Interaction "naturelle" en environnements immersifs - Démonstrateur multimodal et validation sur des applications scientifiques

Damien Touraine

UNIVERSITÉ PARIS XI
UFR SCIENTIFIQUE D'ORSAY

THÈSE
Présentée pour obtenir
Le GRADE de DOCTEUR EN SCIENCES
DE L'UNIVERSITÉ PARIS XI ORSAY
par

Damien TOURAINE

Sujet :

Interaction « naturelle » en environnements immersifs

Démonstrateur multimodal et validation
sur des applications scientifiques

Soutenue le

Lundi 7 Avril 2003

Membres du jury :
- Roger Reynaud, Professeur à l'IEF (Orsay) - *Président du jury*
- Pierre Boulanger, Professeur à l'Université d'Alberta (Canada) - *Rapporteur*
- Philippe Fuchs, Professeur à l'école des mines de Paris - *Rapporteur*
- Bruno Arnaldi, Professeur à l'IRISA (Rennes)
- Patrick Bourdot, Chargé de recherche au LIMSI-CNRS
- Angel Osorio Sainz, Directeur de recherche au LIMSI-CNRS - *Directeur de thèse*

Table des matières

Liste des figures

Remerciements

Je tiens à remercier tous ceux sans qui cette thèse n'aurait pu se dérouler dans de si bonnes conditions.

Tout d'abord, je remercie Angel Osorio d'avoir dirigé ma thèse.

Je tiens particulièrement à remercier Patrick Bourdot, qui pour ma thèse « fût tout, mais ne fût rien » (administrativement parlant, j'entends). Au-delà de son rôle d'encadrant de ma thèse, je respecte son esprit de « Directeur de Recherche » qui permit tant de choses. En espérant que cet esprit fasse aboutir notre prochain projet : EVE.

Je remercie, également les membres de mon jury : Pierre Boulanger et Philippe Fuchs d'avoir rapporté sur ma thèse. Je vous remercie également d'avoir accépté d'amender mon document afin de l'améliorer. Je souhaite aussi un bon séjour en France à Pierre.

Je remercie également Bruno Arnaldi qui a su mettre en place et diriger le projet RNTL PERF-RV, projet d'une grande importance pour le Réalité Virtuelle en France. Je voudrais y associer Roger Reynaud qui permet d'étendre mes travaux dans le domaine de la Réalité Augmentée dans le cadre du projet VARVIC.

Je tiens également à remercier le CNRS pour sa dotation spéciale qui permit d'acquérir le dispositif immersif μse. Ce dispositif m'a permis de valider l'ensemble de mes travaux.

Je remercie, dans son ensemble, le LIMSI-CNRS et plus particulièrement les membres du groupe Geste et Image qui m'ont accueilli dans leur équipe.

Olivier, Joe et Bruno : encore une fois, désolé de ne pas avoir « lâché » le *fly* à temps ... Maintenant que mon lit repose au 508, n'hésitez pas : je vous le prête ! Quand à Jean-marc, ne désespère pas : tu arriveras à faire comprendre « stéréo » à ce fichu ordinateur. Il m'a fallu 3 ans pour dompter la bête ! Même si la dernière phrase à la mode est « Mets cet objet dans ma main » !

Plus personnellement, je remercie, ma famille qui me supporte depuis plus de 25 ans. Enfin, je tiens à associer à cet ensemble de remerciements, Frédéric qui m'a prodigué maints conseils et n'a pas hésité à essuyer les plâtres en relisant mon mémoire. A ce sujet, j'ai une pensé particulière pour tous ceux qui ont eu la gentillesse bienveillante de relire ma thèse.

Introduction

De nos jours, de plus en plus de laboratoires et d'industriels développent ou utilisent des systèmes de Réalité Virtuelle (RV). La plupart du temps, les interactions sur ces systèmes se réduisent à transcrire dans les environnements virtuels 3d les interactions Fenêtres, Icônes Menus et Pointeur (cf. WIMP - *Windows*, *Icons*, *Menus* and *Pointing*). De plus, à l'instar des grands industriels de l'automobile, les applications de Réalité Virtuelle sont souvent cantonnées à la visualisation de données dans des formats très appauvris (surfaces polyédriques) malgré la topologie parfois complexe de certains objets (cf. CAO). A l'heure actuelle, où le domaine de la Communication Humain-Machine (CHM) commence à atteindre un niveau de fiabilité acceptable, une de mes problématiques est d'intégrer toutes les modalités disponibles pour interagir dans les environnements immersifs. Dans ce contexte, j'ai étudié comment intégrer plusieurs concepts et paradigmes susceptibles de rendre plus « naturelles » les interactions immersives.

Cependant, avant de commencer l'exposé de mes travaux, il importe de remarquer que la perception humaine dépend de beaucoup d'aspects cognitifs. En effet, sur la base de ses expériences perceptuelles, l'humain acquiert une imagerie mentale (tant visuelle, qu'auditive ou haptique) à partir de laquelle il construit toute une connaissance spatiale sur l'univers réel. Il utilise souvent cette connaissance pour développer des stratégies cognitives dans l'appréhension des mondes virtuels. Ainsi, étudier les interactions « naturelles » suppose la prise en considération de ces aspects cognitifs. Cependant, en amont de ces considérations qui relèvent pour partie de la psychologie cognitive, mon travail était d'avantage axé sur l'intégration des diverses modalités. De ce point de vue, le qualificatif d'interaction « naturelle » est bien entendu un abus de langage (je reviendrai ce point en section F, page 23).

Dans le cadre d'applications 3d immersives ou non, plusieurs auteurs ont déjà montré que le modèle WIMP a pour défaut majeur d'imposer à l'utilisateur de changer fréquemment de contexte de travail (cf. : zone de travail 3d / zone de commande). De plus, dans certains cas, l'utilisateur n'a pas la possibilité de saisir un pointeur pour selectionner un bouton ou un menu. C'est par exemple le cas pour les applications chirurgicales dans lesquelles le praticien ne peut pas lâcher ses instruments. Une solution est de proposer une approche multimodale de l'interaction : à partir de plusieurs événements issus de périphériques tels que les reconnaissances vocale et gestuelle, le système est capable de générer un événement sémantique de plus haut niveau. Cependant, ce type de système impose un certain nombre

de contraintes au gestionnaire de périphériques. Ainsi, pour mettre en œuvre une telle approche en RV, je présente dans le **chapitre 2** l'architecture distribuée de la plate-forme EVI3d (cf. Environnements Virtuels et Interaction 3d). La distribution des traitements associés aux périphériques nous a été imposée par la mise en œuvre de certains périphériques. En effet, les systèmes de reconnaissance utilisent généralement beaucoup de ressources. Afin d'éviter d'en allouer au détriment du système de rendu, nous avons préféré exporter ces tâches sur d'autres calculateurs. Plus globalement, le but de l'architecture EVI3d est de permettre la flexibilité requise par l'implémentation de nouveaux paradigmes interactifs. En effet, avec les systèmes actuels, nous sommes encore trop souvent limités dans les études ergonomiques et cognitives par des contraintes inhérentes au mode de gestion des périphériques immersifs ainsi qu'à la gestion des événements qui en résulte.

Le deuxième volet de ma thèse part du constat que les environnements logiciels actuels ne permettent pas la modularité requise par l'architecture logicielle proposée ainsi que pour les interactions « naturelles » étudiées. En effet, aucune librairie ne prévoit l'interaction au travers de périphériques distribués et d'un serveur multimodal d'événements. De plus, ces librairies sont globalement optimisées pour la gestion de scènes statiques et utilisent souvent l'ensemble des ressources de la machine. Ces librairies sont donc peu propices à l'interaction sur des bases de données scientifiques dynamiques. En effet, ce type de scène est souvent basé sur un amas de facettes évoluant au cours de temps. Il faut alors libérer des ressources pour calculer ces évolutions. Le **chapitre 3** présente l'ensemble logiciel que j'ai conçu pour autoriser cette distribution des applications scientifiques de Réalité Virtuelle, tout en offrant les fonctionnalités standards de gestion des dispositifs immersifs existants. Un point clef du noyau géométrique de la plate-forme logicielle EVI3d est la métaphore de « véhicule ». Ce concept permet de dissocier complètement le monde virtuel du monde réel, tout en constituant le point de passage obligé entre ces deux mondes. Issu des travaux du LIMSI-CNRS en matière de contrôle des navigations virtuelles, ce concept est devenu la base même de notre noyau géométrique. Au-delà de l'architecture dédiée à la gestion des interactions « naturelles » et à la distribution des traitements, ce noyau géométrique permet le développement d'applications de RV sur les dispositifs immersifs les plus variés. Ce chapitre se poursuit par la présentation du système de navigation sur lequel j'ai travaillé.

Au demeurant, mes travaux de thèse n'ont pas été menés sans un certain travail de conceptualisation sur la Réalité Virtuelle. Dans ce contexte, le **premier chapitre** de cette thèse présente un aperçu des concepts de base de la RV et fait le point sur l'état de l'art en matière d'interaction immersive. Ensuite, je définis la terminologie employée dans le reste de mon document et précise les différents types d'objets rencontrés en Réalité Virtuelle. Je terminerai ce premier chapitre en passant en revue les différentes modalités interactives disponibles en RV, ainsi que les périphériques associés.

Enfin, dans le **quatrième chapitre** de ma thèse, je montrerai les contributions, les validations ainsi que les perspectives de mes travaux de thèse. Il sera découpé en deux ensembles de sections. Un premier ensemble sera consacré aux validations et aux évaluations. Dans cette partie, j'aborderai les évaluations des latences du serveur distribué

d'événements (cf. EVserveur) et montrerai l'ordonnancement des différents composants internes du noyau géométrique. J'achèverai ce volet par les validations mises en œuvre en présentant le démonstrateur multimodal mis en place sur la base de mes travaux, et en faisant le point sur les différentes applications à caractère scientifique utilisant l'architecture EVI3d et son noyau géométrique. Le second ensemble mettra en avant les perspectives à moyen et à long terme. Je commencerai par montrer comment l'intégration d'un canal de type flot de données dans l'architecture EVI3d permettra entre autre l'interfaçage entre un calculateur graphique et un super calculateur. Ensuite, je décrirai les protocoles d'évaluation envisagés pour la validation ergonomique du système de contrôle des navigations.

Les outils actuellement proposés dans le cadre de la Réalité Virtuelle sont tous insuffisants pour mener à bien des études approfondies dans le cadre de la cognition humaine. En effet, certains paramètres tels que ceux relatifs au traitement des latences sont incontrôlables. De plus, ils sont peu propices pour mener des expérimentations sur de nouveaux paradigmes interactifs comme par exemple la multimodalité. Enfin, peu de ces systèmes proposent la répartition des traitements interactifs sur plusieurs calculateurs. Mes travaux portent donc sur la conception d'un système permettant, entre autres, de proposer une solution à ces trois problèmes.

Chapitre 1

Positionnements conceptuels sur la Réalité Virtuelle et l'Interaction 3d

Sommaire

1 Introduction

Il existe de nombreuses définitions de la Réalité Virtuelle (RV). En fait, l'interaction en environnements immersifs est à l'intersection de différents domaines, de sorte qu'il est très difficile de donner des définitions admises par l'ensemble des communautés impliquées.

Par contre, il est communément admis que l'apparition de la RV est très liée à l'interaction sur les objets de mondes virtuels. Cependant, on peut aussi considérer que des périphériques utilisés pour manipuler ces objets seraient restés impuissants sans l'émergence des systèmes de rendu des mondes virtuels (carte graphique temps réel, outils de CAO, logiciels d'animation,...).

Une autre approche pour cerner les attendus conceptuels de la RV est l'analyse des scènes qu'elle manipule. En effet, les mondes virtuels peuvent être composés de plusieurs types d'objets. De plus, la structure des objets peut influencer le mode d'interaction que nous avons avec eux.

Un thème récurrent dans la plupart des définitions de la RV [FMP01] est la notion de réalisme. Toutes les applications de la RV, et en particulier les applications scientifiques, ne cherchant pas a priori une modélisation réaliste des scènes virtuelles, notre propos se focalisera sur ce qui rend « naturelle » l'interaction immersive.

Ce premier chapitre vise donc à donner un aperçu des concepts de base de la RV, ainsi qu'à faire le point sur l'état de l'art en matière d'interaction immersive. Pour ce faire, je commence par préciser la terminologie employée dans ce document pour décrire mes travaux. J'étudie ensuite les différents types d'objets disponibles en Réalité Virtuelle. Enfin, les deux dernières sections de ce chapitre développent des considérations sur les périphériques en entrée et en sortie et leurs applications à l'interaction immersive.

2 Concepts et définitions

Etant donné que l'interaction en environnements immersifs est à l'intersection de domaines aussi variés que l'informatique graphique, la communication humain-machine, ainsi que les sciences cognitives et sociales, il est très difficile de donner des définitions admises par l'ensemble des communautés impliquées. Aussi, il m'a paru nécessaire de préciser, dès le début de ce premier chapitre, les termes employés dans ce document en explicitant certains de mes positionnements conceptuels par rapport à la RV.

2.1 Réalité Virtuelle et Augmentée

La Réalité Virtuelle et Augmentée est le domaine de l'informatique qui vise à interagir avec des objets numériques décrits dans la mémoire d'un ordinateur.

Pour mener différentes tâches plus ou moins collaboratives, relatives à des applications tridimensionnelles, on peut voir la réalité virtuelle et augmentée comme un ensemble d'outils matériels et logiciels qui permet de gérer de façon immersive et « réaliste » des interactions sensorimotrices d'opérateurs humains, d'une part sur des objets réels via un monde virtuel, d'autre part sur des entités virtuelles plus ou moins contraintes par l'univers réel [Bou02].

Il y a deux classes d'entités virtuelles.

Entités purement virtuelles

Ce sont des entités dont le modèle original n'a d'existence que dans l'imagination de leur concepteur. Il n'existe aucun lien entre ce type d'entité et une quelconque entité équivalente dans le monde réel. C'est le cas des dinosaures de *Jurassic Park*, dont la représentation est entièrement basée sur des modèles issus d'études paléontologiques. A ce jour, par exemple, aucune preuve n'a été apportée sur la pigmentation exacte de leur épiderme, de sorte que les textures employées ne résultent que de l'imagination humaine. Ce sont des entités purement virtuelles.

Cependant, chaque entité virtuelle doit être susceptible d'avoir un comportement propre et possède pour ce faire des descriptions de lois physiques ou à défaut de mouvements prédéterminés qui pilotent ses interactions avec les autres entités de ce monde virtuel. Si l'entité n'a aucun comportement propre, alors, elle sera inerte dans le monde virtuel. Ces entités virtuelles peuvent se décomposer en deux types :

– *Humanoïdes.* Les humanoïdes sont des entités qui représentent des êtres humains. Cependant, ces êtres humains obéissent à des règles prédéfinies et rien qui ne soit préalablement prévu ne peut influencer son comportement. En d'autres termes, même si à un instant t il peut refléter un comportement social standard, la réponse à un stimulus inconnu sera indéterminée. Il n'existe donc pas de lien direct entre un humanoïde et un être humain complet.

– *Objets purement virtuels.* Les comportements propres de ces objets peuvent être proche de ceux d'un objet équivalent dans le monde réel. Cependant, la richesse d'un objet virtuel est sa potentialité à violer des lois physiques du monde réel. Au-delà de ces aspects comportementaux, il n'existe aucune corrélation entre ces objets et un quelconque objet dans le monde réel.

Entités ayant une représentation concrète dans le monde réel

D'un autre côté, pour décrire plus précisément les interactions des opérateurs humains et des objets réels avec le monde virtuel, la figure 1.1 montre aussi qu'il nous faut considérer deux autres types d'entités : les « avatars d'humains » et les « avatars d'objets ». Contrairement à [Sta02] qui limite le concept d'avatar aux êtres humains, nous nous rapprochons de [SC02] qui introduit aussi cette notion dans le cadre d'objets réels.

Ces avatars sont : soit issus de la numérisation d'une entité réelle, soit un modèle qui pilote un objet existant dans l'univers réel. Par exemple : la synthèse par EDF du sanctuaire d'Athena à Delphes par numérisation des différents éléments architecturaux, puis reconstruction du puzzle [Ath96].

– *Avatars d'humains.* Un avatar d'humain est une entité dont le comportement est calqué sur celui d'un être humain dans le monde réel. Cela requiert de capter le comportement de l'être humain dans le monde réel pour le retranscrire en « temps réel » au travers de son avatar dans le monde virtuel. Il faut ici noter l'absence d'« intelligence » de l'avatar d'humain par rapport à un humanoïde. En effet, ce dernier doit avoir un comportement autonome dans le monde virtuel.

  L'acception la plus courante d'avatar s'applique à la représentation des êtres humains dans le monde virtuel. Notre point de vue est d'augmenter cette représentation d'une contrepartie logicielle et interactive. C'est-à-dire que pour nous, un avatar humain est un module logiciel qui permet à chaque opérateur humain de communiquer avec le monde virtuel. Ainsi, au-delà d'une représentation plus ou moins réaliste d'une portion ou de la totalité du corps d'un utilisateur, l'avatar humain est une entité qui gère plus fondamentalement les périphériques[1] d'entrées et de sorties associés à l'opérateur humain. Il inclut ainsi tous les composants logiciels utiles à l'interac-

1. Le terme *périphérique* peut être interprété de deux façons différentes :
– **Extension de l'ordinateur.** C'est l'acception communément admise. Il s'agit d'un module matériel qui ne fait pas partie de l'unité centrale de l'ordinateur. Par extension, nous parlons aussi de périphérique lorsqu'il s'agit d'un module matériel ou logiciel qui joue le rôle d'interface entre le monde extérieur et l'ordinateur. Ce monde extérieur est, dans le cas de beaucoup de périphériques, destiné à l'interaction humain-machine (clavier, souris, capteur de mouvements ...).
– **Interface de l'utilisateur.** L'acception précédente est centrée ordinateur. Mon point de vue est que cette notion peut être inversée. Ainsi, l'acception dont je me sers dans mon document est celle où les périphériques sont des extensions de l'être humain qui permettent à ce dernier de se « connecter » au monde virutel. Cependant, je reviendrai sur les différents types de périphériques lors des sections 5 et 6 du présent chapitre.

tion humain-machine, comme par exemple : les traitements des événements sensorimoteurs, la gestion des interacteurs associés (« widget [2] » 3d, objets métaphoriques,...), le contrôle de la navigation, la gestion des scènes virtuelles (pour adapter la complexité de la scène à la position, l'orientation et l'action de l'utilisateur) et toute l'interface avec le noyau fonctionnel propre à l'application de Réalité Virtuelle ou Augmentée.

– *Avatars d'objets* De façon relativement analogue, un avatar d'objet est une entité qui permet à un objet réel d'interagir avec le monde virtuel. Sa représentation au sein du monde virtuel peut être définie à partir de processus de reconstruction 3d appliqués sur l'objet réel. Inversement, des systèmes robotiques peuvent permettre à l'avatar d'objet de modifier ou manipuler l'objet réel associé depuis le monde virtuel. C'est par exemple le cas lors de téléopération où l'utilisateur manipule l'avatar d'objet. Ce dernier agit alors sur son équivalent réel pour refléter la configuration du monde virtuel.

  S'il n'est plus en correspondance avec son objet réel, un avatar d'objet devient alors un objet purement virtuel. Quand l'environnement virtuel permet de gérer de tels changements d'états, alors il devient possible d'utiliser un tel système comme support à des tâches de conception ou de simulation d'objets (cf. CAO).

Réalité Augmentée L'augmentation nécessite, par définition, que la géométrie de la réalité soit connue de l'ordinateur. En d'autres termes, la Réalité Augmentée requiert une reconstruction du monde réel. Ainsi, l'existence des avatars d'objets est ce qui caractérise les applications de Réalité Augmentée.

Schéma coneptuel de la RV&A

En résumé, on peut observer que ce schéma conceptuel de la Réalité Virtuelle et Augmentée (figure 1.1) décrit les principales classes d'interactions immersives. Les deux entités supérieures représentent les interactions dans l'univers réel. Les applications purement orientées Réalité Virtuelle ne mettent en œuvre que les trois entités de gauche et l'entité en bas à droite : opérateur humain, avatar humain, humanoïde et objet virtuel. Les applications de téléopération peuvent être vues comme des systèmes ne faisant intervenir que les quatre entités supérieures : opérateur humain, avatar humain, avatar d'objet et objet réel. Finalement, l'implication de l'ensemble des entités de ce schéma décrit de façon complète un système de Réalité Augmentée.

Le fondement d'un environnement virtuel est donc de procurer à des utilisateurs des fonctionnalités interactives sensorimotrices (principalement : visuelles, sonores et gestuelles) pour mener des activités immersives « réalistes » sur des scènes tridimensionnelles numériques, éventuellement issues de numérisations sur l'univers réel, voire en interaction

2. Un widget est un élément graphique qui permet de faire une action (bouton, menu déroulant, barre de défilement, potentiomètre. . .).

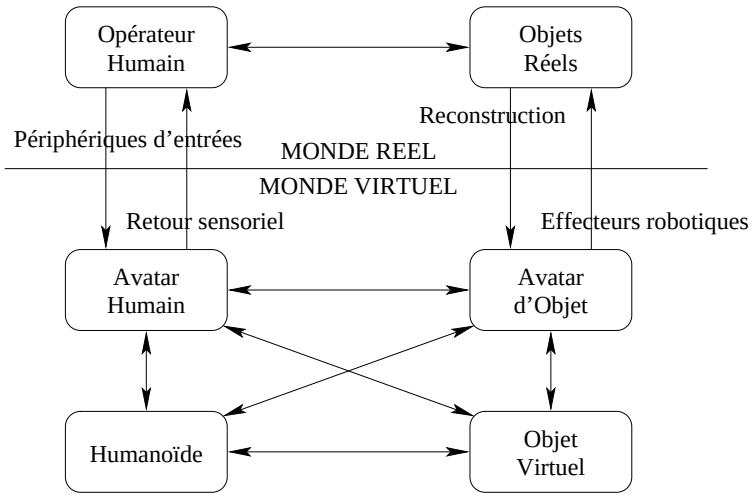

FIG. 1.1 – *Schéma conceptuel de la RV&A.*

avec celui-ci. Il en résulte que de tels systèmes requièrent des processus complexes d'analyse et de synthèse d'informations (capteurs, reconstruction 3d, retours perceptifs, actionneurs robotiques...). Au demeurant, une des problématiques de la Réalité Virtuelle et Augmentée est de doter les entités du monde virtuel d'aspects et de comportements réalistes : géométriques, physiques, mais aussi cognitifs. Par ailleurs, pour structurer sémantiquement les combinaisons en entrée et en sortie des événements sensori-moteurs, différents travaux portent aussi sur le traitement multimodal de ces interactions [CMO+99]. Enfin, un autre type de préoccupations est de permettre des tâches coopératives entre utilisateurs au sein d'un même ou via plusieurs environnements virtuels.

Cette façon de présenter la Réalité Virtuelle et Augmentée ne prétend pas être une définition parfaite de ce domaine. Ce qui vient d'être décrit n'est que le schéma conceptuel qui m'a servi de base à la structuration de mes travaux.

Interaction avec les objets

Au niveau de l'interaction, il est intéressant de constater que l'ensemble des interactions que j'ai étudié pour la Réalité Virtuelle est applicable à la Réalité Augmentée. Par contre toutes les interactions de Réalité Augmentée ne sont pas toujours utilisables en Réalité Virtuelle.

Par exemple, l'un des paradigmes uniquement applicable en Réalité Augmentée est celui de la modification directe de l'objet réel. Ainsi, avec des capteurs de déformation, nous pouvons appliquer ces modifications à l'avatar d'objet. Ce paradigme n'est pas envisageable dans le cas de la réalité virtuelle où, par définition, il n'existe pas de relation explicite entre un objet virtuel et un objet équivalent dans l'univers réel. En effet, il y a très souvent des relations, au moins du point de vue cognitif pour l'utilisateur, mais elles ne sont pas

forcément explicitées dans le modèle des objets virtuels, ni explicitables par le système.

L'une des premières contraintes est de visualiser les objets virtuels. Nous reviendrons sur ce point en section C

2.2 Monde virtuel

Le monde virtuel est le lieu de représentation de tous les objets virtuels et augmentés d'une application.

- Le monde virtuel n'a de limites que celles de la précision de l'ordinateur ;
- Son référentiel propre est nommé référentiel de la scène ;
- Ce dernier n'est a priori corrélé avec aucun référentiel réel (c'est-à-dire que le référentiel du monde virtuel n'est pas obligatoirement fixé à un endroit précis dans l'univers réel) ;
- Le monde virtuel peut d'ailleurs être anisotropique (il peut par exemple avoir une composante selon l'axe des X dilatée par rapport à celle sur l'axe des Y) ;
- L'utilisateur peut dilater ou contracter son échelle propre à sa guise ;
- Les lois physiques qui régissent le monde virtuel ne sont pas forcément celles du monde réel ;
- Le monde virtuel peut contenir des propriétés mathématiques exotiques.

Par exemple, les trois composantes de position peuvent être totalement absentes de ce monde virtuel. Elles peuvent laisser la place à des scalaires représentant des pressions et des vecteurs de vorticité. Ces deux dernières composantes sont fortement utiles pour les études d'écoulements en mécanique des fluides.

D'un autre côté, des lois peuvent stipuler que la lumière a une vitesse finie et que nous pouvons accélérer des objets à des vitesses proches de cette dernière. Ce monde serait le terrain d'essai idéal pour valider certains concepts de la relativité restreinte.

Cependant, le monde virtuel n'est pas obligatoirement limité à quatre dimensions (3 pour la position et une pour le temps). Ainsi, en ajoutant des champs gravitationnels dans ce monde (sous la forme de trois dimensions supplémentaires), un système de RV permettrait de valider les théories de la relativité générale. Cependant, ce monde n'est déjà plus euclidien[3] !

De plus, les lois physiques du monde virtuel peuvent stipuler que l'énergie est discrétisée sous la forme de « quanta » d'énergie et que tout élément est une onde électromagnétique. Ainsi, ce monde serait idéal pour étudier les théories quantiques. Par exemple, une représentation possible de ce monde est celle où la probabilité de présence d'une particule donnée est rendue sous la forme de transparence : la valeur de transparence au point considéré est directement égale à la probabilité de présence de la particule en ce point.

3. En effet, en vertu de la relativité générale, la plus courte distance entre deux points n'est plus la ligne droite.

Inversement, le monde virtuel peut avoir un nombre de dimensions inférieur à quatre. Par exemple, ce monde peut représenter une information bidimensionnelle telle que la succession des molécules dans un brin d'ADN[4]. Ainsi, la représentation de ce brin selon une table de conformation spatiale[5] permet des études tridimensionnelles des molécules génomiques [GH00].

On peut aussi développer un monde à u dimensions où u est réel ($u \in \Re^{+*}$). Le développement d'un monde de dimensions non entières sur des ordinateurs est d'autant plus compliqué que ces derniers possèdent par nature une représentation discrète. Cependant, un tel monde virtuel permettrait d'étudier des objets fractals[6].

2.3 Environnements immersifs

Un environnement immersif est un dispositif qui permet à l'homme de se plonger dans un univers sans relation explicite avec le monde réel. Afin d'immerger l'utilisateur dans le monde virtuel, il faut lui transmettre des informations sur ce monde et lui permettre de modifier ce monde. Le degré d'immersion sera plus ou moins important selon la quantité et la qualité des informations transmises.

Au niveau cognitif, l'immersion peut être réalisée à différents degrés. En effet, l'être humain peut se faire une représentation mentale du monde virtuel à partir d'informations (même fragmentaires) issues de ce dernier. Cependant, la charge cognitive sera inversement proportionnelle à la quantité d'information sur le monde virtuel fourni à l'utilisateur. Cette charge doit cependant être pondérée par la qualité de la représnetation perceptive de cette information.

L'immersion n'est possible qu'à travers des périphériques qui vont exciter des canaux sensori-moteurs. Le canal sensoriel primaire de l'être humain est la vue. Ainsi, la sensation d'immersion sera plus importante avec un large champ visuel, ce qui explique l'engouement pour les environnements immersifs de grande taille (et les grands écrans en général).

Afin de permettre une interaction « naturelle » (voir paragraphe F, page 23), on fait ici l'hypothèse que l'environnement immersif doit induire une charge cognitive, sur chaque modalité, équivalente à celle associée à la perception du monde réel. C'est pourquoi, contrairement à l'acception courante de la littérature, même s'il est possible de synthétiser le monde virtuel sur l'écran d'une station de travail, je n'attribue pas le statut d'environnement immersif à ce type de dispositif. Cette restriction se justifie par le fait que les bords de l'écran d'une station graphique sont beaucoup trop visibles par rapport à la taille de la fenêtre de visualisation, ce qui « casse » la sensation immersive attendue et induit potentiellement une surcharge cognitive.

Nous parlons d'une surcharge potentielle car la charge cognitive d'une tâche est variable en fonction de l'utilisateur et de son expertise. Par exemple, à force de pratiquer, le chirur-

4. L'aspect bidimensionnel est représenté par l'abscisse curviligne le long du brin selon un axe et le choix parmi les quatres molécules de base (Adénine, Cytosine, Guanine et Thymine) selon l'autre axe.

5. Une table qui fournit le changement de référentiel à appliquer entre deux molécules (A, C, G ou T) successives du brin.

6. Un objet fractal est un objet mathématique de dimension non entière.

gien verra sa charge cognitive diminuer pour des tâches d'endoscopie sur un petit écran. Ainsi, en utilisant le même type de dispositif pour le rendu d'un monde virtuel chirurgical, nous pouvons atteindre un niveau de cognition équivalent. Cependant, le praticien est un cas particulier du large panel des utilisateurs potentiels. Ainsi, la degré d'immersion d'un dispositif immersif dépend non seulement de l'utilisateur, mais en plus de son expertise.

2.4 L'interaction

L'interaction est le sujet premier de mes recherches lors de ma thèse. D'après le dictionnaire :

– *Interaction* : Actions mutuelles de plusieurs corps ou systèmes susceptibles de s'influencer.

Dans le cas qui nous intéresse, ces interactions sont celles de l'utilisateur qui travaille sur les objets virtuels, mais aussi, celles issues des objets qui ont une influence sur le comportement de l'utilisateur.

A Changements dimensionnels

Dans les faits, les environnements immersifs sont entièrement basés sur une interaction tridimensionnelle. C'est-à-dire que l'interaction se fait depuis le monde réel et les retours sensoriels sont à destination du monde réel. Par la suite, j'appellerai *monde isomorphe au réel*[7] tout monde euclidien tridimensionnel doté ou non d'une composante temporelle.

Cependant, l'intérêt des environnements immersifs est leur capacité à rendre les mondes virtuels de tous types. Or, le paragraphe précédent sous-entend que les environnements immersifs sont plus adaptés au rendu de mondes isomorphes au réel. Ainsi, il faut donc trouver le moyen de transformer les mondes non isomorphes en des mondes isomorphes au réel. Je nommerai *espace de représentation d'un monde virtuel*, le monde isomorphe au réel qui est l'image issue d'une transformation d'un monde virtuel donné (voir figure 1.2). Pour passer du monde virtuel vers son espace de représentation, plusieurs algorithmes existent dont nous verrons les plus courants dans les sections qui traitent des différentes modalités.

Cependant, ces algorithmes deviennent limités, lorsqu'il s'agit de transformer une interaction décrite dans l'espace de représentation, dans le monde virtuel d'origine. En effet, dans la transformation vers l'espace de représentation, les algorithmes imposent généralement une réduction de dimensions. Or, cette réduction induit une extrapolation des dimensions annihilées lors du passage dans l'espace de représentation. En d'autres termes, s'il y a injection du monde virtuel vers l'espace de représentation, il n'y a pas bijection entre le monde virtuel et son espace de représentation. Le problème est similaire à celui qui limite l'utilisation de la souris (2d) pour décrire une interaction dans un monde à trois dimensions.

Nous pouvons voir, figure 1.2, que l'espace de représentation est décomposable en plusieurs sous-espaces. Ces sous-espaces ne sont en fait pas tous isomorphes au réel. En effet,

7. sous-entendu : monde isomorphe à l'univers réel

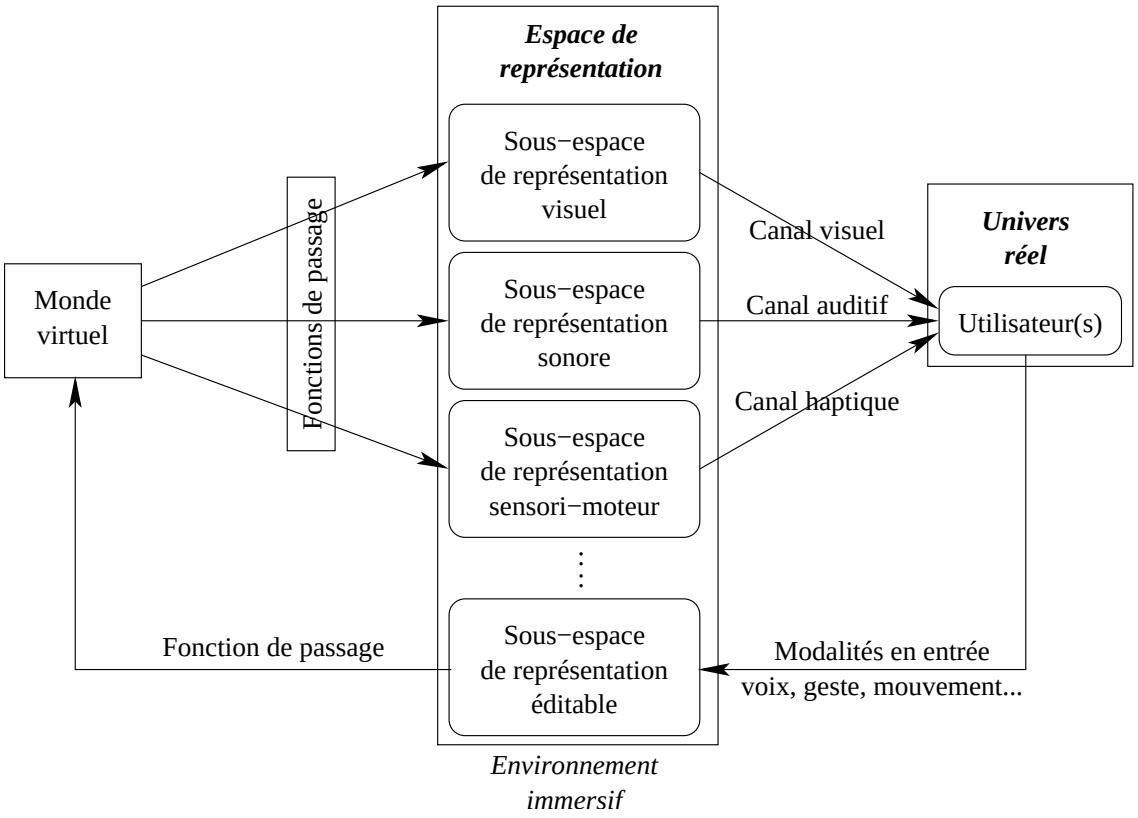

FIG. 1.2 – *Articulation de l'espace de représentation, décomposé en sous-espaces propres à chaque modalité, entre le monde virtuel et l'univers réel.*

dans ce qui précède nous avons uniquement abordé le sous-espace associé à la modalité visuelle. Ce dernier doit être isomorphe à l'univers réel. Il existe d'autres sous-espaces qui ne le sont pas, tel que celui dédié à la modalité sonore.

D'un autre côté, il peut exister des espaces de représentation qui permettent à l'utilisateur d'éditer le monde virtuel. Ainsi, le sens des flèches dans la figure 1.2 représente les relations transitives dans le monde virtuel. Cependant, je n'ai pas représenté les relations internes à l'environnement immersif. Par exemple, il n'est pas rare que le capteur de mouvement de la main affecte le sous-espace de représentation visuel pour imposer un déplacement de l'avatar de la main (cf. retours lexicaux evoqués par exemple dans [Bel95]).

Si ces considérations sont conceptuellement intéressantes pour situer mon travail, elles sortent cependant du champ effectif des investigations de ma thèse. En effet, dans la suite de ce mémoire je me limiterai à l'étude des mondes isomorphes au réel. Nous ne pouvons en effet pas envisager de créer un système universel d'interaction sur un monde de dimension appartenant à $\Re^{+*}$ car chaque monde a ses propres lois et donc, ses propres transformations vers son espace de représentation propre. Au demeurant, les paradigmes

et solutions, qui permettent d'interagir avec le monde virtuel isomorphe au réel, proposés dans ce document sont suffisamment génériques pour être aussi utilisés pour interagir avec les espaces de représentation de mondes non isomorphes au réel. C'est pourquoi, dans la suite de ce document, pour des raisons de commodité, je parlerai « abusivement » de monde virtuel pour désigner en fait l'espace de représentation associé au monde virtuel sur lequel nous travaillons.

B Evénements

Toute interaction se fait grâce aux événements. Cela est dû à la technologie sous-jacente : l'ordinateur.

En effet, les systèmes d'exploitation les plus utilisés sont basés sur du *multi-tâche préemptif*. Ainsi, par défaut, les tâches attendent d'être réveillées par l'ordonnanceur du système. Ce dernier réveille une tâche donnée quand un événement demandé par cette dernière arrive dans le système (une action sur une touche du clavier, un mouvement de la souris, des donnés issues du disque sont disponibles sur le bus...).

Cependant, il existe plusieurs classes d'événements.

Evénements discrets

Un événement discret est basé sur une valeur ponctuelle et/ou non continue. Par exemple, les touches d'un clavier ou les « clics » souris sont des interactions discrètes. En d'autres termes, il n'y a pas, a priori, de corrélation envisageable entre deux événements issus du même périphérique. Au demeurant, certains traitements multimodaux visent parfois à corréler de tels événements.

Evénements continus

Inversement, d'autres événements sont basés sur des domaines de valeurs continues. Par exemple, le mouvement de la souris sur un tapis est continu et appartient à un domaine de variation décrivant l'ensemble de l'écran. Il existe une corrélation implicite entre deux événements de ce type puisque leurs valeurs sont proches.

Evénements périodiques

L'ultime aspect d'un événement continu est sa périodicité. C'est-à-dire que les signaux issus de ce type de périphérique peuvent être considérés comme un flux continu de données à une fréquence donnée. Dans les faits, on peut assimiler les événements périodiques à des événements continus.

Il existe des systèmes discrets qui sont périodiques comme le clavier. Le phénomène périodique est fourni par le pilote du périphérique qui régénère l'événement selon une fréquence programmable. Cependant, lorsque les types de données sont décorrélés, il n'est pas rare que la périodicité d'un événement soit remplacée par celle d'un événement plus récent. Typiquement, lorsqu'une touche du clavier est maintenue enfoncée, le signal répétitif

engendré sera supplanté par celui d'une autre touche. Pire encore, le relâchement de cette seconde touche ne rétablira pas le signal répétitif de la touche initiale si celle-ci est restée enfoncée.

C Interaction sur le monde virtuel

Il existe plusieurs types d'interactions applicables sur un environnement immersif ou une scène virtuelle.

Commande d'état du système

Au niveau le plus élémentaire, un premier mode d'interaction vise à pouvoir transmettre des commandes au système. Ces commandes sont fondamentales pour changer l'état du système.

Parmi les commandes, il peut y avoir la demande de chargement d'une base de données. D'un autre côté, les changements du mode graphique d'un dispositif ou ceux de l'état des périphériques sont aussi des commandes qui ont un impact déterminant sur les interactions de plus haut niveau.

Manipulation des objets

L'une des interactions de haut niveau qui est typique à la RV concerne la manipulation des objets virtuels. Cependant, avant de les manipuler, il faut les créer. C'est pourquoi, la plupart du temps, la manipulation des objets commence par une interaction discrète. De plus, lors de la manipulation des objets, il faut des interactions continues, par exemple pour déplacer les objets.

Observation

Après avoir créé un objet, on a souvent besoin de l'observer en détail, afin de déterminer si sa forme et son comportement sont conformes aux spécifications. Il faut donc que le système soit capable de permettre à un utilisateur de « tourner » autour de l'objet afin de l'étudier en détail. Cette étude peut être facilitée par la mise en place de zooms.

Navigation

Dans les environnements immersifs, les utilisateurs ont toujours un champ d'action relativement limité. En effet, l'utilisateur est contraint dans ses déplacements par les zones d'influence des différents capteurs. La navigation permet à un utilisateur de se déplacer dans le monde virtuel sans être affecté par les limites physiques réelles du dispositif. Cette observation est à l'origine d'un nouveau concept de « véhicule » et des travaux que nous avons menés sur le contrôle des navigations virtuelles (voir section 3.1 du chapitre 3).

D Substitution événementielle

Dans certains cas, nous sommes limités en termes de périphériques sur le nombre de types d'événements. Par exemple, dans le cas des jeux, la souris n'est pas suffisante pour décrire le contrôle de la rotation du joueur ainsi que sa translation. Ainsi, il peut être utile de substituer un type d'événement à un autre.

Evénements discrets pour un contrôle continu des navigations

Dans le cas de la navigation, il faut laisser à l'utilisateur le choix du point de vue. Or, par définition, un événement discret aura une discrétisation qui rend imprécis ce choix. Généralement, lorsque des événements discrets sont utilisés, ils sont cumulés avec un pas incrémental de déplacement. Il faut que ce pas soit suffisamment grand pour éviter que le déplacement ne soit trop lent. Avec un incrément trop petit par rapport à la scène, l'utilisateur mettra trop de temps pour atteindre son but. A contrario, il faut que cet incrément soit suffisamment fin pour permettre à l'utilisateur de choisir confortablement son point de vue. C'est pourquoi, certains systèmes prévoient plusieurs grandeurs d'incrément selon différentes modalités ou différentes actions sur la même modalité. Ce type de navigation est celui que l'on trouve sur les jeux vidéo.

Il existe aussi d'autres systèmes plus évolués qui tiennent compte d'une information temporelle pour adapter l'incrément. Par exemple, ces systèmes considèrent que plus l'utilisateur réitère son événement, plus il veut aller loin. Cela est mis en œuvre en appliquant un pas croissant en fonction du temps de manipulation de la position de l'utilisateur. Cependant, cela est parfois contradictoire, puisqu'il faudrait aussi pouvoir ralentir le mouvement lorsque l'utilisateur s'approche de son objectif pour lui permettre de choisir finement son point de vue.

De plus, certains types d'événements discrets n'ont pas de mode de répétition. C'est-à-dire que pour les répéter, il faut les réitérer. Par exemple, on peut configurer l'interface graphique d'un ordinateur pour ne pas « répéter » les événements claviers, de sorte que pour répéter la touche enfoncée, il faut la relâcher et l'enfoncer à nouveau. Même si ce n'est pas le mode de fonctionnement standard des interfaces clavier, il existe des périphériques (cf. wand) ou des modalités (cf. reconnaissance vocale) qui ne peuvent pas être configurés autrement.

Les problèmes décrits ci-dessus ne sont pas uniquement propres au contrôle des navigations. Ils s'appliquent aussi à toutes les interactions de type continu. Ainsi, de manière générale, il est difficile d'utiliser des événements de type discret pour simuler des événements continus.

Evénements continus pour un contrôle discret

A cause de l'ordinateur, tout champ de valeurs, même continu, est obligatoirement discrétisé. Ainsi, on peut dire finalement que les événements continus sont aussi discrets. Cependant, dans le cadre du contrôle des navigations virtuelles, ces événements continus

requièrent un pas variable. A l'inverse des événements discrets, ils permettent une grande marge de manœuvre modifiable par l'utilisateur en fonction du capteur utilisé.

Contrairement au mode précédent, il est plus facile de simuler un contrôle discret par un événement continu. En effet, dans ce cas, on utilise généralement un système de seuillage qui en fonction de la valeur renvoyée par le capteur, enclenchera ou n'enclenchera pas le contrôle. Cependant, c'est réduire le large champ des variations possible du capteur à deux états.

E Limitations

Cybersickness

Il existe, cependant, des problèmes physiologiques et comportementaux avec les environnements virtuels. Par exemple, il n'est pas rare que l'utilisateur qui visite un monde virtuel, soit sujet à des perturbations oculométriques, des désorientations, voire des maux de tête ou des nausées.

Le nom donné à ces maux, proposé par McCauley et Sharkey, est « cybersickness », en français « mal des mondes virtuels ». Ce n'est pas une maladie ou un état pathologique, mais plutôt une réponse normale à un stimulus inhabituel.

Le problème apparaît notamment lorsque l'utilisateur navigue dans le monde virtuel : l'oreille interne, qui contrôle le sens de l'équilibre de l'utilisateur, lui signale qu'il est immobile alors que ses yeux voient le contraire. Ainsi, il y a un conflit entre la sensation renvoyée par les yeux et celle renvoyée par le sens de l'équilibre (cf. système vestibulaire de l'oreille interne). C'est un trouble proche du mal de mer. Ce mal est renforcé par la sensation d'immersion dans le monde virtuel. Ainsi, devant un petit écran, cette sensation est assez rare, probablement du fait que l'utilisateur perçoit aussi le monde réel. Par contre, dans un environnement immersif avec une scène suffisamment grande pour « entourer » l'utilisateur, le cybersickness sera relativement fréquent.

[LaV00] fait le point sur les éléments morphologiques qui entrent en jeu dans le mécanisme du cybersickness ainsi que sur les théories actuelles sur les conséquences psychologiques du conflit sensoriel. De plus, l'auteur explique les facteurs agravants qui engendrent le cybersickness.

Latence

L'un des éléments qui pourrait participer au cybersickness est la latence dans l'affichage (le terme de latence est la traduction la plus proche du terme anglais *lag*). Cette latence introduit un délai entre une action dans le monde réel et son équivalent dans le monde virtuel. Ainsi l'utilisateur qui déplace un objet virtuel dans sa main peut voir un retard dans le déplacement de l'objet.

Les causes de cette latence ont plusieurs origines. En effet, dans la chaîne des traitements appliqués aux événements, peu de tâches sont parallélisables. Dans un processus

standard de simulation, il y a généralement 4 étapes successives à respecter :

- Acquisition des données : il s'agit de la latence intrinsèque du capteur. Il ne faut pas oublier que dès que l'on active des filtres anti-bruit, la latence du capteur augmente [8].
- Transmission vers le processus graphique : le temps de transmission d'un élément sur un port série est directement proportionnel à la fréquence de transmission sur ce port. De plus, le format des données est primordial : généralement, l'utilisateur aura le choix entre un format binaire et un format ASCII. Le premier est préférable au second car il est plus compact.
- Traitement graphique : c'est le délai sur lequel nous avons le moins de contrôle. En effet, il se base sur le temps de rendu qui est directement fonction du nombre de facettes, donc de la complexité de la scène. A ce niveau, il est intéressant de faire intervenir des algorithmes de gestion de scènes afin d'alléger ce processus.
- Fréquence de rafraîchissement d'écran : il faut aussi noter que la fréquence de rafraîchissement verticale introduit une latence. A 60 Hz, la latence introduite peut atteindre 16 ms. A 120 Hz, la latence sera inférieure à 8,4 ms.

Une latence nulle n'est pas envisageable, car il y aura toujours des délais de transmission ainsi que des traitements ayant de la latence. Par contre, il existe plusieurs solutions pour alléger les effets de la latence. Au-delà des réductions basées sur une utilisation optimale des systèmes, nous pouvons mettre en place des filtres prédictifs. Par exemple, avec des filtres de Kallmann on peut prévoir la position de l'utilisateur à l'instant d'affichage suivant. Cependant, cela sous-entend que nous ayons une idée précise de l'instant où cet affichage aura lieu. Or, principalement à cause de l'étape de traitement graphique, la latence n'est pas constante au cours du temps.

Ainsi, une des pistes de recherche envisageable serait de faire en sorte que la latence soit constante au cours du temps (aussi bien au sein d'une même session qu'au cours de plusieurs sessions différentes). Ceci semble par exemple nécessaire pour des expériences en sciences cognitives. Il arrive en effet que celles-ci produisent des résultats complètement différents bien qu'étant basées sur le même protocole expérimental. Ainsi, la différence de latence entre deux expériences est l'un des facteurs majeurs sur lequel les expérimentateurs n'ont aucun contrôle. En d'autres termes, il peut être pertinent de se poser la question : *Faut-il essayer de diminuer la latence au maximum, quitte à ce qu'elle ne soit pas constante ou faut-il plutôt essayer de faire en sorte qu'elle soit constante, à défaut d'être minimale ?*

F Interaction « naturelle »

Il est maintenant temps d'expliquer le qualificatif « naturelle » qui apparaît dans le titre de ma thèse. Ce terme a plusieurs acceptions. Celle qui m'intéresse le plus est : *Qui respecte les lois de la nature.* Elle est directement corrélée à la définition de sous-espace de représentation dans un environnement immersif. En effet, les lois de la nature requièrent la mise en place d'un espace isomorphe au réel.

8. à titre d'exemple, la documentation du Fastrack Pohlemus précise qu'en paramétrant les filtres avec des valeurs trop sélectives, on peut atteindre des latences intrinsèques de l'ordre de 162ms.

J'appelle interaction « naturelle » une interaction qui se rapproche du comportement naturel. Ainsi, par exemple, le fait de prendre une tasse virtuelle dans la main, par l'intermédiaire d'un système de reconnaissance gestuel associé à un gant numérique, pour la poser sur une table est naturel. Par contre, le fait de cliquer sur un bouton pour demander à ce que la tasse soit mise sur la table n'est pas naturel. En d'autres termes, une tâche dans le monde virtuel est considéré comme naturelle si la charge cognitive est de même nature que celle d'une tâche équivalente dans le monde réel. Je rejoins en cela la définition d'environnement immersif que j'ai choisie dans la section 2.3.

Cependant, il est pertinent de se poser la question de savoir s'il est vraiment important que l'interaction soit naturelle. En effet, nos actions sont qualifiées de « naturelles » car elles sont adaptées à l'environnement réel et à notre morphologie. Etant donné que les environnements immersifs permettent de rendre n'importe quel type de monde virtuel (voir section2.2, page15), il est maintenant possible de mettre en place des interactions moins contraintes par les lois physiques. Par exemple, dans l'espace réel tout est fini, exception faite de l'univers lui-même. Ainsi, la feuille de papier que vous tenez entre vos mains est finie (elle possède des frontières). Cela impose au texte qu'elle contient d'être structuré sous forme de lignes et de pages. Dans le domaine connexe à celui de la Réalité Virtuelle qu'est internet, le texte est toujours représenté sous forme de lignes. Cependant, l'idée de page au sens « unité verticale de texte ayant une dimension constante » a disparu. Ainsi, tout texte peut être représenté sous la forme d'une unique page dont la longueur est variable d'un document à un autre. Dans ce cas, nous avons gagné une dimension supplémentaire. De plus, dans certaines applications, comme les applications scientifiques, il n'y a pas forcément de comportement « naturel » à mettre en place, compte tenu des objets.

Avant de proposer à l'utilisateur de nouveaux paradigmes interactifs qui seraient contraires aux lois physiques, il faut lui proposer des interactions qu'il a l'habitude d'utiliser. De plus, si nous voulons étudier et évaluer la pertinence de nouveaux paradigmes, il faut pouvoir les comparer avec ceux déjà en place, c'est-à-dire à ceux du monde réel. Cela requiert la mise en place d'une grande partie des interactions naturelles, voire même de leur totalité.

2.5 Conclusion sur les définitions

Nous avons vu qu'il existait principalement deux types d'interactions : celles continues et celles discrètes. Nous avons aussi vu que chaque interaction doit être associée à un unique type d'événement. C'est pourquoi nous considérons que les périphériques matériels et les modalités sont interchangeables pourvu qu'ils fournissent le même type d'information. Par exemple, une commande vocale peut avantageusement remplacer une interaction clavier. Cependant, au niveau de l'interaction, il faut toujours garder à l'esprit qu'il peut y avoir des effets indésirables tels que le cybersickness lorsque l'on interagit avec le monde virtuel. De plus, même si l'intégration d'interactions « naturelles » peut être criticable vis-à-vis de la Réalité Virtuelle, il faut les prendre en compte, ne serait-ce que pour évaluer les

nouveaux paradigmes interactifs.

Nous avons observé que les mondes virtuels peuvent être totalement décorrélés en termes de lois mathématiques ou physiques par rapport à l'univers réel. Cela nous a amené à introduire la notion d'*espace de représentation* associé à un monde virtuel sur lequel s'appliquent effectivement les interactions dans un environnement immersif. Enfin, nous avons vu que l'interaction peut prendre plusieurs formes. Mais surtout, les périphériques ou modalités d'interaction d'un même type (discret ou continu) sont interchangeables pour une même action.

3 Buts et usages

Nous avons précédemment parlé des mondes virtuels et des interactions. Cependant, nous devons aussi parler des applications ainsi que des limitations de ces mondes. Il est communément admis que l'apparition de la RV&A est très liée à l'interaction sur des objets de mondes virtuels et donc à la panoplie des périphériques utilisés pour manipuler ces objets. Cependant, on peut aussi considérer que ces périphériques seraient restés impuissants sans l'émergence de systèmes de rendu des mondes virtuels. Au-delà des architectures et cartes graphiques temps réels, les outils de CAO et autres logiciels d'animation ont été des contributions majeures pour ces systèmes de rendu, comme le deviennent actuellement les outils logiciels de simulation audio ou haptiques. Les citations de cette section n'ont pas pour but d'être exhaustives. Elles servent simplement à illustrer mon propos.

3.1 Revue de projets

La revue de projets est une utilisation fréquente de la RV par les industriels : l'ensemble des corps de métiers concernés se réunit autour d'un prototype et étudie les détails de ce prototype virtuel afin de faciliter l'intégration du produit en cours de conception.

La Réalité Virtuelle permet d'éviter la construction de prototypes physiques préliminaires. En effet, un prototype virtuel permet de remplacer ces fabrications coûteuses, et ce d'autant que les délais de modification de la maquette virtuelle sont moindres. [DJL+00] décrit notamment HIVE (Human Integrating Virtual Environment), un système permettant une tâche de conception entre deux sites distants.

3.2 La Conception Assistée par Ordinateur(CAO)

A l'heure actuelle, la CAO utilise peu la Réalité Virtuelle. En effet, les concepteurs travaillent principalement sur papier. L'étape suivante de la conception d'un objet vise à prendre l'ensemble des plans et à les synthétiser dans l'ordinateur. Enfin, seulement, les concepteurs peuvent évaluer en Réalité Virtuelle la pertinence de leur conception. Ils travaillent ainsi sous la forme de revue de projet. Cependant, si une modification doit intervenir suite à la revue de projet, les concepteurs retournent à leur planche à dessin et le processus reprend. Ainsi, l'un des grands chantiers de recherche est d'introduire la Réalité Virtuelle au cours même du processus de conception des objets [Bou02, Con02, Gou01, BKG98].

3.3 La simulation

La simulation consiste à travailler, dans le monde virtuel, sur la modélisation d'un phénomène au sein de l'espace de représentation.

On peut simuler des objets en leurs donnant une géométrie et/ou des comportements propres. Mais, la simulation consiste surtout à étudier un objet en modifiant sa géométrie,

sa structure ou son comportement. C'est là, par exemple, que se situe l'un des chantiers de l'application de la RV à la conception d'objets.

On distingue deux types de simulation :

Simulation « hors ligne »

Dans cette simulation les caractéristiques de l'objet simulé ne peuvent être modifiées que par l'intermédiaire d'un processus qui doit être sous-traité à un module totalement dissocié de l'interaction qui l'engendre.

Dans ce type de simulation, le bouclage entre le traitement des modifications et la maquette résultante n'est pas en « temps interactif[9] ». En outre, ce bouclage s'appuie souvent sur une organisation du travail (cf. : concepteur/opérateur de saisie/ingénieurs).

Simulation « en ligne »

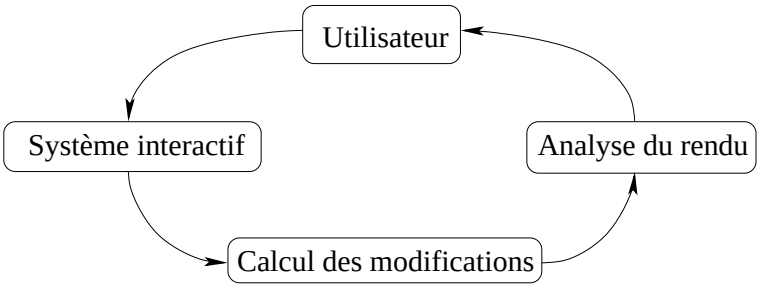

FIG. 1.3 – *Une boucle de simulation comporte quatre éléments fondamentaux.*

Une simulation en ligne est une simulation où le traitement des modifications est opéré en « temps interactif ». Ainsi, une demande de modification de comportement ou de géométrie de l'objet est prise directement en compte par le système pour régénérer un objet cohérent avec les nouvelles spécifications. L'utilisateur peut alors voir sans trop de délai son objet évoluer en fonction des contraintes qu'il lui applique.

Comme on peut le voir, dans le cas de la simulation « en ligne », il se crée une boucle où l'utilisateur joue le rôle du manipulateur. Cette boucle, décrite figure 1.3, est appelée *boucle de simulation*. Il est intéressant de constater que cette boucle de simulation est déjà présente dans la figure 1.2, page 18. La répartition des éléments dans la figure 1.2 est plus générique en ce sens qu'elle est indépendante de la tâche de simulation décrite figure 1.3.

9. On peut dire d'un système qu'il a une réponse en temps interactif si la latence perçue par l'utilisateur est tolérable au regard de la tâche qu'il effectue.

3.4 Usage scientifique de la RV

L'utilisation de la RV pour des applications scientifiques vise souvent à représenter et analyser des données complexes. Ces données peuvent avoir tout type de caractère. Elles peuvent être unidimensionnelles et enrichies d'informations tridimensionnelles (cf. ADN). Elles peuvent avoir cinq dimensions (cf. Mécanique des Fluides).

Ainsi, l'un des problèmes des applications scientifiques est d'avoir à représenter un monde de dimension non standard. C'est pourquoi, la première tâche pour le développement d'une application scientifique est de déterminer l'ensemble des données et son espace de représentation. Cependant, les applications scientifiques peuvent aussi avoir pour objectif des simulations « en ligne ». Par exemple, dans le cas de la Mécanique des Fluides, on peut imaginer que l'utilisateur « manipule » des sources de flux et/ou des obstacles dans le monde. L'ordinateur a ensuite la charge de représenter les déformations associées à ces déplacements. Ainsi, une des caractéristiques fortes de ces applications est leur possibilité de requérir une très grande puissance de calcul[10].

3.5 Usage thérapeutique de la RV

L'idée sous-jacente aux applications thérapeutiques [Riv00] est que, si les environnements immersifs sont aussi réalistes que leurs concepteurs le pensent, alors, il est possible de traiter en Réalité Virtuelle certaines pathologies comportementales telles que des phobies.

Par exemple, certains patients atteints de phobie arachnéenne peuvent être traités en les plaçant dans un monde virtuel contenant des araignées. L'intérêt majeur est que la scène phobique est entièrement contrôlable. Ainsi, dans une première phase du traitement, le monde virtuel permet une transition douce des images fixes vers des images d'arachnides en mouvement. Ensuite, dans une seconde phase, le monde virtuel permet d'avoir un arachnide complètement réactive, à comportement paramétrable en fonction de l'évolution du traitement contre la phobie.

Plus globalement, bon nombre d'objets peuvent être synthétisés dans le monde virtuel. Ainsi, l'environnement immersif permet de traiter la plupart des phobies, sans pour autant nécessiter les animaux ou objets associés à celles-ci. Cependant, les modèles synthétisés doivent être suffisamment réalistes pour donner l'illusion du réel.

3.6 Apprentissage par la RV

Une autre application intéressante de la Realité Virtuelle est celle relative à l'apprentissage. Dans beaucoup de corps de métiers relativement manuels, la pratique est importante. Cependant, dans certains domaines, il est impensable de faire des essais sur matériaux réels. Par exemple, dans le domaine de la maintenance aéronautique, certaines pièces très coûteuses ou fragiles ne peuvent être manipulées sans une grande expérience sous peine

10. Le lecteur pourra se référer à la section 5.1, page 151, pour avoir une évaluation de la puissance de calcul nécessaire à ce type d'application.

d'être irrémédiablement détériorées. De leur côté, les pilotes sont obligés de faire un certain nombre d'heures de vol en simulateur afin d'avoir l'habilitation de piloter des avions chargés de passagers. Ou encore, dans le cas de la chirurgie [HKW+98], la Réalité Virtuelle peut remplacer l'apprentissage sur des animaux ou des cadavres.

Dans ce cas, la Réalité Virtuelle permet de répéter des processus critiques, sans pour autant mettre en cause des vies humaines ou détériorer des matériaux. En fait, tout domaine qui requiert des manipulations délicates peut nécessiter un apprentissage sur simulateur dans un environnement immersif.

3.7 Assistance à la maintenance

Dans le domaine de la maintenance, les dépanneurs étaient précédemment obligés de transporter l'ensemble des documentations. Ensuite, les documents techniques ont été numérisés pour être contenus dans une base de données informatique. Cependant, dans le cas de montages complexes, il peut être difficile de localiser les pièces et les points de montage. Ainsi, l'ultime étape a été de faire porter au technicien une paire de lunettes immersive [WFM+96]. Ces lunettes affichent, en surimpression, les pièces et les points critiques d'une opération de maintenance.

3.8 Les jeux vidéo

Comme nous l'avons vu précédemment, la Réalité Virtuelle commence à l'interaction sur un monde virtuel. Donc, les jeux vidéo font partie de la RV&A puisqu'ils permettent à un utilisateur d'agir sur un monde numérique. C'est sans conteste le domaine qui a tiré vers le bas le coût de certains matériels de Réalité Virtuelle (i.e. les cartes graphiques). Ces systèmes sont souvent encore très primitifs. Les interactions sont généralement limitées au clavier et à la souris. Certains concepteurs proposent l'utilisation de manettes de jeu, tandis que quelques consoles de jeux y intègrent un retour sensori-moteur. Les mondes virtuels sont généralement composés de scènes architecturales ou urbaines, de landes désertiques ou de zones aériennes où le plaquage de textures est utilisé de façon intensive.

3.9 Conclusion

On a vu plusieurs applications de la Réalité Virtuelle, mais certaines sont cantonnées à un simple rendu alors que d'autres permettent l'interaction sur des objets virtuels. De plus, ces systèmes requièrent souvent une expertise de la part de l'utilisateur dans le domaine de la RV, tandis que d'autres plus « grand public » finissent, à force de pratique, par amener l'opérateur à acquérir ladite expertise.

4 Les différents types de scène

Les mondes virtuels peuvent être composés de plusieurs types d'objets. De plus, la structure des objets, qui ne correspond pas forcément à leur type, peut influencer le mode d'interaction que nous avons avec eux.

4.1 Types d'objets

Il existe deux systèmes de classification des objets : la métrique et la topologie.

A Topologie

Topologie connue

Les applications de Réalité Virtuelle actuelles sont, pour la plupart, composées d'objets topologiquement connus. C'est-à-dire que la représentation initiale des objets repose sur un ensemble de primitives « simples » telles que des surfaces planes ou paramétriques. C'est le cas, par exemple, d'une automobile ou d'une scène architecturale.

Topologie non connue

Par opposition aux objets topologiquement connus, il existe des objets dont la représentation initiale n'est pas composée de primitives géométriques. Ceci est, par exemple, le cas de nombreuses applications scientifiques, comme celles de Mécanique des Fluides. Cette représentation est souvent issue d'un monde non isomorphe avec une injection pour passer dans l'espace de représentation. Ainsi, la fonction de passage détruit généralement toute l'information topologique. Il existe un autre système qui fournit ce type d'objets. Les scanners 3d fournissent un nuage de points d'un objet réel, ensemble de points à partir duquel, les utilisateurs essayent de recréer la topologie de l'objet.

Dans de telles situations, l'espace de représentation correspond la plupart du temps à une collection de facettes dont les connexités ne sont pas a priori connues. De plus, la gestion de la granularité de l'approximation polyédrique dépend presque toujours du mode de génération (la fonction de passage du monde virtuel à l'espace de représentation ou la finesse de l'acquisition du scanner 3d). Il en résulte qu'il faut généralement appliquer des algorithmes complexes pour retrouver les connexités sur ces approximations polyédriques.

B Métrique

Au-delà de leurs informations topologiques, nous distinguons deux grandes classes d'objets : les objets « monolithiques » et les objets « caverneux ».

Objets monolithiques

Ce sont des objets dont la partie intérieure n'est pas « représentée ». Ainsi, une visite de l'intérieur n'est pas envisageable et/ou souhaitée.

Le type de navigation utilisé est une navigation centrée sur l'objet. Ainsi, il suffit d'un simple référentiel placé au centre de gravité de l'objet. Il ne reste qu'à trouver un moyen pour faire tourner l'objet autour de ce référentiel pour pouvoir étudier l'objet.

Dans ce contexte, l'un des systèmes de navigation le plus efficace est celui du *arc-ball* [Sho92].

Objets caverneux

Ces objets ont pour caractéristique d'être « visitables ». Autrement dit, leur structure interne est intéressante à étudier pour l'utilisateur. Il s'agit, par exemple, de scènes architecturales et urbaines.

Cependant, dans de telles structures, il y a généralement plusieurs points d'intérêt. C'est-à-dire qu'une visite associée à de tels objets ne peut pas être limitée à un référentiel placé au centre de gravité de l'objet.

De plus, le fait que l'intérieur de l'objet soit intéressant n'empêche pas que l'extérieur présente plusieurs points d'intérêt. Ainsi, l'interaction avec les objets caverneux peut, dans certains cas, être de même nature que celle utile aux objets monolithiques.

C Conclusion sur les objets

Nous avons vu que les objets à topologie non connue présentent des inconvénients. En particulier, l'interaction sur ce type d'objet est beaucoup plus complexe, à cause de l'approximation polyédrique d'une surface dont on ne connaît pas le formalisme mathématique. Par exemple, une détection de collision sur une surface de Bézier[11] est beaucoup plus rapide à déterminer si elle exploite l'algorithme de DeCasteljau que si elle a lieu sur une approximation polyédrique déconnectée des propriétés intrinsèques à ce type de surface[12].

En fait, un objet appartient forcément aux deux systèmes de classification. En effet, un objet à topologie connue peut être monolithique ou caverneux. Il en va de-même pour un objet dont la topologie est inconnue.

4.2 Structures de données volumiques

Une structure de données volumique est une représentation de la scène. Elle doit permettre d'implémenter les différents algorithmes propres à la synthèse graphique ainsi que ceux associés à la manipulation des objets.

Graphe de scène

Tous les objets virtuels ont un système de positionnement (référentiel) propre. Ce référentiel peut définir la position de l'objet par rapport à une autre (i.e. : la main par rapport au bras). Tous les objets sont : soit définis par rapport à un autre objet, soit par

11. Réalisable à partir d'une égalisation d'équations entre la trajectoire et la surface.
12. qui requiert une détection de collisions sur chacune des facettes de l'approximation.

rapport à l'origine du monde virtuel. Notons que cette description s'entend sans bouclage transitif : la représentation des positions de tous les objets est un arbre où chaque nœud représente un référentiel global à plusieurs objets, chaque branche est un changement de référentiel d'un nœud de l'arbre à un autre et chaque feuille est un objet monolithique. C'est la notion d'arbre de scène.

Dans certains cas, il peut être utile de représenter un objet plusieurs fois, comme par exemple des fauteuils dans une salle de concert. Ce concept d'instanciation induit une évolution de l'arbre initial vers une structure en graphe : on parle alors de *graphe de scène*.

La gestion de scène

La plupart du temps, les scènes se composent de beaucoup trop d'objets pour être rendues en temps réel. C'est-à-dire que même si les calculateurs graphiques ont une puissance croissante, les applications de Réalité Virtuelle requièrent toujours plus de puissance. Le premier but des algorithmes de gestion de scène vise donc l'optimisation de l'affichage.

C'est ainsi qu'il faut faire en sorte de simplifier la scène avant de l'afficher. Les simplifications peuvent être de plusieurs types (Algorithme de visibilité, centrée objet ou scène ou par simplification polygonale). Voir, par exemple [Bou02] pour un état de l'art sur le sujet.

4.3 Les scènes CAO

Dans le cas de scènes utilisées en Conception Assistée par Ordinateur (CAO), les objets sont toujours décrits par des objets géométriques simples. Deux modèles s'affrontent : ceux à base de voxels[13] et ceux à base de frontières. Ces deux modèles géométriques ont introduit chacun des algorithmes d'édition et de représentation. Requicha [Req80], présente un premier schéma récapitulant l'ensemble des modes de représentation et d'édition.

Pour les voxels, la méthode la plus employée est l'énumération spatiale. Ainsi, l'espace est découpé en un certain nombre de petits éléments volumiques (les voxels) et selon la présence ou l'absence de matière, le voxel changera d'état (vide, plein ou mixte). Au demeurant, la représentation la plus employée en CAO est le modèle de représentation par frontières ou « Boundary Representation » (B-Rep). Il s'agit d'un graphe qui permet de séparer l'intérieur de l'extérieur d'un objet en représentant les relations de connextié entre les éléments qui décrivent la métrique de sa surface. Ce dernier modèle peut être composé de faces planes, mais peut aussi contenir des surfaces courbes.

Au-delà des modèles de représentation, les outils de CAO proposent des modèles d'édition permettant de combiner plusieurs objets. En effet, un objet CAO est généralement une composition de plusieurs entités de base. Pour ce faire, ils peuvent utiliser des arbres de type Constructive Solid Geometry (CSG) qui permettent de composer des objets grâce aux différentes opérations topologiques possibles (intersection, union et soustraction). No-

13. Les voxels sont des unités volumiques (typiquement, des petits cubes) qui s'apparentent aux pixels de l'image.

tons qu'un arbre CSG est proche d'une arbre de scène, à ceci près que dans ces derniers, les objets sont par définition séparables.

4.4 Les scènes scientifiques

L'une des spécificités des scènes scientifiques est le caractère quasi aléatoire des amas de facettes obtenus. En effet, les scènes scientifiques sont souvent issues de mondes non isomorphes au réel qui requièrent donc une réduction de dimensions. Ainsi, nous avons généralement des objets à topologie non connue. C'est pourquoi les algorithmes conventionnels de gestion de scènes ne sont pas forcément efficaces pour optimiser les temps de rendu.

4.5 Les scènes composites

Il existe des scènes qui peuvent contenir plusieurs types d'objets ou types de sous-scènes relatifs aux différents types décrits ci-dessus. La plupart du temps, elles sont structurées selon un graphe de scène. Dans ce cas, la navigation doit être suffisamment élaborée pour permettre un déplacement à « grande distance » ainsi qu'un mouvement de faible amplitude : on parle alors de scène à granularité variable.

4.6 Conclusion sur les différents types de scènes

Les interactions qui m'ont servi de référence dans mes travaux sont celles qui sont applicables à l'ensemble des ces différentes scènes. Tous les modes d'interaction que nous allons décrire dans les sections suivantes de ce chapitre, ainsi que toute la complexité de la représentation propre aux différents objets ou différentes scènes doivent être pris en compte.

5 De l'Humain vers le Monde Virtuel

Dans les dispositifs immersifs, il existe plusieurs types de périphériques pour obtenir des informations sur l'état et l'activité de l'utilisateur (configuration géométrique, actions sur l'application et sur le monde virtuel). On peut les classer dans deux catégories : les systèmes basiques et ceux dotés de traitements évolués (cf. reconnaissance...).

5.1 Systèmes « basiques »

Ces systèmes sont en fait entièrement basés sur le calcul déterministe de valeurs issues de capteurs spécifiques.

Capteurs à 6 degrés de liberté avec récepteurs « embarqués »

Pour obtenir une information de position et d'orientation avec 6 degrés de liberté, il existe deux technologies de base, les récepteurs électromagnétiques et ceux à ultrasons.

Les systèmes électromagnétiques ont été les premiers utilisés. Ils sont basés sur l'émission d'un champ électromagnétique par une antenne. En fonction de l'intensité selon chaque axe et la phase du champ perçu par un capteur, il est en effet possible de déterminer la position et l'orientation de ce capteur dans l'espace. Cependant, le système est obligé d'alterner le champ d'une direction canonique à une autre en passant par les trois axes. Cela pose un problème de fréquence lors de l'utilisation de plusieurs capteurs où chaque capteur doit pouvoir traiter son propre triplet de champs à une cadence suffisante.

Des systèmes plus récents sont basés sur l'émission d'ultrasons. Ces systèmes se servent de la triangulation pour déterminer la position de chaque capteur. Une simple triangulation ne permettant pas d'obtenir l'orientation du capteur, il faut multiplier les microphones ultrasons sur le capteur. En effet, pour avoir une bonne précision sur la position, il suffit d'écarter les émetteurs fixes. Cependant, la précision de l'orientation dépend directement de la discrimination entre les récepteurs. Cette discrimination est directement fonction de leur écartement. Mais la taille du capteur doit être limitée pour ne pas gêner l'utilisateur. Ainsi, les constructeurs sont obligés de rajouter une centrale inertielle au système de capture afin d'améliorer la précision sur l'orientation tout en gardant une taille raisonnable pour l'ergonomie du dispositif de capture. Dans les faits, la centrale inertielle calcule aussi l'accélération, permettant ainsi d'améliorer la précision en corrélant plusieurs sources d'information : une prédiction de la position future avec les centrales inertielles et l'information renvoyée par les capteurs ultrason. Jusqu'à très récemment, cette technologie avait un poids non négligeable à cause de la centrale inertielle.

L'un des plus importants problèmes de ces deux technologies est celui des réflexions parasites.

En effet, pour les systèmes électromagnétiques, les réflexions sont causées par des composants ferromagnétiques qui agissent comme des courants de Foucault générés par les capteurs. Ainsi, il suffit de rajouter un matériel magnétique entre l'antenne émettrice et

le capteur pour que les données soient faussées. En fait, il n'existe aucune autre solution que celle d'imposer des composants non magnétiques dans l'espace d'interaction.

Concernant les systèmes à ultrasons, les réflexions proviennent des surfaces lisses. Les réflexions seront d'autant plus importantes que le matériau est rigide. Le problème des réflexions parasites sur les écrans a été contourné par la multiplication des émetteurs et des récepteurs. Ce qui fait que là où trois microphones suffisent théoriquement pour trianguler une position, ces capteurs utilisent jusqu'à une dizaine de microphones.

Les algorithmes associés sont généralement parallélisables. Ainsi, ces dernières années ont vu l'apparition de systèmes complets de capture de mouvements : « motion capture ». Ces systèmes permettent de capter simultanément plusieurs positions et orientations dans un même espace en se basant sur plusieurs récepteurs. Ils sont souvent utilisés dans le domaine du sport, mais aussi dans celui des jeux vidéo pour créer des corpus de mouvements réalistes qui sont ensuite appliqués à des personnages de synthèse.

Capteurs à 6 degrés de liberté par caméra infrarouge

La technologie infrarouge permet de ne pas avoir de récepteurs « embarqués ». En effet, certains systèmes de capture par caméra infrarouge sont basés sur la réflexion des ondes infrarouges par des matériaux spécifiques. En plaçant des boules de ces matériaux sur l'objet à traquer, on peut obtenir sa position dans le champ de vision des caméras. Pour ce faire, on détermine la position du marqueur dans l'espace par triangulation en connaissant la position des caméra les unes par rapport aux autres. De plus, en plaçant plusieurs boules sur un sous-objet rigide et indéformable, on peut obtenir l'orientation de ce dernier.

Au-delà du problème de réflexion sur des miroirs ou sur des objets parasites, il y a le problème de l'occultation. Imaginons que nous mettons un réflecteur dans la paume de la main et que nous fermions le poing : dans ce cas, le système sera incapable de déterminer la position du capteur.

De plus, les boules doivent être relativement grosses, afin de laisser une trace visible sur les capteurs des caméras quelle que soit la distance entre ces boules et lesdites caméras. Or, cela empêche de mettre plusieurs sphères sur une petite surface. Qui plus est, le fait d'obtenir un point ne suffit pas pour déterminer un référentiel 6 degrés de liberté (6DDL). Le nombre minimum de capteurs pour obtenir un référentiel 6DDL est de 3. Ainsi, il faut multiplier les réflecteurs afin de caractériser complètement l'orientation.

Pour palier ces deux gros défauts, ce type de dispositif multiplie généralement les caméras. Une autre approche, qui fait l'objet des travaux de thèse d'*Olivier Magneau*, doctorant au sein LIMSI-CNRS, est d'utiliser de façon optimale et contrôlée le positionnement des capteurs sur les objets à traquer [MBG02b, MBG02a].

Ces recherches sont d'autant plus pertinentes que l'infrarouge possède un énorme intérêt par rapport aux deux technologies précédentes. De par la manière dont cette technologie est utilisée, le système embarqué est totalement passif et la transmission du signal ne requiert aucun médium particulier. Ainsi, il n'y a aucun cordon ombilical entre le

réflecteur et le système de traitement. Contrairement aux autres systèmes, il n'est pas nécessaire de mettre en place un système de transmission sans fils. La seule limitation à la multiplication des réflecteurs, d'un coût très faible, est la puissance de calcul nécessaire au suivi de l'ensemble des capteurs et au traitement du signal issu des caméras.

Il existe aussi le système Optotrak qui utilise des diodes lumineuses. Contrairement aux systèmes décrit précédemment, l'Optotrak requiert des éléments embarqués actifs. De plus, ce système acquiert les données en temps réel.

Autres systèmes de capture de position

Il existe d'autres systèmes permettant à l'ordinateur d'obtenir la position de l'utilisateur. Par exemple, la société Boeing a implémenté un système qui perçoit les influx nerveux au travers de la peau et qui arrive ainsi à déterminer le déplacement du bras sur lequel il est appliqué.

Un autre système utilisé est celui du bras articulé. Ainsi, en recomposant la succession des rotations et translations subie par le bras articulé, on peut déterminer avec beaucoup de précision la position de l'extrémité (en fait, ces systèmes sont ceux qui ont la plus grande précision). Ce bras peut être utilisé pour supporter un visiocasque, tel que sur le Boom^{TM}.

D'autres systèmes se basent sur une centrale inertielle pour obtenir l'orientation des objets. En fait, contrairement au système ultrason, les systèmes inertiels ne fournissent qu'une information sur l'orientation et l'accéleration.

Configuration de la main

La solution consiste à mettre des capteurs qui mesurent la flexion de chaque articulation. Là encore, il y a deux technologies.

Les systèmes basés sur des fibres optiques sont les plus anciens. Une lumière est émise à une extrémité de la fibre et une cellule photosensible capte la lumière reçue à l'autre bout. Ainsi, on peut connaître la déformation de la fibre en fonction de l'intensité absorbée par la fibre. Cette déformation est directement dépendante de la flexion des articulations du doigt sur lequel la fibre est fixée.

Les systèmes à base de jauges de contraintes fixées sur chaque doigt sont actuellement les plus fiables. En effet, les fibres optiques s'usent et ont un comportement non homogène avec le temps.

Un autre problème est la morphologie de la main des utilisateurs. Les élongations des différents capteurs de la main ne suffisent pas à définir la géométrie exacte de la main : ces valeurs ne sont pas absolues et nous n'en connaissons pas la métrique de référence. On est donc obligé de faire un calibrage du gant avant chaque utilisation. Cette calibration permet de connaître la relation qui existe entre l'étirement de chaque capteur et les angles des articulations associées. Le système de reconnaissance est en effet plus robuste au changement d'utilisateur lorsque nous travaillons sur les angles plutôt que sur l'étirement.

Ainsi, deux personnes ayant des mains de taille différentes peuvent travailler avec la même base d'apprentissage.

Widget 2D et 3D

Initialement conçus par les laboratoires de *Xerox* à *Palo Alto*, les menus et la métaphore de bureau sont les concepts d'interaction les plus répandus à l'heure actuelle sur nos ordinateurs. Comme ces concepts sont maintenant devenus universels, beaucoup d'environnements immersifs se contentent de les reprendre en les plaçant dans un espace à trois dimensions. Par exemple, le plan des menus a été redressé pour apparaître comme un plan virtuel. D'autres ajoutent des décorations telles que des boutons ayant une profondeur, des textes en relief ... Cependant, conceptuellement, nous sommes toujours en deux dimensions.

Hormis quelques paradigmes, comme [GC01] qui propose de placer un menu dans une sorte de « rubic's-cube », aucun système ne fait preuve d'originalité pour remplacer les menus bidimensionnels. Une explication peut être l'impossibilité physiologique qu'a l'homme de focaliser simultanément sa vue sur plusieurs plans de profondeur. En l'espèce, l'utilisateur doit pouvoir se focaliser sur le plan concerné pour effectuer une tâche précise comme la sélection de valeurs, de données ou d'un état via un « clic ». Pour ce type d'interaction, l'idéal est de pouvoir gommer les autres plans pour éviter de parasiter l'affichage. En conséquence, si l'on désire permettre une interaction multi-plan, il faut que l'ordinateur soit capable de déterminer le plan d'intérêt de l'utilisateur.

En fait, l'affichage de ces menus est très gênant, car il oblige l'utilisateur à changer de contexte visuel pour passer de l'objet 3d aux menus 2d et réciproquement. Cette permutation de contexte peut augmenter la charge cognitive de l'utilisateur en particulier lorsque son activité porte sur des scènes complexes et qu'elle met en œuvre un grand nombre de menus. De plus, les capteurs de position ont une précision relativement limitée et ils introduisent des vibrations qu'il faut filtrer avant de déterminer la direction du pointeur.

5.2 Systèmes à base de reconnaissance

Les systèmes de reconnaissance sont des sur-couches à des périphériques de plus bas niveau. Il existe principalement deux types de systèmes de reconnaissance. Les systèmes à base d'apprentissage « générique » et ceux à base d'apprentissage « spécifique ».

Voix

Les deux types de reconnaissance existent pour la voix.

La reconnaissance dite spécifique s'apparente au multi-locuteur. Le vocabulaire doit être déterminé au préalable de manière précise, afin qu'un échantillon de personnes aux caractéristiques vocales différentes permette au système de savoir comment interpréter le signal de n'importe quel utilisateur.

D'un autre côté, quand la reconnaissance de la parole est générique, cela revient à ce que le système puisse reconnaître un texte quelconque. Cependant, seul l'utilisateur ayant fait l'apprentissage et quelques personnes ayant des caractéristiques vocales similaires seront reconnues par le système. Par contre, le vocabulaire pourra changer au fil du temps, sans avoir à refaire un nouvel apprentissage.

Geste

Il y a deux méthodes pour intégrer le geste dans une application. Une première consiste, par exemple pour le geste de préhension, à directement calculer l'intersection entre chaque objet de la scène et l'extrémité de chaque doigt.

Cependant, au-delà de ces interactions « en contact », il est nécessaire de disposer d'algorithmes de reconnaissance. Ces algorithmes sont généralement plus efficaces mais permettent surtout de décrire des gestes dans l'espace, sans contact direct avec les objets. Par exemple, une fois qu'un geste de désignation a été reconnu et que l'objet concerné a été identifié, la reconnaissance de geste peut servir à manipuler « à distance » les objets.

Le système de reconnaissance de gestes dont nous nous servons actuellement a été développé par A. Braffort et B. Bossard du groupe Geste et Image du LIMSI et s'appuie sur l'algorithme de *Rubine* [Rub91]. Il se base sur une étude statistique des angles de chaque articulation.

Notons que la mise au point d'un tel système générique de reconnaissance n'est pas encore d'actualité. Ceci requiert la mise au point d'un « outil » de description du langage utilisé. En effet, on est incapable, à ce jour, de faire un apprentissage générique avec un descripteur de langage modifiable. Il est intéressant d'observer cependant que cet apprentissage est proche d'un système multi-signeur. Par exemple, j'utilise actuellement un jeu de paramètres qui n'est pas issu d'un apprentissage sur mes propres gestes, mais sur ceux d'une tierce personne.

Caméra

La capture par caméras dans le domaine visible pose des problèmes supplémentaires par rapport aux systèmes avec des réflecteurs infrarouges. Il est impossible, contrairement à l'infrarouge, d'« éclairer » abondamment la scène afin de faire ressortir les points d'intérêt. En effet les environnements immersifs sont généralement relativement sombres [14].

A cela s'ajoutent des problèmes d'analyse d'image inhérents à cette approche. Avec l'infrarouge, nous pouvons faire en sorte de sélectionner les matériaux pour éviter les matériaux sensibles aux infrarouges à la fréquence voulue. Le corps humain et les vêtements étant par nature non sensibles à ces fréquences. Ainsi, le travail d'analyse d'image associé à la capture par infrarouge consiste principalement à :

– rechercher les pics d'infrarouge dans les images (les boules réflectrices) ;

14. Cela est principalement dû à la technologie utilisée (cf. note de bas de page 16, page 44)

– faire le suivi des trajectoires pour dissocier l'ensemble des réflecteurs visibles par la caméra ;
– mettre en correspondance 3d toutes les caméras.

Dans le cas de la lumière visible, il faut arriver à définir la position de points clefs sur l'utilisateur. Or, il est difficilement envisageable d'utiliser des marqueurs spatiaux, étant donné que nous ne pouvons pas obliger l'utilisateur à ne pas porter les couleurs associées aux marqueurs. En d'autres termes, la lumière visible impose une analyse complète de l'image et non de pics d'intensité. Cela passe par une détection des contours des corps dans l'image, détection qui à elle seule est à l'origine de nombreux travaux en analyse d'image. De plus, il faut arriver à recréer la configuration tridimensionnelle du corps humain par fusion de l'information issue de toutes les caméras. A tout cela, s'ajoute le problème des réflexions parasites sur les écrans, les miroirs, les parties métalliques ou les lunettes.

Enfin, il est important de noter que les caméras du commerce sont prévues pour fonctionner à 25Hz. Si cette fréquence est suffisante pour la sensation de mouvement visuel par le cerveau, elle est insuffisante pour servir à la capture des mouvements en RV&A. Par exemple, comme nous le verrons plus loin (section C), il faut un rafraîchissement de la position de l'utilisateur de l'ordre de 100Hz pour pouvoir proposer un système optimal de RV avec audio 3d.

Latence des systèmes de reconnaissance

Les systèmes de reconnaissance posent au demeurant quelques problèmes pour les interactions temps réel. En effet, ces systèmes ne peuvent pas garantir la reconnaissance tant que la fin d'un processus de reconnaissance n'a pas eu lieu.

Plus globalement, dans la plupart des systèmes de reconnaissance, il existe une incertitude sur la date de début et sur la date de fin de chaque reconnaissance. Par exemple, un mot prononcé peut n'être que le premier mot d'une phrase. Dans ce cas, la reconnaissance doit attendre un certain temps après la fin de la prononciation du mot pour être sûr que ce mot ne fera pas partie d'une commande plus longue. Ainsi, même si le système de reconnaissance est capable de déterminer avec exactitude l'instant de fin, il est obligé d'attendre un certain temps après la fin pour être capable d'annoncer avec certitude l'information.

5.3 Problème de saisie des données alphanumériques

Dans le cadre de certaines applications comme la Conception Assistée par Ordinateur (CAO), il existe un problème pour dimensionner les objets dans des dispositifs immersifs. Cette tâche fondamentale consiste à donner aux pièces à synthétiser les dimensions voulues.

Le problème est que l'incertitude des capteurs de position et de mouvement combinée avec la faible stabilité et précision de la main limitent très fortement l'usage des interactions gestuelles pour définir avec exactitude des cotations.

Plusieurs méthodes ont été proposées : Doug Bowman et al. utilisent un *pinch-glove*[15]

15. Un cyberglove équipé de capteurs pour déterminer s'il y a contact entre les extrémités de deux doigts.

pour simuler un clavier [BWCL01].

Une autre méthode en cours d'investigation au LIMSI-CNRS est celle de donner aux utilisateurs la possibilité d'interagir via un assistant personnel (PDA : *Palm* ou autres). Ce type d'équipement dispose d'un système de saisie des données (par exemple : reconnaissance d'écriture ou, de manière moins répandue, clavier intégré) couplé à un système d'échange de données avec un ordinateur fixe. Certains proposent des protocoles de connexions sans fils permettant des échanges « grande distance » (le protocole *BlueTooth* peut atteindre 100 mètres). L'idée est donc de récupérer les commandes, les paramètres et le texte saisie sur le PDA pour impacter de façon précise sur le monde virtuel.

5.4 Fusion multimodale

Dans le cadre de l'interaction de tous les jours, les humains combinent plusieurs modalités. L'exemple typique est celui du paradigme « pose ça ici » associé à deux sélections (voir la thèse de Yacine Bellik [Bel95], mais aussi [NC93, OCW+00]).

Dans ce cas, il faut que le système soit capable de corréler l'information issue de plusieurs modalités. Sur cet exemple, il doit être capable de prendre un objet au moment de la première désignation vocale (« ça »), puis de le relâcher lors de la seconde désignation vocale (« ici ») : on parle dans ce cas de co-référence.

Parmi tous les critères pour déterminer si deux événements doivent être fusionnés, il y a celui de la *proximité temporelle*. Ce critère vise à ne pas fusionner deux événements qui ont lieu à des intervalles temporels trop éloignés.

C'est ici que les problèmes de latence des systèmes de reconnaissance deviennent critiques. En effet, pour être sûr de définir le résultat d'une fusion, il faut attendre que tous les événements potentiellement intéressants soient arrivés. En d'autres termes, le temps de latence du système de fusion multimodale correspond à la latence du système de reconnaissance le plus lent majoré par le temps de transmission des événements. Cela peut atteindre plusieurs dixièmes de seconde.

5.5 Conclusion

Au-delà des différents types de capteurs qui permettent de suivre les mouvements de l'utilisateur, une approche importante à prendre en compte pour la gestion des interactions de l'humain sur le monde virtuel, est celle qui se propose d'utiliser des systèmes à base de reconnaissance. Si ces systèmes permettent d'améliorer le réalisme des interactions de l'homme sur le monde virtuel, ils introduisent des latences qui peuvent être préjudiciables au traitement multimodal qu'il convient aussi de mettre en œuvre.

Notons qu'à l'heure actuelle nous sommes encore limités, en situation immersive, pour gérer de façon efficace la saisie des données alphanumériques, interactions au demeurant particulièrement fréquentes en CAO ou pour certaines applications scientifiques.

6 Du Monde Virtuel vers l'Humain

En contrepartie des périphériques qui permettent à l'utilisateur d'interagir sur le monde virtuel, il faut aussi que les retours sensori-moteurs vers l'utilisateur soient les plus réalistes possibles.

6.1 Modalité Visuelle

Le retour visuel est le premier auquel nous pensons lorsque nous parlons de dispositifs immersifs. Cela est dû au fait que la vue est le sens dont l'être humain se sert le plus.

A Changement dimensionnel

Nous avons vu à la section 2.2 que le monde virtuel pouvait avoir un nombre de dimensions supérieur à trois, voire de valeur non entière. Cependant, le canal visuel est limité à deux dimensions plus une sensation de profondeur et de mouvement (ce qui ajoute une dimension supplémentaire). En d'autres termes, l'utilisateur ne peut percevoir que quatre dimensions par la modalité visuelle. Dans les faits, la plupart des mondes virtuels utilisés ont une composante temporelle. Comme cette dernière correspond parfaitement (moyennant une dilatation ou une contraction du temps) au monde réel je considérerai cette composante comme implicite dans la suite de cette sous-section.

Ainsi, le problème majeur du rendu tridimensionnel d'un monde virtuel est de trouver un algorithme permettant de passer de ce monde vers un monde isomorphe au réel.

La conversion la plus simple correspond à la transformation d'un monde de dimension inférieur au monde cartésien. La plupart du temps, cette conversion est faite par l'intermédiaire d'un « plaquage » de dimension N sur l'espace ou un sous-espace du monde isomorphe au réel.

Pour transformer un monde de dimension supérieur à un monde cartésien, il faut faire appel à des algorithmes plus évolués. Parmi les algorithmes les plus utilisés, notamment en Mécanique des Fluides, il y a ceux qui permettent des représentations à base d'iso-surfaces. On affiche dans l'espace une surface qui sépare l'espace des valeurs d'une quatrième dimension (la composante temporelle étant implicitement la cinquième) en valeurs inférieures et valeurs supérieures à une valeur de référence. Cette méthode est particulièrement appropriée à des champs de valeurs à variation continue.

Une autre méthode est celle de la texture 3d. Elle consiste à considérer chaque voxel du monde virtuel et à lui affecter une couleur en fonction de la couleur du ou des champs associés. Ensuite, nous affectons une valeur de transparence à l'ensemble des voxels du monde virtuel. Il suffit alors d'afficher l'ensemble des voxels. Cette solution permet un affichage de beaucoup plus d'informations. Cependant, elle est moins utilisée, car elle requiert un apprentissage de l'utilisateur, puisque ladite transparence peut avoir différents sens. Par exemple, quand cette solution sert à représenter les orbitales atomiques d'un atome ou d'une molécule, la transparence représente la probabilité d'existence d'un électron à un endroit donné et la couleur, son vecteur vitesse.

B Le relief

Le relief est le fait pour l'utilisateur de percevoir la profondeur des objets dans la scène. L'utilisateur peut percevoir l'information de profondeur selon plusieurs modes.

Focalisation

Il faut noter que l'œil est un système optique complexe. En particulier le cristallin est une lentille que le cerveau contrôle pour adapter la distance focale (par tension ou relâchement des bords du cristallin). Ainsi, le cerveau accommode l'œil sur un point d'intérêt par une action sur la forme du cristallin. Grâce au retour de l'information de déformation du cristallin, l'homme perçoit la profondeur des objets et donc dispose d'une sensation monoculaire du relief.

Binocularité

La sensation de relief résulte aussi du fait que nos deux yeux perçoivent la même scène sous deux points de vue légèrement décalés.

La stéréoscopie est le principe de rendu qui utilise la binocularité humaine pour donner la sensation du relief à un utilisateur. Ce principe consiste à séparer les canaux visuels associés à chaque œil.

Cependant, la projection sur un plan d'un dispositif immersif dépend du point de vue de l'utilisateur (cf. figure 1.4). Ainsi, les systèmes de Réalité Virtuelle doivent connaître la position de l'utilisateur à chaque pas temporel, afin de pouvoir calculer un relief exact. Ceci implique que la scène doit être recalculée en permanence afin de tenir compte de la position de l'utilisateur.

Il est intéressant de constater que les personnes ayant une faiblesse à un œil perçoivent en particulier le relief via l'information de focalisation du cristallin.

Mouvement

Au demeurant, tourner autour d'un objet permet aussi à un utilisateur de percevoir la profondeur de chaque partie de l'objet.

Le principe de l'alioscopie [Fau01, Ali] se base justement sur le mouvement. Il s'agit de retarder l'image sur un des deux yeux grâce à des filtres adéquats. Ainsi, lors des mouvements de « travelling » au cinéma, on peut recréer le relief. L'intérêt de cette approche est qu'il n'est pas nécessaire de faire les prises de vue avec des appareils stéréoscopiques. De même, le rendu se fait sur un simple écran, sans équipements spéciaux, à tel point que l'on peut s'en servir dans toutes les salles de cinéma. Cependant, ce type de visualisation en relief n'est pas possible sans travelling.

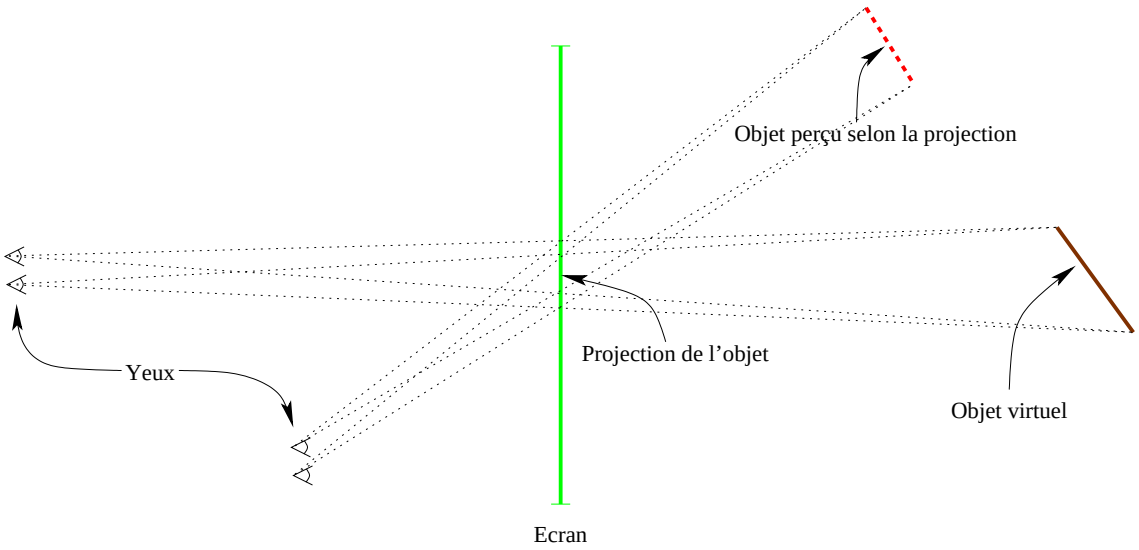

FIG. 1.4 – *La projection, pour un utilisateur juste en face de l'écran, est exacte vis-à-vis de l'objet en traits pleins. Par contre, lorsque l'utilisateur est décalé, l'objet virtuel est faux.*

C Les périphériques

Il existe principalement deux grandes classes de périphériques permettant de percevoir le relief par stéréoscopie.

Casques immersifs

Ce sont historiquement les premiers dispositifs immersifs à caractère stéréoscopique. Il existe des évolutions comme les lunettes see-through (« voir au travers ») qui permettent de superposer le monde virtuel sur le monde réel.

L'intérêt majeur des casques est que l'utilisateur est complètement immergé dans le monde virtuel sans aucune référence au monde extérieur. Il permet aussi pour chaque utilisateur d'avoir son propre canal visuel.

La contrepartie de l'isolation visuelle complète de l'utilisateur est qu'il n'est pas conscient des limites imposées par l'espace réel. Ainsi, pour les interactions avec d'autres utilisateurs, il faut nécessairement introduire des avatars. Un autre point à noter est que les casques sont peu propices aux revues de projets avec plusieurs utilisateurs.

Il y a quelques années, la technologie ne permettant pas de créer des écrans embarqués de grande définition, les seuls systèmes qui offraient une bonne qualité visuelle étaient les systèmes de type BOOM^{TM}. Outre que ces périphériques sont maintenant dépassés, leur inconvénient majeur est qu'ils doivent être portés par un bras articulé à cause de leur poids non négligeable.

Au delà des problèmes de définition, les casques immersifs ont généralement un champ

visuel réduit.

Ecriture rétinienne Les dispositifs nommés Virtual Retinal Display(VRD) permettent au système d'écrire directement l'information sur la rétine à l'aide d'un laser déviable dans un plan. Certains permettent aujourd'hui des produire des scènes colorés (cf. HIT lab, Seattle). [CSF02] propose notamment d'intégrer un tel dispositif, dans le cadre de la Réalité Augmentée, à un système de capture infra-rouge pour obtenir un système de superposition d'objets virtuelles sur le monde réel.

Grands dispositifs

Les dispositifs immersifs de type murs ont été pensés et développés à l'Electronic Visualisation Laboratory (EVL) par *Carolina Cruz Neira* [CNDD93]. Dans cette catégorie, nous classons tout type de dispositif avec écran fixe. Deux grandes catégories s'affrontent.

Les écrans à projection directe sont particulièrement appropriés pour les revues de projets. De plus, ils peuvent être installés dans des espaces relativement étroits, car l'espace nécessaire à la projection se confond avec celui de l'interaction.

Bien que les systèmes rétro-projetés requièrent plus d'espace pour la projection, ce type de dispositif est beaucoup plus riche en termes d'interaction. En effet, dans les systèmes projetés, l'utilisateur peut créer une ombre portée sur l'écran, ce qui empêche une interaction proche de l'écran et réduit la sensation d'immersion visuelle.

De plus, les murs rétro-projetés peuvent aller de la simple station de travail jusqu'à la CAVE complète avec 6 faces voire jusqu'à des systèmes reconfigurables (cf. RAVE, MOVE) en passant par les tables de travail immersives (Benchs).

D Technologies stéréoscopiques

Parmi les points importants pour les dispositifs à base de murs, il y a la stéréoscopie. Plusieurs technologies existent.

Stéréoscopie active

Cette technologie utilise le balayage de l'écran. Le but est d'afficher l'image correspondant à un des deux yeux, pendant qu'un dispositif occulte la vision de l'autre œil. Ce système requiert une synchronisation avec le calculateur graphique.

La stéréoscopie active est celle utilisée dans la plupart des dispositifs de type murs.

Un des problèmes majeurs de ce genre de technologie est que les seuls projecteurs capables à ce jour d'atteindre des fréquences très élevées sont des projecteurs tri-tubes[16] ou CRT. Les fréquences requises sont de l'ordre de 100Hz. Cependant, il faut aussi qu'ils aient une décroissance suffisante de la luminance des pixels. En effet, il ne faut pas que l'éclairage d'un pixel nécessaire à l'un des yeux persiste pour l'autre œil alors qu'il devrait être éteint.

16. Un tri-tube est un projecteur composé de trois tubes cathodiques (un rouge, un vert et un bleu). De plus, cette technologie est la seule à permettre de très grandes définitions (3000x2500).

Cela a été un problème pendant très longtemps notamment pour le tube vert. Le défaut de ce type de projecteur est qu'ils sont très faiblement puissants en termes d'intensité lumineuse (environ 300 Lumens). Cela impose de travailler dans une salle obscure.

Il existe une deuxième sorte de projecteurs qui est basée sur une technologie émergeante : les micro-miroirs, dite DLP ou DMD. Cela consiste en une puce qui gère chaque élément d'un maillage de petits miroirs reflétant ou non la lumière dans la direction de l'écran en fonction d'un signal électrique. En fait, la seule limite de puissance lumineuse de ce système est celle de l'absorption calorifique de la paroi qui reçoit la lumière lorsque le faisceau est dévié hors de l'écran. La définition maximale de ces puces est de 1280x1024 pixels. La technologie ne permet pas d'augmenter la définition de ces puces. Cela est très peu comparé à une technologie comme les LCD qui peuvent atteindre les 1600x1200 pixels. Cette technologie reste très onéreuse[17] mais risque d'évoluer dans les prochaines années en fonction de son impact dans le milieu du cinéma. Les technologies DLP permettent d'atteindre des fréquences adéquat pour la stéréoscopie active (les sociétés Christies et Barco proposent des DLP permettant du 1280x1024 à 100Hz). Cependant, les DLP ne permettent pas la flexibilité des projecteurs tri-tubes[18], car le signal est obligatoirement inscrit dans une grille régulière de 1280 par 1024 pixels.

Stéréoscopie par polarisation circulaire

Afin de dissocier l'image destinée à l'œil droit par rapport à celle de l'œil gauche, la stéréoscopie polarisée circulairement utilise deux projecteurs avec des filtres. Pour ce faire, il suffit d'envoyer l'image associée à chaque œil sur le bon projecteur. L'utilisateur, équipé de lunettes avec les filtres associés, peut percevoir la stéréoscopie. C'est d'autant plus intéressant que les lunettes sont très légères et peu onéreuses comparativement à celles dédiées à la stéréoscopie active.

Cette technologie se développe beaucoup actuellement. Cependant, la qualité n'est pas encore parfaite.

Dans la pratique, tous les projecteurs permettent de faire ce type de stéréoscopie. Cependant, les projecteurs LCD sont généralement préférés aux projecteurs tri-tube (du fait du coût de ces derniers et de leur faible intensité lumineuse). De plus, un mécanisme intéressant permet d'optimiser l'utilisation des projecteurs LCD. En effet, ils sont composés d'une source de lumière blanche. A la sortie de cette source, un filtre polarise la lumière. Ainsi, chaque pixel est géré par un filtre polarisant réactif à un champ électromagnétique. C'est pourquoi la lumière qui sort d'un projecteur LCD est naturellement polarisée. L'optimisation consiste à faire en sorte que la polarisation de la lumière corresponde parfaitement à celle que nous attendons en sortie pour la stéréoscopie.

Un des problèmes majeurs des technologies à base de polarisation est la dépolarisation qu'induit certains matériaux. En effet, si les matériaux qui interviennent dans le chemin

17. Cela peut atteindre dix fois le prix d'une technologie tri-tube.

18. La technologie des tri-tubes, ne comprenant pas de grille, nous permet d'atteindre des définition de 1600 par 1200 ou, encore, d'avoir une définition carré de 1152 par 1152, à 120 Hz.

optique sont mal choisis, ils peuvent dépolariser la lumière. C'est pourquoi, les miroirs ainsi que les écrans sont toujours constitués de matériaux spécifiques.

Stéréoscopie par polarisation linéaire

La polarisation linéaire, plus ancienne, est basée sur le même principe que la polarisation circulaire. La différence est le type de polarisation.

La polarisation linéaire interdit un mouvement de tangage de la tête de l'observateur car la polarisation peut s'inverser et permuter ainsi l'image droite et l'image gauche : les mouvements de l'utilisateur sont donc limités à 5 degrés de liberté. Cela empêche l'utilisation d'une telle stéréoscopie dans le cadre d'un plancher ou d'un plafond. En effet, dans le cas de murs verticaux, l'utilisateur aura d'instinct la notion de haut et de bas, donc, de directivité du signal. Cette notion n'a plus de sens dans le cas de parois horizontales.

Stéréoscopie par anaglyphe

Il s'agit d'une stéréoscopie basée sur des filtres de couleurs (généralement rouge et cyan). L'utilisateur porte alors une paire de lunettes dont le verre gauche (de couleur rouge) filtre le vert et le bleu alors que le verre droit (de couleur cyan) filtre le rouge. A noter qu'il existe des filtres basés sur d'autres couleurs. Ainsi, en générant l'image, il suffit de masquer le rouge pour synthétiser l'image associée à l'œil droit et masquer le bleu et le vert pour synthétiser l'image pour l'œil gauche.

Il semblerait que n'importe quel type de projecteur puisse être utilisé pour ce type de stéréoscopie. Au niveau des couleurs, il y a cependant une petite subtilité pour les trois types de projecteurs. En effet, les tri-tubes fonctionnent en couleurs additives. C'est-à-dire qu'ils partent du noir et qu'ils ajoutent une composante de rouge, de vert ou de bleu. D'un autre côté, les DLP ou les LCD, utilisent des couleurs soustractives : ils partent d'une lumière blanche et retirent une composante de cyan, magenta ou jaune sur trois faisceaux dissociés à l'origine. Le résultat est toujours le même. Cependant, selon les fréquences des couleurs choisies par les constructeurs, il se peut que ces couleurs ne correspondent pas à celles de référence utilisées pour la plupart des lunettes anaglyphes.

Stéréoscopie anaglyphique étendue

C'est un nouveau concept développé par la société allemande TAN[19] sous le nom de Infinitec^{TM}qui demande à être validé. L'idée est de fonctionner sur la base de filtres passe-bande[20] sur chacune des trois couleurs. Par exemple, avec un filtre passe-bande bleu qui laisse passer la lumière comprise entre $0.47\mu m$ et $0.48\mu m$ sur l'œil droit et un filtre $0.46\mu m$ et $0.47\mu m$ sur l'œil gauche, il suffit d'émettre à $0.475\mu m$ pour l'œil droit et $0.465\mu m$ pour l'œil gauche. Dans les deux cas, on voit du bleu. Ils sont légèrement décalés, mais en prenant des filtres plus fins, on peut les rapprocher et ainsi atténuer le décalage. En utilisant le

19. Cette société a maintenant été absorbée par la société BARCO.
20. Un filtre passe-bande est un filtre qui ne laisse passer qu'une bande du spectre du signal complet.

même principe sur les deux autres composantes, on peut ainsi filtrer les images distinctes pour les deux yeux.

Cependant, la technologie n'est pas encore au point. En effet, il faut trouver des filtres passe-bandes qui autorisent simultanément trois bandes (une par couleur principale) relativement étroites et dans les fréquences qui nous intéressent. Mais surtout, il faut reproduire le miracle avec trois autres bandes légèrement décalées.

En ce qui concerne les projecteurs, la technologie tri-tube est dans ce cas à bannir. En effet, les développeurs de tri-tubes se soucient d'avantage de trouver des phosphores assez rapides pour éviter que le vert ne « bave » d'un point de vue stéréoscopique à un autre que de trouver des phosphores suffisamment distincts, bien que dans les mêmes longueurs d'onde. C'est donc vers les LCD ou les DLP que l'on doit se tourner. En effet, le principe de base de ces deux technologies est l'utilisation d'une lumière blanche séparée en trois faisceaux pour être chacun colorié en rouge, vert ou bleu puis filtré par pixel. Ainsi, l'utilisation de filtres adéquats placés dès la conception du projecteur sur chaque faisceau lumineux pourrait satisfaire les contraintes de ce type de stéréoscopie.

En théorie, on peut avoir une infinité de filtres différents. Cependant, à l'heure actuelle, cette technologie ne semble pas encore au point pour obtenir une stéréoscopie satisfaisante.

Auto-stéréographie

L'auto-stéréographie est le fait pour l'utilisateur de percevoir le relief sans l'aide de lunettes ou d'appareil visuel complexe. Parmi les méthodes, il y a celles qui requièrent une grande gymnastique des yeux. Par exemple, l'auto-stéréogramme oglige l'utilisateur à focaliser un point à l'infini pour percevoir la profondeur d'un objet sur un motif répétitif. Il existe d'autres méthodes basées sur deux images côte à côte. Dans ce cas l'utilisateur doit faire converger ses yeux pour percevoir le relief de l'objet.

Cependant, une technologie plus prometteuse et moins fatiguante pour les yeux est celle des « barrières ». Elle consiste en une grille placée à quelques centimètres de l'écran de telle manière à filtrer les pixels que l'utilisateur peut voir. Ainsi, il suffit de filtrer les pixels à afficher selon le masque pour l'oeil droit ou pour l'oeil gauche. Cette technologie est relativement proche du système lenticulaire qui s'applique sur l'écran et grâce à un réseau vertical permet de distinguer l'image associée à l'oeil droit de celle correspondant à l'oeil gauche. L'Electronic Visualization Laboratory de Chicago, dans [SMD+01] propose une extension du principe de Barrier (Varrier) qui permet, grâce à un capteur associé à la position de la tête, de tenir compte du point de vue de l'utilisateur. Cela permet, en temps réel, de définir la position de la grille par rapport au point de vue.

E Les calculateurs

Une part importante du travail de rendu d'un monde virtuel est effectuée par le calculateur graphique.

Ce dernier doit avoir la capacité de calculer la projection de la scène en temps réel. La notion de temps réel dans le rafraîchissement graphique est de l'ordre de 24 images

par seconde. Cela est dû à la persistance rétinienne qui permet de donner une illusion de mouvement par un affichage « échantillonné ».

Cependant, il ne faut pas oublier que nous sommes dans le cas stéréoscopique. C'est-à-dire que là où un film requiert un point de vue, il faut calculer deux points de vue. Il faut donc avoir une vitesse de rafraîchissement de l'ordre de 48 images par seconde.

A noter que dans le cas d'environnements immersifs, il faut calculer une image stéréoscopique par écran. Ainsi, il faut avoir, si possible, la puissance de calcul nécessaire au rafraîchissement de 48 images par seconde et par écran.

Cette puissance ne pouvant être absorbée par l'unité de calcul central (Central Processing Unit : CPU), il faut souvent avoir une carte graphique dédiée par écran.

Machines parallèles (de type Onyx)

Historiquement, les premières machines permettant de piloter des dispositifs immersifs étaient des machines Silicon Graphics[21].

Les machines de type Onyx, Onyx2, puis Onyx3 sont entièrement dédiées au pilotage d'environnements immersifs. Elles sont basées sur une architecture parallèle de type cc-NUMA (cache coherent - Non Uniform Memory Access). Cette architecture travaille sur un ensemble de nœuds CPU interconnectés par une architecture hyper cube. Chaque nœud est connecté à une carte graphique haute définition. Nous pouvons ainsi avoir théoriquement autant de cartes graphiques que de nœuds.

Au-delà du calcul graphique de grande qualité, ces machines autorisent l'utilisation de quatre buffers de rendu, contrairement aux deux habituels[22]. Ces quatre buffers sont en fait les deux buffers standard dupliqués afin d'obtenir un jeu complet pour l'œil droit et un autre pour l'œil gauche. Au niveau de l'implémentation, la carte graphique va successivement chercher l'information dans le buffer de droite puis dans le buffer de gauche. Les cartes sont également capables de fournir l'information sur le buffer actuellement en cours d'affichage. Cette information fait partie des nombreux éléments que l'utilisateur peut paramétrer en reprogrammant le générateur de signal graphique.

Un autre point à ne pas oublier est que le signal de synchronisation stéréoscopique (genlock) doit être identique sur l'ensemble des écrans du dispositif. Or, ce signal est directement dépendant du rafraîchissement vertical. Ainsi, ces machines doivent être capables d'autoriser une synchronisation des cartes graphiques les unes par rapport aux autres.

Grappes de PCs

Aujourd'hui, les prix pratiqués pour les machines parallèles est d'autant plus prohibitif que des solutions basées sur des « clusters » de PCs[23] émergent de plus en plus. Cependant,

21. Cette société s'appelle aujourd'hui SGI.

22. Les deux buffers sont ceux utilisés dans le cas du double-buffer avec un buffer arrière, utilisé pour construire la scène et un buffer avant qui est celui affiché.

23. « cluster » de PCs ou « grappes » de PCs en Français.

le problème des grappes de PCs est de plusieurs ordres :

- Il faut trouver des cartes graphiques qui autorisent une synchronisation comme celle décrite pour les machines parallèles. Cela comprend la synchronisation verticale entre cartes ainsi que l'émission du signal de synchronisation stéréoscopique avec des lunettes actives.
- Là où les machines parallèles partagent la mémoire entre tous les processus qui pilotent les cartes graphiques, les PCs des grappes ne peuvent pas la partager. Ainsi, une part importante de la grappe doit être dédiée à la distribution de l'information, notamment par l'adjonction d'un réseau très haut débit. Nous verrons section 6, page 154, l'utilisation de réseaux haut débit dans le cadre de grappes de PCs.

F Limites de l'approche stéréoscopique

Les deux techniques d'immersion ci-dessus (HMD ou murs) introduisent cependant plusieurs problèmes.

Problème de l'objet interposé

Imaginons que nous sommes dans un environnement immersif. Supposons un objet en émergence par rapport à l'écran. Si le pointeur de la souris est sur cet objet, la perception est complètement faussée car le pointeur se trouve au niveau du plan de convergence, c'est-à-dire l'écran, alors que l'objet est devant celui-ci.

Lorsque l'utilisateur cherche à interposer sa main entre l'objet et l'écran, le problème est rigoureusement le même. En effet, la perception de la profondeur est perturbée lorsqu'un objet virtuel masque un objet qui est théoriquement derrière. Le problème est aussi de même nature lorsque l'utilisateur est devant une station de travail : en effet, l'interaction est limitée à la fenêtre de l'écran. Il en résulte que les bords de l'écran coupent la scène même si les objets virtuels émergent de ce dernier. Nous reviendrons sur cela section C, page 109.

Notons par ailleurs que l'interactivité est limitée lorsque deux utilisateurs veulent travailler sur un même objet situé entre les deux utilisateurs : l'un des deux utilisateurs peut occulter les objets à l'autre utilisateur.

Limitation du nombre d'utilisateurs

L'une des limites des technologies actuellement disponibles est le nombre d'utilisateurs concernés par la stéréoscopie. En effet, avec des casques, pour « embarquer » un utilisateur supplémentaire, il faut ajouter un nouveau casque. Dans le contexte des murs, on peut avoir de nombreux utilisateurs, mais un seul aura un relief exact, ou au plus 2 groupes d'utilisateurs (multi stéréoscopie).

C'est donc une limitation fondamentale à ces deux types de technologie.

6.2 Modalité auditive

Il est intéressant de noter que la richesse du son est nettement supérieure à celle de la vue. En effet, là où l'œil humain à un champ de vision d'au plus 180°, le champ auditif est lui de 360°autour de la tête.

Les travaux présentés dans ce mémoire de thèse ne portent pas sur l'audio 3d. Au demeurant, il convenait de signaler les potentialités de cette modalité.

A Changement dimensionnel

Par définition, contrairement aux aspects visuels, le son comporte quatre dimensions. Intrinsèquement, le son est bidimensionnel (une composante d'amplitude et une composante temporelle). Cependant, la répartition spatiale du son est très riche car elle permet de déterminer la localisation des sources.

Concernant les changements dimensionnels à proprement parler, il n'y a aucun algorithme « miracle » (comme les iso-surfaces dans le domaine visuel). Cependant, le son est beaucoup plus riche que l'image. Tout comme l'image, le son comporte une composante fréquentielle. De plus, l'être humain est capable de percevoir les harmoniques[24]. Ainsi, on peut imaginer transposer une information multidimensionnelle dans des compositions d'harmoniques.

B Le relief

Il existe des systèmes qui permettent aux signaux audio de fournir une information de profondeur sur un objet. Tout d'abord, le système le plus répandu dans le règne animal est celui du sonar : l'animal émet un son et il est capable d'en déterminer la profondeur en fonction du délai de la réflexion sur les objets de la scène. Par exemple, les chauves-souris se servent d'ondes ultrason et les baleines d'ondes dans le spectre sonore audible. Les personnes aveugles se servent aussi de la réflexion du son sur les objets pour se faire une représentation mentale de la pièce qui les entoure. Nous sommes également capables d'estimer la taille d'un espace grâce aux réflexions des sons sur ses parois. C'est notamment le cas dans les églises, où l'atténuation du son lors des réflexions sur les parois intérieurs est très faible. Ainsi, nous pouvons avoir une idée de la hauteur des voûtes, grâce au retard de la perception d'un son par rapport à son émission.

De même, grâce à l'effet Doppler-Fizeau[25], nous sommes capables de connaître une profondeur relative d'un objet en mouvement par rapport à nous.

24. L'harmonique d'une note musicale est un son dont la fréquence est un multiple d'une puissance de deux par rapport à la fréquence fondamentale de la note initiale (ie. : $f_{harmonique} = f_{fondamentale} * 2^n$). Ce son, en se superposant à la note permet de décrire des signaux périodiques plus complexe que de simples sinusoïdes. La composition des harmoniques d'une note caractérise le timbre de chaque instrument.

25. Cet effet crée un décalage fréquentiel sur un phénomène ondulatoire lorsqu'il y a déplacement relatif d'une source et d'un récepteur. Perceptivement, cela se traduit par un décalage vers l'aigu des sons lorsque la source se rapproche et vers le grave lorsque la source s'éloigne.

Cependant, ces systèmes ne sont pas utilisables en toutes circonstances. Ainsi, contrairement à la modalité visuelle, la perception de la profondeur d'une source sonore est très difficile à ressentir.

En fait, il y a une part cognitive importante dans la perception de l'éloignement d'une source. En fonction de la nature d'un son, notre perception de sa profondeur sera plus ou moins fine. Par exemple, à niveau sonore identique, la perception de la distance d'un orage sera totalement différente de celle de la distance d'un oiseau.

C Perception de la direction

Un point important à noter est que l'oreille est incapable, seule, de déterminer l'origine d'un son. Le fait de coupler deux oreilles permet de percevoir un cercle d'où le son vient potentiellement grâce à l'écart du temps qui sépare le son perçu par une oreille par rapport à l'autre (voire figure 1.5). Cependant, cela dépend du milieu dans lequel nous le percevons. En effet, dans l'eau, le son se propage plus vite. Ainsi, dans ce milieu nous devenons incapables de percevoir l'écart de temps entre les deux oreilles.

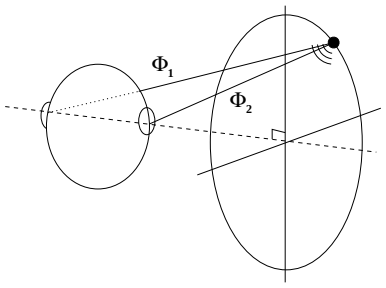

FIG. 1.5 – *Une source sonore perçue par deux capteurs est située sur un cercle d'incertitude.*

L'écoute binaurale ne permettant que de percevoir un cercle où est située la source sonore, il faut réduire les incertitudes sur le cercle afin de percevoir l'origine d'un son. Ainsi, tous les efforts dans les environnements immersifs consistent à rendre la sensation de l'origine d'un son.

Mouvement Pour réduire l'incertitude sur la position d'une source sonore, nous utilisons couramment le mouvement. En déplaçant les capteurs que sont les oreilles, nous sommes capables de déterminer plusieurs cercles contenant l'origine. Ainsi, la source se trouve à l'intersection des différents cercles. Cependant, cette méthode n'est pas suffisante lorsque le son est impulsionnel, c'est-à-dire lorsque le temps pendant lequel il est produit est très court.

Fonctions de transfert inter-aural Les oreilles ne sont pas que de simples capteurs. Elles sont environnées par la tête, le buste et le corps entier. Ainsi, un son qui vient d'une

source sonore extérieure au corps sera modifié par les différentes réflexions et absorptions dues à la peau et aux cheveux. C'est ainsi que nous sommes capables, en écoutant les déformations sonores, de retrouver l'origine d'un son. Ces déformations sont aussi appelées fonctions de transfert inter-aural [Mol92, WWK91] car le corps agit comme un filtre avec une fonction de transfert spécifique au corps humain.

Dans les faits, ces fonctions de transfert sont propres à chaque personne. Pour une écoute aux haut-parleurs, les fonctions de transfert inter-aural seront naturellement présentes. Par contre, il faut les recréer dans le cadre d'une écoute au casque, car le son est directement diffusé auprès de l'oreille. De plus, comme l'utilisateur a la possibilité de se déplacer en gardant le casque sur les oreilles, il faut rafraîchir la position de ce dernier pour adapter la position de toutes les sources virtuelles. Cette fréquence de rafraîchissement doit être au moins de 100Hz afin que l'utilisateur ne soit pas perturbé à long terme par une latence et ne perçoive pas de saccades dans la position du son (cette fréquence est équivalente à celle de 24 images par seconde pour l'échantillonage du mouvement sur la modalité visuelle).

D Codage du son

Le son 3d n'est pas du tout calculé comme l'image. En effet, là où l'image est composée de facettes qu'il faut positionner dans le monde virtuel, le son est une somme de sources sonores distribuées en 3d dans l'espace virtuel. Qui plus est, la facette n'a pas obligatoirement de composante temporelle. A contrario, l'aspect ondulatoire du son impose une dynamicité à ce codage.

A l'origine, la sonorisation ne contenait qu'un unique canal. Puis, est venue la stéréophonie. C'est le système actuellement le plus utilisé en dehors du milieu professionnel. L'avènement des lecteurs DVD avec un codage dolby^{TM}5.1 est en passe de détrôner ce standard. Il fournit les deux canaux de la stéréophonie, mais en plus deux autres canaux latéraux, ainsi qu'un canal vocal (en position centrale par rapport aux canaux stéréophoniques) et un canal de basses. Il existe d'autres évolutions dans le domaine grand public comme le codage DTS^{TM} [26] qui introduit un septième canal dans sa version 6.1 encodant le signal d'un haut-parleur arrière. En fait, tous ces codages font évoluer la stéréophonie pour augmenter l'immersion auditive.

Cependant, les codages grand public ne sont pas satisfaisants car ils ne tiennent pas compte d'une élévation des sources sonores. En fait ces codages sont issus du monde du cinéma où les écrans sont plus étendus horizontalement que verticalement. De plus, il est très difficile, dans les salles de cinéma, d'ajouter un canal sonore pour une source en dessous de l'utilisateur. C'est pourquoi, très tôt, le milieu de la recherche a développé des systèmes multicanaux qui permettent de simuler n'importe quelle configuration audio. L'idée globale est de spatialiser l'ensemble des sources sonores dans ce modèle. Ensuite, en faisant la somme (fusion mathématique des signaux), on obtient l'ambiance sonore globale. Enfin, il suffit de passer de ce système au système de rendu final utilisé pour obtenir une

26. Le codage DTS est un codage proche du Dolby 5.1 mais dont la qualité est supérieure grâce à une compression de données inférieure.

spatialisation complète du son [Jot96].

L'un des systèmes les plus intéressants est le système ambiphonique. Il consiste à coder l'ensemble des contributions à différents ordres. L'ordre 0 consiste à capter le signal avec un microphone omnidirectionnel (soit un premier canal). L'ordre 1 est celui composé par une capture selon chaque orientation de chaque direction canonique (6 canaux). Dans les faits, il est difficile de capter au-delà de l'ordre 1. Ainsi, les systèmes ambiphoniques encodent généralement 7 canaux, tout comme le système DTS^{TM}6.1, mais le signal est mieux réparti spatialement. De plus, le codage ambiphonique autorise des conversions dans beaucoup d'autres systèmes tel que le Dolby^{TM}, la stéréophonie, le DTS^{TM}6.1, etc.

E Périphériques de rendu

Les casques audio

De même qu'il existe deux types de casques visuels (HMD ou see-through), il existe deux types de casques audio.

Les casques fermés ont pour caractéristique principale d'empêcher tout son extérieur de venir perturber le signal audio. Ainsi, l'utilisateur se trouve dans une « bulle » où seuls les sons du monde virtuel lui parviennent.

A contrario, les casques ouverts laissent passer le signal de l'extérieur. Dans le cas d'une immersion, cela permet à plusieurs utilisateurs d'interagir dans le monde virtuel sans que le système soit obligé de récupérer la voix de chaque utilisateur pour la « spatialiser » et la distribuer auprès des autres.

Avantages Les casques permettent à chaque utilisateur d'avoir son propre « point d'écoute » sur la scène audio. Ils permettent également, selon le cas, de supprimer les bruits ambiants. Ces bruits peuvent provenir des calculateurs, des vidéos-projecteurs ou bien des systèmes d'aération ou de climatisation.

Inconvénients L'utilisation des casques impose un calcul supplémentaire pour spatialiser le son. Il faut tenir compte de la fonction de transfert inter-aural pour rendre le son au bon endroit pour l'utilisateur. Ainsi, l'ajout d'un utilisateur supplémentaire, se traduit par l'augmentation de la charge à demander au calculateur. De plus, afin que le son soit rendu correctement, il faut grarantir un système de capture de mouvement à 100Hz.

Les « murs » audio

L'analogie au mur d'image est le mur de haut-parleurs. Il peut aller du simple haut-parleur aux murs contenant plus d'une dizaine de haut-parleurs.

Les murs permettent de faire une spatialisation de chaque objet puis d'en faire la somme. Le rendu est alors parfait pour un utilisateur à condition que les objets soient situés à l'arrière du dispositif audio par rapport à l'utilisateur (voir figure 1.6).

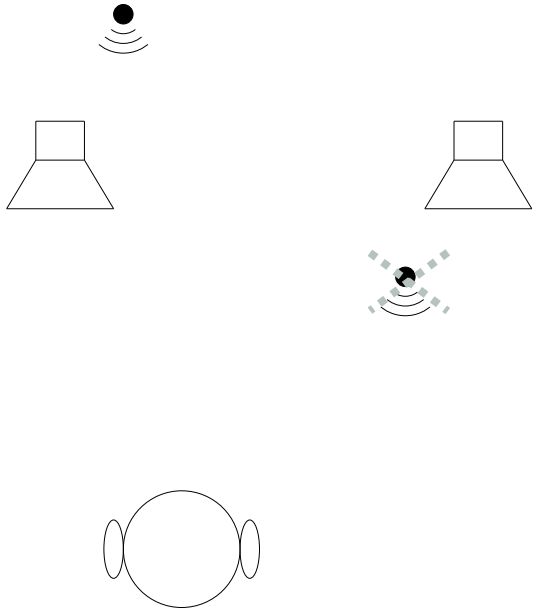

FIG. 1.6 – *Placement d'une source sonore par rapport aux différents générateurs sonores.*

Par contre, dans le cas de salles relativement bruyantes, le bruit ambiant perturbera le signal. De plus, comme dans le cas du mur graphique, il sera par exemple impossible de simuler un objet situé entre deux utilisateurs. En effet, un premier utilisateur occultera la source sonore pour le second utilisateur. En outre, la simulation 3d d'une source sonore pour deux ou plusieurs utilisateurs n'est pas possible avec des haut-parleurs. Ce problème peut être levé par les casques audio.

Un autre problème concerne les perturbations mutuelles des écrans visuels et du signal audio. Afin de ne pas gêner la vision, les haut-parleurs doivent être placés derrière les écrans. Cela suppose des écrans transparents à l'audio. Par exemple, dans certaines installations existantes, les écrans vibrent dès que le système sonore émet des sons graves.

Les hologrammes auditifs

Contrairement aux hologrammes visuels qui sont impossibles à réaliser sur des scènes dynamiques, la technologie des hologrammes auditifs donne quelques résultats. L'intérêt majeur d'une telle technologie réside dans la possibilité pour deux ou plusieurs utilisateurs d'interagir sur un même objet audio (émetteur de son).

Par contre, cette technologie n'est pas encore arrivée à maturité. Actuellement, les hologrammes auditifs fonctionnent correctement dans un plan à deux dimensions, mais pas dans un volume. De plus, le système requiert des haut-parleurs de petite taille donc « passe-

haut[27] » répartis tous les dix centimètres le long d'un cercle autour de la zone interactive. Cela crée une gêne entre les haut-parleurs et le chemin optique pour la projection.

6.3 Modalité sensorimotrice

Les retours sensori-moteurs, mis en œuvre par des périphériques haptiques, permettent à l'utilisateur de percevoir des sensations de pression ou de contact sur le corps humain. Ils peuvent servir, par exemple, à simuler la poignée d'une valise en empêchant l'utilisateur de fermer le poing. Cependant, afin de rendre correctement les sensations de contact, il faut utiliser des systèmes pour calculer en temps réel les collisions des différentes parties du corps avec l'ensemble de la scène virtuelle. Ces retours sensori-moteurs sont de deux types différents.

Retours internes

Ces retours haptiques ont un point d'appui qui n'est pas fixe par rapport au référentiel fixe dans lequel est placé l'utilisateur. Par exemple, le Cybergrasp a pour point d'appui le dos de la main.

Le défaut majeur de ces périphériques est de ne pas fournir un retour d'effort global et donc de ne pas empêcher certains mouvements. Ils permettent notamment à un utilisateur d'avoir un retour d'effort lors de la préhension d'un élément comme le combiné d'un téléphone. Cependant, ils n'empêcheront pas la main de l'utilisateur de traverser la table sur laquelle est posé ledit téléphone.

Retours externes

Les retours externes ont pour point d'appui un point fixe de l'espace où se déplace l'utilisateur. Par exemple, le Phantom est un bras articulé solidaire d'une table.

Cela pose un autre type de problème. En effet, les bras articulés limitent l'espace d'interaction, sans oublier qu'ils perturbent le champ de vision dans l'environnement immersif (on parle alors de leur caractère « intrusif »).

Il y a cependant des solutions comme le Spidar [LCS97]. Il est constitué de plusieurs fils reliés à des moteurs placés aux extrémités du dispositif. On voit mal au demeurant comment plusieurs personnes peuvent utiliser dans un même dispositif un tel système.

6.4 Olfactif

Des systèmes olfactifs existent qui permettent de rendre une centaine d'odeurs différentes. Il existe, par exemple, le iSmell, de DigiScents Inc et le Pinoke de AromaJet.com. Cependant, ce type de périphérique est avant tout dédié et pensé pour des applications sur le web. Ainsi, leur utilisation dans le cadre de la Réalité Virtuelle n'est pas du tout appropriée.

27. Un filtre passe-haut est un filtre qui ne laisse passer que les hautes fréquences, donc les aigus mais pas les graves.

6.5 Interfaces bio-technologiques

Hollywood montre des interactions humaines avec des mondes virtuels au travers d'interfaces neuronales comme dans le film *Matrix*. Cette technologie en est à ses balbutiements. En effet, les chercheurs viennent seulement de trouver un moyen de connecter un neurone à une puce de silicium [ZF01].

Les travaux actuels, axés sur le milieu médical, portent surtout sur la réhabilitation de fonctions laisées par des accidents ou des maladies. Cependant, il existe des retombées intéressantes de ces techniques dans le domaine de la Réalité Virtuelle. Par exemple, des chercheurs sont actuellement en train de travailler sur un système permettant de contrôler l'oreille interne. Ce type de dispositif pourrait, par exemple, permettre de réduire le risque de cybersickness dans les environnements immersifs.

Il convient de souligner que ces approches posent des questions d'éthique. Ainsi, l'utilisation courante de telles technologies prendra un certain temps.

6.6 Fission multimodale

Nous avons vu plus haut que l'utilisateur se sert de la fusion multimodale de l'information pour agir sur le monde qui l'entoure.

D'un autre côté, il est courant que l'utilisateur se serve de plusieurs modalités pour avoir des informations sur le monde qui l'entoure. Par exemple, une personne qui marche sur une route sera informée par le canal auditif qu'une voiture arrive derrière lui. Cependant, c'est le canal visuel qui lui signalera que le véhicule n'est pas sur la même voie que lui.

La fission multimodale permet à une information de se propager selon plusieurs canaux sensoriels.

Dans les environnements immersifs, il n'est pas rare qu'il y ait des collisions entre objets virtuels. Ainsi, en utilisant les canaux sonores et visuels, il est possible d'informer l'utilisateur même si celui-ci tourne la tête à l'événement qui l'intéresse.

6.7 Substitution de modalités et systèmes pseudo-haptiques

Comme nous l'avons vu dans la partie des systèmes haptiques, l'un de leurs grands défauts est l'encombrement, voire le caractère intrusif, de ce type de dispositif.

Une des solutions est le remplacement des modalités haptiques par d'autres modalités. On peut en effet substituer à la collision « physique » de la main sur un mur un effet visuel ou sonore.

C'est pourquoi certains travaux visent à leurrer le cerveau en lui donnant l'illusion qu'il agit physiquement sur un objet. Ces systèmes sont dits pseudo-haptiques.

Par exemple, *Anatole Lecuyer* [Lec01, LKC+01] a étudié, dans sa thèse, un système permettant de simuler des raideurs de ressorts sur une space-ball alors que ce périphérique est incapable de produire un tel stimulus. La space-ball lui permet d'obtenir la force exercée par l'utilisateur, et en retours, il simule visuellement l'enfoncement du ressort. La modalité visuelle donne alors à l'utilisateur une information sur la rigidité du ressort (ralentissement

de la partie mobile dont il montre la représentation 3d) en relation avec le retour d'effort qu'aurait perçu l'utilisateur s'il avait appuyé sur un ressort réel.

7 Conclusion

Ce chapitre était consacré à l'étude des concepts et des définitions qui ont servi de base à mes travaux dans le domaine de la Réalité Virtuelle.

Le domaine de la Réalité Virtuelle et Augmentée couvre un large éventail de disciplines. De plus, si l'on commence à intégrer des considérations sur la perception, cela revient vite à tenir compte de problèmes cognitifs liés à la Réalité Virtuelle. Nous avons, par exemple, vu que dans le cas du son, la perception des distances dépend beaucoup des aspects cognitifs. Cette composante est importante dans tous les domaines associés aux retours sensori-moteurs du monde virtuel vers l'être humain. En d'autres termes, sur la base de ses expériences perceptuelles, l'humain acquiert une imagerie mentale (tant visuelle, qu'auditive ou haptique) à partir de laquelle il construit toute une connaissance spatiale sur l'univers réel [Den79]. Il utilise souvent cette connaissance pour développer des stratégies cognitives dans l'appréhension des mondes virtuels. A titre d'exemple anecdotique, une tasse à café virtuelle aura beau être loin de l'utilisateur, elle semblera proche si sa taille semble supérieure à celle d'une tasse réelle. Cependant, même si j'ai étudié certains de ces aspects lors de mes travaux, la contribution de ma thèse ne concerne pas ce domaine des sciences cognitives mais d'avantage les moyens matériels et logiciels pour mettre en œuvre des interactions « naturelles ».

La palette des applications possibles autour de la Réalité Virtuelle est très vaste. Même si de nombreuses applications se cantonnent à la visualisation stéréoscopique d'objets, il en est d'autres qui requièrent une interaction beaucoup plus importante. Ce type de système peut être proposé au grand public comme aux spécialistes. Cependant, ces derniers utilisent souvent la Réalité Virtuelle comme un support pratique à une connaissance théorique (cf. conception informatique, apprentissage, maintenance assistée...).

Sur le plan matériel et logiciel, les types de périphériques utilisés peuvent introduire des contraintes sur le système chargé de les gérer. La plupart des environnements immersifs sont conçus autour du canal visuel. Comme le montre cet aperçu conceptuel et bibliographique, le son et l'haptique sont des canaux sensoriels importants pour l'homme. Ils commencent à se généraliser dans le domaine de la Réalité Virtuelle [Wen98, WWF98]. Les autres canaux sensoriels posent par contre beaucoup plus de contraintes calculatoires, voire éthiques.

Ces considérations sur les environnements immersifs sont à la base de toute ma réflexion sur les contraintes imposables à une architecture distribuée pour environnements immersifs. Ainsi, l'architecture de la plate-forme EVI3d avait pour ambition de permettre l'intégration de l'ensemble des systèmes, dispositifs et périphériques décrits ci-dessus.

Chapitre 2

Architecture logicielle de EVI3d

Sommaire

1 Introduction

L'un des deux principaux travaux que j'ai effectué lors de ma thèse a été d'étudier, de spécifier et de concevoir une architecture logicielle permettant d'intégrer l'ensemble des contraintes inhérentes à l'interaction réaliste en environnements immersifs.

Dans cet esprit, nous avons été particulièrement préoccupés par la gestion de modalités avancées telle la reconnaissance vocale ou gestuelle. Au demeurant, nos interactions doivent rester relativement familières à l'utilisateur. C'est pourquoi les périphériques de base de la RV ont été conservés tout en proposant de les spécialiser à certaines modalités pour rendre leur usage plus naturel.

La motivation de ce travail sur l'étude et la mise en place de l'architecture EVI3d réside dans l'absence de véritable plate-forme de Réalité Virtuelle pour faire des études cognitives avancées. En effet, aucune librairie ne permet la mise en place de paradigmes interactifs évolués mettant en œuvre par exemple des événements de très haut niveau sémantique dans une architecture distribuée. De plus, le double canal de l'architecture EVI3d permet de dissocier deux types de connexion réseau fondamentalement différents. Cela permet une répartition plus homogène de la charge réseau ainsi que de la charge de calcul.

Dans ce chapitre, je vais commencer par décrire les systèmes de gestion des périphériques qui existent actuellement. Ensuite, je décrirai la partie événementielle de l'architecture EVI3d : l'EVserveur [4]. De plus, je mettrai en avant certains périphériques et leurs spécificités. J'aborderai ensuite la gestion multimodale des événements [1]. Enfin, je montrerai comment nous pouvons gérer les périphériques haptiques [3].

2 Travaux existants

L'un des problèmes fondamentaux que posent les Environnements Virtuels est le réalisme des interactions entre les utilisateurs, les objets de notre univers réel et les représentations numériques qui composent le monde virtuel. Gérer ce réalisme suppose une grande puissance de calcul. Ainsi, une grande partie des travaux récents s'est intéressée à la gestion distribuée d'applications de RV&A afin de pouvoir les répartir, via le réseau, sur plusieurs calculateurs.

Le but de certains des travaux évoqués est d'étudier les problèmes logiciels liés à la collaboration d'Environnements Virtuels distants. D'un point de vue applicatif, ce type de problème vise la gestion de tâches collaboratives qui est probablement l'un des verrous à faire sauter pour pouvoir faire de la RV le support du bureau d'étude du futur. Mais, en raison de leur complexité, les applications en Environnements Virtuels ont également besoin de la coopération de grand nombre de processus de plus bas niveau. En effet, la gestion des comportements interactifs des objets dans de tels environnements (rendu sensori-moteur, simulation de phénomènes physiques...) nous place dans une situation où une machine autonome est insuffisante pour assurer à elle seule les ressources de calcul utiles à de tels systèmes.

La gestion d'événements est importante à maîtriser si l'on souhaite que les Environnements Virtuels puissent contrôler correctement les interactions entre les utilisateurs et les objets tant réels que virtuels. Dans ce contexte, il importe de souligner que mes travaux partent de l'hypothèse que les événements ont a priori des occurrences apériodiques. Ceci n'est pas incompatible avec le fait que certains événements sont périodiques ou bien que leurs traitements doivent être gérés de façon synchrone. Certains processus visent même à évaluer si des événements ne sont pas « presque » synchrones (cf. cohérence temporelle en multimodalité).

Plusieurs approches d'architecture logicielle distribuée existent pour les applications de RV&A et [SZ99] est l'un des ouvrages les plus exhaustifs sur la question. Il y a principalement deux types d'architectures : les systèmes qui se concentrent sur les « échanges de données » et ceux qui s'intéressent à la gestion d'événements. Nous nous intéresserons, dans une première partie de ce tour d'horizon, aux approches orientées « échange de données ». Ces approches imposent l'usage de réseaux à large bande, où chaque application gère de façon propriétaire l'interface utilisateur / scène virtuelle. Puis, nous analyserons dans une deuxième partie les « systèmes génériques de gestion d'événements » pour en extraire leurs concepts principaux. Si ceux-ci sont très répandus et très utilisés dans des interfaces traditionnelles, ils ne sont pas prévus pour gérer les périphériques utilisés dans des Environnements Virtuels. Enfin nous ferons un point sur les « systèmes événementiels dédiés » aux Environnements Virtuels qui présentent encore certaines limitations.

2.1 Approches orientées échange de données

Les principaux systèmes de ce type s'appuient sur le concept de *mémoire partagée*. Ces systèmes répartissent via le réseau une réplique entière (ou *miroir*) de la base de données

de la scène virtuelle utilisée. Ainsi, chaque machine cliente peut avoir un accès à tous les objets de la scène.

A Principaux systèmes

Systèmes militaires

Au commencement, il y eut le programme de recherche ENIAC (Electronic Numerical Integrator And Computer). Son but principal était de simuler des explosions nucléaires pour concevoir la bombe atomique. Ainsi, la première simulation par ordinateur fut développée par l'armée américaine et les systèmes de simulation les plus importants ont longtemps été enfermés dans leurs laboratoires ou les écoles militaires. Dans ce contexte, l'ancêtre des logiciels distribués de simulation est SIMNET (SIMulation NETwork) [Cos99]. Ce logiciel a été conçu en 1983 par un jeune scientifique de l'agence ARPA (Advanced Research Projects Agency), un programme militaire qui a également développé l'ARPA-NET qui est devenu Internet dans les années 80. SIMNET est maintenant un système obsolète.

[PBD91] introduit en 1991 le modèle DIS (Distributed Interactive Simulation). Ce modèle vise la simulation à très grande échelle et est entièrement basé sur le concept de mémoire partagée via le réseau. Le principe fondamental de DIS est d'avoir plusieurs machines coopérant sans avoir besoin de serveur. Cette absence de serveur signifie qu'aucun contrôle n'est opéré sur les flux de données. En conséquence, chaque machine doit traiter toutes les informations envoyées par chacune des autres machines. DIS a été utilisé par la Naval Postgraduate School américaine pour développer en 1993 son propre système de simulation NPSNET [ZPOM93].

Une évolution du système DIS apparaît en 1998 avec HLA (High level Architecture), un système développé par le département américain de la défense [HLA98]. HLA fournit une architecture commune pour la modélisation géométrique et la simulation. Cette architecture définit une API[1] générique pour que des applications indépendantes puissent échanger des informations. Le protocole d'échange de données de HLA est sur le point d'être reconnu comme un standard par le comité IEEE. Le but principal de ce protocole est de transférer de gros volumes de données. Pour ce faire, ce protocole suppose que ce type d'échange est possible si les variations de ces informations sont définies sur des domaines continus mais bornés. Si des signaux continus (captés par une souris, un traqueur ou un gant numérique ...) peuvent être facilement implémentés, ce protocole ne semble pas vraiment adapté à des interactions ponctuelles (événements clavier, reconnaissance vocale ou gestuelle ...). De plus, qu'ils soient continus ou ponctuels, ces signaux doivent pouvoir être combinés ; en d'autres termes, la dimension multimodale de l'interaction humaine n'est pas prévue dans HLA. Un autre point critique semble être la gestion des informations indéfinies. Ce type d'information peut être retourné par des périphériques ou par des systèmes de reconnaissance. C'est par exemple le cas lorsque l'utilisateur sort du champ d'un capteur, ou lorsqu'un système de reconnaissance de la parole n'arrive pas à reconnaître certains

1. Une API(Advanced Programming Interface) permet à un logiciel de s'interfacer avec la librairie associée.

mots. Dans ce contexte, des informations au contenu indéfini peuvent alors transiter, ce qui paraît difficilement compatible dans HLA avec le fait que les domaines de variation des informations doivent être continus et bornés.

Systèmes civils

L'un des premiers systèmes civils employant le concept de la mémoire partagée est DIVE (Distributed Interactive Virtual Environment) [CH93]. Lorsque le projet DIVE fut lancé, l'immersion dans une scène virtuelle s'effectuait à l'aide de périphériques visuels de type HMD^{TM}. Il en résulte que ce système s'est concentré sur la gestion et le partage d'avatars humains. Cependant, dans des dispositifs de type CAVE^{TM}ou RAVE^{TM} [2], ou encore avec un système panoramique de type *Reality Center*^{TM}, les lunettes actives rendent les avatars relativement inutiles puisque les utilisateurs, placés au sein d'un même dispositif, se voient mutuellement et perçoivent leur propre corps. Dans le contexte de ces grands dispositifs immersifs, le seul cas où la gestion des avatars redevient nécessaire est celui où la coopération entre utilisateurs se fait à distance via plusieurs de ces dispositifs.

AVIARY, pour sa part, formalise d'avantage les Environnements Virtuels [SW94]. Ce système introduit la notion d'« artefact », c'est-à-dire une entité qui englobe les objets purement virtuels aussi bien que les représentations des humains (i.e. avatars). Le monde virtuel est représenté comme un « environnement » obéissant à un ensemble de lois (comme la pesanteur) qui gèrent toutes sortes d'objets et pour ce faire une application contrôle le comportement de chaque objet. Une autre notion importante de ce système est que l'utilisateur agit en tant que module de cet « environnement ».

CAVERN (CAVE Research Network) est un regroupement d'industriels et de laboratoires de recherche equipés de CAVE ou de tables de travail immersives reliées par des connexions à haut débit. Ils ont développé CAVERNsoft [LJD97] pour permettre une collaboration entre ces différents sites. Ce logiciel se focalise sur les connexions haut débit afin de pouvoir travailler avec des calculateurs graphique puissant, notamment sur des applications de visualisation scientifique. Contrairement à d'autres systèmes, CAVERNsoft se base sur un système réseau « fiable » [3]. La seconde version de ce système [PCK^{+}00] est conçues de manière modulaire. De plus, ils proposent aux utilisateurs des couches réseau plus bas-niveau. Cette couche bas niveau est prévue pour permettre l'échange de tout type d'informations entre le client et le serveur (notamment grâce à la classe de conversion de données). Cependant, sa structure se prête mal à la gestion et à la distribution des périphériques de RV. Etant donné qu'il permet principalement l'échange entre sites distants, rien n'est prévu pour le passage des messages chargés de la gestion et de la synchronisation des périphériques. De plus, ce système semble trop orienté TCP/IP [4] ou UDP/IP [5], ce qui crée des difficultés dans l'utilisation de protocoles réseau optimisés.

2. acronyme de *Reconfigurable Automatique Virtual Environment*.
3. On pourra se référer à la section B, page 65 pour comprendre le terme fiable.
4. acronyme de *Transmission Control Protocol on Internet Protocol layer*.
5. acronyme de *User Datagram Protocol on Internet Protocol layer*.

L'un des premiers projets français sur les environnements distribués est OpenMASK [DCDK98] (General Animation and Simulation Platform) développé par l'équipe SIAMES de l'IRISA [DCDK98]. Ce système est basé sur une architecture « producteur / consommateur ». Ce principe très générique pouvait répondre à nos besoins, mais n'était pas librement disponible au moment où débutèrent nos recherches. Nous reparlerons de ce système en conclusion et perspective. Chaque module est appelé successivement par le noyau du système avec un ensemble de paramètres. Les paramètres sont des « branchements » réalisés à partir de la sortie d'autres modules. Ces branchements peuvent être interpolés ou extrapolés. GASP a évolué dernièrement vers OpenMASK. Il semblerait que les spécifications n'aient pas beaucoup changé entre les deux versions.

PaVRML est une extension de VRML (Virtual Reality Markup language) [Rag94] développée dans le cadre du projet européen PAVR (Platform for Animation and Virtual Reality) [FFH+99]. La solution proposée par PaVRML est plus générique que toutes les autres car elle permet la coopération entre un grand nombre de plates-formes graphiques, telles que : OpenInventor™, Alias Wavefront™, SoftImage™, Performer™... L'approche consiste à partager un fichier de données dans un répertoire réseau à travers des systèmes standard d'échange de fichiers tels que NFS (Network File System, utilisé par la plupart des OS Unix) ou Samba (système de gestion de fichiers à distance utile pour se connecter avec le NetBios de Microsoft). Le format des fichiers PaVRML inclut la gestion d'objets animés via des ajouts internes[6] de certains logiciels standards. Ceci permet de gérer le comportement d'objets virtuels distribués sur plusieurs Environnements Virtuels. Le défaut majeur de cette approche est qu'elle ne fournit pas de solution générique pour la gestion des événements et des périphériques de RV&A. En d'autres termes, cette gestion reste à la charge de chaque plate-forme logicielle.

Systèmes ludiques

Il faut noter que les simulateurs distribués les plus répandus sont utilisés pour des jeux. En effet, ils permettent à plusieurs joueurs répartis sur un réseau de collaborer pour exterminer des « ennemis » dans des mondes virtuels. Dans ce contexte, il paraît pertinent d'analyser les protocoles réseau de logiciels comme Quake world [Jan94] ou Half-life. La philosophie de l'approche distribuée de ces jeux est simple : ils utilisent la même scène (la carte du jeu) pour tous les joueurs, tandis que les objets de la scène sont peu complexes, parfaitement connus et dotés de comportements prédéfinis. Les joueurs et les objets ont un nombre limité de mouvements possibles. En conséquence, les interactions logicielles (y compris celles entre sites) sont triviales, car les seuls échanges consistent à communiquer la position et l'état de chaque joueur. Ce type de jeu utilise des objets autonomes dont certains peuvent être paramétrés par les utilisateurs. Au demeurant, ce paramétrage est très limité et ne permet pas aux joueurs d'introduire des informations autres que celles prévues par le concepteur du logiciel.

6. Un ajout interne (pluggin, en anglais) est un module logiciel qui vient s'ajouter au sein d'un logiciel afin d'y incorporer des fonctionnalités supplémentaires.

B Discussions

Approches multicast

Les systèmes qui utilisent les concepts de DIS (tels que NPSNET ou HLA) visent la gestion de bases de données de très grande taille et peuvent ainsi gérer jusqu'à 100.000 objets en interaction (cf. soldats, unités...) pour le cas de DIS. De plus, chaque objet peut être vu comme un processus séparé. Aussi, pour distribuer ce type de données, une solution consiste à utiliser les outils IP *multicast* (ou *Internet Protocol multicast*).

L'IP multicast [AFM92] est un des meilleurs outils pour gérer la mémoire partagée. Il permet à un groupe d'ordinateurs d'échanger des données sur un réseau dont on ne connaît pas la topologie logique du réseau (contrairement au protocole *broadcast* qui est bloqué par les routeurs). Pour être membre d'un groupe, il suffit à chaque application cliente d'en faire la requête. Selon la configuration et l'état actuel du réseau, cette requète sera transmises au contrôleur local, autrement appelé *multicast router*. Quand une application cliente veut soumettre une information aux autres membres de son groupe, ceci se fait en une seule fois car elle n'a pas à adresser individuellement cette information à chaque membre du groupe. Le principe même des protocoles multicast est en effet de laisser aux membres destinataires la tâche de récupérer sur le réseau les informations qui les concernent.

Cependant, comme le signale [PMB99], un rapport du LSMA[7], l'approche multicast présente quelques problèmes, en particulier :

- Généralement, la couche de transport utilisant l'IP multicast est l'UDP. Ainsi, le protocole IP multicast n'est pas fiable. Or, il ne s'agit pas d'un protocole connecté (ou *connected protocol*) de sorte qu'il ne possède pas de contrôle de flux. Cela signifie que des paquets de données peuvent être dupliqués ou perdus sans aucun avertissement. Il existe des systèmes multicast basés sur TCP (protocole connectée) [LP96]. Cependant, le système de contrôle de flux, sature le réseau dans le cadre de cette utilisation. La conséquence directe est une très grande perte de performances du réseau.
- Le protocole multicast est principalement adapté aux réseaus fortements diffusifs. Ainsi, ce protocole n'apporte rien aux réseaux locaux commutés (ou *switched networks*). Un réseau commuté est un réseau où des connexions sont établies dynamiquement sans aucune intervention de l'émetteur : contrairement au réseau Ethernet, ATM (Asynchronous Transfer Mode) est typiquement un réseau commuté. Cependant, la philosophie « point à point » de ce genre de réseau ne permet pas l'utilisation des méthodes broadcast requises pour le multicast. En effet, pour que chaque client puisse recevoir l'information, il faut que le commutateur recopie l'information sur chaque canal. Cela revient à « sérialisé » l'information. Il n'y a en revanche aucun problème de ce genre avec le réseau Ethernet qui sont fortement diffusif. De plus, cette sérialisation est beaucoup moins génante dans le cas des WAN (Wide Area Network), puisque les clients sont généralement regroupé au sein de LAN.

7. le groupe de travail en Large-Scale Multicast Applications du IETF (Internet Engineering Task Force) : http://www.ietf.org/html.charters/lsma-charter.html.

TCP est plutôt destiné à une utilisation dans le cadre de transfert de gros volumes de données (FTP, HTTP, ...). Inversement, UDP, couche de transport de prédilection pour le multicast, était plutôt destiné à des échanges d'événements (X-window, NTP). Cependant, dans le cadre des applications de RV, il faut que nous possédions l'ensemble des événements du système. En effet, les événements que nous perdons peuvent engendré un « trou » dans la commande courante. C'est pourquoi, le protocole TCP sera préféré au protocole UDP. Inversement, le transfert des bases de données n'a pas de persistance dans le cadre d'une simulation sur plusieurs ordinateurs. En effet, la base de donnée à une instant t sera « écrasée » par celle de l'instant $t + 1$. Ainsi, la perte de données sera transitoire. Cela justifie l'utilisation de protocole UDP sur IP multicast, beaucoup moins lourd en terme de bande passante qu'une transmission unicast par TCP/IP. En d'autres termes, on préfèrera faire du multicast plutôt que du multi-unicast.

En résumé, il y a deux possibilités conceptuellement antinomiques. La meilleure utilisation du protocole multicast est l'échange des données massives (par exemple, la base de données entière d'une scène virtuelle avec beaucoup d'objets). Inversement, si les données à échanger sont de faible taille (i.e. des données fortement compressées ou des informations compactes, par exemple de type fonctionnelles ou sémantiques, associées à un nombre restreint d'objets), le multicast n'est pas approprié. Cette conclusion est renforcée par le manque de fiabilité du protocole multicast. Le fait est que si les données ne sont pas compressées ou que leur volume est très important, une erreur de transmission induit de faibles pertes, de sorte que l'usage du protocole multicast est bien une solution. Par contre, si des paquets sont détruits ou dupliqués sur des données fortement compressées ou sur de petites bases de données, les conséquences sont bien entendu plus dommageables. Ainsi, avec les applications basées sur les concepts de DIS, l'utilisation du multicast est complètement justifiée, mais un objet doit envoyer plusieurs fois la même information, car seule cette redondance permet d'éviter le manque de fiabilité de l'IP multicast.

Approches « producteur/consommateur »

Les systèmes de type *producteur/consommateur* travaillent sur un ensemble de processus voire de *threads*[8] qui sont passés en revue (cf. *scheduler*) par le système d'exploitation (OS) ou par le noyau de la plate-forme logicielle utilisée pour le développement d'une application de RV&A. Avec ces systèmes (tels que DIVE, AVIARY, OpenMask et ceux de type DIS) les pilotes de périphériques sont typiquement des modules producteurs d'information. Cependant, le fait que la machine doive alterner le traitement de plusieurs tâches peut introduire des latences importantes. En outre, certains de ces systèmes travaillent suivant un signal de synchronisation pour gérer l'alternance de traitement de chaque tâche. Cela permet de passer ces tâches en revue une à une. Mais au lieu qu'un événement extérieur à l'application active une tâche, l'événement concernant un processus ne commence à être traité que lorsque c'est au tour de ladite tâche d'être traitée.

8. des processus à mémoire « commune », à ne pas confondre avec des processus à mémoire « partagée » où la mémoire est distribuée (voire divisée) entre plusieurs processus.

Pour les systèmes de type DIS, la gestion d'une application est vue comme des interactions entre *objets autonomes*. Cependant, dans certaines applications de RV&A, en particulier pour l'exploration de phénomènes scientifiques, il peut être difficile d'extraire plusieurs objets à partir d'une scène virtuelle. Par exemple, dans les applications de mécanique des fluides, l'algorithme des *Marching cubes* [LC87] produit une surface polyédrique qui est une entité unique (même si la surface possède plusieurs composantes connexes) qui ne peut pas être éclatée en objets autonomes (excepté en une multitude de facettes triangulaires). D'un autre côté, la granularité de certaines bases de données à caractère scientifique est très fine, de sorte qu'il y a trop d'objets pour pouvoir les gérer de façon autonome. Par exemple, dans le cas de la visualisation d'ADN (Acide DésoxyriboNucléique), il est fréquent de devoir gérer plus de cinq millions de molécules A, C G ou T (Adénine, Cytosine, Guanine ou Thymine) ! Ainsi, j'ai orienté mes travaux vers la conception d'un système où chaque processus peut gérer plusieurs objets.

Quand DIVE, PaVRML ou certains jeux distribués sont lancés, il n'y a aucun moyen de charger, relancer ou détruire (i.e. : effacer de la mémoire) dynamiquement n'importe quelle partie de ses composants logiciels. Pour certains systèmes, on ne peut ajouter aucun composant sans recompiler l'application. Ainsi, les liens entre les composants sont préétablis et ne peuvent pas changer pendant une session de travail. Ceci pose de sérieux problèmes, comme par exemple de ne pas pouvoir changer de périphériques ou en introduire de nouveaux sans relancer une application de RV&A, voire sans la recompiler.

Les approches de type mémoire partagée supposent la transmission de gros volumes de données et ces données sont relativement normalisées (cf. PaVRML et les systèmes de type DIS). Avec certains de ces systèmes, le développeur d'applications ne peut pas toujours intervenir sur les fonctions d'affichage. Quand il peut le faire, ces fonctions sont généralement basées sur les bibliothèques graphiques de haut niveau. Par exemple, *OpenGL optimizer* ou *Performer* [Opt98, Per98] fournissent généralement des outils avancés d'affichage mais le contexte qui doit être utilisé pour les mettre en œuvre fait qu'ils sont difficiles à combiner avec de nouveaux paradigmes de gestion de scène.

2.2 Systèmes génériques de gestion d'événements

Echanger la cause des interactions, en l'occurrence des événements, plutôt que leurs effets, à savoir un flux de nouvelles données sur les objets modifiés, est typiquement le principe qui oppose l'approche événementielle à celle précédemment décrite. L'utilisation principale des « systèmes de notification d'événements » est de permettre la gestion de périphériques. Dans cet esprit, plusieurs modèles ont été proposés.

Il y a eu d'abord le modèle *PUSH*. Un serveur dessert plusieurs abonnés et dès qu'un périphérique produit un événement, le serveur l'envoie à tous ses abonnés. Ce genre de système exige une file d'attente d'événements à l'intérieur de chaque client. L'approche opposée, ou modèle *PULL* [PHSJ97], permet aux clients de ne demander des événements au serveur que lorsqu'ils le souhaitent. En conséquence, c'est le serveur qui gère la file d'attente des événements. Cependant, le principe même de ce modèle double le nombre

des transactions, puisque l'envoi de tout événement est précédé d'une requête du client au serveur. Une solution semblable est celle où les clients se connectent à un serveur central et font des demandes d'entrée-sortie. Ce genre d'approche peut être intéressant pour les retours haptiques si l'on souhaite que les clients puissent adresser les informations de retour sensoriels au périphérique haptique via le serveur (ce qui n'est pas l'approche courante puisque, pour réduire les latences, le pilote d'un périphérique dispose généralement d'un miroir de la base de données de la scène).

Dans ce qui suit, nous allons passer en revue les principaux systèmes de notification d'événements et étudier leurs propriétés en termes de réseau.

X Window

A la suite de Macintosh, un groupe du MIT développa le système *X Window* afin de doter *Unix* d'une interface graphique. Pour interagir correctement avec les fenêtres, ils ont introduit le concept d'événement gérant non seulement les périphériques d'entrée (clavier, souris) mais aussi des paramètres de fenêtrage : paramètres de création ou d'activité (cf. focus), modification de format... Dans ce qui suit, nous appelons *événement standard* tout événement dont la gestion est prévue dans le système X Window.

Avec un dispositif de type CAVE^{TM}, RAVE^{TM}, Reality Center^{TM}ou HMD^{TM}, il est évident que le clavier et la souris n'ont d'intérêt que pour la personne qui supervise, depuis un poste de contrôle, l'ensemble de l'Environnement Virtuel ainsi que l'application en cours d'utilisation. Dans le cas du fenêtrage, les événements X Window chargés de gérer les paramètres ne sont pas non plus appropriés, puisque le rendu graphique utilise l'ensemble de l'écran.

En fait, pour l'ergonomie de l'immersion virtuelle, les périphériques utiles à ce type d'interaction (cf. capteurs de position et de mouvement, gant numérique, périphérique haptique, systèmes de reconnaissance du geste ou de la parole, systèmes audio 3d...) ne produisent aucun événement standard de nos interfaces graphiques traditionnelles. D'autre part, X Window n'est pas censé contrôler des applications collaboratives. De ce point de vue, le système *OMG Corba* semble être une approche plus intéressante.

OMG Corba

L'OMG (*Object Management Group*), dont le but était de promouvoir les langages de modélisation de type objet, est à l'origine de la spécification Corba (*Common Object Request Broker Architecture*) [COR], cahier des charges d'un protocole réseau de haut niveau. Le but de ce protocole est de gérer des services, aussi bien locaux que via le réseau, sans que les clients n'aient à savoir où résident les fournisseurs de ces services. Une notion importante de la norme Corba est de rendre transparents les protocoles des réseaux sur lesquels s'effectuent les échanges d'informations.

Le *Corba Event Service* vise la gestion d'événements [Sch99]. Ce service utilise le concept de « canal d'événements » chargé de fournir des événements à tous les clients d'un service. Un canal d'événements est directionnel et peut être connecté à plusieurs

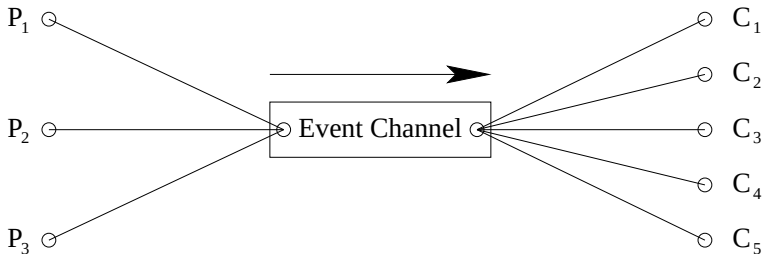

FIG. 2.1 – *Event channel de Corba.*

périphériques (voir figure 2.1). Toutefois plusieurs remarques s'imposent. Tout d'abord, le fait qu'il soit directionnel implique que les échanges entre un fournisseur et un consommateur ne peuvent pas s'inverser sans que soit créé un second canal d'événements. Or, il n'est pas rare en RV&A qu'un client doive envoyer des messages à ses fournisseurs, par exemple pour relancer un pilote de périphérique ou changer les états de celui-ci... Dans cet esprit, Corba ne fournit qu'un service de distribution d'événements. Il n'offre aucun service pour la gestion de périphériques ou de leurs pilotes (charger, lancer, détruire les pilotes inutilisés, modifier leurs états, annoncer de nouvelles connexions...). Ensuite, un canal d'événements ne peut pas avoir une structure hiérarchique. Une telle structure requiert qu'un module logiciel joue le rôle d'interface entre un canal père et des canaux fils. Un autre problème est le nombre des connexions. Si l'on veut pouvoir gérer de façon indépendante les pilotes de périphériques sous Corba, chaque périphérique doit avoir son propre canal d'événements, ce qui risque de d'entrainer une surcharge du réseau. De plus, il est possible de gérer tous les événements via une seule application contrôlant tous les périphériques (un gestionnaire de périphériques). Mais comme tous les périphériques n'intéressent pas forcément tous les clients, chaque périphérique doit posséder son propre canal d'événements. Il en résulte que ce gestionnaire de périphérique devra distribuer les événements dans différents canaux d'événements, tandis que les clients devront fusionner dans une file d'attente les événements qui les concernent pour les traiter séquentiellement. Si un des clients a besoin des événements de tous les périphériques, on arrive au schéma critique de la figure 2.2.

Un autre inconvénient du canal d'événements de Corba est qu'on ne peut généralement pas maîtriser les temps de latence dans les transferts d'événements. [HLS97] propose donc une solution temps réel qui, bien que basée sur le Corba Event Service, garantit que la distribution des événements s'opère en dessous d'un certain seuil de latence. Notons que si une faible latence dans la distribution des événements est nécessaire dans une approche où les modules d'une application de RV&A sont distribués sur plusieurs calculateurs, il n'est pas pour autant pertinent de rechercher un temps réel « dur[9] ». En outre, pour factoriser les développements, il est utile d'être multi-OS ou au moins POSIX[10] [Lew94]. Sur ce point

9. celui-ci implique généralement le changement de l'OS, or les applications actuelles de RV&A utilisent OpenGL et fonctionnent donc souvent sous IRIX (l'OS des machines SGI).

10. Portable Operating System interface for UnIX : Linux, SunOS, IRIX...

les auteurs semblent indiquer que leur implémentation de Corba est aussi multi-OS. Leur approche consiste à utiliser un canal d'événements plus actif. Cependant leur solution ne peut pas nous satisfaire, car cette activité consiste à faire une sorte de fusion « précoce » des événements, par exemple en les corrélant temporellement avant de les diffuser. Il en résulte que ces macro événements vont devoir être décomposés par les modules clients, donc en particulier par celui de fusion multimodale pour lequel ces corrélations ne seront pas forcément valides. En outre, les différents pilotes de périphériques ont une implémentation statique qui rend nécessaire de reconstruire le système dès qu'un nouveau périphérique doit être intégré.

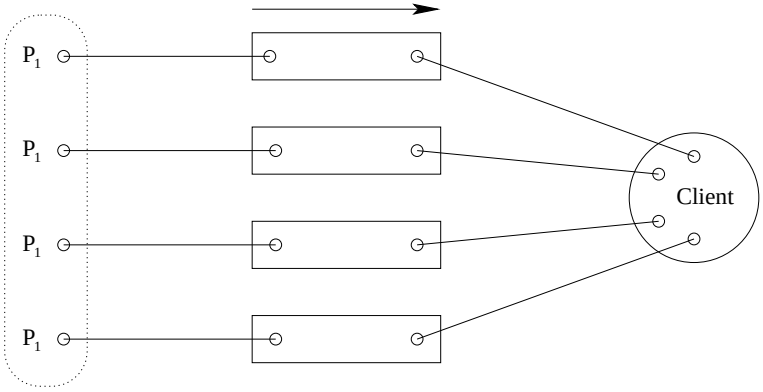

FIG. 2.2 – *Cas critique de l'utilisation de l'event channel de Corba.*

Corba présente donc plusieurs limitations. De façon plus générale, la représentation des données transmises n'est pas optimale. Sous Corba, l'en-tête de chaque lot de données comprend environ 100 octets, de sorte que la diffusion d'un simple événement d'un capteur 6 DDL (Degrés De Liberté) requiert 5 fois plus de place que ce qui est normalement nécessaire (puisque ces données requièrent seulement 24 octets).

De plus, l'annuaire de Corba gère trop d'informations pour pouvoir réaménager dynamiquement les connexions entre calculateurs en fonction de leur charge. Plus globalement, Corba fournit beaucoup trop d'outils. En fait, l'OS doit charger la totalité du ORB (Object Request Broker) alors qu'une application de RV&A ne requiert qu'une toute petite partie de Corba.

Autres systèmes génériques événementiels et conclusion

Quelques autres systèmes génériques à caractère événementiel existent.

Par exemple, [RGS95] propose une bibliothèque pour des communications temps réel entre processus où les échanges réseau utilisent le protocole multicast. Comme nous l'avons expliqué plus haut (cf. paragraphe B, page 65), ce protocole réseau manque de sécurité et de fiabilité, tandis que l'usage d'un OS temps réel pose des problèmes avec les bibliothèques les plus courantes en RV&A (cf. paragraphe 2.2, page 69). A l'opposé [BBHM95] propose

un serveur d'événements qui n'a besoin, ni d'un OS temps réel, ni du protocole de multicast. Cependant la solution proposée a deux défauts importants : d'une part l'expéditeur d'événements opère une fusion multimodale trop précoce (cf. paragraphe 2.2, page 70), d'autre part un callback[11] est utilisé pour chaque événement.

2.3 Approches orientées serveur d'événements de RV&A

X Window ne gère pas les applications distribuées et collaboratives, tandis que Corba est un système lourd à utiliser. Cependant, il existe pour la RV&A quelques approches à base de serveurs d'événements.

Systèmes couramment utilisés

Un travail intéressant est *VRjuggler* [JCNB00]. Ce système permet l'utilisation de périphériques à forte latence (comme les systèmes de reconnaissances de la parole et du geste) car il dispose d'un système de datation précis qui permet de tester ces latences. Au demeurant, VRjuggler n'est pas censé gérer ces latences lorsque les périphériques sont distribués sur un réseau.

Un autre système, *Trackd*, est un produit commercial de *VRCO*[12], est en fait le gestionnaire de périphériques des plus importantes bibliothèques commerciales[13] dédiées au développement d'applications de RV&A. Trackd s'appuie sur trois composants. Un gestionnaire de capteurs est associé à la machine où à laquelle ils sont connectés. A travers le réseau ou une liaison série RS 232, les données sont fournies à un processus démon[14] de la machine où réside en elle-même l'application de RV&A. Enfin, l'application se connecte au démon local à travers une mémoire partagée. Notons que Trackd tourne également sous Windows. Cependant, le principal défaut de trackd est l'absence de gestion centralisée des périphériques. Il en résulte qu'il est incapable de gérer une synchronisation temporelle entre les différents serveurs.

Une autre solution pour la gestion de capteurs est *DTK*[15] (*DIVERSE Tool Kit*) fourni avec l'API de DIVERSE[16]. Contrairement à Trackd, DTK est un logiciel libre (via une Lesser General Public License : LGPL) et son mécanisme de mémoire partagée est utilisable pour une application distribuée sur un réseau. Par contre, aucun portage ne semble prévu sur Windows.

11. fonction d'un module A, fournie par référence (i.e. pointeur de fonction) à un module B lors d'un processus d'initialisation, de sorte que le module B puisse exécuter quand il en a besoin la fonction du module A : un exemple type est l'appel d'une fonction d'un module « application » à partir d'une interaction gérée par un module « interface ».

12. http://www.vrco.com.

13. Il est par exemple utilisé par *CAVElib*^{TM}(une API de VRCO pour la gestion des dispositifs de type CAVE^{TM}), mais aussi par le *WorldToolKit* de *Sense8* (http://www.sense8.com/) et par *D-vise* de *Division*.

14. Un démon est un processus Unix qui tourne en tâche de fond et qui permet de gérér un service réseau (serveur web, serveur de mail, serveur ftp...) ou un service local (enregistreur chronologiques, gestionnaire de cartes PCMCIA ou de périphériques...).

15. http://www.diverse.vt.edu/dtk/

16. Une interface de programmation d'application comparable à CAVElib^{TM}développée par Virginia Tech.

DTK et Trackd sont des serveurs d'événements mais ne permettent pas de collaboration événementielle entre les modules distribués d'une application de RV&A. Ces serveurs sont associés à d'autres outils logiciels qui fournissent ce genre de service. Leur structure est très proche de ce que nous recherchions. Cependant, ces solutions ne sont pas optimisées pour pouvoir gérer n'importe quel type d'événement.

Les serveurs d'événements existants visent des périphériques au signal régulièrement échantillonné tels que les capteurs de position et d'orientation tout à fait ordinaires pour la RV&A. Ce type de serveur scrute de manière permanente la mémoire partagée. N'étant pas en situation d'attente d'événements, le traitement des événements aux occurrences apériodiques prend beaucoup de ressource CPU. De plus, les systèmes à base de mémoire partagée requièrent des mécanismes de verrouillage de la mémoire, ce qui n'est pas sans conséquence pour l'usage de la CPU par l'ensemble des processus actifs de l'ordinateur. En outre, dans une approche à base de mémoire partagée, chaque client a accès à tous les événements. Ceci complique le filtrage des événements qui, de notre point de vue, est utile pour garantir la qualité de distribution des événements.

Systèmes plus confidentiels

Enfin, des solutions ad hoc autour de X Window sont possibles. En l'occurrence, dans le contexte du projet MIX3D [BKG98], mes prédécesseurs ont développé le **MServeur** [BKG95] pour gérer quelques modalités non-standards (reconnaissance de la parole, interactions avec écran tactile). Cette extension du serveur X fournissait en particulier des outils pour dater les événements issus de processus à forte latence.

VRPN

VRPN [IHS+01] (Virtual Reality Peripheral Network) est un système « open source » assez récent. Il présente l'intérêt d'être proche de nos préoccupations. En effet, il implémente un système de synchronisation temporelle. De plus, il implémente également un pilote de périphérique vocal[17].

Cependant, il présente plusieurs limitations. Comme nous le verrons plus loin (section 6.5) le système de reconnaissance vocale ViaVoice n'est pas suffisamment performant pour permettre une fusion multimodale optimale. De plus, la structure client serveur de VRPN est limitée à un unique serveur et un unique client situés sur 2 machines. Dans ces conditions, la datation des événements n'est pas réalisé dans l'esprit d'une possible distribution des traitements des interactions sur plusieurs calculateurs.

Discussions

DTK et Trackd ne sont pas censés contrôler des événements datés. Or la datation des événements est une des premières choses à maîtriser afin de pouvoir fusionner des modalités issues de traitements à forte latence, tels que les systèmes de reconnaissance de

17. Par l'intermédiaire de ViaVoice

la parole ou de gestes. VRPN peut être intéressant, mais est limité à des architectures non distribuées. De son côté, le MServeur de MIX3D n'était qu'un pseudo serveur d'événements au sens où il n'a pas été conçu pour supporter des architectures en réseau : le MServeur et l'application cliente tournaient sur une même machine, tandis que les connexions avec les machines en charge des traitements à forte latence n'opéraient que via des liens RS 232. C'est cependant en partant de l'expérience du MServeur acquise au sein du LIMSI-CNRS que j'ai élaboré le cahier des charges d'un gestionnaire distribué de périphériques et d'événements pour le développement d'applications de RV&A à interface multimodale.

2.4 Conclusion sur les architectures distribuées existantes

A l'époque où cette étude bibliographique fut achevée, il apparaissait donc qu'aucune architecture logicielle ne donnait de solution globale pour la programmation d'applications de RV&A à interface multimodale. En effet, au-delà de la diversité des périphériques à gérer, il importe de pouvoir confier à des ressources calculatoires distinctes des traitements qui requièrent des calculateurs dédiés (calculs de simulation, retours haptiques, audio 3d) ou qui présentent des latences (reconnaissance de la parole et du geste). En outre, pour que le second type de traitement puisse contribuer à une gestion multimodale des interactions, le système distribué proposé doit pouvoir assurer une datation fiable des événements ce qui implique de gérer la synchronisation des horloges des différents calculateurs. S'ajoute à ce cahier des charges de base, plusieurs objectifs qui concourent à l'ergonomie des applications de RV&A basées sur une telle distribution de ressources. Ainsi, le gestionnaire des périphériques et de distribution des événements doit être dynamiquement configurable, pour pouvoir ajouter tout composant de périphérique sans nécessité de compiler à nouveau le système distribué, ni de relancer l'application pendant la session de travail. Il doit être compatible avec X Window, par exemple pour pouvoir développer un module de contrôle de l'application de RV&A doté lui-même d'une interface classique. Un tel module n'est pas destiné aux utilisateurs de l'application de RV&A, mais permet à un opérateur humain de superviser le bon fonctionnement de l'application et du dispositif matériel utilisé. Enfin ce système distribué doit pouvoir être porté sur divers protocoles réseau (exceptés ceux de type multicast pour les différentes raisons déjà exposées), tandis qu'il doit pouvoir tourner sur les principaux systèmes d'exploitation (POSIX aussi bien que Windows).

3 Principe général de l'architecture EVI3d

Dans la section précédente, nous avons montré qu'une interaction avancée en RV&A suppose, d'une part une gestion fine des événements issus de divers périphériques, pour certains associés à des processus de reconnaissance, mais exige également de prévoir la gestion de flux important de données. Pour ce qui concerne les événements, ceux-ci doivent être datés avec précision (sur l'expérience de MIX3D nous avions évalué que cette datation devait avoir une latence inférieure à $10ms$), tandis que l'usage de systèmes de reconnaissance requiert des ressources calculatoires indépendantes des calculateurs graphiques, haptiques ou audio.

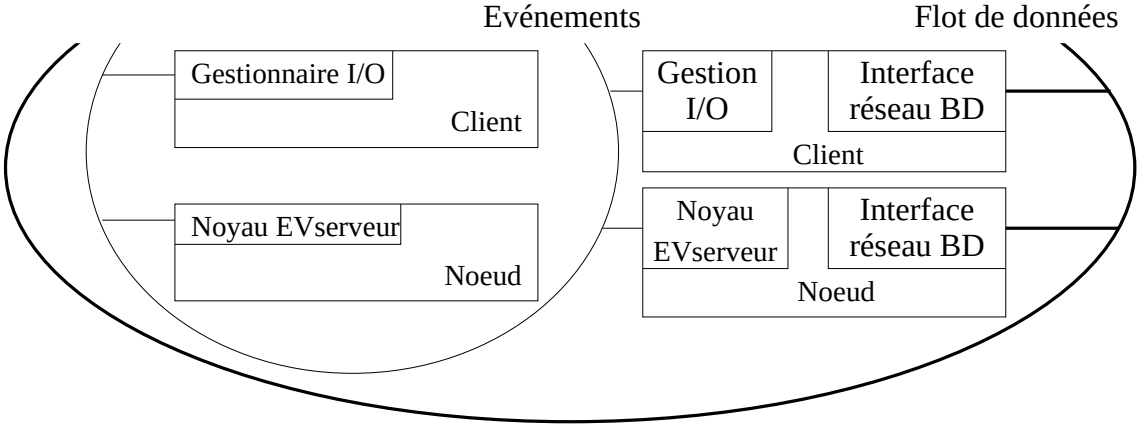

FIG. 2.3 – *Schéma général de l'architecture EVI3d.*

Au-delà de la gestion des périphériques de RV&A, l'architecture distribuée de la plateforme EVI3d vise à combiner les avantages d'une approche événementielle avec la gestion de flux de données (voir figure 2.3). En effet, un système distribué de RV&A doit pouvoir assurer deux types de services en matière d'échanges d'information : des transferts de grandes quantitées de données pour la création des miroirs complets ou partiels de bases de données sur différents calculateurs, mais aussi l'actualisation continue de ces données via des échanges d'informations sur les variations des données au sein de bases de données initiales (i.e. gestion de deltas d'information, dans une démarche analogue au format de fichier MPEG). En conséquence, l'architecture EVI3d contrôlera à terme deux canaux :

– ***Un réseau orienté « distribution d'événements »*** : Entièrement contrôlé par l'EVserveur (via les composants logiciels `Noyau de l'EVserveur` et `Gestionnaire E/S EV` dont nous parlerons aux paragraphes 4.3 et 4.4, page 80), ce premier canal connecte par principe tous les calculateurs utiles à une application de RV&A, tandis

que les informations qui y transitent sont structurées en événements. Il en résulte que ce canal peut être un réseau à faible bande passante (dans notre cas, un simple réseau Ethernet 10 mb/s). Certains de ces événements peuvent servir à adresser des requêtes à un module pour lui signifier d'actualiser les données de la BD miroir utile à un calculateur. Ce canal permet aussi d'échanger des messages de synchronisation pour connecter d'autres modules et organiser les transferts d'informations sur le second canal.

– ***Un réseau orienté « flux de données »*** : Ce second canal est supposé gérer un flot continu d'informations (ou *stream*) via un réseau haut débit. De natures diverses (BD 3d, signaux image ou audio...), ces informations ont surtout pour caractéristique d'être denses. Cette densité peut résulter de l'aspect dynamique de données qui en elles-mêmes peuvent être moyennement volumineuses (cf. figure 4.6, page 151). Au démarrage d'une application de RV&A, ce canal permet de distribuer sur certains calculateurs les miroirs de bases de données. Plus fondamentalement, ce canal doit pouvoir servir à adresser des lots de données destinés à actualiser lesdits miroirs de bases de données. Ceci est nécessaire lorsque ces actualisations requièrent trop de ressources CPU pour être opérées localement après un échange de requêtes via le canal événementiel (cf. solution évoquée précédemment).

La figure 2.3 montre donc les quatre composants de base de l'architecture EVI3d. Ces composants sont conçus pour résider sur des machines séparées. Dans cette architecture, seuls quelques modules logiciels d'une application de RV&A sont effectivement connectés via le réseau de type flux de données. A l'inverse, tous les modules d'une telle application communiquent ensemble par l'intermédiaire du réseau de distribution d'événements à l'aide de l'EVserveur.

Le fait de dissocier les deux types de données est plus pertinent comparé à l'approche utilisèe par la plupart des autres systèmes distribués dédiés à la Réalité Virtuelle. En effet, les événements ont un comportement différent des flots de données. Ces derniers ont un volume important, mais leur fréquence de rafraîchissement sera plutôt de l'ordre de 24 Hz. Cette fréquence est suffisante pour assurer une perception fluide du mouvement qui est typiquement ce que l'on recherche dans l'étude de comportement dynamique d'objets scientifiques. Inversement, la taille des événements est beaucoup plus petite (de l'ordre de 30 octets), mais nous avons vu (section C, parge 52) que certains systèmes requièrent une fréquence de 100Hz. Par ailleurs, au niveau de la latence pour l'adaptation de la stéréoscopie (voir section E, page 22), il est important que ces événements aient une fréquence élevée.

Ainsi, nous disposons de deux modes de transfert de données totalement antinomiques. C'est pourquoi, contrairement aux autres systèmes, nous désirons les dissocier avant de les transmettre sur le réseau. Cela permet une plus grande souplesse dans l'utilisation du réseau. Par exemple, pour le canal flux de données, nous pouvons préférer l'utilisation d'un réseau de type « raw » (réseau qui consiste à transmettre sur le médium l'ensemble de la base de donnée, sans notion de structuration de ces dernières sous la forme de paquets). Inversement, pour le canal événementiel, nous avons constaté qu'un réseau standard de

type *Ethernet* pouvait suffire pour la plupart des périphériques de Réalité Virtuelle. De plus, cette dichotomie évite de parasiter mutuellement les deux canaux.

Dans l'implémentation, nous pouvons faire transiter les deux canaux sur le même médium au travers de « canaux virtuels ». Cela évite de doubler systématiquement les systèmes réseau entre toutes les machines contribuant à la simulation. De plus, la présence d'un canal événementiel est obligatoire, afin de synchroniser les machines. Par contre, le canal flux de données n'est pas obligatoire. Par exemple, dans le cas de notre démonstrateur multimodal (voir section 4, page 149), les objets sont relativement simples et figés. Donc, il n'y a pas de calcul 3d distribué sur plusieurs machines.

Dans ce qui suit, je décrirai les aspects les plus importants de l'EVserveur, le gestionnaire distribué de périphériques et d'événements qui sert donc d' « épine dorsale » à l'architecture d'EVI3d. Par contre, l'articulation de l'EVserveur avec la gestion de flux de données ne sera qu'évoquée, nos recherches sur cette partie n'étant pas encore achevées.

4 La connexion événementielle - EVserveur

4.1 le gestionnaire de périphériques de RV&A

Gestion dynamique des périphériques

Tout d'abord, il convient que l'EVserveur soit en mesure d'accomplir toutes les tâches inhérentes à un gestionnaire de périphériques. En d'autres termes, il devait pouvoir charger, lancer, détruire et changer la configuration de n'importe quel pilote de périphérique. En raison de la nature a priori apériodique des événements, tous les pilotes de périphériques doivent fonctionner comme des tâches concurrentes. Ainsi, sur une machine donnée et au sein d'un processus dédié à l'EVserveur appelé `nœud EV`, chaque pilote de périphérique possède son propre *thread* (cf. note n^o 8 page 66). L'utilisation de processus de type thread en lieu et place de simples processus permet d'éviter les problèmes de partage de données en mémoire.

Un problème qui intervient souvent lors d'une simulation dans un environnement immersif est la stabilité des périphériques. Ainsi, il n'est pas rare qu'un périphérique se déconnecte ou se place dans une situation de blocage. De plus, nous pouvons avoir besoin d'ajouter un périphérique en cours de simulation. C'est pourquoi, une des contraintes du système est de pouvoir lancer des pilotes de périphériques ou le relancer sans avoir à interrompre un EVserveur en cours de fonctionnement. Pour ce faire, les pilotes de périphériques sont des objets partagés dynamiques basés sur la fonction système `dlopen()`. Ainsi, il est possible de développer et de tester de nouveaux pilotes de périphériques sans avoir à recompiler ou à relancer l'EVserveur. En résumé, un pilote de périphérique est un objet partagé, dynamiquement chargé par le EVserveur et conçu pour être lancé comme un *thread* séparé.

Une autre préoccupation est de pouvoir contrôler avec l'EVserveur des périphériques classiques de type clavier ou souris. Certaines applications doivent en effet pouvoir être utilisées indifféremment, avec les périphériques de RV&A d'un dispositif immersif ou sur un terminal graphique de bureau.

Pour disposer d'une gestion homogène des périphériques, l'extension *XTrap* [ACJ91] est utilisée de manière à récupérer les événements X avant qu'ils ne soient traités par le serveur X. Ceci permet à l'EVserveur de n'avoir qu'une seule connexion avec X et non pas une connexion avec chaque fenêtre X, puisqu'en effet le serveur X n'envoie un événement qu'à la fenêtre qui a le *focus*. Ce choix a l'avantage d'éviter de devoir faire jouer aux clients X un rôle actif vis-à-vis de l'EVserveur, de sorte que celui-ci est totalement indépendant des clients X. Par contre, le concept de focus a dû être ré-implémenté dans l'EVserveur, afin de pouvoir distinguer par exemple, si les événements gérés par l'EVserveur concernent l'espace immersif ou ceux d'une sous-application tierce (par exemple, les fenêtres qui permettent de superviser l'ensemble de l'application de RV).

Pour des raisons historiques (cf. le MServeur de MIX3D, section 2.3, page 72), le premier dispositif non standard géré avec l'EVserveur fut donc un système de reconnaissance de la parole (en l'occurrence, le système ViaVoice^{TM}). De notre point de vue, la voix est

l'une des modalités interactives les plus adaptées à la gestion de commandes ponctuelles. Cependant, les applications de RV&A exigent également des commandes continues (par exemple, pour contrôler les navigations virtuelles), pour lesquelles l'utilisation des interactions vocales n'est pas ergonomique (chose que mes prédécesseurs avaient en particulier mis en évidence sur MIX3D). Le deuxième type d'événement non standard contrôlé par l'EVserveur fut donc celui relatif aux capteurs de mouvements 6 DDL (en l'occurrence, le système Logitech^{TM}en technologie ultrason, puis les systèmes Polhemus^{TM}, Flock of Birds^{TM}, et MotionStar^{TM}en technologie électromagnétique).

Gestion des événements

Nous nous sommes rendus compte que les informations renvoyées par les capteurs de position 6DDL sont complètement disparates en termes d'arrangement comme en terme d'échelle. C'est ainsi que certains capteurs renvoient leurs valeurs en pouces pendant que d'autres les fournissent dans une unité non standard avec des cœfficients de conversion vers le système métrique ou des unités de mesures anglo-saxonnes. Le problème est plus délicat encore au niveau des orientations. En effet, même s'il y a un consensus pour l'usage des angles d'Euler, l'organisation des axes et l'application des angles sur les axes ainsi que l'ordre d'application des angles restent différents selon les périphériques. C'est pourquoi nous avons unifié tous les événements 6DDL. Dans un souci d'uniformisation, et afin de n'avoir aucun problème de protocole dans notre architecture Client-Serveur, nous avons défini une hiérarchie des classes d'événements en entrée (cf. figure 2.4). Outre les événements non-standards évoqués précédemment, cette hiérarchie de classes couvre toutes sortes de dispositifs tels que des souris 3d ou les signaux de différents modèles de gants numériques (Dataglove^{TM}, CyberGlove^{TM}, 5DT^{TM}). Ainsi, le développeur d'une application de RV&A n'a plus à tenir compte des caractéristiques matérielles et techniques relatives aux périphériques qu'il utilise.

Comme pour les périphériques d'entrée, ont trouve une hiérarchie des classes pour les événements en sortie (cf. figure 2.5). Cette hiérarchie n'inclut pas les affichages ou les effets sonores, car ces périphériques de sortie ont des signaux lourds qui ne sont pas destinés à transiter par l'EVserveur. De plus, dans cette figure, nous abordons les périphériques sous leur formes sémantiques. En effet, l'aspect retour interne ou externe n'est qu'un état d'un type de retour possible.

Une partie de cette hiérarchie concerne les événements issus des périphériques haptiques. Outre la distinction entre « retours d'efforts » et « sensations tactiles », cette hiérarchie prévoit une représentation spécifique pour les « retours sensoriels symboliques », à savoir ceux issus de systèmes de vibration dont disposent par exemple certaines manettes de jeux interactifs. Notons cependant que la gestion des périphériques haptiques dans l'architecture EVI3d est encore en cours d'évaluation, tandis qu'une partie des informations n'est pas supposée circuler via l'EVserveur (voir section 6, page 91).

Enfin, la classe « message » définit des événements utilisés pour la transmission d'informations entre plusieurs clients. Ce genre d'événement est en particulier utile pour la com-

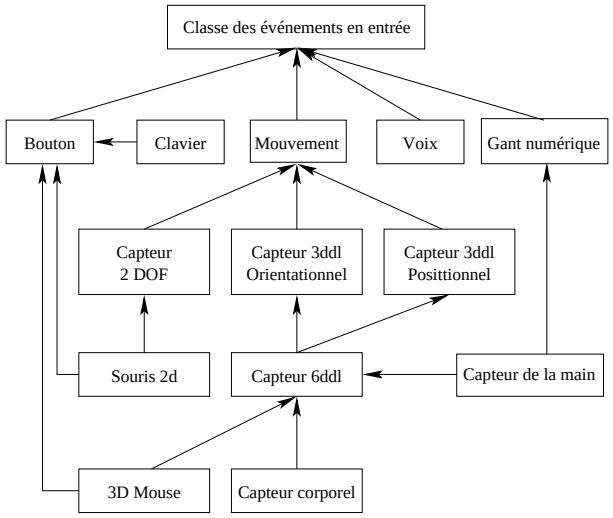

FIG. 2.4 – *Hiérarchie de classes d'événements en entrée.*

munication entre modules de plus haut niveau, comme par exemple, entre des périphériques de capture de mouvements et un système de reconnaissance de gestes, ou encore entre ces périphériques et divers modules monomodaux et un système de gestion multimodale des interactions (voir section 5, page 86).

4.2 Structure hiérarchique de l'EVserveur

Pour un client, l'EVserveur n'est qu'un seul et unique module logiciel représenté par le `nœud EV` auquel il est relié. Ceci est rendu possible grâce aux `Noyau de l'EVserveur` et `Gestionnaire d'entrée/sortie EV` qui masquent toutes les connexions et tous les besoins de transmission entre l'EVserveur et ses clients et dont nous parlerons dans les sections suivantes.

Les `nœuds EV` (figure 2.7) servent à distribuer l'EVserveur en plusieurs processus cohabitant sur le réseau. Cependant cette communication entre `nœuds EV` doit se faire avec un coût minimal. Pour maîtriser la charge du réseau les `nœuds EV` sont hiérarchiquement structurés. Excepté la racine de la hiérarchie, chaque `nœud EV` est client d'un `nœud EV` père. Par ailleurs, tout `nœud EV` achemine les événements des périphériques qu'il gère à ses clients directs, à savoir : son `nœud EV` père, ses `nœuds EV` fils et les clients de l'EVserveur directement connectés à ce `nœud EV`. Les avantages de cette structuration hiérarchique de l'EVserveur sont :

- une meilleure organisation de la diffusion d'information ;
- une taille optimale de la table de routage de chaque `nœud EV` ;
- la possibilité d'introduire dans chaque `nœud EV` des filtrages (comme par exemple des fonctions de firewall) ;

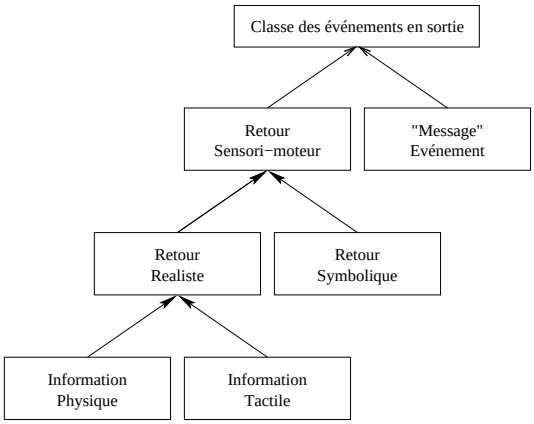

FIG. 2.5 – *Hiérarchie de classes d'événements en sortie.*

– une synchronisation temporelle plus facile des machines (voir le paragraphe 4.7, page 84).

4.3 Noyau de l'EVserveur

Une machine contribuant à une application de RV&A possède un et un seul `nœud EV` dès lors qu'elle doit gérer un ou plusieurs périphériques utiles à ladite application. Pour fournir des informations complètes et correctes aux clients de l'EVserveur, chaque `nœud EV` doit connaître à tout moment tous les services que les autres `nœuds EV` sont censés délivrer. Pour ce faire, chaque `nœud EV` possède un thread particulier qui constitue le noyau local de l'EVserveur appelé `Noyau de l'EVserveur` (voir figure 2.6). C'est le rôle du `Noyau de l'EVserveur` de chaque `nœud EV` que de distribuer cette information auprès des autres `nœuds EV`. Plus précisément le `Noyau de l'EVserveur` est censé gérer la distribution et la réception des événements, leur datation, la base de données qui identifie les clients du `nœud EV` et, plus globalement, administrer l'ensemble du `nœud EV` (cf. `administrateur EV serveur` dans la figure).

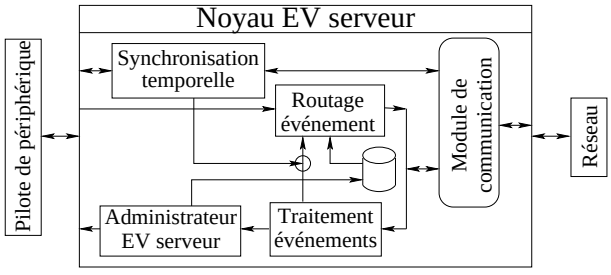

FIG. 2.6 – `noyau de l'EVserveur`.

Comme l'EVserveur est susceptible de distribuer à plusieurs clients un même événement, un système de filtrage a été introduit dans le `Routage d'événement` du `Noyau de l'EVserveur`. Pour des raisons de sécurité et comme la commande xhost de X Window, lorsqu'un client demande à recevoir un nouveau type d'événement, le `Noyau de l'EVserveur` consulte une liste dédiée pour vérifier si ce client est autorisé à recevoir les événements de ce type.

Mais le `Noyau de l'EVserveur` est également censé contrôler la qualité du service. Ainsi, si un `nœud EV` tombe en panne, la partie `Noyau de l'EVserveur` des `nœuds EV` qui lui étaient directement connectés supervise l'élection du remplaçant de ce `nœud EV` défaillant.

4.4 Gestionnaire d'entrée/sortie

D'un autre côté, chaque client du EVserveur inclut un composant logiciel spécial contrôlant la connexion du client à son `nœud EV`. Ce composant, appelé le `Gestionnaire E/S EV`, permet également au client d'envoyer des messages à n'importe quel pilote de périphérique ou client contrôlé ou relié à l'EVserveur. Ce type de communication est utile au module chargé de superviser les interactions (cf. client multimodal), afin qu'il puisse adresser des requêtes aux pilotes des périphériques (par exemple, pour interroger ou modifier l'état d'un périphérique). Mais une telle fonctionnalité est par ailleurs importante dans le contexte d'une architecture distribuée, quand on souhaite que plusieurs applications ou modules clients puissent échanger des informations afin de coopérer.

Les systèmes à base de notification d'événements utilisent généralement le principe de la fonction `XNextEvent()` de X Window. Située à l'intérieur d'une boucle infinie, cette fonction verrouille l'accès au processus traitant les événements, dans l'attente d'un quelconque événement. Quand un événement survient, cette fonction autorise ledit processus à traiter l'événement en question. A noter que cette fonction n'est à nouveau active que lorsque le processus a fini de traiter le dernier événement. En d'autres termes nous sommes en présence d'une notification différée des événements.

Pour une compatibilité avec les applications actuelles développées sous X Window, l'EVserveur peut fonctionner de la même manière. Cependant, le côté client de notre architecture est plus complexe. Dans une application développée sous l'EVserveur, la partie cliente est elle-même constituée de différents *threads*. Ainsi les clients peuvent supporter une notification immédiate ou différée des événements.

En présence d'une notification immédiate des événements, le système fournit les événements dès que ceux-ci sont produits. En d'autres termes, le système fonctionne dans un mode de préemption, en s'appropriant la tâche en cours de sorte à traiter immédiatement les événements. Inversement, une notification différée des événements consiste à les placer dans une file d'attente et à ne les fournir que lorsque la tâche les demande au serveur. Dans ce cas le système n'a aucun droit de préemption sur la tâche en cours : comme nous l'avons dit précédemment, c'est sur ce principe que le serveur X fonctionne. Dès qu'un événement arrive, le `Gestionnaire E/S EV` appelle une *fonction virtuelle* afin d'enregistrer cet événement dans une file d'attente. Ainsi, les événements sont disponibles

pour des appels différés. Pour que le client puisse travailler en mode immédiat, il suffit au développeur de l'application de spécialiser la classe `VEclient` en redéfinissant une *fonction virtuelle* héritée par cette spécialisation, de manière à pouvoir récupérer les événements avant qu'ils ne soient placés en file d'attente.

Disposer d'une notification différée des événements est essentielle. Par exemple, pendant l'affichage de la scène, les événements relatifs à la modification d'un objet ne doivent pas être ignorés au risque de rendre l'interaction difficile à faire : incohérence de l'affichage par rapport aux actions effectives de l'utilisateur. Une solution classique est de prévenir ce problème dans le codage même de l'application. Ce codage explicite devient inutile lorsque les événements sont enregistrés dans une file d'attente et qu'ils sont rappelés pour être automatiquement traités dès que l'affichage est terminé. A contrario, un avantage important de la notification immédiate des événements, est de permettre des interactions pendant que l'application est en train de gérer des processus longs à traiter. Une commande, par exemple supposée annuler un tel processus, sera traitée trop tardivement en présence d'une notification différée.

Notons enfin que l'utilisation d'une unique fonction de callback pour n'importe quel type d'événement (i.e. la fonction qui redéfinit la *fonction virtuelle* évoquée précédemment) est une approche particulièrement intéressante parceque polyvalente : le développeur est dispensé de programmer un callback spécifique pour chaque classe d'événement. De plus, le `Gestionnaire E/S EV` n'a pas à être mis à jour, si de nouveaux types de périphériques apparaissent sur le marché, ou encore si des pilotes de périphériques sont intégrés a posteriori dans l'EVserveur.

4.5 Le Module de Communication

Pour rendre génériques les transmissions réseau du EVserveur, les composants `Noyau de l'EVserveur` et `Gestionnaire E/S EV` doivent être complètement indépendants des considérations de réseau. Aussi un *Module de Communication* (MC) a été développé pour contrôler les transmissions. Ceci permet de s'adapter à l'évolution des matériels et des protocoles de réseau. Ainsi, le MC a été développé sur le réseau standard Ethernet 10 mb/s, mais est extensible aux protocoles de réseau tels que MPI ou Myrinet.

Les transmissions avec le MC sont standardisées :

- les adresses des clients et des `nœuds EV` ne sont pas directement connues de l'EVserveur. Le MC a une *table de conversion*. De fait, l'EVserveur est en lui-même indépendant du type du réseau.
- deux fonctions principales sont proposées par le MC :
  - l'une est employée par *Routage d'événement* pour envoyer un message à un ensemble de clients ;
  - l'autre, le *Traitement événements* lui-même, agit comme un callback pour que l'EVserveur puisse recevoir les messages issus des clients.
- les données échangées via ces deux fonctions principales sont « linéarisées » (i.e. données qui ont subi un processus de concaténation structurelle) pour permettre au

MC de gérer n'importe quelle classe d'événement.

Pour ce faire, le MC utilise deux sous-modules, l'un pour la « linéarisation » des données, l'autre pour l'interface de transmission.

Une solution pour le sous-module de linéarisation est d'enchaîner les données en faisant précéder chacune d'elles par un entête qui spécifie leur type. Cependant, de plus petits paquets sont possibles en utilisant des identifiants (ID) actifs sur les événements. Chaque type d'événement conserve un même ID pendant une session, de sorte que le sous-module de linéarisation n'a qu'à enchaîner les données dont les structures sont connues. L'idée est de permettre ainsi à l'EVserveur d'avoir toujours le même type de données à gérer en ce qui concerne la communication réseau. Pour pouvoir changer le sous-module de linéarisation rapidement, son codage est indépendant du MC. De plus, lors de la linéarisation de nombre, nous prenons soin de transformer les données pour faire en sorte que les données codées soient indépendante de la machine qui émet le message ainsi que de celles qui reçoivent ce message (notamment, différence entre le « little endian », le « big endian » et le « middle endian »[18]). Ainsi, nous avons pris le partie de transformer l'ensemble des nombres en « big endian ».

Le codage de l'interface de transmission est également indépendant du MC. Le premier but de cet autre sous-module est de déterminer le type de connexion à utiliser avec un autre nœud (client ou serveur). Par exemple, si un client est sur la même machine qu'un `nœud EV`, ce sous-module doit établir une connexion en mémoire partagée, tandis qu'il utilisera une connexion réseau dans l'autre cas. De plus, l'interface de transmission doit pouvoir changer dynamiquement les connexions avec les autres nœuds. Ceci est utile quand un `nœud EV` est en panne. Dans ce cas, tous les nœuds qui dépendent du nœud défaillant doivent réorienter leurs connexions vers celui qui le remplacera (cf. voir fin de la section 4.3, page 81).

4.6 Sûreté des connexions

Un aspect important de l'EVserveur est que la distribution des événements ne se fait pas via une seule connexion UDP/IP multicast, mais au contraire, comme CAVERNsoft, à l'aide de plusieurs connexions de TCP/IP. Comme nous l'avons signalé au paragraphe B page 65 de la section 2, le protocole UDP/IP n'est pas fiable. A l'inverse, le protocole TCP/IP évite les duplications ou les pertes de messages. D'autres problèmes se posent avec l'IP multicast, comme les problèmes de sécurité qu'induit ce protocole par la capacité de quiconque à s'abonner à un groupe de multicast. Avec IP multicast, pour que seuls les clients autorisés accèdent à un service donné, il faudrait disposer pour chaque client autant de connexions que d'abonnements autorisés pour celui-ci. Grâce à l'approche TCP/IP, un client de l'EVserveur ouvre seulement une connexion avec un `nœud EV` et ne la ferme

18. Ces trois différents types d'« endianness » correspondent au mode d'encodage des nombres par les processeurs. La différence intervient dans l'ordre des octets pour former un nombre sur plusieurs octets. Pour être indépendant des machines, il faut transformer tous les nombres dans un formalisme commun pour les transmettre via le réseau.

jamais. Tous les événements voyagent par cette connexion, tandis que la partie `Noyau de l'EVserveur` peut contrôler si un client est autorisé à s'abonner à la réception de tel ou tel type d'événements.

Dans notre système, nous ne contrôlons pas encore la qualité de service. En effet, nos applications sont actuellement déployées sur une petite échelle (six machines). Ainsi, les défaillances sont à l'heure actuelle relativement rares. C'est pourquoi, bien que faisant partie du cahier des charges initial, ce problème n'a pas été l'une de nos priorités au regard des problématiques de validation de l'architecture.

4.7 La synchronisation temporelle

Comme nous l'expliquerons plus loin (section 5, page 86), nous avons besoin de dater les événements et les messages qui circulent via l'EVserveur. Cependant, en présence d'une approche distribuée, cette datation se heurte à des problèmes de synchronisation temporelle.

Précision requise

De nos jours encore, les horloges internes de chaque ordinateur ne sont pas synchronisées et présentent des déviations spécifiques. En conséquence, l'EVserveur intègre un mécanisme de synchronisation temporelle pour assurer une datation fiable des événements et messages [Arn99]. Le cahier des charges initial était de fournir une datation dont la latence maximale devait être inférieure à $10ms$ sur un réseau Ethernet 10 mb/s. Cette précision était a priori suffisante car depuis notre étude bibliographique sur ce sujet à l'époque du projet MIX3D, tout indiquait qu'au-dessous de $10ms$ les utilisateurs humains ne peuvent pas distinguer deux événements[19].

Cependant ce seuil de latence n'est qu'un élément indicatif sur l'exactitude que doit avoir la datation des événements. En effet, ce seuil concerne la sensation qu'a l'utilisateur vis-à-vis du traitement temps réel de ses interactions. Si la datation a une latence inférieure à ce seuil, cela ne signifie pas que les événements parviennent à l'utilisateur en moins de $10ms$. Il faut également que la distribution et le traitement des événements soient réalités dans cet intervalle temporel.

Distribution de la datation

Pour traiter efficacement ces événements, toute la stratégie de l'architecture EVI3d est de distribuer ces traitements sur des calculateurs dédiés. D'un autre côté, l'ensemble de la partie EVserveur de cette architecture est pensé pour optimiser les délais de transmission des informations. C'est pourquoi il convenait, non seulement d'abaisser le plus possible le temps de latence de la synchronisation temporelle des calculateurs, mais aussi de s'assurer,

19. Il convient de noter que les modalités haptiques et audio 3d, ou le système vestibulaire peuvent être beaucoup plus exigeants que $10ms$ de latence (cf. comptes-rendues des journées de l'action spécifique « Réalité Virtuelle et Cognition » du département STIC du CNRS).

au-delà de l'intégration de cette contrainte dans le cahier des charges de l'EVserveur, que la transmission des événements était bien largement inférieure à ce seuil.

Le principe du système de synchronisation est d'obtenir un temps universel (ou UT) à partir d'un « maître ». La hiérarchie de `nœuds EV` étant en elle-même une approche « maître/esclave », cette hiérarchie est utilisée pour synchroniser temporellement les `nœuds EV`, sans qu'ils aient besoin de connaître le maître qui contrôle ce temps universel.

Outre le principe de cette synchronisation, la précision de la datation des événements résulte aussi d'un accès optimal (i.e. direct et « ad hoc ») à la partie matérielle des ordinateurs que nous utilisons (en l'occurrence, des stations SGI et des PCs sous Windows ou Linux). Des matériels spécifiques[20] seraient nécessaires pour une meilleure précision que celle permise par notre mécanisme de synchronisation.

Du fait de la localisation du système de synchronisation temporelle dans la partie `Noyau de l'EVserveur` de chaque `nœud EV`, tout événement d'un périphérique est directement daté par le `nœud EV` où réside le pilote dudit périphérique, alors qu'un message entre clients est daté lors de son passage dans le premier `nœud EV` qui va le prendre en charge.

Précision des données

L'unité de base de notre système de synchronisation est la microseconde : c'est l'unité logique la plus proche autorisant de descendre en dessous de la milliseconde en termes de précision. Au niveau de l'implémentation, nous avons opté pour des entiers sur 64 bits au lieu d'entiers standards sur 32 bits. En utilisant la microseconde comme unité de base, un entier sur 32 bits permet une autonomie de seulement 1 heure 11 minutes 34 secondes 967 ms 296 μs, alors qu'un entier 64 nits donne une autonomie de plusieurs centaines de milliers d'année !

Cependant, par définition, un entier 64 bits est plus lourd en termes de bande passante réseau. C'est d'autant plus lourd que nos événements comportent deux informations temporelles : les dates de début et de fin de l'événement. Là où une heure (codée sur 32 bits) ne suffit pas pour définir la base de temps, elle suffit amplement pour définir la durée de l'événement. Ainsi, pour les événements nous ne transmettons que la date de départ en 64 bits. La date de fin d'un événement est transmise de façon implicite sous la forme d'un delta de temps par rapport au début du-dit événement, delta de temps qui peut se satisfaire d'un codage sur 32 bits.

20. Par exemple, un périphérique susceptible de recueillir l'UT que savent gérer les satellites du système GPS (*Global Positioning System*).

5 La fusion multimodale des événements

Nous nous sommes basés sur le système de fusion multimodal LIMSI-Draw développé par Yacine Bellik [Bel95] lors de sa thèse pour développer notre système de fusion multimodale. Il est constitué de trois parties : les *interpréteurs*, une *file d'attente* et un *gestionnaire de fusion*.

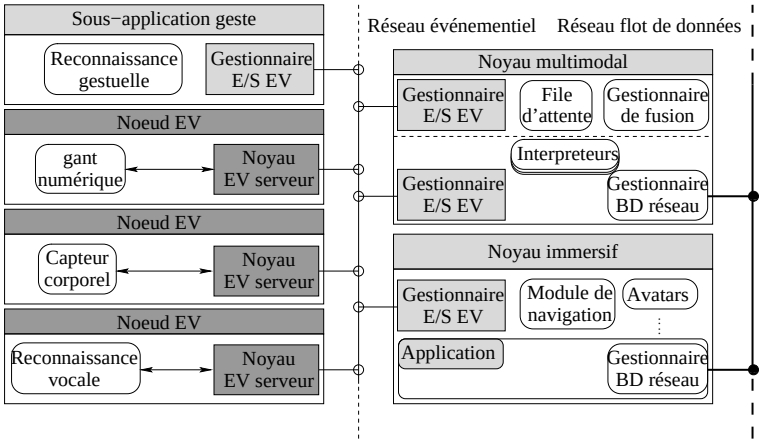

FIG. 2.7 – *Schéma de l'instanciation d'une application utilisant la fusion multimodale.*

5.1 Interprétation

Comme nous le verrons un peu plus loin, le processus de fusion multimodale est indépendant de l'application. En conséquence, les événements qui lui sont fournis doivent non seulement être correctement datés, mais doivent également être porteurs d'informations de haut niveau (i.e. relatives à l'application et à la sémantique de l'interaction). Comme cela a déjà été expliqué section 4.7, chaque événement est daté dès qu'il traverse le `Noyau de l'EVserveur` qui pilote le périphérique associé. Pour être sémantiquement « augmentés », ces événements sont pré-traités par des *interpréteurs* qui dépendent de chaque modalité.

La figure 2.7 montre comment nous instancions une application utilisant la fusion multimodale. Ainsi, un certain nombre de périphériques sont reliés au système par l'intermédiaire de `nœuds EV`. Ils envoient notamment les événements au noyau immersif, aux systèmes de reconnaissance évolués (ici, la reconnaissance gestuelle) ainsi qu'au noyau multimodal. Ce dernier, après avoir généré les événements sémantiquement augmentés, les renverra à l'application par l'intermédiaire de son `nœud EV`au noyau géométrique. Dans ce cas, le noyau multimodal agit comme le système de reconnaissance gestuelle : il récupère l'information sur les périphériques par l'intermédiaire de l'EVserveur et les renvoie, toujours au travers de ce dernier, sur le réseau à destination du noyau graphique. En outre

le noyau multimodal récupère également les informations issues de la reconnaissance gestuelle.

La fonction des interpréteurs est de traduire les événements de bas niveau produits par les périphériques (coordonnées 3d renvoyées par les capteurs de position, chaînes de caractères produites par le système de reconnaissance de la parole...), en des informations de plus haut niveau susceptibles d'être exploitées par l'application.

Par exemple, un geste de désignation, composé d'un événement capteur de position 6DDL, d'un événement gant et d'un événement reconnaissance gestuelle, peut avoir plusieurs significations différentes. En effet, il y a d'abord la configuration propre de la main. La reconnaissance d'un geste de pointé peut, par exemple, activer un pointeur laser (laser beam). En analysant alors la position de chaque objet dans la scène par rapport à ce pointeur, nous pouvons en déduire que le geste désigne tel ou tel objet. Ces différentes significations sont issues de l'interprétation de l'événement gestuel. Chaque objet intersecté générera sa propre interprétation. Cependant, il faut ajouter des informations supplémentaires telles que, par exemple, la profondeur de chaque objet. En effet, selon la commande, on peut avoir à fusionner avec celle-ci des informations relatives au premier objet intersecté ou aux autres objets pointés par le geste en question.

Pour revenir au cas général, quand des événements ont été interprétés et datés, l'information est envoyée au système multimodal de fusion. Un interpréteur est donc associé à chaque modalité d'entrée. Chaque interpréteur possède son propre *modèle de langage* ainsi qu'un *domaine d'information* spécifique pour réaliser la transformation des événements :

- ***les modèles de langage sont statiques.*** Ils contiennent un ensemble d'informations sémantiques invariantes nécessaires pour opérer l'interprétation. Pour un gant numérique, ce sera le modèle de langage gestuel composé de la sémantique de chaque geste du langage. Pour un système de reconnaissance de la parole, ce sera un modèle du langage parlé contenant la signification de chaque mot ou phrase.
- ***le domaine d'information est dynamique.*** Il s'agit de la description de l'environnement de l'application. Cet environnement évolue dans le temps en fonction de l'avancement de la tâche. Il contient une description de chaque « objet » de l'application. Le domaine d'information est dynamiquement mis à jour dès que la BD des objets est modifiée. Pour permettre une telle mise à jour, les interpréteurs peuvent avoir besoin d'une connexion sur le réseau orienté « flux de données ».

D'autres modules peuvent aussi être destinataires des événements de bas niveau produits par les pilotes des périphériques. C'est typiquement le cas des retours « lexicaux », tels que la visualisation de l'avatar d'une main. Mais plus fondamentalement, certaines modalités exigent un module de reconnaissance intermédiaire entre le pilote du périphérique et le module d'interprétation. Ainsi, dans l'architecture EVI3d, la structure générique d'une modalité est composée de trois modules : le pilote de périphérique en lui-même, un processus de reconnaissance et un interpréteur. La figure 2.8 montre le pré-traitement entier des événements avant qu'ils soient adressés à un module de fusion multimodale. Un point

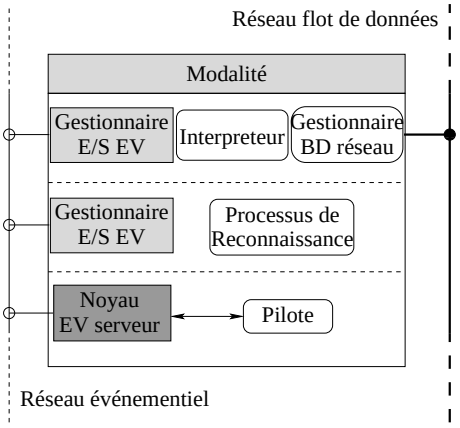

FIG. 2.8 – *Structure générique d'une modalité dans l'architecture EVI3d : les 3 niveaux de pré-traitement d'un événement avant la fusion multimodale.*

important à noter est que, grâce à l'architecture EVI3d, chaque composant de ce processus a la capacité de résider sur différentes machines (cf. lignes pointillées).

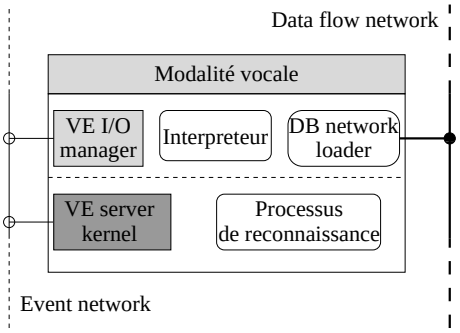

FIG. 2.9 – *Prétraitement de la modalité vocale.*

A l'inverse, certains de ces composants peuvent être fusionnés ou divisés, par exemple en fonction du contexte matériel rencontré :

– ***Reconnaissance de la parole.*** L'information bas niveau fournie par le pilote de périphérique est un signal audio qui, dans le principe, devrait transiter par un réseau de type « flux de données ». Cependant, il n'est souvent pas possible de contrôler le signal vocal de ces systèmes. Dans ce cas, le pilote du périphérique et le processus de reconnaissance sont combinés au sein d'un même composant qui ne fournit que les mots reconnus. La figure 2.9 montre la structure logicielle du pré-traitement pour cette modalité. C'est suivant ce schéma que nos démonstrateurs sont en mesure d'utiliser ViaVoice^{TM}. Notre version de ce système de reconnaissance de la parole ne

tournant que sous Windows, le `Noyau de l'EVserveur` a donc été porté et validé sous cet OS (en sus de IRIX et Linux).

– ***Reconnaissance de gestes.*** La reconnaissance de gestes requiert la gestion simultanée de plusieurs périphériques (figure 2.10). Pour chaque main, il convient par exemple de piloter un gant numérique et un capteur 6 DDL de mouvements. Pour reconnaître certains types de gestes (cf. main « dominante » / main « dominée ») le système de reconnaissance doit disposer des signaux relatifs aux deux mains. Actuellement, ce schéma a été validé sur la reconnaissance de gestes isolés d'une seule main, module développé au sein du groupe G&I sur la base des travaux de *Rubine* [Rub91].

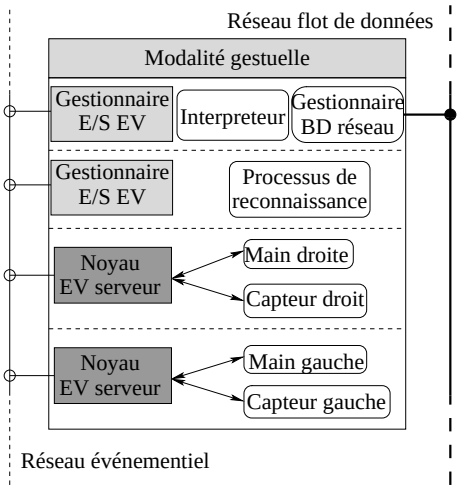

FIG. 2.10 – *Prétraitement de la modalité gestuelle.*

5.2 La file d'attente des événements

Une file d'attente reçoit l'information des interpréteurs de toutes les modalités. Cette information est classée suivant les trois catégories d'un modèle d'interaction utilisateur : <*commande*, *arguments*, *valeurs*>. Elle est de plus triée en fonction de la date de début de production de cette information.

5.3 Le gestionnaire de fusion

Le gestionnaire de fusion analyse l'information contenue dans la file d'attente dans le but de reconstruire la commande initiale de l'utilisateur puis de lancer son exécution. Le processus consiste alors à identifier une commande dans la file d'attente, tandis que le *modèle d'application* (cf. figure 2.11) en donne la description. Après l'analyse des autres informations dans la file d'attente, le gestionnaire assigne des valeurs aux arguments de la

commande. Quand tous les arguments sont évalués, le gestionnaire lance l'exécution de la commande si celle-ci est valide.

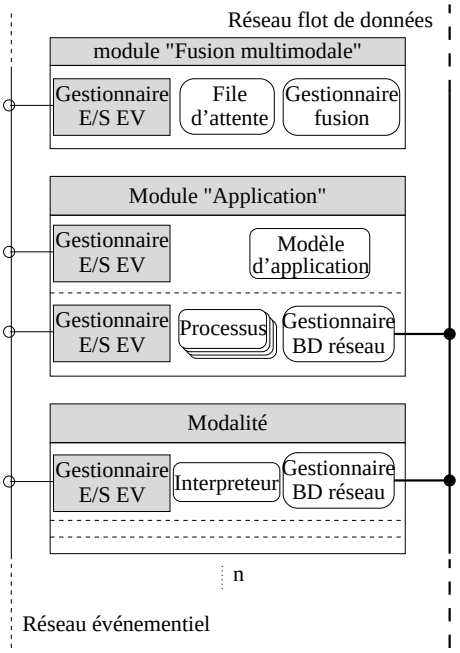

FIG. 2.11 – *Le système multimodal dans l'architecture d'EVI3d*

Le gestionnaire de fusion manipule dans l'ordre chronologique les informations contenues dans la file d'attente. Le processus de fusion utilise un ensemble de règles définies à partir de l'analyse des différentes configurations possibles qui peuvent apparaître dans la file d'attente. Le résultat du traitement dépend du type de l'information courante, de l'état de l'information précédente et de *conditions d'appariement*.

Un traitement clef opéré par le gestionnaire de fusion est donc l'assignation des bonnes valeurs aux bons arguments. L'appariement résulte du respect de trois contraintes :

- la ***complémentarité logique*** des informations : par exemple, on ne peut pas apparier deux valeurs ou une valeur et une commande, tandis que a contrario, il est possible d'apparier une valeur et un argument ;
- la ***compatibilité de type*** : pour apparier un argument et une valeur, le type de la valeur doit être compatible avec le type de l'argument ;
- la ***proximité temporelle*** : une valeur et un argument peuvent être appariés si leurs dates de production sont dans une fenêtre temporelle assez étroite.

La figure 2.11 montre l'intégration du processus de fusion multimodale dans l'architecture d'EVI3d. Chaque entité connectée au réseau orienté « distribution d'événements » peut être vue en tant que processus distinct. De plus, ces différentes entités peuvent résider sur différentes machines.

6 Gestion des retours haptiques

Contrairement aux retours visuels, qui ne requièrent qu'une cadence de 24 Hz (principalement dûe à la persistance rétinienne), les retours haptiques exigent une fréquence de 1 kHz. Chaque itération est un ensemble de calculs qui inclut la distance entre la position virtuelle du périphérique et chaque objet de la scène.

VRPN est un des premiers pilotes de périphériques gérés en réseau qui permet une telle interaction. Sa gestion haptique est basée sur l'utilisation des bibliothèques comme : GHOST (General Haptic Open Software Toolkit) distribué par SenSable pour ses stylets haptiques (cf. Phantom^{TM}), VHT (Virtual Hand Toolkit) distribué par VTi (Virtual Technologies) pour son gant numérique à exosquelette (cf. CyberGrasp^{TM}) et H-COLLIDE [GLGT99].

Les interactions entre le client et les pilotes utilisent la même connexion pour tous les périphériques d'une même application. Ainsi, la scène entière est contrôlée en utilisant cette connexion. Ce système fonctionne parfaitement dans le cas d'une scène statique (i.e. objets aux positionnements rigides) dont les interactions de modification se limitent à des mouvements globaux qui affectent l'ensemble de la scène. Cependant, dans le cas des données dynamiques, le transfert de ce type d'information sur un seul réseau induit des risques de surcharge (a fortiori sur un réseau à faible bande passante).

Le double canal que propose l'architecture EVI3d offre une solution générique pour la gestion distribuée des périphériques haptiques. Ainsi, dans le cas de petites mises à jour, telles que des mouvements d'une scène statique, les informations sont transmises via le EV-serveur par le réseau orienté « distribution d'événements ». Inversement, quand la scène subit des modifications importantes, du fait de l'interaction haptique ou plus fondamentalement du fait de son caractère dynamique propre (cf. écoulements turbulents), le pilote peut alors souscrire à un groupe multicast d'un calculateur de simulation qui lui fournira ces données dynamiques via un réseau de type « flux de données » (cf. `Gestionnaire BD réseau`).

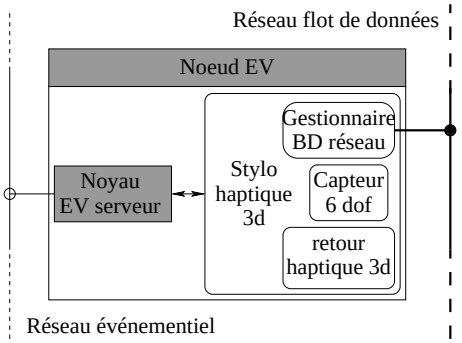

FIG. 2.12 – *Intégration du périphérique de type Phantom^{TM}.*

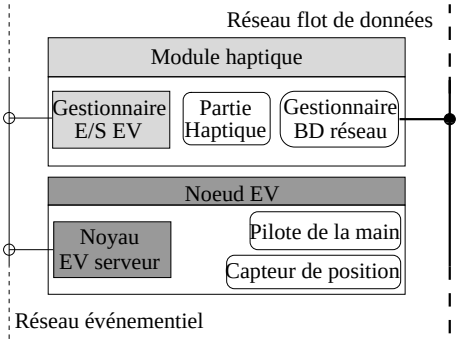

FIG. 2.13 – *Intégration du périphérique de type CyberGrasp^{TM}.*

La figure 2.12 montre que l'architecture EVI3d permet d'inclure directement le DB network loader dans le pilote d'un `nœud EV`, comme c'est le cas pour le pilote du périphérique Phantom$^{\text{TM}}$. De plus, l'architecture EVI3d permet de gérer le cas où la partie haptique doit être physiquement dissociée de la partie qui gère les entrées. Par exemple, avec un périphérique de type CyberGrasp$^{\text{TM}}$, le EVserveur permet la communication entre le pilote d'un gant numérique (les événements de flexion et de position de la main, en partie basse du `nœud EV` de la figure 2.13) et le pilote des retours d'efforts de l'exosquelette. Ainsi, le pilote du gant numérique peut être situé sur l'ordinateur qui gère également un processus de reconnaissance de gestes ainsi que son interpréteur de modalité associé (cf. figure 2.10), alors que les retours haptiques peuvent être gérés par une autre machine pour fournir la cadence nécessaire à la commande robotique.

7 Conclusion sur l'architecture EVI3d

Au-delà de l'implémentation de l'architecture EVI3d, le travail principal effectué lors de ma thèse a été de réflechir à des systèmes interactifs toujours plus « naturels ». Les librairies actuelles à base de mémoire partagée (DIS, HLA, NPSNET, ...) ne sont généralement pas optimisées dans ce cadre puique leur point fort est d'échanger un volume de données important. Les librairies à base d'événements sont par contre plus appropriées à ce type de traitement. Cependant, la plupart des librairies, dont les deux plus importantes, trackd et VRjuggler, ne permettent pas de faire intervenir un paramètre temporel dans leur gestion des événements. La seule librairie qui déroge à la règle est VRPN. Cependant, elle ne permet pas la répartition des périphériques sur plusieurs machines. Cela pose problème pour maintenir une horloge commune à l'ensemble des systèmes.

Ainsi, l'apport majeur de la librairie EVI3d est sa capacité à distribuer l'ensemble des périphériques sur un ensemble hétérogène de machines, tout en conservant une base de temps commune. De plus, son double canal événement/flux de données apporte une nouvelle approche dans la gestion des environnements immersifs. Cette approche est motivée par l'aspect antinomique des deux types d'information. Elle permet de dédier chaque type d'information à un canal optimisé à cet effet. Cependant, conscient de l'aspect pratique, il est possible de faire transiter les deux modes par un seul canal. De la même manière, il est aussi d'inhiber le canal par flux de données.

Bien que nous ayons pensé intégralement cette architecture dans l'optique du double canal, nous n'avons pas encore pu implémenter le canal flux de données. Cependant, nous travaillons activement sur ce sujet et, sur la base du cahier des charges à respecter par ce canal que nous avons conçu, nous avons déjà quelques pistes concernant les réseaux possibles. Ces considérations seront vues en détail dans la section 6, page 154 du chapitre sur les perspectives.

En ligne de mire, nous avions la réalisation d'un système autorisant la mise en place de paradigmes permettant une interaction « naturelle » avec le monde virtuel. Nous avons commencé avec la réalisation d'un démonstrateur multimodal que j'exposerai dans la section 4 du chapitre consacré aux validations et aux évaluations. Ces interactions « naturelles » doivent être validées par des études ergonomiques et cognitives sur notre système. Cela fait également partie de nos perspectives.

Chapitre 3

Noyau géométrique de EVI3d

Sommaire

1 Introduction

Le second travail important effectué lors de ma thèse est la conception d'un noyau géométrique. Ce dernier permet notamment le développement d'applications de RV sur les dispositifs immersifs les plus variés.

Ce travail fut motivé par le fait qu'aucune librairie actuellement disponible ne paraît facilement interfaçable avec l'architecture EVI3d. De plus, dans le cadre d'études cognitives, il faut pouvoir contrôler un certain nombre de paramètres. Parmi ceux-ci, il faut être capable de permuter les affichages afin d'obtenir une fréquence de rafraîchissement constante.

Ainsi, après avoir présenté les travaux actuels dans ce domaine et leurs connexions avec les systèmes de gestion des périphériques décrits dans le chapitre précédent, je présenterai notre propre environnement. Je montrerai comment il s'articule avec ou sans l'EVserveur. Ensuite je détaillerai l'interfaçage avec une application standard. Puis, j'aborderai les problèmes de calibration des différents systèmes de référentiels. Je consacrerai une partie de ce chapitre au système de contrôle des navigations virtuelles que nous avons étudié [2]. Je ferai le point sur les utilitaires annexes mis en place pour assister le développeur. Enfin, je conclurai en décrivant un démonstrateur multimodal développé pour valider la plateforme logicielle EVI3d.

2 Travaux existants

Un point fondamental que doit traiter le système de pilotage géométrique d'un environnement immersif est la gestion des jonctions entre les plans images. De façon plus générale, c'est la gestion d'un système graphique multi-fenêtre. Cette gestion doit être optimale afin de permettre en particulier la stéréoscopie.

2.1 Librairies graphiques

Librairies graphiques standard

A la base des librairies graphiques actuelles, se trouvent deux principaux standards industriels : OpenGL [1] et DirectX [2].

OpenGL fut développé dans les années 80 par la société Silicon Graphics. La primitive de base est la facette [WNDS99, WNDS98]. Ainsi, quelle que soit sa complexité, tout objet virtuel doit pouvoir être approximé de façon polyédrique. A l'origine, OpenGL (ou plutôt GL) était la librairie servant d'interface avec les cartes graphique des stations graphiques SGI. Il s'agit donc d'un langage de commande qui permet de piloter ces cartes. Depuis que OpenGL est devenue autonome, elle n'est plus directement reliée aux cartes graphiques de SGI. Par contre, la contrainte majeure actuellement imposée à OpenGL est que toutes ses fonctionnalités doivent être réalisables par des algorithmes qui peuvent être implémentés en *hardware* de sorte à permettre un rendu temps réel.

La librairie DirectX est une librairie proposée par Microsoft. Elle est basée sur des objets plus haut niveau comme par exemple les splines. Cependant, le défaut majeur de cette librairie est qu'elle n'est disponible que pour les systèmes de type Windows.

Librairies à graphe de scène intégré

Il convient de noter que certaines librairies graphiques offrent aujourd'hui des fonctionnalités de gestion des niveaux de détails. Par exemple, *Open Inventor* (librairie initialement créée par Silicon Graphics) représente la scène à l'aide d'un graphe dont les nœuds décrivent les propriétés de leur sous-graphe associé [OIv94]. Au sein de ce graphe, le nœud `SoLevelOfDetails` permet de choisir un sous-graphe parmi plusieurs en fonction de la taille de la boîte englobante des objets définis par les sous-graphes.

De son côté, la librairie *Performer* de Silicon Graphics, conçue pour le rendu temps réel sur des systèmes multiprocesseurs [Per98, Per94], est également basée sur un graphe avec différents types de nœuds. Ainsi, le nœud `pfLOD` est dédié à la sélection des représentations en fonction de la distance à l'observateur, de sa taille dans l'espace image et du champ de vision. Le nœud `pfMorph` permet de faire un morphing entre deux niveaux de détails successifs, de manière à rendre la transition invisible en faisant cœxister deux modèles

1. http://www.opengl.org
2. http://www.microsoft.com/windows/directx

pendant la phase de transition. Le nœud `pfBillboard` gère des imposteurs (facettes texturées) constamment orientés vers l'observateur qui se substituent à un objet 3d complexe. Cette librairie offre de plus des *surfaces actives* ou ASD (*Active Surface Definition*) pour structurer les niveaux de détails de terrains et permet leur affichage progressif et non uniforme (plus de détails dans les parties proches de l'observateur) [Per98]. Enfin, un point important de Performer est sa capacité à évoluer. En effet, cette librairie permet d'intégrer des nœuds propres au client dans son graphe de scène. Ces nœuds peuvent prendre tout type de données.

Même si ces librairies utilisent la cohérence temporelle, il apparaît néanmoins que les outils qu'elles proposent ont souvent un impact sur le temps de calcul. Dans l'ensemble, ces fonctionnalités n'apportent un gain réel que lorsque les simplifications sont importantes.

Enfin, elles ont le grand défaut de n'exister que pour un ensemble assez restreint de plate-formes (Performer n'existe, par exemple, que pour IRIX [3] et pour Linux). Leur prix et le fait que ce soit des produits fermés (caractéristiques fixes et non explicitées) constituent aussi un frein pour s'en servir comme support à des activités de recherche.

2.2 Gestion d'un système ayant plusieurs plans graphiques

Plusieurs systèmes, généralement associés aux librairies citées précédemment et aux systèmes de gestion des périphériques décrits dans le chapitre précédent permettent de gérer des environnements immersifs.

A Systèmes Commerciaux

Les premières librairies à avoir vu le jour ont été développées par les pionniers de la Réalité Virtuelle. Cependant, elles sont devenues payantes.

CAVElib

C'est la librairie la plus répandue à l'heure actuelle dans le milieu industriel. Etant payante, c'est l'une des premières à garantir le support technique. Cependant, elle est particulièrement onéreuse.

Cette librairie est basée sur l'usage des processus (ou *process*) pour séparer les tâches de rendu. Le principal problème de ce type de solution est qu'il faut en permanence synchroniser les différents espaces mémoire. Par exemple, des données aussi triviales que la position de l'observateur, ou des composants de son avatar (position et configuration de chaque main, du corps, ...) doivent être « recopiés » pour chaque *process* qui les utilise. Cela est une bonne chose car nous verrons plus loin que cela permet de « figer » la scène à redessiner pour l'ensemble des contextes.

Jusqu'à ce jour, la seule interface disponible pour connecter des périphériques pour CAVElib est Trackd. Ainsi, CAVElib se hérite de tous les défauts de ce gestionnaire de périphériques (voir section 2.3, page 71).

3. IRIX est l'OS des machines SGI

Vega

Vega est une librairie commerciale, qui possède un certain nombre d'atouts. L'un de ses avantages est de pouvoir gérer des grappes de PCs. En effet, nous avons vu précédemment qu'une grappe de PCs peut être une solution économique pour piloter des environnements immersifs. Cependant, il faut être capable de répartir l'information pertinente entre tous les ordinateurs composant la grappe, ce qui est le cas de Vega.

Un autre point fondamental de Vega est sa capacité à gérer la synthèse de sons spatialisés. En effet, pour nous, la composante visuelle n'est pas la seule permettant d'interagir avec le monde virtuel. Ainsi, Vega permet d'enrichir le monde virtuel de comportements sonores. Nous avons vu précédemment que Performer permettait d'intégrer tout type de nœuds. Ainsi, dans les faits, les développeurs de Vega ont enrichi Performer de leurs propres nœuds sonores [Inc98].

World Toolkit

World Toolkit est une librairie permettant la synthèse et la manipulation d'objets très différents. Ce système permet en particulier de gérer la synthèse d'un monde immersif. Cependant, elle possède un défaut majeur : tout le rendu est effectué dans le processus principal. Il est donc difficilement concevable d'effectuer un calcul lourd en parallèle au rafraîchissement du rendu de la scène qui est fonction de la position de l'utilisateur. De plus cette librairie ne tient pas compte des périphériques haptiques.

Multipipe SDK (MPK)

Cette librairie, développée par SGI et qui existe uniquement sous IRIX, est plutôt inclassable. En effet, elle était pendant un certain temps OpenSource, jusqu'au moment où elle est devenue *multithread*[4], date à laquelle SGI l' intégré dans la vente de son OS. On constate habituellement que c'est plutôt le schéma inverse que subissent habituellement les librairies : elles commencent par être payantes, puis elles deviennent OpenSource. MPK n'a qu'une seule fonctionnalité : fournir les services de base pour la gestion d'un environnement immersif. Ainsi, il ne faut pas espérer y intégrer la gestion de périphériques immersifs, ou bien de plusieurs utilisateurs.

Performer

Performer gère beaucoup plus de fonctionnalités que celles requises pour la gestion d'un simple graphe de scène. Dans les faits, Performer répond également à tous les besoins d'une application développée pour un environnement immersif. Cependant, plusieurs limites apparaissent à ce niveau.

4. précédemment, elle était multi-process.

Par exemple, ce système nous impose de nous servir d'un graphe de scène de la même librairie. Nous avons vu plus haut que cela ne convient pas toujours à certaines applications, en particulier en simulation scientifique.

De plus, Performer n'est qu'une librairie permettant de gérer les pipes graphiques. Ainsi, le développeur est obligé d'implémenter l'ensemble des éléments connexes tels que : la lecture d'un fichier de configuration éventuel, la configuration des différents écrans graphiques et la gestion des périphériques.

Enfin, parmi les problèmes que pose l'utilisation de Performer, c'est son manque de portabilité. En effet, Performer n'est vraiment complet que pour IRIX, puisque les distributions de Linux et de X-window ne permettent pas toutes l'utilisation de Performer.

B Librairies OpenSources

Depuis quelques années, le phénomène OpenSource atteint également le domaine de la Réalité Virtuelle. Ainsi, par exemple, *Carolina Cruz Neira*, qui est à l'origine de CAVElib, a lancé son équipe dans le développement d'un système OpenSource de gestion des environnements immersifs : VR juggler.

VR juggler

Etant donné qu'elle est OpenSource, et qui de plus est développée par un ténor de la Réalité Virtuelle, cette librairie est aujourd'hui la plus répandue dans le milieu de la recherche scientifique. Tout comme CAVElib, elle apporte tous les services de base pour la gestion d'un environnement immersif.

Elle permet plusieurs fonctionnalités importantes associées à la gestion des environnements immersifs. Avant tout, elle est capable d'intégrer des contextes de rendu Performer. Cela signifie qu'elle peut gérer des applications développées sous Performer, donc utiliser par exemple des graphes issus de cette librairie ou de tout autre logiciel que cette dernière est capable de lire (formats DXF, OBJ et VRML notamment).

Nous avons vu que Performer est capable d'inclure des nœuds sonores. C'est pourquoi les développeurs de VR juggler proposent d'intégrer eux aussi un rendu sonore dans leur librairie, avec ou sans nœud Performer.

Au niveau de la gestion des événements, VR juggler possède sa propre librairie de gestion des événements. De plus, il a une interface vers Trackd, la librairie associée à CAVElib.

Cependant, VR juggler est une librairie relativement limitée. Ainsi, par exemple, il n'existe aucune interface avec VRPN. De plus, elle est peu appropriée à la gestion des applications scientifiques. En effet, nous avons vu que de telles applications requièrent, par nature, un calcul en parallèle à l'affichage. Ce calcul est totalement découplé de l'affichage sauf en certains instants clefs où les données scientifiques sont transmises au module d'affichage. Or, VR juggler découpe les *process* de modification de la base de données d'affichage selon trois modes : <*Avant*, *Pendant*, *Après*> l'affichage. Ce découpage est donc peu compatible avec la notion de calcul en parallèle continu jusqu'à un « point de rencontre ».

De plus, ce problème est accentué par l'impossibilité de réinitialiser l'application sans détruire et relancer complètement cette dernière.

Dans les faits, plusieurs auteurs de nouvelles librairies reprochent une grande complexité dans la configuration de VR juggler. Nous même avons éprouvé des difficultés à mettre en place VR juggler. Mon analyse est qu'elle gère beaucoup trop de choses en confondant les interactions. Par exemple, interaction continue et interaction discrète sont confondues (à l'instar du fait que sont proposé des fonctionnalités de navigation basées sur des interactions clavier).

Diverse

La librairie Diverse a été développée par Virginia Tech. Elle est principalement basée sur Diverse ToolKit (DTK) que nous avons présenté précédemment (voir section 2.3, page 71). Cette librairie gère également l'ensemble des aspects d'un environnement immersif, avec pour objectif de simplifier l'implémentation d'une application. Cependant, Diverse a un gros défaut pour la synchronisation des affichages. En effet, avec cette librairie, il peut exister un décalage temporel entre les différentes images issues des différents plans de l'environnement immersif.

FreeVR

Cette librairie relativement récente est due à *William Shermann* du NCSA, par ailleur, co-auteur d'un des plus récents ouvrages de synthèse sur la RV [SC02]. Son but premier est de fournir une API OpenSource complètement compatible avec CAVElib, c'est-à-dire qu'en changeant les noms des fonctions de l'API, le comportement et l'algorithmie puissent rester inchangés. Ainsi, ses fonctionnalités sont très proches de cette dernière librairie. Au niveau de la gestion des périphériques, FreeVR se base sur son propre système, une interface avec Trackd et une autre avec VRPN.

Une idée intéressante de FreeVR est l'utilisation d'un serveur *telnet* en local sur la machine. Ce dernier permet de configurer le système de gestion de l'environnement immersif sans être obligé d'intégrer une interface de type menu. Mais surtout, il permet de piloter l'application à distance au travers du réseau. Actuellement, l'interface est relativement pauvre, mais pourrait s'étoffer dans les versions futures.

VTK

VTK (Visualization ToolKit) [SML96] est une librairie basée sur le principe que le temps de développement d'une application peut être grandement réduit en utilisant un langage interprété. Ainsi, les concepteurs proposent d'implémenter une application sous la forme d'un langage comme Tcl/Tk ou Java. Afin de ne pas trop pénaliser l'exécution du rendu, VTK utilise une sous-couche compilée et une librairie graphique propre. Ainsi, le code développé sur VTK pourra rester valable si OpenGL est remplacée par une autre librairie.

Cependant, VTK ne permet pas le rendu dans un environnement immersif. C'est pourquoi les utilisateurs sont obligés d'y adjoindre une librairie comme vtkcave, VTK2CAVE ou VtkActorToPF pour permettre un rendu immersif. Alors que la première est purement OpenGL, la seconde utilise tous de même une interface avec les machines SGI. La troisième est beaucoup plus contraintes, puisqu'elle permet de convertir une scène VTK en scène Performer.

C Discussion

Nous avons vu que toutes les librairies fournissent sensiblement les mêmes services de base pour la gestion d'un environnement immersif.

Cependant, certaines librairies ne permettent pas de synchronisation des affichages entre les différents plans. Cela peut être préjudiciable dans le cas d'un environnement immersif avec plusieurs plans.

De plus, peu d'entre elles se préoccupent des calculs qui doivent continuer à être effectués en parallèle au rafraîchissement, comme cela est par exemple utile dans certaines applications scientifiques (voire notamment VRjuggler, page 99). Rappelons que ce rafraîchissement sert à adapter le rendu (et en particulier la projection stéréoscopique) au point de vue de l'utilisateur.

Contrairement aux objets les plus courants en RV&A, un objet de simulation scientifique peut être extrêmement paramétrable, au point où il peut requérir une interface propre. Dans le cas d'un environnement immersif, cette interface doit pouvoir être affichée sur un autre écran que ceux du dispositif. Plus globalement, une telle interface est nécessaire pour « contrôler » l'ensemble de l'application immersive. Cela signifie généralement que l'interface doit pouvoir être totalement ou partiellement exportable sur un autre ordinateur. C'est notamment le cas de FreeVR, avec son interface Telnet.

Enfin, un autre point critique commun à beaucoup de ces librairies est qu'elles semblent consommer l'ensemble des ressources calculatoires de l'ordinateur pour le rendu immersif. Cela est contraire à leur utilisation dans le cadre d'applications scientifiques. En effet, la partie calcul scientifique requiert généralement beaucoup de puissance CPU. Cependant, une solution à ce problème consiste à utiliser une machine multiprocesseurs. Ainsi, le calcul immersif est effectué sur l'un des processeurs, pendant que les calculs scientifiques se déroulent sur les autres. Performer multipipe permet cela. Cependant, cette librairie impose d'autres contraintes, tel que le graphe de scène qui l'empêche d'être optimale dans certains cas.

2.3 Conclusion sur les librairies existantes

Nous avons vu dans le cadre de cette section que OpenGL était la librairie graphique multi-plateforme de référence. C'est pourquoi, nous ne pouvions pas imaginer développer une librairie de gestion d'un environnement immersif sans OpenGL. Par contre, nous n'utiliserons aucune librairie de plus haut niveau comme Performer ou OpenInventor. En effet, elles introduisent des facilités dans la gestion de graphes de scènes. Cependant, ces

facilités ne sont pas toujours compatibles avec la structure même de nos bases de données scientifiques.

Bien que nous ne nous soyons pas servis de librairies de gestion des environnements immersifs, nous nous en sommes inspirés pour concevoir notre propre librairie. En effet, la gestion de base d'un environnement immersif est toujours la même : calcul de projection exacte pour un point de vue donné sur plusieurs plans. Par contre, toutes les librairies actuellement disponibles ne permettent pas de gérer l'ensemble des contraintes que nous imposent en particulier les applications scientifiques.

3 Gestion du dispositif immersif

La fonctionnalité de base d'un environnement immersif est de savoir gérer l'ensemble des écrans qui le compose pour qu'un ou plusieurs utilisateurs puissent voir le monde virtuel. Cependant, dans le cadre de nos considérations sur les interactions, nous devons intégrer d'autres composants tout aussi fondamentaux, à notre sens.

Par exemple, afin d'unifier nos paradigmes, nous avons fait évoluer le concept de *véhicule* que DIVE fut l'un des premiers à utiliser. Dans cette section, j'illustrerai ensuite la gestion de deux types de projection stéréoscopique (HMD et Mur), et ce en particulier dans le cadre d'une application de CAO. Ensuite, nous mettrons en avant plusieurs contraintes telles que l'interaction multi-utilisateurs, ou le multi-fenêtrage sur une seule carte graphique.

3.1 Concept de véhicule

Partons du constat simple que l'espace d'interaction de l'utilisateur n'est pas infini. En effet, il est contraint par les longueurs de câbles ainsi que par les limites d'influence des capteurs de position. Les murs de l'environnement immersif ou les câbles de connexion avec d'un casque HMD sont des limites encore plus importantes. En effet, si l'on peut imaginer des systèmes de capture sans fils, il est inconcevable de supprimer les écrans.

Donc l'utilisateur ne peut physiquement pas aller au-delà de certaines limites propres au dispositif matériel. Ceci nous amène logiquement au concept de véhicule : *le véhicule est l'espace dans le monde virtuel défini par l'intersection de l'ensemble des zones d'influence des capteurs, des longueurs de câbles et des limites physiques de l'environnement immersif.*

Volume d'interaction

Bien que le véhicule dans le monde virtuel n'ait généralement pas d'existence en tant qu'entité virtuelle perceptible (sauf si cette métaphore est susceptible d'être rattachée à un type d'interaction particulier à l'application, comme pour un simulateur de conduite), son équivalent dans le monde réel représente un volume bien délimité par les contraintes des différents périphériques. De plus, il est intéressant de constater que l'utilisateur ne peut s'en échapper sans perdre un canal d'interaction. Plus particulièrement, l'utilisateur ne peut ni interagir ni se déplacer hors de ce volume du monde réel. Ainsi, j'appellerai *volume d'interaction* l'espace correspondant au véhicule dans le monde réel.

« Correspondance » entre l'univers réel et le monde virtuel

Tous les mouvements de l'utilisateur captés dans le volume d'interaction sont retranscrits à l'identique dans le véhicule. En fait, la correspondance des deux éléments fait que leurs référentiels sont également en bijection (voir la bijection entre l'univers réel et l'espace de représentation de la section A de la page 17). C'est-à-dire qu'un capteur défini dans le référentiel du volume d'interaction peut être retranscrit sans aucune modification

dans le monde virtuel. En ce sens, le véhicule se rapproche de la notion de « rooms[5] » de [RH94]. Ainsi, la métaphore du véhicule sert de « pont » entre le monde virtuel et l'univers réel.

Afin de simplifier l'ensemble des tâches de gestion de l'environnement immersif, nous définirons tous les paramètres de l'interaction dans un référentiel lié au volume d'interaction.

Echelles

L'échelle est une question pertinente pour les applications scientifiques. En effet, on peut aussi bien vouloir travailler sur un brin d'ADN que souhaiter simuler un modèle astrophysique.

Cependant, le volume d'interaction a sa métrique propre qui ne peut ni être dilatée, ni être condensée puisqu'elle est à l'échelle humaine. Pour nous, l'unité de mesure correspondant à l'échelle humaine est le mètre. Il en résulte que pour simplifier nos traitements, la première chose à faire pour chaque périphérique est de le redéfinir dans un système métrique.

Contrairement au volume d'interaction, l'échelle du véhicule, elle, n'est pas contrainte. Ainsi, c'est en faisant varier l'échelle de ce dernier que nous pouvons changer l'utilisateur d'échelle.

Navigation du véhicule

Il est intéressant de voir que l'utilisateur a toute latitude pour visiter le volume d'interaction. C'est-à-dire qu'il peut voyager partout dans ce volume sans être contraint par quoi que ce soit. Ses moyens d'investigation sont ceux de tous les jours : le déplacement de son corps.

Nous avons vu plus haut qu'il peut être intéressant d'autoriser un mouvement plus important dans le monde virtuel. Notre point de vue est que cette navigation est autorisée par le déplacement du véhicule. Ainsi, le terme véhicule prend toute sa valeur. C'est-à-dire qu'il existe une relation constante hors navigation entre le référentiel de la scène et celui du véhicule. A chaque étape de la navigation cette relation évolue jusqu'à ce que l'utilisateur stabilise le véhicule pour permettre l'étude des éléments présents dans ce dernier.

Métaphore du véhicule réel

Un véhicule réel est généralement équipé d'une ou plusieurs fenêtres sur le monde extérieur. Là où dans une voiture il y a les fenêtres, on retrouve les écrans de l'environnement virtuel. Cette métaphore s'applique également aux casques immersifs, à ceci près que le point de vue est solidaire de la fenêtre de façon analogue à un casque de moto.

5. Une traduction en Français de ce terme est « pièce ».

Du point de vue de l'utilisateur

L'utilisateur doit avoir besoin de sentir les limites du volume d'interaction. En effet, l'utilisateur doit savoir s'il essaye de prendre un objet hors de la limite d'influence du capteur ou de la pièce.

Dans l'idéal, l'utilisateur ne perçoit pas les limites induites par les murs. La chose est encore plus problématique avec les casques HMD où l'utilisateur ne voit même pas les limites physiques du dispositif (cordons ombilicaux, capteurs, ...). C'est pourquoi une évolution de ce système intégrera une frontière du volume d'interaction dans le cas ou l'utilisateur s'en approcherait trop. Nous pourrions par exemple matérialiser les limites du volume par un hexaèdre (inscrit dans l'intersection de tous les volumes interactifs) aux frontières translucides.

3.2 Projection perspective

Le principe de la projection perspective est de projeter l'ensemble des points de l'espace sur un plan selon des droites concourantes en un point nommé centre de projection. Cette projection est celle dont l'être humain se sert. En effet, le centre de projection n'est autre que le point de focalisation du cristallin[6], c'est-à-dire le centre de l'œil.

En projection perspective, deux lignes parallèles ne forment pas deux droites parallèles. La seule exception est lorsque les droites sont parallèles au plan de projection. En fait, deux droites parallèles ne faisant pas partie de ce cas particulier donneront l'impression de se rencontrer au point de fuite.

A Stéréoscopie

Nous avons vu précédemment que la stéréoscopie permet de percevoir le relief. Dans ce cadre, nous utilisons la projection perspective. Cependant, il ne suffit pas de prendre deux points de vue différents sur la scène. Il faut tenir compte de certaines considérations géométriques.

Le fait de fournir deux projections à l'utilisateur ne suffit pas pour lui faire percevoir le relief. En effet, il faut intégrer un ensemble de paramètres ergonomiques.

Contrainte épipolaire

Soit O_g et O_d deux points de vue et P un point de la scène 3d (cf. figure 3.1). Les points O_g, O_d et P définissent un plan appelé *plan épipolaire*. Les *droites épipolaires* (ou *lignes épipolaires*), g et d, sont les intersections du plan épipolaire et des pl ans image. Les images p_g et p_d de P sont les projections de P le long des lignes de vues O_gP et O_dP. Les points p_g et p_d sont donc dans le plan O_gO_dP et sur les droites épipolaires. On constate donc que le point p_g de l'épipolaire gauche g doit correspondre à un point p_d de l'épipolaire droite d.

6. Notons qu'il s'agit d'une approximation qui consiste à assimiler le cristallin à une lentille mince.

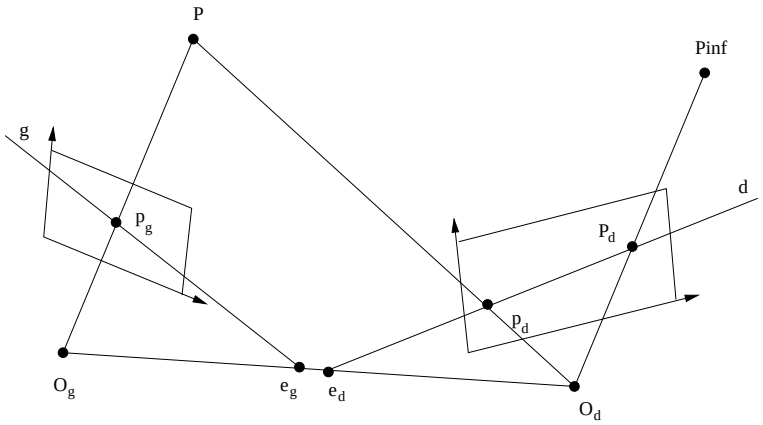

FIG. 3.1 – *Contrainte épipolaire.*

Examinons maintenant où se trouve le point p_d lorsque P se déplace sur la droite O_gP, c'est à dire lorsque P est plus ou moins loin des caméras. Soit e_d l'*épipole* droit. e_d est l'image de la caméra gauche O_g par la caméra droite O_d. Ainsi, lorsque P se rapproche de O_g, son image p_d dans l'image droite se rapproche de e_d. Lorsque P est très éloigné des points de vue, il se projette en $p_{d\infty}$, intersection du plan image droit et de la droite passant par O_d parallèle à la direction O_gP. D'où l'énoncé de la contrainte épipolaire : le point p_d, correspondant du point p_g dans l'image droite, appartient nécessairement au segment épipolaire $[e_d,p_{d\infty}]$.

Cette contrainte est à la base de la projection stéréoscopique. Elle est fondamentale pour garantir que l'utilisateur arrive à fusionner correctement l'information issue des deux yeux afin de reconstituer le relief.

Plan de convergence

Dans le cas des murs, la solution la plus simple et la plus communément utilisée pour respecter la contrainte épipolaire des images est de les projeter sur le même plan. Nous appelons ce plan, le plan de convergence. Cependant, dans le cas de la réalité virtuelle, ce plan ne suffit pas ! Nous avons vu que le véhicule ouvre une fenêtre sur le monde virtuel. Or, au-delà d'appartenir à un plan, une fenêtre est par définition bornée. Ainsi, non seulement les plans de projection des deux points de vue doivent être confondus pour garantir l'épipolarité, mais en plus, les deux fenêtres de projection doivent être confondues. Notons que nous pouvons placer cette fenêtre de projection partout dans le monde réel. Cependant, il est préférable de la placer au niveau de l'écran. Sinon, nous risquons des décalages aux jointures entre les écrans.

Par contre, pour les casques immersifs, les deux plans de projection associés à la stéréoscopie peuvent même ne pas être parallèles puisque chacun doit exactement se superposer aux deux écrans du casque.

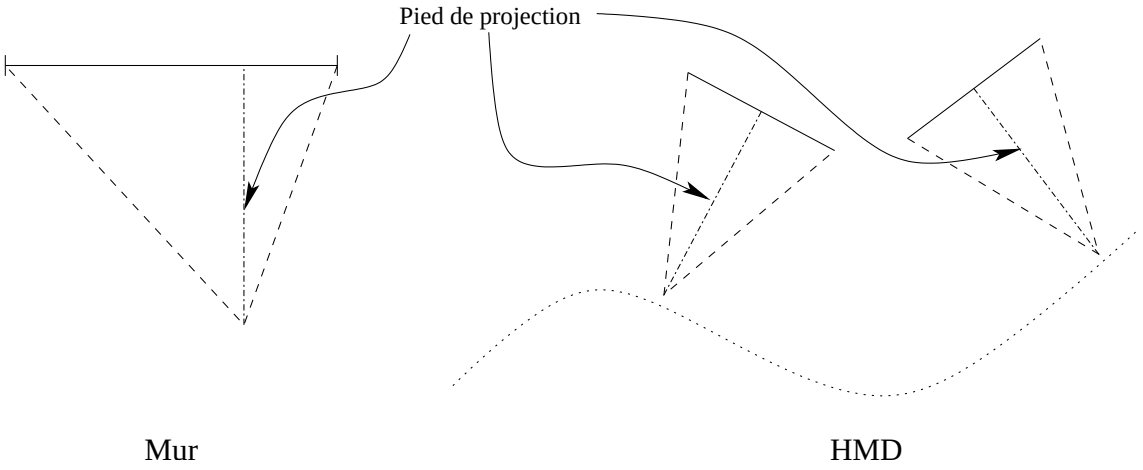

FIG. 3.2 – *Le pied de projection est la projection orthogonale de point de vue sur l'écran. En fonction du type de dispositif, il se déplacera dans l'espace réel.*

Enfin il convient d'introduire également le concept de *pied de projection* (voir figure 3.2). Il s'agit de la projection orthogonale du point de vue sur le plan de projection. Il sert à définir le repère de l'écran par rapport au point de vue.

Disparité

La disparité entre deux images, directement issue de l'écartement des yeux, est fondamentale. En effet, c'est cette disparité qui permet de percevoir le relief.

Dans les premières applications implémentées, nous avions introduit une fonctionnalité permettant de changer la disparité en modifiant artificiellement l'écartement des yeux. Au demeurant, il importe de noter que ceci est une « tricherie » qui induit une erreur lorsque l'interaction de l'utilisateur requiert une perception exacte de la profondeur des objets. Ainsi, comme l'explique [ZHR99], en jouant artificiellement sur l'écartement des yeux, l'utilisateur risque de confondre les profondeurs et vouloir attraper un objet virtuel qui n'est géométriquement pas à l'endroit où il est perçu (figure 3.3).

Un terme souvent utilisé pour qualifier l'angle de perception des deux yeux associé à cette disparité est la *parallaxe*. Il existe cependant une limite physiologique à cette parallaxe. Par exemple, si vous mettez votre main à quelques centimètres des yeux, vous aurez du mal à forcer un strabisme convergent pour fixer la paume de la main[7]. Ainsi, avec des objets relativement proches de l'utilisateur dans le monde virtuel, le problème sera identique et risque d'engendrer des troubles à l'utilisateur. La solution que nous pensons mettre en œuvre est d'ôter ces objets de la scène. Cependant, ce traitement requiert une

7. Il est important de noter que si l'on voit flou, ce n'est pas à cause de la limite de parallaxe. C'est parce que le cristallin n'arrive pas à focaliser sur une distance aussi courte.

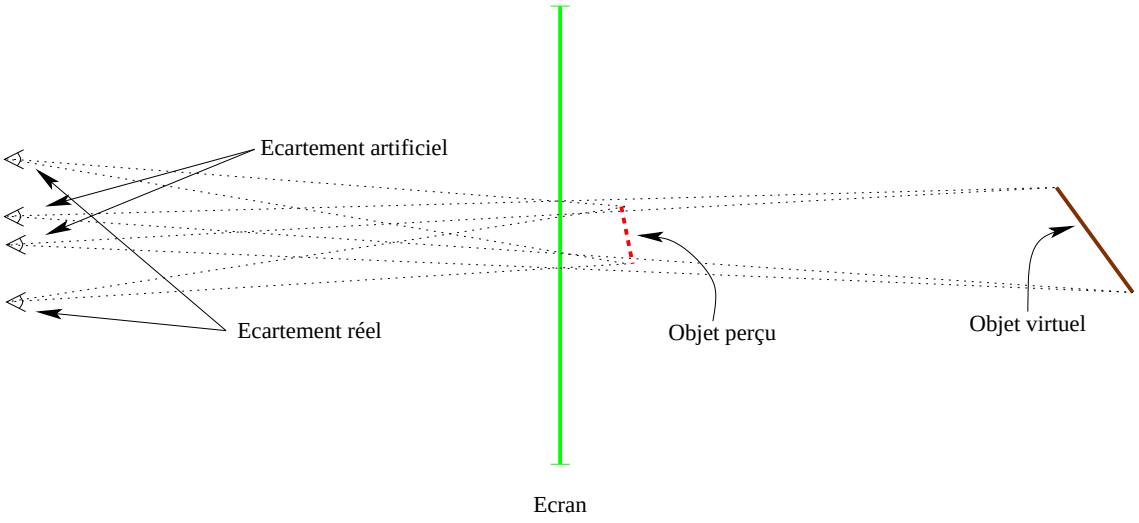

FIG. 3.3 – *Lorsqu'on rapproche artificiellement la position des points de vues virtuelles, les objets sembleront se rapprocher. Inversement, lorsqu'on les écarte, les objets s'éloigneront.*

information sur la distance des objets les plus proches de l'utilisateur. Sur des scènes complexes, cette information n'est malheureusement disponible qu'avec des algorithmes de gestion de scène ad hoc.

Un autre problème peut survenir. Par exemple, il n'est pas rare que l'utilisateur ne souhaite pas que la stéréoscpie s'adapte à son point de vue (notamment lors de présentation devant un public). Or, avec tous les systèmes, il existe une position par défaut. Cette dernière est toujours relativement loin de l'écran. Cependant, si l'utilisateur s'approche de l'écran, il risque de dépasser cette valeur limite de parallaxe. Dans ce cas, nous jouons sur l'écartement virtuel des yeux en le réduisant afin d'éviter de dépasser la limite de parallaxe. En effet, dans ce cas, le but principal est généralement le confort de visualisation du public et non la précision des mouvements dans le monde virtuel.

B Distinction géométrique entre les murs et les HMDs

Dans le premier chapitre, nous avons abordé les différences entre les murs et les HMDs. Ces différences influencent même le calcul de la projection (revoir Figure 3.2).

Le cas des HMDs est le plus simple. En effet, comme l'écran est généralement centré par rapport à l'œil, le pied de projection est au milieu de l'écran. Ainsi, les deux projections sont symétriques. De plus, il faut tenir compte de la translation entre les deux yeux. La position de l'utilisateur intervient dans le changement de référentiel du système complet composé par les lunettes.

Au niveau des murs, la projection doit être centrée au niveau de l'œil de l'utilisateur. Ainsi, contrairement au cas des HMDs, le pied de projection n'est pas centré et bouge

en permanence par rapport à l'écran. C'est pourquoi la projection sur les murs requiert une projection dissymétrique en fonction du point de vue. On doit donc, pour obtenir la position de chaque œil, déterminer la position et l'orientation de la tête de l'utilisateur par rapport à l'écran. La position de chaque œil se déduit par deux translations relativement au référentiel de la tête.

Implémentation des différentes stéréoscopies Nous avons vu que dans le cas de la stéréoscopie active, la solution optimale requérait à ce jour l'utilisation de projecteurs tritubes. Ce type de matériel, associé à un calculateur capable de fournir quatre buffers, est la première implémentation que nous ayons mise au point. De petites modifications nous ont permis de gérer la stéréoscopie anaglyphique. En effet, il suffit de remplacer l'appel d'écriture dans le buffer droit ou gauche par une écriture dans un seul buffer au demeurant filtré selon la composante rouge ou bien les couleurs vert et bleu (cf. cyan).

Pour les autres modes stéréoscopiques (polarisation, casque HMD, anaglyphe étendu, ...) une solution, pour envoyer un signal cohérent aux écrans des périphériques visuels à moindre coût calculatoire, est d'utiliser un boîtier de dérivation séparant le signal stéréoscopique en deux signaux monoscopiques. Cependant, nous proposons d'utiliser de multiples sorties graphiques sur la même machine. Ainsi, nous ouvrons deux fenêtres, chacune affectée à un des deux signaux de la stéréoscopie. Ensuite, notre environnement gère l'affichage de ces deux fenêtres, tandis qu'il distribue ces couples de fenêtres sur chaque écran d'une CAVE. Chaque fenêtre est donc considérée comme l'un des deux canaux visuels. Les informations précisant si l'on veut autoriser la stéréoscopie active, anaglyphe, passive pour un des deux yeux sont entièrement paramétrables via un fichier de configuration.

C Problème des bords des écrans

Nous avons vu précédemment que les bords de l'écran peuvent gêner la perception du relief. Pour les reliefs en profondeur, notre cerveau est habitué à cette situation dans le monde réel. Pour s'en convaincre, il suffit de regarder par la fenêtre les objets situés dehors. Par contre, nous sommes particulièrement sensibles au fait que les bords des écrans coupent des objets situés virtuellement devant la fenêtre de visualisation.

Eviter que les objets soient coupés par les bords de l'écran est impossible ! En fait, ce qui gêne le plus l'utilisateur, c'est qu'il voit l'objet coupé par l'œil opposé au bord avant que l'objet ne le soit par l'œil situé du même côté que ledit bord (voir figure 3.4 à gauche). Cela est en effet contraire au mode de fonctionnement « naturel » de la stéréoscopie (en profondeur) lorsqu'elle est coupée par une fenêtre de visualisation (réelle ou virtuelle).

Ce problème n'intervient pas dans les jonctions entre deux murs, puisque les objets coupés se prolongent sur l'écran connexe. De plus, dans le cas des casques, le plan de convergence est tellement proche de l'œil de l'utilisateur que les objets sont systématiquement placés derrière ce plan. Ainsi, le problème n'apparaît que dans le cas de la station de travail ou dans celui de bords de murs.

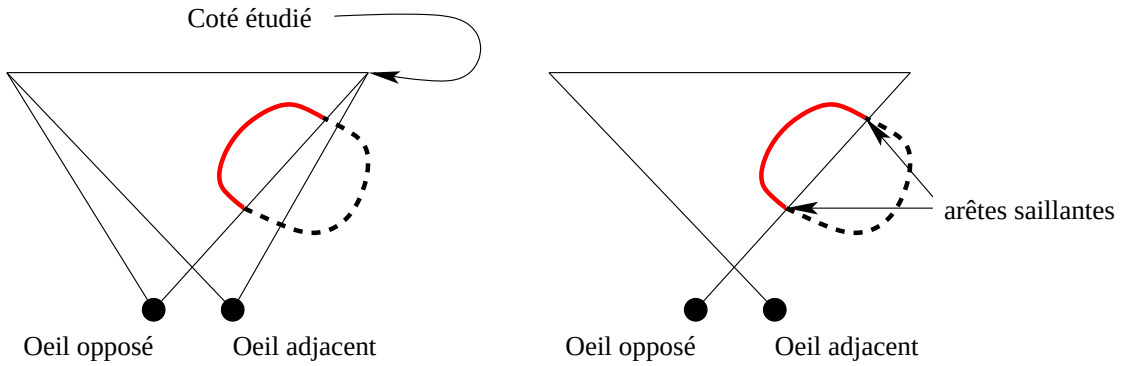

FIG. 3.4 – *Schéma explicatif de l'œil adjacent et de l'œil opposé par rapport au bord droit de l'écran.*

Si l'utilisateur a des problèmes d'ambiguïté aux frontières des écrans, il faut essayer de couper les parties de la scène qui posent ces problèmes. C'est pourquoi j'ai proposé d'ajouter des plans de coupe, pour chaque œil, coupant selon les plans incidents de l'autre œil. Cela a même pour conséquence de renforcer la stéréoscopie en ajoutant des arêtes saillantes aux objets au niveau de leurs intersections avec ces plans de coupes supplémentaires (voir figure 3.4 à droite).

Au demeurant, on remarquera que si l'œil a quatre plans de coupe supplémentaires, seulement deux plans sont nécessaires. En effet, les deux autres sont naturellement inclus dans les régions coupées propres à l'œil courant.

D Zoom et changement d'échelle

Une notion importante de la perception visuelle dans l'univers réel est celle du zoom optique. Elle consiste à modifier l'optique pour ouvrir ou fermer l'angle du point de vue. Géométriquement parlant, il suffit de changer l'ouverture des plans latéraux[8] de la projection perspective pour rapprocher ou éloigner. Ainsi, pour un point de vue fixe à un endroit donné, en fermant les plans latéraux de la projection on changera l'ouverture d'angle. C'est-à-dire que l'utilisateur aura l'illusion de se déplacer vers l'objet car les objets grandiront.

Cependant, cette méthode n'est pas appropriée dans le cas des dispositifs immersifs constitués de plusieurs murs. En effet, l'ouverture des plans latéraux est imposée par les jonctions entre les écrans. Ainsi, il faut trouver une autre solution pour agrandir ou diminuer la taille des objets.

8. L'ouverture des plans correspond à l'ouverture de la focale en optique.

La sous-fenêtre

La méthode « simple » est celle d'une *sous-fenêtre*. Cette sous-fenêtre n'étant pas contrainte, on peut sans problème y faire des « zooms ». Cependant, il faut faire ressortir cette sous fenêtre en matérialisant son contour afin d'éviter une ambiguïté entre le point de vue réel et la partie « zoomée ».

Le changement d'échelle

L'autre solution est de changer l'*échelle de la scène*. En réalité, la scène a son échelle propre et il n'est pas envisageable de la changer. Ainsi, la seule solution est de changer la métrique du véhicule.

Changer cette échelle revient donc à effectuer une homothétie au centre du référentiel du volume d'interaction. Cela a pour conséquence principale de changer la granularité de la perception de la scène par l'utilisateur. Par exemple, cela peut influencer le pas d'incrément de la navigation, en considérant que l'échelle est directement liée au point visé (ou focus). Un autre aspect important de cette homothétie est qu'elle va modifier les plans de coupe avant et arrière de la scène par rapport à l'utilisateur. Dans ces conditions, si l'utilisateur est situé au niveau du centre du référentiel, il n'y aura aucune différence perceptible hormis les objets qui vont apparaîtront ou disparaîtront de la profondeur de son champ de vision.

3.3 Projection orthographique

La projection orthographique est une projection dont le centre est à l'infini. C'est l'une des projections de prédilection de la CAO. En effet, c'est la seule qui permet à deux droites parallèle de rester parallèles après la projection. Ainsi, il est possible de définir des cotations « exactes » quelque soit le point de vue.

Projection orthographique stéréoscopique

Nous avons vu plus haut que la stéréoscopie utilise la projection perspective. Cependant, est-il possible d'utiliser une projection orthographique pour faire ressortir le relief?

La réponse triviale est non ! En effet, cette projection écrase toute déformation par rapport au plan de convergence. Or, ce qui donne la sensation pour un objet d'être loin ou proche, c'est justement la perspective qui grossit les objets proches et diminue les objets lointains. Le relief stéréoscopique ne fait que souligner cette information en nous permettant de composer de façon instantanée deux points de vue sur la même scène.

La sensation de la projection orthographique est celle d'un plan de construction. Comme le centre de cette projection est à l'infini, le seul point intéressant est la position de la projection de ce centre sur le plan. En effet, en fonction de la position de ce point par rapport à la fenêtre, la scène se déplacera entièrement dans la direction orthogonale au plan.

Si on ne crée aucune disparité à partir d'une projection orthographique, ceci revient à la placer dans le plan de convergence (i.e. généralement le plan d'un écran). Si on double cette projection orthographique en lui donnant une disparité égale à l'écartement des yeux de l'utilisateur, cela revient à placer la projection orthographique de la scène à l'infini du point de vue de la stéréoscopie. On peut imaginer, à partir de ces deux cas de base, de faire varier plus ou moins l'émergence sur la profondeur de ce plan de projection orthographique.

Autre visualisation utilisée en CAO

La projection orthographique n'est pas le seul mode de visualisation utilisé en CAO. En effet, les concepteurs se servent souvent de trois plans de projection, un pour chaque axe canonique. Il s'agit alors de prendre une boîte et de projeter l'objet de l'étude sur chacune des faces. Ensuite, on déplie la boîte et on retrouve la projection latérale et la projection de dessus respectivement à gauche et au-dessus de la projection de face.

Extension aux murs

Nous avons vu précédemment que la stéréoscopie dans le cadre de la projection orthographique ne pouvait servir qu'à déplacer le plan par rapport à l'écran.

Mode 1 : **Planche à dessin virtuel** Une première méthode de visualisation dans un système à base de murs immersifs est de projeter orthographiquement la scène sur un plan, puis de placer le plan dans le monde virtuel pour être rendu par le dispositif immersif en stéréo. C'est l'idée de la planche à dessin virtuelle.

Mode 2 : **boîte de CAO fermée** Une évolution de cette première méthode est de transformer la planche à dessin pour la spatialiser. On commence par relever les projections latérales et supérieures de la boîte décrite plus haut. Ainsi, nous pouvons synthétiser en trois dimensions la boîte ouverte en stéréo. Cela permet, pour le concepteur de clarifier les projections. Nous pouvons même envisager de rajouter des plans de projection intermédiaires inclinables.

Mode 3 : **boîte immersive** Une troisième approche est la synthèse, sur chacun des écrans d'une CAVE, de la projection usuelle aux concepteurs CAO. Ainsi, on prend la boîte décrite ci-dessus et au lieu de la déplier, on affiche ses faces sur les écrans du dispositif.

Afin d'augmenter les potentialités et l'ergonomie d'un tel système, nous pouvons également ajouter une synthèse stéréoscopique de l'objet. Cependant, on risque une confusion entre l'objet virtuel et sa projection. C'est pourquoi nous plaçons la projection orthographique à l'infini du point de vue stéréoscopique (voir ci-dessus).

Quoiqu'il en soit, il faut que les 3 modes soient proposés au choix. En effet, bien que nous encouragions de nouvelles interactions plus ergonomiques, nous devons permettre à l'utilisateur de retrouver ses marques en fournissant les outils dont il se sert habituellement.

3.4 La calibration

Un des problèmes des environnements immersifs est de faire en sorte que la métrique virtuelle des objets corresponde à celle de l'univers réel. Pour ce faire, il faut concevoir des systèmes permettant de connaître le changement de référentiel exact entre celui des capteurs de mouvements et celui du véhicule. C'est l'objet de la calibration.

Calibration grossière Un fichier de configuration contient des informations qui décrivent l'ensemble des opérations permettant de passer du référentiel des capteurs à celui du volume d'interaction. Cette calibration est grossière lorsque c'est le développeur qui détermine les dimensions et les angles de rotation à l'aide d'instruments manuels de mesure.

Calibration précise La seule solution pour atteindre la précision nécessaire à l'interaction, est de se servir du capteur de mouvement pour calculer la géométrie du dispositif. Cependant, il est difficile de positionner le capteur sur la structure étant donné qu'il est rare qu'elle ne contienne pas quelques éléments ferro-magnétiques perturbant localement la mesure.

La solution choisie est celle d'un pointeur laser solidarisé au capteur de position. En pratique, on génère une mire contenant cinq points par écran et, à l'aide du laser, on pointe successivement ces points. La mise en correspondance entre projection réelle et virtuelle est la base d'un calcul où l'inconnue est la géométrie de l'écran exprimée dans le repère de base des capteurs. Ainsi, on peut obtenir la géométrie de chaque écran et, par extension, celle du dispositif.

Vis-à-vis de la définition de la géométrie du dispositif dans le fichier de configuration, il faut que le référentiel du véhicule reste cohérent avec la position des écrans. En effet, quelle que soit la méthode de calibration, si la position des écrans est précisée par rapport à un certain référentiel, il faut que cela perdure. La solution choisie a été de figer le passage du référentiel du volume d'interaction à celui du premier écran sur la base de l'information fournie par le fichier de configuration (cf. figure 3.5). Ensuite, on calcule le référentiel de cet écran dans celui du capteur de position, et ainsi, on obtient le changement de référentiel du capteur de position vers le référentiel du volume d'interaction.

3.5 L'aspect multi-utilisateur

Un point important que nous ne ferons qu'évoquer dans ce mémoire est la possibilité pour deux utilisateurs d'interagir sur un même objet. Or, afin d'interagir correctement à plusieurs sur les objets, il faut que le relief soit exact pour chaque utilisateur. Différentes méthodes sont possibles pour gérer l'ensemble des paramètres utiles à la gestion de plusieurs utilisateurs.

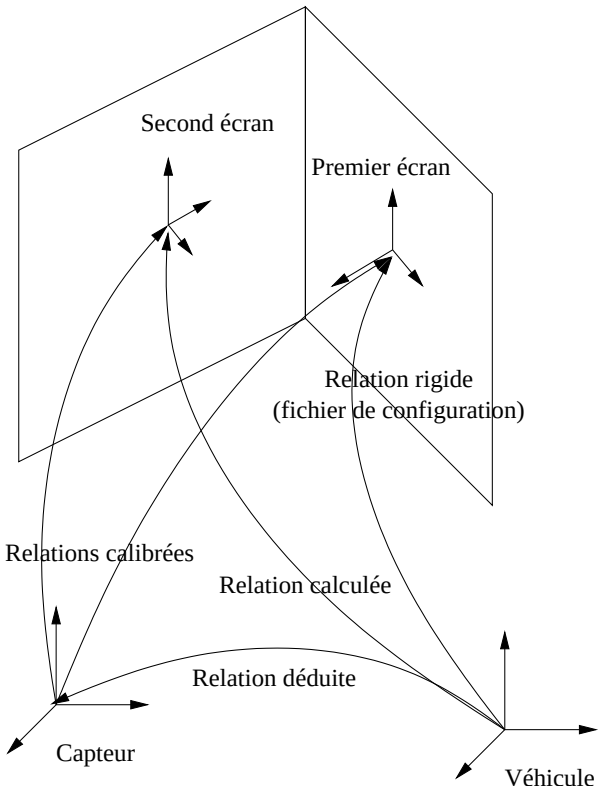

FIG. 3.5 – *Calibration fine : la relation du premier écran au référentiel est rigide.*

Gestion commune des affichages

Nous pouvons, dans un premier temps, concevoir qu'une unique boucle gère l'actualisation de l'affichage de tous les utilisateurs. L'intérêt majeur de cette approche est d'éviter des problèmes d'asynchronisme, lorsque la scène subit de son côté une mise à jour (modification des objets qu'elle contient). En effet, les utilisateurs ayant des comportements autonomes, il est difficile, en l'absence d'une telle boucle, de synchroniser le rafraîchissement des affichages stéréoscopiques des utilisateurs.

Gestion dissociée des affichages

Cependant, cette gestion commune des affichages ne convient pas toujours. Par exemple, en supposant que l'un des utilisateurs soit le démonstrateur traqué et qu'il montre une scène à un ensemble d'autres utilisateurs, les fréquences de rafraîchissement pour chaque utilisateur sont différentes. En effet, l'utilisateur traqué aura tendance à demander une stéréoscopie exacte, alors que les autres n'auront aucun tracking et « subiront » une stéréoscopie moyenne. Ainsi, un rafraîchissement de la scène est opéré pour les utilisa-

teurs passifs alors que ce rafraîchissement est inutile.

De façon analogue, si un utilisateur donné a un point de vue assez pauvre sur le monde virtuel, la scène peut être très rapide à dessiner. D'autre part, si un autre utilisateur regarde un endroit du monde virtuel très complexe, le temps pour l'afficher peut être très long. Ainsi, nous obtenons deux utilisateurs avec des temps d'affichage complètement différents. Si nous implémentons une seule et même boucle pour tous les utilisateurs, nous allons pénaliser l'utilisateur ayant besoin d'un temps de rafraîchissement plus court que les autres.

3.6 Plusieurs fenêtres pour une seule carte graphique

A l'heure actuelle, même les cartes graphiques à faible coût possèdent plusieurs sorties vidéo, alors que précédemment, cette capacité était réservée aux calculateurs graphiques important.

Le problème est que même s'il y a deux sorties, le signal graphique transite par le même composant. Ainsi, si nous avons deux fenêtres sur la même carte, pilotée par deux *thread* différents, chaque *thread* doit avoir son propre contexte graphique. Le choix de séparer les différents contextes OpenGL sur des *threads* différents, même s'ils concernent la même carte graphique, est imposé par les blocs de mémoire propres à chaque contexte. Même si nous devons avoir une base de données commune à toutes les tâches d'affichage, chaque contexte doit posséder son propre bloc mémoire contenant, par exemple, les indices des textures ou des display-lists. C'est d'autant plus important que dans les cas de gestion de scènes, l'état courant du graphe de scène est propre à chaque fenêtre. C'est pourquoi, une séparation en *threads* par contexte permet d'utiliser des facilités mises en place par les systèmes d'exploitation pour la création de blocs de mémoire spécifiques.

Cependant, pour dessiner simultanément sur chacun des *threads*, le système d'exploitation va être obligé de permuter entre tous les contextes en fonction du *thread* actuellement activé par l'ordonnanceur (*scheduler*). Cette tâche de permutation est tellement lourde que les développeurs d'applications commerciales (3DS max par exemple) font en sorte que l'ensemble des affichages s'effectuent avec un unique contexte OpenGL.

Dans les faits, nous pouvons avoir besoin de plusieurs contextes OpenGL sur une même carte graphique (par exemple, un contexte par sortie graphique). Nous retrouvons alors le problème de lourdeur de permutation des contextes. Ainsi, plutôt que d'autoriser une permutation, la solution que nous avons proposée et validée sur les machines de type SGI est un sémaphore « bloquant » le *thread* de chaque contexte OpenGL d'une même carte graphique. Ainsi, l'ordonnanceur ne peut pas poursuivre l'exécution d'un autre *thread* ayant un contexte graphique sur la même carte. Pour la gestion des machines SGI, notre environnement est donc en mesure de disposer d'un sémaphore par carte graphique chargé de gérer chaque *thread* associé aux différents contextes OpenGL de ces cartes. Notons que les informations nécessaires à la mise en place de cette fonctionnalité ne résultent pas d'une documentation existante des appels systèmes sous IRIX, mais de l'analyse de l'implémentation de ces mêmes appels sous Linux. Sur les autres types de machine, nous

n'avons, à l'heure actuelle, aucun moyen de déterminer si deux contextes appartiennent à la même carte graphique. C'est pourquoi, sur ces machines non-SGI, notre environnement ne prévoit qu'un seul sémaphore chargé de bloquer l'ensemble des contextes de toutes les cartes graphique.

3.7 Sauvegarde des paramètres

Dans le cadre de la stéréoscopie, on doit passer deux fois dans la boucle affichage (une fois par œil). En fait, il faut multiplier ce chiffre par le nombre de plans stéréoscopiques. Mais si un paramètre est modifié entre les deux affichages graphiques, la scène sera incohérente entre les deux points de vue, ce qui pose problème.

La première étape est de « figer » les paramètres importants. C'est ainsi qu'avant chaque nouvel affichage par utilisateur, on sauvegarde les paramètres associés à ce dernier. Nous nous servons de cette sauvegarde pour calculer les affichages pendant que les paramètres originaux sont modifiés par les *threads* s'occupant de l'interaction.

3.8 Gestion de la latence du système

Nous avons vu précédemment (section E, page 22) que l'idéal serait d'obtenir une latence nulle entre la capture de l'information par le périphérique et son rendu dans l'environnement immersif. Cependant, cela est une utopie. Nous avons vu également qu'une autre solution serait, à défaut, de contrôler le temps de latence de telle manière qu'il soit constant. Rappelons que l'EVserveur est prévu pour intégrer un système de synchronisation temporelle. De plus, la latence interne à chaque périphérique est constante. Ainsi, on peut connaître la latence totale d'un événement à partir du moment où il est reçu dans la structure de l'EVserveur. Cependant, nous avons vu que la permutation des affichages (*swap de buffers*) pouvait être le paramètre de latence le plus variable. Ainsi, nous prévoyons d'implémenter un système permettant de synchroniser l'ensemble des affichages sur un signal de synchronisation extérieur. Ce signal pourra ainsi être celui d'un synchronisateur central chargé de maintenir constante la latence du système.

4 Connexion à l'EVserveur

Nous avons vu dans le chapitre précédent, que l'EVserveur permet de gérer l'ensemble des périphériques de Réalité Virtuelle. Les événements issus des périphériques arrivent avec trois informations permettant de les identifier : l'identifiant du nœud de l'EVserveur, l'identifiant du pilote (ou *driver*) ainsi que celui interne au périphérique.

Pour que les clients de l'EVserveur puissent utiliser ces informations, l'EVserveur précise une fois pour toutes, au moment du lancement d'un périphérique ou d'un client, les paramètres des périphériques. Ces paramètres contiennent l'ensemble des descriptifs du périphérique sous forme textuelle et sous forme d'identifiants.

Ainsi, la première étape est de stocker l'information associée à l'ensemble des périphériques propres à l'application. Cette information permet, moyennant la lecture du fichier de configuration associé, de savoir à quel périphérique doit être associé l'événement arrivant.

Mais le système permet beaucoup plus. En effet, il fonctionne sur la base d'identifiants uniques attribués à chaque périphérique. Ces identifiants sont fournis à l'application utilisant le noyau géométrique. Ainsi la fonction de traitement des événements ne contient qu'un seul test sur la valeur de cet identifiant (`switch ... case` en C).

4.1 Boucle d'événements

Généralement, les logiciels fonctionnent avec une boucle d'événements. C'est-à-dire que le programmeur implémente une boucle contenant successivement le traitement des informations issues des périphériques suivi de la procédure d'affichage associée.

Cette méthode suffit dans la plupart des cas, puisque l'affichage est souvent directement dépendant des événements extérieurs. Cependant, dans notre cas, nous avons vu que les temps de rafraîchissement et de traitement sont différents pour chaque utilisateur. De plus, les événements issus de périphériques sont cadencés à des fréquences relativement disparates.

Par exemple, nous pouvons rencontrer des temps de latence importants dus, d'une part à la nature de la modalité interactive, d'autre part à la complexité du traitement demandé à la machine par l'utilisateur. C'est le cas, lorsqu'un utilisateur demande par une interaction vocale le changement du paramètre d'une iso-surface où, au-delà de l'interaction, le système va nécessiter un certain temps pour calculer l'amas de facettes associé à cette nouvelle valeur d'iso-surface.

De façon générale, tout traitement d'événement peut prendre un temps indéterminé. Ainsi, nous risquons d'augmenter le temps entre deux rafraîchissements. C'est pourquoi nous proposons de rafraîchir l'affichage de manière indépendante par rapport au traitement des événements. Cela pose tout de même des problèmes de synchronisation entre ces deux composants fondamentaux de l'application. Par exemple, nous devons tenir compte du traitement des événements de mouvement de la tête de l'utilisateur pour mettre à jour les projections.

4.2 Gestion des focus

Nous avons vu, lors de la présentation de l'EVserveur, que nous avons implémenté un pilote X-window afin d'éviter d'être gêné par le focus. Cependant, il faut tout de même éviter que lorsqu'une touche du clavier est enfoncer pour taper une commande sur une console, les sous-applications s'occupant de la partie immersive de l'application réagissent alors qu'elles ne sont pas concernées. Ainsi, nous avons dû repenser le concept de focus. Il est important de rappeler que les événements arrivant à l'application sont contextualisés à une fenêtre. De plus, chaque événement est transmis selon deux modes distincts.

Evénement dupliqué Ces événements sont envoyés à la première fenêtre de chaque utilisateur du noyau géométrique. Dans ce cas, quel que soit le focus, chaque « utilisateur » autonome du noyau géométrique reçoit cet événement.

Evénement focusé Ces événements ne sont envoyés qu'à la fenêtre ayant actuellement le focus X-window. Un des intérêts majeurs de cette solution est de permettre, par exemple, d'implémenter des *arcballs* par rapport au référentiel propre de la fenêtre courante. Cependant, ces événements peuvent ne pas être transmis à l'application dans le cas où aucune des fenêtres n'aurait actuellement le focus X-window. Ainsi, un seul utilisateur recevra l'événement.

Comme le focus est directement corrélé à la position de la souris, ce mode de délivrance des événements n'est pertinent que pour les événements issus de X-window, c'est-à-dire ceux qui ont une signification en dehors de l'interaction immersive (clavier, souris, ...).

4.3 Débrayage de l'EVserveur

Il est clair que toute la puissance de l'architecture EVI3d provient de la gestion complète d'un environnement immersif intégrant plusieurs périphériques. Cependant, dans des cas plus simples, comme le debuggage, nous devons être capables de lancer l'application sans être obligés de charger l'ensemble de l'EVserveur. C'est pourquoi il est possible de « débrayer » le système de connexion à l'EVserveur pour le remplacer par une gestion locale des événements X-window.

Dans tous les cas (avec ou sans connexion à l'EVserveur), une boucle dans un *thread* à part gère l'ensemble des événements X-window. En effet, nous avons besoin de savoir quand on doit redessiner une fenêtre, par exemple, lorsqu'elle redevient active devant une autre fenêtre ou lorsqu'il faut la dimensionner à nouveau. Ainsi, le cas où nous travaillons hors EVserveur n'est qu'un cas particulier où les événements X-window clavier et souris sont redirigés dans la partie client de l'EVserveur pour prendre le même chemin que les événements habituels (voire Figure 3.6).

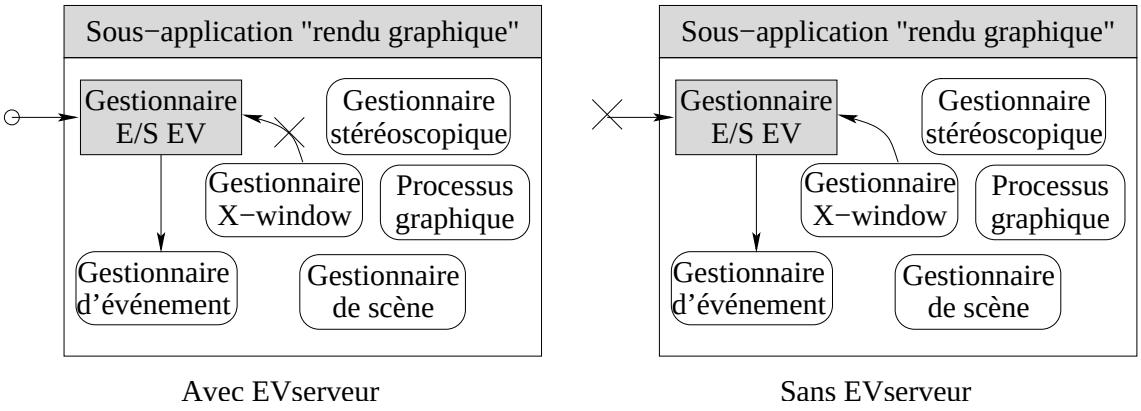

FIG. 3.6 – *Selon que l'on est ou non connecté à l'EVserveur, les événements issus de X-window sont envoyés à l'application ou non.*

4.4 L'application de contrôle

Nous avons constaté que dans la plupart des dispositifs immersifs de grande taille, il y a toujours un tuteur avec les utilisateurs et un « administrateur » derrière une console pour gérer le fonctionnement global de l'application. Nous appelons « applications de contrôle » le module terminal qui permet de lancer l'application.

D'un autre côté, comme nous l'avons vu précédemment, les applications orientées simulations scientifiques requièrent toujours un jeu de *widgets* permettant de paramétrer les différents éléments scientifiques. Par exemple, il n'est pas rare, dans les applications de mécanique des fluides, d'avoir besoin de régler les valeurs d'iso-surfaces, le champ intéressant,... C'est ainsi qu'une simple console avec un clavier ne suffisent pas pour étudier les objets scientifiques.

Au-delà de l'interaction multimodale des utilisateurs en immersion, notre point de vue est qu'il faut aussi intégrer une interface graphique complète (widget) pour permettre une interaction la plus variée possible. Cependant, cela ne peut pas se faire sans certaines concessions. En effet, il faut dissocier l'affichage graphique de la partie widget. Pour la plupart des applications, la fenêtre graphique est incluse dans une plus grande fenêtre dont la partie supérieure est occupée par un menu déroulant. Dans le cas d'applications de Réalité Virtuelle, les fenêtres graphiques sont obligatoirement contraintes par le dispositif. De plus, nous avons déjà signalé qu'il convenait de ne pas polluer l'espace de travail par des menus et autres widgets.

En s'inspirant de Gimp[9], qui sépare la fenêtre contenant les boutons de celles contenant les images à manipuler, nous proposons d'exporter la fenêtre contenant l'ensemble des widgets sur un autre écran. En fait, comme les cartes graphiques sont déjà occupées par le rendu de l'environnement immersif et qu'il est impensable de permuter les fenêtres pour

9. http://www.gimp.org

permettre d'avoir accès à un quelconque menu, nous ouvrons cette fenêtre sur un autre terminal.

Nous avons vu précédemment que l'EVserveur permettait la transmission d'une information d'un client à un autre. Ce mécanisme est utilisé pour transmettre les événements issus des widgets directement aux clients. En fait, nous avons développé un module applicatif spécifique dont le rôle de base est celui du contrôleur de l'EVserveur. Il intègre également un ensemble de menus permettant de gérer l'environnement immersif, comme par exemple de passer en stéréoscopie, de lancer la navigation ou encore de définir la calibration géométrique du dispositif immersif. par exemple, dans le cas de défaillance du système de reconnaissance vocale, il faut pouvoir tout de même piloter le dispositif immersif. Cette application a également la capacité de charger dynamiquement et gérer un ou plusieurs modules impliqués dans l'application RV. Par exemple, l'un de ces modules peut être celui servant à paramétrer les valeurs d'iso surfaces intéressantes, ou bien¡encore, les champs à visualiser.

Il faut noter que ce type de module doit non seulement être capable de générer des événements en direction de l'application, mais il doit aussi être capable de récupérer et de traiter les événements issus d'autres périphériques. En effet, imaginons que dans nos menus, un bouton permette de passer en stéréoscopie, en secours de la reconnaissance vocale. Dans le cas où l'utilisateur utilise la voix, il faut que l'application traite l'événement afin que le bouton change de représentation de sorte de refléter l'état courant du système.

Nous avons fait plusieurs choix pour la gestion de ces applications de contrôle déportée. Le premier choix a été celui associé au chargement dynamique. Il a été motivé pour prévoir le cas où l'application tourne sans EVserveur. En effet, dans ce cas, c'est l'application gérant l'affichage graphique qui charge elle-même le module. Cela permet tout de même au client de pouvoir interagir sur l'application. Il se retrouve alors dans une situation proche de celle de Gimp avec une fenêtre contenant les boutons et une autre contenant l'objet graphique. Par contre, nous sommes contraints dans cette démarche par les problèmes de compatibilité de version d'exécutable. En effet, sur une architecture de type SGI, il est impossible de charger un module 32 bits dans un noyau géométrique 64 bits et inversement. Cela peut poser des problèmes lorsque les librairies de gestion des widgets ne sont disponibles que dans un seul format (32 bits, par exemple), et que l'application centrale requiert un autre format comme celui en 64 bits pour effectuer plus rapidement les calculs sur des nombres flottants en double précision.

Un autre choix que nous avons fait a été de ne pas imposer à l'utilisateur une librairie graphique particulière (Motif, GTk, Qt, ...). La seule fonctionnalité que fournit le programme est la connexion à l'EVserveur et un *thread* à part exclusivement réservé à l'application de contrôle. Cela pose par contre, d'autres problèmes, car chaque librairie a sa propre implémentation pour autoriser le multi-thread ! Ce choix est encore en phase d'expérimentation, car nous n'avons pas une vue globale de la gestion multi-thread de l'ensemble des librairies graphiques actuellement disponibles.

5 Système de navigation

Nous avons vu précédemment que l'utilisateur doit avoir la possibilité de choisir son point de vue sur la scène. Cela ne peut être possible qu'avec la possibilité pour ce dernier de naviguer.

5.1 Travaux existants

De nombreux paradigmes existent pour permettre à l'utilisateur de se déplacer dans un monde virtuel. Certains consistent à avoir une interaction centrée objets, d'autres tiennent compte d'une analogie du monde virtuel par rapport à l'univers réel, d'autres s'appuient sur des métaphores de vols aériens. C'est en relation avec ces trois axes que nous allons faire le point des approches existantes.

A Navigation autour d'un objet

Certains systèmes de navigation visent à faire tourner les objets sur eux-mêmes. Il est intéressant de constater que l'on a besoin de composer ces rotations avec une translation. En effet, nous devons pouvoir prendre du champ par rapport aux objets.

Rotation de base avec la souris

Cette approche consiste simplement à prendre les mouvements horizontaux et verticaux de la souris pour paramétrer les composantes θ et ϕ du point de vue de l'utilisateur exprimé en coordonnées sphériques. Cependant, ce système présente plusieurs problèmes :

- si l'utilisateur est juste au-dessus de pôle Nord de l'objet, en jouant sur la composante horizontale de la souris, l'objet tournera autour d'un axe orthogonal au plan de l'écran.
- si l'utilisateur continue pour passer de l'autre côté, en se plaçant au-dessus de l'équateur, une translation horizontale de la souris aura un effet inverse.

Arcballs

Pour pallier aux problèmes de la rotation associée à la souris, [Sho92] propose l'utilisation des Arcballs. Nous en avons déjà parlé, mais je dois en préciser la philosophie. Ils considèrent aussi bien les concepts mathématiques que les facteurs humains. La direction de rotation de l'objet correspond à celle du déplacement de la souris. Cependant, contrairement aux autres méthodes basées sur la souris, il n'y a pas de phénomène d'hytérésis. Cela facilite l'annulation de rotations incrémentales.

Manipulation de l'objet

Dans les systèmes précédents, nous manipulions le point de vue pour faire tourner l'objet. Il existe un autre mode de « navigation », dans lequel l'utilisateur prend l'objet

et le manipule pour le déplacer dans le monde virtuel. Ce mode d'interaction requiert un capteur de position et une solidarisation de l'objet à ce capteur.

B Navigation dans le plan

Certaines scènes sont composées d'objets « isomorphes » à ceux de l'univers réel. C'est le cas, par exemple, de scènes architecturales ou de scènes issues de relevés topographiques.

Ces navigations intègrent tout de même des changements d'altitude. En effet, dans le cas de scènes architecturales, on peut toujours changer de niveau (escaliers, ascenseurs, ...). Ainsi, avec ce type de navigation, l'application doit être capable de gérer les détections de collisions avec les escaliers ou les murs de l'ascenseur pour permettre une élévation de l'utilisateur cohérente avec le moyen utilisé pour le changement d'« étage ».

Treadmil

Le *treadmil*, ou *shopping cart metaphor* [Jr.88], consiste simplement à mettre un tapis roulant [10], dans l'environnement immersif. Ce tapis permet de se déplacer dans le monde virtuel en marchant réellement. Dans les faits, les utilisateurs ajoutent une barre pour permettre de changer la direction de la marche. Cette méthode a cependant un défaut majeur : il introduit un objet occultant qui parasite la vision dans le cas d'environnements immersifs avec un plancher stéréoscopique.

Walking-in-place

Une extension du *treadmil* est de supprimer l'aspect matériel du tapis roulant [SSU93, FPB88], ce qui peut paraître contradictoire avec le concept même. En fait, la solution utilisée par les concepteurs est d'utiliser un système de reconnaissance des mouvements de la tête pour analyser si l'utilisateur simule la marche.

Ainsi, pour avancer, l'utilisateur marche sur place et le système est capable de « déterminer » la vitesse à laquelle il avance. Outre l'absence d'un système matériel encombrant, il permet également de définir la direction dans laquelle avancer. En effet, il suffit de suivre l'orientation de l'utilisateur au moment où il simule sa marche.

Wand

Le *Wand* est un périphérique contenant plusieurs boutons, un capteur de position à 6 degrés de liberté et un mini-joystick : beaucoup d'environnements immersifs se servent de ce type de périphérique.

L'utilisation la plus courante est celle que tous les jeux vidéo mettent en œuvre : lorsque l'on pousse le joystick vers l'avant ou vers l'arrière, cela revient à faire avancer ou reculer l'utilisateur. Une action à droite ou à gauche impose une rotation à l'utilisateur dans la direction demandée.

10. treadmil en anglais.

C Navigation dans l'espace

Les scènes isomorphes à l'univers réel ne sont pas les seules possibles dans le monde virtuel. C'est d'ailleurs rarement le cas pour les applications scientifiques. Ainsi, il faut pouvoir se désolidariser du sol.

Si précédemment, une simple interaction continue à 2 degrés de liberté pouvait suffire, il faut maintenant une interaction à 6 degrés de liberté pour pouvoir tourner dans tous les sens et se déplacer dans toutes les directions.

Déplacement par débrayage

Il ne faut pas oublier que l'utilisateur peut se déplacer à l'intérieur du véhicule. Ainsi, il peut « naviguer » dans le monde virtuel, mais seulement pour la partie qui est en émergence par rapport aux écrans du véhicule. Dans ce cas, le véhicule ne suit pas le mouvement de l'utilisateur. Lorsque l'utilisateur reviendra à sa place initiale, il retrouvera le point de vue qu'il avait alors sur la scène.

Ainsi, l'idée est de débrailler le véhicule [WO90a]. C'est-à-dire que lorsque l'utilisateur le désire, le véhicule suivra son mouvement. Et il le bloquera lorsqu'il désire se remettre à une position lui permettant d'« aller » plus loin. En fait, ce mode est analogue à celui de soulever la souris 2d pour lui redonner de l'amplitude de mouvement.

Navigation par le vol

La méthode du « *Flying* » consiste à pointer une direction avec un capteur à six degrés de liberté. Ainsi, l'utilisateur suivra un déplacement linéaire dans la direction pointée. Cette méthode nécessite, cependant, un système de mise en service et d'arrêt de déplacement dans la direction voulue.

Ce type d'interaction ne permet pas de changer les composantes de tangage et de roulis. Ainsi, l'utilisateur ne peut pas envisager de changer de point de vue pour regarder vers le haut ou bien de tourner autour d'un axe parallèle au regard. Ceci n'est pas trop gênant pour des scènes virtuelles ayant une notion de haut et de bas. Au demeurant, dans le cadre d'applications requérant des survols, telles que des applications moléculaires ou particulaires, la notion de vol est sans fondement.

Extension des systèmes plans dans l'espace

Nous avons vu précédemment que les systèmes de navigation à deux dimensions sont limités dans le plan. La sous-section précédente nous montrait que les systèmes de type *Flying* nécessitent un actionneur permettant de déclencher le déplacement. Pour éviter ces manques réciproques, les développeurs les ont réunis en un système de navigation complet.

Le pointeur associé au système de navigation de type *Flying* peut être celui intégré au *Wand* ou celui associé à un capteur de mouvement dans la main de l'utilisateur. Dans le cas du *Wand*, un bouton peut servir à changer l'orientation du point de vue dans le monde virtuel.

Widgets

Les *widgets* sont souvent utilisés dans le cas de la navigation. Par exemple, DIVE introduit des boutons 3d dans le monde virtuel. En fonction d'un « clic » l'utilisateur peut décider de se déplacer dans telle ou telle direction.

Si ce type de contrôle des navigations peut être nécessaire, il est de notre point de vue loin d'être « naturel ». En effet, nous avons précédemment parlé de la substitution des interactions. Ainsi, comme nous l'avons vu section 2.5 du premier chapitre, il n'est pas ergonomique de substituer une interaction de type continu par une autre de type discret. Ainsi, ce type de navigation, dans le plan ou dans l'espace est difficilement envisageable.

Marqueurs spaciaux

Les *marqueurs spaciaux*, ou landmarks en anglais, sont en fait issus d'une pré-étude de la scène où l'utilisateur aura préenregistré certaines positions optimales pour étudier l'objet.

Bien que ce système ne soit pas à proprement parler un système de navigation, il permet tout de même de visiter le monde virtuel. Contrairement aux autres de navigation qui sont continus, celui-ci est discrétisé.

De plus, il requiert toujours, en amont, un système de navigation continue pour sélectionner les points de vue. Inversement, les systèmes de navigation classiques utilisent généralement ce type de marqueurs spatiaux. En effet, il peut toujours être utile de pouvoir retrouver un point de vue stable lorsque l'utilisateur se « perd » dans son monde virtuel. Ce système peut évoluer vers une possibilité de définir les marqueurs dynamiquement au cours de la navigation pour pouvoir les retrouver ultérieurement.

De plus, deux modes permettent de rejoindre un point de vue prédéfini :

- l'utilisateur peut se rendre instantanément au point de vue demandé ;
- il peut aussi « visiter » le monde virtuel en se rendant au point de vue considéré.

D Conclusion sur les travaux existants

Les différents modes de navigation dans un monde virtuel devraient tous être disponibles au sein d'une même application. En effet, selon les situations, l'utilisateur pourra préférer un système de type *Wand*, des widgets 3d ou autres.

De plus, il faut pouvoir combiner n'importe quel type de contrôle continu de la navigation avec des systèmes de marqueurs spatiaux pour les raisons évoquées ci-dessus.

Cependant, aucun des systèmes spécifiés précédemment ne satisfait nos critères de navigation complètement libre. Ainsi, en se basant sur l'étude bibliographique ci-dessus, nous avons développé un nouveau système de navigation permettant de le faire.

De plus, hormis le walking-in-place, tous les systèmes de navigation requièrent la monopolisation de la main par un périphérique orienté navigation. Cela est contraire à notre philosophie consistant à dédier chaque modalité à son mode de fonctionnement le plus

naturel. La main étant souvent accaparée par la manipulation des objets, ne peux en plus gérer la navigation.

5.2 L'origine de la métaphore du véhicule

Notre métaphore du véhicule est issue du concept de volant 6 DDL constituant notre système de contrôle des navigations virtuelles.

Dans un véhicule, le conducteur utilise des actionneurs afin de modifier sa direction ou sa vitesse. En fait, ces actionneurs sont des capteurs qui, selon la valeur du déplacement, vont amplifier ou atténuer la modification de la trajectoire. Ainsi, en fonction de l'enfoncement de la pédale d'accélérateur, la voiture va accélérer ou décélérer. Il en va de même pour le volant qui fait tourner les roues.

Dans le cas du volant, il est intéressant de constater qu'il existe une configuration du volant où la voiture continue tout droit. Ce point d'équilibre existe également avec l'accélérateur : l'échelle de modification n'est alors pas sur la vitesse, mais sur l'accélération.

C'est le concept de base de notre système de navigation : on applique à la scène l'écart entre le référentiel neutre et son référentiel courant. A noter que ce référentiel neutre correspond au point d'équilibre décrit plus haut. C'est-à-dire que le véhicule sera immobile lorsque l'utilisateur sera cohérent en position et en orientation avec le référentiel. Quand je dis que l'on applique cet écart, il faut comprendre que l'on « ajoute » l'écart à la position du véhicule[11].

5.3 Point d'application du système de navigation

Dans les faits, le capteur de position permettant de faire évoluer la navigation peut être placé sur n'importe quelle partie du corps humain.

La première position à laquelle nous pensons est celle de la main. Elle présente l'intérêt d'être proche de celle utilisée dans tous les environnements immersifs en ce sens que la plupart des systèmes de navigation sont localisés dans la main.

Cependant, nous ne voulons pas surcharger les tâches de la main. En effet, nous désirons lui réserver les tâches de préhension, désignation... C'est pourquoi nous avons préféré appliquer notre modèle au capteur associé à la position de la tête. C'est ainsi que ce système a été nommé HCnav (Head Control virtual Navigation [BDA99]).

Dans ce contexte, il est même probable que notre système sera capable de réduire les effets désagréables du *cybersickness*. Nous pensons que ce mal est amplifié lorsque l'utilisateur contrôle la navigation par la main. En effet, il n'existe aucune relation entre la main et le système vestibulaire. A l'inverse, le fait de contrôler la navigation par les mouvements de la tête doit pouvoir atténuer les effets du *cybersickness*, puisque dans ce cas, nous stimulons le système vestibulaire.

11. Le fait d'ajouter un changement de référentiel à un référentiel donné consiste à multiplier ce changement au référentiel donné.

5.4 Référentiel neutre

Le problème est que la morphologie de chaque utilisateur change. Ainsi, il est inconcevable de choisir arbitrairement un référentiel neutre, car il risque de correspondre à un échantillon relativement étroit de la population. Par exemple, si on choisit un référentiel neutre trop élevé en position, les utilisateurs de petite taille ne l'atteindront jamais. Inversement, les personnes de grande taille seraient obligées de se pencher pour s'y placer.

C'est pourquoi, nous devons demander à l'utilisateur de déterminer lui-même la position et l'orientation de son référentiel neutre. C'est ce que nous appelons la calibration du référentiel neutre. Ce référentiel doit être celui où l'utilisateur éprouve un maximum de confort et qu'il est susceptible de retrouver facilement. Cependant, il est important de noter qu'au cours de la navigation, l'utilisateur fatigue. Ainsi, il faut que le système autorise une re-calibration en cours de simulation.

5.5 Volumes de tolérance

On constate qu'au cours du temps, l'utilisateur ne peut pas respecter les paramètres du référentiel neutre (x,y,z,α,β,γ). Cela nous oblige à tenir compte de variations dans le contrôle de navigation.

Pour ce faire, nous prenons en considération une sphère de tolérance autour de la position de calibration. Tant que l'utilisateur est à l'intérieur de cette sphère, nous ne tenons pas compte de sa position pour mettre à jour son point de vue.

De même, l'utilisateur ne peut pas forcément maintenir la tête dans une orientation précise. C'est pourquoi notre système tient également compte de cônes de tolérance. C'est-à-dire que si l'ensemble des angles entre le référentiel courant et celui de calibration est inférieur à des valeurs limites, alors la composante orientationnelle n'est pas utilisée pour mettre à jour la position du véhicule.

Ainsi, l'algorithme est le suivante :

- étant donné $\Delta M_i = [\delta x,\delta y,\delta z,\delta\alpha,\delta\beta,\delta\gamma]$, les paramètres d'Euler de l'écart relatif du capteur 6 DDL à l'instant t_i ;
- étant donné $dm_i = [dx,dy,dz,d\alpha,d\beta,d\gamma]$, les paramètres d'Euler du mouvement relatif de l'utilisateur dans le monde virtuel à l'instant t_i ;

  si $\Delta M_i^j > \epsilon_j$, $j \in \{1,3\}$

  alors, calculer le déplacement translationnel de l'utilisateur dm_i^j, $j \in \{1,3\}$;

  si $\Delta M_i^j > \epsilon_j$, $j \in \{4,6\}$

  alors, calculer la rotation de l'utilisateur dm_i^j, $j \in \{4,6\}$;

  si $\Delta M_i^j < \epsilon_j$, $j \in \{1,6\}$

  alors on garde le même point de vue.

Nous verrons plus loin comment calculer le mouvement dm_i à partir de ΔM_i (section 5.11). Cependant, ce petit algorithme signifie :

a - Nous n'avons rien à faire si les mouvements relatifs du capteur 6 DDL restent à l'intérieur des deux volumes de tolérance...

b - Lorsque le capteur de position reste à l'intérieur de la sphère de tolérance mais que le capteur d'orientation sort des cônes de tolérance, alors le mouvement de l'utilisateur est purement rotationnel autour de l'utilisateur. Il est intéressant de constater que si l'utilisateur reste à la position de calibration, mais continue d'être hors des cônes, le monde tourne autour de lui comme s'il en était le centre.

c - Si le capteur de position est hors de la sphère de tolérance, mais que les directions restent à l'intérieur des cônes, alors le mouvement sera purement translationnel.

d - Enfin, si le capteur de position et orientation sort complètement de la sphère et des cônes, alors le mouvement sera composé des 6 degrés de liberté.

5.6 Transition entre la navigation et l'adaptatif

Nous avons vu que si l'utilisateur veut stabiliser son point de vue, il faut qu'il soit dans la sphère et les cônes de tolérance. En fait, il y a deux modes de fonctionnement.

Dans un premier temps, le mode standard d'observation d'une scène virtuelle se traduit dans les calculs par la projection de la scène vis-à-vis du point de vue de l'utilisateur. Dans ce mode l'utilisateur peut bouger sans problème sans crainte de perdre le point de vue sur la scène. Cependant, ses déplacements sont limités à l'espace du volume d'interaction.

C'est seulement lorsqu'il passe dans le mode navigationnel qu'il peut voyager sur de longues distances dans le monde virtuel. Ce passage entre ces deux modes peut se faire via une commande vocale.

A propos de la calibration du référentiel neutre, deux situations se présentent.

Calibration à la demande Outre que l'utilisateur peut oublier de faire cette calibration, l'inconvénient de cette approche est que'elle oblige à un déclenchement en deux temps de la navigation. Ceci n'est malheureusement pas très ergonomique pour les personnes peu familiarisées avec le système de navigation.

Calibration automatique Dans ce cas, si l'utilisateur ne prend pas garde à ce repositionner vers le centre du volume d'interaction, il risque, au cours de la navigation qu'il vient de déclencher, de se trouver limité par les frontières du volume.

5.7 Cœfficient d'atténuation

Le signal brut des capteurs ne peut pas être directement appliqué au contrôle de cette navigation. En effet, en admettant que l'utilisateur tourne la tête de 45°, en huit pas temporels, l'utilisateur a virtuellement tourné sur lui-même. Ainsi, à 24 images pas seconde, nous effectuerions trois tours en une seconde !

C'est pourquoi nous sommes obligés d'appliquer des atténuations. Cependant, nous n'avons actuellement trouvé aucun algorithme mathématique permettant de calculer simultanément l'interpolation sur les translations ainsi que sur les rotations. Plus fondamentalement, nous avons vu précédemment que pour gérer la sphère de tolérance ainsi que les

cônes, il faut être capable de dissocier le traitement de la position par rapport à celui sur l'orientation.

Atténuation sur la translation Afin de calculer l'atténuation sur la translation, nous utilisons une simple interpolation linéaire entre la position de calibration et la position courante. Nous pouvons même appliquer un cœfficient d'atténuation différent sur chacune des 3 composantes de la translation.

Interpolation linéaire sphérique Pour le calcul de l'atténuation sur l'orientation, nous utilisons les quaternions. Les quaternions sont un formalisme mathématique pour représenter les rotations. Ils sont basés sur quatre coordonnées : un angle de rotation et un axe de rotation. Il existe un ensemble d'opérateurs permettant de passer de ce formalisme à des matrices et inversement. Il est important de noter que les quaternions ne s'adressent qu'aux rotations. Ainsi, les matrices issues d'un quaternion ont une composante translationnelle nulle.

De plus, ce formalisme mathématique inclu un opérateur lui permettant l'interpolation linéaire entre deux quaternions clefs. Ces interpolateurs s'appellent les SLERP (Spherical lineare intERPolation) [Sho85]. Ils ont la capacité de faire, sur le plus court chemin rotationnel entre deux quaternions, une atténuation rotationnelle pour un paramètre u donné. Il existe cependant des limitations aux SLERP : dans certains cas extrêmes, ils peuvent introduire des oscillations, faussant ainsi l'interpolation. Cependant, ces cas ne font pas partie de notre cahier des charges.

La première étape est de convertir l'orientation de la calibration et l'orientation courante en quaternions. Ensuite, nous appliquons les fonctions SLERP sur le chemin rotationnel entre ces deux instants clefs. Enfin, nous convertissons le résultat en matrice. La matrice obtenue sera combinée à la translation atténuée dans les équations que nous verrons en section 5.11.

Notons que cette approche considère l'orientation comme un tout : contrairement à l'atténuation sur la translation, nous ne pouvons pas atténuer de façon différenciée les orientations selon chacun des trois axes. Nous en verrons les conséquences dans la section suivante.

5.8 Anisotropie de la morphologie humaine

Nous avons vu précédemment que le système ne permettait pas d'atténuer différemment les orientations selon chacun des axes. La tête a des degrés de liberté d'amplitude relativement proches en rotation, on peut se satisfaire de cette situation. Celle-ci est donc compatible avec l'impossibilité pour l'interpolation des quaternions d'être anisotrope. Cependant, ce qui est acceptable pour la rotation ne l'est plus pour la translation.

En effet, les degrés de liberté dans le plan horizontal ont une amplitude nettement supérieure à ceux de l'axe vertical. Par exemple, le champ de variation horizontale est celui que l'utilisateur peut parcourir en marchant dans le volume d'interaction. De même,

il peut fléchir les genoux pour « descendre » dans le monde virtuel. Par contre, pour « monter », il peut essayer de monter sur la pointe des pieds, mais cela ne lui donne pas un degré de liberté suffisant. Il peut aussi essayer de calibrer une position avec les genoux fléchis.

C'est pourquoi, nous avons introduit des cœfficients (K_{max}) correspondant aux amplitudes moyennes autorisées par la morphologie selon chaque axe. Ainsi, pour compenser cette anisotropie, nous normalisons l'ensemble des composants translationnels du mouvement par ces K_{max}. Cette normalisation doit être faite dans les deux directions de chaque axe.

A propos de ces cœfficients, trois remarques doivent être faites :

- La sphère de tolérance autour de la position neutre peut être définie comme des pourcentages de ces K_{max}.
- Nous tenons compte du volume d'interaction pour définir ces cœfficients. C'est-à-dire que les limites des différents périphériques peuvent entrer en jeu dans leurs définitions.
- Enfin, nous pouvons tenir compte d'une évolution de ces K_{max}. En effet, l'utilisateur peut se fatiguer et donc faire que le volume d'interaction ne soit plus identique à ce qu'il était initialement. Dans ce cas, nous devons pouvoir changer les cœfficients dynamiquement.

Un travail que nous allons entamer durant la phase de validation ergonomique du système de navigation sera de caractériser ces cœfficients en fonction de la morphologie moyenne d'un individu.

5.9 Modification des cœfficients d'atténuation

Rappelons que l'unité de mesure de l'ensemble des capteurs est le mètre. Sur une scène dont la taille totale est de l'ordre de l'angström (ie. dizaine d'atomes), le fait de visiter la scène sans appliquer d'atténuation reviendrait à étudier la Joconde depuis un Concorde en vol. Inversement, visiter une galaxie sans amplification du mouvement, reviendrait à visiter la grande muraille de Chine à dos d'escargot !

En fait, une navigation sur de petits objets induit une vitesse relative plus importante (cf. les lois de similitude). Ainsi, nous devons tenir compte de la taille de la scène pour contrôler la vitesse relative de la navigation. C'est-à-dire que la première interpolation à appliquer est de faire en sorte que la navigation ne soit pas trop rapide ou trop lente par rapport à la métrique de la scène.

Nous nous sommes rendus compte que nous pouvions avoir intérêt à changer les valeurs de ces atténuations au cours de la simulation. Imaginons que nous sommes dans une situation de revue de projet sur le prototype virtuel d'une voiture. Ainsi, les cœfficients refléteront la taille de la voiture dans son ensemble. Cependant, en cours d'étude, les ingénieurs motoristes peuvent se poser des questions sur la course des soupapes et le tuyau de l'échappement par rapport à l'espace sous le capot. Dans ce type de situation, il faut être capable de changer la granularité de la navigation pour permettre de tourner

autour de la soupape. C'est-à-dire que nous devons diminuer la vitesse de navigation en augmentant l'atténuation appliquée au système navigationnel.

Parcourir une scène ne demande qu'une atténuation des composantes translationnelles de la navigation. En effet, la distance parcourue dans la scène est dépendante de la vitesse tangentielle et non de la vitesse angulaire.

A contrario, le contrôle de la vitesse angulaire est important lorsque l'utilisateur cherche à décrire une trajectoire stable autour d'un objet. Notre système ne contrôlant que la vitesse tangentielle, la distance à l'objet observé sera une information majeure pour paramétrer la navigation. Cette distance à l'objet nous renseigne en effet directement sur la granularité de la scène intéressant l'utilisateur, à un instant donné.

5.10 Fréquence de rafraîchissement et des capteurs

Actuellement, la navigation est mise à jour à chaque nouvel affichage de la scène. C'est-à-dire que si la scène est particulièrement dense, la navigation sera lente. Dans ces conditions, la navigation est non seulement dépendante des cœfficients, mais aussi du taux de rafraîchissement. De plus, cela signifie que la navigation sera plus rapide lorsqu'il y a peu d'objets à afficher où qu'ils ne sont pas très gros (relativement à la fenêtre de visualisation), ce qui est le cas lorsque les objets sont loin ou lorsque l'utilisateur regarde un endroit peu dense de la scène.

Une autre solution serait de mettre à jour la navigation à chaque fois qu'un événement arrive. Cependant, le problème inverse apparaît : la navigation peut évoluer sans aucun feedback visuel à l'utilisateur. En effet, avec des scènes dont le temps de rafraîchissement est trop long, le principe de navigation fait que les incréments de mouvement issus des événements de tracking vont s'accumuler sans que l'on s'en rende compte. Ceci est d'autant plus accentué par le fait que l'utilisateur, ne voyant pas la scène évoluer, va amplifier son action et risque donc de perdre beaucoup plus facilement la scène de vue.

La solution optimale vise donc à garantir le temps de rafraîchissement grâce à la gestion de scène.

5.11 Aspects mathématiques

Processus principal

L'analogie du véhicule renforce la séparation du processus entre le monde réel et le monde virtuel. Cependant, notre paradigme utilise trois postulats de base.

- **Postulat 1** : La position de l'utilisateur dans le volume d'interaction correspond exactement à sa position dans le véhicule :

$$^{(Dv_i)}N_i = ^{(Dr_i)} N_i$$

- **Postulat 2** : La mise à jour de la position de l'utilisateur n'est effectuée qu'après le mouvement du véhicule dans le monde virtuel :

$$^{(Dv_{i+1})}O_i = ^{(Dr_i)} N_i$$

– **Postulat 3** : Le mouvement atténué (dm_i) est le même entre le monde réel et le monde virtuel :

$$^{(Uv_i)}dm_i = ^{(Ur_c)}\ dm_i$$

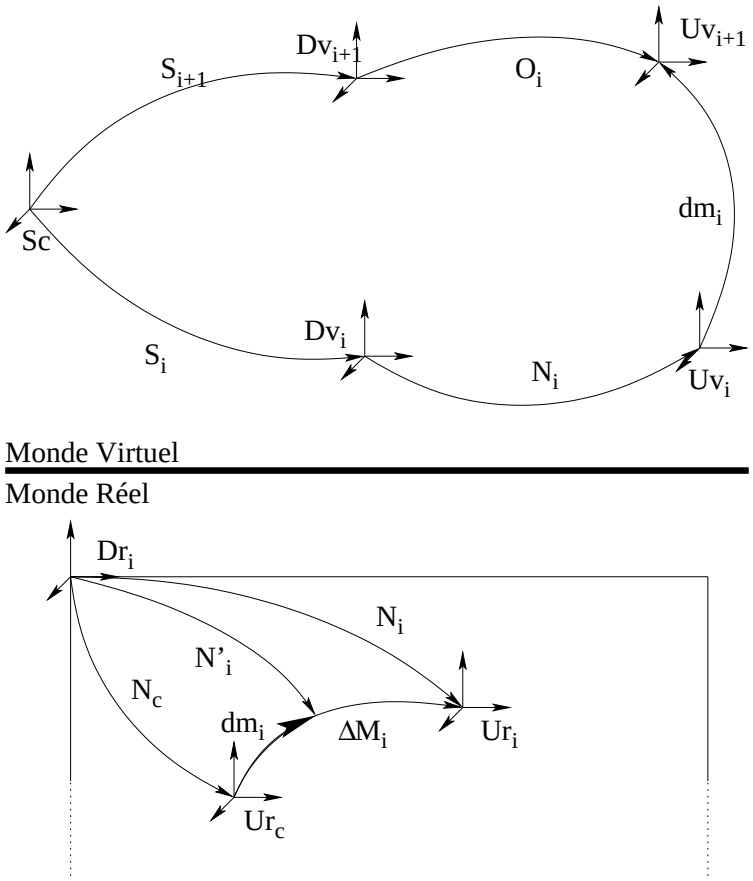

FIG. 3.7 – *Cette figure représente l'ensemble des changements de référentiels dans le monde virtuel et l'univers réel du système HCnav.*

En regardant la figure 3.7.a, nous pouvons poser :

$$^{(Sc)}S_{i+1}^{-1} = ^{(Uv_{i+1})}\ O_i.^{(Uv_i)}dm_i^{-1}.^{(Dv_i)}N_i^{-1}.^{(Sc)}S_i^{-1}$$

Après avoir changé le référentiel de O_i du référentiel (Uv_{i+1}) au référentiel (Dv_{i+1}) et avoir appliqué les trois postulats énoncés plus haut, l'équation principale de navigation devient :

$$^{(Sc)}S_{i+1}^{-1} = ^{(Dr_i)}\ N_i.^{(Ur_c)}dm_i^{-1}.^{(Dr_i)}N_i^{-1}.^{(Sc)}S_i^{-1} \tag{3.1}$$

Calcul de l'atténuation

Afin de simplifier les opérations, dm_i est décomposé entre N_c et N'_i (voir la partie inférieure de la figure 3.7.a) qui est une interpolation entre N_c et N_i.

$$^{(Ur_c)}dm_i^{-1} =^{(Dr_c)} N_c.^{(Dr_c)}N_i'^{-1} \tag{3.2}$$

En introduisant le résultat du processus d'atténuation (équation 3.2) dans le processus principal (équation 3.1), l'équation résultante devient :

$$^{Sc}S_{i+1}^{-1} =^{(Dr_i)} N_i.^{(Dr_c)}N_i'^{-1}.^{(Dr_c)}N_c.^{(Dr_i)}N_i^{-1}.^{Sc}S_i^{-1} \tag{3.3}$$

Il est important de noter que dans cette formule, chaque transformation est appliquée dans son propre référentiel (cf. figure 3.7.a). C'est pourquoi, dans la suite, nous n'indiquerons plus les référentiels d'application de chaque changement.

Globalisation aux HMDs et aux murs

Nous avons vu que le concept de véhicule s'applique aussi bien aux murs qu'aux HMDs. Ainsi, nous devons tenir compte de ces deux types de dispositifs dans le processus de navigation. Dans le cas des HMDs, nous devons appliquer le mouvement de l'utilisateur au mouvement du véhicule pour obtenir une projection cohérente de V_i^{-1}:

$$V_i^{-1} = N_i^{-1}.S_i^{-1}$$

En conséquence, le calcul global (équation 3.3) doit être découpé en deux étapes. Pour fournir le référentiel du véhicule indispensable aux traitements communs aux murs et aux HMDs, on commence par calculer :

$$S_{i+1}^{-1} = N_i.N_i'^{-1}.N_c.V_i^{-1} \tag{3.4}$$

Pour fournir le référentiel indispensable à l'affichage dans le cadre du HMD et mettre à jour l'équation 3.4, on calcule ensuite :

$$V_{i+1}^{-1} = N_{i+1}^{-1}.S_{i+1}^{-1} \tag{3.5}$$

6 Le démonstrateur multimodal

Le démonstrateur multimodal a pour objectif de démontrer toutes les interactions que nous sommes actuellement capables de gérer. Dans les faits, il s'agit du logo du LIMSI représenté en trois dimensions. Le logo étant immobile dans le référentiel de la scène représente une référence fixe du monde virtuel.

En plus du logo, nous avons placé trois objets simples : un cylindre, un cône et une théière. Ces trois objets sont ceux sur lesquels nous pouvons effectuer nos interactions. Comme ils sont indéformables, les seules interactions que nous nous autorisons sont celles associées au déplacement de ces objets.

Il faut noter que cette scène est particulièrement légère en termes de polygones. Elle est donc peu appropriée pour des études où les latences d'affichages doivent être prises en compte. Ainsi, ce démonstrateur n'est prévu que pour montrer la faisabilité d'une interaction multimodale dans un environnement immersif.

6.1 Les utilitaires développés pour le démonstrateur

Nous avons conçu plusieurs éléments supplémentaires pour simplifier la vie du développeur ! Ces éléments sont maintenant intégrés au noyau géométrique. Ils sont donc disponibles pour toutes les applications utilisant ce noyau.

Les détections de collisions

Afin de faciliter les développements, nous avons mis au point un système de détection de collisions. Notons que les scènes dont nous nous servons ont peu d'objets. De plus, tous les objets sont statiques. En fait, la seule mobilité qu'ont ces objets concerne la possibilité de les prendre et de les déplacer. C'est-à-dire qu'ils n'ont pas de déplacement propre autonome, afin de pouvoir les attraper plus facilement.

De plus, afin de simplifier les calculs, nous utilisons une boîte englobante. Cela n'est pas complètement aberrant par rapport à la réalité. En effet, avant de faire une détection de collision fine permettant de savoir si les objets intersectent, nous devons effectuer une détection « grossière » afin d'éviter de surcharger le système.

Collision avec un point Avant tout, nous devons pouvoir être capables d'attraper les objets pour pouvoir les manipuler ! C'est-à-dire que nous recevons du système de reconnaissance un signal correspondant à la préhension. Il faut ensuite définir l'objet à attraper. Pour ce faire, nous utilisons le centre de la paume de la main et nous regardons s'il appartient à la boîte englobante. Il est intéressant de rappeler que le produit scalaire du vecteur allant du centre de la main au centre de la boîte par le vecteur normal à un des côtés de la boîte correspond à la distance selon ce dernier vecteur au centre. Ainsi, une simple comparaison par rapport à la largeur de la boîte permet de savoir si le point est intérieur à la boîte.

Collision avec un pointeur laser Concernant le pointeur laser, le problème est un peu plus délicat. En effet, non seulement nous devons savoir si le pointeur traverse la boîte, mais en plus, il serait bon de connaître la profondeur de l'objet par rapport à l'origine du pointeur. La solution que nous avons choisie est en fait assez proche de l'algorithme de « clipping » de Sutherland-Hogdman [SH74] : elle consiste à tester le pointeur laser selon chaque plan de la boîte. Etant donné un plan et une demi-droite, nous pouvons connaître l'abscisse du point d'intersection depuis l'origine de la droite. De plus, il doit y avoir deux abscisses : une pour chacun des deux plans parallèles composant la frontière de la boîte selon un des trois axes. Ces deux abscisses constituent les limites du segment de la demi-droite appartenant potentiellement à la boîte. Si un point n'appartient pas à ce segment, alors il est clairement hors de la boîte.

Nous pouvons faire la même étude sur un autre axe. Si le segment associé au second axe n'a aucun point commun avec le segment du premier, cela signifie que le second ne fait pas partie de la boîte. Inversement, cela signifie également que le premier segment ne fait pas partie de la boîte. Pour savoir s'il y a un point commun, il suffit de comparer les abscisses des extrémités de chacun des segments pour voir s'ils se chevauchent. La conclusion, dans ce cas, est que la demi-droite ne coupe pas la boîte. On peut ainsi réduire le segment résultant en prenant l'intersection des deux segments précédents. En faisant la même étude avec le troisième axe, on peut savoir si la demi-droite coupe la boîte. De plus, en conservant les abscisses du segment, on est même capable de fournir la profondeur du point d'entré dans la boîte ainsi que celle du point de sortie.

Avatar de la main

Un autre outil que nous avons implémenté est l'avatar de la main. Cet avatar a pour intérêt de fournir l'ensemble des outils indispensables à l'interaction gestuelle.

Nous avons vu plus haut que les angles de chaque gant devaient être calibrés. En fait, l'avatar dépend lui aussi d'une calibration.

Il est intéressant de constater que nous avons besoin de l'élongation de chaque capteur alors que précédemment nous avons vu que la calibration de la main visait justement à convertir en valeurs angulaires uniforme ces élongations.

Quoiqu'il en soit nous devons obligatoirement connaître les longueurs de chaque articulation de la main, afin d'avoir, comme pour la calibration de capteurs, une superposition exacte de la main réelle avec sa projection dans le monde virtuel. A l'heure actuelle, l'avatar ne peut pas être calibré selon la morphologie de l'utilisateur. Les valeurs sont codées « en dur » dans la librairie.

Parmi les outils importants pour l'interaction, il y a la possibilité d'obtenir le référentiel du pointeur laser lorsque celui-ci est lié à la direction pointée par l'index. Ce référentiel est centré à l'extrémité du doigt, l'axe Y de ce référentiel donne la direction du pointeur. Cet outil, tout comme l'affichage de l'avatar, requiert que la main soit correctement calibrée.

Enfin, l'avatar de la main fournit plusieurs informations remarquables sur celle-ci. Parmi les informations que le système est capable de fournir, il y a le centre de la paume

ainsi que les positions de l'extrémité de l'index et du pouce.

6.2 Interaction de base

Les premières interactions que nous avons implémentées sur ce monde virtuel sont celles qui servent de base à tout environnement immersif : le système d'adaptation à la stéréoscopie et plusieurs systèmes de navigation.

Nous pouvons naviguer dans le monde virtuel selon deux modes : le système de navigation basé sur le *Wand*, le système de navigation HCNav. L'implémentation de ces deux systèmes est requise par les besoins d'évaluation ergonomique dont nous disposons. Nous avions initialement envisagé de faire cette validation sur une application scientifique. Cependant, elle ne peut être efficace que si elle se fait sur des systèmes simples et facilement assimilables par un utilisateur quelconque. Ainsi, la complexité de scènes scientifiques nuira au protocole d'évaluation.

6.3 Interaction gestuelle

La première interaction gestuelle implémentée, fut la saisie des objets. Il a suffit de mettre en service le module gèrant la reconnaissance gestuelle développée par A. Braffort et B. Bossard au sein du groupe « Geste et Image » (cf. section B du chapitre précédent). Cette saisie est basée sur le geste reconnu et la configuration de la main. Le centre de la main est transmis au système de détection de collisions. Ce modèle d'interaction est difficilement compatible avec les systèmes de navigation monopolisant la main. En effet, pour prendre un objet, il faudrait poser l'interacteur servant au contrôle de navigation.

Nous avons aussi implémenté la modalité du pointeur laser. Il utilise le système de reconnaissance et l'utilitaire de gestion de l'avatar humain. Il permet de désigner des objets. Il peut également servir à déterminer une direction de navigation dans le cas de l'évolution des systèmes de navigation plans.

6.4 Interaction vocale

Les interactions vocales ont été les premières interactions non-standards que nous avons intégrées. Cependant, la maîtrise des problèmes de reconnaissance fut longue à obtenir : même si le système est dit polyvalent, il y a des mots qu'il reconnaît plus que d'autres. Par exemple, le terme « relief » a dû être préféré au terme « stéréo » en raison du mauvais score de reconnaissance de ce dernier. De même, il a fallu prendre les mots « Calibrer » et « Navigation » qui ont de meilleures chances d'être reconnus que « Calibration » et « Naviguer ». Dans les faits, le vocabulaire du démonstrateur fut trouvé expérimentalement.

6.5 Interaction multimodale

Viavoice, le système de reconnaissance vocal commercial le plus répandu, ne fournit pas la trame temporelle d'une phrase reconnue. Il ne fournit que la date de début ainsi que la date de fin. De plus, la référence temporelle de Viavoice est relativement fluctuante.

Pour ces deux raisons, nous n'avons pas pu implémenter de système de fusion multimodale optimal. Par exemple, on considère que le mot désignant un objet est à peu près au milieu de la phrase. Pour définir la date du mot de désignation, nous prenons la moyenne entre le début et la fin de la phrase. Ainsi, l'objet désigné sera celui qui aura été en intersection avec la main ou le pointeur laser issu du doigt à l'instant le plus proche de cette date.

La fusion multimodale que nous avons implémentée dans ce démonstrateur ne couvre pas l'ensemble des contraintes. En effet, dans l'absolu, nous devons être capables d'attraper un objet tout en naviguant. Dans les faits, il faudrait transmettre la position et l'orientation du véhicule à chaque instant au système de fusion multimodale. Les informations concernant le véhicule suffisent car tous les périphériques sont définis dans ce dernier et le gestionnaire de fusion collecte déjà les informations issues de tous les périphériques. Au demeurant, nous n'avons pas encore implémenté cet envoi.

L'interaction multimodale la plus évoluée du démonstrateur consiste à désigner un objet dans le monde virtuel et à demander à l'ordinateur de le mettre dans la main de l'utilisateur. En d'autres termes, il s'agit d'une interaction illustrant le fameux paradigme du « pose ça ici » [Bel95]. Dans les faits, un pointeur laser se matérialise à l'extrémité du doigt lorsque le système de reconnaissance gestuel signale une désignation. Ensuite, en prononçant la phrase, l'utilisateur déplace le pointeur laser afin de lui faire rencontrer l'objet (la boîte englobante de l'objet apparaît). Mais, tant que l'ordinateur chargé de reconnaître le langage n'a rien reconnu, le système ne réagit pas. Au moment où l'ordinateur a reconnu la phrase clef, il émet un « bip » et l'objet vient automatiquement se caler dans la main. L'effet est d'autant plus impressionnant lorsque l'utilisateur a déplacé son pointeur laser vers un autre objet et que c'est l'objet désigné au moment de la diction qui arrive dans la main.

6.6 Interactions combinées

Les imprécisions sur le système de reconnaissance vocale ne permettent pas d'obtenir un système fiable lorsque le véhicule se déplace dans le monde virtuel. Ainsi, nous ne nous permettons pas de combiner la fusion multimodale avec une navigation virtuelle.

Par contre, notre système permet de combiner cette navigation avec une préhension d'objet. Ainsi, on peut prendre un objet et naviguer dans le monde virtuel tout en le gardant dans la main. Ce mode d'interaction est par exemple très utile dans le cas où la scène est composée de plusieurs objets éparpillés et que nous souhaitons les regrouper tous en un endroit donné. Il suffit d'aller les chercher un par un et de les prendre sous le bras pour les remettre à leur place. Ce type de manipulation n'est possible qu'avec le système de navigation que nous avons développé. En effet, dans le cas du *Wand*, nous serions obligés de « lâcher » l'objet pour pouvoir prendre l'interacteur. Cela est évidemment impossible dans le cas d'un geste de préhension à deux mains. Cependant, cela peut poser un problème lorsque le capteur servant à prendre les objets est également utilisé pour désigner une direction de navigation.

7 Conclusion sur le noyau géométrique de EVI3d

Nous avons vu que le noyau géométrique permettait de gérer complètement tout dispositif immersif. Ces fonctionnalités de base sont communes à un grand nombre de librairies. Cependant, parmi toutes celles-ci, peu apportent des solutions satisfaisantes en termes d'applications scientifiques. Par exemple, CAVElib et VR juggler ont tendance à monopoliser l'ensemble des ressources de la machine. D'un autre côté, ils ne permettent pas d'entrer dans le code pour ajouter un système de synchronisation d'affichage permettant de contrôler la latence totale du système. Notre noyau géométrique fait d'ores et déjà partie de plusieurs applications. Nous reviendrons sur ces dernières dans les section 4 et 5 du chapitre suivant. L'étude de la latence totale du système est l'un des éléments qui manque dans notre noyau géométrique. C'est pourquoi nous sommes actuellement en train de travailler sur la mise au point d'un protocole d'étude de la latence globale. De plus, nous pensons augmenter notre noyau géométrique de la capacité de rendre sur des écrans portables. Cela assurerait que notre système soit polyvalent en permettant l'utilisation de tout type de dispositif. En fait, en réfléchissant plus avant, on se rend compte que les casques immersifs ne sont qu'un certain type d'écran embarqué. Parmi les améliorations du système, nous aurons toujours deux types d'écran : les écrans fixes par rapport au véhicule et ceux attachés à un périphérique de capture. Cependant, ils couvriront l'ensemble des besoins actuelles.

Au niveau du système de navigation, la solution que nous proposons permet de naviguer dans un monde virtuel sans monopoliser la main. Il y a encore des lacunes dans notre système de navigation : à l'heure actuelle, il nous est impossible d'empêcher un utilisateur de traverser une paroi. Ainsi, nous allons travailler à ajouter un système permettant de contraindre la navigation. Nous aborderons ces points dans la section 7.1 de la page 158.

Chapitre 4

Validations du système EVI3d et perspectives

Sommaire

1 Introduction

Ce chapitre est consacré à l'implémentation et aux validations des systèmes mis en place dans le cadre de ma thèse. De plus, je ferai le point sur les perspectives pour l'évolution de mes travaux.

Concernant l'implémentation, je vais commencer par préciser le mode de fonctionnement de certains périphériques spécifiques. De plus, je montrerai les évaluations du système de transmission événementielle. Je ferai le point sur l'implémentation du noyau géométrique (aussi bien au niveau de la gestion de l'affichage qu'au niveau de la gestion des événements). J'évoquerai ensuite des problèmes que nous avons rencontrés avec certaines applications, problèmes qui nous ont conduit à faire évoluer le mode de fonctionnement de l'application de contrôle. Pour valider mes travaux je montrerai comment on peut utiliser l'interaction multimodale au sein d'un démonstrateur. J'aborderai aussi l'implémentation de deux applications scientifiques sur le noyau géométrique. Enfin, au niveau des perspectives, je ferai le point sur le canal flux de données de l'architecture EVI3d ainsi que sur le système de navigation.

2 Canal événementiel

Nous avons vu précédemment que le canal événementiel permettait de gérer les événements de Réalité Virtuelle. En terme de validation, nous avons réalisé un certain nombre de travaux connexes. Par exemple, le mode de fonctionnement de certains périphériques impose des contraintes ou pose des problèmes qu'il faut lever. Par ailleurs, nous devons faire le point sur les performances de ce canal.

2.1 Détails de certaines implémentations

L'ensemble des périphériques de Réalité Virtuelle ne pose généralement aucun problème avec les systèmes standard de gestion des périphériques. Par contre, les problèmes surgissent lorsque nous utilisons des périphériques qui ne sont pas intialement prévus pour la RV. Ainsi, les systèmes de reconnaissance imposent généralement des contraintes supplémentaires.

A La reconnaissance vocale

La parole, contrairement aux périphériques de RV n'est pas basée sur un domaine numérique borné. En effet, le champ de variation est infini et fait partie d'un ensemble qui ne peut être quantifié. C'est pour cela qu'il faut configurer les systèmes à apprentissage générique pour leur donner le vocabulaire qu'ils doivent reconnaître.

Dans les faits, les systèmes de reconnaissance génériques requièrent une configuration. Cette information n'est pas connue a priori du pilote de périphérique. C'est pourquoi, cette information doit être fournie au pilote dynamiquement. Ainsi, au-delà des messages issus de l'EVserveur, il faut permettre à un client de renvoyer un message directement au pilote. Cette fonctionnalité a été l'une des premières implémentée.

Cependant, afin d'éviter que les messages issus du client ne soient défectueux, nous avons ajouté la possibilité pour chaque pilote d'implémenter un bout de librairie librement accessible au client. Le pilote de la voix est le premier a avoir disposé de cette fonctionnalité. Le client peut ainsi définir le vocabulaire du module de reconnaissance en affectant à chaque mot à reconnaître un identifiant. C'est cet identifiant qui lui sera transmis à chaque fois que le mot en question sera reconnu.

Par ailleurs, pour la fusion multimodale dont nous parlerons section 5, il faut que le système de reconnaissance vocale soit capable de reconstituer la séquence temporelle de la prononciation de la phrase. Ainsi, il doit pouvoir, pour chaque mot, définir sa date de début et sa date de fin dans la phrase prononcée.

B La reconnaissance gestuelle

Nous avons vu plus haut que nous avons implémenté une hiérarchie de classes d'événements. Cette hiérarchie intègre un événement gant numérique. Nous avons eu plusieurs problèmes avec ce type de périphérique. En effet, le nombre de degrés de liberté pour la main peut varier du simple au triple. Ainsi, les gants 5DT fournissent 7 DDL alors que les gants

Cyberglove peuvent aller jusqu'à 22. C'est pourquoi notre événement intègre un tableau de dimension non prédéfinie avec un masque précisant si l'information est disponible dans le gant pour une articulation donnée.

Comme les gants changent en terme de nombre de degrés de liberté fournis, les paramètres utilisés lors de l'apprentissage pour un gant ne correspondent pas à ceux utiles à un autre type de gant. Par exemple, nous nous sommes aperçus que la caractérisation du geste de désignation se faisait principalement à l'aide de l'abduction entre l'index et le majeur. Cette information n'est disponible qu'avec certains gants haut de gamme. Ainsi, pour les gants ayant moins de degrés de liberté, le système devra caractériser par d'autres informations de geste, comme l'opposition de l'index aux autres doigts.

Le système de reconnaissance gestuelle génère lui aussi un événement. La solution choisie est de ne pas dupliquer les valeurs associées au gant dans l'événement associé issu du système de reconnaissance. Par contre, nous récupérons la date de l'événement du gant qui arrive pour estampiller l'événement de reconnaissance généré. Dans les faits, cela signifie que lorsqu'un événement de reconnaissance gestuelle arrive, il faut rechercher l'événement gant associé sur la base de la date. Par exemple, lorsque l'on a un geste de désignation, il faut trouver la configuration du gant pour pouvoir connaître la position et l'orientation du pointeur laser.

C La connexion de périphériques via le réseau

Les systèmes de type MotionStar^{TM}requièrent une grande bande passante entre le périphérique et l'ordinateur qui le gère. Cela est dû au nombre de périphériques autonomes ainsi qu'à la fréquence des acquisitions. Cette bande passante ne peut plus se satisfaire des *bus* habituels. Ainsi, les concepteurs de ces périphériques les connectent directement au réseau. Dans ce cas, un client réseau doit se connecter au serveur du périphérique[1]. Avec l'EVserveur, nous pourrions imaginer avoir un client par machine, chaque client requérant les informations de ce périphérique au sein de l'application de réalité virtuelle. Cependant, les protocoles prévus pour le MotionStar^{TM}n'autorisent qu'un seul client. C'est pourquoi, un seul EVserveur capture les événements issus de ces périphériques et les transmet, après les avoir datés, dans le flux standard des événements.

Dans les faits, les concepteurs du MotionStar^{TM}utilisent un ordinateur de type PC dans lequel ils enfichent l'ensemble des cartes nécessaires pour la capture. Ensuite, grâce à un logiciel propriétaire, ils distribuent sur le réseau l'information dont nous avons besoin.

Malheureusement, nous ne pouvons pas implémenter un EVserveur sur la machine livrée avec le périphérique étant donné que l'OS n'est pas approprié à la gestion multi-tâches de périphériques. De plus, aucun pilote de périphérique n'est disponible pour ces cartes sous un OS plus évolué.

Cependant, c'est une chose sur laquelle nous restons en veille. Nous aurions en effet un grand avantage à utiliser un EVserveur en local. Par exemple, avec le logiciel actuellement utilisé, il est impossible de changer un paramètre du MotionStar^{TM}sans interrompre

1. Les périphériques sont généralement reliés au calculateur par un *bus* série de type « RS232 ».

l'ensemble des acquisitions.

2.2 Evaluation des performances du système

Nous avons vu dans la section E, page 22 que la latence du système peut être un phénomène limitant dans le cadre d'applications de RV. Cependant, avant de pouvoir évaluer ces latences, il faut que nous soyons capables de la mesurer. Cette mesure peut être réalisée au niveau de la synchronisation temporelle distribuée par l'EVserveur. Ainsi, nous commencerons par étudier la précision de cette synchronisation. De plus, nous ne pouvons pas encore contrôler l'ensemble des latences du système. Par contre, nous pouvons essayer de minimiser le transfert d'informations entre le périphérique et le noyau géométrique.

A Précision de la datation

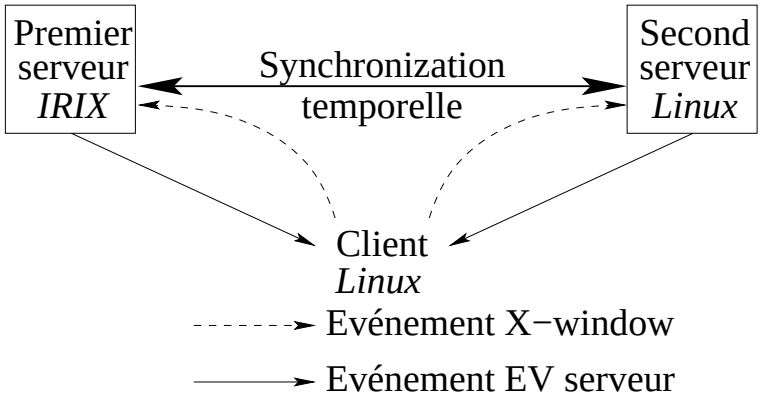

FIG. 4.1 – *Système utilisé pour évaluer la précision de la synchronisation temporelle des différents nœuds.*

Pour tester la précision du système de synchronisation temporelle, nous avons utilisé trois calculateurs de la plate-forme de RV&A (voir la figure 4.1). Un client de l'EVserveur et un `nœud EV` sont placés sur deux machines distinctes gérées par Linux. Ce serveur est lui-même connecté à un `nœud EV` maître (du temps universel) situé sur une troisième machine qui tourne sous IRIX. Enfin, un pilote de type clavier est utilisé sur chacun des deux serveurs. Le protocole consiste à effectuer une interaction clavier sur l'ordinateur où est situé le client, puis à comparer l'écart temporel existant entre les deux événements produits par ces pilotes respectivement sur les serveurs maître et esclave.

Sur une centaine de mesures de ce type, la latence moyenne obtenue est de $(0{,}9 \pm 0{,}1)$ ms, ce qui est 10 fois inférieur au seuil de notre cahier des charges.

Même si le résultat de cette évaluation paraît plutôt positif, il convient d'observer que nos conditions expérimentales ne sont pas optimales. En effet, pour pouvoir réaliser rapidement ce test, le pilote de clavier utilisé par les deux `nœuds EV` est défini comme un

client X Window connecté au serveur X du calculateur où est situé le client de l'EVserveur ! En d'autres termes, ces résultats auraient dû être meilleurs, si l'on avait pris le temps d'éliminer les latences induites par ce recours « grossier » au protocole de X Window. En conséquence, il est prévu de refaire ce test avec un sysème sonore : nous allons mettre en place un système de capture sonore émettant un son « ponctuel » sur l'entrée son de deux ordinateurs. Ainsi, avec un système de seuillage, nous pourrons calculer la différence de synchronisation entre les deux machines. Dès lors, les pilotes sonores des deux `nœuds EV` pourront recevoir le même signal et l'écart mesuré sera effectivement celui de la latence du système de synchronisation temporelle entre deux serveurs dont l'un est l'esclave de l'autre.

Cependant, comme nous l'avons dit précédemment, le bon respect du seuil de 10 ms n'a qu'une valeur indicative sur la qualité du mécanisme de synchronisation temporelle. Plus que le respect de ce seuil, c'était d'évaluer la précision des datations obtenues qui importait dans ce test. C'est en effet celle-ci qui influe sur l'écart temporel minimum utilisable par le critère de cohérence temporelle de la fusion multimodale (cf. section 5, page 86). Par contre, ce seuil redevient pertinent vis-à-vis du second test, celui de la latence de la distribution des événements et messages via l'EVserveur.

Cette valeur de précision est une valeur indicative d'un seuil minimal. En effet, ces tests n'ont pas été réalisés dans le cadre d'une charge réseau importante. Ainsi, une autre évaluation sera mise en œuvre prochainement pour déterminer la précision de la datation lorsque le réseau est très chargé par des transferts de données lourds tels que des chargements de fichiers.

B Latence de transmission des événements

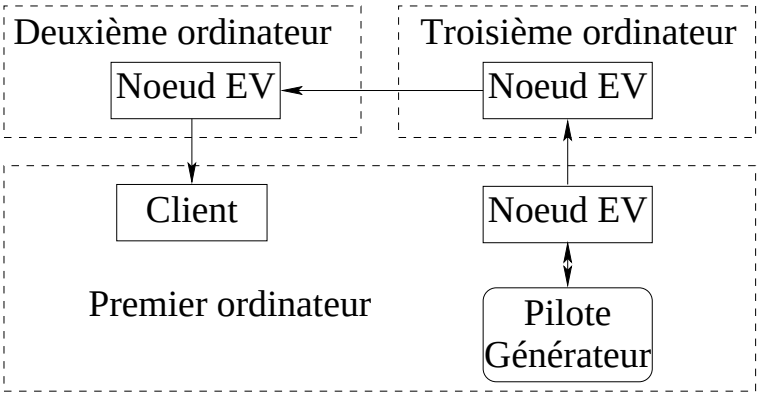

FIG. 4.2 – *Système utilisé pour évaluer la latence de transmission au travers de l'EVserveur.*

Dans ce second test, trois calculateurs sont de nouveau utilisés, mais cette fois ils tournent tous sous IRIX avec chacun un `nœud EV` : un maître et un esclave, lui-même

maître d'un second esclave. En outre, ce second esclave gère un client de l'EVserveur situé sur le même ordinateur que le `nœud EV` maître de l'ensemble du système (celui qui est le plus éloigné de ce client en terme de connexions). Ainsi, nous avons entre ces trois ordinateurs un enchaînement de trois connexions gérées par l'EVserveur (voir la figure 4.2). Enfin, le `nœud EV` maître de l'ensemble du système possède un pilote de périphérique spécial destiné à simuler une production d'événements à différentes fréquences.

Dans le cadre de ce protocole de test, les événements ne sont plus datés par le système de synchronisation temporelle de la partie `VE server kernel` des `nœuds EV`. C'est la fonction `gettimeofday` qui donne cette fois la date locale de la machine, juste avant que le `nœud EV` n'émette les événements produits par son pilote. Après avoir voyagé via les `nœuds EV` des deux autres machines, les événements parviennent au client de l'EVserveur situé sur la machine d'où ils sont partis. Quand le client reçoit un événement, il appelle la même fonction pour comparer la date de réception avec celle qu'avait cet événement au moment de sa production.

Au cours des premières évaluations de ce test, celui-ci a mis en évidence un mauvais fonctionnement du système IRIX avec des latences des transmissions à près de 100 ms, pour des fréquence de transmissions élevée (au-delà de 1 kHz). Afin de résoudre ce problème, qui n'apparaît pas sous Linux (et qui, faute de temps, n'a pas encore été évalué sous Windows), l'EVserveur inclut un traitement d'accusé de réception (ou ACK) différent de celui géré par TCP/IP. Ce traitement, qui dans le principe peut ralentir les transferts, permet d'obtenir une latence moyenne de $(3 \pm 0{,}2)$ ms pour les fréquences de production d'événements comprises entre 1 Hz et 10 kHz. Rappelons que cette latence est observée sur 3 connexions. On peut donc estimer au tiers de cette valeur la latence moyenne par connexion.

L'étape suivante de cette évaluation consiste à voir ce que deviennent les résultats de ce test en présence de machines dotées d'OS différents (IRIX, Linux, Windows). De même que pour la précision de la datation, ces tests sont un petit peu restrictifs. Nous allons donc refaire ces tests dans des conditions de charge réseau importante.

3 Gestion du dispositif immersif

3.1 Implémentation

Les librairies EVI3d fonctionnent à l'aide du polymorphisme C++. Il faut alors hériter d'une classe et sur-définir des fonctions virtuelles.

Structure multi-thread pour l'affichage

Un point fondamental est de pouvoir utiliser les différentes cartes graphiques en parallèle. Ainsi, chaque fonction d'affichage est incluse dans un *thread* à part contenant son propre contexte OpenGL.

`Init()` - Cette méthode permet à l'utilisateur de définir un ensemble de paramètres constants au cours de la session. Elle est appelée après chaque activation de la classe courante par l'ensemble des threads d'affichage. A noter que si cette méthode est appelée, toutes les autres méthodes associées à l'affichage le sont aussi.

`Reinit()` - Elle est appelée à la demande de l'utilisateur. Le rôle de cette fonction est par exemple de définir les listes d'affichages[2]. En effet, comme les listes d'affichages ne peuvent pas être partagées entre tous les contextes, le développeur peut demander l'appel de cette fonction uniquement lorsqu'il faut changer les listes d'affichages.

`CommonReinit()` - Cette méthode permet une initialisation commune à l'ensemble des initialisations (par `Reinit()`) d'un utilisateur. Elle est appelée par le thread propre de l'utilisateur. Elle permet, par exemple, de synchroniser l'ensemble des initialisations sur un jeu de données commun. Elle n'est appelée qu'avant un `Reinit()`, donc, uniquement lorsque cette dernière est activée.

`DrawGL()` - Cette fonction est appelée une fois au début de l'application et sur demande de l'utilisateur. C'est elle qui contient l'ensemble des fonctions OpenGL permettant de redessiner la scène. Il est important de noter que cette fonction est appelée autant de fois que la stéréoscopie et le dispositif l'exigent. Ainsi, il ne faut surtout pas envisager d'y effectuer une modification de la scène puisqu'elles seront faites un nombre non constant de fois.

`SwapParameter()` - Cette fonction sert à figer la scène et tous les paramètres s'y rapportant comme la position du véhicule. Il est important de noter que cette fonction est appelée dans le thread de l'utilisateur.

2. *display-lists* en anglais, système mis en place sur OpenGL pour accélérer le rendu des objets contants sur plusieurs affichages.

Schémas de l'ordonnancement des threads de l'affichage

La section précédente explique l'utilisation de chaque méthode propre à l'affichage. Cependant, il est utile de préciser, sur des cas types, comment s'ordonnancent les différents threads et les différentes méthodes.

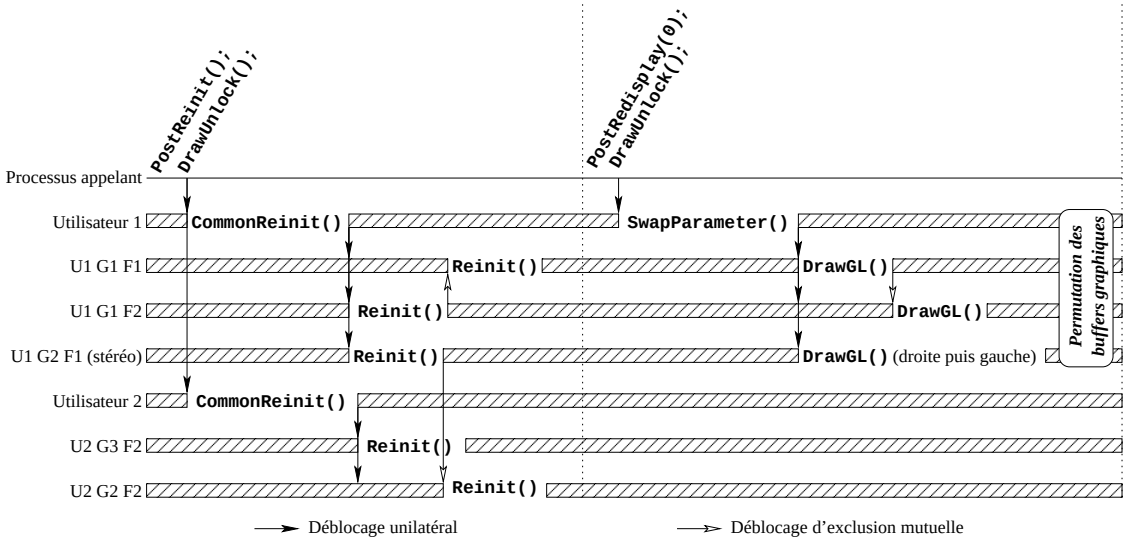

FIG. 4.3 – *Exemple d'organisation des différents threads associés au noyau géométrique.*

La figure 4.3 représente l'ensemble des tâches effectuées selon les différents appels de fonction. Le temps s'écoule de droite à gauche. Chaque ligne verticale représente un thread à part. Dans cet exemple, il y a deux utilisateurs. Le premier a trois fenètres : deux sur le même canal graphique (U1G1F1 et U1G1F2) et une troisième, stéréoscopique, sur le second canal de la machine (U1G2F1). Le second a deux fenêtres : une le canal graphique 3 (U2G3F1) et une sur le même canal que la troisième fenètre du premier utilisateur (U2G2F1). Les boites hachurées représentent les threads bloqués. Le texte entre deux zones hachurées correspond à la méthode surdéfinie appelée par le noyau géométrique. Les flèches unidirectionnelles représentent un thread qui en libère un autre. Lorsqu'elles sont vide, il s'agit d'un lock d'exclusion mutuel pour protéger la ressource qu'est le canal graphique. Les flèches à double direction représentent une exclusion mutuelle entre deux threads (principalement à cause de l'impossibilité pour deux threads d'avoir accès au même canal). La ligne horizontale du haut représente le thread appelant les fonctions. Nous avons deux enchaînements différents (`PostReinit` puis `PostRedisplay`). Le postredisplay n'est demandé que pour le premier utilisateur. Un autre point important est que la permutation des affichage n'est effectuée qu'au moment ou tous les threads d'affichages (`DrawGL`).

La figure 4.4 présente l'organisation des différents threads lorsque l'on fait un affichage bloquant. Ce cas n'a qu'un seul utilisateur. Ce dernier a deux fenètres sur deux canaux graphiques différents.

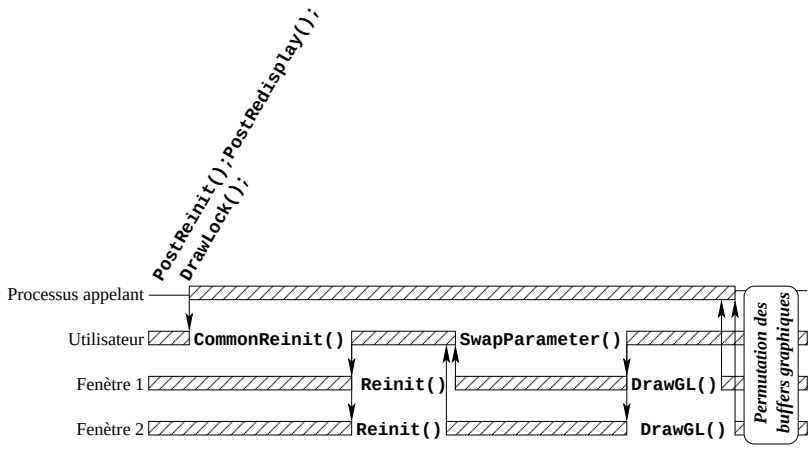

FIG. 4.4 – *Second exemple d'organisation des différents threads associés au noyau géométrique.*

Gestion des périphériques

Au-delà de la gestion de l'affichage, le noyau géométrique est également prévu pour intégrer les périphériques issus de l'EVserveur.

`SetPeripherique()` - Cette méthode permet à l'application de connaître les périphériques proposés. Elle est appelée à chaque fois qu'un périphérique se connecte à la structure de l'EVserveur (soit en cours de session, soit au démarrage, lors de la connexion de l'application à l'EVserveur).

Cette méthode est notamment utilisée lorsque nous avons des périphériques de type reconnaissance vocale requiérant une paramétrisation (voir section A, page 140). En effet, avec cette méthode, le système peut connaître les paramètres de connexion du pilote et donc lui envoyer sa configuration.

`UnSetPeripherique()` - Cette méthode est le contraire de la méthode précédente. Elle informe l'application qu'un périphérique se déconnecte de l'EVserveur.

`ProcessEvent()` - C'est la méthode de réception des événements. Les informations qui lui sont envoyées sont filtrées selon le point d'intérêt (ou focus) décrit à la section 4.1, page 77.

Application de l'échelle

Nous avons vu que nous pouvions changer l'échelle du véhicule afin de faire en sorte que l'utilisateur puisse travailler sur sa scène avec un point du vue de taille correcte sur

le monde virtuel. Cependant, il existe des scènes dont la métrique est importante. C'est par exemple le cas de scènes architecturales dans lesquelles il faut respecter l'échelle de la scène, globalement à l'échelle humaine. Nous avons aussi vu que l'utilisateur pouvait avoir des points d'intérêts différents en cours de session. Ainsi, le noyau géométrique peut changer d'échelle à volonté.

Nous pouvons définir l'échelle selon deux modes :

- *Echelle relative*. Ce mode permet d'utiliser des scènes dont l'échelle importe moins que l'étude de l'objet dans la scène. C'est par exemple le cas de scènes scientifiques. En effet, à l'échelle moléculaire, ce qui importe est l'étude de la conformation spatiale. La définition de l'échelle est faite à partir de la longueur de la diagonale de la boite englobante de la scène fournie par l'utilisateur. Le facteur d'échelle est tel que cette diagonale correspond à la diagonale d'un premier écran du dispositif immersif. Ainsi, nous sommes garantis que la scène tient intégralement dans cet écran. Ainsi, l'utilisateur aura toujours un objet visible dans son environnement immersif.
- *Echelle absolue*. Cette échelle est directement issue de la valeur fournie par l'utilisateur. Ainsi, dans le cas d'un bâtiment à l'échelle 1, l'utilisateur aura sa véritable taille. Cela permet à un utilisateur de visiter un espace tout en analysant les proportions par rapport à sa perception propre.

3.2 Evolution de l'application de contrôle

Le développement de l'application de contrôle peut poser des problèmes. Par exemple, il y a des impossibilités pour certaines librairies de cohabiter avec un contexte X-window dans un thread à part. D'autres librairies imposent l'utilisation de méthodes spécifiques qui doivent être appelées dans un ordre précis. Mais, ces méthodes sont incompatibles avec la fonction X-window `XInitThreads`, importante pour la gestion multithread de X-window.

Ainsi, nous faisons actuellement évoluer le noyau géométrique de EVI3d pour intégrer cette application sous la forme d'un processus à part. Ce travail requiert une modification plus profonde de l'architecture EVI3d. En effet, il faut que les événements issus de l'EVserveur puissent être dupliqués pour être envoyés simultanément sur les différents processus de l'application. Afin de pouvoir les dupliquer, nous avons intégré le système de partage de l'information dans un module plus complexe. Ainsi, ce dernier permet maintenant d'exporter l'ensemble des événements dans un fichier pour pouvoir « rejouer » la scène ultérieurement.

4 Validation sur le démonstrateur multimodal

Pour valider les aspects multimodaux de l'architecture EVI3d et de son noyau géométrique, nous avons implémenté un démonstrateur. Celui-ci est basé sur une scène comportant trois objets (une théière, un cône et un cylindre) manipulables et un fond solidaire au référentiel de scène du monde virtuel (voir figure 4.5). Bien que minimaliste, cette scène nous permet de valider et étudier les points clefs ainsi que les contraintes de l'introduction de la multimodalité dans des applications de RV.

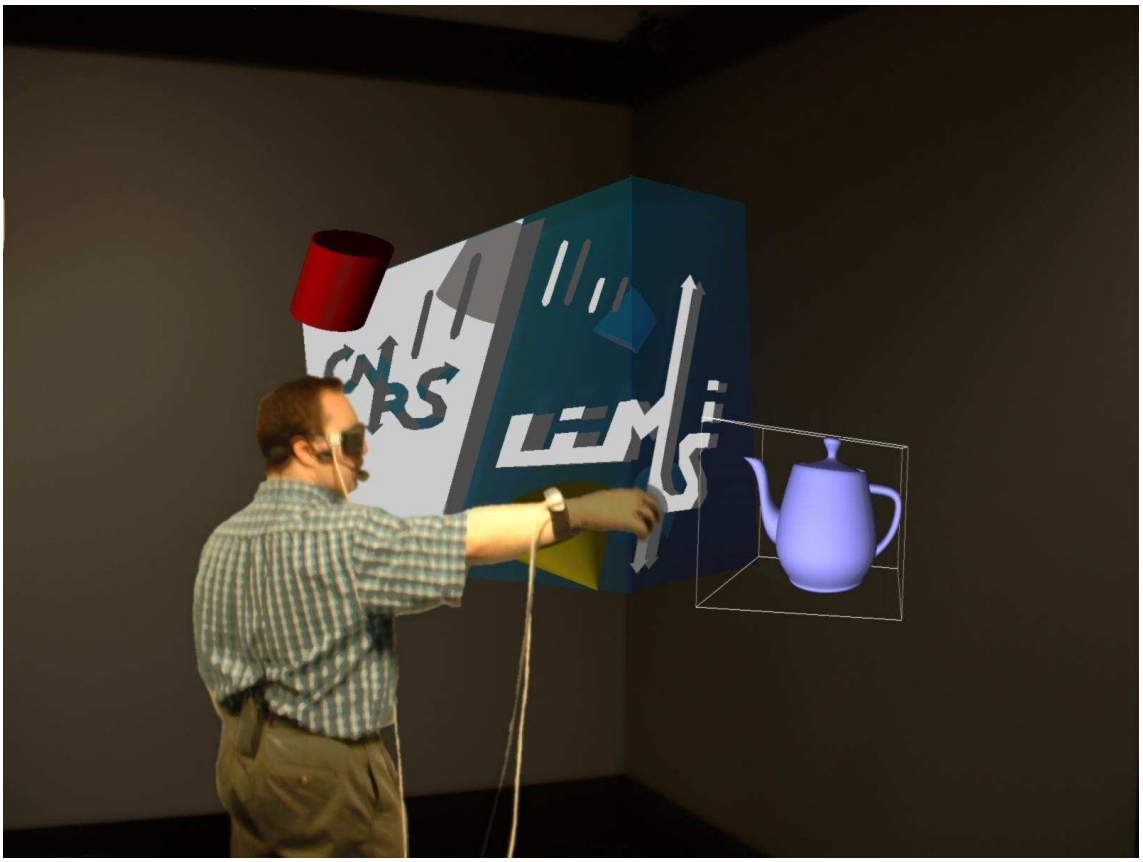

FIG. 4.5 – *Travail multimodal sur le démonstrateur.*

4.1 Evolution vers un système quadridimensionnel en réalité virtuelle

Le modèle que nous avons mis en œuvre est issu de la généralisation à quatre dimensions d'un modèle prévu initialement avec trois dimensions pour des périphériques standards (les 2 dimensions de l'espace image plus la dimension temporelle). Cette généralisation n'est pas en effet sans problème.

Le problème le plus important est celui de la stabilité et de la fréquence des données. Dans le cadre des applications initiales (cf. Mserveurde MIX3D [BKG95, BKG98]), l'interaction étant effectuée à l'aide d'une souris, voire d'un écran tactile. Or, ces deux périphériques ne sont pas en permanence solidaires avec le corps humain. C'est pourquoi la corrélation entre la voix et ces périphériques était relativement simple. En effet, dans le cadre de l'écran tactile, il suffisait de prendre la position renvoyée par ce dernier entre le temps de début et le temps de fin de génération du signal vocal. Il en va de même pour le pointeur souris qui ne bouge pas ou très peu pendant toute l'élocution.

En Réalité Virtuelle, les pointeurs sont attachés au corps humain. C'est-à-dire qu'ils sont sujets à des mouvements « parasites ». De même que dans le monde réel nous précisons nos paroles par des gestes déictiques, l'utilisateur en environnement virtuel appuiera la phrase « prend cet objet » en faisant en sorte de désigner l'objet visé au moment où il prononcera le mot « cet ». Cependant, il aura tendance à bouger, en risquant de désigner par moment d'autres objets. C'est pourquoi il ne faut pas corréler les mots avec n'importe quel objet. Dans les faits, la solution que nous avons choisie est d'analyser les occurrences de tous les objets les plus proches et de prendre celui le plus souvent désigné dans le laps de temps de la prononciation des mots de désignation.

Il existe aussi un problème d'ambiguïté. En effet, dans le cas où plusieurs objets seraient dans la ligne du pointeur laser, il faudrait permettre à l'utilisateur de préciser la profondeur voulue au moment de la commande. C'est d'autant plus problématique que les objets éloignés sont par définition difficiles à pointer. Dans le monde réel, il n'est pas rare qu'en désignant des objets, nous soyons obligés de préciser notre pensée. Nous n'avons pas trouvé de solution réellement efficace pour gérer ce type d'indetermination. Pour aider l'utilisateur à préciser sa pensée, nous ne faisons que mettre en valeur les objets soupçonnés de l'intéresser afin de l'aider à affiner son choix.

4.2 Regroupement des interpréteurs

Nous avons décrit plus haut le rôle des interpréteurs (cf. section 5.1, page 86). Dans les faits, les interpréteurs requièrent une connaissance complète de la scène pour pouvoir fonctionner correctement. Cette connaissance est par exemple indispensable pour pouvoir déterminer les détections de collisions entre d'éventuels pointeurs et des objets de la scène.

Or, nous avons vu que dans le principe, le réseau haut débit distribue l'intégralité de la scène sur tous les ordinateurs de la simulation qui la requièrent. Cependant, il est peu réaliste de donner l'accès au réseau haut débit à l'ensemble des ordinateurs gérant chaque périphérique susceptible d'intervenir dans des fusions d'événements.

Nous avons donc choisi de centraliser l'ensemble des interpréteurs sur la même machine. Qui plus est, pour la machine en question, nous avons utilisé une machine disposant d'une puissance supérieure. Ainsi, cette machine est en mesure de gérer l'ensemble des détections de collisions du système. Le fait de les centraliser permet, de plus, de factoriser un certain nombre de traitements propres tels que ceux de la gestion de scène.

5 Validation sur des applications scientifiques

Nous avons également mis en place le système EVI3d sur trois applications. Je présente ici les deux applications scientifiques.

L'utilisation d'EVI3d dans ces applications leurs apporte tous les aspects de gestion d'environnement immersifs que ces dernières ne possédaient pas. Ainsi, elles sont maintenant capables de fonctionner sur n'importe quel type d'environnement immersif. De plus, elles peuvent intégrer tous les périphériques que gère EVI3d.

5.1 Mecanique des fluides

L'application de mécanique des fluides, dont nous avons des représentations figure 4.6, permet de visualiser et d'étudier des bases de données tri-dimensionnelles de vorticité et de pression sous la forme d'iso-surfaces. Une iso-surface représente la limite entre les valeurs supérieures et inférieures à une valeur donnée.

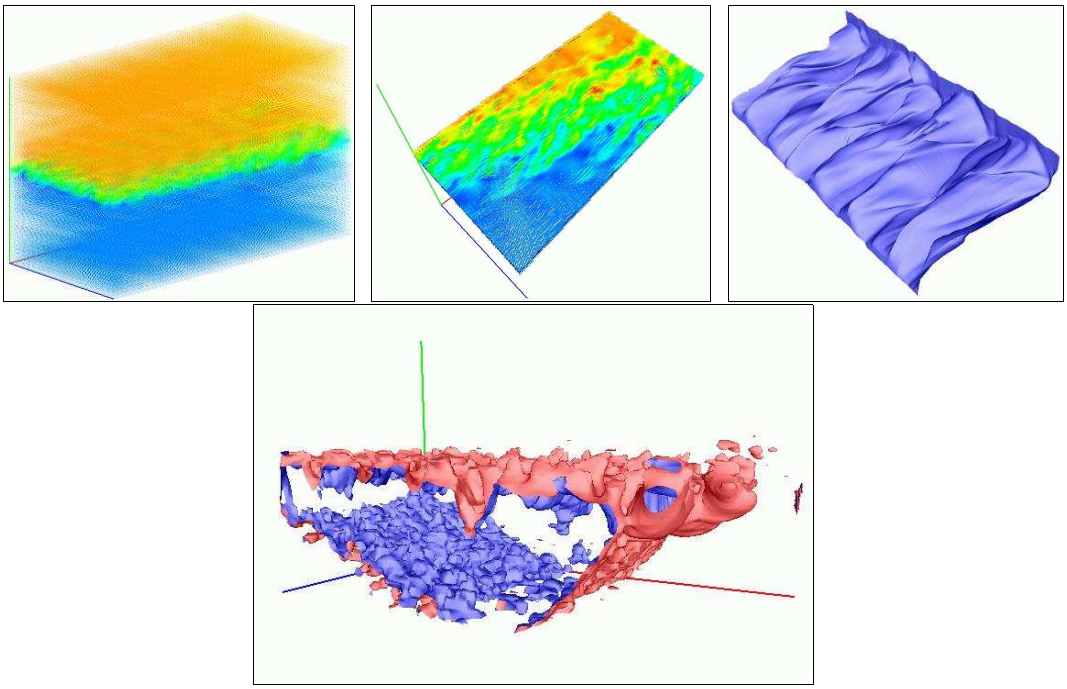

FIG. 4.6 – *L'application Mécanique des fluides développée sur la plate-forme EVI3d : Visualisation de la BD d'un instant « t » de la simulation d'un mélange compressible turbulent (haut-gauche) ; Coupe sur la jonction entre les deux couches (haut-milieu) ; Surface de propagation de particules iso-temporelles (haut-droite) ; Iso-surface de pression d'environ 500.000 facettes dont il convient de gérer l'évolution dynamique au cours de la simulation(bas).*

Elle permet de plus de faire apparaître les besoins calculatoires de ce type d'application. En effet, notre base de données actuelle est une grille tri-dimensionnelle de deux millions de cellules. Sur cette grille, on applique un algorithme tel que le *marching cubes* qui parcourt toute la grille pour créer un ensemble de facettes par cellule. De plus, notre base de donnée est extraite d'une base de données dynamique (au moins d'une cinquantaine d'instants différents). Or, afin de rendre l'application dynamique, il faut atteindre 24 images pas seconde afin que l'utilisateur ait vraiment la sensation de mouvement. Cela signifie qu'il faut arriver à calculer 24 iso-surfaces par seconde. Dans certains cas, il peut être intéressant de visualiser simultanément plusieurs surfaces. Il faut donc être capable de parcourir la base de données au moins 24 fois par seconde.

L'intégration d'EVI3d lui permet avant tout d'ouvrir une fenêtre graphique avec un contexte OpenGL puisqu'elle est intégralement basée sur EVI3d. Au niveau des périphériques que cette application utilise, il y a les capteurs de mouvements qui permettent de gérer la projection ainsi que la reconnaissance de la parole. Enfin, EVI3d lui apporte, indirectement, le système de contrôle de navigation mis au point lors de ma thèse.

5.2 ADN-Viewer

Contrairement à l'application précédente directement développée sur l'architecture EVI3d, ADN-Viewer a migré récemment sur la plate-forme EVI3d (cf. figure 4.7, page 153). Ainsi, le problème majeur de cette application fut de concilier le programme existant et toutes ses fonctionnalités avec le noyau géométrique. Par exemple, il a fallu homogénéiser le système de référentiel propre à l'application (centré caméra) avec le système du noyau géométrique (centré véhicule). Cependant, le problème majeur de cette application a été d'intégrer le système de fenêtre de l'application initiale. En effet, le but premier de cette application est de permettre l'étude de la conformation spatiale de brins d'ADN. En elle-même, cette application requiert un ensemble de contrôles permettant, par exemple, de selectionner des gènes ou des portions d'ADN. Ces fenêtres, font maintenant partie intégrante de l'application de contrôle sous la forme d'un thread à part. L'interface développée sous Qt [3] semble peu appropriée à l'utilisation en multi-thread avec des contextes X-window qui ne lui appartiennent pas. Cela nous a conduit à trouver un autre mode de fonctionnement des applications de contrôle.

Au-delà de son utilisation dans un environnement immersif, cette application a un grand bénéfice à utiliser EVI3d. Outre la gestion d'un environnement immersif de type CAVE ainsi que l'usage du système de HCnav, ADN-Viewer peut utiliser le Wand pour désigner les gènes et disposer ainsi d'une interaction plus fine sur les brins d'ADN.

3. Librairie d'interface graphique développée par TrollTech.

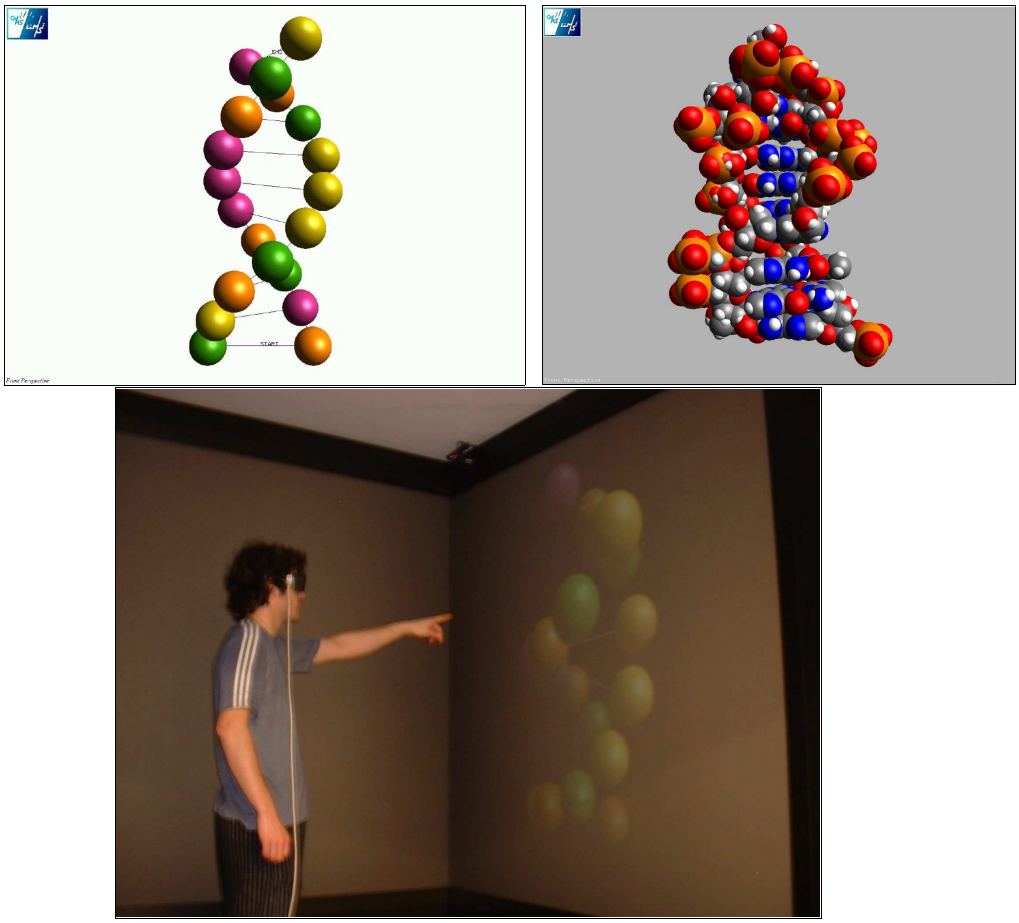

FIG. 4.7 – *L'application ADN-Viewer portée sur la plate-forme EVI3d : Partie du gène YAL004W du chromosome I de S. cerevisiae ; Représentation génique où chaque molécule A, C, G ou T est représentée par une sphère (haut-gauche) ; Représentation nucléique où chaque atome est représenté par une sphère (haut-droite) ; Analyse immersive de la représentation génique (bas).*

6 Le flux de données

Nous avons vu plus haut qu'un canal est dédié au flux de données. Dans les faits, ce canal n'a pas encore été mis en place. Cependant, nous avons dû, dès l'étude de l'architecture EVI3d, aménager une place pour ce mode de transmission. Cette section est donc issue de l'étude conceptuelle que nous avons menée dans ce cadre. Nous allons donc dans un premier temps regarder les différents cas d'utilisation d'un tel réseau. De plus, je montrerai sur un exemple d'application avec des ordres de grandeur en terme de bande passante requise. Enfin, j'aborderai les différentes technologies disponibles pour atteindre cette bande passante.

6.1 Répartition des calculs sur plusieurs calculateurs

Dans les faits, la gestion de flux de données est indispensable aux applications scientifiques. En effet, un grand nombre d'application de ce type utilise des bases de données dynamiques qu'il faut visualiser en environnement immersif. Ainsi, elles s'accompagnent de processus de calcul importants pour passer du monde virtuel (numérique) à l'espace de représentation. De plus, la production de ce monde virtuel est elle-même généralement issue d'un traitement complexe. Plusieurs choix de répartition peuvent être faits.

Par exemple, l'ensemble des calculs non graphiques peut être réalisé sur un super-calculateur. Ainsi, la liaison haut débit n'a à transmettre que la liste des facettes à afficher. Cependant, cette méthode n'est pas complètement optimale. En effet, il faut souvent permettre à l'application de connaître un certain nombre d'informations topologiques. Ces informations sont par exemple pertinentes dans le cas de la détection de collisions ou plus simplement pour les calculs d'éclairement (cf. pseudo-normale).

D'un autre côté, une situation où la synthèse du monde virtuel est réalisée sur le super-calculateur, et le passage du monde virtuel vers l'espace de représentation sur le calculateur graphique, n'est pas non plus optimale. Dans ce cas, les aspects de gestion de scène peuvent être grandement accélérés grâce à une connaissance approfondie des modèles initiaux. Par exemple, certaines équations de mécanique des fluides peuvent être résolues localement. Ainsi, une partie des calculs sur les données à régénérer peut être laissée au calculateur graphique.

Une troisième situation consisterait à effectuer l'ensemble des calculs, y compris la détection de collisions, sur le super-calculateur. Cependant, il peut y avoir plusieurs limitations : ce dernier est déjà très chargé en calculs. Ainsi, il n'est pas garanti qu'il pourra faire le calcul de détection de collisions en temps réel. De plus, d'autres composants comme le rendu sonore peuvent faire appel à la géométrie de la scène qui est, dans ce cas, enfermée dans le super-calculateur. Enfin, dans le domaine des super-calculateurs, la polyvalence des processeurs n'est plus le syandard. Ainsi, l'architecture optimale associée à une application scientifique peut être contradictoire en regard des calculs nécessaires à la détection de collisions utiles à l'interaction. En conséquence, cela pénaliserait l'application et gaspillerait de la puissance machine.

En terme de généricité de notre architecture, nous avons décidé de laisser ce choix au concepteur final de l'application. Ainsi, selon les applications, la répartition peut être plus ou moins optimisée.

6.2 Calcul sur grappe

Par ailleurs, l'autre finalité du canal flux de données de l'architecture EVI3d est de également permettre l'utilisation de grappes de PCs (*cluster* en anglais) pour le rendu en environnement immersif. Or, cela pose d'autant plus de problèmes que par définition, il faut que tous les ordinateurs associés à l'affichage disposent de la même image de la base de données. Cette image commune doit donc être distribuée.

En fait, la solution la plus simple est de considérer que tous les ordinateurs sont de puissance équivalente, donc les calculs scientifiques sont effectués en local par chaque ordinateur, et en parallèle. Cependant, cette méthode induit du gaspillage en terme de puissance CPU.

La solution à moindre coût est de répartir l'ensemble des calculs sur les machines d'une grappe. Ainsi, avec des mécanismes d'échange partiel de la base de données, on peut synchroniser l'ensemble des calculateurs. Cependant, cette méthode est très complexe et requiert une grande connaissance des mécanismes de calcul parallèle réparti. De plus, l'optique principale de mon étude consiste à remplacer un calculateur graphique dédié par un ensemble de petits calculateurs reliés en grappe. Ce n'est donc pas dans l'optique de la répartition des calculs que j'ai abordé la gestion des grappes lors de ma thèse.

L'approche envisagée vise plutôt la connexion d'un super-calculateur, chargé de synthétiser le monde virtuel, avec une grappe par l'intermédiaire d'un réseau haut débit. Dans cette optique, il faut que le protocole réseau autorise l'envoi de la même information simultanément sur plusieurs calculateurs. En effet, cela permet un gain de temps proportionnel au nombre de machines dédiées au rendu en environnement virtuel. Dans les faits, cela interdit les réseaux commutés comme les réseaux ATM.

6.3 les différents types de réseaux utilisables

Ordre de grandeur de la bande passante requise

Avant de traiter des différents réseaux envisageables, il convient de préciser la contrainte principale à appliquer à ce réseau : la bande passante requise.

Pour évaluer cette bande passante, ma référence en terme de volume de données est l'application Mécanique des fluides, vue précédemment : une scène dynamique contenant 800.000 facettes (il s'agit, par exemple, de la somme des facettes des deux iso-surfaces ayant servies pour la visionneuse stéréoscopique du LIMSI-CNRS des vœux de l'an 2000 [6]). Ces facettes sont composées de trois sommets au format numérique *float*. Ainsi, cette scène a une taille de 9,6 Mo. Or, l'unité standard de mesure de bande passante pour les réseaux est le nombre de bits par seconde. Ainsi, cette base de données de référence a une taille de 76,8 Mb/s. Cependant, afin d'obtenir une illusion du mouvement associé à la dynamicité

de la base de données, il faut générer 24 images par seconde. Ainsi, la bande passante totale nécessaire pour à un affichage de la base de données ci-dessus est de 1,8 Gigabits par seconde (Gb/s).

Evolution des protocoles standards

Depuis maintenant plusieurs décennies, le réseau le plus répandu est Ethernet. Au fil des années, il a évolué en partant de 10 Mb/s, tandis que la dernière norme commercialisée est à 1 Gb/s.

Cependant, le grand défaut de ce type de réseau est que plusieurs paquets de données peuvent entrer en collision, c'est-à-dire qu'il n'y a pas de priorités. Ainsi, plusieurs ordinateurs peuvent envoyer simultanément une information sur le médium. Il est reconnu qu'un réseau de type Ethernet ne peut être exploité qu'à 70% de sa bande passante totale. Ainsi, sur un réseau Gigabit, uniquement 700 Mb/s sont disponibles.

Un point critique concernant le Gigabit Ethernet est la taille des paquets. En effet, cette taille, pour des raisons de compatibilité ascendante n'a pas évolué depuis le début de la norme. Ainsi, en une seconde, il arrive 100 fois plus de paquets sur un réseau Gigabit par rapport à un réseau 10 Mb/s. Cela engendre une surcharge pour la CPU interrompue pour traiter les paquets venant du réseau. Cela a poussé les concepteurs de cartes réseaux à proposer une évolution du protocole en multipliant par 6 la taille des paquets. Ce protocole, appelé *Jumbo Frames* ne fait malheureusement pas partie de la norme officielle.

Au regard de notre base de données de référence, il est évident que les réseaux de type Gigabit n'ont pas une bande passante suffisante. Il existe cependant une norme évoluée en cours de préparation : le réseau 10 Gigabit. Cependant, cette norme ne pévoit toujours pas l'utilisation de Jumbo Frames. Ainsi, les paquets auront toujours la même taille que celle fixée lors du protocole Ethernet dans les années 80.

Cependant, l'intérêt d'utiliser des réseaux de type Gigabit Ethernet dès maintenant est leur compatibilité avec les anciennes normes. Par ailleurs, ils permettent de réaliser des tests à moindre coût, car les composants de base sont beaucoup plus répandus que ceux de tout autre type de réseau haut débit.

Les réseaux évolués

A ce niveau de puissance, les différents types de réseau n'obéissent pas à des normes auxquelles plusieurs constructeurs souscrivent. Il s'agit plutôt de sociétés distinctes proposant leur propre réseau.

Par exemple, la société Myricom a développé les réseaux Myrinet[4]. Ce type de réseau a une bande passante de 2 Gb/s. Par contre, ce réseau n'est pas prévu pour effectuer du broadcast.

Il existe d'autres réseaux comme les réseaux HIPPI[5] (High-Performance Parallel Interface), avec une bande passante allant de 800 Mb/s à 1600Mb/s. Ce dernier réseau a

4. http://www.myricom.com
5. http://www.hippi.org

donné naissance à un réseau plus évolué : GSN [6]. Ce dernier est en fait un multiplexage du réseau HIPPI en parallélisant 8 réseaux HIPPI 800. Ainsi, il peut atteindre 6,4 Gb/s.

Un autre système a vu le jour dernièrement : Infiniband [7]. Ce type de réseau vise à mettre en place une architecture novatrice au sein même des ordinateurs. Il ne se connecte pas comme une extension sur un bus d'entrés/sortie de l'ordinateur mais joue le rôle de bus d'inter-connexion entre les prériphériques et les CPUs. En fait, sur ce réseau sont branchés un certain nombre de ressources (des disques, des cartes Ethernet, cartes fibre optique, des cartes vidéos, des cartes graphique, etc) ainsi que des nœuds processeurs. La bande passante maximum prévue par la norme est de 30 Gb/s.

Ces différents réseaux présentent l'inconvénient majeur d'être peu répandus et sont donc très onéreux. De plus, il faut toujours ajouter des « switchs » qui ne sont pas standards.

Conclusion sur les réseaux haut débit

Nous avons vu qu'il existait plusieurs types de réseaux. Ceux orientés très haut débit ne sont pas forcément les plus appropriés dans les phases d'essais. C'est pourquoi, nous nous sommes contentés, pour l'heure, d'étudier le canal flux de données de notre architecture sur une solution à base d'un réseau Gigabit Ethernet. Dans les faits, afin de pallier la faible bande passante des réseaux Gigabit, cette solution devra être combinée avec des systèmes de gestion de scène afin d'optimiser les transmissions. Ainsi, la solution finale sera un juste équilibre entre la bande passante disponible sur le réseau et la gestion de scène afin d'optimiser les transmissions.

Il convient également de noter que cette étude a été faite dans l'optique d'une intégration du canal haut débit dans l'architecture. En effet, nous avons vu que ce type de canal était important pour certains types d'application. Ainsi, même si conceptuellement cette composante est prévue dans l'architecture, nous n'en sommes qu'à l'étude de faisabilité et n'avons donc pas encore commencé à modéliser son intégration.

6. http://www.hnf.org
7. http://www.infinibandta.org

7 Système de navigation

Le système de navigation que nous avons mis au point requiert encore quelques améliorations. Par exemple, il faut pouvoir contraindre le mouvement de l'utilisateur. De plus, nous devons évaluer ce mode de navigation pour connaître sa pertinence.

7.1 Contraintes de la scène

Glissement sur le mur

Pour une interaction immersive « naturelle », aucune partie du corps de l'utilisateur ne doit traverser les objets virtuels. Ainsi, un système de détection de collisions doit influencer et être influencé par notre système de navigation.

Dans le cas de scènes architecturales, celles-ci sont généralement composées d'entités verticales et horizontales (murs, sols). Par contre, la visualisation immersive d'applications scientifiques comme la mécanique des fluides, s'appuie sur des surfaces dynamiques beaucoup plus complexes. Ainsi, le détecteur de collisions requiert alors des algorithmes de gestion de scène pour atteindre le temps réel nécessaire à notre système de navigation.

D'un autre côté, en présence d'une navigation contrainte par des détections de collisions, l'utilisateur peut être bloqué dans des « impasses ». C'est-à-dire qu'il a été capable d'entrer dans un goulot d'étranglement, mais qu'il est incapable de retrouver la sortie.

Pour résoudre ce problème, nous travaillons actuellement sur un système de contrainte du déplacement du véhicule qui permet un glissement sur les surfaces des objets d'une scène.

Contrainte du sol

Nous avons suggérer ci-dessus que l'utilisateur ne devait pas traverser ni les murs, ni le plancher. Cependant, il faut plus généralement pouvoir contraindre le mouvement dans un plan parallèle au sol. Cela est fondamental, par exemple dans la visite de scènes architecturales.

Contrairement à tous les autres systèmes de navigation, le nôtre est par essence à six degrés de liberté. Ainsi, nous sommes obligés de le contraindre pour permettre une navigation dans le plan.

La solution la plus simple consiste à systématiquement annuler la composante perpendiculaire au plan de contrainte. Le problème est que cette solution ne permet pas à un utilisateur d'aller dans les étages d'une architecture.

La solution choisie est de filtrer toute composante de translation « verticale » dans le delta à appliquer sur la position de la scène par rapport à celle du véhicule. C'est-à-dire qu'en fonction de la position courante de l'utilisateur, on peut déterminer un plan courant de déplacement. Il suffit alors de contraindre la matrice dm_i^{-1} dans ce plan (i.e. le plan du sol, de l'escalier ou de l'étage).

7.2 Protocole d'évaluation du système de navigation

Nous avons déjà fait tester le système de navigation par un grand nombre d'utilisateurs. Ces premiers retours nous ont encouragés à poursuivre dans cette piste. Cependant, certains utilisateurs ont essayé d'utiliser le système en sens inverse. Par exemple, au lieu de partir à gauche lorsque l'utilisateur tourne la tête à droite, ils auraient préféré que la scène parte sur la droite. Ce point est à éclaircir, mais il est facilement ajustable puisqu'il suffit d'utiliser un coefficient d'atténuation négatif au niveau de la rotation. L'utilisation de ce système nécessite un apprentissage. Cependant, alors que certaines personnes n'arrivent pas à contrôler suffisamment le système au bout de dix minutes (le temps est surtout limité par le temps de la démonstration face à un public), d'autres personnes comprennent le système en 2 minutes. Ayant eu deux types de publics (scientifique lors de nos rencontres et grand public lors des Fêtes de la Science 2002), nous n'avons pas remarqué des prédispositions particulières pour certains utilisateurs. Avec le recul, nous nous rendons compte que notre discours sur le sujet s'affine et nous permet de faire comprendre plus rapidement le principe du système de navigation.

Toutes ces considérations ne sont basées que sur des observations empiriques. Aussi, nous travaillons actuellement à une évaluation du système de navigation main libre décrit dans le troisième chapitre de cette thèse. Cependant, il faut pouvoir le comparer à d'autres systèmes déjà existant. Ainsi, sur la base du *wand*, nous avons implémenté un second système de navigation dans l'espace.

L'application utilisée sera probablement basée sur une scène architecturale. En effet, les scènes scientifiques sont trop complexes et donc peu adaptées à un large public. Deux modes de navigation seront proposés lors de cette évaluation :

- à l'aide du *Wand*, la navigation se faisant dans un plan horizontal via son mini joystick. Le plan peut monter et descendre en utilisant les boutons.
- à l'aide du système « HCnav » couplé aux mouvements de tête. Ce mode permet de naviguer en six degrés de liberté.

On ajoute un « marqueur » (boule jaune avec flèche symbolisant une direction de visée), que l'on pourra montrer à l'expérimentateur suivant divers points de vue (subjectif, général..). L'objectif de l'utilisateur est de rejoindre le référentiel complet (en position et en orientation) du marqueur durant le temps de l'expérience. Après l'expérience, on pourra visualiser la trajectoire suivie dans le monde virtuel. (Cette trajectoire est enregistrée sur disque afin de permettre un dépouillement des résultats).

Il faut noter plusieurs remarques faites, notamment par les ergonomes que nous avons déjà rencontrés à ce sujet :

- Les deux modes de navigation sont très différents. On doit donc s'attendre à des comportements différents (ce qui est intéressant en soi mais brouille la comparaison).
- Il faut également proposer un système avec le même comportement que le système de navigation main libre mais couplé au mouvement du Wand. De plus, le mouvement induit doit probablement être rapporté à la tête de l'utilisateur (et non à la main), de sorte que le maniement soit plus aisé.

- Il faudra enregistrer les mouvements de tête dans l'espace réel pour juger du confort et du comportement global de l'expérimentateur.
- Une information pertinente sera l'orientation de la tête de l'utilisateur. En effet, cela permettra d'avoir une idée sur le regard mouvement du regard de l'utilisateur. En post-traitement, nous pourrons également faire de la triangulation sur les différentes positions de l'utilisateur afin de connaître ses points d'intérêts visuel.

Parmi les autres éléments sur lesquels il faudra encore réfléchir, il y a le positionnement de la cible dans la scène en fonction de la base de données choisie. Il faudra aussi permettre de changer la vitesse de navigation (par l'intermédiaire de la modification des coefficients d'atténuation) en cours de navigation. Dans le même principe, il faudra pouvoir changer la vitesse de navigation au *wand*.

Conclusion

Ce mémoire de thèse retrace l'ensemble des travaux de recherche que j'ai accompli durant ces trois années. Nous avons vu que, même si je ne me suis pas concentré sur les aspects cognitifs sous-jacents à l'interaction en Réalité Virtuelle, il s'agit d'un point important servant de fil conducteur à l'ensemble de ma démarche. En effet, mon but était de fournir une plate-forme logicielle permettant de valider et de comparer l'ergonomie de tout type de paradigme interactif que le domaine de la Communication Humain-Machine est susceptible d'apporter à la RV.

Les concepts exposés dans le **premier chapitre** mettent en lumière un certain nombre de problèmes et de contraintes qu'il faut lever pour atteindre cet objectif. Par exemple, les mondes virtuels au sens de la section 2.2 page 15 ne sont pas forcément isomorphes à l'univers réel. Ainsi, la première contrainte que nous nous sommes fixée est que les interactions mises en place sont toutes assez génériques pour être utilisées sur l'espace de représentation de n'importe quel monde virtuel. De plus, nous avons vu que le canal sensoriel le plus répandu pour le retour de l'information de l'ordinateur vers l'utilisateur est la vue. Au demeurant, les autres canaux ont tendance à être laissés de côté, même si les aspects audio 3d ou haptiques commencent à apparaître dans les environnements immersifs. De plus, les systèmes actuels sont encore peu enclins à travailler sur des données scientifiques qui imposent des contraintes en terme de charge de calcul et de transmission de données.

Dans le **deuxième chapitre**, j'ai présenté une architecture distribuée pour la RV, dont l'originalité est d'être articulée sur deux canaux de communication. L'application gèrant le rendu graphique est considérée, dans notre cas, comme un module interactif. Cependant, afin de soulager la partie graphique de certaines charges de calcul liées à l'interaction, plusieurs périphériques et modules sont exportés grâce à l'architecture sur d'autres machines. Cette distribution peut être imposée par des besoins spécifiques des connexions matérielles ou de systèmes d'exploitation. Mais l'architecture de la plate-forme EVI3d permet de plus d'intégrer un ensemble de paradigmes interactifs « naturels », tant par l'utilisation de périphériques plus adaptés que de modules logiciels évolués (traitements de signaux, processus de reconnaissance...). Un des points clefs de cette interaction « naturelle » est la possibilité de fusionner des événements issus de modalités diverses (cf. fusion multimodale). Afin de valider cette architecture pour la gestion de telles interactions, j'ai plus particulièrement modélisé et mis en œuvre des interactions immersives basées sur

la combinaison de commandes vocales avec les gestes co-verbaux associés. En outre, l'architecture de la plate-forme EVI3d est dotée d'un système de synchronisation temporelle indispensable à ce type de fusion puisque le traitement des interactions est réparti sur plusieurs calculateurs. De plus, nous avons prévu, dès la conception de notre architecture, l'utilisation d'un canal haut débit pour les flux de données. Cependant, il n'est encore qu'à l'état de cahier des charges.

Les applications gérées par les systèmes actuels de RV manipulent généralement des objets statiques ou qui ont des comportements dynamiques prédéfinis. En d'autres termes, les concepts de base de ces systèmes n'autorisent pas de charges de calcul temps réel propres à l'application. Dans les faits, ces systèmes accaparent les ressources de calcul au détriment des traitements utiles à l'application. De plus, ils n'autorisent pas la synchronisation des rendus en présence de calculs en parallèle. Il n'est pas non plus prévu de synchronisation pour un contrôle fin de la latence. D'un autre côté, les applications scientifiques ont souvent à gérer un domaine de représentation non isomorphe à l'univers réel. Lorsqu'elles sont isomorphes, la granularité est variable et les traitements annexes sont importants. C'est vis-à-vis de ces deux contraintes que nous n'avons pas trouvé de système susceptible de gérer les applications scientifiques de manière optimale. Dans ce contexte, le **troisième chapitre** de ma thèse présentait les principales originalités du noyau géométrique que j'ai dû concevoir pour répondre à ces contraintes. Ce noyau s'appuie en particulier sur une métaphore de « véhicule » applicable à tout type de dispositif immersif. Sorte de pont entre le réel et le virtuel, cette métaphore permet à l'utilisateur d'agir sur le monde virtuel via les périphériques situés dans l'univers réel. En fait, cette métaphore de « véhicule » provient des travaux auxquels j'ai contribué sur le contrôle des navigations virtuelles basé sur le signal d'un simple capteur de mouvement 6 DDL. C'est le système HCnav qui permet de contrôler les navigations de l'utilisateur dans l'ensemble du monde virtuel en fonction de ses mouvements dans l'univers réel.

Cette architecture distribuée ainsi que le noyau géométrique évoqué constituent l'essentiel de la plate-forme EVI3d. La généricité de mes travaux permet à cette plate-forme d'être d'ores et déjà utilisée par 5 applications ou démonstrateurs de Réalité Virtuelle du LIMSI-CNRS (application de Mécanique des Fluides, de Génomique et de CAO immersive ; mais aussi démonstrateur du système HCnav et de l'interaction multimodale en RV).

Le **quatrième chapitre** est entièrement basé sur la validation et l'évaluation de mes travaux. Parmi les perspectives futures, nous allons généraliser le noyau géométrique à tous les types de périphériques. Cela pourra être mis en œuvre facilement étant donné qu'il ne s'agit que d'une généralisation des casques immersifs déjà implémentés. L'amélioration du mode de fonctionnement de l'application de contrôle est, quant à lui plus complexe, car il requiert un réaménagement interne de l'EVserveur. Cependant, il s'agit d'un problème fortement pénalisant sur lequel il faudra mettre l'accent. Un point qui sera facilement mis en œuvre sera la synchronisation du noyau géométrique sur un contrôleur extérieur pour les latences. Cela va nécéssiter la mise en place d'un protocole d'évaluation de la

latence totale. Nous allons aussi travailler sur le flux de données. Ce canal permet aux applications, en particulier pour celles à caractère scientifique, d'échanger des données avec un super-calculateur. A moyen terme, cette communication haut débit ouvre également des perspectives pour distribuer le rendu en environnement immersif sur un ensemble de petits calculateurs ou grappe. Enfin, nous allons mettre en place une évaluation ergonomique de notre système de navigation qui intègrera des contraintes imposées par la scène.

Au-delà de ces perspectives d'implémentation, mes travaux de thèse débouchent enfin sur deux types de perspectives de recherche.

Tout d'abord, l'intégration de traitements avancés sur divers signaux, ainsi que la possibilité de combiner plusieurs modalités, nous permettent maintenant d'aborder les problématiques cognitives associées à l'utilisation de paradigmes spécifiques pour interagir sur des mondes virtuels. En particulier, nous allons pouvoir évaluer l'apport des interactions multimodales dans des tâches immersives, tandis que des évaluations ergonomiques comparatives pourront être menées entre le système HCnav et différents systèmes de navigation virtuelle. A plus long terme, différents travaux sont aussi possibles dans les problématiques des substitutions sensorimotrices, ou celles relatives aux interactions coopératives.

Par ailleurs, mes travaux de recherche sur les interactions immersives pour les applications scientifiques vont pouvoir s'amplifier. Au-delà de la distribution d'une application scientifique de RV entre le calculateur graphique et un super calculateur, le rapport entre l'augmentation des débits et la croissance de la complexité des scènes devrait m'amener à étudier le partage des fonctionnalités de gestion de scènes entre ces deux types de calculateurs. Cette problématique d'optimisation de la communication haut débit sera aussi au cœur de mes préoccupations pour l'extension de l'architecture de la plate-forme EVI3d à la gestion de clusters de PC pour des applications immersives aux feedbacks tant visuels, qu'audios ou haptiques.

Liste des publications

Conférences et workshops internationaux avec actes et comité de lecture

[1] **D. Touraine**, P. Bourdot, Y. Bellik, and L. Bolot. A framework to manage multimodal fusion of events for advanced interactions within Virtual Environments. In *Proc. of Eurographics Workshop for Virtual Environments*, Barcelona (Spain), May 2002.

[2] P. Bourdot and **D. Touraine**. Polyvalent display framework to control virtual navigations by 6DOF tracking. In *Proc. of IEEE International Virtual Reality Conference 2002*, Orlando (Florida), March 2002.

[3] **D. Touraine** and P. Bourdot. VEserver : a manager for input and haptic multisensorial devices. In *Proc. of 10th IEEE international workshop on Robot and Human Communication*, pages 2-7, Bordeaux - Paris (France), september 2001.

[4] P. Bourdot and **D. Touraine**. A Hierarchical Server to Manage Multimodal and Collaborative Interactions within Virtual Environments. In *Proc. of the 19th IASTED International Conference on Applied Informatics*, Innsbruck (Austria), February 2001.

Mémoires de diplômes

[5] **D. Touraine**. *Visualisation stéréoscopique et navigation interactive appliquées à des écoulements.* Rapport de DEA « Système Electronique de Traitement de l'Information », Faculté des Sciences d'Orsay, Univesité Paris-Sud, 1999.

Autres formes de publication

[6] P. Bourdot, C. Tenaud, **D. Touraine**, L. Doris, G. Chastagner. Visionneuse stéréoscopique d'une scène en relief calculée par l'application "Mécanique des Fluides" de la "plate-forme EVI 3d (Environnements Virtuels et Interactions 3d)" ; LIMSI-CNRS, Décembre 1999.

Bibliographie

[ACJ91] D. Annicchiarico, R. Chelser, and A. Jamisson. Xtrap: The xtrap architecture. Technical report, Digital Equipment Corporation, July 1991.

[AFM92] S. Armstrong, A. Freier, and K. Marzullo. *Mulicast transport protocol*. IETF Large-Scale Multicast Applications, February 1992. Request For Comment (RFC) : 1301.

[Ali] Site web de l'alioscopie. http://www.alioscopy.com.

[Arn99] L. Arnal. Etude et mise en place d'un système de synchronisation d'horloge dans un environnement distribué. Master's thesis, Univeristy of Paris XI, May 1999. Document in french.

[Ath96] EFA - EDF - Ecole d'Architecture de Nancy-Maison de l'Archéologie de Bordeaux. *Marmaria, Le Sanctuaire d'Athéna à Delphes*, 1996. ISBN 2-86958-085-1.

[BBHM95] J. Bacon, J. Bates, R. Hayton, and K. Moody. Using events to build distributed applications. In *the second international workshop on services in distributed networked environments*, pages 148 – 155. IEEE computer society press, June 1995.

[BDA99] P. Bourdot, M. Dromigny, and L. Arnal. Virtual navigation fully controlled by head tracking. In *Proc. of International Scientific Workshop on Virtual Reality and Prototyping*, Laval (France), June 1999.

[Bel95] Y. Bellik. *Interfaces multimodales : concepts, mode'les et architectures*. PhD thesis, Université Paris-Sud, 1995. Document in french.

[BKG95] P. Bourdot, M. Krus, and R. Gherbi. Management of non-standard devices for multimodal user interfaces under Unix / X11. In *International Conference on Cooperative Multimodal Communication*, Eindhoven (The Netherland), May 1995.

[BKG98] P. Bourdot, M. Krus, and R. Gherbi. Cooperation between reactive 3D objects and a multimodal X window kernel for CAD. In *Lecture Notes in Artificial Intelligence (1374) on Multimodal Human-Computer Communication : Systems, Techniques and Experiments*, pages pp. 188–212. Springer, January 1998.

[Bou02] P. Bourdot. Reconstruction et interaction 3d : contribution à la réalité virtuelle et augmentée, July 2002. Habilitation à diriger des recherches - Note et documens LIMSI numro 2002-11.

[Bow99] D. Bowman. *Interaction techniques for common tasks in immersive virtual environments*. PhD thesis, Georgia institut of technology, June 1999.

[BWCL01] D. Bowman, C. Wingrave, J. Campbell, and V. Ly. Using pinch gloves for both natural and abstract interaction techniques in virtual environments. In *Proc. of HCI International*, pages 629–633, New Orleans (USA), August 2001.

[CH93] C. Carlsson and O. Hagsand. DIVE - a platform for multi-user virtual environments. *Computer & graphics*, 17(6):663 – 669, 1993.

[CMO+99] P. R. Cohen, D. McGee, S. Oviatt, L. Wu, J. Clow, R. King, S. Julier, and L. Rosenblum. Multimodal interaction for 2d and 3d environments. In *IEEE Computer Graphics and Applications*, volume 19(4), pages 10–13. IEEE Press, 1999.

[CMU+97] R. Cole, J. Mariani, Uszkoreit, A. Zaenen, and V. Zue. *Survey of the state of the art in human language technology*. Cambridge University Press, 1997.

[CNDD93] C. Cruz-Neira, D.J.Sandin, and T. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *Computer Graphics (Proceedings of SIGGRAPH '93)*, pages 135–142, August 1993.

[CO95] P. R. Cohen and S. L. Oviatt. The role of voice input for human-machine communicatio. In *Proceedings of the National Academy of Sciences*, volume 92(22), pages 9921–9927. Washington, D. C. National Academy of Sciences Press, 1995.

[Con02] T. Convard. Interaction immersive avec des objets 3d réactifs. Master's thesis, Univeristy of Paris XI, August 2002. Document in french.

[COR] Corba for beginners. http://cgi.omg.org/corba/beginners.html.

[Cos99] L. Neil Cosby. SIMNET - an insider's perspective, 1999. http://www.sisostds.org/webletter/siso/iss_39/art_202.htm.

[CSF02] W. Chinthammit, E. Seibel, and T. Furness. Unique shared-aperture display with head or target tracking. In *Proc. of Virtual Reality 2002 Conference (VR 2002)*, March 2002.

[DCDK98] S. Donikian, A. Chauffaut, T. Duval, and R. Kulpa. GASP : from modular programming to distributed execution. In IEEE, editor, *Computer Animation'98*, pages 79 – 87, Philadelphie, USA, June 1998.

[Den79] M. Denis. *Les images mentales*. Collections le psychologue - presses universitaires de France, 1979.

[Div] The diverse toolkit. http://www.diverse.vt.edu/dtk/.

[DJL+00] M. Daily, J. Jerald, C. Lee, K. Martin, D. McInnes, P. Tinker, and R. Smith. Distributed design review in virtual environments. In *Proc. of Third International Conference on Collaborative Virtual Environments (CVE2000)*, September 2000.

[Dow] L. Downs. Cs184: Using quaternions to represent rotation. Electronic version: http://http.cs.berkeley.edu/ ∼laura/ cs184/ quat/ quaternion.html.

[Fau01] S. Faudeux. Relief, l'autre dimension. In *SONOVISION*, November 2001. Document in french.

[FDFH00] J. Foley, A. Van Dam, S. Feiner, and J. Hughes. *Computer graphics - principles and practice.* Addison wesley, April 2000.

[FFH+99] F. Faure, C. Faisstnauer, G. Hesina, A. Aubel, M. Escher, F. Labrosse, J.-C. Nebel, and J.-D. Gascuel. Collaborative animation over the network. In *Computer animation'99*, Geneva, 1999.

[FMP01] P. Fuchs, G. Moreau, and J.-P. Papin, editors. *Le traité de la réalité virtuelle.* Les presses de l'École des Mines, 2001.

[FPB88] Jr F. P. Brooks. Grasping reality through illusion : Interactive graphics serving science. In *Prooceeding of CHI'88 : Invited keynote address at Conf. on Computers and Human Interaction*, pages 1–11, May 1988.

[GC01] J. Grosjean and S. Coquillart. Command & control cube: a shortcut paradigm for virtual environments. In *Proc. of IPT-EGVE'2001*, Stuttgart (Germany), May 2001.

[GH00] R. Gherbi and J. Herisson. ADN_viewer: a software framework for 3d modeling and stereoscopic visualisation of the genome. In *Proc. of international conference Graphicon'2000*, pages 14–21, august-september 2000.

[GLGT99] A. Gregory, M. Lin, S. Gottschalk, and R. Taylor. H-collide: A framework for fast and accurate collision detection for haptic interaction. In *IEEE Virtual Reality*, 1999.

[Gon95] Y. Gong. Speech recognition in noisy environments: A survey. In *Speech Communication*, volume 6, pages 261–291, 1995.

[Gou01] D. Goularas. Interaction immersive en réalité virtuelle : application de cette problématique à la cao. Master's thesis, Univeristy of Paris XI, August 2001. Document in french.

[Han97] C. Hand. A survey of 3d interaction techniques. *Computer graphics forum*, 16(5):269 – 281, December 1997.

[HKW+98] J. Hahn, R. Kaufman, A. Winick, Y. Park, R. Lindeman, K.-M. Oh, R. Walsh, N. Al-Ghreimil, T. Carleton, M. Loew, and S. Sankar. Training environment for inferior vena caval filter placement. In *Proc. of Medicine Meets Virtual Reality 6*, San Diego (USA), Jannuary 1998.

[HLA98] US Depatement of defense. *High Level Architecture (HLA) Interface Specification*, April 1998. submission to the IEEE committy.

[HLS97] T. H. Harrison, D. L. Levine, and D. C. Schmidt. The design and performance of a real-time corba object event service. In *the OOPSLA'97 conference*, pages 184 – 200, Octobre 97.

[IHS+01] R. Taylor II, T. Hudson, A. Seeger, H. Weber, J. Juliano, and A. Helser. Vrpn: A device-independent, network-transparent vr peripheral system. In *Proc.*

of Virtual Reality Software & Technology 2001 (VRST'01), Banff, Alberta (Canada), November 2001.

[Inc98] Paradigm Simulation Inc. Vega programmer's guide, 1998.

[Jan94] S. Jankowski. Quake protocol information. http://www.activesw.com/people/steve/qprotocol.html, 1994.

[JCM+97] M. Johnston, P. R. Cohen, D. McGee, S. L. Oviatt, J. A. Pittman, and I. Smith. Unification-based multimodal integration. In *Proc. of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid (Spain), 1997.

[JCNB00] C. Just, C. Cruz-Neira, and A. Baker. Performance measurement capabilities of vr juggler : real-time monitoring of immersive applications. In *The 4th international immersive projection technology workshop*, 2000.

[Jot96] J.M. Jot. Synthesizing three-dimensional sound scenes in audio or multimedia production and interactive human-computer interfaces. In *5th International Conference: Interface to Real & Virtual Worlds*, May 1996.

[Jr.88] F. P. Brooks Jr. Grasping reality through illusion - interactive graphics serving science. In *Proc. CHI'88*, pages 1 – 11, New York, May 1988.

[KAL+00] J. Kelso, L. Arsenault, C. Logie, F. das Neves, and R. Kriz. Diverse graphics interface for performer programmer's guide. Technical report, University Visualization and Animation Group of the Advanced Communication and Information Technology Center, Virginia Tech, July 2000. http://www.diverse.vt.edu/dtk/.

[KCC95] A. Kheddar, R. Chellali, and P. Coiffet. Implementation of head-behavior based control for navigation within virtual reality applications. In *Proc. of IEEE International Conference on Systems, Man and Cybernetics: Intelligent systems for the 21st century*, volume 5, pages 4644–4649, October 1995.

[Lak96] J. Lakos, editor. *Large-Scale C++ Software Design.* Addison-Wesley, July 1996.

[LaV00] J. LaViola. A discussion of cybersickness in virtual environments. *ACM SIGCHI Bulletin*, 32:47 – 56, 2000. ISBN 0736-6906.

[LC87] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics*, volume 21, July 1987.

[LCS97] B. Laroussi, Y. Cai, and M. Sato. New haptic device for human-scale virtual environment: Scaleable spidar. In *Proc. of ICMR 97*, Tamperre (Finland), Jannuary 1997.

[Lec01] A. Lecuyer. *Contribution à l'étude des retours haptique et pseudo-haptique et de leur impact sur les simulations d'operations de montage/démontage en aéronautique.* PhD thesis, Université Paris-Sud, December 2001. Document in french.

[Lew94] D. Lewine. *POSIX Programmer's guide.* O'Reilly and associates incorporated, March 1994.

[LJD97] J. Leigh, A. Johnson, and T. DeFanti. Cavern: A distributed architecture for supporting scalable persistence and interoperability in collaborative virtual environments. *Virtual Reality Research, Development and Applications*, 2.2:217–237, December 1997.

[LKC+01] A. Lecuyer, A. Kheddar, S. Coquillart, L. Graux, and P. Coiffet. A haptic prototype for the simulations of aeronautics mounting/unmounting operations. In *Proc. of IEEE ROMAN'2001*, Bordeaux-Paris (France), September 2001.

[LOF99] J. A. Larson, S. L. Oviatt, and D. Ferro. Designing the user interface for pen and speech applications. In *Conference on Human Factors in Computing Systems (CHI'99)*, 1999.

[LP96] J.C. Lin and S. Paul. Rmtp: A reliable multicast transport protocol. In *Proc. of IEEE Infocom*, March 1996.

[MBG02a] O. Magneau, P. Bourdot, and R. Gherbi. 3D tracking based on infrared cameras. In *International Conference on Computer Vision and Graphics*, Zakopane, Poland, September 2002.

[MBG02b] O. Magneau, P. Bourdot, and R. Gherbi. Positionning and identification of markers for 3D tracking. In *VIRTUAL CONCEPT*, Biarritz, France, October 2002.

[MCW00] D. R. McGee, P. R. Cohen, and L. Wu. Something from nothing: Augmenting a paper-based work practice with multimodal interaction. In *Proceedings of the Designing Augmented Reality Environments Conference*, pages 71–80. ACM Press, 2000.

[Mil92] D.L Mills. *Network time protocol (version 3): specification, implementation and analysis*. University of Delaware, March 1992. Request For Comment (RFC) : 1305.

[Mol92] H. Moller. Fundamentals of binaural technology. In *Applied Acoustics*, volume 36, pages 171–218, 1992.

[NC93] L. Nigay and J. Coutaz. A design space for multimodal systems : concurent processing and data fusion. In *Proc. of INTERCHI'93*, pages 172–178. ACM Press, april 1993.

[OC00] S. L. Oviatt and P. R. Cohen. Multimodal systems that process what comes naturally. In *Communications of the ACM*, volume 43(3), pages 45–53, 2000.

[OCW+00] S. Oviatt, P. Cohen, LZ. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro. Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. In *Human-computer interaction*, volume 15 (4), pages 263–322, 2000.

[ODK97] S. L. Oviatt, A. DeAngeli, and K. Kuhn. Integration and synchronization of input modes during multimodal human-computer interaction. In *Proceedings of Conference on Human Factors in Computing Systems (CHI'97)*, pages 415–422. New York: ACM Press, 1997.

[OFK+01] Alice Oh, Harold Fox, Max Van Kleek, Aaron Adler, Krzysztof Gajos, Louis-Philippe Morency, and Trevor Darrell. Evaluating look-to-talk: A gaze-aware interface in a collaborative environment. In *Proc. of CHI 2002*, Minneapolis, November 2001.

[OIv94] Silicon Graphics, Inc. *Optimizing Open Inventor(tm) Applications*, 1994.

[Opt98] Silicon Graphics, Inc. *OpenGL Optimizer Programmer's Guide: An Open API for Large-Model Visualization*, 1998.

[PBD91] C. Pinon-Bouwens and LB. Mc Donald. Distributed interactive simulation: interoperability of dissimilar training devices for team training. In Twenty-Third Annual Summer Computer Simulation Conference, editor, *the 1991 Summer Computer Simulation Conference*, volume xxiii+1194, pages 934 – 939, CSC, San Diego, CA, USA, 1991.

[PCK+00] K. Park, Y. Cho, N. Krishnaprasad, C. Scharver, M. Lewis, J. Leigh, and A. Johnson. Cavernsoft g2: A toolkit for high performance tele-immersive collaboration. In *Proc. of the ACM Symposium on Virtual Reality Software and Technology 2000*, October 2000.

[Per94] Silicon Graphics, Inc. *IRIS Performer Programmer's Guide*, 1994.

[Per98] Silicon Graphics, Inc. *IRIS Performer programmer's guide*, 1998.

[PHSJ97] I. Pyarali, T. Harrison, D. C. Schmidt, and T. D. Jordan. Proactor : An architecturale pattern for demultiplexing and dispatching handlers for asynchronous events. In *4th Pattern Languages of Programming conference*, Allerton Park, Illinois, September 1997.

[PMB99] M. Pullen, M. Myjak, and C. Bouwens. *Limitation of internet protocol suite for distributed simulation in the large multicast environment.* IETF Large-Scale Multicast Applications, February 1999. Request For Comment (RFC) : 2502.

[Rag94] D. Raggett. Extending www to support platform independent virtual reality. In Internet Soc, editor, *Proceedings of INET'94/JENC5. INET'94, the Annual Conference of the Internet Society held in conjuction with 5th Joint European Networking Conference (JENC5)*, Reston, VA, USA, 1994. 2 vol. 464 pp. p.242/1, 242/3-6 vol.1.

[Req80] A. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computer survey*, 12:437–464, December 1980.

[RGS95] R. Rajkumar, M. Gagliardi, and L. Sha. The real-time publisher/subscriber inter-process communication model for distributed real-time systems : design and implementation. In *1st IEEE Real-time technology and applications symposium*, May 1995.

[RH94] W. Robinett and R. Holloway. The visual display transformation for virtual reality. Technical Report TR94-031, presence, September 1994.

[Riv00] G. Riva. *Virtual Reality in neuro-psycho-physiology - Cognitive, clinical and methodological issues in assessment and rehabilitation.* IOS PRESS, 2000.

[RS01] G. Reitmayr and D. Schmalstieg. An open software architecture for virtual reality interaction. In *Proc. of Virtual Reality Software & Technology 2001 (VRST'01)*, Banff, Alberta (Canada), November 2001.

[Rub91] D. Rubine. *The automatic recognition of gestures*. PhD thesis, Carnegie Mellon University, 1991.

[SC02] W. Sherman and A. Craig. *Understanding Virtual Reality: Interface, Application, and Design*. Morgan Kaufmann Publishers, September 2002. ISBN 1-55860-353-0.

[Sch] D. C. Schmidt. The adaptive communication environment (ace). http://www.cs.wustl.edu/ schmidt/ACE.html.

[Sch94] D. C. Schmidt. Reactor: An object behavioral pattern for concurrent event demultiplexing and event handler dispatching. In *1st Pattern Languages of Programs Conference*, August 1994.

[Sch99] D. C Schmidt. An overview of omg corba event services, 1999. http://www.cs.wustl.edu/ schmidt/.

[SH74] I. Sutherland and G. Hodgman. Reentrant polygon clopping. In *CACM*, volume 17(1), pages 32–42, January 1974.

[Sho85] K. Shoemake. Animating rotation with quaternion curves. In *Proc. of SIGGRAPH'85*, volume 19, pages 245–254, 1985.

[Sho92] K. Shoemake. Arcball: A user interface for specifying three-dimensional orientation using a mouse. In *Proc. of Graphics Interface '92*, pages 151–156, 1992.

[SJM+97] A. Shaikh, S. Juth, A. Medl, I. Marsic, C. Kulikowski, and J. Flanagan. An architecture for multimodal information fusion. In *Proceedings of the Workshop on Perceptual User Interfaces (PUI'97)*, pages 91–93, 1997.

[SMD+01] D. Sandin, T. Margolis, G. Dawe, J. Leigh, and T. DeFanti. The varrier auto-stereographic display. *SPIE*, 4297:104–211, June 2001.

[SML96] W. Schroeder, K. Martin, and W. Lorensen. The design and implementation of an object-oriented toolkit for 3d graphics and visualization. In *Proc. of the conference on Visualization '96*, 1996.

[SSU93] M. Slater, A. Steed, and M. Usoh. The virtual treadmill : A naturalistic method for navigation in immersive virtual environments. In *First Eurographics Workshop on Virtual Environments, Polytechnical University of Catalonia, ed. Martin Goebel*, pages 71–83, September 1993.

[Sta02] Kay M. Stanney, editor. *Handbook of Virtual Environment - Design, Implementation, and Applications*. Human factors and ergonomics, 2002.

[SW94] D. N. Snowdon and A. J. West. The AVIARY VR-system. a prototype implementation. *6th ERCIM workshop*, June 1994.

[SZ99] S. Singhal and M. Zyda. *Networked Virtual Environments*. ACM Press, July 1999. ISBN=0-201-32557-8.

[Tra99] H. Tramberend. Avocado: A distributed virtual reality framework. In *Proc. of the IEEE Virtual Reality*, pages 14–21, March 1999.

[UAW+99] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks Jr. Walking >> Walking-in-place >> Flying in virtual environments. In *Proc. SIGGRAPH 99*, pages 359 – 364, August 1999.

[Wen98] E. Wenzel. Three-dimensional virtual acoustic displays. presence: teleoperators and virtual environments. In *SIGCHI Bulletin*, pages 80–107, 1998.

[WFM+96] A. Webster, S. Feiner, B. MacIntyre, B. Massie, and T. Krueger. Augmented reality in architectural construction, inspection and renovation. In *Proc. of Third ASCE Congress for Computing in Civil Engineering*, Anaheim (CA), June 1996.

[WNDS98] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL(R) Reference Manual*. Addison Wesley, August 1998.

[WNDS99] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 1.2 (3rd Edition)*. Addison Wesley Longman, August 1999.

[WO90a] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. In *Proc. of Symposium on Interactive 3D Graphics*, pages 175– 183, 1990.

[WO90b] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. In ACM SIGGRAPH, editor, *Proc. of 1990 Symposium on Interactive 3D Graphics*, pages 175– 183, 1990.

[WWF98] E. Wenzel, F. Wightman, and S. Foster. Development of a 3d auditory display system. In *SIGCHI Bulletin*, volume 20(2), pages 52–57, 1998.

[WWK91] E. Wenzel, F. Wightman, and D. Kistler. Localization with non-individualized virtual display cues. In Addison-Wesley ACM Press, editor, *CHI'91*, pages 351–359, 1991.

[ZF01] G. Zeck and P. Fromherz. Noninvasive neuroelectronic interfacing with synaptically connected snail neurons immobilized on a semiconductor chip. In *Proc. of the National Academy of Sciences 98*, 2001.

[Zha93] S. Zhai. Investigation of feel for 6dof inputs : Isometric and elastic rate control for manipulation in 3d environments. In *Proc. of the Human Factors and Ergonomics Society*, 1993. 37th annual meeting.

[Zha95] S. Zhai. *Human Performance in Six Degree of Freedom Input Control*. PhD thesis, University of Toronto, 1995. Electronic version: http://vered.rose.toronto.edu/people/shumin_dir/papers/PhD_ Thesis/top_page.html.

[ZHR99] W. Zachary, L. Hodges, and W. Ribarsky. The analytic distortion induced by false-eye separation in head-tracked stereoscopic displays, 1999.

[ZPOM93] M. J. Zyda, D. R. Pratt, W. D. Osborne, and J. G. Monahan. NPSNET: Real-time collision detection and response. *The Journal of Visualization and Computer Animation*, 4(1):13–24, January–March 1993.