Université de Strasbourg

Thèse présentée pour obtenir le grade de docteur
Université de Strasbourg

❧

Département d'Informatique, Laboratoire ICube
École Doctorale « Mathématiques, Sciences de l'Information et de l'Ingénieur » (MS2I)

Discipline : Informatique

Proposition d'un Protocole Web pour la Collaboration Multi-Support en Environnement 3D

UMI3D

Par : **Julien CASARIN**

Sous la direction de Dominique Bechmann, professeure des Universités

Membres du jury:

- **Rapporteur :** Indira Thouvenin, Prof. Contractuelle, UTC Compiègne
- **Rapporteur :** Samir Otmane, Prof. des Universités, Évry
- **Examinateur :** Michel Beaudouin-Lafon, Prof. des Universités, Paris-Sud
- **Examinateur :** Antonio Capobianco, Maître de Conférences, Strasbourg
- **Examinateur :** Jean-François Gaudy, Executive Vice President, Chief Innovation & Digital Officer , GFI (encadrant CIFRE)

Date de soutenance : 30 Septembre 2019

Résumé

Les récentes avancées technologiques dans les domaines de la Réalité Virtuelle (RV) et de la Réalité Augmentée (RA) ont donné naissance à un ensemble varié de supports permettant, de visualiser des médias 3D, et d'interagir avec eux. Cette grande diversité d'outils, permettant de manipuler le contenu 3D, apporte des perspectives nouvelles pour le support du travail collaboratif par la RA et la RV. En effet, le travail collaboratif implique couramment des acteurs ayant des métiers et des rôles différents. Leurs besoins en termes de visualisation, et surtout d'interaction avec le contenu 3D sont donc différents. S'il existe aujourd'hui des outils permettant la conception d'expériences collaboratives multi-supports, ceux-ci demandent aux concepteurs de médias 3D une expertise importante en développement d'applications de RV/RA. Ils doivent d'une part connaître les bonnes pratiques de conception propres à chaque support. Il faut d'autre part que les concepteurs aient une certaine expérience en développement d'interfaces plastiques. Si ce n'est pas le cas, une application différente est nécessaire pour chaque support.

Le point de vue défendu dans cette thèse est qu'une nouvelle architecture des applications collaboratives en RV/RA est nécessaire de sorte à permettre leur conception sans connaissance par les développeurs des spécificités des supports.

Après avoir présenté les outils existants pour créer des applications multi-supports en RV/RA, nous proposons une nouvelle architecture des applications 3D. Cette architecture sépare le média 3D du dispositif utilisateur. Pour permettre la communication entre le média et les différents supports, nous introduisons le protocole web UMI3D qui permet au dispositif de se connecter, grâce à un navigateur, à tout média 3D utilisant ce protocole. Le navigateur permet d'une part, le chargement du contenu 3D du média, et génère d'autre part dynamiquement une interface utilisateur ayant une ergonomie adaptée au support. Cette interface utilisateur permet la réalisation par l'utilisateur de contrats d'interaction établis par le média grâce au protocole. Le principal apport de la méthode de conception des médias 3D proposée est de ne pas demander au concepteur du média de compétences techniques spécifiques aux dispositifs de RV/RA. Par ailleurs, dans le cas où les navigateurs UMI3D sont existants pour les dispositifs utilisés, la conception d'un média 3D multi-support demande le même effort de développement que la conception d'une application 3D mono-support.

Abstract

Recent technological advances in the fields of Virtual Reality (VR) and Augmented Reality (RA) provide a new infrastructure for the visualization of 3D media, and to interact with them. This diversity of tools, allowing to manipulate 3D content, brings new perspectives for the support of cooperative work by AR and VR. Indeed, collaborative work involves actors with different roles and trades. Their needs in terms of visualization, and especially of interaction with 3D content are therefore different. If there are nowadays tools allowing the design of collaborative experiences for multiple devices, these tools require designers to have significant expertise in developing AR/VR applications. First, they must be aware of the design best-practices for each device. Second, the designers must have some experience in developing plastic interfaces. If it is not the case, designing a different application is needed for each device.

The point of view we defend in this thesis is that a new architecture is needed for collaborative AR/VR applications to allow their conception without to enquire the developers to have a knowledge of each device specificities.

After presenting the existing tools for creating AR/VR applications for multiple devices, we propose a new architecture for 3D applications. This architecture separates the 3D media from the user device. To allow communication between the media and the different devices, we introduce the UMI3D web protocol which allows the device to connect to any 3D media using a UMI3D browser. The browser allows the loading of 3D content from the media, and dynamically generates an adapted user interface for a given device. This user interface allows the user to draw up interaction contracts established by the media using the protocol. The main contribution of the proposed 3D media architecture is to avoid asking the media designer for technical skills about specific AR/VR devices. Furthermore, in the case where UMI3D browsers are existing for the devices used, the design of a multi-devices 3D media requires the same development effort as the design of a single-device 3D application.

Remerciements

Je tiens pour commencer à remercier le Groupe Gfi, et plus particulièrement son directeur de l'Innovation et du Digital Monsieur Jean-François Gaudy, pour m'avoir fait confiance et avoir financé cette thèse. Je voudrais par ailleurs remercier l'ensemble des membres de mon équipe à la direction de l'Innovation de Gfi. Ils/elles réalisent depuis maintenant plus d'un an un travail remarquable pour poursuivre et valoriser les travaux présentés dans ce mémoire.

Ces travaux ont par ailleurs été réalisés avec l'aide de l'équipe Informatique Géométrique et Graphique (IGG) du laboratoire ICube. Je souhaite donc remercier tous les doctorants et membres de l'équipe qui ont su m'apporter leurs conseils, leur aide et leur soutien durant ces trois années.

J'adresse également mes remerciements à ma directrice de thèse, Madame Dominique Bechmann, Professeure à l'Université de Strasbourg, qui a toujours su me canaliser et m'orienter dans la bonne direction.

Je souhaite par ailleurs remercier l'ensemble des membres de mon jury de thèse. Leurs questions et leurs conseils ont largement contribué à améliorer ce manuscrit.

Je remercie enfin les membres de ma famille qui m'ont toujours soutenu, ainsi que celle qui m'accompagne, m'encourage (et me supporte) chaque jour depuis le début de mes travaux.

Introduction

Contexte des travaux

Les récentes avancées technologiques dans les domaines de la Réalité Virtuelle (notée RV par la suite) et de la Réalité Augmentée (notée RA) ont donné naissance à un ensemble varié de supports permettant, de visualiser des médias 3D, et d'interagir avec eux. Cette grande diversité d'outils, permettant de manipuler le contenu 3D, apporte des perspectives nouvelles pour le support du travail collaboratif par la RA et la RV. En effet, le travail collaboratif implique couramment des acteurs ayant des métiers et des rôles différents. Leurs besoins en termes de visualisation, et surtout d'interaction avec le contenu 3D sont donc différents. C'est pourquoi nous pourrions nous attendre à trouver de nombreuses applications existantes, permettant la collaboration multi-support en milieu professionnel. Si quelques applications de ce type existent en effet depuis plusieurs années, elles sont bien moins développées que ce que l'on pourrait supposer au vu des cas d'usages potentiels. De plus, la grande majorité des applications disponibles sur plusieurs supports ne propose pas le multi-support pour tirer parti des complémentarités entre supports, mais plutôt pour être accessibles à un plus grand nombre d'utilisateurs. Le parti pris est donc, à l'inverse, de proposer l'expérience la plus similaire possible sur les différents supports. Notre point de vue est que, malgré les cas d'usages potentiels d'une collaboration multi-support, les outils existant pour la conception de telles expériences collaboratives restent trop contraignants pour être largement adoptés. D'une part, parce que la plupart des outils et standards de développement multi-support ne permettent que peu de tirer parti des complémentarités des supports, ou demandent une forte expertise de la part des concepteurs de médias 3D. D'autre part, l'approche plus classique consistant à développer une application distincte optimisée pour chaque support, implique des temps et des coûts de développement très importants. Bien qu'optimale, cette approche n'est donc utilisée que dans de rares cas, suffisamment critiques pour justifier des développements longs et contraignants.

Objectifs des travaux

L'objectif de cette thèse est de rendre plus accessible la conception de systèmes de Réalité Augmentée (RA) et/ou de Réalité Virtuelle (RV) supportant le travail collaboratif. Comme présenté dans la section précédente (Contexte des travaux), ceci implique idéalement l'utilisation, par chaque collaborateur, du support (i.e. du dispositif de RA ou de RV) le plus adapté à son rôle dans le processus collaboratif, c'est à dire aux besoins

en termes de visualisation et d'interaction qui découlent des tâches lui étant attribuées. L'ergonomie de l'application doit par ailleurs s'adapter aux différents supports utilisés de sorte à exploiter au maximum leurs spécificités.

Si comme nous le rappellerons dans le Chapitre 1, il existe aujourd'hui des outils permettant la conception de telles expériences, ceux-ci demandent aux concepteurs de médias 3D une expertise importante en développement d'applications de RA/RV. Ils doivent d'une part connaître les bonnes pratiques de conception (ergonomie, performances requises) propres à chaque support. Il faut d'autre part que les concepteurs aient une certaine expérience en développement d'interfaces dont l'ergonomie et la disposition s'adaptent automatiquement au support (voir section 1.3.4 Plasticité des interfaces utilisateur). Si ce n'est pas le cas, une application différente est nécessaire pour chaque support.

Le point de vue défendu dans cette thèse est qu'une nouvelle architecture des applications collaboratives en RA/RV est nécessaire de sorte à permettre leur conception sans connaissance par les développeurs des spécificités des supports.

Critères de succès et orientation des travaux

Comme énoncé dans la section "Objectifs des travaux", notre objectif premier est de proposer une nouvelle architecture pour la conception des applications collaboratives en RA/RV. Pour valider/infirmer cette architecture, nous avons définis les conditions de succès suivantes :

- La solution proposée doit valider les attendus vis à vis d'une solution de travail assisté par ordinateur (CSCW).
- L'exploitation des fonctionnalités spécifiques à l'un des support utilisé ne doit pas être restreinte.
- La solution proposée doit permettre de concevoir une application 3D multi-supports sans connaissance préalable des supports et de leurs propriétés.
- L'adaptation de l'interface utilisateur au support ne doit pas être prise en charge par le concepteur de l'application collaborative.

L'évaluation et l'orientation de nos travaux a donc été déterminée prioritairement en fonction de ces critères plutôt que vers une étude approfondie des performances utilisateur propres aux différents systèmes présentés. En effet, la proposition dans le Chapitre 1 d'un modèle de donnée permettant une description générique de l'interaction en environnement 3D aurait pu entrainer une réorientation de cette thèse vers une étude approfondie, de l'interprétation de ce modèle par les différents supports d'une part, et de l'expérience utilisateur découlant de l'utilisation de ce modèle d'autre part. Notre choix a toutefois été de privilégier dans un premier temps une vérification de la faisabilité technique d'une interprétation de ce modèle. En effet, si le modèle proposé permettait par construction de valider trois de nos quatre critère de succès, il restait une incertitude majeure quand à

sa capacité à répondre aux attendus propres au support du travail collaboratif assisté par ordinateur (notre premier critère). De plus, si le modèle proposé permettait bien de valider la faisabilité de l'architecture proposée, celui ci comportait des lacunes, à notre sens encore trop importantes (comme l'absence de suivis de la navigation dans l'environnement 3D, ou encore l'absence d'interactions naturelles avec l'environnement 3D) pour être soumis en l'état à des utilisateurs. Des travaux annexes sont toutefois en cours pour compléter ce modèle d'abstraction et évaluer son interprétation par les supports de RA/RV. Une présentation de ces travaux a donc été incluse dans ce manuscrit (Chapitre 4 et Chapitre 5) de sorte à faciliter sa compréhension, et à donner de premiers indicateurs de performance.

Plan du document

Dans le but de justifier notre proposition d'une nouvelle architecture logicielle, nous réalisons dans le Chapitre 1 un état de l'art des solutions et standards existants pour la création d'applications multi-supports en RV et en RA. Cette synthèse des outils existants et de leurs limites respectives nous a permis de conclure qu'une nouvelle méthode de conception des applications collaboratives de RA/RV est nécessaire pour répondre à l'ensemble des objectifs fixés.

Nous proposons donc dans le Chapitre 2 une première contribution [19]. Cette contribution propose une nouvelle architecture logicielle pour les applications collaboratives en RA/RV. L'architecture en question s'appuie sur un protocole web (que nous avons appelé UMI3D pour "Unified Model for Interaction in 3D Environment") permettant à tout support de RA/RV d'interagir avec un environnement virtuel (ou augmenté) hébergé sur un serveur. Ce chapitre présente donc l'approche utilisée pour la conception de l'architecture proposée, à savoir l'application à l'AR/VR des principes de conception propres aux systèmes de gestion d'interface utilisateur (UIMS), ainsi que la séparation architecturale des paradigmes de l'interaction homme-machine. Nous expliquons par la suite de quelle manière nous avons construit, en nous appuyant sur les travaux existants concernant la manipulation de degrés de liberté, un modèle de données permettant une description agnostique des interactions possible avec un média 3D. La suite de ce chapitre décrit une preuve de concept dans laquelle nous relatons la manière dont nous avons réalisé une première implémentation du protocole UMI3D et de l'architecture proposée. Dans le but de convaincre que notre modèle de données permet bien d'implémenter des manipulations simples (translations, orientations, changements d'échelle), mais également des manipulations complexes (intégration de plusieurs tâches dans une même manipulation), nous concluons cette partie par un ensemble d'expérimentations techniques (c'est à dire des exemples d'interactions réalisables avec le protocole UMI3D).

Dans une deuxième contribution [21], nous démontrons que l'architecture proposée précédemment, ainsi que l'utilisation du protocole UMI3D permettent bien, sous certaines

conditions, de répondre aux attendus d'un système de travail collaboratif assisté par ordinateur (CSCW). Pour se faire, nous proposons d'une part une boîte à outil donnant un exemple d'implémentation des requis techniques pour le support de la collaboration en environnement 3D (présentés en conclusion du Chapitre 1). D'autre part, nous illustrons l'utilisation de ces outils par conception d'un jeu de construction collaborative. Ce cas d'étude nous permet dans un premier temps de discuter des implications de nos travaux sur le processus de conception des médias 3D, et dans un second temps de vérifier à travers une étude utilisateur que les mécanismes attendus durant une session de travail collaboratif sont bien observés.

Nous présentons par la suite dans les chapitres 4 et 5 des travaux préliminaires réalisés en collaboration avec Nicolas Pacqueriaud et Antoine Ladrech [1]. Le Chapitre 4 présente une extension du protocole UMI3D permettant la représentation des utilisateurs par des avatars. Cette extension nous permet de lever les limites identifiées dans le Chapitre 3 concernant le support du travail collaboratif avec UMI3D. Le Chapitre 5 aborde quand à lui les problématiques liées à l'évaluation des médias 3D et surtout des navigateurs UMI3D. Pour palier à ces problèmes, nous expliquons comment nous avons utilisé une plateforme d'analyse de données massive (Big Data) pour construire un système d'évaluation des navigateurs UMI3D. Pour finir, nous présentons les résultats d'une première étude comparative menée avec ce système.

Pour conclure ce document, nous présentons les limites actuelles du protocole UMI3D, ainsi que les pistes d'extension que nous avons identifiées pour dépasser ces limites.

1. Nicolas Pacqueriaud et Antoine Ladrech sont ingénieurs recherche et développement dans l'équipe dont je suis responsable chez Gfi Informatique.

Table des matières

Chapitre 1

Collaboration Multi-Support en Réalité Virtuelle et en Réalité Augmentée

Dans ce chapitre nous rappelons dans un premier temps les différentes définissions de la Réalité Virtuelle (RV), de la Réalité Augmentée (RA), et de la Réalité Mixte (RM). Nous nous relevons en particulier l'absence de définition communément admise pour la Réalité Mixte. Au vu de la complexité du vocabulaire, et surtout des ambiguïtés possible entre le matériel (support) et le logiciel (média), nous introduisons par la suite notre définition d'un média 3D pour l'ensemble du document. Dans un deuxième temps, nous proposons une revue synthétique des outils et méthodes existantes pour la conception de médias 3D multi-supports. Cette revue nous permet de mettre en évidence les limites des approches actuelles vis à vis des besoins industriels en termes de conception d'expériences collaboratives multi-supports. Pour terminer ce chapitre nous établissons, à travers la littérature propre au domaine du travail collaboratif assisté par ordinateur (CSCW), une liste des attendus pour un média 3D. Il s'agit ici d'établir l'ensemble des propriétés qui doivent être vérifiées pour valider/infirmer qu'une nouvelle méthode de conception des médias 3D permet bien le support du travail collaboratif.

1.1 Réalité Virtuelle, Réalité Augmentée

Cette section introduit le vocabulaire et les principales définitions propres aux domaines de la Réalité Virtuelle, de la Réalité Augmentée, ainsi que de la collaboration assistée par ordinateur.

Réalité Virtuelle. La Réalité Virtuelle (RV) est une technologie permettant d'immerger une ou plusieurs personnes dans un **environnement virtuel**. Un environnement virtuel est un espace 3D entièrement généré par ordinateur. On appellera par la suite support ou dispositif de Réalité Virtuelle, tout appareil permettant de générer une expérience de Réalité Virtuelle comme par exemple, les casques de Réalité Virtuelle (notés Hmd) ou encore les CAVEs.

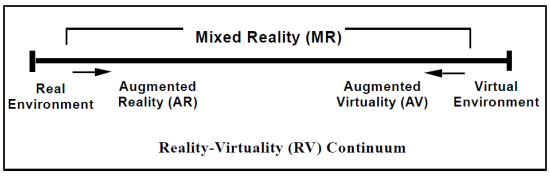


FIGURE *1.1 – Le continuum de Milgram 1994. [61]*

Réalité Augmentée. La Réalité Augmentée (RA) est une technologie permettant de superposer ou d'insérer du contenu virtuel généré par ordinateur (objet 3D, son, odeur, etc...) dans un environnement réel. La résultante, idéalement perçue par l'utilisateur comme formant un environnement unique, est appelée **environnement augmenté**. Dans le cadre très vaste de la Réalité Augmentée, le niveau d'interaction entre les objets réels et virtuels est très variable. Il peut aller du positionnement graphique des objets virtuels sur une image réelle [3, 85] jusqu'à une inclusion réaliste des objets virtuels dans le monde réel [51].

Virtualité Augmentée. La Virtualité Augmentée (VA) permet quand à elle d'insérer des éléments, ou des informations, provenant de l'environnement réel où évoluent le ou les utilisateurs dans l'expérience de Réalité Virtuelle. S'agissant principalement, d'une extension de la Réalité Virtuelle, les deux termes sont couramment regroupés sous la dénomination de Réalité Virtuelle.

Le continuum de Milgram. Dans [61], Milgram et Kishino définissent la Réalité Mixte comme étant le continuum entre le réel et le virtuel. (représenté Figure 1.1). Tout environnement incluant des éléments réels et des éléments virtuels peut donc être considéré comme étant une expérience de Réalité Mixte (RM). Il existe ainsi deux approches technologiques pour produire ce type d'expérience : La Réalité Augmentée, et la Réalité Virtuelle (via la VA). La notion de continuum réel-virtuel est devenue plus concrète avec les avancées technologique récentes, qui ont permis à des dispositifs de Réalité Virtuelle tels celui présenté dans [55] de produire par Virtualité Augmentée des expériences similaires à celles couramment réalisées avec des dispositifs de RA.

Pas de consensus concernant la définition de la Réalité Mixte. Si la définition originelle de la Réalité Mixte a bien été proposée par Milgram et Kishino comme évoqué précédemment, cette définition fait aujourd'hui débat dans la communauté scientifique aussi bien que dans l'industrie. Une étude récente de Speicher et al. [79] a permis de mettre ce débat en évidence en montrant, d'une part qu'aucune définition de la Réalité Mixte n'est majoritaire dans la littérature, et d'autre part qu'il existe au moins six définitions largement adoptées pour la Réalité Mixte. Pour arriver à ces conclusions, Speicher et al. s'appuient sur une revue de littérature de 68 articles, ainsi que sur l'audition de dix experts

académiques et industriels. Il en ressort six notions, relatives au concepts de Réalité Mixte, pouvant être couramment reliées aux articles et/ou aux définitions des experts :

- **Continuum** (35,3% des articles, 3/10 experts). Il s'agit de la définition issue du continuum de Milgram. La RM est ici un mélange d'objet virtuels et réels visibles dans un même support. Il en résulte un environnement situé entre la réalité et un environnement totalement virtuel.
- **Synonyme** (23,5% des articles, 2/10 experts). Pour une partie de la communauté, le terme de Réalité Mixte est totalement interchangeable avec celui de Réalité Augmentée.
- **Collaboration** (11,8% des articles, 0/10 expert). Dans certains cas, la RM est considérée comme étant la collaboration entre des utilisateurs évoluant en RA, et des utilisateurs évoluant en RV. Les articles adoptant cette définition s'intéressent à la manière de reconstituer, en environnement virtuel, l'environnement réel dans lequel évoluent le/les utilisateurs de RA. [28]
- **Combinaison** (7,4% des articles, 2/10 experts). Pour une autre partie de la communauté, la RM désigne soit des applications combinant des fonctionnalités propres à la RA et des fonctionnalités propres à la RV, soit des supports ayant la capacité à produire à la fois des expériences de RV et des expériences de RA.
- **Alignement** (23,5% des articles, 0/10 expert). La notion d'alignement se réfère soit à la synchronisation entre un espace physique et un espace virtuel, soit à l'inverse à l'alignement d'un environnement virtuel à la réalité. Si Speicher et al. pointent ici un recouvrement partiel avec les notion de Collaboration et de Combinaison, cette définition nous semble fortement lié aux thématiques de jumeau numérique. Ces travaux visent entre autre à utiliser des objets connectés (IoT) soit pour le pilotage en environnement virtuel d'objets réels, soit pour la supervision en VR d'objets réels distants.
- **RA plus avancée** (7,4% des articles, 3/10 experts). Enfin, pour une partie de la communauté, la RM consiste en une amélioration des expériences de RA par une meilleure compréhension de l'environnement réel par le support.

Cette absence de consensus sur la définition de la RM n'est de notre point de vue pas problématique car elle enrichit de débat scientifique autour des technologies immersives. Elle induit toutefois certaines difficultés de communication au sein de la communauté. En particulier, l'absence de terme désignant l'ensemble des expériences de RA, RV et RM nous a souvent fait défaut pour expliquer nos travaux (le terme de réalité étendue XR est plutôt employé pour désigner l'ensemble des domaines de la RV/RA/RM que pour désigner une expérience). Pour cette raison, nous introduisons par la suite la notion de média 3D.

1.2 Notion de média 3D

1.2.1 Paradigmes de l'interaction homme-machine

Dans [6], Beaudouin-Lafon présente trois paradigmes pour l'interaction homme-machine :

— **Computer-as-a-tool.** Le paradigme 'computer-as-a-tool' regroupe toutes les fonctionnalités qui étendent les compétences de l'utilisateur. L'une des illustrations les plus simples de ce paradigme est le déplacement d'un curseur 2D par la souris d'un ordinateur. Dans ce cas, l'utilisateur acquière la compétence de déplacer le curseur sans avoir à penser la manière dont il déplace la souris. Plus généralement on peut considérer comme des outils la plupart des fonctionnalités directement associées à la manipulation par l'utilisateur d'un périphérique d'entrées.

— **Computer-as-a-partner.** Ce paradigme correspond à l'ensemble des moyens de communication anthropomorphes des systèmes informatiques. Ceci regroupe par exemple l'interaction en langage naturel, ainsi que les agents virtuels qui ont vocation à exécuter de manière autonome des tâches pour aider un ou des utilisateurs.

— **Computer-as-a-media.** Le paradigme 'computer-as-a-media' désigne les moyens et systèmes informatiques utilisés comme moyen de communication entre utilisateurs humains. On peut par exemple classer dans ce paradigmes les systèmes de conférence vidéo et de messagerie. La définition de ce paradigme peut de notre point de vue également être élargie à tout moyen de mettre un même contenu digital, qu'il soit ou non interactif, à disposition d'un groupe ou d'une communauté d'utilisateurs.

Ces paradigmes permettent de classifier les interfaces homme-machine (IHM), ou du moins leurs modules. Une interface complexe pouvant en effet proposer des interactions appartenant à différents paradigmes. Par ailleurs, Beaudouin-Lafon [6] pointe justement que ces trois paradigmes de l'interaction homme-machine sont étudiés par trois communautés scientifiques distinctes : Le paradigme 'computer-as-a-tool' est principalement étudié par la communauté s'intéressant à l'interaction homme-machine (IHM). Le paradigme 'computer-as-a-partner' est quand à lui relatif au domaine de l'intelligence artificielle (IA). Enfin, le paradigme 'computer-as-a-media' est le propre de la communauté scientifique intéressée par le travail collaboratif assisté par ordinateur (CSCW).

1.2.2 Application à la collaboration en Réalité Virtuelle et en Réalité Augmentée

Dans le but de proposer une définition du média 3D commune à la RA, la RV et la RM, nous proposons d'appliquer les paradigmes de l'interaction précédemment décris aux domaines de la RV/RA/RM :

— **Computer-as-a-tool.** Dans le cadre d'une expérience de RV/RA/RM, un en-

semble de fonctionnalités est associé à la manipulation par l'utilisateur du support. Ces fonctionnalités permettent de mettre en œuvre des outils pour l'interaction entre l'utilisateur et l'environnement virtuel ou augmenté. Une première partie de ces outils, liée aux périphériques de sorties, permet à l'utilisateur de percevoir/ressentir visuellement et/ou par ses autres sens la dimension digitale de l'environnement. La seconde partie de ces outils, liée aux périphériques d'entrées, permet à l'utilisateur d'agir sur l'environnement dans lequel il évolue.

— **Computer-as-a-media.** Si l'on considère la définition donné précédemment pour le paradigme 'computer-as-a-media', celle-ci peut s'appliquer a une partie de ce que l'on décrit habituellement comme étant un environnement virtuel ou augmenté. En effet, l'ensemble des contenus digitaux comme les objets 3D virtuels ou les artéfacts (en RA) font par définition partie d'un média. Par ailleurs, l'ensemble des fonctionnalités de l'application visant à permettre la collaboration telles que la communication orale et visuelle, ou encore l'existence de rôles utilisateur sont également à rattacher à ce paradigme. Enfin, les interactions possible avec les contenus sont à notre sens indissociables de ces contenus et font donc également parties du média.

— **Computer-as-a-partner.** Comme présenté précédemment lors de sa définition, les interfaces en langage naturel parfois utilisées en RA/RV pour l'interaction main libre (exemple : système de menus vocaux des lunettes de RA HoloLens), se rattachent au paradigme 'computer-as-a-partner'. On retrouve également ici les agents virtuels ou personnages non jouables (PNJ) qui sont animés par intelligence artificielle.

1.2.3 Définition d'un média 3D

Au vu des concepts et composantes propres aux applications de RV/RA/RM, et surtout des paradigmes de l'interaction auxquels ils se rattachent, nous proposons/adoptons les définitions suivantes pour la suite de ce document :

Média 3D. Nous définissons un média 3D comme un système informatique, accessible simultanément par un ensemble d'utilisateurs, regroupant les éléments suivant :

— Un ensemble d'objets digitaux organisés en trois dimensions par rapport à un référentiel virtuel (RV) et/ou des points d'ancrage géo-spatiaux (RA).

— Un ensemble d'interactions possibles entre les utilisateurs potentiels et le contenu digital du média.

— Des moyens techniques permettant (et organisant) la collaboration entre utilisateurs.

Support. Nous désignerons par la suite par support tout dispositif de RA/RV (Ceci inclut les périphériques d'entrées et de sorties) permettant l'implémentation d'outils pour l'interaction avec un ou des médias 3D.

1.3 Méthodes & standards pour la conception de médias 3D multi-supports

1.3.1 Les moteurs de jeu

Les outils de conception les plus répandus en RA et en RV sont les moteurs de jeu tels que Unity, Unreal Engine, ou encore Blender (voir Tableau 1.1). Ces moteurs de jeu ont pour objectif commun la simplification de la création de médias 3D interactifs. La plupart d'entre eux permettent de compiler, et de distribuer des applications 3D pour plusieurs supports et plateformes. Si ces moteurs permettent au concepteur de médias 3D l'usage, et donc l'apprentissage, d'un unique outil pour développer les médias 3D multi-supports, il reste nécessaire de compiler séparément l'application pour chaque support. De plus l'usage de boîtes à outils spécifiques au support peut être nécessaire pour la gestion des périphériques d'entrées et de sorties. Enfin, si l'usage du moteur de jeu permet de mutualiser une partie du code source de l'application pour les différents supports (i.e. en particulier l'implémentation des règles métier), l'interface utilisateur ne peut être conservée que si les dispositifs supportés sont fortement semblables. C'est à dire si les dispositifs ont des périphériques d'entrées/sorties similaires, et que l'interface utilisateur convient à l'ensemble de ces supports (du point de vue de l'ergonomie).

1.3.2 Normalisation des périphériques d'entrées/sorties

L'une des approches existantes, pour améliorer le partage d'une même interface entre plusieurs supports, consiste à normaliser la connexion et les échanges de données entre l'application 3D et les périphériques d'entrées/sorties. C'est par exemple l'approche du middleware MiddleVR [53]. Cette boîte à outils permet de simplifier le développement d'applications de RV/RA en donnant au concepteur des outils haut-niveaux pour gérer les périphériques d'entrées/sorties dans le moteur de jeu Unity. Ceci permet notamment de remplacer un périphérique d'entrées (par exemple un contrôleur 6D) ou de sorties (un écran 2D par un Hmd) par un autre sans modifier le code de l'interface.

Plus récemment, des standards de programmation tels que WebVR [1] [35] ou encore OpenXR [64] ont été proposés dans cette même optique de normalisation de l'accès aux périphériques. WebVR permet au concepteur de créer des applications de RV à destination d'un navigateur Web en disposant d'un ensemble de fonctionnalités de base, communes

1. WebVR a récemment fusionné avec WebAR pour devenir WebXR.

TABLE *1.1 – Principaux moteurs de jeu. Une liste exhaustive est disponible dans [94]*

Moteur de jeu	Plateformes	Langage des scripts
Blender	Windows, Linux, OS X, Solaris	Python
CryEngine	Windows, OS X, Linux, PlayStation 3, PlayStation 4,Wii U, Xbox 360, Xbox One, iOS, Android	Lua, C#
OGRE	Linux, Windows, OS X, NaCl, WinRT,Windows Phone 8, iOS and Android	C++
Three.js	HTML5, Windows, Linux, OS X, iOS and Android	JavaScript
Unity	Toute plateforme existante	C#, Cg, HLSL
Unreal Engine	Toute plateforme existante	GLSL, Cg, HLSL, UnrealScript, C++, Blueprints

à tous les dispositifs de RV. OpenXR normalise quant à lui dans une API (Application Programmation Interface) la manière dont un périphérique décrit et expose ses données d'entrées et de sorties. OpenXR permet donc d'obtenir, pour les supports l'implémentant, le même résultat que MiddleVR d'un point de vue de la facilité d'accès aux périphériques. Il est par ailleurs a noter que la plupart des moteurs de jeu tels que Unity disposent aujourd'hui d'interfaces de programmation (API) implémentant cette approche soit directement [86], soit par des extensions telles que VRTK [91].

1.3.3 Modèles d'abstraction basés sur les fonctionnalités communes

Nous avons vu précédemment, qu'en plus de normaliser l'accès aux périphériques dans une application web, WebVR permet de concevoir des applications 3D multi-supports, à condition de n'utiliser que les propriétés communes des supports, ou de mettre en place une adaptation de l'interface utilisateur au support (voir section suivante).

Ce principe d'abstraction du support par un ensemble de fonctionnalités, communes à l'ensemble de dispositifs supportés, est à la base d'une très large majorité des boîtes à outils dédiées à la 3D multi-support. C'est par exemple le cas de RUIS [81], une boîte à outils facilitant le développement de médias 3D pour une liste finie de supports. L'objectif de RUIS est de rendre accessible à des concepteurs non-experts, comme des étudiants, la conception d'applications de RV. L'utilisation d'un ensemble de fonctionnalités simples, communes aux dispositifs supportés, est donc ici particulièrement pertinente. De manière générale, cette approche est plutôt adaptée aux applications 3D pour lesquelles on souhaite reproduire l'expérience utilisateur la plus similaire possible quelque soit le support. C'est typiquement le cas de la majorité des jeux vidéo.

La boîte à outil #Five, présentée dans [15], constitue à l'inverse un premier pas vers une abstraction du support basée sur l'interaction. Cette boîte à outils fournit une aide, indépendante du support, pour répartir des interactions entre plusieurs utilisateurs et dispositifs, tout en fournissant une manière générique de décrire les interactions possibles avec les objets virtuels, grâce à leur propriétés éditables. La limite de cette approche est que le modèle d'abstraction du support est ici réservé à la conception des règles de collaboration. #Five doit donc se combiner avec une abstraction orientée fonctionnalités pour le reste de l'application (navigation, contrôle d'application, interactions individuelles avec le contenu)

1.3.4 Plasticité des interfaces utilisateur

Si l'effort de standardisation dans l'interfaçage avec les périphériques permet techniquement de simplifier l'accès aux périphériques des différents supports, ceci ne règle pas la problématique d'ergonomie soulevée par les différences entre dispositifs (i.e. l'interface

utilisateur adaptée n'est pas la même selon le support). La thématique de la plasticité des interfaces s'intéresse à ce problème en recherchant des outils et méthodes pour concevoir des applications 3D dont les interfaces s'adaptent dynamiquement aux périphériques utilisés.

1.3.4.1 Définition

La plasticité a été définie, pour tout système interactif, comme étant "la capacité qu'a un système interactif à résister aux variations des caractéristiques physiques du système et de l'environnement tout en préservant son utilisabilité" [83, 62].

Il est important de signaler ici que cette définition n'inclut pas uniquement l'adaptation d'une interface à un support et ses propriétés. On peut, notamment dans le domaine de la RV, faire mention des nombreux travaux existants sur l'adaptation d'aides à l'apprentissage en environnement virtuel. Dans ce cas, l'adaptation de l'interface ne concerne que peu (voir pas) l'adaptation au support mais plutôt une adaptation au contexte utilisateur. Un premier exemple est présenté par Burkhardt et al. dans [17]. Dans cet article, les auteurs introduisent un système d'aides pour l'apprentissage en simulateur de la conduite automobile. Ce système comprend principalement un affichage adaptatif d'aides à la prise d'informations contextuelles par le conducteur. La problématique est ici de concevoir une interface pertinente pour une infinité (ou un nombre extrêmement élevé) de situations possibles. L'adaptation d'aides visuelles en fonction du type d'erreur utilisateur, tel que proposé par Jeanne dans [45], constitue un second exemple. Ici, l'adaptation de l'interface est également faite par rapport au contexte utilisateur. Toutefois, si le premier exemple basait principalement l'adaptation sur le contexte de l'environnement virtuel, elle est ici plutôt basée sur la connaissance (progressivement consolidée) des erreurs passées de l'utilisateur.

1.3.4.2 Interfaces adaptables et adaptatives

De manière plus générale, les interfaces plastiques peuvent être classifiées en deux catégories : les interfaces adaptables et les interfaces adaptatives [65]. Un système dit adaptable fournit à l'utilisateur des outils lui permettant de modifier certaines des caractéristiques de ce même système. L'utilisateur prend donc dans ce cas l'initiative de l'adaptation du système en fonction de son contexte d'usage. À l'inverse, un système est dit adaptatif s'il est capable de modifier automatiquement ses propres caractéristiques en fonction des besoins de l'utilisateur. Dans ce cas l'adaptation de l'interface est à l'initiative du système ce qui peut, d'après Oppermann [65], dans certains cas ne pas être parfaitement intégré par les utilisateurs. Pour réduire ce problème, Oppermann propose un système combinant les deux types d'adaptations. Celui propose en effet de manière adaptative à l'utilisateur de réaliser lui-même des adaptations de l'interface. L'utilisateur

a ainsi une meilleure compréhension des adaptations du système.

1.3.4.3 Cas particulier de l'adaptation au support

Dans le cas particulier d'une interface utilisateur multi-support, la définition générale de la plasticité implique que cette interface préserve son utilisabilité, sur tous les supports ciblés, pour être considérée comme s'adaptant au support. Pour les interfaces 3D, ceci nous indique que la seule normalisation des périphériques d'entrées et de sorties n'est pas suffisante pour considérer que cette interface multi-support remplit les critères de la plasticité. L'interface doit en effet pour cela également, si les supports ont des différences importantes, adapter la présentation et l'ergonomie de l'interface utilisateur. Si il est trivialement possible de proposer des outils d'adaptation dans une interface 3D (c'est le cas pour la configuration des contrôles utilisateur dans une large majorité de jeux vidéo), nous soutenons que l'adaptation de l'interface et son ergonomie globale ne peuvent pas être laissés à l'entière discrétion de l'utilisateur. C'est pourquoi le concepteur du système doit impérativement rendre adaptatives au support les interfaces 3D.

L'application la plus répandue de l'adaptativité des interfaces utilisateur à différents supports se trouve certainement dans le domaine du développement web avec le 'Responsive Web design' (noté RWD). On parlera plutôt de site web adaptatif en français. Le principe du RWD consiste à faire varier la présentation et la mise en page d'un site web en fonction d'un certain nombre de paramètres identifiés pour les systèmes interactifs 2D : Taille de l'écran, type d'interaction (tactile, souris & clavier, vocale), accès aux capteurs du dispositif, mobilité. Pour ce faire, le concepteur du site web doit définir l'interface spécifique à chaque combinaison de valeurs des paramètres qu'il souhaite prendre en compte.

Comme relevé par Lacoche dans [54], la transcription directe des principes du RWD aux applications de RV/RA est extrêmement couteuse en développement. En effet, le nombre de développements nécessaires est directement corrélé au nombre de supports, ainsi qu'à la diversité des périphériques d'interaction à supporter et des environnements utilisateur. La diversité des moyens d'interaction en environnement 3D, mais surtout de l'environnement utilisateur (allant de l'immersion en RV dans un environnement finis et sécurisé, à la RA mobile en environnement réel ouvert), rendent irréaliste la prise en compte par le concepteur de toute les possibilités. Si des solutions comme MiddleVR [35], VR Juggler [12], ou encore Colaviz [32] permettent de réduire le volume de développements nécessaires, il reste toujours plus complexe de concevoir une application de RA/RV pour plusieurs supports que de la concevoir pour un seul support. Ces outils restent donc réservés à des concepteurs ayant une certaine connaissance préalable de l'outil et du développement multi-support en RA/RV.

1.3.5 Limites actuelles pour la conception de médias 3D multi-supports

Au vu des outils et méthodes de développement présentés précédemment, nous sommes en mesure d'affirmer que ces outils permettent de manière satisfaisante, et accessible par une large majorité de développeurs 3D (grâce aux moteurs de jeu), de concevoir un média 3D pour un unique support. En effet, la normalisation des périphériques d'entrées et de sorties est aujourd'hui ,soit directement intégrée aux moteurs de jeu pour les supports dits "grand public" (smartphone, HMD), soit apportée par l'utilisation de middlewares. Cette normalisation permet à tout développeur 3D de concevoir de manière relativement similaire des médias 3D à destination d'un support, voir d'un type de support. Toutefois, le passage à un média 3D à destination de supports variés (et potentiellement asymétriques) impose au concepteur de ce média d'implémenter des mécanismes rendant l'interface utilisateur adaptative au support. Comme exposé en conclusion de la section précédente, l'adaptation à des supports de RV/RA variés demande la prise en compte d'un grand nombre de paramètres et nécessite de plus une bonne connaissance des outils (frameworks) permettant l'adaptation. S'ajoute à cela la nécessité d'une connaissance avancée de l'ergonomie et plus généralement des bonnes pratiques de conception d'interfaces utilisateurs pour chaque support. La conception de médias 3D multi-supports par les méthodes existantes reste donc, de notre point de vue réservée à une communauté de concepteurs confirmés (si ce n'est experts). Enfin comme présenté précédemment dans notre description des différents paradigmes de l'interaction homme-machine, la communauté s'intéressant à la plasticité des interfaces n'est pas exactement la même que celle s'intéressant au travail collaboratif assisté par ordinateur, et donc à la conception de médias. Il faut donc s'attendre à ce que les concepteurs de médias 3D aient au mieux une connaissance de base concernant l'adaptation des interfaces utilisateur au support.

Par ailleurs, un deuxième inconvénient de l'approche est le fait que l'adaptation de l'interface utilisateur au support soit portée par le média 3D. Ceci implique que pour deux médias donnés, l'interface utilisateur pourra être totalement différente pour un même dispositif. Ceci est un frein à l'établissement par les utilisateurs finaux, d'habitudes d'usage propres à leur dispositif. Les utilisateurs doivent donc, pour chaque média, découvrir une nouvelle ergonomie, et donc produire un effort cognitif important avant d'être performants dans l'utilisation de l'application.

Pour ces raisons, nous sommes convaincus qu'il nous faut explorer d'autres solutions que le développement d'une unique application s'adaptant à tous les supports, pour atteindre l'objectif de médias 3D offrant aux utilisateurs une ergonomie optimale quelque soit les supports utilisés. Il nous semble en particulier intéressant d'expérimenter une nouvelle architecture des médias 3D, de sorte à permettre que l'adaptation de l'interface utilisateur soit portée par le support plutôt que par le média 3D. Il est important de

noter que, du point de vue utilisateur, cette approche doit produire le même type d'expérience qu'une interface utilisateur adaptative, remplissant les critères de la plasticité. En effet, le fait que l'application s'adapte au support, ou que le support s'adapte au média, doit être transparent pour l'utilisateur. Il doit simplement constater que les interfaces sont différentes pour deux supports différents, et quelles sont adaptées à leurs supports respectifs.

1.4 CSCW : Travail collaboratif assisté par ordinateur

'Computer-Supported Cooperative Work' ou CSCW est un domaine de recherche s'intéressant à la conception des systèmes informatiques ayant pour but le support du travail collaboratif (notés CSCWS par la suite). La littérature propre à ce domaine définit l'ensemble des propriétés que doit avoir un système de support du travail collaboratif. Ces propriétés nous permettent de déduire la liste des fonctionnalités, que nous devons démontrer comme étant implémentables, avant de considérer toute nouvelle méthode de conception des média 3D comme étant adaptée au support de la collaboration.

1.4.1 Quels prérequis pour le support de la collaboration ?

1.4.1.1 Contexte partagé

Dans [23] Churchill présente l'existence d'un contexte de travail partagé comme l'une des propriétés de base des CSCWS. Ce qui signifie que les utilisateurs du système partagent certains contenus et retours visuels (ou artéfacts), et ont une certaine connaissance des activités (i.e. des interactions) passées et présentes des autres utilisateurs. Cette dernière propriété peut se résumer à un sentiment de compréhension mutuelle entre les utilisateurs. Dans le but d'implémenter cette propriété, le système de support de la collaboration (CSCWS) doit pouvoir générer des artéfacts partagés lorsque les utilisateurs interagissent avec le média [30]. Cette fonctionnalité est d'autant plus importante quelle est utile pour la communication et les débats entre les utilisateurs [27, 36].

1.4.1.2 Mécanismes de coordination

L'amélioration de la coordination des utilisateurs durant des activités coopératives est l'un des buts principaux des CSCWS [60, 75, 71, 39]. Dans le but de spécifier les conditions de l'organisation de la collaboration dans un CSCWS, Schmidt et Simone [75] proposent de distinguer deux composantes pour la collaboration : le travail coopératif (cooperative work) et le travail d'articulation (articulation work). Le travail coopératif est ici décrit comme étant l'interdépendance entre les tâches exécutées par les différents acteurs et le

référentiel de travail commun. De son côté, de travail d'articulation recouvre l'ensemble des contraintes utilisées pour réduire la complexité de la collaboration. Cela inclut l'ensemble des moyens permettant de coordonner, planifier ou aligner des activités interdépendantes. Par ailleurs, Schmidt et Simone présentent le travail coopératif et le travail d'articulation comme se définissant récursivement. Ils expliquent cela par la nécessité de laisser les utilisateurs définir eux même l'articulation du travail collaboratif. Cette négociation par les utilisateurs de l'articulation de leur travail constitue en effet une tâche coopérative. L'ensemble de ces propriétés implique, pour concevoir des applications collaboratives, que le concepteur doit disposer d'outils de développement permettant d'implémenter des interactions (i.e. des tâches) coopératives et parallèles. Enfin, nous devons fournir des fonctionnalités permettant d'articuler et organiser au cours du temps la distribution des interactions.

1.4.1.3 Malléabilité

L'un des problèmes couramment rencontrés avec les CSCWS est le manque de malléabilité des mécanismes de coordination. Dans [75] Schmidt et Simone définissent la malléabilité des systèmes de support de la collaboration comme l'implication des utilisateurs dans la définition des mécanismes de coordination. La malléabilité étant ici présentée comme l'un des attendus fondamentaux des CSCWS. Plus précisément, Schmidt et Simone soutiennent que les acteurs devraient pouvoir ajouter ou modifier des mécanismes de coordination durant l'exécution du système. Les prérequis pour implémenter cette fonctionnalité sont, d'une part l'utilisation d'une boîte à outils suffisamment flexible pour le développement du système collaboratif, et d'autre part de rendre les mécanismes d'articulation visibles et sémantiquement compréhensibles pour les utilisateurs. L'idée que nous avons besoin d'outils flexibles dédiés au développement de systèmes collaboratifs est partagée par Dourish dans [29]. Dans cet article, Dourish théorise différents niveaux pour la malléabilité et la flexibilité des CSCWS. Le plus faible niveau, mais le plus simple à implémenter, consiste à donner aux utilisateurs la possibilité de configurer un ensemble de paramètres durant l'exécution du système. Le niveau le plus élevé consiste quand à lui à permettre aux utilisateurs de modifier le fonctionnement même du système durant l'exécution. Un exemple de ce niveau de malléabilité supérieur serait de laisser les utilisateurs définir eux même la manière dont ils se répartissent les interactions en fonction d'une stratégie de travail qu'ils souhaitent adopter. Finalement, Dourish soutient que la principale raison au manque de malléabilité dans les systèmes collaboratifs provient du manque de flexibilité des outils de développement dédiés aux CSCWS. Ceci s'explique par le choix des concepteurs d'outils de développement de favoriser la simplicité et la vitesse de développement en proposant uniquement des comportements pré-configurés au détriment des possibilités d'usage.

1.4.1.4 Rôles utilisateur

La plupart des CSCWS sont conçus pour soutenir des processus coopératifs complexes. Ces processus impliquent divers acteurs avec des rôles et des droits d'interaction très différents. Dans [23] Churchill et Snowdon soulignent qu'il découle, de cette diversité de rôles utilisateur, un besoin de points de vue multiples et flexibles pour les CSCWS. Cela signifie qu'un système de support de la collaboration devrait idéalement permettre des variations dans les contenus et les artéfacts visibles en fonction du rôle de l'utilisateur dans la collaboration. De même, les CSCWS devraient pouvoir distribuer différentes tâches aux utilisateurs en fonction de leurs droits. Ceci doit être vu comme des contraintes dans la malléabilité du système : Les utilisateurs doivent pouvoir définir ensemble leur stratégie de coopération, mais cette stratégie doit rester compatible avec chacun des droits sur le média 3D.

1.4.1.5 Activités individuelles

Au cours d'une situation de travail coopératif, les différents acteurs alternent continuellement entre activités partagées et individuelles [23, 75]. Ces alternances dans le type d'activité ont été conceptualisées par Schmidt et Simone comme représentant l'interdépendance entre entre les contextes de travail personnel et partagé. S'il parait évident que les différents acteurs doivent être conscients de l'impact des activités individuelles sur le contexte de travail partagé, nous soutenons que tous les artéfacts et retours visuels générés par les activités individuelles ne doivent pas être partagés. Seuls les artéfacts ayant une utilité pour la compréhension sémantique des activités individuelles doivent être partagés pour éviter une surcharge cognitive des autres utilisateurs. Il en résulte la nécessité de supporter des retours visuels individuels dans les CSCWS.

1.4.1.6 Conscience des autres

La conscience des autres (Awareness of others) a été initialement définie comme la compréhension des activités des autres acteurs. Cette compréhension définit, avec notre propre activité, un contexte de travail patagé [30]. Cette définition à la première personne a été étendue dans certains travaux plus récents pour inclure l'idée d'une conscience mutuelle ou partagée (mutual or shared awareness) [82, 74]. Cela signifie que la notion de conscience des autres inclut dorénavant l'idée d'une compréhension récursive des activités de chacun.

La transcription et l'application des modèles de conscience des autres aux environnements 3D ont déjà été étudiés par le passé, notamment par Benford et al. [7] qui propose un modèle spatial de l'interaction. Le point central de ce modèle est que tout objet nécessite un moyen technique, appelé Aura, pour lui permettre d'interagir avec les autres objets. De plus, l'Aura du média d'un objet (on utilise ici le terme d'objet d'un point de

vue générique, il peut donc aussi bien s'appliquer à un contenu 3D qu'à un utilisateur) est définie comme étant un sous-espace dans lequel il est possible d'interagir avec ce média à un instant donné. La notion de conscience (ou awareness) entre deux objets se définit à travers les concepts de Focus et de Nimbus. Dans le cas des médias 3D, le Focus représente l'attention spatiale de l'objet. Le Nimbus d'un objet est quand à lui défini comme un sous-espace du média 3D dans lequel l'objet rend disponible des informations à son sujet.

Bien qu'il soit, comme relevé par Ouramdane dans son manuscrit de thèse [67], le modèle d'interaction spatiale de Benford peut être dans des cas particulier appliqué sous forme de fonctions mathématiques définissant le Nimbus, le Focus et l'awareness, celui-ci reste uniquement théorique si l'on reste dans une approche générale. Ce modèle est toutefois suffisant pour déduire un certain nombre de requis pour les médias 3D. En effet, si nous appliquons par exemple ces concepts à deux utilisateurs A et B qui collaborent dans un environnement 3D, la conscience entre A et B est définie par leur Focus et leur Nimbus. Par ailleurs, il est important de noter que, si l'utilisateur A interagit avec un objet C, le Nimbus de C fournit des informations sur l'activité de A à l'utilisateur B, ce qui augmente la conscience qu'a B de A. En pratique, cela signifie que la production de retours visuels partagés, lors de l'interaction des utilisateurs avec le média 3D, augmente la conscience des autres utilisateurs. La seule condition à cela est qu'il faut que tous les utilisateurs soient capable de mettre en relation ces retours visuels avec le contexte de travail partagé.

Pour finir, dans le cas spécifique d'une collaboration à distance et/ou en situation d'immersion dans un environnement virtuel collaboratif, il est nécessaire d'introduire une représentation 3D des utilisateurs (i.e. des avatars) pour palier au manque de communication visuelle entre les acteurs [58, 23, 9]. Si ce n'est pas naturellement le cas, il faut également mettre en place un moyen de communication vocale pour permettre aux utilisateurs de cordonner leurs actions.

1.4.2 Application au support de la collaboration par un média 3D

L'ensemble des requis généraux évoqués précédemment, ainsi que les spécificités propres à leur application au cas particulier des médias 3D, nous conduisent à la liste d'attendus suivantes pour un média 3D :

— **Contenus et artéfacts partagés.** De sorte à permettre l'établissement entre utilisateurs d'un contexte de travail partagé, il est nécessaire que ceux-ci visualisent un certain nombre de contenus 3D communs, ainsi que des artéfacts résultant de leurs interactions avec ces contenus. Ces artéfacts (ou retours visuels) partagés permettent à un utilisateur de comprendre l'activité des autres utilisateurs et de

la rapporter aux objectifs de travail communs.

— **Artefacts individuels.** Si il est bien nécessaire de partager des retours visuels informant l'ensemble des collaborateurs de l'activité d'un utilisateur, certaines activités de cet utilisateur sont individuelles. Ces activités individuelles sont liées aux tâches unitaires que doit effectuer seul l'utilisateur pour remplir un objectif (ou une tâche) nécessaire au travail commun. Il n'est en général pas souhaitable de partager des artéfact liés à la réalisation d'activités individuelles, sauf s'il permettent de donner des informations sur la réalisation d'un objectif partagé. Ces artéfacts individuels sont toutefois nécessaires pour l'utilisateur réalisant une activité individuelle. D'un point de vue plus technique, cela signifie que le concepteur d'un média 3D doit pouvoir définir dans quelles conditions un retour visuel est partagé, soit à l'ensemble des utilisateurs, soit uniquement à l'utilisateur dont l'action sur le média induit ce retour visuel.

— **Mécanismes de coordination.** L'objectif des mécanismes de coordination est l'organisation et la planification des travaux individuels et coopératifs au cours du temps. Ceci implique d'une part qu'un média 3D doit être capable de distribuer entre les utilisateurs un ensemble de tâches. Ceci se traduit par des interactions, potentiellement différentes, mises à disposition des utilisateurs. Le média 3D doit d'autre part être en mesure de faire évoluer dynamiquement cette distribution des tâches, soit en fonction de l'évolution du contexte de travail individuel et/ou partagé, soit suite à une action des utilisateurs (voir Malléabilité).

— **Malléabilité.** Comme définis précédemment, la malléabilité d'un système supportant le travail collaboratif se définit comme l'implication des utilisateurs dans la définition des mécanisme de coordination. Concrètement, cela veut dire que le système fournit aux utilisateurs des outils permettant l'adaptation des mécanismes de coordination du système. La notion de niveaux de malléabilité théorisée par Dourish dans [29] se transcrit donc par l'importance des adaptations que peuvent apporter les utilisateurs aux mécanismes de coordination. Par exemple, un système permettant aux utilisateurs de sélectionner le mode de fonctionnement des mécanismes de coordination parmi une liste de fonctionnements préconçus sera considéré comme faiblement malléable. A l'inverse, un système permettant aux utilisateurs de modifier à tout moment l'attribution de chaque tâche (c'est à dire de chaque interaction) sera considéré comme hautement malléable.

— **Rôles utilisateur.** La notion de rôle utilisateur est directement liée au processus coopératif supporté par le système. L'implémentation par média 3D de rôles pour les utilisateurs consiste à définir, à partir des règles métier propres au processus, quels contenus 3D doivent et/ou peuvent être visualisés par un certain type d'utilisateur. De même, le rôle de l'utilisateur implique potentiellement des restrictions dans l'accès à certaines interactions. Ceci induit des contraintes à respecter pour les

mécanismes de coordination du système, et par extension des limitations pour la malléabilité du système. En effet, ni les mécanismes de coordination, ni leur adaptation par les utilisateurs, ne doivent conduire à une répartition des tâches contraires aux règles métier définies par le processus. Il est à noter que dans certains cas, un utilisateur peut cumuler plusieurs rôles ou en changer durant l'utilisation du système.

— **Représentation des utilisateurs.** L'un des requis propres à la collaboration en environnement 3D consiste à permettre aux utilisateurs de dispositifs immersifs (c'est principalement le cas des supports de RV) de visualiser les autres utilisateurs et leur mouvement. Cette fonctionnalité est notamment nécessaire pour qu'un utilisateur ait conscience des autres utilisateurs et de leurs activités. De manière concrète, ceci implique que le média 3D soit informé par les supports du mouvement des utilisateurs de sorte à pouvoir restituer visuellement cette information aux autres utilisateurs (typiquement en animant des avatars reproduisant les mouvements des utilisateurs).

— **Communication vocale.** De même qu'une représentation des utilisateurs est nécessaire lorsqu'ils ne peuvent pas se voir de manière naturelle, le média 3D doit fournir un moyen de communication vocale aux utilisateurs si la collaboration est distante.

Chapitre 2

UMI3D : Proposition d'un système de gestion d'interface utilisateur pour les média 3D

Ce chapitre a pour objet la proposition d'une nouvelle architecture pour les médias 3D. Cette architecture s'appuie sur un protocole de communication web que nous avons appelé UMI3D pour permettre à tout support de RA/RV d'interagir avec les médias 3D l'implémentant.

Nous revenons dans un premier temps sur les principes d'architecture logicielle propres aux systèmes de gestion d'interface utilisateur (UIMS), et expliquons comment nous les avons appliqué aux médias 3D de sorte à proposer une architecture séparant les paradigmes de l'interaction homme-machine. Nous décrivons dans un second temps la manière dont nous avons, dans la conception du protocole UMI3D, utilisé les travaux existants concernant la manipulation de degrés de liberté, pour définir un modèle de données permettant une description agnostique des interactions possible avec un média 3D. Enfin, nous présentons dans un troisième temps une preuve de concept dans laquelle nous relatons la manière dont nous avons réalisé une première implémentation du protocole UMI3D et de l'architecture proposée.

2.1 Les systèmes de gestion d'interface utilisateur

2.1.1 Notion de UIMS.

Le domaine des systèmes de gestion des interfaces utilisateur ou UIMS (pour User Interface Management System) a pour but principal la recherche et la conception d'outils de développement permettant de simplifier la création d'interfaces utilisateur. Bien que la définition exacte d'un UIMS varie fortement dans la littérature [5] depuis la première utilisation du terme par Kasik [47], on considère actuellement communément un UIMS comme un mécanisme permettant de séparer la logique métier des interfaces utilisateur lors de la conception d'un logiciel [63], ou plus généralement d'un système d'information. Les exemples de UIMS les plus connus et appliqués sont les patrons d'architecture logiciel [18] Model-View-Controller (MVC) et son extension Model-View-Presenter (MVP) qui

sont couramment utilisés dans le domaine du développement web [57]. Concrètement, ces modèles permettent de séparer la définition des structures de données, les règles métier régissant le comportement de l'application, et l'interface utilisateur. L'intérêt principal étant de pouvoir réutiliser les structures de données et règles de gestion dans plusieurs modes de présentation, comme par exemple des interfaces utilisateur dédiées à différents supports (page web, ordinateur de bureau, smartphone, etc...).

2.1.2 Application aux médias 3D.

Dans le but de simplifier la création de médias 3D multi-supports, il nous semble nécessaire de faire une revue des systèmes de gestion d'interface utilisateur existants dans les domaines de la Réalité Virtuelle et de la Réalité Augmentée. L'ensemble des outils et standards précédemment présentés dans notre état de l'art peut correspondre à la large définition des UIMS. Toutefois, il n'existe à notre connaissance, pas d'équivalent aux patrons de conception qui permettrait de concevoir un média 3D et ses interactions séparément des interfaces utilisateur propres aux différents supports. La principale question à résoudre pour proposer ce type de modèle consiste à trouver comment abstraire le média 3D du support, sans pour autant réduire les possibilités d'interaction des utilisateurs finaux au plus grand dénominateur communs des fonctionnalités des dispositifs.

Approche naïve pour créer une application 3D. L'approche la moins structurée pour créer des applications 3D est résumée Figure 2.1. Elle consiste à développer conjointement (dans un même script si l'on utilise un moteur 3D) l'interface utilisateur, la logique métier de l'application, et la gestion du contenu 3D. Si cette approche est parfois suffisante pour créer une application simple, destinée à un support unique, elle ne convient pas pour la conception des applications multi-supports. En effet, l'interface utilisateur n'étant pas séparée du contenu et de la logique métier, l'ensemble de l'application doit être redéveloppée pour chaque support. Au delà des temps de développement accrus, cela pose également un problème de maintenabilité des différentes applications : Chaque modification de la logique de l'application doit être répliquée spécifiquement pour chacun des supports, ce qui entraine des risques d'erreurs et des différences de comportement entre les différents supports.

Application des principes de Model-View-Presenter aux applications 3D. Une méthodologie de conception plus avancée, présentée Figure 2.2, consiste à appliquer les principes des patrons d'architecture logicielle comme Model-View-Presenter aux applications 3D. Il s'agit de séparer clairement, dans des scripts ou des modules différents, le code informatique gérant la manipulation du contenu, la logique métier de l'application, et l'interface utilisateur. Ceci permet, si l'on utilise un moteur 3D supportant plusieurs

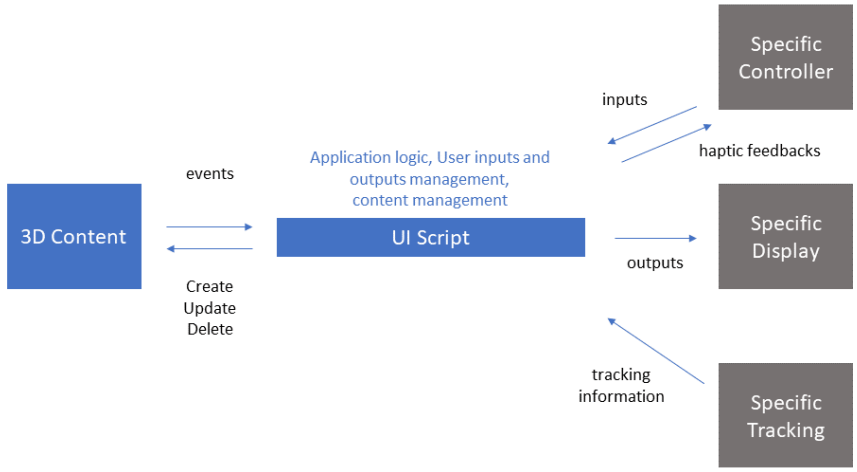


FIGURE *2.1 – Architecture naïve pour les applications 3D*

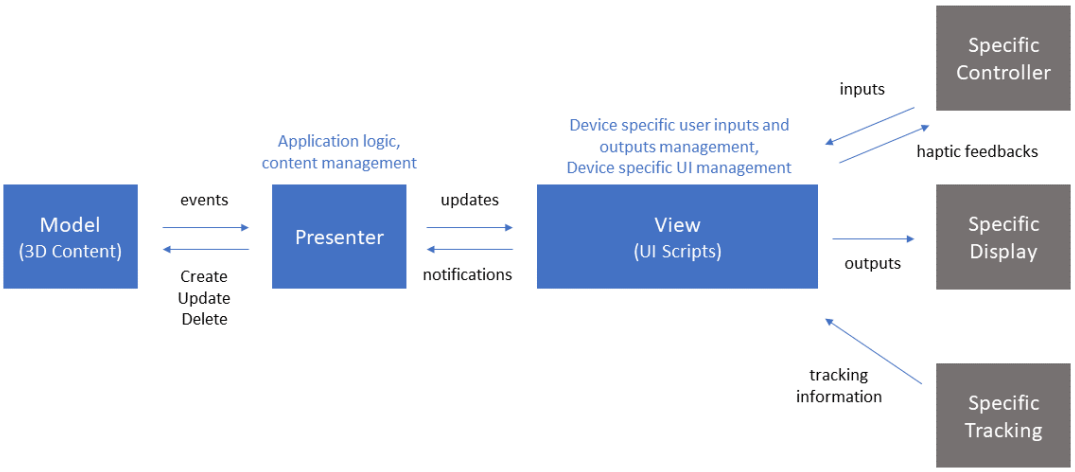


FIGURE *2.2 – Application des principes de Model-View-Presenter aux applications 3D*

supports comme par exemple Unity, de partager entre les supports une même implémentation de la gestion du contenu et de la logique métier. Cette approche présente deux avantages principaux par rapport à l'approche 'naïve' : Premièrement le comportement de la logique métier et de la gestion du contenu est garanti identique sur tous les supports. Deuxièmement, les développements spécifiques aux supports sont limités aux seules interfaces utilisateur, ce qui réduit sensiblement le coût en temps de développement du multi-support. Toutefois, une mise à jour de la logique métier, ou du système de gestion du contenu, implique toujours de recompiler, et de redistribuer l'application pour chaque support. De même, une modification significative de la logique métier peut impliquer la modification des interfaces utilisateurs, donc des développements spécifiques aux supports supplémentaires.

Positionnement des standards existants. Les standards et middleware existants pour le développement multi-support en RV et RA sont principalement tournés vers la

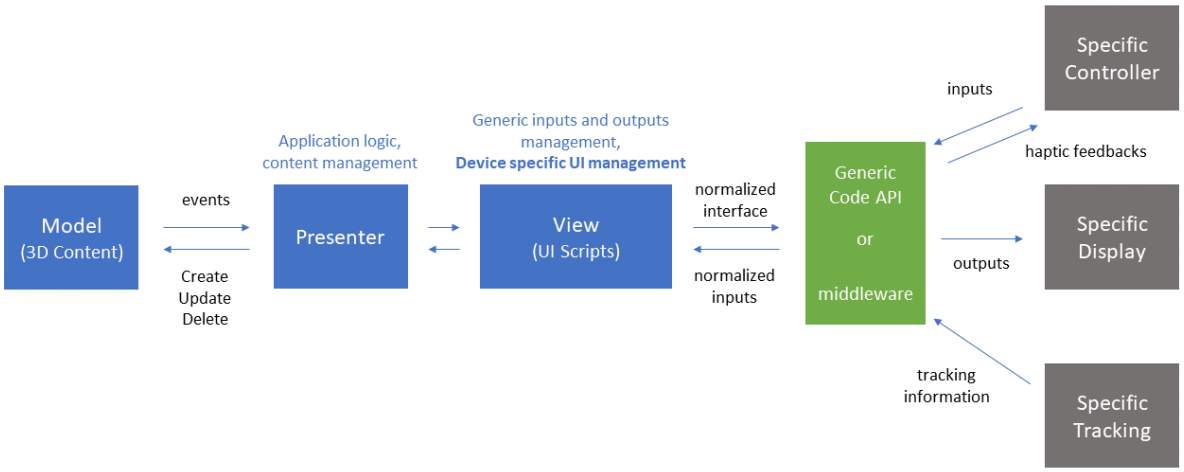


FIGURE *2.3 – Positionnement des standards et outils existants (Middle VR, WebVR, OpenXR, etc.)*

création d'une interface unique pour plusieurs supports. Cette méthode, lorsqu'elle est, comme proposé Figure 2.3, combinée avec les principes du patron d'architecture MVP, permet de partager une même version de l'application à plusieurs supports. Toutefois, comme développé dans notre état de l'art (Chapitre 1), ces outils ne permettent pas de créer une unique application pour l'ensemble des supports sans réduire l'interface aux seules fonctionnalités communes aux supports. Cette approche permet toutefois de partager une même interface, et donc une même implémentation, entre des dispositifs aux propriétés et aux périphériques semblables. Par exemple, il serait ainsi possible d'avoir une implémentation unique pour l'ensemble des casque de RV (Hmd) du marché. Cette approche réduit donc encore sensiblement le nombre total d'implémentations, et donc les temps de développement requis par une application multi-support. Toutefois, aucuns des inconvénients de l'approche MVP ne disparait totalement : Une mise à jour de la logique métier de l'application entraine toujours un besoin de recompiler, et redistribuer l'application pour chacun des supports, et peut toujours demander des mises à jour des interfaces propres aux différentes catégories de supports. Enfin, il faut noter que cette problématique, est encore plus marquée pour un parc d'applications. Par exemple, si un éditeur d'applications multi-supports souhaite rendre disponibles ses applications sur un nouveau support, et que ce dispositif est d'un nouveau type, alors cet éditeur doit concevoir une nouvelle interface pour chaque application. Il doit par ailleurs dans tous les cas recompiler toutes les applications pour ce support, ce qui peut représenter des temps de déploiement très importants.

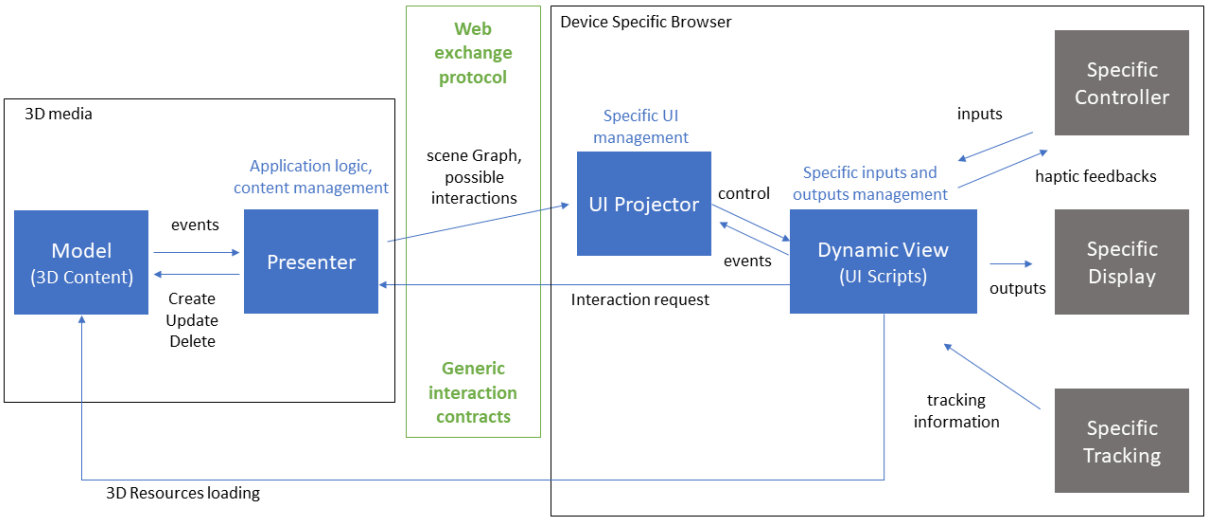


FIGURE *2.4 – Architecture proposée pour des médias 3D permettant la collaboration multi-supports en RV/RA.*

2.2 Séparation des paradigmes de l'interaction homme-machine

Dans le but de proposer une nouvelle architecture des médias 3D permettant une abstraction des supports, et ne réduisant pas l'interaction à l'utilisation des seules fonctionnalités communes aux supports, nous proposons d'étudier la manière dont s'interfacent les différents paradigmes de l'interaction dans les applications 3D. Nous sommes en effet convaincus, qu'une telle abstraction des supports est possible à condition d'isoler, d'un point de vue architectural, les paradigmes de l'interaction et de normaliser leurs échanges.

2.2.1 Proposition d'une nouvelle architecture des médias 3D séparant les paradigmes de l'interaction

Si l'on applique les paradigmes de l'interaction aux applications de RA et RV basées sur l'approche Model-View-Presenter (Figure 2.2), les modules Model et Presenter appartiennent respectivement aux paradigmes 'computer-as-a-media' et 'computer-as-a-tool'. La classification de la partie Presenter, qui contient la logique métier et gère la mise à jour de l'interface est moins évidente. En effet, si la conception des règles métier de l'application fait partie du média, la gestion des mises à jour de l'interface ne se fait aujourd'hui pas, ou rarement, indépendamment des fonctionnalités apportées par l'interface. Notre point de vue est qu'une séparation architecturale claire entre les paradigmes 'computer-as-a-tool' et 'compuper-as-a-media' permettrait de créer des médias 3D en totale indépendance des supports qui les manipulerons. Ceci passe par une standardisation des échanges 'media'-'tool' qui ne soit pas basée sur les fonctionnalités supposées des supports.

Dans le but de mettre en œuvre la séparation des paradigmes de l'interaction 'computer-as-tool' et 'computer-as-a-media' évoquée dans le paragraphe précédent, nous proposons une nouvelle architecture des applications 3D. Cette architecture, détaillée Figure 2.4, se base sur l'utilisation d'un protocole d'échange web pour la communication entre un média 3D et un support de RV/RA quelconque. Ce protocole, que nous avons appelé UMI3D pour 'Unified Model for Interaction in 3D environment' [19], constitue un modèle d'abstraction du dispositif basé sur les interactions possibles avec le média 3D. Le média 3D établit des contrats d'interaction génériques qui sont transmis aux différents supports grâce à un modèle de données qui sera présenté ultérieurement. Chaque support doit disposer d'un navigateur capable de réaliser ces contrats. Pour cela, le navigateur interprète les contrats d'interaction, en générant et en mettant à jour dynamiquement une interface utilisateur adaptée au support. Cette étape constitue de notre point de vue une opération de projection qui permet de répartir les contrats d'interaction reçus entre les interfaces graphiques (menus, boutons, formulaires, etc) et les périphériques d'entrées utilisateur (souris, clavier, contrôleurs 6-D, etc). Par ailleurs, la réaction résultant de la réalisation d'un contrat d'interaction, est à la charge du média 3D (via son module presenter) car définie par sa logique métier. Le principal avantage selon nous de cette architecture, est qu'elle permet dans une certaine mesure, de reproduire le comportement observé entre les objets réels et les outils physiques. En effet, le support de RV/RA constitue, avec son navigateur, un outil qui peut être réutilisé pour interagir avec différents médias 3D. Ceux-ci réagiront différemment en fonction de leur logique métier et de leurs interactions possibles. Il est par ailleurs à noter que si la standardisation de la communication 'tool-media' permet techniquement à tout support d'interagir avec tout média 3D, ce n'est pas forcément judicieux. En effet, de même qu'il est possible, mais inadapté d'utiliser une pièce de monnaie pour resserrer une vis, certain supports de RV/RA resteront plus efficace que d'autres pour interagir avec un média 3D. Le second avantage de l'architecture proposée est quelle permet de réduire à une unique application (un navigateur dédié) la connexion d'un nouveau support de RV ou de RA à n'importe quel média 3D existant, et ce sans avoir à déployer ces médias spécifiquement pour le support en question (i.e. grâce à la connexion web au média). De même, l'évolution d'une application métier, c'est à dire d'un média, ne nécessite pas de recompiler cette application pour chaque support. Ce média est automatiquement à jour pour tout support s'y connectant après la modification. Enfin, les médias 3D conçu pour la collaboration temps réel avec l'architecture proposée, permettent nativement la collaboration entre supports de RV ou de RA différents.

2.2.2 Complémentarité avec les standards existants

Il est d'autre part à noter que l'architecture proposée dans le paragraphe précédent peut se combiner avec les standards de développement multi-supports existant. On peut

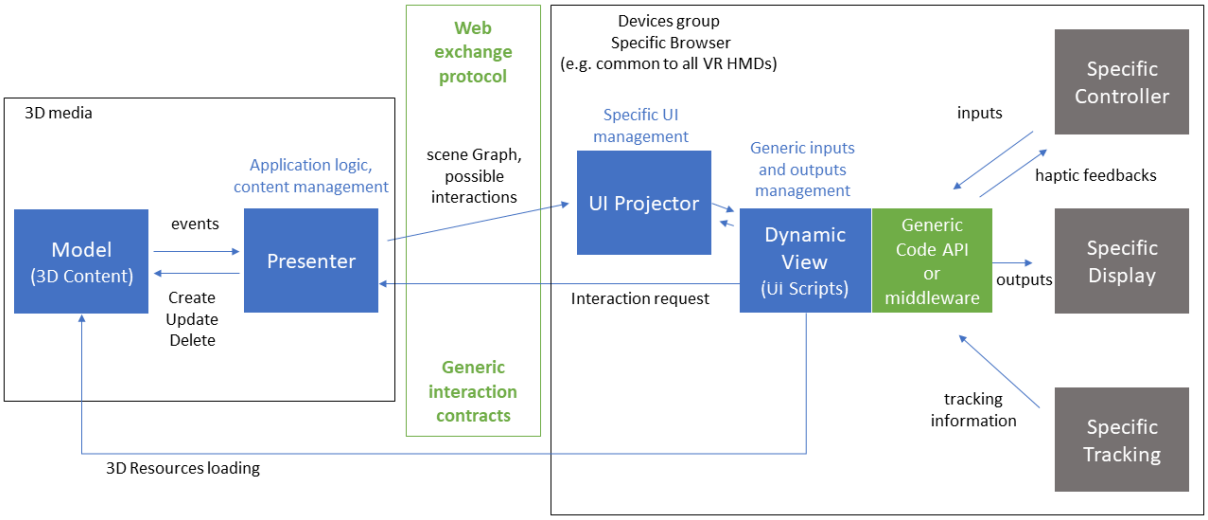


FIGURE *2.5 – Combinaison avec les approches existantes*

ainsi réduire le nombre de navigateurs UMI3D à développer. En effet, ces outils sont tout à fait appropriés pour créer une même application pour des supports semblables. Par exemple, l'utilisation du standard OpenXR pourrait permettre de concevoir un unique navigateur pour plusieurs Hmd de RV, et d'y connecter pour l'interaction tout contrôleur à six degrés de liberté. On obtiendrait alors l'architecture globale présentée Figure 2.5.

2.3 Modèle d'abstraction de UMI3D

Cette section reprend et étend la présentation originelle du modèle d'abstraction de UMI3D introduite dans [19]. Le modèle d'abstraction en question permet, à un média 3D distant, d'établir des contrats d'interaction avec tout support de RV/RA, sans connaissance de la nature du support et de ses fonctionnalités.

2.3.1 Principales tâches en interaction 3D

On considère communément que les interactions entre un utilisateur et un environnement 3D peuvent être classées selon quatre types de tâches : la manipulation, la sélection, le contrôle d'application et la navigation. Cette classification, illustrée Figure 2.6, est généralement attribuée Bowman [16].

— **Sélection.** La tâche de sélection permet à un utilisateur de désigner un ou un ensemble d'objets 3D. Cette tâche se décompose généralement en deux étapes : une phase de pointage (ou de désignation) durant laquelle l'utilisateur désigne un ou des objets, et une action de validation permettant à l'utilisateur de confirmer sa volonté d'interagir avec le ou les objets.

— **Manipulation.** La tâche de manipulation constitue le principal moyen pour un utilisateur d'agir sur un ou un ensemble d'objets 3D. Elle permet, après la sélec-

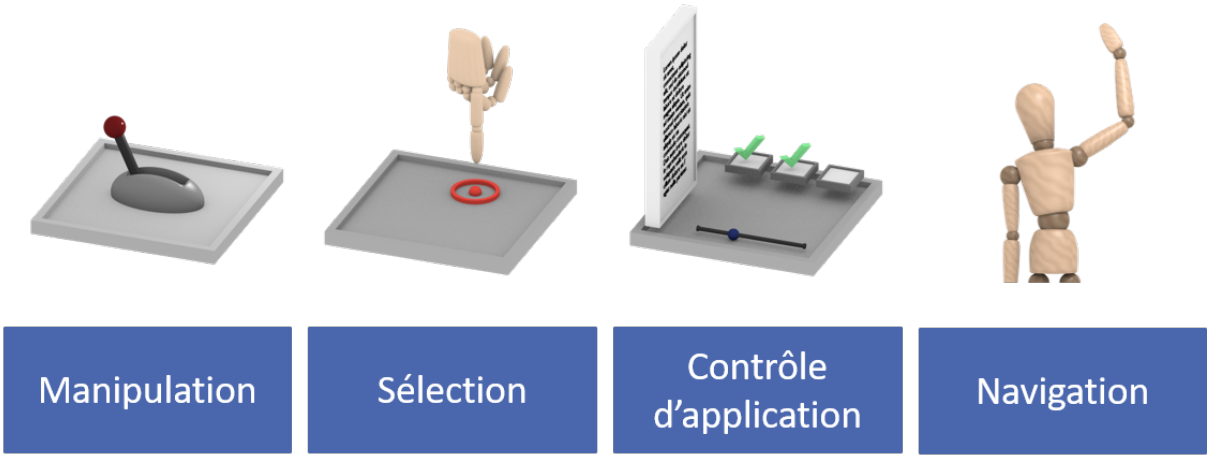


FIGURE *2.6 – Classification des tâches en interaction 3D.*

tion de(s) l'objet(s) 3D d'agir sur ses paramètres (position, orientation, couleur, comportement, ...). Si cette tâche inclut théoriquement le paramétrage de toute propriété de l'objet, certaines études [43, 88, 89, 87] semblent limiter la notion de manipulation à l'exécution par l'utilisateur de tâches de translation et/ou de rotation à un ou plusieurs degrés de liberté. Par ailleurs, il peut parfois être question de manipulation lorsque la configuration d'un paramètre de l'application implique la manipulation de degrés de liberté.

— **Navigation.** La navigation désigne l'ensemble des outils et méthodes permettant d'une part à l'utilisateur de se déplacer dans un environnement 3D, et d'autre part de calculer la position de l'utilisateur par rapport à un point fixe de l'environnement 3D.

— **Contrôle d'application.** La notion de contrôle d'application désigne de manière générale l'interaction de l'utilisateur avec les paramètres globaux de l'application. On y inclut généralement toutes tâches ne se rapportant à aucunes des trois autres tâches, et permettant la communication entre l'utilisateur et le système : menus graphiques, sélection d'outils, commandes gestuelle, commandes vocales.

2.3.2 Interactions avec un média 3D

2.3.2.1 Principe des contrats d'interaction

Le fonctionnement du protocole d'échange UMI3D repose principalement sur l'établissement de contrats d'interaction entre le média 3D et les différents navigateurs UMI3D propres à chaque support de RV/RA. Il s'agit ici de contrats d'interface : le média décrit au navigateur une interface web lui permettant d'agir sur certains de ses paramètres ou sur les paramètres des contenus qui le composent. Le navigateur doit alors générer une interface permettant à l'utilisateur d'émettre une requête (ou demande) d'interaction à

destination de cette interface. Le fonctionnement détaillé de ces contrats est le suivant :

- **Chargement de scène et description des interactions possibles.** Comme présenté Figure 2.7, le média 3D communique avec le support grâce à un canal de communication web bidirectionnel établi entre le média et le navigateur UMI3D dédié au support. Cette connexion permet, dans un premier temps, le chargement par le navigateur de la scène 3D dans son état initial. Le média 3D décrit alors au navigateur un ensemble de contrats d'interaction qui définissent les interactions possibles à cet instant pour l'utilisateur.
- **Génération d'une interface utilisateur.** contrats sont définis par la transmission, standardisée au format JSON, d'une combinaison d'objets que nous appelons des briques d'interaction. Ces briques constituent un nombre réduit de classes permettant de décrire les interactions possibles avec le média. Lors de la réception d'un contrat d'interaction par le navigateur, celui-ci effectue une opération dite de projection, pour générer ou mettre à jour l'interface utilisateur spécifique au support, et ainsi rendre possible la réalisation par l'utilisateur du contrat d'interaction.
- **Action de l'utilisateur.** Lors de la manipulation de l'interface générée par l'utilisateur, le média 3D est notifié en temps réel, c'est à dire à une fréquence similaire à celle de la boucle de rendu du support (typiquement 30 à 90 Hz), de la manière dont l'utilisateur remplit le contrat. Le média peut alors, en fonction des règles métier qui le régissent, produire une réaction à l'action de l'utilisateur.
- **Réaction du média 3D.** Lors de la réalisation ponctuelle (message unique) ou continue (30 à 90 Hz) d'un contrat d'interaction, le média 3D peut produire une réaction. Celle-ci peut être une manipulation du contenu de la scène 3D, qui est alors notifiée en temps réel (typiquement 30 à 90 Hz) aux différents supports connectés au média. Il peut également s'agir d'une modification des contrats d'interaction mis à la disposition de l'utilisateur.

Collaboration temps réel. La notification en temps réel (typiquement 30 à 90 Hz) à tous les supports connectés au média 3D permet la synchronisation de la scène 3D entre différents utilisateurs. L'architecture proposée permet donc nativement l'interaction collaborative de plusieurs supports avec le média 3D. La collaboration n'est ici garantie que d'un point de vue technique, c'est à dire sans la garantie que le système réponde bien aux attendus d'un système de support du travail collaboratif.

Scénario interactif du média 3D. La manière dont évoluent en temps réel le contenu 3D et la répartition entre les utilisateurs des interactions possibles avec le média, constitue le scénario interactif du média 3D. Les outils de conception mis en place pour définir ce scénario ne seront pas présentés plus en détail ici, mais feront par la suite l'objet du Chapitre 3. Nous nous concentrerons ici principalement sur la description des différentes

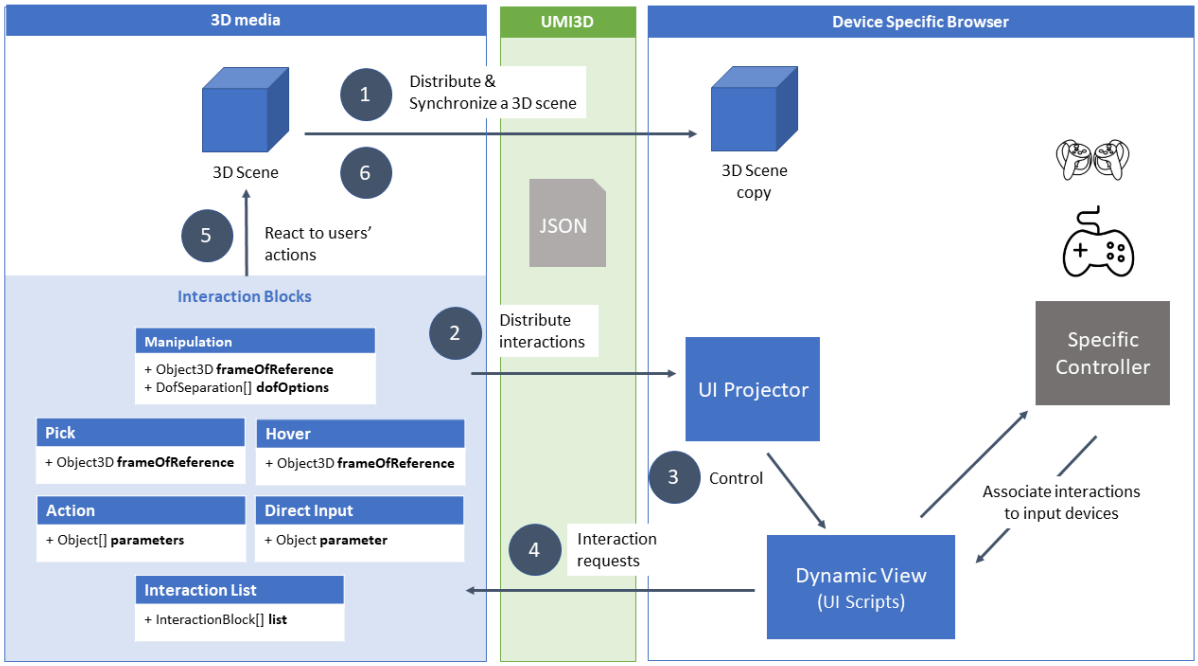


FIGURE *2.7 – Fonctionnement du protocole UMI3D : définition de contrats d'interaction avec un support quelconque à l'aide d'un nombre finis de briques d'interaction.*

briques d'interaction.

2.3.2.2 Briques d'interaction

La principale difficulté pour concevoir le protocole UMI3D a consisté à définir un modèle de données, agnostique des différents supports, pour définir les interactions possibles avec un média 3D. Pour permettre aux différents supports d'exploiter tous leurs périphériques d'entrées et toutes leurs fonctionnalités, il fallait que ce modèle ne s'appuie aucunement sur les fonctionnalités et périphériques supposés des supports. Par exemple, la notion de click est associée à l'utilisation d'une souris, ou plus largement d'un périphérique de pointage, disposant d'un actionneur physique (un bouton par exemple). On préférera donc, dans le cadre du protocole UMI3D, décrire une tâche de sélection qui est une interaction agnostique. Nous avons donc établis une liste des types d'interactions possibles avec un média 3D. Chaque type d'interaction constitue une brique d'interaction, qui se matérialise sous la forme d'une classe dont les instances peuvent être transmises aux supports dans le format JSON. Pour finir, nous avons organisé ces briques d'interactions en trois catégories : pointage et sélection, manipulation, et contrôle d'application.

Comme expliqué dans la présentation de l'architecture logicielle propre à UMI3D, nous séparons d'interface utilisateur, générée dynamiquement par un navigateur UMI3D, de la définition des interactions possibles avec le média 3D. Ceci implique que la classification des tâches de Bowman [16] ne s'applique pas directement à nos briques d'interaction. En effet, si l'interface générée par un navigateur UMI3D propose bien à l'utilisateur toutes

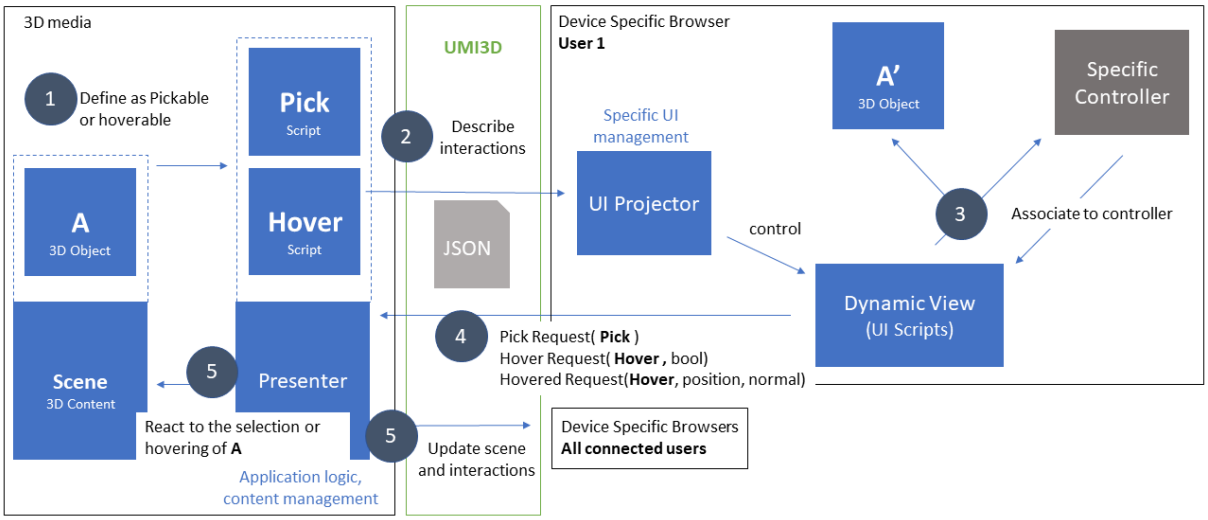


FIGURE *2.8 – Fonctionnement des briques d'interaction Pick et Hover permettant la selection et le pointage d'objets 3D.*

ces interactions, celles-ci ne se retrouvent pas forcément dans les données échangées entre le média et le navigateur. Par exemple, les menus graphiques permettant à l'utilisateur de sélectionner un outil ou une tâche sont générés par le navigateur. Leur aspect n'est donc pas décrit par le média 3D.

2.3.2.3 Pointage et sélection

Le pointage et la sélection d'objets 3D sont des interactions parmi les plus courantes en environnement 3D. Toutefois, la manière de désigner ou de sélectionner un objet 3D diffère fortement d'un support à l'autre. En effet, même si la métaphore de la main virtuelle [80] est couramment utilisée pour la sélection avec les mains, de même que la métaphore du rayon virtuel [13] (Ray-Casting) est répandue pour la sélection distante, d'autres techniques existent. D'autre part, l'implémentation d'une même technique peut varier d'un support à l'autre. Par exemple dans le cas de la technique de Ray-Casting, la manière de manipuler le rayon lancé dépend totalement de la nature du périphérique d'interaction (souris, contrôleur 6-D). Notre point de vue est donc que le choix de la technique de sélection doit être laissé à la discrétion du navigateur.

Pour permettre la définition de tâches de pointage et de sélection, nous définissons donc deux briques d'interaction, respectivement Hover et Pick. Le fonctionnement de ces briques est détaillé Figure 2.8 [1] : Lors de la conception du média 3D, on associe une brique pick et/ou hover à un objet 3D de la scène (1), ainsi qu'une réaction potentielle sur la scène et/ou sur les interactions à disposition de l'utilisateur (5). Ces briques sont ensuite transmises au navigateur (2). Celui-ci associe alors, au pointage et/ou à la sélection de la copie de l'objet 3D par le dispositif de pointage du support (3), la notification d'une

1. les (#) sont ceux de la figure.

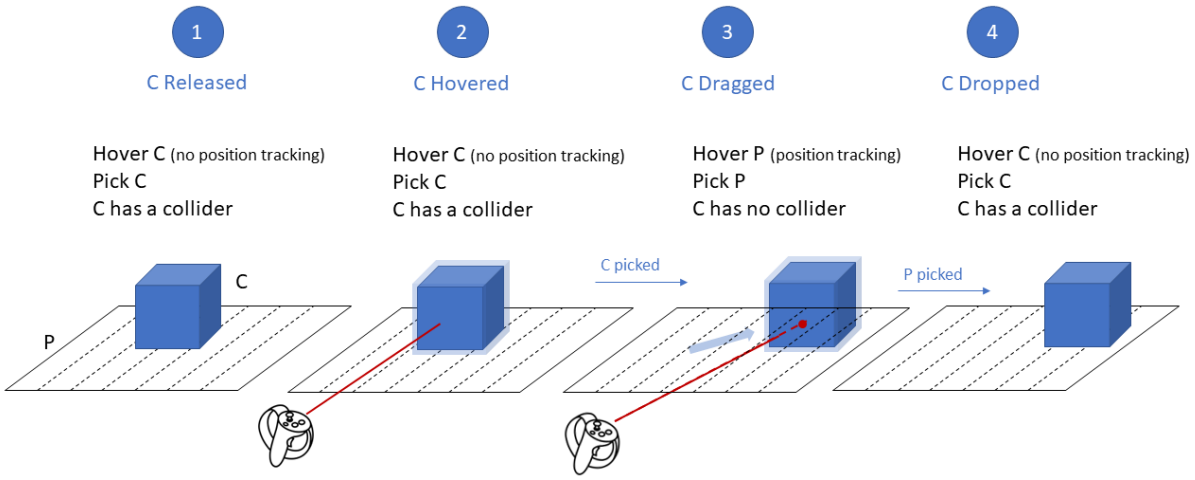


FIGURE *2.9 – exemple : glissé-déposé sur un plan 2D.*

demande d'interaction au média 3D (4). Le média 3D réagit en actionnant les réactions définies précédemment (5). Si la notification d'une demande de sélection n'a pas d'autre argument que la brique Pick associée, nous introduisons deux types de requêtes pour le pointage (i.e. pour la brique Hover). La première d'entre elle (Hover Request) notifie ponctuellement le média du début ou de la fin d'un pointage de l'objet. La seconde (Hovered Request) permet, durant le pointage, d'informer le média de la position pointée sur la surface de l'objet, ainsi que de la normale à la surface en ce point. Pour limiter les flux de données, le paramétrage de la brique Hover permet au concepteur du média d'utiliser ou non ce deuxième évènement.

Exemple : implémentation d'une fonctionnalité de glissé-déposé sur un plan 2D. Pour illustrer les utilisations possibles des briques d'interaction Pick et Hover, on peut par exemple concevoir une fonctionnalité de glissé-déposé d'objet 3D en utilisant uniquement ces deux briques. Comme présenté Figure 2.9, cet exemple nécessite à minima deux objets : un plan 2D (P) et un objet 3D (le cube C). À l'état initial (1), deux briques d'interaction Pick et Hover permettent à l'utilisateur d'avoir un retour visuel au survol de C (2), et de le sélectionner (transition 2-3). Lors de la sélection de C, les briques Pick et Hover liées à C sont remplacées par des briques Pick et Hover liées au plan P. De plus, le retour visuel produit par le survol précédent de C est conservé pour la durée de la sélection. Enfin, la boîte de collision de C est supprimée pour permettre le survol de P à travers C. L'ensemble de ces transformations permet d'entrer dans la phase de glissé-déposé (3) durant laquelle la position pointée sur la surface de P est synchronisée en temps réel avec la position de C. Lors de la sélection de P (transition 3-4), les interactions reviennent à leur état d'origine, et le retour visuel produit sur C est supprimé.

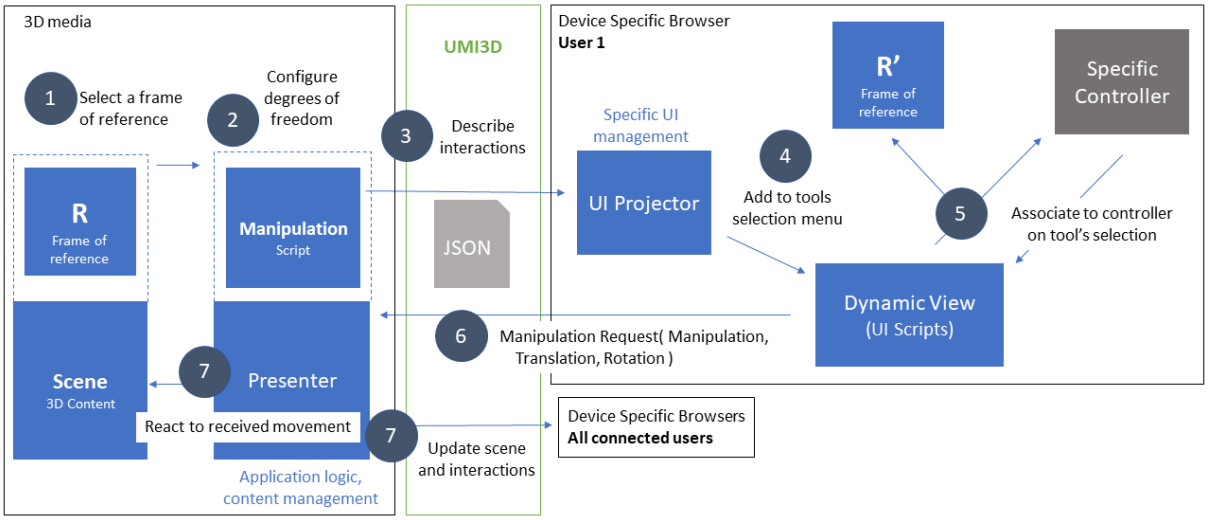


FIGURE 2.10 – *Fonctionnement de la brique manipulation.*

2.3.2.4 Manipulation par contrats de mouvement

Intérêt à utiliser les degrés de liberté pour l'abstraction du dispositif. La réalisation par l'utilisateur de tâches à un ou plusieurs degrés de liberté permet la majorité des interactions avec le contenu d'un environnement 3D. Habituellement, on associe directement à la manipulation des degrés de liberté d'un périphérique d'entrée, une manipulation des degrés de liberté d'un objet virtuel. Dès 1994, les travaux de R.J.K. Jacob & al. [43] ont montré que le dispositif, dont la structure de contrôle est la plus proche de la structure perceptuelle de la tâche réalisée, est le plus adapté pour exécuter cette tâche. Dans des travaux plus récents, M. Veit & al [88] ont montré que les utilisateurs décomposent instinctivement certaines tâches complexes dans le but de les simplifier. De plus, cette décomposition varie au cours de la tâche en fonction de la vitesse d'exécution et/ou de la précision souhaitée(s) [89]. Enfin, il apparaît qu'une séparation de certains degrés de liberté lors de la conception de l'interaction conduit à de meilleures performances de l'utilisateur si cette décomposition est conforme à la manière dont l'utilisateur simplifierait de lui-même la tâche [88, 87]. L'ensemble de ces travaux nous pousse à penser qu'il serait possible de décrire la manipulation d'un objet 3D uniquement en décrivant la structure de la tâche par ses degrés de liberté. Il nous paraît par ailleurs approprié de compléter cette description par une information sur la manière dont les degrés de liberté doivent idéalement être intégrés/séparés lors de la réalisation de la tâche.

Brique Manipulation. La brique d'interaction manipulation, dont le fonctionnement est détaillé Figure 2.10 permet la manipulation d'objets 3D, ou plus généralement de paramètres à plusieurs degrés de liberté. Pour concevoir une tâche de manipulation, on associe une brique manipulation à un objet 3D de la scène qui définit le référentiel de la manipulation (1), ainsi qu'une réaction potentielle sur un ou plusieurs éléments de la scène

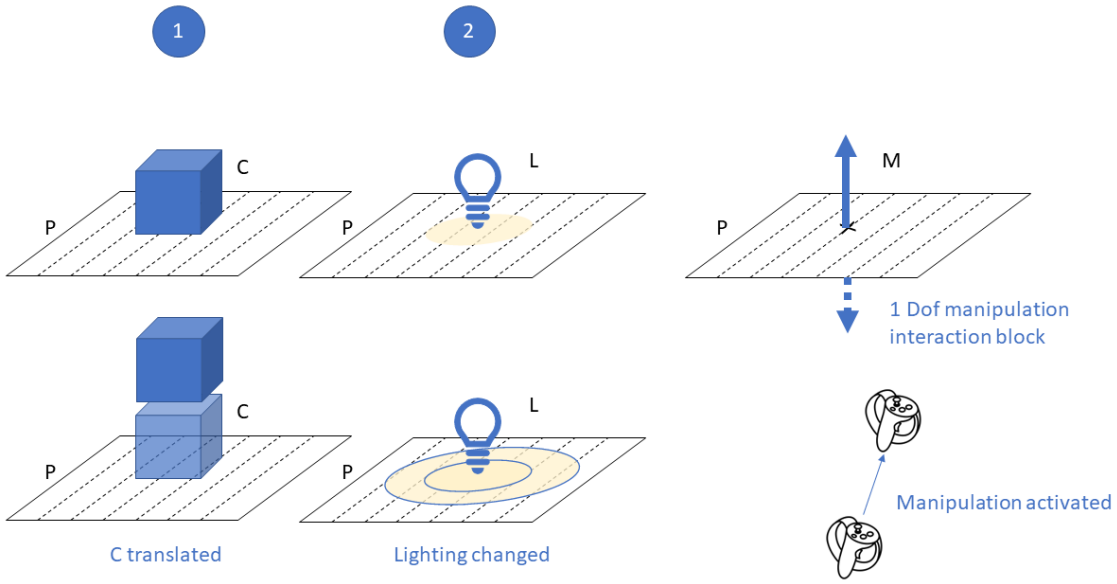


FIGURE *2.11 – Manipulations à un degré de liberté.*

(7). La configuration des degrés de liberté de la manipulation (2) permet d'une part de définir les degrés de liberté de la tâche, et d'autre part, de proposer plusieurs possibilités de séparation/intégration de degrés de liberté. Le fait que le support utilisateur soit supposé inconnu à la conception demande en effet de proposer des décompositions alternatives de la tâche pour le cas où le support serait incompatible avec un regroupement de degrés de liberté. Lors de la transmission de la brique au navigateur (3), des menus sont générés pour permettre à l'utilisateur d'activer l'outil (i.e. la manipulation) reçu. Lors de la sélection de l'outil (5), le navigateur associe, à la manipulation d'un périphérique, l'envoie au média 3D d'une demande de manipulation (6). Le média 3D réagit alors en manipulant un ou plusieurs éléments de la scène (7).

Exemple : manipulations à un degré de liberté. Le principal intérêt du modèle d'abstraction du support mis en place consiste à pouvoir décrire et réaliser une grande variété d'interactions à partir d'un nombre réduit de classes (i.e. les briques d'interaction). On peut donc comme présenté Figure 2.11 définir des réactions différentes pour l'activation d'une même brique d'interaction. Dans cet exemple, nous définissons deux média 3D contenant tout deux un plan P, et une manipulation à un degré de liberté définissant une translation orthogonale à P dans le référentiel de P. Dans le cas (1), le mouvement transmis par le support à l'activation de la manipulation est directement appliqué au cube C. Il en résulte une translation de C orthogonalement à P. Dans le cas (2) le mouvement transmis est utilisé pour contrôler la portée de la source de lumière L.

FIGURE *2.12 – Brique d'interaction manipulation et contrôle de l'intégration et de la séparation de degrés de liberté.*

Intégration et séparation des degrés de liberté. Dans le but de permettre au concepteur du média 3D de contrôler l'intégration ou la séparation des degrés de liberté, la brique manipulation contient un système de regroupement des degrés de liberté à options multiples présenté Figure 2.12. Ceci permet d'une part de demander la séparation de degrés de liberté, lorsque cette séparation implique un gain de performance comme avancé dans [88, 87]. Cela permet d'autre part d'indiquer aux supports, ne permettant pas d'intégrer tous les degrés de liberté requis, des alternatives pour la décomposition de ces degrés de liberté. Par exemple, on indique ainsi à un dispositif souris-clavier comment décomposer une translation à trois degrés de liberté en une translation 2D et une translation 1D. On obtient alors le comportement présenté Figure 2.13. Dans cet exemple, la configuration souhaitée (option 1) est une intégration des trois degrés de liberté de la tâche de translation. Dans le cas de l'utilisation d'un périphérique à trois ou six degrés de liberté, le navigateur UMI3D associe à la translation en trois dimensions du périphérique, l'activation conjointe des trois degrés de liberté. Pour les périphériques à moins de trois degrés de liberté, l'intégration des degrés de liberté demandée par l'option 1 n'est pas réalisable. Le concepteur du média 3D définit donc une option 2 demandant l'intégration des degrés de la translation dans le plan horizontal, et la séparation de la translation verticale. Dans ce cas, le navigateur UMI3D génère un menu permettant de sélectionner soit la translation planaire, soit l'axe vertical.

2.3.2.5 Contrôle d'application

Le contrôle d'application regroupe tous les moyens mis à disposition de l'utilisateur pour paramétrer l'application et/ou sélectionner des outils : menus, boutons, saisie de texte. La création du média 3D se faisant sans connaissance préalable des supports, il n'est pas possible de prévoir la nature de l'interface utilisateur. En effet, l'interface la plus adaptée peut varier en fonction des caractéristiques du dispositif (dimensions de l'affichage, taille de l'écran, ...). Nous proposons de décrire, de manière générique, les moyens de paramétrage par deux briques d'interaction : Action et DirectInput.

Action L'objet Action permet le déclenchement d'un évènement. Cet objet peut avoir pour argument une liste de paramètres. Une Action sans paramètres correspond typique-

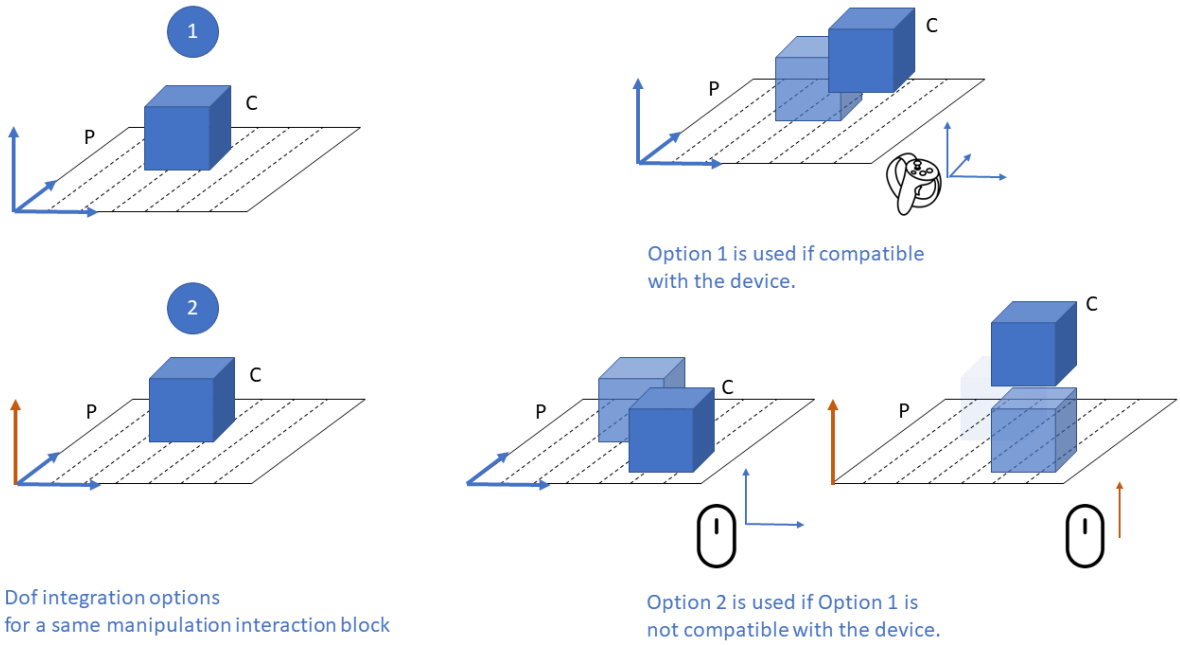


FIGURE *2.13 – Manipulation à trois degré de liberté.*

ment à un bouton dans l'interface finale. Une action avec des paramètres correspond à un formulaire de saisie.

Direct Input L'objet Direct Input permet de modifier en temps réel un paramètre de l'application ou d'un objet 3D (valeur numérique, champ de texte, ...).

Interaction List Si les deux briques précédemment décrites sont suffisantes pour décrire tous les éléments nécessaires au contrôle d'application, elles ne permettent pas toujours de générer une interface utilisateur optimale. En effet, la disposition et le regroupement des éléments similaires de l'interface sont essentiels pour l'ergonomie. Nous introduisons donc une dernière brique d'interaction appelée Interaction List qui permet de guider la création de menus et sous-menus lors de la génération de l'interface utilisateur. La brique Interaction List est utilisée pour regrouper un ensemble d'autres briques d'interaction qui ont une valeur sémantique ou des usages proches. Il s'agit d'indiquer quels éléments doivent éventuellement être regroupés dans l'interface utilisateur.

2.3.2.6 Cas de la navigation

L'absence de brique d'interaction dédiée à la navigation constitue la différence la plus marquée entre la définition des tâches en environnement 3D de Bowman [16] et notre modèle d'abstraction de l'interaction. Une tâche de manipulation est en effet constituée de deux composantes : une (ou des) technique(s) de navigation permettant à l'utilisateur de se déplacer dans l'environnement 3D, et des outils permettant le calcul de la position de

l'utilisateur par rapport à un point fixe de l'environnement. Notre point de vue concernant la technique de navigation est que celle-ci ne devrait pas être définie par le média 3D. Effet le choix d'une technique de navigation adaptée dépend fortement du support et de ses périphériques. Nous laissons donc, à ce stade, l'implémentation de la tâche de navigation à l'entière discrétion du navigateur UMI3D. Nous incluons toutefois dans le protocole UMI3D un système de téléportation permettant au média de donner au navigateur sa position initiale dans la scène 3D. A l'inverse, nous détaillerons dans le Chapitre 3 les raisons pour lesquelles il est absolument nécessaire pour le support de la collaboration que le média 3D soit informé en temps réel de la position et des geste de l'utilisateur. L'implémentation et l'utilisation de cette fonctionnalité seront par la suite l'objet du Chapitre 4 qui présente nos travaux préliminaires sur le suivi des utilisateurs par un média 3D. Ce suivi constitue en effet une problématique complexe du fait de la nature inconnue des supports. Le concepteur n'a de fait pas connaissance de la complétude des données, c'est à dire des points (ou articulations) du corps de l'utilisateur dont la position sera notifié au média 3D en temps réel.

2.4 Implémentation de médias et navigateurs UMI3D

2.4.1 Plateforme SVEP : preuve du concept UMI3D

Pour valider le modèle d'abstraction du protocole UMI3D, nous avons implémenté une plateforme Web permettant d'héberger simultanément plusieurs médias 3D communiquant par le protocole UMI3D. Cette plateforme a été développée dans la technologie Node.js [2] ce qui nous permet notamment de créer rapidement des scènes 3D avec la librairie Three.js [25]. Un média 3D hébergé sur la plateforme est composé, d'un script définissant conjointement une scène 3D et des briques d'interaction, et de fichiers ressources correspondant aux modèles 3D et textures. Enfin, Socket.io a été utilisé pour créer un WebSocket permettant la communication temps réel entre les environnements 3D et les supports. Les choix technologiques pour cette première implémentation ont été faits dans le but d'une validation rapide du modèle. Une utilisation ultérieure de moteurs de jeu tels que Unity ou Unreal Engine était déjà planifiée dans le but d'enrichir les environnements 3D développés et d'accroître les performances. L'architecture globale de cette plateforme appelée SVEP (pour Shared Virtual Environment Platform) est présentée dans la Figure 2.14.

2. Node.js est un framework JavaScript largement utilisé pour la conception d'applications web.

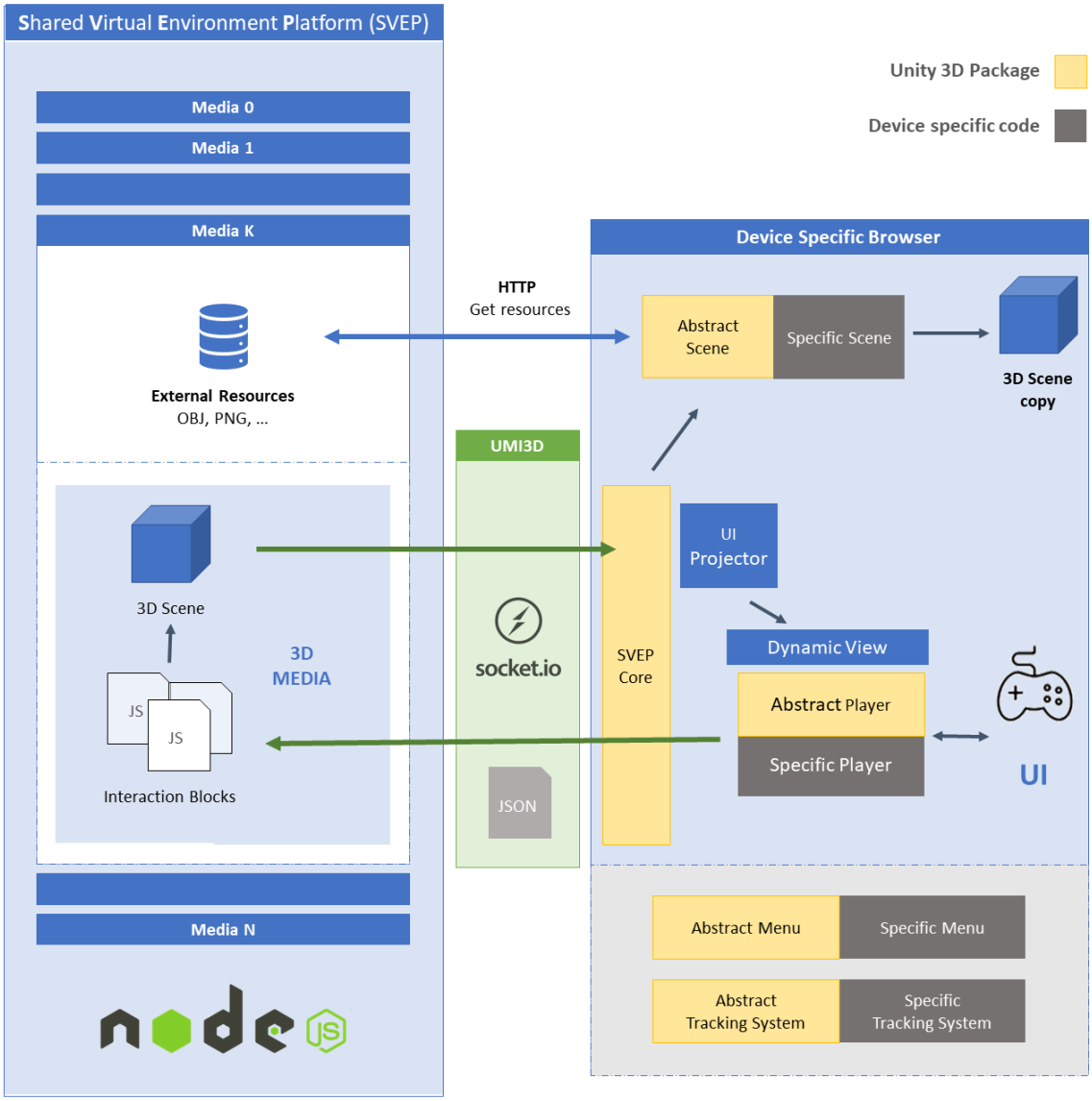


FIGURE *2.14 – SVEP*

2.4.2 Implémentation de navigateurs UMI3D avec Unity 3D

Dans le but d'accélérer l'implémentation du navigateur UMI3D pour un support donné, nous avons développé un une extension pour le moteur de jeu Unity 3D. Cette extension permet de faire abstraction de la connexion réseau avec la plateforme SVEP lors du développement d'un navigateur. De plus, il permet de mutualiser l'architecture de tous les navigateurs utilisant Unity (Figure 2.14). Un navigateur UMI3D conçu avec le package Unity est composé des 5 modules suivants :

SVEP Core. Le module SVEP Core est en charge de la communication réseau entre le navigateur UMI3D et les médias 3D hébergés par la plateforme (Figure 2.15). Il convertit par ailleurs les messages échangé au format JSON avec le média 3D en Objet de transfert de données[3]. Enfin, il pilote les autres modules.

Scene Le module Scene permet de charger dynamiquement les objets virtuels du média 3D et de les synchroniser en temps réel. Le module Scene doit étendre la classe Abstract Scene du package Unity. L'utilisation d'une classe abstraite permet de limiter les développements pour un client donné à l'implémentation d'une stratégie de chargement dynamique adaptée aux capacités du dispositif. Cette stratégie définit notamment dans quel ordre le navigateur charge les contenus 3D, s'il en charge plusieurs en parallèle, ou encore les matériaux et shaders appliqués aux contenus chargés. Cette stratégie peut éventuellement également s'adapter à la qualité de la connexion entre le média et le navigateur (bande passante, débit). Plusieurs dispositifs peuvent donc utiliser la même implémentation de ce module s'ils ont des performances semblables (mémoire vive, mémoire graphique). Sinon, une stratégie de chargement différente est mise en place, avec éventuellement un canal de rendu et des shaders différents.

Player. Le module Player est chargé de la projection dynamique des briques d'interactions sur le dispositif. Si une classe abstraite du package Unity permet de mutualiser les interfaces avec le module SVEP Core, il est nécessaire d'implémenter spécifiquement la projection des différentes briques d'interaction sur le dispositif.

Menu. Le module Menu permet à l'utilisateur de naviguer entre les différents environnements 3D mis à sa disposition par la plateforme SVEP.

Tracking System. Le module Tracking System est implémenté uniquement pour les supports de Réalité Augmentée ou Mixte. Il est en charge du positionnement dynamique

3. Un objet de transfert de données (data transfer object ou DTO en anglais) est un patron de conception utilisé dans les architectures logicielles objet. [93]

des objets virtuels en fonction des objets réels de l'environnement (marqueurs, positions Gps, ...).

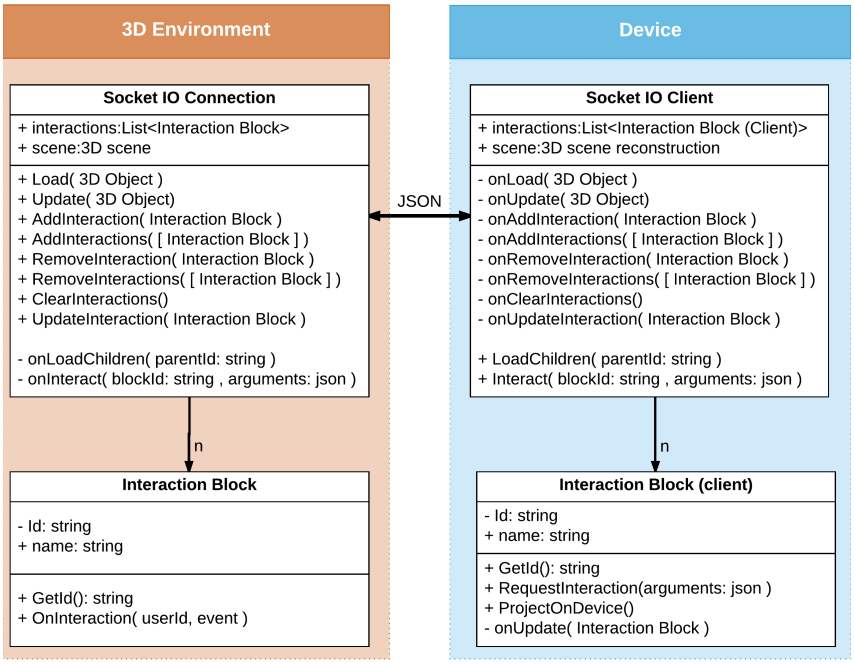


FIGURE *2.15 – Communication entre un navigateur UMI3D et un média 3D*

2.4.3 Premières expérimentations

Cette section présente des expérimentations techniques développées et testées par des membres de notre équipe. Leur objectif n'est, à ce stade des travaux, pas une évaluation des performances utilisateurs lors de l'utilisation du protocole UMI3D, mais une vérification du fonctionnement technique de l'architecture proposée. La manière dont peuvent être menées, et surtout interprétées, des études utilisateur complètes (c'est à dire incluant des indicateurs de performances objectifs) de média 3D basés sur notre architecture sera exposée et débattue dans le Chapitre 5.

FIGURE *2.16 – Premiers navigateurs UMI3D*

TABLE *2.1 – Supports expérimentés*

Support	1	2	3
Affichage	Smart TV	écran tactile	Oculus rift Hmd
Navigation	Mouse & keyboard	ARToolKit	Oculus rift 6D
Tracking System	.	ARToolKit & Camera	.
Interaction	Souris & clavier	Tactile	Oculus rift remote
Degrés de liberté de l'interaction	2	2	1+1
Menus	2D	2D	plan 2D dans la scène

2.4.3.1 Supports expérimentés

Dans le but de vérifier l'interopérabilité des médias 3D développés pour la plateforme SVEP, nous avons implémenté trois navigateurs UMI3D pour différents supports (Figure 2.16). Nous avons ainsi pu vérifier que le développement d'une seule application, un navigateur UMI3D, permet bien l'interaction entre l'ensemble des médias 3D existant et un nouveau dispositif. Les points notables de l'implémentation sont résumés ci-dessous ainsi que dans le Tableau 2.1.

Implémentation du module Scene. La puissance de calculs et les capacités graphiques de nos dispositifs étant semblables, ils partagent une même implémentation du module Scene.

Implémentation du module Tracking System. Seul le support 2 est capable d'afficher un environnement 3D en Réalité Augmentée. Il est donc le seul dispositif implémentant le module Tracking System. Ce module permet comme dans [48] d'effectuer, en temps réel, le recalage avec des marqueurs 2D à l'aide de l'ARToolKit.

Implémentation du module Menu & contrôle d'application généré par le module Player. Pour les dispositifs 1 et 2, l'interface utilisateur regroupant le module Menu ainsi que le contrôle d'application généré par le module Player, est une interface 2D traditionnelle superposée à la scène 3D. Toutefois, l'Oculus Rift du dispositif 3 ne permet pas ce type d'interface. Une interface en trois dimensions a donc été conçue pour ce dispositif. Il s'agit simplement d'une interface 2D, insérée dans un plan de la scène 3D.

2.4.3.2 Média 1 : Manipulation d'objet virtuel & retours visuels

Dans le but d'évaluer notre modèle, nous avons développé et publié sur la plateforme SVEP, un environnement 3D permettant à un utilisateur de manipuler un objet virtuel (Figure 2.17). Notre objectif était de vérifier, que le modèle permet la manipulation d'un

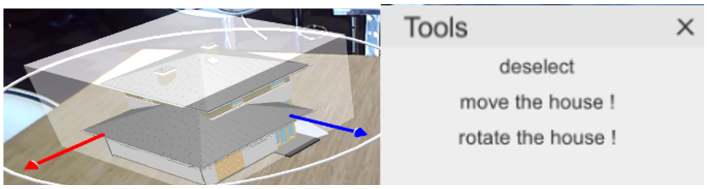


FIGURE *2.17 – Intégration de degrés de liberté*

FIGURE *2.18 – Séparation de degrés de liberté*

objet virtuel par un dispositif quelconque, qu'une réaction aux manipulations de l'utilisateur peut être générée dynamiquement, et enfin qu'il est bien possible de mettre à jour en temps réel les interactions dont dispose l'utilisateur de sorte à scénariser ses échanges avec le média.

Scénario d'interaction. L'utilisateur dispose de deux modes d'interaction proposés dans ce scénario. Le mode Manipulation permet la manipulation des degrés de liberté de l'objet qui peut ainsi être positionné par translation dans le plan XY et orienté par une rotation d'axe Z. Une action permet de plus de passer dans le mode Configuration. Ce deuxième mode donne à l'utilisateur un ensemble d'interactions lui permettant de configurer les degrés de liberté du mode manipulation. La Figure 2.19 montre le menu généré par le dispositif 1 (voir Tableau 2.1) pour le mode de configuration. La configuration permet de choisir si les manipulations se font de manière continue ou discrète. Dans le cas d'une rotation discrète, le pas de rotation est configurable. Les translations selon les axes X et Y peuvent être intégrées (Figure 2.17) ou séparées (Figure 2.18). Enfin, une sélection de l'objet entraine un retour au mode de manipulation.

Il est à noter que la spécification du protocole UMI3D a évolué depuis la présentation de cette expérimentation dans [19]. Il n'est dorénavant plus possible d'indiquer, via la brique d'interaction Manipulation, si le mouvement est discret ou continu. En effet, cette distinction pouvant être réalisée par le média dans l'interprétation du mouvement continu reçu, elle n'a pas été conservée. D'un point de vue utilisateur, il n'y aurait aucune différence, et le paramètre discret/continu fonctionnerait de la même manière.

Retours visuels. Le dispositif n'ayant connaissance ni de la nature sémantique des manipulations, ni de l'objet manipulé, il ne lui est pas possible de générer lui-même un retour visuel durant l'interaction de sorte à indiquer à l'utilisateur, la prise en compte de

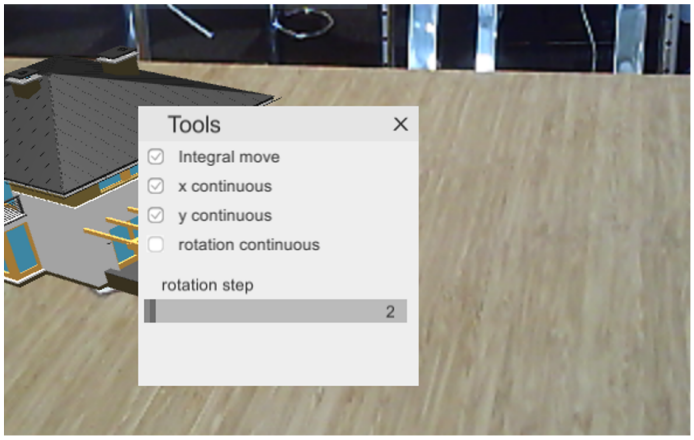


FIGURE *2.19 – Configuration d'une manipulation*

son interaction, et les mouvements possibles. Ce retour doit donc être généré directement par le média lors de l'activation des briques d'interaction. La Figure 2.20 résume les différents retours générés par le média 3D. Lors du pointage, la boîte de sélection est affichée en transparence autour de l'objet. Si l'objet est sélectionné, des flèches ou des cercles sont affichés pour indiquer les translations ou les rotations possibles. Enfin les flèches et cercles affichés lors de la sélection changent de couleurs lors de l'activation des briques d'interaction qu'ils représentent.

2.4.3.3 Média 2 : Manipulation simultanée de plusieurs objets

Si l'implémentation du Média 1 a permis de valider les principes de base de notre modèle, ce test ne permet pas à lui seul d'évaluer la pertinence du modèle pour la scénarisation d'interactions complexes. Nous présentons ci-dessous un environnement complémentaire montrant comment la définition d'un mouvement générique par des briques d'interaction permet de scénariser la manipulation simultanée de plusieurs objets. Avec ce test, nous voulions nous assurer que la brique d'interaction Manipulation ne permette pas uniquement de modéliser des translations et des rotations d'objets virtuels, mais également des interactions complexes tel que le grossissement, la sélection multiple ou encore le déclenchement d'évènements. L'environnement présenté ci-dessous montre qu'il est possible de modéliser de telles manipulations par un mouvement générique à partir d'une brique d'interaction Manipulation.

Scénario d'interaction. L'environnement est composé d'une pile de plateaux. Le but de l'utilisateur est de trier les plateaux en fonction de leur taille. La seule opération autorisée consiste à retourner le haut de la pile (Figure 2.21).

Sélection à un degré de liberté. Une brique à un degré de liberté est utilisée en association avec un mouvement vertical. Un mouvement de haut en bas ou de bas en

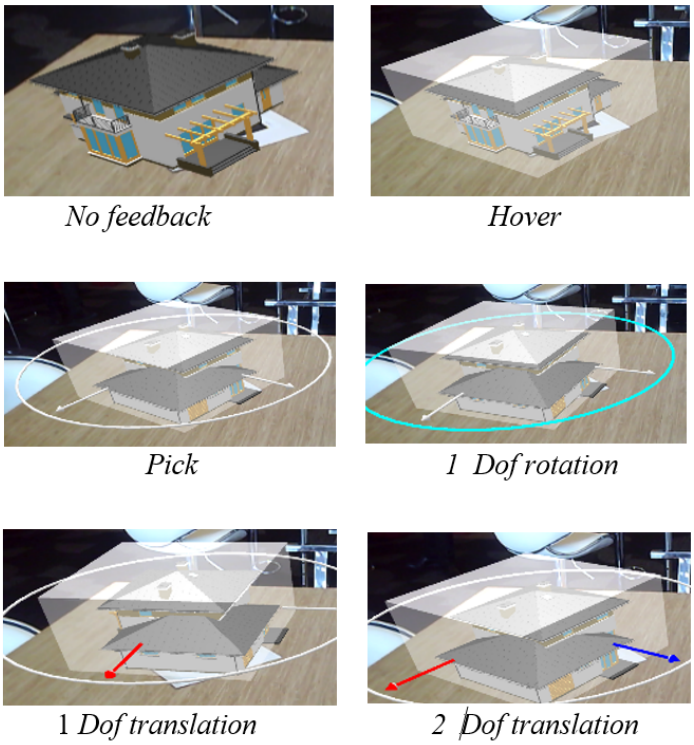


FIGURE *2.20 – Retour visuels*

FIGURE *2.21 – Le jeu du crêpier*

haut permet d'ajouter ou d'enlever un plateau à la sélection (Figure 2.22). Le média 3D définit l'amplitude du mouvement utilisé pour incrémenter/décrémenter la sélection d'un plateau. La possibilité de sélectionner les plateaux en plusieurs fois permet à l'utilisateur d'éventuellement corriger sa sélection.

Tâche intégrée. Un deuxième degré de liberté est utilisé pour déclencher le retournement de la sélection en l'associant à un mouvement horizontal. L'intégration de ces deux degrés de liberté par une brique d'interaction Manipulation permet à l'utilisateur de sélectionner et retourner le haut de la pile d'un seul mouvement (Figure 2.23).

2.4.4 Conclusion des premières expérimentations

Par construction, le portage sur un nouveau support de l'ensemble des médias 3D développés avec UMI3D ne dépend pas du nombre de médias mais dépend de la difficulté à implémenter la projection des briques d'interaction sur le dispositif. L'utilisation de UMI3D offre donc un gain de portabilité croissant lorsque le nombre de médias augmente. Au vu des tests présentés dans cette section, il apparait que la composition et l'évolution dynamique des briques d'interaction permettent bien de scénariser et de décrire de manière générique les interactions avec un média 3D. Par ailleurs, le protocole permet naturellement la collaboration entre des dispositifs de natures différentes (il suffit de connecter simultanément plusieurs supports au média via UMI3D). Toutefois, il reste à ce stade nécessaire d'établir les bonnes pratiques de conception des médias 3D. En effet, si la collaboration est mécaniquement rendu possible par l'architecture de notre modèle, il reste à démontrer que UMI3D permet de concevoir des médias répondant efficacement aux attentes vis à vis d'un système de support du travail collaboratif.

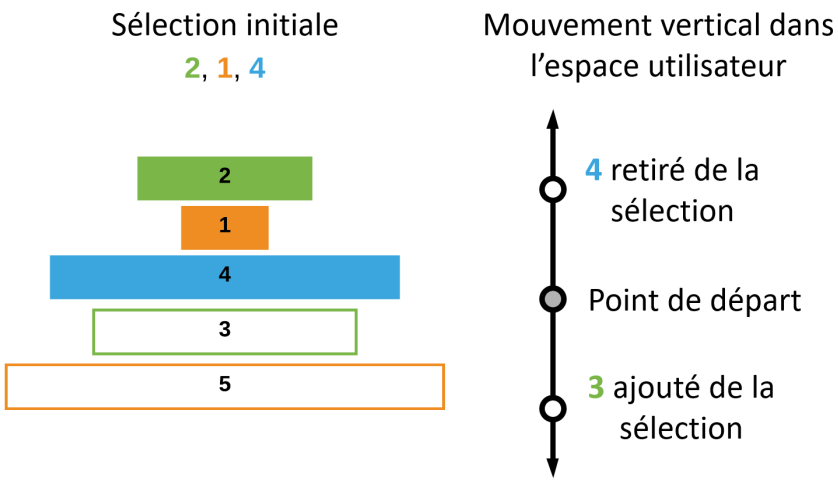


FIGURE 2.22 – *Sélection à un degré de liberté*

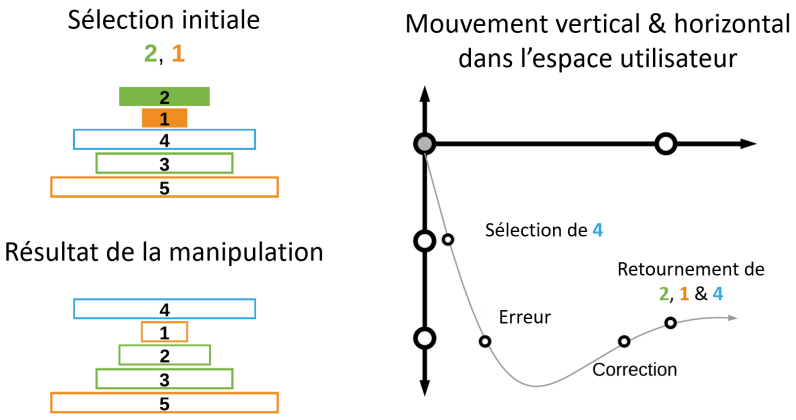


FIGURE *2.23 – Intégration de la sélection et du retournement*

Chapitre 3

Support de la Collaboration avec UMI3D

Au cours du chapitre précédent nous avons introduis le protocole UMI3D (Unified Model for Interaction in 3D environment). Ce protocole web permet à des concepteurs d'environnements 3D de créer des médias 3D, indépendamment des supports de RA/RV des utilisateurs finaux. Si UMI3D permet bien, d'un point de vue technique, de connecter simultanément plusieurs utilisateurs à un même média 3D, il reste nécessaire de vérifier que UMI3D répond bien aux attendus des systèmes de support du travail coopératif (computer-supported cooperative work systems). L'objectif de ce chapitre est donc de vérifier que le travail collaboratif peut bien être supporté en utilisant UMI3D pour la conception d'un média 3D. Pour ce faire, nous établissons la liste des attendus pour le support de la collaboration par un média 3D restant à vérifiées suite à nos travaux précédant (voir Chapitre 1.4.2 Application au support de la collaboration par un média 3D). Nous présentons par la suite une boîte à outils permettant une implémentation de ces attendus dans un média 3D utilisant le protocole UMI3D. Cette boîte à outil, ainsi que les exemples d'utilisation fournis montrent qu'il existe au moins une implémentation permettant de valider ces attendus dans un média conçu avec UMI3D. Enfin, nous présentons une étude de cas qui nous permet d'une part de discuter les implications de UMI3D pour la création d'un média 3D, et d'autre part de vérifier dans une étude utilisateur que les attendus généraux vis à vis d'un système supportant le travail collaboratif sont bien observables à l'utilisation de ce média.

3.1 Quels requis du travail collaboratif assisté par ordinateur UMI3D sont validés par UMI3D ?

Comme développé dans le chapitre précédent, le concept principal du protocole UMI3D est de décrire l'interaction entre un média 3D et un support quelconque par un nombre limité de briques d'interaction. Le protocole permet par ailleurs nativement de connecter plusieurs utilisateurs à un média 3D, et de synchroniser en temps réel l'environnement 3D lors de l'interaction avec ces utilisateurs. S'il n'y a pas de limitations techniques à la collaboration de plusieurs utilisateurs avec UMI3D, rien ne garantit en l'état que UMI3D soit

TABLE *3.1 – UMI3D : Support des fonctionnalités requises par les CSCWS*

Fonctionnalités	co-située & non immersive	distante ou immersive
Contenus et artéfacts partagés	supportés	supportés
Artefacts individuels	à vérifier	à vérifier
Rôles utilisateur	à vérifier	à vérifier
Mécanismes de coordination	à vérifier	à vérifier
Malléabilité	à vérifier	à vérifier
Avatars	non supportés	non supportés
Communication	supportée	non supportée

adapté au support du travail coopératif. Afin de le vérifier, nous avons résumé Tableau 3.1 les fonctionnalités requises pour la collaboration ainsi que leur support par UMI3D. Les fonctionnalités notées "à vérifier" sont celles qui ne sont pas trivialement supportées par UMI3D, et dont nous allons démontrer le support dans ce chapitre. Pour les fonctionnalités signalées comme n'étant pas supportées, leur support n'est pas incompatible avec UMI3D mais demandera une extension future du protocole.

La synchronisation temps réel de tout contenu appartenant au média 3D étant le comportement natif de UMI3D, les retours visuels partagés sont nativement supportés. A l'inverse, comme le protocole ne permet aujourd'hui pas d'informer le média 3D de la position de l'utilisateur, il n'est pas encore possible de générer des avatars. De même, le protocole ne fournit pour l'instant pas de canal de communication vocale pour les cas de collaboration à distance.

3.2 Boîte à outil pour la création de médias 3D

Dans le but de simplifier la création et l'exécution de médias 3D utilisant le protocole UMI3D, et d'en étudier les propriétés pour la collaboration, nous avons choisis de développer une boîte à outil basée sur le moteur de jeu Unity 3D. Nous avons pour cela implémenté des primitives (des scripts C# appelés MonoBehaviour), que les développeurs peuvent instancier depuis l'éditeur graphique de Unity, afin de créer des objets virtuels et des interactions. Cette boîte à outils contient par ailleurs des scripts et objets pré-configurés (appelés prefabs) contenant l'ensemble des composants en charge de la connexion et du dialogue avec les navigateurs UMI3D (i.e. avec les supports des utilisateurs). Ces fonctionnalités sont identiques à celles de l'implémentation NodeJS présentée au chapitre précédant et dans [19].

Dans le reste de cette section, nous présentons nos attentes quand à la contribution de notre boîte à outils au développement de systèmes de support du travail coopératif (CSCWS). Nous expliquons ensuite en détail comment cette boîte à outils peut être utilisée pour la conception d'interactions, ainsi que les primitives supplémentaires (propriétés d'objet asynchrones, filtrage des contenus visibles, filtrage des interactions disponibles, et patrons d'interaction) que nous proposons pour la gestion de la collaboration dans des environnements UMI3D. Ces primitives sont réparties en trois catégories : développement d'interaction, graphe des états utilisateur, et malléabilité.

3.2.1 Contribution pour la conception des CSCWS

Avec cette boîte à outils, nous souhaitons fournir une implémentation de UMI3D disposant de tous les avantages propres aux moteurs de jeu, comme une interface graphique permettant de glisser-déposer des objets et/ou des primitives dans la scène, ou encore la possibilité de créer, configurer et supprimer des objets et primitive durant l'exécution du média. Par ailleurs notre boîte à outils permet de conserver ces propriétés de conception à l'exécution, y compris lorsque plusieurs utilisateurs et supports sont connectés au média en cours de conception.

Nous soutenons que la combinaison, des avantages pour la conception de médias 3D d'un moteur de jeu tel que Unity, et de la couche d'abstraction des supports 3D offerte par UMI3D ouvrira de nouvelles perspectives pour la conception de médias 3D. Par exemple, la possibilité d'expérimenter un nouveau support (c'est à dire un nouvel outil) pour l'interaction avec un média UMI3D existant va accélérer la classification des propriétés interactives des supports, et améliorer le choix des supports dans les applications industrielles. De plus, les temps de développement réduits devraient laisser plus de temps aux concepteurs de médias 3D pour se focaliser sur les cas d'usages. Enfin, nous espérons que les possibilités de conception à l'exécution conduiront à des itérations plus courtes dans la conception orientée-utilisateur des systèmes collaboratifs à trois dimensions.

3.2.2 Développement d'interactions

3.2.2.1 Artéfacts partagés

Nous voyons deux catégories de retours visuels (ou artéfacts) qui soient intéressants à partager dans un média 3D : les réactions de l'environnement aux actions des utilisateurs, et le Nimbus des objets composant l'environnement 3D. Le Nimbus informe les utilisateurs des changements concernant ces objets d'une part, et d'autre part des interactions des autres utilisateurs avec ces objets. La première catégorie d'artéfacts constitue les mises à jour du contexte de travail partagé. La seconde catégorie d'artéfacts permet quand à elle d'augmenter la conscience qu'ont les utilisateurs de ces changements dans le contexte de

```
1  public class TranslateZ : Dof {
2
3
4    public Object3D movedObject;
5
6
7
8    protected override void init()
9    {
10     base.init();
11     frameOfReference = movedObject;
12     addInteractionListener(onInteraction);
13   }
14
15   void onManipulated( User user, Movement
         evt)
16   {
17     movedObject.position += evt.translation;
18   }
19
20 }
```

Listing 3.1 – Héritage d'une brique d'interaction

```
1  public class MyScript : MonoBehaviour {
2
3    public Object3D movedObject;
4    public DofSeparation translationOptions;
5    private Dof dof;
6
7    void start()
8    {
9      dof = new Manipulation();
10     dof.dofOptions = translationOptions;
11     dof.frameOfReference = movedObject;
12     dof.addInteractionListener(onManipulated);
13   }
14
15   void onManipulated( User user, Movement
         evt)
16   {
17     movedObject.position += evt.translation;
18   }
19
20 }
```

Listing 3.2 – Création dynamique d'une interaction

travail partagé, ainsi que leur conscience des activités des autres utilisateurs.

Pour ces fonctionnalités, nous avons implémenté un script (MonoBehaviour) Unity par brique d'interaction composant UMI3D. De plus, pour permettre au concepteur du média d'implémenter des retours visuels, chaque brique d'interaction dispose d'un évènement, émis lorsque la brique est activée par un navigateur UMI3D. Cet évènement contient les paramètres éventuels de la demande d'interaction reçue (par exemple le mouvement généré pour une brique manipulation). Il y a dès lors deux possibilités pour associer une réaction à une brique d'interaction : étendre une brique d'interaction de base en créant un nouveau script héritant de la brique en question, ou ajouter depuis un autre script ou l'interface graphique, un délégué (fonction à exécuter lors de l'émission de l'évènement) à l'évènement de la brique d'interaction. Les Listing 3.1 et Listing 3.2 montrent comment ces deux stratégies peuvent être appliquées dans le but de permettre à un utilisateur de translater un objet avec un degré de liberté. La première option (Listing 3.1) est plutôt adaptée lorsque le patron d'interaction implémenté est réutilisé plusieurs fois dans le média, ou si le concepteur souhaite le sauvegarder dans un prefab (objet pré-configuré). Cette approche peut de plus être utilisée pour construire une bibliothèque d'interactions de haut niveau. La seconde option (Listing 3.2) est quand à elle plus utile pour générer dynamiquement des interactions à partir d'un script différent.

Par défaut, chaque objet virtuel est synchronisé en temps réel par le média. C'est pourquoi, lors d'un retour visuel suite à une interaction, les modifications des propriétés des objets virtuels sont partagées à tous les navigateurs, et donc à tous utilisateurs, connectés au média 3D.

3.2.2.2 Artéfacts individuels

Comme nous l'avons souligné dans notre présentation des prérequis pour le support de la collaboration, les artéfacts résultant d'activités individuelles des utilisateurs ne doivent pas être partagés. Par exemple, si le concepteur du média souhaite indiquer à un utilisateur

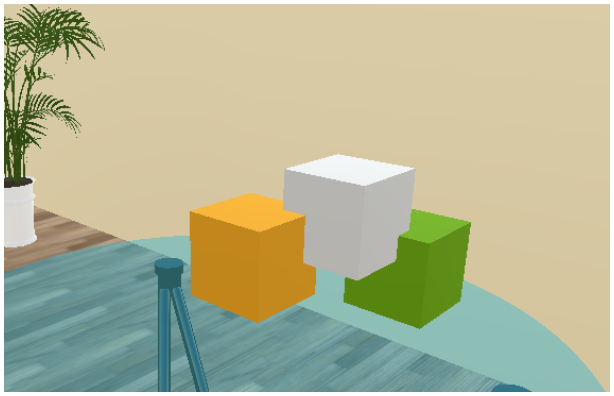


FIGURE *3.1 – Exemple : Sélection et survol d'objets virtuels*

qu'il est possible de sélectionner un objet, en changeant sa coloration lors de son survol par l'outil de sélection du support, le partage de ce retour visuel peut être perçu comme du bruit par les autres utilisateurs. Dans ce cas, la génération d'un retour visuel non reliable au contexte de travail partagé constitue une surcharge cognitive pour les utilisateurs. Ce retour visuel peut ainsi avoir un impact négatif sur la perception des autres utilisateurs et/ou sur la perception du contexte de travail. À l'inverse, dans un autre média, ou simplement dans une phase de travail différente, ce même retour visuel peut fournir un moyen à l'utilisateur de montrer l'objet aux autres utilisateurs. Dans ce cas, le retour visuel constitue un outil de communication essentiel au dialogue et à la coordination des utilisateurs. Il doit alors être partagé.

Pour permettre cette coexistence des artéfacts individuels et partagés, notre boîte à outil laisse le concepteur du média choisir quels paramètres et propriétés des objets virtuels doivent être synchronisés. Cela signifie que le concepteur peut choisir, pour chaque transformation d'un objet, de partager ou non la transformation. Pour ce faire, nous avons implémenté une classe ObjectProperty qui permet de décrire une propriété pouvant avoir une valeur dépendant de l'utilisateur. Ces propriétés peuvent être dynamiquement définies comme étant synchrones ou asynchrones en fonction de l'évolution du contexte de travail (individuel et partagé). Il est par ailleurs possible d'accéder (Listing 3.3) ou d'affecter (Listing 3.4) la valeur par utilisateur de ces propriétés sans connaissance du statut actuel (i.e. synchrone ou asynchrone). Un exemple d'usage de ces propriétés asynchrones est visible Figure 3.1. Ici, le cube à droite ne devient vert que pour l'utilisateur le survolant avec son outil de sélection.

La notion d'artéfact individuel ne concerne pas uniquement la transformation asynchrone de certaine propriétés des objets virtuels. Il peut plus simplement s'agir d'un objet virtuel temporairement ou définitivement ajouté ou supprimé de la scène. Dans ce cas, une fonctionnalité permettant de rendre visible l'artefact pour un unique utilisateur est nécessaire. La gestion par utilisateur de la visibilité des contenus étant également nécessaire pour la création de rôles utilisateur, cette fonctionnalité sera décrite par la suite

```
public class Object3DProperty<T>
{
  T value;
  Dictionary<User, T> asyncValues;
  bool isAsync;

  public T GetValue(User user)
  {
    if (isAsync && asyncValues[user]))
    {
      return asyncValues[user];
    }
    else
    {
      return value;
    }
  }

  ...

}
```

Listing 3.3 – Accès à la valeur par utilisateur d'une propriété d'un objet virtuel

```
public class Object3DProperty<T>
{
  T value;
  Dictionary<User, T> asyncValues;
  bool isAsync;

  ...

  public void SetValue(User user,T val)
  {
    if (isAsync)
    {
      asyncValues[user] = val;
    }
    else {
      value = val;
    }
  }


}
```

Listing 3.4 – Affectation de la valeur par utilisateur d'une propriété d'un objet virtuel

(voir Filtrage des contenus visibles).

3.2.3 Graphe des états utilisateur

L'implémentation de l'Aura des objets constitue l'une des composantes essentielle à la conception d'un CSCWS. Cela consiste en la mise en place d'une solution technique pour définir quelles interactions sont fournies à chaque utilisateur en fonction de son rôle, des contextes de travail individuel et partagé, et de la manière dont les utilisateurs coordonnent et articulent leur travail (i.e. des mécanismes de coordination).

Pour simplifier le développement de l'Aura des objets, nous proposons aux concepteurs de média 3D l'utilisation d'une interface graphique. Cette interface permet de dessiner un graphe fini des états possible pour les utilisateurs. Chaque utilisateur connecté au média constitue un jeton positionné sur l'un des états du graphe. Ce jeton peut être déplacé dynamiquement à tout moment en fonction des actions de l'utilisateur, des actions des autres utilisateurs, ou d'évènements automatiques provenant du média lui-même (par exemple la fin d'une tâche minutée). La suite de cette section explique comment ce graphe d'états utilisateur peut-être utilisé par le concepteur du média pour filtrer les objets visibles, contrôler les interactions disponibles, et enfin créer des rôles utilisateur.

3.2.3.1 Filtrage des contenus visibles

Pour permettre le filtrage des contenus virtuels, nous avons implémenté un filtre de visibilité abstrait sous la forme d'un script MonoBehaviour. Pour créer un filtre de contenu spécifique, le concepteur du média doit simplement créer une classe (i.e. un nouveau script) héritant de ce filtre abstrait. Il suffit ensuite d'ajouter ce comportement à un objet virtuel de la scène. Si l'on souhaite faire varier la visibilité des contenus en fonction de l'état de l'utilisateur, il suffit d'inclure l'état de l'utilisateur comme paramètre pour la

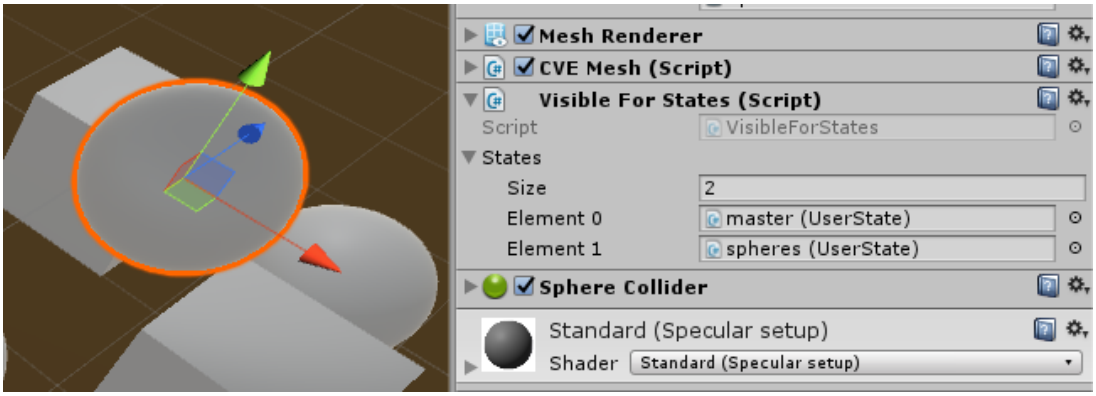


FIGURE *3.2 – Application d'un filtre de contenu*

détermination de la valeur de retour (valeur booléenne : Vrai|Faux) du filtre. Dès lors, un changement d'état de l'utilisateur pourra modifier le retour du filtre, et ainsi modifier la visibilité de l'objet. Dans la Figure 3.2, nous présentons un usage possible de ces filtres. Nous appliquons un filtre à une sphère de sorte à la rendre visible pour un ensemble d'états utilisateur. Dans cet exemple, les utilisateurs visualisent un ensemble de cubes et de sphères. Selon l'état utilisateur, l'utilisateur visualise un seul ou les deux types d'objets. Le filtre visible sur la figure montre que les sphères ne sont visibles que pour les états utilisateur 'master' et 'spheres'.

3.2.3.2 Filtrage des interactions disponibles

De manière similaire au filtrage des contenus visibles, nous avons implémenté un filtre abstrait pour la disponibilité des interactions. Celui-ci permet au concepteur du média 3D de contrôler la distribution des interactions entre les utilisateurs au cours du temps. Ces filtres permettent par exemple, comme présenté Figure 3.1 de limiter, à un utilisateur à la fois, la sélection d'un objet (un cube dans cet exemple). Dans cet exemple, le cube est respectivement coloré en orange ou en vert lorsqu'il est sélectionné ou respectivement pointé (survolé par l'outil de sélection) par un utilisateur. Le filtre permet ici d'empêcher les manipulations concurrentes du cube. Pour ce faire, le filtre rend indisponibles, les briques d'interaction permettant la le pointage et la sélection, lorsque le cube est déjà sélectionné. Dans l'exemple donné, un autre utilisateur a sélectionné le cube orange, il n'est donc ni possible de le pointer, ni de le sélectionner car le média 3D a, au moment de la sélection du cube, notifier aux utilisateur l'expiration de ces contrats d'interaction. A l'inverse, l'utilisateur dispose toujours de contrats d'interaction permettant le pointage et la sélection des deux autres cubes. Le cube vert est notamment vert (et non blanc) car il est pointé par cet utilisateur. Dans d'autres cas, il est possible de distribuer/filtrer les interactions en fonction du rôle des différents utilisateurs.

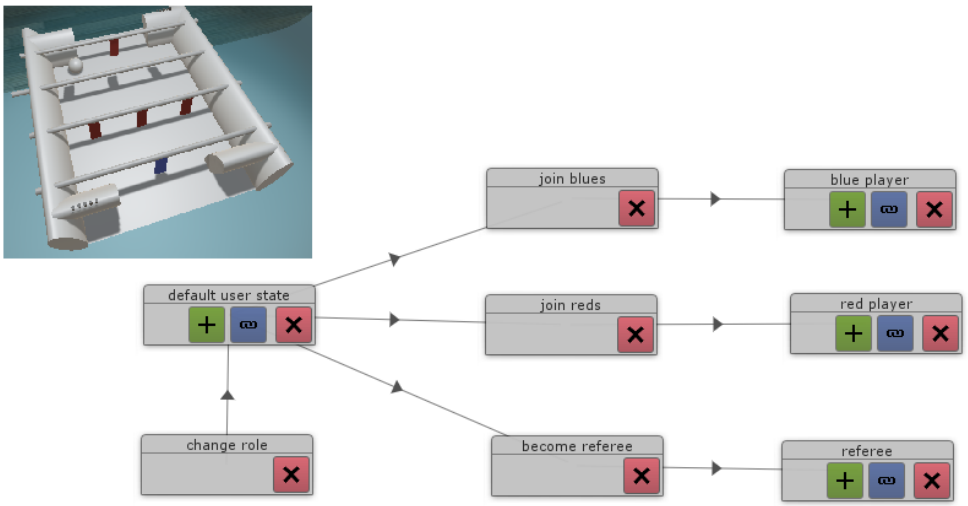


FIGURE *3.3 – Exemple : Graphe des états utilisateur pour un jeu de babyfoot virtuel*

3.2.3.3 Création de rôles utilisateur

L'implémentation de rôles utilisateur devient possible grâce à l'ensemble des fonctionnalités présentées dans cette section. Ces fonctionnalités peuvent être utilisées pour distribuer dynamiquement différentes interactions et contenus virtuels en fonction de l'utilisateur. Pour cela, il suffit de créer dans le graphe d'états utilisateur différentes branches pour les différents rôles. Il est par ailleurs possible de sécuriser l'accès à l'une de ces branches en créant un formulaire de connexion à l'aide de la brique d'interaction Action. Pour illustrer la création de rôles utilisateurs avec notre boîte à outil, nous avons implémenté un jeu de babyfoot virtuel (Figure 3.3). Ce jeu contient trois rôles d'utilisateurs : joueur bleu ('blue player' sur la figure), joueur rouge ('red player') et arbitre ('referee'). Les informations permettant de tirer et de lancer une nouvelle balle sont réparties selon les rôles. Le moteur physique de Unity a été utilisé pour simuler et partager les mouvements et collisions de la balle.

3.2.3.4 Mécanismes de coordination

Le graphe des états utilisateur peut également être utilisé pour définir des mécanismes de coordination. En effet, en ajoutant des filtres aux interactions permettant la navigation dans le graphe d'états, le concepteur du média 3D peut définir des tâches à exécuter en parallèle par les utilisateurs, ou encore définir des séquences de tâches à effectuer chronologiquement. Il est également possible de déverrouiller certaines interactions et/ou chemins du graphe lorsque les utilisateurs atteignent des objectifs prédéfinis. Enfin, il est possible d'implémenter des filtres d'interactions de sorte à assigner certaines tâches à un utilisateur, ou à un groupe d'utilisateurs.

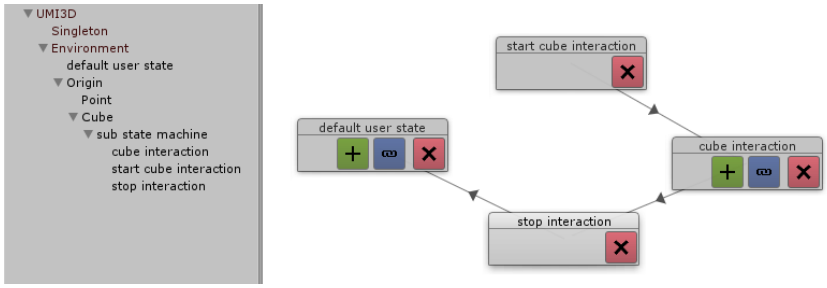


FIGURE *3.4 – Stockage d'un sous-graphe d'états utilisateur dans un objet préfabriqué.*

3.2.4 Malléabilité

3.2.4.1 Instanciation de patrons d'interaction prédéfinis

Dans le but de simplifier l'instanciation de patrons d'interaction ou de mécanismes de coordination, les états utilisateur sont eux même des scripts Unity faisant partie intégrante de la scène. Ceci permet de stocker dans une bibliothèque d'objet préfabriqués (i.e des prefabs), non seulement des objets qui peuvent apparaitre plusieurs fois dans la scène, mais aussi la manière dont les différents rôles d'utilisateur pourront interagir avec ces objets. L'intérêt, est de permettre aux utilisateurs d'interagir instantanément avec le contenu lors de son instanciation, sans avoir à générer par code les nouvelles interactions. Le graphe des états utilisateur est donc dynamique et évolue automatiquement lors de l'instanciation de ces patrons d'interaction. Prenons l'exemple de l'interaction avec un ensemble de cubes dont le sous-graphe d'états utilisateur est similaire à celui présenté Figure 3.4. Nous définissons de plus un filtre pour bloquer les manipulations concurrentes du cube. Enfin nous plaçons dans la scène les états utilisateur et le filtre de sorte à ce que le cube soit l'un de leurs parents communs. Dès lors, à chaque fois que le cube est dupliqué, le nouveau cube peut instantanément être manipulé de la même manière que le cube d'origine. Enfin, l'évolution du graphe des états utilisateur permet aux utilisateurs de passer directement d'un cube sélectionné à un autre sans manipulation concurrente. En effet, le graphe d'état du cube faisant partie de l'objet préfabriqué (prefab) permettant l'instanciation d'un nouveau cube, la manière d'interagir avec cet objet est automatiquement définie à l'instanciation. Par ailleurs, si le graphe présenté Figure 3.4 interdit la sélection d'un cube déjà sélectionné, il n'impose pas que l'utilisateur soit dans l'état par défaut (aucun cube sélectionné) pour sélectionner un autre cube. Un utilisateur peut donc relâcher un cube en en sélectionnant un autre.

3.2.4.2 Modifications à l'exécution

L'utilisation de Unity3D, comme un serveur pour nos médias 3D, permet d'utiliser l'éditeur graphique du moteur de jeu pour ajouter ou modifier des mécanismes de coordination pendant l'exécution du média. S'il n'est pas possible d'implémenter de nouveaux

scripts à l'exécution, l'instanciation ainsi que le paramétrage de comportements est possible. Ceci permet par exemple d'ajouter, de supprimer ou encore de modifier des contenus 3D, des interactions et des filtres à l'exécution de sorte à redéfinir les mécanismes de coordination. Si cette approche rend le système hautement malléable (c'est à dire permet une redéfinition à l'exécution des mécanismes de coordination), elle a comme contrainte forte de demander que l'un des acteurs de la collaboration, ou un superviseur, maîtrise Unity 3D ainsi que notre boîte à outil pour permettre aux acteurs de redéfinir eux-même la manière dont il travaillent.

Une autre possibilité pour apporter de la malléabilité aux environnements UMI3D consiste à utiliser les briques d'interaction Action et Direct Input (voir Figure 2.7) pour laisser aux utilisateurs le contrôle de certains paramètres de l'application. Cette solution apporte un plus faible niveau de malléabilité, mais ne requiert pas de compétences de développement de la part des utilisateurs.

Pour finir, nous avons implémenté un comportement abstrait (présenté Listing 3.5) que les patrons d'interaction ou de collaboration doivent hériter pour simplifier leur instanciation, et surtout leur remplacement à l'exécution. Le concepteur de l'application regroupe un ensemble de comportements et de propriétés d'un objet virtuel dans un MonoBehaviour A (i.e. un nouveau script). Il devient alors possible de concevoir des patrons d'interaction dédiés aux objets de type A. Dès lors, tout objet instanciant le comportement A ne peut déployer qu'un unique comportement InteractionTemplate<A> à un moment donné. Toutefois, si un objet dispose de plusieurs MonoBehaviour regroupant différentes propriétés (par exemple si les propriétés de l'objet sont organisées en différents médias), il est possible de déployer simultanément un patron d'interaction pour chaque groupe de propriétés (i.e. pour chaque MonoBehaviour). Enfin, un patron d'interaction étant un objet préfabriqué, il peut lui-même contenir des objets 3D, et il est donc possible de définir de nouveaux patrons pour ces objets. En conséquence, les patrons d'interaction et de collaboration peuvent être imbriqués.

Nous fournirons d'avantage de détails sur les possibilités de modification du média à l'exécution dans l'étude de cas présentée section 3.3.

3.2.5 Interface de conception

L'interface de conception de notre boîte à outils est présentée Figure 3.5. L'ensemble des fonctionnalités présentées jusqu'ici sont directement intégrées à l'interface graphique de Unity. Il est donc possible de combiner toute fonctionnalité de Unity avec les fonctionnalités de la boîte à outils. Nous fournissons de plus un panneau sur mesure pour la conception du graphe des états utilisateur. Cette interface graphique permet au concepteur d'ajouter, de supprimer, ou d'éditer les états utilisateurs, ainsi que les transitions d'état. Par ailleurs, il est possible de créer des sous-graphes pour rendre le graphe principal plus

```
public abstract class InteractionTemplates<A> : MonoBehaviour where A:MonoBehaviour
{
    public abstract void Load();
    public abstract void Destroy();
    public A target;

    public static void Load(InteractionPattern pattern, Object3D obj)
    {
        DestroyExistingTemplate(obj);
        InteractionTemplates<A> instance = Instanciate(pattern);
                                instance.target = target;
    }

    public static void DestroyExistingTemplate(Object3D obj)
    {
        ...
    }

    private void Start()
    {
        Load();
    }

}
```

Listing 3.5 – Déploiement à l'exécution de patrons d'interaction et de Collaboration

lisible. L'usage principal de ces sous-graphes est le stockage du graphe d'interaction des objets de la scène. Par exemple, le graphe d'interaction du cube présenté Figure 3.4 devrait être un sous-graphe du graphe d'états du média. Ces sous-graphes peuvent être visualisés depuis leur graphe parent grâce à un bouton 'ouvrir'. Dans la Figure 3.5, c'est le cas des deux états les plus à droite du graphe visible dans la Figure 3.5 (partie 4 de l'interface).

3.2.6 Processus de conception

Dans la Figure 3.6, nous proposons un processus pour la conception des médias 3D avec notre boîte à outil. Premièrement, des tâches de développement ('scripting' sur la figure) sont nécessaires afin d'implémenter le comportement interne du média (i.e. ses règles métier), ainsi que pour créer des patrons d'interaction et de collaboration. Ces tâches ne peuvent pas être réalisées à l'exécution, d'une part car le moteur Unity 3D doit compiler chaque script C# hors exécution avant qu'il ne soit disponible pour le concepteur, et d'autre part car ces implémentations doivent être suffisamment testées avant d'être présentées à des utilisateurs finaux. Deuxièmement, des sessions de conception à l'exécution ('run-time design' sur la figure) permettent au concepteur de modifier les contenus, les interactions, et les mécanismes de coordination à l'exécution. En parallèle, des utilisateurs potentiels peuvent tester le média en collaborant à l'aide de différents supports. Ceci permet au concepteur de recevoir un retour immédiat sur le média, et ainsi de l'améliorer. Finalement, le point clé pour une application optimale de ce processus réside dans l'identification préalable des besoins utilisateur en termes de contenus, d'interaction et de collaboration. Ceci permet idéalement de développer une bibliothèque de contenus et d'interactions suffisante pour ne pas multiplier les tâches de développement postérieures au début des sessions de conception à l'exécution. On pourrait ainsi regrouper l'ensemble de ces sessions pour limiter la mobilisation des utilisateurs finaux à un unique atelier de

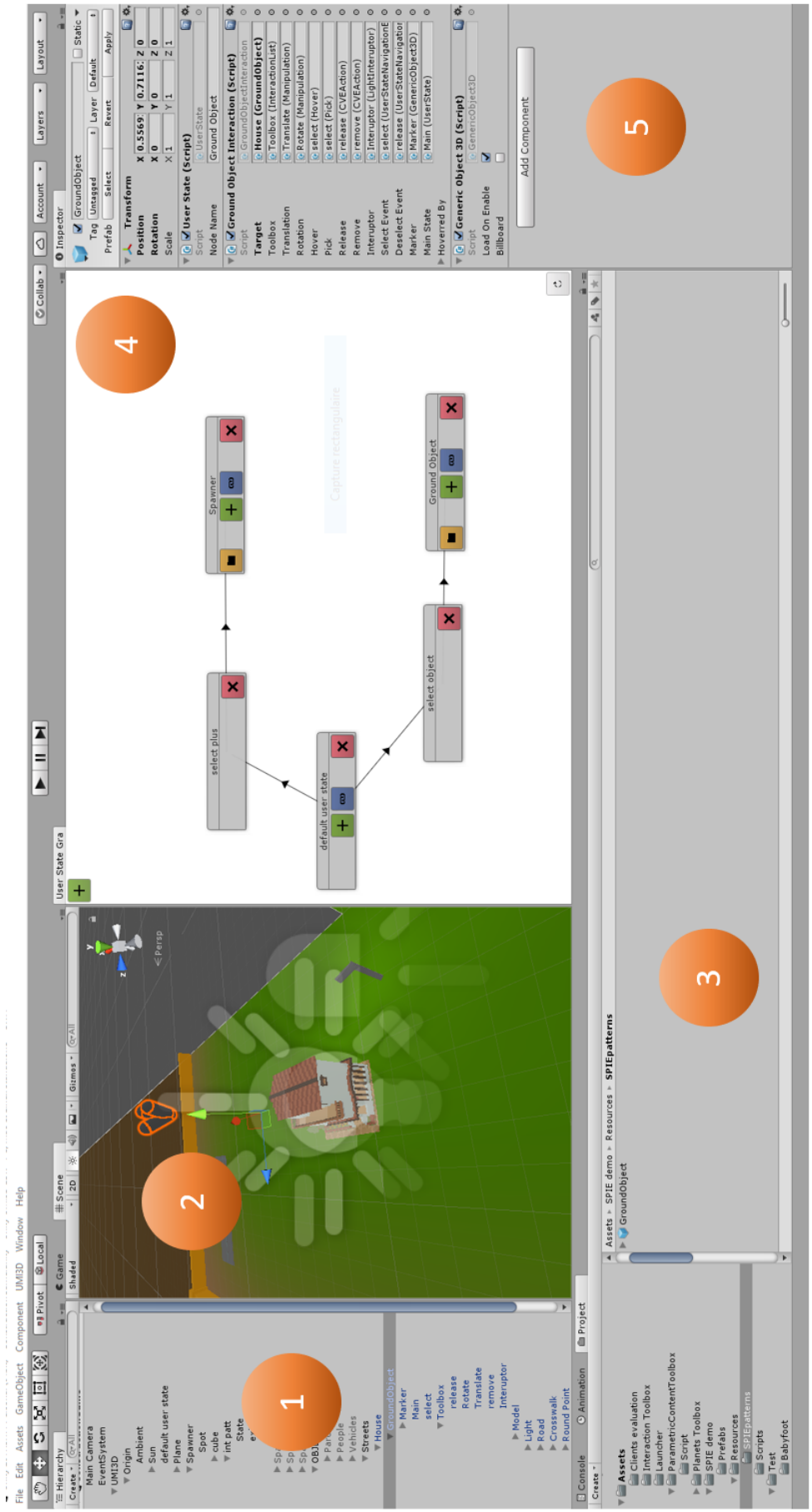


FIGURE *3.5 – 1 : Hiérarchie de la scène Unity ; 2 : Éditeur de scène de Unity ; 3 : Bibliothèque d'objets et de scripts MonoBehaviour ; 4 : Graphe des états utilisateur ; 5 : Propriétés de l'objet sélectionné*

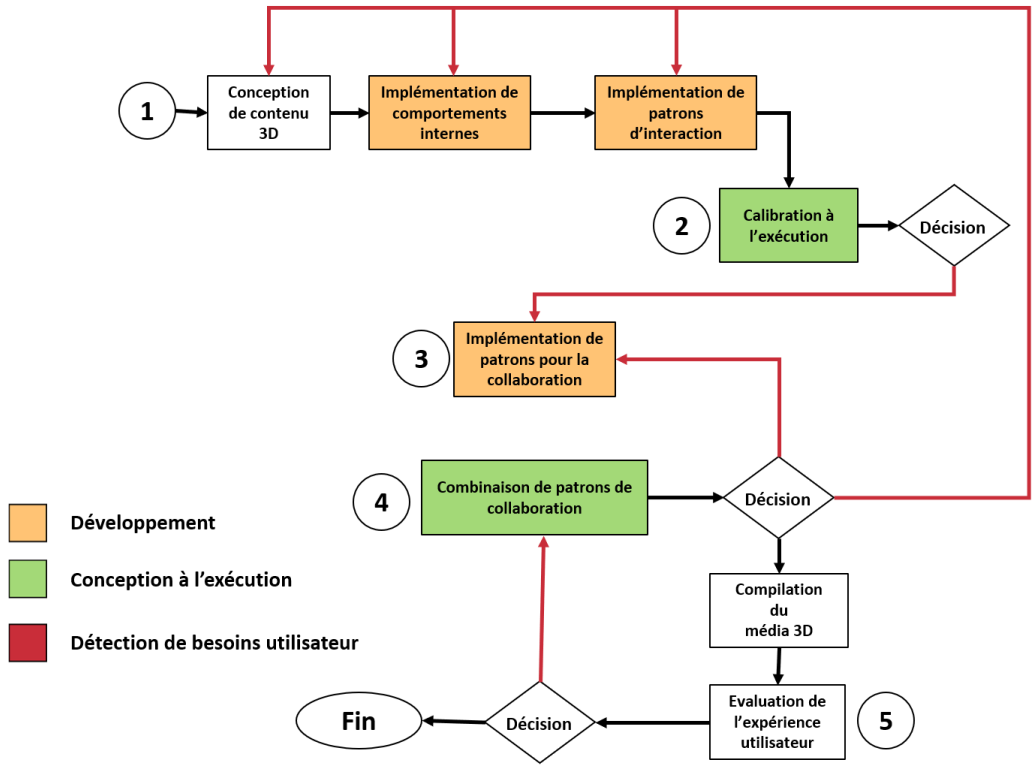


FIGURE *3.6 – Proposition d'un processus de conception des médias 3D*

co-conception du média 3D. La section suivante sera intégralement consacrée à une étude de cas dans laquelle nous utilisons ce processus de conception pour créer un média 3D.

3.3 Étude de cas : co-conception d'un jeu de construction collaborative

Cette section présente un exemple concret de média 3D conçu grâce à notre boîte à outils. Ce média consiste en un jeu de construction collaborative (Figure 3.7). Notre objectif initial était de permettre la construction d'un village en positionnant des habitations, des rues, des parcs ou encore des véhicules. Pour cela, trois supports, avec les

FIGURE *3.7 – Étude de cas*

navigateurs UMI3D associés, ont été proposés aux utilisateurs : un ordinateur de bureau proposant une interaction classique avec souris et clavier (dispositif 1), un Oculus Rift et ses deux contrôleurs Touch à 6 degrés de liberté (dispositif 2), ainsi qu'un Oculus Rift proposant une interaction directe avec les mains grâce à un capteur Leap Motion (dispositif 3). Nous avons toutefois été amenés à étendre les objectifs et fonctionnalités du média initial, suite à l'identification de nouveaux besoins et souhaits des utilisateurs finaux, durant les sessions de conception à l'exécution. La Figure 3.8 résume ces évolutions autant dans le contenu proposé, que dans la définition des interactions et dans l'organisation même des phases de travail. Dans la suite de cette section, nous présentons les différents patrons d'interaction et de collaborations implémentés pour ce cas d'usage. Nous relatons ensuite le déroulé et le rôle des sessions de 'conception à l'exécution', dans le processus de conception du média. Nous expliquons également les interactions supplémentaires que nous avons définis pour apporter de la malléabilité dans la répartition des tâches et les objectifs des utilisateurs. Finalement, nous analysons les résultats d'une étude utilisateur montrant la manière dont des utilisateurs potentiels ont réagis face au média final.

3.3.1 Patrons d'interaction et de collaboration

L'ensemble des patrons d'interaction et de collaboration que nous présentons ici sont hérités du patron de sélection et manipulation d'objet déjà présenté dans la Figure 3.1. Lorsque l'objet est survolé par l'outil de sélection d'un utilisateur, un retour visuel individuel est donc produit pour cet utilisateur. Si l'utilisateur sélectionne ensuite cet objet, l'utilisateur passe dans l'état principal du sous-graphe d'état de l'objet. Dès lors, un retour virtuel partagé à tous les autres utilisateurs leur indique que l'objet est manipulé par un collaborateur. Le retour visuel est différent pour l'utilisateur ayant sélectionné l'objet de sorte à ce qu'il différencie cet objet de ceux manipulés par les autres utilisateurs. De plus, l'utilisateur obtient différentes interactions avec l'objet sélectionné selon la nature exacte du patron d'interaction déployé sur l'objet. Enfin, si les sélections concurrentes d'objet sont interdites par le graphe des états utilisateur, les utilisateurs peuvent avoir des interactions concurrentes en cas de collision entre les objets qu'ils manipulent.

Le média contient quatre types de patrons de sélection et manipulation d'objet : Objets urbains (Ground object), Véhicules (Vehicles), Personnages (characters) et Créateurs de contenus (content spawners).

3.3.1.1 Objets urbains

Les objets urbains constituent l'ensemble des objets 3D qui peuvent être déplacés sur le plateau de jeu à deux dimensions (par exemple des habitations, des arbres, ou encore des routes). Ces objets urbains peuvent être translatés en 2D dans le plan du plateau, ou tourné autour de l'axe normal au plateau. Pour cela, nous utilisons la brique d'interaction

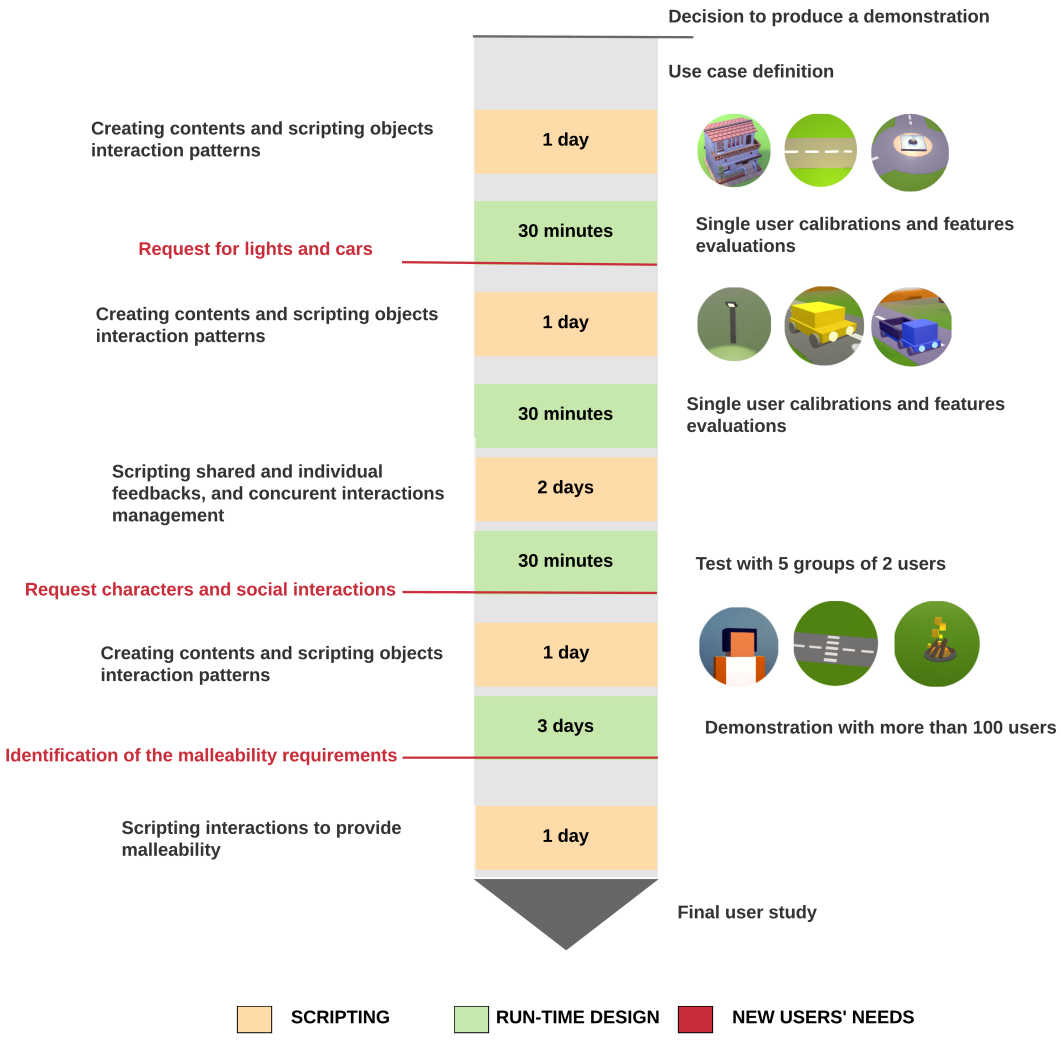


FIGURE 3.8 – *Étude de cas : historique de conception*

FIGURE *3.9 – Interaction avec les objets urbains. A gauche, l'utilisateur 'User 2' manipule un lampadaire. À droite, l'utilisateur 'User 1' manipule un arbre.*

Manipulation. Par ailleurs, le moteur physique de Unity est utilisé pour détecter les collisions entre les objets urbains, et pour les maintenir à l'intérieur de l'espace de jeu. Ces interactions physiques ne sont actives que pendant la manipulation des objets. Enfin, les patrons de collaboration des objets urbains produisent des retours visuels asymétriques durant la manipulation des objets. Un indicateur blanc lors du survol d'un objet urbain signifie que cet objet est disponible à la sélection. Cet indicateur blanc est individuel et n'est donc pas visible pour les autres utilisateurs. L'indicateur devient jaune pour l'utilisateur s'il a sélectionné l'objet. Dans ce cas, l'indicateur devient visible pour les autres utilisateurs et prend pour eux la couleur rouge qui indique la manipulation par un autre utilisateur. L'ensemble de ces retours visuels, résumé Figure 3.9, renforce la conscience des autres utilisateurs durant la manipulation des objets urbains.

3.3.1.2 Personnages et Véhicules

Les personnages et les véhicules sont des extensions des objets urbains (i.e. leurs classes respectives héritent de la classe GroundObject). Un pattern d'interaction dédié est disponible pour chacun de ces deux types d'objet. Ainsi, ils peuvent déployer leur propre patron d'interaction ou celui commun aux objets urbains selon la situation. Le patron d'interaction des véhicules fournit une interaction de conduite 'drive' en remplacement de l'outil de translation et rotation des objets urbains. De la même manière, le patron d'interaction propre aux personnages fournit une interaction de marche à pied 'walk'. Cette interaction génère, dans son retour visuel, une animation des jambes du personnage manipulé. Enfin, suite à une demande des utilisateurs en session de conception à l'exécution, nous avons ajouté la possibilité de se munir/démunir d'un morceau de guimauve dans l'interaction avec un personnage (à l'aide de la brique d'interaction Action).

FIGURE *3.10 – Interactions sociales. User 2 (à gauche) attend que le personnage, contrôlé par User 1 (à droite), traverse la route avant de continuer à conduire la voiture rouge.*

3.3.1.3 Créateurs de contenus

Plusieurs créateurs de contenus, répartis aux coins du plateau de jeu, permettent aux utilisateurs d'instancier de nouveaux objets urbains, personnages et véhicules durant la partie. Un nouvel objet est automatiquement sélectionné par l'utilisateur l'ayant créé de sorte à ce qu'il le positionne à l'endroit de son choix. Cela permet de plus de libérer immédiatement le créateur de contenu qui redevient accessible pour les autres utilisateurs.

3.3.2 Sessions de conception à l'exécution

3.3.2.1 Calibration des interactions

Le but de la calibration d'une interaction, et plus particulièrement d'une manipulation, consiste à définir des coefficients de sensibilité en fonction d'un ensemble d'indicateurs choisis au préalable (précision, temps d'exécution de la tâche, facilité de prise en main). Dans notre cas, ne disposant pas à ce stade d'outils de mesure objectifs (voir Chapitre 5), la calibration consiste à un réglage de la sensibilité, pendant l'exécution de la tâche, en fonction des retours du testeur.

Pour cette étude, nous avons utilisé trois navigateurs UMI3D existants et déjà calibrés. Ceci signifie que ces navigateurs ont été configurés de sorte à générer des amplitudes de mouvement similaires lors de l'interprétation d'une même brique d'interaction Manipulation (voir Figure 2.7). C'est pourquoi nous avons pu réduire les tests de jeu (playtest), des différentes interactions et patrons de conception précédemment présentés, à un seul support. Ces tests consistent principalement en la recherche des coefficients de sensibilité pour les différentes briques d'interaction de type Manipulation. De cette manière, nous définissons à l'exécution les vitesses de translation et de rotation des objets urbains (ceci inclus les personnages et véhicules) de sorte à obtenir un compromis entre la vitesse d'exécution des tâches et leur précision. Après avoir calibré ces interactions avec le navigateur

UMI3D dédié aux ordinateur de bureau, nous avons réalisé les mêmes tests avec les autres supports de sorte à vérifier la bonne calibration des navigateurs. Concrètement, nous exécutons de manière indépendante la calibration avec chaque support, puis vérifions que les coefficients de sensibilité choisis sont subjectivement proches. Dans le cas de navigateurs bien calibrés, les différences constatées doivent être minimes. Si nous avons rapidement détecté que certains supports étaient plus adaptés à certaines tâches, ou simplement plus précis pour ces tâches, nous n'avons pas trouvé de différence notable dans la meilleure calibration des interactions. Ce qui confirme la validité de la calibration de nos navigateurs et garanti qu'un seul navigateur est suffisant pour calibrer les interactions de notre média 3D.

3.3.2.2 Définition des mécanismes de coordination

Après avoir validé les patrons d'interaction des objets 3D, nous avons développé des patrons de collaboration spécifiques à notre cas d'étude. Par exemple, nous avons créé l'indicateur de sélection déjà présenté section 3.3.1.1. Pour ce faire, nous avons commencé par créé la géométrie de l'indicateur à partir des primitives géométriques de Unity (sphère, plan, cube, cylindre). Nous avons ensuite défini dans une session de conception à l'exécution, les règles de visibilité de notre indicateur. Nous avons également utilisé cette session pour définir les règles définissant la couleur de l'indicateur en fonction du contexte. Le résultat le plus intéressant de la session a été la mise en évidence, par les utilisateurs, de la nécessité d'avoir une couleur d'indicateur asymétrique (voir Figure 3.9), lors de la sélection d'un objet, pour distinguer un objet sélectionné par un autre utilisateur de l'objet que l'on a soi-même sélectionné. Enfin, la manière dont les utilisateurs passent d'un objet sélectionné à un autre a également été choisie dans cette session.

3.3.2.3 Identification de nouveaux besoins utilisateur

Si les sessions de conception à l'exécution ont largement satisfait nos attentes de part les possibilités de conception rapide offertes, elles ont également montrées des propriétés intéressantes pour augmenter l'engagement des utilisateurs finaux dans le processus de conception des médias 3D. En effet, les itérations très courtes de notre processus de design ont selon nous permis de faire entrer les utilisateurs dans une démarche de co-création d'une application les satisfaisant. Cet engagement accrus nous a permis d'identifier de nouveaux contenus, comme les personnages et les éclairages, à ajouter au média. Cette démarche a été poussée jusqu'à la proposition d'un nouveau mode de jeu qui n'avait pas été spécifié initialement. En effet, les utilisateurs ont exprimé une certaine frustration à ne pas pouvoir jouer, avec des interactions adaptées, dans le village qu'ils venaient de créer. C'est pourquoi nous avons conçus des interactions dédiées pour ce mode 'social' (voir Figure 3.10 et la section 3.3.1.2). De plus, lors de l'évaluation de ce mode, les utilisateurs

nous ont régulièrement demandé à alterner entre ces deux modes d'interaction selon qu'ils souhaitaient construire (ou modifier) le village, ou y jouer. Ceci a généré un besoin de malléabilité supplémentaire, pour laisser la possibilité aux utilisateurs de basculer à tout instant du mode de jeu 'social' au mode 'construction' et inversement.

3.3.3 Paramètres de malléabilité

Afin de fournir une certaine malléabilité aux utilisateurs finaux, sans demander l'intervention d'un concepteur d'environnement UMI3D, nous avons utilisé des briques d'interaction pour leur permettre de configurer certains paramètres identifiés en session de conception à l'exécution. Tout d'abord, nous avons implémenté deux rôles utilisateur. Le 'master' (maître en anglais), qui a la possibilité de configurer les paramètres de malléabilité et de transmettre cette responsabilité au 'player' (joueur) de son choix. Nous avons ensuite définis, à l'aide de briques Direct Input, des interactions permettant au 'master' de définir les types de contenus pouvant être ajoutés par chaque 'player'. Ceci permet aux utilisateurs de définir ensemble une répartition des tâches. Enfin, l'utilisateur 'master' contrôle le basculement entre les modes de jeu 'social' et 'construction'.

3.3.4 Étude utilisateur

Dans le but de vérifier que les requis, établis pour un système de support de la collaboration, sont bien observés lors de l'utilisation de notre média 3D, nous avons conduit une étude utilisateur. Pour ce faire, nous avons demandé à huit groupes de deux utilisateurs de construire un village, d'y reproduire certaines interactions sociales, puis de s'évaluer eux même en répondant à un questionnaire. Les participants étaient des personnes volontaires de notre organisation. Ils avaient des expériences très variées de l'interaction en environnement virtuel (incluant aussi bien des personnes expérimentant la RV pour la première fois que des professionnels du domaine).

Limitation des problèmes de communication. Dans le but de limiter les effets négatif sur la collaboration liés au manques moyens de représentation des utilisateurs (à ce stade des travaux) et de communication distante, nous avons conçu un média 3D peu immersif et adapté nos conditions expérimentales. Le contenu interactif du média 3D étant en effet limité à un plateau de jeu, les utilisateurs de supports de RV n'évoluent pas dans un environnement interactif à l'échelle 1. Par ailleurs, le choix d'une collaboration limitée à deux utilisateurs maximise la conscience des autres. Il n'est en effet pas nécessaire pour un utilisateur de déterminer lequel de ses collaborateurs est à l'origine des transformations du média qu'il observe. Par ailleurs les retours visuels présentés Figure 3.9 facilitent la communication entre utilisateurs. Pour finir, Les espaces physiques des deux utilisateurs

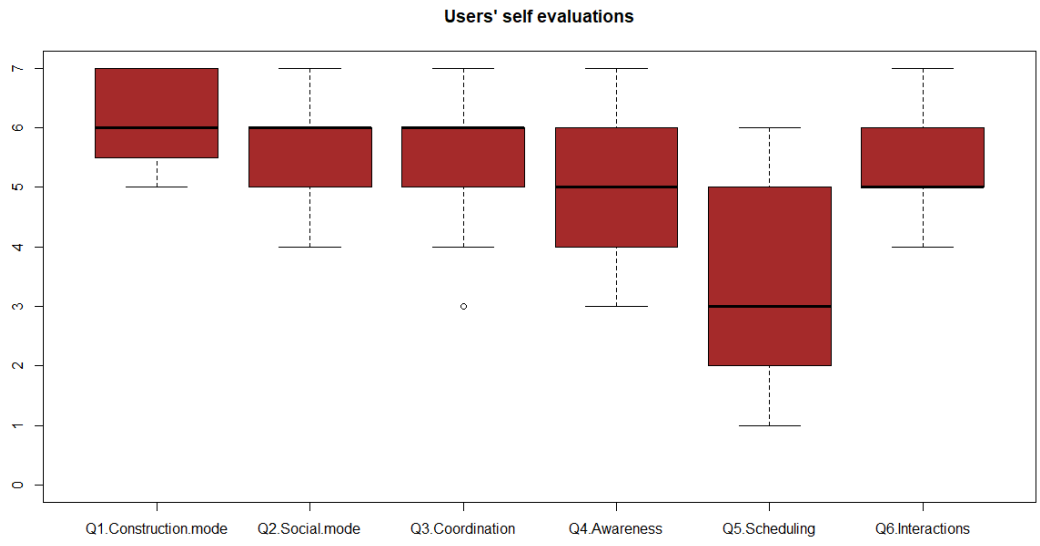


FIGURE *3.11 – Résultats de l'étude utilisateur*

ont été positionnés l'un à côté de l'autre de sorte à permettre la communication vocale entre eux.

Questionnaire utilisateur. Dans le questionnaire, nous leur avons demandé d'évaluer différents aspects de l'application à l'aide d'échelles de Likert (Une partie des résultats est présentée Figure 3.11). Cela incluait des questions générales sur la satisfaction utilisateur lors des modes de jeu 'construction' et 'social' (Q1 : "Êtes-vous satisfait par le village que vous avez construit ?", Q2 : "Comment évaluez vous le jeu dans le village"), une auto-évaluation de la collaboration avec leur partenaire de jeu (Q3 : "Arriviez-vous à vous coordonner ?") ainsi que de la conscience des autres (Q4 : "Compreniez-vous bien les actions de votre partenaire ?"), et une évaluation de la qualité des interactions (Q6). Enfin, nous avons laissé aux utilisateurs suffisamment de place pour écrire librement leur commentaires et ainsi mieux comprendre leurs évaluations.

Les résultats de cette étude, présentés Figure 3.11, montrent que tous les utilisateurs se sont déclarés satisfait par le mode 'construction'. La plupart d'entre eux ont également une bonne opinion de l'expérience proposée par le mode 'social'. Par ailleurs, une majorité d'utilisateur a été convaincue par la qualité des interactions proposées. Certains utilisateurs nous ont par contre notifié un manque de précision dans les tâches de positionnement lors de l'utilisation des navigateurs UMI3D pour Oculus Rift (dispositifs 2 et 3). Ce défaut de précision est logiquement particulièrement marqué pour le dispositif 3, basé sur l'interaction avec les mains, du fait de la difficulté pour l'utilisateur à arrêter la manipulation au moment souhaité. Ceci s'explique par l'impossibilité pour l'utilisateur à contrôler finement la détection de fin de geste réalisée par le capteur Leap Motion. Ces manques de précision ne constituent à nos yeux pas un défaut de conception des navigateurs UMI3D, mais sont inhérents aux propriétés et qualités des supports utilisés.

Lors de l'expérimentation, nous avons pu constater la capacité des utilisateurs à tirer parti des complémentarités entre supports pour surmonter ces difficultés. Cette prise de conscience, sans intervention de notre part, des capacités propres aux supports a poussé les utilisateurs à modifier la répartition des tâches de sorte à optimiser leur performance commune. En particulier, certains utilisateurs ont réorganisé leur travail de sorte à ce que l'orientation précise des objets urbains soit réalisée à la souris :

> "Les difficultés rencontrées avec un dispositif sont surmontées par les autres dispositifs, il faut se coordonner. Par exemple, le manque de précision de la rotation est résolu en utilisant la souris."

Nous avons par ailleurs obtenu une plus grande disparité des résultats concernant la conscience des autres (Q4) et planification des tâches (Q5). La disparité en terme de conscience des autres peut s'expliquer par la disparité de notre population test vis à vis de leur fréquence d'usage de la RV et plus particulièrement des Hmd. Nous avons effectivement constaté que les plus faibles résultats à cette question (Q4) provenaient d'utilisateurs ayant utilisé un Hmd pour la première fois durant notre expérience. Leurs commentaires montrent que la découverte et l'apprentissage de l'interface ont focalisés une partie considérable de leur attention, réduisant ainsi le temps passé à observer les actions de leur collaborateur. La disparité des résultats quand à la planification des tâches nous a semblé plus surprenante. Pour évaluer ce point, nous avions demandé aux utilisateurs s'ils avaient été dérangé par les objets, qu'ils avaient préalablement positionnés dans la zone de jeu lors de la construction (Q5). Les commentaires et nos observations montrent qu'une partie des utilisateurs (ceux ayant majoritairement réussis à planifier la construction) ont testé le média 3D en début de construction, et ainsi découvert les règle physiques de l'environnement. Ces utilisateurs ont adapté leur planification à l'impossibilité de faire passer un objet urbain à travers un autre (i.e. du fait de la détection des collision entre objets). Les autres utilisateurs ont découvert ces règles en cours de partie, ce qui les a obligé à déconstruire une partie de leur travail. Ce contretemps a généré une certaine frustration chez ces utilisateurs. Une solution pourrait être l'ajout d'un paramètre de malléabilité supplémentaire permettant de temporairement désactiver la détection des collisions dans le média.

3.4 Discussion

Dans ce chapitre nous avons présenté le contenu de notre boîte à outils dédiée à la conception de médias 3D avec UMI3D. Nous avons par ailleurs illustré la manière dont ces outils peuvent être utilisé pour répondre aux attendus vis à vis des systèmes de support du travail coopératif (CSCWS). Nous avons en particulier démontré le support de toutes les fonctionnalités pointées comme 'à vérifier' Tableau 3.1. Nous avons par ailleurs réalisé une étude de cas concrète en concevant un média 3D permettant à des

utilisateurs de collaborer avec des dispositifs asymétriques (Hmd et écran 2D). Nous sommes donc en mesure d'affirmer que UMI3D est pleinement adapté au support du travail collaboratif dans le cas d'une collaboration non immersive. Pour que UMI3D soit adapté à la collaboration dans toutes les situations, une extension du protocole permettant une représentation des utilisateurs (i.e. des avatars) sera présentée dans le Chapitre 4. Enfin, des travaux supplémentaires pourraient être menés pour vérifier qu'il est bien possible d'atteindre par une interface graphique, à la portée d'utilisateurs finaux, le haut niveau de malléabilité fourni par la boîte à outils au concepteur du média 3D.

Chapitre 4

Travaux Préliminaires pour le Suivi des Utilisateurs et l'Animation d'Avatars

Ce chapitre présente des travaux préliminaire, réalisés en collaboration avec Nicolas Pacqueriaud. L'objectif de ces travaux est d'étendre le protocole UMI3D de sorte à inclure un système, indépendant du support, pour le suivi utilisateur. Un tel système permettrait notamment la représentation des utilisateurs par des avatars.

De nombreuses études ont été menées sur la conception (et l'animation) d'avatar et sur l'impact de leur présence sur les utilisateurs. Ce chapitre s'appuie donc sur une présentation de ces travaux, puis propose de les adapter à notre architecture des médias 3D. Nous nous sommes en particulier intéressé à la normalisation des données décrivant le mouvement de l'utilisateur, ainsi qu'à la possible adaptation par le média 3D de l'animation et/ou de l'apparence des avatars en fonction de ces mêmes données.

4.1 Travaux relatifs à la représentation des utilisateurs en environnement 3D

4.1.1 Avatars et théorie de l'incarnation

Dans [8], Benford et al. définissent les avatars comme étant "la mise à disposition des utilisateurs d'images corporelles appropriées permettant de représenter pour les autres utilisateurs et également pour eux-mêmes". Les auteurs ajoutent de plus que ces représentations permettent de situer les utilisateurs en environnement virtuel, ainsi que de partager leur identité et leur activité.

Dans [50], Kilteni et al. affirment que l'usage de l'immersion en RV a permis d'étudier d'en quelle mesure il est possible de ressentir les mêmes sensations envers un corps virtuel (dans un environnement virtuel immersif), qu'envers notre corps biologique. Dans cet article, l'auteur aborde cette question en faisant référence à l'incarnation ("Sense of Embodiment") qu'il définit comme une structure composée :

— du sens de l'auto-localisation ("Sense of Self-Location") qui correspond l'impression

de savoir où et comment les membres d'un corps sont situés dans l'espace.
— du sens de la capacité d'action ("Sense of Agency") qui correspond à un sentiment de contrôle de ses actions et de leurs conséquences.
— du sens de la propriété du corps ("Sense of Body Ownership") qui correspond à la reconnaissance d'un corps comme étant son propre corps.

Enfin, des études expérimentales sont proposées pour l'amélioration de l'incarnation durant l'immersion en environnement virtuel.

4.1.2 Création et Représentation

Un avatar a pour objet la représentation d'une entité dans un espace différent de son espace d'origine (c'est à dire son environnement physique pour un utilisateur). Des travaux ont donc été menés pour développer des processus de création d'un avatar reproduisant l'apparence d'un utilisateur. Par ailleurs, des outils de personnalisation de l'avatar son parfois mis à disposition de l'utilisateur.

Dans [26], De Groot décrit la création d'un système de Réalité Virtuelle pouvant accueillir plusieurs utilisateurs simultanément. Ce système permet aux utilisateurs d'interagir entre eux et/ou avec les objets de l'environnement. Ce système contient en particulier un module gérant la création d'un avatar pour chaque utilisateur. Ce module est interfacé avec d'autres modules esclaves qui sont dédiés à la gestion de la visibilité des avatars. Chacun de ces modules prend en charge l'animation d'un utilisateur particulier dans l'espace virtuel des autres utilisateurs. De Groot précise de plus qu'il est possible d'animer certaines parties des avatars grâce aux mouvements de périphériques d'entrées (comme par exemple des gants de Réalité Virtuelle ou des traqueurs de position). A chaque interaction, les différents modules esclaves reçoivent des instructions du module maître pour définir leurs mouvements.

Ebrahimi et al. [33] testent quand à eux l'influence du niveau de fidélité de l'avatar d'un utilisateur sur son estimation des distances dans un environnement virtuel. Dans cette étude, nous nous intéressons plus particulièrement aux différentes représentations mises en place par Ebrahimi et al. pour leurs expérimentations. Ces représentations mettent en œuvre trois niveaux de détails différents pour les avatars. Le premier consiste à utiliser un avatar complet immersif, de taille similaire à l'utilisateur. Le second niveau correspond à un niveau de fidélité plus faible et ne représente que les jointures du corps(épaules, coudes, poignets). Enfin, le dernier mode de représentation testé dans leurs expérimentations consiste à ne rendre visible que les contrôleurs utilisés par l'utilisateur.

D'autres méthodes de création d'avatars pour environnements virtuels collaboratifs (CVE) sont abordées dans [37] par Gamelin et al. afin de donner un cadre théorique à leur étude de l'impact de la fidélité des avatars. Leurs travaux se focalisent sur deux types de représentation. Leur première approche consiste à reconstruire dans l'environnement

virtuel un nuage de points à partir de la vidéo d'un utilisateur enregistrée par un capteur de profondeur. Leur seconde approche repose sur la création en amont d'un avatar à l'image de l'utilisateur et son animation en temps réel grâce à un capteur. Les mouvements reconnus sont appliqués à un squelette pour animer l'avatar.

D'autres travaux se concentrent quand à eux sur la personnalisation d'avatars virtuels [31, 2]. Dans [31], Ducheneaut et al. s'intéressent à la manière dont les utilisateurs personnalisent leurs avatars ainsi qu'aux outils à leur disposition pour ce faire. Il en ressort une tendance de l'utilisateur à tenter de se rapprocher d'une apparence idéale, tout en s'éloignant de son corps réel et des défauts qu'il s'attribue. De plus, les participants tendent à passer plus de temps sur la personnalisation des parties immédiatement visibles et reconnaissables, tels que les cheveux par exemple. Dans [2], Ahmed et al. nous présentent une méthode de construction d'avatars personnalisés (taille, proportions) à partir de vidéos d'une personne en activité, prises depuis différents points de vue. Il est alors possible d'adapter un modèle par défaut avec les enregistrements et d'y appliquer les textures reconnues grâce aux différents points de vue.

Enfin, Hasenfratz et al. présentent dans [40] un système permettant de reconstituer une représentation 3D à partir de plusieurs cubes pouvant être lissés par traitement d'images sur un flux vidéo. Il est alors possible pour l'utilisateur d'interagir physiquement et en temps réel avec les objets de l'environnement par le biais de son avatar reconstitué.

4.2 Extension de UMI3D pour le suivi des utilisateurs

Les objectifs de ces premiers travaux sur le support des avatars par UMI3D sont, d'une part de permettre au média 3D de suivre l'évolution des utilisateurs dans la scène 3d le composant, et d'autre part de fournir aux concepteurs de médias 3D les outils nécessaires à la conception et l'animation d'avatars dans un média 3D.

La principale difficulté à implémenter un suivi des mouvements de l'utilisateur par un média 3D (utilisant UMI3D) provient du fait que le média 3D ne connait pas la nature du dispositif de l'utilisateur. En effet, la quantité d'information disponible sur le mouvement utilisateur peut varier d'une simple position caméra pour un écran 2D à un suivi complet de l'ensemble du corps pour un support de VR équipant chaque partie du corps d'un capteur de position. Dans le but de fournir au média l'ensemble des informations disponible, il s'agit de déterminer un modèle de données permettant une description partielle ou exhaustive des mouvements de l'utilisateur.

Nous proposons de baser ce modèle de données (résumé Figure 4.1) sur la transmission des coordonnées de points d'articulations du corps humain. Le navigateur UMI3D de chaque support utilise ce modèle pour envoyer en temps réel (30 à 90 Hz) la position

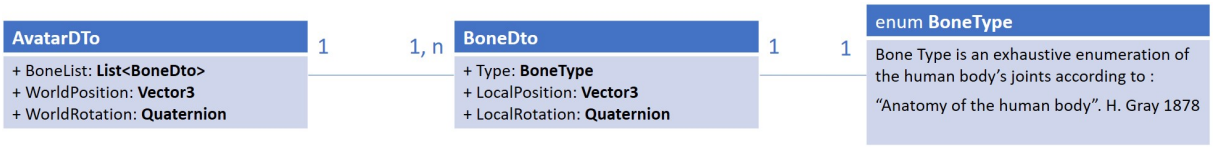


FIGURE *4.1 – Modèle de données pour le suivi utilisateur dans UMI3D.*

de points d'articulation connus ou interpolés par le support. Les points d'articulations dits interpolés correspondent à des points dont la position exacte n'est pas mesurée par un capteur, mais calculée à partir de la position des points d'articulation mesurés et des connaissances du navigateur sur le contexte d'usage du support. Il est par exemple possible d'utiliser l'approche présentée dans [11], et illustrée Figure 4.2, pour interpoler certaines articulations à partir des relations physiques entre les articulations et les membres.

Pour finir, de sorte à ce que toute donnée disponible sur le mouvement utilisateur puisse être transmise (ce qui inclut les données qu'aucun dispositif actuel n'est en mesure de capturer) et utilisée par le média 3D, nous définissons nos articulations parmi une liste exhaustive des articulations du corps humain [38].

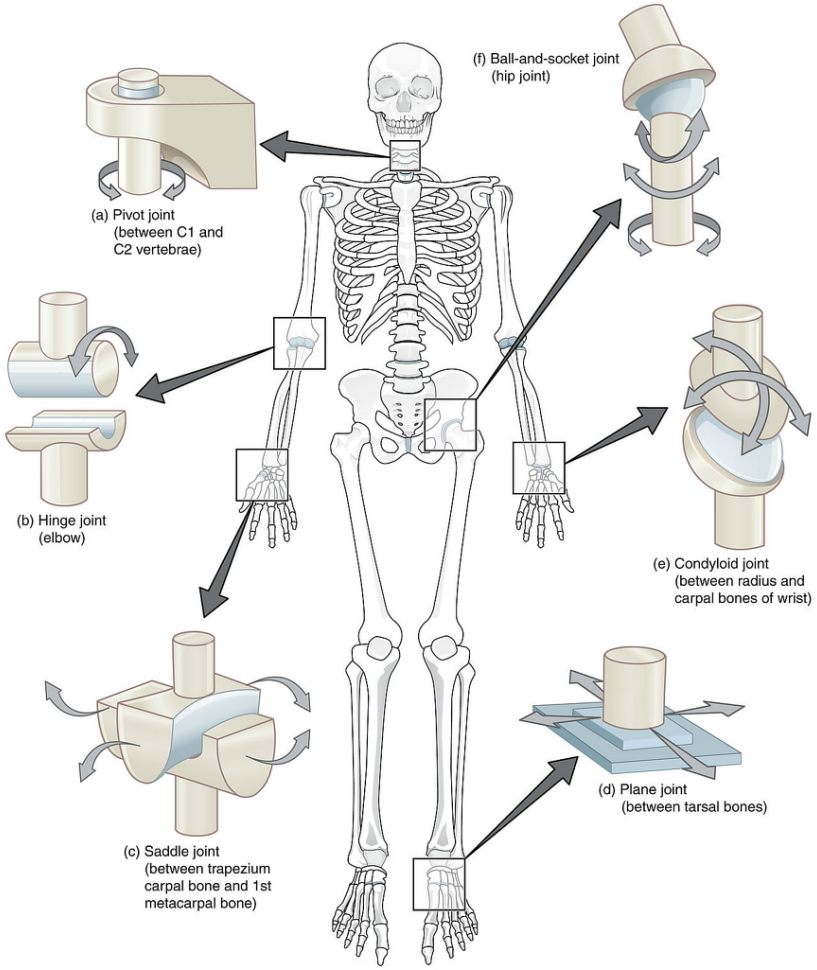


FIGURE *4.2 – Les différents types d'articulations synoviales et leur emplacement dans le corps. [11]*

4.3 Approches possible pour la représentation des utilisateurs

Le point de départ de notre réflexion a été de déterminer les différentes manières dont un concepteur de media 3D pourrait utiliser le suivi utilisateur pour permettre aux utilisateurs se percevoir mutuellement. Nous avons ainsi distingué deux approches possibles :

— La première approche consiste à fournir des avatars de même fidélité à tous les utilisateurs. Dans ce cas, si un support fournit le mouvement de jointures non présentes pour le type d'avatar choisi, ces données sont simplement ignorée. À l'inverse, si les données de jointures utilisées par l'avatar ne sont pas présente, le média 3D devra adapter automatiquement l'animation de l'avatar en fonction des données présentes. La principale limite de cette approche est que le réalisme de l'animation des différents avatars pourra, dans un même média, varier de manière visible selon les supports. Il nous faudra donc dans nos travaux futurs évaluer l'impact sur la perception des autres utilisateurs de ces variations. De même une étude utilisateur approfondie devra être menée de sorte à déterminer les outils nécessaires aux concepteurs de médias 3D pour définir l'adaptation de l'animation aux données reçues.

— Une seconde approche pourrait consister à adapter la fidélité de l'avatar à la qualité des données reçues. Ainsi, deux utilisateurs de supports différents pourraient collaborer avec des avatars asymétriques. On pourrait par exemple imaginer un avatar représentant tout le corps d'un utilisateur co-exister avec un avatar composé uniquement d'une tête et deux mains. Dans ce cas, l'adaptation de l'animation devrait être plus simple pour le concepteur (moins d'articulations dont la position doit être interpolée), mais celui-ci devra concevoir d'avantage d'avatars. En effet, si plusieurs modes de représentation sont proposés en fonction des données, il est nécessaire de décliner chaque personnage dans l'ensemble de ces modes. Cette deuxième approche nous semble intéressante dans des cas d'usages ne nécessitant pas un haut niveau de réalisme, et proposant une grande diversité d'interaction. Le niveau de fidélité de son propre avatar, ainsi que celui des avatars des autres utilisateurs, pourraient aider les utilisateurs a percevoir/présupposer de leur capacité d'interaction avec l'environnement et les guider dans une éventuelle répartition des tâches. Aucune étude n'adressant à notre connaissance cette hypothèse, celle-ci constitue une piste intéressante pour la poursuite de nos travaux sur la représentation des utilisateurs lors de l'interaction avec un média 3D.

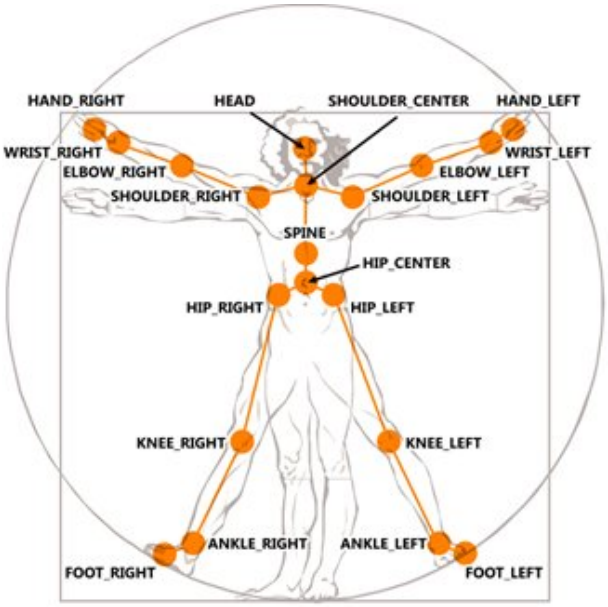


FIGURE *4.3 – Articulations du squelettes humain fourni par le capteur Kinect V2.*

4.4 Exemple d'implémentations dans des navigateurs UMI3D

L'objectif de cette partie est d'illustrer l'usage de notre méthode de suivi de l'utilisateur pour la conception d'un système d'avatars s'adaptant à la qualité des données fournis par les différents supports. De sorte à mieux distinguer les adaptations, nous avons choisis d'adapter l'apparence de l'avatar aux données plutôt que d'adapter son animation. Pour tester le système conçu, nous avons implémenté la fonctionnalité de suivi utilisateur dans les trois navigateurs UMI3D déjà présenté dans le Chapitre 3 (3.3 Étude de cas : co-conception d'un jeu de construction collaborative), à savoir : un ordinateur de bureau proposant une interaction classique avec souris et clavier, un Oculus Rift et ses deux contrôleurs Touch à 6 degrés de liberté, ainsi qu'un Oculus Rift proposant une interaction directe avec les mains grâce à un capteur Leap Motion. Pour l'implémentation du suivi utilisateur pour les différents supports, nous avons sélectionné nos articulations parmi les jointures d'un squelette simplifié semblable à celui du capteur Kinect illustré Figure 4.3.

Ordinateur de bureau Pour l'ordinateur, nous avons associé l'écran à la tête de l'avatar, représentée par la caméra dans le moteur de jeu Unity3D et la souris à une main de l'utilisateur. La souris ne représente qu'une seule main car elle ne permet de contrôler qu'un seul pointeur. Le clavier n'est pas représenté car nous considérons qu'il ne permet pas de transmettre un mouvement de l'utilisateur.

Nous avons donc construit un premier squelette constitué d'une tête, d'un tronc, d'un bras diminué pour la main non dominante, et d'un bras complet pour la main dominante

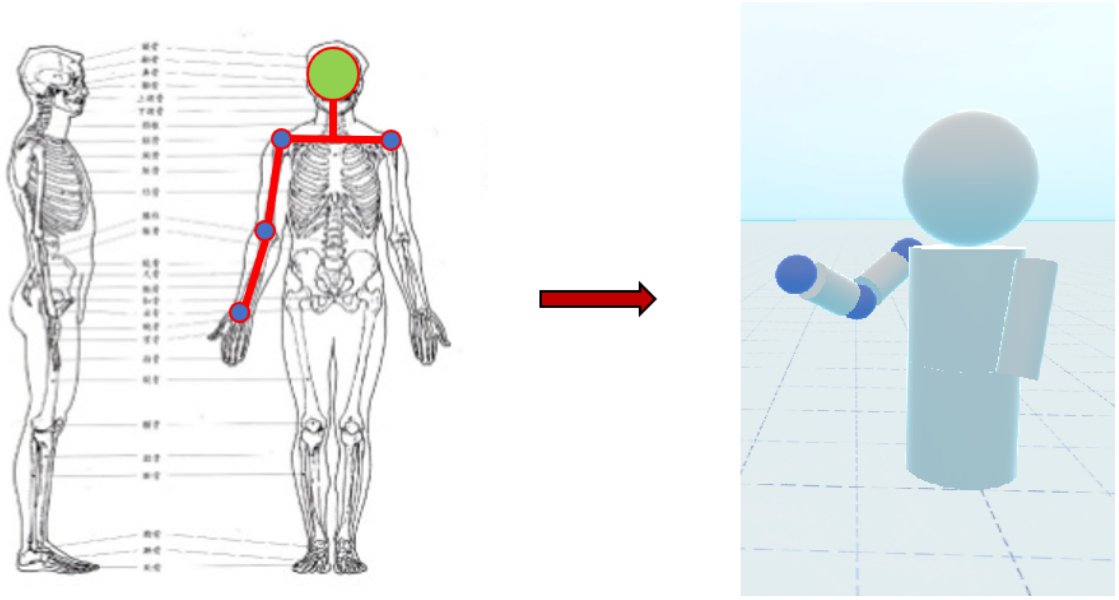


FIGURE *4.4 – Squelette pour un ordinateur de bureau.*

(c'est-à-dire qu'il est lui-même composé d'une épaule, d'un bras, d'un coude, d'un avant-bras et d'un poing). Ce squelette est visible dans la Figure 4.4.

Par la suite, nous avons établi des liens physiques entre les parties du corps mises en jeu. Tout d'abord, il nous semblait approprié que le tête et le tronc soient fixes l'un par rapport à l'autre, de même que le tronc et le bras non dominant ce dernier n'étant pas animé. L'ensemble reste alors orienté de manière cohérente avec la caméra lors de des déplacements de l'utilisateur. Ces déplacements seront d'ailleurs issus des déplacements de la caméra du moteur de jeu lors de l'utilisation de l'application.

Nous avons ensuite définis des contraintes physiques pour bras dominant qui est lui mobile. Notre première idée a été d'animer la main avec les mouvements du curseur de la souris. Pour rappel, le bras dominant se compose d'une épaule, d'un bras, d'un coude, d'un avant-bras et d'une main. D'abord, nous avons rendu l'épaule fixe par rapport au tronc. Ensuite, nous avons positionné le bras à la suite cette épaule. Le seul mouvement qui lui est autorisé est la rotation centrée sur l'épaule d'axe parallèle à l'axe de révolution du tronc. Dans le but de maintenir la main dans le champ de vision de l'utilisateur, sa rotation est limitée entre 0 et 115 degrés vers l'intérieur du corps. Le coude est quant à lui placé dans la continuité du bras et fixe par rapport à lui. L'avant-bras est positionné par rapport au coude de la même façon que le bras par rapport à l'épaule, c'est à dire avec une rotation autour du coude à deux degrés de liberté dans une limite de 70 degrés. Nous estimons que ces premières limites imposées permettent d'avoir un premier rendu de mouvement similaire à un corps humain lorsque la main se déplace.

Cet arrangement a toutefois généré plusieurs inconsistances et n'était par ailleurs pas intuitif pour l'utilisateur. Nous avons donc définis une position de repos qui est activée lorsque le curseur de la souris indique un point n'étant pas à la portée du bras de

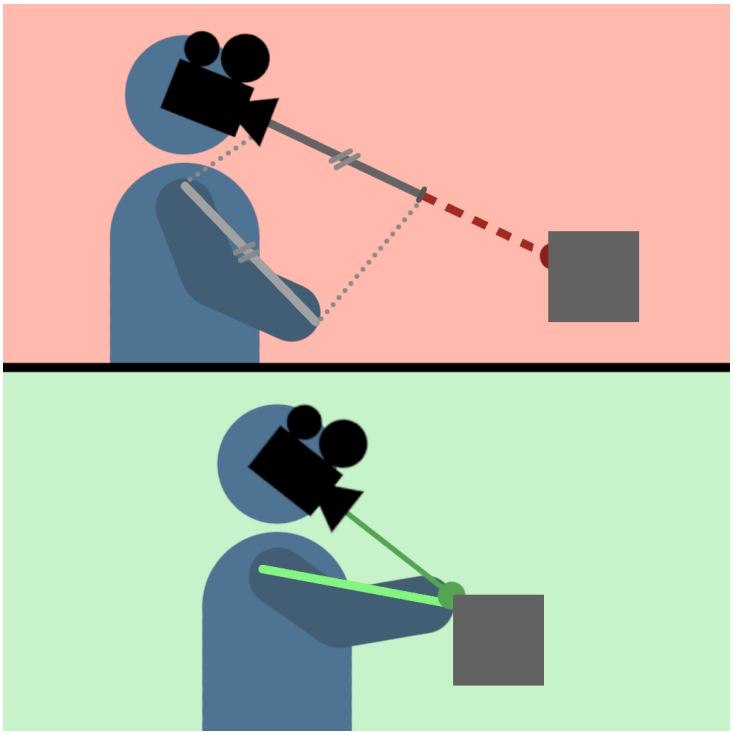


FIGURE 4.5 – *Position de repos de la main pour l'ordinateur de bureau.*

l'utilisateur (Figure 4.5).

Oculus Rift et contrôleurs 6-D Pour l'Oculus Rift équipé de contrôleurs Touch, la tête étant équipée du casque, il est naturel de l'associer à celle du squelette. De plus, les deux mains étant chacune dotée d'un contrôleur, il est aussi possible de le représenter. Nous obtenons alors un squelette composé d'une tête, d'un tronc et de deux bras complets. On peut voir le résultat à la Figure 4.6.

Concernant la physique de ce squelette, elle est relativement proche de celle du squelette de l'ordinateur. Nous avons réutilisé la même disposition et les mêmes contraintes physiques pour les deux bras de ce squelette, en y ajoutant pour les sous éléments bras une rotation possible autour de chaque épaule selon l'axe des deux épaules.

Les mains du squelette sont positionnées en temps réel sur les contrôleurs de l'utilisateur. Nous avons observé de premiers résultats encourageant sur le suivi des bras du squelette avec cette méthode. Cependant, des incohérences se créent dans l'anatomie du squelette lorsqu'un utilisateur pose ses contrôleurs ou lorsqu'il a une taille légèrement différente de celle du squelette, pouvant générer une déviance de certaines parties pour le reste de la simulation.

Oculus Rift et Leap Motion Enfin, pour l'Oculus avec Leap Motion, la tête est comme précédemment équipée du casque. On la représentera de la même façon pour le squelette de cet équipement. De plus, le Leap Motion est un dispositif similaire à une caméra permettant le suivi des mains d'une ou de plusieurs personnes. Il est donc possible

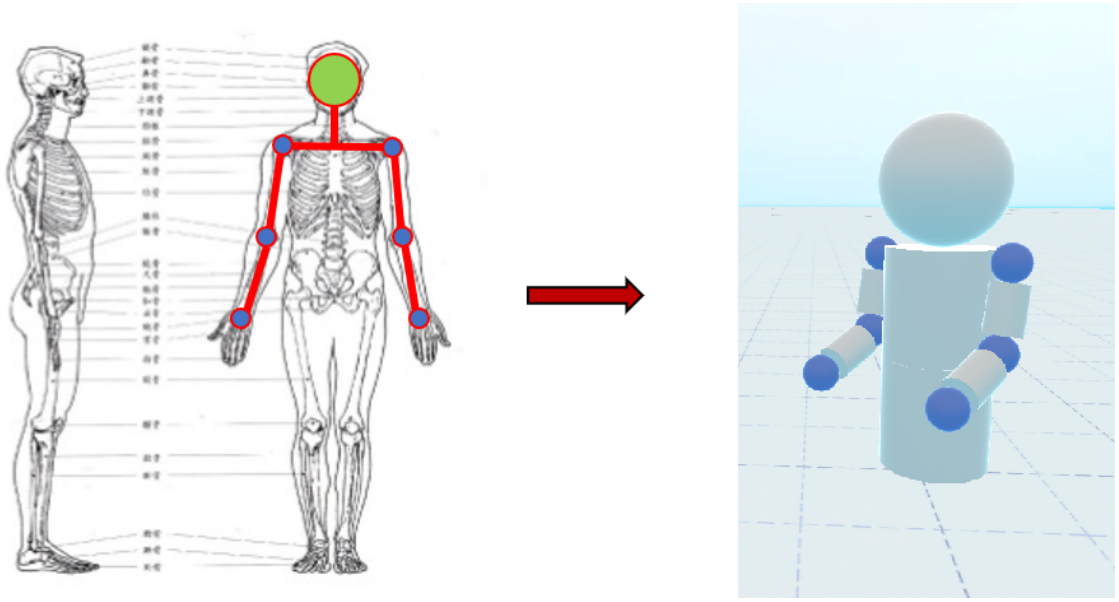


FIGURE *4.6 – Squelette pour un Oculus Rift & contrôleur 6-D.*

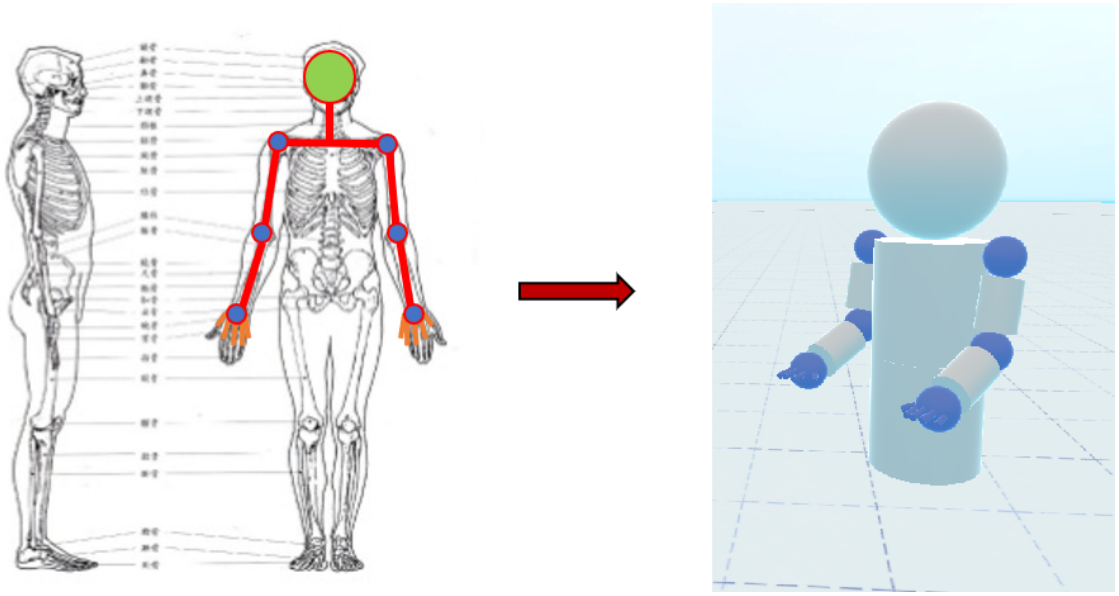


FIGURE *4.7 – Squelette pour un Oculus Rift & capteur Leap Motion.*

d'associer les mains de l'utilisateur aux mains du squelette. Il en résulte alors un squelette proche du précédent. On aura ainsi une tête, un tronc et deux bras complets. Cependant, puisque le Leap Motion est également capable de détecter et positionner les doigts et articulations de la main (ce qui permet, entre autres, de savoir si un doigt est tendu ou plié), nous ajoutons cinq doigts à chaque main. (Figure 4.7).

La physique de ce squelette est exactement la même que pour le dispositif précédent. Cependant, les mains du squelette sont placées à l'endroit où le Leap Motion estime que les mains de l'utilisateur se situent. C'est pourquoi certaines anomalies n'apparaissent pas sur ce dispositif. Par ailleurs, nous avons mis au point un système permettant de changer l'apparence de chacun des doigts du squelette, en fonction de leur extension (plié ou tendu).

Choix d'un avatar par le média 3D L'interpolation des articulations d'un avatar à partir du mouvement des périphériques étant une thématique largement étudiée, nous n'y avons pas consacré d'avantage d'efforts à ce stade. Les animations proposées étant en effet suffisante pour valider le fonctionnement de notre système de suivi utilisateur. Pour illustrer l'adaptation automatique de l'avatar associé à un utilisateur en fonction de son support, nous avons implémenté le processus suivant :

- Lors de la connexion d'un nouvel utilisateur au média 3D, celui-ci attend la première transmission par le support de la pose du corps de l'utilisateur pour choisir un avatar.
- Lors de la première transmission de la position utilisateur, le média génère l'avatar visible Figure 4.7 si les données contiennent au moins les positions de la tête, des bras, des mains et des doigts de l'utilisateur. L'avatar visible Figure 4.6 est généré si une partie des données sont absentes mais la tête, deux bras et une main sont présents. Enfin si l'avatar ne contient que les données de la tête et d'une main, c'est l'avatar visible Figure 4.4 qui est généré. Si aucun des avatars ne peut être associé aux données, l'utilisateur n'est pas représenté.
- Lors de la mise à jour par le support de la pose de l'utilisateur, l'avatar animé en temps réel (30 à 90 Hz) et synchronisé pour chaque participant grâce à la synchronisation de scène de UMI3D.

4.5 Analyse des latences

Si les mécanismes de synchronisation de scène de UMI3D permettent bien d'animer des avatars, et ainsi de représenter les utilisateurs, ces animations se font avec une certaine latence réseau. Le mouvement de l'utilisateur est en effet transmis au média 3D par le réseau, le média détermine ensuite la nouvelle position des avatars avant de la redistribuée par le réseau aux différents utilisateurs. Si une latence faible n'est pas un problème pour la

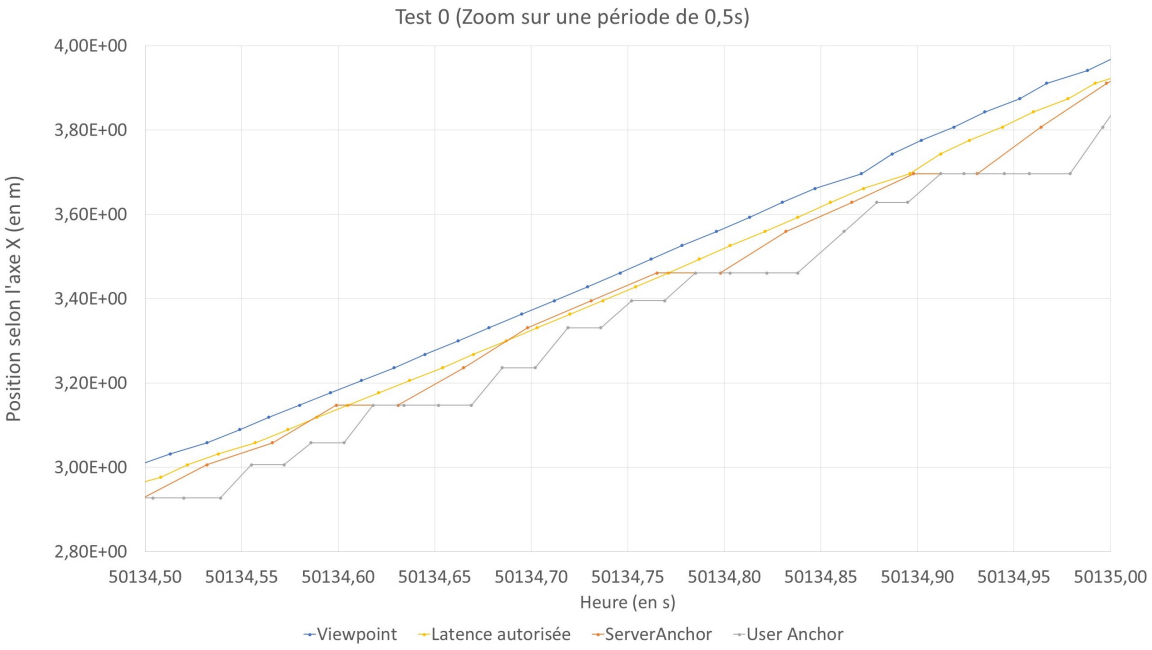


FIGURE *4.8 – lorem ipsum*

visualisation des autres utilisateurs, celle-ci a un impact très négatif lorsqu'il s'agit de son propre avatar entrainant un accroissement du mal du simulateur (Motion Sickness). Sur ce point la littérature s'accorde sur une latence maximale tolérable d'environ 20 millisecondes [69, 34, 92].

Pour déterminer si l'animation de l'avatar d'un utilisateur peut lui être transmise par le mécanisme de synchronisation habituel de UMI3D, nous avons réalisé un ensemble des tests de performances. L'un de ces tests, illustré Figure 4.8 a consisté à exécuter sur une même machine un média 3D et un navigateur UMI3D. L'exécution sur la même machine du média (serveur) et du navigateur (client) permet de limiter au maximum les latences réseau. Durant un déplacement de l'utilisateur, nous avons relevé la position de la caméra du navigateur (viewpoint), la position de l'articulation correspondante de l'avatar pour le média 3D (ServerAnchor), ainsi que pour le navigateur (UserAnchor). En affichant la position de la caméra retardée de la latence maximale autorisée, on constate que même dans des conditions réseau idéales, le positionnement de son propre avatar admet une latence largement supérieure à la limite fixée [1]. Nous en concluons que les mécanismes de synchronisation de UMI3D ne sont pas adaptés à la visualisation de son propre avatar.

1. En réalité, la position de la caméra peut déjà admettre si la navigation utilise une capture de position réelle de l'utilisateur.

4.6 Modèle de suivi utilisateur proposé

Au vu des problématiques de latence soulevées, nous proposons de n'utiliser les mécanismes de synchronisation de scène de UMI3D que pour la visualisation des avatars des autres utilisateurs. Nous proposons donc que chaque navigateur anime lui-même son propre avatar. Le media 3D transmet pour cela l'avatar au navigateur avec une description de la manière dont cet avatar doit être relié aux articulations du corps de l'utilisateur. Le navigateur relie alors directement les jointures de l'avatar aux articulations de l'utilisateur permettant son animation. L'utilisateur peut alors visualiser son propre avatar sans latences réseaux. Le mécanisme de suivi de l'utilisateur décris précédemment ainsi que la synchronisation de scène de UMI3D sont quand à eux utilisés pour partager l'avatar aux autres utilisateurs. Il en résulte le fonctionnement résumé par la Figure 4.9. Dans le but de réduire les latences dans le partage d'un avatar aux autres utilisateurs, nous étudierons à l'avenir la possibilité d'un partage pair à pair des mouvements des utilisateurs, par exemple en utilisant le protocole WebRTC.

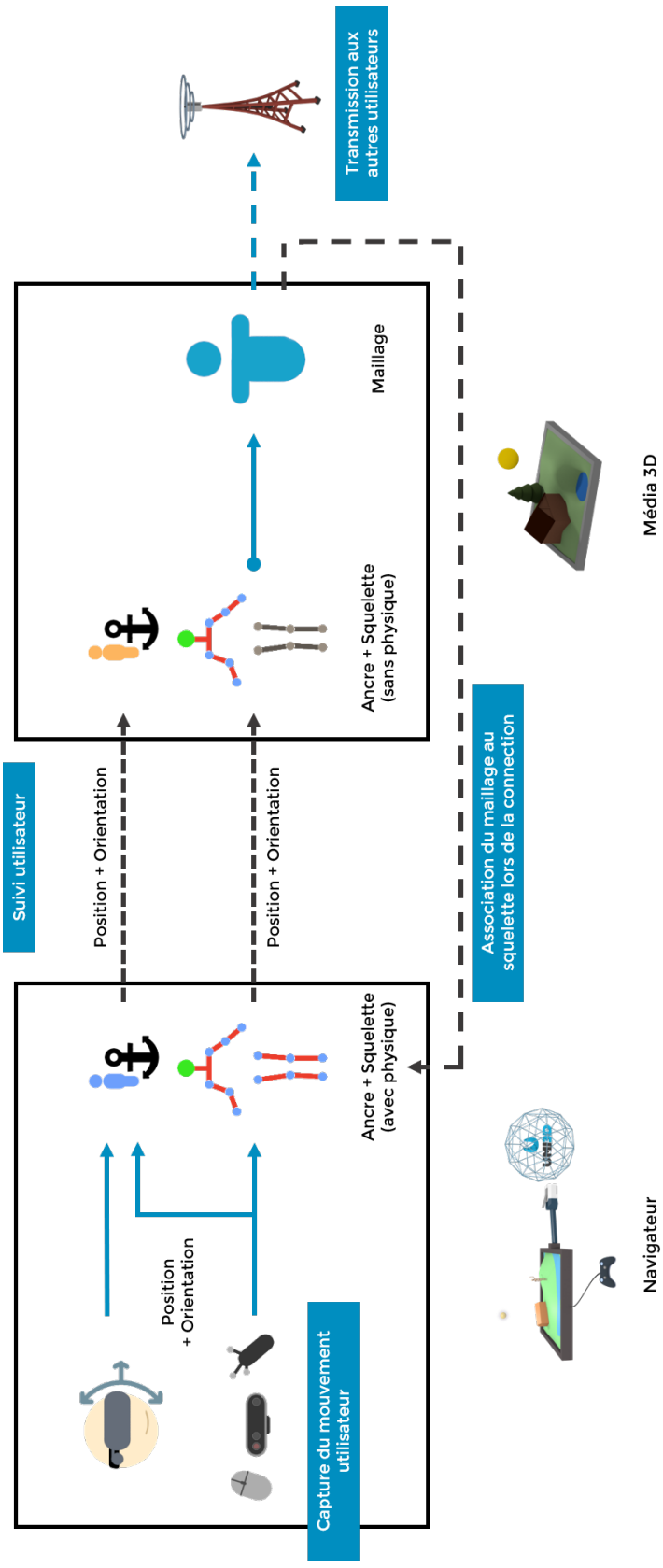


FIGURE *4.9 – Visualisation d'avatars avec UMI3D*

Chapitre 5

Travaux Préliminaires pour l'Évaluation des Performances Utilisateur avec UMI3D

Ce chapitre présente des travaux préliminaires, réalisés en collaboration avec Antoine Ladrech [20]. L'objectif de ces travaux est la proposition d'outils et de méthodes pour l'évaluation de l'expérience utilisateur lors de l'interaction avec média 3D utilisant le protocole UMI3D. Le premier objet de ces travaux est de fournir aux futurs concepteurs de navigateurs UMI3D un moyen fiable de les calibrer et de les évaluer. Le second consiste à permettre aux concepteurs de médias 3D d'évaluer la manière dont les utilisateurs des différents navigateurs interagissent avec ces médias. La principale difficulté pour mener ces évaluations provient de l'architecture client-serveur de UMI3D. Ce que nous qualifions habituellement de technique d'interaction pour la manipulation est en effet répartis entre deux systèmes distincts : Le media définit une technique de manipulation en fonction d'un contrat de mouvement. Il s'agit d'associer une réaction à la création d'un mouvement par un navigateur UMI3D. Le navigateur implémente quand à lui une technique de génération de mouvement lui permettant de satisfaire un contrat de mouvement. Le concepteur du média ou du navigateur ne maîtrise donc qu'une moitié de l'implémentation des interactions. Il est toutefois nécessaire de collecter des informations provenant des deux systèmes pour analyser la manière dont les utilisateurs interagissent avec un média. Dans ce chapitre nous présentons donc un système de capture de données permettant de réaliser ces analyses en croisant des données provenant du média 3D avec des données provenant du navigateur UMI3D.

5.1 Évaluation des périphériques et techniques d'interaction

L'évaluation des périphériques et techniques d'interaction occupe une place centrale dans la littérature concernant l'interaction 3D. Celle ci est en effet trivialement nécessaire pour mesurer l'apport d'un nouveau périphérique ou d'une nouvelle technique. Il est dans cet optique difficile d'évaluer seul un périphérique ou une technique, et l'on préfère

généralement comparer les performances avec celles obtenues par un autre périphérique ou une autre technique d'interaction. La procédure la plus répandue consiste donc à organiser des sessions de tests durant lesquelles un groupe d'utilisateurs réalise une même séquence de tâches avec plusieurs périphériques ou plusieurs techniques d'interaction.

Les participants doivent idéalement soit couvrir une large diversité de profils utilisateur (age, main dominante, expérience de l'interaction 3D, ...), soit provenir d'une population d'utilisateur ciblée, par exemple des utilisateurs novices comme dans [73]. Les utilisateurs peuvent également être répartis en différents groupes testant les techniques/périphériques dans un ordre différent. Cela permet de limiter les biais dans la comparaison en évitant que tous les utilisateurs aient une meilleure connaissance des tâches à réaliser sur un support. Si le nombre d'utilisateurs est suffisamment important, il est possible de limiter chaque groupe à un(e) seul(e) périphérique(technique) et ainsi éviter qu'il ait une progression des utilisateurs durant l'expérimentation. Dans le cas de l'évaluation d'un support (et de ses périphériques), les tâches évaluées doivent être représentatives de l'ensemble des interactions possibles en environnement 3D (sélection, manipulation, navigation et contrôle d'application). Ces tâches doivent par ailleurs être de difficulté variable de sorte à évaluer les supports selon plusieurs indicateurs comme le temps d'exécution ou la précision.

Dans le cadre de nos travaux préliminaires, nous nous sommes principalement intéressé aux tâches de manipulation d'objets 3D. Dans ce cadre les principaux aspects étudiées sont la sélection de l'objet à manipuler [72, 66], le positionnement de l'objet par translation et rotation vers un positionnement cible [10, 43, 89, 87], ainsi que l'étude de la trajectoire de l'objet manipulé [43, 88, 89, 87]. Des métriques, calculées durant l'exécution de la tâche ou à postériori, sont alors analysées et comparées en fonction du support ou de la technique d'interaction. Si certaines métriques sont universelles (temps de complétion, succès/échec), d'autres dépendent de la tâche. Certaines ne sont par exemple applicables qu'aux tâches de positionnement comme : la distance finale entre l'objet et la position cible, ou la distance totale parcourue par l'objet. Comme chaque métrique permet de quantifier différents aspects de l'exécution d'une tâche, il est possible d'orienter l'évaluation du support selon les métriques choisies. Par exemple, si un périphérique a été conçu à des fins de précision, la distance finale entre l'objet et la position cible aura plus d'importance dans l'analyse que le temps de complétion de la tâche.

Pour appliquer ces méthodes à l'évaluation des navigateurs UMI3D ou de l'interaction de ces navigateurs avec un media 3D donné, il est dans un premier temps nécessaire de calibrer les navigateurs UMI3D, c'est à dire de garantir qu'ils rempliront de manière similaire les contrats d'interaction proposés par un média 3D. En effet, si la réponse attendue à un contrat d'interaction donné n'est pas normée, il sera impossible pour les concepteurs de média d'une part, et pour les concepteurs de navigateurs d'autres part d'obtenir des médias 3D manipulables avec tout navigateur ou inversement des navigateurs capables de manipuler tout média 3D. Il devient alors possible de comparer des navigateurs en

fixant le choix d'un média 3D, ou de même d'évaluer une technique d'interaction en proposant, depuis le média 3D, plusieurs contrats d'interaction différents pour la réalisation d'une même tâche. Le deuxième élément nécessaire est l'implémentation d'un système de capture de données permettant de capturer et croiser des données sur l'évolution de la tâche provenant du média 3D, avec des données sur l'activité de l'utilisateur provenant du navigateur UMI3D.

5.2 Calibration d'un navigateur UMI3D

Pour affirmer qu'un navigateur UMI3D est bien calibré, il faut vérifier qu'il remplit les contrats d'interaction proposés par un média 3D d'une manière similaire aux autres navigateurs. Il ne s'agit ici pas de d' affirmer que ce navigateur permet de réaliser ces contrats avec les mêmes performances, mais uniquement de vérifier si les navigateurs sont bien utilisables pour les mêmes tâches. Le principal point de calibration concerne l'interprétation par le navigateur de la brique d'interaction 'Manipulation'. Il s'agit en effet de la seule brique d'interaction pour laquelle les paramètres de la réponse sont inconnus lors de l'établissement du contrat d'interaction. Cette brique donnant en effet aux supports la possibilité d'envoyer en temps réel (30 à 90 Hz) un mouvement (définit par des degrés de liberté attendus) au média. Il s'agit donc de vérifier que les navigateurs UMI3D produisent des amplitudes de mouvement similaires lors de l'activation de cette brique d'interaction. Si ce n'est pas le cas, les interactions mises à disposition par le média, grâce à cette brique, ne seront utilisables que par les supports générant les mêmes amplitudes de mouvement que les supports dont disposait le concepteur du média. Dans le but de calibrer les navigateurs UMI3D, nous proposons donc un média 3D contenant un ensemble de tâches de référence qui doivent, si possible, être réalisables par tout navigateur UMI3D.

Média de référence pour la calibration. Comme expliqué précédemment, nous avons conçu un média 3D dédié à la calibration des navigateurs UMI3D. Ce média permet à l'utilisateur de réaliser tour à tour des tâches de translation (Figure 5.1) et de rotation. La tâche consiste simplement à positionner un cube dans un cube cible de taille légèrement supérieure (marge d'erreur). Dans le cas de la rotation, le cube est déjà à la même position que la cible, et seule l'orientation reste à manipuler. Chaque tâche doit être exécutée dans un délai de 30 secondes. Enfin, le média 3D permet à l'utilisateur de recommencer indéfiniment chaque tâche de sorte à permettre une éventuelle calibration du navigateur entre deux tests.

Pour permettre une calibration la plus simple possible, les briques d'interaction 'Manipulation' permettant l'exécution des différentes tâches fonctionnent à la manière d'un passe plat : le mouvement transmis par le navigateur est directement appliqué à l'objet sans aucune transformation (pas de coefficient de sensibilité du côté du média 3D, pas de

lissage/filtrage du mouvement transmis, ...). Par ailleurs, lors de nos sessions de tests, nous exécutons si possible (ce n'est pas le cas sur certains support mobiles) nos navigateurs depuis Unity de sorte à pouvoir les calibrer pendant l'exécution des tests en fonction des retours des utilisateurs. Il en résulte des sessions de calibration à l'exécution relativement similaires au processus de conception à l'exécution des média 3D décris dans le Chapitre 3.

Exemple : Calibration de la translation 3-D d'un navigateur UMI3D. Nous avons par exemple utilisé cet environnement pour calibrer la génération d'un mouvement de translation à trois degrés de liberté avec notre navigateur Oculus Rift et contrôleurs 6-D. Ce navigateur implémente en effet une technique permettant de générer en continu une translation lorsque le contrôleur 6-D s'éloigne d'une position neutre (voir Figure 5.1). Lorsqu'une brique d'interaction 'Manipulation' propose de réaliser un mouvement de translation à trois degrés de liberté, l'actionnement continu d'un bouton du contrôleur déclenche la génération d'un mouvement (Il s'agit du bouton avec l'étiquette "Move" sur la Figure 5.1). Lorsque le mouvement commence, un repère virtuel est affiché (par le navigateur) à la position de départ du contrôleur pour indiquer la position d'équilibre (3 sur la Figure 5.1). Le mouvement est généré en continu dans la direction 'position neutre' vers 'centre du contrôleur' avec une amplitude proportionnelle à la distance entre ces deux points (2 et 3 sur la Figure 5.1). Une amplitude maximum est définie pour limiter l'impact des erreurs de manipulation du contrôleur. Dans ce cas, la calibration a consisté à définir le volume de la zone neutre (c'est à dire la distance du point neutre à laquelle un mouvement commence à être généré), la distance maximale (c'est à dire la distance du point neutre à laquelle l'amplitude maximale est atteinte), ainsi que la sensibilité du système (c'est à dire la vitesse de génération de la translation en m/s lorsque l'amplitude est maximale).

FIGURE *5.1 – Exécution, avec le navigateur Oculus Rift et contrôleurs 6-D, d'une tâche de référence pour la translation à 3 degrés de liberté.*

5.3 Évaluation des navigateurs UMI3D

Nous avons donc présenté, dans la première partie de ce chapitre, les besoins en terme de calibration des navigateurs UMI3D. Si la procédure de calibration proposée permet de s'assurer sans efforts importants qu'un navigateur UMI3D répondra bien de la manière attendue aux contrats d'interaction établis par les médias 3D, elle ne fournit pas de moyen objectif pour mesurer les performances utilisateur. Pour permettre le calcul d'indicateurs de performance objectifs, nous avons utilisé une plateforme d'analyse de données massives (Big Data) pour stocker, puis croiser et enfin analyser des données capturées simultanément, à chaque image (60 à 90 Hz), par un média 3D (position/orientation d'un objet manipulé, début/fin/succès/échec d'une tâche) et par un navigateur (mouvements des périphériques, boutons actionnés). La capture de ces données brutes, c'est à dire non transformées en indicateurs, par une solution Big Data est intéressante car elle permet le calcul d'indicateurs à posteriori. Les données expérimentales peuvent ainsi être stockées à long terme et être réutilisées plus tard lors de la conception d'un nouveau navigateur. Le premier avantage est de ne pas avoir à reproduire les expérimentations pour un navigateur dont les données sont disponibles. Le second avantage est de permettre de comparer le nouveau navigateur à tous les navigateurs existants (à condition qu'ils aient été évalués avec ce système). Enfin, les métriques intéressantes n'étant pas forcément les mêmes selon les avantages recherchés pour un nouveau navigateur, le stockage de données brutes permet le calcul à posteriori de nouveaux indicateurs pour une expérimentation passée.

5.3.1 Plateforme d'analyse de données

Le fonctionnement d'une plateforme d'analyse de données massives n'étant pas le sujet de nos travaux, nous ne nous attarderons pas longuement sur le fonctionnement de ces plateformes. Nous en expliquons toutefois quelques principes en base qui sont nécessaires à la compréhension des travaux.

Stockage des données. La plateforme que nous avons utilisée pour nos travaux s'appuie sur une technologie de stockage appelée **Data Lake Storage** qui est fournie par Microsoft Azure (Cloud de Microsoft). Elle permet le stockage de grand volume de données (jusqu'à plusieurs Petabytes), brutes ou consolidées, dans un système de fichiers sécurisé [70]. Ce système est basé sur le framework **Hadoop** permettant la conception de système de données distribuées en sous-fichiers [14, 77]. Le stockage en une multitude de fichiers permet notamment la parallélisation massive du calcul de métriques et/ou d'analyses statistiques. Une même requête pouvant en effet être exécutée en parallèle sur plusieurs fichiers avant une fusion des résultats. Pour ce faire les données analysées doivent avoir la même structure. Pour simplifier l'écriture et l'exécution de requêtes, il est possible

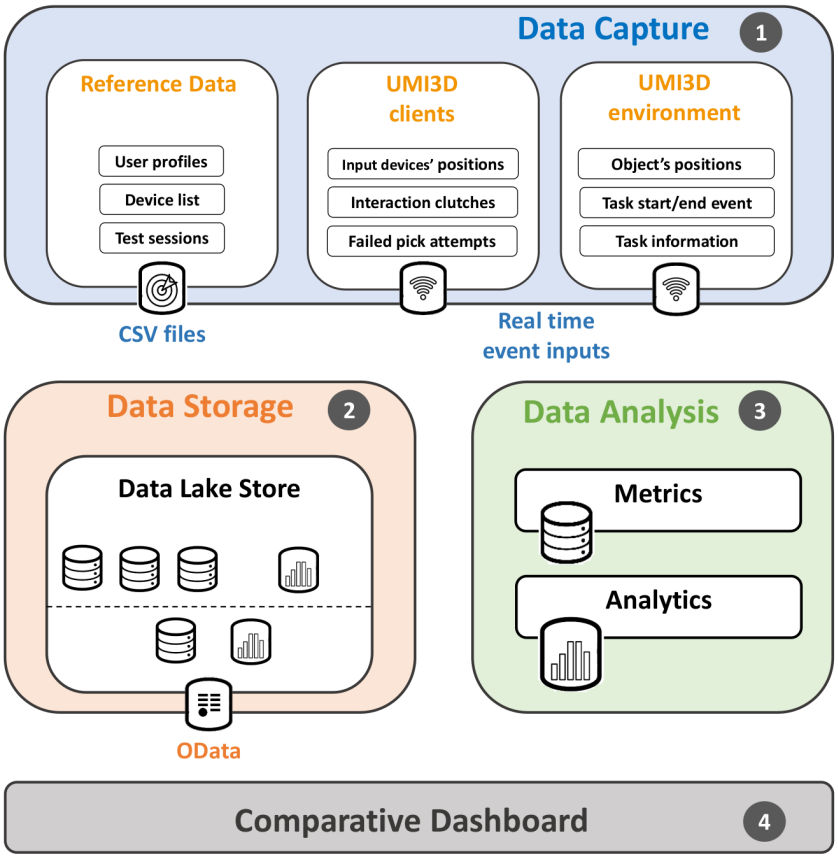


FIGURE 5.2 – *Utilisation d'une plateforme Big Data pour l'évaluation des navigateurs UMI3D.*

de regrouper les données de même structure dans un composant appelé **Data Set**. Ce composant peut ensuite être interrogé à la manière d'une table dans une base de données.

Utilisation avec UMI3D. La Figure 5.2 nous permet d'illustrer la manière dont nous utilisons la plateforme d'analyse de données pour la conception de notre outil d'évaluation des navigateurs UMI3D :

— Les données expérimentales brutes sont stockées massivement dans un **Data Lake** (2 sur la Figure 5.2). L'idée étant de stocker à long terme les données expérimentales de sorte à pouvoir les ré-exploiter à posteriori dans de nouvelles études comparatives.

— Des points d'entrées appelés **Real Time Events** (1 sur la Figure 5.2) permettent à la plateforme de recevoir des flux de données continus (des dizaines de messages chaque secondes) et de les trier en différents **Data Sets**. Une application, comme nos médias 3D et nos navigateurs, peut donc envoyer des données à tout moment en utilisant une clé de sécurité qui lui est dédiée.

— Une fois les données brutes stockées, la plateforme permet l'écriture et l'exécution de **requêtes U-SQL** (3 sur la Figure 5.2) qui est un langage de requête (similaire

à SQL) dédié à l'analyse de données sur cette plateforme [84]. Ce langage permet également l'exécution de script R [4], un langage largement répandu pour l'analyse statistique [41]. Ces requêtes peuvent être exécutées pour toute donnée stockée par la plateforme, et il est également possible de stocker le résultat des analyses dans de nouveaux **Data Sets**.

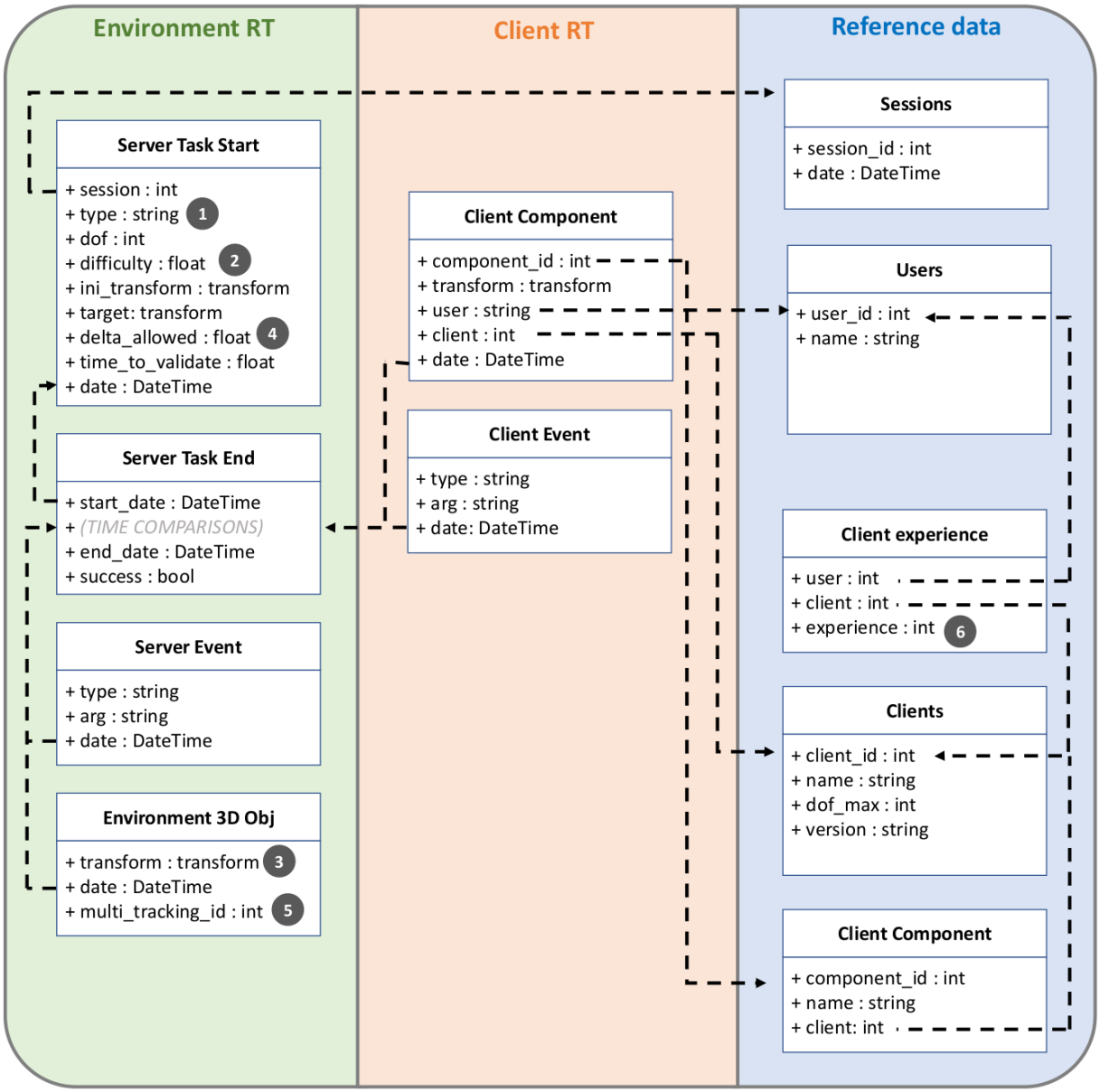
— Pour finir, il est possible d'afficher les données et résultats d'analyses stockés dans la plateforme grâce au protocole de requête OData [22]. (4 sur la Figure 5.2)

5.3.2 Média de référence pour l'évaluation des navigateurs UMI3D

Pour cette étude préliminaire, nous nous sommes concentrés sur la réalisation de tâches de sélection, translation et rotation par les navigateurs UMI3D. Nous avons pour ce faire réutilisé le média utilisé pour la calibration des navigateurs et l'avons complété en ajoutant une tâches de sélection de difficulté variable (la taille de l'objet à sélectionner est plus ou moins grande). Nous obtenons par ainsi cinq types de tâches, chacune étant de difficulté variable : translation à 1, 2 ou 3 degrés de liberté, rotation à 1 degré de liberté, et sélection d'un objet. La difficulté des tâches de translation et rotation est définie par la taille de la cible dans laquelle l'objet doit être positionné. Plus la taille de la cible (toujours plus grande que l'objet manipulé) est proche de la taille de l'objet, plus l'erreur tolérée est faible. Pour éviter les faux positifs dans la détection du succès des tâches de positionnement (par exemple un objet traversant la cible sans s'y arrêter), l'objet manipulé doit être maintenu un certain temps dans la cible avant que la tâche soit considérée comme terminée. De même, un temps limite est attribué pour chaque tâche de sorte à ce que tous les utilisateurs ne passent pas un temps trop conséquent à tenter de réussir les tâches les plus complexes. Nous avons enfin inclus à nos interactions des aides visuelles permettant à l'utilisateur de savoir si l'objet manipulé est bien positionné.

5.3.3 Capture et croisement des données

Nous avons déjà évoqué précédemment la nécessité de combiner des données disponibles sur le média 3D avec des données issues des navigateurs pour évaluer l'interaction avec UMI3D. Pour ce faire le média et le navigateur envoient en temps réel (60 à 90 Hz) des données à la plateforme d'analyse de données. Une liste exhaustive des données que nous avons collectées est présentée par la Figure 5.3. En plus des flux de données du média et du navigateur, cinq fichiers CSV doivent être exportés manuellement sur la plateforme (voir la section 'Reference Data' sur la Figure 5.2 et la Figure 5.3). Ces données de références permettent de relier les données provenant du média et du navigateur à une même session de tests, à un navigateur donné (fiche descriptive du navigateur, du support et de leurs propriétés), et éventuellement à un profil de testeur anonymisé (utile si l'on veut comparer les données en fonction des profils utilisateur : expert/novice par exemple).



1. Different types of tasks that can be performed by the user. (E.g. Translation, Rotation and Selection)
2. Difficulty of the task rated from 1 (very easy) to 10. (supposedly impossible)
3. 'Transform' actually represents the 10 float of a Unity Transform. (i.e. position, scale and rotation)
4. Depends on the type of the task, represents one way of quantifying the largest error allowed to achieve it.
5. If several objects are being tracked in the same task, they need a different multi_tracking_id.
6. The experience of a user on a device is rated with an integer from 1 to 7.

FIGURE 5.3 – *Données capturées pour l'évaluation des navigateurs UMI3D.*

Métrique	**T**	**R**	**S**
Succès/Échec	☑	☑	☑
Temps de complétion	☑	☑	☑
Distance finale de la cible	☑	☐	☐
Distance parcourue	☑	☐	☐
Distance Angulaire de la cible	☐	☑	☐
Distance Angulaire parcourue	☐	☑	☐
Nombre de repositionnement du périphérique d'interaction (Clutches)	☑	☑	☐

TABLE *5.1 – Métriques calculées pour les tâches de Translation (T), Rotation (R) et Sélection (S)*

5.3.4 Indicateurs calculés pour nos navigateurs

Pour cette étude préliminaire, notre but étant de valider le fonctionnement de notre système d'évaluation, nous avons arbitrairement sélectionné, et calculé des métriques sur la plateforme Big Data. Notre choix c'est porté sur des métriques simple et interprétables visuellement sur un graphique. Nous avons par ailleurs essayer de couvrir les différents aspects de la complétion de tâche comme la vitesse d'exécution et la précision. Cela nous a conduit à sélectionner les métriques présentées Table 5.1.

L'ensemble des métriques est calculée pour chaque tâche par une requête U-SQL exécutée à partir des données brutes présente sur la plateforme Big Data. Les résultats sont stockés dans un **Data Set** différent pour chaque type de tâche, dans le même **Data Lake** où sont stockées les données brutes.

5.3.5 Analyses statistiques

Différentes requêtes sont ensuite appliquées à ces **Data Sets** pour conduire des analyses statistiques sur les métriques. Nous calculons dans un premier temps les moyennes, médianes et écarts type. Ces valeurs peuvent facilement être affichées sur des graphes et permettent donc une approche rapide et très visuelle de l'étude. Elles ne sont en revanche pas suffisantes pour tirer des conclusions sur les impacts potentiels des navigateurs sur les résultats. Il est donc nécessaire d'effectuer des tests statistiques plus approfondis. Grâce à la plateforme d'analyse de données, nous pouvons intégrer du code R dans nos requêtes, ce qui est très pratique étant donné que tous les tests statistiques classiques sont déjà implémentés en R. L'étude ANOVA en est un très bon exemple puisqu'elle s'applique bien à nos métriques.

Pour réaliser une étude ANOVA, nous avons développé notre propre script R, et l'avons intégré dans la plateforme. Le script calcule d'abord toutes les combinaisons de paramètres de tâches (type, degrés de liberté, difficulté) et trie les données en fonction de ces combinaisons. Le script effectue ensuite un test exact de Fisher pour vérifier que la distribution de nos résultats est bien Gaussienne. Enfin, si la distribution est validée, nous pouvons

réaliser l'ANOVA et récupérer des valeurs comme la p-value (qui mesure la probabilité d'une corrélation entre un facteur et les résultats) ou la puissance statistique qui quantifie la pertinence de l'étude en fonction de paramètres tels que la taille de l'échantillon analysé.

5.3.6 Visualisation des données

Pour simplifier la lecture de ces analyses nous regroupons nos résultats dans des tableaux de bord interactifs donnant une vue globale et comparative des performances des supports sur les tâches étudiées. Ces tableaux de bord sont accessibles via une application web et ne récupèrent que les résultats des requêtes stockées sur la plateforme. Ceci permet de n'effectuer aucun calcul lors de la visualisation, ce qui la rend beaucoup plus réactive. Les tableaux de bord sont par ailleurs divisés en plusieurs parties qui offrent des points de vue différents des performances des supports.

Dans les tableaux de bord implémentés (Figure 5.4), des histogrammes donnent une vue lisible de la distribution des données de toutes les tâches pour un indicateur donné. Les boîtes à moustaches (Boxplots) permettent une vue plus globale de ces mêmes données tandis que les graphes de moyennes (Average Plots) affichent les moyennes de toutes les métriques pour un type de tâche précis. Nous avons également affiché les résultats notre étude ANOVA sur les tableaux de bord. Elle présente les valeurs clés de l'étude (p-value du test de Fisher, p-value de l'ANOVA et puissance statistique) accompagnées d'une phrase explicative sur la décision à prendre en accord avec les seuils standards pour les p-values (< 0.05).

De plus, les sections sont individuellement interactives et permettent à l'utilisateur de trier les résultats en fonction du type de tâche, de la difficulté et du nombre de degrés de liberté, ainsi que les dispositifs et métriques qu'il souhaite afficher.

5.3.7 Premiers résultats

Dans cette section, nous présentons les résultats d'une session de test réalisée et analysée avec notre système. Le seul objectif de cette session est de démontrer le bon fonctionnement du système, mais en aucun cas de tirer des conclusions quant aux performances des navigateurs UMI3D utilisés pour l'étude.

L'étude a été conduite avec un groupe de douze utilisateurs. Même si nous n'avons pas utilisé leurs niveaux d'expérience sur les différents dispositifs pour l'analyse, nous les avons collectés pour un usage futur. Nous avons utilisé les trois navigateurs UMI3D présentés dans les Chapitres 3 et 4 : un navigateur pour ordinateur de bureau avec une interaction clavier/souris (désigné par CS pour le reste de la section), un pour l'Oculus Rift avec un capteur LeapMotion pour la détection des mains (LM) et un pour l'Oculus Rift avec les manettes Oculus Touch à six degrés de liberté (OT). Chaque utilisateur a réalisé la même

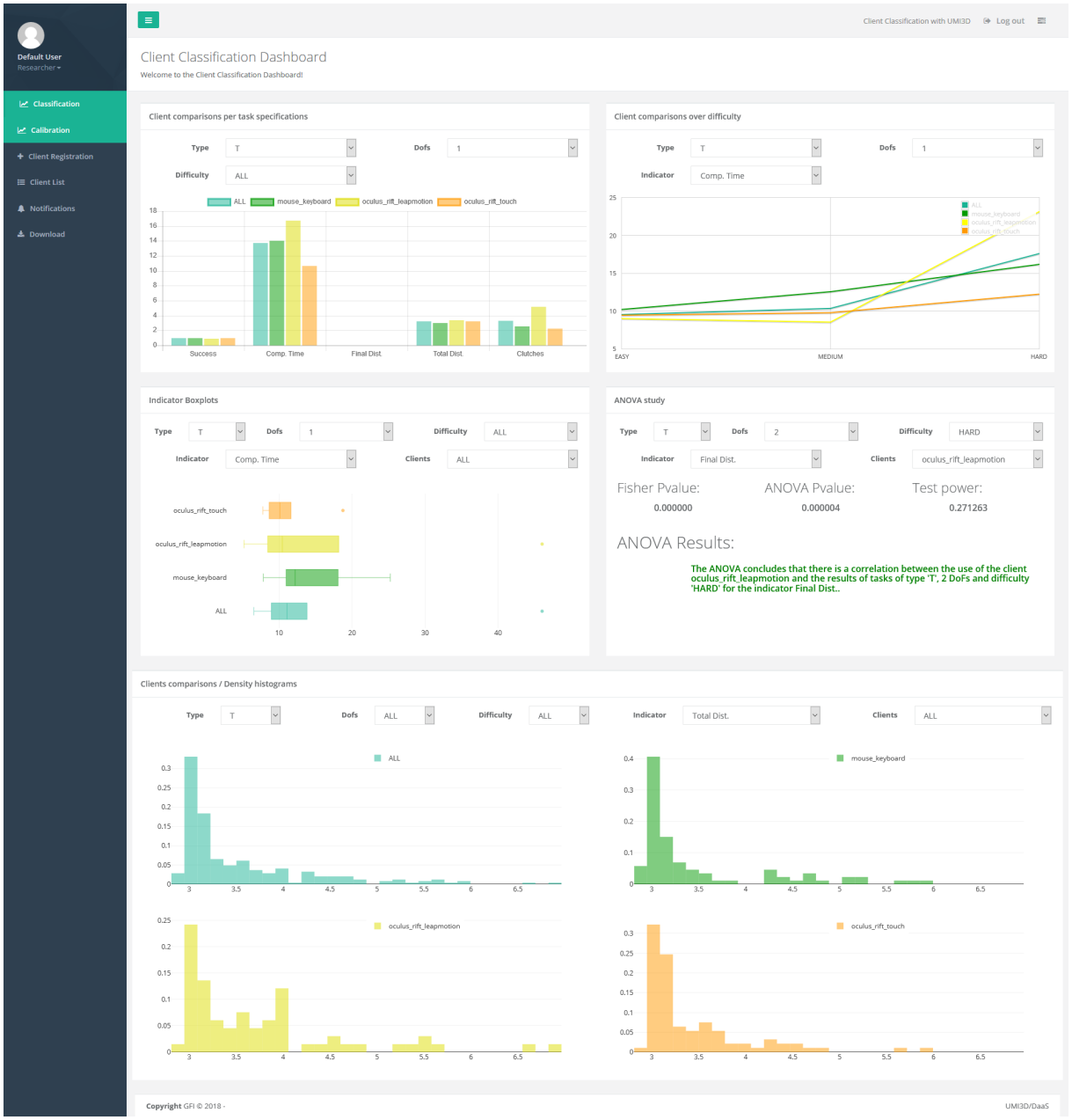


FIGURE 5.4 – *Aperçu des tableaux de bord*

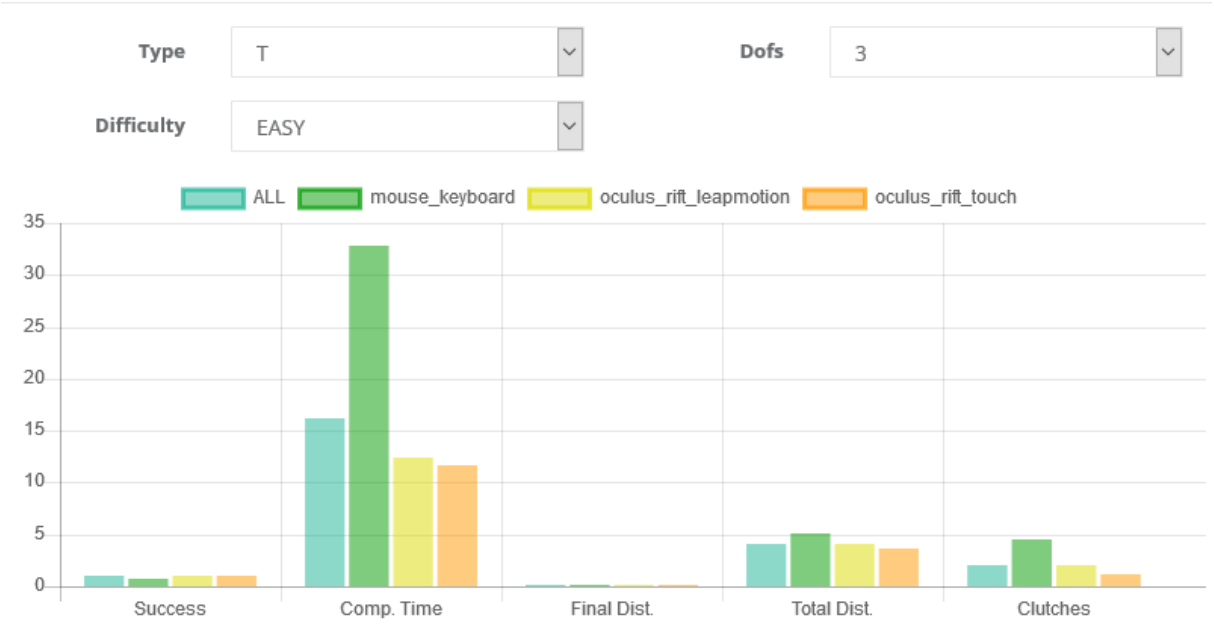


FIGURE 5.5 – *Moyennes des métriques pour la tâche de translation avec trois degrés de liberté et une faible difficulté*

procédure en deux étapes. Premièrement, ils ont été invités à réaliser successivement trois fois une tâche de chaque de type sur chaque navigateur. Cela leur a permis d'apprendre le processus de validation des tâches ainsi que les méthodes d'interaction des différents navigateurs. Nous avons ensuite généré aléatoirement une séquence de vingt-cinq tâches par navigateur. La dimension aléatoire concernait le type, le nombre de degrés de liberté et la difficulté des tâches. L'ordre dans lequel les utilisateurs réalisaient leurs séquences était également aléatoire. Afin d'éliminer l'impact du processus d'apprentissage lors de l'analyse, nous n'avons collecté des données que lors de la seconde phase. Une fois les tâches effectuées, nous avons exécuté toutes les requêtes U-SQL nécessaire au calcul des métriques et des résultats statistiques.

Nous présentons ici des résultats notables, qui nous ont permis de nous assurer de la fiabilité de notre système.

— Comme le graphe Figure 5.5 le montre, la moyenne du temps de complétion, de la distance totale et du nombre de repositionnement du dispositif sont plus élevés pour le navigateur CS pour les tâches de translation avec trois degrés de liberté et une faible difficulté. Ces résultats reflètent possiblement le fait que le navigateur CS ne permet pas l'intégration de trois degrés de liberté à la fois. Les translations 3D sont décomposés par ce navigateur en une translation 2D et une translation 1D. Pour alterner entre les deux translations, l'utilisateur doit sélectionner manuellement l'option souhaitée dans l'interface. Cela cause nécessairement une perte de temps en comparaison aux navigateurs LM et OT qui ont une intégration des mouvements 3D. Cela entraine également une augmentation du nombre de repositionnement

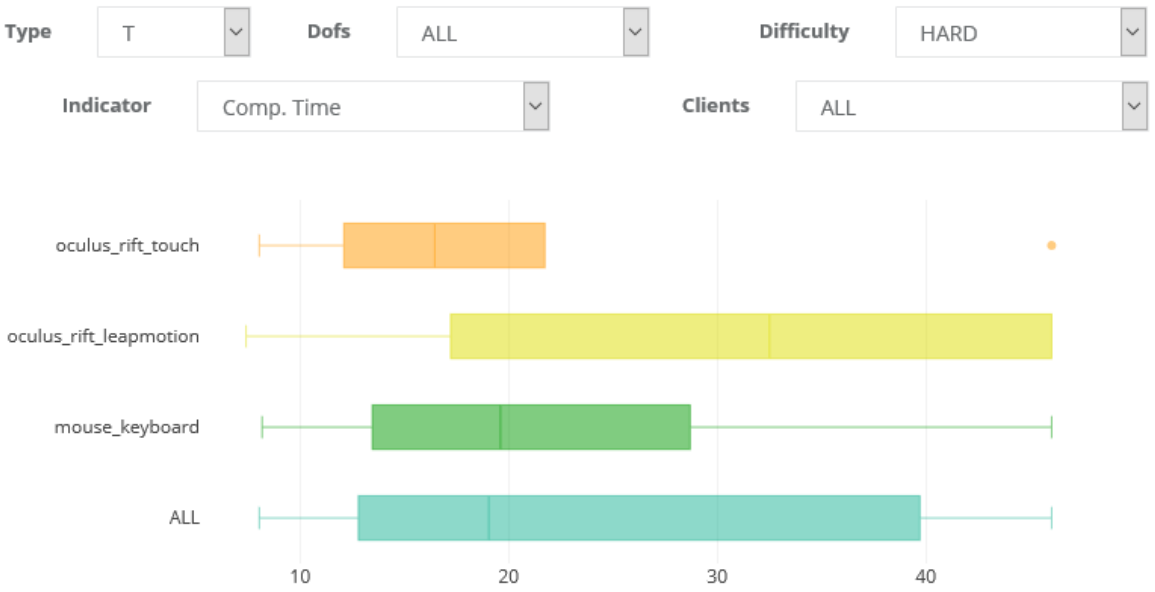


FIGURE *5.6 – Boites à moustaches du temps de complétion pour les tâches de translation avec une haute difficulté*

(l'utilisateur doit lâcher le cube à chaque fois qu'il sélectionne une option. De plus, la décomposition des degrés de liberté empêche les utilisateurs de déplacer le cube droit vers la cible, ce qui explique l'augmentation de la distance parcourue.

— Les boîtes à moustaches de Figure 5.6 représentent la distribution du temps de complétion pour les tâches de translation avec une haute difficulté. Pour le navigateur LM, la valeur médiane est plus élevée que pour les deux autres navigateurs. Le troisième quartile atteint même la valeur maximale qui correspond à la limite de temps donnée pour les tâches. Ce type de tâche semble donc plus difficile à effectuer sur le navigateur LM. Une cause serait peut-être le manque de précision de la technique d'interaction implémentée pour la manipulation sur ce navigateur. Elle implique en effet que l'utilisateur ferme le poing au début de la manipulation et l'ouvre à la fin (activation de la technique par une posture). Ce geste est assez inexact et dépend fortement des imprécisions du capteur LeapMotion. A l'inverse, les navigateur CS et OT effectuent cette action avec un bouton, ce qui est plus précis. De plus, le navigateur LM ne permet pas à l'utilisateur de se déplacer dans l'environnement 3D, ce qui est un handicap supplémentaire.

— Le graphe de Figure 5.7 montre l'évolution du taux de succès en fonction de la difficulté lors des tâches de sélection. Nous pouvons noter que les résultats sont plus faibles pour le navigateur LM que pour les deux autres. La technique d'interaction pour la sélection pourrait en être la cause pour deux raisons. Premièrement, l'objet est sélectionné via un pointeur dirigé selon l'index, ce qui dépend des imprécisions du capteur. Deuxièmement, le pointeur doit rester 0.5 secondes sur l'objet pour le

FIGURE 5.7 – *Évolution du taux de succès en fonction de la difficulté pour les tâches de sélection*

sélectionner, ce qui est une fois de plus moins efficace que la sélection des autres navigateurs qui s'effectue en appuyant sur un bouton.

Les résultats observés semblent correspondre aux limitations connues des navigateurs, ce qui nous permet de valider notre système. Cependant, la pertinence de nos données est à remettre en question du fait du peu de données collectées lors de l'étude. En effet, l'étude ANOVA que nous avons effectué nous permet d'accéder à une valeur de puissance statistique qui quantifie la probabilité de rejeter légitimement l'hypothèse nulle (i.e. l'hypothèse d'indépendance entre les facteurs testés et les résultats). Cette puissance statistique dépend notamment de la taille de l'échantillon de données utilisé. Nous avons noté que cette valeur était acceptable (>0.8) quand le filtrage n'était pas trop restrictif (par exemple lorsque l'on considère toutes les tâches d'un même type) mais est beaucoup plus faible dans le cas où le type, le nombre de degrés de liberté ou la difficulté sont filtrés. Enfin, il serait intéressant de compléter notre visualisation de l'étude ANOVA par un affichage des résidus qui confirmerait/infirmerait la bonne distribution des données.

Chapitre 6

Conclusion : Implications et Limites Actuelles de UMI3D

La problématique de cette thèse consistait à rendre plus accessible la conception d'expériences de Réalité Virtuelle (RV) et de Réalité Augmentée (RA) multi-supports, pour le support du travail coopératif. Nos travaux ont en ce sens conduis à la proposition d'une nouvelle architecture des médias 3D permettant de limiter la conception d'une telle expérience au développement d'un serveur appelé média 3D. La conception de ce média ne demande par ailleurs pas au(x) concepteur(s) d'expertise technique propre aux différents supports de RA/RV. Pour pouvoir interagir avec ces médias 3D, chaque support de RA/RV doit disposer d'un navigateur UMI3D dédié. Nous avons par ailleurs démontré la faisabilité technique de l'architecture proposée en l'implémentant à l'aide du moteur de jeu Unity 3D. Nos travaux ont enfin montré que l'architecture en question permet bien, d'une part d'implémenter des médias 3D supportant le travail collaboratif, et d'autre part qu'il est possible de fournir aux concepteurs de médias et de navigateurs UMI3D des outils permettant leur évaluation par des études utilisateurs objectives (mesure d'indicateurs de performance et analyses statistiques).

6.1 Résumé du problème et des contributions

Après avoir présenté dans le Chapitre 1 les outils existants pour créer des applications multi-supports en RA/RV, et expliqué leurs limites vis à vis de la problématique, nous avons proposé dans le Chapitre 2 une nouvelle architecture des applications 3D. Cette architecture sépare le média 3D (i.e. le contenu 3D, ses interactions possibles, et les règles métier) du dispositif utilisateur. Pour permettre la communication entre le média et les différents supports, nous avons introduit le protocole web UMI3D qui permet au dispositif de se connecter, via un navigateur UMI3D, à tout média 3D utilisant ce protocole. Le navigateur permet d'une part, le chargement du contenu 3D du média, et génère d'autre part dynamiquement une interface utilisateur ayant une ergonomie adaptée au support. Cette interface utilisateur permet la réalisation par l'utilisateur de contrats d'interaction établis par le média grâce au protocole. Le principal apport de la méthode de conception des médias 3D proposée est de ne pas demander au concepteur du média

de compétences techniques spécifiques aux dispositifs de RV/RA. Par ailleurs, dans le cas où les navigateurs UMI3D sont existants pour les dispositifs utilisés, la conception d'un média 3D multi-support demande le même effort de développement que la conception d'une application 3D mono-support.

Les Chapitre 3 et 4 nous permettent quant à eux de démontrer que l'architecture proposée et le protocole UMI3D répondent bien aux attendus d'un système informatique dédié au support du travail coopératif. Le Chapitre 3 démontre en effet que le protocole UMI3D, tel que présenté dans le Chapitre 2, permet l'implémentation de tous les requis pour le travail collaboratif assisté par ordinateur en environnement 3D, à l'exception du partage d'une représentation des utilisateurs et d'un canal de communication vocale. Si la communication vocale peut trivialement être implémentée en utilisant des standards dédiés comme WebRTC [1], le suivi et la représentation des utilisateurs n'était alors pas possible. Nous avons donc, dans le Chapitre 4, étendus le protocole UMI3D de sorte à permettre cette représentation des utilisateurs.

La boîte à outils présentée dans le Chapitre 3, ainsi que son utilisation dans une étude de cas, ont par ailleurs permis d'étudier les nouvelles possibilités offertes par ce protocole pour la conception de média 3D. En particulier, la combinaison d'un moteur de jeu (Unity dans notre cas) et de UMI3D nous a conduit à proposer une nouvelle méthode orientée-utilisateur pour la conception des médias 3D. Si l'utilisation de méthodes orientées utilisateur n'est pas nouvelle, la nouveauté de notre approche consiste à impliquer les utilisateurs dans une phase de conception et de paramétrage habituellement réservée au concepteur. Cette phase implique en effet habituellement de compiler puis de distribuer les modifications de l'interface pour chaque support. Notre implémentation permettant de modifier le média 3D pendant son exécution, il est possible d'avoir des itérations rapprochées avec des utilisateurs potentiels (conception à l'exécution), et ainsi de les impliquer fortement dans un processus de co-création du média (i.e. d'un média répondant à leurs problématiques métier). Nous sommes donc convaincus que l'utilisation de UMI3D, et l'application de cette méthode, peuvent conduire à une implémentation plus performante de solutions concrètes pour la collaboration en RA/RV, et à une meilleure adoption de ces solutions par les utilisateurs potentiels.

6.2 Limites du modèle d'abstraction actuel

Si l'ensemble des travaux présentés dans cette thèse nous permet de valider la faisabilité et la pertinence du modèle proposé pour le support de la collaboration en RA/RV, ce modèle reste toutefois incomplet. Si l'on souhaite adresser l'ensemble des cas d'usage de la collaboration avec les supports de RA/RV. Il faudrait pour cela à minima étendre le

1. L'implémentation de WebRTC a été récemment réalisée dans nos navigateurs UMI3D à l'aide de l'extension de Unity 3D dédiée à ce protocole.

modèle d'abstraction du protocole UMI3D pour traiter les problèmes ci-dessous.

6.2.1 Réalité Augmentée et Mixte

Pour commencer, si rien n'empêche aujourd'hui d'utiliser UMI3D pour interagir avec un média 3D en Réalité Augmentée ou en Réalité Mixte, la problématique de la pose 6-D du support reste ouverte. En effet, l'une des fonctionnalité de base en RA et RM est le positionnement, relatif au support de l'environnement réel ou d'objets réels. Nous regroupons les différentes manières d'établir la pose du support en trois catégories :

— Le positionnement, dans le référentiel du support, d'objets physiques par des marqueurs (QR Codes, images) ou leur modèle 3D. Il existe pour ce faire de nombreuses solutions ouvertes comme ARToolKit [48] ou SolAR [78], et propriétaires comme Vuforia [68], VisionLib [90] ou Wikitude [95].

— Le suivi des mouvements du support par des algorithmes de SLAM (Cartographie et localisation simultanées). Cette technique permet notamment, lorsqu'un objet virtuel est déposé en RA dans un environnement réel, de fixer la position apparente de cet objet dans l'environnement réel grâce au calcul image à image du mouvement du support. Ces algorithmes ont par ailleurs été étendus à la relocalisation simultanée de plusieurs utilisateurs par l'algorithme de CoSlam [97] et ses variantes [59, 76, 46].

— Le positionnement du support par rapport à des ancres géospatiales. Il s'agit ici de positionner des artéfacts virtuels et le support dans un repère défini par ses coordonnées dans le référentiel terrestre (latitude, longitude, altitude). Cette technique est propre à des travaux récents sur l'interaction spatialisée (Spatial Computing) visant à l'émergence d'un Cloud (nuage) dédié à la RA [2]. On peut ici citer les travaux de 6D.ai [1] qui propose une relocalisation géospatiale en six degrés de liberté grâce à un nuage de point centralisé de l'environnement réel, ou encore les travaux de la communauté Open AR Cloud pour définir les verrous technologiques restant pour l'implémenttion d'un AR Cloud multi-support [44].

Si certains standards ont été proposés par le passé, notamment pour la description de positions géospatiales (GML [24]) et/ou la description d'objets réels (ARML [56]), ils restent mal (voir pas) supportés par les outils de développements professionnels dédiés à la RA. De même, il n'y a aujourd'hui pas d'évidence concernant l'architecture et les algorithmes à utiliser pour la relocalisation simultanée de plusieurs supports de RA.

2. Le terme AR Cloud, proposé en 2017 par Inbar dans [42] est la dénomination utilisée pour désigner une technologie permettant le partage de la position d'ancres virtuelles dans le monde réel.

6.2.2 Description des zones navigables

La navigation dans les médias UMI3D est actuellement entièrement gérée par les navigateurs. Seul la définition par le média d'un 'mode' de navigation (marche, vol, orbite) a été jusqu'ici testé, sans nous convaincre. En effet, ne s'agissant que d'une indication sur la manière dont l"'utilisateur était sensé ce déplacer, aucune contrainte n'était réellement imposée aux navigateurs. Cette approche ne permettait ainsi pas de décrire la manière dont l'utilisateur doit être capable de se déplacer dans un média 3D (zones navigables, zones interdites, points de téléportation, ascenseurs, véhicules, ...). Une des approches que nous souhaitons explorer à l'avenir est l'établissement d'un champ tri-dimensionnel dynamique de potentiel décrivant le coût de déplacement d'un point à un autre. Cette approche est notamment utilisée en robotique pour le déplacement autonome de véhicules [49].

6.2.3 Interaction directes entre un avatar et un média 3D

L'envoi par les navigateurs UMI3D d'informations sur le déplacement de l'utilisateur devrait, en plus de permettre une représentation des utilisateurs comme présenté dans le chapitre 4, fournir un moyen d'aborder efficacement l'interaction physique entre le corps de l'utilisateur et le media 3D. L'avatar de l'utilisateur étant en effet définis par le média 3D, celui-ci peut par exemple générer une réaction aux collisions entre l'avatar et les autres objets 3D le composant. Cette possible simulation physique ne nous semble toutefois pas suffisante. Nous étudions en effet actuellement la possibilité de lier certaines briques d'interaction avec un objet de sorte à guider/contrôler sa manipulation directe, par exemple par les mains de l'utilisateur. Nous envisageons par ailleurs la proposition d'une extension du protocole UMI3D de sorte à permettre une description générique de la physique du média 3D pour la génération de retours haptiques (ou pseudo-haptiques) par les supports en ayant la capacité. Cette description permettrait par exemple à un support disposant de périphériques à retour d'effort de simuler dynamiquement une interaction physique réaliste entre le média 3D et l'utilisateur. Le retour haptique pourrait alors être calculé localement au vu de la fréquence élevée demandée par le calcul de retours haptiques (typiquement 1000 Hz). À l'inverse, la mise à jour paramètres physiques des objets (poids, vitesse, élasticité) pourrait être rafraichis par le média (30 à 90 Hz).

6.2.4 Interprétation des performances utilisateur

Les outils et résultats préliminaires présentés dans le chapitre 5 nous permettent d'affirmer qu'il est possible de mener des études utilisateur objectives de l'interaction avec UMI3D. Il faut pour cela coupler des données collectées en temps-réel par le média et par le navigateur UMI3D de l'utilisateur. Les données ainsi consolidées permettent alors

le calcul d'indicateurs de performance et la conduite d'études statistiques courantes pour l'évaluation de l'interaction en environnement 3D. Il est en particulier possible de comparer aisément les performances utilisateurs obtenus avec différents supports (i.e. avec différents navigateurs UMI3D) lors de l'interaction avec un même média 3D. On pourra donc conclure qu'un navigateur est plus ou moins adapté à l'activation de certaines briques d'interaction. Il est par ailleurs possible d'évaluer de manière globale la performance d'un navigateur dans l'exécution d'une tâche, c'est à dire la capacité de l'utilisateur à contrôler dynamiquement, par son navigateur, la réalisation d'un contrat d'interaction en fonction de la réaction produite par le média. Nous avons identifié un paramètre ayant une influence importante sur la performance des manipulations dans ces conditions : L'utilisateur doit, comme pour une technique d'interaction classique, pourvoir intuitivement associer la manipulation de ses périphériques d'entrées à la réaction produite par le média 3D. Les travaux de R.J.K. Jacob & al. [43] nous indiquent ici que pour faciliter cette appropriation, la structure perceptuelle de la tâche doit être adaptée à la structure de contrôle du périphérique. Ceci implique à notre sens, d'une part que la traduction en briques d'interaction de la tâche doit retranscrire au mieux sa structure perceptuelle. D'autre part la technique d'interaction mise en œuvre par le navigateur, pour activer les briques d'interaction, doit avoir une structure perceptuelle adaptée à ces briques, et impliquer l'utilisation d'un(ou des) périphérique(s) d'entrées du support dont la structure de contrôle est adaptée aux structures de la tâche et de la technique d'activation des briques d'interaction. Au vu du manque logique de connaissance sur l'interprétation des briques d'interaction par les différents supports, nous soutenons que de nombreuses études utilisateur, concernant des applications variées des briques d'interaction de UMI3D, devront être menées avant que nous ayons un recul suffisant pour expliquer finement les performances, c'est à dire identifier dans toute situation si d'éventuels résultats négatifs proviennent de l'association par le média d'une réaction inadaptée aux briques d'interaction choisies, ou plutôt d'une mauvaise interprétation de ces briques par le navigateur UMI3D. Enfin, nous pensons que la création de ces connaissances passera par une réflexion de fond sur la terminologie a adopter pour décrire la réalisation d'une tâche par un navigateur UMI3D. La notion de 'technique d'interaction' nous semble en effet ici inadaptée au vu de la fragmentation de cette technique entre le média 3D et le navigateur.

6.3 Améliorations possibles de nos méthodes de conception et d'évaluation

6.3.1 Optimisation du nombre de tests utilisateur

Lors de l'étude de cas présentée en fin de Chapitre 3, nous avons montré que UMI3D peut être combiné à un moteur de jeu pour permettre des itérations très courtes (mo-

dification du média durant l'exécution des tests utilisateurs) avec les utilisateurs finaux. Leur implication dans la conception de solution est ainsi accrue. Toutefois, la chronologie présentée Figure 3.8 montre bien que plusieurs sessions ont été nécessaires à la conception de l'application finale. Or la mobilisation d'utilisateurs finaux reste problématique, au moins dans un contexte professionnel, car elle demande une mise à disposition de ressources, et éventuellement d'un environnement de travail réel. Il pourrait donc être intéressant d'améliorer notre méthode de conception de sorte à regrouper les sessions de conception à l'exécution. Une piste de travail pourrait être l'adaptation de la méthode 'Anticipated User Experience and Conceptual Development' présentée par Kukka et al. dans [52]. Cette méthode permettrait de pré-concevoir sans développement l'expérience finale attendue par les utilisateurs finaux, et ainsi d'établir le concept de la solution. Cela permettrait également de mieux qualifier les contenus et interactions à préparer en vue d'une unique session de conception à l'exécution. Il est toutefois également possible que ce regroupement soit contre-productif, car ne laissant pas suffisamment de temps aux utilisateurs et au concepteur pour maturer le concept de la solution.

6.3.2 Transformation des méthodes d'évaluation

Si la nouvelle méthode de conception que nous proposons simplifie les tests utilisateur en réduisant le nombre de mobilisations des utilisateurs, la captation et l'analyse de données objectives (i.e. d'indicateurs de performance mesurés et non issus de questionnaires utilisateur) sont complexifiées par l'architecture distribuée de notre modèle. En effet, seule une partie de l'information, notamment l'évolution de la scène 3D et les évènements métier, est localisée sur le média 3D (i.e. dans un serveur). Les données liées aux mouvements des utilisateurs, et surtout à leur manipulation des interfaces utilisateurs, ne sont disponibles que dans les navigateurs UMI3D. Il en résulte une nécessité de mise en commun et de croisement de ces données, de sorte à permettre des mesures et analyses empiriques de l'utilisation des médias 3D conçus avec UMI3D. Cette mise en relation des données du média et des navigateurs permettra d'appliquer l'ensemble des méthodes existantes pour l'évaluation de la collaboration et des environnements 3D. Par exemple, en adaptant les travaux de Xu et al. [96], il serait possible des prédire les zones d'intérêts utilisateur (contenus attirants l'attention des utilisateurs) du média 3D. Pour ce faire il faudrait, entre autre, mettre en relation les interactions avec les contenus (pointage, sélection, et manipulation), avec l'attention visuelle des utilisateurs.

Pour faciliter les études utilisateurs dans le cadre de l'utilisation de UMI3D, nous avons réalisé des travaux préliminaires [20] visant à connecter nos outils de développement (notre boite à outils pour la conception de médias, ainsi que notre package Unity pour la conception de navigateurs UMI3D) à une plateforme cloud d'analyse de données massives. Nous avons ensuite utilisé cette connexion pour créer un système de compa-

raisons de techniques d'interaction. Cet exemple a été retenu car nous pensons que la possibilité de partager un même média 3D entre plusieurs équipes de recherche (étudiant des dispositifs différents) peut améliorer la comparaison de données expérimentales entre équipes. Cela garantirait d'une part l'absence de biais dans l'implémentation du média et des interactions (utilisation du même exécutable / du même serveur), et pourrait permettre de ne capter des données que pour le dispositif étudié, la comparaison se faisant avec des données antérieures provenant d'une autre équipe.

6.4 Vers un standard pour la collaboration en environnement 3D

Pour conclure nos travaux, nous nous sommes interrogés sur la démarche à adopter pour lever les limites actuelles du protocole UMI3D et l'industrialiser. Lors de cette réflexion, nous avons envisagé deux possibilités. La première aurait été l'industrialisation d'un 'framework' (infrastructure de développement en français) propriétaire. Cette solution a été rapidement écartée car elle nous semblait en opposition avec les objectifs même de UMI3D. En effet, l'objectif étant de pouvoir connecter tout dispositif de RV/RA, à tout média 3D, cela n'aurait pas été envisageable pour une unique organisation de fournir le support pour tout nouveau dispositif, y compris unique comme un simulateur industriel. C'est pourquoi les deux organisations à l'origine de ce projet de thèse, Gfi et le laboratoire ICube de l'Université de Strasbourg, ont récemment fait le choix de fonder le Consortium UMI3D. Ce projet collaboratif aura pour but de réunir l'ensemble de la communauté RA/RV autour de la standardisation de UMI3D, sous la forme d'un protocole d'échange libre de droit. Il restera donc possible de concevoir librement des navigateurs pour les médias 3D existants, ou inversement des médias 3D utilisables par les navigateurs existants.

Bibliographie

[1] 6D.ai. 6D.ai web page. `https://www.6d.ai`, 2019. Online ; accessed 10 July 2019.

[2] Naveed Ahmed, Edilson De Aguiar, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Automatic generation of personalized human avatars from multi-view video. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 257–260. ACM, 2005.

[3] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE Comput. Graph. Appl.*, 21(6) :34–47, November 2001.

[4] Roger Barga, Valentine Fontama, and Wee Hyong Tok. *Integration with R*, pages 81–101. Apress, Berkeley, CA, 2015.

[5] Michel Beaudouin-Lafon. User interface management systems : Present and future. In Sabine Coquillart, Wolfgang Straßer, and Peter Stucki, editors, *From Object Modelling to Advanced Visual Communication*, pages 197–223, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[6] Michel Beaudouin-Lafon. Designing interaction, not interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 15–22, New York, NY, USA, 2004. ACM.

[7] Steve Benford, John Bowers, Lennart E. Fahlén, and Chris Greenhalgh. Managing mutual awareness in collaborative virtual environments. In *Proceedings of the Conference on Virtual Reality Software and Technology*, VRST '94, pages 223–236, River Edge, NJ, USA, 1994. World Scientific Publishing Co., Inc.

[8] Steve Benford, John Bowers, Lennart E Fahlén, Chris Greenhalgh, and Dave Snowdon. User embodiment in collaborative virtual environments. In *Chi*, volume 95, pages 242–249. Citeseer, 1995.

[9] Gary Bente, Sabine Rüggenberg, Nicole C. Krämer, and Felix Eschenburg. Avatar-mediated networking : Increasing social presence and interpersonal trust in net-based collaborations. *Human Communication Research*, 34(2) :287–318, 2008.

[10] François Bérard, Jessica Ip, Mitchel Benovoy, Dalia El-Shimy, Jeffrey R. Blum, and Jeremy R. Cooperstock. Did "minority report" get it wrong ? superiority of the mouse over 3d input devices in a 3d placement task. In Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates, and Marco Winckler, editors, *Human-Computer Interaction – INTERACT 2009*, pages 400–414, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[11] J. Gordon Betts, Peter DeSaix, Eddie Johnson, Jody E. Johnson, Oksana Korol, Dean H. Kruse, Brandon Poe, James A. Wise, Mark Womble, and Kelly A. Young. Openstax, anatomy and physiology, 2017. `https://openstax.org/details/books/anatomy-and-physiology`.

[12] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. Vr juggler : a virtual platform for virtual reality application development. In *Proceedings IEEE Virtual Reality 2001*, pages 89–96, March 2001.

[13] Richard A Bolt. *"Put-that-there" : Voice and gesture at the graphics interface*, volume 14. ACM, 1980.

[14] Dhruba Borthakur. The hadoop distributed file system : Architecture and design. *Hadoop Project Website*, 11(2007) :21, 2007.

[15] Rozenn Bouville, Valérie Gouranton, Thomas Boggini, Florian Nouviale, and Bruno Arnaldi. #five : High-level components for developing collaborative and interactive virtual environments. In *2015 IEEE 8th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 33–40, March 2015.

[16] Douglas A. Bowman. *Interaction Techniques for Common Tasks in Immersive Virtual Environments : Design, Evaluation, and Application*. PhD thesis, Atlanta, GA, USA, 1999. AAI9953819.

[17] J-M. Burkhardt, V. Corneloup, C. Garbay, Y. Bourrier, F. Jambon, V. Luengo, A. Job, Ph. Cabon, A. Benabbou, and D. Lourdeaux. Simulation and virtual reality-based learning of non-technical skills in driving : critical situations, diagnostic and adaptation. *IFAC-PapersOnLine*, 49(32) :66 – 71, 2016. Cyber-Physical & Human-Systems CPHS 2016.

[18] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture - Volume 1 : A System of Patterns*. Wiley Publishing, 1996.

[19] Julien Casarin, Dominique Bechmann, and Marilyn Keller. A unified model for interaction in 3d environment. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, VRST '17, pages 23 :1–23 :7, New York, NY, USA, 2017. ACM.

[20] Julien Casarin, Antoine Ladrech, Tristan Tchilinguirian, and Dominique Bechmann. A umi3d-based interactions analytics system for xr devices and interaction techniques. In *Proceedings of the 2019 IEEE VR Conference*, VR '19, Washington, DC, USA, 2019. IEEE Computer Society.

[21] Julien Casarin, Nicolas Pacqueriaud, and Dominique Bechmann. Umi3d : A unity3d toolbox to support cscw systems properties in generic 3d user interfaces. *Proceedings of the ACM on Human-Computer Interaction*, 2018.

[22] David Chappell. Introducing odata. *Data Access for the Web, The Cloud, Mobile Devices, and More*, pages 1–24, 2011.

[23] Elizabeth F. Churchill and David Snowdon. Collaborative virtual environments : An introductory review of issues and systems. *Virtual Reality*, 3(1) :3–15, Mar 1998.

[24] Simon Cox, Adrian Cuthbert, Ron Lake, and Richard Martell. Geography markup language (gml) 2.0. *URL : http ://www. opengis. net/gml/01-029/GML2. html*, 2001.

[25] Brian Danchilla. *Three.js Framework*, pages 173–203. Apress, Berkeley, CA, 2012.

[26] Marc De Groot. Multi-user virtual reality system for simulating a three-dimensional environment, #jul# 16 2002. US Patent 6,421,047.

[27] Alan Dix. *Computer Supported Cooperative Work : A Framework*, pages 9–26. Springer London, London, 1994.

[28] Alan Dix, Adrian Friday, Boriana Koleva, Tom Rodden, Henk Muller, Cliff Randell, and Anthony Steed. Multiple spaces. In *Spaces, Spatiality and Technology*, pages 151–172. Springer, 2005.

[29] Paul Dourish. Using metalevel techniques in a flexible toolkit for cscw applications. *ACM Trans. Comput.-Hum. Interact.*, 5(2) :109–155, June 1998.

[30] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work*, CSCW '92, pages 107–114, New York, NY, USA, 1992. ACM.

[31] Nicolas Ducheneaut, Ming-Hui Wen, Nicholas Yee, and Greg Wadley. Body and mind : a study of avatar personalization in three virtual worlds. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1151–1160. ACM, 2009.

[32] Florent Dupont, Thierry Duval, Cédric Fleury, Julien Forest, Valérie Gouranton, Pierre Lando, Thibaut Laurent, Guillaume Lavoué, and Alban Schmutz. Collaborative Scientific Visualization : The COLLAVIZ Framework. In *JVRC 2010 (2010 Joint Virtual Reality Conference of EuroVR - EGVE - VEC)*, Fellbach, Germany, October 2010.

[33] Elham Ebrahimi, Leah S Hartman, Andrew Robb, Christopher C Pagano, and Sabarish V Babu. Investigating the effects of anthropomorphic fidelity of self-avatars on near field depth perception in immersive virtual environments. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1–8. IEEE, 2018.

[34] Mohammed S Elbamby, Cristina Perfecto, Mehdi Bennis, and Klaus Doppler. Toward low-latency and ultra-reliable virtual reality. *IEEE Network*, 32(2) :78–84, 2018.

[35] Falko Kuester Emad Barsoum. Webvr : an interactive web browser for virtual environments. volume 5664, pages 5664 – 5664 – 8, 2005.

[36] Susan R. Fussell, Robert E. Kraut, and Jane Siegel. Coordination of communication : effects of shared visual context on collaborative work. In *Proceedings of the 2000 ACM Conference on Computer-supported Cooperative Work*, CSCW '00, pages 21–30, 2000.

[37] Guillaume Gamelin, Amine Chellali, Cédric Dumas, and Samir Otmane. Fidélité de l'avatar du partenaire distant dans un environnement virtuel immersif : effets sur les interactions spatiales. In *30eme conférence francophone sur l'interaction homme-machine*, page 7p, 2018.

[38] Henry Gray. *Anatomy of the human body*, volume 8. Lea & Febiger, 1878.

[39] Tom Gross. Supporting effortless coordination : 25 years of awareness research. *Computer Supported Cooperative Work (CSCW)*, 22(4) :425–474, Aug 2013.

[40] Jean-Marc Hasenfratz, Marc Lapierre, and François X Sillion. A real-time system for full body interaction with virtual worlds. In *Eurographics Symposium on Virtual Environments*, pages 147–156, 2004.

[41] Ross Ihaka and Robert Gentleman. R : a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3) :299–314, 1996.

[42] Inbar, Ori. Arkit and arcore will not usher massive adoption of mobile ar. `https://medium.com/super-ventures-blog/arkit-and-arcore-will-not-usher-massive-adoption-of-mobile-ar-da3d87f7e5ad`, 2017. Online ; accessed 09 July 2019.

[43] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and M. Preston Mullen, Jr. Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact.*, 1(1) :3–26, March 1994.

[44] Jan Erik Vinje, Colin Steinmann, Steven Swanson, Christine Perey, Arka Bala, Chris Nunes, Dave Lorenzini, Rolf Kruse, Jason Fox, Tony Hodgson, James Jackson, Marco Tillmann, Scott Simmons, Preston McCauley, Michael Vogt, Alina Kadlubsky, John Daly, Suzan Oslin, and Julien Casarin. State of the ar cloud report - 2019. `https://www.openarcloud.org/reports/sotarc-2019`, 2019. Online ; accessed 09 July 2019.

[45] Florian Jeanne, Indira Thouvenin, and Alban Lenglet. A study on improving performance in gesture training through visual guidance based on learners' errors. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, VRST '17, pages 24 :1–24 :10, New York, NY, USA, 2017. ACM.

[46] Marco Karrer, Patrik Schmuck, and Margarita Chli. Cvi-slam—collaborative visual-inertial slam. *IEEE Robotics and Automation Letters*, 3(4) :2762–2769, 2018.

[47] David J. Kasik. A user interface management system. *SIGGRAPH Comput. Graph.*, 16(3) :99–106, July 1982.

[48] H. Kato, K. Tachibana, M. Billinghurst, and M. Grafe. A registration method based on texture tracking using artoolkit. In *2003 IEEE International Augmented Reality Toolkit Workshop*, pages 77–85, Oct 2003.

[49] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[50] Konstantina Kilteni, Raphaela Groten, and Mel Slater. The sense of embodiment in virtual reality. *Presence : Teleoperators and Virtual Environments*, 21(4) :373–387, 2012.

[51] Boriana Koleva, Holger Schnädelbach, Steve Benford, and Chris Greenhalgh. Traversable interfaces between real and virtual worlds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pages 233–240, New York, NY, USA, 2000. ACM.

[52] Hannu Kukka, Minna Pakanen, Mahmoud Badri, and Timo Ojala. Immersive street-level social media in the 3d virtual city : Anticipated user experience and conceptual development. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 2422–2435, New York, NY, USA, 2017. ACM.

[53] Sebastien Kuntz. Middlevr a generic vr toolbox. In Tobias Höllerer, Victoria Interrante, Anatole Lécuyer, and J. Edward Swan II, editors, *VR*, pages 391–392. IEEE Computer Society, 2015.

[54] Jérémy Lacoche. *Plasticity for user interfaces in mixed reality.* PhD thesis, Université Bretagne Loire (ComuE) , Institut de recherche en informatique et systèmes aléatoires (Rennes), 2016.

[55] Stan Larroque and Julien Casarin. Para : experimental device for virtual and augmented reality. In *Proceedings of Digital Optics for Immersive Displays (DOID 2018)*, volume 10676, 2018.

[56] Martin Lechner. Ogc augmented reality markup language 2.0 (arml 2.0), version 1.0. 2015.

[57] A. Leff and J. T. Rayfield. Web-application development using the model/view/controller design pattern. In *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, pages 118–127, Sep. 2001.

[58] Jason Leigh and Andrew E. Johnson. Supporting transcontinental collaborative work in persistent virtual environments. *IEEE Comput. Graph. Appl.*, 16(4) :47–51, July 1996.

[59] Giuseppe Loianno, Yash Mulgaonkar, Chris Brunner, Dheeraj Ahuja, Arvind Ramanandan, Murali Chari, Serafin Diaz, and Vijay Kumar. A swarm of flying smartphones. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1681–1688. IEEE, 2016.

[60] Thomas W. Malone and Kevin Crowston. What is coordination theory and how can it help design cooperative work systems ? In *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work*, CSCW '90, pages 357–370, New York, NY, USA, 1990. ACM.

[61] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality : a class of displays on the reality-virtuality continuum. volume 2351, pages 282–292, 1995.

[62] Raquel Oliveira, Sophie Dupuy-Chessa, and Gaëlle Calvary. Verification of plastic interactive systems. In Jürgen Ziegler, editor, *i-com : Vol. 14, No. 3*, page 192–204, Berlin, 2015. De Gruyter.

[63] Dan Olsen. *User Interface Management Systems : Models and Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[64] OpenXR. OpenXR web page. `https://www.khronos.org/openxr`, 2019. Online ; accessed 07 July 2019.

[65] Reinhard Oppermann. Adaptively supported adaptability. *Int. J. Hum.-Comput. Stud.*, 40(3) :455–472, March 1994.

[66] Michael Ortega and Laurence Nigay. Airmouse : Finger gesture for 2d and 3d interaction. In Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates, and Marco Winckler, editors, *Human-Computer Interaction – INTERACT 2009*, pages 214–227, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[67] Nassima Ouramdane. *Vers un système d'assistance à l'interaction 3D pour le travail et le télétravail collaboratif dans les environnements de réalité virtuelle et augmentée*. PhD thesis, 2008. Thèse de doctorat dirigée par Mallem, Malik Informatique Evry-Val d'Essonne 2008.

[68] PTC. Vuforia Engine web page. `https://engine.vuforia.com/engine`, 2019. Online ; accessed 08 July 2019.

[69] Kjetil Raaen and Ivar Kjellmo. Measuring latency in virtual reality systems. In *International Conference on Entertainment Computing*, pages 457–462. Springer, 2015.

[70] Raghu Ramakrishnan, Baskar Sridharan, John R. Douceur, Pavan Kasturi, Balaji Krishnamachari-Sampath, Karthick Krishnamoorthy, Peng Li, Mitica Manu, Spiro Michaylov, Rogério Ramos, Neil Sharman, Zee Xu, Youssef Barakat, Chris Douglas, Richard Draves, Shrikant S. Naidu, Shankar Shastry, Atul Sikaria, Simon Sun, and Ramarathnam Venkatesan. Azure data lake store : A hyperscale distributed file service for big data analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 51–63, New York, NY, USA, 2017. ACM.

[71] Alberto B Raposo, Adailton J.A da Cruz, Christian M Adriano, and Léo P Magalhães. Coordination components for collaborative virtual environments. *Computers & Graphics*, 25(6) :1025 – 1039, 2001. Artificial Life.

[72] Mathieu Raynal, Emmanuel Dubois, and Bénédicte Schmitt. Towards unification for pointing task evaluation in 3d desktop virtual environment. In Andreas Holzinger, Martina Ziefle, Martin Hitz, and Matjaž Debevc, editors, *Human Factors in Computing and Informatics*, pages 562–580, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[73] Élisabeth Rousset, François Bérard, and Michaël Ortega. Two-finger 3d rotations for novice users : Surjective and integral interactions. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 217–224, New York, NY, USA, 2014. ACM.

[74] Kjeld Schmidt. *Cooperative Work and Coordinative Practices : Contributions to the Conceptual Foundations of Computer-Supported Cooperative Work (CSCW)*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[75] Kjeld Schmidt and Carla Simonee. Coordination mechanisms : Towards a conceptual foundation of cscw systems design. *Computer Supported Cooperative Work (CSCW)*, 5(2) :155–200, Jun 1996.

[76] Patrik Schmuck and Margarita Chli. Ccm-slam : Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics*, 36(4) :763–781, 2019.

[77] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–10, May 2010.

[78] SolAR. SolAR framework web page. `https://solarframework.github.io/`, 2019. Online ; accessed 08 July 2019.

[79] Maximilian Speicher, Brian D. Hall, and Michael Nebeling. What is mixed reality ? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 537 :1–537 :15, New York, NY, USA, 2019. ACM.

[80] David J Sturman, David Zeltzer, and Steve Pieper. Hands-on interaction with virtual environments. In *Proceedings of the 2nd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 19–24. ACM, 1989.

[81] Tuukka M. Takala. Ruis : A toolkit for developing virtual reality applications with spatial interaction. In *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*, SUI '14, pages 94–103, New York, NY, USA, 2014. ACM.

[82] Josh Tenenberg, Wolff-Michael Roth, and David Socha. From i-awareness to we-awareness in cscw. *Computer Supported Cooperative Work (CSCW)*, 25(4) :235–278, Oct 2016.

[83] David Thevenin and Joëlle Coutaz. Plasticity of user interfaces : Framework and research agenda. In *Interact*, volume 99, pages 110–117, 1999.

[84] Aphrodite Tsalgatidou, M Pantazoglou, and G Athanasopoulos. Specification of the unified service query language (usql). *Dept. of Informatics & Telecom. University of Athens*, 2006.

[85] J. Tumler, F. Doil, R. Mecke, G. Paul, M. Schenk, E. A. Pfister, A. Huckauf, I. Bockelmann, and A. Roggentin. Mobile augmented reality in industrial applications : Approaches for solution of user-related issues. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 87–90, Sept 2008.

[86] Unity. Unity XR Input web page. `https://docs.unity3d.com/Manual/xr_input.html`, 2019. Online ; accessed 08 July 2019.

[87] M Veit, A Capobianco, and D Bechmann. An experimental analysis of the impact of touch screen interaction techniques for 3-d positioning tasks. In *Proceedings of the 2011 IEEE Virtual Reality Conference*, VR '11, pages 75–82, Washington, DC, USA, 2011. IEEE Computer Society.

[88] Manuel Veit, Antonio Capobianco, and Dominique Bechmann. Influence of degrees of freedom's manipulation on performances during orientation tasks in virtual reality environments. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, pages 51–58, New York, NY, USA, 2009. ACM.

[89] Manuel Veit, Antonio Capobianco, and Dominique Bechmann. Dynamic decomposition and integration of degrees of freedom for 3-d positioning. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, VRST '10, pages 131–134, New York, NY, USA, 2010. ACM.

[90] Visometry. VisionLib web page. `https://visionlib.com/`, 2019. Online ; accessed 08 July 2019.

[91] VRTK. VRTK Documentation web page. `https://vrtoolkit.readme.io/docs`, 2019. Online ; accessed 08 July 2019.

[92] Séamas Weech, Sophie Kenny, and Michael Barnett-Cowan. Presence and cybersickness in virtual reality are negatively related : a review. *Frontiers in psychology*, 10 :158, 2019.

[93] Wikipedia. Wikipedia data transfer object. `https://en.wikipedia.org/wiki/Data_transfer_object`, 2019. Online ; accessed 10 July 2019.

[94] Wikipedia. Wikipedia list of game engines. `https://en.wikipedia.org/wiki/List_of_game_engines`, 2019. Online ; accessed 07 July 2019.

[95] Wikitude. Wikitude web page. `https://www.wikitude.com`, 2019. Online ; accessed 08 July 2019.

[96] Pingmei Xu, Yusuke Sugano, and Andreas Bulling. Spatio-temporal modeling and prediction of visual attention in graphical user interfaces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3299–3310, New York, NY, USA, 2016. ACM.

[97] Danping Zou and Ping Tan. Coslam : Collaborative visual slam in dynamic environments. *IEEE transactions on pattern analysis and machine intelligence*, 35(2) :354–366, 2012.