Les limites de VRML pour les comportements interactifs : étude de cas

Pierre Cubaud, Alexandre Topol, Dan Vodislav

Centre d'Etudes et de Recherche en Informatique
Conservatoire National des Arts et Métiers
{cubaud, topol, vodislav}@cnam.fr

RESUME

Aujourd'hui, de nombreux outils existent pour la création et la visualisation de scènes 3D sur Internet. La plupart des modeleurs exportent leurs données en VRML pour être mises en ligne et visualisées par un *plugin*. La diffusion d'une scène, même de taille conséquente, ne pose plus véritablement de problèmes. Cependant, les options de navigation sont toujours prises en charge par le *plugin* VRML plutôt que spécifiées dans le fichier. Les interactions avec les différents objets composant une scène reposent quant à elles sur un système de routage d'événements qui nous semble un mécanisme peu naturel. Enfin, pour créer des interactions " évoluées " l'auteur d'une scène doit connaître un langage de script ; ce qui a pour conséquence, soit de réduire le nombre de concepteurs potentiels, soit de limiter les fonctionnalités des scènes. En nous appuyant sur un exemple d'interface 3D pour une bibliothèque numérique, nous présentons les mécanismes d'interactions de VRML que nous considérons comme limitants et nous proposons quelques éléments de réponse.

MOTS CLES : VRML, 3D, interaction, comportements.

INTRODUCTION

Depuis que les cartes graphiques avec fonctions 3D câblées sont devenues un matériel standard pour les micro-ordinateurs, nous avons assisté à un déplacement progressif des problèmes rencontrés pour la mise à disposition d'une scène 3D. A l'heure actuelle, le débit des nouvelles lignes de communication et la puissance des cartes 3D autorisent à mettre en ligne des scènes plus complexes, plus fournies en informations, en restant sûr que l'utilisateur pourra les visualiser. La principale préoccupation des auteurs de scènes n'est alors plus de réduire le temps de chargement ou d'augmenter la fluidité de la navigation en limitant le contenu d'une scène, mais de fournir aux utilisateurs des outils permettant une navigation optimale.

Le langage VRML (*Virtual Reality Modeling Language*) [3] s'est imposé, en raison de son monopole à ses débuts, comme un standard pour la description et la diffusion de contenus 3D. C'est même un des rares exemples de standard " Internet " devenu un standard international (ISO/IEC 14772). Même s'il n'était conçu à l'origine que pour gérer des environnements immersifs multi-utilisateurs sur Internet, son statut de standard nous a poussé à l'utiliser dans le cadre de nos expérimentations sur les bibliothèques numériques [4]. Dans sa deuxième version (VRML97), la structuration des différents objets en graphe de scène répond à un besoin de simplification et de clarification de la description. Cette nouvelle organisation convient à un plus grand nombre d'auteurs et plus particulièrement aux non programmeurs. Cependant, la spécification des animations – qu'elles soient automatiques ou dépendantes des interactions – requiert la connaissance d'un langage de programmation dès lors qu'elles sont complexes. Et, même pour un programmeur chevronné, l'écriture de scripts pour la gestion des comportements est une étape laborieuse. On constate par exemple que, selon les *plugins*, il existe des différences d'implémentation de l'interface entre le moteur d'animations et le langage de script pouvant mener à des dysfonctionnements graves. Dans ce contexte, l'écriture de comportements interactifs n'est pas une tâche aisée.

Nous proposons donc dans cet article de montrer les limites du système d'animations de la norme VRML97 à base d'événements et de scripts. Nous cherchons aussi à donner une direction vers laquelle le langage pourrait évoluer pour intégrer la spécification des animations et des interactions basiques sur les objets graphiques décrits. Pour illustrer notre étude, nous présenterons d'abord nos expérimentations sur les bibliothèques numériques en 3D. Nous indiquerons ensuite les problèmes de l'architecture mise en place lors de ces tests. Enfin nous donnerons quelques solutions envisagées pour améliorer la gestion des interactions.

CONTEXTE DE L'ETUDE

Le CNAM a engagé en 1998 la numérisation d'une sélection de son fonds ancien d'ouvrages et de revues scientifiques et techniques. Depuis fev. 2000, les textes numérisés sont librement consultables sur le serveur WWW " Conservatoire Numérique des Arts et Métiers " (CNUM) [5]. Les recherches et la consultation des livres se font grâce à une interface construite par des scripts CGI générant des pages HTML [6].

Nous voulions profiter du haut débit de notre réseau

local (et des accès câble ou ADSL aujourd'hui) pour offrir une interface plus sophistiquée. Interpellés par l'extrait suivant de A. Mengel nous avons décidé de présenter les reliures des livres au lecteur : " Les livres s'affirment grâce à leurs titres, leurs auteurs, leurs places dans un catalogue ou une bibliothèque, leurs illustrations de couvertures. […] Je juge un livre à sa couverture ; je juge un livre à sa forme. " [8]. Nous pensons que ces informations sont des données importantes pour choisir un ouvrage, particulièrement dans un fonds ancien. Notre choix s'est rapidement porté sur une interface 3D décrite en VRML et construite à la volée par des scripts CGI [4]. Notre idée était de nous inspirer de la roue à livres de Ramelli (figure 1a). Elle autorise une lecture croisée de plusieurs ouvrages ouverts à la dernière page consultée. Le lecteur la tourne à l'aide d'un pédalier ; les mains restant disponibles pour feuilleter. Dans notre interface de sélection (figure 1b) les dos des livres sont placés dans un cylindre en fonction de leur pertinence par rapport à deux critères ; nombre jugé suffisant pour la recherche dans un corpus. Les livres pertinents sont initialement face à l'utilisateur. Par un déplacement vertical, il peut visualiser ceux ne satisfaisant qu'un seul critère. Afin de présenter tout le fonds, les autres ouvrages occupent la surface restante du cylindre. Cette expérimentation nous a montré les possibilités offertes par une interface décrite à l'aide du langage VRML, mais les nœuds *Script* n'y étaient pas utilisés.

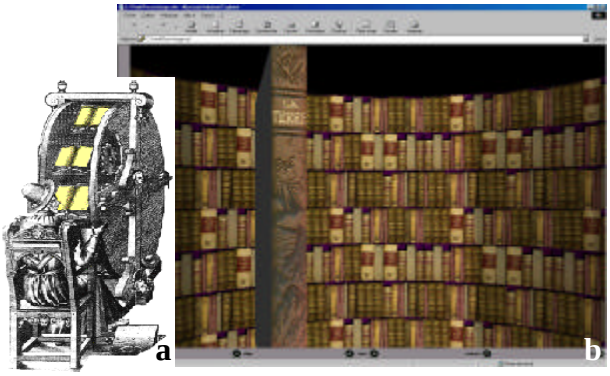

Figure 1 : Roue à livre (a) Interface de sélection (b).

La numérisation des ouvrages du CNUM nous a permis ensuite d'étudier une interface pour la lecture des livres. Parmi les différentes alternatives étudiées, une interface 3D semblait là encore convenir. Elle reste consistante avec l'interface de localisation et permet au lecteur d'organiser son espace de lecture grâce aux zooms et aux vues en perspective de la même façon que l'interface du Web Forager permet d'agencer les WebBooks [2]. C'est sur cette expérimentation précisément que nous portons notre attention puisqu'elle utilise les scripts Java pour modifier ou créer du code VRML de manière à gérer les interactions.

EXPERIMENTATION

Pour notre essai d'interface 3D de consultation en ligne, il nous a paru important de permettre aux utilisateur d'organiser eux-mêmes l'emplacement et l'orientation des ouvrages consultés. Pour cela, nous avons choisi d'utiliser des fenêtres 3D pouvant contenir n'importe quel objet décrit en syntaxe VRML. C'est dans une de ces fenêtres que sont présentés tous les livres à l'utilisateur. Pour lier cette interface de consultation à l'interface de recherche il suffirait de présenter les groupes de livres dans autant de fenêtres 3D. Le choix d'utiliser un élément présent dans la GUI 2D n'est pas un hasard. Nous pouvons ainsi profiter de la connaissance que possède déjà l'utilisateur pour manipuler les fenêtres.

Lorsque la scène VRML est chargée, elle n'est constituée que d'une seule fenêtre contenant les reliures des ouvrages de la bibliothèque CNUM. Le passage de la souris sur un livre fait apparaître une étiquette indiquant l'auteur et le titre du livre. Un clic de souris sur un livre crée une nouvelle fenêtre au premier plan contenant la reliure du livre pour l'identifier rapidement, sa notice et un bouton par tome du livre pour une consultation directe. Un clic sur un de ses boutons change le contenu de cette fenêtre. La notice et les boutons vers les différents tomes du livre disparaissent. Ils laissent place à l'image de la première page du tome sélectionné et aux boutons de navigation (page suivante et précédente, première et dernière page, et retour à la notice). Le lecteur peut ouvrir autant de livres qu'il souhaite et plusieurs fois le même si cela lui est utile. La figure 2 montre des copies d'écran de cette interface de lecture.

Les fenêtres 3D subissent les transformations du point de vue sur la scène. Cependant, l'utilisateur peut interdire à ces transformations de prendre effet sur une fenêtre en la punaisant. De cette manière, elle paraît être immobile. Les différentes interactions sur une fenêtre sont simples :

- la punaiser et la dé-punaiser,
- la déplacer selon ses axes X et Y par sa barre de titre,
- la fermer.

Implémentation

Puisque nous désirons utiliser des fenêtres que le lecteur puisse ouvrir et fermer à sa guise, il faut un mécanisme permettant de créer et d'enlever du code VRML à la scène intiale. La seule possibilité consiste à utiliser les méthodes *CreateVrmlFromString* de la classe *Script* et *destroy* de la classe abstraite *Vector* servant à énumérer toutes les fenêtres. Le code du fichier VRML principal ne contient aucune définition d'objets mais uniquement les nœuds suivants :

- *Group* contenant toutes les fenêtres,
- *ProximitySensor* servant à inhiber les transformations de la caméra pour les fenêtres punaisées,
- *Script* définissant la classe Java principale *MainScript*, gestionnaire de la scène.

La méthode *initialize*, lancée à l'initialisation de ce script (donc à celle de la scène) construit le code VRML de la

première fenêtre en fonction des livres présents dans la bibliothèque CNUM. Un prototype de livre est utilisé. Il définit son apparence et déclare le script *BookScript* qui gère les interactions sur ce livre. Lorsqu'un clic de souris a lieu sur le dos d'un livre, un événement est propagé

Figure 2 : Interface de consultation.

vers ce script qui demande la création d'une fenêtre supplémentaire au gestionnaire de fenêtre. Le code VRML des objets constituant les informations de ce livre est inséré dans la nouvelle fenêtre. Les boutons de consultation d'un livre sont gérés de la même manière : ils émettent un événement qui change le mode de la fenêtre (passage du mode notice au mode lecture) et qui crée l'interface de lecture gérée par la classe *PageScript*.

Les fenêtres 3D sont prises en charge par la classe *Window3D* et leur code VRML est généré dans la méthode *create*. Ce code contient un nœud *Script* déclarant l'utilisation de la classe *Window3Dscript* qui sert à récupérer les interactions sur les fenêtres. Il est chargé également de calculer la transformation inverse et de l'appliquer à la fenêtre si elle doit rester immobile. Les transformations du point de vue sont récupérées grâce au nœud *ProximitySensor* dont la boîte définissant ses frontières de fonctionnement est arbitrairement grande afin qu'il émette toujours les changements. Les déplacements d'une fenêtre se font par " glisser-déposer " en cliquant sur sa barre de titre. Ils sont réalisés par un routage de l'événement sortant *translation_changed* du nœud *PlaneSensor* (qui renvoie les déplacements de la souris dans le plan de la barre de titre) vers l'attribut *set_translation* du nœud *Transform* contenant toute la fenêtre.

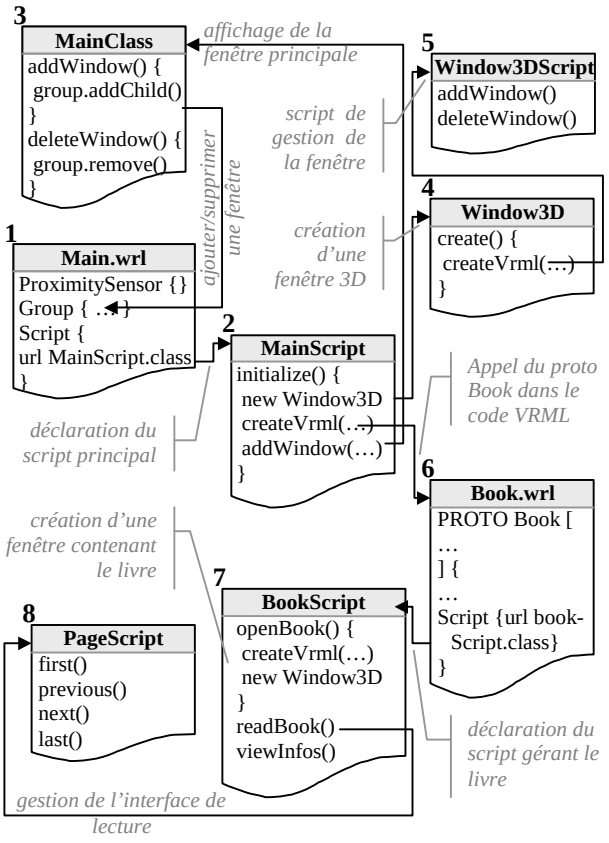

Figure 3 : Architecture logicielle du système de consultation.

L'architecture logicielle décrite est schématisée dans la figure 3. Bien que les comportements soient simples, ils représentent plus de 1200 lignes de code en Java pour 160 lignes de VRML, soit la quasi-totalité de la scène téléchargée.

PROPOSITION D'EXTENSIONS DE VRML

L'effort de programmation avec le modèle " bas niveau " ci-dessus n'est pas à la mesure des résultats obtenus. En aucun cas, un concepteur non programmeur ne pourra mettre à profit le système de scripts pour enrichir les interactions dans une scène VRML. Or ce sont probablement les graphistes ou les artistes (donc des non programmeurs) qui sont en mesure de créer les scènes VRML les plus intéressantes. Nous pensons donc que l'utilisation des événements et des scripts est trop limitée, non pas pour ce qu'ils permettent de faire, mais plutôt pour le nombre de concepteurs potentiels à qui ils s'adressent. De plus, pour les scènes " évoluées ", les scripts représentent la plus grande partie du code diffusé. De ce fait, l'apport de VRML en tant que langage support est faible. Enfin, les scripts gérant de telles scènes sont difficilement distribués par leurs auteurs. Ils

constituent en effet toute l'intelligence de la scène qu'un auteur ne veut pas toujours rendre visible pour de multiples raisons : ils sont leur savoir-faire, parce que leur code n'est pas propre ou mal commenté, etc. A ce jour, il est bien difficile de trouver sur Internet des scènes VRML scriptées utilisées à des fins non anecdotiques, ou de démonstration.

Le succès à large échelle de VRML repose sur une description des animations et des comportements aussi compréhensible que celle des objets. Cette simplification implique de réduire la place réservée aux scripts gérant les interactions. Nous introduisons pour cela des nœuds de comportements applicables à tout objet. Nous optons également pour le remplacement des routages par des mécanismes simples de contrôle des liens entre nœuds. Nous différons sur ce point avec [1] qui analyse fort bien les manques de VRML mais qui souhaite le faire évoluer vers un langage encore plus basé sur la programmation. C'est aussi le cas de l'architecture EMMA mêlant des objets 2D et 3D mais au prix d'une programmation complexe [7]. Avec notre solution, nous ne remettons pas en cause le modèle de description que nous trouvons puissant et proche des API 3D récentes (telles que Java3D, Direct3D, Fahrenheit). Cependant, pour pallier l'inexistence d'un modèle haut niveau pour décrire les comportements nous l'augmentons sans pour autant dénaturer sa structure.

Pour la scène qui nous sert d'exemple, nous aurions souhaité les extensions suivantes :

- un nœud *Clickable* gérant les clics de souris et fournissant les trois apparences classiques : appuyée, relâchée et *roll-over*. Il permettrait de gérer l'affichage des étiquettes informatives et l'interaction menant à l'ouverture d'un livre.
- un nœud *Movable* signifiant que les interactions de l'utilisateur affectent ses nœuds descendants. Ce nœud permettrait de positionner et d'orienter les objets grâce à des interactions et non plus par navigation comme précédemment. En effet, pour déplacer une seule fenêtre dans le poste de lecture en VRML+Java, il faut punaiser toutes les fenêtres sauf celles que l'on souhaite déplacer.
- un mot clé *AFFECTS* pour éliminer la partie routage du fichier VRML. Lorsque le nœud de comportement ne modifie que ses nœuds descendants, ce mot clé n'est pas nécessaire. Il l'est pour intégrer directement dans le graphe de scène les routages d'événements vers un nœud non descendant (comme dans le code suivant).

Pour avoir une idée plus précise du code VRML obtenu, le pseudo-code ci-dessous décrit une fenêtre 3D avec les nœuds précédents. Cette nouvelle syntaxe remplace 450 lignes de Java servant à gérer les fenêtres et le fichier VRML est réduit à ?125 lignes.

```
DEF Window3D Transform {
    children [              # Les nœuds constituant la fenêtre
        Shape { […] }
        […]
        Shape { […] }
        DEF TitleBar Movable {
            # Les interactions autorisées
            canTranslate TRUE
            canRotate TRUE
            # Les événements émis par ce nœud
            translation AFFECTS Window3D.translation
            rotation AFFECTS Window3D.rotation
            children […]    # Les nœuds constituant la barre
        }
    ]
}
```

CONCLUSION ET TRAVAUX FUTURS

Les mécanismes d'interactions offerts par VRML ne conviennent pas aux non programmeurs. Les extensions proposées visent à fournir un minimum d'outils pour ces non experts afin qu'ils puissent créer des scènes réactives. Les scripts resteront cependant nécessaires pour effectuer des opérations non spécifiques à la 3D.

Nous avons donc proposé d'intégrer la gestion des animations et des comportements dans la description de la scène VRML par l'adjonction de nouveaux nœuds. Et bien que cette approche soit présentée dans le cadre particulier d'une interface 3D d'accès à une bibliothèque numérisée, elle est cependant généralisable car elle se base sur des comportements génériques. La syntaxe des fichiers n'est que légèrement modifiée et devient consistante : utilisation d'un seul graphe de scène pour décrire les objets graphiques et leurs comportements. La définition des nœuds de comportements standards à intégrer fera l'objet d'une prochaine étude. Pour ajouter les nouveaux nœuds lors de la phase d'implémentation, nous utiliserons notre client VRML97 développé à d'autres fins [9]. Ainsi, ce langage VRML étendu pourra être utilisée pour le prototypage d'applications 3D, dans l'esprit des nombreux outils disponibles pour les environnements " plats ".

BIBLIOGRAPHIE

1. Balaguer, J.-F. Less Is More : The Power of Simplicity. In *proc. of VRML'99*, February 23-26, 1999.
2. Card, S., Robertson W., York W. The WebBook and the Web Forager : An Information Workspace for the World-Wide-Web. In *proc. of ACM CHI'96*. Vancouver, April 13-18, 1996.
3. Carson, Georges S., Puk, Richard F. and Carey, Rikk. Developing the VRML97 International Standard. *IEEE Computer Graphics and Applications*, March/April 1999, pp. 52-58 et `http://www.vrml.org/Specifications`
4. Cubaud, P., Thiria C., Topol A. Experimenting a 3D Interface for the Access to a Digital Library. In *proc. of ACM DL'98*, Pittsburg, July 1998.
5. Cubaud, P., Topol, A. A WWW-based digital library for antiquarian collections. RR CEDRIC 99-09.
6. le Conservatoire Numérique des Arts et Métiers.

http://cnum.cnam.fr.
7. Marrin, C. *Beyond VRML*. http://www.marrin.com/vrml/private/EmmaWhitePaper.html
8. Mengel, A. *Une histoire de la lecture*. Edition Actes Sud, 1998, p.155.
9. Topol, A. Immersion of Xwindow Applications into a 3D Workbench. *In proc. of ACM CHI'2000*, The Hague, Netherlands, April 2000.