Qu'est-ce-que Java3D ?

API en Java utilisant les techniques de programmation graphique 3 dimensions. Le package J3D utilise notamment les graphes de scènes (VRML) et la spacialisation du son (OpenGL, Direct3D ...)

Le package Java3D ne fait pas partie du standard Java

Indépendant du Hardware

Deux parties :

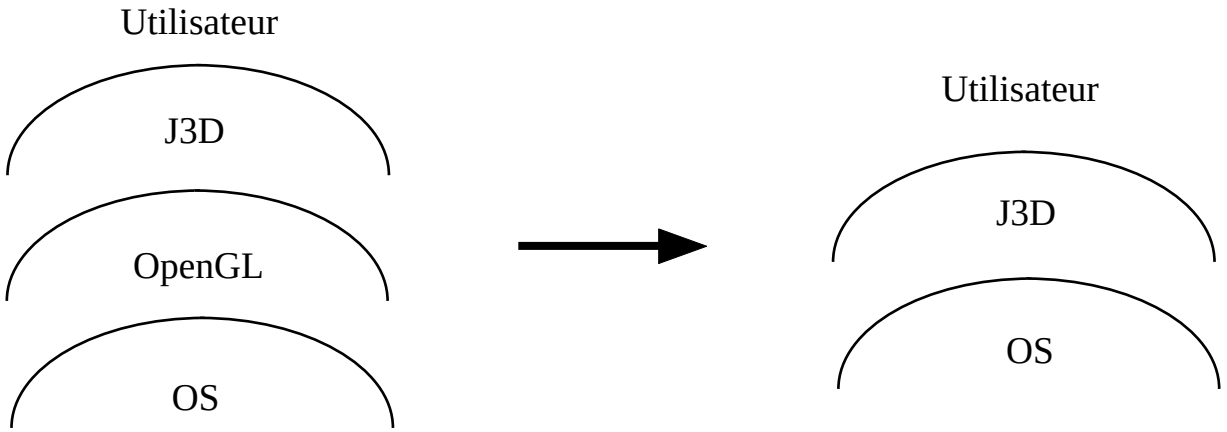
Les spécifications de l'API

L'implémentation (Avril 99 par Sun)
Sun encourage les développeurs à implémenter directement J3D au niveau du Hardware (carte dédiée)

Différences entre Java3D et OpenGL, Direct3D, PHIGS ...

API Java3D se situe au même niveau que OpenGL, PHIGS ...
et peut être utilisée sur les mêmes systèmes (UNIX, Windows ...)

Java3D est conçu pour être directement implémenté au niveau du Hardware mais à l'heure actuelle, il utilise Open GL pour faire l'interface entre le langage Java et la machine



Performances

Java3D est conçu pour utiliser au mieux les performances graphiques du Hardware, donc ces performances dépendent de la machine (processeur) mais surtout de la qualité de la carte graphique

on peut également utiliser des logiciels permettant d'améliorer les performances

La page de référence est : **http://java.sun.com/products/java-media/3D/index.html**

A partir de cette page vous pouvez charger le SDK de Java 3D par le Java Developper Connection (JDC), l'inscription est gratuite.
Le SDK pour Java 3D est disponible pour Solaris2 et Windows 95/NT.

après installation :

- Extraire les exemples et les sources utiles
  cd REPERTOIRE_D_INSTALLATION
  jar xvf j3dexamples.jar
  jar xvf j3dutilssrc.jar

- Pour la documentation javadoc de Java 3D
  jar xvf javadoc_core.jar
  jar xvf javadoc_utils.jar

- Pour les exemples VRML et documentation javadoc de VRML
  jar xvf v97examples.jar
  jar xvf javadoc_vrml97.jar

pour tester, aller dans les répertoires de programmes test :

cd REPERTOIRE_D_INSTALLATION/Java3D/examples/REP_TEST (pourWindows)
javac *.java
java PROGRAMME_TEST ou appletviewer FICHIER_TEST.html

Pour exécuter ces programmes, il faut parfois les lancer avec les options :
pour java : **java -mx64m**
pour appletviewer : **appletviewer -J-mx64m**
Bien avoir le répertoire . dans CLASSPATH.

Les applications Java J3D fonctionnent ainsi que les applets avec appletviewer.
Les applets ne fonctionnent pas encore dans les browsers.

Pour Windows

Windows 95 ou NT 4.0, le JDK 1.2 beta 3 (ou suivant) de Javasoft. éventuellement de OpenGL 1.1 ou d'un accélérateur OpenGL 1.1 disponible à partir de la page de chargement de Java 3D.

Variables d'environnement :
- Ajouter dans PATH le répertoire : REPERTOIRE_D_INSTALLATION\bin
- Ajouter dans CLASSPATH les répertoires
  REPERTOIRE_D_INSTALLATION\lib\appext\j3dutils.jar
  REPERTOIRE_D_INSTALLATION\lib\appext\vrml97.jar
  REPERTOIRE_D_INSTALLATION\lib\sysext\j3dcore.jar
  REPERTOIRE_D_INSTALLATION\lib\sysext\vecmath.jar
  REPERTOIRE_D_INSTALLATION\lib\sysext\j3daudio.jar

Remarque :

Sous Windows 95, la taille des variables d'environnement (comme CLASSPATH) peut être limitée. Pour permettre une longue chaîne dans une variable : sélectionner Properties sur l'icône MSDOS qui lance command.com dans l'onglet "Programme", remplacer la Ligne de commande contenant C:\WINDOWS\COMMAND.COM par C:\WINDOWS\COMMAND.COM /e:4096

Pour Solaris

Solaris 2.5 ou plus
JDK 1.2 beta 3 (ou suivant) de Javasoft.
OpenGL 1.1 pour Solaris avec le patch 106022-02

Variables d'environnement :

- Ajouter dans LD_LIBRARY_PATH le répertoire :
  REPERTOIRE_D_INSTALLATION/Java3D/bin/sparc/green_threads

- Ajouter dans CLASSPATH les répertoires
  REPERTOIRE_D_INSTALLATION/Java3D/lib/appext/j3dutils.jar
  REPERTOIRE_D_INSTALLATION/Java3D/lib/appext/vrml97.jar
  REPERTOIRE_D_INSTALLATION/Java3D/lib/sysext/j3dcore.jar
  REPERTOIRE_D_INSTALLATION/Java3D/lib/sysext/vecmath.jar
  REPERTOIRE_D_INSTALLATION/Java3D/lib/sysext/j3daudio.jar

Structure d'un programme Java3D

Un programme Java3D décrit un graphe de scène ou "monde virtuel". Celui-ci indique :

les objets 3D composant la scène

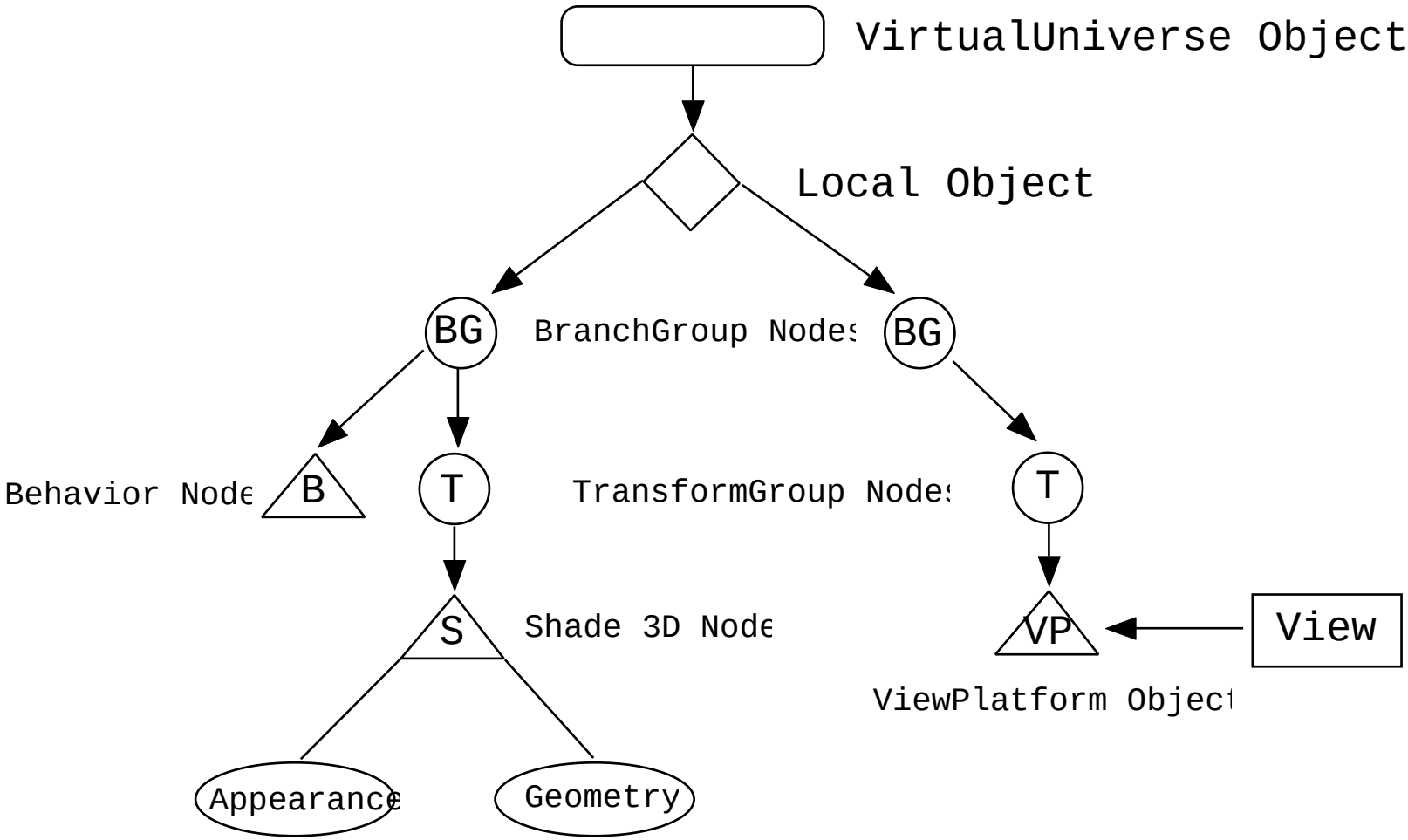
les attributs de chacun de ces objets (formes, couleur, texture, réflexion du matériau, ...) ainsi que leurs attributs de comportement (positionnement, mouvement, ...)

la perspective de visualisation (le point d'observation, l'éclairage, ...)

Ce graphe est décrit par une arborescence qui est parcouru par le moteur 3D lors de l'exécution Cette arborescence commence par un objet **`VirtualUniverse`** qui contient un objet **`Locale`** décrivant un système de coordonnées haute résolution. Dans ce système de coordonnées, sont alors ajoutés les objets 3D (**`Shape3D`**) en décrivant leur forme (**`Geometry`**) et leur **`Appearance`** (couleur, texture, etc.), leurs manipulations possibles par l'utilisateur du programme (**`Behavior`**) ainsi que le point de vue d'observation (**`ViewPlatform`**).

Il peut, dans un programme Java3D, y avoir plusieurs objets VirtualUniverse et plusieurs objets Locale rattachés à chacun d'entre eux mais très souvent un seul suffit.

Structure d'un programme Java 3D



Un exemple Java3D : HelloUniverse

HelloUniverse.html

```
 <APPLET CODE=HelloUniverse.class WIDTH=1000
                                 HEIGHT=650>
 </APPLET>
```

L'essentiel de ce code est :

```
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.*;

public class HelloUniverse extends Applet {
  public HelloUniverse() {
     setLayout(new BorderLayout());
     Canvas3D c = new Canvas3D(null);
     add("Center", c);

 // Creation d'une scène
 // Ajout de cette scène dans univers virtuel
 // avec un système de coordonnées par défaut.
     BranchGroup scene = createSceneGraph();
     SimpleUniverse u = new SimpleUniverse(c);
     u.addBranchGraph(scene);
  }
```

Un exemple Java3D : HelloUniverse

```
public BranchGroup createSceneGraph() {
// Creation de la racine de la branche principale
  BranchGroup objRoot = new BranchGroup();

// Creation de l'unité de traitement (transform group)
// du futur objet.
// Ajout de cette unité dans le BranchGroup
  TransformGroup objTrans = new TransformGroup();
  objRoot.addChild(objTrans);

// Description de l'unité de traitement :
// c'est une rotation suivant l'axe des y affine
// qui commencera dès le début du programme et
// qui est exécutée pendant toute sa durée.
// Ajout de cette rotation dans l'inuté de traitement.
  Transform3D yAxis = new Transform3D();
  Alpha rotationAlpha = new Alpha(-1,
    Alpha.INCREASING_ENABLE,
    0, 0,
    4000, 0, 0,
    0, 0, 0);

   RotationInterpolator rotator =
   new RotationInterpolator(rotationAlpha,objTrans,
       yAxis, 0.0f, (float) Math.PI*2.0f);
       objTrans.addChild(rotator);

        return objRoot;
  }
 }
```

Bibliographie Java3D

Livres

The Java 3D API Specification ; Henry Sowiral, Kevin Rushforth, Michael Deering, ed Addison Wesley. ISBN 0-201-32576-4. Collection The Java Series.

Java for 3D and VRML worlds ; Rodger Lea, Kouichi Matsuda, Ken Miyashita, ed. New Riders. ISBN 1-56205-689-1

URL Remarquables

http://java.sun.com/javaone/javaone98/tracksTOC.html#06
Les conférences de JavaOne (Mars 1998) sur Java Media (Java 2D, 3D, Sound, Speech).

http://java.sun.com/products/java-media/3D/
La page de départ pour la technologie Java 3D.

http://www.sun.com/desktop/java3d/demos/
Des dévellopement important avec Java 3D. Moteurs pour la réalité virtuelle, application médicale, algorithme de lancé de rayon, moteur de création de monde VRML, ...

Bibliographie Java3D

Mailing list

java3d-interest
mailing list concernant Java3D. Pour y souscrire, envoyer un mail à javamedia-request@sun.com avec
subscribe java3d-interest
dans le corps du message.

news groups

Les newsgroups concernant Java sont de la forme comp.lang.java.* (environ une quinzaine) ainsi que fr.comp.lang.java.

SDK pour Java 3D

Sun SDK à http://developer.java.sun.com/developer/earlyAccess/java3D/
C'est au Java Developer Connection (JDC). L'inscription est gratuite.

Java et VRML

J3D : code compilé
VRML : code interprété

VRML Consortium et Sun ont travaillé ensemble pour écrire un package permettant d'interfacer Java3D et VRML

Deux manières d'interfacer Java et VRML :

- on utilise un visualiseur VRML : on peut appeler des applets Java dans le code VRML (EAI)
- on utilise un visualiseur indépendant (écrit en Java) : on peut appeler des scènes VRML dans le code Java

VRML : Généralités

Définition : Virtual Reality Modeling Language. En fait, une définition plus juste serait : format de description de scènes 3D interactives avec hyperlien pour l'Internet.

Principes :

unité de base est la **scène**
format des fichiers : texte
VRML 1.0 -> description de scènes statiques
depuis la version 2.0 -> possibilité d'animer les scènes

La notion de moteur graphique : la scène VRML doit être affichée et doit répondre aux caractéristiques qu'à souhaité mettre en oeuvre son auteur : c'est le rôle du moteur graphique.

Il fonctionne le plus souvent en co-navigateur avec Netscape ou Internet Explorer

Domaine d'utilisation : visualisation dans les domaines scientifiques et techniques, présentations mutimédias, pages Web, jeux ...

VRML : concepts de base

la notion de noeud : notion proche de la structure ou de la classe d'objet

Deux noeuds particulièrement intéressants : `Transform et Group`
Regroupement physique et logique de noeuds

Le noeud `Inline`
Appel de fichiers externes dans une scène (autre scène, image, son, vidéo, script ...)

Ces concepts apportent une modularité indispensable au développement d'applications importantes, fiables et structurées

VRML : exemple de scène statique


```
#VRML V2.0 utf8
Group {
   children [
      DEF Boule Transform {
        translation -1 0 0
        children [
           Shape {
             appearance Appearance   {
                material Material {
                   diffuseColor 1 0 0
                }
             }
             geometry Sphere {
                radius 1
             }
           }
        ]
      } # fin de Transform
      DEF Cube Transform {
        translation 1 0 0
        children [
           Shape {
             appearance Appearance {
                 material DEF mat Material {
                       diffuseColor 0 0 1
                       transparency 0.5
                 }
             }
             geometry Box { size 2 2 2 }
           }
        ]
      } # fin de Transform
    ] # fin de children
} # fin de Group
```

VRML : prototypage

La notion de prototype VRML (`PROTO` et `EXTERPROTO`) est proche de la notion de classe "abstract" en Java (classes dont seules les signatures des méthodes sont définies, pas les corps)

Il s'agit d'un fichier (dont la première ligne comporte le mot clé `PROTO`) contenant des définitions de nœuds, dont les attributs (`field` et `exposedField`) sont initialisés à des valeurs par défaut. Une scène (ou un autre prototype), peut inclure un prototype (mot clé `EXTERPROTO`), et surcharger les valeurs d'un ou plusieurs attributs définis dans le prototype initial grâce à la clause `IS`

Ce mécanisme permet d'offrir une interface unique pour paramétrer un ensemble d'objets encapsulés les uns dans les autres

VRML : prototypage exemple


```
#VRML V2.0 utf8

PROTO Point [
   field SFColor couleur 1 0 0
]
{
 Transform {
   children [
     Shape {
       Appearance Appearance {
         Material Material {
         diffuseColor IS couleur
         }
       }
       geometry Sphere { radius 1 }
     }
   ]
 }
}
```

La définition du prototype `Point`

```
#VRML V2.0 utf8

EXTERNPROTO Point [
Field SFColor couleur
] ["Point.wrl"]

Transform {
  Children [
    Point {
      couleur 0 0 1
    }
  ]
}
```

Une instance de `Point`

VRML : le système d'événements

Depuis la version 2.0, il est possible d'animer les objets, grâce à la mise en place d'un système d'événements

Certains objets ont la possibilité d'émettre et/ou de recevoir des événements (attributs `eventIn` et `eventOut`) vers d'autres objets de la scène. Il n'est pas possible de communiquer directement avec des composants extérieurs au monde VRML, excepté par l'intermédiaire d'un nœud particulier, le `TouchSensor`, permettant uniquement de récupérer les événements émis par la souris

En plus des attributs `eventIn` et `eventOut`, certains objets possèdent des attributs `exposedField`. Derrière ce terme, se cachent deux événements implicites :

un événement entrant (`set_`attr`)`, qui permet de modifier la valeur de l'attribut "attr" avec la valeur reçue ;

un événement sortant (attr_`changed`), qui permet d'émettre la valeur de l'attribut "attr" à chaque modification de celui-ci.

VRML : le système de routage

L'ensemble des événements est piloté par un système de routage (mots clé `ROUTE .. TO`) qui permet de propager un événement (sortant) d'un objet vers un autre objet (événement entrant). Grâce à ce mécanisme et à la richesse des types de nœuds, il est possible de créer des animations complexes

La seule manière de modifier la valeur d'un attribut en VRML est de lui envoyer un événement par ce système de routage.

Remarque importante : les événements entrant et sortant d'une clause `ROUTE .. TO` doivent impérativement être du même type (par exemple booléen).

VRML : événement et routage -> exemple

L'exemple montre comment on peut allumer un projecteur (nœud `PointLight`) par un clic souris. Dans cet exemple, il nous faut définir une zone sensible au curseur de la souris (nœud `TouchSensor`). Nous avons choisi une sphère.

Cet exemple illustre l'utilisation des attributs `eventIn` et `eventOut`.

Le mécanisme est identique avec les attributs `exposedField.`

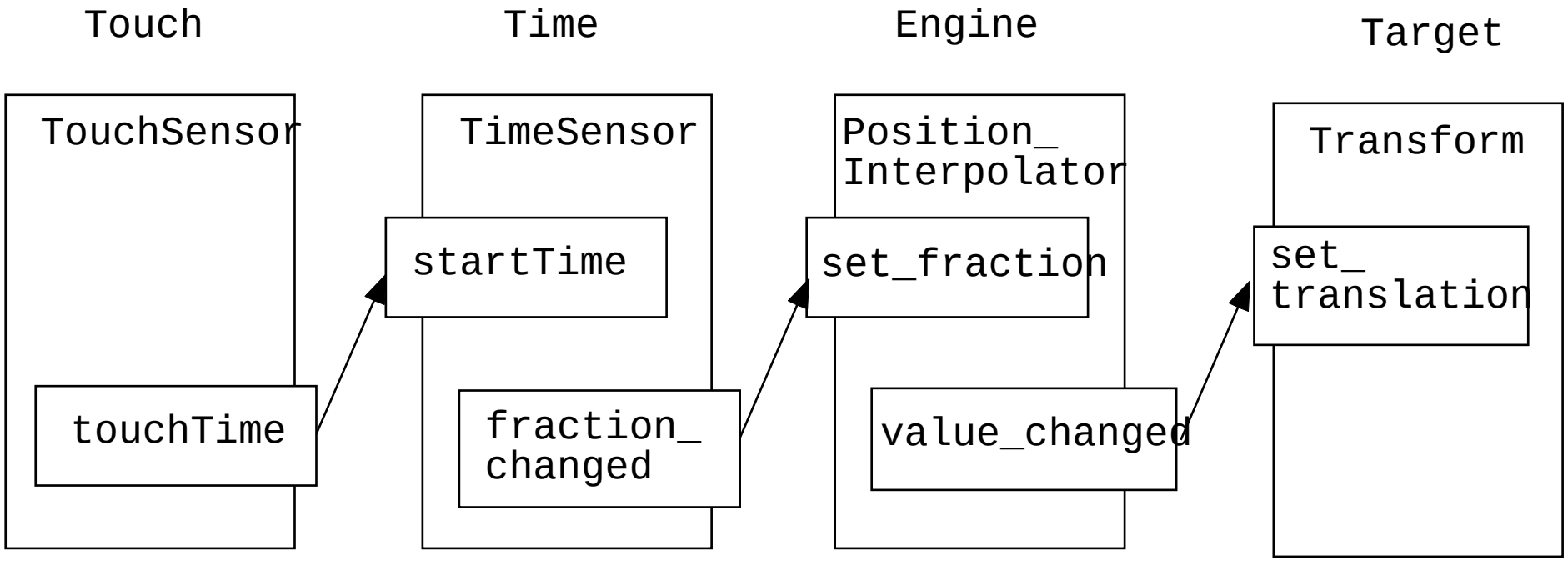
VRML : exemple de scène dynamique


```
#VRML V2.0 utf8

Group {
  children [
     DEF LIGHT_CONTROL TouchSensor {}
     DEF SP Transform {
       children [
         Shape {
           appearance Appearance   {
             material Material {diffuseColor 1 0 0}
           }
           geometry Sphere {
             radius 1
           }

         }
       ]
     }
  ]
}
Group {
  children [
     DEF LIGHT PointLight {
       location -6 4 2
       on       FALSE
     }
  ]
}
ROUTE LIGHT_CONTROL.isActive TO LIGHT.on
```

VRML : propagation des événements



Pour que l'utilisateur crée ses propres événements, VRML met à sa disposition le nœud `Script`.

Comme la plupart des nœuds, le nœud `Script` peut recevoir et émettre des événements
Un événement reçu par un nœud `Script` déclenche l'exécution d'une fonction écrite par l'utilisateur.

Un script peut être prototypé, donc exporté dans une scène et instancié.

Le principe est simple : lorsqu'un nœud de type `Script` reçoit un événement, il passe la valeur de l'événement au script JavaScript spécifié par l'attribut `url` de ce nœud. Le script peut alors mettre à jour la valeur de l'événement sortant du nœud `Script`.

La fonction définie dans le script JavaScript reçoit en paramètre la valeur de l'événement lorsque celui-ci est émis. Notons que le nom de la fonction doit être le même que le nom de l'événement entrant.

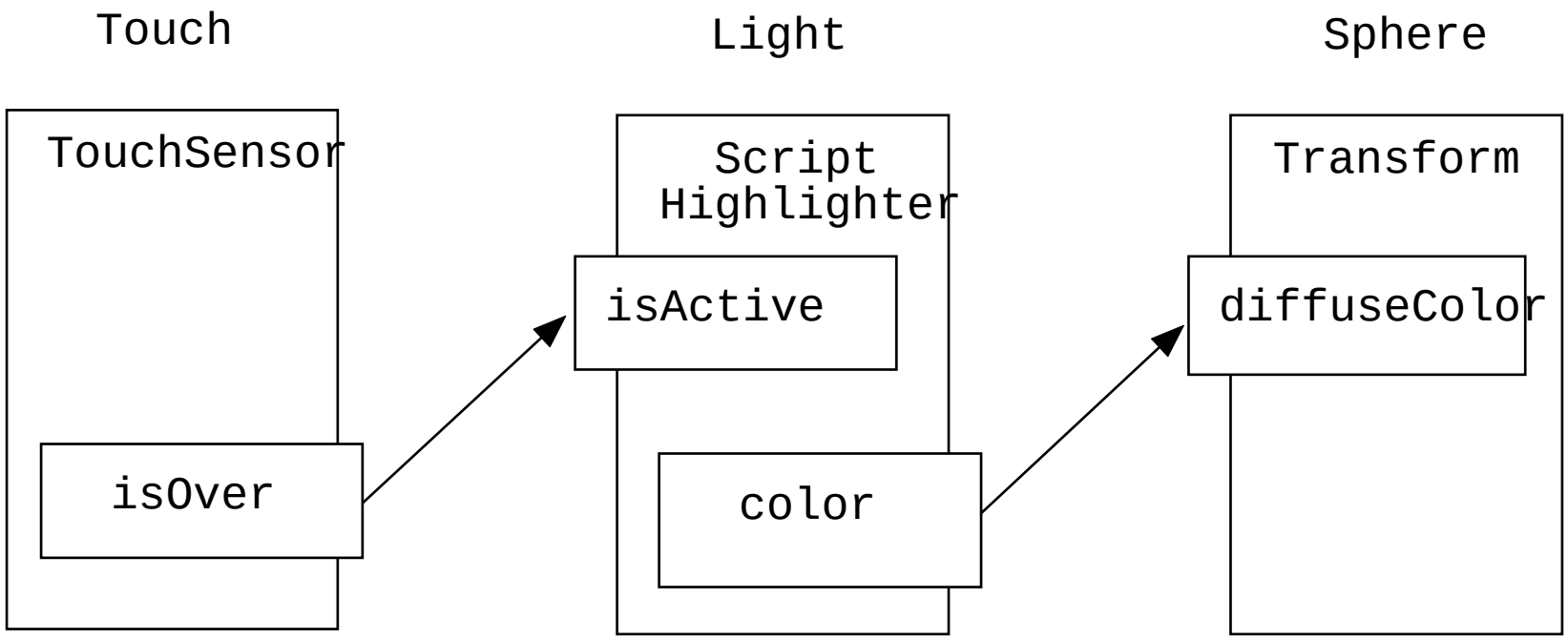
```
function touchTime (timeValue) {
    startTime = timeValue + 5;
}
```

VRML : les scripts - exemple


```
#VRML V2.0 utf8

PROTO Highlighter [
   eventIn SFBool isActive
   eventOut SFColor color
   field SFColor activeColor 0.8 0.8 0.8
   field SFColor inactiveColor 0.5 0.5 0.5
]
{
  Script {
    eventIn  SFBool  isActive IS isActive
    eventOut SFColor color IS color
    field SFColor activeColor IS activeColor
    field SFColor inactiveColor IS inactiveColor
    url [
      "javascript :
         function isActive (eventValue) {
           if (eventValue == true)
             color = activeColor;
           else
             color = inactiveColor;
         } "
    ]
  }
}
ROUTE Touch.isOver TO Light.IsActive
ROUTE  Light.color TO Sphere.diffuseColor
```

VRML : les scripts - exemple



VRML : conclusion

Points forts :

- norme
- co-navigateurs gratuits
- syntaxe simple
- animations de scènes
- grandes possibilités graphiques

Points faibles :

- récent
- quelques limitations
- pas d’outils pour communiquer avec l’extérieur

EAI : Introduction

Pour communiquer entre le monde VRML et le monde extérieur, une interface entre les deux est nécessaire

Cette interface est appelée L'External Authoring Interface (EAI)
Il s'agit d'un ensemble de fonctions, pouvant être exécutées par un environnement extérieur à VRML, qui permettent d'agir sur les objets d'une scène VRML et inversement

Ces fonctions sont regroupées dans un package du langage Java, qui doit être inclus dans le fichier utilisant ces fonctions. Ce package est installé avec le co-navigateur

L'EAI utilise le système d'événements de VRML

EAI : mécanisme général

Pour qu'une applet Java communique avec le monde VRML, elle doit d'abord récupérer ce "monde", soit une instance de la classe `Browser` du co-navigateur
La récupération de cette instance est réalisée par la méthode **`getBrowser()`**

Une fois cette variable récupérée, il est possible d'atteindre n'importe quel objet (nœud VRML) de la scène, sous réserve que cet objet ait un nom (clause `DEF`) et **soit pas défini à l'extérieur de la scène**
la méthode **`getNode("nom du nœud")`** à la variable retournée par **`getBrowser()`**
Les événements d'un nœud récupéré par `getNode` sont accessibles par les méthodes **`getEventIn("nom de l'événement")`** et **`getEventOut("nom de l'événement")`**

applet Java, décrivant la modification de l'attribut `translation` d'un objet de type `Transform` de nom "`Mover`".

```
browser=vrml.external.Browser.getBrowser(this);

Node mover = browser.getNode("Mover");
EventInSFVec3f trans = mover.getEventIn("set_translation") ;

Float value[3] = new float[3] ;
Value[0]=5 ; value[1]=0 ; value[2]=1 ;
Translation.setValue(value) ;
```

EAI : appel des applets Java

Il y a deux façons d'appeler les applets Java dans VRML.

intégrer l'applet pré compilée (`.class`) au sein d'un script. L'applet sera exécutée au lancement du script.

définir un champ particulier dans un fichier HTML, pouvant comporter le fichier principal de la scène VRML et toutes les applets devant agir sur les objets de cette scène.

```
<html>
<head>
<title> Applet Test </title>
</head>
<BODY>
<embed src="YSBON.wrl" height="340" width="440"> </embed>
<applet code="Box.class" width="350" height="320" mayscript> </applet>
</BODY>
</html>
```

EAI : conclusion

Points forts :

- seule manière actuellement de communiquer avec l'extérieur
- utilisation de la puissance de Java

Points faibles :

- baisse des performances du co-navigateur VRML
- limitation importante
- les applets doivent appartenir à la même frame HTML

```
import java.applet.*;
import vrml.*;
import javax.media.j3d.Canvas3D;

public class Vrmlet extends Applet {
   Canvas3D canvas;
   String[]   url;
   Browser  browser;
   int w,h;
   public void init() {
    url = new String[1];
    url[0] = getParameter("URL");
    w = Integer.parseInt(getParameter("WIDTH"));
    h = Integer.parseInt(getParameter("HEIGHT"));
    canvas = new Canvas3D(null);
    browser = new Browser(canvas);
    browser.setAWTContainer(this);
    add("Center",canvas);
    canvas.resize(w,h);
    canvas.show();
    showStatus("Loading "+url[0]);
     try {
        browser.loadURL(url, null);
     }
     catch (Exception e) {
       e.printStackTrace();
     }
    showStatus("Showing "+url[0]);
   }
}
```

```
#VRML V2.0 utf8

Viewpoint {
    position 0 0 8
    orientation 0 1 0 0
}
Group{
 children [
    DEF T1 Transform {
       children [
          DEF T1S SphereSensor{
             spinKick .5
             autoSpinFrameWait 0
             autoSpin TRUE
          }
          Shape {
             appearance Appearance {
                material Material {
                   diffuseColor 0.0 0.0 1.0
                }
                texture ImageTexture {url "crosshair.jpg"}
             }
             geometry Box {}
          }
          Transform {
             translation 3 0 0
             children [
                Shape {
                   appearance Appearance {
                      material Material {
                         diffuseColor 1.0 1.0 0.0}
                   }
                   geometry Sphere { radius .1 }
                }
             ]
          }
       ]
    }
 ]
}
```

```
<html>
<head>
<title>
Vrmlet test page.
</title>
</head>
<APPLET code="Vrmlet.class" WIDTH="512" HEIGHT="512">
<PARAM name="URL" value="db1.wrl">
</APPLET>
</html>
```

VRML : Bibliographie

Le site français
http://apia.u-strasbg.fr/vrml
http://apia.u-strasbg.fr/vrml/tutoriel/
Liste de diffusion française vrml@u-strasbg.fr
http://apia.u-strasbg.fr/vrml/liste/

Les sites incontournables
Le consortium VRML : http://www.vrml.org
VRML Developper's Journal: http://www.VRMLDevelopersJournal.com
Le site CosmoSoftware : http://www.cosmosoftware.com
Le site de silicon Graphics http://www.sgi.com
Excellent tutoriel (en anglais)http://sim.di.uminho.pt/vrml/
Tout sur VRML (site très complet) : http://vrml.sdsc.edu http://vrml.miningco.com

VRML et Java - External Authoring Interface
http://www.cosmosoftware.com/developer/index_allabout.html
http://www.symbolstone.org/technology/java/jvrml/index.html
http://vrml.miningo.com/library/js/bljs.htm

VRML et Java3D
http://www.javasoft.com/products/javamedia/3D/forDevelopers/interest_group.html
http://www.web3d.org/WorkingGroups/vrml-java3d/

Deux livres :
The VRML 2.0 Handbook - Building Moving Worlds on the Web.
Jed Hartman & Josie Wernecke - Silicon graphics, Inc. Addison Wesley Developers Pres

Teach yourself VRML 2 in 21 days.
Chris Marrin & Bruce Campbell - Sams Net editions.