Thèse

THÈSE INSA Rennes
sous le sceau de l'Université Européenne de Bretagne
pour obtenir le titre de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Informatique

présentée par
Cédric FLEURY
ÉCOLE DOCTORALE : MATISSE
LABORATOIRE : IRISA – UMR 6074

Modèles de conception pour la collaboration distante en environnements virtuels distribués : de l'architecture aux métaphores

Thèse soutenue le 15 juin 2012
devant le jury composé de :

Jean-Marc JÉZÉQUEL
Professeur des universités, Université de Rennes 1 / *Président*
Sabine COQUILLART
Directrice de recherche, Inria Grenoble - Rhône-Alpes / *Rapporteur*
Jean-Pierre JESSEL
Professeur des universités, Université Toulouse III - Paul Sabatier / *Rapporteur*
Pascal GUITTON
Professeur des universités, Université de Bordeaux 1 / *Examinateur*
Jacques TISSEAU
Professeur des universités, ENI de Brest / *Examinateur*
Thierry DUVAL
Maître de conférences, Université de Rennes 1 / *Co-encadrant de thèse*
Valérie GOURANTON
Maître de conférences, INSA de Rennes / *Co-encadrante de thèse*
Bruno ARNALDI
Professeur des universités, INSA de Rennes / *Directeur de thèse*

Remerciements

Plus qu'une simple poursuite d'études, cette thèse aura été pour moi une expérience de vie forte aussi bien sur le plan professionnel que sur le plan humain. Les rencontres et les collaborations de ces trois années et demi auront toutes été plus enrichissantes les unes que les autres tant au niveau scientifique qu'au niveau relationnel. Je tiens à remercier toutes les personnes qui ont contribué directement ou indirectement à cette thèse.

En premier lieu, je souhaite remercier Jean-Marc Jézéquel de m'avoir fait l'honneur de présider mon jury de thèse. Je remercie également Sabine Coquillart et Jean-Pierre Jessel d'avoir accepté de relire mon manuscrit de thèse et de m'avoir transmis des commentaires constructifs. J'adresse aussi mes remerciements à Pascal Guitton et Jacques Tisseau pour avoir participé à mon jury de thèse et pour avoir porté un regard critique sur mes travaux de doctorat.

Ensuite, je tiens à remercier Bruno Arnaldi pour avoir accepté de diriger cette thèse et pour avoir stimulé ma curiosité scientifique. Je souhaite bien évidemment remercier Thierry Duval et Valérie Gouranton de m'avoir encadré durant cette thèse. Merci pour votre disponibilité et votre gentillesse au quotidien. Merci également à vous deux de m'avoir accompagné dans mes premières expériences d'enseignement. Je remercie également l'Agence Nationale pour la Recherche (ANR) qui a financé cette thèse au travers du projet Collaviz. Au passage, j'en profite pour remercier tous les partenaires du projet Collaviz (Alban Schmutz, Benoît Vautrin, Christophe Mouton, Jean-Marie Davesnes, etc.) pour m'avoir fait découvrir le monde du travail avec tant de bonne humeur. Je tiens à remercier aussi Anthony Steed pour m'avoir accueilli à l'*University College Of London* pendant trois mois et Ian Grimstead pour être venu me rendre visite à Rennes. Merci à tous les deux d'avoir pris part à mes travaux de recherche.

Par ailleurs, je veux remercier le laboratoire IRISA pour les très bonnes conditions de travail dont j'ai pu bénéficier pendant ma thèse. Je souhaite aussi remercier toutes les personnes de l'équipe VR4i et ex-Bunraku pour la bonne ambiance qui règne dans les couloirs. Merci à Alain Chauffaut et Ronan Gaugne d'avoir pris part à mes travaux et de m'avoir aidé dans la mise au point de mes expérimentations, dans les salles de réalité virtuelle (Avalon, puis Immersia). Je tiens à remercier également toutes les personnes qui ont participé de près ou de loin au développement du *framework* Collaviz, je pense bien

sûr à Laurent Aguerreche, mais aussi aux stagiaires Charles Perin, Florent Goetz et Hugo Marchadour. Merci aussi à mes collègues de bureau (Huyen, Laurent, Samuel, Stéphanie et Vincent) pour avoir partagé mon quotidien.

Je tiens à remercier aussi toutes les personnes qui m'ont aidé à concilier études et sport de haut niveau durant toutes mes années d'études. Je pense tout particulièrement à Marie-Jo Pedrono, Pascale Sébillot, et bien sûr à Thierry Duval et Valérie Gouranton. Merci pour m'avoir permis d'atteindre un bon équilibre personnel.

Dans le même ordre idée, je tiens à remercier tous mes amis qui ont également contribué à cet équilibre personnel. Je pense d'abord à mes amis de Rennes, ceux qui viennent de l'INSA, ceux qui viennent plutôt de la pointe Bretagne et tous les autres. Merci à ceux qui sont venus assister à ma soutenance. Par ailleurs, je pense aussi à mes amis et co-équipiers de voile (Manu, Louis, Simon, ceux du CIV et les autres). Merci d'avoir partagé ces belles navigations et ces chouettes moments avec moi.

Plus particulièrement, je tiens à dire un grand merci à Gwénolé. Des cours à rattraper après mes absences lors des compétitions jusqu'au *template* de ce manuscrit, merci pour ton aide si précieuse durant toutes ces années d'études. Merci aussi pour ta disponibilité, ton soutien et tes conseils dans les moments difficiles. Enfin, merci pour ton amitié et ces bons moments passés au quotidien, de la colocation à Lorient jusqu'aux repas à la cafet de l'IRISA.

Je ne remercierai jamais assez mes parents pour leur aide, leurs conseils et leur soutien constant durant cette thèse, mais aussi depuis toujours. Merci d'avoir toujours fait le maximum pour moi, et de continuer après tant d'années. Merci aussi à ma sœur Delphine pour sa gentillesse et son courage dans les moments difficiles. Même si tu n'étais pas à la soutenance, je sais que le cœur y était. Enfin, merci à Delphine (et oui, une deuxième) pour son soutien précieux dans une période qui n'était pas la plus facile. Merci aussi à toi pour tous les bons moments passés ensemble.

Table des matières

Introduction

Contexte d'étude

Après avoir longtemps été un domaine exploratoire réservé aux laboratoires de recherche puis aux grandes entreprises, la réalité virtuelle se démocratise considérablement de nos jours grâce au développement de nouvelles technologies tant au niveau matériel que logiciel. Les dispositifs matériels (cartes graphiques, écrans 3D, tablettes tactiles, dispositifs de reconnaissance de mouvements) étant devenus abordables, les applications grand public pour le loisir utilisent de plus en plus de techniques de réalité virtuelle. Profitant de cette démocratisation et de la réduction des coûts qui en découle, de nombreuses entreprises choisissent d'avoir recours à la réalité virtuelle pour faire évoluer leurs méthodes de travail dans des domaines variés comme la formation des salariés, la promotion de leurs produits, le design, la conception industrielle, etc.

Parallèlement à cela, la collaboration à distance devient une nécessité pour les entreprises et les universités qui s'équipent de plus en plus de dispositifs de vidéoconférence et autres moyens de communication, comme le prouve l'inauguration récente de trois salles de télé-présences immersives à l'Université Européenne de Bretagne (UEB). Ce type de collaboration permet de diminuer le temps, le coût et l'empreinte écologique des déplacements professionnels au travers le monde. Dans le domaine de la réalité virtuelle, la collaboration de plusieurs utilisateurs dans un environnement virtuel partagé est une perspective essentielle pour l'avenir. En effet, elle permet à des utilisateurs, qui se trouvent sur des lieux géographiquement éloignés, de se retrouver dans un même environnement virtuel. Ces utilisateurs vont ainsi pouvoir travailler ensemble dans l'environnement virtuel pour réaliser différentes tâches comme, par exemple, de la formation à distance, des revues de projet ou de la co-expertise.

Cette thèse s'inscrit dans le contexte du projet ANR [1] Collaviz [2] qui s'intéresse à la visualisation collaborative de données scientifiques. L'objectif de ce projet est de permettre à des experts, situés sur des lieux géographiquement distants, de se retrouver dans un même environnement virtuel pour examiner ensemble des résultats de simulation numérique. D'un côté, le traitement de ces données scientifiques impose d'avoir une structure souvent centralisée capable de traiter et de stocker de très gros volumes de données (cluster, centre de calcul). D'un autre côté, nous voulons que les utilisateurs puissent facilement

1. Projet supporté par l'Agence National pour la Recherche : ANR-08-COSI-003
2. `www.collaviz.org`

collaborer à distance sur les résultats obtenus en utilisant une connexion internet standard et des dispositifs matériels hétérogènes allant d'une simple station de travail à une salle immersive de réalité virtuelle en passant par des machines à faible puissance (tablette tactile, *smartphone*). Dans le projet ANR Collaviz, cette collaboration entre utilisateurs peut prendre des formes très différentes comme de la simple visualisation collaborative de données scientifiques, du placement d'annotations dans ces données, mais aussi de la co-manipulation d'objets virtuels.

Problématique et objectifs de la thèse

Faire collaborer des utilisateurs dans un même environnement virtuel partagé est une tâche qui peut s'avérer extrêmement compliquée. Dans le monde réel, la collaboration entre plusieurs personnes n'est pas toujours une tâche évidente et nécessite une compréhension mutuelle parfaite entre les protagonistes. Dans un environnement virtuel, les contraintes technologiques (latence réseau, champ de vision réduit, etc.), l'utilisation de dispositifs matériels différents pour chaque utilisateur (capacités de perception et d'interaction différentes) et le fait de se trouver dans un monde virtuel (ne reproduisant pas toujours le monde réel) peuvent rendre la compréhension entre utilisateurs plus difficile. Une mauvaise compréhension a inévitablement des conséquences négatives sur la collaboration. Nous avons identifié trois points importants pour permettre une collaboration efficace dans un environnement virtuel. Le système mettant en œuvre un environnement virtuel collaboratif doit être capable de :

1. Permettre à chaque utilisateur de percevoir le même état de l'environnement virtuel, au même moment. En effet, assurer une bonne cohérence de l'environnement virtuel entre chaque site participant à une session collaborative est la condition préalable à une bonne compréhension mutuelle entre les utilisateurs.
2. Intégrer différents outils pour faciliter les interactions et la communication entre les utilisateurs, et adapter ces outils en fonction des dispositifs matériels qu'ils utilisent (visualisation, restitution sonore, interaction, etc.) afin de leur offrir des capacités d'interaction appropriées.
3. Permettre à chaque utilisateur de comprendre quelles sont les capacités d'interaction des autres utilisateurs, c'est-à-dire ce qu'ils perçoivent de l'environnement virtuel et ce qu'ils peuvent y faire.

Les travaux réalisés dans le cadre de cette thèse se concentrent principalement sur la façon de concevoir des environnements virtuels collaboratifs afin de permettre une collaboration efficace entre les utilisateurs. En conséquence, les principaux objectifs de cette thèse sont directement liés aux trois points évoqués précédemment. Le premier objectif est de proposer des solutions pour améliorer la cohérence de l'environnement virtuel tout en offrant aux utilisateurs une réactivité satisfaisante lors des interactions. Ces solutions doivent prendre en compte les contraintes techniques (réseaux, puissance des machines, etc.), mais aussi les tâches que les utilisateurs auront à effectuer dans l'environnement virtuel. Le deuxième objectif est de définir comment intégrer des utilisateurs avec des dispositifs matériels différents dans un même environnement virtuel. D'un point de vue technique, un environnement virtuel collaboratif doit pouvoir être déployé facilement sur des dispositifs matériels variés, et les différentes techniques d'interaction doivent également être adaptées à ces dispositifs. D'un point de vue perceptuel, les utilisateurs doivent être

en mesure de comprendre quelles sont leurs capacités d'interactions en fonction des dispositifs matériels qu'ils utilisent et quelles sont celles des autres utilisateurs. Le troisième objectif est de proposer de nouvelles métaphores pour améliorer la perception, la communication et les interactions entre utilisateurs. Enfin, ces trois objectifs sont regroupés par la volonté de concevoir un environnement virtuel collaboratif qui puisse être déployé entre différents sites géographiquement éloignés et qui puisse servir pour diverses applications et expérimentations collaboratives au sein du projet ANR Collaviz.

Organisation du manuscrit

Ce manuscrit de thèse est composé en six chapitres. Les deux premiers chapitres présentent un état de l'art des travaux sur les environnements virtuels collaboratifs, tandis que les autres chapitres présentent les contributions réalisées dans le cadre de cette thèse.

Le chapitre 1 pose le cadre scientifique de nos travaux. Il présente les grandes lignes de la réalité virtuelle, la manière dont les utilisateurs sont immergés et interagissent dans un environnement virtuel, et les différentes solutions proposées dans la littérature pour permettre aux utilisateurs de collaborer dans les environnements virtuels.

Le chapitre 2 détaille les architectures et les mécanismes permettant de maintenir la cohérence d'un environnement virtuel distribué dans les systèmes existants, ainsi que les architectures logicielles utilisées pour concevoir des applications collaboratives.

Dans le chapitre 3, nous proposons un modèle pour définir où doivent être stockées et traitées les données relatives à un environnement virtuel collaboratif afin d'obtenir le meilleur compromis entre cohérence de l'environnement virtuel et réactivé lors des interactions. Notre modèle permet d'adapter dynamiquement la distribution des données en fonction des contraintes techniques et des tâches que les utilisateurs auront à effectuer dans l'environnement virtuel.

Dans le chapitre 4, nous proposons un modèle d'architecture logicielle permettant de concevoir un environnement virtuel collaboratif en séparant les données de l'environnement virtuel, de la façon dont ces données sont distribuées sur le réseau et de la façon dont elles sont restituées aux utilisateurs. Notre modèle permet de concevoir un environnement virtuel collaboratif indépendamment de la couche réseau utilisée et de la distribution des données, ainsi que des dispositifs matériels utilisés par chaque utilisateur et des systèmes qui leur sont associés (système de rendu graphique, de rendu sonore, etc.).

Dans le chapitre 5, nous proposons un modèle pour intégrer les utilisateurs dans un environnement virtuel en tenant compte de l'espace physique d'interaction qui les entoure. Ce modèle appelé *Cabine Virtuelle d'Interaction Immersive* permet de mettre en œuvre des techniques adaptées pour interagir et collaborer dans un environnement virtuel collaboratif en fonction de l'environnement réel qui entoure les utilisateurs et des dispositifs matériels à leur disposition. Notre modèle propose également un moyen de décrire les capacités de perception de l'environnement virtuel et d'interaction de chaque utilisateur.

Le chapitre 6 montre comment les différentes contributions de cette thèse ont permis de mettre en place un environnement virtuel collaboratif entre Rennes et Londres, dans le cadre du projet ANR Collaviz. Cet environnement virtuel collaboratif a permis de réaliser une série d'expérimentations pour comparer une manipulation mono-utilisateur avec une co-manipulation à deux utilisateurs lors du positionnement d'un plan de *clipping* dans des données scientifiques. Ces expérimentations ont permis de valider les contributions de cette thèse lors d'une collaboration réelle entre deux sites distants.

Chapitre 1

Réalité virtuelle et collaboration

Définir la réalité virtuelle est une tâche complexe tant les domaines scientifiques connexes, les techniques utilisées, ainsi que les domaines d'applications sont nombreux et variés. Nous avons choisi d'utiliser la définition générale proposée par B. Arnaldi, P. Fuchs et J. Tisseau dans le traité de la réalité virtuelle [Fuchs *et al.*, 2006a] : *« La réalité virtuelle est un domaine scientifique et technique exploitant l'informatique et des interfaces comportementales en vue de simuler dans un monde virtuel le comportement d'entités 3D, qui sont en interaction en temps réel entre elles et avec un ou des utilisateurs en immersion pseudo-naturelle par l'intermédiaire de canaux sensori-moteurs. »*

D'une part, cette définition met en avant le fait que la réalité virtuelle permet à un utilisateur (ou plutôt à plusieurs utilisateurs dans le cas collaboratif de cette thèse) d'être en interaction avec un monde artificiel crée grâce à l'informatique, que nous appelons « monde virtuel ». La réalité virtuelle peut être vue comme une boucle dans laquelle les utilisateurs vont à la fois percevoir les éléments du monde virtuel et agir sur ces mêmes éléments que nous appelons « objets virtuels ». Leurs actions modifient la perception qu'ils ont du monde virtuel, tandis que cette perception affecte la façon dont ils agissent, d'où la notion de boucle (cf. figure 1.1).

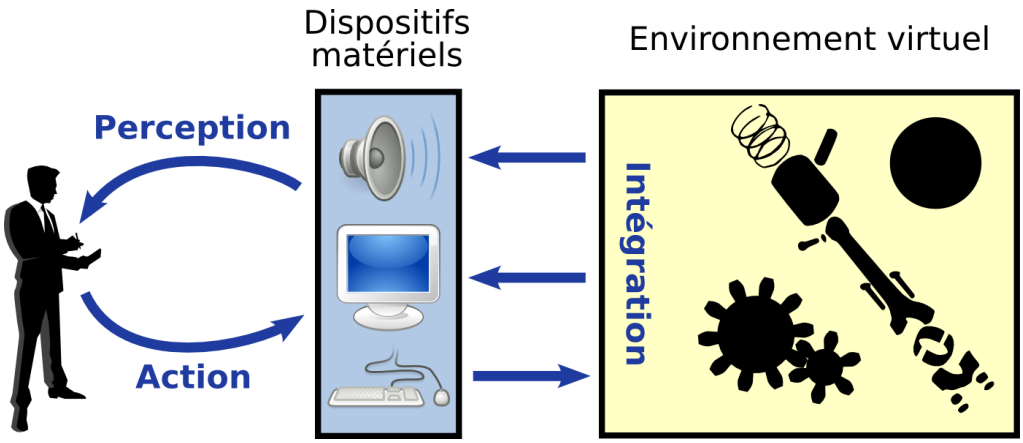


FIGURE 1.1 – Boucle perception/action décrivant un système de réalité virtuelle.

D'autre part, cette définition introduit la notion d'immersion multi-sensorielle des utilisateurs dans l'environnement virtuel. Cette immersion a pour but de créer chez les utilisateurs la sensation d'être présents dans le monde virtuel, comme expliqué par [Slater et Usoh, 1993]. Le traité de la réalité virtuelle [Fuchs *et al.*, 2006a] fait une distinction conceptuelle entre la notion d'immersion et la notion de présence. L'immersion est définie par le fait d'exposer des utilisateurs à un environnement virtuel au moyen de dispositifs occultant en partie leur perception de l'environnement réel qui les entoure. Par extension, la présence est définie comme l'état émotionnel que cette immersion induit chez les utilisateurs en leur permettant de se projeter de manière inconsciente dans le monde virtuel. Cette deuxième notion est plus subjective car elle dépend de l'état mental des utilisateurs. Par exemple, il est possible de considérer qu'une forme de présence est atteinte lorsqu'un lecteur est immergé dans son livre et ne prête plus attention au monde qui l'entoure. Dans une application de réalité virtuelle, cette sensation de présence permet aux utilisateurs soit de réaliser leurs actions d'une façon plus intuitive en leur évitant d'avoir à se projeter consciemment dans l'environnement virtuel, soit d'être placés dans un état émotionnel plus proche de celui dans lequel ils seraient s'ils réalisaient ces actions dans le monde réel.

Le monde virtuel dans lequel les utilisateurs sont immergés peut prendre des formes très diverses. Une vision restrictive présente la réalité virtuelle comme ayant pour but principal de « copier » le plus fidèlement possible le monde réel. Cependant, selon les applications, il peut être plus intéressant de créer un monde virtuel différent de la réalité. En effet, comme le monde virtuel est modélisé grâce à l'informatique, le concepteur est libre de choisir les représentations et les lois physiques qui sont les plus adaptées aux tâches à effectuer. Par exemple, dans le cas de la visualisation de données scientifiques, il peut être intéressant d'ajouter des informations qu'il ne serait pas possible de voir dans le monde réel, comme la température, la pression, etc. Il peut aussi être pratique de permettre aux utilisateurs de se déplacer en volant ou en se téléportant dans le monde virtuel.

Dans la section 1.1, nous présentons la boucle perception/action d'un environnement virtuel et la façon d'y intégrer les utilisateurs. Dans la section 1.2, nous détaillons ensuite la collaboration en environnement virtuel et les solutions proposées pour intégrer plusieurs utilisateurs au sein d'une même boucle perception/action. Ces solutions doivent permettre aux utilisateurs de se comprendre dans un monde virtuel qui n'a pas forcément les mêmes aspects et les mêmes lois physiques que le monde réel.

1.1 Environnement virtuel

Même si les termes de « monde virtuel » et d'« environnement virtuel » ont une signification très proche, nous marquons une légère distinction entre les deux. Pour nous, le monde virtuel correspond à l'ensemble des données qui permet de modéliser le monde artificiel. Tandis que l'environnement virtuel prend en compte, en plus, les utilisateurs au sein de ce monde virtuel : il correspond à la représentation multi-sensorielle que les utilisateurs vont avoir du monde virtuel. Nous pourrons dire qu'un objet virtuel est présent dans le monde virtuel, alors qu'un utilisateur est immergé dans l'environnement virtuel.

Comme présenté dans la figure 1.1, un environnement virtuel peut être caractérisé par les trois termes suivants : perception, action et intégration. Nous détaillons dans la partie 1.1.1, les moyens permettant de faire percevoir l'environnement virtuel aux utilisateurs. Dans la partie 1.1.2, nous présentons les techniques utilisées pour permettre aux utilisa-

teurs d'agir sur l'environnement virtuel. Enfin, dans la partie 1.1.3, nous étudions, d'un point de vue plus technique, les solutions existantes pour intégrer les utilisateurs au sein de la boucle perception/action en fonction en leur environnement réel et des dispositifs matériels qu'ils utilisent.

1.1.1 Perception de l'environnement virtuel

Faire percevoir un monde virtuel à des utilisateurs consiste à stimuler leurs différents canaux sensoriels pour occulter partiellement leur perception du monde réel. Cette restitution multi-sensorielle permet de faire naître chez ces utilisateurs une sensation de présence dans l'environnement virtuel. Même si des retours visuels et auditifs sont le plus souvent utilisés pour cela, il peut aussi être important de tenir compte des facteurs proprioceptifs, haptiques, olfactifs, etc. Par ailleurs, la nature interactive de l'environnement virtuel joue également un rôle important dans l'immersion des utilisateurs. En effet, le fait de les impliquer émotionnellement dans une tâche à accomplir augmente fortement leur sensation de présence dans l'environnement virtuel.

Actuellement, il est quasiment impossible d'occulter entièrement la perception que les utilisateurs ont du monde réel sur tous les canaux sensoriels. Par exemple, il est difficile d'occulter la perception que les utilisateurs ont de leur propre corps ou de la gravité dans le monde réel. L'immersion ne consiste donc pas seulement à simuler un monde virtuel pour chacun des canaux sensoriels, mais aussi à mettre en relation le monde virtuel et le monde réel pour certains de ces canaux. Cela revient à « détourner » l'environnement réel afin qu'il participe à l'immersion dans l'environnement virtuel. Par exemple, il est beaucoup plus facile de restituer le toucher d'un objet particulier en introduisant le vrai objet réel dans l'environnement virtuel (interface tangible ou *prop*).

Dans la partie 1.1.1.1, nous présentons les différents retours sensoriels permettant d'immerger les utilisateurs dans l'environnement virtuel, ainsi que certains des dispositifs matériels utilisés. Cependant, nous ne présentons pas une liste exhaustive de tous les dispositifs utilisés en réalité virtuelle. Une description très complète de ces dispositifs est proposée dans le deuxième tome du traité de la réalité virtuelle [Fuchs *et al.*, 2006b]. Dans la partie 1.1.1.2, nous détaillons plus particulièrement comment la mise en correspondance du virtuel avec le réel permet d'améliorer la sensation de présence.

1.1.1.1 Restitution multi-sensorielle

Selon les applications ciblées par le système de réalité virtuelle, il faut définir quels sont les canaux sensoriels à stimuler chez les utilisateurs afin de leur faire percevoir l'environnement virtuel d'une façon appropriée à la tâche qu'ils doivent réaliser. Un des facteurs les plus importants pour l'immersion est d'assurer une bonne synchronisation temporelle et une bonne cohérence spatiale entre ces différents retours sensoriels. En effet, une désynchronisation entre les retours sensoriels peut troubler les utilisateurs, détériorer leur sensation de présence et même, dans certains cas, leur donner le mal des simulateurs (*cybersickness*). Par ailleurs, il est aussi important de minimiser la latence lors des interactions, c'est-à-dire minimiser le temps que le système met pour répondre aux actions des utilisateurs en leur renvoyant les retours sensoriels adéquats.

FIGURE 1.2 – *Head Mounted Display.*

FIGURE 1.3 – *WorkBench* (photo © PSA).

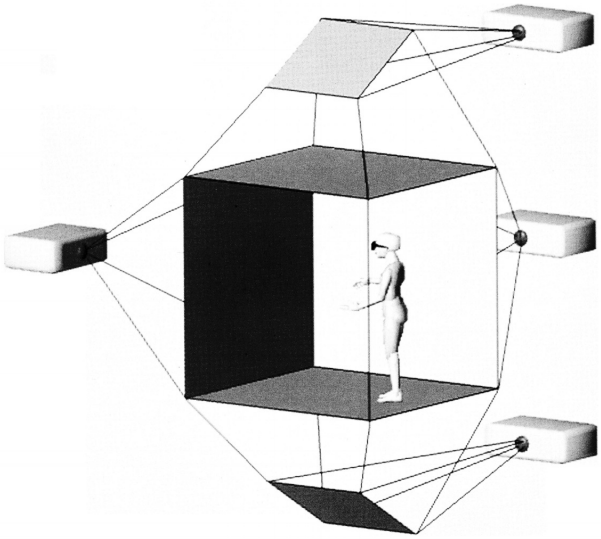


FIGURE 1.4 – Premier schéma de la littérature décrivant un CAVE® [Cruz-Neira *et al.*, 1992].

Restitution visuelle. Une restitution visuelle est indispensable dans la plupart des applications de réalité virtuelle. En effet, il est extrêmement rare qu'une application n'utilise pas un retour visuel. Il est possible de distinguer quatre facteurs qui favorisent l'immersion visuelle des utilisateurs dans l'environnement virtuel :

- un large champ de vision (dans le cas idéal, au moins égal à celui des humains),
- une vision stéréoscopique afin de permettre la perception des profondeurs,
- une haute résolution afin d'exploiter au mieux l'acuité visuelle des utilisateurs,
- une immersion totale du regard des utilisateurs, c'est-à-dire que le dispositif de visualisation permette aux utilisateurs de toujours voir le monde virtuel, même s'ils bougent ou tournent la tête dans le monde réel.

Deux types de dispositif de visualisation permettent une immersion totale du regard des utilisateurs : le visiocasque (*Head Mounted Display (HMD)* [Sutherland, 1968] - cf. figure 1.2) et le visiocube (CAVE® [Cruz-Neira *et al.*, 1992] - cf. figure 1.4). Les *HMD* n'offrent pas un grand champ de vision, ni une bonne résolution car ils sont constitués de petits écrans placés près des yeux de l'utilisateur. Les CAVE® permettent généralement d'avoir une meilleure résolution et un champ de vision non limité par le dispositif matériel, mais l'espace de déplacement des utilisateurs est souvent restreint à l'intérieur du dispositif.

D'autres dispositifs de visualisation utilisent des écrans suffisamment grands pour que les utilisateurs n'en voient pas les limites lorsqu'ils sont en train d'interagir dans le dispositif. Ces dispositifs généralement appelés « salle immersive » sont composés au moins de deux grands écrans ou d'un écran cylindrique. Ces salles immersives offrent également une immersion importante du regard des utilisateurs, mais qui n'est cependant pas totale.

Pour les applications où les utilisateurs restent quasi-immobiles devant le dispositif ou lorsqu'ils interagissent avec un monde virtuel constitué d'un seul objet central, les dispositifs de visualisation de type « bureau immersif » suffisent pour avoir une bonne

immersion du regard. Ils sont généralement composés d'un grand écran incliné ou de deux écrans perpendiculaires (*WorkBench* [Krüeger et Fröehlich, 1994] - cf. figure 1.3).

De nombreux autres dispositifs de visualisation peuvent être utilisés pour les applications de réalité virtuelle en allant du grand écran sur lequel sont projetées des images grâce à un vidéo-projecteur stéréoscopique jusqu'au simple écran monoscopique. Ces dispositifs n'offrent qu'une immersion partielle du regard des utilisateurs et ne répondent pas à tous les critères favorisant l'immersion, mais ils peuvent suffire à créer chez les utilisateurs une sensation de présence en adéquation avec les besoins de l'application. De nos jours, les écrans stéréoscopiques se développent de plus en plus pour le grand public : écrans auto-stéréoscopiques, télévisions 3D, etc. Ces écrans peuvent être une bonne solution pour développer des applications de réalité virtuelle pertinentes à moindre coût.

Restitution sonore. De nombreux systèmes de réalité virtuelle utilisent des retours sonores pour faire percevoir aux utilisateurs l'ambiance sonore du monde virtuel, les bruits que font des objets ou les entités virtuels, les bruits induits par les utilisateurs lors de leurs interactions, les contacts entre des objets virtuels, etc. Dans la plupart des applications de réalité virtuelle, ces retours sonores sont couplés au moins aux retours visuels. Pour favoriser la sensation de présence des utilisateurs dans le monde virtuel, ces retours sonores doivent être spatialisés. La spatialisation du son est réalisée par une modulation du son contrôlée par logiciel en fonction du dispositif de restitution sonore utilisé (hauts parleurs avec des dispositions différentes ou casque mono-utilisateur).

La spatialisation du son est très complexe. Elle est, à elle seule, un domaine de recherche particulier. Nous présentons simplement ici les deux approches principalement utilisées. La première consiste à jouer sur la différence de niveau sonore entre les deux oreilles des utilisateurs, ainsi que sur le temps d'arrivée du son aux deux oreilles [Kapralos, 2003]. Cela crée l'illusion d'une provenance du son différente de la position physique des haut-parleurs. Cette technique peut être utilisée avec un casque mono-utilisateur car elle utilise uniquement une différence de son entre les deux oreilles. Par contre, quelque soit le dispositif utilisé, elle nécessite de connaître la position de la tête des utilisateurs. La deuxième approche cherche à reconstruire de manière physiquement réaliste le champ acoustique dans la zone qui entoure les utilisateurs [Kleiner *et al.*, 1993]. À l'inverse de la première, cette approche peut permettre d'avoir plusieurs auditeurs simultanés, mais elle nécessite des dispositif plus complexes, ainsi que des ressources de calcul plus importantes.

Restitution haptique. La restitution haptique regroupe les retours tactiles, ainsi que les retours proprioceptifs, c'est-à-dire les retours qui ont trait à la perception du corps dans l'espace. Ce type de restitution est moins développé dans les applications de réalité virtuelle que les restitutions visuelles et sonores, principalement à cause des dispositifs mécaniques complexes qu'il nécessite souvent. Cependant, il permet d'augmenter de manière importante la sensation de présence des utilisateurs et tend à se développer de plus en plus.

Premièrement, les retours tactiles cherchent à recréer la sensation du toucher généralement au niveau des doigts. Le but est de faire croire aux utilisateurs qu'ils touchent une surface virtuelle avec une texture particulière. Le principe consiste à stimuler les récepteurs sensoriels de la peau grâce à des stimulations vibratoires ou électriques. Cependant, il n'existe pas de solutions suffisamment génériques pour reproduire la sensation du toucher pour n'importe quelle surface. De plus, ces solutions permettent généralement de ne faire ressentir que des surfaces plates, et il est difficile de donner aux utilisateurs l'impression

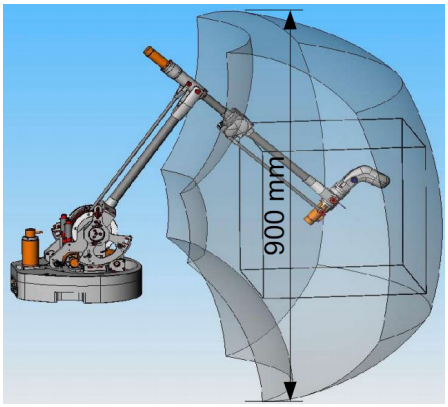


FIGURE 1.5 – Espace d'interaction associé à un bras à retour d'effort (photo © Haption).

FIGURE 1.6 – Exo-squelette appliquant un retour d'effort sur la main (photo © CyberGlove Systems).

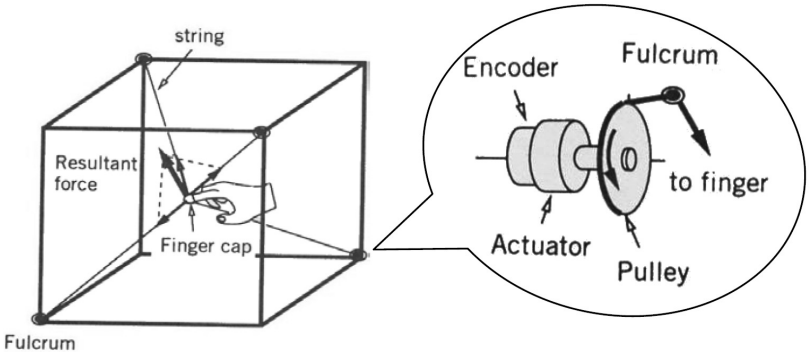


FIGURE 1.7 – Schéma décrivant le fonctionnement d'un SPIDAR [Sato, 2002].

qu'ils touchent un objet en 3D. Lorsque les utilisateurs doivent manipuler un objet bien particulier (outil spécifique à l'application), il peut être intéressant d'utiliser une interface tangible, c'est-à-dire d'introduire l'objet réel dans le monde virtuel afin de permettre à l'utilisateur d'avoir un retour tactile réaliste.

Deuxièmement, les retours proprioceptifs ont pour but de donner aux utilisateurs l'impression que leur corps ou une partie de leur corps est en contact physique avec l'environnement virtuel. Cela augmente fortement leur sensation de présence car ils ont l'impression d'agir physiquement sur le monde virtuel et inversement. Pour réaliser cette interaction, les dispositifs à retour d'effort utilisés ont une double fonctionnalité : d'une part, ils captent les mouvements des utilisateurs et leur permettent de manipuler les objets virtuels, et d'autre part, ils restituent aux utilisateurs les forces qui s'appliquent à ces objets dans le monde virtuel. Il existe de nombreux types de dispositifs à retour d'effort : les bras à retour d'effort (cf. figure 1.5), les exo-squelettes pour la main, le bras ou le corps (cf. figure 1.6), ainsi que les dispositifs à fils de type SPIDAR [Sato, 2002] (cf. figure 1.7).

Pour certaines tâches, en particulier pour celles de locomotion, il peut être pertinent pour la sensation de présence de mettre en relation les mouvements réels des utilisateurs avec des actions dans le monde virtuel. En effet, lorsque les utilisateurs réalisent les mouvements réels, ils ont les retours proprioceptifs appropriés à la tâche qu'ils effectuent. Pour

cela, les mouvements des utilisateurs doivent être captés grâce à un système de *tracking*, comme par exemple un système utilisant des capteurs magnétiques ou des caméras infrarouges. Dans certains cas, il faut en plus « accompagner » les utilisateurs afin de les soustraire aux contraintes physiques. Par exemple, dans le cas de la marche en environnement virtuel, les utilisateurs peuvent marcher dans le monde réel, mais ils risquent vite d'atteindre les limites du dispositif de visualisation ou du système de *tracking*. Pour éviter ce problème, [Darken *et al.*, 1997] proposent de faire marcher les utilisateurs sur un tapis roulant à deux dimensions afin de les garder au centre de leur dispositif immersif. Cependant, ce type de solution est coûteux et complexe à mettre en œuvre.

Enfin, nous incluons dans les retours proprioceptifs ce que les utilisateurs perçoivent au niveau de leur système vestibulaire. Ce système permet de détecter l'orientation du corps dans l'espace, ainsi que ses accélérations/décélérations en fonction de la gravité. Dans le monde réel déjà soumis à la gravité terrestre, il est quasiment impossible de masquer la perception que les utilisateurs ont de la gravité. Il est donc seulement possible d'en modifier la perception grâce à des dispositifs matériels complexes, encombrants et coûteux (simulateurs, plateformes mobiles, etc.). Ces dispositifs permettent d'éviter qu'il y ait un décalage entre la vision des utilisateurs et leur perception vestibulaire, car ce décalage entraine le mal des simulateurs, comme étudié par exemple par [Ujike, 2009].

Autres restitutions sensorielles. Il existe d'autres facteurs qui contribuent à la sensation de présence des utilisateurs dans l'environnement virtuel. Cependant, ils sont utilisés uniquement dans certaines applications très spécifiques. Nous en faisons donc juste une rapide énumération. Premièrement, il peut être intéressant de changer la température et l'humidité dans le dispositif immersif afin de recréer les conditions climatiques du monde virtuel, comme avec le GEPAT développé sous la responsabilité de Laurent Todeschini (DGA). Deuxièmement, il existe des dispositifs qui permettent de restituer les odeurs, comme celui proposé par [Matsukura *et al.*, 2011]. Ces dispositifs peuvent soit diffuser une odeur d'ambiance, soit restituer des odeurs plus localisées afin de pouvoir changer l'odeur sentie par les utilisateurs en fonction de leurs déplacements dans le monde virtuel. Cependant, il est difficile de gérer précisément la diffusion et la persistance de ces odeurs. Enfin, certaines applications utilisent une restitution gustative afin de faire percevoir aux utilisateurs le goût d'aliments virtuels, comme celle proposée par [Narumi *et al.*, 2011].

1.1.1.2 Co-localisation

Lorsque des utilisateurs sont immergés dans un environnement virtuel, il est quasiment impossible d'occulter entièrement la perception qu'ils ont de l'environnement réel (gravité, limites de déplacement physique, perception du corps, etc.). Dans la partie précédente, nous avons expliqué qu'il est parfois plus simple, voire bénéfique pour la sensation de présence, d'intégrer des éléments du monde réel dans le monde virtuel, comme le corps des utilisateurs ou une interface tangible. Qu'ils soient volontairement ou involontairement introduits dans le monde virtuel, ces objets réels doivent être mis en relation avec leurs représentations virtuelles pour chacun des canaux sensoriels. Par exemple, un outil réel manipulé par un utilisateur doit avoir une action ou une ombre dans le monde virtuel (retour visuel), il doit restituer les chocs avec les objets virtuels (retour haptique) et ces chocs doivent générer des sons spatialisés (retour auditif). Cette mise en correspondance spatiale entre les objets réels et leurs représentations virtuelles, que nous appelons « co-localisation », est un facteur

important pour la sensation de présence des utilisateurs. En effet, si l'objet réel n'est pas représenté sur tous les canaux sensoriels ou s'il existe un décalage entre sa position réelle et celle de ses représentations virtuelles, cet objet réel aura du mal à « s'intégrer » dans le monde virtuel : les utilisateurs auront plus de mal à se projeter dans l'environnement virtuel et cela peut nuire à leur sensation de présence.

Les objets réels co-localisés dans le monde virtuel peuvent être de natures très différentes : la salle immersive et ses limites physiques, la tête ou le corps des utilisateurs, des outils d'interaction, etc. La co-localisation de la tête *(head-tracking)* permet d'adapter la vision que les utilisateurs ont du monde virtuel. Par ailleurs, les dispositifs d'interaction réels sont souvent mis en relation avec leur action dans le monde virtuel, mais la co-localisation peut être généralisée à tout élément réel qui est intégré dans le monde virtuel.

Co-localisation de la tête des utilisateurs. Dès lors que les utilisateurs peuvent se déplacer physiquement au sein du dispositif immersif qu'ils utilisent (CAVE®, salle immersive, *HMD* dans une salle équipée d'un système de *tracking*, etc.), il est nécessaire de co-localiser la tête des utilisateurs avec la représentation qu'ils perçoivent du monde virtuel. En effet, il faut déterminer la position de la tête des utilisateurs *(head-tracking)* pour adapter en fonction de cette position les informations qui sont envoyées sur les différents canaux sensoriels. Les utilisateurs peuvent ainsi se déplacer physiquement dans le dispositif immersif tout en ayant des retours sensoriels cohérents avec leur position. Ils peuvent alors explorer l'environnement virtuel (sur de petites distances) ou se placer pour interagir avec un objet virtuel de façon intuitive grâce à leurs déplacements dans le monde réel. Cela augmente de manière importante la sensation de présence dans l'environnement virtuel.

Pour réaliser cette co-localisation, il faut d'abord établir une corrélation entre le repère réel du dispositif immersif et le repère global du monde virtuel. Cette corrélation peut être amenée à changer dynamiquement si les utilisateurs peuvent naviguer dans le monde virtuel (sur des distances plus grandes que celles du dispositif immersif). Pour cela, il faut repérer la position de la tête des utilisateurs dans le monde réel grâce à un système de *tracking*, comme ceux utilisant des capteurs magnétiques ou des caméras infrarouges. Cette position réelle est mise en relation avec une position dans le monde virtuel afin d'adapter en conséquence le modèle visuel, sonore, etc. Ces adaptations pour chacun des retours sensoriels sont entièrement dépendantes du modèle et du dispositif de restitution utilisés. Afin d'illustrer ces adaptations, nous prendrons l'exemple du retour visuel, car cette adaptation est couramment utilisée dans de nombreux dispositifs de visualisation.

Pour les dispositifs de visualisation utilisant des écrans (par opposition au *HMD*) et une vision stéréoscopique, il est nécessaire d'adapter les images affichées en fonction de la position de la tête de l'utilisateur. [Arsenault et Ware, 2000] mettent en avant le fait que toute image en perspective a un point appelé « centre de perspective ». Dès qu'une image est vue d'une position différente du centre de perspective, des distorsions apparaissent dans l'image. Lorsqu'un utilisateur se déplace physiquement par rapport à l'écran, cette distorsion des images entraîne des mouvements des formes 3D dans le monde virtuel (cf. figure 1.8). Ces mouvements ne sont pas réalistes, mais surtout ils dégradent la sensation de présence et peuvent être très dérangeants pour cet utilisateur. Il est donc nécessaire de déformer la pyramide de vue en fonction de la position de sa tête, c'est-à-dire de recalculer les images en décalant le centre de perspective (cf. figure 1.9).

Lorsque plusieurs utilisateurs sont présents dans un même dispositif de visualisation, il faut être capable d'afficher une image particulière pour chacun des utilisateurs (et même

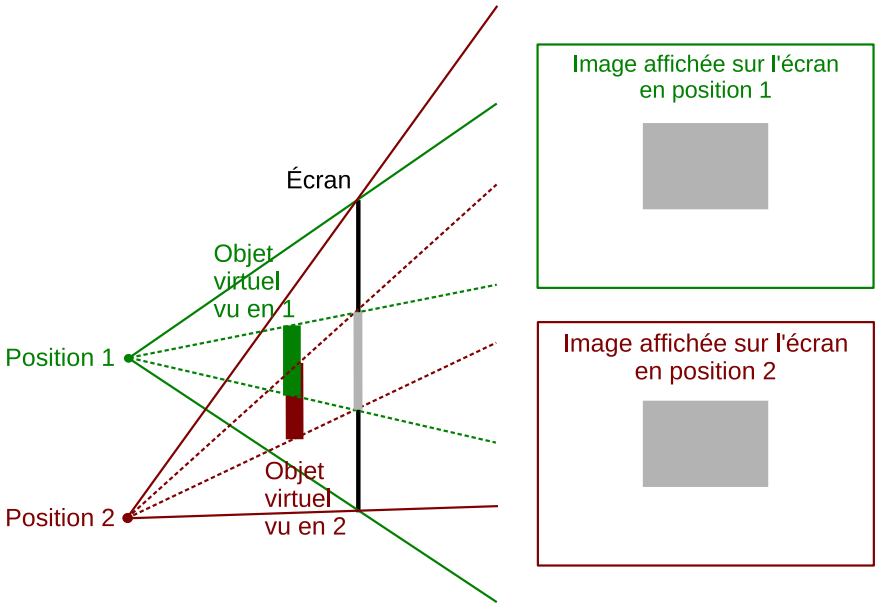


FIGURE 1.8 – Sans déformation de la pyramide de vue : quelle que soit la position de l'utilisateur (1 ou 2), l'objet a la même position grise sur l'écran. Avec une vision stéréoscopique, pour un objet virtuel situé en avant de l'écran, l'utilisateur situé en 1 voit l'objet à la position verte. S'il se déplace en 2, il verra l'objet à la position rouge. Lors de son déplacement, il verra donc l'objet « bouger » de la position verte à la position rouge. Il se produit un mouvement équivalent (sens inversé) pour les objets virtuels situés en arrière de l'écran.

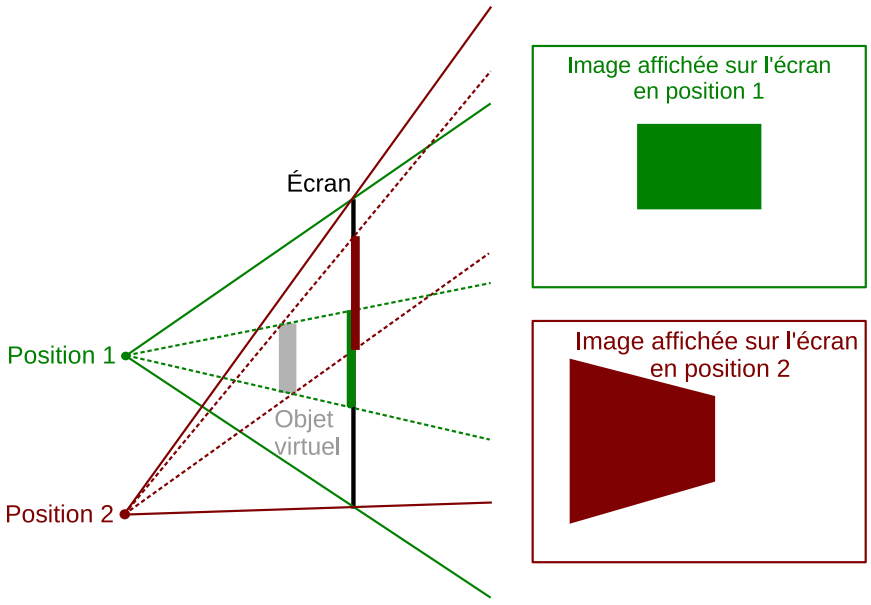


FIGURE 1.9 – Avec déformation de la pyramide de vue : l'image est calculée en fonction de la position de l'utilisateur. Lorsque l'utilisateur se trouve en 1, l'écran affiche l'objet virtuel à l'endroit vert. S'il se déplace en 2, l'image est recalculée afin d'afficher l'objet virtuel à l'endroit rouge. Quelle que soit la position de l'utilisateur (1 ou 2), il verra l'objet à la position grise. Il en est de même pour les objets virtuels situés en arrière de l'écran.

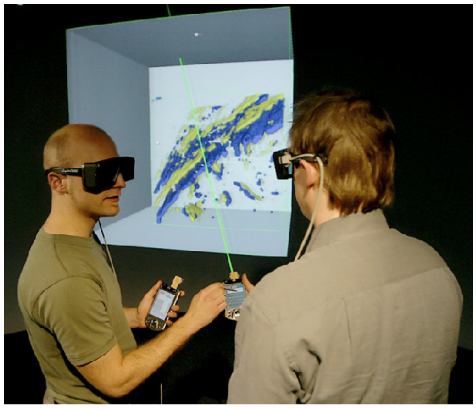


FIGURE 1.10 – PDA matérialisant la source du rayon virtuel [Simon, 2005].

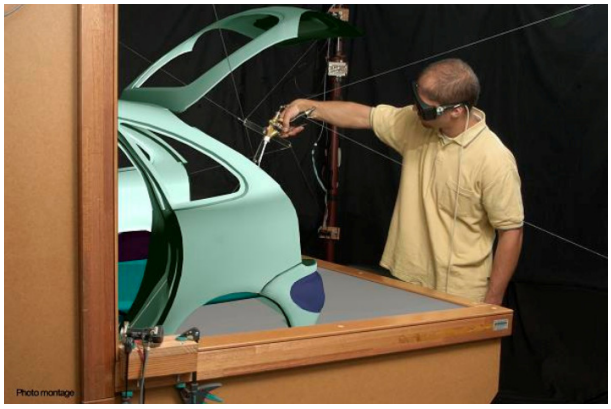


FIGURE 1.11 – Interface tangible servant à poser du mastic virtuel [Ortega et Coquillart, 2005].

deux avec la stéréoscopie) pour adapter le retour visuel de chacun d'entre eux. En effet, vu que le retour visuel dépend de la position de la tête de chaque utilisateur et qu'ils n'ont pas forcément la même position dans le dispositif, il faut afficher un point de vue particulier par utilisateur sans que ce point de vue ne soit visible par les autres. Des dispositifs, comme celui proposé par [Agrawala *et al.*, 1997], permettent d'afficher plusieurs vues simultanées en jouant sur la fréquence d'affichage et sur la polarisation des images, mais ils sont encore rares. Pour remédier à ce problème, [Marbach, 2009] propose de fusionner les images de plusieurs utilisateurs ayant des points de vues différents afin d'afficher une image légèrement déformée qui contente tous les utilisateurs.

Co-localisation de l'outil d'interaction. Une co-localisation est souvent réalisée sur les outils d'interaction que les utilisateurs manipulent afin d'associer l'objet réel manipulé avec sa représentation et, par extension, son action sur le monde virtuel. Cette co-localisation de l'outil d'interaction permet d'augmenter le réalisme des actions réalisées par les utilisateurs, et donc la précision et la rapidité d'exécution de celles-ci. Par exemple, [Simon, 2005] effectue une co-localisation de PDA réels qui servent de source à un rayon virtuel permettant de manipuler des objets virtuels (cf. figure 1.10). Ou encore, [Ortega et Coquillart, 2005] réalisent la co-localisation d'un pistolet à mastic réel permettant de poser du mastic virtuel sur une portière de voiture virtuelle. L'utilisation de l'objet réel qui serait normalement utilisé pour faire l'action dans le monde réel (interface tangible) permet d'augmenter la sensation de présence grâce à un retour tactile réaliste (cf. figure 1.11). L'outil d'interaction peut être soit un objet réel permettant de matérialiser l'outil virtuel, soit directement une partie du corps de l'utilisateur (pour les techniques de manipulation utilisant une main virtuelle, par exemple). De la même façon que pour la tête des utilisateurs, des capteurs de localisation sont utilisés pour repérer cet outil d'interaction dans le monde réel et ainsi recalculer la position de ses représentations dans le monde virtuel. Enfin, au niveau visuel, pour permettre une bonne mise en correspondance quelle que soit la position de l'utilisateur qui manipule l'outil, il est important de réaliser en même temps une co-localisation de sa tête. En effet, si cet utilisateur bouge uniquement la tête (mais pas l'outil d'interaction), il faut malgré tout recalculer les images affichées afin que l'utilisateur voie bien l'outil réel et sa représentation virtuelle correctement alignés.

Généralisation de la co-localisation. La co-localisation peut être généralisée à l'intégration de n'importe quel objet réel qui a un rôle ou un impact dans l'environnement virtuel. Premièrement, il peut être intéressant de co-localiser dans le monde virtuel des objets réels qui ont un rôle dans la perception ou les interactions des utilisateurs. Cela peut permettre des retours sensoriels supplémentaires ou plus réalistes (en particulier, pour les retours haptiques). Par exemple, il est possible d'imaginer co-localiser une table réelle pour y poser des outils virtuels. Deuxièmement, il peut également être nécessaire de co-localiser certains objets, parties ou phénomènes du monde réel qui ne peuvent pas être masqués et qui impactent la sensation de présence des utilisateurs. C'est le cas, par exemple, des limites de déplacement physique dans un dispositif immersif (définies par les écrans de visualisation ou par l'espace repéré par le système de *tracking*). Ces limites peuvent être co-localisées afin de les faire percevoir aux utilisateurs, et ainsi de leur permettre d'adapter leurs interactions afin de ne pas franchir ces limites.

1.1.2 Action sur l'environnement virtuel

De nombreuses techniques sont utilisées pour permettre aux utilisateurs d'agir sur l'environnement virtuel qu'ils perçoivent. [Hand, 1997] propose de classer les techniques génériques en trois catégories : la manipulation des objets virtuels (interaction), la manipulation du point de vue (navigation), et le contrôle de l'application. Le contrôle de l'application décrit la communication entre les utilisateurs et le système qui contrôle l'environnement virtuel. Dans la partie 1.1.2.1, nous étudions d'abord les différentes techniques de navigation, puis nous présentons dans la partie 1.1.2.2, les techniques interactions qui permettent aux utilisateurs de manipuler des objets virtuels. Dans cette partie, nous ne détaillons pas le contrôle de l'application étant donné que ce type d'interaction ne correspond pas aux actions directes des utilisateurs sur l'environnement virtuel.

1.1.2.1 Techniques de navigation

La navigation est l'interaction qui consiste à manipuler un objet virtuel particulier : le point de vue d'un utilisateur. Cela permet de modifier la perception qu'il a du monde virtuel. Généralement, cet utilisateur manipule son propre point de vue, mais il est possible d'imaginer qu'un utilisateur puisse manipuler le point de vue d'un autre dans un environnement virtuel collaboratif. Dans le cas d'un environnement virtuel multi-échelle, la navigation regroupe le déplacement du point de vue, mais aussi les changements d'échelle et donc de champ de vision (niveau de zoom) comme l'explique [Hand, 1997].

Par extension, [Bowman *et al.*, 2004] précisent que la navigation peut également intégrer la recherche d'itinéraire dès lors que les utilisateurs se déplacent dans le monde virtuel. Cette partie cognitive considère les tâches de se repérer dans le monde virtuel et d'y trouver leur chemin. Cependant, nous n'abordons pas dans cette partie la recherche d'itinéraire ou de chemin, ni l'algorithmique qui va avec (recherche du plus court chemin, etc.).

Modes de navigation « classique ». La navigation est la tâche la plus couramment effectuée dans les environnements virtuels : la plupart des applications de réalité virtuelle permettent aux utilisateurs de naviguer dans le monde virtuel. La navigation est donc largement décrite dans la littérature traitant des interactions. Nous réalisons dans cette partie un panorama des paradigmes « classiques » de navigation en se basant sur le livre de [Bowman *et al.*, 2004] et le rapport de Master Recherche de [Thomas, 2005].

Dans un environnement virtuel, les utilisateurs ne sont pas soumis aux contraintes matérielles du monde réel ce qui permet d'obtenir une plus grande souplesse lors de la navigation (au niveau de la vitesse, de la continuité du déplacement, etc.). Cependant, cela peut parfois être déroutant pour les utilisateurs. Nous présentons les différents modes de navigation « classique » en les classant selon le référentiel utilisé pour le déplacement.

Navigation égocentrique. Cette catégorie regroupe les modes de navigation où les déplacements se font relativement à l'utilisateur. Généralement, cet utilisateur spécifie dans son repère une vitesse et une direction de déplacement. Il existe principalement deux modes de navigation égocentrique :

- **La métaphore de la marche :** la marche est le mode de déplacement qui est le plus intuitif. Comme dans la vie de tous les jours, l'utilisateur se déplace selon deux dimensions toujours à la même hauteur par rapport au sol du monde virtuel (3 degrés de liberté). Ce mode de navigation peut être contrôlé grâce à des mouvements physiques de l'utilisateur (marche sur place [Slater *et al.*, 1995], marche sur un tapis roulant [Brooks *et al.*, 1992]), à la métaphore d'un véhicule comme par exemple l'*Uniport* de Sarcos (vélo), ou à des techniques plus classiques comme le déplacement dans la direction du regard ou de la main [Mine, 1995b], ou encore le déplacement dans la direction spécifiée par un périphérique d'interaction (*joystick*, etc.).
- **La métaphore du vol :** le déplacement du point de vue se fait librement dans les trois dimensions (6 degrés de liberté). Bien que ce mode de navigation soit une technique facile à programmer, [Hand, 1997] souligne qu'il n'est pas vraiment naturel et facile à utiliser pour les utilisateurs. Ce mode de navigation est souvent contrôlé grâce à un périphérique d'interaction à trois dimensions [Ware et Osborne, 1990], mais il est possible d'utiliser la direction du regard ou de la main [Mine, 1995b], ainsi que la métaphore du poste de pilotage d'un engin « volant » (simulateur de vol).

Navigation exocentrique. Cette catégorie regroupe les modes de navigation où les déplacements ne sont pas spécifiés par rapport à un utilisateur, mais par rapport à un élément du monde virtuel. Ce type de navigation peut permettre une rotation relative autour d'un objet virtuel afin de faciliter son observation, ou un placement automatique par rapport à cet objet comme le propose la technique *Navidget* [Hachet *et al.*, 2008]. [Bowman *et al.*, 2004] incluent également dans cette catégorie la technique *Grabbing the air* [Mapes et Moshell, 1995], qui permet de se déplacer en se « tirant » dans le monde virtuel avec les mains. Cette technique est généralement contrôlée par les gestes des utilisateurs.

Navigation assistée. Au lieu de spécifier une vitesse et une direction de déplacement, les utilisateurs choisissent directement une destination à atteindre ou un chemin à suivre. Il existe de nombreux moyens pour permettre aux utilisateurs de sélectionner la nouvelle position de leur point de vue. Nous pouvons citer par exemple :

- L'utilisation d'un rayon virtuel pour pointer la destination [Bowman *et al.*, 1999].
- L'utilisation d'un monde en miniature (*World In Miniature (WIM)* - [Stoakley *et al.*, 1995]) qui est une sorte de carte 2D ou 3D du monde virtuel. Les utilisateurs peuvent positionner directement leur représentation dans le monde en miniature à l'endroit où ils veulent se rendre [Pausch *et al.*, 1995]. Cette technique permet aussi de placer le champ de vision directement dans la direction souhaitée.

– L'utilisation d'une liste de chemins prédéfinis dans laquelle les utilisateur n'ont qu'à choisir le chemin qu'ils veulent parcourir. Cette technique de navigation est généralement utilisée pour des visites « guidées » dans le monde virtuel [Galyean, 1995].

Une fois la nouvelle destination choisie par les utilisateurs, ce type de navigation n'a pas besoin de tenir compte du temps de déplacement ni du trajet à suivre pour se rendre à cette nouvelle position. Diverses techniques de déplacements peuvent donc être utilisées, et elles n'ont pas forcément besoin d'être réalistes. Nous pouvons distinguer parmi ces techniques de déplacements :

- **La métaphore de la « téléportation » :** les utilisateurs se retrouvent instantanément à la destination choisie [Ruddle *et al.*, 2000]. Cela permet un déplacement très rapide, mais peut être relativement déroutant pour les utilisateurs.
- **L'interpolation des positions :** cette technique consiste à interpoler le déplacement entre la position de départ et celle d'arrivée [Mackinlay *et al.*, 1990], ce qui permet de remédier au problème de désorientation provoqué par la « téléportation ».
- **La métaphore de la visite guidée :** cette métaphore permet de faire découvrir à l'utilisateur des éléments importants de l'environnement virtuel en le faisant passer par un chemin permettant de percevoir ces éléments [Elmqvist *et al.*, 2007].

Modes de navigation multi-échelle. Le changement d'échelle peut être ajouté à la navigation comme un degré de liberté supplémentaire, car il contribue à modifier la perception que l'utilisateur a du monde virtuel [Zhang et Furnas, 2002][Zhang et Furnas, 2005]. De manière générale, le changement d'échelle se fait de façon uniforme selon les 3 dimensions : les utilisateurs ne contrôlent pas le changement d'échelle sur chacune des 3 dimensions séparément (il n'y a qu'un seul degré de liberté). Dans la majorité des cas, le changement d'échelle d'un utilisateur est associé à une modification de sa vitesse de déplacement [Kopper *et al.*, 2006]. L'usage veut que plus un utilisateur est petit dans le monde virtuel, moins sa vitesse de déplacement sera importante et inversement. Cela permet à l'utilisateur de modifier le compromis entre sa vitesse de déplacement et la précision de son déplacement. Il existe deux modélisations possibles pour permettre les changements d'échelle des utilisateurs dans un monde virtuel :

– la taille des utilisateurs est modifiée et le monde virtuel reste à la même échelle,
– l'échelle du monde virtuel est modifiée et les utilisateurs gardent la même taille.

Dans le cas d'un environnement virtuel collaboratif, ce deuxième mode est plus contraignant car il impose que tous les utilisateurs aient la même échelle dans le monde virtuel. En effet, dès lors qu'un utilisateur change d'échelle, il oblige tous les autres utilisateurs à changer d'échelle étant donné qu'il modifie l'échelle du monde virtuel partagé.

La possibilité de changer d'échelle peut être considérée comme un degré de liberté supplémentaire qui vient s'ajouter à la navigation « classique ». En effet, les utilisateurs peuvent augmenter ou réduire leur taille en plus des actions de déplacement. Les différentes techniques de navigation multi-échelles peuvent être séparées en deux catégories : les techniques manuelles et les techniques automatiques.

Changements d'échelle manuels. Les utilisateurs modifient directement leur taille dans le monde virtuel en utilisant deux commandes : l'une permettant de grandir et l'autre de rétrécir. Cette technique est très basique, mais elle est souvent utilisée dans les environnements virtuels multi-échelles car elle est très simple à implémenter. Elle peut être

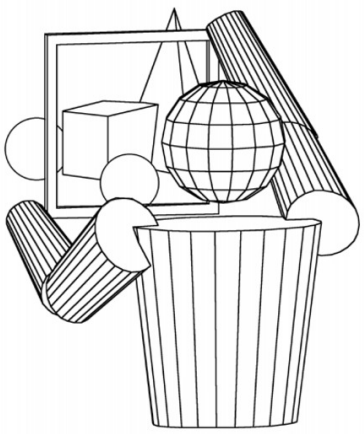


FIGURE 1.12 – Sélection de la zone à zoomer en définissant un rectangle avec les mains [Mine *et al.*, 1997].

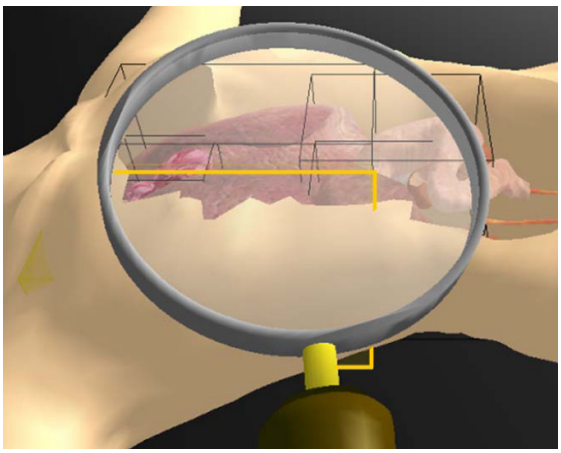


FIGURE 1.13 – Boîtes englobantes permettant de visualiser les différents niveaux de détail [Kopper *et al.*, 2006].

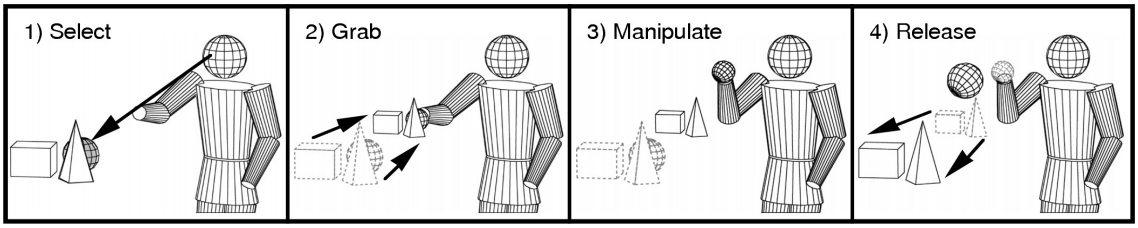


FIGURE 1.14 – « Scaled-world Grab » : changement d'échelle automatique pour la manipulation d'objets [Mine *et al.*, 1997].

couplée à n'importe quel mode de navigation « classique » étant donné qu'elle gère indépendamment le degré de liberté associé à l'échelle. D'après [Kopper *et al.*, 2006], cette technique n'est pas facile à contrôler pour les utilisateurs (difficulté à contrôler en même temps le changement d'échelle et le déplacement), et elle n'est pas toujours performante : par exemple, il est difficile d'utiliser le changement d'échelle pour jouer sur les vitesses de déplacement afin d'avoir le meilleur compromis précision/rapidité de déplacement en fonction des actions que les utilisateurs effectuent.

La technique *Head-butt Zoom* proposée par [Mine *et al.*, 1997] peut également être classée dans les changements d'échelle manuels. Cette technique permet de passer facilement d'une vue normale à une vue détaillée (niveau de zoom plus important). Pour sélectionner la vue détaillée, les utilisateurs matérialisent avec leurs deux mains un rectangle dont l'emplacement définit la zone à zoomer et dont la taille définit le coefficient d'agrandissement (cf. figure 1.12). Puis, par un simple mouvement du corps, l'utilisateur peut changer de vue pour passer à la vue détaillée. Ce mode de sélection se rapproche de celui des logiciels 2D (de traitement d'image ou de dessin) où la zone à zoomer est définie en traçant la diagonale d'un rectangle avec la souris.

Changements d'échelle automatiques. [Kopper *et al.*, 2006] ont montré que le changement d'échelle automatique, lorsqu'il est possible, est plus efficace que le changement d'échelle manuel. Cependant, ce changement automatique nécessite des informations sup-

plémentaires pour déterminer quelle serait l'échelle appropriée. Les techniques mettant en œuvre des changements d'échelle automatiques sont les suivantes :

- **Les boîtes englobantes :** [Kopper *et al.*, 2006] proposent de décomposer le monde virtuel en une hiérarchie de boîtes englobantes qui représentent chacune un niveau de détails. Chacune des boîtes est associée à une échelle qui correspond à la bonne taille pour voir les détails présents dans la boîte. Une « loupe » permet de voir les différentes boîtes englobantes (cf. figure 1.13). Les auteurs proposent deux techniques de navigation :
  - Les utilisateurs sélectionnent grâce à la « loupe » la boîte englobante qu'ils veulent explorer (niveau de détail inférieur). Ils sont alors automatiquement mis à l'échelle associée à la boîte et placés en son centre. Ils peuvent alors naviguer dans la boîte en utilisant la métaphore du vol. Une simple commande permet de retourner dans la boîte de niveau supérieur.
  - Les utilisateurs peuvent se déplacer dans le monde virtuel en utilisant la métaphore du vol. Dès qu'ils entrent dans une boîte associée à une échelle plus petite, ils sont automatiquement réduits. À l'inverse, dès qu'ils ressortent d'une boîte, ils sont automatiquement agrandis à l'échelle de la boîte de niveau supérieur.

  D'après les expériences réalisées par [Kopper *et al.*, 2006], la première technique semble être la plus efficace pour réaliser une tâche précise, tandis que la deuxième permet plus de liberté de déplacement afin d'explorer le monde virtuel.
- ***Scaled-world grab* :** cette technique, proposée par [Mine *et al.*, 1997], se rapproche de la technique *Grabbing the air* vue précédemment avec en plus la notion de changement d'échelle. Elle consiste à réduire l'échelle du monde virtuel afin que l'objet sélectionné par un utilisateur puisse être à portée de mains de ce dernier (cf. figure 1.14). [Mine *et al.*, 1997] précisent que cette technique, classiquement utilisée pour manipuler des objets, peut permettre aussi de se déplacer dans le monde virtuel. En effet, il suffit qu'un utilisateur devienne grand, attrape un objet dans la direction vers laquelle il veut se déplacer, se tire vers cet objet et reprenne une échelle plus petite. Il peut ainsi atteindre toutes les destinations visibles d'un simple geste.

1.1.2.2 Techniques d'interaction

Les interactions qui permettent aux utilisateurs d'agir sur le monde virtuel peuvent être très différentes en fonction des tâches que les utilisateurs doivent effectuer dans l'environnement virtuel. Dans certains cas, il faut trouver des techniques qui permettent aux utilisateurs d'effectuer des actions similaires à celles qu'ils effectueraient dans le monde réel. Dans d'autres cas, il faut trouver des métaphores d'interaction qui permettent aux utilisateurs d'agir de manière plus simple et plus efficace afin de leur permettre de réaliser des tâches qu'ils ne pourraient pas faire ou, du moins, pas aussi facilement dans le monde réel. En conséquence, il existe un très grand nombre de techniques et de métaphores d'interaction qui varient en fonction des différentes applications, mais aussi en fonction des dispositifs immersifs utilisés. Dans cette partie, nous réalisons un rapide panorama des différentes techniques couramment utilisées dans les environnements virtuels en se basant sur le livre de [Bowman *et al.*, 2004] et sur le manuscrit de thèse de [Aguerreche, 2010].

Comme pour la navigation, les techniques d'interaction sont parfois classées en deux catégories : égocentrique et exocentrique. Les interactions égocentriques regroupent les techniques où les utilisateurs agissent depuis leur propre point de vue comme ils le feraient

dans le monde réel. Les interactions exocentriques regroupent les techniques où les utilisateurs prennent un point de vue extérieur au monde virtuel afin d'agir plus facilement. Les utilisateurs peuvent prendre ce point de vue extérieur en utilisant, par exemple, un monde en miniature [Stoakley *et al.*, 1995] ou la technique du *Scaled-world grab* [Mine *et al.*, 1997]. Néanmoins, lorsque les utilisateurs interagissent de manière égocentrique ou exocentrique, les techniques utilisées pour manipuler les objets virtuels sont souvent similaires. En effet, les tâches à effectuer sont généralement identiques que les utilisateurs agissent directement sur le monde virtuel ou sur une « maquette » du monde virtuel. Dans les deux cas, les interactions peuvent être décomposées en deux tâches : la sélection et la manipulation.

Sélection. La tâche de sélection est essentielle dont toutes les interactions dépendent. En effet, avant toute manipulation, il faut que les utilisateurs désignent les objets virtuels avec lesquels ils veulent interagir. Pour cela, il existe différentes techniques qui sont plus ou moins bien adaptées à l'application et aux dispositifs immersifs utilisés :

- **Main virtuelle :** la technique de sélection la plus naturelle consiste à représenter la main d'un utilisateur par une main humaine dans le monde virtuel [Jacoby *et al.*, 1994]. Cette main permet de sélectionner les objets par le simple fait de les toucher. La représentation virtuelle de la main peut être co-localisée avec la main réelle de l'utilisateur afin qu'il ait l'impression que cette main virtuelle est sa vraie main. En plus des changements de position, le mouvement des doigts peut être reproduit sur la main virtuelle si l'utilisateur porte un dispositif permettant de repérer la position de ses doigts comme un gant de données. Cette technique est particulièrement bien adaptée au *HMD* car l'utilisateur ne voit pas ses mains réelles avec un tel dispositif. Cependant, cette technique a un champ d'action limité qui correspond à la longueur de bras de l'utilisateur : il ne peut sélectionner que les objets virtuels qui sont à portée de sa main et il est obligé de se déplacer pour sélectionner des objets distants.
- **Métaphore du « Go-Go » :** cette technique proposée par [Poupyrev *et al.*, 1996] est une extension de la main virtuelle. Elle permet à un utilisateur d'accéder à des objets éloignés sans avoir à se déplacer dans le monde virtuel. Lorsque la main réelle de l'utilisateur dépasse un seuil d'éloignement par rapport à son corps, la relation entre le déplacement de la main virtuelle et celui de la main réelle n'est plus linéaire, mais exponentielle. L'utilisateur peut ainsi sélectionner précisément les objets virtuels situés près de lui, mais il peut aussi aller sélectionner des objets lointains grâce à des déplacements plus importants de sa main virtuelle. Par contre, la co-localisation de la main de l'utilisateur n'est pas conservée avec cette technique.
- **Curseur 3D :** de la même façon que le curseur de la souris permet de désigner un point précis dans l'espace 2D du bureau de l'ordinateur, le curseur 3D [Zhai *et al.*, 1994] permet de désigner un point précis de l'espace 3D de l'environnement virtuel. Il peut être déplacé dans tout le monde virtuel afin de sélectionner les objets qui s'y trouvent. Ce curseur 3D peut être contrôlé de deux façons différentes :
  - en position grâce, par exemple, à un bras à retour d'effort ou un objet repéré par un système de *tracking* (comme un *flystick*). Le curseur 3D peut être co-localisé avec l'objet réel qui sert à le contrôler. Cependant, la relation entre l'objet réel et le curseur 3D peut aussi être « débrayée » afin de pouvoir replacer l'objet réel et d'emmener le curseur 3D plus loin que le champ d'action du dispositif : la co-localisation n'est alors plus assurée.

  - en vitesse grâce, par exemple, à un *joystick* ou une souris 3D *(SpaceMouse)*. Bien entendu, dans ce cas, la co-localisation n'est pas possible.

- **Rayon virtuel :** cette technique utilise un rayon qui va être projeté dans le monde virtuel [Mine, 1995a]. Ce rayon permet de sélectionner les objets virtuels qu'il intersecte. Il permet ainsi à un utilisateur de sélectionner des objets distants simplement en les pointant. Cet utilisateur spécifie l'origine et la direction du rayon virtuel grâce à sa main ou à un objet réel. La représentation du rayon virtuel est souvent co-localisée avec la main ou l'objet réel qui sert à matérialiser son origine. Cette technique est donc particulièrement bien appropriée aux dispositifs offrant une immersion visuelle importante (de type CAVE® ou salle immersive) car l'utilisateur voit vraiment le rayon virtuel partir de sa main ou de l'objet réel.
- **Technique sur « image plan » :** cette technique permet à un utilisateur de sélectionner des objets virtuels en interagissant avec une « image plan », c'est-à-dire une projection 2D de la scène virtuelle 3D [Pierce *et al.*, 1997]. Pour cela, l'utilisateur doit pointer l'objet à sélectionner avec ses doigts ou un objet dédié : la sélection se fait sur le premier objet qui intersecte la ligne qui part des yeux de l'utilisateur et qui passe par ses doigts ou l'objet dédié. Il existe de nombreuses solutions pour pointer l'objet à sélectionner sur l'« image plan » : l'utilisateur place l'objet entre son pouce et son index, pose son index sur l'objet, place sa paume sous l'objet, encadre l'objet avec ses deux mains, occulte l'objet avec un objet spécifique, etc. Pour mettre en œuvre cette technique, la tête de l'utilisateur et l'objet dédié à la sélection doivent être repérés pour déterminer la ligne virtuelle qui pointe vers l'objet à sélectionner.

Manipulation. La manipulation est la tâche qui consiste à spécifier ou modifier les propriétés des objets du monde virtuel (leur position, leur orientation, leur forme, leur couleur, etc.). Cette tâche est intimement liée à la sélection car il n'est pas possible de manipuler un objet virtuel sans l'avoir sélectionné au préalable. Les techniques de sélection vues dans la partie précédente permettent également de réaliser certaines tâches simples de manipulation telles que des translations ou des rotations. Pour cela, une commande permet aux utilisateurs de spécifier qu'ils veulent sélectionner l'objet virtuel désigné et passer en mode manipulation. Généralement, l'objet virtuel se retrouve alors « accroché » à l'outil d'interaction et subit alors les même déplacements que cet outil. Par exemple, lorsqu'un utilisateur utilise une main virtuelle, il peut fermer sa main pour sélectionner un objet virtuel. Il passe alors en mode manipulation et il peut déplacer l'objet qui est en contact avec la main virtuelle simplement en bougeant sa main.

Cependant, les techniques utilisées pour la sélection ne sont pas toutes bien adaptées à la manipulation et elles ne permettent pas toujours de réaliser les modifications souhaitées. Par exemple, il est difficile d'utiliser le rayon virtuel pour effectuer des rotations autres que celles autour de l'axe du rayon. De plus, lorsque l'objet virtuel sélectionné se situe loin des utilisateurs, il est difficile pour eux d'être précis dans leurs manipulations. Pour remédier à ces limitations, certaines techniques utilisent une combinaison de plusieurs métaphores. La méthode HOMER proposée par [Bowman et Hodges, 1997] utilise un rayon virtuel pour sélectionner les objets virtuels, puis une main virtuelle attachée à l'objet sélectionné pour le manipuler plus facilement. D'autres techniques proposent, en plus, de rapprocher l'objet virtuel des utilisateurs une fois qu'il a été sélectionné. Cela permet aux utilisateurs de manipuler plus précisément l'objet et de mieux voir ce qu'ils font. Lorsque la manipulation

est terminée, l'objet virtuel est replacé à sa place initiale dans le monde virtuel. Cependant, cette solution n'est pas envisageable pour des interactions collaboratives car il n'est pas toujours possible de rapprocher l'objet virtuel de deux utilisateurs en même temps.

Pour des manipulations plus complexes, il peut être nécessaire de réaliser différentes actions distinctes sur un même objet virtuel. De plus, ces différentes actions peuvent mettre en œuvre des techniques d'interactions très différentes. Il est donc nécessaire que les utilisateurs précisent l'action qu'ils souhaitent appliquer à l'objet sélectionné. Pour effectuer ce choix, nous distinguons deux fonctionnements différents :

- Le choix de l'outil d'interaction détermine l'action qui peut être réalisée par un utilisateur et donc les objets avec lesquels il peut interagir (c'est-à-dire les objets qui possèdent les bonnes propriétés correspondant à cette action). Les utilisateurs peuvent choisir l'outil d'interaction simplement en appuyant sur un bouton de leur périphérique d'interaction, en prenant un objet virtuel représentant cet outil dans le monde virtuel, ou en le sélectionnant dans un menu. Par exemple, dans le modeleur graphique 3DM [Butterworth *et al.*, 1992], l'utilisateur choisit l'outil d'interaction au travers d'un menu 3D qui est présent directement dans le monde virtuel.
- Le choix de l'objet virtuel détermine les actions qui peuvent lui être appliquées par l'utilisateur qui l'a sélectionné : les « Smart Objects » [Kallmann et Thalmann, 1999] indiquent eux-mêmes les actions que les utilisateurs peuvent leur appliquer.

1.1.3 Intégration des utilisateurs dans l'environnement virtuel

Dans cette partie, nous présentons d'un point de vue plus technique les différentes propositions pour intégrer les utilisateurs au sein de la boucle perception/action en tenant compte de leur environnement réel et des dispositifs matériels qu'ils utilisent. Les solutions pour intégrer les utilisateurs et leur environnement réel peuvent être classées en trois catégories avec des objectifs graduels. Dans la partie 1.1.3.1, nous détaillons les solutions qui permettent d'abstraire les dispositifs matériels lors de la conception d'un système de réalité virtuelle afin de faciliter le déploiement d'une application sur des dispositifs matériels variés. Ces solutions permettent de mettre en relation de manière générique les retours sensoriels et les actions des utilisateurs avec les dispositifs qu'ils utilisent. Dans la partie 1.1.3.2, nous présentons les systèmes de réalité virtuelle qui intègrent les dispositifs matériels des utilisateurs au sein de l'environnement virtuel. Cela permet d'assurer une co-localisation de ces dispositifs en adaptant leurs retours sensoriels ou leurs actions en fonction de leur position dans le monde virtuel. Enfin, dans la partie 1.1.3.3, nous verrons que certaines applications nécessitent d'intégrer en plus dans l'environnement virtuel l'espace d'interaction réel associé à un dispositif matériel. Cela permet de faire percevoir aux utilisateurs les limites de leurs capacités de perception et d'interaction dans l'environnement virtuel, ainsi que d'adapter les techniques d'interaction en fonction de ces limites.

1.1.3.1 Abstraction des dispositifs matériels

De nombreux systèmes ont été proposés pour permettre le développement d'applications de réalité virtuelle. Même si les premiers d'entre eux étaient très spécifiques à des caractéristiques techniques données (dispositifs d'entrée/sortie, système d'exploitation, etc.), la plupart des systèmes cherchent à s'abstraire au maximum des dispositifs matériels utilisés. Cela permet d'utiliser une même application de réalité virtuelle avec plusieurs types de dis-

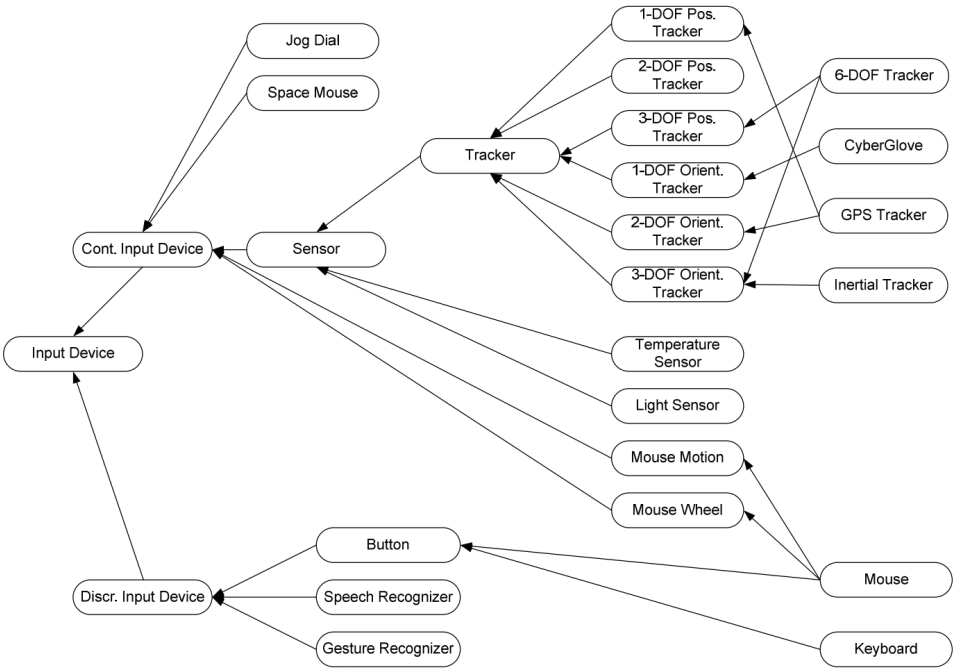


FIGURE 1.15 – Hiérarchie d'interfaces pour les dispositifs d'entrée [Ohlenburg *et al.*, 2007].

positifs matériels ou avec des configurations différentes. Comme présenté dans [Aguerreche, 2010], il existe différentes manières de mettre en œuvre cette abstraction.

Afin de séparer le code principal d'une application de réalité virtuelle et celui associé aux dispositifs matériels utilisés, MR Toolkit [Shaw *et al.*, 1993] propose de répartir les différents processus de l'application en trois couches. La première est la couche matérielle : chaque processus de cette couche correspond à un dispositif matériel et il communique avec la seconde couche par l'intermédiaire de *sockets*. Une seconde couche fournit une interface haut-niveau pour la couche matérielle et fait le lien avec la troisième couche. Enfin, la troisième couche correspond à l'application principale.

GNU/MAVERIK [Hubbold *et al.*, 1999] utilise un système de *callbacks*. Chaque dispositif matériel doit implémenter ces *callbacks* pour faire remonter ses informations (positions 3D, actions, etc.) au reste de l'application. Les développeurs peuvent ainsi concevoir leur application en se basant sur ces *callbacks* sans se soucier des dispositifs matériels qui seront utilisés. Il faudra néanmoins que les dispositifs matériels soient capables de fournir toutes les informations requises par l'application.

VR-Juggler [Bierbaum *et al.*, 2001] se base sur un mécanisme d'héritage : les périphériques d'interaction sont regroupés en plusieurs catégories (périphériques de positions, périphériques analogiques, etc.) ayant chacune une interface associée. Intégrer un nouveau dispositif matériel consiste à implémenter les interfaces qui lui correspondent. L'application peut ainsi être développée en se basant seulement sur les interfaces des différents types de périphériques qui sont nécessaires dans cette application. D'une façon similaire, DEVAL *(DEVice Abstraction Layer for VR/AR)* [Ohlenburg *et al.*, 2007] définit une couche générique d'abstraction des dispositifs matériels qui est structurée en une hiérarchie d'interfaces associées aux différents types de dispositifs matériels (cf. figure 1.15). Cette couche d'abstraction peut ainsi être facilement étendue à d'autres types de dispositifs.

Dans les méthodes présentées précédemment, l'abstraction des dispositifs matériels est réalisée au niveau logiciel à l'intérieur du système. Cependant, une autre solution consiste à déplacer la couche d'abstraction en dehors du système. VRPN [Taylor *et al.*, 2001] est constitué d'un ensemble de serveurs pour chacun des différents dispositifs matériels. Un serveur VRPN est généralement exécuté sur une machine séparée qui est reliée directement à un dispositif matériel. Le serveur envoie par le réseau les données provenant de ce dispositif à la machine qui exécute l'application principale. L'application principale a donc juste à traiter les actions qu'elle reçoit des serveurs. Ces actions arrivent sous une forme abstraite (indépendante du dispositif matériel comme des positions 3D, des évènements, etc.) ce qui permet à l'application de s'adapter à n'importe quel dispositif matériel pour lequel il existe un serveur VRPN. Cette solution a connu un succès important et intègre des serveurs pour un grand nombre de dispositifs matériels de réalité virtuelle.

1.1.3.2 Intégration des dispositifs matériels

Les solutions précédentes permettant de s'abstraire des dispositifs matériels considèrent ces dispositifs seulement comme des services qui envoient des données en entrée ou reçoivent des données en sortie. Cependant, il peut être nécessaire de considérer chacun de ces dispositifs comme étant liés à une entité avec une position dans le monde virtuel. Certains systèmes proposent de modéliser les dispositifs matériels dans l'environnement virtuel afin d'adapter les retours qu'ils transmettent aux utilisateurs ou les actions qu'ils permettent d'effectuer sur le monde virtuel en fonction de leur position dans le monde virtuel.

Par exemple, pour les dispositifs de visualisation, il est possible de placer une camera virtuelle qui définit les images affichées sur le dispositif de visualisation qui lui est associé. [Robinett et Holloway, 1992] proposent une hiérarchie de systèmes de coordonnées pour modéliser un système de réalité virtuelle utilisant un visio-casque *(HMD)*. Cette hiérarchie de repères permet de modifier la position, l'orientation et l'échelle d'un utilisateur dans le monde virtuel, tout en maintenant la co-localisation entre la position réelle de la tête de cet utilisateur et celle de la camera virtuelle. Dans un environnement virtuel collaboratif, [Zhang et Furnas, 2005] proposent une approche similaire en utilisant la particularité du graphe de scène de Java3D [Sowizral et Deering, 1999]. Le concept de la `ViewPlatform` de Java3D définit un système de coordonnées particulier pour décrire le dispositif de visualisation de chaque utilisateur. Cette `ViewPlatform` permet donc à chaque utilisateur d'utiliser la même application en l'adaptant à son propre système de visualisation avec l'idée de *« write once, view everywhere »*.

De façon plus générale, Dive [Hagsand, 1996][Frécon et Stenius, 1998] et ensuite Diverse [Kelso *et al.*, 2002] permettent de créer un environnement virtuel extensible et reconfigurable, indépendamment des dispositifs matériels utilisés. Ils utilisent une structure de données particulière pour décrire la scène du monde virtuel à la manière d'un graphe de scène. Cette structure de données est basée sur un système de modules qui peuvent être liés à différents dispositifs matériels. Par exemple, [Steed, 2008] propose d'utiliser les abstractions d'un véhicule, de l'utilisateur et du corps de l'utilisateur pour adapter la navigation aux dispositifs matériels qui peuvent être utilisés. Ainsi, ces abstractions sont « pilotées » en fonction des périphériques d'interaction dont les utilisateurs disposent :

- Si l'utilisateur dispose uniquement d'un *joystick*, il contrôlera seulement le véhicule.
- Si l'utilisateur dispose en plus d'un système de *tracking* qui permet de le repérer dans son dispositif immersif, il pourra en plus déplacer son abstraction dans le véhicule.

– Si ce système de *tracking* permet de repérer de manière indépendante chacun des membres de l'utilisateur, il pourra en plus animer l'abstraction de son corps à l'intérieur du véhicule.

D'une façon similaire, *Simple Virtual Environment (SVE)* [Kessler *et al.*, 2000] permet de concevoir des applications de réalité virtuelle dont la configuration des dispositifs matériels sera spécifiée uniquement lors de l'exécution de l'application. Pour cela, il propose de clairement séparer le modèle décrivant l'environnement virtuel et les dispositifs matériels lors de la conception de l'environnement virtuel. Le modèle de l'environnement virtuel inclut une description de la scène 3D, ainsi qu'une représentation virtuelle de l'utilisateur. Cette représentation virtuelle sera ensuite contrôlée par les dispositifs d'entrée et définira les retours envoyés à l'utilisateur au travers des dispositifs de sortie.

Enfin, la *Cabine Virtuelle d'Immersion* [Duval et Chauffaut, 2006] propose d'utiliser un ensemble de systèmes de coordonnées pour coupler la navigation et les interactions dans les environnements virtuels collaboratifs multi-échelle. Grâce à cette *Cabine Virtuelle d'Immersion*, les outils d'interaction sont repérés dans un système de coordonnées propre à chaque utilisateur. Cela permet aux utilisateurs de naviguer (y compris en changeant d'échelle) dans le monde virtuel en transportant avec eux leurs outils d'interaction.

1.1.3.3 Intégration des espaces d'interaction réels

Les solutions précédentes proposent d'intégrer certains dispositifs matériels dans l'environnement virtuel, mais elles ne tiennent pas compte des espaces d'interaction 3D qui leur sont associés. Cependant, certaines applications nécessitent d'intégrer ces espaces réels dans l'environnement virtuel afin de prendre en compte les possibilités de perception et d'interaction que les dispositifs offrent aux utilisateurs. Cette prise en compte permet, d'une part, d'adapter les techniques d'interaction en fonction des possibilités de perception et d'interaction des utilisateurs et, d'autre part, de leur faire percevoir ces possibilités.

Espace de déplacement physique Le modeleur graphique 3DM [Butterworth *et al.*, 1992] propose de matérialiser la zone couverte par le système de *tracking* par un tapis volant virtuel *(Magic Carpet)* représenté par un cercle rouge sur le sol du monde virtuel (cf. figure 1.16). L'utilisateur qui utilise un visiocasque *(HMD)* peut se déplacer physiquement

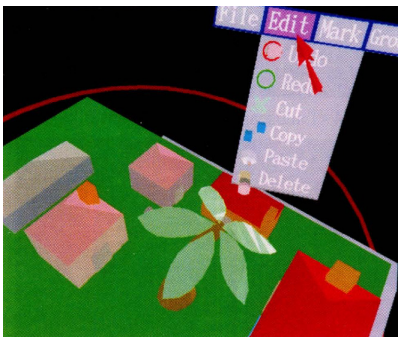


FIGURE 1.16 – *Magic Carpet* représentant la zone du système de *tracking* dans 3DM [Butterworth *et al.*, 1992].

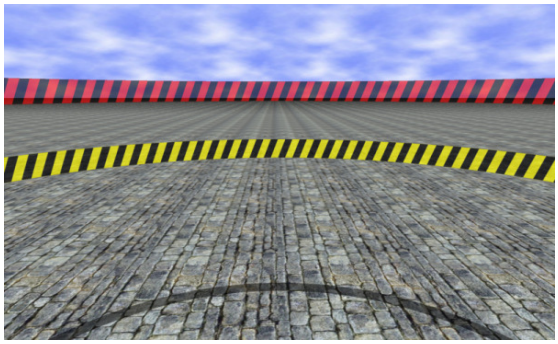


FIGURE 1.17 – *Magic Barrier Tape* représentant les limites physiques de la salle immersive [Cirio *et al.*, 2009].

sur ce tapis volant : ses déplacements réels sont mis en relation avec ses déplacements sur le tapis volant afin de lui permettre d'effectuer des tâches d'observation ou d'interaction de manière intuitive. Pour la navigation sur de grandes distances, l'utilisateur peut également déplacer le tapis volant dans le monde virtuel grâce à un outil spécifique. Le menu 3D pour accéder aux différents outils d'interaction peut être posé sur le tapis volant : l'utilisateur se déplace ainsi dans le monde virtuel en transportant ses outils.

De façon similaire, la *Magic Barrier Tape* [Cirio *et al.*, 2009] permet de représenter les limites de l'espace dans lequel les utilisateurs peuvent se déplacer physiquement par un ruban virtuel de signalisation (cf. figure 1.17). Cela permet de faire prendre conscience aux utilisateurs des limites de déplacement que leur impose le dispositif de visualisation. Lorsqu'un utilisateur se trouve trop proche de la frontière de l'espace de déplacement, un second ruban rouge apparait afin de lui signaler qu'il ne doit pas s'approcher plus de cette frontière. Pour naviguer sur des distances plus importantes sans dégrader la sensation de présence dans l'environnement virtuel, les utilisateurs ont simplement à « pousser » le ruban virtuel de signalisation vers l'endroit où ils souhaitent aller.

D'autres techniques utilisées pour permettre aux utilisateurs de marcher de façon naturelle dans les environnements virtuels doivent aussi intégrer l'espace de déplacement physique, même si elles ne le représentent pas dans le monde virtuel. Elles doivent tenir compte de la position des utilisateurs dans l'espace de déplacement physique afin d'adapter la technique de navigation pour que les utilisateurs restent dans l'espace restreint imposé par le dispositif matériel. Nous pouvons citer, par exemple, les techniques de redirection de la marche [Razzaque, 2005][Cirio *et al.*, 2012], les techniques de replacement [Williams *et al.*, 2007], ou les techniques superposant des espaces virtuels [Suma *et al.*, 2012].

Espace d'interaction haptique La technique de la *Bubble* [Dominjon *et al.*, 2005] propose de représenter l'espace d'interaction d'un dispositif à retour d'effort par une sphère semi-transparente autour du curseur que l'utilisateur manipule dans l'environnement virtuel (cf. figure 1.18). Cette *Bubble* permet non seulement aux utilisateurs de percevoir le champ d'action du dispositif, mais elle permet aussi d'adapter la technique d'interaction de ce dispositif. En effet, lorsque le curseur est à l'intérieur de la *Bubble*, il est contrôlé en position et permet ainsi d'agir avec les objets virtuels. Par contre, lorsqu'il est en dehors de la *Bubble*, il est contrôlé en vitesse et permet de déplacer la *Bubble* (cf. figure 1.19).

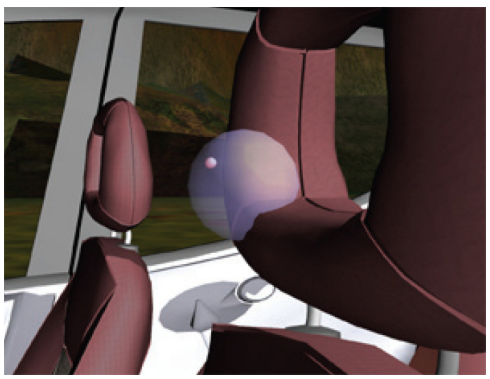


FIGURE 1.18 – *Bubble* représentant l'espace d'interaction du dispositif à retour d'effort [Dominjon *et al.*, 2005].

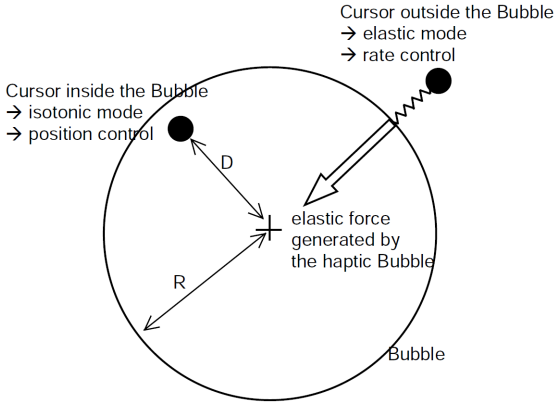


FIGURE 1.19 – Schéma de la *Bubble* avec les deux modes d'interaction [Dominjon *et al.*, 2005].

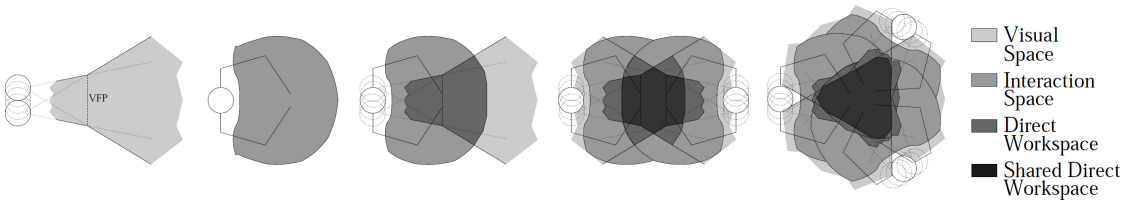


FIGURE 1.20 – Première description d'espaces d'interaction [Mulder et Boschker, 2004].

Espace de visualisation Le *Hand Held Display (HHD)* [Amselem, 1995] est un écran LCD dont la position est déterminée par un capteur de localisation. Le système repère également la position de la tête de l'utilisateur et peut ainsi modéliser l'espace de visualisation associé à cet écran qui est défini par la pyramide de vue qui part de la tête et passe par l'écran. Cette co-localisation de l'espace de visualisation permet d'adapter les images virtuelles affichées sur l'écran en fonction des positions de la tête de l'utilisateur et de l'écran dans le monde réel. L'écran peut être vu comme une fenêtre sur le monde virtuel que l'utilisateur peut déplacer.

Cependant, les concepteurs des applications présentées précédemment ne décrivent pas de façon explicite l'existence de tels espaces d'interaction. Les premiers à décrire des espaces d'interaction multi-sensoriels sont [Mulder et Boschker, 2004]. Ils proposent d'associer à un dispositif immersif (basé sur un miroir) un espace de visualisation : *visual space* et un espace de manipulation : *interaction space* (les utilisateurs peuvent passer les mains sous le miroir pour manipuler « directement » les objets virtuels). Ils définissent le *direct workspace* comme l'intersection de ces deux espaces. Plusieurs de ces dispositifs peuvent être juxtaposés afin de permettre à plusieurs utilisateurs de collaborer dans un *shared direct workspace* qui est l'intersection des *direct workspaces* de chaque utilisateur (cf. figure 1.20). Néanmoins, cette description n'est pas un modèle générique qui s'adapte à d'autres dispositifs immersifs, ni à d'autres techniques d'interaction. Par exemple, elle ne permet pas aux utilisateurs de naviguer indépendamment des autres utilisateurs car la relation spatiale entre les *direct workspaces* doit être maintenue. De plus, elle ne propose pas de solution pour faire percevoir ces espaces d'interaction aux utilisateurs.

1.2 Collaboration

La collaboration en réalité virtuelle consiste à intégrer plusieurs utilisateurs dans la boucle perception/action d'un même environnement virtuel comme le montre la figure 1.21. De cette façon, les actions de chaque utilisateur sur le monde virtuel modifient la perception que tous les utilisateurs ont de ce monde. Les utilisateurs peuvent ainsi avoir des interactions sociales au travers du monde virtuel et réaliser ensemble différentes tâches. La collaboration en environnement virtuel a de nombreuses applications comme, par exemple, la co-expertise de données scientifique, la formation à distance, l'apprentissage de manipulation collaborative, etc. Les utilisateurs qui collaborent dans un environnement virtuel peuvent être présents dans la même salle de réalité virtuelle et utiliser le même dispositif matériel, ou se trouver à distance sur des lieux géographiquement éloignés et utiliser des dispositifs matériels différents. Nous parlons alors de collaboration locale et de collaboration distante. Dans le cadre de cette thèse, nous nous intéressons principalement à la collaboration distante même si une grande partie de nos travaux peut être utilisée également pour la collaboration locale.

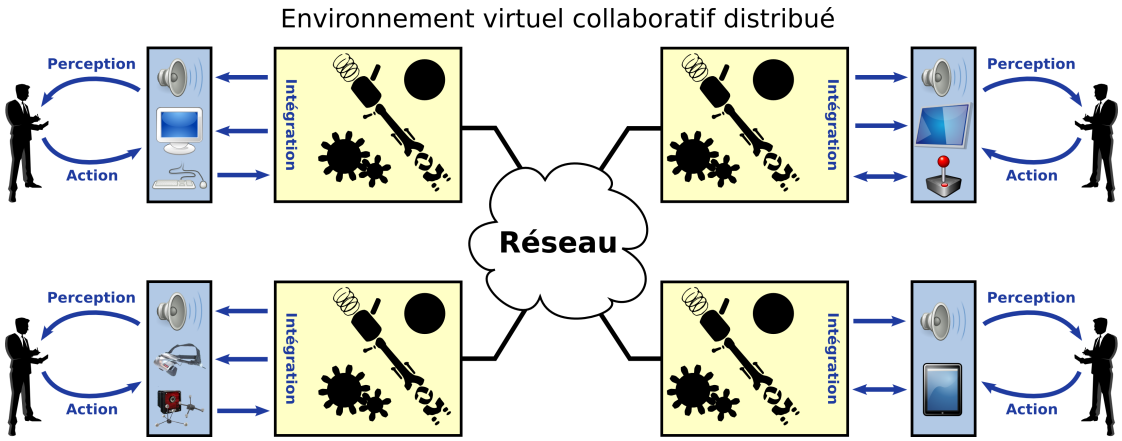


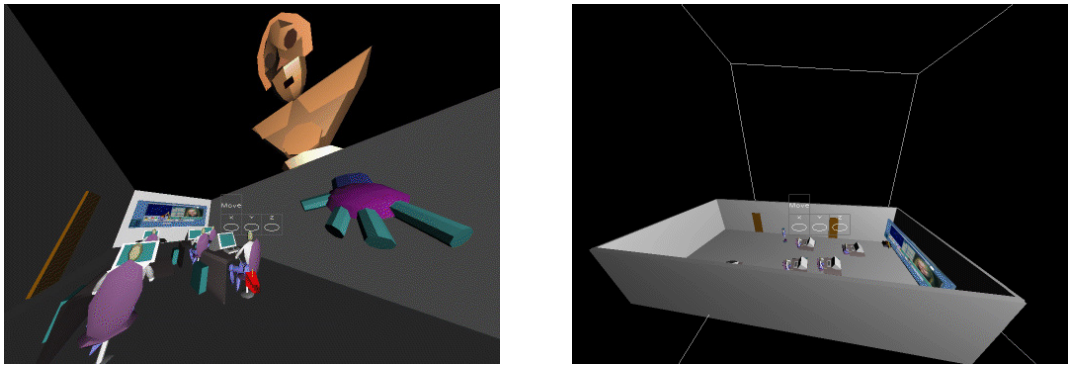
FIGURE 1.21 – Collaboration de plusieurs utilisateurs dans un même environnement virtuel.

Pour permettre aux utilisateurs de collaborer, il faut leur fournir des outils pour faciliter leur compréhension mutuelle et pour leur permettre d'interagir ensemble. Dans la partie 1.2.1, nous présentons les solutions proposées dans la littérature pour améliorer la perception entre les utilisateurs. Dans la partie 1.2.2, nous étudions les moyens permettant aux utilisateurs de communiquer entre eux. Dans la partie 1.2.3, nous détaillons les techniques utilisées pour permettre aux utilisateurs de se déplacer avec les autres lorsqu'ils veulent observer ou interagir ensemble. Enfin, dans la partie 1.2.4, nous présentons les techniques pour manipuler les objets virtuels à plusieurs.

1.2.1 Perception des autres utilisateurs

Le facteur préalable à toute collaboration est que chaque utilisateur soit capable de percevoir les autres dans l'environnement virtuel. Une mauvaise perception des autres peut entrainer des incompréhensions entre utilisateurs. En conséquence, les utilisateurs doivent pouvoir percevoir où se trouvent les autres, mais aussi ce qu'ils voient et ce qu'ils font. Il peut également être intéressant d'ajouter des informations supplémentaires sur leurs capacités de perception et d'interaction : ce qu'ils peuvent voir en fonction du dispositif de visualisation qu'ils utilisent, les objets virtuels avec lesquels ils peuvent interagir, etc.

Cette perception des autres utilisateurs peut se faire sur différents canaux sensoriels, dont principalement la vue, l'ouïe et le toucher. Au niveau visuel, la technique la plus utilisée est de représenter chaque utilisateur par un avatar dans le monde virtuel qui peut être plus ou moins réaliste comme le présentent [Benford *et al.*, 1995]. Il est également possible d'utiliser un monde en miniature (comme le *WIM* de [Stoakley *et al.*, 1995]) pour représenter les utilisateurs sur cette sorte de « carte » du monde virtuel et ainsi permettre à chaque utilisateur de facilement localiser les autres. D'une façon un peu similaire, CALVIN [Leigh *et al.*, 1996] propose aux utilisateurs de prendre une vue intérieure *(mortal's view)* ou extérieure *(deity's view)* du monde virtuel (cf. figure 1.22). Les utilisateurs avec la vue extérieure peuvent facilement voir où se trouvent les autres et ce qu'ils font. Néanmoins, dans un environnement virtuel multi-échelle, lorsque les utilisateurs n'ont pas la même échelle, ils n'ont pas forcément la même perception du monde virtuel. Cela peut être une source d'incompréhension entre les utilisateurs comme l'expliquent [Zhang et Furnas, 2002]. Par exemple, un utilisateur qui est grand peut avoir l'impression d'être loin



(a) (b)

FIGURE 1.22 – (a) Vue intérieure et (b) vue extérieure du monde virtuel [Leigh *et al.*, 1996].

d'un utilisateur qui est petit, et inversement. Au niveau sonore, il peut être intéressant de faire percevoir aux utilisateurs de façon spatialisée la parole, mais aussi les bruits des actions provenant des autres. Ces retours sonores sont essentiels à la communication, mais fournissent aussi des informations sur la position et les actions des utilisateurs. Au niveau haptique, lorsque plusieurs utilisateurs interagissent sur un même objet virtuel avec des dispositifs haptiques, ils peuvent percevoir les forces que les autres appliquent à l'objet. Ces informations kinesthésiques permettent de ressentir la présence et les actions des autres.

Les informations supplémentaires sur les capacités de perception et l'interaction des utilisateurs dépendent principalement des dispositifs matériels qu'ils utilisent. Pour faire percevoir ces informations aux utilisateurs, il est donc nécessaire de tenir compte des espaces d'interaction réels associés à ces dispositifs matériels comme pour les applications présentées dans la partie 1.1.3.3. Par exemple, [Fraser *et al.*, 1999] proposent de matérialiser la pyramide de vue des utilisateurs sous forme d'une représentation en « fil de fer » (cf. figure 1.23). Cette représentation doit être adaptée en fonction du dispositif immersif et de l'espace de visualisation qui lui est associé. Pour faire percevoir qu'un utilisateur interagit avec un objet virtuel, ils proposent de changer l'aspect de l'objet afin de montrer qu'il est manipulé et d'étirer le bras de l'utilisateur en direction de l'objet afin de montrer aux autres quel est l'utilisateur qui le manipule. Par extension, le *Spatial model of interaction* proposé par [Benford *et al.*, 1994] est une approche intéressante pour décrire

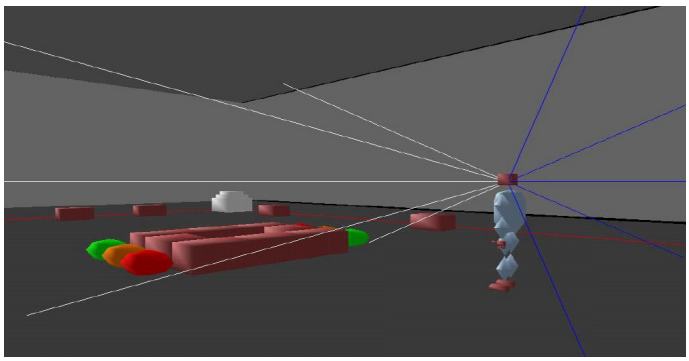


FIGURE 1.23 – Représentation de l'espace de visualisation de l'utilisateur [Fraser *et al.*, 1999].

la perception multi-sensorielle des utilisateurs. Ce modèle spatial définit un *focus* et un *nimbus* pour chaque utilisateur présent dans l'environnement virtuel. Le *focus* correspond à la zone dans laquelle l'utilisateur a une perception sensorielle des autres ou des objets virtuels. Le *nimbus* correspond à la zone dans laquelle les autres ont une perception sensorielle de cet utilisateur. Ces zones ne sont pas forcément identiques pour le *focus* et le *nimbus* et elles peuvent avoir des formes ou des tailles différentes en fonction du retour sensoriel auquel elles correspondent (visuel, sonore, haptique, etc.). Même si le *focus* et le *nimbus* ne représentent pas l'environnement réel des utilisateurs, ils dépendent aussi des caractéristiques des dispositifs matériels utilisés.

1.2.2 Communication entre utilisateurs

Comme dans le monde réel, la communication entre les utilisateurs est essentielle pour permettre une collaboration efficace. La parole est le moyen de communication le plus simple à mettre en œuvre et le plus naturel pour les utilisateurs. En conséquence, de nombreuses plateformes de réalité virtuelle intègrent la voix des utilisateurs. Si la plateforme en elle-même n'intègre pas la voix, il est toujours possible d'utiliser un logiciel de voix sur IP pour remédier à ce manque. Comme le précisent [Hindmarsh *et al.*, 1998], la communication orale apporte aux utilisateurs beaucoup d'informations complémentaires pour la collaboration et permet éventuellement de compenser une mauvaise perception de l'environnement virtuel ou des autres utilisateurs. Par exemple, elle permet aux utilisateurs de compléter leur expertise du monde virtuel lorsqu'ils ont des points de vue différents ou de se coordonner lorsqu'ils réalisent une co-manipulation. Même si la communication orale est souvent pertinente lorsque deux utilisateurs sont en train de réaliser une tâche ensemble, elle peut entrainer des discontinuités lors des interactions comme l'expliquent [Bowers *et al.*, 1996]. De plus, ce mode de communication est difficile à utiliser lorsque les utilisateurs sont nombreux ou lorsque l'environnement est bruyant.

Des outils d'interaction ou des outils dédiés peuvent aussi être utilisés pour la communication visuelle entre utilisateurs. Par exemple, un rayon virtuel peut être utilisé par un utilisateur pour montrer aux autres des détails intéressants du monde virtuel comme il le ferait avec un pointeur laser dans le monde réel. Certaines applications permettent en plus aux utilisateurs de laisser des annotations dans le monde virtuel sous diverses formes (texte, son, vidéo, etc). Ces annotations sont particulièrement pertinentes pour l'étude de données scientifiques comme le présentent [Schild *et al.*, 2009]. De plus, ce mode de communication permet une collaboration différée dans le temps car un utilisateur peut laisser des annotations à un moment donné et un autre utilisateur peut venir les voir plus tard.

1.2.3 Navigation collaborative

Comme l'expliquent [Snowdon *et al.*, 1995], les environnements virtuels collaboratifs sont du type *WYSINWIS (What Your See Is Not What I see)* dans le sens où les utilisateurs peuvent prendre des points de vue différents sur le monde virtuel afin de compléter leur perception mutuelle. Néanmoins, il est important pour cela que ces utilisateurs perçoivent en même temps le même état de l'environnement virtuel. Pour aider à la navigation, un utilisateur peut prendre un point de vue extérieur au monde virtuel comme dans CALVIN [Leigh *et al.*, 1996] afin de diriger les autres utilisateurs qui naviguent ou même de les placer directement comme un objet virtuel pour leur offrir un point de vue intéressant.

Même si les utilisateurs peuvent naviguer indépendamment, il peut être intéressant de leur permettre de synchroniser leur déplacement afin qu'ils puissent se suivre dans le monde virtuel ou se placer au même endroit pour avoir le même point de vue (si un utilisateur veut montrer quelque chose à un autre). [Duval *et al.*, 2008] proposent de permettre aux utilisateurs de définir un ensemble de points de vue intéressants que les autres pourront ensuite parcourir pour explorer des données scientifiques. Par ailleurs, [Yang et Olson, 2002] proposent trois modes pour permettre à deux utilisateurs de naviguer ensemble :

- les deux utilisateurs ont la même vue mais un seul dirige le déplacement,
- l'un des utilisateurs suit avec un *offset* celui qui dirige le déplacement,
- l'un des utilisateurs à une vue extérieure au monde virtuel et seul l'autre se déplace.

Enfin, [Dodds et Ruddle, 2008] proposent une navigation en groupe où les utilisateurs peuvent se déplacer indépendamment, mais où le système offre des fonctionnalités pour les aider à rester en groupe. Il leur permet, par exemple, de suivre automatiquement un membre du groupe ou d'être placé au centre du groupe (moyenne des positions des membres). Lors de ces déplacements automatiques, ils continuent à contrôler indépendamment leur orientation afin de pouvoir observer le monde virtuel comme ils le souhaitent.

1.2.4 Co-manipulation

Contrairement aux techniques d'interaction présentées dans la partie 1.1.2.2, la co-manipulation consiste à manipuler un objet virtuel à plusieurs utilisateurs en même temps. Interagir à plusieurs avec le même objet virtuel permet soit de réaliser des tâches compliquées, soit de reproduire la manière réelle de manipuler cet objet. Pour mettre en œuvre une co-manipulation, il faut être capable de combiner les actions provenant de plusieurs utilisateurs en même temps. [Aguerreche, 2010] propose de classer les solutions existantes en deux catégories : la séparation des degrés de liberté (partie 1.2.4.1) et les accès concurrents à un même degré de liberté (partie 1.2.4.2).

1.2.4.1 Séparation des degrés de liberté

La séparation des degrés de liberté permet à plusieurs utilisateurs d'accéder indépendamment à des degrés de liberté différents. Cela permet d'éviter les modification concurrentes sur les mêmes paramètres d'un objet. Classiquement, six degrés de liberté sont associés à un objet virtuel : trois pour les translations et trois pour les rotations. Cependant, il est possible de modifier de cette façon de nombreux autres paramètres de l'objet virtuel comme sa taille, sa couleur, etc. [Pinho *et al.*, 2002] proposent deux méthodes pour réaliser la séparation des degrés de liberté :

- **Les techniques coopératives homogènes :** les utilisateurs utilisent chacun un même outil d'interaction mono-utilisateur présenté dans la partie 1.1.2.2.
- **Les techniques coopératives hétérogènes :** les utilisateurs utilisent des outils d'interaction différents (par exemple : un rayon virtuel pour qu'un utilisateur applique les translations et une main virtuelle pour qu'un autre applique les rotations).

[Pinho *et al.*, 2002] concluent que la technique de séparation des degrés de liberté permet de faire plus facilement et plus rapidement des manipulations difficiles, mais aussi des ajustements plus précis que lorqu'un utilisateur réalise seul la manipulation.

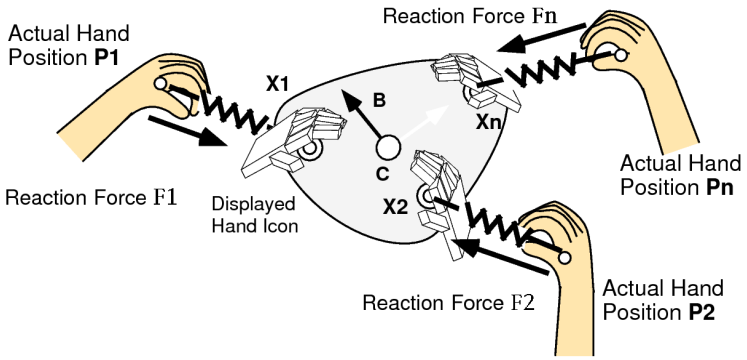
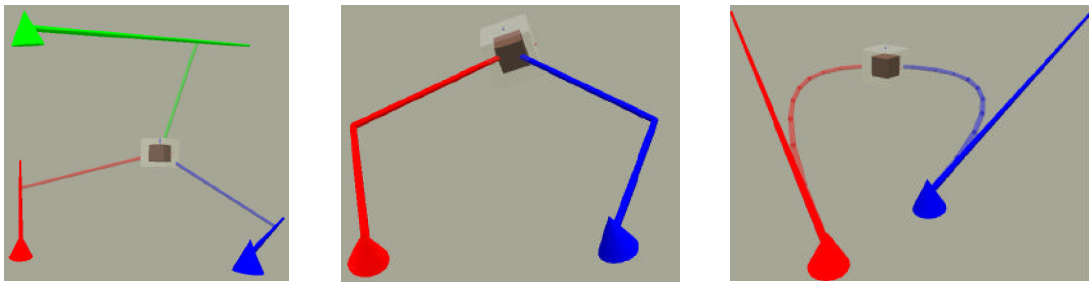


FIGURE 1.24 – Application de forces sur un objet virtuel grâce à un ressort entre chacune des mains virtuelles et le point d'application de leur force [Noma et Miyasato, 1997].

1.2.4.2 Accès concurrents à un même degré de liberté

Pour permettre à des utilisateurs d'accéder en même temps aux mêmes degrés de liberté d'un objet virtuel, il faut mettre en œuvre des techniques pour gérer les modifications concurrentes. Une première solution consiste à faire la moyenne des valeurs données par les outils d'interaction de chaque utilisateur afin de calculer la valeur du paramètre contrôlé comme la position ou l'orientation d'un objet virtuel. [Noma et Miyasato, 1997] proposent une deuxième solution qui consiste à appliquer des forces sur les points de l'objet saisi par les utilisateur grâce à une main virtuelle. La position et l'orientation de l'objet virtuel sont le résultat de l'équilibre des forces appliquées par les utilisateurs. Afin d'éviter les mouvements brusques et les instabilités, un système de ressort est utilisé entre la main virtuelle et le point d'application de la force sur l'objet virtuel (cf. figure 1.24). Si deux utilisateurs seulement veulent réaliser ensemble cette manipulation physique (basée sur des forces), il est difficile de contrôler les six degrés de liberté de l'objet virtuel (en particulier la rotation) parce qu'il tourne de façon non contrôlée autour de l'axe formé par les deux points saisis. [Aguerreche *et al.*, 2009] proposent une technique de manipulation à trois mains : un utilisateur possède deux points de contrôle, alors que l'autre n'en possède qu'un seul. Ces trois points définissent un plan qui sert de support pour manipuler l'objet. Cette technique permet de contrôler plus facilement et d'une façon plus réaliste les six de degrés de liberté de l'objet virtuel. Si les utilisateurs effectuent une collaboration locale, [Aguerreche *et al.*, 2010] proposent d'utiliser en plus une interface tangible re-configurable pour matérialiser les trois points de contrôle.



(a) (b) (c)

FIGURE 1.25 – Rayons (a) élastiques, (b) coudés, et (c) courbés [Duval et Fenals, 2002].

Par ailleurs, il faut aussi trouver des solutions pour représenter les actions appliquées par les utilisateurs sur l'objet virtuel afin de permettre à chacun de comprendre quelle est son action et quelles sont les actions des autres sur l'objet. Si les utilisateurs utilisent des dispositifs à retour d'effort, il est possible de faire ressentir au travers d'un retour haptique les actions des autres. Autrement, il est possible d'utiliser des retours visuels comme une ligne extensible pour matérialiser le « ressort » entre le point saisi de l'objet et l'outil servant à le manipuler, ou des rayons coudés ou courbés pour matérialiser l'action appliquée à l'objet par les différents utilisateurs, comme proposé par [Riege *et al.*, 2006] ou [Duval et Fenals, 2002] (cf. figure 1.25).

1.3 Synthèse

Un système de réalité virtuelle peut être décrit par une boucle perception/action qui permet aux utilisateurs d'accéder à un monde virtuel. Intégrer un utilisateur dans cette boucle consiste à lui faire percevoir l'environnement virtuel au niveau de ses différents sens (perception) et à lui permettre d'interagir avec les objets virtuels (action). Induire une sensation de présence chez cet utilisateur nécessite une bonne immersion multi-sensorielle de l'utilisateur dans l'environnement virtuel, mais aussi une bonne co-localisation dans le monde virtuel les parties de l'environnement réel de l'utilisateur qui ne sont pas « masquées » par l'environnement virtuel (volontairement ou involontairement). Il existe de nombreuses solutions et de nombreux dispositifs matériels pour réaliser cette immersion multi-sensorielle. Pour interagir avec l'environnement virtuel, l'utilisateur doit aussi être capable de manipuler les différents objets du monde virtuel, dont un objet bien particulier : son point de vue sur le monde virtuel. Modifier son point de vue lui permet de se déplacer librement dans le monde virtuel. Il existe de nombreuses techniques pour naviguer et interagir dans un environnement virtuel qui diffèrent en fonction des dispositifs matériels utilisés et des tâches que les utilisateurs doivent réaliser. De plus, permettre aux utilisateurs d'interagir avec l'environnement virtuel participe aussi à la sensation de présence.

La collaboration dans un environnement virtuel consiste à intégrer plusieurs utilisateurs dans une même boucle perception/action afin que chacun puisse agir sur le monde virtuel, mais qu'ils perçoivent tous le même monde. Ainsi ces utilisateurs vont pouvoir réaliser ensemble des tâches plus ou moins complexes dans l'environnement virtuel. Cependant, cette collaboration n'est pas évidente et elle nécessite une bonne compréhension entre les utilisateurs. Même s'il existe des solutions pour permettre aux utilisateurs de percevoir les autres, de communiquer, de naviguer et de réaliser des manipulations ensemble dans un environnement virtuel collaboratif, il subsiste des besoins tant au niveau technique que conceptuel pour améliorer la collaboration en environnement virtuel.

Besoins techniques

Les différentes solutions présentées dans ce chapitre pour permettre à des utilisateurs de collaborer dans un environnement virtuel ne prennent généralement pas en compte les difficultés techniques pour mettre en œuvre un tel environnement virtuel collaboratif. Cependant, pour améliorer la collaboration entre ces utilisateurs, il est aussi nécessaire d'améliorer les performances au niveau technique des environnements virtuels collaboratifs.

Premièrement, dans le cadre de cette thèse, nous nous intéressons plus particulièrement à la collaboration à distance. Dès lors que les utilisateurs se situent à distance, les délais

de communication sur le réseau peuvent introduire des difficultés supplémentaires pour la collaboration, car ils entrainent des désynchronisations entre les états de l'environnement virtuel que perçoivent chacun des utilisateurs. Si les utilisateurs ne perçoivent pas le même état de l'environnement virtuel au même moment, cela peut poser problème pour la communication, les co-manipulations et, plus généralement, les interactions entre utilisateurs. Il faut donc trouver des solutions pour garantir que tous les utilisateurs perçoivent un état cohérent de l'environnement virtuel sur chaque site tout en leur offrant un temps de réponse acceptable lorsqu'ils interagissent.

Deuxièmement, il existe également des contraintes techniques pour intégrer dans un même environnement virtuel des utilisateurs utilisant des dispositifs matériels différents, allant de la simple station de travail jusqu'à la salle immersive de réalité virtuelle. En effet, il faut que le système de réalité virtuelle puisse être adapté en fonction de ces dispositifs matériels. Nous avons présenté dans la partie 1.1.3.1 les nombreuses solutions existantes pour s'abstraire des dispositifs matériels lors de la conception d'un système de réalité virtuelle. Cependant, nous souhaitons également que le système soit capable de s'adapter aux différents composants logiciels que les utilisateurs utilisent pour gérer le graphe de scène, faire le rendu graphique et sonore, etc. Cela éviterait d'obliger tous les utilisateurs à utiliser les mêmes composants logiciels (librairies graphiques, sonores, etc.) et chaque utilisateur pourrait choisir les plus adaptés en fonction des dispositifs matériels qu'il utilise.

Dans le chapitre 2, nous présentons un état de l'art plus détaillé des solutions existantes pour résoudre ces deux problématiques de conception afin d'identifier ce qui permettrait d'améliorer les performances techniques des environnements virtuels collaboratifs.

Besoins conceptuels

En plus des contraintes techniques vues dans la partie précédente pour adapter un système de réalité virtuelle à différents dispositifs matériels, il ressort de l'état de l'art un besoin récurrent de tenir compte dans l'environnement virtuel des différents espaces d'interaction associés à ces dispositifs (cf. figure 1.26). Les systèmes de réalité virtuelle doivent intégrer cet environnement réel qui entoure chacun des utilisateurs afin :

- D'adapter les techniques d'interaction proposées aux utilisateurs. En effet, pour adapter la façon dont les actions des utilisateurs sont traitées, il peut être nécessaire de modéliser l'espace de déplacement physique de l'utilisateur comme pour les techniques de marche « naturelle » [Razzaque, 2005][Williams *et al.*, 2007], l'espace d'interaction d'un dispositif haptique comme pour la *Bubble* [Dominjon *et al.*, 2005] ou l'espace de visualisation comme pour le *HHD* [Amselem, 1995].
- De permettre aux utilisateurs de percevoir leurs possibilités de perception et d'interaction comme, par exemple, leur espace de déplacement physique avec le tapis volant [Butterworth *et al.*, 1992] et la *Magic Barrier Tape* [Cirio *et al.*, 2009] ou leur espace d'interaction avec un dispositif haptique avec la *Bubble* [Dominjon *et al.*, 2005].
- De permettre aux utilisateurs de percevoir les possibilités de perception et d'interaction des autres utilisateurs. En effet, pour améliorer la collaboration, il est pertinent de faire percevoir aux utilisateurs les espaces d'interaction des autres, comme le proposent [Fraser *et al.*, 1999] pour l'espace de visualisation.

Cependant, même si certaines de ces applications intègrent un espace d'interaction particulier associé à un dispositif matériel, les concepteurs de ces applications ne décrivent pas de façon explicite l'existence d'un tel espace et la manière de l'intégrer ne peut pas être

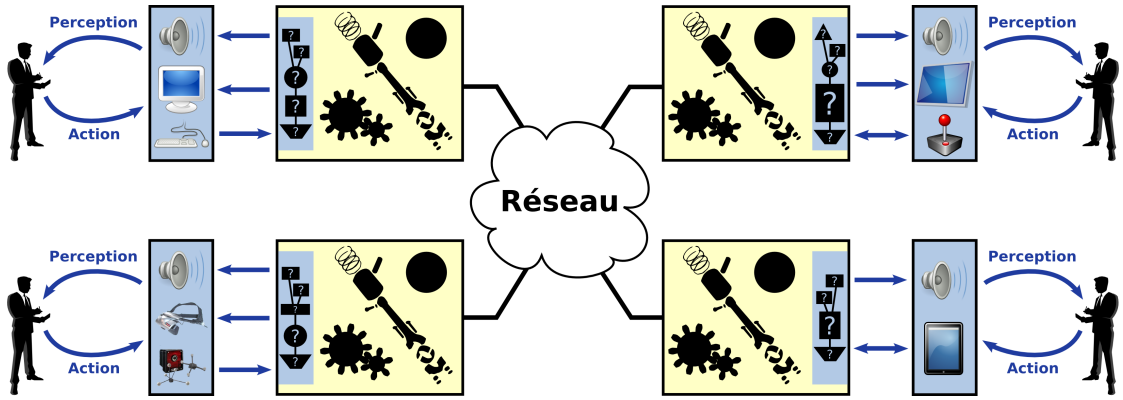


FIGURE 1.26 – Intégrer les espaces d'interaction réels dans l'environnement virtuel.

généralisée à d'autres espaces d'interaction. Seuls [Mulder et Boschker, 2004] décrivent des espaces d'interaction multi-sensoriels. Néanmoins, cette description n'est pas un modèle générique qui s'adapte à différents dispositifs immersifs, ni à d'autres techniques d'interaction. Elle ne propose pas non plus de solution pour intégrer ces espaces d'interaction dans le monde virtuel, ni pour les faire percevoir aux utilisateurs. Ces aspects plus conceptuels de modélisation et intégration dans l'environnement virtuel, des espaces d'interaction réels qui entourent chaque utilisateur sont traités dans le chapitre 5.

Chapitre 2

Architectures des environnements virtuels collaboratifs

Les travaux réalisés dans le cadre de cette thèse s'intéressent principalement à la collaboration distante. Pour permettre à des utilisateurs situés sur des lieux géographiquement éloignés de collaborer, il faut mettre en œuvre un environnement virtuel distribué entre ces différents sites. Nous avons établi au chapitre précédent que pour permettre une collaboration efficace, tous les utilisateurs doivent percevoir en même temps un état cohérent de l'environnement virtuel, même si les modifications de cet environnement virtuel doivent être propagées sur le réseau entre les différents sites. Il faut donc définir comment modéliser les objets présents dans le monde virtuel et où les traiter afin d'assurer une bonne cohérence de l'environnement virtuel. Nous avons aussi conclu au chapitre précédent qu'il serait intéressant de modéliser l'environnement virtuel indépendamment des composants logiciels utilisés pour le restituer aux utilisateurs (ces composants logiciels peuvent être différents en fonction des dispositifs matériels utilisés par chacun des utilisateurs).

Nous étudions dans la section 2.1, les systèmes existants qui mettent en œuvre un environnement virtuel distribué, ainsi que les mécanismes utilisés pour assurer une bonne cohérence de cet environnement. Puis, nous détaillons dans la section 2.2, les architectures logicielles qui permettent de modéliser une application collaborative indépendamment de la partie réseau et des composants logiciels particuliers utilisés par chacun des utilisateurs.

2.1 Environnement virtuel distribué et maintien de cohérence

Depuis le milieu des années 80 jusqu'à nos jours, un grand nombre d'environnements virtuels distribués a été conçu pour répondre aux besoins spécifiques d'applications diverses et variées. Ces environnements virtuels distribués comme tout système distribué nécessitent de maintenir la cohérence entre les états de chacun des utilisateurs.

Dans un premier temps, nous présentons dans la partie 2.1.1, une rapide description d'ordre générale des différents systèmes étudiés dans cette section. Puis, nous montrons dans le reste de la section que les caractéristiques, en terme de cohérence et de performance d'un environnement virtuel distribué, sont fortement corrélées au choix de l'architecture du système. En effet, parmi les différents choix techniques possibles, certaines solutions permettent de répondre plus ou moins bien aux différents besoins d'un environnement virtuel

collaboratif distribué. Par exemple, certaines architectures permettent de maintenir une forte cohérence entre les différents états de l'environnement virtuel mais introduisent de la latence lorsque les utilisateurs interagissent. À l'inverse, d'autres architectures autorisent quelques incohérences entre les états de l'environnement perçus par les différents utilisateurs mais offrent un temps de réponse bien meilleur lors des interactions. Généralement, les états de l'art existants classent les environnements virtuels distribués en fonction soit de la façon dont les ordinateurs sont connectés entre eux [Delaney *et al.*, 2006b][Joslin *et al.*, 2004], soit de la façon dont les données relatives à l'environnement virtuel sont distribuées sur ces ordinateurs [Macedonia et Zyda, 1997][El Zammar, 2005]. Même si ces deux caractéristiques sont souvent liées entre elles, il existe de plus en plus de solutions hybrides qui mixent ces deux choix en fonction des besoins de l'environnement virtuel collaboratif. Comme présenté dans [Fleury *et al.*, 2010b], nous avons donc choisi de classer les solutions existantes, d'une part, en fonction du type d'architecture réseau du système (partie 2.1.2) et, d'autre part, en fonction de la distribution des données (partie 2.1.3). Enfin, dans la partie 2.1.4, nous présentons les mécanismes qui permettent d'assurer une bonne cohérence de l'environnement virtuel.

2.1.1 Systèmes étudiés

Parmi les systèmes étudiés, les deux plus anciens sont *Reality Build for Two* [Blanchard *et al.*, 1990] et MR Toolkit [Shaw et Green, 1993]. *Reality Build for Two* est souvent considéré comme le premier environnement virtuel distribué. Il permet de mettre en relation dans un même monde virtuel deux utilisateurs équipés de HMD et de gants de réalité virtuelle leur permettant d'interagir. Il a été développé par la société *VPL Research, Inc.* fondée par Jaron Lanier et spécialisée dans ce genre de dispositifs de visualisation et d'interaction. MR Toolkit est un système plus expérimental développé par l'université d'Alberta au Canada. Il permet de mettre en relation plusieurs utilisateurs distants dans un même environnement virtuel collaboratif. Il a été utilisé, par exemple, pour mettre en place une démonstration d'un jeu de Handball à plusieurs joueurs.

L'environnement virtuel à vocation militaire SIMNET [Calvin *et al.*, 1993] est également un des plus anciens. Il aurait été instancié par le *DARPA (Defense Advanced Research Projects Agency)* dés 1983 avec le support de l'armée américaine. Il a été utilisé pour connecter différents simulateurs de véhicule (chars, hélicoptères, avions, etc.) au travers d'un réseau à grande échelle (WAN) pour effectuer des simulations de combat. Le but de ce projet était de prouver qu'un environnement virtuel distribué pouvait être utilisé pour entraîner des soldats de façon rentable. SIMNET a aussi été utilisé pour tester et évaluer de nouvelles techniques de combat ou de nouveaux types de véhicule lors de batailles virtuelles. Le système NPSNET [Macedonia *et al.*, 1994] a également été conçu pour les simulations militaires avec un nombre de participants importants sur des réseaux à grande échelle. Il a été développé par la *Naval Postgraduate School (NPS)* à Monterey en Californie (USA). La distribution de l'environnement virtuel est basée sur une décomposition en différents objets et sur des événements entre ces objets. Cet environnement virtuel distribué a été l'un des précurseurs au niveau de la distribution réseau avec l'utilisation du *multicast*, de mécanismes de réductions des communications réseaux et de l'informatique parallèle pour distribuer les traitements sur les machines des différents utilisateurs.

Les systèmes RING [Funkhouser, 1995], BrickNet [Singh *et al.*, 1995] et VIPER [Torguet et Caubet, 1995] se concentrent principalement sur la distribution sur le réseau. RING

a été conçu par le laboratoire de télécommunications AT&T Bell. Il propose d'intégrer un grand nombre d'utilisateurs dans un même environnement virtuel en réduisant les communications sur le réseau et les traitements en fonction de ce que voit chaque utilisateur. Il s'adapte donc particulièrement bien aux environnements virtuels où il y a de nombreuses occlusions comme un bâtiment ou une ville. Bricknet a été développé à l'université nationale de Singapour *(NUS)* et sa particularité réside principalement dans le fait que chaque utilisateur peut avoir des objets virtuels privés ou publiques. Il a permis de développer un environnement virtuel pour la formation à distance où des élèves peuvent avoir chacun des objets privés pour réaliser les manipulations demandées, mais ils peuvent aussi partager ces objets lorsqu'ils ont besoin de les montrer à d'autres élèves ou à l'instructeur. Enfin, VIPER a été conçu à l'université Paul Sabatier de Toulouse et propose de modéliser un environnement virtuel distribué en se focalisant sur les échanges (symbolisés par des stimulus) entre les entités virtuelles.

Le système VISTEL *(Virtual Space teleconferencing system)* [Yoshida *et al.*, 1995] a été développé par l'*ATR Research Lab* au Japon. Comme son nom l'indique, ce système propose d'étendre un système de téléconférence en un espace virtuel dans lequel les utilisateurs ne font pas seulement que parler, mais peuvent aussi partager des objets 3D pour améliorer leurs possibilités de collaboration.

La plate-forme logicielle SPLINE *(Scaleable PLatform for Interactive Environments)* [Waters *et al.*, 1997], développée par le *Mitsubishi Electric Research Lab*, permet de créer des environnements virtuels distribués adaptés à la collaboration entre plusieurs utilisateurs distants. Les applications de ce système portent principalement sur les relations sociales dans un monde virtuel avec les utilisateurs qui interagissent au travers de leur avatar. L'application la plus célèbre, appelée *"Diamond Park"*, permet à des utilisateurs de se promener à vélo dans un parc virtuel en pédalant sur des vélos dans le monde réel.

Les systèmes PaRADE [Roberts et Sharkey, 1997] et DIVE [Frécon et Stenius, 1998] s'attaquent tous les deux au problème des réseaux hétérogènes : ils ont pour but de connecter entre eux, dans un même environnement virtuel, des utilisateurs qui utilisent des réseaux aux caractéristiques différentes. PaRADE *(Predictive Real time Architecture for Distributed Environments)* a été conçu à l'université de Reading en Angleterre. Il permet d'intégrer dans un même monde virtuel des utilisateurs avec des machines de puissance hétérogène, connectées soit par un réseau local (LAN), soit par une réseau étendu (WAN). Il cherche à minimiser la bande passante utilisée par un environnement virtuel distribué en mettant en place des techniques pour prévoir les événements qui peuvent survenir dans l'environnement virtuel. DIVE *(Distributed Interactive Virtual Environment)* a été développé par le *Swedish Institute of Computer Science*. DIVE est basé sur un ensemble de processus de communication exécutés sur les machines des différents utilisateurs connectées par un réseau LAN ou une réseau WAN. Chaque processus représentant soit un utilisateur humain, soit une application autonome, communique avec une base de données qui stocke une description des objets du monde virtuel afin de garantir sa cohérence.

Le système SPIN-3D [Dit Picard *et al.*, 2002] a été développé par le laboratoire d'informatique fondamentale de Lille dans un projet financé par France Télécom R&D. Il permet de créer une salle de réunion virtuelle où chacun des utilisateurs est représenté par un avatar situé autour de la table d'une salle de réunion. Les utilisateurs peuvent interagir ensemble et partager des documents durant la session collaborative. La particularité de SPIN-3D réside dans le fait que seul les objets virtuels partagés et les documents partagés sont synchronisés entre les utilisateurs sur le réseau, et non toute la scène virtuelle.

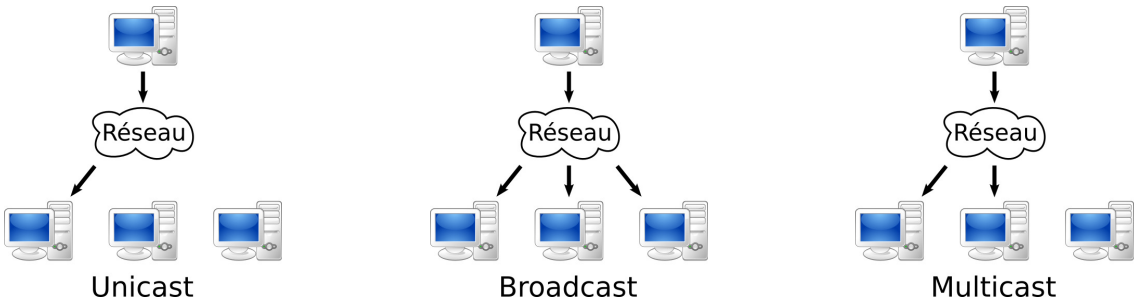


FIGURE 2.1 – Méthodes de communication réseau.

La plate-forme OpenMASK *(Modular Animation and Simulation Kit)* [Margery *et al.*, 2002], développée à l'IRISA/Inria Rennes, a pour but d'offrir un système modulaire pour concevoir des applications de réalité virtuelle. Elle est basée sur un noyau principal qui gère l'exécution des objets virtuels à des fréquences différentes, ainsi que la communication entre ces objets. Elle permet aussi de répartir les différents traitements de l'environnement virtuel sur les machines utilisées par les utilisateurs lors de la session collaborative.

La plate-forme de réalité virtuelle ASSET *(Architecture for Systems of Simulation and Training)* [El Merhebi *et al.*, 2007], conçue à l'université Paul Sabatier de Toulouse, a été utilisée pour implémenter différents algorithmes permettant d'améliorer la distribution de l'environnement virtuel sur le réseau. En particulier, elle a permis de tester une architecture réseau basée sur plusieurs serveurs, ainsi que des techniques de filtrage des communications réseau en fonction de ce que les utilisateurs perçoivent du monde virtuel.

Le système ATLAS [Lee *et al.*, 2007] a été développé par l'*Information and Communications University (ICU)* en Corée. Il fournit un mécanisme de gestion des accès concurrents aux objets virtuels basé sur une prédiction déterminée en fonction des centres d'intérêts de chacun des utilisateurs. Il a utilisé pour réaliser des applications diverses allant de la conception collaborative assistée par ordinateur aux jeux vidéo multi-joueurs.

Enfin, le système ShareX3D [Jourdain *et al.*, 2008] est un prototype développé dans le cadre du sous-projet V3D initié par EDF R&D au sein du projet SCOS de l'Agence Nationale pour la Recherche (ANR). Ce prototype est une application en ligne qui permet a plusieurs utilisateurs de partager un environnement virtuel en étant connecté à un serveur par des connections HTTP utilisant la technique du *"long polling"*.

2.1.2 Architecture réseau du système

Un environnement virtuel collaboratif distribué est généralement constitué de plusieurs ordinateurs inter-connectés et souvent géographiquement dispersés que nous appelons « nœuds ». Il existe principalement trois méthodes de transmission des données qui permettent à ces nœuds de communiquer entre eux sur le réseau (cf. figure 2.1) :

- *unicast* : transmission uniquement d'un nœud à un autre (point-à-point),
- *broadcast* : transmission d'un nœud à tous les autres nœuds du réseau,
- *multicast* : transmission d'un nœud à un sous-ensemble de nœuds du réseau.

[Delaney *et al.*, 2006b] distinguent deux organisations possibles des ordinateurs sur le réseau pour un environnement virtuel distribué : les architectures pair-à-pair (partie 2.1.2.1) et les architectures client-serveur (partie 2.1.2.2). [Lee *et al.*, 2007] proposent d'ajouter les architectures hybrides qui combinent ces deux types d'architecture (partie 2.1.2.3).

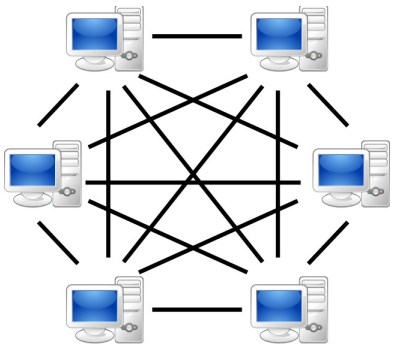


FIGURE 2.2 – Architecture pair-à-pair.

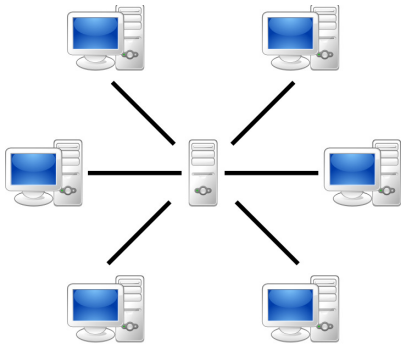


FIGURE 2.3 – Architecture client/serveur.

2.1.2.1 Architecture pair-à-pair

Les informations sont transmises directement entre les nœuds des utilisateurs qui évoluent dans l'environnement virtuel (cf. figure 2.2). Parmi les systèmes qui utilisent une telle architecture, nous pouvons citer *Reality Build for Two* [Blanchard *et al.*, 1990], MR Toolkit [Shaw et Green, 1993], SIMNET [Calvin *et al.*, 1993], NPSNET [Macedonia *et al.*, 1994] et SPIN-3D [Dit Picard *et al.*, 2002]. Lorsque le nombre d'utilisateurs croît fortement, le nombre de messages à transiter sur le réseau peut être très important, en particulier avec un mode de transmission *unicast* (si le système contient N utilisateurs, un nœud doit envoyer $N-1$ messages lorsqu'il veut transmettre une mise à jour). Il peut également être difficile de contacter rapidement tous les nœuds participant à la session collaborative, ce qui complexifie le maintien de la cohérence entre les différents nœuds.

2.1.2.2 Architecture client/serveur

Cette architecture fournit un accès central auquel tous les utilisateurs sont connectés (cf. figure 2.3). Ce serveur central peut gérer la communication entre les différents utilisateurs et stocker des données. Nous pouvons citer comme exemples d'architecture client-serveur les systèmes RING [Funkhouser, 1995], BrickNet [Singh *et al.*, 1995], VISTEL [Yoshida *et al.*, 1995] et ShareX3D [Jourdain *et al.*, 2008]. Cependant, avec cette architecture, lorsque le nombre d'utilisateurs croît fortement, un goulot d'étranglement peut apparaître au niveau du serveur, dû à un nombre trop important de requêtes entrantes.

2.1.2.3 Architecture hybride

Pour palier les limitations de ces deux types d'architecture, de plus en plus de systèmes utilisent des architectures hybrides, qui utilisent à la fois des communications entre les nœuds et des communications avec un serveur. Cela peut permettre par exemple d'accélérer les communications entre les utilisateurs tout en maintenant plus facilement la cohérence de l'environnement virtuel grâce au serveur.

Dans SPLINE [Waters *et al.*, 1997], plusieurs serveurs se répartissent les flots d'information grâce à des connexions *unicast* entre serveurs (cf. figure 2.4). Les utilisateurs sont connectés à un seul serveur grâce à un serveur qui gère uniquement les connexions. Cette architecture permet d'éviter le goulot d'étranglement de l'architecture client/serveur et permet de connecter des utilisateurs avec besoins particuliers. ASSET [El Merhebi *et al.*,

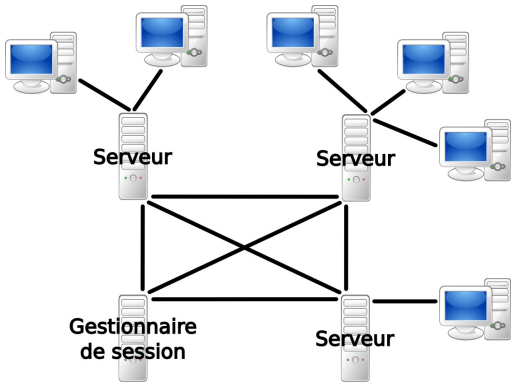


FIGURE 2.4 – Plusieurs serveurs connectent des utilisateurs à un environnement virtuel.

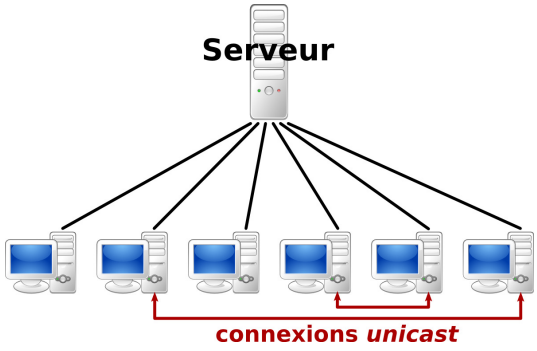


FIGURE 2.5 – Plusieurs connexions *unicast* supplémentaires sont établies entre des utilisateurs qui collaborent ensemble.

2007] utilise également plusieurs serveurs pour connecter plusieurs utilisateurs à un même environnement virtuel. Cet environnement virtuel est divisé en régions spatiales et chaque serveur est responsable d'une région. Un serveur particulier est en charge de la connexion des utilisateurs et de la subdivision spatiale. Même si les serveurs gèrent les entités de leur région, il est parfois nécessaire d'échanger des informations avec les serveurs qui gèrent les régions voisines. Pour cela, chaque serveur communique les informations de sa région aux serveurs intéressés grâce à un groupe de diffusion *multicast*.

[Anthes *et al.*, 2004] proposent une autre architecture hybride pour faciliter la collaboration entre des utilisateurs proches (en terme de distance dans le monde virtuel). Les utilisateurs sont tous connectés à un même serveur. Cependant, lorsque des utilisateurs sont proches dans le monde virtuel, des connexions *unicast* sont établies entre eux afin d'augmenter la cohérence de l'environnement virtuel entre ces utilisateurs et donc de faciliter leur collaboration (cf. figure 2.5).

La plate forme OpenMASK [Margery *et al.*, 2002] est également une architecture hybride. En effet, l'architecture est de type pair-à-pair pour l'envoi des mises à jour et d'événements. Par contre, elle est de type client/serveur pour le service de nommage des objets virtuels et pour les éventuels ajouts dynamiques de site en cours de session.

2.1.3 Distribution des données

[Macedonia et Zyda, 1997] précisent qu'une des décisions les plus critiques lors de la conception d'un environnement virtuel collaboratif est de choisir où mettre les données relatives à l'environnement virtuel et aux objets qui le constituent. Il faut déterminer sur quels nœuds placer les données (données géométriques, textures, etc.), mais aussi sur quels nœuds exécuter le processus relatif à chacun de ces objets. Nous distinguons trois modes de distribution des données : le mode centralisé (partie 2.1.3.1), le mode homogènement répliqué (partie 2.1.3.2) et le mode partiellement répliqué (partie 2.1.3.3).

2.1.3.1 Mode centralisé

Certains systèmes comme VISTEL [Yoshida *et al.*, 1995] utilisent une seule base de données partagée par tous les utilisateurs de l'environnement virtuel collaboratif. Toutes les données relatives à l'environnement virtuel sont contenues sur un serveur central et

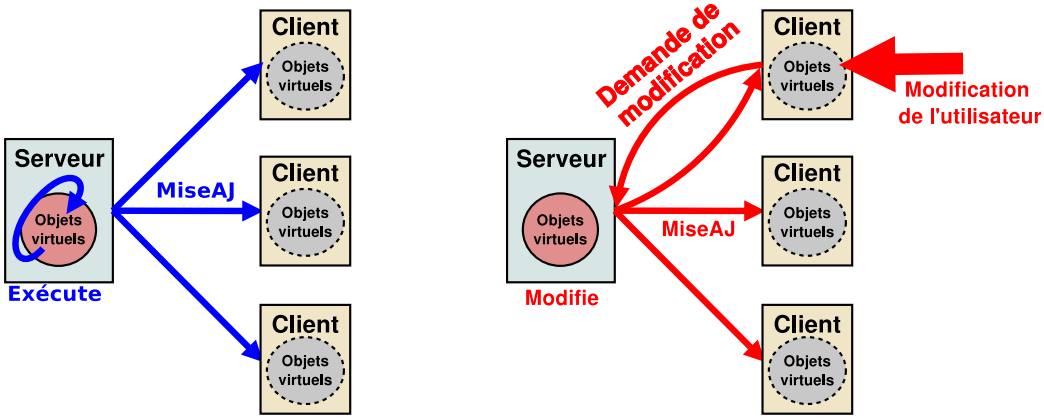


FIGURE 2.6 – (a) Exécution du comportement des objets et (b) modification d'un objet dans le cas d'un mode centralisé

tous les traitements relatifs aux objets sont exécutés sur ce serveur (cf. figure 2.6(a)). Les utilisateurs viennent se connecter au serveur pour participer à la session collaborative (cette approche impose une architecture réseau de type client/serveur). Lorsqu'un utilisateur veut modifier l'état d'un objet, il envoie une requête au serveur. Le serveur traite alors la demande de modification, puis transmet une mise à jour à tous les nœuds, y compris à celui qui a fait la requête (cf. figure 2.6(b)). Cette méthode assure une forte cohérence de l'environnement virtuel entre les nœuds et permet de ne pas dupliquer les données. Cependant, elle a deux inconvénients majeurs :

- Dès que la latence réseau entre les clients et le serveur est importante, un délai apparait entre les actions des utilisateurs et les modifications effectives de l'environnement virtuel par le serveur. Ce délai peut être extrêmement gênant pour les utilisateurs.
- Lorsque le nombre d'utilisateurs devient important, un goulot d'étranglement risque d'apparaître sur le serveur car ce dernier doit envoyer des messages de mise à jour à tous les nœuds à la fois.

2.1.3.2 Mode homogènement répliqué

La méthode la plus souvent utilisée dans les environnements virtuels collaboratifs est d'initialiser l'état de chaque nœud participant à la session collaborative avec une même base de données contenant toutes les informations sur le monde virtuel (terrain, modèles géométriques, textures, comportement des objets,...) comme, par exemple, dans MR Toolkit [Shaw et Green, 1993]) ou SPIN-3D [Dit Picard *et al.*, 2002]. Les données peuvent être déjà présentes sur le nœud de l'utilisateur lorsqu'il se connecte à la session (comme dans la plupart des jeux vidéo où un CD-Rom est utilisé). Dans le cas contraire, ces données seront répliquées au moment de la connexion initiale soit à partir des autres nœuds déjà présents dans la session, soit à partir d'un serveur. Pendant la session, chacune des bases de données évolue librement sur les nœuds des utilisateurs et tous les traitements relatifs aux objets sont exécutés en local (cf. figure 2.7(a)). Pour maintenir la cohérence, seuls les changements d'état des objets et parfois des évènements particuliers (collision entre deux objets,...) sont transmis sur le réseau afin que tous les utilisateurs puissent mettre à jour

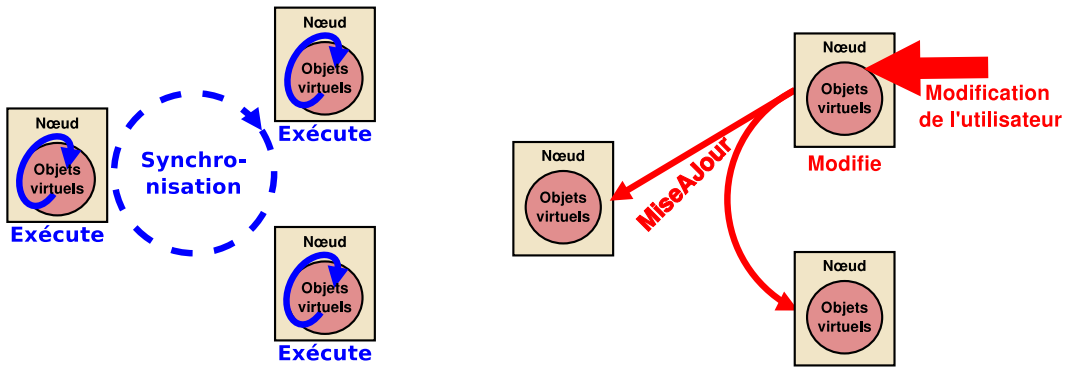


FIGURE 2.7 – (a) Exécution du comportement des objets et (b) modification d'un objet dans le cas d'un mode homogènement répliqué

leur base de données (cf. figure 2.7(b)). Cette méthode a deux avantages principaux :

- Le nombre et surtout la taille des messages à transiter sur le réseau est relativement faible étant donné que seulement des messages de mise à jour sont envoyés.
- La latence est très faible lorsque l'utilisateur interagit car les objets manipulés sont traités en local sur son nœud. Les modifications se font d'abord sur sa copie locale avant d'être envoyées aux autres par des messages de mise à jour.

Cependant, cette méthode comporte également des inconvénients :

- Si le volume des données de l'environnement virtuel augmente, le volume des données de chaque nœud augmente aussi. Cette solution n'est donc pas envisageable pour des volumes de données importants (données scientifiques, CAO, etc.).
- Cette approche est relativement peu flexible, en particulier, si les utilisateurs veulent ajouter des nouveaux objets qui ne sont pas prévus dans la base de données initiale.
- Il est difficile de gérer les droits d'accès des utilisateurs aux objets. Par exemple, deux utilisateurs peuvent manipuler en même temps le même objet en local et le conflit apparaît uniquement lors de la mise à jour de cet objet. Il est donc nécessaire de mettre en place des mécanismes pour gérer les accès concurrents des utilisateurs aux objets virtuels (cf. partie 2.1.4.2).
- L'environnement virtuel peut devenir incohérent entre les nœuds à cause de la latence et de la perte de certaines données sur le réseau lors des mises à jour. De plus, il n'est pas envisageable de mettre entre place de la co-manipulation avec cette solution.

2.1.3.3 Mode partiellement répliqué (hybride)

Plusieurs systèmes utilisent des approches hybrides pour palier certains des inconvénients des deux modes présentés précédemment. Ces approches proposent de répartir les données ou les processus entre les différents nœuds pour éviter de multiplier les ressources nécessaires et pour faciliter le maintien de la cohérence.

Pour éviter de dupliquer toutes les données sur les nœuds de tous les utilisateurs, les données peuvent être réparties entre ces différents nœuds. La base de données peut alors être considérée comme une mémoire partagée et distribuée entre les nœuds. Par exemple, dans DIVE [Frécon et Stenius, 1998], seule une partie des données est activement répliquée. Ainsi, au lieu de télécharger les données du monde virtuel en entier, seuls les objets dont

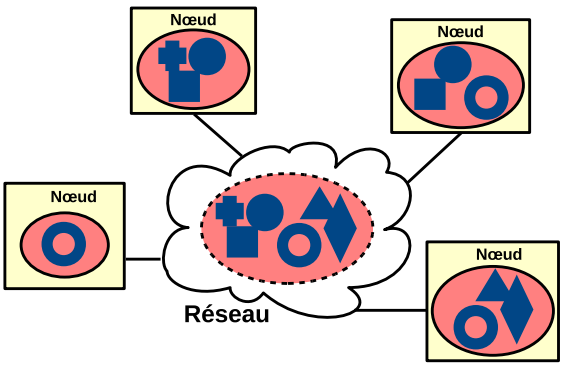


FIGURE 2.8 – Données partiellement répliquées chez différents utilisateurs.

l'utilisateur a besoin sont copiés sur son nœud (cf. figure 2.8). Par contre, si l'utilisateur a besoin d'autres objets au cours de la session, il devra les re-télécharger dynamiquement. DIVE utilise des connexions *unicast* pour gérer les communications entre les utilisateurs (transfert de messages ou d'objets lorsque c'est nécessaire), et il utilise aussi un serveur pour garder une trace de toutes les données présentes chez les différents utilisateurs afin de pouvoir sauvegarder l'état de l'environnement virtuel en cas de déconnexion brutale de certains utilisateurs ou pour relancer la session là où elle a été laissée.

L'avantage de cette méthode est qu'elle peut s'étendre à un nombre d'utilisateurs très important ou à un gros volume de données qu'il n'est pas nécessaire de dupliquer pour chaque utilisateur. Par contre, son inconvénient est le coût de communication élevé pour transmettre les données supplémentaires et pour garantir la cohérence. Lors de la modification de l'état d'un objet, il faut transmettre des messages de mise à jour à tous les autres utilisateurs même s'ils n'ont pas l'objet en question. En effet, dans le cas où le nombre d'utilisateurs est important, un utilisateur peut ne pas connaître quels sont les autres utilisateurs qui possèdent l'objet qu'il vient de modifier.

La plus grande difficulté des systèmes utilisant une base de données partiellement répliquée est de trouver comment répliquer efficacement les données manquantes sans que l'immersion de l'utilisateur dans l'environnement virtuel soit perturbée. D'après [Lee *et al.*, 2007], deux solutions sont principalement utilisées :

- Le transfert par priorité, qui consiste à sélectionner uniquement les objets virtuels qui sont dans le champ de vision de l'utilisateur et à transmettre ces objets en utilisant des niveaux de détails (*LODs*) ou des techniques multi-résolution en fonction de leur apport visuel dans la scène pour l'utilisateur. Cette technique permet de maximiser la qualité graphique et la performance des interactions en équilibrant les détails graphiques des objets virtuels avec les possibilités de transmission du réseau.
- Le pré-chargement des données, qui consiste à pré-charger en cache les données dont l'utilisateur aura potentiellement besoin afin qu'elles soient immédiatement disponibles lorsqu'il les demandera. Il faut trouver un moyen de prédire de façon efficace quels objets vont intéresser en premier l'utilisateur afin de définir une priorité de pré-chargement. Pour cela, la distance entre l'utilisateur et les objets virtuels est traditionnellement utilisée en supposant que l'utilisateur va être intéressé par les objets dont il s'approche. D'autres solutions proposent d'ajouter la direction de déplacement de l'utilisateur ou d'utiliser le type des objets pour déterminer le champ d'intérêt de cet utilisateur. Cependant, plus le nombre d'objets devient important, plus il est difficile de prévoir quels objets vont intéresser les utilisateurs.

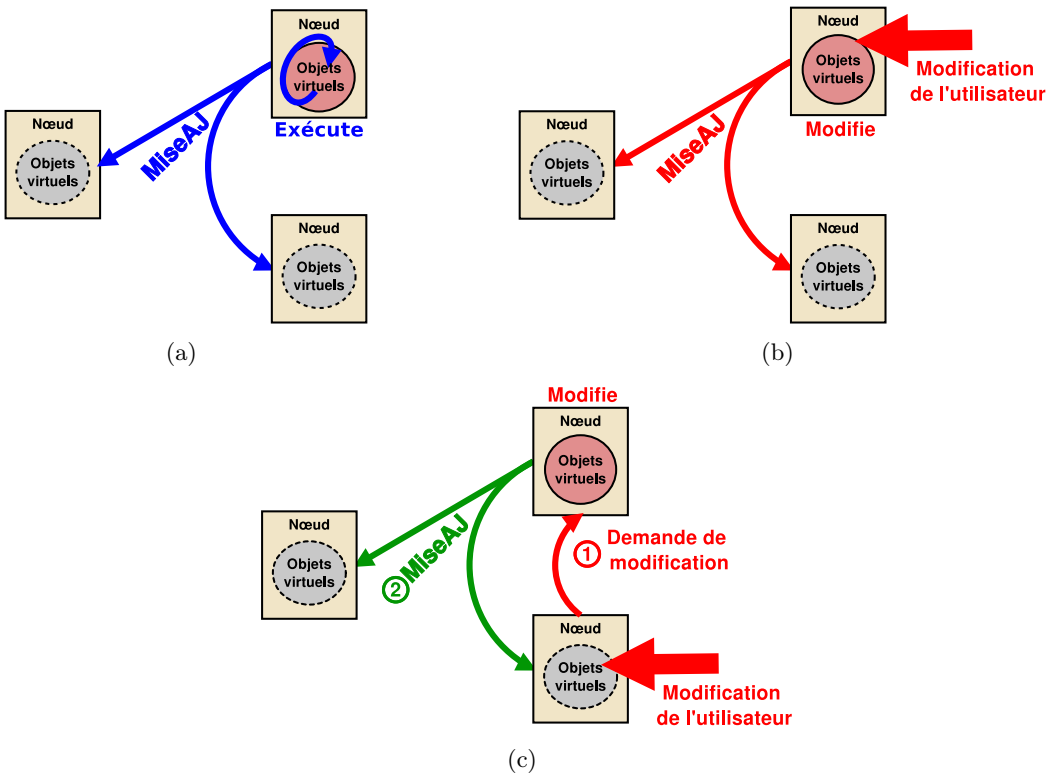


FIGURE 2.9 – (a) Exécution du comportement des objets, (b) modification directe d'un objet par son référent, (c) modification indirecte d'un objet par son miroir dans OpenMASK.

Pour éviter des coûts de communication trop importants lors des mises à jour, BrickNet [Singh *et al.*, 1995] utilise une architecture client/serveur où le serveur garde une trace des objets virtuels, des droits d'accès à ces objets et des utilisateurs qui possèdent ces objets. Les données de l'environnement virtuel sont également partiellement répliquées sur les nœuds des utilisateurs et les comportements des objets sont exécutés sur le processus local de chaque nœud. Ainsi, lorsqu'un utilisateur veut modifier un objet virtuel, il commence par demander au serveur les droits d'accès à cet objet (un seul utilisateur à la fois peut modifier un objet). Lorsqu'il obtient les droits d'accès de l'objet, il peut le modifier en local avant qu'une mise à jour soit transmise au serveur et répercutée seulement aux autres utilisateurs qui possèdent cet objet. Cette approche permet de faciliter le maintien de la cohérence et la gestion de la concurrence sur les objets grâce à l'utilisation d'un serveur. Cependant, nous retrouvons les limitations de l'architecture client/serveur.

Dans OpenMASK [Margery *et al.*, 2002], les données de l'environnement virtuel sont aussi répliquées sur le nœud de chaque utilisateur. Cependant, les processus relatifs aux objets ne s'exécutent pas sur chacun des nœuds. En effet, seul le nœud d'un des utilisateurs est responsable du processus définissant le comportement d'un objet. Pour cela, OpenMASK utilise le paradigme de référentiel et de miroirs comme présenté par [Duval et Margery, 2000b]. Le référentiel est assigné à un processus sur un nœud en particulier. C'est lui qui définit le comportement de l'objet. Sur les autres nœuds participant à la session collaborative, des miroirs sont créés pour représenter ce même objet. Un miroir possède la même interface que son référentiel distant (mêmes entrées, mêmes sorties,...). Cepen-

dant, il n'effectue aucun calcul en interne et se contente de « suivre » le comportement du référentiel (cf. figure 2.9(a)). Le noyau d'OpenMASK se charge de maintenir la cohérence entre le référentiel et ces miroirs en établissant une connexion entre les deux comme expliqué par [Duval et Margery, 2000a]. Lorsqu'un utilisateur manipule un objet dont le référentiel se trouve sur son nœud, l'état de l'objet est modifié en local, puis une mise à jour est envoyée aux autres nœuds (cf. figure 2.9(b)). Par contre, si le référentiel de l'objet que l'utilisateur manipule ne se trouve pas sur son nœud, l'état de l'objet sera d'abord modifié sur le nœud distant du référentiel, avant que ce nœud ne renvoie une mise à jour à tous les autres nœuds, y compris à celui de l'utilisateur qui interagit (cf. figure 2.9(c)). Ce deuxième cas de figure augmente le temps de réponse lors des interactions surtout si la latence réseau est importante. Par contre, cette solution permet de répartir la charge de calcul sur les différents nœuds participant à la session. Elle permet aussi de garantir une meilleure cohérence de l'environnement virtuel et de gérer de façon implicite les accès concurrents aux objets car seul le référentiel peut être modifié.

Enfin, VIPER [Torguet et Caubet, 1995] propose trois solutions pour distribuer les entités d'un environnement virtuel. Premièrement, les entités peuvent être homogènement répliquées sur tous les nœuds de la session. Deuxièmement, les entités peuvent être répliquées uniquement sur les nœuds qui ont besoin de les posséder. Troisièmement, les entités peuvent être traitées uniquement sur un nœud particulier, mais elles restent néanmoins accessibles à distance pour les autres entités. Cette troisième solution est utilisée pour les entités qui effectuent seulement des traitements et qui n'ont pas de représentation dans le monde virtuel. Le traitement de ces entités peut ainsi être répartis sur différents nœuds.

2.1.4 Mécanismes de maintien de la cohérence

En plus de l'architecture réseau et de la distribution des données, il existe des mécanismes qui permettent d'améliorer la cohérence de l'environnement virtuel. Dans la partie 2.1.4.1, nous détaillons les solutions existantes pour synchroniser ensemble les ordinateurs participant à la session collaborative. Dans la partie 2.1.4.2, nous présentons les solutions utilisées pour gérer les accès concurrents aux données lorsque la distribution des données ne garantit pas à elle seule la cohérence quand plusieurs utilisateurs tentent de modifier le même objet en même temps. Enfin, dans la partie 2.1.4.3, nous expliquons certains mécanismes utilisés pour réduire des coûts de communication et améliorer la cohérence entre les nœuds lorsque la bande passante sur le réseau est faible.

2.1.4.1 Synchronisation

Le temps est un élément fondamental d'un environnement virtuel collaboratif. Cependant, la notion de temps peut varier d'un système à un autre. [Delaney *et al.*, 2006a] distinguent deux notions de temps différentes selon les environnements virtuels distribués :

- **Le temps absolu** : le temps dans l'environnement virtuel est basé sur une horloge périodique synchronisée entre les différents nœuds de la session grâce au temps universel coordonné (UTC).
- **Le temps logique ou causal** : le temps dans l'environnement virtuel est basé sur une horloge logique qui n'est pas synchronisée très précisément entre les différents nœuds. Le temps peut être vu comme une séquence ordonnée d'évènements et si aucun nouvel évènement arrive, il n'avance pas.

Dans un environnement virtuel collaboratif parfaitement cohérent, tous les utilisateurs ont le même état global au même instant du temps absolu. Mais, ce cas parfait ne peut jamais être atteint à cause de la latence réseau. [Delaney *et al.*, 2006a] présentent différentes solutions pour garantir une plus ou moins bonne cohérence dans le temps en fonction de la réactivité souhaitée lors les interactions.

Synchronisation par blocage. La synchronisation par blocage, utilisée dans RING [Funkhouser, 1995], est la solution la plus simple pour assurer la cohérence dans un environnement virtuel collaboratif. Elle consiste à bloquer tous les utilisateurs jusqu'à ce que ces derniers aient effectué le traitement de tous les évènements du pas de temps de simulation en cours. Ainsi, aucun utilisateur n'avance son horloge de simulation tant qu'il n'a pas reçu tous les acquittements des autres utilisateurs pour signifier qu'ils sont prêts pour le prochain pas de temps. Cette solution impose que les évènements soient traités dans l'ordre. Elle garantit ainsi une parfaite cohérence de l'environnement virtuel, mais pas une bonne réactivité du système lors des interactions. En effet, lorsque la latence est importante ou que les transmissions ne sont pas fiables, il faut parfois attendre longtemps que tous les utilisateurs aient envoyé leur acquittement. De plus, les pas de temps de simulation ne sont pas forcément constants. Il n'y a donc pas toujours le même temps de réponse lors des interactions ce qui peut être perturbant pour les utilisateurs.

Cohérence globale imposée. Cette technique consiste à retarder le traitement des évènements à la fois locaux et distants d'une durée déterminée. Cette durée correspond en général à une borne maximale de la latence réseau. Le fait de retarder les traitements permet de maximiser les chances que tous les évènements distants (envoyés par les autres utilisateurs) soient bien arrivés avant de traiter l'ensemble des évènements. Cette méthode permet d'avoir une forte cohérence globale entre tous les nœuds de la session grâce à l'utilisation d'une horloge absolue. Par contre, lors des interactions, elle introduit un retard plus ou moins important selon l'architecture et les capacités du réseau. Néanmoins, ce retard est constant durant toute la session.

Cohérence globale retardée. À l'inverse de la cohérence globale imposée, l'objectif de cette technique est de maintenir une cohérence globale de façon asynchrone. Les utilisateurs perçoivent le même état de l'environnement virtuel, mais à des instants différents. Chaque évènement est marqué avec un *timestamp* (grâce à une horloge logique) afin que l'état global puisse être reconstruit localement en respectant l'ordre logique des évènements. Une horloge logique est maintenue chez chacun des utilisateurs. Néanmoins, le fait que la cohérence globale soit retardée peut nuire dans certains cas à la collaboration.

Synchronisation par distorsion temporelle. La synchronisation par distorsion temporelle est un mécanisme optimiste, qui consiste à traiter chaque évènement dès qu'il arrive. Les évènements sont également marqués avec un *timestamp*. Lorsque un évènement arrive avec un *timestamp* plus ancien que les évènements qui viennent d'être exécutés, un mécanisme doit annuler le traitement de tous les évènements exécutés en avance (*rollback*), exécuter le message qui vient d'arriver et ré-exécuter à nouveau tous les autres évènements jusqu'à rattraper le temps courant. De plus, le mécanisme doit aussi envoyer des messages pour annuler les messages incorrects envoyés précédemment (*rollback propagation*).

Cette méthode de synchronisation permet aux utilisateurs d'avoir des interactions réactives, car elles sont exécutées directement. Cependant, elle ne peut être utilisée que lorsque les retours arrière sont rares, car ils sont extrêmement gênants pour les utilisateurs. Certains systèmes proposent d'afficher quelques images clés lors de la ré-exécution des évènements pour faciliter la compréhension des utilisateurs lors d'un retour arrière. Par ailleurs, cette méthode nécessite de stocker les évènements reçus pour les ré-exécuter en cas de retour arrière.

Gestion du temps prédictive. Cette méthode consiste à prédire les évènements et à les envoyer sur le réseau avant qu'ils n'arrivent. Ce concept a été proposé en premier par [Roberts et Sharkey, 1997] dans le système PaRADE afin de résoudre les problèmes de maintien de cohérence de l'environnement virtuel lorsque les utilisateurs collaborent au travers d'un réseau ayant une latence inhérente. Bien évidemment, ce mécanisme ne peut pas s'appliquer à tous les objets étant donné qu'ils n'ont pas tous un comportement prévisible. En particulier, il est difficile de prédire les actions des utilisateurs. Par exemple, PaRADE utilise ce mode de gestion pour la détection de collision.

Les évènements sont prédits localement, marqués avec un *timestamp* et envoyés aux autres utilisateurs de la session où ils seront exécutés au moment approprié (défini par le *timestamp*). Pour que la gestion prédictive ait un intérêt, il faut que la durée entre l'envoi de l'évènement prédit et son traitement soit supérieure à la latence réseau. Sinon le message arrivera après le moment où il doit être traité. Si la prédiction s'avère fausse, le système doit mettre en œuvre un retour arrière afin de corriger les erreurs. Pour minimiser ces retours arrière, PaRADE propose de transmettre les évènements prédits juste à temps grâce à une estimation de la latence réseau calculée en étudiant le RTT (*round-trip delay time*) du réseau grâce à l'envoi de messages spécifiques.

Synchronisation en utilisant le serveur. Dans certaines architectures client/serveur, le serveur peut être utilisé pour gérer la synchronisation des évènements en plaçant une horloge logique sur le serveur. C'est le cas dans ShareX3D [Jourdain *et al.*, 2008] où le serveur maintient un *timestamp* pour le point de vue des utilisateurs (ils partagent tous le même point de vue sur la scène). Lorsque le serveur reçoit un message de mise-à-jour du point de vue, il incrémente le *timestamp*. Ainsi, lorsqu'un utilisateur demande s'il existe un point de vue plus récent au serveur, il envoie dans sa requête la valeur de son *timestamp* local. Le serveur compare alors son *timestamp* avec celui de l'utilisateur qui fait la requête :

- Si le *timestamp* de l'utilisateur est égal à celui du serveur, alors le point de vue de l'utilisateur est à jour et sa requête est mise en attente.
- Si le *timestamp* de l'utilisateur est plus ancien que celui du serveur, alors le serveur envoie le dernier point de vue à l'utilisateur et la valeur du *timestamp* correspondant.

Cette solution permet de s'assurer de façon simple que les utilisateurs ont bien le point de vue le plus récent. Par contre, elle ne permet pas de s'assurer que tous les évènements soient traités par chaque nœud. En effet, un nœud traite le dernier évènement reçu par le serveur et non chacun des évènements reçus dans l'ordre. Cela n'est pas gênant dans ShareX3D car le serveur envoie directement la position absolue du point de vue à l'utilisateur. Ce dernier n'a donc pas besoin des positions précédentes pour placer le point de vue.

2.1.4.2 Gestion de la concurrence

Seuls les systèmes qui utilisent le mode de distribution des données centralisé (les données peuvent être modifiées uniquement sur le serveur) ou ceux qui utilisent un système de référentiel et de miroirs comme OpenMASK [Margery *et al.*, 2002] (seul le référentiel peut être modifié par les utilisateurs) garantissent la cohérence de l'environnement virtuel lors d'accès concurrents. À partir du moment où les informations sont réparties sur les nœuds de chacun des utilisateurs (comme dans le cas du mode de distribution homogènement répliqué), ces utilisateurs peuvent accéder et modifier localement les données avant que les modifications ne soient transmises aux autres utilisateurs. Cela peut entraîner des incohérences dans l'environnement virtuel et il est nécessaire de gérer les accès concurrents des utilisateurs aux données. Pour gérer les accès concurrents aux données lorsque seul un utilisateur peut modifier un objet virtuel en même temps (interactions non collaboratives), [Lee *et al.*, 2007] distinguent trois modes de contrôle de la concurrence :

Mode pessimiste. Comme dans BrickNet [Singh *et al.*, 1995], ce mode garantit que seul un utilisateur modifie les paramètres d'un objet à la fois grâce à un système de blocage. Lorsqu'un utilisateur veut manipuler un objet, il demande à en devenir le propriétaire. Un objet ne peut avoir qu'un seul propriétaire à la fois. Seul le propriétaire courant d'un objet peut modifier les paramètres de cet objet. Cette solution permet d'écarter tout problème de modifications concurrentes. Cependant, lorsque la latence réseau est importante ou que le nombre d'utilisateurs est grand, le temps nécessaire pour acquérir la propriété d'un objet peut être important et donc entrainer des interactions moins réactives. Par exemple, dans le cas d'une architecture pair-à-pair, l'utilisateur qui veut interagir avec un objet doit envoyer sa demande à tous les autres utilisateurs et attendre leur réponse pour être sûr qu'aucun autre utilisateur n'est déjà le propriétaire de cet objet.

Mode optimiste. Ce mode permet aux utilisateurs de modifier les objets sans vérifier l'apparition d'éventuels conflits. Ainsi, les utilisateurs peuvent interagir d'une façon plus naturelle avec les objets. Cependant, en cas de conflit, il est nécessaire d'effectuer une « réparation ». Celle-ci peut être complexe et impose aussi à certains utilisateurs d'effectuer à nouveau leur action, ce qui peut être assez fastidieux si cela se produit souvent.

Mode basé sur la prédiction. Comme dans PaRADE [Roberts et Sharkey, 1997] ou ATLAS [Lee *et al.*, 2007], ce mode se situe entre le mode pessimiste et le mode optimiste. Il consiste à prédire pour chaque objet quels sont les utilisateurs qui peuvent potentiellement le manipuler, et à établir des priorités entre ces propriétaires potentiels. Dès que la prédiction s'avère fausse pour un utilisateur, celui-ci donne la propriété de l'objet à l'utilisateur suivant dans la liste de propriétaires potentiels. Cette prédiction est généralement basée sur la position et le comportement des utilisateurs (vers où il se déplace, vers où il regarde, etc.).

2.1.4.3 Réduction des communications

Pour permettre des interactions performantes et une bonne cohérence de l'environnement virtuel dès que le nombre d'utilisateurs devient important ou que les contraintes réseaux sont fortes, il est important de mettre en place des techniques pour réduire au

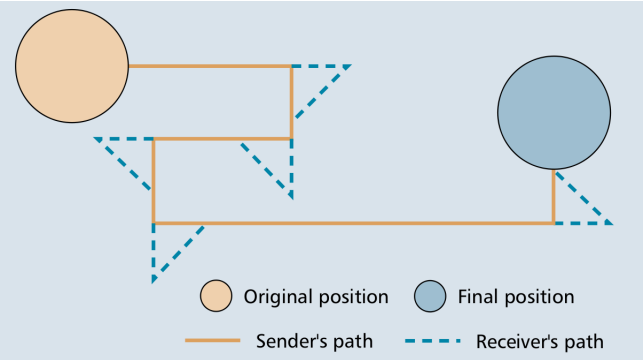


FIGURE 2.10 – Un exemple de *dead-reckoning* [Joslin *et al.*, 2004].

maximum les données transmises sur le réseau afin de surmonter ces contraintes matérielles. En effet, le fait d'optimiser les communications réseaux peut permettre de contrer les effets négatifs de la latence réseau et des faibles capacités en termes de bande passante.

Dead-Reckoning Les techniques de *dead-reckoning*, proposées dans SIMNET [Calvin *et al.*, 1993], puis dans NPSNET [Macedonia *et al.*, 1994], permettent de réduire le nombre de messages transitant sur le réseau pour mettre à jour les états des objets virtuels sur les différents nœuds de la session. Le *dead-reckoning* est basé sur : une formule de prédiction, un seuil d'erreur, et une formule de convergence. Au lieu d'envoyer et de recevoir à chaque pas de temps de simulation des messages de mise à jour pour décrire le mouvement de tous les objets virtuels, la formule de prédiction est utilisée pour estimer leur trajectoire. Le calcul de cette formule est réalisé pour chaque objet sur chaque nœud de la session grâce aux dernières informations reçues (comme, par exemple, la vitesse et la position de l'objet). Chaque nœud de la session gère une copie « fantôme » de l'objet afin de comparer son état courant et la prédiction qui est faite par les autres. Lorsqu'un évènement survient sur un nœud pour un objet particulier (action de l'utilisateur, collision, etc.), les autres utilisateurs ne peuvent plus prédire fidèlement l'état de cet objet. Dans ce cas, le seuil d'erreur entre l'état courant et la prédiction est atteint sur ce nœud. Le nœud envoie alors des messages de re-mise à jour pour cet objet à tous les autres utilisateurs. Les autres utilisateurs utilisent la formule de convergence pour recaler leur version de l'objet avec les nouvelles informations qu'ils viennent de recevoir (cf. figure 2.10).

Filtrage des données Toujours dans l'optique de réduire le nombre de messages transmis sur le réseau, il est possible d'utiliser des techniques de filtrage des données. Ces techniques consistent à essayer d'envoyer les messages seulement aux utilisateurs concernés par les informations contenues dans les messages. Elles permettent aussi de réduire le nombre de messages reçus par les utilisateurs et donc le temps de calcul nécessaire pour les traiter.

Ces techniques de filtrage utilisent traditionnellement des zones d'intérêt *(area of interest)*. La gestion de ces zones d'intérêt a pour objectif de réduire le nombre des destinataires des messages de mise à jour des objets virtuels. Au lieu de partager tous les éléments d'un environnement virtuel entre tous les utilisateurs, ce partage est réduit à un sous-ensemble d'objets en tenant compte des intérêts de chaque utilisateur. Les objets partagés seront filtrés selon des critères spatiaux ou fonctionnels. Par exemple, un utilisateur peut n'être

intéressé que par les objets situés à proximité de lui [Waters *et al.*, 1997] ou dans son champ de vision [Funkhouser, 1995]. Ainsi, lorsqu'un utilisateur modifie un objet, il envoie des messages de mise à jour seulement aux autres utilisateurs présents dans la même zone d'intérêt. D'un point de vue technique, cette diffusion restreinte des messages est généralement effectuée grâce à l'utilisation d'un serveur qui est capable de savoir à quelles zones d'intérêt les utilisateurs appartiennent ou par l'abonnement des utilisateurs à un groupe de diffusion *multicast* lorsqu'ils entrent dans une nouvelle zone d'intérêt.

Migration La migration consiste à faire migrer le processus qui exécute le comportement d'un objet d'un nœud à un autre. Dans la plupart des plateformes qui utilisent cette technique, le but est de mieux répartir les charges de calcul sur les différents nœuds participant à la session. Cependant, dans les plateformes qui utilisent un système de référentiel et de miroirs comme dans OpenMASK [Margery *et al.*, 2002], cette technique peut être utilisée pour permettre des interactions plus performantes lorsque la latence réseau est importante. En effet, elle permet de faire migrer le référentiel sur le nœud de l'utilisateur qui interagit [Duval et Zammar, 2006] afin d'éviter de devoir envoyer des messages de modification au nœud qui possède le référentiel et d'attendre les messages de mise à jour en retour pour recevoir les résultats des actions de l'utilisateur. Après migration, cet utilisateur peut interagir directement en local avec le référentiel avant d'envoyer des mises à jour aux autres utilisateurs. Cela permet de réduire le nombre de messages envoyés sur le réseau et de rendre les interactions plus réactives.

Compression des données Les messages de mise à jour sont souvent de simples vecteurs ou des commandes. Ces commandes peuvent parfois être encodées avec un codage d'Huffman, mais en règle générale la compression est rarement utilisée lors de la transmission des messages de mise à jour sur le réseau [Joslin *et al.*, 2004]. Cependant, avec l'apparition d'interactions de plus en plus complexes comme, par exemple, les mouvements d'un humain virtuel (avec jusqu'à 186 degrés de liberté différents), les messages de mise à jour peuvent être de taille plus importante et peuvent nécessiter de la compression.

Par ailleurs, dès qu'un mode de distribution des données partiellement répliqué est utilisé, il peut être nécessaire de transmettre de nouvelles données (textures, géométries, etc.) à un utilisateur qui en a besoin. Dans ce cas, la compression des données et la façon de les transmettre (niveau de détails, etc.) sont importantes pour ne pas trop perturber les interactions des utilisateurs lorsque des données supplémentaires sont rechargées.

Agrégation de paquets Au lieu d'envoyer directement un paquet sur le réseau pour chaque message à transmettre, cette solution propose de regrouper plusieurs messages dans une même unité de données plus grande comme présenté par [Delaney *et al.*, 2006b]. Cela permet de réduire la taille des données transmises car il est possible de regrouper les informations redondantes (identifiant de l'objet, *timestamp*, etc.). Cela permet surtout de réduire le nombre de messages à transmettre : lorsque le réseau limite le nombre de messages transmis par unité de temps, il est ainsi possible d'envoyer plus rapidement les informations. Cependant, cette méthode peut introduire un peu de latence (certains messages sont forcément retardés) et elle peut poser des problèmes de reconstruction des messages (par exemple, lorsque certains messages ne concernent pas tous les utilisateurs).

2.2 Architecture logicielle d'une application collaborative

Comme nous l'avons présenté au chapitre 1, nous souhaitons intégrer dans un même environnement virtuel des utilisateurs avec différents dispositifs matériels, mais aussi avec différents composants logiciels associés à ces dispositifs (librairies graphiques, sonores, etc.). Même s'il existe de nombreuses solutions pour concevoir une application de réalité virtuelle en faisant abstraction des dispositifs matériels (cf. partie 1.1.3.1), il n'existe pas à notre connaissance de système de réalité virtuelle qui permette de modéliser un environnement virtuel en le séparant des composants logiciels utilisés pour le restituer aux utilisateurs. Cependant, dans le domaine des interactions homme-machine (IHM [1]), de nombreux modèles d'architecture logicielle proposent de séparer la partie principale de l'application de la partie qui fait l'interface avec les utilisateurs. Nous présentons ces modèles dans la partie 2.2.1. Dans le cas d'une application collaborative distante, l'architecture logicielle doit également prendre en compte la dimension multi-utilisateur et la distribution sur le réseau des données de l'application. Dans la partie 2.2.2, nous étudions donc comment ces modèles d'architecture logicielle ont été adaptés pour la collaboration dans le domaine du travail coopératif assisté par ordinateur (TCAO [2]).

2.2.1 Architecture logicielle d'un système interactif

Les systèmes interactifs nécessitent de modéliser la structure principale et les fonctionnalités d'une application indépendamment de la façon dont les utilisateurs vont interagir avec cette application. Cela permet d'adapter l'interface utilisateur en fonction des dispositifs matériels utilisés tout en gardant le même cœur applicatif. Pour séparer la partie principale de l'application et l'interface utilisateur, les modèles logiciels existants dans le domaine de l'IHM choisissent généralement de diviser les applications en trois parties : la partie abstraite de l'application, la représentation qui en est faite aux utilisateurs et une partie qui fait le lien entre ces deux parties. Nous avons choisi de présenter certains modèles couramment utilisés qui illustrent les différentes approches pour mettre en œuvre cette division. Certains modèles proposent de décomposer l'application en plusieurs composants fonctionnels comme le modèle Arch présenté dans la partie 2.2.1.1. D'autres modèles décrivent le système comme un ensemble d'agents, c'est le cas des modèles MVC et PAC détaillés dans la partie 2.2.1.2. Enfin, il existe des approches hybrides comme le modèle PAC-Amodeus décrit dans la partie 2.2.1.3.

2.2.1.1 Modèles basés sur une décomposition fonctionnelle

Le modèle Arch [UIMS, 1992] décompose une application interactive en cinq composants fonctionnels (cf. figure 2.11). Nous retrouvons les trois parties citées précédemment :

- le *Domaine spécifique* correspond à l'abstraction de l'application,
- la *Boîte à outils d'interaction* est la partie en contact direct avec l'utilisateur,
- le *Contrôleur de dialogue* assure le lien entre ces deux parties.

Pour assurer une meilleure indépendance entre ces trois composants, le modèle Arch ajoute en plus des adaptateurs entre chacun des composants : l'*Adaptateur de domaine* et la *Présentation*. Cette indépendance entre les composants permet de changer facilement d'interface utilisateur ou de modifier le cœur applicatif sans impacter le reste.

1. en anglais : *Human-Computer Interaction (HCI)*
2. en anglais : *Computer Supported Cooperative Work (CSCW)*

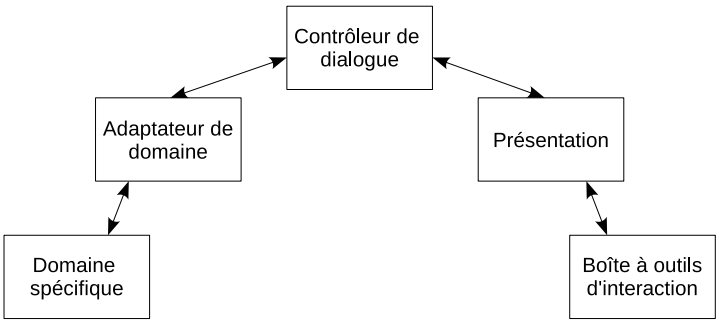


FIGURE 2.11 – Modèle Arch.

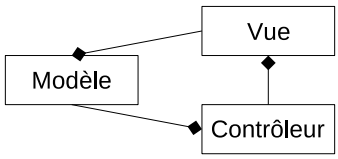


FIGURE 2.12 – Modèle MVC.

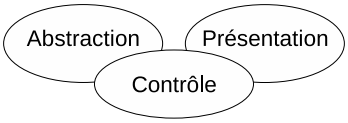


FIGURE 2.13 – Modèle PAC.

2.2.1.2 Modèles multi-agents

Les modèles multi-agents structurent un système interactif comme un ensemble d'unités de traitement appelées agents. Ce concept d'agents décompose l'application à un niveau plus fin afin d'obtenir une application plus modulaire qui peut être facilement modifiée, parallélisée et distribuée. Le modèle MVC *(Modèle-Vue-Contrôleur)* [Reenskaug, 1979][Goldberg, 1990] choisit de diviser chaque agent en trois parties (cf. figure 2.12) :

- le *Modèle* représente les données, ainsi que les règles pour y accéder,
- la *Vue* définit la façon de présenter les données du *Modèle* aux utilisateurs,
- le *Contrôleur* traduit les interactions des utilisateurs effectuées dans la *Vue* sous forme d'actions qui sont ensuite envoyées au *Modèle*.

Cependant, le modèle MVC n'impose pas de séparation formelle entre le *Modèle* et la *Vue*, et ces deux composants sont parfois fortement couplés.

Le modèle PAC *(Présentation-Abstraction-Contrôle)* [Coutaz, 1987] est également basé sur un concept d'agents constitués chacun de trois facettes (cf. figure 2.13). Au premier abord, ce modèle peut sembler similaire au modèle MVC avec la *Présentation* qui joue le rôle de la *Vue*, l'*Abstraction* qui joue le rôle du *Modèle* et le *Contrôle* qui joue le rôle du *Contrôleur*. Cependant, le comportement de chacune de ces facettes est légèrement différent du comportement de leur équivalent dans le modèle MVC :

- l'*Abstraction* correspond à la sémantique et aux fonctionnalités de l'application,
- la *Présentation* décrit l'interface avec les utilisateurs, c'est-à-dire les entrées et les sorties de l'application telles qu'elles sont perçues par les utilisateurs,
- le *Contrôle* effectue le lien et assure la cohérence entre la facette d'*Abstraction* et la facette de *Présentation*.

De par la structure du modèle PAC, l'*Abstraction* et la *Présentation* ne peuvent pas communiquer directement. Par ailleurs, les agents PAC sont organisés de façon hiérarchique et chaque agent peut communiquer avec les autres uniquement au travers de leur *Contrôle*. Le *Contrôle* agit donc comme un médiateur qui filtre les communications entre l'*Abstraction* et la *Présentation*, mais aussi entre chacun des agents PAC.

2.2.1.3 Modèles hybrides

Le modèle PAC-Amodeus [Nigay et Coutaz, 1991] est un modèle hybride qui combine à la fois un découpage fonctionnel et une décomposition en agents. Il est basé à la fois

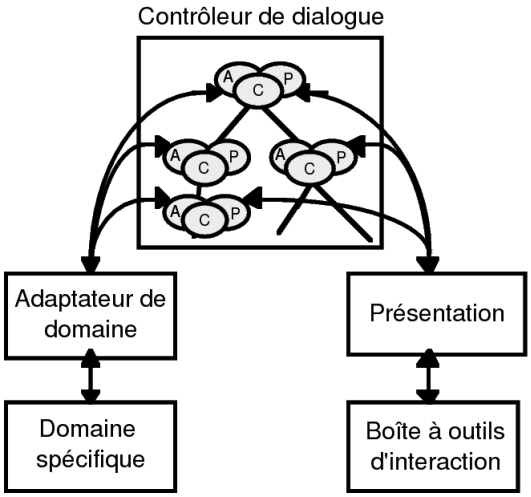


FIGURE 2.14 – Modèle PAC-Amodeus.

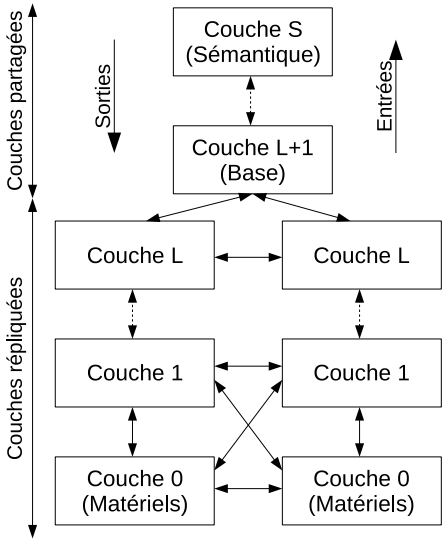


FIGURE 2.15 – Modèle de Dewan.

sur le modèle Arch et sur le modèle PAC : le *Contrôleur de dialogue* du modèle Arch est organisé en une hiérarchie d'agents PAC dont les *Abstractions* sont reliées à l'*Adaptateur de domaine* et les *Présentations* à la *Présentation* de Arch (cf. figure 2.14). Cette solution hybride permet de combiner les avantages des deux modèles.

2.2.2 Architecture logicielle d'un système collaboratif

Pour permettre à des utilisateurs de collaborer au travers d'un système interactif, il faut que le système soit multi-utilisateur. Cependant, interagir à plusieurs utilisateurs dans un même système pose de nombreux problèmes (cohérence, distribution réseau, etc.) et impose de revoir la conception de ce système. Dans le domaine du TCAO, de nouveaux modèles d'architecture logicielle ont donc été proposés pour adapter les différents modèles provenant de l'IHM à la collaboration : il existe des modèles basés sur des couches d'abstraction comme le modèle de Dewan décrit dans la partie 2.2.2.1, et des modèles basés sur des agents comme les modèles AVL et CoPAC présentés dans la partie 2.2.2.2. D'autres modèles proposent en plus de décrire les différentes fonctionnalités offertes aux utilisateurs pour la collaboration : dans la partie 2.2.2.3, nous détaillons le modèle de conception *clover* et la manière dont le modèle PAC* le met en œuvre. Enfin, il existe aussi des modèles hybrides pour la collaboration comme le modèle Clover que nous présentons dans la partie 2.2.2.4.

2.2.2.1 Modèles basés sur des couches d'abstraction

Certains modèles proposent de structurer un système interactif multi-utilisateur par un ensemble de couches d'abstraction en allant du niveau spécifique aux dispositifs matériels jusqu'au niveau spécifique à l'application. Le modèle de Dewan [Dewan, 1999] propose une architecture logicielle générique pour les systèmes multi-utilisateurs qui numérote ces couches en partant de 0 pour la couche matérielle jusqu'à S pour la couche sémantique spécifique à l'application (cf. figure 2.15). Ce modèle propose que les couches de base du système (de S à $L+1$) soient partagées entre tous les utilisateurs et que les couches

inférieures (de L à 0) soient répliquées dans des branches séparées pour chaque utilisateur. Ainsi chaque utilisateur peut adapter le système en fonction des dispositifs matériels qu'il utilise, mais il peut aussi avoir des couches logicielles propres qui lui permettent d'avoir des fonctionnalités personnalisées. Les flots d'informations en entrée et en sortie sont répercutés verticalement entre les couches (de 0 à S pour les entrées et dans le sens inverse pour les sorties) et horizontalement entre les couches équivalentes lorsqu'elles sont répliquées dans des branches différentes.

Le modèle de Dewan peut être utilisé pour implémenter une version multi-utilisateur du modèle Arch. Pour cela, le *Domaine spécifique* de Arch est partagé entre tous les utilisateurs et les autres composants de Arch sont répliqués dans des branches séparées pour chacun des utilisateurs.

2.2.2.2 Modèles basés sur des agents

D'autres modèles proposent d'étendre les modèles multi-agents comme MVC ou PAC afin de répondre aux besoins des systèmes multi-utilisateurs en gardant la décomposition de l'application en un ensemble d'agents. Le modèle ALV *(Abstraction-Lien-Vue)* [Hill, 1992] vise les systèmes construits autour d'un noyau fonctionnel centralisé. Pour chaque agent, nous retrouvons un composant central *(Abstraction)* qui est partagé entre tous les utilisateurs. Chaque utilisateur dispose d'une interface propre *(Vue)* qui est adaptée à son rôle. Des composants de liaison *(Lien)* assurent la cohérence entre la *Vue* qui leur est associée et l'*Abstraction* partagée en propageant les modifications dans les deux sens. Ces *Liens* permettent de garantir l'indépendance entre l'*Abstraction* partagée et les différentes *Vues*. Cependant, le modèle ALV est principalement pertinent dans le cas d'une architecture réseau centralisée, c'est-à-dire lorsqu'un serveur gère le noyau fonctionnel.

Par ailleurs, le modèle CoPAC [Salber, 1995] propose d'adapter le modèle PAC-Amodeus pour la collaboration en ajoutant une facette de *Communication* à chacun des agents PAC utilisés dans le *Contrôleur de dialogue*. Cette facette de *Communication* se situe exactement sur le même plan que les facettes d'*Abstraction* et de *Présentation*. Le *Contrôle* assure donc la communication entre ces trois facettes. La facette de *Communication* permet de propager et de recevoir les informations relatives à cet agent sur le réseau. Cependant, le modèle CoPAC ne spécifie pas comment ces informations sont transmises sur le réseau et comment la cohérence entre les différentes versions de chaque agent PAC est maintenue entre les différents sites.

2.2.2.3 Modèles basés sur une description fonctionnelle de la collaboration

Le modèle de conception *clover* [Calvary *et al.*, 1997] propose de décomposer les fonctionnalités spécifiques à la collaboration en trois domaines (cf. figure 2.16) :

- l'espace de production correspond à l'ensemble des objets que les utilisateurs élaborent et manipulent ensemble,
- l'espace de coordination maintient les relations temporelles et la cohérence de ces objets en fonction des activités des utilisateurs,
- l'espace de communication permet une communication au sens échange d'informations entre les utilisateurs.

Ce modèle de conception est mis en œuvre dans le modèle PAC* [Calvary *et al.*, 1997] qui est une extension du modèle CoPAC : il est donc basé aussi sur le modèle PAC-Amodeus. Les trois domaines fonctionnels du modèle de conception *clover* sont répartis dans les trois

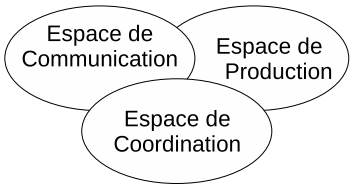


FIGURE 2.16 – Modèle de conception *clover*.

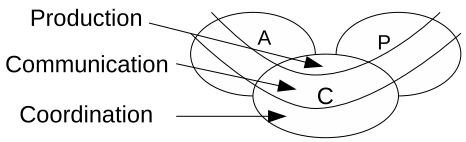


FIGURE 2.17 – Modèle PAC*.

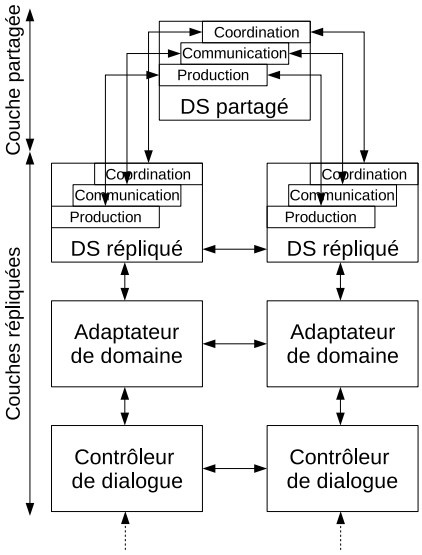


FIGURE 2.18 – Modèle Clover (la *Présentation* et la *Boîte à outils d'interaction* du modèle Arch situées sous le *Contrôleur de dialogue* ne sont pas représentées).

facettes de chacun des agents PAC (cf. figure 2.17). Le modèle PAC* a la particularité de pouvoir regrouper ensemble des agents PAC mono-utilisateurs et des agents PAC multi-utilisateurs (qui suivent le modèle de conception *clover*).

2.2.2.4 Modèles hybrides

Le modèle Clover [Laurillau et Nigay, 2002] est un modèle hybride à la fois basé sur des couches d'abstraction et sur une description fonctionnelle de la collaboration : il utilise une implémentation multi-utilisateur du modèle Arch basée sur le modèle de Dewan à laquelle viennent s'ajouter les trois domaines fonctionnels du modèle de conception *clover*. Le *Domaine spécifique (DS)* du modèle Arch est divisé en deux composants : le *DS partagé* et le *DS répliqué*. Ces deux composants implémentent chacun les trois domaines fonctionnels du modèle de conception *clover* (cf. figure 2.18). Mis à part le *DS partagé*, tous les composants du modèle Arch sont répliqués dans des branches séparées pour chacun des utilisateurs. Contrairement au modèle PAC* où le *Domaine spécifique* est complètement partagé entre les utilisateurs, le modèle Clover permet d'avoir à la fois un domaine fonctionnel partagé par tous les utilisateurs et un domaine fonctionnel privé à chaque utilisateur.

2.3 Synthèse

Concevoir un système capable de mettre en œuvre un environnement virtuel collaboratif impose de faire de nombreux choix techniques. D'une part, il faut choisir la manière de distribuer l'environnement virtuel sur le réseau entre les différents nœuds participant à une session collaborative. Ces choix de distribution impactent directement les performances du système en particulier au niveau de la cohérence entre les états de l'environnement virtuel que chaque utilisateur perçoit, ainsi qu'au niveau de la réactivité lors des interactions. D'autre part, il faut choisir une architecture logicielle pour modéliser l'environnement vir-

tuel. Ce choix d'architecture détermine la portabilité du système, c'est-à-dire sa capacité à s'adapter à la fois aux différents modes de distribution réseau, mais aussi aux différents composants logiciels utilisés pour faire l'interface avec les utilisateurs.

Choix de distribution sur le réseau

Pour distribuer un environnement virtuel collaboratif entre différents sites, il faut choisir une architecture réseau pour connecter les nœuds entre eux, mais aussi une façon de distribuer et de traiter les données relatives à l'environnement virtuel entre ces nœuds (cf. figure 2.19). Cependant, ces deux caractéristiques ne sont pas obligatoirement corrélées, et de plus en plus de solutions mélangent les choix d'architecture réseau avec les choix de distribution des données (cf. table 2.1). Même si certains mécanismes permettent d'améliorer la cohérence de l'environnement virtuel, les performances du système au niveau de la cohérence et de la réactivité sont principalement déterminées par les deux choix précédents. L'architecture réseau est souvent imposée par les contraintes techniques du système, mais l'idéal est de pouvoir mettre en place une architecture réseau hybride utilisant à la fois des serveurs et des communications pair-à-pair entre les nœuds. Par contre, la distribution des données peut être choisie plus librement lors de la conception d'un système.

Les différents modes de distribution des données ont leurs propres avantages en termes de cohérence, de réactivité, de quantité de traitements à effectuer sur chacun des nœuds, et de nombre de communications réseaux entre les nœuds. Ainsi, chaque mode de distribution permet d'obtenir un compromis particulier entre la cohérence de l'environnement virtuel et la réactivité du système. Généralement, les systèmes existants choisissent le mode de distribution qui correspond le mieux aux besoins de l'application qu'ils souhaitent mettre en œuvre. Mais il n'existe pas de solution générique qui puisse s'adapter aux besoins des différents types d'application. De plus, il serait intéressant de pouvoir aussi choisir le compromis cohérence/réactivité en fonction du rôle de chaque objet dans le monde virtuel et des tâches que les utilisateurs sont en train d'effectuer. Ces besoins de choisir la distribution des données indépendamment pour chaque objet et de pouvoir la modifier de façon dynamique pendant la session sont étudiés dans le chapitre 3

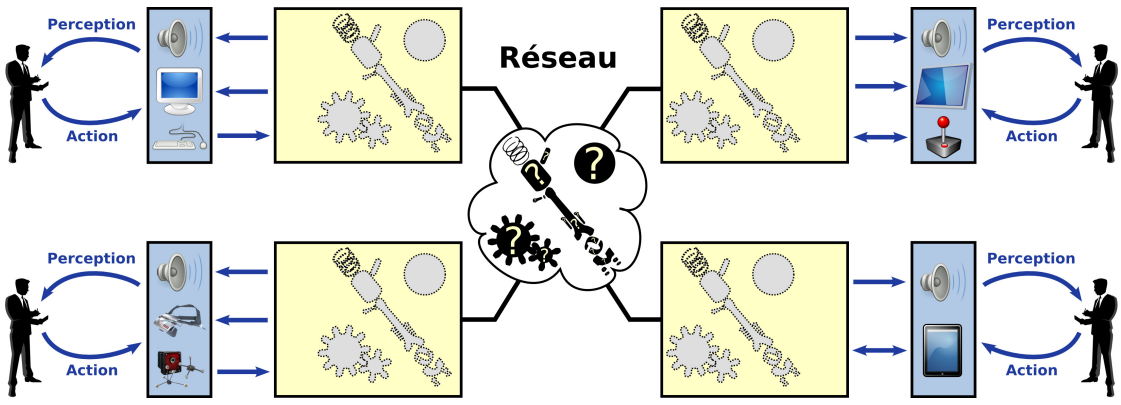


FIGURE 2.19 – Distribuer les données de l'environnement virtuel sur les différents nœuds.

Environnement virtuel collaboratif	Architecture réseau	Distribution des données
VISTEL [Yoshida *et al.*, 1995]	client/serveur	centralisée
RING [Funkhouser, 1995]	client/serveur	homogènement répliquée
BrickNet [Singh *et al.*, 1995]	client/serveur	partialement répliquée
ShareX3D [Jourdain *et al.*, 2008]	client/serveur	homogènement répliquée
SPIN-3D [Dit Picard *et al.*, 2002]	pair-à-pair	homogènement répliquée
DIVE [Frécon et Stenius, 1998]	pair-à-pair (+ serveur de sauvegarde)	partialement répliquée
PaRADE [Roberts et Sharkey, 1997]	hybride	homogènement répliquée
[Anthes *et al.*, 2004]	hybride	homogènement répliquée
OpenMASK [Margery *et al.*, 2002]	hybride	partialement répliquée (utilise le paradigme de référentiel/miroirs pour l'exécution du comportement des objets)
SPLINE [Waters *et al.*, 1997]	hybride	partialement répliquée
ATLAS [Lee *et al.*, 2007]	hybride	partialement répliquée

TABLE 2.1 – Caractéristiques de certains environnements virtuels distribués.

Choix de l'architecture logicielle du système

À notre connaissance, il n'existe pas de modèle d'architecture logicielle spécifique pour les environnements virtuels collaboratifs distribués. Cependant, il existe de nombreuses solutions dans le domaine des interactions homme-machine pour modéliser un système interactif indépendamment de la manière d'interfacer les utilisateurs avec ce système. Ces modèles permettent d'adapter une application aux différents dispositifs matériels et composants logiciels particuliers utilisés par les utilisateurs. Ces modèles ont ensuite été adaptés

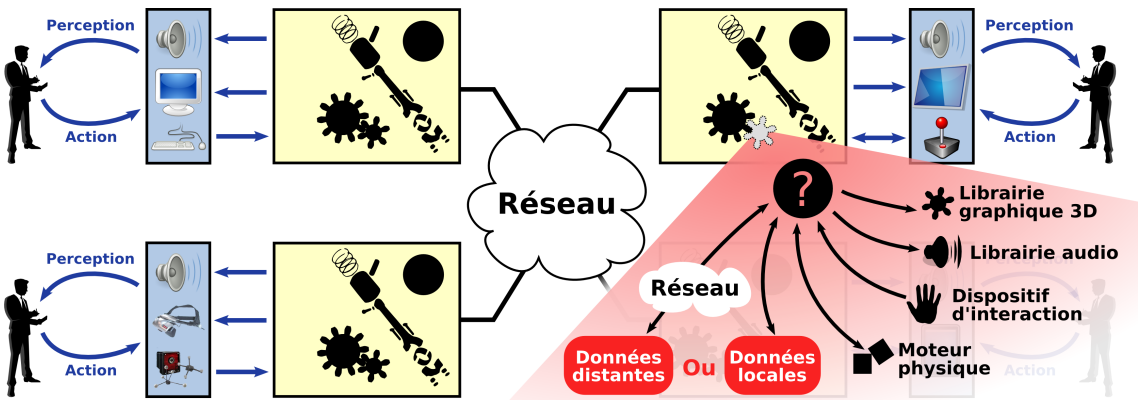


FIGURE 2.20 – Modéliser l'environnement virtuel indépendamment de la distribution réseau et des composants logiciels spécialisés.

aux systèmes multi-utilisateurs dans le domaine du travail coopératif assisté par ordinateur pour permettre à plusieurs utilisateurs de partager les mêmes données afin de pouvoir collaborer à distance.

Ces modèles peuvent servir de base pour définir un nouveau modèle pour mettre en œuvre un environnement virtuel collaboratif. Ce nouveau modèle doit permettre de concevoir l'environnement virtuel indépendamment de l'architecture réseau et de la distribution des données. De plus, il doit être capable de s'adapter aux différents systèmes matériels et logiciels que les utilisateurs peuvent être amenés à utiliser (cf. figure 2.20). Les modèles basés sur des agents comme le modèle ALV [Hill, 1992] ou CoPAC [Salber, 1995] nous semblent particulièrement bien adaptés pour mettre en œuvre un environnement virtuel collaboratif. En effet, chaque objet du monde virtuel peut être modélisé par un agent ce qui permettrait d'adapter la distribution réseau indépendamment pour chaque objet. Cependant, le modèle ALV peut difficilement s'adapter à plusieurs modes de distribution puisqu'il est basé sur un *Noyau Fonctionnel* centralisé. Par ailleurs, le modèle CoPAC ne décrit ni la manière de communiquer entre les agents, ni la manière de distribuer les données sur le réseau. Ce besoin de définir un nouveau modèle d'architecture logicielle adaptée aux environnements virtuels collaboratifs distribués est traité dans le chapitre 4.

Chapitre 3

Modèle d'adaptation dynamique de la distribution des données

Chacun des trois modes de distribution des données (centralisé, répliqué, hybride) présentés dans le chapitre 2 permet d'atteindre un compromis spécifique entre la cohérence de l'environnement virtuel et la réactivité du système. Les systèmes existants choisissent le mode de distribution de façon à obtenir un compromis adéquat en fonction des besoins de l'application et des contraintes techniques. Cependant, même si chaque mode de distribution a ses propres avantages, il n'existe pas de solution générique qui réponde aux besoins de tous les environnements virtuels distribués. Si nous voulons que l'environnement virtuel collaboratif puisse être utilisé avec différents types d'applications et des contraintes réseaux fortes, il est difficile de choisir un mode de distribution en particulier (cf. figure 3.1). En effet, les contraintes réseaux accentuent les différences entre chacun des modes, et chaque type d'applications impose des besoins particuliers. Par exemple, une application où les utilisateurs ne font qu'observer des données nécessite une forte cohérence, mais pas une réactivité importante. À l'inverse, une application où des utilisateurs manipulent séparément des objets virtuels exige une bonne réactivité, mais pas forcément une forte cohérence. Au sein d'une même application, le rôle des objets virtuels ou les tâches effectuées par les utilisateurs peuvent également nécessiter d'ajuster le compromis entre cohérence et réactivité.

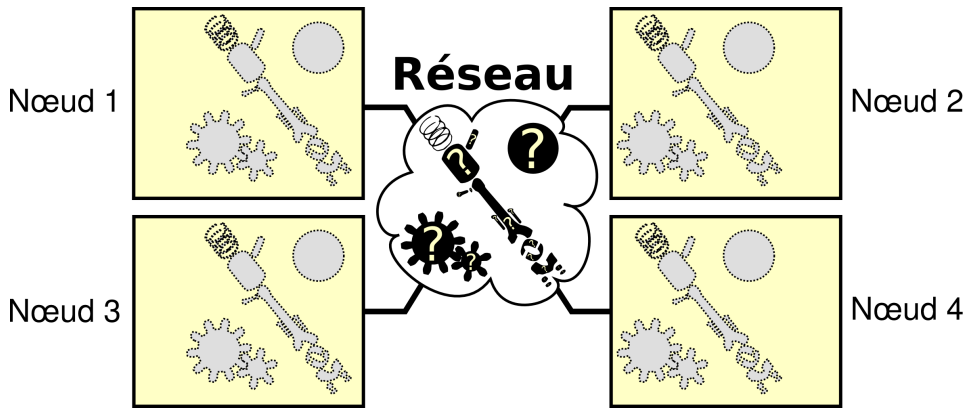


FIGURE 3.1 – Comment distribuer les données d'un environnement virtuel sur le réseau ?

Dans ce chapitre, nous proposons un modèle pour concevoir les environnements virtuels distribués qui permet d'adapter la distribution des données indépendamment pour chaque objet virtuel et de façon dynamique durant la session [Fleury *et al.*, 2010c]. Ce modèle est basé sur un paradigme de référents et de proxys, présenté dans le partie 3.1. Ce paradigme permet de mettre en œuvre les trois modes de distribution (partie 3.2), de choisir un mode particulier pour chaque objet virtuel (partie 3.3) et de modifier dynamiquement le mode choisi durant la session (partie 3.4). Enfin, dans la partie 3.5, nous présentons l'instanciation de ce modèle dans le *framework* Collaviz avec une architecture réseau de type client/serveur.

3.1 Paradigme de référents et de proxys

Notre modèle d'adaptation dynamique de la distribution des données est basé sur un paradigme de référents et de proxys. Sur chacun des nœuds participant à la session collaborative, chaque objet virtuel est représenté soit par un référent, soit par un proxy :

- **Un référent** effectue les tâches suivantes :
  - il stocke les données relatives à l'objet,
  - il exécute le comportement de l'objet,
  - il traite les demandes de modification de l'objet provenant des utilisateurs,
  - il envoie des messages de mise à jour aux proxys qui lui sont associés.
- **Un proxy** met simplement à jour les représentations virtuelles de l'objet (visuelles, sonores, physiques, etc.) lorsqu'il reçoit des messages de mise à jour provenant d'un référent. Un proxy n'exécute aucun autre traitement localement et il se contente de transmettre à son référent les demandes de modification provenant des utilisateurs.

Cependant, à la différence du paradigme de référentiel/miroirs utilisé dans OpenMASK [Margery *et al.*, 2002], il peut y avoir pour un même objet soit un seul référent, soit des référents sur tous les nœuds afin de pouvoir implémenter plusieurs modes de distribution des données. Lorsqu'un même objet possède plusieurs référents, il faut en plus gérer les modifications concurrentes faites par ces différents référents. Nous avons également mis en place un mécanisme de migration permettant de transformer un proxy en référent, et inversement. Cela permet de modifier la position du ou des référent(s) dans un même mode de distribution, mais cela permet aussi de changer facilement le mode de distribution de l'objet. Pour faciliter la migration, les proxys peuvent stocker, en plus, une copie locale des données de l'objet à partir des informations qu'ils reçoivent dans les messages de mise à jour. Ainsi, les proxys possèdent toujours l'état courant de l'objet et il est facile de les transformer en référent lorsqu'il est nécessaire. Par contre, cela impose une duplication des données relatives à l'objet sur tous les nœuds participant à la session.

3.2 Trois modes de distribution des données

En fonction de la localisation du ou des référent(s) sur les nœuds participant à la session collaborative, trois modes de distribution des données peuvent être implémentés sans modifier le code de l'application : le mode centralisé, le mode hybride et le mode répliqué.

Nous détaillons dans la partie 3.2.1, l'implémentation de chaque mode et les caractéristiques qui en découlent en termes de cohérence et de réactivité. Dans la partie 3.2.2, nous réalisons une synthèse quantitative des caractéristiques de chacun des modes.

3.2.1 Caractéristiques de chaque mode

Pour chacun des modes de distribution (***MD***), nous évaluons la cohérence de l'environnement virtuel et la réactivité lors des interactions en fonction de l'implémentation qui en est faite avec le paradigme de référents et de proxys. Pour cela, nous quantifions le décalage de cohérence (***DC***) et la latence lors des interactions (***LI***) par rapport à la latence réseau (***lat***). Pour effectuer cette évaluation quantitative, nous ne considérons que la latence réseau et pas le temps nécessaire pour traiter les objets virtuels sur chacun des nœuds afin de simplifier la comparaison entre les trois modes de distribution. Pour chaque mode, nous évaluons aussi le nombre de communications réseau (***CR***) et la quantité de traitements à effectuer sur chacun des nœuds (***QT***).

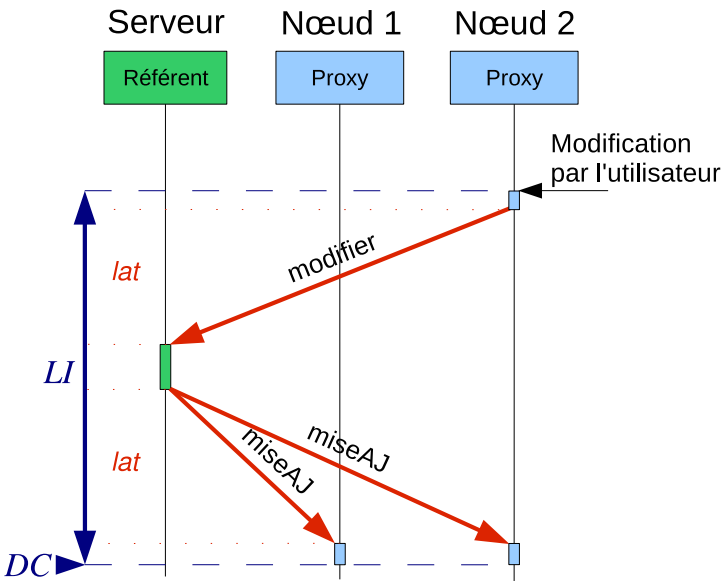


FIGURE 3.2 – Modification d'un objet avec le mode centralisé.

3.2.1.1 Mode centralisé

Pour mettre en œuvre le mode centralisé pour un objet virtuel, un seul référent est placé sur un serveur (ce mode impose d'avoir un serveur accessible pour chacun des nœuds). Tous les autres nœuds possèdent un proxy. Ainsi, toutes les données relatives à l'objet sont stockées et traitées sur le serveur. Les demandes de modification de l'objet sont toutes envoyées par les proxys au serveur, qui renvoie ensuite des mises à jour à tous les nœuds (cf. figure 3.2). La table 3.1 présente les différents avantages et inconvénients de ce mode. Nous pouvons remarquer que ce mode permet d'assurer principalement une bonne cohérence de l'environnement virtuel.

MD	**Avantages**	**Inconvénients**
DC	Quasiment nul si les nœuds ont des connexions similaires ou s'ils sont synchronisés entre eux (cf. partie 2.1.4.1).	
LI		Égale à deux fois ***lat***.
CR	Tous les traitements sont effectués sur le serveur (pas besoin d'une grande puissance de calcul sur les nœuds).	
QT		Beaucoup de communications réseaux (les modifications faites par les utilisateurs, mais aussi celles dues au comportement de l'objet, potentiellement à chaque pas de temps, doivent être transmises par le serveur à tous les nœuds).

TABLE 3.1 – Caractéristiques du mode centralisé.

3.2.1.2 Mode hybride

Pour mettre en œuvre le mode hybride pour un objet virtuel, un seul référent est placé sur un nœud particulier. Tous les autres nœuds possèdent un proxy pour cet objet. Ainsi, toutes les données relatives à l'objet sont stockées et traitées sur ce nœud particulier. Les demandes de modification sont toutes envoyées par les proxys à ce nœud, qui se charge

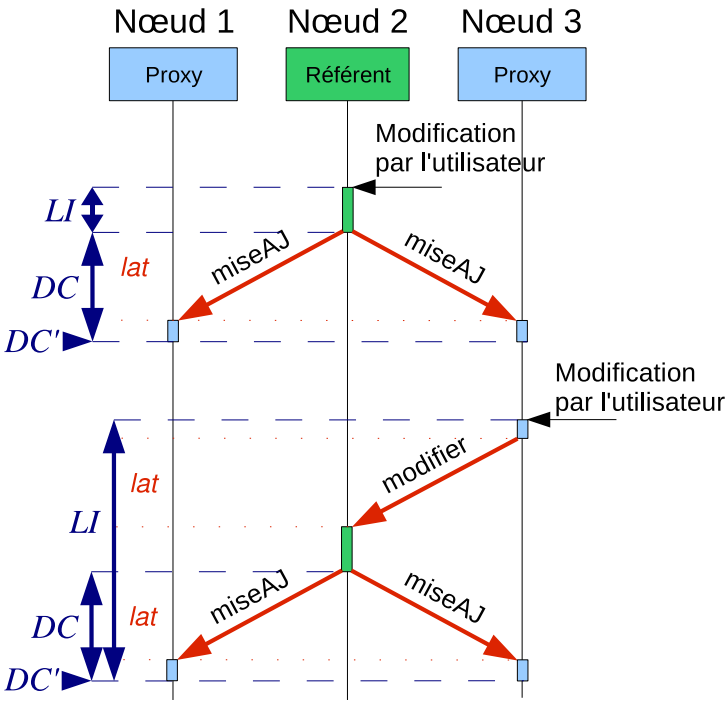


FIGURE 3.3 – Modification d'un objet avec le mode hybride.

ensuite de mettre à jour tous les autres nœuds qui participent à la session (cf. figure 3.3). Le nœud qui est le référent peut être changé dynamiquement durant la session collaborative en utilisant le processus de migration.

Si un serveur est connecté à la session (pour traiter les objets utilisant le mode centralisé par exemple), il peut aussi être utilisé dans ce mode. Dans ce cas, le serveur conserve un proxy pour l'objet en mode hybride afin de garder une copie à jour de l'objet sur le serveur. Cela permet de préparer le serveur pour un éventuel changement de mode, mais aussi d'assurer la persistance de l'environnement virtuel.

La table 3.2 présente les différents avantages et inconvénients de ce mode. Nous pouvons remarquer que ce mode permet d'obtenir un compromis entre cohérence et réactivité. Afin de tirer le meilleur parti de ce mode, le principe est de faire migrer le référent sur le nœud de l'utilisateur qui interagit avec l'objet. Il est possible d'obtenir ainsi une bonne réactivité pour cet utilisateur et une bonne cohérence entre les nœuds de tous les autres utilisateurs. Le cas où un utilisateur interagit avec un objet dont le référent est situé sur un nœud distant doit être évité au maximum.

MD	**Avantages**	**Inconvénients**
DC	Quasiment nul pour tous les nœuds qui ont des proxys s'ils ont des connexions similaires ou s'ils sont synchronisés entre eux (cf. partie 2.1.4.1).	Égal à ***lat*** pour le nœud qui possède le référent par rapport aux autres (il est en avance de ***lat***).
LI	Quasiment nulle pour le nœud qui possède le référent.	Égale à deux fois ***lat*** pour tous les nœuds qui ont des proxys.
CR	Tous les traitements sont localisés sur un nœud en particulier (on peut répartir les traitements en priorité sur les nœuds les plus puissants).	
QT		Beaucoup de communications réseaux (les modifications faites par les utilisateurs, mais aussi les changements dus au comportement de l'objet, potentiellement à chaque pas de temps, sont à transmettre par le référent à tous les autres nœuds).

TABLE 3.2 – Caractéristiques du mode hybride.

3.2.1.3 Mode répliqué

Pour mettre en œuvre le mode répliqué pour un objet virtuel, chaque nœud possède un référent pour l'objet. Dans ce cas, comme les référents n'ont pas de proxy associé, ils ont un comportement un peu particulier : ils n'envoient pas de mises à jour pour le comportement aux autres référents du même objet, mais ils leur envoient des mises à jour lorsque des modifications locales ont été faites par l'utilisateur. Ainsi, les données relatives à l'objet sont stockées et traitées localement sur chacun des nœuds. Pour que les nœuds aient tous le même état de l'objet, il faut les synchroniser entre eux afin d'obtenir la même

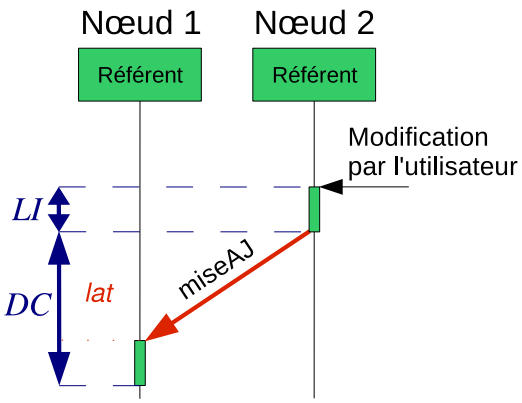


FIGURE 3.4 – Modification d'un objet avec le mode répliqué.

vitesse d'exécution du comportement de l'objet sur chaque nœud. Nous avons expliqué dans la partie 2.1.4.1 qu'il existe plusieurs solutions pour assurer cette synchronisation. Les demandes de modification sont aussi traitées localement sur le nœud de l'utilisateur qui interagit, puis ce nœud envoie une mise à jour à tous les autres nœuds (cf. figure 3.4). Ce fonctionnement impose de gérer les modifications concurrentes provenant des différents nœuds. En effet, avec ce mode, chaque utilisateur peut modifier l'objet en local sur son nœud et les conflits ne surviennent que lors de l'envoi des messages de mise à jour. Nous avons expliqué dans la partie 2.1.4.2 qu'il existe plusieurs politiques pour gérer les accès concurrents à un même objet.

Si un serveur est connecté à la session, il peut aussi être utilisé dans ce mode. Dans ce cas, le serveur possède aussi un référent pour l'objet et il peut faire évoluer ainsi sa propre version de l'objet. Si un mécanisme de synchronisation est utilisé, le serveur doit aussi être synchronisé avec les autres nœuds. Le serveur conserve ainsi une copie à jour de l'objet pour préparer un éventuel changement de mode ou pour assurer la persistance.

La table 3.3 présente les différents avantages et inconvénients de ce mode. Nous pouvons remarquer que ce mode permet principalement une bonne réactivité lors des interactions.

MD	**Avantages**	**Inconvénients**
DC		Égal à ***lat*** entre deux nœuds de la session.
LI	Quasiment nulle pour tous les utilisateurs.	
CR		Tous les traitements sont dupliqués sur chacun des nœuds.
QT	Peu de communications réseau (seules les modifications traitées localement sont transmises d'un nœud à tous les autres).	

TABLE 3.3 – Caractéristiques du mode répliqué.

3.2.2 Comparaison des trois modes

La table 3.4 récapitule les valeurs de décalage de cohérence (***DC***) entre les nœuds et de latence lors des interactions (***LI***) quantifiées en fonction de la latence réseau (***lat***). Seul le mode centralisé permet une forte cohérence entre tous les nœuds, tandis que le mode répliqué permet une bonne réactivité. Si le référent est situé sur le nœud de l'utilisateur qui interagit, le mode hybride permet d'obtenir le meilleur compromis.

Mode de distribution	***LI***	***DC***
Centralisé	2 ***lat***	0
Hybride *Interaction avec le référent*	0	***lat*** *(référent ↔ proxys)* 0 *(proxy ↔ proxy)*
Hybride *Interaction avec un proxy*	2 ***lat***	***lat*** *(référent ↔ proxys)* 0 *(proxy ↔ proxy)*
Répliqué	0	***lat***

TABLE 3.4 – ***LI*** et ***DC*** pour chacun des modes de distribution des données.

3.3 Adaptation spécifique à chaque objet

Contrairement aux systèmes existants où les données relatives aux objets virtuels sont distribuées de la même façon, notre modèle permet de choisir un mode de distribution de façon indépendante pour chaque objet. En effet, le paradigme de référents et de proxys offre la possibilité de définir séparément, pour chaque objet virtuel, si tous les nœuds doivent posséder un référent (mode répliqué) ou si seulement un nœud particulier doit posséder un référent. Dans ce second cas, il est aussi possible de choisir séparément pour chaque objet quel nœud doit posséder le référent et donc quels nœuds auront un proxy. Notre modèle assure également l'« interopérabilité » entre les objets virtuels en permettant à des objets qui ne sont pas forcément dans le même mode de distribution de communiquer ensemble. Les aspects techniques de cette « interopérabilité » seront détaillés dans le chapitre 4.

Dans un environnement virtuel, tous les objets n'ont pas le même rôle et les mêmes capacités d'interaction. Chaque objet virtuel a des besoins spécifiques en termes de cohérence et de réactivité. Le mode de distribution des données d'un objet peut alors être choisi en fonction de son rôle dans le monde virtuel. Par exemple, un objet manipulé par un seul utilisateur à la fois nécessite une forte réactivité, il peut être traité localement sur le nœud de cet utilisateur (mode hybride). Néanmoins, si cet objet est un pointeur 3D utilisé pour montrer des éléments intéressants du monde virtuel à d'autres utilisateurs, il nécessite une forte cohérence afin que tous les utilisateurs le voient pointer au même endroit, au même moment. Il peut être plus judicieux de traiter cet objet sur le serveur afin d'assurer la meilleure cohérence possible (mode centralisé). Enfin, nous pouvons prendre l'exemple d'objets « animés », c'est-à-dire qui sont modifiés par leur comportement à chaque pas de temps. Ces objets ne sont pas manipulés par les utilisateurs, ils ne nécessitent donc pas une réactivité importante. Il peut être intéressant de traiter ces objets soit sur le serveur (mode centralisé) si les machines des utilisateurs n'ont pas une puissance suffisante pour traiter beaucoup d'objets, soit sur chacun des nœuds (mode répliqué) si les connexions réseaux sont limitées et qu'il est préférable de réduire les communications entre les différents nœuds.

3.4 Adaptation dynamique durant la session

Grâce au mécanisme de migration décrit dans la partie 3.1, le paradigme de référents et de proxys permet de transformer facilement durant la session collaborative un référent en proxy, et inversement. Pour chaque objet virtuel, il est donc possible de modifier au cours de la session les nœuds qui possèdent un référent. D'une part, cela permet de changer dynamiquement le référent de nœud lorsque le mode hybride est utilisé, comme par exemple pour faire migrer le référent sur le nœud de l'utilisateur qui interagit. D'autre part, cela permet de changer dynamiquement le mode de distribution des données d'un objet durant la session. Effectivement, déplacer le référent à partir d'un nœud particulier vers le serveur permet de passer du mode hybride au mode centralisé, et inversement. Tandis que transformer tous les proxys en référents permet de passer du mode hybride ou centralisé au mode répliqué, et inversement.

Cette possibilité de modifier de façon dynamique le mode de distribution d'un objet virtuel permet d'adapter le compromis entre cohérence et réactivité de cet objet en fonction des activités des utilisateurs dans l'environnement virtuel, mais aussi des problèmes techniques qui peuvent survenir durant la session. Voici quelques critères liés à l'activité des utilisateurs qui peuvent servir pour initier des changements dynamiques de mode :

- Est-ce que l'objet est manipulé par un ou plusieurs utilisateurs ?
- Est-ce que l'objet se situe à proximité d'un ou plusieurs utilisateurs ?
- Est-ce que l'objet est dans le champ de vision d'un ou plusieurs utilisateurs ?

Au niveau technique, il peut aussi être intéressant d'adapter dynamiquement la distribution des données en fonction des troubles réseaux, des ralentissements de certaines machines, etc. Cela permet de remédier temporairement à ces problèmes techniques et d'améliorer les performances globales du système. Par exemple, lorsque le nœud qui possède le seul référent d'un objet (mode hybride) est déconnecté, ce référent peut être déplacé temporairement sur le serveur (mode centralisé) afin de pouvoir continuer à traiter le comportement et les modifications de cet objet.

3.5 Instanciation avec une architecture réseau client/serveur

Dans le cadre du projet ANR Collaviz, ce modèle d'adaptation dynamique de la distribution des données a été utilisé pour développer un *framework* dédié à la visualisation collaborative distante de données scientifiques. Le champ d'application de ce projet est très vaste : bien que les utilisateurs aient principalement besoin d'observer des données scientifiques, ils sont aussi amenés à réaliser des manipulations pour exploiter ces données, comme placer des annotations, des plans de coupe, etc. En conséquence, les différents types d'applications de cet environnement virtuel collaboratif imposent des besoins différents en termes de cohérence et de réactivité.

Traiter et visualiser des données scientifiques imposent également certaines contraintes techniques. Premièrement, il faut être capable de stocker de gros volumes de données. Deuxièmement, le traitement de ces données nécessite une puissance de calcul importante, équivalente à celle d'un *cluster* de calcul. Troisièmement, les données manipulées sont généralement des données industrielles et donc confidentielles. Il faut donc pouvoir sécuriser l'accès à ces données. Pour répondre à ces contraintes, le projet ANR Collaviz a choisi

d'utiliser une architecture client/serveur avec un serveur capable de stocker de gros volumes de données, d'héberger des moteurs de traitement et de gérer l'accès à ces données de façon sécurisée. Afin de garantir une sécurité maximale et un déploiement facile, le projet impose que toutes les communications entre les nœuds passent par le serveur. De plus, le projet a choisi d'utiliser des connexions standards ou des connexions sécurisées en fonction du type de l'application. Il est donc possible d'utiliser soit une couche réseau TCP pour les connexions standards, soit une couche réseau HTTP/HTTPS capable de passer les *firewalls* des entreprises et de garantir des connexions sécurisées. En conséquence, nous avons adapté notre modèle de distribution des données à une architecture réseau de type client/serveur et à différents types de connexions.

Dans la partie 3.5.1, nous décrivons l'impact de l'utilisation d'une architecture réseau client/serveur sur les différents modes de distribution des données de notre modèle. Puis, dans la partie 3.5.2, nous détaillons comment nous avons pu profiter de cette architecture client/serveur pour mettre en place des mécanismes centralisés améliorant la cohérence de l'environnement virtuel. Enfin, dans la partie 3.5.3, nous présentons les mesures de performance que nous avons réalisées pour les différents modes de distribution sur cette architecture client/serveur.

3.5.1 Impacts sur les trois modes de distribution

L'utilisation d'une architecture réseau de type client/serveur et de connexions sécurisées rajoutent de la latence réseau (***lat***) et augmentent les différences entre les différents modes de distribution des données. Nous décrivons dans cette partie l'impact de ces contraintes techniques sur chacun des trois modes de distribution des données de notre modèle d'adaptation dynamique.

Mode centralisé. L'utilisation d'une architecture réseau de type client/serveur n'a pas d'impact sur le mode centralisé vu que les communications passent déjà par le serveur dans ce mode. Les valeurs de $\boldsymbol{DC}$ et $\boldsymbol{LI}$ restent donc les mêmes.

Mode hybride. Le fait que les communications doivent passer obligatoirement par le serveur impacte fortement le mode hybride (cf. figure 3.5). La valeur de $\boldsymbol{LI}$ est doublée lorsque l'utilisateur qui interagit possède un proxy. En effet, dans ce cas, la demande de modification doit passer par le serveur pour être transmise au référent et il en est de même pour la mise à jour qui revient dans le sens inverse. En conséquence, la valeur de $\boldsymbol{DC}$ entre le nœud qui possède le référent et les autres est aussi doublée.

Mode répliqué. Pour le mode répliqué, seule la valeur de $\boldsymbol{DC}$ est doublée car les modifications faites en local doivent passer par le serveur pour être envoyées aux autres nœuds de la session (cf. figure 3.6).

La table 3.5 récapitule les valeurs de $\boldsymbol{DC}$ et $\boldsymbol{LI}$ quantifiées en fonction de la latence réseau (***lat***) lorsqu'une architecture réseau client/serveur est utilisée. Nous pouvons remarquer que les différences entre les modes sont doublées. Le cas où un utilisateur interagit avec un proxy dans le mode hybride est particulièrement à éviter car la latence lors des interactions est extrêmement importante dans cette configuration.

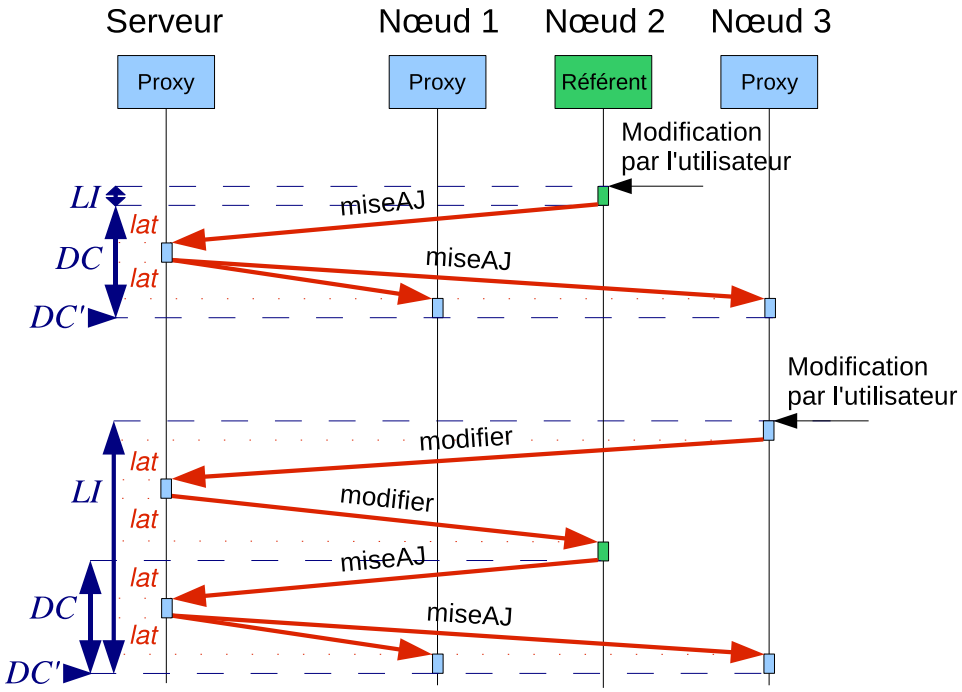


FIGURE 3.5 – Modification d'un objet : mode hybride et architecture client/serveur.

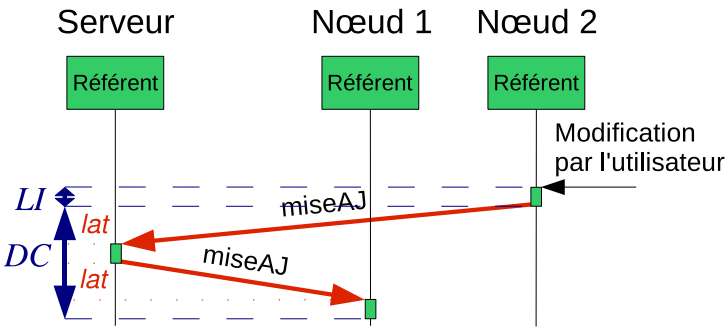


FIGURE 3.6 – Modification d'un objet : mode répliqué et une architecture client/serveur.

Mode de distribution	***LI***	***DC***
Centralisé	2 ***lat***	0
Hybride *Interaction avec le référent*	0	2 ***lat*** *(référent ↔ proxys)* 0 *(proxy ↔ proxy)*
Hybride *Interaction avec un proxy*	4 ***lat***	2 ***lat*** *(référent ↔ proxys)* 0 *(proxy ↔ proxy)*
Répliqué	0	2 ***lat***

TABLE 3.5 – ***LI*** et ***DC*** pour chaque mode de distribution et une architecture client/serveur.

3.5.2 Mécanismes centralisés pour améliorer la cohérence

En plus de l'architecture réseau et de la distribution des données, nous avons présenté dans l'état de l'art de la partie 2.1.4 certains mécanismes pouvant être mis en place pour assurer une meilleure cohérence de l'environnement virtuel. Tant qu'à utiliser une architecture réseau client/serveur, nous avons choisi de tirer avantage du serveur pour mettre en place plus efficacement ces mécanismes. Nous présentons dans cette partie la manière dont le serveur est utilisé pour sauvegarder l'état de l'environnement virtuel (partie 3.5.2.1), pour synchroniser ensemble les nœuds (partie 3.5.2.2) et pour gérer les droits d'accès et les accès concurrents aux objets virtuels (partie 3.5.2.3).

3.5.2.1 Sauvegarde de l'état de l'environnement virtuel

Notre modèle d'adaptation dynamique de la distribution des données permet au serveur d'être considéré comme un nœud supplémentaire qui participe à la session :

- Dans le mode centralisé, le serveur possède le référent de l'objet.
- Dans le mode hybride, le serveur possède un proxy de l'objet qui lui permet de recevoir et de stocker les données à jour pour cet l'objet.
- Dans le mode répliqué, le serveur possède un référent de l'objet qu'il exécute comme les autres nœuds afin de gérer sa propre version de l'objet.

Ainsi, quel que soit le mode de distribution des données choisi pour chaque objet virtuel, le serveur a toujours une copie à jour des données relatives à tous les objets du monde virtuel. Premièrement, cela permet d'avoir une entité centrale qui est capable de fournir une version courante du monde virtuel lorsqu'un nouvel utilisateur se connecte à la session. Deuxièmement, la sauvegarde de l'état courant du monde virtuel sur le serveur est aussi utile pour assurer la persistance au cas où des utilisateurs possédant les référents de certains objets seraient déconnectés de façon involontaire (à cause de troubles réseaux, par exemple). Enfin, le serveur permet également une persistance dans le temps : à la fin de la session collaborative, le serveur peut automatiquement sauvegarder l'état courant du monde virtuel afin de le restaurer ultérieurement pour permettre une collaboration dans le temps entre différents utilisateurs.

3.5.2.2 Synchronisation

Une synchronisation entre les nœuds participant à la session peut être effectuée dans les trois modes de distribution. Pour les objets utilisant le mode répliqué, elle assure que les comportements des objets soient exécutés en même temps sur chacun des nœuds. C'est le seul moyen de garantir une cohérence pour l'exécution des comportements dans ce mode (il subsiste néanmoins un décalage de cohérence lorsque les utilisateurs modifient les objets). Pour les objets utilisant le mode centralisé ou hybride, la cohérence est déjà assurée par le mode de distribution des données, mais la synchronisation peut permettre d'améliorer encore la cohérence en imposant que les messages de mise à jour soient traités en même temps sur chacun des nœuds.

Cette synchronisation est basée sur un verrou *(lockstep synchronisation)* comme dans RING [Funkhouser, 1995] et OpenMASK [Margery *et al.*, 2002] : chaque nœud doit attendre que tous les autres aient fini d'exécuter le pas de temps de simulation courant avant d'exécuter le pas de temps suivant. Sur chaque nœud, un pas de temps de simulation peut

être décomposé entre trois étapes :

1. traiter les messages reçus depuis le dernier pas de temps (dont les mises à jour),
2. exécuter le comportement des objets pour lesquels ce nœud est le référent,
3. envoyer les messages produits durant l'étape 2.

Le serveur est utilisé pour mettre en œuvre cette synchronisation. Il envoie un message à tous les nœuds contenant le numéro du pas de temps qu'ils sont autorisés à exécuter, puis il attend de recevoir des messages d'acquittement de tous les nœuds pour ce pas de temps (même numéro) avant d'envoyer le message pour le pas de temps suivant. Grâce à ce système de numérotation des pas de temps, une désynchronisation de quelques pas de temps peut être autorisée pour éviter les blocages dus à la latence réseau. L'algorithme 3.1 décrit la synchronisation du côté d'un nœud et l'algorithme 3.2 celle du côté du serveur.

Cependant, ce type de synchronisation peut poser problème lorsque les nœuds connectés à la session sont hétérogènes. En effet, dès qu'un nœud met du temps à traiter un pas de temps à cause d'une faible puissance de calcul ou d'une connexion réseau lente, ce nœud va ralentir tous les autres nœuds de la session. Nous avons donc proposé de mettre en place plusieurs groupes de synchronisation (cf. figure 3.7) :

- un groupe qui est fortement synchronisé avec le serveur,
- N groupes qui ont chacun une synchronisation de plus en plus faible (tous les nœuds d'un même groupe sont néanmoins synchronisés ensemble),
- un groupe qui n'est pas synchronisé du tout.

La composition de ces groupes peut être changée dynamiquement durant la session. Lorsqu'un nœud ralentit, à lui seul, tout un groupe de synchronisation pendant au moins une période déterminée (lorsque ce n'est pas un trouble ponctuel), il est déplacé dans un groupe avec une synchronisation plus faible. À l'inverse, lorsqu'un nœud attend systématiquement les autres nœuds de son groupe, il pourra être déplacé dans un groupe avec un synchronisation plus forte. En conséquence, les nœuds les plus lents perçoivent l'environnement virtuel avec du retard par rapport aux autres nœuds, mais ils sont de toute façon limités par leur faible puissance de calcul ou par une connexion réseau trop lente.

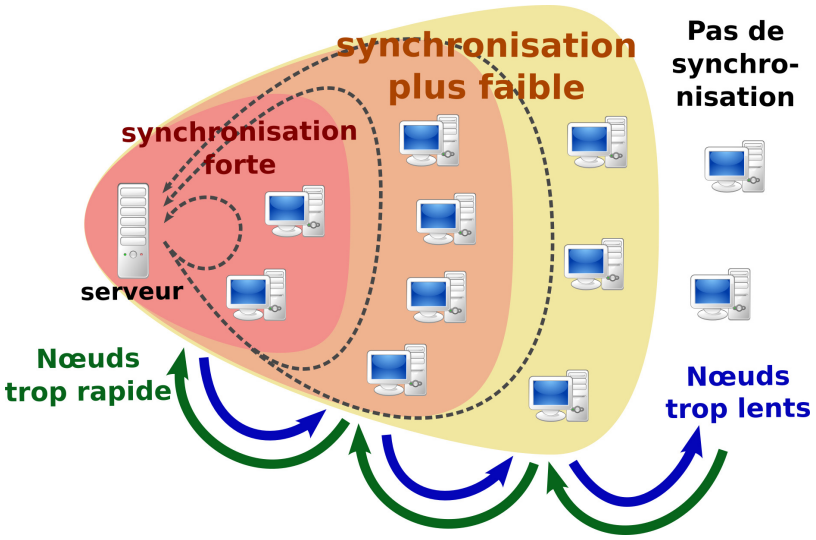


FIGURE 3.7 – Plusieurs groupes de synchronisation gérés par le serveur.

```
_estInitialisé : Booléen ← FAUX ;      // Indique si le client a été initialisé ou pas
_estSynchronisé : Booléen ← VRAI ;     // Indique si le client est synchronisé ou non
_pasDeTemps : Long ← 0 ;               // Numéro du pas de temps courant
_pasDeTempsATraiter : Long ← 0 ;       // Numéro du pas de temps qui peut être exécuté

Fonction traiter_message_recu( message : Message ) :
    // Si le message est un message de synchronisation
    Si (message.type() = Message.SYNCHRO) Alors
        données Sync : Données ← message.données() ;
        // Si le nœud n'a pas encore été initialisé
        Si (NON _estInitialisé) Alors
            // Initialiser le nœud à partir du pas de temps courant de la session
            _pasDeTemps ← donnéesSync.prendre(PAS_DE_TEMPS)−1 ;
            _estInitialisé ← VRAI ;
        Fin Si
        // Si le nœud a été placé dans le groupe non synchronisé
        Si (donnéesSync.prendre(GROUPE) = Groupe.RELÂCHÉ) Alors
            _estSynchronisé ← FAUX ;
        Sinon
            _estSynchronisé ← VRAI ;
        Fin Si
        // Mettre à jour le pas de temps que le nœud peut exécuter
        _pasDeTempsATraiter ← donnéesSync.prendre(PAS_DE_TEMPS) ;
    Fin Si
    ... // Traitement des autres types de messages
Fin

Fonction exécuter_nœud() :
    Si (_estInitialisé) Alors
        Si (_estSynchronisé) Alors
            // Si le nœud est autorisé à exécuter le prochain pas de temps)
            Si (_pasDeTemps < _pasDeTempsATraiter) Alors
                exécuter_un_pas_de_temps() ;
                _pasDeTemps++ ;
                // Envoyer un message d'acquittement au serveur
                // contenant le numéro du pas de temps exécuté
                donnéesAcq : Données ;
                donnéesAcq.ajouter(PAS_DE_TEMPS, _pasDeTemps) ;
                acquittement : Message ← nouveau(Message.ACQ, donnéesAcq) ;
                envoyer_au_serveur(acquittement) ;
            Fin Si
        Sinon
            exécuter_un_pas_de_temps() ;
            _pasDeTemps++ ;
        Fin Si
    Fin Si
Fin
```

ALGORITHME 3.1 – Algorithme de synchronisation du côté d'un nœud.

```
_groupes : Groupe[N+2] ;   // Tableau stockant les groupes de synchronisation
_pdtGroupes : Long[N+2] ; // Tableau stockant le pas de temps de chaque groupe
_retardAccepté : Entier ;     // Nombre de pas de temps de retard accepté

Fonction groupe_près( numGroupe : Entier ) : Booléen
    estPrès : Booléen ← VRAI ;
    Pour chaque nœud : Nœud de _groupes[numGroupe].nœuds() faire
        // Récupérer le pas de temps reçu lors du dernier acquittement de ce nœud
        acqNd : Long ← nœud.dernier_acquittement_recu() ;
        // Si le nœud a plus de retard que le nombre de pas de temps de retard accepté
        Si (acqNd < (_pdtGroupes[numGroupe] − _retardAccepté) ) Alors
            estPrès ← FAUX ; // Le groupe n'est pas près
        Fin Si
    Fait
    Retourner estPrès ;
Fin

Fonction envoyer_message_synchro( numGroupe : Entier ) :
    // Préparer les données pour le message de synchronisation
    donnéesSync : Données ;
    donnéesSync.ajouter(GROUPE, numGroupe) ;
    donnéesSync.ajouter(PAS_DE_TEMPS, _pdtGroupes[numGroupe]) ;
    // Créer et envoyer le message avec les données à tous les nœuds du groupe
    msgSync : Message ← nouveau(Message.SYNCHRO, donnéesSync) ;
    _groupes[numGroupe].envoyer_à_tous(msgSync) ;
Fin

Fonction exécuter_serveur() :
    // Si les nœuds fortement synchronisés sont près à exécuter le pas de temps suivant
    Si (groupe_près(Groupe.SYNC_FORTE)) Alors
        // Incrémenter le pas de temps du groupe fortement synchronisé
        _pdtGroupes[Groupe.SYNC_FORTE]++ ;
        // Envoyer un message de synchronisation à tous les nœuds du groupe
        envoyer_message_synchro(Groupe.SYNC_FORTE) ;
        // Exécuter le pas de temps sur le serveur
        exécuter_un_pas_de_temps() ;
    Fin Si

    // Pour tous les autres groupes plus faiblement synchronisés
    Pour num : Entier de Groupe.SYNC_FORTE+1 à Groupe.RELÂCHÉ−1 faire
        // Si le groupe ne va pas plus vite que le groupe synchronisé avec le serveur
        Si (_pdtGroupes[Groupe.SYNC_FORTE] < _pdtGroupes[num]) Alors
            // Si les nœuds du groupe sont près à exécuter le pas de temps suivant
            Si (groupe_près(num)) Alors
                _pdtGroupes[num]++ ;
                envoyer_message_synchro(num) ;
            Fin Si
        Fin Si
    Fait
Fin
```

ALGORITHME 3.2 – Algorithme de synchronisation du côté du serveur.

3.5.2.3 Gestion des droits et des accès concurrents

Pour permettre un accès sécurisé aux données et ainsi garantir leur confidentialité, nous avons mis en place un système de droits d'accès aux objets virtuels assuré par le serveur. Le serveur stocke un niveau d'accès pour chaque objet du monde virtuel. Le niveau d'accès 0 est le niveau le plus restrictif afin de pouvoir ajouter autant de niveaux d'accès (moins restrictifs) que nécessaire. De plus, nous avons aussi défini une notion de critère d'accès aux objets pour les utilisateurs. Nous avons utilisé principalement trois critères d'accès :

- le droit de voir un objet,
- le droit de modifier un objet,
- le droit de créer ou de supprimer un objet.

Cependant, cette liste des critères d'accès peut être étendue autant que nécessaire. Pour chaque utilisateur, le serveur stocke un niveau d'accès associé à chacun de ces critères. Ainsi, si l'utilisateur a un niveau d'accès supérieur ou égal à celui de l'objet, il pourra lui appliquer l'action définie dans le critère d'accès. Par exemple, si un utilisateur a les droits d'accès suivants `[voir=0, modifier=2, créer/supprimer=5]`, il pourra alors voir tous les objets du monde virtuel, modifier tous les objets sauf ceux de niveau 0 et 1, et créer ou supprimer des objets de niveau supérieur ou égal 5 .

Le serveur est également utilisé pour gérer les accès concurrents aux objets virtuels. Nous avons mis en place une gestion pessimiste de la concurrence comme dans BrickNet [Singh *et al.*, 1995] : seul un utilisateur peut modifier un objet virtuel à la fois. Le serveur stocke pour chaque objet l'identifiant de l'utilisateur qui est son propriétaire, c'est-à-dire celui qui peut actuellement modifier cet objet. Étant donné que toutes les communications passent par le serveur, ce dernier ne peut contrôler que les demandes de modification ou les mises à jour après modification (selon le mode de distribution utilisé) proviennent bien du propriétaire de l'objet avant de transmettre les messages à tous les autres nœuds de la session. Pour modifier un objet virtuel, un utilisateur doit d'abord demander au serveur à en devenir le propriétaire. Avant de lui accorder la propriété, le serveur vérifie qu'il n'y a pas déjà un propriétaire pour cet objet et que cet utilisateur a bien les droits d'accès nécessaires pour modifier l'objet (niveau d'accès supérieur ou égal à celui de l'objet pour la modification). Si un autre utilisateur est déjà le propriétaire de l'objet, l'utilisateur qui a fait la demande doit attendre que l'autre ait fini de modifier l'objet.

Même si seulement un utilisateur peut manipuler un objet à la fois, nous avons pu mettre en œuvre des co-manipulations. Dans ce cas, chacun des utilisateurs impliqués dans une co-manipulation utilise un outil particulier qu'il est le seul à manipuler (pas de problème de propriété). Il existe alors deux solutions pour effectuer une co-manipulation : soit l'objet manipulé par plusieurs utilisateurs s'abonne aux informations des outils afin d'intégrer les valeurs qu'ils proposent, soit les outils peuvent appliquer une contrainte physique sur un objet et une co-manipulation est possible lorsque plusieurs outils appliquent une contrainte sur un même objet.

3.5.3 Mesures de performance

Pour évaluer les performances de notre modèle avec une architecture client/serveur, nous avons effectué des mesures pour les trois modes de distribution des données. Après avoir décrit les conditions expérimentales (partie 3.5.3.1), nous présentons les mesures de la latence lors des interactions et du décalage de cohérence (partie 3.5.3.2), puis celles de la quantité de traitements et du nombre de communications réseau (partie 3.5.3.3).

3.5.3.1 Conditions expérimentales

Les mesures de performance ont été effectuées avec deux clients connectés à un serveur distant. Les clients utilisent deux stations de travail similaires équipées d'un processeur Intel® Xeon®, tandis que le serveur est hébergé par un ordinateur portable équipé d'un processeur Intel® Core™2. Les deux clients sont connectés au serveur soit par un réseau local (LAN), soit par un réseau étendu (WAN). Pour le réseau LAN, les ordinateurs sont connectés sur le réseau Ethernet à 100 Mbit/s du laboratoire IRISA en utilisant la couche TCP du *framework* Collaviz. Pour le réseau WAN, ils sont connectés par Internet en utilisant la couche HTTP du *framework* Collaviz. Dans ce second cas, les clients sont situés au laboratoire IRISA sur le campus de Beaulieu et le serveur se trouve dans le centre ville de Rennes. En conséquence, la connexion par Internet entre les clients et le serveur passe par Paris : de Rennes à Paris par le réseau à grande vitesse des universités *Renater* sur environ 450 km (Rennes, Caen, Rouen, Paris), puis de Paris à Rennes par un réseau Internet standard sur environ 350 km.

Afin de ne pas surcharger les ordinateurs et le réseau, les pas de temps de simulation ne sont pas exécutés aussi vite que possible sur les clients et sur le serveur. Nous imposons une fréquence d'exécution des pas de temps de simulation qui peut varier entre 30, 60 et 120 Hz. Nous avons expliqué, dans la partie 3.5.2.2, qu'un pas de temps de simulation est composé de trois phases : traitement des messages reçus, exécution du comportement et envoi des messages générés. En conséquence, la temporisation entre chaque pas de temps pour tenir la fréquence peut introduire un délai entre la réception et le traitement des messages sur chacun des nœuds par lesquels ils transitent. Ce délai est au maximum légèrement inférieur à la durée d'un pas de temps à la fréquence définie, c'est-à-dire 33 ms à 30 Hz, 17 ms à 60 Hz ou 8 ms à 120 Hz. Ce délai peut avoir une importance non négligeable sur les valeurs $\boldsymbol{DC}$ et $\boldsymbol{LI}$ surtout lorsque la latence réseau est faible (comme sur le réseau LAN).

Enfin, les deux clients peuvent être synchronisés ou désynchronisés. En utilisant le mécanisme de synchronisation présenté dans la partie 3.5.2.2, les deux clients peuvent être synchronisés ensemble en étant placés tous les deux dans le groupe fortement synchronisé avec le serveur ou ils peuvent être désynchronisés en étant placés tous les deux dans le groupe qui n'a pas de synchronisation.

3.5.3.2 Mesures de *LI* et *DC*

Pour effectuer les mesures de $\boldsymbol{LI}$ et $\boldsymbol{DC}$, nous avons créé un environnement virtuel simplifié contenant seulement un objet virtuel. Lorsqu'un client interagit avec cet objet, nous mesurons le temps nécessaire pour que ce client reçoive la modification ($\boldsymbol{LI}$), ainsi que la différence de temps entre les moments où cette modification arrive sur les deux clients ($\boldsymbol{DC}$). Ces mesures ont été réalisées avec ou sans synchronisation entre les clients et pour des fréquences de 30, 60 ou 120 Hz. La table 3.6 présente les résultats obtenus sur le réseau LAN et la table 3.7 ceux obtenus sur le réseau WAN. Ces résultats correspondent à des valeurs moyennes pour une modification, obtenues après 2000 modifications de l'objet dans chaque cas. Pour le mode hybride, nous avons pu mesurer uniquement le décalage de cohérence entre le réfèrent et le proxy, et pas celui entre deux proxys. Cependant, ce décalage entre proxys est identique à celui mesuré dans le mode centralisé étant donné que les mises à jour des proxys passent dans tous les cas par le serveur. De manière générale, les valeurs obtenues avec les deux types de réseaux correspondent aux valeurs théoriques présentées dans la table 3.5.

Sync.	Mode de distribution	***LI***			***DC***		
		Fréquence (Hz) *(Pas de temps en ms)*			Fréquence (Hz) *(Pas de temps en ms)*		
		30 *(33)*	60 *(17)*	120 *(8)*	30 *(33)*	60 *(17)*	120 *(8)*
Non	**Centralisé**	64.7	29.4	20.1	4.5	4.2	2.8
	Hybride - *Réfèrent*	0.005	0.005	0.005	39.0	22.0	12.6
	Hybride - *Proxy*	104.4	55.4	35.5	42.9	25.0	17.2
	Répliqué	0.005	0.005	0.005	37.2	21.8	12.4
Oui	**Centralisé**	65.1	36.2	27.6	0.0	0.2	0.2
	Hybride - *Réfèrent*	0.005	0.005	0.005	45.5	24.3	14.4
	Hybride - *Proxy*	120.0	56.1	42.7	46.1	24.3	18.0
	Répliqué	0.005	0.005	0.005	42.3	25.2	14.0

TABLE 3.6 – Valeurs de ***LI*** et ***DC*** (en ms) lors d'une modification sur un réseau LAN.

Sync.	Mode de distribution	***LI***			***DC***		
		Fréquence (Hz) *(Pas de temps en ms)*			Fréquence (Hz) *(Pas de temps en ms)*		
		30 *(33)*	60 *(17)*	120 *(8)*	30 *(33)*	60 *(17)*	120 *(8)*
Non	**Centralisé**	200.6	180.4	160.4	18.6	23.7	22.1
	Hybride - *Réfèrent*	0.004	0.004	0.004	209.3	207.3	207.4
	Hybride - *Proxy*	405.9	387.2	373.0	195.1	202.3	195.5
	Répliqué	0.003	0.003	0.003	215.4	216.4	214.2
Oui	**Centralisé**	207.7	203.2	195.5	1.4	1.8	1.9
	Hybride - *Réfèrent*	0.004	0.004	0.004	166.0	164.2	161.7
	Hybride - *Proxy*	410.4	400.1	396.6	166.6	163.6	164.1
	Répliqué	0.003	0.003	0.003	166.0	162.9	162.5

TABLE 3.7 – Valeurs de ***LI*** et ***DC*** (en ms) lors d'une modification sur un réseau WAN.

Lorsque la latence réseau est faible (inférieure à la durée d'un pas de temps de simulation), nous remarquons dans la table 3.6 que la fréquence d'exécution des pas de temps de simulation impacte fortement les valeurs de ***LI*** et ***DC***. En effet, nous pouvons estimer la latence réseau à quelques millisecondes dans le cas du réseau LAN et nous remarquons que la durée du pas de temps de simulation intervient à chaque fois que le message arrive sur un des nœuds. Par exemple, pour ***LI***, il faut attendre environ deux fois la durée d'un pas de temps dans le cas du mode centralisé (réception du message de modification sur le serveur, puis réception de la mise à jour sur le client), et quatre fois dans le cas du mode hybride si l'interaction se fait avec un proxy. Si les deux clients ne sont pas synchronisés, il est possible que l'exécution des pas de temps se « chevauche » entre les différents nœuds et qu'il n'y ait pas forcément à attendre le pas de temps entier à chaque fois. Par contre, si les deux clients sont fortement synchronisés, il faudra attendre quasiment un pas de temps entier à chaque fois. Cela explique que ***LI*** soit légèrement meilleure dans le cas non synchronisé que dans le cas synchronisé. Il en est de même pour ***DC*** entre le référent et les proxys dans le mode hybride et pour ***DC*** entre deux référents dans le mode répliqué.

Par contre, pour le mode centralisé, la synchronisation réduit légèrement la valeur de ***DC*** car les deux clients attendent la réception du même message et le délai introduit par le cadencement des pas de temps est le même sur les deux clients.

Lorsque la latence réseau est plus importante (supérieure à la durée d'un pas de temps de simulation), nous remarquons dans la table 3.7 que la synchronisation permet d'améliorer significativement la cohérence (***DC***). Dans le cas du réseau WAN, nous pouvons estimer la latence réseau à environ 80 ms. Il peut donc se passer plusieurs pas de temps pendant la transmission du message sur le réseau. Le temps à attendre lors de la réception d'un nouveau message sur chacun des nœuds est donc plus aléatoire. Si les clients ne sont pas synchronisés, il y a davantage de risques que les messages de mise à jour ne soient pas traités en même temps. Par ailleurs, comme dans le cas du réseau LAN, l'augmentation de la fréquence d'exécution des pas de temps permet en moyenne de diminuer un peu ***LI***.

3.5.3.3 Mesures de la charge de calcul et des communications réseaux

Pour effectuer les mesures de charge de calcul sur les différents nœuds de la session et du nombre de messages échangés entre ces nœuds, nous avons créé un environnement virtuel simplifié avec seulement 12 objets virtuels animés (avec un comportement qui évolue en fonction du temps). Nous avons effectué ces mesures en utilisant le réseau WAN, une fréquence d'exécution des pas de temps de 60 Hz et une synchronisation forte. Nous avons comparé trois configurations différentes :

- les 12 objets utilisent le mode centralisé *(12 Centr.)*,
- 4 objets utilisent le mode centralisé et les 8 autres objets utilisent le mode hybride avec 4 référents sur un des clients et 4 référents sur l'autre client *(3X4 Hybr.)*,
- les 12 objets utilisent le mode répliqué *(12 Réplic.)*.

Pour chacune de ces configurations, nous avons mesuré le temps nécessaire pour exécuter l'ensemble des comportements des objets et le nombre de messages de mise à jour reçus sur chacun des nœuds. La figure 3.8 donne les valeurs moyennes pour un pas de temps de simulation (moyennes effectuées sur 2000 pas de temps de simulation).

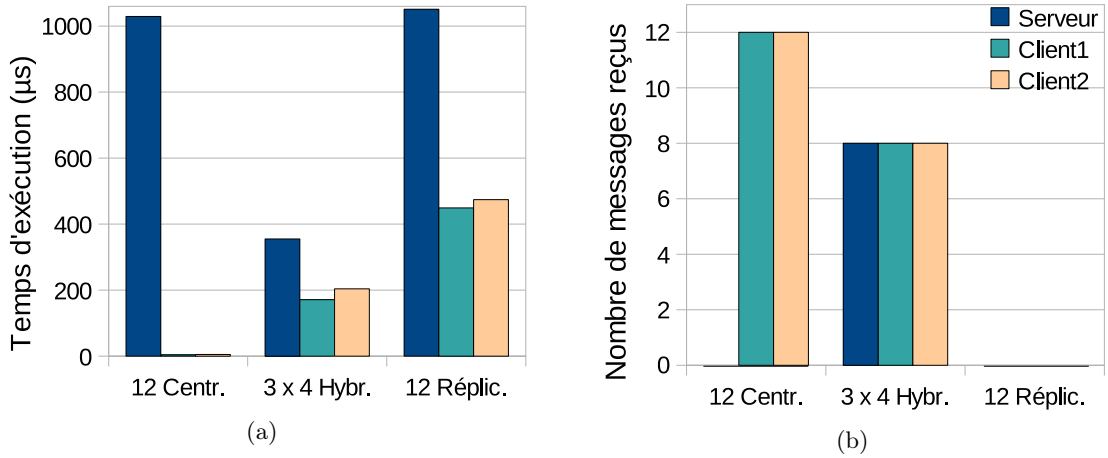


FIGURE 3.8 – (a) Temps d'exécution des comportements des objets et (b) nombre de messages de mise à jour reçus sur chaque nœud pour traiter les 12 objets virtuels animés.

Dans le mode centralisé *(12 Centr.)*, tous les traitements sont effectués sur le serveur et les deux clients reçoivent des messages de mise à jour pour les deux 12 objets. Dans le mode hybride *(3X4 Hybr.)*, les traitements sont répartis sur les trois nœuds, mais il y a aussi beaucoup de communications réseaux entre les nœuds (8 messages reçus sur chaque nœud). À l'inverse, dans le mode répliqué *(12 Réplic.)*, les traitements sont dupliqués sur chacun des nœuds, mais aucun message n'est envoyé sur le réseau. Pour les configurations *3X4 Hybr.* et *12 Réplic.*, la différence sur la figure 3.8(a) entre le temps d'exécution sur le serveur et les temps d'exécution sur les clients est simplement due à une puissance du CPU plus faible sur le serveur de test que sur les clients.

3.6 Conclusion

Pour améliorer la collaboration entre des utilisateurs dans un environnement virtuel distribué, il est essentiel d'offrir à ces utilisateurs un compromis adéquat entre la cohérence de l'environnement virtuel et la réactivité du système lorsqu'ils interagissent. Afin d'obtenir un environnement virtuel distribué générique qui puisse s'adapter aux contraintes techniques et aux besoins de différents types d'applications, nous avons proposé un modèle d'adaptation dynamique de la distribution des données de l'environnement virtuel. Ce modèle permet de mettre en œuvre trois modes de distribution des données qui ont chacun des avantages et des inconvénients au niveau de la cohérence et de la réactivité, mais aussi au niveau de la quantité de traitements à effectuer sur chaque nœud et du nombre de communications sur le réseau entre ces nœuds (cf. table 3.8). Étant donné que tous les objets n'ont pas les mêmes besoins en fonction de leur rôle dans le monde virtuel, notre modèle permet de choisir le mode de distribution des données de façon indépendante pour chaque objet virtuel. De plus, pour s'adapter aux actions des utilisateurs et à d'éventuelles perturbations techniques, le choix du mode de distribution des données d'un objet virtuel peut être modifié dynamiquement pendant la session.

Dans le cadre du projet ANR Collaviz, nous avons instancié notre modèle de distribution des données avec une architecture réseau client/serveur afin de mettre en œuvre un environnement virtuel distribué pour la visualisation collaborative de données scientifiques. Même si cette architecture réseau impose que toutes les communications passent par le serveur, chacun des trois modes de distribution garde ses caractéristiques propres et offre donc un intérêt particulier en fonction des besoins de l'application. Par ailleurs, nous avons pu profiter du serveur pour mettre en place les mécanismes de sauvegarde de l'environnement virtuel, de synchronisation et de gestion des accès concurrents. Des mesures de performance de l'environnement virtuel distribué ont été effectuées pour les trois modes de distribution

Caractéristiques	Mode centralisé	Mode hybride	Mode répliqué
Cohérence de l'environnement virtuel	$++$	$+$	$--$
Réactivité lors des interactions	$--$	$+-$	$++$
Répartition des traitements	$++$	$++$	$--$
Nombre de communications réseaux	$--$	$--$	$++$

TABLE 3.8 – Récapitulatif des avantages et inconvénients de chacun des trois modes de distribution des données proposés.

des données avec des connexions réseau différentes et plusieurs configurations du système. Ces mesures ont permis de vérifier les caractéristiques de chacun des trois modes et de montrer l'impact de la synchronisation des nœuds sur le système : même si elle n'améliore pas les performances lorsque la latence réseau est faible comme sur les réseaux LAN, elle permet d'augmenter significativement la cohérence lorsque la latence réseau est importante comme sur les réseaux WAN.

Enfin, les changements dynamiques de mode de distribution interviennent, pour l'instant, à la demande des utilisateurs ou lors d'événements simples : migration du référent sur le nœud de l'utilisateur qui interagit, migration du référent sur le serveur lors que l'utilisateur qui possède le référent se déconnecte de la session, etc. Il serait intéressant de trouver un moyen d'automatiser les changements de mode de distribution en proposant une façon plus générique de décrire les critères et les règles qui gouvernent ces changements. Il serait envisageable de définir un langage qui décrive la politique de changement de mode de distribution de chaque objet virtuel. Il faudrait également effectuer des mesures supplémentaires avec les métriques proposées pour évaluer l'impact de ces changements automatiques sur les performances.

Chapitre 4

Modèle d'architecture logicielle pour les environnements virtuels collaboratifs

Même si il existe plusieurs modèles d'architecture logicielle permettant de concevoir des systèmes collaboratifs, l'état de l'art présenté au chapitre 2 montre qu'il n'existe pas de solution adaptée pour modéliser les environnements virtuels collaboratifs distribués. Pour mettre en œuvre le modèle d'adaptation dynamique de la distribution des données présenté au chapitre précédent, nous avons besoin d'un modèle d'architecture logicielle qui soit capable de modéliser les objets virtuels indépendamment de la façon dont ils sont distribués sur le réseau. De plus, ce modèle d'architecture logicielle doit aussi être capable de modéliser l'environnement virtuel indépendamment de la façon dont il est représenté et restitué aux utilisateurs : il doit pouvoir s'adapter aux différents composants logiciels (librairies graphiques, sonores, hatiques, etc.) utilisés en lien avec les dispositifs matériels des utilisateurs (cf. figure 4.1).

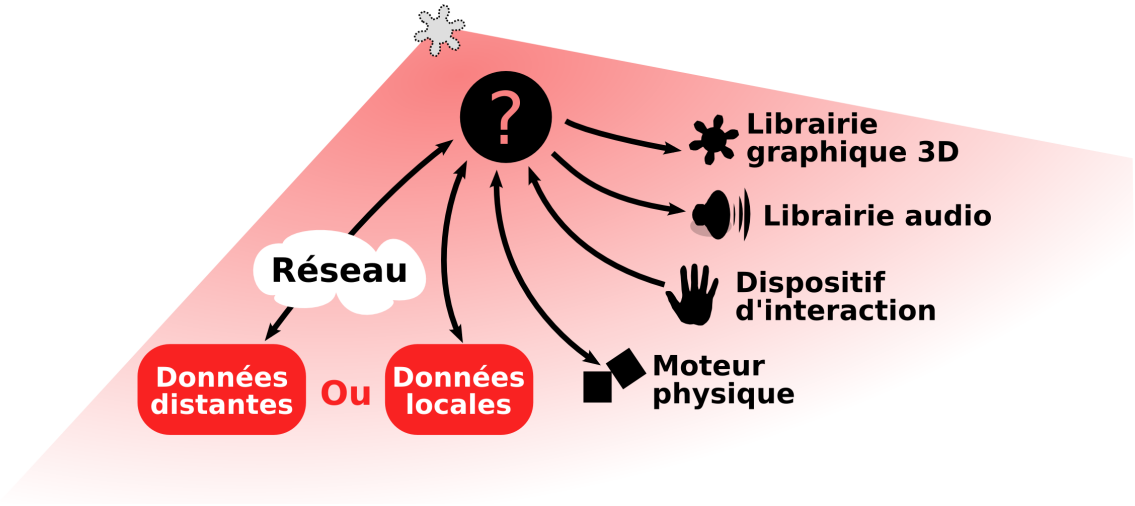


FIGURE 4.1 – Comment modéliser les objets d'un environnement virtuel collaboratif ?

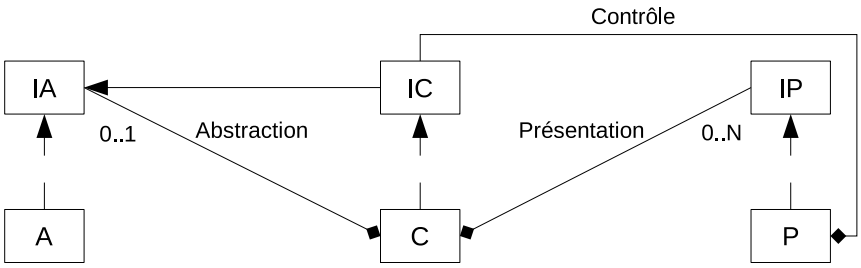


FIGURE 4.2 – Modèle PAC avec des interfaces faisant le lien entre les différentes facettes.

Dans ce chapitre, nous proposons un nouveau modèle d'architecture logicielle appelé PAC-C3D qui étend le modèle PAC pour les environnements virtuels collaboratifs [Duval et Fleury, 2011b]. Dans la partie 4.1, nous présentons d'abord l'interprétation du modèle PAC que nous avons utilisée. Nous décrivons ensuite le modèle PAC-C3D dans la partie 4.2 et la manière de développer un environnement virtuel collaboratif avec ce modèle dans la partie 4.3. Enfin, dans la partie 4.4, nous expliquons comment ce modèle a été utilisé pour mettre en œuvre un environnement virtuel collaboratif offrant une interopérabilité entre plusieurs librairies graphiques et intégrant un moteur physique.

4.1 Interprétation de l'architecture logicielle PAC

Le modèle PAC [Coutaz, 1987], présenté dans la partie 2.2.1.2, est un modèle multi-agent. Nous pensons que ce genre de modèle est bien adapté aux environnements virtuels car chaque objet virtuel peut être décrit comme un agent avec des données particulières et un comportement propre. Le modèle PAC est composé de trois facettes : la *Présentation*, l'*Abstraction* et le *Contrôle*. Cependant, nous proposons d'utiliser une interprétation particulière du modèle PAC présentée dans [Duval et Tarby, 2006]. Cette interprétation se distingue principalement par le fait qu'elle définit le *Contrôle* comme un « proxy » de l'*Abstraction*. La figure 4.2 présente cette interprétation du modèle PAC légèrement modifiée afin qu'elle permette d'associer plusieurs *Présentations* à un même *Contrôle*. Chaque objet du monde virtuel est décrit par trois interfaces :

- L'interface de l'*Abstraction* (**IA**) : elle déclare les méthodes permettant de modifier l'objet, d'exécuter son comportement et de récupérer les valeurs de ses paramètres.
- L'interface de la *Présentation* (**IP**) : elle déclare les méthodes permettant de mettre à jour la représentation de l'objet (visuelle, sonore, haptique, etc.).
- L'interface du *Contrôle* (**IC**) : elle hérite de **IA** afin d'offrir les mêmes fonctionnalités que l'*Abstraction* et de jouer son rôle de « proxy ». Le *Contrôle* peut ainsi réguler l'accès à l'*Abstraction* et maintenir la cohérence entre l'*Abstraction* et les *Présentations*.

Généralement, les *Présentations* sont fortement dépendantes des composants logiciels utilisés pour restituer l'environnement virtuel aux utilisateurs, qui eux-mêmes dépendent fortement du système et des dispositifs matériels. L'interface de la *Présentation* permet de séparer, au niveau du *Contrôle*, la partie principale de l'environnement virtuel collaboratif de ces composants logiciels spécifiques. De plus, l'interface du *Contrôle* assure que les *Présentations* et l'*Abstraction* soient indépendantes de l'implémentation du *Contrôle*.

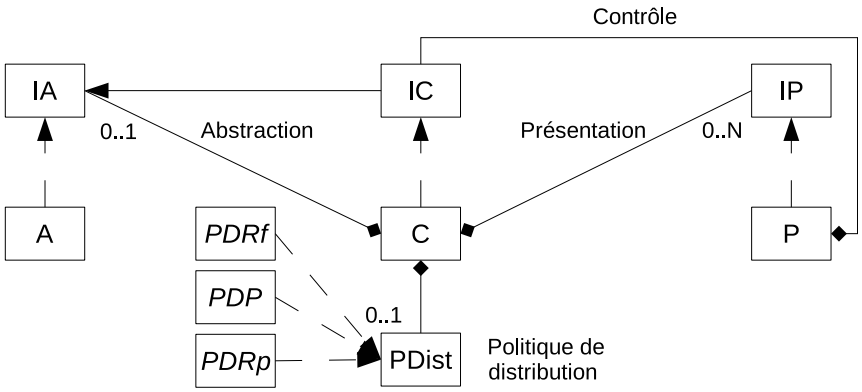


FIGURE 4.3 – Modèle PAC-C3D mettant en œuvre différentes politiques de distribution des données au niveau du *Contrôle*.

4.2 Modèle d'architecture logicielle PAC-C3D

Nous avons établi, dans la partie 2.2.2, qu'il existe plusieurs solutions pour prendre en compte l'aspect collaboratif d'un système interactif. Un modèle basé sur des agents nous semble particulièrement bien adapté aux environnements virtuels collaboratifs car les aspects de collaboration peuvent être intégrés à chaque agent. Ainsi, chaque agent décrivant un objet virtuel peut gérer d'une façon particulière la distribution de ses données sur le réseau. Par ailleurs, nous pensons que les aspects de la distribution sur le réseau ne doivent impacter ni l'*Abstraction* ni la *Présentation* afin de ne pas avoir à adapter le code décrivant les objets ou leurs représentations en fonction du mode de distribution des données choisies. Nous avons donc choisi de traiter les aspects collaboratifs uniquement au niveau du *Contrôle* sous la forme d'une politique de distribution (cf. figure 4.3). Cette politique de distribution définit comment synchroniser les versions d'un même objet sur les différents nœuds afin de maintenir la cohérence entre ces différentes versions. Nous avons proposé trois politiques de distribution :

- La politique de distribution Référent ***(PDRf)*** : un *Contrôle* ***PDRf*** demande à son *Abstraction* d'exécuter le comportement de l'objet. Il lui transmet également les demandes de modification (locales ou distantes) de l'objet. Dès que l'*Abstraction* a été modifiée par le comportement ou par la demande d'un utilisateur, il diffuse des mises à jour à tous les *Contrôles* PDP qui lui sont associés sur les autres nœuds.
- La politique de distribution Proxy ***(PDP)*** : un *Contrôle* ***PDP*** intercepte les demandes de modification locales de l'objet et les transmet au *Contrôle* ***PDRf*** qui lui sont associé sur le nœud référent. Il traite également les mises à jour qu'il reçoit de son *Contrôle* ***PDRf*** associé.
- La politique de distribution Répliquée ***(PDRp)*** : un *Contrôle* ***PDRp*** demande à son *Abstraction* d'exécuter le comportement de l'objet. Il lui transmet seulement les demandes de modification locales de l'objet. Dès que l'*Abstraction* a été modifiée par une demande locale de l'utilisateur (et pas par le comportement), il diffuse des mises à jour à tous les *Contrôles* ***PDRp*** qui lui sont associés sur les autres nœuds. En conséquence, il doit aussi traiter les mises à jour provenant de ses *Contrôles* ***PDRp*** associés. Si l'objet a été aussi modifié en local par l'utilisateur, il faut en plus qu'il gère les conflits potentiels entre la modification locale et les mises à jour reçues.

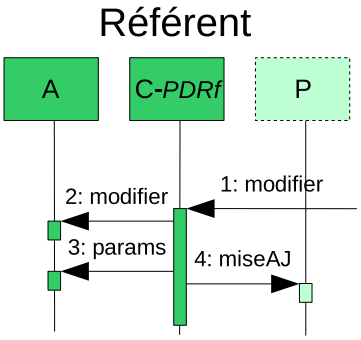


FIGURE 4.4 – Implémentation d'un référent avec PAC-C3D.

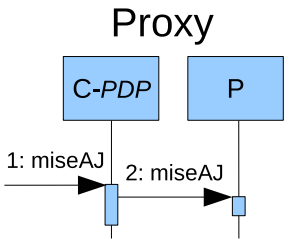


FIGURE 4.5 – Implémentation d'un proxy avec PAC-C3D.

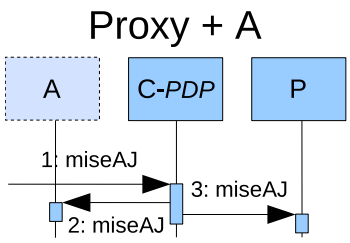


FIGURE 4.6 – Implémentation d'un proxy avec une *Abstraction* avec PAC-C3D.

4.3 Mise en œuvre d'un environnement virtuel collaboratif avec PAC-C3D

Le modèle d'architecture logicielle PAC-C3D permet de mettre en œuvre un environnement virtuel collaboratif distribué en séparant le cœur de l'application de la façon dont cet environnement est distribué sur le réseau et de la façon dont il est restitué aux utilisateurs. Dans la partie 4.3.1, nous expliquons comment nous avons mis en œuvre le modèle d'adaptation dynamique de la distribution des données présenté dans le chapitre précédent en utilisant le modèle PAC-C3D. Dans la partie 4.3.2, nous étudions comment ce modèle permet de découpler les données de l'environnement virtuel des différentes représentations virtuelles qui en sont faites. Enfin, nous détaillons la façon de gérer l'ensemble des objets PAC-C3D de l'environnement virtuel dans la partie 4.3.3 et la façon de créer ces objets dans la partie 4.3.4.

4.3.1 Adaptation dynamique de la distribution des données

Le modèle d'architecture logicielle PAC-C3D permet d'implémenter facilement le paradigme de référents et de proxys. Pour chaque objet du monde virtuel, il faut développer les trois facettes du modèle PAC-C3D en spécifiant si nécessaire les trois interfaces associées avec les fonctionnalités particulières de l'objet. À partir de ces trois facettes, il est possible d'implémenter facilement pour un objet :

- un référent en instanciant sur le nœud une *Abstraction*, un *Contrôle* et éventuellement des *Présentations* si ce nœud n'est pas le serveur (cf. figure 4.4),
- un proxy en instanciant un *Contrôle* et une *Présentation* sur le nœud (cf. figure 4.5),
- un proxy avec une *Abstraction* en instanciant les trois facettes sur le nœud (cf. figure 4.6). Dans ce cas, l'*Abstraction* n'effectue aucun traitement, elle stocke seulement une version à jour des données de l'objet afin de faciliter la migration du référent sur ce nœud, le cas échéant.

Cette deuxième solution d'implémentation pour le proxy facilite la migration du référent sur le nœud car il n'y a pas besoin de transférer les données de l'objet au moment de la migration. Cette solution est donc intéressante lorsqu'il est nécessaire de faire migrer souvent et rapidement le référent de l'objet comme, par exemple, lorsque le référent migre dynami-

quement sur le nœud de l'utilisateur qui interagit avec cet objet. Cependant, cette solution a l'inconvénient de dupliquer les données de l'objet sur chacun des nœuds. Pour chaque objet virtuel, il est possible de choisir indépendamment si les proxys doivent posséder une *Abstraction* ou non, en fonction des besoins relatifs à cet objet.

Cette implémentation du paradigme de référents et de proxys grâce au modèle PAC-C3D garantit la séparation entre les données de l'environnement virtuel et la façon dont elles sont distribuées sur le réseau. Même si les objets virtuels ne sont pas dans le même mode de distribution, ils peuvent communiquer entre eux par l'intermédiaire de leur *Contrôle*. Cela permet de choisir le mode de distribution des données de façon indépendante pour chaque objet virtuel. De plus, pour changer le mode de distribution d'un objet, il suffit de changer la politique de distribution au niveau du *Contrôle* et, éventuellement, de créer une *Abstraction* à jour sur le nœud qui possédera le nouveau référent. Cela permet de changer le mode de distribution dynamiquement durant la session.

4.3.1.1 Instanciation du mode centralisé

Lorsqu'un objet utilise le mode centralisé, un seul référent est situé sur le serveur et des proxys sont utilisés sur les autres nœuds de la session. La figure 4.7 décrit les échanges entre les facettes PAC-C3D réparties sur un serveur et deux clients lors d'une modification d'un objet utilisant le mode centralisé :

1-3 L'utilisateur effectue une action sur l'objet au travers de la *Présentation* disponible sur le nœud 1. Cette *Présentation* demande à son *Contrôle* d'effectuer la modification. Comme le *Contrôle* du nœud 1 utilise la politique ***PDP***, il transmet la demande de modification au *Contrôle* ***PDRf*** sur le serveur.

4-6 Le *Contrôle* ***PDRf*** sur le serveur transmet la demande de modification à son *Abstraction* qui exécute les traitements nécessaires. Puis, le *Contrôle* ***PDRf*** demande à l'*Abstraction* les nouvelles valeurs des paramètres de l'objet après la modification et transmet une mise à jour aux *Contrôles* ***PDP*** sur les deux clients.

7 Les *Contrôles* ***PDP*** sur chacun des clients reçoivent la mise à jour avec les nouveaux paramètres de l'objet et transmettent cette mise à jour à leur *Présentation*.

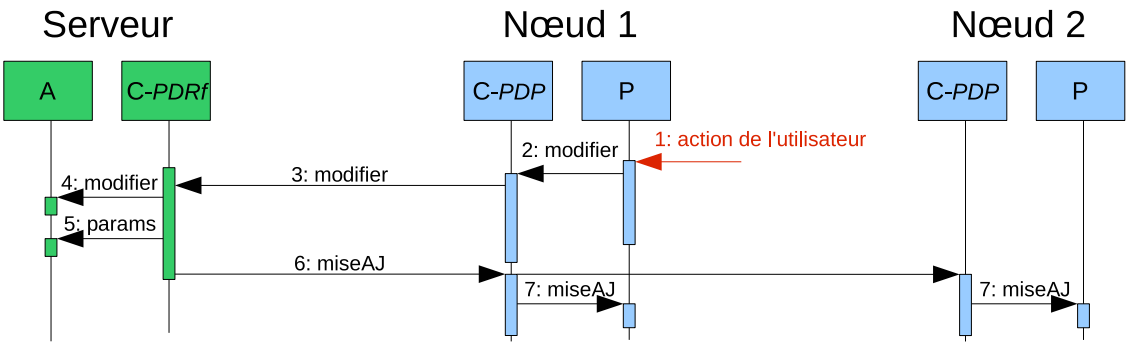


FIGURE 4.7 – Instanciation du mode centralisé avec le modèle PAC-C3D.

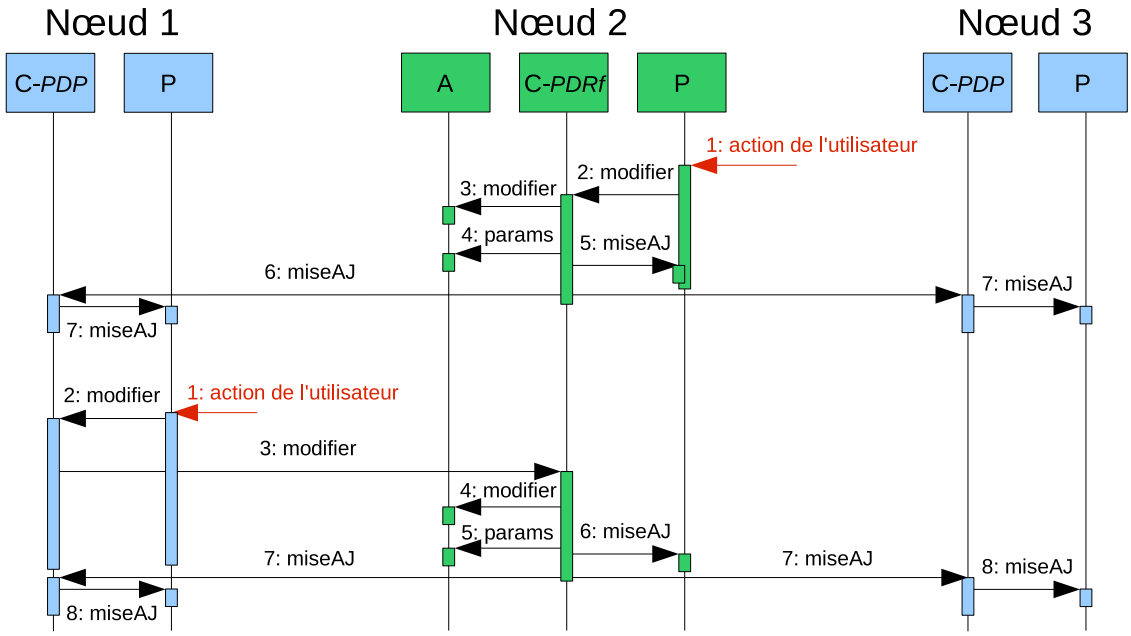


FIGURE 4.8 – Instanciation du mode hybride avec le modèle PAC-C3D.

4.3.1.2 Instanciation du mode hybride

Lorsqu'un objet utilise le mode hybride, un seul référent est situé sur un nœud particulier de la session et des proxys sont utilisés sur les autres nœuds. La figure 4.8 décrit les échanges entre les facettes PAC-C3D réparties sur trois nœuds lors d'une modification d'un objet utilisant le mode hybride. Il existe alors deux cas de figure. Premier cas, l'utilisateur qui interagit possède le référent de l'objet sur son nœud comme dans la première partie de la figure 4.8 :

1-3 L'utilisateur effectue une action sur l'objet au travers de la *Présentation* disponible sur le nœud 2. Cette *Présentation* demande à son *Contrôle* d'effectuer la modification. Comme le *Contrôle* du nœud 2 utilise la politique de distribution ***PDRf***, il demande directement à son *Abstraction* de traiter la modification.

4-6 Le *Contrôle* ***PDRf*** demande ensuite à son *Abstraction* les nouvelles valeurs des paramètres de l'objet après la modification. Il met à jour sa *Présentation*, puis transmet une mise à jour aux *Contrôles* ***PDP*** sur les deux autres nœuds.

7 Les *Contrôles* ***PDP*** sur les nœuds 1 et 3 reçoivent la mise à jour avec les nouveaux paramètres de l'objet et transmettent cette mise à jour à leur *Présentation*.

Deuxième cas, l'utilisateur qui interagit possède un proxy de l'objet sur son nœud comme dans la deuxième partie de la figure 4.8. Les échanges entre les facettes peuvent alors s'apparenter aux échanges présentés dans la partie précédente pour un objet utilisant le mode centralisé. Le nœud qui possède le *Contrôle* ***PDRf*** joue alors le rôle du serveur.

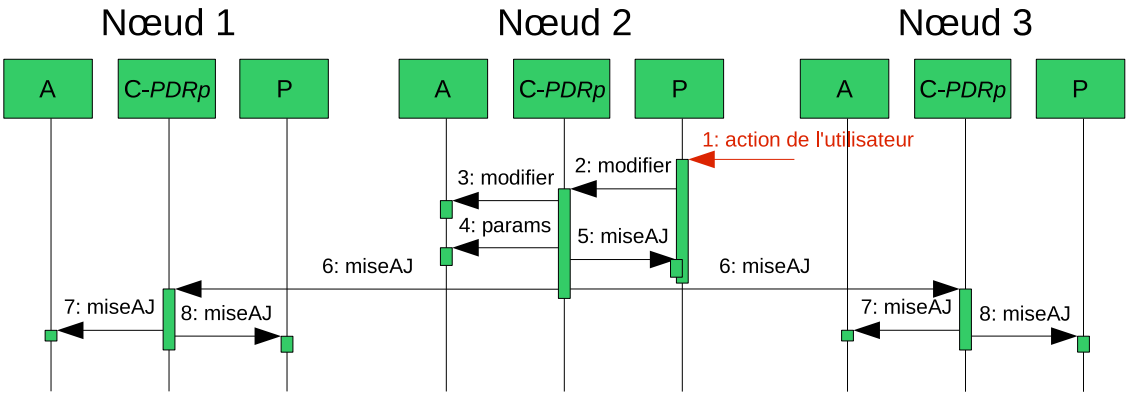


FIGURE 4.9 – Instanciation du mode répliqué avec le modèle PAC-C3D.

4.3.1.3 Instanciation du mode répliqué

Lorsqu'un objet utilise le mode répliqué, des référents sont utilisés sur tous les nœuds de la session. La figure 4.9 décrit les échanges entre les facettes PAC-C3D réparties sur trois nœuds lors d'une modification d'un objet utilisant le mode répliqué :

1-3 L'utilisateur effectue une action sur l'objet au travers de la *Présentation* disponible sur le nœud 2. Cette *Présentation* demande à son *Contrôle* d'effectuer la modification. Comme le *Contrôle* utilise la politique de distribution ***PDRp***, il demande directement à son *Abstraction* de traiter la modification.

4-6 Le *Contrôle* ***PDRp*** sur le nœud 2 demande ensuite à son *Abstraction* les nouvelles valeurs des paramètres de l'objet après la modification. Il met à jour sa *Présentation* de l'objet, puis transmet une mise à jour aux *Contrôles* ***PDRp*** sur les autres nœuds.

7-8 Les *Contrôles* ***PDRp*** sur les nœuds 1 et 3 reçoivent la mise à jour avec les nouveaux paramètres de l'objet. Ils mettent à jour leur *Abstraction* en gérant les conflits éventuels qui peuvent se produire avec des modifications locales. Ils transmettent ensuite la mise à jour à leur *Présentation*.

4.3.2 Découplage des représentations virtuelles

L'interface de la *Présentation* (**IP**) du modèle PAC-C3D garantit une forte indépendance entre les données d'un objet virtuel et les différentes *Présentations* qui lui sont associées. Cela permet de séparer distinctement la modélisation de l'environnement virtuel des différentes représentations qui en sont faites. Ainsi, chaque objet du monde virtuel peut être associé facilement à plusieurs types de *Présentation*. Chacune de ces *Présentations* permet de représenter l'objet d'une façon différente en utilisant des composants logiciels particuliers et adaptés aux dispositifs matériels que les utilisateurs utilisent. En plus de maintenir la cohérence entre l'*Abstraction* et les *Présentations*, le *Contrôle* permet également de maintenir la cohérence entre les différentes *Présentations*.

Premièrement, un même objet peut avoir plusieurs types de *Présentation* sur le même nœud. Cela permet de mettre en place une restitution multi-sensorielle de l'objet en associant cet objet à une *Présentation* pour le système de rendu graphique 3D, à une *Présentation* pour la librairie audio, à une *Présentation* pour le moteur physique en charge des retours haptiques, etc. (cf.figure 4.10).

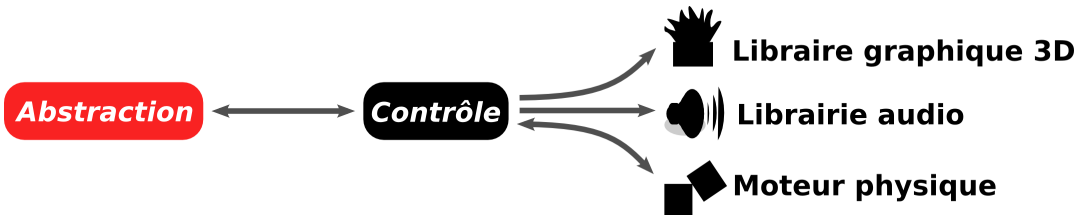


FIGURE 4.10 – Plusieurs *Présentations* pour un même objet virtuel sur le même nœud.

Deuxièmement, un même objet peut avoir une *Présentation* différente sur chacun des nœuds de la session. Cela permet d'utiliser des *Présentations* associées à des composants logiciels différents en fonction de ce qui est le plus adapté aux dispositifs matériels des utilisateurs. Le modèle PAC-C3D permet non seulement d'éviter de dupliquer le code du comportement des objets dans chaque composant logiciel, mais il permet aussi à ces composants logiciels de communiquer entre eux. Il assure donc en quelque sorte une interopérabilité entre ces différents composants logiciels. Si un utilisateur interagit par l'intermédiaire d'un de ces composants, les modifications seront répercutées dans tous les autres composants sur les autres nœuds. Grâce à cette interopérabilité, il est également envisageable qu'un des composants logiciels associés à une *Présentation* puisse utiliser des fonctionnalités disponibles uniquement dans un autre de ces composants.

Par ailleurs, certaines *Présentations* « actives » peuvent effectuer des modifications sur l'objet qui leur est associé (sans l'intervention des utilisateurs). Par exemple, un moteur physique peut agir sur la position de l'objet en fonction des contacts de cet objet avec les autres objets du monde virtuel. Si plusieurs *Présentations* « actives » sont utilisées pour un même objet, il peut être nécessaire que le *Contrôle* gère les modifications concurrentes faites par ces différentes *Présentations*.

Enfin, le système d'interaction qui permet aux utilisateurs d'agir sur les objets virtuels au travers de différents périphériques peut être vu comme un ensemble particulier de *Présentations*. En effet, chaque objet interactif sur lequel les utilisateurs peuvent agir est associé à une *Présentation* liée à un périphérique d'interaction particulier. Cette *Présentation* transmet les actions des utilisateurs sur le périphérique au *Contrôle* de cet objet interactif (cf. figure 4.11). Ce lien entre l'objet virtuel et le périphérique d'interaction est décrit dans un fichier de configuration.

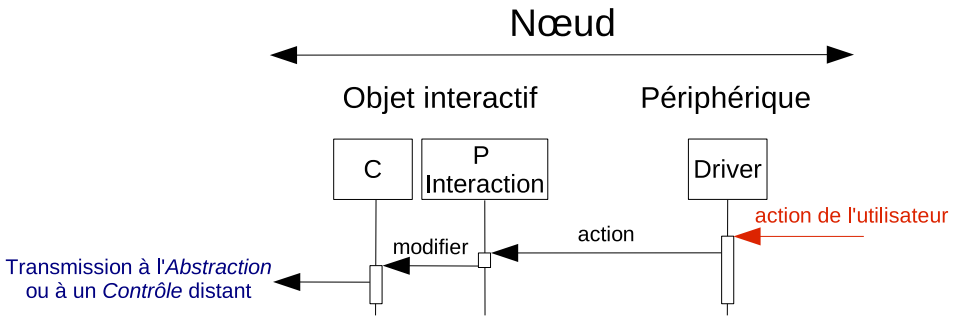


FIGURE 4.11 – *Présentation* liée à un périphérique d'interaction.

4.3.3 Gestion des objets virtuels

Chacun des objets présents dans le monde virtuel est modélisé par un agent PAC-C3D. Pour gérer l'ensemble de ces objets, nous utilisons un gestionnaire d'objets qui est également structuré en utilisant le modèle PAC-C3D. Ce gestionnaire d'objets est donc décomposé en trois facettes chacune décrite par une interface :

- l'*Abstraction* stocke des données simples sur le monde virtuel (nombre d'objets, etc.),
- le *Contrôle* conserve la liste des *Contrôles* de tous les objets virtuels et gère les demandes de création et de suppression d'objet dans le monde virtuel,
- chacune des *Présentations* propose une représentation particulière du monde virtuel et stocke les *Présentations* de tous les objets virtuels associées à cette représentation. Par exemple, pour une librairie graphique 3D, la *Présentation* du gestionnaire d'objets correspond au graphe de scène du monde 3D et les *Présentations* des objets virtuels correspondent aux éléments du graphe de scène qui représentent graphiquement ces objets. De même, pour un moteur physique, la *Présentation* du gestionnaire d'objets correspond au monde physique et les *Présentations* des objets virtuels correspondent aux entités physiques qui seront utilisées pour la simulation.

Les *Abstractions* des objets virtuels ont accès au *Contrôle* du gestionnaire d'objets et donc à la liste des *Contrôles* de tous les autres objets du monde virtuel. Ainsi, chaque objet peut agir ou communiquer avec les autres en accédant à leur *Contrôle*. Cela garantit une interopérabilité entre les objets même s'ils utilisent des modes de distribution différents.

4.3.4 Création des objets virtuels

Afin de permettre à l'environnement virtuel d'évoluer dynamiquement, chacune des facettes PAC du gestionnaire d'objets possède une *factory*. Cette *factory* est capable de créer la facette appropriée en fonction du type de l'objet. Nous avons expliqué dans la partie précédente que les *Abstractions* de chaque objet virtuel ont accès au *Contrôle* du gestionnaire d'objets. Lorsqu'un objet virtuel veut créer un nouvel objet, son *Abstraction* demande au *Contrôle* du gestionnaire d'objets de créer ce nouvel objet.

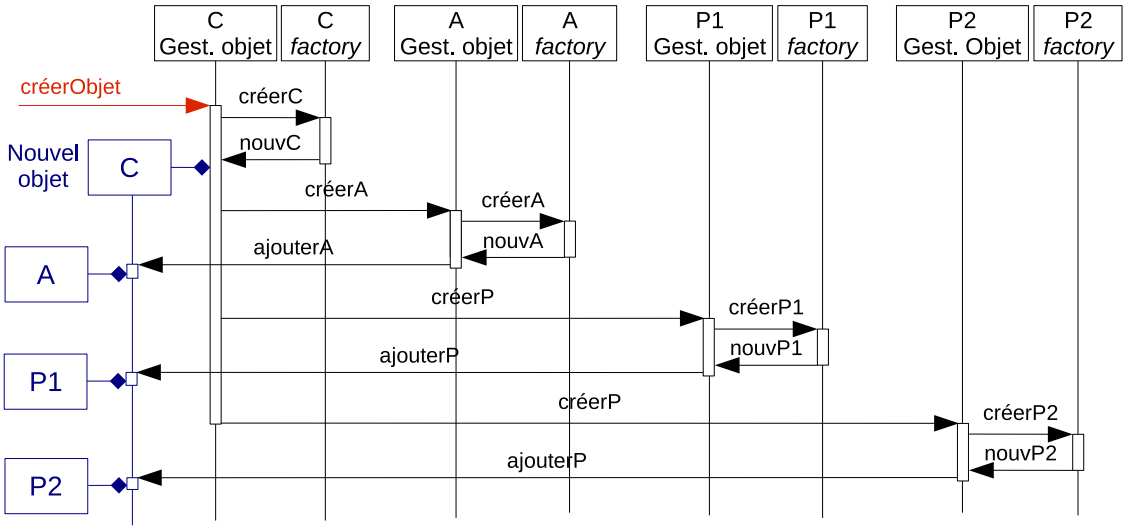


FIGURE 4.12 – Création des différentes facettes PAC-C3D à l'aide des différentes *factory*.

Lorsqu'un *Contrôle* du gestionnaire d'objets reçoit une demande de création, il transmet d'abord la demande de création à tous les autres nœuds de la session. Sur chacun des nœuds (cf. figure 4.12), le *Contrôle* du gestionnaire d'objets demande à sa *factory* de créer le *Contrôle* du nouvel objet. Si l'objet a besoin d'une *Abstraction* sur ce nœud, le *Contrôle* du gestionnaire d'objets transmet la demande de création, ainsi que le nouveau *Contrôle* de l'objet à son *Abstraction*. L'*Abstraction factory* se charge de créer l'*Abstraction* adéquate pour le nouvel objet et de la lier à son *Contrôle*. De même, le *Contrôle* du gestionnaire d'objets transmet ensuite la demande de création et le *Contrôle* du nouvel objet à chacune de ses différentes *Présentations*. Chaque *Présentation factory* se charge de créer la *Présentation* adaptée à ce nouvel objet, de la lier à son *Contrôle* et de l'ajouter dans la représentation de l'environnement virtuel (graphe de scène, monde physique, etc.).

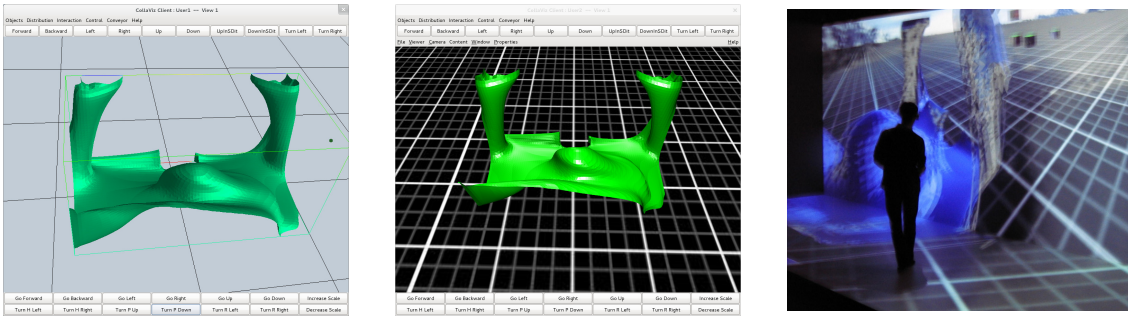
4.4 Utilisation de PAC-C3D dans le cadre du projet Collaviz

Une des contraintes du projet ANR Collaviz est que le *framework* utilisé puisse être facilement déployé sur des dispositifs matériels variés allant de l'ordinateur portable au système immersif en passant par les *smartphones*. Il a été décidé que tous les développements du projet seraient réalisés en Java afin de permettre un déploiement facile sur des systèmes d'exploitation différents. Java a également été choisi car il offre la possibilité d'exécuter les applications sur les machines des utilisateurs sans faire aucune installation préalable *(Java Web Start)*.

Pour répondre à cette contrainte lors de la conception de notre *framework* Collaviz, nous avons utilisé le modèle d'architecture logicielle PAC-C3D. Les parties *Abstraction* et *Contrôle* des objets virtuels et du gestionnaire d'objets ont été développées en Java afin de garantir une forte indépendance par rapport au système d'exploitation utilisé. Cependant, il n'existe pas de librairie graphique en Java qui soit adaptée à tous les types de dispositif de visualisation à la fois. Nous avons donc choisi d'utiliser des librairies graphiques différentes afin de s'adapter à plusieurs types de dispositif de visualisation. Dans la partie 4.4.1, nous expliquons comment le modèle PAC-C3D a permis d'intégrer ces différentes librairies graphiques dans le *framework* Collaviz et, dans la partie 4.4.2, nous détaillons comment ce modèle rend possible l'interopérabilité entre ces différentes librairies graphiques. Nous avons également choisi d'intégrer un moteur physique afin de détecter les collisions entre les objets du monde virtuel. Cela permet, par exemple, de faciliter le placement d'annotations dans des données scientifiques ou de proposer de nouvelles façons d'explorer ces données. Nous présentons dans la partie 4.4.3, la façon d'intégrer cette *Présentation* « active » grâce à PAC-C3D. Enfin, dans la partie 4.4.4, nous détaillons les différents retours sensoriels qui ont été associés à un outil d'interaction pour faciliter l'exploration de données scientifiques dans le cadre d'une collaboration avec le chercheur Ian Grimstead de l'université de Cardiff.

4.4.1 Intégration de différentes librairies graphiques

Notre *framework* Collaviz intègre trois librairies graphiques Java. Ces trois instanciations des librairies graphiques permettent d'avoir trois *viewers* différents pour visualiser l'environnement virtuel (cf. figure 4.13). Chacune de ces librairies graphiques possède des caractéristiques particulières et les utilisateurs peuvent ainsi choisir le *viewer* qui est le mieux adapté aux dispositifs matériels qu'ils utilisent :



(a) (b) (c)

FIGURE 4.13 – *Viewers* (a) Java3D, (b) jMonkeyEngine et (c) jReality du *framework* Collaviz permettant de visualiser le même monde virtuel.

- **Java3D** [1] est bien adapté pour les simples stations de travail et il permet en plus de faire du *Java Web Start* (pas d'installation nécessaire),
- **jMonkeyEngine** [2] peut être utilisé pour les stations de travail, mais il a surtout la particularité de pouvoir être déployé facilement sur les périphériques mobiles (*smartphone*, tablette, etc.) utilisant le système Android [3].
- **jReality** [4] est bien adapté pour les dispositifs immersifs car il propose une distribution du graphe de scène sur plusieurs machines (qui peuvent gérer les différents écrans d'une salle immersive), de la stéréovision active, ainsi qu'une déformation de la pyramide de vue en fonction de la position de la tête de l'utilisateur. De plus, jReality intègre également une librairie audio pour sonoriser les objets virtuels.

Pour chacune de ces librairies graphiques, un ensemble de *Présentations* a été développé afin de compléter la modélisation PAC-C3D des objets virtuels et du gestionnaire d'objets. Chaque objet du monde virtuel a donc une *Présentation* visuelle pour chaque librairie graphique, plus une *Présentation* sonore si nous voulons que l'objet ait un son dans jReality. Cette multiplication des *Présentations* peut sembler être un coût de développement supplémentaire. Cependant, avec la notion d'héritage entre les objets, il n'est pas toujours nécessaire de développer une nouvelle *Présentation* pour chaque nouvel objet. En effet, il est souvent possible de réutiliser la *Présentation* de l'objet dont le nouvel objet hérite. Il existe principalement trois *Présentations* importantes pour chaque librairie graphique :

- la *Présentation* du gestionnaire d'objets qui instancie le graphe de scène,
- la *Présentation* d'un objet virtuel de base qui définit sa représentation graphique,
- la *Présentation* d'une caméra virtuelle qui définit d'où et comment le rendu graphique doit être fait.

En conséquence, il faut créer une *Présentation* pour un nouveau type d'objet uniquement lorsque ce type d'objet a des besoins spécifiques en termes de rendu graphique, comme par exemple pour une lumière dans le monde virtuel.

1. `http://java3d.java.net`
2. `http://jmonkeyengine.com`
3. `http://www.android.com`
4. `http://www3.math.tu-berlin.de/jreality`

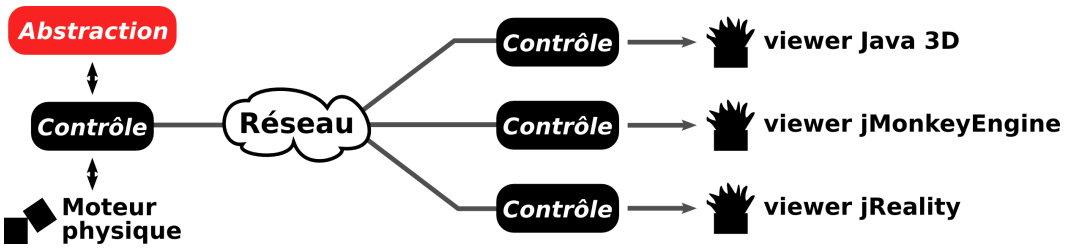


FIGURE 4.14 – Différents *Présentations* pour un même objet sur des nœuds différents.

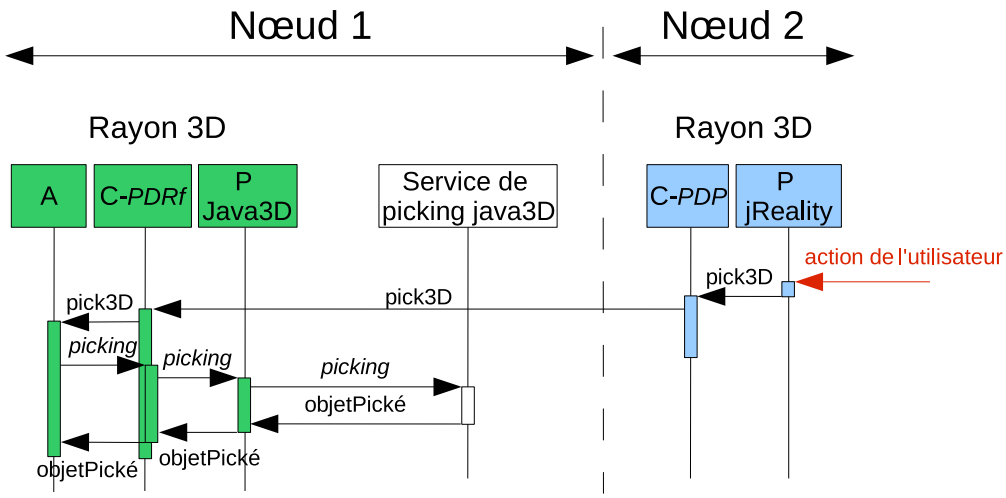


FIGURE 4.15 – Utilisation du service de *picking* fournit par Java3D.

4.4.2 Interopérabilité entre différentes librairies graphiques

Grâce à l'intégration de ces trois librairies graphiques dans le *framework* Collaviz, chaque utilisateur d'une même session collaborative peut choisir le *viewer* qui est le plus adapté aux dispositifs matériels qu'il utilise indépendamment des *viewers* choisis par les autres utilisateurs (cf. figure 4.14). Le modèle PAC-C3D assure l'interopérabilité entre ces trois *viewers* : toute modification faite dans un des *viewers* sera automatiquement répercutée dans les autres. De plus, il est possible pour un *viewer* d'utiliser des fonctionnalités supplémentaires disponibles dans les autres *viewiers*. Pour cela, il faut que le référent de l'objet soit sur le nœud dont le *viewer* possède la fonctionnalité. C'est le cas, par exemple, pour le *picking* 3D. Afin de ne pas re-développer nous même un service de *picking* dans la partie principale de l'environnement virtuel, nous avons choisi d'utiliser ceux proposés par les librairies graphiques et donc de demander à la *Présentation* graphique de faire le *picking*. Dans un premier temps, nous n'avions pas implémenté de service de *picking* dans notre *viewer* jReality, mais nous pouvions quand même bénéficier d'un service de *picking* dans jReality en déplaçant le référent de l'objet qui avait besoin du *picking* sur un nœud qui possédait un *viewer* Java3D et qui pouvait ainsi demander à la *Présentation* Java3D d'effectuer le *picking* (cf. figure 4.15). Par la suite, ce service a été implémenté afin de pouvoir faire des déploiements jReality sans être obligé de lancer simultanément un client utilisant le *viewer* Java3D.

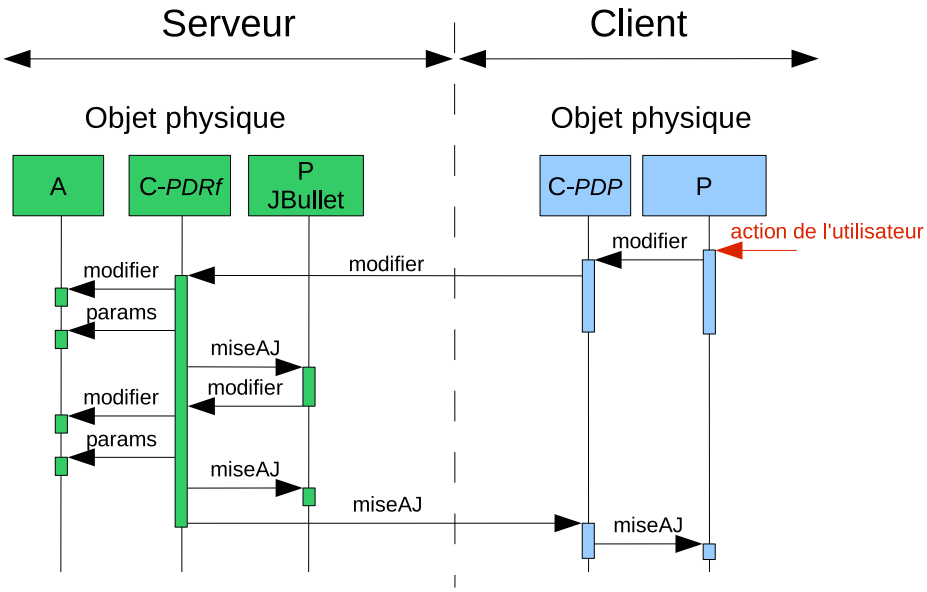


FIGURE 4.16 – Modification de l'objet par une *Présentation* « active » de JBullet.

4.4.3 Intégration d'un moteur physique

Afin de rendre possible la détection de collisions entre les objets du monde virtuel, le moteur physique JBullet [5], qui est le portage Java de Bullet, a été intégré au *framework* Collaviz. Pour cela, un ensemble de *Présentations* particulières a été créé pour faire le lien avec JBullet. Afin d'éviter les conflits entre plusieurs simulations physiques, cet ensemble de *Présentations* peut seulement être instancié sur un nœud. Dans le cas idéal, il doit être instancié sur le serveur s'il possède une grosse puissance de calcul (cf. figure 4.14), mais il peut également être instancié sur un autre nœud si besoin.

La *Présentation* du gestionnaire d'objets correspond au monde physique de JBullet et elle exécute la simulation physique. Lorsqu'un objet est créé avec des propriétés physiques associées, la *Présentation* physique du gestionnaire d'objets crée une *Présentation* physique pour cet objet grâce à sa *factory* et l'ajoute dans le monde physique. Cette *Présentation* « active » de l'objet peut modifier les données relatives à l'objet (en passant par son *Contrôle*) afin d'appliquer à l'objet les collisions et les contraintes physiques (cf. figure 4.16). Ainsi, à chaque pas de temps, la *Présentation* physique de l'objet est d'abord mise à jour en fonction des actions des utilisateurs. Puis, le moteur physique est prévenu qu'un pas de temps de simulation a été exécuté afin qu'il recalcule les contacts et les forces appliqués à l'entité physique de l'objet, ainsi que sa nouvelle position. Enfin, la *Présentation* physique de l'objet renvoie une demande de modification au *Contrôle* avec la position recalculée afin de mettre à jour l'*Abstraction* et les autres *Présentations* « passives » de cet objet. De cette façon, il est possible d'intégrer un moteur physique dans un environnement virtuel collaboratif et faire percevoir son action dans tous les *viewers* utilisés par les utilisateurs.

5. `http://jbullet.advel.cz`

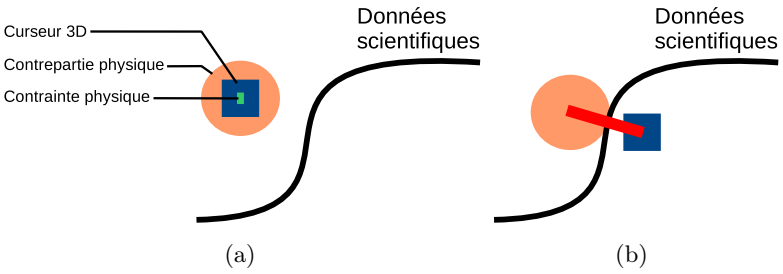


FIGURE 4.17 – Curseur 3D et sa contrepartie physique : (a) libre ou (b) en collision avec les données scientifiques.

FIGURE 4.18 – 2 LEDs éclairées sur la *Wiimote*.

4.4.4 Association de différents retours sensoriels

Afin de faciliter l'exploration de données scientifiques, nous avons proposé, en collaboration avec le chercheur Ian Grimstead de l'université de Cardiff, d'associer plusieurs retours sensoriels à l'outil d'interaction utilisé par les utilisateurs. Ce couplage avec plusieurs représentations virtuelles est rendu possible grâce au modèle PAC-C3D (cf.figure 4.10).

Pour interagir avec les données scientifiques, un utilisateur manipule un curseur 3D grâce à une *Wiimote* de la console *Wii* de Nintendo®. Cette *Wiimote* permet de capter les actions de l'utilisateur, mais aussi de mettre en place des retours sensoriels supplémentaires. Pour interagir à partir d'un dispositif immersif, cette *Wiimote* peut également être repérée par un système de *tracking* afin d'avoir un champ d'action plus important.

Pour détecter les contacts entre le curseur 3D et les données scientifiques, une contrepartie physique est associée au curseur 3D grâce à une contrainte physique. Cette contrepartie est un objet physique qui peut entrer en contact avec les éléments du monde virtuel et en particulier avec les données scientifiques. Lorsque le curseur 3D « passe » au travers d'un objet, la contrainte physique entre l'objet et sa contrepartie ne peut plus être maintenue, et la distance entre les deux parties augmente (cf. figure 4.17).

La contrainte physique est modélisée comme un objet PAC-C3D particulier. Elle est paramétrée par le curseur 3D et par la contrepartie physique qui lui est associée. La contrainte physique calcule la distance entre ces deux objets. D'une part, elle est associée à une *Présentation* dans le monde physique de JBullet qui permet d'instancier réellement une contrainte dans le moteur physique. D'autre part, cette contrainte physique peut aussi être associée à des *Présentations* supplémentaires afin de mettre en œuvre, en fonction de la distance entre les deux objets, plusieurs retours sensoriels pour les utilisateurs :

- **Un retour visuel :** un cylindre est affiché dans le monde virtuel afin de faire le lien entre le curseur 3D et sa contrepartie physique. Lorsque la distance entre les deux objets est faible, le cylindre est de couleur verte. Si cette distance augmente, alors la couleur du cylindre vire progressivement au rouge, avant de devenir totalement rouge lorsque le distance dépasse un seuil pré-déterminé.
- **Un retour sonore :** la librairie audio de jReality a été utilisée pour associer un son à la contrainte. Lorsque la distance associée à la contrainte est inférieure à un seuil, aucun son n'est émis par le système. Puis, plus cette distance augmente, plus le son émis par jReality devient important.

- **Un retour lumineux :** les quatre LEDs disponibles sur la *Wiimote* ont permis d'offrir un retour lumineux aux utilisateurs (cf. figure 4.18). Lorsque la distance associée à la contrainte est inférieure à un seuil, toutes les LEDs sont éteintes. Puis, au fur et à mesure que la distance augmente, le nombre de LEDs allumées augmente.
- **Un retour haptique :** la *Wiimote* a également permis d'offrir un retour haptique grâce aux vibrations qu'elle permet de générer. Lorsque la distance associée à la contrainte est inférieure à un seuil, la *Wiimote* ne vibre pas. Puis, plus cette distance augmente, plus les vibrations de la *Wiimote* deviennent importantes. Ce type de retour offre un moyen simple de ressentir la présence des données scientifiques sans avoir besoin d'utiliser des dispositifs à retour d'effort complexes.

Ces retours sensoriels offrent aux utilisateurs des informations supplémentaires pour mieux percevoir quand leur curseur 3D entre en contact avec la surface des données scientifiques. Le découpage de ces retours sensoriels en un ensemble de *Présentations* séparées offre une bonne modularité : chaque utilisateur peut choisir d'associer une ou plusieurs *Présentations* à la contrainte physique en fonction des retours sensoriels souhaités. Cette association entre les objets virtuels et leurs *Présentations* est décrite dans un fichier de configuration.

4.5 Conclusion

Le modèle d'architecture logicielle PAC-C3D propose une extension du modèle PAC pour adapter ce modèle aux environnements virtuels collaboratifs 3D. Chaque objet du monde virtuel peut être modélisé par un agent PAC-C3D sur chacun des nœuds qui participe à la session collaborative. Chaque agent est composé de trois facettes rendues indépendantes par un ensemble d'interfaces. L'*Abstraction* est en charge des données relatives à l'objet, ainsi que de son comportement. Les *Présentations* correspondent aux différentes représentations virtuelles de cet objet. Enfin, le *Contrôle* est en charge du maintien de la cohérence entre l'*Abstraction* et les *Présentations*, mais aussi entre les différentes versions de l'objet sur chacun des nœuds.

Grâce à cette décomposition, le modèle PAC-C3D permet de mettre en œuvre le paradigme de référents et de proxys, et d'instancier le modèle d'adaptation dynamique de la distribution des données, proposé dans le chapitre précédent. Trois politiques de distribution au niveau du *Contrôle* permettent d'implémenter les trois modes de distribution des données proposés. La politique de distribution du *Contrôle* peut être choisie de façon indépendante pour chaque objet virtuel et modifiée de façon dynamique durant la session, ce qui correspond parfaitement aux besoins de notre modèle de distribution des données.

Par ailleurs, le modèle PAC-C3D permet de modéliser l'environnement virtuel indépendamment des représentations virtuelles qui lui sont associées. Cela permet d'assurer un fort découplage entre la modélisation de l'environnement virtuel et les composants logiciels qui sont utilisés pour le restituer aux utilisateurs. D'une part, un objet peut avoir plusieurs *Présentations* différentes sur un même nœud afin d'offrir une restitution multi-sensorielle aux utilisateurs. D'autre part, un objet peut avoir également des *Présentations* différentes sur les différents nœuds de la session afin de choisir celles qui sont les mieux adaptées aux dispositifs matériels utilisés. Dans le cadre du projet ANR Collaviz, ces deux types d'instanciation de *Présentations* multiples ont été utilisés. Premièrement, un curseur 3D utilisant plusieurs retours sensoriels différents a été mis en place pour explorer des données scientifiques. Deuxièmement, le *framework* Collaviz intègre trois librairies graphiques

(Java3D, jReality et jMonkeyEngine) qui permettent de visualiser l'environnement virtuel à partir de dispositifs matériels différents. Le moteur physique JBullet a également été intégré au *framework* Collaviz. Le modèle PAC-C3D permet d'assurer l'interopérabilité entre ces différentes librairies. Certaines de ces librairies ont été développés en collaboration avec un stagiaire de Master 2 (*viewer* jReality) et deux ingénieurs (*viewer* jMonkeyEngine et intégration du moteur physique JBullet).

L'intégration du moteur physique JBullet dans le *framework* Collaviz a également permis de montrer qu'il est possible d'utiliser des *Présentations* « actives ». En effet, ces *Présentations* physiques permettent d'ajouter un comportement physique aux objets virtuels. Cependant, si un objet virtuel possède plusieurs *Présentations* « actives », il risque d'avoir des conflits entre les modifications proposées par ces différentes *Présentations*. Pour aller plus loin dans l'intégration de *Présentations* « actives », il faudrait trouver un moyen de gérer les priorités entre ces différentes *Présentations* afin d'éviter les conflits.

Chapitre 5

Modèle d'intégration des espaces d'interaction réels des utilisateurs

Pour offrir aux utilisateurs des interactions adaptées, de nombreuses applications de réalité virtuelle doivent tenir compte de l'environnement réel qui entoure chacun des utilisateurs. Pour chaque utilisateur, cet environnement réel peut être caractérisé par un ensemble de volumes 3D réels, associés aux dispositifs matériels qu'il utilise. Chaque volume 3D réel, que nous appelons « espace d'interaction », définit les possibilités de perception ou d'interaction d'un utilisateur au travers d'un dispositif matériel particulier (et donc au niveau d'un canal sensoriel spécifique). Dans le chapitre 1, nous avons vu que certaines applications nécessitent d'intégrer l'espace d'interaction d'un dispositif matériel particulier, dans l'environnement virtuel. Par exemple, le *Hand Held Display (HHD)* [Amselem, 1995] nécessite de modéliser l'espace de visualisation associé à un écran mobile, la technique de la *Bubble* [Dominjon *et al.*, 2005] considère l'espace d'interaction d'un dispositif à retour d'effort, et le modeleur graphique 3DM [Butterworth *et al.*, 1992] intègre l'espace de déplacement physique de l'utilisateur repéré par un capteur magnétique (l'espace de déplacement physique est défini par la portée du capteur magnétique). Selon les applications,

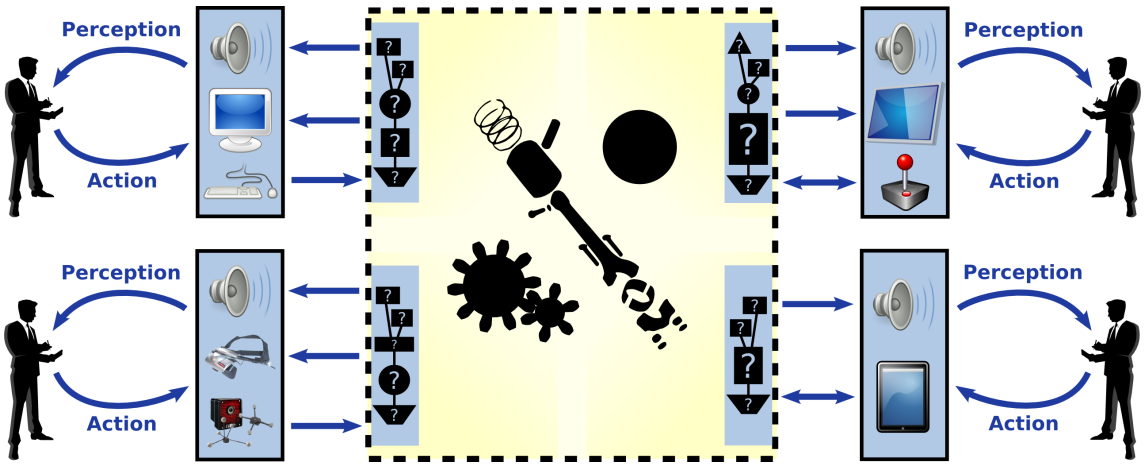


FIGURE 5.1 – Comment intégrer les espaces d'interaction réels de chaque utilisateur dans l'environnement virtuel ?

le fait d'intégrer ces espaces d'interaction dans l'environnement virtuel permet :

1. d'adapter les interactions et la collaboration entre utilisateurs en fonction de leurs possibilités de perception et d'interaction,
2. de faire comprendre à chaque utilisateur quelles sont ses propres possibilités de perception et d'interaction, mais aussi quelles sont celles des autres utilisateurs dans le cas d'un environnement virtuel collaboratif.

Cependant, ces applications ne décrivent explicitement ni l'existence, ni les caractéristiques de tels espaces d'interaction. L'intégration de ces espaces est souvent faite au cas par cas et il n'existe pas de solution générique qui permette de traiter de façon homogène les deux points cités précédemment.

Même si d'un point de vue technique, le modèle d'architecture logicielle PAC-C3D présenté au chapitre précédent permet d'adapter une application de réalité virtuelle aux différents dispositifs matériels et aux composants logiciels qui leur sont associés, ce modèle ne prend pas en compte les aspects conceptuels au niveau de l'interaction des utilisateurs. Il en est de même pour les solutions permettant de s'abstraire des dispositifs matériels dans les systèmes de réalité virtuelle, présentées dans la partie 1.1.3.1.

Dans la partie 1.1.3.2, nous avons vu qu'il existe également des solutions pour intégrer certains dispositifs matériels dans l'environnement virtuel. En particulier, le concept de *Cabine Virtuelle d'Immersion* [Duval et Chauffaut, 2006] permet de coupler la navigation et les autres interactions dans un environnement virtuel collaboratif multi-échelle. Ce concept, qui a été l'initiateur des travaux présentés dans cette section, propose de repérer les outils d'interaction dans le repère de l'utilisateur afin qu'il puisse naviguer avec, dans le monde virtuel. Cependant, ce concept ne décrit pas l'environnement réel de chaque utilisateur comme un ensemble d'espaces d'interaction et, en conséquence, il ne propose pas, non plus, de solutions pour intégrer ces espaces dans l'environnement virtuel.

Enfin, comme présenté dans la partie 1.1.3.3, [Mulder et Boschker, 2004] propose une première description d'espaces d'interaction multi-sensorielles. Cependant, cette description est très spécifique à un dispositif matériel particulier et considère seulement un espace de visualisation et un espace d'interaction. Elle ne propose pas de structure générique pour intégrer ces espaces d'interaction dans l'environnement virtuel.

Dans ce chapitre, nous proposons donc un modèle appelé *Cabine Virtuelle d'Interaction Immersive* (CVII) qui intègre de façon générique les espaces d'interaction réels des différents utilisateurs, dans un environnement virtuel collaboratif [Fleury *et al.*, 2010a]. Dans la partie 5.1, nous expliquons d'abord le concept de CVII. Dans la partie 5.2, nous décrivons comment la CVII modélise et intègre les espaces d'interaction des utilisateurs dans l'environnement virtuel. Dans la partie 5.3, nous présentons les différentes fonctionnalités pour l'interaction et la collaboration qui peuvent être mises en œuvre grâce à la CVII. Enfin, dans la partie 5.4, nous détaillons certaines mises en œuvre intéressantes qui utilisent le modèle de CVII.

5.1 Concept de *Cabine Virtuelle d'Interaction Immersive* (CVII)

La *Cabine Virtuelle d'Interaction Immersive* propose une solution générique pour tenir compte des espaces d'interaction des utilisateurs lors de la conception et de l'utilisation d'un système de réalité virtuelle. Elle propose une modélisation de haut niveau pour décrire, organiser et contrôler ces espaces d'interaction dans un environnement virtuel collaboratif

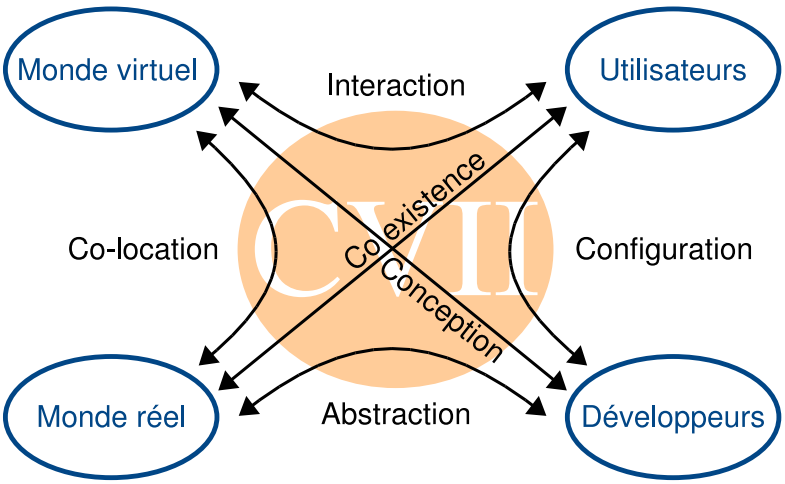


FIGURE 5.2 – Le concept de *Cabine Virtuelle d'Interaction Immersive*.

quels que soient les dispositifs matériels utilisés par les utilisateurs. La CVII permet de faire le lien entre le monde réel et le monde virtuel, mais elle peut aussi être vue comme faisant le lien entre les utilisateurs finaux et les concepteurs de l'application. La CVII peut ainsi être caractérisée par les six termes suivants (cf. figure 5.2) :

- **Co-existence :** les utilisateurs se situent dans le monde réel où ils peuvent agir sur les objets réels et sur les dispositifs d'interaction.
- **Conception :** les développeurs conçoivent l'environnement virtuel et choisissent les techniques d'interaction et de collaboration qui seront utilisées.
- **Interaction :** les utilisateurs interagissent dans l'environnement virtuel afin de réaliser des tâches collaboratives avec les autres utilisateurs.
- **Co-localisation :** l'espace 3D du monde réel est mis en relation avec l'espace 3D du monde virtuel afin de lier les objets réels à une représentation dans l'environnement virtuel (cf. partie 1.1.1.2).
- **Abstraction :** pour concevoir de façon générique les applications de réalité virtuelle, les développeurs peuvent s'abstraire des dispositifs matériels utilisés par les utilisateurs dans le monde réel.
- **Configuration :** les développeurs peuvent configurer les applications de réalité virtuelle en fonction des besoins des utilisateurs.

5.2 Modèle de CVII

Le modèle de *Cabine Virtuelle d'Interaction Immersive* est basé sur une hiérarchisation des espaces d'interaction des utilisateurs. Nous décrivons cette hiérarchisation dans la partie 5.2.1 et la façon dont elle est intégrée dans l'environnement virtuel dans la partie 5.2.2. Dans la partie 5.2.3, nous présentons la structure d'un point de vue logiciel de la CVII. Enfin, dans la partie 5.2.4, nous détaillons les opérateurs qui permettent de manipuler cette structure.

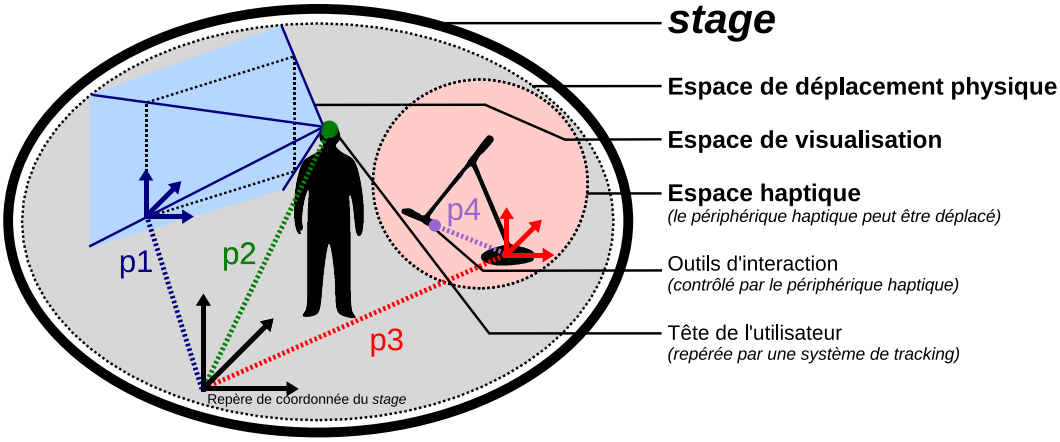


FIGURE 5.3 – Le *stage* regroupe les différents espaces d'interaction de l'utilisateur.

5.2.1 Hiérarchisation des espaces d'interaction

La CVII propose de modéliser l'environnement réel de chaque utilisateur comme une hiérarchie d'espaces d'interaction. Un espace d'interaction peut être défini comme un volume fonctionnel qui décrit l'espace 3D dans lequel l'utilisateur va pouvoir percevoir le monde virtuel ou agir dessus grâce à un dispositif matériel particulier. Chaque type de dispositif matériel a donc un espace d'interaction particulier associé :

- **L'espace de déplacement physique** correspond à l'espace dans lequel l'utilisateur peut déplacer son corps dans un dispositif immersif. Il est souvent défini par la position des écrans de visualisation ou par les limites du système de *tracking*.
- **L'espace de visualisation** ne correspond pas seulement à la surface des écrans, mais aussi à l'espace de l'environnement virtuel qui peut être vu par l'utilisateur. Il est souvent défini par la pyramide qui part de sa tête et qui passe par les écrans.
- **L'espace sonore** correspond à l'espace dans lequel l'utilisateur peut percevoir les sons du monde virtuel en fonction des dispositifs de restitution sonore qu'il utilise.
- **L'espace d'interaction** d'un périphérique correspond à l'espace dans lequel l'utilisateur peut agir sur le monde virtuel en utilisant ce périphérique.
- **L'espace haptique** correspond à l'espace dans lequel l'utilisateur peut ressentir les efforts du monde virtuel grâce à un dispositif à retour d'effort. Ce type de dispositif a généralement un espace d'interaction relativement restreint (cf. figure 1.5).

Chaque espace d'interaction peut contenir des objets virtuels ou des représentations virtuelles associées à des objets réels en fonction de ses propriétés sensorielles. De plus, chaque espace d'interaction peut inclure à l'intérieur de son volume d'autres espaces d'interaction d'où la notion de hiérarchie. Par exemple, si un utilisateur peut déplacer un dispositif haptique dans une salle immersive, l'espace haptique de ce dispositif sera inclus dans l'espace de déplacement physique associé à la salle. L'espace d'interaction qui définit la base de la hiérarchie est appelé *stage* et il englobe tous les autres espaces d'interaction d'un même utilisateur (cf. figure 5.3). Tous les espaces d'interaction sont repérés directement par rapport au *stage* ou indirectement par rapport à un autre espace d'interaction inclus dans le *stage*. Ainsi, cette modélisation des espaces d'interaction les uns par rapport aux autres permet de décrire l'organisation de l'environnement réel de chaque utilisateur.

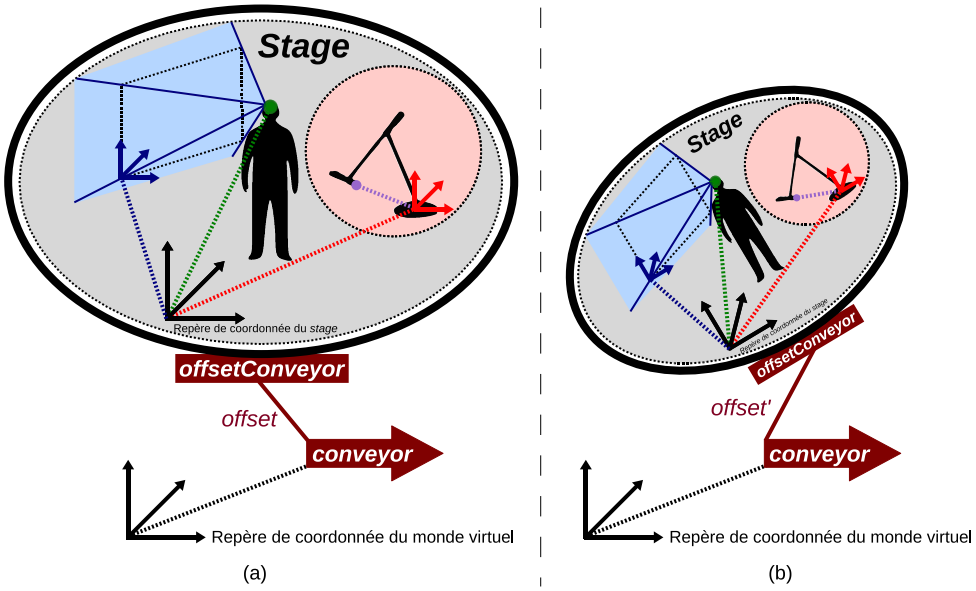


FIGURE 5.4 – En (a), le *conveyor* et l'*offsetConveyor* intègrent le *stage* dans le monde virtuel. En (b), l'*offset* entre le *conveyor* et l'*offsetConveyor* a été modifié au niveau de la position, de l'orientation et de l'échelle.

5.2.2 Intégration des espaces d'interaction dans l'environnement virtuel

La CVII peut être considérée comme une abstraction de l'environnement réel de chaque utilisateur afin de permettre aux développeurs de concevoir des applications de réalité virtuelle indépendamment des dispositifs matériels utilisés. Les développeurs ont seulement à considérer une instance générique de la CVII au moment de la conception, et cette instance sera spécifiée avec la description des espaces d'interaction de chaque utilisateur au moment de la configuration. Pour intégrer cette instance de la CVII dans l'environnement virtuel de façon générique, le modèle de CVII se base sur deux composants principaux :

- **Le *stage* :** comme nous l'avons expliqué dans la partie précédente, le *stage* regroupe l'ensemble des espaces d'interaction d'un utilisateur et les localise dans son propre repère de coordonnées. Les développeurs doivent donc seulement prendre en compte la notion de *stage* lors de la conception d'un application sans avoir à connaitre les espaces d'interaction qui seront inclus à l'intérieur lors de la configuration.
- **Le *conveyor* :** c'est le point d'intégration du *stage* dans l'environnement virtuel. Le *conveyor* a une position, une orientation et une échelle dans le monde virtuel, et il définit la technique de navigation utilisée dans l'environnement virtuel. Afin de pouvoir implémenter les différentes techniques de navigation, nous proposons de décomposer ce composant en deux parties en rajoutant un *offsetConveyor*. Cet *offsetConveyor* est attaché au *conveyor* avec un *offset* de position, de rotation et d'échelle. Le *stage* est directement « attaché » à l'*offsetConveyor* (cf. figure 5.4(a)). Cette décomposition va permettre, par exemple, à un utilisateur de naviguer dans une direction tout en regardant dans une autre en changeant l'offset entre *offsetConveyor* et le *conveyor* (cf. figure 5.4(b)). Lorsque le *conveyor* (et donc l'*offsetConveyor*) sont déplacés dans le monde virtuel en utilisant la technique de navigation définie par le *conveyor*, le *stage* et tous les espaces d'interaction inclus à l'intérieur sont déplacés avec lui (y compris lors des changements d'échelle).

En conséquence, le *conveyor* et l'*offsetConveyor* sont exclusivement virtuels, tandis que le *stage* est une représentation virtuelle liée à l'environnement réel des utilisateurs. Avec cette décomposition, il faut choisir où fixer la limite en définissant quelles parties du monde réel doivent être modélisées dans l'environnement virtuel. En effet, comme il n'est pas possible d'intégrer tout le monde réel dans le monde virtuel, il faut définir quel doit être le dernier espace réel à intégrer dans l'environnement virtuel : nous proposons de fixer cette limite au premier espace réel qui ne peut pas être bougé dans le monde réel lorsque l'utilisateur interagit. Par exemple, dans le cas d'une salle immersive, la limite du *stage* est la salle en entier. Par contre, dans le cas d'une plateforme mobile comme un simulateur de vol, il faudra intégrer l'espace entourant la plateforme afin de pouvoir répercuter ses mouvements dans l'environnement virtuel.

5.2.3 Structure de la CVII

Pour mettre en œuvre le modèle de CVII, nous utilisons une structure, indépendante du graphe de scène et des autres représentations de l'environnement virtuel, afin de lier entre eux les différents composants de la CVII, un peu comme dans Dive [Frécon et Stenius, 1998] (cf. partie 1.1.3.2). Cette structure est mise en œuvre grâce au modèle d'architecture logicielle PAC-C3D présentée dans le chapitre précédent. L'objet de base de l'environnement virtuel est le `SharedVirtualObject` : cet objet PAC-C3D prend en compte les aspects de distribution des données sur le réseau et de gestion des différentes représentations virtuelles grâce à la décomposition en trois facettes (*Présentation*, *Abstraction*, et *Contrôle*). Il est

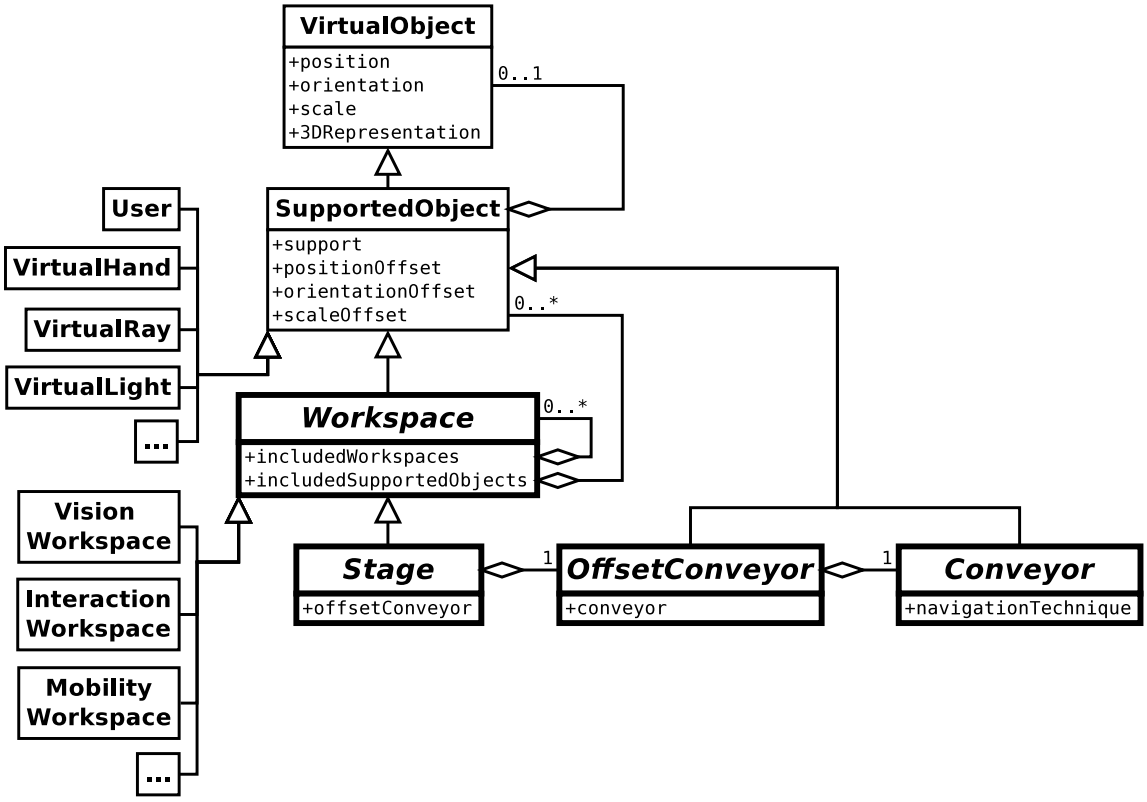


FIGURE 5.5 – Diagramme UML représentant les différents composants de la CVII.

intégré dans l'environnement virtuel par le gestionnaire d'objets comme présenté dans la partie 4.3.3. Les objets servant à modéliser la CVII héritent de cet objet de base afin de pouvoir bénéficier de l'architecture PAC-C3D et de l'intégration dans une structure qui leur permet de communiquer entre eux. Tous ces objets sont donc décomposés en trois facettes PAC-C3D. Cependant, dans cette partie, nous ne détaillons pas les trois facettes à chaque fois afin de ne pas compliquer les explications et le schéma. La figure 5.5 présente l'organisation des différents composants de la CVII :

- Le composant principal est le `SupportedObject` qui hérite du `SharedVirtualObject`. Chaque `SupportedObject` est rattaché à un autre `SharedVirtualObject` avec un *offset* pour la position, l'orientation et l'échelle. De cette façon, le `SupportedObject` suit l'évolution du comportement de son « support » notamment au niveau de son positionnement dans le monde virtuel. Afin de ne pas avoir à demander en permanence à son « support » s'il a été modifié, le `SupportedObject` s'abonne aux modifications de son « support » afin de recevoir les nouvelles valeurs de ses paramètres dès que ce dernier est modifié. Ainsi, le « support » propage les modifications qu'il subit aux objets dont il est le « support ». Le `SupportedObject` peut être spécialisé afin de modéliser les différents objets du monde virtuel (`VirtualHand`, `VirtualRay`, etc.).
- Le `Workspace` permet de modéliser un espace d'interaction de la CVII. Il hérite du `SupportedObject` auquel il ajoute la notion d'un volume 3D. Il peut donc contenir d'autres `Workspaces` ou n'importe quels autres `SupportedObjects`. Le `Workspace` peut également être spécialisé afin de proposer un comportement et des fonctionnalités adaptées à chacun des espaces d'interaction de l'utilisateur (`VisionWorkspace`, `InteractionWorkspace`, etc.).
- Le `Stage` est un `Workspace` particulier qui est lié obligatoirement à un `OffsetConveyor`. Nous avons expliqué qu'il inclut les différents `Workspaces` d'un utilisateur.
- L'`OffsetConveyor` est lui-même rattaché à un `Conveyor` avec éventuellement un offset au niveau de la position, de l'orientation, ou de l'échelle.
- Le `Conveyor` définit la technique de navigation utilisée.

5.2.4 Opérateurs de la CVII

Le modèle de CVII fournit un ensemble d'opérateurs pour gérer et manipuler la structure présentée dans la partie précédente. Ces opérateurs permettent de mettre en œuvre différentes techniques d'interaction et de collaboration adaptées en fonction des dispositifs matériels utilisés par chaque utilisateur. Dans cette partie, nous appellons « position » le regroupement de la position, de l'orientation et de l'échelle d'un objet.

Premièrement, nous distinguons un ensemble d'opérateurs de base (**Bo**) pour chacun des différents composants de la CVII :

- Pour le `SupportedObject`, des opérateurs permettent :

  Bo1 : de modifier ses paramètres propres (couleurs, géométrie, etc.),
  Bo2 : de modifier sa position dans le repère global,
  Bo3 : de modifier sa position dans son repère propre,
  Bo4 : de modifier ses *offsets* dans le repère de son support,
  Bo5 : de changer son « support »,
  Bo6 : de calculer sa position dans un autre repère.

- Pour le `Workspace`, des opérateurs permettent en plus :
  Bo7 : de modifier la forme de son volume 3D associé,
  Bo8 : d'ajouter ou enlever des `SupportedObjects` à l'intérieur de ce `Workspace`,
  Bo9 : d'ajouter ou enlever d'autres `Workspaces` à l'intérieur de ce `Workspace`,
  Bo10 : de calculer les distances entre un objet inclus et les frontières du volume 3D.
- Pour le `Stage`, un opérateur permet en plus :
  Bo11 : de changer son `OffsetConveyor` associé.
- Pour l'`OffsetConveyor`, un opérateur permet en plus :
  Bo12 : de changer son `Conveyor` associé.
- Pour le `Conveyor`, un opérateur permet en plus :
  Bo13 : de changer la technique de navigation utilisée.

Deuxièmement, afin d'offrir des fonctionnalités de collaboration plus complexes, nous avons également proposé un ensemble d'opérateurs de plus haut niveau (**Ho**) réutilisant les opérateurs de base. Ces opérateurs permettent :

Ho1 : d'attacher les `Stages` de plusieurs utilisateurs à un même `OffsetConveyor`,
Ho2 : d'attacher plusieurs `OffsetConveyors` à un même `Conveyor`,
Ho3 : d'attacher un `Conveyor` à un autre objet du monde virtuel (navigation relative),
Ho4 : de permettre à un `Conveyor` de prendre la même position qu'un autre `Conveyor`,
Ho5 : de restreindre les degrés de liberté lors du déplacement d'un objet,
Ho6 : de contraindre le déplacement d'un objet à respecter certaines conditions,
Ho7 : d'imposer que l'orientation d'un objet suive une direction particulière,
Ho8 : de calculer les intersections de plusieurs `Workspaces`.

Bien entendu, cette liste d'opérateurs de haut niveau n'est pas exhaustive et elle ne prétend pas couvrir toutes les opérations possibles dans un environnement virtuel collaboratif. Elle peut être complétée en fonction des besoins des utilisateurs et des différentes actions à réaliser dans de nouvelles applications.

5.3 Fonctionnalités de la CVII

Grâce à sa structure et à ses opérateurs, le modèle de CVII permet de mettre en œuvre de nombreuses fonctionnalités pour l'interaction et la collaboration dans un environnement virtuel collaboratif. Ces fonctionnalités peuvent être adaptées en fonction de l'environnement réel de chaque utilisateur et des dispositifs matériels qu'il utilise. En particulier, le modèle de CVII permet de décomposer la navigation dans le monde virtuel selon trois niveaux (partie 5.3.1), d'intégrer différents outils d'interaction dans l'environnement virtuel (partie 5.3.2), de faire percevoir aux utilisateurs leurs capacités de perception et d'interaction (partie 5.3.3) et de leur offrir des fonctionnalités pour la collaboration (partie 5.3.4).

5.3.1 Navigation

Afin d'adapter les différentes techniques de navigation à l'environnement réel de chaque utilisateur, nous distinguons trois niveaux de navigation pour un utilisateur utilisant une CVII. Ces trois niveaux ne sont pas utilisés dans tous les cas, mais ils permettent d'adapter la CVII en fonction de l'application et des dispositifs matériels utilisés.

1. Pour naviguer au delà de l'espace de déplacement physique, les utilisateurs peuvent déplacer leur *conveyor* en utilisant la technique de navigation qui lui est associée. Les différentes techniques de navigation de la littérature (cf. partie 1.1.2.1) peuvent être implémentées en changeant la position, l'orientation et l'échelle du *conveyor* grâce aux opérateurs proposés dans la partie précédente. Par exemple, il est possible d'implémenter les métaphores « classiques » de navigation comme celle du vol (**Bo3**), celle de la marche (**Bo3** et **Ho5**) ou celle de la téléportation (**Bo2**).
2. Afin de mettre en place des techniques de navigation plus complexes, les utilisateurs peuvent également modifier les *offsets* entre l'*offsetConveyor* et le *conveyor* (**Bo4**). Cela permet d'offrir plus de liberté aux utilisateurs en leur permettant, par exemple, de regarder librement dans la direction qu'ils souhaitent lors des déplacements automatiques (visite guidée, suivi de chemin, etc.), ou de suivre un autre utilisateur avec un *offset* qu'ils peuvent modifier lors d'une navigation collaborative. Cette décomposition en deux parties (*offsetConveyor* et *conveyor*), nous a aussi permis d'implémenter une technique permettant de se déplacer intuitivement par rapport à un objet virtuel, même si cet objet est en mouvement dans le monde virtuel (cf. figure 5.6). Lorsqu'un utilisateur sélectionne un objet virtuel, son *conveyor* est automatiquement attaché à cet objet avec un *offset* nul en translation (**Ho3**) et l'*offset* de son *offsetConveyor* est recalculé pour que l'utilisateur reste à la même place dans le monde virtuel (**Bo4**). En faisant tourner simplement son *conveyor* (**Bo3**), l'utilisateur peut tourner très facilement autour de l'objet qui devient ainsi le centre de rotation du *conveyor*. En modifiant l'*offset* de son *offsetConveyor* (**Bo4**), l'utilisateur peut également réaliser des *« travelings »* le long de l'objet. Enfin, si l'utilisateur déplace légèrement son *conveyor* (**Bo2** ou **Bo3**), il peut aussi déplacer le centre de rotation sur une partie particulière de l'objet.
3. Les utilisateurs peuvent se déplacer physiquement dans l'espace de déplacement physique inclus dans leur *stage*. En effet, le modèle de CVII permet d'intégrer, dans l'environnement virtuel, l'espace dans lequel chaque utilisateur peut se déplacer dans le monde réel. Les déplacements réels de cet utilisateur sont ainsi co-localisés avec ses

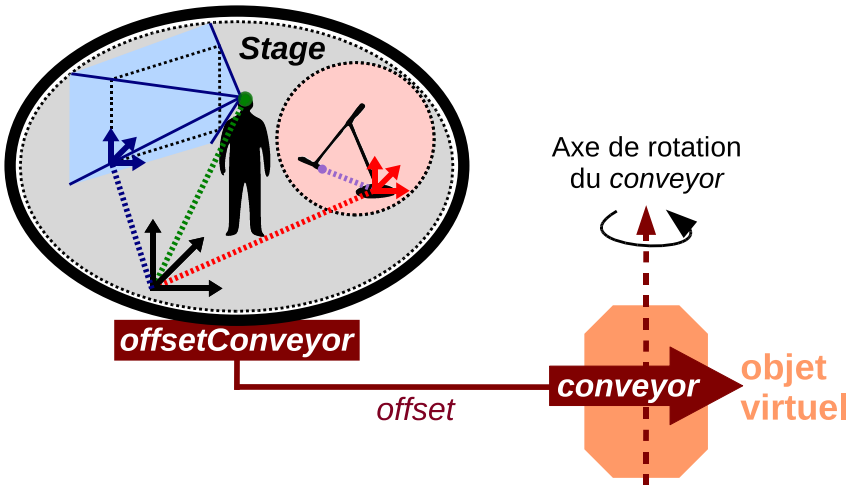


FIGURE 5.6 – Le *conveyor* est attaché à un objet virtuel afin que l'utilisateur puisse tourner intuitivement autour de cet objet.

déplacements dans le monde virtuel ce qui lui permet d'interagir et de se placer de manière intuitive dans l'environnement virtuel. Pour cela, la position de l'utilisateur est repérée par un système de *tracking* afin de mettre à jour son *offset* dans le repère de son espace de déplacement physique (**Bo4**). De cette façon, la co-localisation de l'utilisateur peut être maintenue quelles que soient la position, l'orientation, mais aussi l'échelle du stage dans le monde virtuel. De plus, la pyramide de vue de l'utilisateur (espace de visualisation) est déformée (**Bo7**) en fonction de la position de sa tête par rapport aux écrans de visualisation *(head-tracking)*. Enfin, nous pouvons inclure dans ce niveau de navigation les techniques où un utilisateur déplace un écran comme le HHD [Amselem, 1995] (cf.partie 1.1.3.3). Dans ce cas, l'écran est repéré par rapport au repère de l'espace de déplacement physique (**Bo4**) et l'espace de visualisation est également recalculé en fonction de cette position (**Bo7**).

5.3.2 Intégration des outils d'interaction

Le modèle de CVII intègre les outils d'interaction et les espaces d'interaction qui leur sont associés dans le *stage*. Ainsi, chaque utilisateur peut se déplacer dans le monde virtuel en transportant avec lui ses outils d'interaction car ils suivent les déplacements du *stage*. De même que pour l'utilisateur, la co-localisation entre des objets réels et leur représentation virtuelle peut être maintenue quelles que soient la position, l'orientation et l'échelle du *stage* dans le monde virtuel.

Certains outils d'interaction sont utilisés pour réaliser directement des interactions 3D avec le monde virtuel (main virtuelle, rayon virtuel, ect.), tandis que d'autres peuvent être considérés comme des *widgets* 3D qui apportent des informations supplémentaires ou permettent d'effectuer une action particulière. Par exemple, un monde en miniature (WIM) ou un slider 3D pour modifier l'échelle de l'*offsetConveyor* peuvent être considérés comme des *widgets* 3D. La structure hiérarchique de la CVII permet aux utilisateurs de placer ces *widgets* 3D relativement au *stage* (**Bo4**) afin d'organiser leur espace de travail d'une façon adaptée aux tâches qu'ils doivent réaliser. Les utilisateurs peuvent ainsi se déplacer en emmenant avec eux l'espace de travail qu'ils ont organisé : nous retrouvons la métaphore du poste de pilotage où les utilisateurs naviguent entourés par divers « instruments » (cf. partie 1.1.2.1). Si les utilisateurs peuvent déplacer les écrans de visualisation, il est possible d'aller plus loin en leur permettant de choisir et de modifier l'emplacement des « fenêtres » de leur cabine virtuelle.

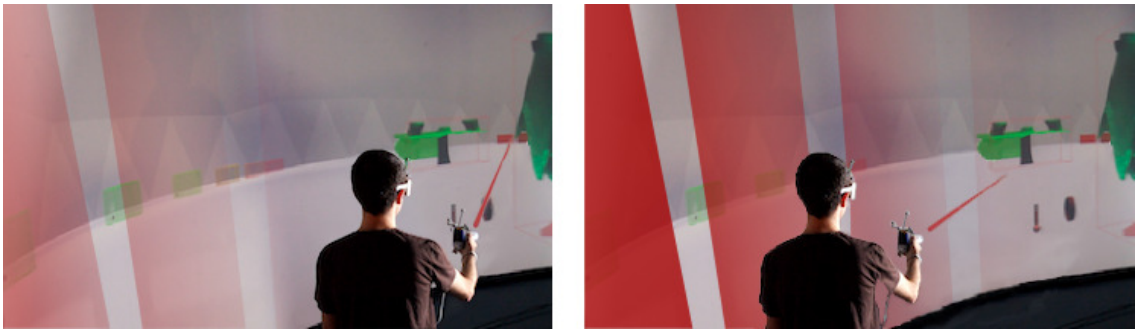
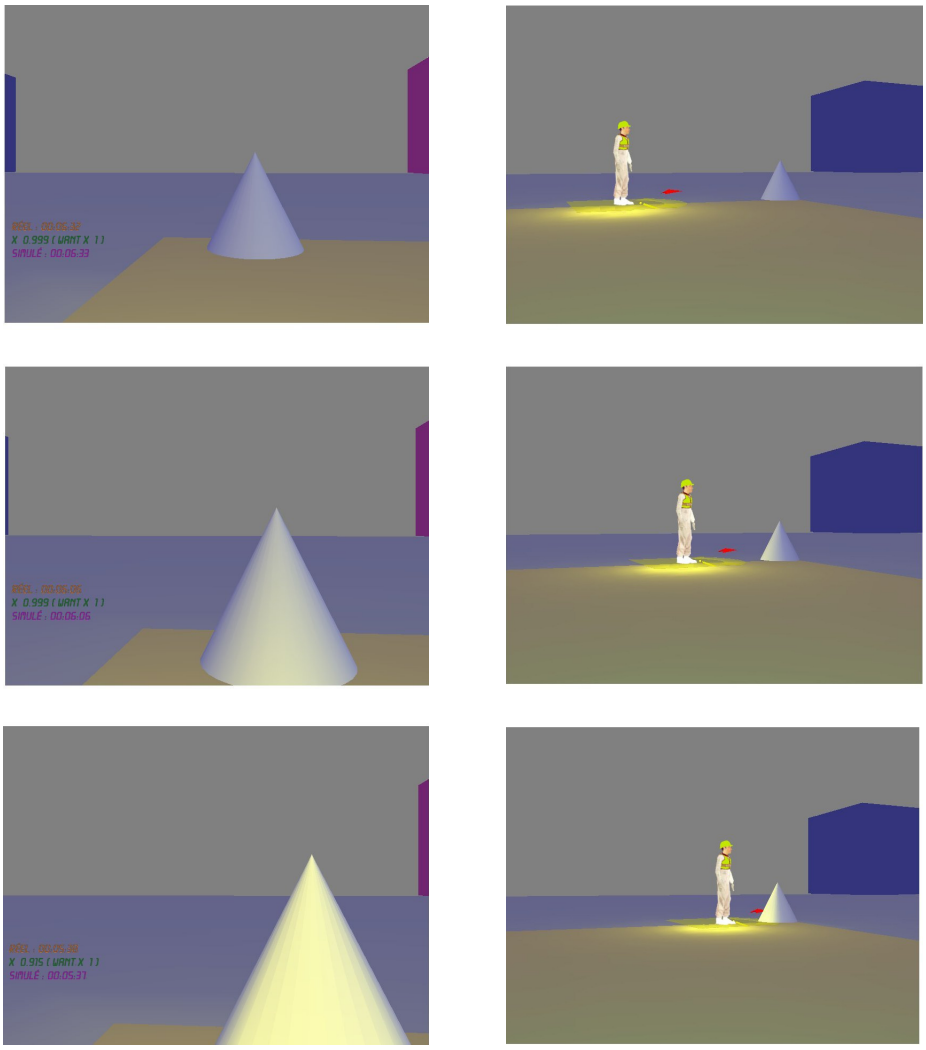


FIGURE 5.7 – Les vitres semi-transparentes matérialisant les écrans de visualisation s'opacifient lorsque l'utilisateur s'en approche.

5.3.3 Perception des limites des espaces d'interaction

Les différents dispositifs matériels utilisés imposent des limitations spatiales lors des interactions des utilisateurs dans l'environnement virtuel à cause de leurs caractéristiques techniques (portée des capteurs de *tracking*, espace de déplacement physique dans un dispositif immersif, espace d'interaction d'un dispositif haptique, etc.). En intégrant les espaces d'interaction associés à chacun de ces dispositifs matériels, le modèle de CVII offre une structure pour représenter les limites de ces espaces dans le monde virtuel afin de faire percevoir aux utilisateurs leurs capacités de perception et d'interaction.

Premièrement, il est intéressant de permettre à chaque utilisateur de comprendre quelles sont ses capacités de perception et d'interaction afin qu'il puisse adapter sa façon d'interagir en conséquence. Par exemple, nous avons proposé d'utiliser des vitres semi-transparentes



(a) (b)

FIGURE 5.8 – L'objet virtuel est illuminé par un reflet jaune lorsqu'il entre dans l'espace d'interaction de la main virtuelle de l'utilisateur associé à la couleur jaune : (a) vu par l'utilisateur lui-même et (b) vu par un autre utilisateur.

pour représenter l'espace de déplacement physique d'un utilisateur. Ces vitres semi-transparentes sont placées virtuellement au niveau des écrans de visualisation d'un dispositif immersif afin d'éviter qu'un utilisateur s'approche trop près de ces écrans. Afin de ne pas surcharger la scène, les vitres sont totalement transparentes lorsque l'utilisateur est loin des écrans. Cependant, lorsque l'utilisateur s'approche des écrans (**Bo10**), les vitres sont opacifiées progressivement (**Bo1**) afin que l'utilisateur puisse adapter sa façon d'interagir pour ne pas toucher les écrans (cf. figure 5.7).

Deuxièmement, dans le cas collaboratif, il est aussi intéressant de permettre à chaque utilisateur de comprendre quelles sont les capacités de perception et d'interaction des autres utilisateurs. Cela permet aux utilisateurs d'adapter leur façon de collaborer et d'améliorer la répartition des tâches en fonction des capacités de chacun. Par exemple, nous avons proposé d'illuminer avec une couleur propre à chaque utilisateur les objets qui se trouvent dans l'espace d'interaction de la main virtuelle qu'ils utilisent (cf. figure 5.8). Ainsi, lorsqu'un objet entre dans l'espace d'interaction de la main virtuelle d'un utilisateur, il est illuminé avec des reflets de la couleur associée à cet utilisateur. Cette représentation de l'espace d'interaction de la main virtuelle a un double rôle. D'une part, elle permet à chaque utilisateur de savoir quels sont les objets qu'il peut atteindre directement et quels sont les objets nécessitant qu'il effectue une tâche de navigation pour les atteindre. D'autre part, elle permet à chaque utilisateur de voir quels objets peuvent être atteints par les autres utilisateurs. Un utilisateur est alors en mesure de demander à un autre utilisateur de manipuler pour lui des objets qui sont dans l'espace d'interaction de ce deuxième utilisateur.

5.3.4 Collaboration au travers de plusieurs CVII

Les environnements virtuels collaboratifs offrent généralement la possibilité aux utilisateurs de se déplacer indépendamment afin d'avoir des perspectives hétérogènes sur le monde virtuel. Afin de permettre à chaque utilisateur de naviguer indépendamment des autres, le modèle de CVII propose d'instancier un *offsetConveyor* et un *conveyor* par utilisateur. Chaque utilisateur peut ainsi déplacer indépendamment son *stage* et les espaces d'interaction inclus dedans en utilisant la technique de navigation de son *conveyor*. Cependant, comme présenté dans la partie 1.2.3, il existe certaines applications où il est intéressant que les utilisateurs puissent naviguer ensemble dans le monde virtuel. La structure, ainsi que les opérateurs de la CVII permettent de mettre en place facilement ces fonctionnalités de navigation collaborative :

1. Si un utilisateur veut rejoindre un autre utilisateur dans le monde virtuel, il est possible de superposer leur deux *conveyors* (**Ho4**).
2. Si un utilisateur veut prendre la même vue qu'un autre utilisateur et conserver cette même vue lorsque l'autre se déplace, il est possible d'attacher le *stage* de chaque utilisateur à l'*offsetConveyor* du deuxième utilisateur (**Ho1**)(cf. figure 5.9). Si ces deux utilisateurs n'utilisent pas le même dispositif de visualisation, il se peut alors que leur point de vue diffère légèrement (champ de vision, positions relatives des écrans par rapport à l'utilisateur, etc.). Dans ce cas, il peut alors être important de représenter leurs espaces de visualisation respectifs afin qu'ils puissent comprendre ce que l'autre voit.
3. Si un utilisateur veut suivre un autre utilisateur avec un décalage (pour pouvoir observer ce qu'il fait, par exemple), il est possible d'attacher les *offsetConveyors* des deux utilisateurs au *conveyor* du deuxième utilisateur (**Ho3**)(cf. figure 5.10). Ainsi

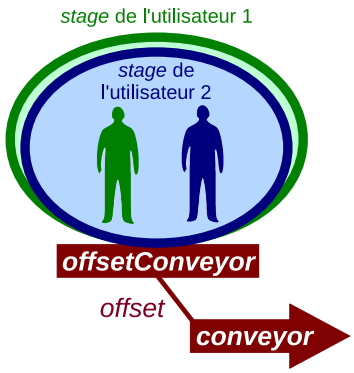


FIGURE 5.9 – Les *stages* de 2 utilisateurs sont attachés au même *offsetConveyor*.

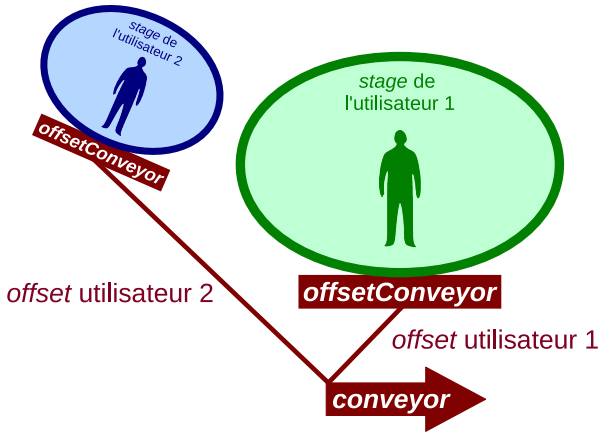


FIGURE 5.10 – Les *offsetConveyors* de 2 utilisateurs sont attachés au même *conveyor* avec des *offsets* différents.

l'utilisateur est libre de choisir un *offset* différent pour son *offsetConveyor* afin d'être en retrait par rapport à l'autre utilisateur.

4. Si un utilisateur veut déplacer un autre utilisateur pour l'emmener voir un détail intéressant du monde virtuel, il peut interagir directement avec son *conveyor* comme il le ferait avec n'importe quel autre objet virtuel pour le déplacer ou le tourner dans le monde virtuel (**Bo2**).

Par ailleurs, il est possible de faire intersecter les espaces d'interaction de plusieurs utilisateurs (**Ho8**) : cela permet de mettre en relation plusieurs espaces réels distants au travers de l'environnement virtuel. Cette mise en relation impose quelques restrictions car les objets réels peuvent apparaître seulement dans un espace réel à la fois. Cependant, si chaque objet réel a une représentation virtuelle, cette représentation peut apparaître dans les espaces d'interaction de chacun des utilisateurs ce qui leur permet de percevoir sa présence. Par exemple, c'est le cas de l'avatar d'un utilisateur associé à son corps dans l'espace de déplacement physique. Si deux espaces de déplacement physique s'intersectent, un utilisateur pourra se tenir debout dans l'espace de déplacement physique de l'autre utilisateur au travers de son avatar.

5.4 Exemples d'utilisation de la CVII

Dans cette partie, nous présentons plusieurs exemples d'utilisation de notre modèle qui illustrent le concept de *Cabine Virtuelle d'Interaction Immersive*. Dans la partie 5.4.1, nous présentons d'abord deux instanciations possibles de la CVII. Dans la partie 5.4.2, nous détaillons une nouvelle métaphore de curseur 2D / rayon 3D permettant d'offrir les mêmes capacités d'interaction que celles d'un rayon virtuel 3D à un utilisateur qui n'utilise pas un système immersif. Dans la partie 5.4.3, nous expliquons comment les différentes fonctionnalités de navigation collaborative de la CVII ont été utilisées dans le cadre d'une expérimentation pour le *3DUI contest 2012*. Enfin, dans la partie 5.4.4, nous proposons un nouveau système de gestion des points de vue pour faciliter l'exploration collaborative de données scientifiques.

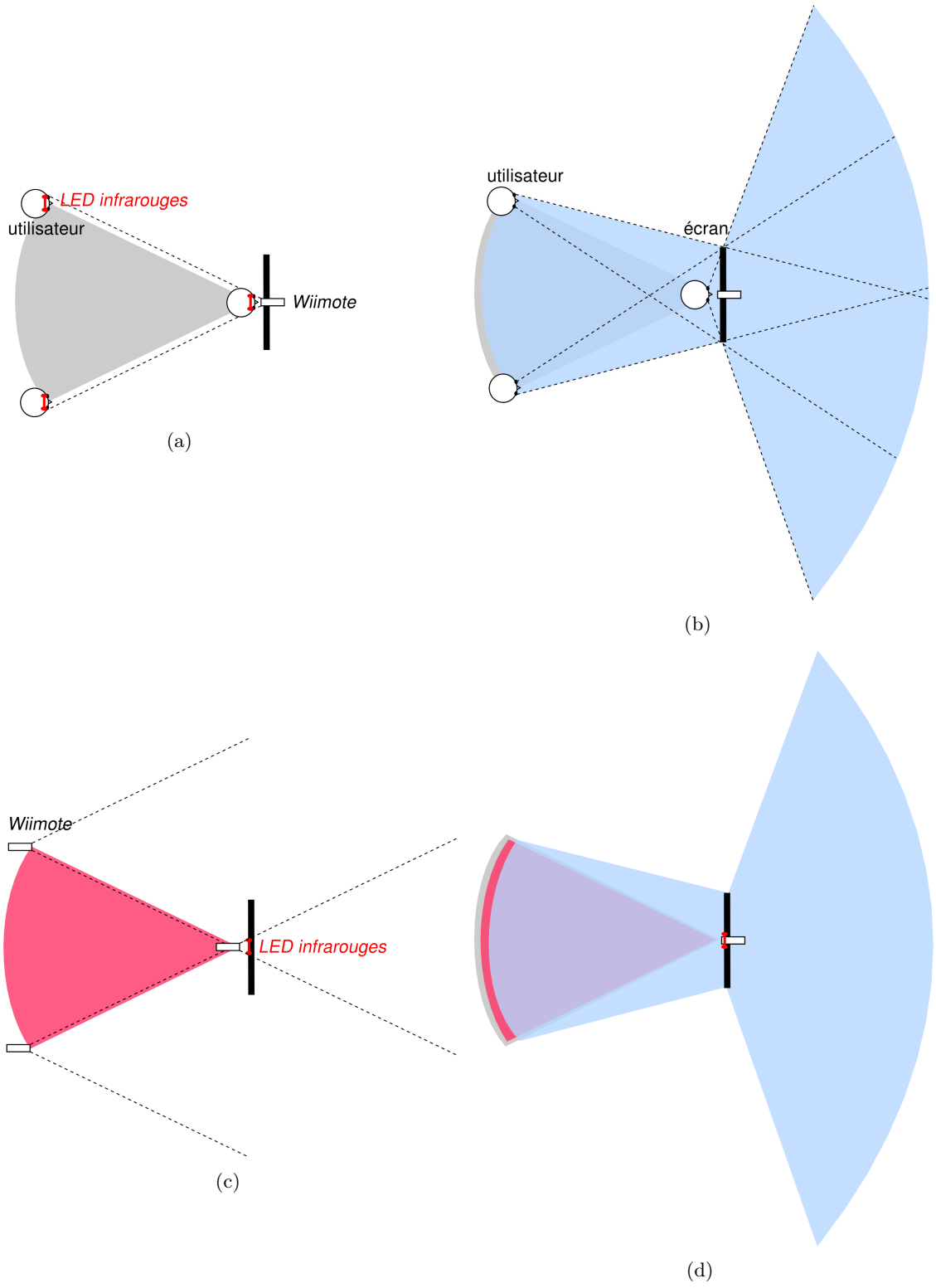


FIGURE 5.11 – Instanciation de la CVII pour une station de travail semi-immersive avec (a) l'espace de déplacement physique, (b) l'espace de visualisation, (c) l'espace d'interaction associé à une *Wiimote* et (d) le *stage* regroupant ces trois espaces.

5.4.1 Deux exemples d'instanciation de la CVII

Le modèle de CVII permet de modéliser l'environnement réel de chaque utilisateur et de l'intégrer dans le monde virtuel. Nous détaillons ici deux exemples d'instanciation de la CVII que nous avons utilisés dans le cadre de nos expérimentations avec deux types de dispositifs matériels différents : une station de travail (partie 5.4.1.1), puis une salle immersive (partie 5.4.1.2).

5.4.1.1 Instanciation pour une station de travail semi-immersive

Pour permettre à des utilisateurs de collaborer dans un environnement virtuel sans avoir besoin d'un système de réalité virtuelle complexe et coûteux, nous avons proposé d'utiliser des *Wiimotes* de la console *Wii* de Nintendo® pour améliorer la sensation de présence pour un utilisateur utilisant une simple station de travail ou un ordinateur portable. Une *Wiimote* est utilisée pour repérer la tête de l'utilisateur et pour déformer l'image affichée (*head-tracking*) afin qu'il ait une sensation de profondeur malgré l'écran monoscopique de sa station de travail ou de son ordinateur portable. Une autre *Wiimote* est utilisée pour lui permettre d'interagir de manière intuitive grâce aux mouvements de sa main.

Pour repérer la tête de l'utilisateur assis devant son bureau, nous avons utilisé la caméra infrarouge de la *Wiimote*. L'utilisateur porte des lunettes sur lesquelles sont fixées deux LED infrarouges et la *Wiimote* est fixée au dessus ou au dessous de l'écran. Les deux LED sont captées par la caméra infrarouge de la *Wiimote* ce qui permet de déterminer la position de la tête dans l'image plan de la caméra, ainsi que la profondeur de la tête en fonction de l'écartement des diodes. Le champ de vision de la caméra infrarouge (33 ° à l'horizontal et 23 ° à la verticale) définit l'espace de déplacement physique dans lequel l'utilisateur doit rester pour être repéré par la *Wiimote* (cf. figure 5.11(a)). À partir de cet espace de déplacement physique, nous pouvons définir l'espace de visualisation en fonction de toutes les positions possibles que peut prendre la tête de l'utilisateur et la position de l'écran (cf. figure 5.11(b)). Enfin, pour interagir, l'utilisateur tient dans sa main la deuxième *Wiimote* et une barre avec deux LED infrarouges est placée au dessus ou au dessous de l'écran. D'une façon similaire à sa tête, la main de l'utilisateur est repérée par rapport à l'écran en fonction de la position et l'écartement des deux LED perçues par la caméra infrarouge de la *Wiimote*. L'utilisateur peut ainsi interagir avec le monde virtuel au moyen d'un curseur 3D par exemple. Le champ de vision de la camera infrarouge de cette deuxième *Wiimote* définit l'espace d'interaction de l'utilisateur (cf. figure 5.11(c)). Ces trois espaces sont regroupés dans le *stage* de l'utilisateur afin de lui permettre naviguer dans l'environnement virtuel en conservant la co-localisation de sa tête et de sa main quelles que soient sa position et son échelle dans le monde virtuel (cf. figure 5.11(d)). Ce *stage* permet également de représenter l'utilisateur derrière son bureau dans le monde virtuel (cf. figure 5.13).

5.4.1.2 Instanciation pour une salle immersive

Afin d'intégrer un utilisateur interagissant dans une salle immersive, nous avons modélisé ce type de salle grâce à la CVII. Notre salle de réalité virtuelle Immersia est actuellement constituée d'un grand écran sur un mur (9,6 × 3 m) et d'un grand écran sur le sol (9,6 × 2,8 m). De plus, un système de *tracking* permet de repérer la position de plusieurs marqueurs à l'aide de caméras infrarouges dans le parallélépipède rectangle défini par les deux écrans. Pour la plupart de nos applications, nous repérons, grâce à deux marqueurs, la position de

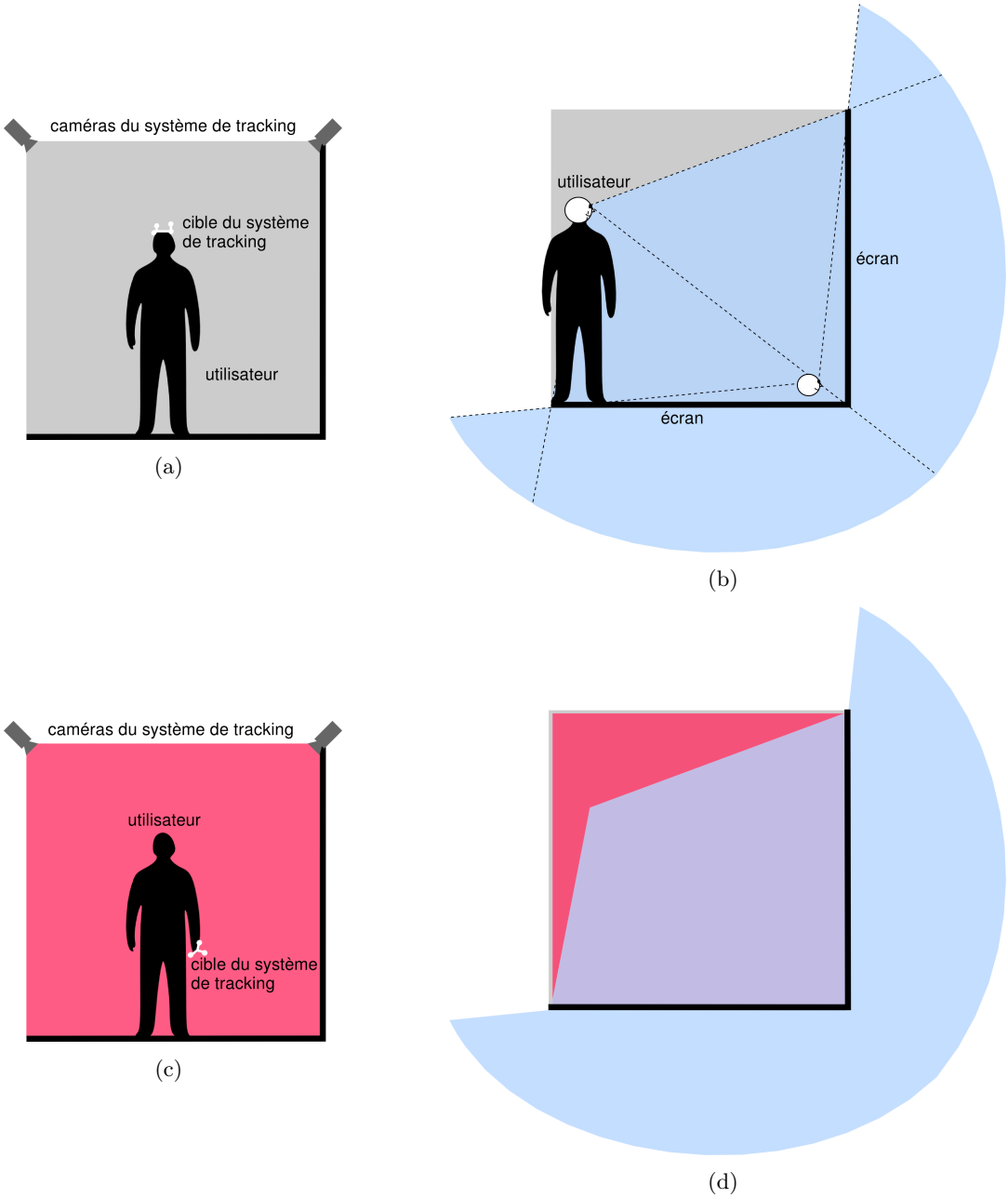


FIGURE 5.12 – Instanciation de la CVII pour une salle immersive avec (a) l'espace de déplacement physique, (b) l'espace de visualisation, (c) l'espace d'interaction et (d) le *stage* regroupant ces trois espaces.

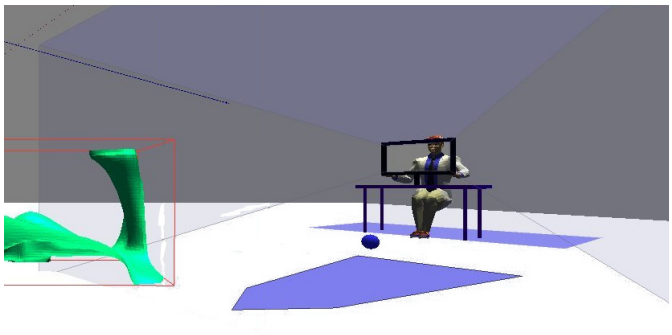


FIGURE 5.13 – Représentation virtuelle de la station de travail semi-immersive vue par un utilisateur extérieur.

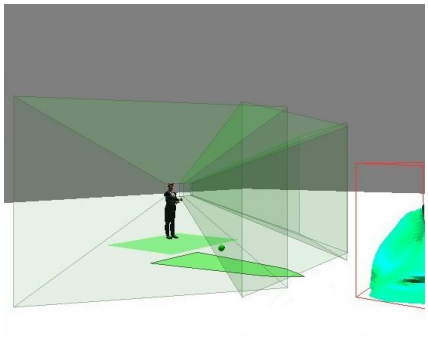


FIGURE 5.14 – Représentation virtuelle de la salle immersive vue par un utilisateur extérieur.

la tête de l'utilisateur et celle de sa main pour lui permettre d'interagir avec les objets du monde virtuel (avec la métaphore de la main virtuelle, par exemple). Le parallélépipède rectangle couvert par les caméras infrarouges du système de *tracking* correspond donc à la fois à l'espace de déplacement physique de l'utilisateur et à l'espace d'interaction (cf. figure 5.12(a et c)). À partir des différentes positions possibles de la tête de l'utilisateur, l'espace de visualisation est défini par l'union des pyramides de vues qui passent par les deux écrans (cf. figure 5.12(b)). Le *stage* de l'utilisateur englobe ces trois espaces et lui permet de naviguer sur des grands distances dans le monde virtuel en emmenant avec lui l'espace réel de la salle immersive (cf. figure 5.12(d)). Ce *stage* permet également de représenter l'utilisateur au milieu de la salle immersive dans l'environnement virtuel (cf.figure 5.14).

5.4.2 Métaphore du curseur 2D/rayon 3D

Afin d'offrir des capacités d'interaction similaires à un utilisateur qui utilise une station de travail et à un utilisateur qui utilise une salle immersive, nous avons proposé une nouvelle métaphore qui adapte la métaphore du rayon virtuel 3D aux dispositifs non immersifs [Duval et Fleury, 2009]. D'une part, un utilisateur se servant d'une simple station de travail peut manipuler cette métaphore avec un périphérique d'interaction 2D aussi facilement

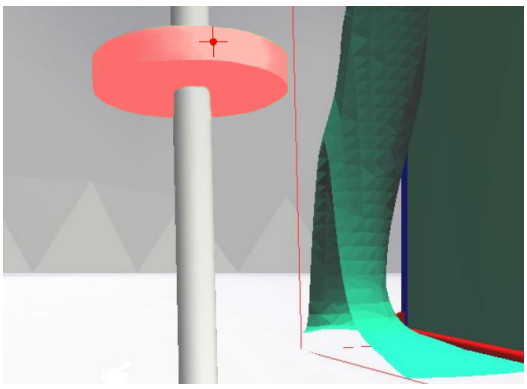


FIGURE 5.15 – Curseur 2D/rayon 3D vu par l'utilisateur qui le manipule.

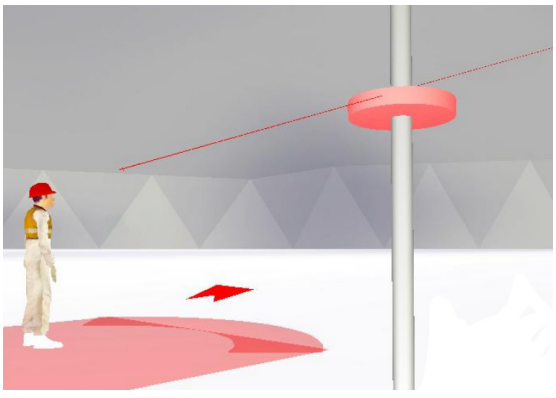


FIGURE 5.16 – Curseur 2D/rayon 3D vu par un autre utilisateur.

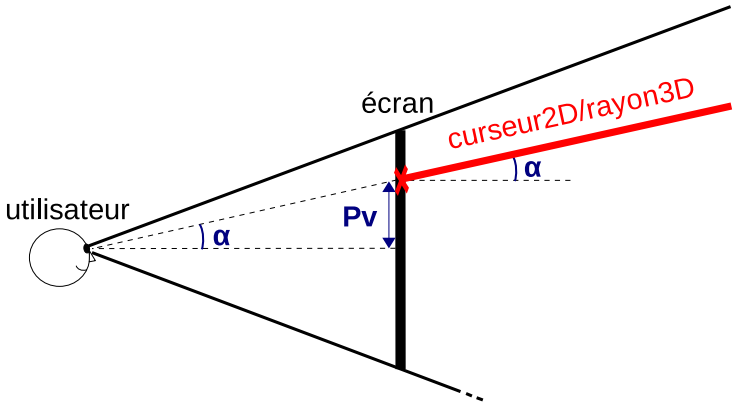


FIGURE 5.17 – L'orientation α du curseur 2D/rayon 3D est déterminée en fonction de sa position **Pv** par rapport au centre de l'écran afin qu'il se projette en un point sur l'écran pour l'utilisateur qui le manipule.

qu'il manipulerait un curseur 2D (cf. figure 5.15). D'autre part, les autres utilisateurs perçoivent cette métaphore comme un rayon virtuel 3D dans le monde virtuel afin qu'ils puissent comprendre quel objet virtuel l'utilisateur est en train de manipuler ou quel détail il est en train de montrer (cf. figure 5.16).

Cette nouvelle métaphore de curseur 2D/rayon 3D permet à un utilisateur de piloter un rayon virtuel 3D comme un curseur 2D. Pour cela, un rayon 3D est ajouté dans le monde virtuel, mais il est situé près du point de vue de l'utilisateur et toujours orienté afin que l'utilisateur le voie uniquement comme un point sur l'écran (cf.figure 5.17). Ce rayon 3D est donc intégré dans l'espace de visualisation de l'utilisateur afin que sa base soit placée au niveau de l'écran et qu'il soit orienté en fonction de la pyramide de vue de l'utilisateur. L'ouverture de la pyramide de vue (et en conséquence l'orientation du rayon 3D) dépendent soit de la valeur par défaut du champ de vision de l'utilisateur dans le monde virtuel, soit de la position de sa tête si elle est repérée par un système de *tracking*.

De cette façon, le curseur 2D/rayon 3D peut être piloté par n'importe quel périphérique 2D comme une souris, une *Wiimote* en utilisant uniquement les positions 2D captées par la caméra infrarouge, etc. Par ailleurs, les mouvements des objets manipulés avec le curseur 2D/rayon 3D peuvent être étendus à des mouvements 3D en permettant de translater ces objets le long du rayon. Pour contrôler une telle translation, il faut utiliser des commandes supplémentaires à celles du périphérique 2D comme, par exemple, la molette de la souris ou la profondeur capturée par la *Wiimote*.

5.4.3 Utilisation des fonctionnalités de navigation collaborative

Dans le cadre du *3DUI Contest 2012*, nous avons mis en place une expérimentation où deux utilisateurs doivent collaborer pour réaliser une tâche de navigation [Nguyen *et al.*, 2012]. La tâche consiste à trouver des cibles cachées dans un bâtiment qui comporte plusieurs étages. Les deux utilisateurs ont des rôles asymétriques : un premier utilisateur se déplace dans le bâtiment et doit trouver les cibles, tandis que le second peut prendre une vue différente afin de l'aider dans sa tâche en lui indiquant le chemin à suivre. Les règles du concours imposent que les utilisateurs ne puissent pas communiquer verbalement, mais uniquement en partageant des objets virtuels 3D.

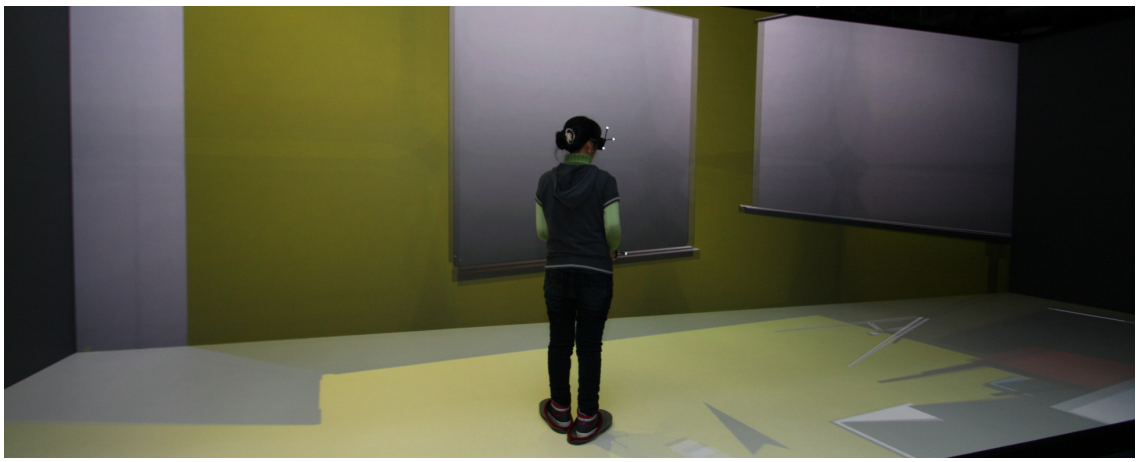


FIGURE 5.18 – Premier utilisateur explorant le bâtiment à la recherche des cibles dans l'environnement virtuel grâce à une salle immersive.

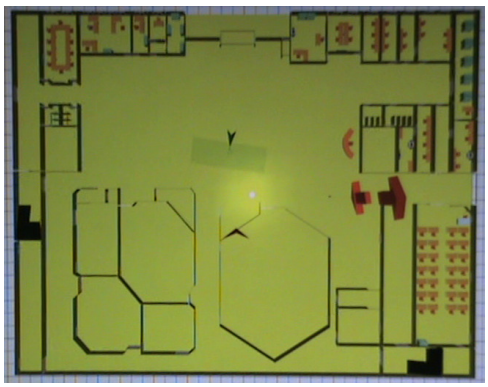


FIGURE 5.19 – Vue 1 du 2nd utilisateur offrant une vue globale du bâtiment.

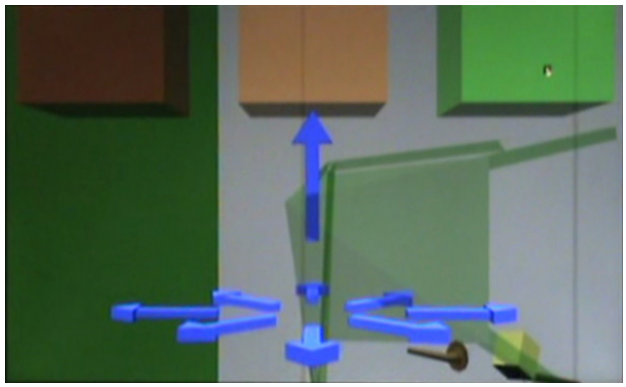


FIGURE 5.20 – Vue 2 du second utilisateur permettant de voir et de guider le premier utilisateur.

Le premier utilisateur utilise une salle immersive afin d'être immergé dans l'environnement virtuel (cf. figure 5.18). Il a un point de vue à la première personne et il peut naviguer dans le bâtiment en utilisant la métaphore de la marche qu'il contrôle avec un *joystick*. Il est également repéré par un système de *tracking*, donc il peut aussi se déplacer physiquement dans le dispositif en marchant afin d'explorer plus précisément le bâtiment (par exemple, il peut se pencher pour regarder sous un table ou derrière une porte). C'est obligatoirement cet utilisateur qui doit « attraper » les cibles. Cependant, il n'a aucune idée d'où elles se trouvent dans le bâtiment, et c'est pourquoi il a besoin d'être aidé.

Le second utilisateur utilise une simple station de travail avec deux écrans qui offrent deux vues différentes sur l'environnement virtuel. La vue 1 propose un point de vue en hauteur qui permet à l'utilisateur de voir le bâtiment en entier avec l'avatar de l'autre utilisateur à l'intérieur (cf. figure 5.19) comme dans un monde en miniature (WIM) ou dans la *deity's view* de CAVLIN [Leigh *et al.*, 1996]. Dans cette vue, il possède également un plan de *clipping* qui lui permet de « scanner » le bâtiment étage par étage afin de trouver les cibles. La vue 2 propose un point de vue légèrement en retrait derrière l'autre utilisateur (fonctionnalité n° 3 de la partie 5.3.4) afin de voir ce qu'il fait dans l'environnement virtuel et de lui fournir les indications nécessaires pour le guider (cf. figure 5.20).

Nous avons proposé plusieurs solutions pour permettre au second utilisateur de guider le premier. Premièrement, le second utilisateur peut afficher des flèches devant le premier utilisateur. Ces flèches sont crées dynamiquement à l'intérieur du *stage* du premier utilisateur afin qu'il puisse voir ces indications devant lui même lorsqu'il navigue. D'un façon similaire, le second utilisateur peut orienter une boussole qui indique la direction à suivre. Le premier utilisateur embarque cette boussole à l'intérieur de son *stage* et peut la placer là où c'est le plus pratique pour lui. Deuxièmement, le second utilisateur peut éclairer le chemin que le premier utilisateur doit suivre grâce à une lampe qu'il manipule dans la vue globale du bâtiment (vue 1). Troisièmement, le second utilisateur peut déplacer le premier utilisateur en manipulant directement son *conveyor* dans la vue 1 (fonctionnalité n° 4 de la partie 5.3.4).

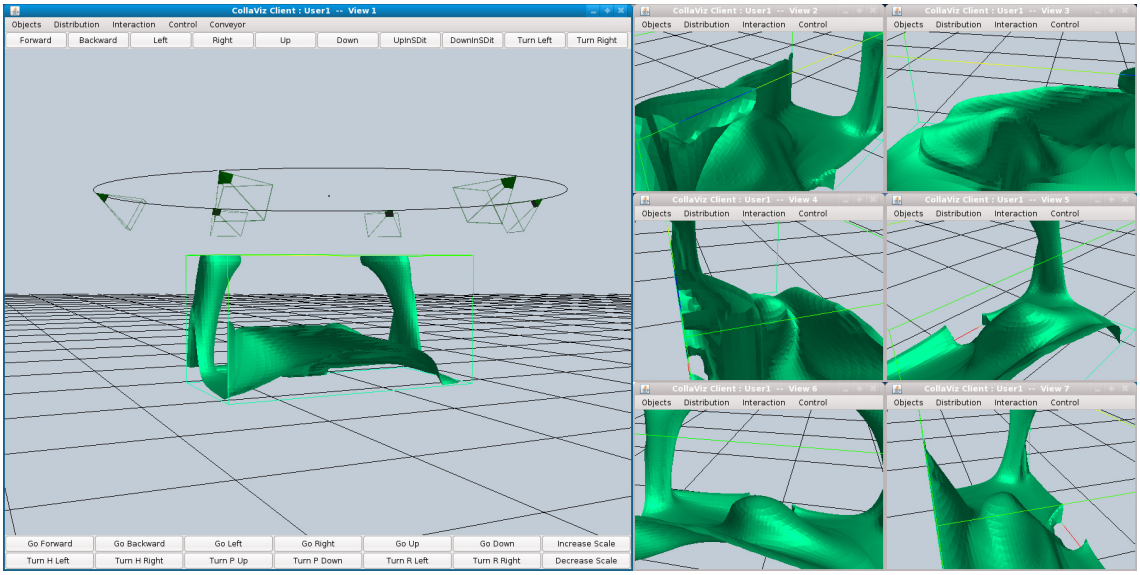


FIGURE 5.21 – Plusieurs caméras virtuelles permettant d'observer des données scientifiques sous différents angles.

5.4.4 Système de caméras pour la visualisation de données scientifiques

Pour faciliter l'exploration des données scientifiques, nous avons proposé un système de positionnement de plusieurs caméras virtuelles relativement aux données. Ce système peut être utilisé soit pour permettre à un utilisateur d'avoir plusieurs points de vue simultanés sur les mêmes données scientifiques, soit pour permettre à plusieurs utilisateurs d'examiner ensemble les mêmes données scientifiques avec des points de vue différents (cf. figure 5.21). Dans les deux cas, le système coordonne le positionnement de ces caméras virtuelles les unes par rapport aux autres afin de permettre aux utilisateurs d'observer la donnée sous plusieurs angles différents en même temps pour repérer des détails similaires ou des phénomènes discrets dans le temps.

Ce système se base sur la décomposition en un *conveyor* et un *offsetConveyor* proposée dans le modèle de CVII. Premièrement, le *conveyor* est rattaché à la donnée à examiner (**Ho3**) avec un *offset* nul afin que la donnée devienne son centre de rotation comme expliqué

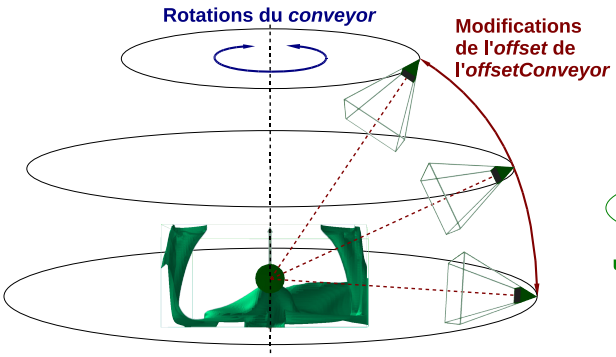


FIGURE 5.22 – L'*offsetConveyor* reste toujours à la même distance de son *conveyor* et les rotations du *conveyor* permettent de tourner autour des données.

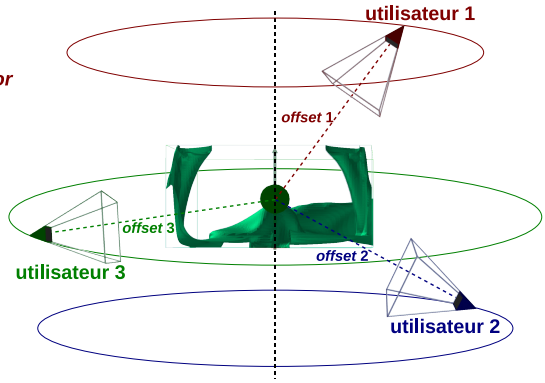


FIGURE 5.23 – Plusieurs *offsetConveyors* peuvent être rattachés au même *conveyor* afin de coordonner leurs rotations autour de la donnée.

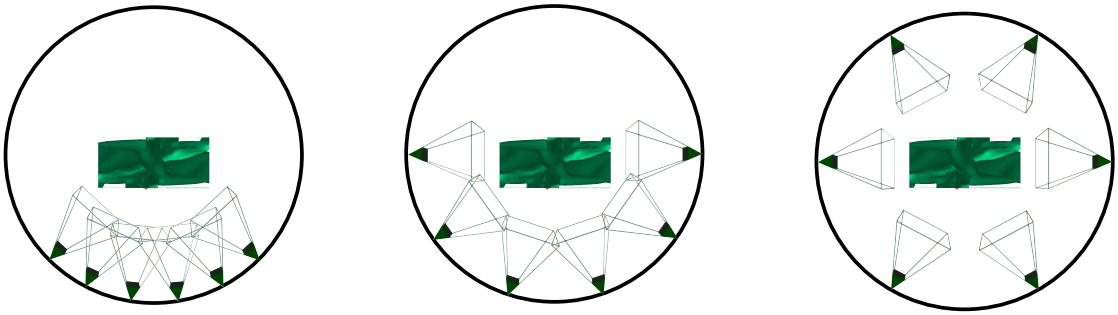


FIGURE 5.24 – Plusieurs distributions d'*offsetConveyors* le long de la même « orbite ».

dans la partie 5.3.1. Deuxièmement, nous proposons d'utiliser un *offsetConveyor* spécial :

- Cet *offsetConveyor* peut être déplacé par l'utilisateur uniquement verticalement en fonction de l'orientation du *conveyor* (**Ho5**).
- Il est contraint à rester toujours à la même distance du *conveyor* : cette distance définit donc une « orbite » de rotation autour de l'objet. En conséquence, lorsque l'utilisateur monte ou descend l'*offsetConveyor* par rapport à la position médiane du *conveyor*, le diamètre de cette « orbite » de rotation diminue (cf. figure 5.22).
- L'orientation de cet *offsetConveyor* est contrainte afin qu'il soit toujours orienté dans la direction de son *conveyor*(**Ho7**). Ainsi le point de vue associé à cet *offsetConveyor* est toujours tourné vers la donnée à examiner.

Pour permettre à un utilisateur d'avoir plusieurs points de vues sur la donnée ou pour permettre à plusieurs utilisateurs d'examiner en même temps la donnée, plusieurs *offsetConveyors* sont rattachés au même *conveyor*. Chacun de ces *offsetConveyors* intègre un *stage* avec au moins un espace de visualisation associé afin de restituer un point de vue sur la donnée à un utilisateur. Les utilisateurs peuvent faire évoluer librement leur(s) *offsetConveyor(s)* en respectant la contrainte de distance par rapport au *conveyor* afin de choisir plusieurs angles d'observation de la donnée (cf. figure 5.23). Cependant, une fois qu'une organisation satisfaisante des points de vue a été trouvée, l'utilisateur qui est le propriétaire du *conveyor* peut faire tourner tous ces points de vue en même temps en faisant tourner le *conveyor*. De plus, pour améliorer la coordination des points de vue, nous avons

proposé d'ajouter des fonctionnalités au niveau du *conveyor* afin de contrôler les différents *offsetConveyors* qui lui sont associés :

- Tous les *offsetConveyors* peuvent être regroupés au même endroit afin que tous les utilisateurs aient la même vue en même temps. Cela peut être intéressant lorsque l'utilisateur a trouvé quelque chose d'intéressant et qu'il veut le montrer aux autres.
- Tous les *offsetConveyors* peuvent être contraints à rester à la même hauteur les uns par rapport aux autres. Cela permet de les regrouper tous sur la même « orbite ».
- Tous les *offsetConveyors* peuvent être distribués homogènement le long de la même « orbite » sur tout le cercle ou sur seulement une portion du cercle (cf. figure 5.24).

Comme pour la rotation du *conveyor*, l'utilisateur qui est le propriétaire du *conveyor* contrôle également l'utilisation de ces fonctionnalités pour distribuer les *offsetConveyors*.

5.5 Conclusion

Intégrer des utilisateurs dans la boucle perception/action d'un environnement virtuel ne consiste pas seulement à brancher des informations en entrée et en sortie sur les dispositifs matériels à leur disposition. En effet, de plus en plus d'applications nécessitent de prendre en compte l'environnement réel qui entoure les utilisateurs, lors de la conception et de l'utilisation d'un système de réalité virtuelle. La *Cabine Virtuelle d'Interaction Immersive* (CVII) propose une solution générique pour modéliser et intégrer l'environnement réel de chaque utilisateur dans un environnement virtuel collaboratif. La CVII permet de mettre en relation le monde réel et le monde virtuel, mais aussi les utilisateurs finaux et les développeurs d'une application de réalité virtuelle. D'une part, elle intègre dans l'environnement virtuel les volumes 3D réels qui entourent les utilisateurs pour leur permettre d'interagir d'une façon adaptée aux capacités de leurs dispositifs matériels. D'autre part, elle permet aux développeurs de s'abstraire de la façon dont est organisée l'environnement réel des utilisateurs lorsqu'ils conçoivent une application de réalité virtuelle.

La CVII propose de modéliser l'environnement réel de chaque utilisateur comme une hiérarchie d'espaces d'interaction multi-sensoriels associés à leurs dispositifs matériels. Ainsi, l'environnement réel de chaque utilisateur est décomposé en plusieurs espaces d'interaction comme un espace de visualisation, un espace de restitution sonore, un espace de déplacement physique, un espace haptique, etc. La CVII propose également une solution générique pour intégrer ces différents espaces d'interaction réels dans l'environnement virtuel. Afin de permettre aux développeurs de s'abstraire de l'environnement réel des utilisateurs, la CVII est constituée de deux parties : le *stage* et le *conveyor*. Le *stage* est un volume 3D qui modélise les espaces d'interaction d'un utilisateur en fonction des dispositifs matériels qu'il utilise. Le *conveyor* est le point d'intégration du *stage* dans le monde virtuel et il définit la technique de navigation utilisée pour le déplacer.

Le modèle est basé sur une structure hiérarchique pour organiser les différents composants de la CVII, et sur des opérateurs pour gérer et manipuler cette structure. Ces opérateurs sont utilisés pour mettre en œuvre différentes fonctionnalités qui permettent aux utilisateur d'interagir et de collaborer d'une façon adaptée aux dispositifs matériels qu'ils utilisent. En particulier, la CVII :

- décompose la navigation en trois niveaux afin de s'adapter aux différents dispositifs matériels (déplacement du *conveyor*, de l'*offsetConveyor* par rapport au *conveyor* ou de l'utilisateur dans son espace de déplacement physique),

- permet aux utilisateurs de transporter leurs outils d'interaction lorsqu'ils naviguent,
- permet aux utilisateurs de comprendre quelles sont leurs capacités de perception et d'interaction et quelles sont celles des autres utilisateurs,
- offre différentes possibilités pour la navigation collaborative.

Ces différentes fonctionnalités ont été utilisées pour réaliser une expérimentation de navigation collaborative dans le cadre du *3DUI Contest 2012* et pour mettre en œuvre un système de caméra pour examiner à plusieurs des données scientifiques dans le cadre du projet ANR Collaviz.

Enfin, le modèle de CVII a principalement été utilisé pour intégrer des espaces de déplacement physique, des espaces de visualisation et des espaces d'interaction. Pour compléter l'étude de ce modèle, il serait intéressant de l'évaluer avec d'autres espaces d'interaction comme des espaces de restitution sonore spatialisée ou des espaces haptiques. Il faudrait également proposer de nouvelles métaphores pour représenter ces nouveaux espaces d'interaction dans le monde virtuel.

Chapitre 6

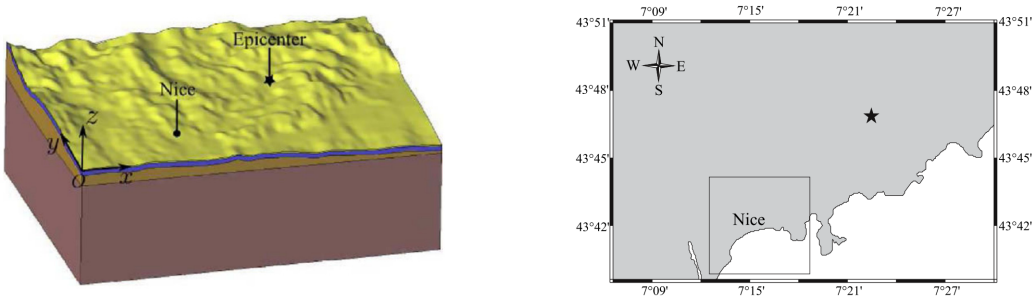
Expérimentations sur l'exploration collaborative de données scientifiques

Dans le cadre du projet ANR Collaviz et du projet Européen Visionair [1], nous avons réalisé une série d'expérimentations sur l'exploration collaborative de données scientifiques en environnement virtuel [Fleury *et al.*, 2012]. Le projet Visionair est un projet de collaboration entre plusieurs laboratoires Européens permettant de mutualiser les infrastructures de réalité virtuelle et d'effectuer des travaux de rechercher en commun. Dans ce contexte, ces expérimentations ont été menées en collaboration avec le professeur Anthony Steed de l'*University College of London* lors d'une mobilité de trois mois à Londres. Elles consistaient à demander à un utilisateur de manipuler un plan de *clipping* pour analyser des données scientifiques soit tout seul, soit avec un autre utilisateur situé à distance (l'un étant à Rennes et l'autre à Londres). L'objectif de ces expérimentations était de comparer une manipulation mono-utilisateur et une manipulation collaborative à distance.

Ces expérimentations ont été mises en œuvre grâce au *framework* Collaviz que nous avons développé durant cette thèse. Ce *framework* permet de déployer un environnement virtuel collaboratif distribué en se basant sur les trois contributions présentées dans les chapitres précédents : un modèle d'adaptation dynamique de la distribution des données (chapitre 3), un modèle d'architecture logicielle pour modéliser les objets virtuels (chapitre 4) et un modèle générique pour intégrer les espaces d'interaction des utilisateurs dans l'environnement virtuel (chapitre 5). Ainsi, ces expérimentations ont permis de valider chacune des contributions de cette thèse lors du déploiement d'un environnement virtuel collaboratif entre deux sites distants.

Dans la partie 6.1, nous présentons le cadre général de ces expérimentations. Puis, dans la partie 6.2, nous expliquons comment le *framework* Collaviz a permis de déployer un environnement virtuel collaboratif entre Rennes et Londres. Enfin, dans la partie 6.3, nous détaillons le déroulement de ces expérimentations et les résultats obtenus.

1. `www.infra-visionair.eu`



(a) (b)

FIGURE 6.1 – (a) Vue 3D de la zone étudiée avec les différentes couches géologiques et (b) carte de la zone étudiée : l'épicentre est indiqué par une étoile [Dupros *et al.*, 2010].

6.1 Cadre général des expérimentations

Lors des expérimentations, les utilisateurs manipulent un plan de *clipping* pour réaliser interactivement des coupes successives dans des données scientifiques. Ces données sont des données sismiques issues de la simulation physique d'un séisme hypothétique, mais réaliste (au niveau de la localisation et de l'intensité) dans la région de Nice (cf. figure 6.1). Elles nous ont été fournies par le BRGM (partenaire du projet ANR Collaviz). Les données utilisées pour les expérimentations sont les isosurfaces de la déformation maximale du sol *(Peak Ground Displacement)* obtenue durant la simulation physique [Dupros *et al.*, 2010]. Ces isosurfaces sont organisées de façon concentrique atour de l'épicentre (cf. figure 6.2).

Dans la partie 6.1.1, nous détaillons la tâche de manipulation que les utilisateurs doivent effectuer. Puis, nous présentons les deux techniques utilisées pour réaliser cette tâche : une manipulation mono-utilisateur (cf. partie 6.1.2) et une co-manipulation où deux utilisateurs (cf. partie 6.1.3).

6.1.1 Tâche à réaliser lors des expérimentations

Pour analyser ces données, les utilisateurs doivent trouver trois points d'intérêt situés à l'intérieur des données. Dans le cadre de nos expérimentations, ces points d'intérêt sont matérialisés par des petites sphères rouges (cf. figure 6.3). Pour examiner l'intérieur des données, les utilisateurs peuvent manipuler un plan de *clipping* qui permet de réaliser une section des données (cf. figure 6.4). Une fois que les utilisateurs ont trouvé les trois points d'intérêt, ils doivent placer le plan de *clipping* de manière à ce qu'il passe précisément par chacun de ces trois points en même temps pour réaliser une section des données qui montre les trois points d'intérêt (cf. figure 6.5). Afin que la technique de navigation utilisée n'ait pas d'impact sur la manipulation à réaliser, les utilisateurs ne peuvent pas naviguer à grande échelle dans le monde virtuel. Ils sont uniquement autorisés à se déplacer en utilisant leurs déplacements physiques dans le dispositif immersif qu'ils utilisent.

6.1.2 Manipulation mono-utilisateur du plan de *clipping*

L'utilisateur manipule seul le plan de *clipping* grâce à une cible repérée par un système de *tracking*. Cette cible est rattachée directement au centre du plan de *clipping* (cf. figure 6.6). Ainsi, l'utilisateur peut manipuler le plan de *clipping* en effectuant des translations et des rotations de son centre selon les 6 degrés de liberté.

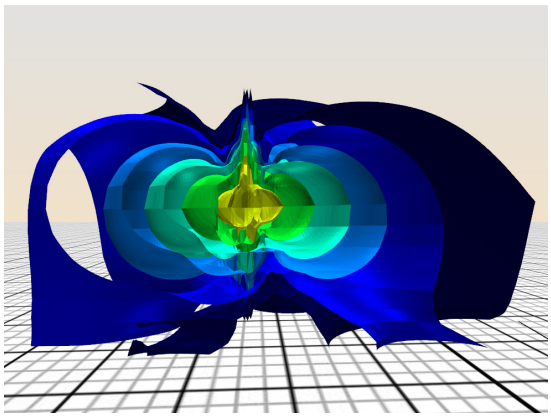


FIGURE 6.2 – Isosurfaces de la déformation maximale du sol autour de l'épicentre.

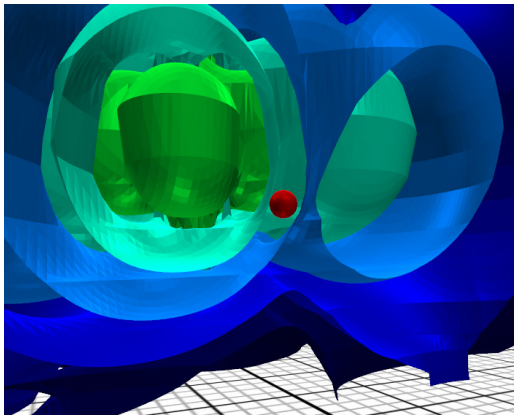


FIGURE 6.3 – Sphère rouge représentant un point d'intérêt dans les données.

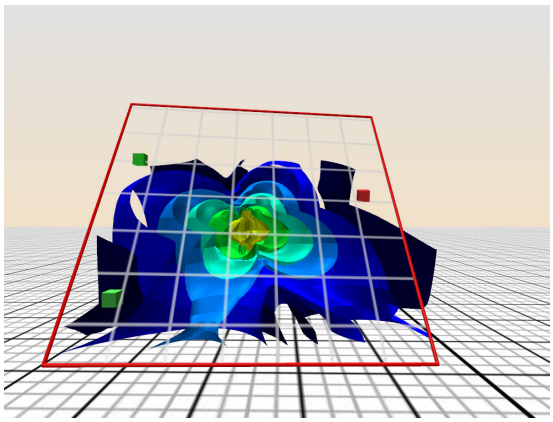


FIGURE 6.4 – Plan de *clipping* permettant de réaliser une section des données.

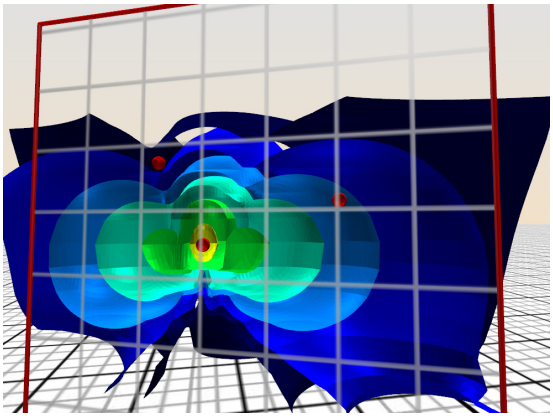


FIGURE 6.5 – Section des données montrant les trois points d'intérêt en même temps.

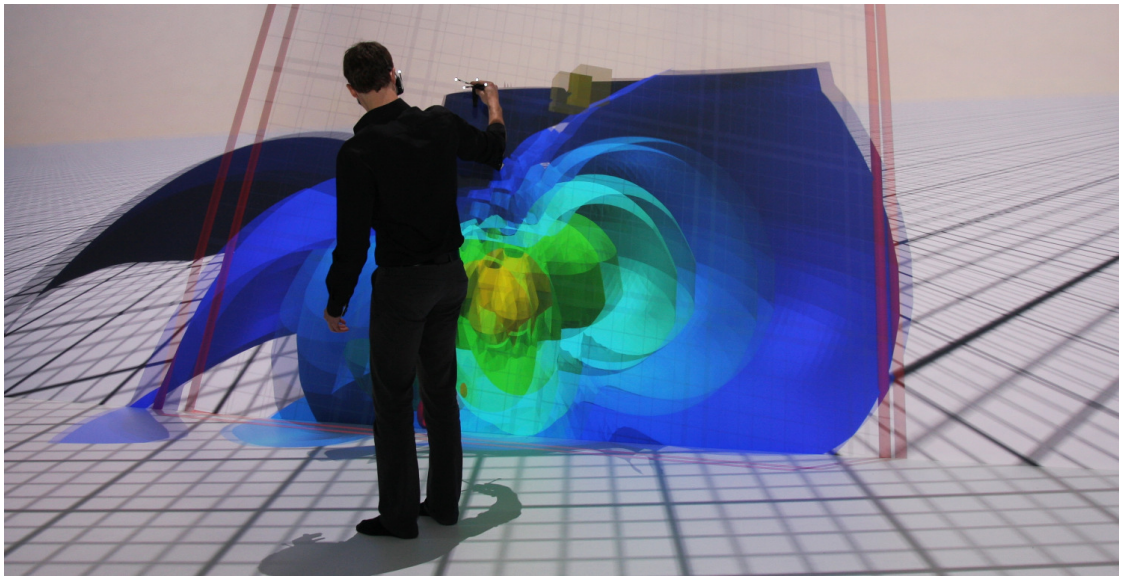
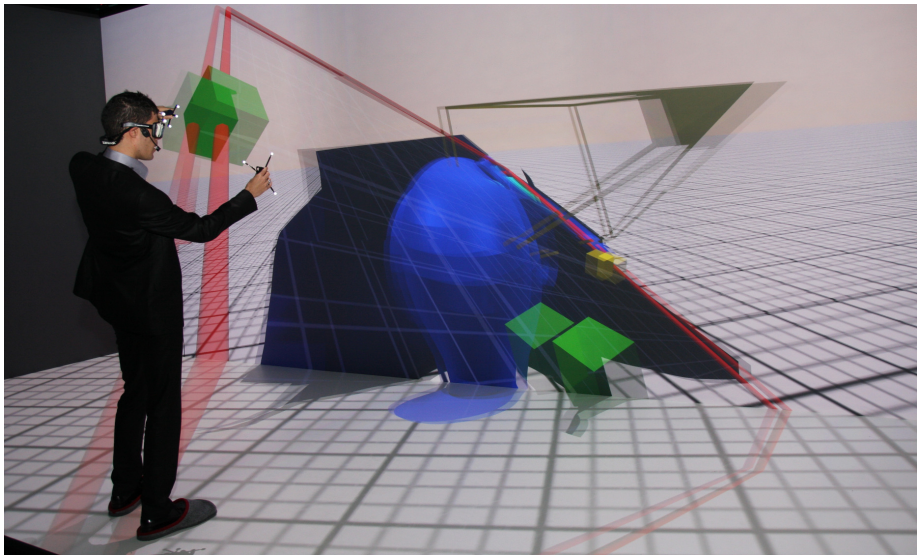
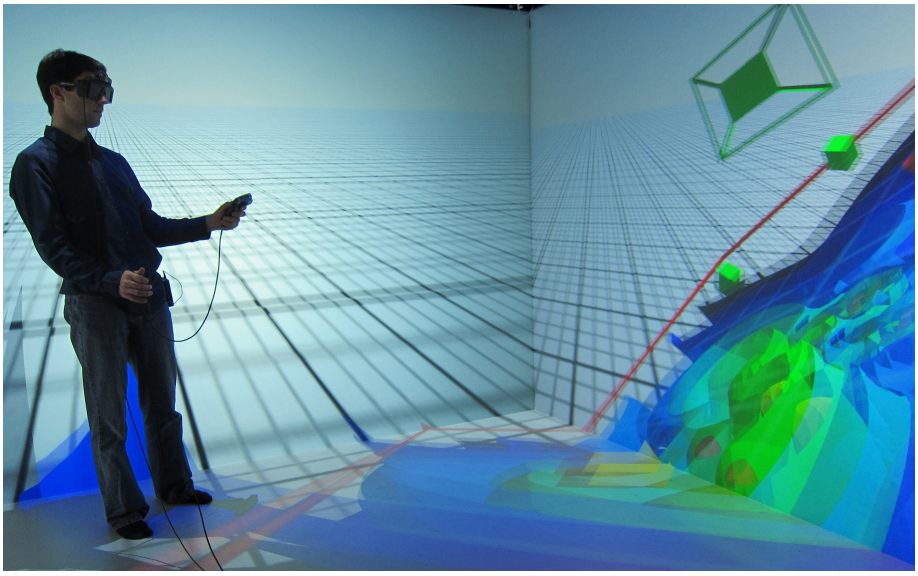


FIGURE 6.6 – Manipulation mono-utilisateur du plan de *clipping* par le centre (cube jaune).



(a)



(b)

FIGURE 6.7 – Co-manipulation du plan de *clipping* grâce à une technique de manipulation à trois mains : (a) l'utilisateur situé à Rennes contrôle deux points (cubes verts) et (b) l'utilisateur situé à Londres contrôle un point (cube jaune visible sur le sol).

6.1.3 Co-manipulation à deux utilisateurs du plan de *clipping*

Pour la manipulation collaborative, nous avons utilisé la technique de manipulation à trois mains proposée par [Aguerreche *et al.*, 2009]. L'utilisateur situé à Rennes utilise ses deux mains pour contrôler deux points virtuels grâce à deux cibles repérées par un système de *tracking*, tandis que celui situé à Londres utilise une seule main pour contrôler un point virtuel grâce à une cible repérée par un système de *tracking*. Les positions de ces trois points virtuels définissent la position du plan de *clipping* et permettent aux deux utilisateurs de le manipuler ensemble (cf. figure 6.7).

Chaque utilisateur est représenté dans le monde virtuel par sa pyramide de vue afin que l'autre puisse comprendre ce qu'il voit, ainsi que par son ou ses point(s) de contrôle du plan afin que l'autre puisse comprendre ce qu'il est en train de faire (cf. figure 6.7). De plus, les deux utilisateurs peuvent communiquer ensemble par la voix grâce à un microphone fixé sur chaque utilisateur et un retour sonore dans chacune des salles immersives.

6.2 Mise en œuvre des expérimentations

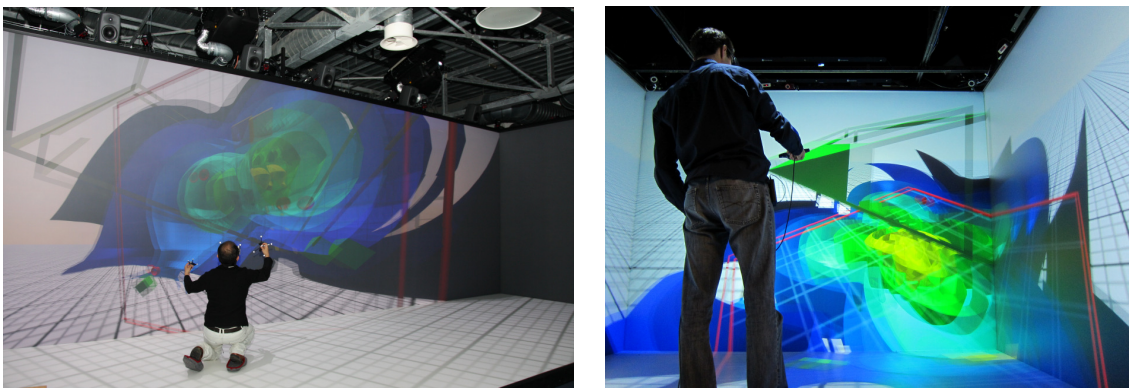
Pour mettre en œuvre ces expérimentations avec une co-manipulation à distance, nous avons déployé un environnement virtuel collaboratif entre Rennes et Londres grâce au *framework* Collaviz. Nous expliquons dans cette partie comment ce *framework* a permis de distribuer les données de l'environnement virtuel sur le réseau (partie 6.2.1) et d'adapter le système aux différents dispositifs matériels utilisés sur chacun des sites (partie 6.2.2).

6.2.1 Distribution des données sur le réseau

Le *framework* Collaviz a permis de distribuer l'environnement virtuel collaboratif entre Rennes et Londres afin de réaliser des co-manipulations à distance. Ce *framework* est basé sur une architecture réseau client/serveur comme décrit dans la partie 3.5. Le serveur était situé à Rennes, mais dans un bâtiment différent de celui où se trouve la salle de réalité virtuelle. Deux clients (l'un à Rennes et l'autre à Londres) étaient connectés à ce serveur en utilisant la connexion TCP proposée par le *framework* Collaviz. Une synchronisation forte était utilisée entre les deux clients. Les temps de latence sur le réseau pouvaient aller jusqu'à 50 ms entre le client de Londres et le serveur. Le modèle d'adaptation dynamique de la distribution des données, présenté dans le chapitre 3, a permis de choisir indépendamment pour chaque objet du monde virtuel un mode de distribution particulier parmi les trois modes proposés : centralisé, hybride ou répliqué. Nous avons donc pu choisir le mode de distribution de chaque objet en fonction de son rôle dans le monde virtuel afin d'avoir le compromis adéquat entre cohérence et réactivité lors des interactions.

Pour la manipulation mono-utilisateur, les données scientifiques étaient traitées sur le serveur (mode centralisé), tandis que le plan de *clipping*, les points d'intérêt et la main virtuelle de l'utilisateur étaient traités sur le nœud de l'utilisateur (mode hybride) afin de garantir la meilleure réactivité possible lors des interactions.

Pour la co-manipulation à deux utilisateurs, les deux points de contrôle de l'utilisateur situés à Rennes étaient traités sur le nœud de Rennes (mode hybride avec le référent à Rennes), tandis que le point de contrôle de l'utilisateur situés à Londres était traité sur le nœud de Londres (mode hybride avec le référent à Londres) afin de garantir la meilleure réactivité possible lorsque les utilisateurs interagissent avec ces points de contrôle. Par contre, les données scientifiques, mais aussi les points d'intérêt et le plan de de *clipping* étaient traités sur le serveur (mode centralisé) afin de garantir une bonne cohérence entre les deux sites. Cela permettait de s'assurer que les deux utilisateurs voyaient le plan de *clipping* au même endroit dans les données, au même moment, et qu'ils pouvaient le manipuler ensemble sans incompréhension entre eux. Avec cette distribution des données, un léger décalage peut survenir entre les points de contrôle manipulés par les utilisateurs et le plan de *clipping* à cause de la latence réseau. Cependant, comme ce décalage n'intervient pas directement sur les points de contrôle que les utilisateurs manipulent, aucun participant n'a ressenti de gène due à cette latence durant les expérimentations.



(a) (b)

FIGURE 6.8 – (a) Salle « Immersia » à Rennes et (b) salle immersive de l'UCL à Londres utilisées lors des expérimentations.

6.2.2 Adaptation aux dispositifs immersifs des deux sites

Pour réaliser les expérimentations sur les deux sites, nous avons déployé le *framework* Collaviz sur chacun des dispositifs matériels disponibles sur place. Chacun des deux sites possède des dispositifs de réalité virtuelle complexes :

- **À Rennes :** la salle de réalité virtuelle est constituée de deux grands écrans de visualisation (env. 9,6 m × 3 m). Elle peut être décrite par un parallélépipède rectangle dont deux des faces correspondent aux deux écrans : le mur du fond et le sol de la salle (cf. figure 6.8(a)). Un système de *tracking* utilisant des caméras infrarouges permet de repérer la tête de l'utilisateur et ses deux mains à l'intérieur de ce parallélépipède rectangle. Huit projecteurs sont utilisés pour projeter les images sur le mur du fond (projection par l'arrière) et trois projecteurs sont utilisés pour le sol (projection par le dessus). Une machine permet de contrôler les huit projecteurs du mur du fond et une autre machine permet de contrôler ceux du sol. Les applications doivent donc être distribuées sur ces deux machines.
- **À Londres :** la salle de réalité virtuelle est constituée de quatre grands écrans de visualisation (env. 3 m × 2,1 m). Elle peut être décrite par un parallélépipède rectangle dont quatre des faces correspondent aux écrans : le mur du fond, le mur de droite, le mur de gauche et le sol de la salle (cf. figure 6.8(b)). Un système de *tracking* à ultrasons permet de repérer la tête de l'utilisateur et une de ses mains à l'intérieur de ce parallélépipède rectangle. Un projecteur est utilisé pour projeter les images sur chacun des écrans (projection par l'arrière pour les trois murs et par le dessus pour le sol). Chaque projecteur est contrôlé par une machine particulière et une cinquième machine permet de contrôler les quatre autres. Les applications doivent donc être lancées sur cette cinquième machine, puis distribuées sur les quatre autres.

Premièrement, la décomposition du modèle d'architecture logicielle PAC-C3D nous a permis de choisir un *viewer* adapté à ces deux dispositifs immersifs. Ainsi, parmi les différentes *Présentations* visuelles proposées pour l'environnement virtuel, nous avons pu utiliser le *viewer* jReality qui offre des fonctionnalités adaptées aux dispositifs de type salle immersive. D'une part, il permet de générer des images stéréoscopiques et de déformer ces images en fonction de la position de la tête de l'utilisateur (*head-tracking)*. D'autre part, il

permet de distribuer le graphe de scène de l'environnement virtuel sur plusieurs machines et de générer une vue particulière de ce graphe de scène sur chacune de ces machines. Cela nous a permis de distribuer notre application sur les différentes machines qui gèrent les projecteurs de chaque salle immersive.

Deuxièmement, le modèle de *Cabine Virtuelle d'Interaction Immersive* nous a permis d'intégrer dans l'environnement virtuel l'espace de visualisation, l'espace de déplacement physique et l'espace d'interaction associés à chacun des deux dispositifs (cf. partie 5.4.1.2). La description de l'espace de déplacement physique nous a permis de réaliser facilement une co-localisation de la tête de l'utilisateur présent dans la salle immersive. À partir de cette co-localisation, la modélisation de l'espace de visualisation nous a permis de configurer le *viewer* jReality avec les positions relatives de chacun des écrans et de générer des images appropriées en fonction de la position de la tête de l'utilisateur. Enfin, la description de l'espace d'interaction nous a permis de réaliser facilement une co-localisation des mains de l'utilisateur avec les points de contrôle du plan de *clipping* afin de lui permettre de manipuler ce plan de façon intuitive.

6.3 Expérimentations de manipulation du plan de *clipping*

Afin de comparer la technique de manipulation mono-utilisateur du plan de *clipping* avec la technique de co-manipulation du plan de *clipping* à deux utilisateurs distants, nous avons réalisé une série d'expérimentations. Les résultats obtenus lors de la première expérimentation (partie 6.3.1) n'étant pas suffisamment significatifs, nous avons mené une deuxième expérimentation (partie 6.3.2) en modifiant les conditions expérimentales à partir des conclusions de la première expérimentation. Les résultats de ces deux expérimentations sont discutés dans la partie 6.3.3.

6.3.1 Première expérimentation

Dans cette première expérimentation, nous avons demandé à 30 participants de réaliser la tâche décrite dans la partie 6.1.1 en utilisant les deux techniques de manipulation du plan de *clipping* présentées dans la partie 6.1.2 et dans la partie 6.1.3. Après une période d'entrainement, nous avons demandé à chaque participant de réaliser 5 fois la tâche pour chaque technique de manipulation. À chaque essai, les trois points d'intérêt étaient replacés dans les données scientifiques en choisissant aléatoirement une configuration parmi plusieurs configurations pertinentes pour la manipulation. Les différentes configurations des trois points d'intérêt étaient les mêmes pour les deux techniques de manipulation et pour tous les participants (mais pas forcément dans le même ordre). À chaque essai, nous avons mesuré le temps nécessaire pour effectuer la tâche, c'est-à-dire pour que le plan de *clipping* passe par les trois points d'intérêt en même temps.

Cependant, les écarts de temps obtenus entre les différents participants sont très importants, et après une analyse statistique des résultats, nous n'avons pas pu tirer de conclusion significative en faveur de l'une ou de l'autre des deux techniques. Nous avons émis trois hypothèses pour expliquer ces résultats non significatifs :

1. La distance entre les trois points d'intérêt était trop faible (et ne variait quasiment pas entre les différentes configurations). En effet, pour certains participants, la tâche était très facile à effectuer et cela ne leur prenait que quelques secondes. Pour ces participants, il est difficile de voir une différence entre les deux techniques. De plus,

avec une la tâche facile à réaliser, le temps nécessaire aux deux participants (qui ne se connaissaient pas) pour se coordonner au début de la co-manipulation pénalise plus cette technique par rapport à la manipulation mono-utilisateur.

2. La recherche des trois points d'intérêt à l'intérieur des données scientifiques introduisait un biais dans l'évaluation des deux techniques de manipulation. En effet, les participants n'avaient pas d'expérience particulière dans l'analyse de données scientifiques et il était très difficile pour certains d'entre eux de trouver les trois points d'intérêt dans les données. Pour ces participants, il est difficile de comparer les deux techniques entre elles car le temps pour réaliser la tache dépend plus de la facilité à trouver les points que du temps nécessaire pour bien placer le plan de *clipping*.
3. Le temps d'apprentissage n'était peut-être pas assez long. En effet, nous avons constaté une nette amélioration du temps nécessaire pour effectuer la tâche au fur et à mesure des essais surtout pour la technique de co-manipulation.

6.3.2 Deuxième expérimentation

Pour obtenir des résultats plus significatifs, nous avons mené une deuxième expérimentation en adaptant le protocole expérimental en fonction des trois hypothèses établies à partir de la première expérimentation. Pour valider l'hypothèse 1, nous avons choisi de faire varier de façon plus importante l'écartement entre les trois points d'intérêt. De plus, pour être sûr de ne pas rencontrer de problèmes similaires à ceux décrits dans les hypothèses 2 et 3, nous avons décidé, d'une part, d'enlever les données scientifiques afin de ne pas mélanger la tâche de manipulation avec la tâche de recherche et, d'autre part, d'allonger la durée de l'entrainement pour les deux techniques de manipulation.

6.3.2.1 Description de l'expérimentation

Dans cette deuxième expérimentation, même s'il n'y a plus de donnés scientifiques, les participants doivent toujours placer un plan devant passer par trois points du monde virtuel en même temps (cf. figure 6.9). Pour manipuler ce plan, les participants utilisent soit une manipulation mono-utilisateur (partie 6.1.2), soit une co-manipulation avec un autre participant situé à distance (partie 6.1.3). Des configurations de trois points avec un écartement entre les points plus important ont été introduites, et les différentes configurations de trois points ont été classées en deux catégories en fonction de la moyenne des distances 2 à 2 entre les trois points :

- une catégorie « **Proches** » où la moyenne des distances est inférieure à 0,6 m,
- une catégorie « **Éloignés** » où la moyenne des distances est supérieure à 1,4 m.

Population. 32 personnes ont participé à cette expérimentation (6 femmes et 26 hommes), âgés de 18 à 50 ans (moyenne : 26, écart type : 6.72). Aucun de ces participants n'avaient participé à la première expérimentation. Ces participants ont été séparés en deux groupes :

- un groupe **G1** de 16 participants à Rennes qui ont réalisé la co-manipulation avec un participant de référence à Londres (qui était toujours le même),
- un groupe **G2** de 16 participants (8 à Rennes, 8 à Londres) qui ont réalisé la co-manipulation avec un autre participant du groupe.

Dans chaque groupe, un participant sur deux réalise d'abord la manipulation mono-utilisateur puis la co-manipulation, et inversement pour les autres participants.

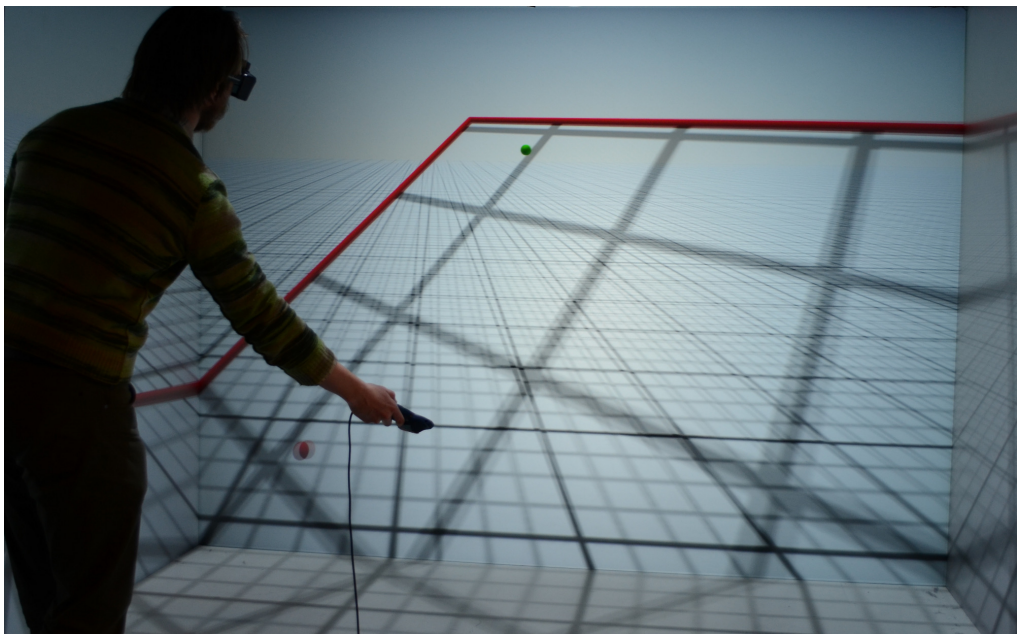


FIGURE 6.9 – 2ième expérimentation de manipulation du plan de *clipping* sans les données.

Procédure. Chaque participant effectue d'abord un entrainement pour les deux techniques de manipulation en respectant l'ordre qui lui a été attribué (manipulation mono-utilisateur puis co-manipulation, ou l'inverse). Après avoir laissé le temps au participant de se familiariser avec l'environnement virtuel, l'entrainement consiste à réaliser 4 fois la tâche avec chacune des techniques. Une fois que le participant a réalisé l'entrainement, il passe à la phase réelle d'expérimentation : il effectue 10 fois la tâche pour chacune des techniques de manipulation toujours en respectant l'ordre qui lui a été attribué. Parmi ces 10 essais, 5 sont effectués avec une configuration de trois points **Proches** et 5 sont effectués avec une configuration de trois points **Éloignés**. Les 5 configurations de trois points **Proches** sont mélangées aléatoirement avec les 5 configurations de trois points **Éloignés** lors des 10 essais. L'expérimentation dure au total environ 45 minutes avec l'entrainement.

Données collectées. Pour chaque essai, nous avons enregistré le temps nécessaire au participant pour réaliser la tâche. Le décompte du temps démarre lorsque le participant active la manipulation du plan en appuyant sur un bouton et s'arrête automatiquement lorsque le plan atteint les trois points. À l'issue de l'expérimentation, les participants remplissaient également un questionnaire subjectif sur les deux techniques de manipulation.

6.3.2.2 Résultats

À partir des données collectées lors de l'expérimentation, nous avons mené des analyses comparatives entre la technique de manipulation mono-utilisateur et la technique de co-manipulation à deux utilisateurs distants. Lors de ces analyses comparatives, nous avons différencié les cas où les points étaient **Proches** et les cas où les points étaient **Éloignés**.

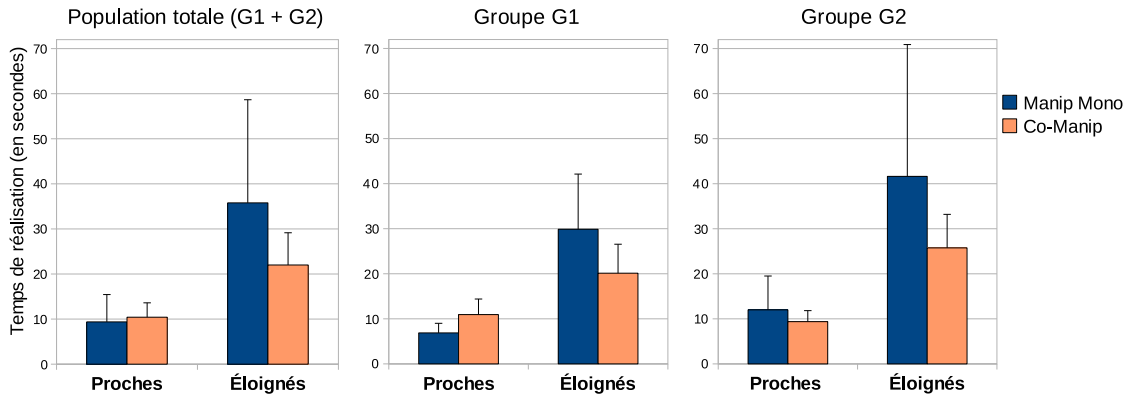


FIGURE 6.10 – Moyennes et écarts types du temps de réalisation de la tâche avec les deux techniques (a) pour toute la population, (b) pour le groupe **G1** et (c) pour le groupe **G2**.

Temps de réalisation de la tâche. Pour chaque participant, nous avons calculé la moyenne du temps de réalisation de la tâche des 5 essais de chaque catégorie de points pour chacune des deux techniques (4 cas). Nous avons ensuite effectué un test de Student (t-test) sur ces moyennes (en secondes) en comparant les deux techniques (Manip Mono, Co-Manip). Nous avons également calculé les moyennes (M) et les écarts types (ET) entre tous les participants pour chacun des 4 cas.

Premièrement, nous considérons toute la population **G1** + **G2** (cf. figure 6.10(a)). Lorsque les points sont **Proches**, le temps de réalisation moyen diffère peu entre la Manip Mono (M = 9.38 sec, ET = 6.06 sec) et la Co-Manip (M = 10.42 sec, ET = 3.19 sec) et cet écart n'est pas significatif ($t(54)$ = -0.76, p = 0.44). Par contre, lorsque les points sont **Éloignés**, le temps de réalisation moyen est largement plus court avec la Co-Manip (M = 22 sec, ET = 7.15 sec) qu'avec la Manip Mono (M = 35.78 sec, ET = 22.88 sec) et cet écart est significatif ($t(54)$ = 2.84, p = 0.006).

Deuxièmement, nous considérons uniquement le groupe **G1** où les participants réalisent la Co-Manip avec toujours le même participant de référence à Londres (cf. figure 6.10(b)). Lorsque les points sont **Proches**, le temps de réalisation est légèrement plus court avec la Manip Mono (M = 6.88 sec, ET = 2.13 sec) qu'avec la Co-Manip (M = 10.94 sec, ET = 3.45 sec) et cet écart est significatif ($t(30)$ = -4.01, $p < 0.001$). Lorsque les points sont **Éloignés**, le temps de réalisation moyen est largement plus court avec la Co-Manip (M = 20.13 sec, ET = 6.43 sec) qu'avec la Manip Mono (M = 29.88 sec, ET = 12.25 sec) et cet écart est significatif ($t(30)$ = 2.82, p = 0.008).

Finalement, nous considérons uniquement le groupe **G2** où les participants réalisent la Co-Manip avec un autre participant du groupe (cf. figure 6.10(c)). Lorsque les points sont **Proches**, le temps de réalisation est légèrement plus court avec la Co-Manip (M = 9.38 sec, ET = 2.45 sec) qu'avec la Manip Mono (M = 12 sec, ET = 7.5 sec), mais cet écart n'est pas significatif ($t(22)$ = 0.96, p = 0.35). Par contre, lorsque les points sont **Éloignés**, le temps de réalisation moyen est largement plus court avec la Co-Manip (M = 25.75 sec, ET = 7.44 sec) qu'avec la Manip Mono (M = 41.63 sec, ET = 29.27 sec), cependant cet écart est peu significatif ($t(22)$ = 1.49, p = 0.149). Nous pensons que cet écart peu

significatif malgré un écart de moyennes important est du à plusieurs raisons :

- Certains participants ont eu beaucoup de mal à réaliser la tâche avec la Manip Mono lorsque les points étaient **Éloignés** (d'où l'écart type important pour la Manip Mono avec les points **Éloignés**).
- Certains binômes ont eu du mal à se coordonner au début pour la Co-Manip (problème pour se comprendre en anglais, etc.) ce qui a entrainé des écarts assez importants entre les binômes.
- Dans ce groupe **G2**, chaque participant réalisait seul la tâche avec la Manip Mono et en binôme la tâche avec la Co-Manip. Il y a donc deux fois plus d'échantillons pour la Manip Mono que pour la Co-Manip.

Questionnaire subjectif. Après l'expérimentation, il était demandé aux participants de remplir un questionnaire de préférences où ils devaient noter de 1 à 7 les deux techniques de manipulation pour les deux catégories de points en fonction de 5 critères subjectifs : la *fatigue*, la *facilité*, la *précision*, le caractère *naturel* et l'*appréciation globale* de la technique. De plus, pour la Co-Manip, il leur était aussi demandé de noter de 1 à 7 pour les deux catégories de points s'ils avaient ressenti le sentiment de *collaborer* avec un autre participant ou non. Nous avons calculé les moyennes et les écarts types des notes attribuées par toute la population (**G1** + **G2**) dans chacun des cas. Puis, pour comparer les réponses concernant les deux techniques, nous avons calculé les différences (Δ) de moyennes entre les deux techniques et nous avons effectué un test de Wilcoxon (Signed Rank) sur les notes attribuées par les participants pour savoir si ces différences sont significatives ou non.

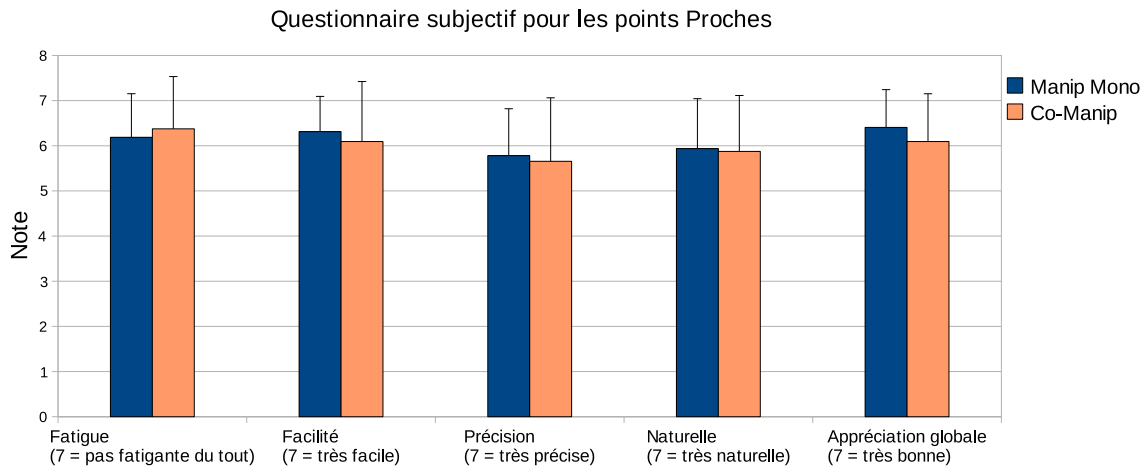


FIGURE 6.11 – Moyennes et écarts types des notes du questionnaire subjectif avec les deux techniques pour les points **Proches**.

Premièrement, si nous considérons les réponses au questionnaire subjectif pour les points **Proches** (cf. figure 6.11), il ne semble pas se dégager de préférences particulières pour une des deux techniques de manipulation. En effet, la différence de moyennes entre les deux techniques est très faible pour chaque critère et l'analyse statistique montre que ces différences ne sont pas significatives : la *fatigue* ($\Delta = 0.19$, $p = 0.21$), la *facilité* ($\Delta = 0.22$, $p = 0.58$), la *précision* ($\Delta = 0.13$, $p = 0.77$), le caractère *naturel* ($\Delta = 0.06$, $p = 1$) et l'*appréciation globale* ($\Delta = 0.31$, $p = 0.08$).

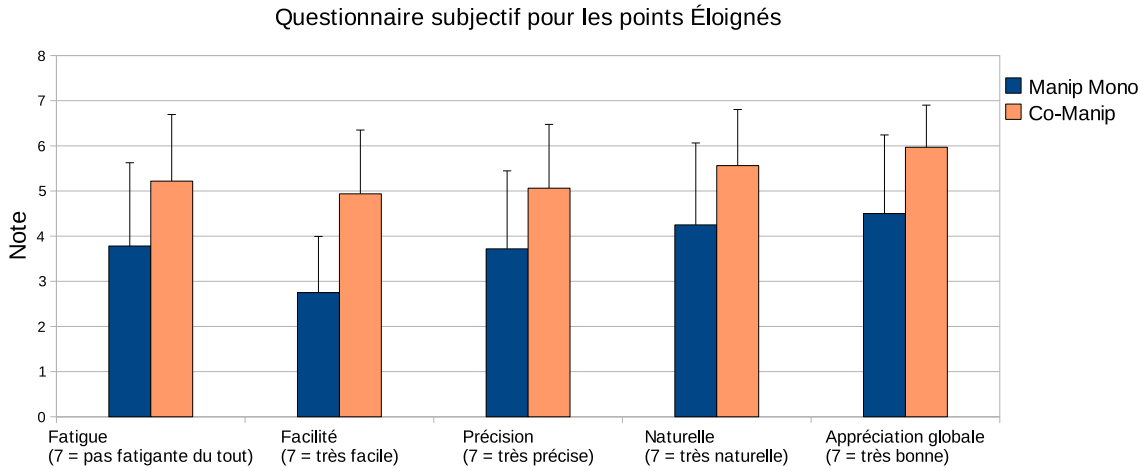


FIGURE 6.12 – Moyennes et écarts types des notes du questionnaire subjectif avec les deux techniques pour les points **Éloignés**.

Deuxièmement, si nous considérons les réponses au questionnaire subjectif pour les points **Éloignés** (cf. figure 6.12), il semble que les participants aient préférés dans l'ensemble la Co-Manip à la Manip Mono. En effet, la différence de moyennes entre les deux techniques est relativement importante en faveur de Co-Manip (supérieure à 1 point) pour chaque critère et l'analyse statistique montre que ces différences sont très significatives : la *fatigue* ($\Delta = 1.44$, $p < 0.001$), la *facilité* ($\Delta = 2.19$, $p < 0.001$), la *précision* ($\Delta = 1.34$, $p < 0.001$), le caractère *naturel* ($\Delta = 1.31$, $p < 0.001$) et l'*appréciation globale* ($\Delta = 1.47$, $p < 0.001$).

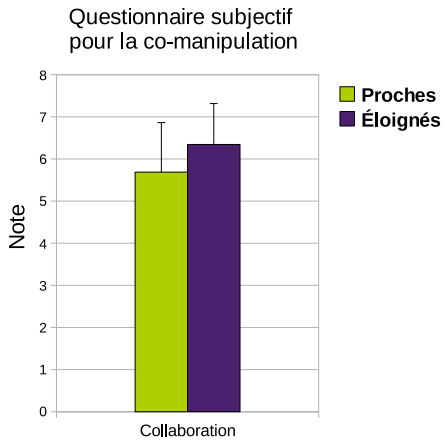


FIGURE 6.13 – Moyennes et écarts types des notes du sentiment de *collaboration* pour la co-manipulation.

Enfin, quelle que soit la distance entre les points, il semble que les participants aient eu un sentiment relativement fort de *collaborer* avec une autre personne distante (cf. figure 6.13) : pour les points **Proches** ($M = 5.69$, $ET = 1.18$) et pour les points **Éloignés** ($M = 6.34$, $ET = 0.97$). De plus, il est intéressant de noter que ce sentiment de *collaboration* est légèrement plus fort lorsque les points sont **Éloignés** et l'analyse statistique montre que cette différence est significative ($\Delta = 0.66$, $p = 0.007$). Cela nous amène à penser que la collaboration avec un autre participant est d'autant plus pertinente que la tâche est difficile (comme lorsque les points sont plus éloignés).

6.3.3 Discussion

Les résultats obtenus lors de la deuxième expérimentation montrent que la technique de co-manipulation à distance est plus efficace que la technique de manipulation mono-utilisateur lorsque les points sont plus éloignés. Cela peut s'expliquer par deux facteurs : lorsque les points sont plus éloignés, premièrement, la tâche de positionnement du plan demande plus de précision et, deuxièmement, les utilisateurs ne peuvent pas voir tous les points en même temps dans leur champ de vision et ils doivent tourner la tête lorsqu'ils ajustent le positionnement du plan. Ces deux facteurs font que la tâche est globalement plus difficile avec les points les plus éloignés. Le temps moyen nécessaire pour accomplir la tâche est alors significativement plus court avec la co-manipulation qu'avec la manipulation mono-utilisateur car chacun des deux utilisateurs peut concentrer son action sur seulement certains points. De plus, pour les points les plus éloignés, le questionnaire subjectif a montré que les utilisateurs ont préféré la co-manipulation à la manipulation mono-utilisateur de manière globale, mais aussi qu'ils l'ont trouvée moins fatigante, plus facile, plus précise et plus naturelle.

Lorsque les points sont plus proches et donc que la tâche à réaliser est plus facile, nous n'avons pas pu dégager de différence significative entre les deux techniques de manipulation lors de cette deuxième expérimentation. En effet, le temps de réalisation moyen de la tâche est plus court soit avec la manipulation mono-utilisateur, soit avec la co-manipulation selon les populations étudiées, et cela de manière non significative. De plus, pour les points les plus proches, les participants n'ont pas exprimé de préférence significative pour l'une ou l'autre des deux techniques. Cette absence de différence significative entre les deux techniques vient corroborer l'hypothèse 1, émise lors de la première expérimentation.

Par ailleurs, il n'y pas d'évolution significative du temps de réalisation de la tâche au fur et à mesure des essais lors de la deuxième expérimentation. Cela nous amène à penser que l'hypothèse 3, émise lors de la première expérimentation, est justifiée et que l'entrainement réalisé lors de la deuxième expérimentation était suffisant. Par contre, rien ne nous permet de conclure en faveur ou contre l'hypothèse 2, émise lors de la première expérimentation. Il serait alors intéressant de réaliser une nouvelle expérimentation en remettant les données scientifiques et en utilisant les configurations de points les plus éloignés. Il pourrait aussi être intéressant de proposer des solutions supplémentaires pour permettre aux utilisateurs de mieux tirer profit de la collaboration lors de la rechercher des points d'intérêts.

Enfin, les résultats moins significatifs obtenus pour le groupe **G2** par rapport à ceux obtenus pour le groupe **G1** montre qu'une expérimentation collaborative est difficile à mettre en place tant les paramètres qui entrent en jeu sont nombreux. En effet, faire collaborer à distance deux personnes qui ne se connaissent pas entraine des difficultés supplémentaires au niveau de la langue, du vocabulaire, des prédispositions à travailler avec une autre personne et de la bonne volonté mise dans cette collaboration.

6.4 Conclusion

Dans le cadre du projet ANR Collaviz et du projet Européen Visionair, nous avons mené une série d'expérimentations sur la collaboration en environnement virtuel, en partenariat avec l'*University College of London*. Ces expérimentations avaient pour but de comparer une manipulation mono-utilisateur avec une manipulation collaborative à distance entre deux participants (l'un se trouvait à Rennes et l'autre se trouvait à Londres). La tâche à

réaliser consistait à placer un plan de *clipping* dans des données scientifiques de manière à ce qu'il montre, en même temps, trois points d'intérêts présents dans les données. Pour la manipulation mono-utilisateur, le participant manipule seul le plan de *clipping*. Pour la manipulation collaborative, les deux participants manipulent ensemble le plan de *clipping* en utilisant la technique de manipulation à trois mains proposée par [Aguerreche *et al.*, 2009]. Même s'il ne semble pas avoir de différences significatives entre les deux techniques lorsque les trois points sont proches, cette série d'expérimentations a montré que la manipulation collaborative à distance est plus efficace que la manipulation mono-utilisateur lorsque les trois points sont plus éloignés, c'est-à-dire lorsque la tâche est plus difficile à réaliser. Il serait intéressant de mener des expérimentations complémentaires pour déterminer le seuil à partir duquel la manipulation collaborative devient significativement plus efficace que la manipulation mono-utilisateur.

Par ailleurs, le *framework* Collaviz développé dans le cadre de cette thèse a été utilisé pour mettre en œuvre l'environnement virtuel collaboratif distribué entre Rennes et Londres. Le modèle d'adaptation dynamique de la distribution des données, présenté dans le chapitre 3, a permis de répartir les données et les traitements des objets virtuels entre les différentes machines du réseau. Nous avons ainsi pu choisir pour chaque objet le meilleur compromis entre cohérence et réactivité en fonction de son rôle lors des expérimentations. Le modèle d'architecture logicielle PAC-C3D, présenté dans le chapitre 4, a permis de coupler facilement le *framework* Collaviz à la librairie graphique jReality. Cette librairie graphique offre des fonctionnalités adaptées aux salles immersives de réalité virtuelle utilisées sur chacun des deux sites. Enfin, la *Cabine Virtuelle d'Interaction Immersive*, présentée dans le chapitre 5, a permis d'intégrer dans l'environnement virtuel les différents espaces d'interaction associés à ces salles immersives afin de co-localiser facilement les utilisateurs. Cette série d'expérimentations réalisée entre Rennes et Londres a permis de valider les trois contributions de cette thèse dans des conditions réelles de collaboration à distance entre deux sites géographiquement éloignés.

Conclusion

Bilan de la thèse

Nos travaux s'inscrivent dans le cadre général de la collaboration en environnement virtuel. Ce type de collaboration offre la possibilité à des utilisateurs situés à distance de se retrouver dans un environnement virtuel afin de réaliser des tâches communes. Elle leur permet également de partager des représentations virtuelles différentes de celles qu'ils percevraient dans le monde réel. L'objectif de cette thèse a été d'imaginer de nouvelles solutions pour concevoir et utiliser les environnements virtuels collaboratifs afin d'améliorer la collaboration à distance entre plusieurs utilisateurs. Dans ce contexte, nous avons proposé trois modèles permettant de formaliser les différents niveaux de conception et d'utilisation d'un environnement virtuel collaboratif. Le premier modèle propose un moyen d'adapter dynamiquement la distribution des données de l'environnement virtuel sur le réseau. Le deuxième modèle propose une architecture logicielle pour concevoir un environnement virtuel indépendamment de la distribution réseau et des composants matériels ou logiciels, spécifiques à chaque utilisateur. D'un point de vue plus conceptuel, le troisième modèle propose une solution générique pour permettre à des utilisateurs se servant de dispositifs matériels différents de collaborer efficacement dans un environnement virtuel.

Pour une bonne compréhension mutuelle entre les utilisateurs, il est essentiel de garantir que ces utilisateurs perçoivent tous, au même moment, le même état de l'environnement virtuel sans que cela dégrade leurs capacités d'interaction. La cohérence d'un environnement virtuel distribué dépend fortement de l'endroit sur le réseau où sont traitées les données relatives aux objets virtuels. Nous avons proposé un modèle d'adaptation dynamique de la distribution des données [Fleury *et al.*, 2010c] qui permet de mettre en œuvre trois modes de distribution grâce à un paradigme de référents et de proxys : un mode centralisé, un mode hybride et un mode répliqué. Chacun de ces modes de distribution offre un compromis particulier en termes de cohérence et de réactivité de l'environnement virtuel collaboratif. Le paradigme de référents et de proxys permet de choisir indépendamment pour chaque objet virtuel, le mode de distribution qui est le plus adapté à son rôle dans le monde virtuel. Ce choix de mode peut également être modifié dynamiquement durant la session collaborative en fonction des actions des utilisateurs ou des éventuelles perturbations techniques. Dans le cadre du projet ANR Collaviz, nous avons instancié et évalué ce modèle avec une architecture réseau de type client/serveur.

Afin de modéliser un environnement virtuel collaboratif indépendamment de la distribution réseau et des composants spécifiques à chaque utilisateur, nous avons proposé le modèle d'architecture logicielle PAC-C3D [Duval et Fleury, 2011b]. D'une part, ce modèle permet de mettre en œuvre le modèle d'adaptation dynamique de la distribution des données présenté dans le paragraphe précédent. D'autre part, il offre la possibilité d'adapter une application de réalité virtuelle aux différents composants matériels et logiciels qui font l'interface avec les utilisateurs. Pour cela, chaque objet virtuel est composé de trois facettes : une *Abstraction* qui stocke et traite les données relatives à l'objet, une ou plusieurs *Présentations* qui effectuent l'interface avec les utilisateurs et un *Contrôle* qui fait le lien à la fois entre les deux autres facettes et entre les différentes versions de cet objet sur les machines de chaque utilisateur. Le modèle PAC-C3D a permis d'intégrer facilement dans le *framework* Collaviz, trois librairies graphiques adaptées chacune à des dispositifs matériels différents, ainsi qu'un moteur physique permettant d'associer un comportement physique à certains objets virtuels. Il a également permis de mettre en place des retours multi-sensoriels en associant plusieurs *Présentations* à un même outil d'interaction.

De nombreuses applications nécessitent de prendre en compte l'environnement réel qui entoure les utilisateurs afin d'adapter les techniques d'interaction et de collaboration, mais aussi pour faire percevoir aux utilisateurs leurs propres capacités de perception et d'interaction, ainsi que celles des autres utilisateurs. Nous avons proposé le modèle de *Cabine Virtuelle d'Interaction Immersive* (CVII) [Fleury *et al.*, 2010a] pour intégrer les espaces d'interaction réels des utilisateurs dans l'environnement virtuel. Ce modèle est composé de deux parties : le *stage* qui modélise l'environnement réel des utilisateurs sous forme d'une hiérarchie d'espaces d'interaction (espace de visualisation, de restitution sonore, de déplacement physique, haptique, etc.) et le *conveyor* qui permet d'intégrer ce *stage* dans le monde virtuel. Le modèle de CVII fournit également un ensemble d'opérateurs permettant de mettre en œuvre différentes fonctionnalités pour l'interaction et la collaboration, ainsi qu'une structure pour représenter les espaces d'interaction des utilisateurs dans l'environnement virtuel. Grâce au modèle de CVII, nous avons pu mettre en place différentes applications collaboratives comme une expérimentation de navigation collaborative dans un bâtiment, dans le cadre du *3DUI Contest 2012*, et un système de caméras pour explorer à plusieurs des données scientifiques, dans le cadre du projet ANR Collaviz.

Ces trois modèles nous ont permis de concevoir et développer un *framework* pour mettre en œuvre des environnements virtuels collaboratifs distribués entre des sites distants, dans le cadre du projet ANR Collaviz. Le *framework* Collaviz est basé sur deux composants principaux : le service de collaboration et la librairie *IIVC*. Le service de collaboration modélise l'environnement virtuel en utilisant la décomposition PAC-C3D et implémente le modèle d'adaptation dynamique de la distribution des données en utilisant une couche de communication réseau développée par un des partenaires du projet ANR Collaviz. La librairie *IIVC* implémente le modèle de *Cabine Virtuelle d'Interaction Immersive* et intègre différentes métaphores de collaboration en utilisant également la décomposition proposée par le modèle PAC-C3D. Le *framework* Collaviz intègre différents modules permettant de représenter l'environnement virtuel : trois *viewers* adaptées à des dispositifs matériels différents grâce aux librairies graphiques Java3D, jReality et jMonkeyEngine, une représentation physique grâce au moteur physique JBullet, et une représentation permettant de faire le lien entre les objets virtuels et les périphériques d'interaction. Le Consortium Scilab (partenaire du projet ANR Collaviz) a également développé un *viewer* graphique supplémentaire dédié à la visualisation de données scientifiques. Le *framework* Collaviz a

été déployé sur différents dispositifs matériels allant de la simple station de travail jusqu'à la salle immersive de réalité virtuelle. Ce *framework* a été utilisé pour des applications de visualisation collaborative de données scientifiques dans le cadre du projet ANR Collaviz [Dupont *et al.*, 2010], mais aussi pour d'autres types d'applications collaboratives comme, par exemple, l'expérimentation de navigation collaborative réalisée dans le cadre du *3DUI Contest 2012* [Nguyen *et al.*, 2012] ou la démonstration présentée à la conférence ICAT 2011 [Duval et Fleury, 2011a].

Enfin, le *framework* Collaviz a permis de réaliser une série d'expérimentations sur l'exploration collaborative de données scientifiques en environnement virtuel [Fleury *et al.*, 2012]. Le but de ces expérimentations était de comparer une manipulation mono-utilisateur avec une manipulation collaborative entre deux utilisateurs situés à distance (l'un se trouvait à Rennes et l'autre se trouvait à Londres). La tâche à réaliser consistait à placer un plan de *clipping* dans des données scientifiques afin d'effectuer une coupe des données qui montre, en même temps, trois points d'intérêts présents à l'intérieur de ces données. D'une part, cette série d'expérimentations a permis de valider les trois modèles proposés dans cette thèse lors d'une collaboration entre des participants situés sur des lieux géographiquement éloignés. D'autre part, elle a permis de montrer que la manipulation collaborative à distance était plus efficace que la manipulation mono-utilisateur lorsque la tâche à réaliser était difficile (lorsque la distance séparant les trois points d'intérêts était plus importante).

Perspectives

Chacun des modèles proposés dans le cadre de cette thèse apporte de nouvelles perspectives pour la conception et l'utilisation des environnements virtuels collaboratifs. Par ailleurs, le *framework* Collaviz développé dans le cadre de cette thèse, ainsi que les expérimentations réalisées entre Rennes et Londres amènent également de nouvelles perspectives de travail pour la collaboration à distance en environnement virtuel.

Premièrement, le modèle d'adaptation dynamique de la distribution des données pourrait être adapté afin de permettre l'étude de nouveaux modes de distribution. En effet, il serait envisageable de modifier légèrement le paradigme de référents et de proxys afin d'avoir plusieurs nœuds « référent » et plusieurs nœuds « proxy » pour un même objet virtuel. Un objet pourrait ainsi avoir des référents sur certains nœuds particuliers et des proxys sur les autres (actuellement, un objet virtuel peut avoir soit un référent et des proxys, soit uniquement des référents). Cela permettrait d'obtenir des modes de distribution qui mélangeraient les caractéristiques du mode hybride et du mode centralisé. Par exemple, il serait possible d'offrir un compromis entre cohérence et réactivité spécifique à chaque utilisateur en fonction des performances de son système (vitesse de sa connexion réseau et puissance de sa machine).

Deuxièmement, le modèle d'architecture logicielle PAC-C3D permet de décomposer chaque objet virtuel en trois facettes afin de séparer les données et le comportement de l'objet, de sa distribution sur le réseau et de ses représentations dans le monde virtuel. Cependant, cette décomposition peut sembler parfois contraignante lors de la conception de nouveaux types d'objet virtuel. Il serait intéressant de proposer un meta-modèle qui permette de créer automatiquement, à partir d'une description de l'objet, les trois facettes PAC-C3D et de faire le lien entre ces facettes. Il serait ainsi possible, d'une part, de générer l'*Abstraction* à partir de la description des données et du comportement de l'objet, et

d'autre part, de définir si le *Contrôle* de l'objet dont cet objet hérite peut être réutilisé ou si un nouveau *Contrôle* spécifique doit être créé. Par contre, il semble difficile d'automatiser la création des *Présentations* car elles dépendent souvent de composants logiciels spécifiques.

Troisièmement, le modèle de CVII permet d'intégrer les espaces d'interaction réels des utilisateurs dans l'environnement virtuel en les modélisant d'une façon hiérarchisée. Il serait intéressant d'évaluer les différentes fonctionnalités d'interaction et de collaboration proposées grâce au modèle de CVII. Par exemple, il serait intéressant d'évaluer la technique qui permet de tourner facilement autour d'un objet pour l'examiner et le système de caméras pour l'exploration collaborative de données scientifiques basé sur cette technique. Par ailleurs, afin de faciliter l'intégration de nouveaux dispositifs matériels, nous pourrions proposer un formalisme standardisé pour décrire les différents espaces d'interaction et leur organisation dans le monde réel, en utilisant un langage de description basé sur XML comme X3D ou Collada. Ainsi, la configuration de l'environnement réel de chaque utilisateur pourrait être stockée dans un fichier XML.

Par ailleurs, les expérimentations de manipulation d'un plan de *clipping* réalisées entre Rennes et Londres amènent de nouvelles expérimentations possibles. En effet, il serait intéressant de faire une expérimentation similaire à la deuxième expérimentation réalisée en affichant les données scientifiques afin de déterminer si ces dernières ont un impact sur la tâche de manipulation. De plus, il serait intéressant de mener une nouvelle série d'expérimentations avec un écart variable entre les trois points à atteindre avec le plan de *clipping* afin de déterminer le seuil à partir duquel cet écart rend la tâche suffisamment difficile pour qu'il soit plus efficace d'utiliser la manipulation collaborative plutôt que la manipulation mono-utilisateur.

De façon plus générale, le *framework* Collaviz basé sur les trois modèles proposés dans cette thèse va pouvoir être utilisé dans le cadre de différentes collaborations futures. Ce *framework* a déjà été installé à l'IRISA/Inria Rennes, à l'UCL à Londres et à l'Inria Sophia-Antipolis. Dans le cadre du projet Européen Visionair, il pourra être utilisé pour de nouvelles expérimentations entre Rennes et Londres. Dans le cadre du projet VCoRE (collaboration entre l'Inria et le Frauhofer IGD), il est aussi prévu que ce *framework* soit utilisé pour mettre en œuvre un environnement virtuel collaboratif qui permette une interopérabilité entre plusieurs librairies de réalité virtuelle comme SOFA [2] ou OpenSG [3]. Dans ce cadre, un nouveau *viewer* basé sur OpenSG a été développé et intégré au *framework* Collaviz grâce à un *binding* entre Java et C++ (OpenSG étant développé en C++).

2. `www.sofa-framework.org`
3. `www.opensg.org`

Bibliographie

Agrawala M., Beers A. C., McDowall I., Fröhlich B., Bolas M. et Hanrahan P. (1997). « The two-user Responsive Workbench : support for collaboration through individual views of a shared space ». *Dans Proceedings of the conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 327–332.

Aguerreche L. (2010). *Partage d'interactions en environnements virtuels : de nouvelles techniques collaboratives basées sur un protocole de dialogue générique*. Thèse de doctorat, INSA de Rennes.

Aguerreche L., Duval T. et Lécuyer A. (2009). « Short paper : 3-Hand Manipulation of Virtual Objects ». *Dans Proceedings of the Joint Virtual Reality Conference of EGVE - ICAT - EuroVR (JVRC)*, pages 153–156.

Aguerreche L., Duval T. et Lécuyer A. (2010). « Reconfigurable tangible devices for 3D virtual object manipulation by single or multiple users ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 227–230.

Amselem D. (1995). « A Window on Shared Virtual Environments ». *Presence : Teleoperators and Virtual Environments*, 4(2):130–145.

Anthes C., Heinzlreiter P. et Volkert J. (2004). « An adaptive network architecture for close-coupled collaboration in distributed virtual environments ». *Dans Proceedings of the ACM SIGGRAPH conference on Virtual Reality Continuum and its Applications in Industry (VRCAI)*, pages 382–385.

Arsenault R. et Ware C. (2000). « Eye-hand co-ordination with force feedback ». *Dans Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 408–414.

Benford S., Bowers J., Fahlén L. E. et Greenhalgh C. (1994). « Managing mutual awareness in collaborative virtual environments ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 223–236.

Benford S., Bowers J., Fahlén L. E., Greenhalgh C. et Snowdon D. (1995). « User embodiment in collaborative virtual environments ». *Dans Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 242–249.

Bierbaum A., Just C., Hartling P., Meinert K., Baker A. et Cruz-Neira C. (2001). « VR Juggler : A Virtual Platform for Virtual Reality Application Development ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 89–96.

Blanchard C., Burgess S., Harvill Y., Lanier J., Lasko A., Oberman M. et Teitel M. (1990). « Reality built for two : a virtual reality tool ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 35–36.

Bowers J., Pycock J. et O'Brien J. (1996). « Talk and embodiment in collaborative virtual environments ». *Dans Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 58–65.

Bowman D. A., Davis E. T., Hodges L. F. et Badre A. N. (1999). « Maintaining Spatial Orientation during Travel in an Immersive Virtual Environment ». *Presence : Teleoperators and Virtual Environments*, 8(6):618–631.

Bowman D. A. et Hodges L. F. (1997). « An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 35–38.

Bowman D. A., Kruijff E., LaViola J. J. et Poupyrev I. (2004). *3D User Interfaces : Theory and Practice*. Addison Wesley.

Brooks, Jr. F. P., Airey J., Alspaugh J., Bell A., Brown R., Hill C., Nimscheck U., Rheingans P., Rohlf J., Smith D., Turner D., Varshney A., Wang Y., Weber H. et Yuan X. (1992). « Six Generations of Building Walkthrough : Final Technical Report to the National Science Foundation ». Rapport technique TR92-026, University of North Carolina at Chapel Hill.

Butterworth J., Davidson A., Hench S. et Olano M. T. (1992). « 3DM : A three dimensional modeler using a head-mounted display ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 135–138.

Calvary G., Coutaz J. et Nigay L. (1997). « From Single-User Architectural Design to PAC* : a Generic Software Architecture Model for CSCW ». *Dans Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 242–249.

Calvin J., Dickens A., Gaines B., Metzger P., Miller D. et Owen D. (1993). « The SIMNET virtual world architecture ». *Dans Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 450–455.

Cirio G., Marchal M., Regia-Corte T. et Lécuyer A. (2009). « The Magic Barrier Tape : A Novel Metaphor for Infinite Navigation in Virtual Worlds with a Restricted Walking Workspace ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 155–162.

Cirio G., Vangorp P., Chapoulie E., Marchal M., Lecuyer A. et Drettakis G. (2012). « Walking in a Cube : Novel Metaphors for Safely Navigating Large Virtual Environments in Restricted Real Workspaces ». *IEEE Transactions on Visualization and Computer Graphics*, 18:546–554.

Coutaz J. (1987). « PAC : An Object Oriented Model for Implementing User Interfaces ». *ACM SIGCHI Bulletin*, 19:37–41.

Cruz-Neira C., Sandin D. J., DeFanti T. A., Kenyon R. V. et Hart J. C. (1992). « The CAVE : audio visual experience automatic virtual environment ». *Communications of the ACM*, 35:64–72.

Darken R. P., Cockayne W. R. et Carmein D. (1997). « The omni-directional treadmill : a locomotion device for virtual worlds ». *Dans Proceedings of the ACM symposium on User Interface Software and Technology (UIST)*, pages 213–221.

Delaney D., Ward T. et McLoone S. (2006a). « On consistency and network latency in distributed interactive applications : A survey – part I ». *Presence : Teleoperators and Virtual Environments*, 15(2):218–234.

Delaney D., Ward T. et McLoone S. (2006b). « On Consistency and Network Latency in Distributed Interactive Applications : A Survey – Part II ». *Presence : Teleoperators and Virtual Environments*, 15(4):465–482.

Dewan P. (1999). « Architectures for Collaborative Applications ». *Trends in Software, special issue on Collaborative Systems*, pages 169–193.

Dit Picard S. L., Degrande S., Gransart C. et Chaillou C. (2002). « VRML data sharing in the spin-3D CVE ». *Dans Proceedings of the ACM conference on 3D Web technology (Web3D)*.

Dodds T. J. et Ruddle R. A. (2008). « Mobile Group Dynamics in Large-Scale Collaborative Virtual Environments ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 59–66.

Dominjon L., Lécuyer A., Burkhardt J.-M., Andrade-Barroso G. et Richir S. (2005). « The "Bubble" Technique : Interacting with Large Virtual Environments Using Haptic Devices with Limited Workspace ». *Dans Proceedings of the IEEE World Haptics Conference (WHC)*, pages 639–640.

Dupont F., Duval T., Fleury C., Forest J., Gouranton V., Lando P., Laurent T., Lavoué G. et Schmutz A. (2010). « Demo - Collaborative Scientific Visualization : The COLLAVIZ Framework ». *Dans Proceedings of the Joint Virtual Reality Conference of EuroVR - EGVE - VEC (JVRC)*.

Dupros F., Martin F. D., Foerster E., Komatitsch D. et Roman J. (2010). « High-performance finite-element simulations of seismic wave propagation in three-dimensional nonlinear inelastic geological media ». *Parallel Computing*, 36(5–6):308–325.

Duval T. et Chauffaut A. (2006). « La cabine virtuelle d'immersion (CVI) : un mode de transport des outils d'interaction dans les univers 3D ». *Dans Proceedings of Conférence Francophone sur l'Interaction Homme-Machine (IHM)*, pages 167–170.

Duval T. et Fenals A. (2002). « Faciliter la perception de l'interaction lors de manipulations coopératives simultanées en environnements virtuels 3d ». *Dans Annex of the Proceedings of the conference of the Association Francophone d'Interaction Homme-Machine (IHM)*, pages 29–32.

Duval T. et Fleury C. (2009). « An asymmetric 2D Pointer/3D Ray for 3D interaction within collaborative virtual environments ». *Dans Proceedings of the ACM conference on 3D Web technology (Web3D)*, pages 33–41.

Duval T. et Fleury C. (2011a). « Demo - Collaboration between Networked Heterogeneous 3D Viewers through a PAC-C3D Modeling of the Shared Virtual Environment ». *Dans Proceedings of the International Conference on Artificial reality and Telexistence (ICAT)*.

Duval T. et Fleury C. (2011b). « PAC-C3D : A New Software Architectural Model for Designing 3D Collaborative Virtual Environments ». *Dans Proceedings of the International Conference on Artificial reality and Telexistence (ICAT)*.

Duval T., Fleury C., Nouailhas B. et Aguerreche L. (2008). « Demo - Collaborative exploration of 3D scientific data ». *Dans Proceedings of the ACM symposium on Virtual reality Software and Technology (VRST)*, pages 303–304.

Duval T. et Margery D. (2000a). « Building objects and interactors for collaborative interactions with GASP ». *Dans Proceedings of the conference on Collaborative Virtual Environments (CVE)*, pages 129–138.

Duval T. et Margery D. (2000b). « Using GASP for Collaborative Interactions within 3D Virtual Worlds ». *Dans Proceedings of the conference on Virtual Worlds (VW)*, pages 65–76.

Duval T. et Tarby J.-C. (2006). « Améliorer la conception des applications interactives par l'utilisation conjointe du modèle PAC et des patrons de conception ». *Dans Proceedings of the conference of the Association Francophone d'Interaction Homme-Machine (IHM)*, pages 225–232.

Duval T. et Zammar C. e. (2006). « A migration mechanism to manage network troubles while interacting within collaborative virtual environments ». *Dans Proceedings of the ACM conference on Virtual Reality Continuum and Its Applications (VRCIA)*, pages 417–420.

El Merhebi S., Torguet P., Jessel J.-P. et Hoelt J.-C. (2007). « ASSET : A Scalable Multicast Multi-Server Based Distributed Virtual Environments System ». *Dans Proceedings of the conference on Cyberworlds (CW)*, pages 179 –186.

El Zammar C. (2005). *Interactions coopératives 3D distantes en environnements virtuels : gestion des problèmes réseau.* Thèse de doctorat, INSA de Rennes.

Elmqvist N., Tudoreanu M. E. et Tsigas P. (2007). « Tour generation for exploration of 3D virtual environments ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 207–210.

Fleury C., Chauffaut A., Duval T., Gouranton V. et Arnaldi B. (2010a). « A Generic Model for Embedding Users' Physical Workspaces into Multi-Scale Collaborative Virtual Environments ». *Dans Proceedings of the International Conference on Artificial reality and Telexistence (ICAT)*.

Fleury C., Duval T., Gouranton V. et Arnaldi B. (2010b). « Architectures and Mechanisms to efficiently Maintain Consistency in Collaborative Virtual Environments ». *Dans Proceedings of the IEEE VR workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 87–94.

Fleury C., Duval T., Gouranton V. et Arnaldi B. (2010c). « A New Adaptive Data Distribution Model for Consistency Maintenance in Collaborative Virtual Environments ». *Dans Proceedings of the Joint Virtual Reality Conference of EuroVR - EGVE - VEC (JVRC)*, pages 29–36.

Fleury C., Duval T., Gouranton V. et Steed A. (2012). « Evaluation of Remote Collaborative Manipulation for Scientific Data Analysis ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*. 8 pages.

Fraser M., Benford S., Hindmarsh J. et Heath C. (1999). « Supporting awareness and interaction through collaborative virtual interfaces ». *Dans Proceedings of the ACM symposium on User Interface Software and Technology (UIST)*, pages 27–36.

Frécon E. et Stenius M. (1998). « DIVE : A Scaleable Network Architecture for Distributed Virtual Environments ». *Distributed Systems Engineering*, 5:91–100.

Fuchs P., Moreau G., Berthoz A. et Vercher J.-L. (2006a). *Le traité de la réalité virtuelle*, volume 1. Presses de l'École des Mines de Paris, 3ième édition.

Fuchs P., Moreau G., Burkhardt J.-M. et Coquillart S. (2006b). *Le traité de la réalité virtuelle*, volume 2. Presses de l'École des Mines de Paris, 3ième édition.

Funkhouser T. A. (1995). « RING : a client-server system for multi-user virtual environments ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 85–93.

Galyean T. A. (1995). « Guided navigation of virtual environments ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 103–105.

Goldberg A. (1990). « Information models, views, and controllers ». *Dr. Dobb's Journal*, 15:54–61.

Hachet M., Decle F., Knodel S. et Guitton P. (2008). « Navidget for Easy 3D Camera Positioning from 2D Inputs ». *Dans Proceedings of the IEEE symposium on 3D User Interfaces (3DUI)*, pages 83–89.

Hagsand O. (1996). « Interactive Multiuser VEs in the DIVE System ». *IEEE MultiMedia*, 3(1):30–39.

Hand C. (1997). « A Survey of 3D Interaction Techniques ». *Computer Graphics Forum*, 16(5):269–281.

Hill R. D. (1992). « The abstraction-link-view paradigm : using constraints to connect user interfaces to applications ». *Dans Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 335–342.

Hindmarsh J., Fraser M., Heath C., Benford S. et Greenhalgh C. (1998). « Fragmented interaction : establishing mutual orientation in virtual environments ». *Dans Proceedings of the ACM conference on Computer Supported Cooperative Work (CSCW)*, pages 217–226.

Hubbold R., Cook J., Keates M., Gibson S., Howard T., Murta A., West A. et Pettifer S. (1999). « GNU/MAVERIK : a micro-kernel for large-scale virtual environments ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 66–73.

Jacoby R. H., Ferneau M. et Humphries J. (1994). « Gestural interaction in a virtual environment ». *Dans Proceedings of the conference on Stereoscopic Displays and Virtual Reality Systems*, pages 355–364.

Joslin C., Di Giacomo T. et Magnenat-Thalmann N. (2004). « Collaborative virtual environments : from birth to standardization ». *IEEE Communications Magazine*, 42(4):28–33.

Jourdain S., Forest J., Mouton C., Nouailhas B., Moniot G., Kolb F., Chabridon S., Simatic M., Abid Z. et Mallet L. (2008). « ShareX3D, a scientific

collaborative 3D viewer over HTTP ». *Dans Proceedings of the ACM conference on 3D Web technology (Web3D)*, pages 35–41.

Kallmann M. et Thalmann D. (1999). « Direct 3D interaction with smart objects ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 124–130.

Kapralos B. (2003). *Auditory Perception and Virtual Environments*. Thèse de doctorat, Department of Computer Science, York University.

Kelso J., Arsenault L. E., Kriz R. D. et Satterfield S. G. (2002). « DIVERSE : A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 183–190.

Kessler G. D., Bowman D. A. et Hodges L. F. (2000). « The Simple Virtual Environment Library : An Extensible Framework for Building VE Applications ». *Presence : Teleoperators and Virtual Environments*, 9(2):187–208.

Kleiner M. J., Dalenbck B.-I. et Svensson P. (1993). « Auralization - An Overview ». *Journal of the Audio Engineering Society*, 41(11):861–875.

Kopper R., Ni T., Bowman D. A. et Pinho M. (2006). « Design and Evaluation of Navigation Techniques for Multiscale Virtual Environments ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, page 24.

Krüeger W. et Fröehlich B. (1994). « The Responsive Workbench ». *IEEE Computer Graphics and Applications*, 14(3):12–15.

Laurillau Y. et Nigay L. (2002). « Clover architecture for groupware ». *Dans Proceedings of the ACM conference on Computer Supported Cooperative Work (CSCW)*, pages 236–245.

Lee D., Lim M., Han S. et Lee K. (2007). « ATLAS : A Scalable Network Framework for Distributed Virtual Environments ». *Presence : Teleoperators and Virtual Environments*, 16(2):125–156.

Leigh J., Johnson A. E., Vasilakis C. A. et DeFanti T. A. (1996). « Multi-perspective collaborative design in persistent networked virtual environments ». *Dans Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 253–260.

Macedonia M. et Zyda M. (1997). « A taxonomy for networked virtual environments ». *IEEE Multimedia*, 4(1):48–56.

Macedonia M. R., Zyda M. J., Pratt D. R., Barham P. T. et Zeswitz S. (1994). « NPSNET : A network software architecture for large scale virtual environments ». *Presence : Teleoperators and Virtual Environments*, 3(4):265–287.

Mackinlay J. D., Card S. K. et Robertson G. G. (1990). « Rapid controlled movement through a virtual 3D workspace ». *Dans Proceedings of the conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 171–176.

Mapes D. P. et Moshell J. M. (1995). « A Two Handed Interface for Object Manipulation in Virtual Environments ». *Presence : Teleoperators and Virtual Environments*, 4(4): 403–416.

Marbach J. (2009). « Image Blending and View Clustering for Multi-Viewer Immersive Projection Environments ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 51–54.

Margery D., Arnaldi B., Chauffaut A., Donikian S. et Duval T. (2002). « OpenMASK : Multi-Threaded or Modular Animation and Simulation Kernel or Kit : a General Introduction ». *Dans Proceedings of the Virtual Reality International Conference (VRIC)*, pages 101–110.

Matsukura H., Nihei T. et Ishida H. (2011). « Multi-sensorial field display : Presenting spatial distribution of airflow and odor ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 119–122.

Mine M. R. (1995a). « ISAAC : A Virtual Environment Tool for the Interactive Construction of Virtual Worlds ». Rapport technique TR95-020, University of North Carolina at Chapel Hill.

Mine M. R. (1995b). « Virtual Environment Interaction Techniques ». Rapport technique TR95-018, University of North Carolina at Chapel Hill.

Mine M. R., Brooks Jr. F. P. et Sequin C. H. (1997). « Moving objects in space : exploiting proprioception in virtual-environment interaction ». *Dans Proceedings of the conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 19–26.

Mulder J. D. et Boschker B. R. (2004). « A Modular System for Collaborative Desktop VR/AR with a Shared Workspace ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, volume 0, page 75.

Narumi T., Kajinami T., Nishizaka S., Tanikawa T. et Hirose M. (2011). « Pseudo-gustatory display system based on cross-modal integration of vision, olfaction and gustation ». *Dans Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 127 –130.

Nguyen T. T. H., Fleury C. et Duval T. (2012). « 3DUI contest - Collaborative Exploration in a Multi-Scale Shared Virtual Environment ». *Dans Proceedings of IEEE symposium on 3D User Interfaces (3DUI)*.

Nigay L. et Coutaz J. (1991). « Building User Interfaces : Organizing Software Agents ». *Dans Proceedings of ESPRIT Conference*, pages 709–717.

Noma H. et Miyasato T. (1997). « Cooperative Object Manipulation in Virtual Space using Virtual Physics ». *Dans Proceedings of the ASME Dynamic Systems and Control Conference (DSCC)*, volume 61, pages 101–106.

Ohlenburg J., Broll W. et Lindt I. (2007). « DEVAL - A Device Abstraction Layer for VR/AR ». *Dans Universal Access in Human Computer Interaction*, volume 4554 de *Lecture Notes in Computer Science*, pages 497–506. Springer.

Ortega M. et Coquillart S. (2005). « Prop-based haptic interaction with co-location and immersion : An automotive application ». *Dans Proceedings of the IEEE workshop on Haptic Audio Visual Environnements and their applications (HAVE)*, pages 23–28.

Pausch R., Burnette T., Brockway D. et Weiblen M. E. (1995). « Navigation and locomotion in virtual worlds via flight into hand-held miniatures ». *Dans Proceedings*

of the conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 399–400.

Pierce J. S., Forsberg A. S., Conway M. J., Hong S., Zeleznik R. C. et Mine M. R. (1997). « Image plane interaction techniques in 3D immersive environments ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 39–43.

Pinho M. S., Bowman D. A. et Freitas C. M. (2002). « Cooperative object manipulation in immersive virtual environments : framework and techniques ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 171–178.

Poupyrev I., Billinghurst M., Weghorst S. et Ichikawa T. (1996). « The go-go interaction technique : non-linear mapping for direct manipulation in VR ». *Dans Proceedings of the ACM symposium on User Interface Software and Technology (UIST)*, pages 79–80.

Razzaque S. (2005). *Redirected Walking*. Thèse de doctorat, University of North Carolina at Chapel Hill.

Reenskaug T. (1979). « The original MVC reports ». `http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf`.

Riege K., Holtkamper T., Wesche G. et Frohlich B. (2006). « The Bent Pick Ray : An Extended Pointing Technique for Multi-User Interaction ». *Dans Proceedings of IEEE symposium on 3D User Interfaces (3DUI)*, pages 62–65.

Roberts D. et Sharkey P. (1997). « Maximising concurrency and scalability in a consistent, causal, distributed virtual reality system, whilst minimising the effect of network delays ». *Dans Proceedings of the 6th IEEE workshops on Enabling Technologies on Infrastructure for Collaborative Enterprises (WET-ICE)*, pages 161–166.

Robinett W. et Holloway R. (1992). « Implementation of Flying, Scaling and Grabbing in Virtual Worlds ». *Dans Proceedings of the symposium on Interactive 3D Graphics (I3D)*, pages 189–192.

Ruddle R. A., Howes A., Payne S. J. et Jones D. M. (2000). « The effects of hyperlinks on navigation in virtual environments ». *International Journal of Human-Computer Studies*, 53(4):551–581.

Salber D. (1995). *De l'interaction individuelle aux systèmes multi-utilisateurs. L'exemple de la Communication Homme-Homme médiatisée*. Thèse de doctorat, Université Joseph Fourier.

Sato M. (2002). « Development of string-based force display : SPIDAR ». *Proceedings of the conference on Virtual Systems and Multi Media (VSMM)*, pages 1034–1039.

Schild J., Holtkämper T. et Bogen M. (2009). « The 3D Sketch Slice : Precise 3D Volume Annotations in Virtual Environments ». *Dans Proceedings of the Joint Virtual Reality Conference of EGVE - ICAT - EuroVR (JVRC)*, pages 65–72.

Shaw C. et Green M. (1993). « The MR Toolkit Peers Package and Experiment ». *Dans Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 463–469.

Shaw C., Green M., Liang J. et Sun Y. (1993). « Decoupled simulation in virtual reality with the MR toolkit ». *ACM Transactions on Informations Systems*, 11:287–317.

Simon A. (2005). « First-person experience and usability of co-located interaction in a projection-based virtual environment ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 23–30.

Singh G., Serra L., Png W., Wong A. et Ng H. (1995). « BrickNet : sharing object behaviors on the Net ». *Dans Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 19–25.

Slater M. et Usoh M. (1993). « Presence in Immersive Virtual Environments ». *Dans Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 90–96.

Slater M., Usoh M. et Steed A. (1995). « Taking steps : the influence of a walking technique on presence in virtual reality ». *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2(3):201–219.

Snowdon D. N., Greenhalgh C. M. et Benford S. D. (1995). « What You See Is Not What I See : Subjectivity in Virtual Environments ». *Dans Proceedings of the Framework for Immersive Virtual Environments conference (FIVE)*, pages 53–69.

Sowizral H. A. et Deering M. F. (1999). « The Java 3D API and Virtual Reality ». *IEEE Computer Graphics and Applications*, 19(3):12–15.

Steed A. (2008). « Some Useful Abstractions for Re-Usable Virtual Environment Platforms ». *Dans Proceedings of the IEEE VR workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*.

Stoakley R., Conway M. J. et Pausch R. (1995). « Virtual reality on a WIM : interactive worlds in miniature ». *Dans Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 265–272.

Suma E. A., Lipps Z., Finkelstein S., Krum D. M. et Bolas M. (2012). « Impossible Spaces : Maximizing Natural Walking in Virtual Environments with Self-Overlapping Architecture ». *IEEE Transactions on Visualization and Computer Graphics*, 18:555–564.

Sutherland I. E. (1968). « A head-mounted three dimensional display ». *Dans Proceedings of the Fall Joint Computer Conference (Fall)*, pages 757–764.

Taylor, II R. M., Hudson T. C., Seeger A., Weber H., Juliano J. et Helser A. T. (2001). « VRPN : a device-independent, network-transparent VR peripheral system ». *Dans Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 55–61.

Thomas S. (2005). « Evaluation de paradigmes de navigation et d'interaction en environnements virtuels coopératifs ». Mémoire de Master, Université de Rennes 1.

Torguet P. et Caubet R. (1995). « VIPER (VIRtuality Programming EnviRonment) : a virtual reality application design platform ». *Dans Proceedings of the Eurographics workshop on Virtual Environments (EGVE)*.

UIMS (1992). « A metamodel for the runtime architecture of an interactive system : the UIMS tool developers workshop ». *ACM SIGCHI Bulletin*, 24(1):32–37.

Ujike H. (2009). « Estimation of Visually Induced Motion Sickness from Velocity Component of Moving Image ». *Dans Virtual and Mixed Reality*, volume 5622 de *Lecture Notes in Computer Science*, pages 136–142. Springer.

Ware C. et Osborne S. (1990). « Exploration and virtual camera control in virtual three dimensional environments ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 175–183.

Waters R., Anderson D., Barrus J., Brogan D., Mckeown S., Nitta T., Sterns I. et Yerazunis W. (1997). « Diamond Park and Spline : A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability ». *Presence : Teleoperators and Virtual Environments*, 6(4):461–480.

Williams B., Narasimham G., Rump B., McNamara T. P., Carr T. H., Rieser J. et Bodenheimer B. (2007). « Exploring Large Virtual Environments with an HMD When Physical Space is Limited ». *Dans Proceedings of the symposium on Applied Perception in Graphics and Visualization (APGV)*, pages 41–48.

Yang H. et Olson G. M. (2002). « Exploring collaborative navigation : the effect of perspectives on group performance ». *Dans Proceedings of the conference on Collaborative Virtual Environments (CVE)*, pages 135–142.

Yoshida M., Tijerino Y. A., Abe S. et Kishino F. (1995). « A virtual space teleconferencing system that supports intuitive interaction for creative and cooperative work ». *Dans Proceedings of the symposium on Interactive 3D graphics (I3D)*, pages 115–122.

Zhai S., Buxton W. et Milgram P. (1994). « The “Silk-Cursor" : investigating transparency for 3D target acquisition ». *Dans Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 459–464.

Zhang X. et Furnas G. W. (2002). « Social interactions in multiscale CVEs ». *Dans Proceedings of the conference on Collaborative Virtual Environments (CVE)*, pages 31–38.

Zhang X. et Furnas G. W. (2005). « mCVEs : Using Cross-Scale Collaboration to Support User Interaction with Multiscale Structures ». *Presence : Teleoperators and Virtual Environments*, 14(1):31–46.

Table des figures

Liste des tableaux

Résumé

La réalité virtuelle collaborative permet à des utilisateurs géographiquement éloignés de se rencontrer dans un environnement virtuel partagé afin d'effectuer ensemble des tâches allant d'une simple observation jusqu'à de la co-manipulation d'objets virtuels. Cependant, les contraintes techniques, l'utilisation de dispositifs matériels différents pour chaque utilisateur, ainsi que le fait de se trouver dans un monde virtuel ajoutent de la complexité et rendent la compréhension entre utilisateurs plus difficile. Nos contributions visent à améliorer la collaboration entre utilisateurs en proposant de nouveaux modèles de conception pour des environnements virtuels collaboratifs distants tant au niveau de l'architecture distribuée qu'au niveau des métaphores de collaboration.

Pour une bonne compréhension mutuelle, il est essentiel que tous les utilisateurs perçoivent le même état de l'environnement virtuel au même moment. Nous proposons une distribution réseau des données relatives à l'environnement virtuel qui peut être adaptée de façon dynamique et indépendante pour chaque objet virtuel. Cette distribution permet d'obtenir le compromis adéquat entre cohérence de l'environnement virtuel et réactivité lors des interactions en fonction des contraintes réseaux, des caractéristiques de l'application, du rôle de chaque objet dans le monde virtuel et des tâches que les utilisateurs sont en train d'accomplir. Pour modéliser un environnement virtuel collaboratif indépendamment de sa distribution réseau et de sa restitution aux utilisateurs (associée à leurs dispositifs matériels), nous proposons une architecture logicielle qui sépare les données de l'environnement virtuel, la partie réseau et les systèmes de rendu graphique, sonore, etc. Cette architecture repose sur une extension du modèle PAC *(Présentation, Abstraction, Contrôle)* pour les applications collaboratives.

Pour permettre une collaboration efficace, il faut être capable d'intégrer des utilisateurs avec des dispositifs matériels différents en offrant à chacun des métaphores d'interaction et de collaboration adaptées. De plus, ces utilisateurs doivent être en mesure de comprendre ce que les autres perçoivent du monde virtuel et ce qu'ils peuvent y faire. Nous avons défini un modèle, appelé *Cabine Virtuelle d'Interaction Immersive*, qui intègre chaque utilisateur dans l'environnement virtuel collaboratif en tenant compte des espaces d'interaction liés à ses dispositifs matériels. Ce modèle est basé sur une structure hiérarchisant les différents espaces d'interaction (visuel, auditif, haptique, etc.) et sur des opérateurs permettant de la modifier et de l'intégrer dans l'environnement virtuel. Le modèle peut ainsi proposer de nouvelles fonctionnalités adaptées aux dispositifs matériels pour naviguer, interagir et collaborer. Il offre également une structure pour représenter les capacités d'interaction de chaque utilisateur dans l'environnement virtuel.

Ces travaux s'inscrivent dans le cadre du projet ANR Collaviz qui porte sur la collaboration à distance d'experts pour l'étude de données scientifiques. Dans ce contexte, nous avons validé nos contributions par des expérimentations de co-manipulation distante entre Londres et Rennes.

Abstract

Virtual reality enables several users located in remote geographical locations to meet themselves in a shared virtual environment to perform a collaborative work such as a simple observation or a co-manipulation of some virtual objects. However, the technical constraints, the use of different material devices for each user and the fact of being in a virtual world increase the complexity and consequently the misunderstanding between users. This PhD work aims to improve the collaboration between users : we propose some new models for designing collaborative virtual environments to deal with the distributed architecture, but also to integrate some new metaphors for the collaboration.

To improve the mutual understanding, we have first to ensure that all users have the same state of the virtual environment at the same time. We propose a new way to distribute the data of the virtual environment. This data distribution can be dynamically and independently adapted on the network for each virtual object. Thus, a good trade-off between consistency of the virtual environment and responsiveness during interaction can be reached according to the technical constraints, the features of the application, the functions that objects fulfil in the virtual world, and the tasks that users are performing. In order to design a collaborative virtual environment independently from the data distribution and from how the data are displayed to the users (on several material devices), we propose a software architecture which clearly separates the data of the virtual environment from the network layer and from the systems for the graphical rendering, the sound rendering, etc. This software architecture is an extension of the PAC model *(Presentation, Abstraction, Control)* for collaboration applications.

For an efficient collaboration, we have to integrate with adapted metaphors for interaction and collaboration several users using different material devices. Moreover, each user must be able to understand how the others perceive the virtual world and what they can do in this virtual world. We describe a new model, called *Immersive Interactive Virtual Cabin*, that makes possible to integrate the users in the virtual environment by taking account of their interaction workspaces. This model is based on a hierarchical structure to organize the different interaction workspaces (visual, sound, haptic, etc.) and on a set of operators to manage and integrate this structure in the virtual environment. In this way, the model can offer new functionalities adapted to the material devices for navigation, interaction and collaboration. It also proposes a structure for representing the interaction capabilities of each user in the virtual environment.

This PhD work is a part of the ANR Collaviz project about collaboration for enabling remote experts to examine together some scientific data. In this context, we have run some experimentations to validate the proposed solutions during remote co-manipulation between London and Rennes.

INSA RENNES

Institut National des Sciences Appliquées de Rennes

20, Avenue des Buttes de Coësmes . CS 14315 . F-35043 Rennes Cedex

Tel : 02 23 23 82 00 – Fax : 02 23 23 83 96