

MOCN Virtuelle

Thierry DUVAL – Serge MORVAN – Jacques TISSEAU

Laboratoire d'Informatique Industrielle
Ecole Nationale d'Ingénieurs de Brest
Technopole Brest-Iroise, Site de la Pointe du Diable
CP 15 – 29608 Brest Cédex

Résumé

Après une rapide description du projet **ARéVi**, nous présentons l'une des composantes de ce projet : une MOCN "Virtuelle". Nous souhaitons développer et utiliser des techniques de réalité virtuelle afin de simuler une MOCN (bien réelle), la télésurveiller, et la téléopérer, tout ceci à l'aide d'une station de travail Silicon Graphics. L'accent est mis ici sur l'aspect simulation, qui est en fait l'étape la plus importante, puisque c'est sur elle que reposeront la télésurveillance et la téléopération. Nous décrivons donc les principales difficultés rencontrées lors du passage d'une machine réelle à une machine virtuelle, puis ce qu'il reste à faire afin de passer à la simulation.

1 Introduction

Le projet **ARéVi** (**A**telier de **R**éalité **V**irtuelle), développé au Laboratoire d'Informatique Industrielle de l'ENIB, est de créer un atelier de Génie Logiciel pour la Réalité Virtuelle. Il doit nous permettre de :

- modéliser des scènes complexes en trois dimensions, cette modélisation comprenant le comportement des objets qui forment ces scènes,
- nous déplacer dans un univers virtuel et interagir avec les éléments qui le composent,
- travailler à plusieurs dans un même univers virtuel (promenades guidées dans des mondes virtuels, travail coopératif 3D, ...).

Notre but est ici d'utiliser une MOCN (Machine Outil à Commande Numérique) comme illustration de nos travaux.

Pour ce faire, nous modélisons cette machine d'un point de vue informatique, tant sur le plan de son architecture physique que sur le plan de son comportement. Ensuite, il sera possible d'en effectuer la simulation. Les étapes suivantes seront sa télésurveillance et sa téléopération.

2 Du réel au virtuel

2.1 Modélisation

D'une façon générale, pour passer d'une application réelle à une application informatique virtuelle, il faut modéliser son architecture et son comportement.

Dans le cas d'une application réelle de type informatique, cette modélisation peut se faire en dupliquant tout ou partie de l'existant (en général, on dupliquera les données afin de ne pas perturber l'application réelle, mais on pourra peut-être utiliser les règles de comportement telles quelles).

Par contre, dans le cas d'une application réelle non informatique, qu'elle soit mécanique, électromagnétique ou de tout autre type, il faut tout créer à partir d'une observation de l'existant : les aspects géométriques [4], cinématiques, la visualisation informatique de l'application, de ses stimuli (les entrées qui peuvent provoquer une modification de l'application), et de son comportement (sa réaction en réponse aux stimuli).

Il faut donc permettre à un client physique (un utilisateur) ou logiciel (un programme) de provoquer les stimuli susceptibles de faire réagir l'application virtuelle.

Enfin, il faut aussi assurer une mise à jour de la visualisation de l'application virtuelle lors d'une modification de son état [6].

2.2 Simulation

La simulation de l'application réelle grâce à l'application virtuelle s'effectue de la façon suivante : en réponse à des stimuli, il faut calculer l'évolution de l'application virtuelle et mettre à jour le modèle qui la représente ainsi que sa visualisation.

Le travail informatique consiste ici à créer un "moteur" de réalité virtuelle gérant automatiquement la mise à jour de la visualisation en fonction des stimuli, via des requêtes à un module "expert" en comportement de la machine [2].

2.3 Télésurveillance

Une fois la simulation réalisée, le passage à la télésurveillance (téléobservation passive, c'est-à-dire sans possibilité d'agir sur la machine, réelle ou virtuelle) peut se faire en substituant aux entrées virtuelles (les stimuli) un état des modifications de l'application réelle.

Pour cela, il faut que le moteur de réalité virtuelle soit décomposé en modules, de façon à ce que ces nouvelles entrées puissent s'insérer sans modification de l'architecture du moteur.

2.4 Téléopération

Le passage à la téléopération est réalisé à son tour à partir des travaux de simulation et de télésurveillance.

2.4.1 Téléopération aveugle

Le moteur de réalité virtuelle doit maintenant traduire les stimuli "virtuels" en des stimuli compréhensibles pour l'application réelle, tout en continuant à visualiser le comportement de l'application virtuelle.

Ce premier type de téléopération est limité car il ne permet pas de voir si l'application réelle réagit effectivement comme il est prévu qu'elle le fasse.

2.4.2 Téléopération avec retour d'information

Pour pallier ce manque de retour d'information, il devient nécessaire de lier la téléopération à la télésurveillance : le moteur de réalité virtuelle doit être en mesure d'accepter des stimuli virtuels pour les transmettre à l'application réelle, tout en étant capable de récupérer des informations réelles et de les visualiser de façon virtuelle.

De plus, le moteur de réalité virtuelle, grâce au module de modélisation de l'application réelle, doit être en mesure d'indiquer au téléopérateur si la situation peut devenir critique pour l'application réelle.

De la même façon, il devient possible d'effectuer de la télésurveillance active (téléobservation + quelques primitives de téléopération comme l'arrêt d'urgence) : un observateur peut contrôler à distance une application et intervenir si la tournure des événements de lui convient pas. Le module

expert de modélisation de l'application réelle peut ici encore activer un signal d'alarme s'il prévoit un mauvais fonctionnement (ou un fonctionnement dangereux) de l'application réelle.

3 Modélisation de la MOCN

Pour passer de notre MOCN réelle à une machine virtuelle, il a donc fallu construire une réplique logicielle de cette machine bien réelle. Pour cela, nous avons utilisé un modeleur 3D [7]. Nous ne détaillerons pas davantage cette partie du projet (laborieuse !). Nous avons ensuite transformé cette modélisation brute en un ensemble d'objets Inventor [8] (Inventor est une bibliothèque de classes C++).

A ce stade, nous ne disposons que d'une simple visualisation statique : le problème est que nous pouvons manipuler cette réplique de la MOCN d'une façon qui peut s'avérer catastrophique pour elle. En effet, nous n'avons pas exprimé quelles sont les liaisons (mécaniques) entre les différentes parties, ni quelles sont les réactions à des sollicitations extérieures que l'on est en droit d'attendre.

Il faut donc exprimer un certain nombre de contraintes afin d'obtenir une structure physique puis un comportement correct. Pour cela, nous avons développé à l'aide du générateur d'Atelier de Génie Logiciel GraphTalk [1] un éditeur de graphes qui nous permet actuellement de modifier l'architecture d'un ensemble d'objets Inventor, et ce de manière interactive.

Ensuite, nous devons spécifier le comportement des différentes parties de la MOCN afin de répondre aux actions en provenance d'un utilisateur (réel ou virtuel). Pour le moment, nous ajoutons ce comportement "à la main" dans le code C++.

Enfin, nous avons développé un interpréteur permettant de traduire des programmes destinés à la MOCN en des actions simples compréhensibles pour notre machine virtuelle.

4 Simulation de la MOCN

Dans un premier temps, nous souhaitons effectuer une simulation du comportement de la MOCN en réponse à des commandes en provenance du panneau de contrôle, ou bien en réponse à des commandes complexes programmées.

Cette simulation comprend en fait plusieurs parties, qui vont être décrites maintenant.

4.1 Visualisation

La MOCN étant modélisée sous la forme d'une hiérarchie d'objets Inventor, sa visualisation peut alors se faire à l'aide du "SceneViewer" d'Inventor. Nous "héritons" alors de toutes les primitives de manipulation associées à cet outil :

- il est possible de se déplacer à l'intérieur de la scène (l'univers) modélisée, et ceci de plusieurs manières (mode "walk" ou "fly").
- il est possible de changer, de façon interactive, le "point de vue" de la scène, c'est-à-dire le positionnement des caméras par lesquelles on voit la scène.
- il est possible d'effectuer toutes sortes de manipulations (interactives, bien sûr) sur les objets qui composent la scène (rotation, translation, facteur d'échelle, ...).

Il faut alors nous imposer certaines restrictions quant à ces diverses possibilités de manipulation : s'il ne paraît pas forcément souhaitable de pouvoir traverser les murs, ce détail est négligeable par rapport aux conséquences visuelles de certaines manipulations permettant par exemple de dissocier littéralement notre MOCN. A l'heure actuelle, nous n'avons pas interdit l'accès à toutes ces manipulations, il faut donc bien faire attention à ne pas casser la machine !

4.2 Gestion des interactions

Comme nous venons de le signaler, le fait d'utiliser les classes C++ d'Inventor nous permet d'agir facilement et de façon interactive sur les objets d'une scène. Le premier problème rencontré, déjà signalé, est que l'accès à ces primitives peut se faire de façon désordonnée et non pertinente. Une façon simple de le résoudre est de s'interdire moralement de telles manipulations, ce qui nous suffira pour le moment.

Le second problème rencontré est que la MOCN doit réagir à des actions particulières en provenance du boîtier de commandes : il faut ici associer des procédures, appelées "callbacks", aux objets concernés afin d'être en mesure de répondre à ces actions. Ceci est pour l'instant réalisé en injectant du code C++ à l'intérieur du code Inventor. A terme, nous souhaitons pouvoir réaliser ceci à l'aide d'un outil de même type que celui permettant de modéliser la géométrie de la machine.

4.3 Traductions des commandes

Grâce à un interpréteur que nous avons développé, nous sommes en mesure de traduire des programmes destinés à la MOCN en des actions simples : il ne nous reste plus qu'à les associer aux composants correspondants de notre machine virtuelle. Il devient alors intéressant de traiter dans un même module l'envoi de ces "tâches logicielles" et l'envoi des requêtes interactives.

C'est donc dans ce module que doivent être définis les différents événements susceptibles de survenir ainsi que les actions à associer à ces événements.

4.4 Architecture logicielle

Un aspect important de cette simulation est celui de son architecture logicielle : comme la simulation servira de base à la télésurveillance et à la téléopération, elle devra être structurée de telle façon que ses différentes parties puissent être aisément réutilisables par la suite.

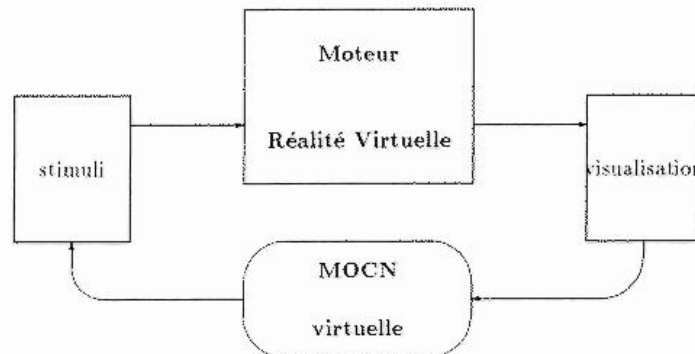


figure 1 : Architecture logicielle de la simulation

Actuellement, l'architecture est la suivante :

- un module de visualisation de la MOCN : simulation et télésurveillance,
- un module de gestion des interactions (acquisition des commandes) : simulation et téléopération,
- un module d'interprétation de commandes : simulation et téléopération,
- un module "expert" contenant une description des contraintes de déplacement de la machine virtuelle : simulation, télésurveillance et téléopération.

5 Extensions du projet

5.1 Télésurveillance de la MOCN

Le deuxième point de ce projet est l'observation à distance de la MOCN à l'aide de notre station de travail. Ceci pourra se faire à l'aide des travaux de simulation, en substituant aux entrées de l'utilisateur des informations de déplacement effectif en provenance de la machine réelle.

Grâce à notre architecture logicielle, nous pouvons récupérer ici le module "visualisation" qui a servi pour la simulation. Il faut venir greffer une partie supplémentaire au module d'interprétation de commandes, de façon à pouvoir lui envoyer les événements de déplacement en provenance de la machine réelle.

5.2 Téléopération de la MOCN

Le troisième et dernier point du projet consiste à commander à distance la MOCN, à partir de notre station de travail.

Ceci peut en fait être une extension de notre simulation avec envoi des instructions (résultant des actions de l'utilisateur) à la MOCN réelle. Il faut réutiliser le module gestion des interactions, et il faut greffer une partie supplémentaire au module de visualisation pour traduire les demandes de déplacement en événements compréhensibles pour la MOCN réelle.

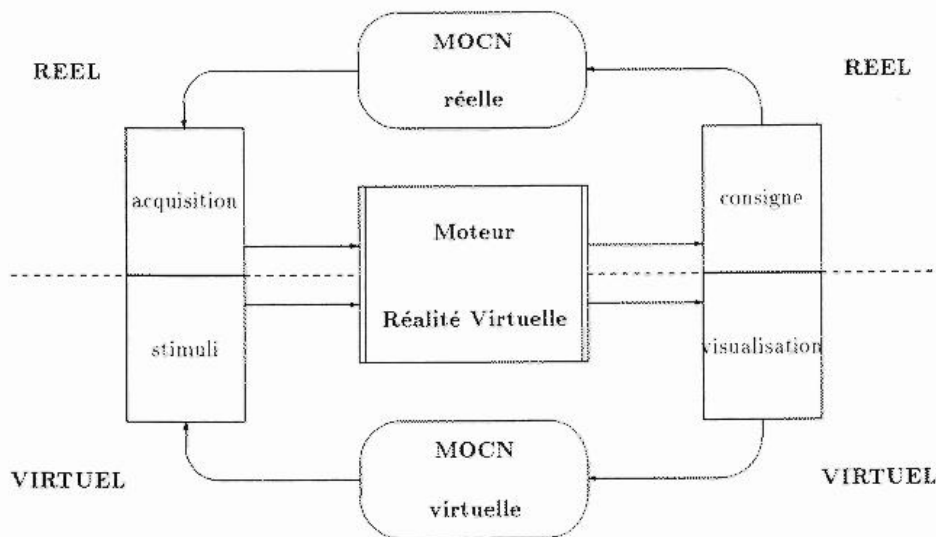


figure 2 : Architecture logicielle de la téléopération

5.3 Architecture du Réseau

Le fait de surveiller, ou de commander, à distance une machine à l'aide d'une station de travail implique l'utilisation d'un protocole de communication. Les données en provenance de notre MOCN sont récupérées à l'aide d'un PC (via un port RS232) connecté au réseau Internet. Ces données sont alors envoyées "au fil de l'eau" à la station de télésurveillance. Pour la téléopération, le processus sera similaire, mais dans les deux sens.

5.4 Améliorations

Actuellement, nous ne nous sommes pas préoccupés de simuler un usinage. Nous envisageons d'associer à notre simulateur un simulateur d'usinage existant (un logiciel de CFAO) qui nous permettrait ainsi de visualiser de façon interactive l'usinage d'une pièce par notre machine.

6 Conclusion

Ce projet nous a permis de mettre en œuvre notre Atelier de Réalité Virtuelle sur un cas réel non trivial : la simulation d'une MOCN.

Nous pouvons valider un certain nombre de concepts, dont les principaux présentés ici sont la modélisation de l'architecture et du comportement de la machine réelle, et l'architecture modulaire du simulateur, ou plus exactement du moteur de réalité virtuelle.

Nous devons maintenant nous attacher au développement d'outils logiciels permettant de spécifier les interactions en provenance d'un utilisateur. Il faut également limiter l'usage de certains modes d'interaction du SceneViewer. Il reste alors à réaliser effectivement les phases de télésurveillance et de téléopération.

Ce travail s'inscrit tout à fait dans les orientations actuelles de la communauté scientifique européenne dans le domaine de la Réalité Virtuelle [5].

Références

- [1] Autodesk, *3D Studio v3*, Autodesk Development BV, Neuchâtel, Suisse, 1993.
- [2] Burdea G., Coiffet P., *La Réalité Virtuelle*, HERMES, 1993.
- [3] Carlson C., Hagsand O., *DIVE : A Platform For Multi-User Virtual Environments*, Computer & Graphics, Vol. 17, n°6, pp 663-669, 1993.
- [4] De Cambray B., *Modélisation 3D : état de l'art*, Technical Report, MASI, France, 1992.
- [5] Encarnaçao J., Göbel M., *European Activities in Virtual Reality*, IEEE Computer Graphics & Applications, pp 66-74, January 1994.
- [6] Figueiredo M., Böhm K., Teixeira J., *Advanced Interaction Techniques in Virtual Environments*, Computer & Graphics, Vol. 17, n°6, pp 655-661, 1993.
- [7] Rank Xerox, *GraphTalk v2.5*, Parallax Software Technologies, Puteaux, France.
- [8] Silicon Graphics Inc., *IRIS Inventor Programmer's Guide*, 1992.