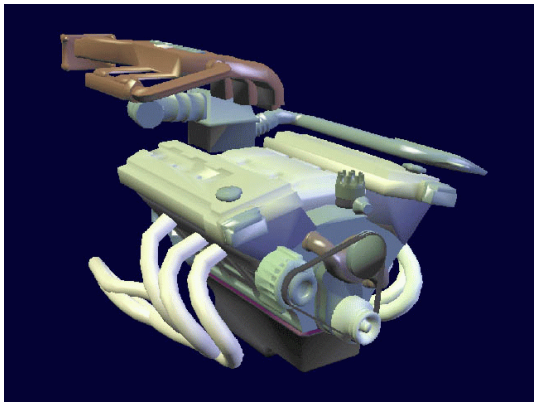




Conception Assistée par Ordinateur







C

Graphe de scènes

Graphe de scènes

- Java3D repose sur la notion de graphe de scènes
  - *Feuilles :* formes 3D, lumières, sons, comportements . . .
  - *Parents :* groupes d'enfants, transformations 3D . . .

  $\Rightarrow$ groupement hiérarchique de formes.

Application Java3D

- Développeur :
  - **Construction et mise à jour d'un graphe à l'aide des classes et des méthodes Java3D**
- Service assuré par Java3D :
  - Rendu à l'écran du contenu du graphe

Rendu

- Ordre de parcours du graphe :
  - Choisi par Java 3D

- Rendu via des threads indépendants et asynchrones
  - Graphique.
  - Son.
  - Comportement (animation).
  - Périphériques d'entrée-sortie.
  - Evénements (détection de collisions).

Vocabulaire

Node : un noeud du graphe de scènes

- *Leaf node :* une feuille
  - Formes, sons, lumières
  - Comportement (animation)
- *Group node :* un noeuds avec enfants
  - Transformations, switch . . .

Node component : les attributs d'un noeud

- La description géométrique d'une forme
- La couleur d'une forme
- Un son à jouer.
- . . .

Classes Java3D

Class Hierarchy

```
java.lang.Object
    javax.media.j3d.SceneGraphObject
        javax.media.j3d.Node
            javax.media.j3d.Group
            javax.media.j3d.Leaf
        javax.media.j3d.NodeComponent
```

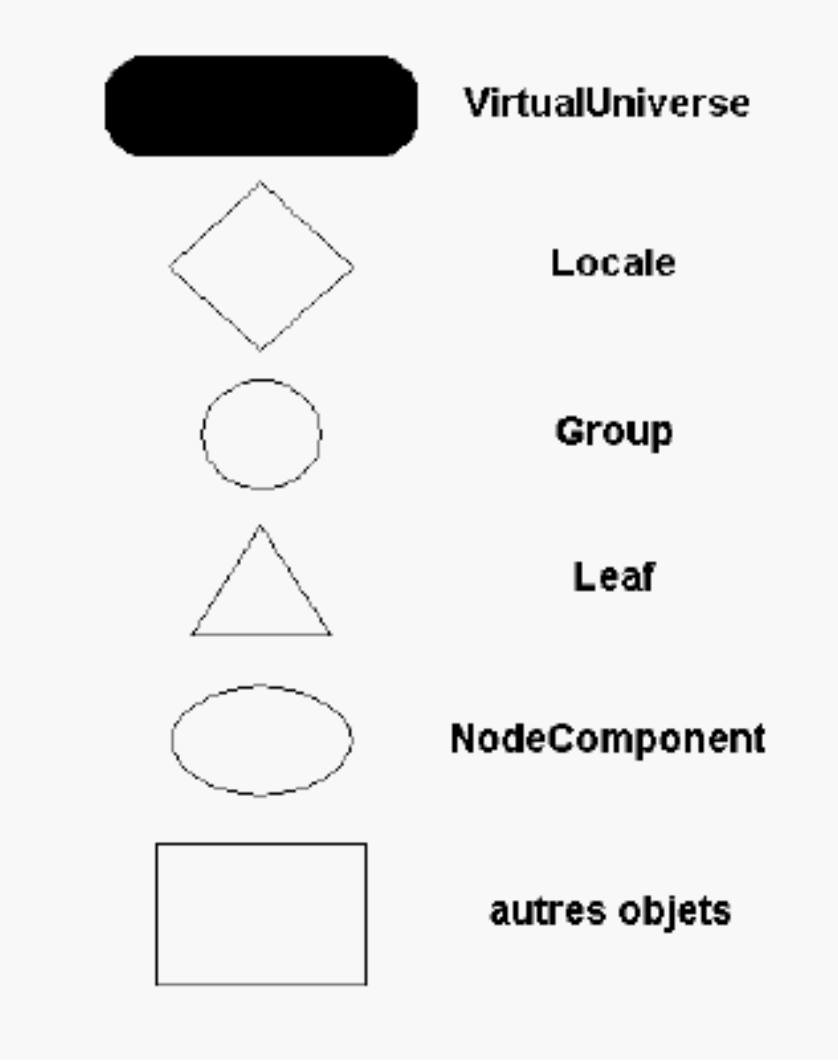
Univers : terminologie

- **Univers Virtuel** (Virtual Universe)
  - Collection de graphes de scène
    - Un par application
    - C'est le sommet du graphe
- **Repère** (Locale)
  - Généralement un par univers
- **Graphe de scènes** (Branch Graph)
  - généralement, plusieurs graphes par repère

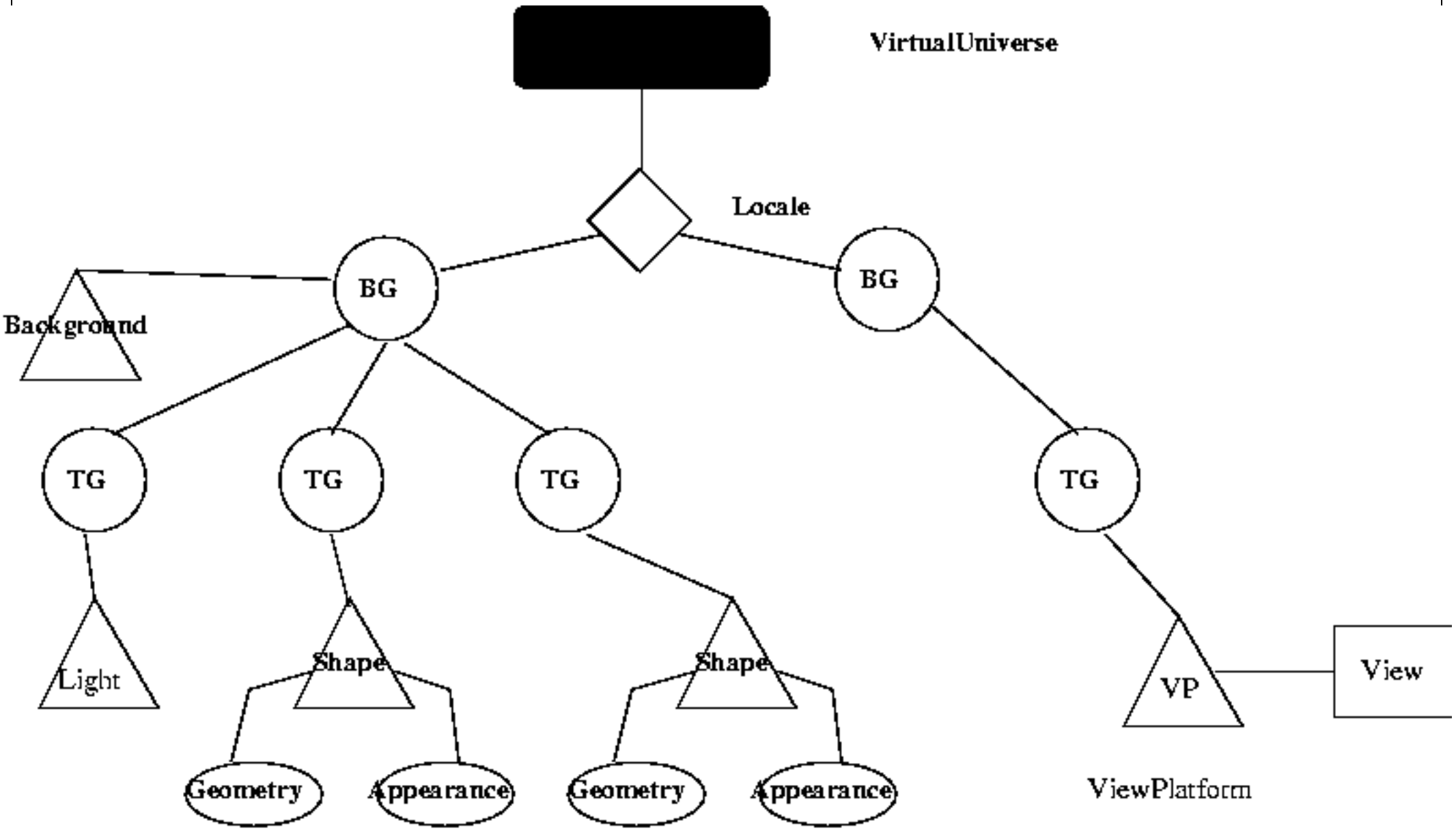
Graphe de scènes : terminologie

- Branches de scènes :
  - Formes, lumières . . .
- Branche de visualisation
  - Génération d'images
  - Généralement une par univers

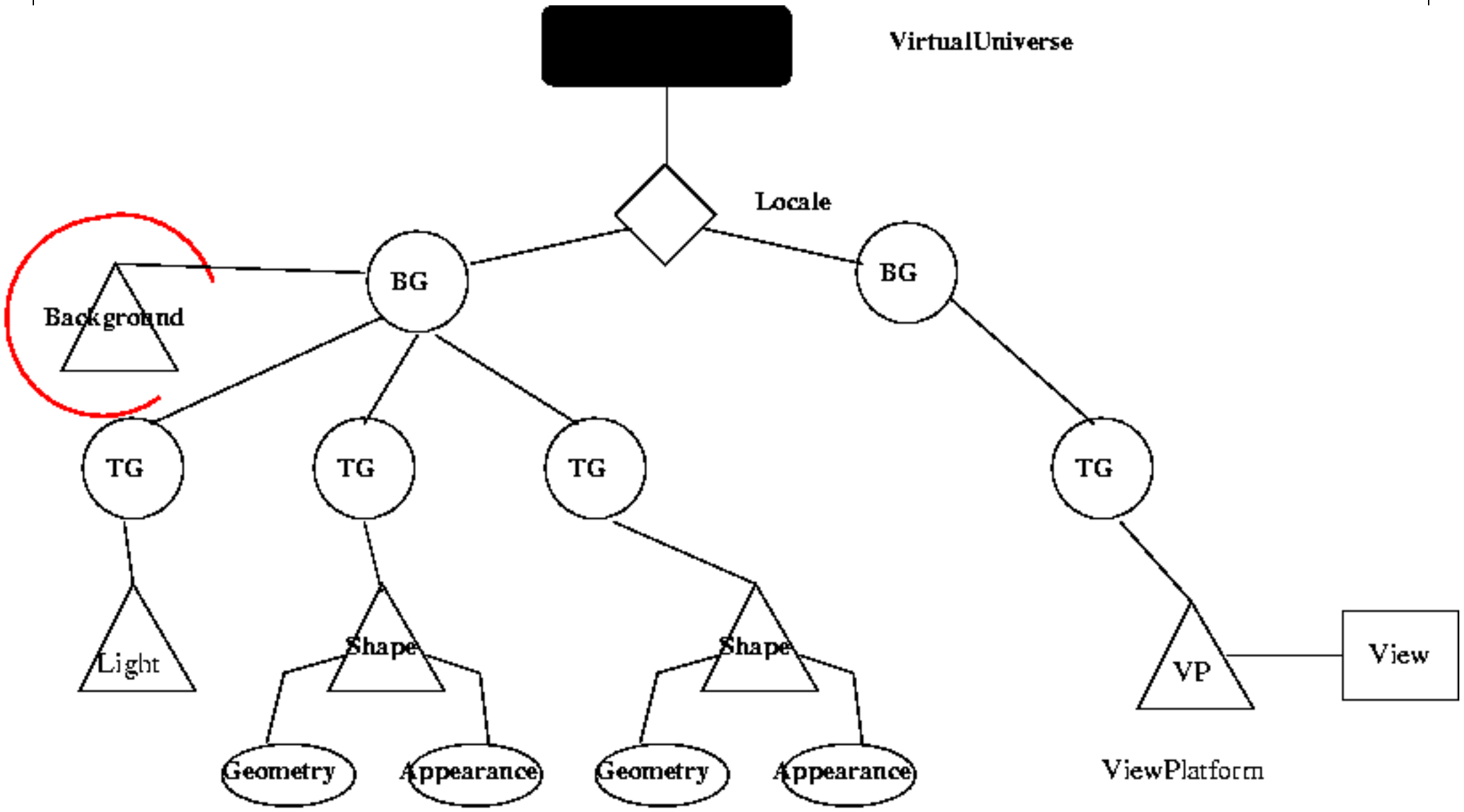
Graphe de scène : représentation graphique



Graphe de scène : représentation graphique



Background


VirtualUniverse
Locale
BG
BG
Background
TG
TG
TG
TG
Light
Shape
Shape
VP
View
Geometry
Appearance
Geometry
Appearance
ViewPlatform

Background

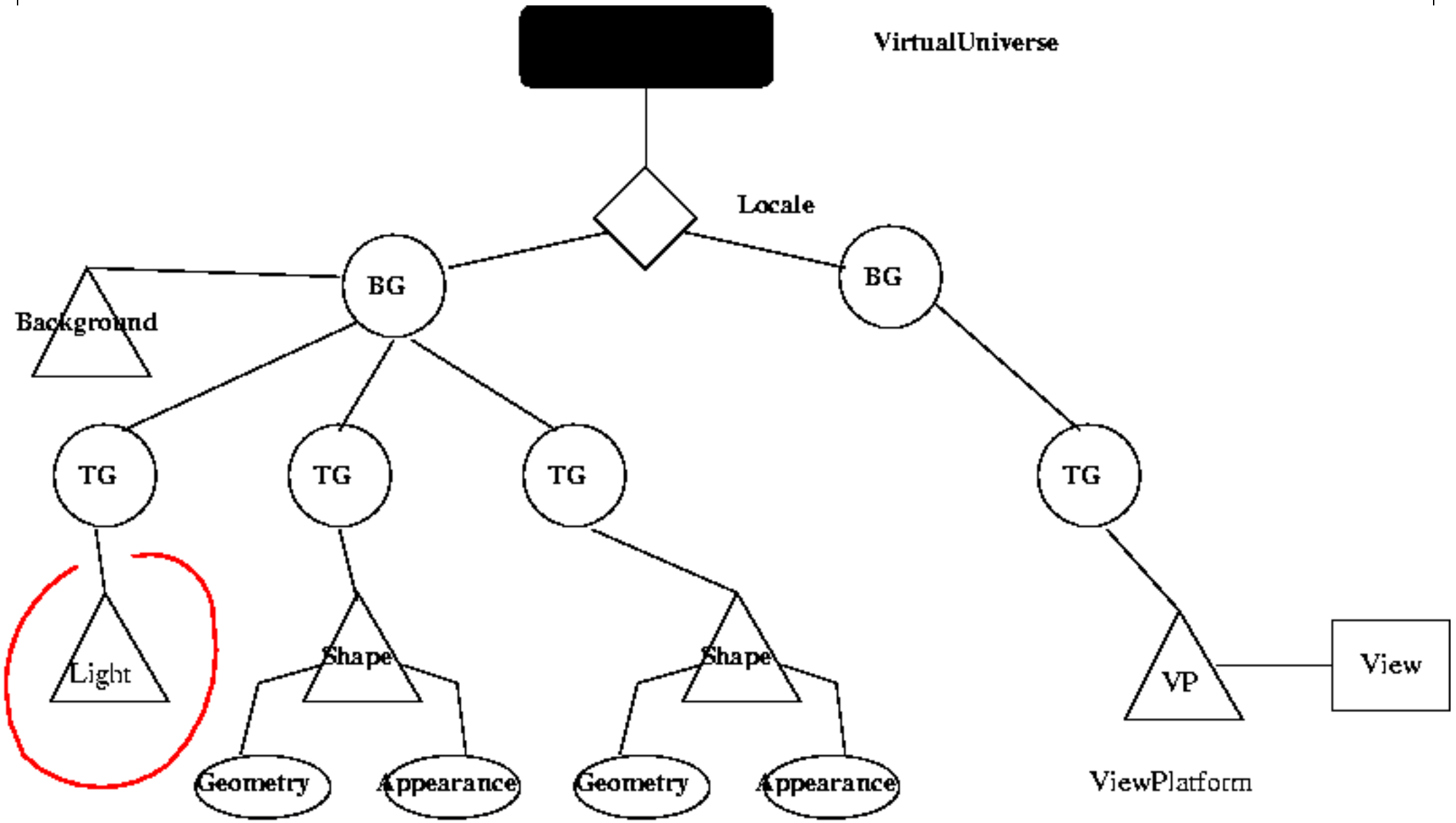
- Définit une couleur ou une image de fond
- Un ou plusieurs par graphe de scène :
  - Zone d'activation
  - Aucun fond actif $\Rightarrow$ fond noir
- *javax.media.j3d.Background*

Exemple

```
BoundingSphere bounds =
    new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
Color3f bgColor = new Color3f(0.05f, 0.05f, 0.7f);
Background bgNode = new Background(bgColor);
bgNode.setApplicationBounds(bounds);
```

- Il reste à mettre le noeud dans le graphe.

Lights


VirtualUniverse
Locale
BG
BG
Background
TG
TG
TG
TG
Light
Shape
Shape
VP
View
Geometry
Appearance
Geometry
Appearance
ViewPlatform

Lights

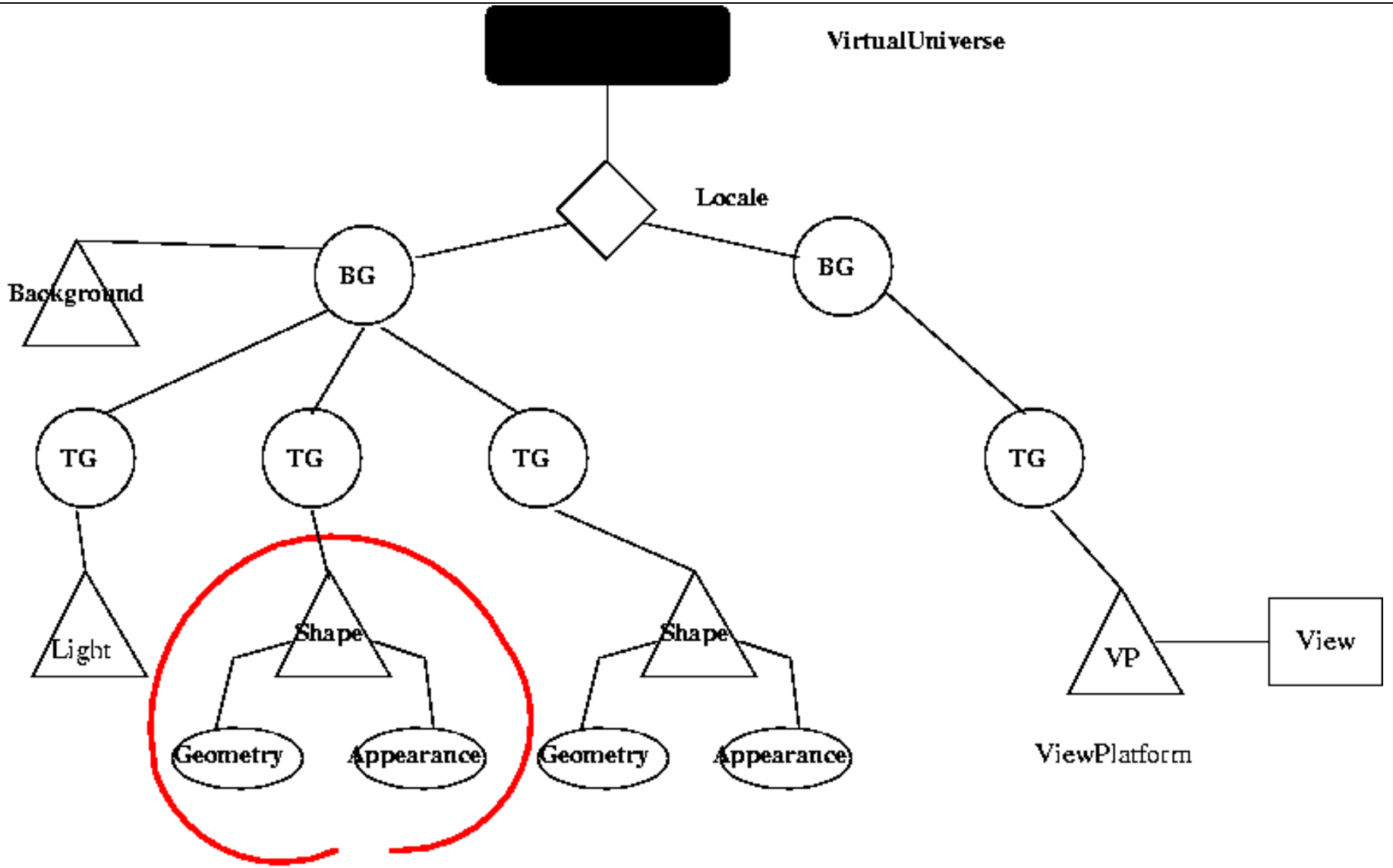
- Classe abstraite :
  - AmbientLight
  - DirectionalLight
  - PointLight
    - Spotlight
- Coûteux
- OpenGL : au plus 7 lumières.
- *javax.media.j3d.Light*

Exemple

```
Color3f light1Color = new Color3f(1.0f, 1.0f, 0.9f);
Vector3f light1Direction  = new Vector3f(4.0f, -7.0f, -12.0f);
Color3f ambientColor = new Color3f(0.1f, 0.1f, 0.1f);

AmbientLight ambientLightNode = new AmbientLight(ambientColor);
ambientLightNode.setInfluencingBounds(bounds);

DirectionalLight light1
    = new DirectionalLight(light1Color, light1Direction);
light1.setInfluencingBounds(bounds);
```

- Il reste à mettre le noeud dans le graphe.

Shape 3D

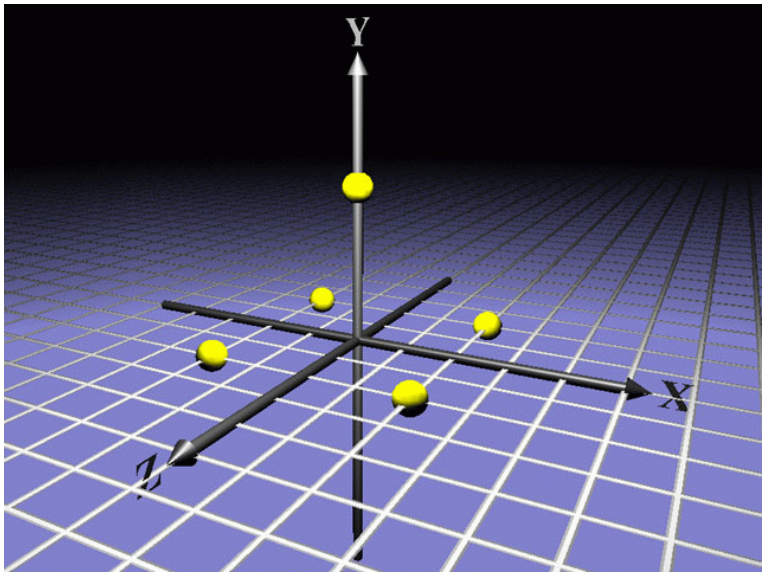

VirtualUniverse
Locale
BG
BG
Background
TG
TG
TG
TG
Light
Shape
Shape
VP
View
Geometry
Appearance
Geometry
Appearance
ViewPlatform

Shape 3D

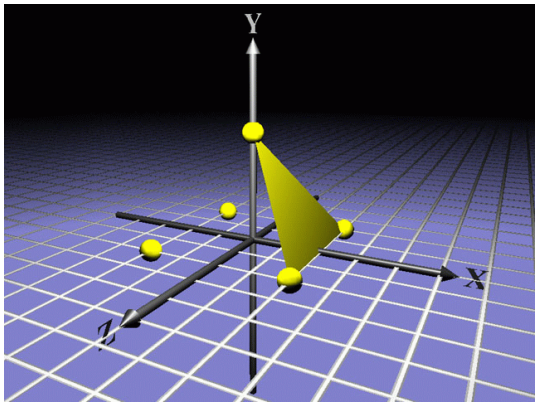
- Shape 3D :
  - Noeud feuille.
  - Deux attributs :
    * Géométrie
    * Apparence

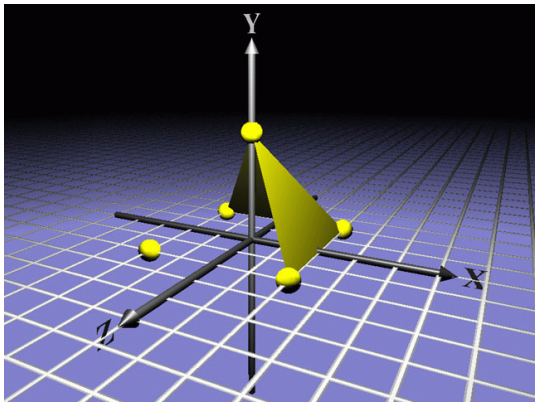
Géométrie

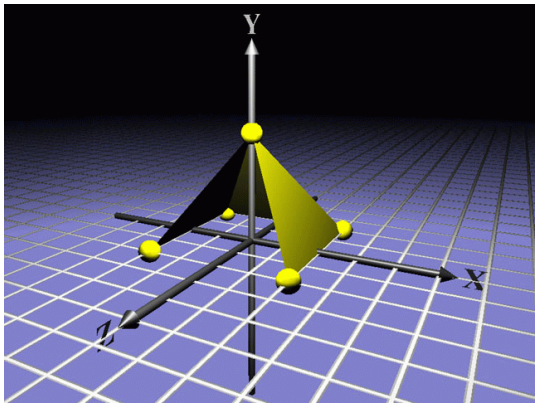
- De manière très générale :

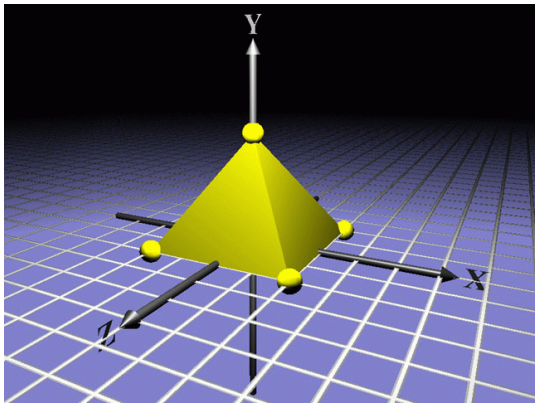


Géométrie

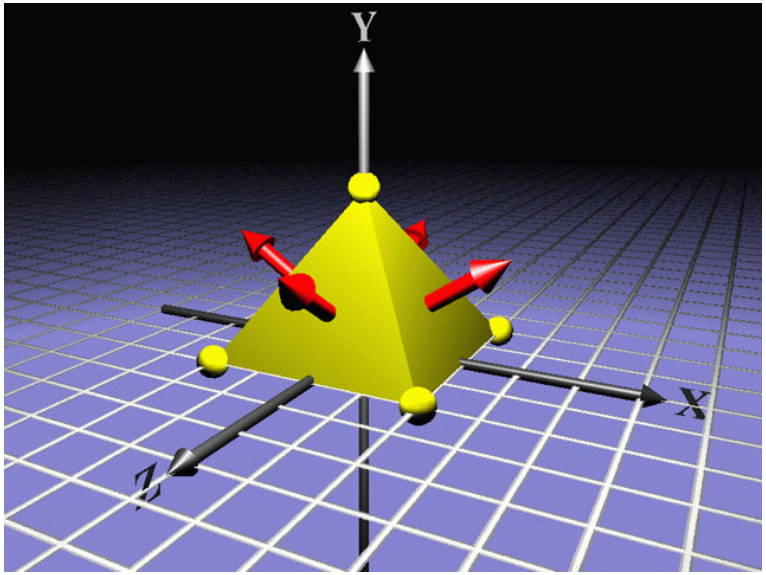








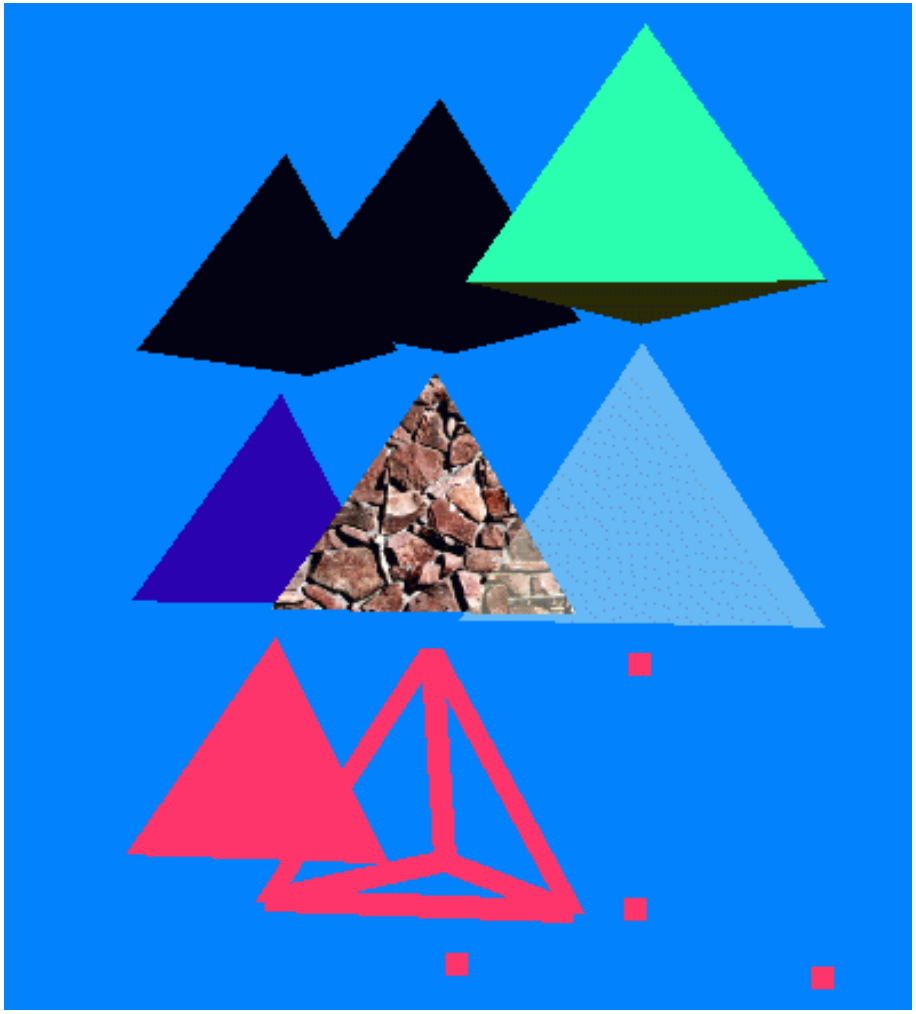
Géométrie



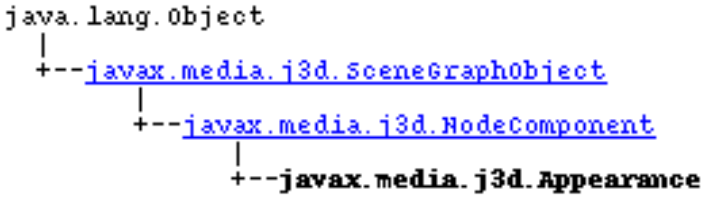
Appparence

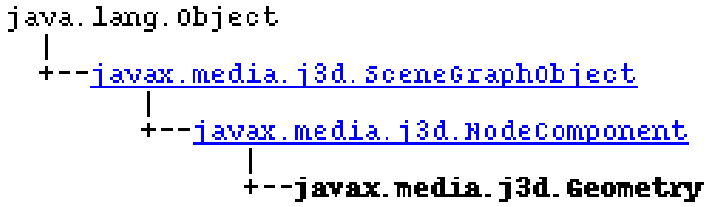
- Contrôle l'aspect de l'objet
  - Couleur.
  - Matériaux (éclairage).
  - Transparence.
  - Texture
  - ...

Exemple



Classes Java





Chargement

- Shape3D $\Rightarrow$ construction de la forme par programme
- Possibilité de chargement de fichiers de données
- Par défaut :
  - format OBJ
  - format Lightwave
- Sur internet :
  - vrml 97
  - 3DS Max
  - ac3d
  - ...

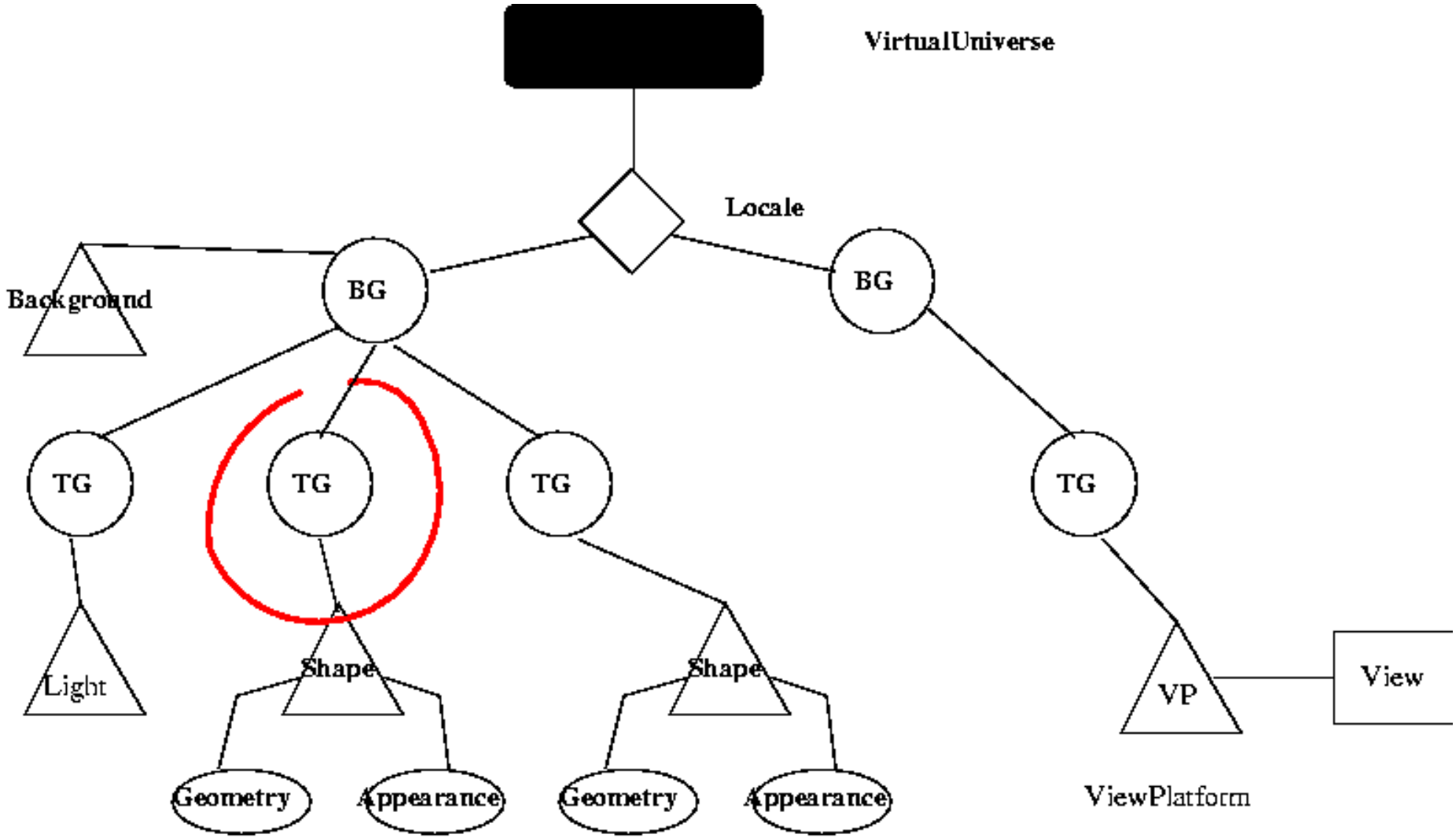
Exemple

```
import com.sun.j3d.loaders.objectfile.ObjectFile ;
import com.sun.j3d.loaders.* ;
import java.io.FileNotFoundException ;


public Node loadObject()
  {
     ObjectFile f = new ObjectFile() ;

     try {
        s = f.load("maison.obj") ;
     } catch (FileNotFoundException error)
     {
       System.err.println(error) ;
       System.exit(1) ;
     }
     return s.getSceneGroup() ;
  }
```

Transform Group

- Noeud groupe :
  - Plusieurs fils possibles
- Noeud Transform :
  - Positionne les fils dans l'espace
  - Possède une Matrice $4x4$ de positionnement *Transform3D*
- *javax.media.j3d.TransformGroup*

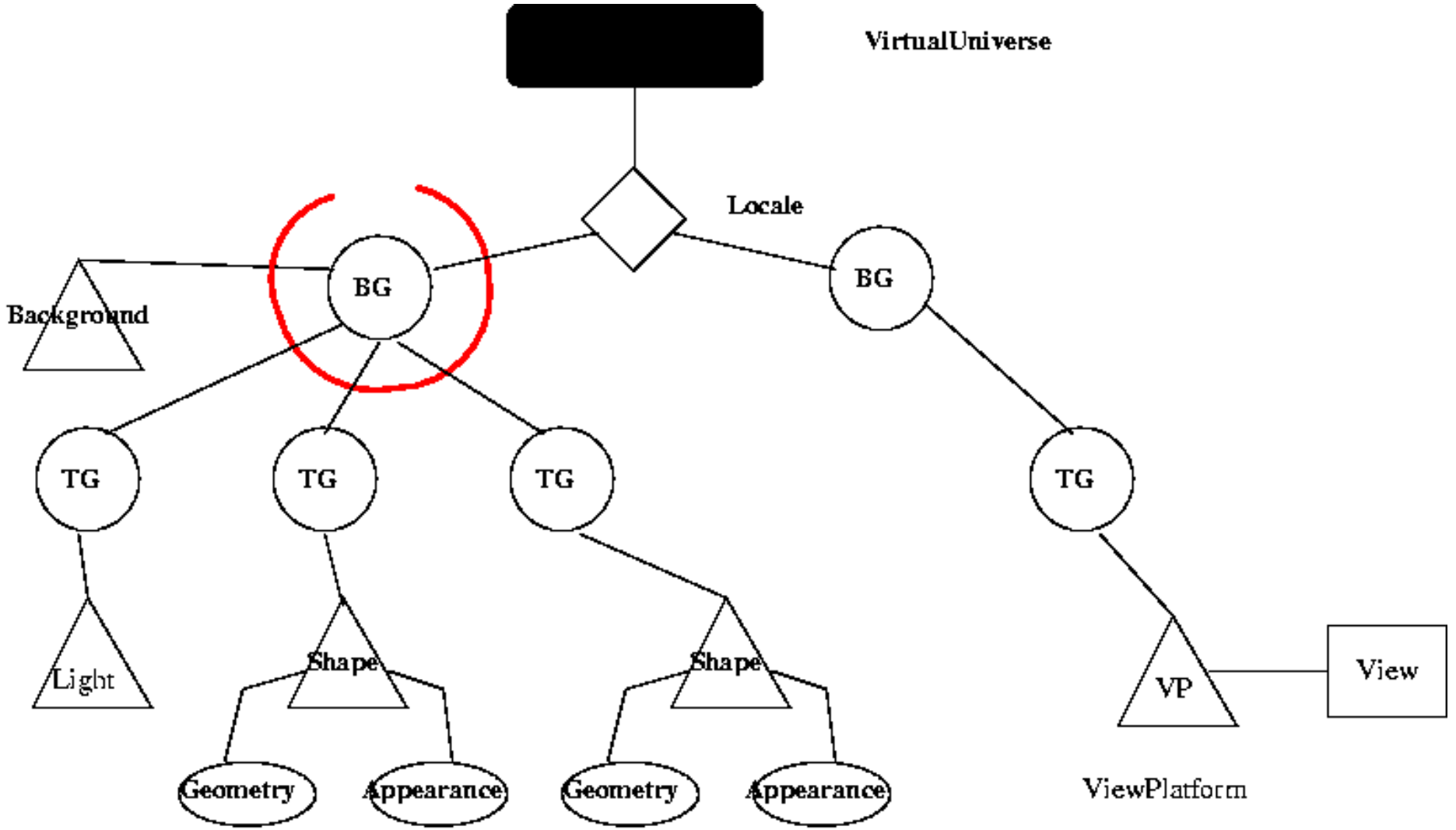
Transform3D



Transform3D

- Représente :
  - Une translation
  - Une rotation
  - Un changement d'échelle
- Méthodes principales :
  - *setTranslation(Vector3d trans)*
  - *setRotation(Quat4f q1) ;*
    - Quaternion = axe + angle de rotation.
  - *setScale(Vector3d scale) ;*
- *javax.media.j3d.Transform3D*

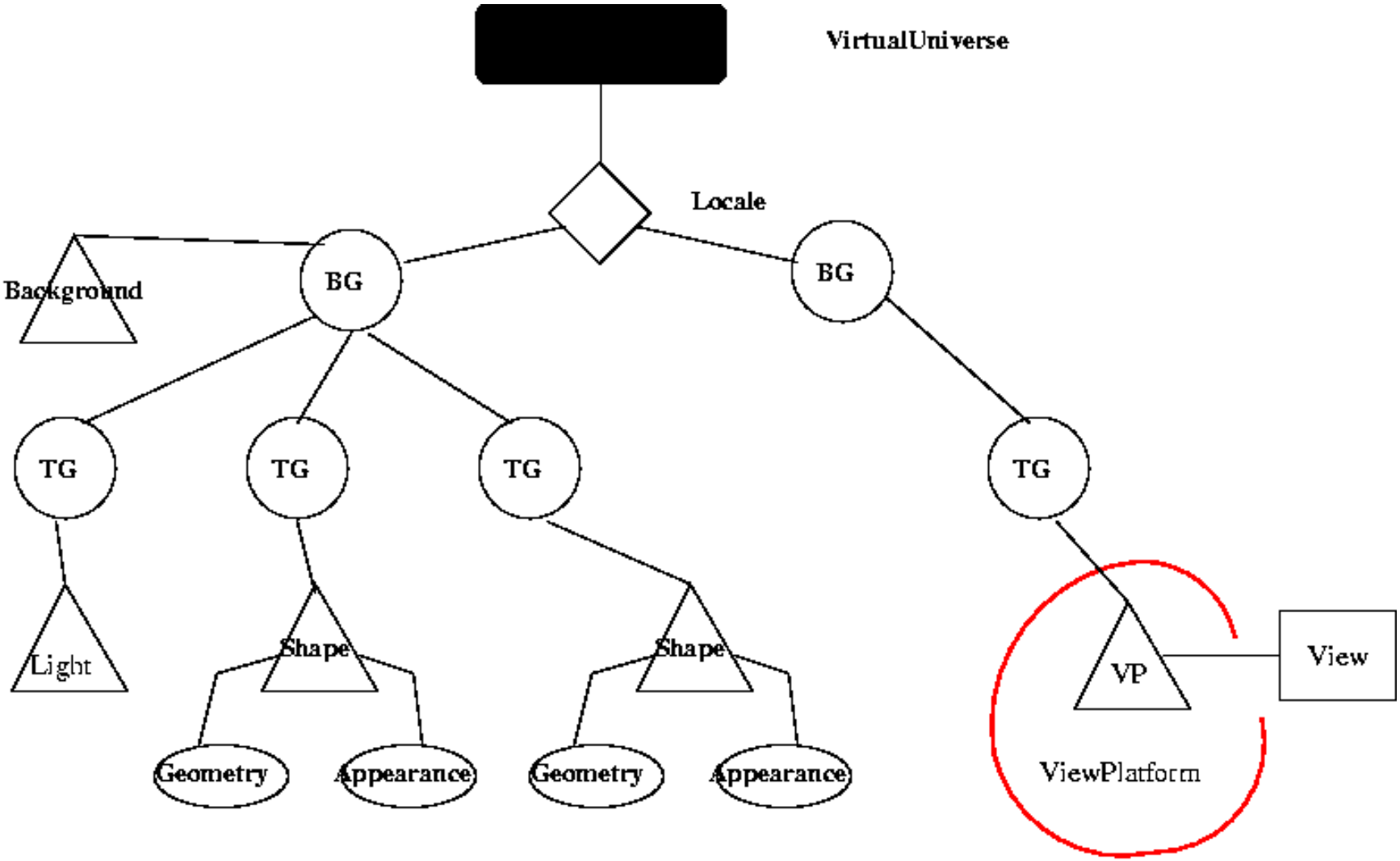
BranchGroup



BranchGroup

- Noeud groupe :
  - Contient un ou plusieurs fils
- Quelques méthodes (*héritées de Group*):
  - *addChild(Node child)*
  - *insertChild(Node child, int index)*
  - *removeChild(int index)*
- Seul noeud pouvant être inséré dans un noeud *Locale*.
- *javax.media.j3d.BranchGroup*

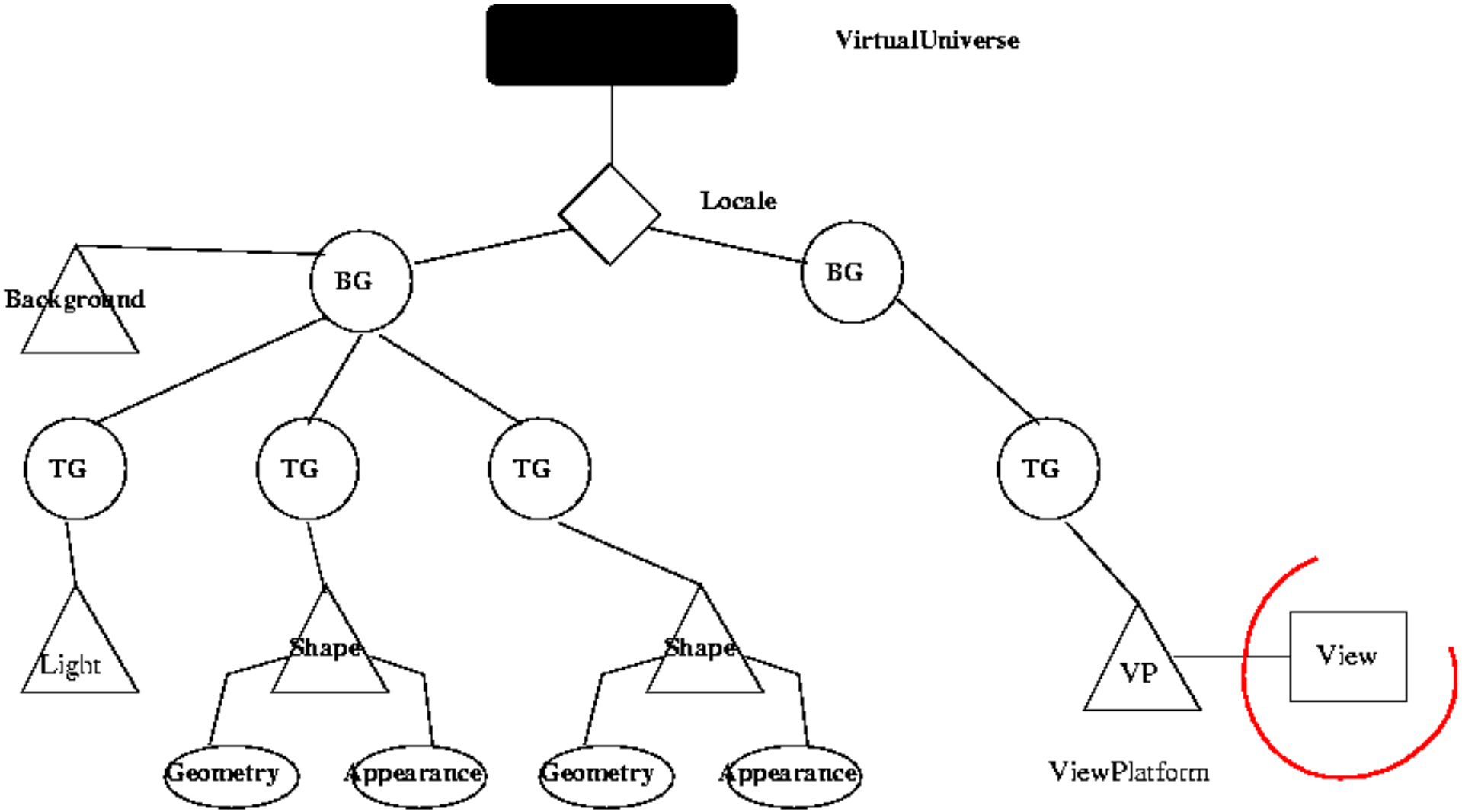
ViewPlatform



ViewPlatform

- Noeud feuille.
- *Caméra* à travers laquelle on navigue dans l'univers.
- Navigation en changeant le noeud *Transform* père de *ViewPlatform*
- *javax.media.j3d.ViewPlatform*

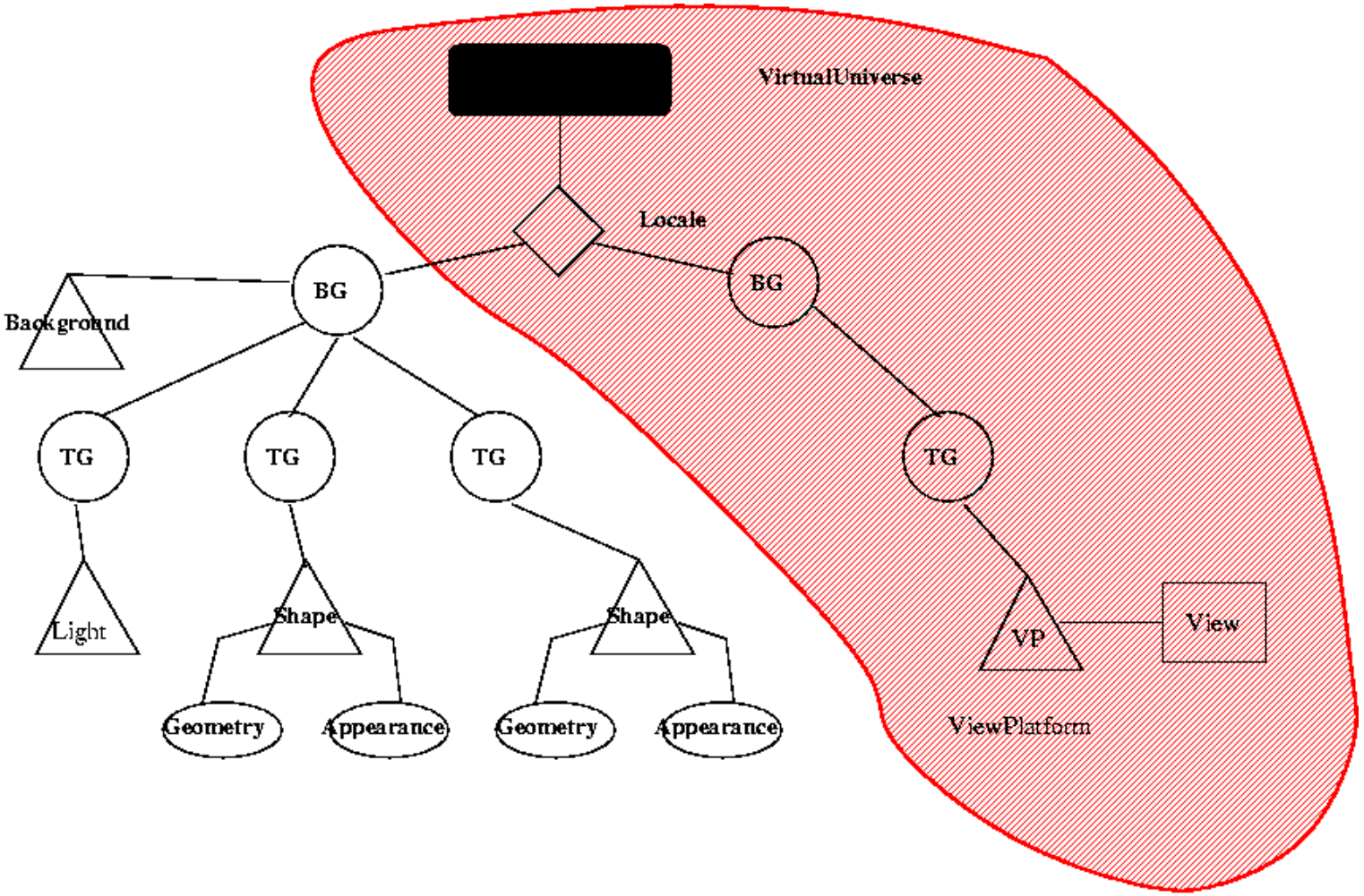
View


VirtualUniverse
Locale
BG
BG
Background
TG
TG
TG
TG
Light
Shape
Shape
VP
View
Geometry
Appearance
Geometry
Appearance
ViewPlatform

View

- Passage d'une scène 3D à une image 2D
- Affichage de l'image 2D dans un *Canvas3D*
- *Canvas3D*
  - Extension du *Canvas* d'*awt*
- Classes :
  - *javax.media.j3d.View*
  - *javax.media.j3d.Canvas3D*

SimpleUniverse



SimpleUniverse

- Objectif :
  - Simplifier la création d'une application Java3D
  - Prend en charge une partie du graphe de scène

SimpleUniverse

- ViewingPlatform :
  - ViewPlatform
  - Géométrie associée :
    - Tableau de bord de voiture
    - etc
- *ViewPlatform getViewPlatform()*

SimpleUniverse

- Viewing :
  - View (Canvas3D)
  - ViewerAvatar
  - etc
- *View getView()*

SimpleUniverse

- Création :
  - *new SimpleUniverse()* :
    - crée une *Frame awt* composé d'un canvas 3D.
  - *new SimpleUniverse(Canvas3D c)* :
    - Permet d'insérer une vue 3D dans une IHM complète

SimpleUniverse

- Quelques méthodes :
  - *void addBranchGraph(BranchGroup g)*
  - *ViewingPlatform getViewingPlatform()*
  - *ViewerPlatform getViewer()*
- *com.sun.j3d.utils.universe.SimpleUniverse*

Terminologie

- Un noeud peut être :
  - Vivant (Live)
  - Compilé (Compiled)

Vivant

- Noeuds d'un *BranchGroup* lors de son ajout à un noeud *Locale* (participent au rendu)
- Exemple :

```
myBranch = new BranchGroup( );
myBranch.addChild( myShape );
myLocale.addBranchGraph( myBranch ); // make live!
```

- Les noeuds ne sont plus vivants lors du retrait :

```
myLocale.removeBranchGraph( myBranch );// not live
```

- Vérification si le noeud est vivant :

```
boolean isLive() ;
```

Compilé

- Optimisation du rendu
- Méthode de *BranchGroup* :

```
void compile() ;
```

- Les noeuds doivent être compilés **avant** d'être rendus vivants :

```
BranchGroup myBranch = new BranchGroup( );
myBranch.addChild( myShape );
myBranch.compile( );
myLocale.addBranchGraph( myBranch );
```

Attributs des noeuds

- Lecture ou écriture des attributs :
  - **Avant** d'être vivant ou compilé.
- Après :
  - Doit être autorisé

  $\Rightarrow$ **Capacités** d'un noeud
- Peu de capacités $\Rightarrow$ plus d'optimisation à la compilation.

Capacités

- Méthodes de *SceneGraphObject*

```
void setCapability( int bit )
void clearCapability( int bit )
boolean getCapability( int bit )
```

Exemple : Shape3D

- Capacités :
  - ALLOW_APPEARANCE_READ
  - ALLOW_APPEARANCE_WRITE
  - ALLOW_GEOMETRY_READ
  - ALLOW_GEOMETRY_WRITE
  - ...

D

Exemple

```
import java.awt.* ;
import java.awt.BorderLayout;
import java.awt.event.*;
import java.awt.GraphicsConfiguration;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.geometry.ColorCube;
import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import javax.vecmath.*;
import com.sun.j3d.loaders.objectfile.ObjectFile ;
import com.sun.j3d.loaders.* ;
import java.io.FileNotFoundException ;
```

```
public class Beethoven extends Frame {

    protected BranchGroup scene ;
    protected SimpleUniverse u ;

    public BranchGroup createSceneGraph() {
// Create the root of the branch graph
BranchGroup objRoot = new BranchGroup();

// Create a bounds for the background and lights
BoundingSphere bounds =
    new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
// Set up the background Color
Color3f bgColor = new Color3f(0.05f, 0.05f, 0.7f);
Background bgNode = new Background(bgColor);
bgNode.setApplicationBounds(bounds);
objRoot.addChild(bgNode);

Color3f light1Color = new Color3f(1.0f, 1.0f, 0.9f);
Vector3f light1Direction  = new Vector3f(4.0f, -7.0f, -12.0f);
Color3f light2Color = new Color3f(0.3f, 0.3f, 0.4f);
Vector3f light2Direction  = new Vector3f(-6.0f, -2.0f, -1.0f);
Color3f ambientColor = new Color3f(0.1f, 0.1f, 0.1f);
```

```
// Second, define the ambient light, and insert it in the branch
AmbientLight ambientLightNode = new AmbientLight(ambientColor);
ambientLightNode.setInfluencingBounds(bounds);
objRoot.addChild(ambientLightNode);

// Lastly, define the directional lights and insert it
DirectionalLight light1
    = new DirectionalLight(light1Color, light1Direction);
light1.setInfluencingBounds(bounds);
objRoot.addChild(light1);

DirectionalLight light2
    = new DirectionalLight(light2Color, light2Direction);
light2.setInfluencingBounds(bounds);
objRoot.addChild(light2);
```

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

Scene s ;
// Create a simple Shape3D node; add it to the scene graph.
ObjectFile f = new ObjectFile() ;

try {
    s = f.load("beethoven.obj") ;
    objTrans.addChild(s.getSceneGroup()) ;
} catch (FileNotFoundException error)
    {
System.err.println(error) ;
System.exit(1) ;
    }

// allows to read the bounding sphere
objRoot.setCapability(BranchGroup.ALLOW_BOUNDS_READ) ;
        // Have Java 3D perform optimizations on this scene graph.
        objRoot.compile();

return objRoot;
    }
```

```
    public Beethoven() {
super("Beethoven") ;
setLayout(new BorderLayout());
        GraphicsConfiguration config =
           SimpleUniverse.getPreferredConfiguration();

Canvas3D c = new Canvas3D(config);
add("Center", c);

// Create a simple scene and attach it to the virtual universe
scene = createSceneGraph();
u = new SimpleUniverse(c);

        // This will move the ViewPlatform back a bit so the
        // objects in the scene can be viewed.

        u.getViewingPlatform().setNominalViewingTransform();
viewAll() ;

u.addBranchGraph(scene);

pack() ;
setSize(300,300) ;
    }
```

```
    {
BoundingSphere mybounds = (BoundingSphere) scene.getBounds() ;
double t[] = new double[3] ;
Point3d center=new Point3d() ;
mybounds.getCenter(center) ;
center.get(t) ;

// on positionne la camera de facon a voir la sphere
TransformGroup cameraPosition = u.getViewingPlatform().
    getViewPlatformTransform() ;

Transform3D trans = new Transform3D() ;
trans.setTranslation(new Vector3f((float)t[0],
  (float)t[1],
  (float)t[2]+(float) mybounds.getRadius()*3));
cameraPosition.setTransform(trans) ;
    }

    //
    // The following allows Beethoven to be run as an application
    // as well as an applet
    //
    public static void main(String[] args) {
new Beethoven().show() ;
    }
}
```