Java VRML (EAI)

Code: java-vrml

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/java-vrml/java-vrml.html**

url: **http://tecfa.unige.ch/guides/tie/pdf/files/java-vrml.pdf**

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 0.2 (modifié le 18/5/00) - BROUILLON donc .....

Prérequis

Module technique précédent: connaissances de VRML (pas de module TIE)

Module technique précédent: java-awt

Abstract

- Introduction à l'EAI (External Authoring Interface) de VRML

Objectifs

- EAI de base ....
- Note: voir aussi le Tecfa VRML Tutorial (section EAI) : http://tecfa.unige.ch/guides/vrml/vrmlman/

A faire

- Java Scripting node (peut-être, car JavaScript fait l'affaire sauf pour le networking)
- un exemple sérieux (genre interface de construction d'un monde liée à une base de données)
- un exemple qui customise un client multi-utilisateur (Deep Matrix ou VNet)

1. Table des matières détaillée

2. Les interfaces de programmation dans VRML

2.1 Scripting nodes vs. External authoring interface

"Scripting Nodes" ISO

- définition de noeds "logiques" à l'intérieur d'une scène VRML
- bonnes implémentations en EcmaScript (avant: VrmlScript et JavaScript)
- mauvaises implémentations en Java (en règle générale), on conseille donc d'utiliser EcmaScript à la place (sauf s'il faut une connectivité réseau)
- les Java Scripting Nodes ne sont pas présentés dans ce module .....
- Usage typique: scènes VRML interactives et animations

EAI

- EAI = External Authoring Interface
- permet d'interagir avec une scène VRML depuis un applet
- possède les même fonctionalités que les scripting nodes (mais parfois plus difficiles à implémenter !) et qq supplémentaires
- fonctionne relativement bien avec les clients majeurs
- Usage typique: clients multi-utilisateurs, interfaces de construction, visualisations dynamiques et interactives

3. Introduction à l'EAI

A. Principe et Standards

- L'EAI est une spécification abstraite qui définit un mécanisme pour envoyer et pour recevoir d'événements depuis un programme externe
- Implémenté avec Java, l'EAI est un jeu de classes que l'on peut appeler depuis un applet pour contrôler un monde VRML
- L'EAI n'est pas un standard, mais une proposition (finalisée en janvier 1999) par le EAI working group

  url: EAI working group: **http://www.web3d.org/WorkingGroups/vrml-eai/**
- La plupart des implémentations (si je ne me trompes pas) suivent le "proposal" par Chris Marrin / SGI qui date de 1977. Il 'sagit d'un "de facto standard".

  url: **http://tecfa.unige.ch/guides/vrml/vrml97/ExternalInterface.html (specification)**

  Il n'existe pas de bon tutorial on- ou off-line à ma connaissance, le seul livre utile est Roehl.B et al. Late Night VRML 2.0 with Java, Ziff Davis Press (mais le code nécessite des révisions). Voir **http://tecfa.unige.ch/guides/vrml/vrmlman/** et voir la FAQ (**http://members.xoom.com/muratak/eaifaq.htm**)

B. Classes Java

- Se trouvent dans le package `vrml.external`
- Important: Les classes sont distribués avec les clients (plugins) VRML. Le développeur doit donc les chercher dans les installations.
  - Cosmo Player 2.1: archive "npcosmop211.jar"
  - Blaxxun: pas d'archive, mais ils sont dans le répertoire "vrml"
  - Je ne sais pas s'il existe des incompatibilités entre les EAI pour différents plugins et clients WWW ... à priori non, mais ce n'est pas sûr.
- A Tecfa: archive avec un extrait des classes Cosmo (sans les scripting nodes)

  url: **http://tecfa.unige.ch/guides/java/classes/vrml-cosmo-eai.jar**
- Note: il faut ces classes juste pour compiler, PAS pour exécuter donc il ne faut pas inclure ces classes dans l'applet. Les plugins vont les trouver sur le PC local !!
- Il ne faut jamais utiliser EAI et scripting nodes (Java) en même temps ! Malheureusement il y conflit entre les classes (à vérifier s'il ne s'agit pas juste d'un conflit de noms)

3.1 Anatomie d'une page Web typique

Une page HTML / VRML / EAI typique contient:

1. du texte HTML
2. embedded VRML browser plug-in
3. un applet Java (visible ou non)

Template:

```
<HTML>
  <HEAD>
    <TITLE>Typical HTML EAI/VRML/JAVA file</TITLE>
  </HEAD>
  <BODY>
    <H1>Typical HTML EAI/VRML/JAVA file</H1>
  <!-- VRML plugged-in scene -->
    <embed src="vrml-scene.wrl" border=0 height="250" width="375">
  <!-- EAI Java applet -->
    <applet code="MyEAIapplet.class" mayscript height="200" width=500">
    </applet>
    </BODY>
  </HTML>
```

url: **http://tecfa.unige.ch/vrml/templates/eai/eai-file.html.text**

Attention: Il faut absolument inclure le paramètre "mayscript" dans le tag applet !!

3.2 Le template de base d'un applet EAI

url: **http://tecfa.unige.ch/guides/vrml/examples/eai/getbrowser/get-browser.html**

url: **http://tecfa.unige.ch/guides/vrml/examples/eai/getbrowser/MyEAIapplet.java**

```
import java.awt.*;
import java.applet.*;
import vrml.external.*;

public class MyEAIapplet extends Applet {
  Browser browser = null;
  public void init() {
    // Paint something to the applet so that you can see something :)
    add (new Label ("This is the Java Applet with a "));
    add (new Button ("No nothing button"));
    // get the Browser
    browser = Browser.getBrowser(this);
    // Test if we really got and print a message to the Java Console
   if (browser == null) {
      System.out.println("FATAL ERROR! no browser :( ");
      return;
      }
   System.out.println("Got the browser: "+browser);
   // Now we could do something with what we got
      }
   }
```

- Cet exemple montre comment obtenir une connexion vers un browser VRML. Un fois qu'on a l'objet "browser" (le plugin VRML) on peut communiquer avec.

3.3 Les opérations de base

A. Obtenir un référence vers le plug-in

voir 3.2 "Le template de base d'un applet EAI" [8]

B. Obtenir une référence d'un noeud VRML

Une fois qu'on a un "handle" vers un noeud, on peut l'observer et le manipuler

C. Lire et écrire des valeurs d'un noeud

équivalent au USE / field et eventOut du scripting node

D. Générer du VRML (from String)

Créer du VRML from string est vital pour créer de nouveaux objets

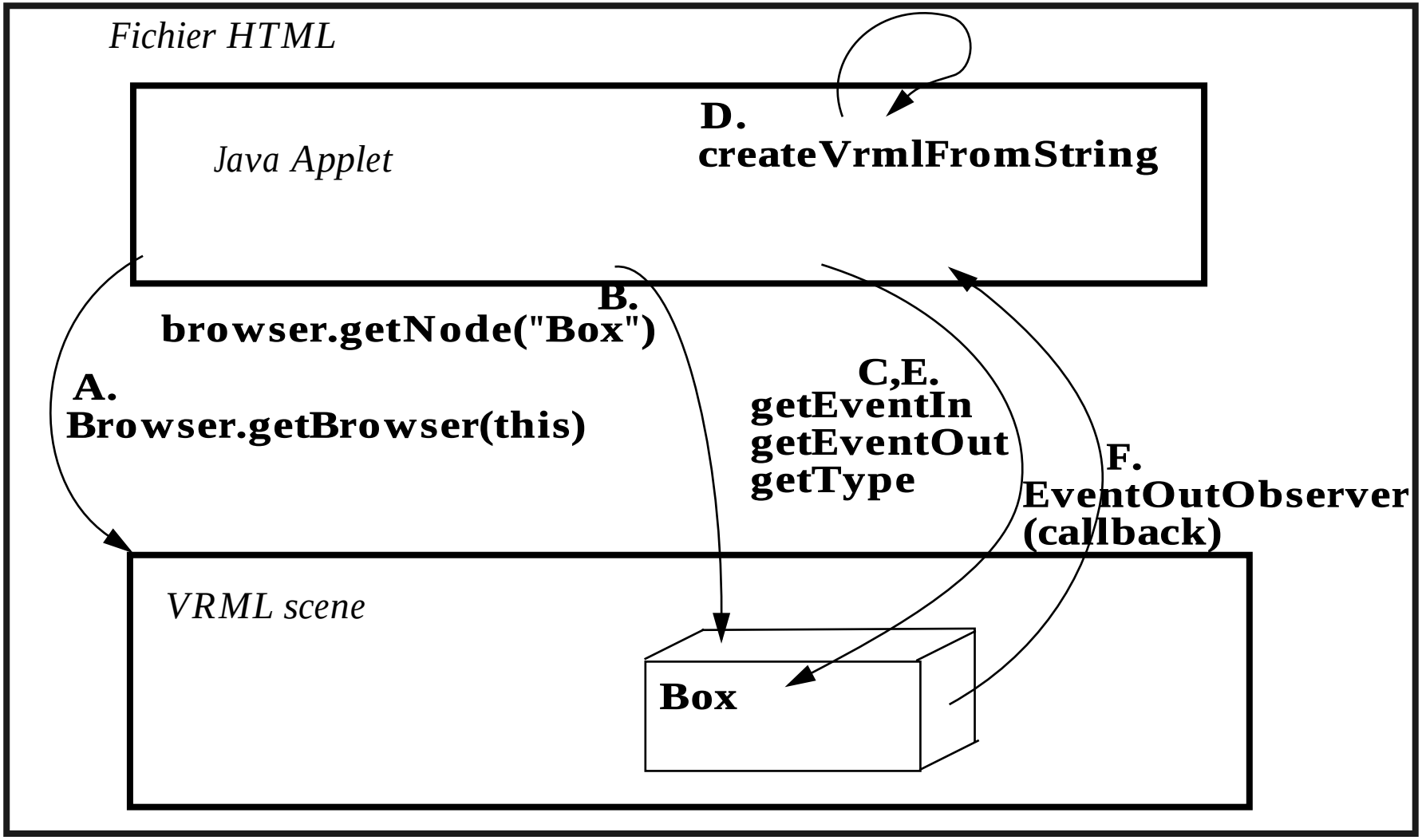
E. Ajouter / enlever des objets et remplacer la scène

ajouter / enlever = même chose que C (on manipule le slot children)

F. Recevoir des événéments de la scène

remplace eventIn du scripting node et les "routes"

Résumé des éléments les plus importantes de l'EAI



4. Techniques EAI de base

4.1 Référence vers un noeud VRML et changer la valeur d'un champs

Exemple 4-1: Changer la couleur d'une boule

url: **http://tecfa.unige.ch/guides/vrml/examples/eai/rgb-change/sphere.wrl.text**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/rgb-change/RGBChangeTest.java**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/rgb-change/rgb-change-test.html**

A. Obtenir la référence

- On fait ce travail typiquement dans la méthode init de l'applet
- Seul on noeud qui est DEFed dans la scène VRML peut être accédé via la méthode getNode de la classe Browser.
- Un noeud VRML est stocké dans un objet de type "Node"

```
Node material = null;
material = browser.getNode("MAT");
```

- Dans la suite on peut utilise les méthodes de la classe Node pour "lire" (eventOut) et "écrire" (eventIn) sur les noeuds.

B. Obtenir une référence vers un EventIn

- Envoyer un eventIn consiste à écrire / transformer un champs d'un noeud VRML
- On peut obtenir une "référence" sur les eventIn de tous les champs de type exposedField et eventIn
  - Rappel: un eventIn vers un exposedField s'appelle "set_xxxx" !!
- Par exemple la définition du noeud "Material" indique qu'on peut modifier tous ses champs (fields) car il s'agit de exposedField(s) :

```
Material {
  exposedField SFFloat ambientIntensity  0.2         # [0,1]
  exposedField SFColor diffuseColor      0.8 0.8 0.8 # [0,1]
  [ .... ]
  exposedField SFFloat transparency      0           # [0,1]
}
```

- Il existe des classes Java / EAI pour chaque type de event VRML, mais il faut faire un type-cast.

```
EventInSFColor diffuseColor = null;
diffuseColor = (EventInSFColor) material.getEventIn("set_diffuseColor");
```

  - **`diffuseColor`** est une référence vers le eventIn "set_diffuseColor" qu'on utilisera dans la suite
  - Les noms de ces classes ressemblent aux noms de types de champs

url: Voir: **Interface Hierarchy de l'EAI Spec**

C. Envoyer un évent (écrire dans un champs d'un noeud)

- la plupart des champs VRML ont plusieurs valeurs (souvent entre 3 et 4).
- donc: on crée et remplit un array qu'on "envoye" dans la suite avec la méthode **`setValue`** de la référence

```
float[] color = new float[3];
......
diffuseColor = (EventInSFColor) material.getEventIn("set_diffuseColor");
......
color[0] = 0.2f; color[1] = 0.2f; color[2] = 0.8f;
diffuseColor.setValue(color);
```

- Il existe 2 situations "classiques" qui font que l'applet veut écrire:
  - l'utilisateur déclenche un événement dans l'applet (appuye sur un bouton par exemple

```
add(redButton = new Button("Red"));
redButton.addActionListener(this);
......
public void actionPerformed (ActionEvent event) {
   Object clickedButton = event.getSource();
   if (clickedButton == redButton)  {
       color[0] = 0.8f; color[1] = 0.2f; color[2] = 0.2f;
       diffuseColor.setValue(color);
      }
```

  - la scène VRML envoye un événement qui sera intercepté et traité par l'applet (voir 4.4 "Recevoir des événements de la scène VRML" [18])

4.2 Lire la valeur d'un champs VRML

Exemple 4-2: Lire la position d'un viewpoint

url: **http://tecfa.unige.ch/guides/vrml/examples/eai/get-node-info/scene.wrl.text**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/get-node-info/GetNodeInfoS.java**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/get-node-info/get-node-info.html**

- même logique que pour écrire une valeur
- L'EAI peut accéder eventOut (y compris ceux des exposedField)
- Notez qu'on accède à l'eventOut d'un field "exposed" en ajoutant "_changed" à son nom (même principe que pour le "set_xxx") !

```
Node entryVP = null;
EventOutSFVec3f positionVP = null;
.....
entryVP = browser.getNode("Entry");
positionVP =(EventOutSFVec3f) entryVP.getEventOut("position_changed");
```

- De nouveau on représente les valeurs d'un champs VRML par un array

```
float[] currentPosition;
currentPosition = positionVP.getValue();
// et on fait quelque chose (ici écrire dans un textarea)
output.append("Entry ViewPoint is at" +
              " x=" + currentPosition [0] +
              " y=" + currentPosition [1] +
              " z=" + currentPosition [2] + "\n");
```

4.3 Create VRML from String et remplacer ou modifier le monde

Exemple 4-3: Créer un "scene graph" et remplacer une scène VRML

url: **http://tecfa.unige.ch/guides/vrml/examples/eai/create/null.wrl.text**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/create/CreateEx0.java**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/create/create-ex0.html**

- Un scène VRML est représenté dans l'applet Java comme un array de Nodes

```
Node[] scene = null;
```

- La méthode createVrmlFromString

```
scene = browser.createVrmlFromString(
          "DEF Camera Viewpoint {\n" +
               "  position 0 0 5}\n" +
          "DEF MySphere Transform {\n" +
               "   children [ \n" +
           ......
               "     ] \n" +
               " } \n"
               );
```

- Remplacer le monde

```
browser.replaceWorld(scene);
```

A. Modifier le monde

Exemple 4-4: Ajouter / enlever un objet dans un grouping node

url: **http://tecfa.unige.ch/guides/vrml/examples/eai/add-remove-test/root.wrl.text**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/add-remove-test/AddRemoveTest.java**
url: **/guides/vrml/examples/eai/add-remove-test/AddRemoveTest.html**

- exactement la même logique que pour changer la valeur d'un champs
  (voir : 4.1, p. 11)
- On transforme la valeur d'un grouping node, comme Group our Transform
  - Voici la définition de Group:

```
Group {
  eventIn      MFNode  addChildren
  eventIn      MFNode  removeChildren
  exposedField MFNode  children      []
  field        SFVec3f bboxCenter    0 0 0     # (-,)
  field        SFVec3f bboxSize      -1 -1 -1  # (0,) or -1,-1,-1
}
```

- Obentir un handle sur les eventIn addChildren et removeChildren

```
Node root = browser.getNode("ROOT");
// EventIns of the root node
EventInMFNode addChildren;
EventInMFNode removeChildren;
// Instantiate (get handle to) the EventIn objects
addChildren = (EventInMFNode) root.getEventIn("addChildren");
```

```
removeChildren = (EventInMFNode) root.getEventIn("removeChildren");
```

- Faire (utiliser la méthode setValue)

```
if (b == addButton) {
    addChildren.setValue(shape);
   }
else if (b == removeButton) {
   removeChildren.setValue(shape);
  }
```

  - shape est un vecteur de type Node (parceque les eventIn sont de type MFNode)

```
// Shape group hierarchy
Node[] shape;
shape = browser.createVrmlFromString("Shape {\n" +
                "  appearance Appearance {\n" +
                "    material Material {\n" +
                "      diffuseColor 0.2 0.2 0.8\n" +
                "    }\n" +
                "  }\n" +
                "  geometry Sphere {}\n" +
                "}\n");
```

4.4 Recevoir des événements de la scène VRML

Exemple 4-5: Un boule qui change de couleur quand on clicke

url: **http://tecfa.unige.ch/guides/vrml/examples/eai/monitor-test/MonitorTouch.java**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/monitor-test/monitor-touch.html**
url: **http://tecfa.unige.ch/guides/vrml/examples/eai/monitor-test/root.wrl**

- Il faut implémenter l'interface `EventOutObserver` et implémenter une méthode `callback`
- Note: selon la proposition de la specification sousmise pour standardisation, cette fonctionalité est implémenté très différemment, elle est remplacé par une interface vers les Routes ....

A. Handle pour un eventOut et enregistrement

```
Node sensor = browser.getNode("TOUCH");
// Get its isActive EventOut
isActive = (EventOutSFBool) sensor.getEventOut("isActive");
// Set up the callback
isActive.advise(this, isActive);
```

B. Le callback

```
public void callback(EventOut who, double when, Object which) {
  // retrieve the state of isActive (see above the advise function)
  EventOutSFBool state = (EventOutSFBool) which;
  // only deal with the event if isActive was true, else the ball would flip back
 if (state.getValue() == true)  {
     System.out.println("callback(): EventOut=" + who
                        + " state =" + state.getValue());
    // Change the color and remember it
    if (colorState == 1) {
       diffuseColor.setValue(redColor);
       colorState = 0;
       }
    else {
       diffuseColor.setValue(greenColor);
       colorState = 1;
  }   }  }
```

- Note: on a un seul callback possible ici, à voir comment gérér plusieurs (les identifier)

4.5 EAI avec SQL

url: **http://tecfa2.unige.ch/guides/vrml/examples/eai/sql-intro/SQLIntro.java**
url: **http://tecfa2.unige.ch/guides/vrml/examples/eai/sql-intro/sql-intro.html**

- à faire !!
- l'applet marche mais ne fait rien d'intéressant. Il devrait générer du VRML à partir de la requête SQL (au lieu de plâtrer les résultats dans une text area).
- Warning: Il faut appeler ca sur tecfa2 (pas tecfa!!)