Plasticity for User Interfaces in Mixed Reality

Jérémy Lacoche

ANNÉE 2016

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale Matisse

présentée par

Jérémy Lacoche

préparée à l'unité de recherche 6074 IRISA
Institut de Recherche en Informatique et Systèmes Aléatoires
ISTIC

Plasticity for User Interfaces in Mixed Reality

Thèse soutenue à Rennes
le 21 juillet 2016

devant le jury composé de :

Guillaume Moreau
Professeur à l'université de Nantes / Président

Torsten Kuhlen
Professeur à l'université de RWTH Aachen / rapporteur

Laurence Nigay
Professeur à l'université de Grenoble / rapporteur

Sabine Coquillart
Directeur de recherche à INRIA Grenoble / examinateur

Thierry Duval
Professeur à Télécom Bretagne / directeur de thèse

Bruno Arnaldi
Professeur à l'INSA de Rennes / examinateur

Eric Maisel
Maître de conférences à l'ENIB / examinateur

Jérôme Royan
Chef de laboratoire R&D à b<>com / examinateur

Remerciements

Cette thèse aura été pour moi une expérience très enrichissante d'un point de vue professionnel mais également d'un point de vue humain. J'espère ici, sans en oublier, remercier toutes les personnes qui ont contribué de près ou de loin à la réussite de mes travaux pendant ces trois ans.

Tout d'abord, je veux remercier Guillaume Moreau de m'avoir fait l'honneur de présider mon jury de thèse. J'aimerais ensuite remercier Laurence Nigay et Torsten Kuhlen d'avoir accepté d'être rapporteurs et de m'avoir donné des remarques très pertinentes et constructives concernant mes travaux. Je tiens également à remercier Sabine Coquillart d'avoir accepté de participer à mon jury de thèse.

Je souhaite ensuite remercier tout particulièrement Thierry Duval pour m'avoir fait confiance en acceptant de diriger cette thèse. Merci pour tes encouragements, tes conseils et ta disponibilité tout aux long de mes travaux de thèse. Tes idées, tes connaissances et ta gentillesse ont été un vrai soutien pendant ces trois ans. Un grand merci également à Jérôme Royan, pour ta confiance, ton suivi au quotidien, tes idées très créatives et la motivation que tu as su m'apporter durant cette thèse. Je tiens également à remercier Bruno Arnaldi et Eric Maisel pour leur encadrement, pour avoir partagé avec moi leurs connaissances et qui on su me guider sur des sujets très variés.

Par ailleurs je remercie l'IRT b<>com pour les conditions idéales dont j'ai pu bénéficier pendant mes trois années de thèse. Participer au démarrage de l'institut a été pour moi une expérience très enrichissante. Je tiens à remercier l'ensemble des membres de l'équipe I2 en particulier Amine et Morgan mes compagnons de thèse pendant ces trois ans. Amine, qui a réussi à supporter mes blagues douteuses pendant toute cette période et qui, même s'il ne fait pas de passes, restera mon partenaire footballistique favori. Morgan, qui a été mon partenaire de travail mais également de conférences en particulier lors des repas de gala! Vous avez été tous les deux d'excellents collègues de travail et un soutien de poids pendant cette thèse. Pour toutes les pauses, les sorties et les moments de rigolade, je vous remercie! Un petit bémol malgré tout, je ne vous remercie pas pour le tabagisme passif que vous m'avez fait subir pendant ces trois années! Merci également à Sébastien et Thomas qui m'ont procuré une aide essentielle dans la réalisation de mes travaux Je tiens ensuite à remercier Delphine, Véronique et Karen qui pendant ces trois ans ont grandement simplifié mes démarches administratives et mes déplacements, et ce toujours dans la bonne humeur. J'aimerais également remercier Florian et Pape mes partenaires de pause, de foot et de Fifa.

Merci également au laboratoire IRISA, en particulier à l'équipe Hybrid de m'avoir accueilli. Merci notamment à Anatole et à Nathalie pour leur gentillesse. Je garderai un très bon souvenir des discussions avec les membres de l'équipe notamment lors des séminaires.

Je tiens ensuite à remercier tous mes amis, en premier lieu mes amis de longue date, Maxime et Simon. Merci également à tous mes amis de l'ESIR, je pense entre autres à David, Guilhem, Jude, Julien, Maxime, Antoine, Delphine et Estelle avec qui j'ai passé cinq années d'études inoubliables.

Je remercie également mes parents qui m'ont toujours soutenu dans la réalisation de mes projets, je vous dois beaucoup dans la réussite de mes études. Merci également à mon oncle Gilles pour son soutien pendant ces trois ans. Merci à vous trois et à Jeans Yves d'être venus m'encourager lors de ma soutenance.

Enfin, un immense merci à Marie pour son soutien et ses encouragements quotidiens pendant ces trois ans. Merci pour ta patience et ton écoute malgré les journées bien remplies. Tu as largement contribué à la réussite de cette thèse. J'aimerais également remercier sa famille pour leurs nombreux encouragements.

Contents

Résumé en Français

Introduction

Dans ce manuscrit de thèse, intitulé «Plasticité pour les interfaces de réalité mixte», nous présentons des travaux de recherche se plaçant dans le contexte des interfaces homme-machine et de la réalité mixte.

Le terme "réalité mixte" fait référence selon Milgram [Milgram and Kishino, 1994] à un continuum allant du monde réel au monde virtuel qui englobe trois domaines particuliers. La "Réalité Virtuelle" consiste à immerger via différent stimuli sensoriels un ou plusieurs utilisateurs dans un environnement virtuel généré par ordinateur dans lequel ils peuvent interagir. La "Réalité Augmentée" quant à elle consiste à intégrer des objets virtuels générés par ordinateur dans un environnement réel. Ces éléments peuvent être superposés à l'environnement réel ou bien combinés avec. La "Virtualité Augmentée" consiste à intégrer et à utiliser dans un environnement virtuel des éléments du monde réel.

Aujourd'hui l'engouement est très fort autour de ce type d'applications grâce à l'arrivée de nouveaux dispositifs d'interaction et d'affichage ainsi qu'à l'augmentation des performances graphiques des ordinateurs et des plateformes mobiles. De nombreux cas d'usages bénéficient de l'utilisation de ces applications comme le divertissement, la formation, la visualisation de données, etc. Toutefois, le développement de ces applications comprend différentes problématiques. Tout d'abord il y un grand nombre de dispositifs d'affichage et d'interaction qui peuvent être utilisés en réalité mixte et tous n'offrent pas les mêmes capacités. Ensuite les interfaces de réalité mixte peuvent utiliser une plus large variété de techniques d'interaction que les interfaces 2D classiques. De même les possibilités de représentation du contenu sont plus importantes grâce à la troisième dimension.

Développer manuellement une application pour chacune des configurations possibles n'est pas une manière flexible pour créer de telles interfaces. Des solutions sont nécessaires pour s'adapter dynamiquement à ce qu'on appelle le contexte d'usage, c'est-à-dire l'ensemble des informations qui permettent de caractériser l'interaction entre l'utilisateur et le système. Cette thèse s'intéresse à la plasticité, un concept venant des interfaces 2D qui vise à traiter ces problématiques. La plasticité d'une interface est sa capacité à s'adapter aux contraintes matérielles et environnementales dans le respect de son utilisabilité [Thevenin and Coutaz, 1999]. La portabilité du code est nécessaire mais n'est pas suffisante pour qu'un système interactif soit considéré plastique. La continuité de l'utilisabilité doit être assurée quel que soit le contexte d'usage. Pour cela, les systèmes plastiques intègrent généralement des mécanismes d'adaptation qui vont prendre en compte un certain nombre d'éléments du contexte d'usage afin de configurer de manière statique ou dynamique une application. Par exemple, un exemple d'adaptation pourrait consister à changer les techniques d'interaction en fonction des périphériques disponibles et des préférences d'un utilisateur.

Considérer ce concept de plasticité lors du développement d'une interface de réalité mixte peut avoir de nombreux avantages comme la réduction des temps et des coûts de maintenance et de développement, la possibilité de diffuser son application très largement sans s'occuper des caractéristiques de l'utilisateur final et de ses périphériques, ou encore une attractivité de l'application améliorée.

Le but de cette thèse est de proposer des modèles et une solution logicielle afin de pouvoir créer des interfaces de réalité mixte plastiques. Ainsi nos contributions s'articulent autour d'un même outil logiciel pour la création de telles interfaces: 3DPlasticToolkit.

Contributions

État de l'art et proposition d'un espace problème

La première partie de cette thèse présente un état de l'art de la plasticité pour les interfaces 2D et pour les interfaces de réalité mixte. Cet état de l'art permet de démontrer quelles sont les façons d'adapter ce type d'interfaces et quelles sont les solutions actuelles. Ainsi nous pouvons proposer un espace problème pour le développement des interfaces de réalité mixte ainsi qu'un ensemble d'exigences qu'une solution doit respecter pour développer ce type d'applications. Cet espace problème sépare la problématique sur quatre axes. Le premier groupe est celui des sources d'adaptations qui correspondent aux éléments du contexte d'usage qui vont entrainer une adaptation. Pour la réalité mixte nous avons identifié 3 sources possibles. Tout d'abord les périphériques disponibles, c'est-à-dire l'ensemble des dispositifs d'interaction et d'affichage connectés à la plateforme ciblée. La seconde source possible correspond aux utilisateurs et leur environnement. Cet ensemble comprend la description des utilisateurs, leurs préférences ainsi que la description des environnements qui les entourent. La dernière source correspond aux données manipulées dans l'application. Ces données peuvent être caractérisées par leur structure ainsi que par leur sémantique. Ces sources d'adaptation sont prises en compte afin de modifier un certain nombre de cibles. Ces cibles d'adaptation constituent le deuxième axe de notre espace problème. Pour la réalité mixte nous avons identifié deux types de cibles. Le premier type correspond aux techniques d'interaction, pour lesquelles on va adapter la façon dont l'utilisateur interagir avec l'application. Ensuite la deuxième cible possible est la visualisation du contenu. En effet, la façon dont est présenté le contenu peut être modifiée, par exemple la manière dont il est disposé dans l'espace 3D ou encore l'aspect visuel de chacun des éléments affichés. Le troisième axe correspond au temps de l'adaptation. L'adaptation est dite statique si elle a lieu entre les sessions ou bien dynamique si des adaptations peuvent également intervenir à l'exécution. Enfin le quatrième axe correspond au contrôleur de l'adaptation. Le premier contrôleur possible est l'utilisateur lui-même qui peut configurer son application. Le deuxième contrôleur possible est le développeur ou déployeur de l'application qui peut configurer le système à distance ou via divers fichiers de configuration. Le troisième contrôleur possible est le système qui peut décider automatiquement des adaptations à effectuer grâce à une détection du contexte d'usage. Au milieu de ces quatre axes on retrouve les moyens d'adaptation qui font référence à la mise en oeuvre des modifications du système interactif. Tout d'abord, le remodelage consiste à modifier localement une application pour satisfaire un contexte d'usage local. Ensuite la redistribution consiste à modifier la distribution des composants d'un système sur plusieurs dimensions : affichage, plateforme et utilisateur. Nous avons ainsi pu extraire un ensemble de sept exigences que doit respecter notre solution logicielle pour le développement d'interfaces de réalité mixte. Ces exigences consistent à couvrir l'ensemble de l'espace problème que nous venons de décrire. Cette solution doit pouvoir prendre en compte l'ensemble des sources d'adaptations, pouvoir impacter l'ensemble des cibles, supporter la dynamicité, et supporter à la fois des adaptations initiées par l'utilisateur, par le développeur et par le système. Les deux moyens d'adaptations, le remodelage et la redistribution, doivent être supportés. Notre dernière exigence est également de proposer une solution facile à utiliser et également facile à étendre pour pouvoir couvrir les problématiques futures.

Adaptation aux dispositifs

Pour aider les développeurs à la création d'interfaces de réalité mixte plastiques, nous avons initié la création de modèles logiciels qui visent à couvrir les différentes exigences de la plasticité. Ces modèles sont intégrés dans une solution logicielle nommée 3DPlasticToolkit. Dans la première partie de cette thèse nous présentons les bases de cette solution, c'est-à-dire ses mécanismes d'adaptation pour le remodelage dynamique. La première partie se concentre sur la prise en compte les dispositifs comme sources d'adaptation. Ces périphériques peuvent être pris en compte afin d'impacter les techniques d'interaction et la visualisation du contenu de l'application. Jusqu'à récemment, la majorité des applications étaient développées pour une plateforme donnée. Aujourd'hui les usages ont évolué, les gens ont accès à une grande variété de plateformes matérielles différentes : ordinateurs personnels, dispositifs mobiles, plateformes immersives. De plus, à l'intérieur de ces catégories,

les périphériques possibles peuvent également être très différents. C'est pourquoi développer une version d'une application pour chacune des possibilités n'est plus une option viable.

Notre première contribution est un modèle pour décrire les dispositifs d'affichage et d'interaction utilisés pour les interfaces de réalité mixte. Ce modèle décrit au format UML peut être étendu et modifié par n'importe quel développeur. Ce modèle expose les données que peuvent restituer ou acquérir les dispositifs. De plus il ajoute des propriétés souvent absentes des taxonomies classiques comme leurs limitations et leurs représentations dans le mode réel. Ainsi, avec notre modèle, un dispositif peut être composés d'un certain nombre d'unités d'entrées, d'unités de sorties et d'objets physiques. Un outil graphique est fourni pour configurer ces dispositifs.

Ensuite nous proposons un nouveau modèle pour développer des composants applicatifs dérivé des modèles PAC[Coutaz, 1987] et ARCH[Arch, 1992]. Ces composants peuvent correspondre à des techniques d'interaction, des widgets 3D, des métaphores de visualisation de données, ou encore des effets visuels. Ce modèle permet de décrire des composants indépendants des dispositifs, indépendants du framework 3D utilisé et qui peuvent supporter différentes représentations. Ainsi, il va permettre d'impacter à la fois les techniques d'interaction et la visualisation du contenu. A l'exécution un composant applicatif est déployé pour chacune des tâches demandées par le développeur. Ces tâches, décrites avec un modèle de tâche classique, permettent de décrire à haut niveau le comportement et les possibilités de l'application.

Pour gérer le déploiement de ces composants en fonction des tâches et des dispositifs disponibles nous proposons un processus d'adaptation dynamique. Ce processus qui gère le remodelage de l'application va réagir aux changements de contexte à l'exécution afin d'ajouter, de supprimer et de modifier les composants applicatifs déployés pour chacune des tâches. Ce processus est basé sur des mécanismes de notation qui vont permettre de maximiser l'utilisabilité de l'application en fonction du contexte d'usage. Ce système de notation est totalement configurable par le développeur.

Avec 3DPlasticToolkit l'utilisateur peut également contrôler et modifier le processus d'adaptation à l'exécution. En effet, notre solution intègre une interface intégrée pour cela : la meta-interface utilisateur. Cette interface permet à l'utilisateur entre autre de remplacer et de modifier les composants applicatifs ou de remplacer un dispositif utilisé par un autre dispositif disponible.

Nous présentons également un modèle pour supporter la redistribution : D3PART ("Dynamic 3D Plastic And Redistribuable Technology"). Dans ce modèle, un processus de redistribution va permettre à l'utilisateur de configurer la distribution de son application via la meta-user interface. Grâce à cette configuration l'utilisateur va pouvoir choisir de migrer et de répliquer ces composants applicatifs entre plusieurs plateformes d'interaction. Cela va ainsi permettre à l'utilisateur de changer d'une plateforme à une autre sans couture via une migration totale, par exemple passer d'une plateforme immersive à un dispositif mobile pour continuer son travail en mobilité. Il va pouvoir combiner plusieurs plateformes grâce à une migration ou une réplication partielle, ou encore pouvoir créer un contexte collaboratif grâce à une réplication totale.

Nous présentons également différents démonstrateurs développés avec nos modèles qui nous permettent de démontrer comment notre ils peuvent être utilisés pour adapter les techniques d'interaction et la visualisation d'une application. Nous présentons ainsi un démonstrateur d'aménagement d'intérieur, un démonstrateur d'interactions collaboratives et également différents effets visuels développé pour régler le problème de "frame cancellation".

Adaptation aux utilisateurs

Nos modèles permettent ainsi de faire des choix de techniques d'interaction et de manières de présenter le contenu en fonction des dispositifs disponibles et des choix du développeur. Ainsi nous ne sommes pas assurés de fournir à l'utilisateur final une application qui va satisfaire ses besoins et ses préférences. La personnalisation a pour but d'adapter l'application en fonction de l'utilisateur. Nous avons ainsi proposé une extension de nos modèles pour considérer l'adaptation aux utilisateurs.

Afin d'identifier quels sont les exigences à respecter pour considérer l'adaptation aux utilisateurs nous avons réalisé une expérimentation préliminaire qui a comparé les préférences et performances de 51 utilisateurs pour trois techniques de sélection 3D sur deux dispositifs de réalité virtuelle différents. Avec cette expérience nous avons pu démontrer que les préférences pour les techniques d'interaction sont très variées. De plus nous avons également constaté que pour un même utili-

sateur la technique préférée ou la plus performante diffère souvent d'un dispositif à l'autre et ce malgré leurs similarités. Les résultats de cette expérimentation ont été exploités afin d'évaluer si un algorithme d'apprentissage supervisé est capable d'anticiper la technique adaptée pour un utilisateur en fonction de son profil ou en fonction des données d'un pré-test préalablement effectué. Les résultats obtenus avec cette approche sont encourageants mais pas encore suffisants afin de vraiment prouver son efficacité parfaite. Un jeu de données comportant plus de participants serait nécessaire pour réaliser plus d'investigations sur l'utilisation de cette méthode d'apprentissage. À partir de ces résultats nous avons proposés un modèle utilisateur qui permet d'inclure pour un utilisateur donné, les données du profil, les préférences, les informations sur l'environnement ou les données de suivi de cet utilisateur. Via l'association entre un utilisateur et chacune des tâches de haut niveau, le processus d'adaptation peut gérer le déploiement des composants applicatifs en fonction des préférences utilisateurs. De plus cette association permet ensuite aux composants applicatifs d'avoir accès aux informations sur l'utilisateur pour adapter leurs propriétés internes. Le déploiement de ces composants s'effectue également via notre processus d'adaptation qui utilise des mécanismes de notation. Ce système de notation prend ainsi compte les préférences de l'utilisateur afin de lui fournir l'application qui va lui correspondre aux mieux. Nous proposons également une intégration de notre méthode d'apprentissage pour calculer automatiquement les notes.

Adaptation aux données

La dernière partie de cette thèse s'intéresse à l'adaptation aux données. En effet, dans une interface de réalité mixte, les utilisateurs peuvent visualiser, modifier et manipuler des données. Ces données sont décrites par des propriétés sémantiques et structurelles qui peuvent avoir un impact sur la façon dont on va interagir avec elles et surtout sur la façon dont on va les présenter. Par exemple, des objets culturels pourraient être visualisés dans un musée virtuel tandis qu'un catalogue de vidéos à la demande pourraient être parcouru à l'intérieur d'un cinéma 3D.

Ainsi nous proposons un modèle complémentaire intégré à 3DPlasticToolkit afin de prendre en compte les données comme source d'adaptation. Pour l'instant l'implémentation du modèle est encore partielle. Dans ce modèle nous proposons tout d'abord la représentation des données sous la forme d'une ontologie gérée avec OWL ("Web Ontology Language"). Ce modèle permet le chargement de base de données dans différents formats et permet d'exposer à la fois la structure et la sémantique des données. Pour visualiser ces données nous donnons la possibilité à l'utilisateur final de créer des requêtes afin de sélectionner un sous ensemble données à visualiser. Ces requêtes peuvent être exprimées via différentes méthodes comme à l'aide d'un champ d'entrée textuel ou à l'aide de commandes vocales. Ensuite nous proposons une mise à jour de nos mécanismes de notation afin de choisir les métaphores de visualisation adaptées aux données qui ont été sélectionnées par l'utilisateur final. Cette notation est basée sur un processus d'optimisation réalisé pour chacune des métaphores compatibles avec la requête de l'utilisateur. Cette optimisation consiste à chercher les meilleurs paramètres de chaque métaphore en fonction de la structure et de la sémantique des données sélectionnées. Une notation par métaphore est ainsi calculée à partir des paramètres trouvés.

Afin d'illustrer ce modèle, nous proposons une première preuve de concept basée sur la visualisation de la base de données TOPIC-TOPOS. Cette base de données contient un ensemble d'éléments relatifs à l'héritage culturel. Le parcours de ces données peut permettre à un utilisateur d'en savoir plus sur certains éléments historiques ou alors de planifier certaines visites.

Conclusion et perspectives

En conclusion cette thèse propose différents modèles afin de créer des interfaces de réalité mixte plastiques. Ces modèles permettent de couvrir l'ensemble de l'espace de problèmes proposé. Ainsi ces modèles permettent de prendre en compte les périphériques, les utilisateurs et les données comme sources d'adaptation. Ces sources peuvent être prises en compte afin de modifier à la fois les techniques d'interaction et la visualisation du contenu. Les adaptations peuvent être statiques et dynamiques et contrôlés par l'utilisateur final, le développeur et le système. Nos modèles supportent le remodelage via un processsus d'adaptation local et la redistribution grâce à un processus intégré.

Ces modèles sont intégrés dans une solution logicielle nommée 3DPlasticToolkit. Cette solution intègre de nombreux périphériques de réalité mixte comme l'Oculus Rift, la Kinect, le HTC vive, le Razer Hydra, la zSpace, etc. De même elle propose également des techniques d'interaction et des métaphores de visualisation de données prêtes à l'usage. Des outils visuels de création permettent de configurer 3DPlasticToolkit. On peut donc dire que 3DPlasticToolkit est prêt à l'emploi pour un développeur qui veut créer une interface de réalité mixte tout en bénéficiant des avantages de la plasticité. Les applications présentées dans cette thèse qui ont été développées avec cette solution permettent de démontrer son efficacité.

La proposition de ces modèles et de cette solution logicielle offre également de nombreuses perspectives de travail. Il serait tout d'abord souhaitable de réaliser une évaluation formelle de notre solution auprès de développeur d'applications afin de mesurer son efficacité. Il serait également possible d'explorer la possibilité de déclencher automatiquement le mécanisme de redistribution en détectant de manière automatique les actions de l'utilisateur. Enfin il serait également intéressant d'adapter le niveau de détail du contenu 3D en fonction des performances des plateformes afin que l'expérience ne soit pas dégradée sur les plateformes avec des faibles puissances de calcul.

List of Acronyms

2D Two-Dimensional
3D Three-Dimensional
AR Augmented Reality
CAD Computer-Aided Design
DOF Degrees Of Freedom
FOR Field Of Regard
FOV Field Of View **GPS** Global Positioning System
HMD Head-Mounted Display
IR Infrared
MDE Model-Driven Engineering
MR Mixed Reality
OS Operating System
PC Personal Computer
SDK Software Development Kit
UML Unified Modeling Language
VE Virtual Environment
VR Virtual Reality
VRML Virtual Reality Modeling Language
WIMP Windows, Icons, Menus, Pointers
X3D eXtensible 3D
XML eXtensible Markup Language

Chapitre 1

Introduction

Context of the thesis

Today, there is a growing interest for Mixed Reality (MR) user interfaces. According to Milgram [Milgram and Kishino, 1994], it includes Augmented Reality (AR), Virtual Reality (VR) and Augmented Virtuality (AV) applications. This new trend can be explained in different ways. First, the impressive progresses made by the consumer electronic industry allow everyone to have access to very powerful GPUs and to high quality display devices and interaction devices. For instance, the timeline shown in Figure 1.1 gives an insight of the evolution of these peripherals. As shown, from the Sutherland's head mounted display (HMD) in 1968 [Sutherland, 1968] to the Oculus CV1 presented in 2015, the gap is significant but this revolution has been made in different steps. Second, today, compared to a few years ago, the creation of 3D applications is simplified. Indeed, game engines such as Unity3D or the Unreal Engine are now free-to-use. Such tools provide graphical tools and built-in components that allow anyone to create interactive 3D applications without knowing any concept of programing and any advanced 3D libraries such as OpenGL™ or DirectX®. Third and last, MR user interfaces have an important number of possible use cases. The most common one concerns entertainment, which includes applications such as video games. However, other professional domains also benefit from the use of MR user interfaces such as scientific data visualization, education, Computer-Aided Design (CAD), e-commerce, architecture, computer-assisted surgery, etc. Using 3D models can really improve attractiveness, interactivity, usability, and efficiency of such applications.

Even with the previously cited development tools, during the development of MR user interfaces, designers and developers are confronted to many issues. Indeed, the large amount of display and interaction devices is an opportunity but is also a problematics during the development of such applications. All devices do not have the same capacities. An application cannot be developed in the same way if it has to run with a mouse and a monitor or with an HMD and motion trackers. Moreover, the way to interact with MR user interfaces is wider than classic 2D user interfaces. Indeed, while classic 2D user interfaces are mostly based on the WIMP (Window, Icon, Mouse, Pointer) interaction paradigm or on tactile 2D interactions, MR user interfaces include a lot of interaction techniques that can be classified into three categories according to Hand [Hand, 1997]: selection/manipulation, navigation and application control. In each category a lot of interaction techniques exist and it can be difficult to choose one that will really fit the application needs, the target interaction devices, and the target users needs and skills. The third dimension included with MR user interfaces also increases the number of ways to present the content in comparison with 2D user interfaces. For instance, for data visualization applications, there exists a lot of visualization metaphors that can be chosen according to different criteria such as the output devices properties or the data intrinsic properties.

Developing manually a version of an application for each possible configuration is not a very flexible way toward adapting it to various features. The combinatorial complexity of such development is important and need to be reduced with specific development tools. In order to deal with such issues, this thesis is centered on plasticity for MR user interfaces. The plasticity concept comes from the field of 2D user interfaces [Thevenin and Coutaz, 1999]. It is the capacity of an interactive

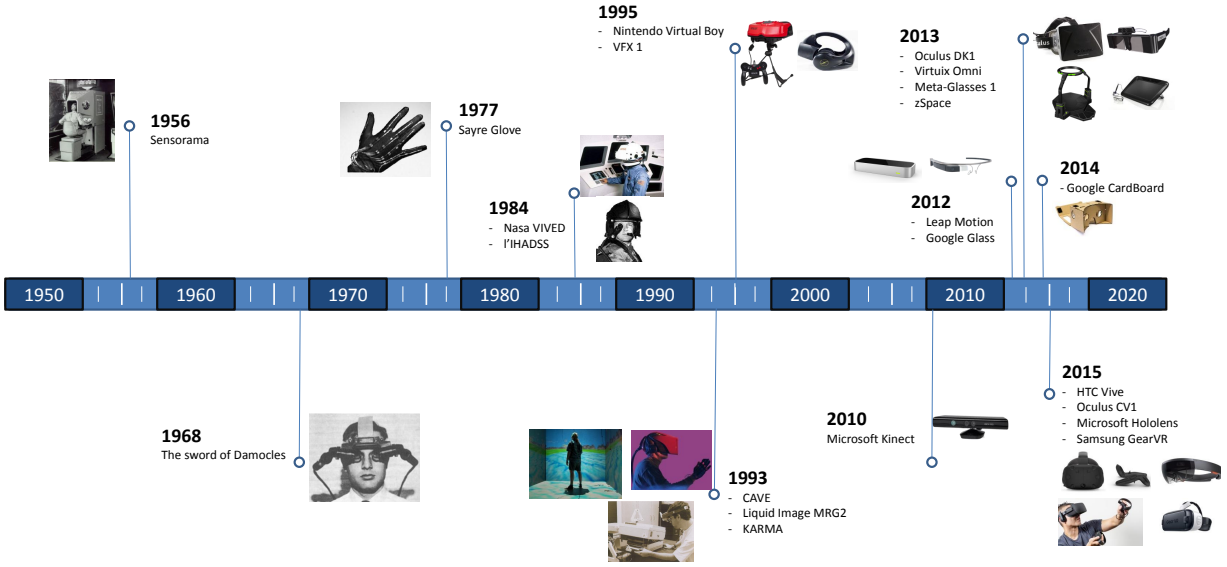


FIGURE 1.1 – A timeline that presents a non exhaustive list of VR and AR devices. Since 2010, there is an increasing number of this kind of devices that reach the consumer market.

system to withstand variations of both the system physical characteristics and the environment while preserving its usability. Code interoperability is a necessary condition but is not sufficient for an interactive system to be considered as plastic. Usability continuity has to be guaranteed too, performances and capabilities have to be at least constant. To do so, plastic interactive systems include an intelligent adaptation process that takes into account the context of use in order to configure the final application. More than just ensuring usability continuity for an application whatever the context of use, plasticity can also reduce maintenance and development times and costs. Plasticity can also help an application to be distributed widely and it can also improve the attractiveness of an application.

This thesis is realized in the context of the ImData project in the technological research institute b<>com. The goal of this project is to develop technologies to navigate in and naturally interact with large datasets. It focuses on the adaptation of the data presentation and of the interaction techniques according to the context of use and also on how to collaboratively interact with these data. Moreover, the project also aims to validate the different technical choices by evaluating the physiological and psychological acceptability of the different proposed solutions.

Objectives

From the assessments of the previous Section we can highlight the fact that software tools are needed by developers if they want to develop plastic MR user interfaces. The objective of this thesis consists in creating such tools. We want to propose new models and concepts for the development of plastic MR user interfaces. Our goal is to integrate these concepts and models into a software solution: a toolkit for the creation of plastic MR user interfaces named 3DPlasticToolkit. To do so, this thesis aims to complete the following objectives:

- We must identify the developers and designers needs and issues when they are confronted to the development of MR user interfaces. We have to deduce from this analysis a set of requirements that our solution has to cover.

- We must identify the current software solutions designed to deal with adaptive and plastic MR user interfaces. We must analyze their advantages and limitations and also demonstrate if they cover all the plasticity requirements for MR user interfaces or not.

- We must provide models and concepts that can cover the different requirements of plasticity and that can solve the limitations of current solutions. We must include these concepts and models in a software solution. This solution must handle the modeling of the context of use and must integrate an intelligent adaptation process that can take this context into account.

- We must create an easy to extend solution. As said before, today the MR domain is really dynamic and the provided solution must be able to handle the future issues and needs.

- We must provide an easy to use solution that can be seamlessly integrated in the developer's workflow. In order to be easy to use, graphical authoring tools must be developed. If possible, this aspect of our solution has to be evaluated.

- We must ensure reusability. Most of the components created for one target project must be possibly reused another application. Ensuring this reusability is the best way to simplify any development and to reduce development times and costs.

- We must help developers to focus on the content of their application and on the user experience. Technical issues induced by MR must be hidden.

Through the different Chapters of this thesis, according to the new concepts and models we introduce, we describe our new software models for the creation of plastic MR user interfaces. We propose 3DPlasticToolkit as an implementation of these models. In order to demonstrate that our solution can cover these different objectives and more specifically the plasticity requirements, we also present different applications that have been developed with 3DPlasticToolkit. As these applications cover different possible use cases of MR, it enables us to demonstrate that our solution can be viable for a lot of domains. Most of the contributions presented in this thesis address adaptation to devices. Indeed, we consider that it is one of the main issues that developers encounter today during the development of MR user interfaces. Nevertheless, this thesis and 3DPlasticToolkit also aims to target the other issues such as adaptation to users and to data.

Dissertation Organization

In this section, we have briefly discussed the context of MR user interfaces and the actual challenges that we address in this thesis.

Chapters 2 presents the design space for the development of MR user interfaces. It focuses on the devices issues as well as on the software ones that any developer has to take into account when developing such application. The goal of this section is to highlight the different adaptation sources and to identify adaptation targets in the field of MR user interfaces.

Chapters 3 focuses on plasticity especially for MR user interfaces. First, it defines the plasticity concept and gives close fields of research. Then we present different related work of 2D and MR user interfaces with adaptation capabilities. The goal of this Chapter is to determine the requirements that our software solution has to cover. We propose a problem space for the development of plastic MR user interfaces. Then the other chapters are chosen according to the identified adaptation sources.

Chapters 4 focuses on device adaptation. In this chapter we propose new models and the basis of our plasticity software toolkit: 3DPlasticToolkit. We present its adaptation process composed of a scoring algorithm and of an integrated user interface for the modification of the adaptation behaviour. This Chapter focuses on detailing how this toolkit can take into account devices as an adaptation source during the development of MR user interfaces.

Chapter 5 presents three different applications developed with 3DPlasticToolkit. It shows how devices can be taken into account with 3DPlasticToolkit in order to adapt the interaction techniques and the content visualization of a MR user interface. This Chapter also demonstrates how 3DPlasticToolkit can really benefit to the development of MR user interfaces with concrete examples.

Chapters 6 focuses on a specific adaptation means possibly impacted by plasticity: Redistribution. Redistribution consists in changing the components distribution of an interactive system

across different dimensions such as platform, display and user. For instance, for a given application, it can make it possible to switch from an interaction platform to another one, to combine these platforms or to create a collaborative context of use. Our contribution is a redistribution process that provides redistribution capabilities to any application developed with our models.

Chapter 7 focuses on User adaptation. We present how any user can be modeled into 3DPlasticToolkit and how this model is taken into account during the adaptation process. A preliminary study has been performed in order to understand the user's adaptation needs for MR user interfaces. The result of this study has also been exploited in order to explore the use of machine learning for the creation of adaptation rules.

Chapter 8 presents how we handle data as an adaptation source. We show how data semantic and data structure is modeled into 3DPlasticToolkit and how it can be taken into account by the toolkit adaptation process. We demonstrate how this feature can be used to visualize a large amount of patrimonial data.

Chapter 9 concludes this thesis an gives some opportunities for future work.

Chapitre 2

Design Space of Mixed Reality User Interfaces

2.1 Introduction

In this chapter we identify the design space of Mixed Reality (MR) user interfaces in order to determine and highlight the different aspects that characterize this domain. We also want to identify the developers needs for the development of MR user interfaces and the limitations of current MR development tools.

With a critical analysis of this design space we aim to demonstrate its complexity and therefore the needs for plasticity in the domain of MR user interfaces. The goal is also to identify a part of the requirements that our software solution has to cover if we really want to meet the developer's requirements.

This Chapter is structured as follows, first, in Section 2.2, we define the different types of MR user interfaces and present some use cases. Second, in Section 2.3, we present the main components involved in the development of MR user interfaces. Next, in Section 2.4, we describe some common tools used to develop MR user interfaces. Last, in Section 2.5, we try to analyze the design space of MR user interfaces in order to explain the needs for plasticity.

2.2 Mixed reality User Interfaces

In 1994, Milgram [Milgram and Kishino, 1994] defines the *virtuality continuum* that establishes a continuity from the real world to the virtual world. This continuum is shown in Figure 2.1. Mixed Reality (MR) refers to anywhere between the extrema of this continuum. According to Bowman et al. [Bowman et al., 2004], the position of an environment on the continuum indicates its level of virtuality from "purely virtual" to "purely real". This position may move during the execution of an application as the user interacts with his environment. Therefore, a MR environment is defined as a place that includes real world and/or virtual world objects. It refers to Augmented Reality (AR), Augmented Virtuality (AV) and Virtual Reality (VR) applications, which are detailed in the following sections.

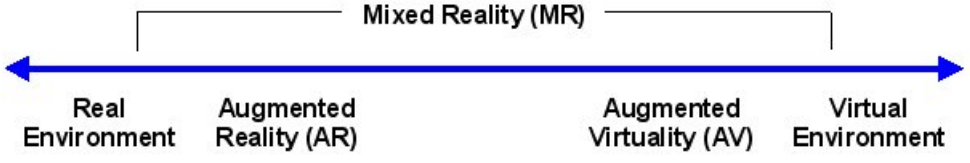


FIGURE 2.1 – The *virtuality continuum* presented by Milgram [Milgram and Kishino, 1994]. A Mixed Reality environment includes real world objects and/or virtual ones. It includes, AR, AV and VR applications.

MR user interfaces are interactive, therefore this field of research is part of the Human-Computer-Interaction (HCI) field of research. In an interactive system, the users' actions are taken into account in order to control an application that provides in exchange multiple feedbacks. Contrary to 2D user interfaces, for MR user interfaces, interactions are performed in a spatial context. In that case we talk about 3D interactions as explained in Section 2.3. Dubois et al. [Dubois et al., 1999] define two possible targets for the user's task in a MR user interface: real world and computer. They respectively correspond to the two terms presented by Milgram [Milgram and Kishino, 1994]: AR and AV. According to Dubois et al. [Dubois et al., 1999], in AR interaction with the real world is augmented by the computer and in AV interaction with the computer is augmented by objects and actions in the real world. Dubois et al. also define two types of augmentation: augmented execution refers to the enhancement of the actions that the user can perform and augmented evaluation refers to the enhancement of the perception. A complementary classification of HCI styles is also proposed by Rekimoto and Nagao [Rekimoto and Nagao, 1995]. First, in a desktop computer, interaction between the user and the computer is isolated from the interaction between the user and the real world. Second in VR, the computer fully surrounds the user and interaction between the user and the real world disappears. Third, in ubiquitous computers environment, the user interacts with the real world and can also interact with computers embodied in the real world. Last, augmented interaction supports the user's interaction with the real world using computer augmented information.

The goal of MR user interfaces is to blur the border between the real world and the virtual world. It gives new opportunities to a lot of application fields. Indeed, it gives developers and designers the possibility to create scenarios that could not be possible in the real world. For instance, imaginary worlds can be created, meaningful data can be displayed upon the real world, a user can be transported to another age or to another place, multiple users from all over the world can be gathered in a same virtual space, etc., the possibilities are limitless.

The specificities of each part of the *virtuality continuum* are detailed in the following Sections.

2.2.1 Virtual Reality

Definition

Virtual Environment (VE) refers to one extrema of the *virtuality continum* defined by Milgram [Milgram and Kishino, 1994] and shown in Figure 2.1. A VE is a computed generated simulation of an artificial world. Virtual Reality (VR) includes applications that consist in immersing one or several users into a VE with which they can interact. When the VE is shared by multiple users, it refers to Collaborative Virtual Environment (CVE).

In order to define Virtual Reality we have chosen the technical definition provided by Fuchs et al. [Fuchs et al., 2011] [Fuchs et al., 2003]:

"Virtual Reality is a scientific and technical domain that uses computer science and behavioural interfaces to simulate in a virtual world the behaviour of 3D entities which interact in real time with each other and with one or more users in pseudo-natural immersion via sensorimotor channels"

From this definition they derive the "perception, cognition, action" loop that is involved in VR. First, the user interacts with a VE through motor interfaces that capture his actions suchs as gestures movements or voices. Then, the simulation takes into account these interactions and modify the VE accordingly. Last, with these modifications, sensorial reactions are transfered to the sensorial interfaces. In most cases, a VR application is multi-sensory as the VE is presented to the end-user through multiple sensory channels. For instance, it can target vision, sound, touch, smell or taste sensory channels. The goal of VR is to provide end-users with a believable experience. Therefore, the three steps of the loop have to be optimized in order to improve the end-user experience. Quality of interaction between objects and users must be ensured. Moreover, the behaviors, appearance and the provided sensorial outputs must be convincing enough. A VE is synthetic as it is created at the time of the simulation and not replayed from a pre-recorded presentation.

In order to define the basic characteristics of VR, Burdea et al. [Burdea and Coiffet, 2003] present the three I's: Immersion, Interaction and Imagination. This is the VR triangle presented

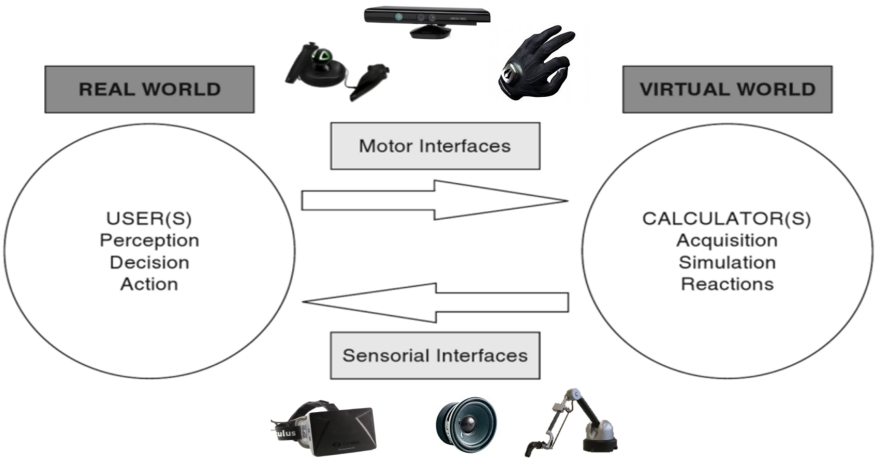


FIGURE 2.2 – The "perception, cognition, action" loop involved in Virtual Reality described by Fuchs et al. [Fuchs et al., 2011]

in Figure 2.3. First, according to Mestre [Mestre et al., 2006] immersion is achieved by removing as many real world sensations as possible and replacing them with sensorial feedbacks provided by the VE. Therefore the level of immersion is directly linked to the fidelity of these feedbacks that depends on the computed simulation and on the output devices used. Presence is a psychological, perceptual and cognitive consequence of immersion. It corresponds to the feeling of "being there", "existing in" perceived by the user. Second, interaction enables the end-user to change the state of the VE in real time. Virtual objects and characters within the VE can react to user's actions and interactively communicate with them. Third and last, imagination is needed to design a VE in order to make it credible and adapted to the target use case. This imagination also enables a designer to create non-realistic worlds such as a world with different physics laws.

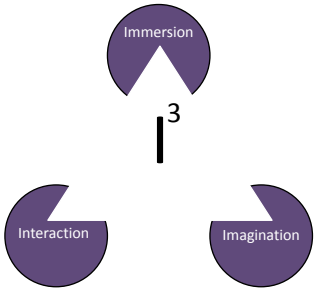


FIGURE 2.3 – The three I's of Virtual Reality [Burdea and Coiffet, 2003]: Immersion, Interaction and Imagination.

Use-cases

In the academic literature and in the industry, we can find a wide variety of possible use-cases of Virtual-Reality. Moreover, today VR is a very trendy domain and many uses-cases still appear. Indeed, the number of possible applications and different virtual worlds is only limited by the human imagination. From this statement, we can deduce that lots of fields can benefit from VR. Here, we have selected some of the most relevant applications domains of VR.

First, the field of **video games** is the most common use case of VE and certainly the most important creator of 3D content. The field of video games can really benefit from VR. Indeed, it can increase the immersion feeling of the user and make him feel that he is not just playing a game but that he is part of this game with the sensation of presence. Video games can be categorized into different types such as:

- **Action**: it includes games such as first-person shooters (FPS), fighting games or platform games. It mainly relies on the player's dexterity and reflexes.
- **Strategy**: it includes games such as real-time strategy and turn-based strategy. It requires skillful thinking in order to achieve victory.
- **Role Playing**: in this kind of games the player embodies an adventurer who has skills that can be improved. Massively multiplayer online role-playing game (MMORPGs) is one example of Role Playing game sub-category.
- **Sports**: it includes sports simulations such as car racing, tennis or football games.

Today, most of video games use classic interactions based on gamepad or keyboard and mouse. However, some recent games tend to use 3D interactions with gestural interfaces. On the consumer market, a first try was made with the Playstation™EyeToy® and more recently with the Nintendo® Wiimotes as shown in Figure 2.4a and with the Microsoft® Kinect™.

(a) (b)

FIGURE 2.4 – Two examples of video games with 3D interaction capabilities.(a) Wii tennis proposes 3D interactions with the accelerometer integrated in the Nintendo® Wiimote. (b) Eve Valkyrie is an immersive spacecraft shooter game that will be available on future VR devices such as the Oculus Rift, the Playstation™VR or the HTC vive.

With the democratization of VR devices such as the Oculus Rift or the Playstation VR, a lot of immersive VR games are in development such as Eve Valkyrie[1], a spacecraft fighting game which is shown in Figure 2.4b or The Climb[2] that is rock climbing simulation.

The second field concerns **education and training**. According to Cobb et al. [Cobb et al., 2008], with VE, designers and developers can recreate scenario that would not be possible in a real situation for cost, risk and logistic issues. Using VR for formation and training can familiarize a trainee to these particular scenarios an can give him a similar experience as if he was confronted to the real situation. With this kind of learning in VE, the goal is to improve the user skills when he will be in the real working environment. One of the other advantages of using VR simulation is that the activity and the progresses of the user can be tracked in order to provide him adapted feedbacks.

In that category, Rickels et al. [Rickel and Johnson, 1999] propose a virtual agent to help a user to inspect a high pressure air compressor aboard a ship. In the context of archaeological research, smith et al. [Smith et al., 2013] present a stereoscopic research and training environment for archaeologists called ArtifactVis2. It makes possible the management and visualization of cultural

[1] https://www.evevalkyrie.com
[2] http://www.theclimbgame.com

datasets within a collaborative virtual 3D system. This work is also an example of immersive data-visualization, which is the next use case of VE that we present. Multiple trainees with different roles can also be integrated in the same learning scenario in a Collaborative Virtual Environment (CVE). In the context of medicine, this is the example given by Claude et al. [Claude et al., 2015] who propose a CVE for learning neurosurgery procedures. This example is illustrated in Figure 2.5.

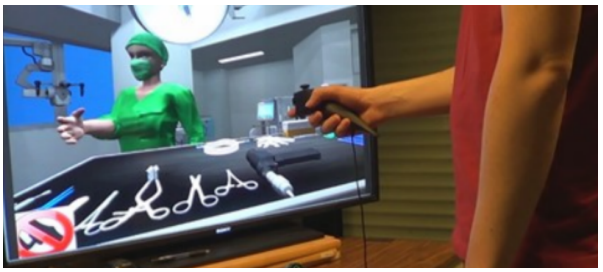


FIGURE 2.5 – A collaborative Virtual Environment for learning neurosurgery procedures proposed by Claude et al. [Claude et al., 2015]

Third, the field of **data-visualization** proposes to immerse one or several users into a VE for analyzing and annotating data. The third dimension provided by VR for the visualization and interactions can improve the user's perception and understanding of the data. It could help him to detect some aspects that would not have been visible in 2D. Moreover, by integrating multiple users into a CVE, we can make several remote users to visualize and interact with the same dataset synchronously as proposed by Fleury et al [Fleury et al., 2012] and illustrated in Figure 2.6.

Regarding the type of data that can be visualized with VR, the most common one concerns scientific data. For instance, Cosmic explorer [Song and Norman, 1993] is a VR applications for investigating the cosmic structure formation. In [Brooks Jr et al., 1990], Brooks et al use VR to visualize and interact with protein molecules. In the same way, Bryson and Levit [Bryson and Levit, 1991] propose the Virtual Windtunnel, a VE for the visualization of three-dimensional unsteady fluid flows. However, immersive data visualization is not limited to scientifc data. Indeed, other kinds of data can also be visualized. For instance Esnault et al. [Esnault et al., 2010] propose to use VR to visualize a Video On Demand (VOD) database. As well, Bonis et al. [Bonis et al., 2009] create personalized VEs for visualizing museum data.

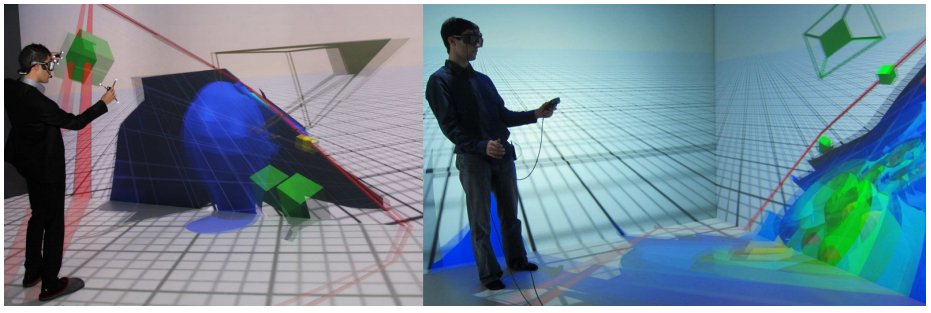


FIGURE 2.6 – A collaboration between two distant users for visualizing and interacting with scientific data presented by Fleury et al. [Fleury et al., 2012]. Here, the two users manipulate together a clipping plane.

2.2.2 Augmented Reality

Definition

With VR, the user is totally immersed into the VE and cannot see the real world around him. In contrast, AR consists in integrating computer-generated virtual objects into the real word [Azuma,

1997]. AR is the first step between real environment and VE on the virtuality continuum presented by Milgram [Milgram and Kishino, 1994]. The virtual objects can be superimposed upon, such as illustrated in Figure 2.7a, or composited, such as illustrated in Figure 2.7b with the real world. AR is interactive in Real-Time. The goals of AR is to display objects and information that the user cannot directly detect with his own senses [Azuma, 1997]. More than just being displayed, these objects and information can also be interactive and manipulable. It can help a user to have a better understanding of his surroundings or to perform better a real-world task. In order to precisely align the real world and the virtual objects, AR systems use calibration and registration processes [Zhou et al., 2008]. Indeed, according to State et al. [Hirota et al., 1996] a virtual object should appear at its proper place in the real world. However, the user cannot correctly determine spatial relationships between real objects and virtual ones. Dynamic registration is needed when the user moves around the real world. The relative position between real and synthetic objects must be constant. In addition, AR rendering is also challenging as the virtual objects must be integrated in a proper way to be perceived correctly. This rendering includes algorithms that compute the lighting and the shadows of virtual objects according the real environment and also that compute eventual occlusion between real and virtual content.

(a) (b)

FIGURE 2.7 – Two examples of AR applications. These two examples mix a real world with virtual information (a) Contextual virtual information is superimposed over the real world with Junaio AR View. (b) The virtual world is composited with the real world. Here a virtual "castle" is displayed on the table.

Use-cases

AR can also be found in a wide variety of domains. Some of them are common with VR. For instance, AR can also be used for gaming and training. We can also cite other fields where AR is used. A lot of these use cases are given by Krevelen et al. [Van Krevelen and Poelman, 2010].

AR can be used by industrials for **assembly and maintenance**. Indeed, AR can be used to display to an operator different informations about an assembly or a maintenance procedure. Using AR can reduce the time and the cost of these procedures and can also avoid mistakes.

For instance, in the automotive industry, BMW experimented AR to improve welding processes on their cars [Echtler et al., 2004] and Volkswagen [Pentenrieder et al., 2007] uses AR for factory design and planning. In the same way, AR is also used in the aeronautics industry such the Boeingwire bundle assembly project [Mizell, 2001] that aims at using AR to guide the assembly of electrical wire bundles. In the energetic energy industry, Klinker et al. [Stricker et al., 2001] show how AR is used by Framatome ANP for the inspection of power plants.

In the same way AR can be used for assistance in the field of **medicine**. In this field, most approaches consist in overlaying the patient with data from medical imagery. It requires real-time in-situ visualization of co-registered heterogeneous data [Mekni and Lemieux, 2014]. For instance, Vogt et al. [Vogt et al., 2006] overlay MR scans on heads and also display a view of the hidden parts of a manipulation tool. To continue, Haouchine et al. [Haouchine et al., 2013] overlay pre-operative

CT scans in order to show vascular networks during hepatic surgery as shown in Figure 2.8. In the same way, Navab et al. [Navab et al., 1999] also display x-ray imagery on top of real objects.

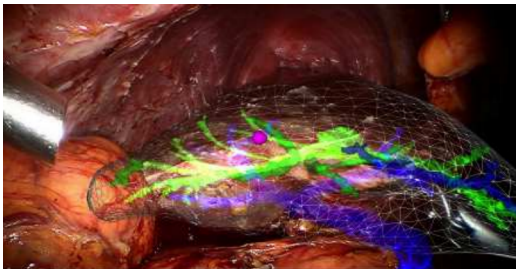


FIGURE 2.8 – An example of AR system for hepatic surgery proposed by Haouchine et al. [Haouchine et al., 2013]. Here, it shows a superimposition of a 3D real-time biomechanical model onto a human liver. This virtual model has been previously acquired through CT scans.

To continue, AR has a lot of **military** applications. Indeed, AR can help soldiers in particular battlefield environments. Some examples of concrete applications are given by Livingston [Livingston et al., 2011] et al. It can be used to improve target acquisition, to help soldiers in understanding their environment and tracking the current battle state (situation awareness). For instance, in [Piekarski et al., 1999] Piekarski et al. propose a collaboration between a VR system and soldiers using an AR system in order to provide them situation awareness. Some works also focus on displaying only the important information to the soldiers in order to not overload their view such as the information filtering algorithm proposed by Julier et al. [Julier et al., 2002].

2.2.3 Augmented Virtuality

In AR, most of the user's view is dedicated to the real world. Virtual Objects are present to enhance this real environment. On the contrary, for Augmented Virtuality (AV) most of the view is dedicated to the VE and some objects from the real world can be inserted in it.

Most use cases of AV concern telepresence. For instance, it includes applications where a distant user can be added in the VE through a window that displays his webcam view such as in [Regenbrecht et al., 2004] or through a 3D reconstructed avatar such as presented by Vasudevan et al. [Vasudevan et al., 2010] and shown in Figure 2.9a. Other applications consist in making the user be aware of his body such as his hands or aware of his close environment such as obstacles or other people in the room. This last case is the one presented by Never Blind In VR [Nahon et al., 2015] and illustrated in Figure 2.9b.

2.2.4 Analysis

VR, AR and AV share a lot of problematics and possible use cases. Each of these domains relies on immersion and interaction with 3D content.

We have seen that the user experience in VR mainly relies on the feeling of presence which can easily disappear if the simulation is not realistic enough. For AR applications, the coherence between the VE and the real world mainly depends on the registration and on the quality of the virtual objects display. AV mixes the different issues encountered in AR and VR.

In addition, the level of virtuality of a given application is not always fixed. For instance, an application could use VR to prototype a virtual object and then propose to see this virtual object in the real world with AR.

Whatever the use-case, these issues are the same so re-usability of software components can be very important for the development of such applications.

To conclude, we can say that a software solution can target these three domains if it can ensure a good interaction with a realistic VE or with virtual objects correctly displayed in the real world. Nevertheless, such a solution must also be able to take into account each particular domain specificities.

(a) (b)

FIGURE 2.9 – Two examples of AV applications. These two examples present how real objects can be injected into a VE. (a) Vasudevan et al. [Vasudevan et al., 2010] represent a 3D reconstruction of the upper bodies of two distant users into a shared VE. (b) "Never Blind VR" [Nahon et al., 2015] represents in the VE some elements from the real world such as close obstacles or other people in the room.

2.3 Interacting with MR user interfaces

The "perception, cognition, action" loop described in Section 2.2 represents the different components needed when developing a MR user interface. In this section we describe these different components.

We demonstrate here that each step of this loop implies a lot of attentions for developers. Indeed, we show here that MR users interfaces can be used with a lot of different display devices and input devices and can involve a lot of interaction techniques. We also describe that even if devices and interaction techniques can be grouped into categories, in each category their properties differ a lot. From these descriptions, we aim to show that developers need software tools that allow them to adapt their applications to this wide variety of devices and interaction techniques.

2.3.1 Input devices

The input devices are part of the last step involved in the "perception, cognition, action" loop. These devices enable the user to communicate with the application. It integrates all peripherals that sense and collect the user data and actions that can have an impact on the VE. According to Ohlenburg et al. [Ohlenburg et al., 2007], an input device is a combination of an hardware component and a software component, sending data into the system, based upon input from reality. The software component may contain a driver, a library, or a software development kit (SDK). For instance, a sensed data can be the positions of the user's hands and a possible action could be the detection of a particular gesture. Input devices can be categorized according to three main points:

- the sensory modality acquired by the device, for instance, a sound, a position, an image, etc.
- the number of DOF of the acquired data, for example, a 3D position or a 2D mouse motion,
- the frequency of the acquisition as some input devices provide continuous data (position, sound) while others provide discrete data (key presses, punctual gesture).

A complete classification for the categorization of input devices is given in Section 4.2.1. Here, we describe the main categories of input devices that are commonly used to interact with VEs.

Desktop and Mobile devices

Desktop and mobile input devices are used for classical 2D user interfaces and also for interactive games. Even if they do not provide data in a 3D space they can also be used in different ways for MR user interfaces. They are often used for 3D applications because they are inexpensive, and are natively supported by most desktop computing platforms [Lindt, 2009]. In this category, the

classical combination 2D Mice/Trackball/Trackpad and Keyboard is used in Desktop environment as the basis of the WIMP (Windows, ICON, Menus and Pointers) interaction paradigm. Then, touch sensing input devices are now commonly used with tablets and smartphones. These devices sense the user's touches. It can include direct touch, for instance with the user's fingers, and also indirect touch when a stylus is used. To continue, game controllers are mainly used for video games especially on consoles. Most of the time game controllers are composed of buttons that give binary discrete values, of triggers that give continuous values between 0 (not pressed) and 1 (totally pressed), and of joysticks that give a 2-DOF continuous positions between [-1, -1] and [1, 1]. These three types of inputs are shown on the Xbox360™ controller illustrated in Figure 2.10. Some recent game controllers also tend to provide more complex inputs such as the tactile surface integrated in the PS4® gamepad or the rotation tracking system integrated in the Wii™ remote plus.

FIGURE 2.10 – The Xbox360™ gamepad is a classic game controller composed of joysticks, triggers and simple buttons. This kind of device is commonly used for playing video games.

Tracking devices

Tracking devices are certainly the most important input for MR user interfaces. They are responsible to continuously detect the position and/or rotation of human body parts or of physical objects in the real world in order to maintain consistency between the real and the virtual world [Rolland et al., 2001]. They can be used in many different ways for MR applications such as interacting with the VE as described in Section 2.3, computing a user-centric stereoscopy or for 3D registration in AR applications. As explained in Section 2.3, tracking devices are frequently used in MR user interfaces because they support the realization of immersive interaction techniques [Lindt, 2009] that increase the user's impression that he is part of the VE. According to Bowman et al. [Bowman et al., 2004], the main characteristics of tracking systems are:

- the range, which refers to the limits between which variation of the acquired data is possible,
- the latency, which refers to the delay between a position/rotation is in a given state and when this state is reported,
- the jitter, which refers to the noise or instability of the acquired data,
- the accuracy, which refers to the precision of the acquired data. How much it is closed to the real value.

Tracking systems can also be categorized according to the type of data they can sense. For example tracking systems exist for gaze estimation, body-parts localization and for objects detection. We

can cite different kinds of tracking technologies such as optical, inertial, GPS (Global Positioning System), magnetic, acoustic or mechanical. Another kind of technology concerns datagloves that are wearable devices used to track the user's hands motions. One common example of optical tracking systems used for motion capture and MR applications is given in figure 2.11 for a VR application proposed by Artanim[3]. This system combines multiples VICON[4] infrared cameras. The infrared light emitted by the cameras are reflected on passive marker patterns, then the system is able to couple the different images acquired by the cameras in order to compute the position and rotation of each pattern. More information about how these different tracking systems work is given in [Rolland et al., 2001], [Foxlin et al., 2002] and [Bowman et al., 2004].

FIGURE 2.11 – A tracking system that combines multiple infrared VICON cameras (in red) in order to track reflective marker patterns. By increasing the number of cameras used, the tracking zone (range) can be enlarged. Here, for a VR application, the system is used in order to track the user's body parts such as his hands and his head.

Depth sensors

Depth sensors acquire 3 dimensional imaging with a real-time compatible frame rate. For a given frame, such sensors deliver an image where each pixel corresponds to the distance of the corresponding object to the sensor. A point cloud of the scene can be extracted from this depth image. According to Castaneda and Navab [Castaneda and Navab, 2011], depth sensors can be classified into multiple categories according to the technology used to acquire the depth data. In the context of MR user interfaces, the most commonly used types of depth sensors are the following ones: stereo-based depth sensors, structured-light-based depth sensors and Time-of-Flight sensors. First, some depth sensors are composed of two cameras and use a triangulation algorithm to reconstruct a depth images from disparity between the two views. Other systems use structured light cameras to project a light pattern into the 3D scene, thus the distortion of this pattern allows the computation of the 3D scene structure. An example of such a sensor if the first version of the Microsoft® Kinect™that is illustrated in Figure 2.12a. An example of point cloud acquired by this sensor is given in Figure 2.12b. Last, Time-of-Flight (ToF) sensors measure the depth of a scene by quantifying the changes that an emitted light signal encounters when it bounces back from objects in a scene [Castaneda and Navab, 2011]. Such a technology is used by the second version of the Kinect™.

Depth sensors can have multiple applications in the field of MR user interfaces. For instance, in the previously cited example Blind In VR [Nahon et al., 2015], a depth sensor was used to reconstruct the user body and his surrounding into a VE. Depth images can also be interpreted by the system in order to give other input data to the application. Indeed, for instance, computer vision algorithms can compute body parts positions and orientations from depth images [Shotton et al., 2013], as well as gaze information [Mora and Odobez, 2012]. It can also be used for 3D registration in AR applications [Henry et al., 2012].

Direct Human Input

Direct human input devices directly acquire the signals emitted by the human body [Bowman

[3] http://www.artanim.ch
[4] http://www.vicon.com

(a) (b)

FIGURE 2.12 – (a) The Microsoft® Kinect™is an example of depth sensor that uses a structured light approach. (b) The point cloud provided by this sensor crosses the depth image with the RGB one.

et al., 2004]. They can be classified into multiple types: speech input devices, brain computer interfaces and bioelectric input devices.

First, speech input devices are based on speech recognition. Weinschenk et al. [Weinschenk and Barker, 2000] define speech recognition as the technologies that enable computers or other electronic systems to identify the sound of a human voice, separate that sound from noise in the environment, and use the messages from the voice as input for controlling the system. Speech input devices are classically composed of a microphone and of a software component that detects what the user says and converts it to application commands. Speech inputs can be used for different purposes when interacting with VEs. For instance, it can be used for application control to display a menu or changing an application parameter such as in [De Sa and Zachmann, 1999]. Speech input can also be used jointly with another input device, in that case we talk about multimodal interactions. This kind of interaction is more detailed in Section 2.3.3.

Second, brain computer interfaces (BCI) consist in recording input commands directly from the signals generated by the human brain activity. The raw signals of the brain are acquired with one or multiple electrodes and are then converted into software commands through signal processing algorithms. According to Bowman et al. [Bowman et al., 2004] BCIs hardware solutions can be noninvasive when the user only wears a headband or a cap with integrated electrodes. A most performing but really invasive approach consists in surgically implanting the electrodes into the motor cortex. OpenViBE [Renard et al., 2010] is an example of software solution which enables to design, test and use BCIs. BCIs were initially used to develop communication and control technologies for people with severe neuromuscular disabilities [Wolpaw et al., 2002]. However, recently a lot of approaches tend to use BCIs in order to interact with VEs [Lotte et al., 2013]. For instance, it can be used for object manipulation such as the brain-based spaceship control proposed by Lotte et al. [Lotte et al., 2008]. An example of application of BCI is given in Figure 2.13b: the mind mirror is an AR application that allows a user to see his brain activity.

Last, bioelectric input is a generalization of BCIs for the entire body. Such devices can record different electrical signal activities from the human body. For instance, the NASA uses such devices to read muscle nerve signals from the forearm in order to control a virtual aircraft [Jorgensen et al., 2000]. In the same way, the Myo[5], illustrated in Figure 2.13a, is an armband that can detect gestures and motions through bioelectric inputs.

2.3.2 Output devices

The output devices (or display devices) are involved in the first step of the "perception, cognition, action" loop. They are responsible to provide the user with the sensorimotor feedbacks computed by the system. According to Ohlenburg et al. [Ohlenburg et al., 2007], an output device is able to represent or to emit information. An output device receives data from the system and affects

[5]https://www.myo.com

(a) (b)

FIGURE 2.13 – Two examples of direct input devices. (a) The Myo armband acquires the forearm bioelectric signals in order to detect gestures and motions. (b) An example of EEG cap used for an AR application: the mind mirror [Mercier-Ganady et al., 2014] allows a user to see his brain activity.

its output to reality. In the same way as input devices, an output device is a combination of an hardware and of a software component. According to Bowman et al. [Bowman et al., 2004], output devices present information to one or more of the user's senses through the human perceptual system. Therefore, they can be categorized according to the human sense they can stimulate:

- visual perception is the capacity to see his surroundings,
- auditory perception is the capacity to perceive sounds,
- haptic perception is the capacity to perceive forces and touches,
- taste is the sensory impression given by food or other substances on the tongue,
- olfaction is the capacity to sense the presence of smell.

In this Section we describe some of the main devices in each category and we give some of their characteristics.

Visual display devices

Visual display devices are the most common output devices used for MR user interfaces. They are responsible to transform a VE rendered by the system into a form perceptible by the Human Visual System (HVS). The goal of these displays is to make the end-user perceive the virtual world in 3 dimensions by stimulating his depth perception. The HVS perceives depth thanks to a variety of depth cues. These cues are separated into two categories. The first one contains monocular cues that are interpreted by each eye. For example, in this category we can find motion parallax, perspective, occlusion or texture gradient. The second category includes binocular cues that require both eyes to be perceived. Stereopsis and convergence are such cues. Visual display devices have multiple characteristics that can impact the level of immersion. These characteristics are given in [Bowman and McMahan, 2007]:

- the Field of View (FOV) is the size of the visual field in degrees that can be seen instantaneously on a display. The FOV of the human visual system is close to 160°,
- the Field of Regard (FOR) is measured in degrees of visual angle. The FOR is the amount of the physical space surrounding the user in which visual images are displayed [Bowman et al., 2004],
- from the display size and resolution we extract the spatial resolution, [Bowman et al., 2004], a measure of quality of the display expressed in dots per inch (dpi),

- stereoscopy provides an additional depth cue to the human visual system: stereopsis. Stereopsis, allows depth perception with the ability of our visual system to merge the two different images acquired by the two eyes,
- with head-based-rendering the rendering system considers the user's head position and rotation in order to create a consistency between the virtual and the real world,
- the refresh rate and the frame rate are expressed in hertz (refreshes per second). The frame rate refers to speed with which images are computed by the system and placed into the frame buffer. The refresh rate is the speed of the visual display device to refresh to display from the frame buffer.

Among the different kinds of visual display devices used for MR user interfaces we can cite simple screens, such as PC monitors, televisions or smartphones and tablet screens. These are the most commons screens used for 3D applications as their prices are relatively low. Some of these screens can be stereoscopic with or without the use of additional glasses. However, because of their small FOR, simple stereoscopic screens do not provide a good level of immersion. They are often used for computer games, Computer Aided Design (CAD) and desktop VR applications. When these screens are hand-worn such as tablets and smartphones they can also be used for AR applications. Then, multiple display systems use multiple screens in order to surround the user and provide an important FOR and an important spatial resolution for immersive VR applications. They are commonly combined with a head tracking system and provide stereopsis. In that category we can for example cite the CAVEs [Cruz-Neira et al., 1992] and Holostages. To continue, Head-Mounted-Displays (HMD) are recently the most commonly used display devices for immersive VR applications. A HMD places the images directly in front of the user's eye. It is composed of one or multiple screens, of lenses and of a tracking system. VR HMDs are closed-view, they totally cover the user eyes in order to hide the real world and to provide an important immersion. It provides stereopsis and a 360°FOR. However, for now HMDs suffer from a low FOV (less than 100°horizontal) and sometimes users experience something called the "tunnel vision" effect which decreases the immersion feeling. Even if some progress have been made recently the resolutions of HMDs is also a critical issue. An example of HMD is given in Figure 2.14a, the Oculus Rift DK2[6]. In addition, see through HMDs used for AR applications let the user see the real world with virtual objects superimposed by optical or video technologies [Azuma, 1997]. Optical see-through HMDs places optical combiners in front of the user's eyes. These combiners are partially transmissive, so that the user can look directly through them to see the real world and also partially reflective in order to display virtual images. On the contrary, video see through HMDs use a classical closed view HMD and display as a background (behind the virtual content) the images captured by two cameras placed in front of the headset (the real scene). For instance, the Epson Moverio BT-300[7] shown in Figure 2.14b is an example of optical see through display. In the same way the Oculus Rift shown in Figure 2.14a can be transformed into a video see through HMD with the addition of the OVRvision system[8]. Other kinds of visual output devices can also be cited such as workbenches, retinal displays and volumetric displays.

Sound output devices

Sound output devices or auditory displays render the sounds generated by the system and that come from the VE. In some cases, in order to provide a realistic simulation, sounds can be spatialized. In that case, we talk of localization, which is the psychoacoustic process of determining the direction from which a sound comes [Sherman and Craig, 2002]. Localization cues allow the end-user to determine the direction and the distance of a sound source. These depth cues are detailed in [Bowman et al., 2004] and [Shilling and Shinn-Cunningham, 2002]. For instance, it includes binaural cues, head-realted transfer functions (HRTFs), reverberation, sound intensity, etc. Two main kinds of auditory displays can be found. First, stereophonic headphones are directly placed on the user's ears and can provide two different sounds to each ear in order to create sound localization effects. One of the advantages of using headphones is the possibility to provide different

[6] https://www.oculus.com/en-us/
[7] http://www.epson.fr/fr/fr/viewcon/corporatesite/products/mainunits/overview/12411
[8] http://ovrvision.com/entop/

(a) (b)

FIGURE 2.14 – Two examples of visual display devices used for MR user interfaces. (a) The Oculus Rift DK2 is an HMD used for VR applications that provides a 100°horizontal field of view and a 360°FOR with a positional and rotational tracking. Its resolution is 960×1080 pixels per eye. (b) The Epson Moverio BT-300 is an example of see-through glasses used for AR applications. It provides a 23°diagonal FOV and depending on the tracking technology used can provide a 360°FOR.

sounds to multiple users. Second, external speakers can be placed at different locations in the real world. It avoids users to wear any additional device but it is more complicated to display different sounds to each user. Simulating localization with external speakers is also more complicated as the emanating sounds can interact with the real environment, and therefore, it can decrease the sound quality of the simulation [Bowman et al., 2004].

(a) (b)

FIGURE 2.15 – Two examples of haptic output devices. (a) The Novint Falcon is a ground-reference haptic device that can apply force on the 3 axis of translation. (b) The Cybergrasp glove is a body-referenced haptic device that can apply forces on each finger in order to prevent the user's fingers from penetrating into a solid virtual object 2.15b.

Haptic devices

According to Burdea [Burdea, 1996], haptic feedback refers to both kinesthetic and tactile feedback. Using haptic feedbacks gives the opportunity to the end-user to "touch" the VE. In most cases, haptic devices can also be categorized into the input device category as they also include tracking systems. One challenging issue for haptic display systems is haptic rendering [Bowman et al., 2004], which refers to the software components used to compute the forces based on a physical simulation.

Kinesthetic (or force) feedback triggers the receptors in the human muscles, joints and tendons. A force feedback device can exert a controlled force by the means of mechanical actuators during the user interaction [Florens et al., 2007]. This type of devices has different properties. First, the number of degrees of freedom on which a force can be applied. Then, the maximum forces that can be applied by the device (in Newton). An haptic device can be ground-referenced if it is placed on the floor, ceiling wall or on the desktop [Bowman et al., 2004]. For instance in that category, the

Novint Falcon[9] illustrated in Figure 2.15a offers haptic feebacks on the three axis of translation, and in the same way the Haption Virtuose^{TM} 6D[10] also offers feedbacks on the three rotation axis. For these two devices, the user has to hold the device effector with one of his hands. SPIDAR systems [Hirata and Sato, 1992] are also ground-referenced haptic devices composed of actuators that provide a force through a set of strings attached to an object or to the user's fingers. On the contrary, body-referenced haptic devices are placed directly on some part of the user's body [Bowman et al., 2004]. In that category we can talk about exoskeleton or hand-force-feedback devices such as the Cybergrasp®[11] glove illustrated in Figure 2.15b. It is also possible to simulate haptic feedback without applying any force. In that case we talk about pseudo-haptic feedback [Lecuyer et al., 2000] that consists in using passive mechanisms combined with visual or sound feedbacks in order to give the user an impression of force feedback. For instance, the Elastic-Arm [Achibet et al., 2015], is a body-mounted elastic armature that links the user's hand to his shoulder in order to make the user perceive a progressive resistance force when he extends his arm.

Tactile feedback is the second type of haptic feedback. It stimulates the user tactile sens in order to feel elements such as the texture of surfaces, temperatures or vibrations. For instance, most today's game controllers integrate vibrotactile actuators in order to produce vibrations during games. Another example is the Cybertouch®[12] glove that has vibrotactile stimulators on each finger and on the palm in order to provide tactile feedbacks.

Smell output devices

In the field of human computer interaction, using smell as a medium is really less unexplored than visual, sound and haptic outputs. Using olfactory stimulations can enforce the immersion feeling of the user. For example, it could be used to simulate the smell of virtual food or the smell of a meadow. This kind of device is very uncommon. However, some academic researchers are working to develop such devices. For instance, Nakamoto et al. [Nakamoto and Minh, 2007] propose a smell output device composed of multiple solenoid valves that can blend 32 component odors. In the same way, Sato et al. [Sato et al., 2008] explored a pulse ejection technique. Moreover, some olfactory displays that use vaporizers have already been commercialized such as Scentair[13] or will be commercialized such as the Feelreal solution [14].

Taste output devices

As well as smell output devices, taste displays are not really commons in interactive applications. Taste can be classified into 5 different basis tastes: : sweetness, sourness, saltiness, bitterness, and umami. For instance, Aminzade [Maynes-Aminzade, 2005] introduces two "edible user interfaces" (EUI). It proposes two low-resolution gustatory devices. First, the BeanCounter, a discrete gustatory device that can dispense jellybeans with different flavors. Second, the TasteScreen is a continuous gustatory device that can blend different flavors that are placed into cartridges and dispense the result as liquid redisdue on a PC monitor. In the same way, Iwata et al. [Iwata et al., 2004] propose the "Food Simulator" [13] that integrates an interface that displays the biting force, auditory information, and the chemical sensation of taste. Taste is rendered by releasing prepared taste components using a micro injector.

Some work also try to create a taste simulation without any taste output devices. For instance, a method introduced by Narumi et al. [Narumi et al., 2011] propose to "Pseudo-gustation" method to change the perceived taste of food when eaten by changing its appearance and scent in an AR application. It exploits the fact that what we sense as taste is affected by what we smell and what we see.

[9] http://www.novint.com/index.php/novintfalcon
[10] http://www.haption.com/site/index.php/fr/products-menu-fr/hardware-menu-fr/virtuose-6d-menu-fr
[11] http://www.cyberglovesystems.com/cybergrasp/
[12] http://www.cyberglovesystems.com/cybertouch/
[13] http://www.scentair.com/
[14] http://feelreal.com/smells

2.3.3 Interactions techniques

In the "perception, cognition, action" loop described before, the action step refers to the interaction part of MR applications. During this step the input devices are manipulated by the user or they directly sense what he is doing and these actions are then reported to the system. If these manipulations are made in order to accomplish a task into the application, we talk about an interaction technique. Indeed, as defined by Bowman et al. [Bowman et al., 2004], an interaction technique is a method to accomplish a task in an application that includes both hardware and software components. When an interaction technique combines different input devices that acquire different sensory modalities, we talk about multimodal interaction. A common example is the combination of gesture and voice recognition systems such as the "Put That There" system proposed by Bolt [Bolt, 1980]. Interacting with VEs implies a wider range of possible interactions than classical 2D user interfaces. Indeed, this kind of interfaces mainly relies on the WIMP (Windows, Icon, Mouse, Pointer) interaction paradigm as well as tactile interactions. These kinds of interactions are often not appropriate for VE. Indeed, to interact with VE, 3D interactions are more adapted. According to Bowman et al. [Bowman et al., 2004], 3D interactions are interactions performed in a spatial context. Interaction techniques are based on metaphors that can be natural, pseudo-natural or symbolic. For 3D interactive systems, three kinds of interaction techniques are proposed by Hand [Hand, 1997]:

- Objects selection and manipulation interaction techniques allow the user to select and move scene objects. In that category, we can cite hand-based interaction techniques that use a virtual model of the user hands such as the Go-Go interaction technique [Poupyrev et al., 1996]. Some other interaction techniques are based on pointers such as the ray-based interaction technique presented by Mine [Mine et al., 1995] illustrated in Figure 2.16a.

- Navigation interaction techniques allow the user to change his point of view. For instance, moving the point of view with a joystick can be considered as a symbolic interaction while walking in the VE as in the real world can be considered as natural. In Figure 2.16b we give an example of interaction techniques that mixes these two paradigms: the Joyman [Marchal et al., 2011] proposes an interaction technique based on a human-scale joystick.

- Application control interaction techniques allow the user to change different application parameters. In that category interaction techniques are mainly based on graphical menus and on voice and gesture commands. For instance, a graphical menu for application control is illustrated in Figure 2.16c.

(a) (b) (c)

FIGURE 2.16 – Three examples of 3D interaction techniques. (a) A ray-based interaction technique for selection and manipulation controlled with a 6-DOF tracking device. (b) The Joyman [Marchal et al., 2011] proposes an interaction technique based on a human-scale joystick for navigating in VEs. (c) A graphical 3D menu for application control

When the interaction implies multiple users at the same time on the same object, we talk about collaborative interactions. This kind of interaction can be necessary for re-creating real situations such as the manipulation of heavy objects that needs multiple persons. According to Margery et al. [Margery et al., 1999], collaborative interaction can be classified onto multiple levels:

- Level 1: the interaction includes the perception of distant users through 3D representation of each user (avatars) and communication (text or audio).
- Level 2: individual interactions enable each user to act on the scene.

  Level 2.1: scene modifications are constrained by scene design.

  Level 2.2: modifications made by a single user are not constrained.
- Level 3: collaborative/codependent interactions enable users to modify the same object at the same time.

  Level 3.1: two users can act on the object in independent ways.

  Level 3.2: two users can act in a codependent way: the resulting reaction of the object is a combination of the different inputs.

2.3.4 Analysis

Compared to 2D user interfaces, MR relies on a lot of possible variety of input devices. We have identified 5 main categories of input devices but we have shown that in each category there are a lot of possible devices with very different properties. If an application is developed for one particular device we can wonder if it can seamlessly be exchanged by another one from the same category or by another one from a different category. Indeed, maybe the end-user of an application will not own the needed devices. We could also imagine a scenario where a device is replaced at runtime by another one. To continue, desktop devices are commonly handled by most of software tools for the creation of interactive systems, but as the other categories are less common they need the installation of specific drivers and the use of particular SDKs. Learning and integrating these SDKs can be difficult and time consuming for developers.

We can make similar observations for output devices than for input devices, MR user interfaces rely on way more possible output devices than 2D user interfaces. The problem of devices replacement is the same, and in the same way as input devices, outputs devices also often need the use of particular drivers and software libraries. More than handling just the different possible devices, this category also implies the difficulty to generate the outputs. Indeed, the application needs to be able to generate the different outputs which include problematics such as managing stereoscopy and head tracking, generating spatialized sounds and computing force feedbacks according to a physical simulation.

Software tools that could help developer to adapt their application to the different input and output devices could be a solution to cover these different issues. These tool must hide the complexity of the different devices. Moreover such tools could also give to an application the capacity to evolve when new devices appear. Indeed, if we want the developer to really focus on the content of his application and on the interaction mechanisms, we need to provide him the tools that already cover these different issues.

To continue, we have shown the wide diversity of interactions used in MR user interfaces. Choosing the interaction techniques for an application can be a difficult task. Indeed, a choice can be influenced by multiple aspects of an application. Of course, the input devices and the output ones have an important impact on these choices. Some interaction techniques can be more adapted to particular devices and even not compatible with others. For instance, it could be complicated to precisely control a virtual hand composed of five fingers with a Flystick. A Dataglove or a hand tracker would be more convenient Moreover, theses choices can also be influenced by the type of the target application and the skills and habits of target users. Indeed, for instance, some 3D interaction techniques can be difficult to use for people who do not use MR user interfaces very often. Adaptation tools may help developers to automatically adapt their applications to these different aspects.

2.4 Development tools

The creation of MR applications first involves the creation of the assets that will be part of the VE. These assets can be of different types, for instance, 3D models created with a modelisation tool

such as 3ds Max[15] or CAD tool such as Autocad [16], or obtained by scanning a real object. These assets can also include animations, shaders, sounds, etc. The goal of 3D development tools is to enable developers to use these different assets in order to create interactive applications. They also aim to help the developer to integrate the devices presented in Sections 2.3.2 and 2.3.1. However they do not propose solutions to automatically adapt an application to these devices. Today, most developers use integrated tools that propose the basis components for most of the steps involved in a 3D application development work-flow. Software middlewares for handling specific features are sometimes combined with these tools. Indeed, today it is really uncommon to start to develop a 3D application directly from scratch with a graphics library or with a 3D engine.

In this section we present the main tools used to develop MR applications. First, the markup languages are designed for the creation of embedded 3D applications. Second, game engines are complete tools that include the main software components for 3D development and that ease the development with visual editors. Last, VR and AR middlewares are frameworks that handle specific features such as devices management, AR registration, stereoscopic rendering, etc.

2.4.1 Markup languages for embedded execution

Markup languages enable users to create interactive 3D applications even with only a little knowledge of coding. The created files can then be embedded in an application that includes the associated execution environment that interprets the language. In order to navigate through 3D worlds on the World Wide Web, the creation of VRML[Carey and Bell, 1997] (Virtual Reality Modeling Language) has been investigated during the 1990s. Its first version was specified in 1994 [17]. VRML is a mutli-platform declarative language for the creation of interactive VEs embedded in applications. VRML is an ISO standard that has been initially designed in the context of Web3D but its execution is not limited to this context and can also be done in a standalone application. VRML includes concepts for the creation of simple and complex 3D shapes, for the integration of 3D sounds, for animation, for interaction, for handling events, for scripting (with VrmlScript, ECMAScript or JavaScript), etc. It also includes the concept of prototypes, which are parametric sub-graphs that can be instantiated in different scenes. The evolution of 3D technologies have resulted in the creation of X3D (extensible 3D)[18] that is a meta-language based on XML retro-compatible with VRML. This ISO standard includes additional features such as shaders, multi-texture rendering, geo-localization, etc. An example of simple X3D file is given in Figure 2.17.

```
<Scene>
        <Background skyColor='1 1 1'/>
        <Viewpoint description='scene view' orientation='-1 0. 0 0.68' position='0 2.9 4.83'/>
        <Shape>
                <Cylinder radius='2' top='false'/>
                <Appearance>
                        <Material/>
                </Appearance>
        </Shape>
</Scene>
```

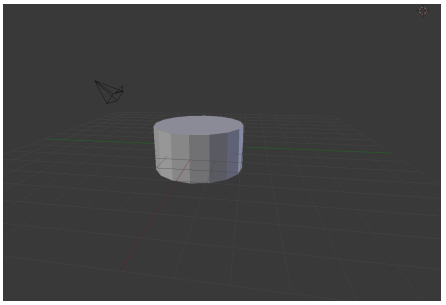


FIGURE 2.17 – An example of a scene described with a X3D file. Here, the scene contains only one cylinder with a radius of 2. The viewpoint refers to the camera pose in the scene (position and orientation). Here, we give the preview of the scene in Blender.

With X3DOM [Behr et al., 2009], X3D files can be integrated natively into web pages. Indeed, this solution proposes an integration of X3D into the DOM tree of HTML5 in order to be supported by all web navigators. Some authoring tools can be found for the creation of VRML and X3D such

[15] http://www.autodesk.fr/products/3ds-max/overview
[16] http://www.autodesk.fr/products/autocad/overview
[17] http://gun.teipir.gr/VRML-amgem/spec/index.html
[18] http://www.web3d.org/standards

as Blender[19] and Sketchup[20].

As VRML and X3D are standards for the creation of VE, ARML (Augmented Reality Markup Language) [Lechner, 2013] is an OGC® standard file format for the creation of augmented reality applications. It allows developers to describe virtual objects in an Augmented Reality (AR) scene with their locations related to the real world. As VRML it can also use ECMAScript as a scripting language. Additionally, it is also possible to dynamically modify the AR scene based on user input.

These file formats provide a lot of possibilities for creating VR and AR applications. Nevertheless they do not provide the same flexibility as modern tools such as game engines for the creation of such applications. Indeed, they do not include optimization capabilities, the possibilities to create complex animations and behaviors are limited and the range of interaction and display devices possibly used is also limited.

2.4.2 Game Engines

Today, in order to develop 3D applications, especially video games, using a game engine is certainly the approach that is the most chosen by developer and designers. According to Gregory [Gregory, 2009], a game engine is a framework that separates the core software components from the art assets, game worlds and rules of play. It can be extensible and then be used as the "foundation" for other 3D applications without important modifications. In the category of game engines we can find for example Unity3D[21], the Unreal Engine[22], Lumberyard [23], or the CryEngine[24]. The main goal of game engines is re-usability. Indeed, with a game engine, all the code and all the assets developed for a particular application can be used seamlessly in another context to develop an another application. Usually a game engine provides the following built-in software components:

- the scene graph is the structure that logically and spatially arranges the different virtual objects of a VE. It is a collection of nodes in a graph or a tree structure. Each node defines a transformation matrix (position, rotation, scale). A node can also be associated with other components such as geometries, animations, behaviors, etc.

- The rendering system handles the display part of the game engine. It converts the 3D models into 2D images. It can render photorealistic effect or non-photorealistic ones. For most game engines, rendering systems are based on rasterization and on graphics APIs such as OpenGL™[25], DirectX® and more recently Metal [26] on iOS and Vulkan™[27].

- The physics engine handles the physical behaviour of virtual objects. It includes a collision detection process and it can emulate the real physical laws as well as imaginary ones.

- The audio system is used to render sounds into a 3D application. It can include mechanisms that simulate 3D sounds, reverberations with the 3D environment and also binaural effects.

- The scripting system proposes one or multiple programming languages in order to create application behaviors or interaction techniques.

- The animation system is used to animate the virtual world. It can include inverse kinematics and key-frame algorithms. For instance the animation system can be used to animate a virtual agent body as well as his face.

- The Graphical User Interface (GUI) system enables a developer or a designer to create 2D or 3D interactive menus for application control purposes.

[19] https://www.blender.org/
[20] https://www.sketchup.com/fr
[21] https://unity3d.com/
[22] https://www.unrealengine.com/
[23] https://aws.amazon.com/lumberyard/
[24] http://cryengine.com/
[25] https://www.opengl.org/
[26] https://developer.apple.com/metal/
[27] https://www.khronos.org/vulkan/

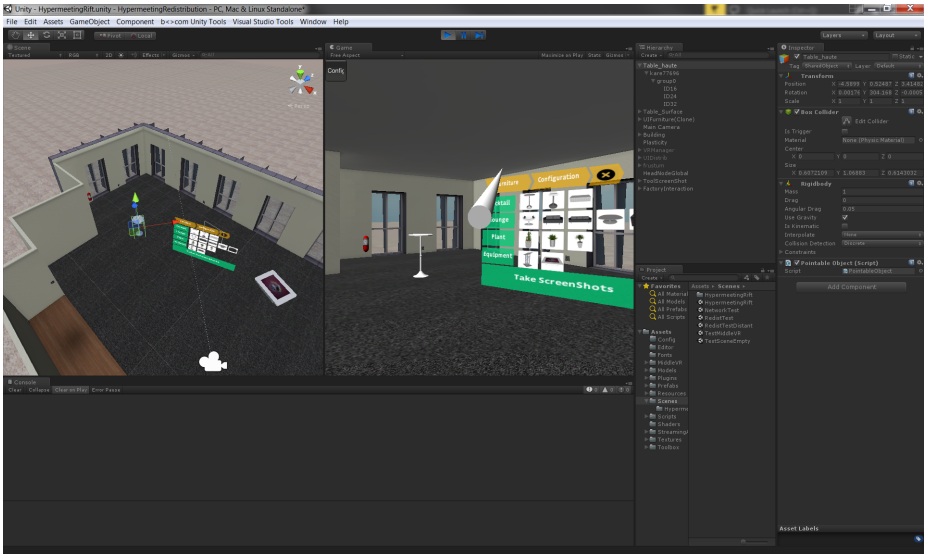


FIGURE 2.18 – The Unity3D visual editor. It allows a developer or a designer to visually create and test a 3D applications.

- The Artificial Intelligence (AI) system allows creating "intelligent" virtual characters. Most of the time this system includes an algorithm that extracts the navigable zones in the 3D world and also path finding algorithms such as A* search algorithm that exploit these zones.

- The network system is useful for creating multi-users applications. It handles state synchronization of the different instances of the same shared applications. For instance, it synchronizes the position and rotation of the virtual 3D objects in order to maintain a consistent 3D world.

- The visual editor provides a user interface for creating and parameterizing a 3D application. The goal is to minimize the programming part of the development. It lets developers visually create a 3D applications by editing parameters, associating behaviors to objects through drags and drops, or moving objects directly into the 3D world. An example of visual editor is given in Figure 2.18.

With modern game engines the created applications can be deployed on a wide variety of computing platforms. It includes PC environments (Windows, MacOS, Linux), mobile platforms (iOS, Android), Web player, native web with WebGL support and also video game consoles. For now, game engines do not integrate an important VR and AR support. Indeed, they mainly integrate desktop devices and they do not provide built-in 3D interactions yet. However, with the growing interest for VR, some game engines such as Unreal and Unity3D now support natively different HMDs.

2.4.3 VR and AR tools

In most cases, 3D development tools such as game engines are not enough for the creation of MR user interfaces. Therefore, some frameworks, toolkits and middlewares have been developed in order to bring to these tools essential features needed by this kind of applications. The goal of these tools is to abstract hardware and software complexities of MR user interfaces in the system in order to help developers in the creation of such applications. Here, we give some of the common tools used for the development of MR user interfaces. These tools and additional ones will be more described in the next Chapters.

For the creation of VR applications, development tools often implement the same basis features. First, they aim to give access to developers to the wide variety of possible input and output devices such as trackers and HMDs. For instance, in that category we can cite VRPN[Taylor et al., 2001], VR Juggler [Bierbaum et al., 2005], the MORGAN framework [Ohlenburg et al., 2004], and the

commercial solution MiddleVR [28]. Second, Appart VRPN, these cited tools also handle stereoscopy and clustering for managing multi-display systems. Last, some of these tools such as MiddleVR also provide a library of ready-to use interaction techniques. Last, VR delopment tools such as MiddleVR, the Morgan framework[Ohlenburg et al., 2004] and OpenMask [Margery et al., 2002] give the possibility to create collaborative virtual environment (CVEs). Such solutions commonly include components for handling network synchronization, collaborative interactions and awareness mechanisms.

For the creation of AR applications, development tools mainly focus on providing calibration and registration algorithms in order to precisely align the real world and the virtual objects. For instance, ARToolkit[29] is an Open-Source toolkit that provides marker-based computer vision algorithms that can run on multiple platforms (Windows, Android, iOS, etc.). In the same way commercial solutions such as Vuforia™[30] and Wikitude [31] also provide markerless solutions.

2.4.4 Analysis

These commonly used tools propose the basic mechanisms for the creation of MR applications. However, they do not provide solutions to really automatically adapt these applications to different contexts of use.

However, today these tools, especially game engines and VR and AR middlewares, are commonly used by the MR developers community. Developers have their habits with these tools and may have a lot of software components developed for them.

Therefore, being able to interface our software solutions with these tools could have two main advantages. First, it would let us not reimplementing a lot of basic features. Second, they could be easily integrated into the developers workflow in order to not disturb them. However, we must not be dependent to a particular development tool to ensure the upgradability of our framework independently of previous tools evolutions.

2.5 Global Analysis

In this chapter we described the design space for the development of MR user interfaces. The goal was to highlight the different components involved in the development process of such applications and the complexity of such development.

As shown in Section 2.2, VR, AR and AV share a lot of concerns and possible use cases. Each of these domains relies on immersion and interaction with 3D content. Each domain has also its own specificities and problematics that must possibly be taken into account.

As demonstrated in Section 2.3, comparing to 2D user interfaces MR relies on a lot of possible variety of input and output devices. We have shown that these different devices can be grouped into categories but we have seen that in each category there is a lot of possible devices with very different properties. Most of these devices need the use of particular SDKs, which can complexify the development process. In the same Section we also show that MR user interfaces rely on a lot of possible interaction techniques. Choosing the interaction techniques for an application can be a difficult task because it depends on a lot of aspects such as the available devices or the user skills and habits. From these observations we can conclude that the context of use of a MR user interface (devices, users, use case, etc.) can have a real impact on its usability.

In Section 2.4, we presented different development tools that can handle some of the specificities of MR user interfaces. These commonly used tools propose the basic mechanisms for the creation of MR applications but they do not provide solutions to really automatically adapt these applications to different contexts of use. Indeed, with most of the presented tools, an application is developed for a particular use-case, with determined interaction techniques for determined devices.

From the analysis performed in Sections 2.2.4 and 2.3.4, we can say that this kind of approach has multiple issues for the development of MR user interfaces. For instance, re-usability: will the

[28] http://www.middlevr.com/
[29] http://artoolkit.org
[30] http://www.vuforia.com/
[31] http://www.wikitude.com/

developed components be compatible for another applications, with other interactions and with other devices? The devices part is a very challenging point, as there is a daily emergence of new MR devices. An application needs to possibly evolve in order to take into account these new devices. To continue, will this application fit all the possible users skills and habits? Indeed, most of people are now very comfortable with 2D user interfaces but 3D interaction is still something new for most of them. Moreover, creating an application for a determined context of use also excludes it to run on a wide variety of other contexts. Maybe, the end-user will not own the devices used in the application and therefore will not be able to use it. Developing a version of an application for each possible configuration has an important combinatorial complexity and therefore is not a very flexible way toward adapting it to various features. Considering adaptation during the creation of such interface is the best way to distribute it widely. Nevertheless, as explained in Section 2.4.4, it is important to be possibly interfaced with these tools. It could have two main advantages. First, it would let us not reimplementing a lot of basic features. Second, they could be easily integrated into the developers workflow in order to not disturb them.

To conclude, we can highlight the need for software adaptation tools in the field of adaptable MR user interfaces. Adaptations tools could help developers to create applications that can be usable on a wide variety of contexts of use and that can satisfy the needs of a lot of users. The concept of plasticity presented in the next Chapter aims to add adaptation features to MR applications. In the next Chapter, we present complementary related work on adaptation for MR user interfaces and more specifically on this concept of plasticity. From these related work, we present the different types of entities for which adaptations can be intended and how a MR user interface can be adapted according to these entities. Our goal is to identify the requirements for the development of plastic MR user interfaces.

Chapitre 3

Plasticity for Mixed Reality User Interfaces

3.1 Problematics and Motivations

As presented in the previous Chapter, the design space for the development of MR user interfaces is very large. Indeed, during such development, designers and developers have to handle a lot of input and output devices, a lot of interaction techniques, a lot of possible kinds of targeted users, and a lot of ways to present content. The third dimension offered by MR user interfaces also complexifies the developments. For instance, as explained by Lindt [Lindt, 2009], a common problem of current MR user interfaces is that they only work with specific interaction devices and techniques. Developing manually a version of an application for each possible configuration is not a very flexible way toward adapting it to various features.

Concretely, the research orientation that led to this thesis has been motivated by the need to develop multiple applications that must be usable in a wide variety of possible context of use. For instance, such an application could consist in laying-out an empty room with furniture. This application is described in the following chapters and most of our examples are based on this application. Its goal is to help people to plan the use of particular premises. Regarding its specifications, the application must be available on a wide variety of platforms, from desktop ones to immersive systems through mobile devices. It must be independent from any concrete interaction devices, and new devices must be easily integrated. In a context of mobility, easy switching from a platform to another one must be supported. For instance, a user may have to switch from an immersive system to a mobile device in order to continue his work while mobile. Collaboration must also be considered in order to let different users work on the same room configuration in the same shared virtual environment. To continue, the application may be used by expert and novice users. All do not have the same interaction skills and preferences. Thus, the application needs to be adapted to each particular user. This adaptation process must be observable and configurable at runtime by the end-user. Furthermore, the different components implemented for this application must be easily reusable for another project. They must be independent from this particular application and from the 3D framework used in the project.

In order to develop an application that can answer to these different requirements, today, solutions propose to adapt MR user interfaces automatically to fit a given context of use. As explained in the next Section, in that case we talk about configurable, auto-configurable, adaptable or adaptive user interfaces, depending on how the adaptation is handled. A recent concept tries to gather these different categories into one field of research: the notion of plasticity for the creation of plastic user interfaces that is also defined in the next Section.

Using such a tool for the development of a MR user interface with adaptation or plasticity capabilities can induce a lot of advantages:

- **Development times and costs are reduced.** As explained by Calvary [Calvary, 2007], with such solutions there is no more need to develop one version of an application for each

particular context of use. Therefore, the development process is simplified and shortened, and so will cost less.

- **Maintenance times and costs are reduced.** Consistency between the different versions of the application is ensured and the differences are only managed by adaptations mechanisms. Therefore, the maintenance of the application is simplified.

- **An application can be distributed widely.** With integrated adaptations mechanisms an application can target a lot of possible users. Indeed, a same application can be adapted to the specific user's platform, to his skills, etc.

- **An application can evolve without supplementary costs.** With plasticity an application can be easily upgraded in order to support new interaction techniques or the latest devices.

- **An application can benefit from Usability continuity.** More than being able to fit a wide variety of contexts of use, an application that supports adaptations can switch from a context to another one without degrading the interaction quality and the interaction possibilities. In that case, we talk about usability continuity.

- **Attractiveness of an application can be improved.** Indeed, relevant adaptations can have a substantial impact on the interest of the user in an interactive system. Adaptations can bring a MR user interface from an impersonal and maybe not optimally usable shape to a personalized one that will meet user's expectations.

In this thesis, we focus on Plasticity because we consider that this concept gathers the different fields of research related to the notion of adaptation for human computer interfaces. This Chapter aims to detail the Plasticity Concept in the context of MR user interfaces, to provide a survey of existing solutions and to propose a problem space for the development of plastic MR user interfaces. The results of this survey and the proposition of this problem space have been published in [Lacoche et al., 2014].

3.2 The Plasticity Concept

The Plasticity concept comes from the field of 2D user interfaces and the first definition of the concept has been given by Thevenin and Coutaz [Thevenin and Coutaz, 1999].

> *Plasticity is the capacity of an interactive system to withstand variations of both the system physical characteristics and the environment while preserving its usability.*

As explained by Calvary et al. [Calvary et al., 2002a], the term "plasticity" is directly inspired from the property of materials that expand and contract under natural constraints without breaking, thus preserving continuous usage. Plasticity takes into account the variation of the context of use [Calvary, 2007]. The conception of an interactive system for multiple contexts of use is not enough. Indeed, a plastic interactive system needs to be able to detect the current context of use and if necessary to change its appearance and behaviour in order to fit this current context.

The plasticity concept includes requirements such as parametrization, customizability, configurability, user-adaptivity, reusability, mobility and portability. In the field of real-time interactive systems (RIS), the plasticity definition is adapted to cover the interaction problematics. The definition of Thevenin and Coutaz means that code portability is a necessary condition but is not sufficient for an interactive system to be considered as plastic. This notion of portability means that the system can be run on all considered platforms but does not ensure a good usability. That is why usability continuity has to be guaranteed too. Performances and the possibilities offered by the system have to be at least constant.

In order to achieve such a goal an interactive system must be adapted. The adaptation design space for plasticity can be modeled onto five different axes:

- **The adaptation controller** determines the entity that decides on the adaptation. It can be the user (end-user or developer) or the system. These adaptations can be chosen by users from a predefined set of parameters: this is user adaptability. They can be chosen automatically by the software: this is system adaptivity. Another level can also be found: the developer/deployer of the application. In that case, the adaptations are chosen by a specialist who can configure the system with programming skills, by editing a configuration file or by using an authoring tool.

- **The adaptation sources** point to the entities for which adaptation is intended. Three examples are given for these entities by Thevenin and Coutaz [Thevenin and Coutaz, 1999]: users, environment and hardware characteristics.

- **The adaptation targets** concern modifications applied by the system and the software components involved in adaptation. For example, it can refer to the application content or the interaction techniques.

- **The temporal dimension of adaptation.** The adaptation mechanism is static if it occurs between sessions and is dynamic if it can also occur at runtime.

- **The adaptation means** are defined by Calvary et al. [Calvary et al., 2004a]. The first one is recasting which consists in locally adapting an application to fit the context of use without modifying its component distribution. For instance, it can consist in locally adding, deleting or replacing application components [Vanderdonckt et al., 2008]. The second one is redistribution, which consists in changing the component distribution of an interactive system on multiple dimensions such as display, platform and user. Redistribution includes migration and replication mechanisms and is more described in Section 6.

Lindt et al. propose a complementary classification of adaptive user interfaces based on the temporal dimension (at runtime or between sessions) and the adaptation controller (here it considers only the user or the system) [Lindt, 2009]. This classification is shown in table 3.1.

adaptation time (when) \ adaptation controller (who)	user	system
Compile time	configurable user interfaces	auto-configurable user interfaces
Runtime	adaptable user interfaces	auto-adaptive user interfaces

TABLE 3.1 – Classification of user interfaces based on adaptation controller and adaptation time from [Lindt, 2009]

Lindt also defines an adaptive user interface as any interface than can be classified in one of the four possible categories. Therefore we can say that an adaptive user interface can be considered as plastic as long as the adaptation is done in order to guarantee a constant user experience or to improve it while considering the current context. The definition should not just be limited to usability continuity but should also be extended to usability improvement.

These definitions suit both 2D or 3D applications. Nevertheless, as demonstrated in Chapter 2, the design space for the development of 3D applications is wider than for 2D ones. Indeed, there is a great difference between these two categories of user interfaces. While 2D user interfaces are mostly based on Windows, Icons, Menus, Pointers (WIMP) and interaction devices like mouses

and touch-screens, 3D user interfaces can use a wider range of interactions techniques, input and output devices and ways to present content. Therefore, the set of possible adaptation sources and adaptation targets for 3D is larger than for 2D. We can deduce that the combinatorial complexity is even bigger for 3D user interfaces. Indeed, developing a code for all possibilities creates a great number of possibilities detailed in the following formula:

$$\begin{aligned} number\ of\ possible\ codes = & Numberof\ adaptation\ controllers \\ & \times Numberof\ Adaptation\ sources \\ & \times Numberof\ adaptation\ targerts \\ & \times Numberof\ adaptation\ time \\ & \times Numberof\ adaptatio\ means \end{aligned}$$

Plasticity aims to solve this combinatorial complexity in order to reduce the development times and costs.

Other fields of research are also concerned by these issues and can then be considered close to plasticity and adaptive user interfaces even if they do not mention it as such. First we can mention intelligent user interfaces that change their behavior to adapt to a person or a task [Ross, 2000]. As an example an autonomous agent may provide users with advice if it notices his/her difficulties to interact with the system. Next, we have adaptive hypermedia [Brusilovsky, 1996] that focuses on the adaptation of hypertext and hypermedia (including 3D [Dachselt et al., 2006]) systems to the user and to the environment [Brusilovsky, 2001] and could be therefore considered as plasticity. By building a user model according to his/her goals, preferences and knowledge, the system can adapt itself to his/her needs. Environment adaptation appeared in adaptive hypermedia with ubiquitous computing. The environment can refer to the user location or to its hardware platform. Therefore, we can compare this kind of adaptation with context aware applications [Schilit et al., 1994] [Chen et al., 2000] that are systems that examine the computing environment (the context) and can react to its changes. As reported by Chen et al. [Chen et al., 2000], there is no unified definition for context but according to Dey et al. [Dey, 2001] context means any information that can be used to characterize the situation of something that is relevant to the interaction between a user and an application (person, place, object). As well as for plastic user interfaces, adaptive hypermedia and context aware applications are systems that examine adaptation sources to find the best adaptation targets in order to maximize the user experience.

In the same way as adaptive Hypermedia, in the web context, responsive web design appeared recently to help developers during websites creation in order to ensure their perfect usability and readability in a large range of platforms [Marcotte, 2010]. With CSS3 media queries, a website is able to adapt automatically its layout to the rendering device. Media types included in the CSS3 specification allow developers to inspect the physical characteristics of the rendering device in order to adapt them automatically. In the case of responsive web design, the only adaptation source is the hardware, especially the size of the rendering screen. The aim is to create and manage one version of a website to serve all the possible platforms such as desktop PCs, smartphones, tablets or connected TVs.

Regarding the 5 axes of adaptations of the plasticity concept, the adaptation sources and targets can take a lot of values when we talk of 3D user interfaces. In the following Sections, we define the different adaptation sources and targets involved in the literature and we give some examples on how they are used in adaptation mechanisms.

3.3 Adaptation Sources

As explained in the previous Section, adaptation sources refer to entities for which adaptation is intended. These entities must be relevant to the interaction between the system and the users to be taken into account. Thus, according to the context definition given in section 3.2, we can generalize by talking about context. An adaptation source refers to something that represents context for the application. First, these properties must be modeled in the system. Secondly, they must be available to the adaptation mechanisms. From the design space described in Chapter 2 and from the different related work about plasticity we can extract three main adaptation sources for MR

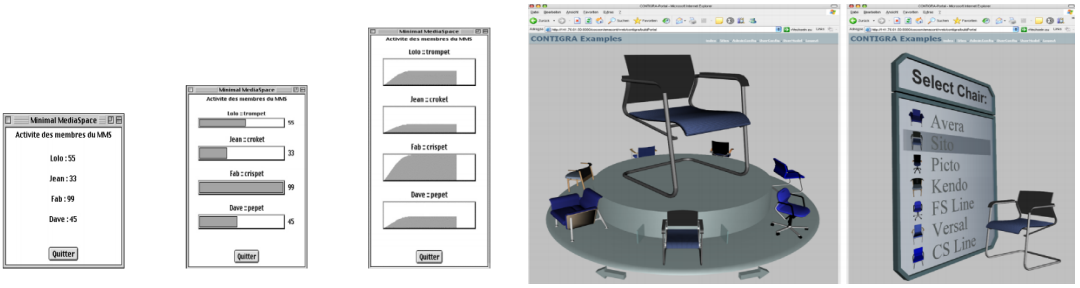
user interfaces. Adaptation to hardware is described in Section 3.3.1, adaptation to users and their environment is detailed in Section 3.3.2 and adaptation to the data is reported in Section 3.3.3.

3.3.1 Devices

There is a wide variety of platforms on which an application can be run and this list grows substantially when we talk about 3D as explained in Sections 2.3.1 and 2.3.2. All platforms do not offer the same capabilities. For instance, without talking about code portability, it is obvious that we cannot run the same version of an application in an immersive cube than on a tablet for virtual reality, or on a tablet with camera than on see-through glasses for augmented reality. In both cases, setups are too different. Typically a platform is one or more computing units connected with input and output devices. This composition may change during runtime, some devices may be disconnected while others may be added. There is a wide range of possible devices used for MR user interfaces and these devices have a lot of different properties as detailed in Sections 2.3.1 and 2.3.2. The properties, the capabilities and also the limitations of these devices can be used for adaptation purposes. Regarding the solutions that take into account the platform capabilities as an adaptation source, three main types of information are taken into account:

- the output devices capabilities,
- the input devices capabilities,
- the computing power of the platform.

The properties of the output devices can impact the sensory feedbacks provided by an application. For instance, the size property of a display screen can be used to dynamically adapt the layout of an application. For instance, as shown in Figure 3.1a Thevenin et al. [Thevenin and Coutaz, 1999] adapt the layout of a 2D user interface according to the screen space variations. In the same way, as shown in Figure 3.1b, for a 3D user interface, Dachselt et al. [Dachselt et al., 2006] propose a solution to switch between multiple 3D layouts according to the screen size.



(a) (b)

FIGURE 3.1 – Two examples of how the layout of an application can be adapted according to the output capabilities of a given platform (here, the screen size). (a) In [Thevenin and Coutaz, 1999], the quantity of information displayed and how this information is represented change according to the screen space variations. (b) In [Dachselt et al., 2006], depending on the screen size the layout of the 3D scene is changed. Here, on the left a ring menu is chosen for a large screen and on the right a floating menu is chosen when the screen is smaller.

To continue, the input devices can impact the way we interact with an application. One critical point is to seamlessly adapt the application interaction techniques to these devices as detailed in Section 3.4.1. As detailed by Lindt in [Lindt, 2005], exchanging an input device by another can result in an impractical interaction technique. For instance, for a 3D-cursor controlled with a 6-DOF tracker, if this tracker is replaced by a speech input device at runtime, the interaction technique is no longer adapted. This assessment is also discussed by Bowman et al. [Bowman et al., 2004].

According to them, many different interaction techniques can be mapped onto any given input device. However, in some cases, these mappings can be unnatural, inefficient and inappropriate.

Regarding the computing units of a system, it can be symbolized by different properties such as the bandwidth of the network or the computing resources available. For example a transfer of large amount of data coupled with a slow connection can affect the user experience. The network and the terminal capabilities can be taken into account to send an adaptable version of a 3D content using level of detail (LOD), such as a 3D agent [Kim et al., 2004], or a 3D model of a city [Royan et al., 2007]. More details about LOD representations will be given in section 3.4.2.

3.3.2 Users and their environment

As briefly discussed in Chapter 2 and in Section 3.1, all users do not have the same interaction habits, the same skills, the same needs. Therefore, some solutions propose to adapt the user interfaces according to the user properties. In that case, we talk about personalization. Using such mechanisms can improve an application usability and its attractiveness. The first step in order to be able to adapt the application to a user is to extract or learn from him/her a profile that contains his/her main characteristics that differentiate him/her from the other users. This is the goal of user modeling. A lot of research work have been published about how to construct user models. Surveys of user modeling are given by Kobsa [Kobsa, 1993] [Kobsa, 2001] and by Fischer [Fischer, 2001]. According to Kobsa [Kobsa, 1995] a user model is defined as a set of information and assumptions on an individual user or on user groups that are used in an adaptation process. Different kinds of user properties can be included into this model:

- **The properties that define the user as a person**. As explained by Calvary [Calvary, 2007], it can include general information such as his/her age his/her gender, his/her size. It can also include information about his/her skills, his/her profession or his/her knowledge.

- **The user preferences and interests.** It can include information about the interaction techniques and the devices that the user prefers and also his/her global or application-oriented interests.

- **The user environment.** It can refer to the physical and social properties of his/her environment [Calvary, 2007]. For instance, it can include the user's location, is he/she inside or outside, his/her GPS position, is he/she at home, at work or in transportation. Moreover, it can also include information about his/her surrounding. For example, it would be interesting to know if is he/she alone or in a crowded environment, if is he in a noisy environment or what is the temperature and the illumination conditions of his/her environment.

- **The user monitoring information.** The interaction between an interactive system and a user can be monitored and then reported into his/her user model. For instance, it can be used to gather the user's interaction habits and interests.

Even if a user model can include a lot of information, most of the time personalization systems only consider a small number of properties in order to perform adaptation. For instance, in [Sparacino, 2003] Sparacino represents a museum visitor just by one criteria among three (greedy, busy, selective). This value is then used to propose him/her the most suited visit program on a multimedia guide. In the case of 2D user interfaces, Motti et al. [Motti et al., 2013] compare different studies on interaction techniques with touch-screen for elderly people and demonstrate that age of users must be a source of adaptation. For example, elderly people will perform better if targets on a touchscreen and spaces between them are bigger. Chittaro et al. [Chittaro et al., 2002] and Dos Santos et al. [dos Santos and Osorio, 2004] present e-commerce applications where the personalization of the 3D worlds are made according to the detected user preferences. By tracking the user action, his/her profile with his/her preferences are edited and the visibility of his/her favorite products is increased. As said, it is also interesting to know more about the environment in which the user is located. The use of this kind of properties is cited by Chen et al. in a call forwarding process [Chen et al., 2000]. A user's phone is ringing in his/her office that is empty, consequently

the system detects his/her location (position property) and automatically forwards the call to the nearby phone. In a second time, the system notices that the user is in a meeting, therefore the software forwards the call directly to the user's voice mailbox. Concerning 3D interactive systems, properties of the user environment can also be useful, e.g in an augmented reality application in which the lighting property of the real scene can be used to get more consistency between the real and the virtual world. For instance, this is done by Kanbara et al. in [Kanbara and Yokoya, 2002] where the evaluation of the real scene lighting is made with a mirror ball technique. A mirror ball on a tracked marker is placed in the real scene. Then, the distribution of light sources in the real scene is estimated by processing the projected area of the mirror ball in the final image captured by the real camera. This distribution is then used for the virtual object rendering.

3.3.3 Data

In an application, the user will interact with data that can have different kinds of properties. For a given application, the data may change, for instance according to a user's request. In adaptation systems that consider data as an adaptation source, two kinds of properties are generally taken into account:

- **Data semantic.** The semantic aims to give a meaning to the data.
- **Data structure.** The structure of the data represents the properties of an entire dataset such as the quantity of data, its spatial or temporal distribution or a specific organization (array, graph etc.).

There are two categories of semantic modeling according to Chevaillier et al. [Chevaillier et al., 2012]. The first one is content-oriented semantic modeling, it provides a representation of an application content and it is commonly based on ontologies. The second one is system-oriented semantic modeling, it aims to give meta-access to the features of the application entities.

In most cases, application designers and developers use the content-oriented semantic even if it has not been specified to the system. For example, Walczak uses X-VRML for designing two different applications manipulating each one a particular kind of data: furniture for the first one and art artifacts for the other [Cellary and Walczak, 2012]. The designer chooses the way he will organize the two 3D scenes by using his/her semantic knowledge about data. That is why he/she chooses the museum representation for the artifacts and the shop for the furnitures. Therefore, in applications where semantic is specified to the system, it can be used as a source of automatic adaptation. For example, Hatala et al. [Hatala et al., 2004] use ontologies to match sounds from a database with museum artifacts and user preferences. In the same way, Bilasco et al. use semantic for object retrieval [Bilasco et al., 2007] and propose rules that specify adaptations to 3D objects with a particular semantic. One example consists in deteriorating geometries of "building" type objects as shown in Figure 3.2. To continue, MASCARET [Chevaillier et al., 2012] uses UML (Unified Modeling Language) in order to model the semantic of 3D objects, procedures and behaviors in order to automatically generate interactive 3D worlds with learning scenarios. This knowledge database can then be used as a source of adaptation by 3D autonomous agents.

Other solutions are also interested in adapting the content visualization of an application according to the data structure. In this category, Esnault et al. [Esnault et al., 2010] present a framework for the creation of visualization metaphors that can take into account the hierarchical structure of a database in order to display its content. One provided example is the visualization of a large catalog of videos on demand. To continue, in the context of web search results, AVE (Adaptive Visualization Environments) presented by Wiza et al. [Wiza et al., 2003] present a new approach to adapt the graphical presentation of search results according to the structure of the data returned by the search engine. The goal of this approach is to maximize the readability of the interface. Information about the data structure such as the overall number of documents returned, the quantity of keywords, the files types, can be used in order to choose a good visualization metaphor. The global result also depends on the user's choices as he/she can influence the selection process and modify different parameters of the final interface such as the color that represents particular types of documents.

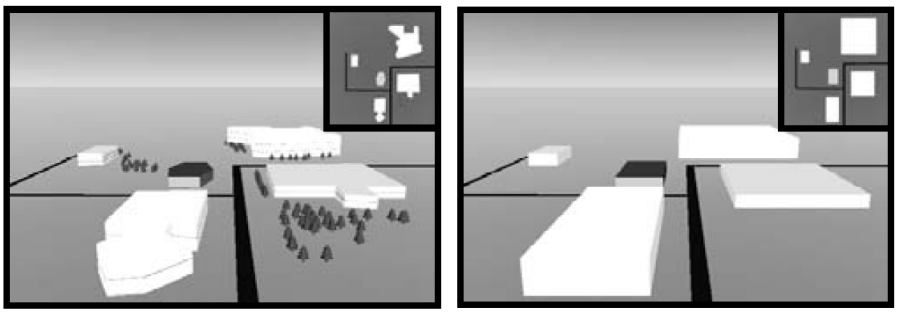


FIGURE 3.2 – In the solution proposed by Bilasco [Bilasco et al., 2007], a 3D scene can be adapted according the semantic of each element. Here, in the image on the right, the geometries of type "building" have been degraded, and the geometries of type "tree" have been removed from the scene.

3.4 Adaptation Targets

As explained in Section 3.2, adaptation targets concern modifications applied by the system and the software components involved in adaptation. Basically, an adaptation mechanism takes sources into account to produce adaptation on some targets. The aim is to change the system in order to respect the plasticity properties developed in section 3.2: usability continuity and attractiveness improvement. We have classified the targets found in the literature into two main categories: interaction techniques modification in Section 3.4.1 and content adaptation in Section 3.4.2.

3.4.1 Interactions

The first type of adaptation that can be applied on an application is the modification of how the user is interacting with this application. As detailed in Section 2.3.3, this part refers to the action step in the "perception, cognition, action" loop of MR user interfaces. In the same Section, we define an interaction technique as a method to accomplish a task in an application which includes both hardware and software components. For MR user interfaces, three kinds of interaction techniques are proposed by Hand [Hand, 1997]: objects manipulation, navigation and application control. In each category, there are a lot of possible techniques that may perform differently depending on the context of use. Therefore, a lot of solutions propose to adapt these interaction techniques according to the context of use. Two types of adaptation can be involved:

- **Interaction technique parameter modification.** Here, the technique can be used in the current context of use and some of its intrinsic parameters can be tuned in order to make the technique optimal for the given context.

- **Interaction technique replacement.** In that case, we consider that an interaction does not fit to the current context of use, it is not optimal or not usable. Therefore, this technique is replaced by another one.

Regarding interaction technique replacement, in some circumstances it is possible that some interaction techniques cannot be used because of this context. For instance, in the paper already cited in section 3.3.1, Lindt [Lindt, 2005] shows that selecting an object with the ray technique is possible with a hand tracker but not with a speech recognizer. If this last device is the only one available, the application will have to use another selection technique such as speech commands. The interaction techniques selection depending on available devices is one of the aims of tools such as Grapple [Green and Lo, 2004], the CATHI framework [Lindt, 2009], and VIARGO [Valkov et al., 2012]. These solutions are more detailed in Chapter 4.

To continue, as said, intrinsic parameters of interaction techniques can also be adapted. For example, Dragicevic and Fekete [Dragicevic and Fekete, 2001] propose to filter the movement of

a cursor in order to stabilize it for people suffering from Parkinson's disease. Similarly, Schon et al. [Schön et al., 2004] change navigational keys sensitivity according to the user performance. If the user is getting lost in the 3D world this sensitivity is damped in order to improve his/her performance. While this method aims to facilitate navigation in 3D worlds, Celentano et al. propose to improve interactions with 3D objects [Celentano et al., 2004]. Interaction techniques are modeled with finite state machines (FSM), then user activity is monitored in order to recognize interaction patterns. The final aim is to anticipate what the user wants to do and therefore shorten his/her interaction.

Some solutions also consider both types of adaptation such as the solution proposed by Octavia et al. [Octavia et al., 2009] that considers both parameters modification and techniques replacement and these are made according to a user model. For instance, two interactions can be exchanged as shown in Figure 3.3 according to the user needs.

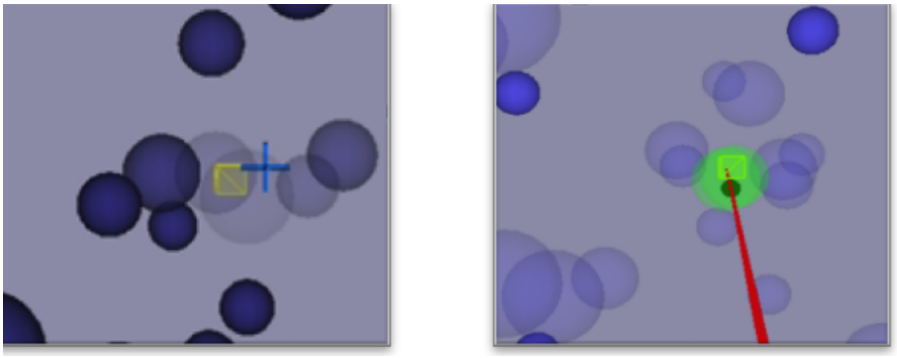


FIGURE 3.3 – Octavia et al. [Octavia et al., 2009] consider interaction techniques parameters modification and techniques replacement according to a user model. For instance, two interactions can be exchanged: a bubble cursor on the left and a 3D ray based interaction technique on the right.

The temporal Augmented Transition Network (tATN) introduced by Latoschik [Latoschik, 2002] goes further FSM by associating transitions with events, guarding constraints and temporal information. tATN are included into the framework presented in [Latoschik, 2005]. This framework enables designers to easily create multimodal interactions with context adaptation and hardware platform portability capabilities.

3.4.2 Content

An interactive system enables a user to interact with some content that can be adapted. This content can be modified according to multiple ways:

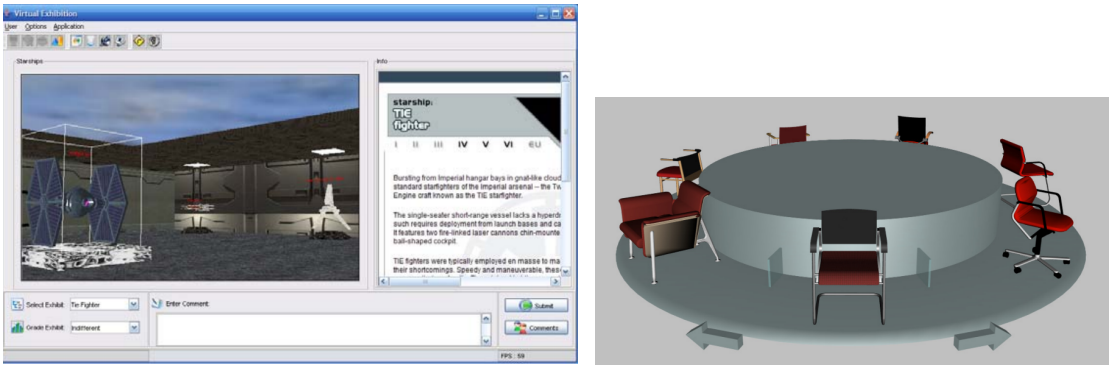
- **Content selection.** In that case the content displayed to the end-user is chosen. For example we could display only the content that satisfies the user needs, interests, preferences or one of his requests. The other available content is omitted.

- **The content intrinsic parameters.** Here, each piece of content is adapted individually. For instance, for a 3D object multiple parameters can be adapted such as a color, a material or its display quality.

- **The content presentation layout.** The content is generally displayed according to a predefined or generated layout. This layout can be adapted to the context of use, for example, modifying the layout in order to fit to screen size as shown in Section 3.3.1.

Regarding content selection, Bonis et al. [Bonis et al., 2009] propose to select the content that matches the user's preferences in order to fulfill virtual museum rooms with 3D objects as shown in Figure 3.4a. It is also possible in 2D user interface as in FlexClock [Grolaux et al., 2001] that

displays a graphic clock and a calendar only if the window is large enough. More globally this is the goal of content-based recommendation systems. As defined by Lops et al. [Lops et al., 2011], such systems try to recommend items similar to those a given user has liked in the past. In the same paper, a literature review of such solutions is provided.



(a) (b)

FIGURE 3.4 – Two examples of how the content of an application can be adapted to the context of use. (a) Bonis et al. [Bonis et al., 2009] propose a method to select the content that matches the user's preferences in order to fulfill virtual museum rooms with 3D objects. Here the museum is fulfilled with science-fiction artifacts (b) Dachselt et al. [Dachselt et al., 2006] propose a solution to adapt the intrinsic parameters of the content according to the context of use. Here, the chair's color is chosen according to the favorite color of the user.

Changing intrinsic parameters of some objects is one of the possibilities offered by Dachselt et al. [Dachselt et al., 2006]. Chair's color is set to the user's favorite color in one of the given examples shown in Figure 3.4b. In the same way, in the e-commerce application [Chittaro et al., 2002] already cited in section 3.3.2, the size of user's favorite products is changed in order to get more visible. Visibility of important information is also the issue of Julier et al. [Julier et al., 2000] who aim at finding the correct opacity for each information display according to its consistency in the current context. Content quality can also be adapted such as an image resolution or the number of vertices and polygons of a 3D mesh. In such a case, we talk about level of detail (LOD). The aim of LOD is to improve or decrease the complexity of a content according to a metric. Creating these different levels of detail for a 3D content can be possible thanks to decimation algorithms such as the method proposed by Schroeder et al. [Schroeder et al., 1992]. This algorithm aims to reduce the number of triangles of a mesh while preserving its topology. Progressive meshes by Hoppe [Hoppe, 1996] are another way for dealing with LOD. It offers a continuous and lossless representation of a 3D mesh that handles different issues such as the smooth choice of LOD, progressive transmission and mesh compression. The metric taken into account in the LOD choice can be one of the adaptation sources detailed in section 3.3. This is the case for the two examples cited in section 3.3.1 where the metric is the hardware capabilities for adapting a 3D agent [Kim et al., 2004] and a 3D city model [Royan et al., 2007]. Adapting the intrinsic parameters of some objects from the VE can also be used in order to improve the physiological acceptability of a MR user interface. For instance, in order to limit the visual discomfort named "frame cancellation" that occurs with stereoscopic screens, Ardouin et al. [Ardouin et al., 2011] propose to clip the objects outside a free-conflict area. This phenomenon is more described in Section 5.3. In the same way, in order to deal with cybersickness, Fernandes and Feiner [Fernandes and Feiner, 2016] propose to dynamically adapt the camera field of view according to its speed and to its angular velocity.

Some examples for adapting the layout of the displayed content can also be found. For instance ENTER [Guinan et al., 2000] can update a scene layout with functions like "Swap" "Move" or "Rotate" called on 3D objects by an adaptation engine that takes into account the user-profile. Layout adaptation is further addressed in [Lee and Green, 2005] by proposing an automatic layout framework. By defining layout policies, a designer can manage a scene by taking into account different constraints such as the user behaviour. In the example cited in Section 3.3.1 and shown in Figure 3.1b, Dachselt et al. [Dachselt et al., 2006] use a rule based system in order to switch

between different content layouts according to the context of use. To continue, Stuerzlinger et al. [Stuerzlinger and Smith, 2002] propose a method for automatically grouping scene objects according to the user's intent and the way humans perceive groups.

3.5 Objectives and challenges

According to the previous Sections, we propose a problem space for the development of plastic MR user interfaces. As explained in Section 3.2, for a plastic user interface, adaptations can occur at different time such as runtime or between sessions. The adaptation controller can be the user, the developer/deployer or the system. We have also identified two adaptation means for the modification of an interactive system: recasting and redistribution. In Section 3.3, we have identified three different adaptation sources for MR user interfaces: hardware, users and their environment and data. In Section 3.4, we have also identified two adaptation targets: content and interaction techniques. This space is given in Figure 3.5.

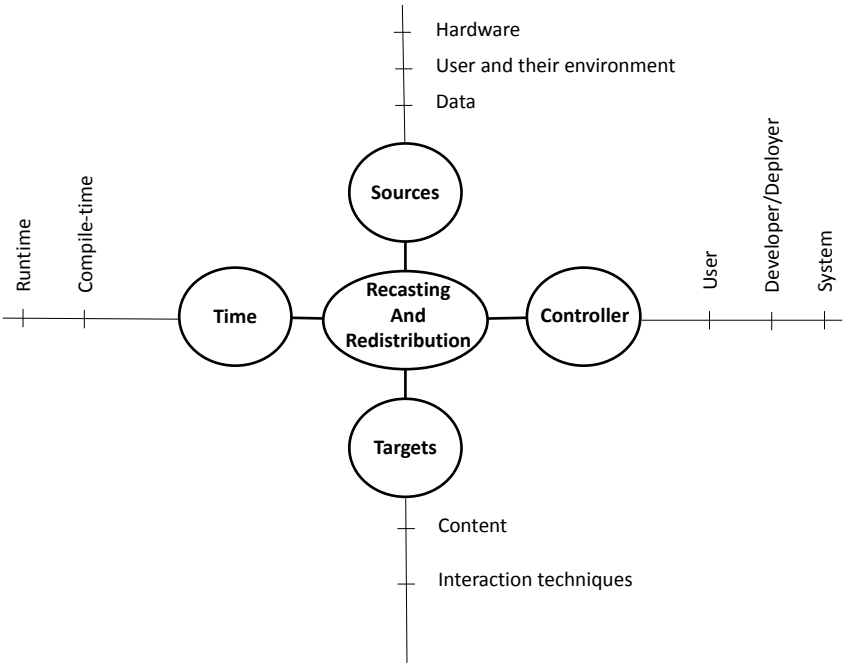


FIGURE 3.5 – The problem space for the development of plastic MR user interfaces.

In the context of this thesis, our goal is to propose software models and tools dedicated to the creation of plastic MR user interfaces. The creation of such solutions creates new challenges. The first difficult point is the representation and the detection of the context of use. The adaptation mechanisms need to be able to get enough information about the users, the hardware and the data in order to perform application modifications. To continue, designing an intelligent adaptation engine that will find and perform the most suited modification of a system is another challenging step. Managing hardware is a really complex stage regarding ubiquitous computing and the growing number of new devices in the context of MR user interfaces as explained in Section 2.3. Moreover, another challenge relates to the right time and the way to react to these new configurations of the system. If the application allows adaptations at runtime, it is substantial to choose the good time for adaptations in order to avoid changing the application too often. The risk of too many modifications is to get the opposite by disturbing the user experience.

From the space problem of plasticity and these different issues we can identify the main challenges that we have to face for the creation of models and tools for the creations of plastic MR

user interfaces. These challenges are represented as a list of requirements that our solutions have to cover:

R1 Deal with the main adaptation sources or be extendable to do so. Therefore, we should provide solutions for the modelization and detection of hardware, users and data. At least, we must provide the possibility to extend our solutions to integrate a missing context model. Each model must be able to represent each piece of context with a small granularity.

R2 Deal with the main adaptation targets or be extendable to do so. The proposed adaptation process must be able to impact the application content and the interaction techniques.

R3 Support the two means of adaptation of plasticity. The adaptation process must be able to modify the application: this is recasting. It must also be able to change its distribution across different dimensions such as platform, display and user: this is redistribution.

R4 Ensure code portability. The created solution must be available on many operating systems (mobile and desktop). Indeed, code portability is a necessary condition of the plasticity property. Moreover, the toolkit needs to be possibly interfaced with the main 3D frameworks and must not depend on a particular one. Indeed, as detailed in Section 2.4 each developer may have his/her own code database for input, rendering and physics management in a particular framework and may not want to redevelop them all to be able to use particular functionalities.

R5 Perform dynamic and static adaptations. The adaptation mechanism is static if it occurs between sessions and is dynamic if it can also occur at runtime. To ensure usability continuity the system needs to be able to detect a context modification such as a device plugged or a new user configuration at runtime and to update the interface accordingly.

R6 Handle user, developer/deployer and system adaptations. The system automatically chooses the best 3D user interfaces according to the adaptation process, this is adaptivity. However, the user must be aware of which adaptation occurs and to be able to modify the aspect of the interface with a set of predefined parameters, this is adaptability. Our adaptation process must be able to decide itself which adaptations are needed in a particular context of use but it also has to give to the end-user a control over the decided modifications. The developer/deployer of the application must also be able to control and to configure the adaption process.

R7 Be flexible, easy to use and to extend for developers and designers. That is why an authoring tool is required to create and configurate MR user interfaces. Indeed, providing an authoring tool is the only way to bring a plasticity toolkit to the consumer market. According to Myers et al. [Myers et al., 2000], the toolkit and its authoring tool must have a low threshold (easy to use and to learn) while having a high ceiling (how much can be done using them).

3.6 Current Software Solutions

In this Section, we recall the main software solutions that deal with Plasticity and Adaptation for MR user interfaces. Our goal is to highlight the advantages and the limitations of current tools in order to propose an original solution. Here, we do not precisely describe each solution but they will be more described in the following Chapters.

For dealing with plastic 2D user interfaces the most common approach consists in using Model-Driven Engineering (MDE). For instance, the CAMELEON conceptual framework [Calvary et al., 2002b] or the UsiXML based markup language [Limbourg et al., 2004] are two MDE approaches for the development of plastic 2D user interfaces. Multiple uses of such approaches have demonstrated that MDE can cover the different requirements of the development of plastic 2D user interfaces. Nevertheless, in the field of 3D, MDE has not been deeply explored and has not been successfully

applied for the development of plastic MR user interfaces that meet all requirements described in Section 3.5. For instance, handling dynamically the wide variety of MR devices and handling the generation of adapted 3D worlds are not really covered by these solutions. Moreover, for now MDE lacks authoring tools for developing applications. Thus, today, solutions based on this concept are not widely used by developers.

The first category of software tools that consider adaptation for MR user interfaces is the category of middlewares already introduced in Section 2.4.3. Solutions such as Collaviz [Dupont et al., 2010], MiddleVR and VR Juggler [Bierbaum et al., 2005] propose to create configurable MR user interfaces. By editing a configuration file manually or with a graphical tool, the created interfaces can target a wide variety of hardware platforms and can support features such as clustering and collaboration. Such solutions mainly focus on adapting the interaction techniques according to the hardware platform and do not consider adaptation according to the users and the data. They also only support the re-configuration between sessions, context-changes cannot be handled at runtime. In the same way, Grappl [Green and Lo, 2004] also include such features and has the advantage to also cover content presentation as an adaptation target by integrating layout policies [Lee and Green, 2005]. Collaviz has also the particularity to integrate a MDE approach [Duval et al., 2014]. With this approach, the deployment of 3D CVEs on various software and hardware platforms can be automated through code generation.

Such solutions go further by including automatic adaptation mechanisms such as Viargo [Valkov et al., 2012] and the CATHI framework [Lindt, 2009]. However, they do not take user as an adaptation source and they cover redistribution only partially (only clustering is handled).

In the context of Web3D, AMACONT [Dachselt et al., 2006] proposes a solution for the creation of adaptive 3D user interfaces. Hardware is considered as an adaptation source but the possible handled platforms are limited to desktop environments. The user properties can be taken into account during the adaptations process.

	Adaptation Sources			Adaptation Targets		Adaptation Time		Adaptation Controller			Adaptation Means	
	Hardware	Users	Data	Content	Interaction Techniques	Runtime	Compile Time	User	Developer/ Deployer	System	Recasting	Redistribution
VR Juggler	Yes	No	No	No	Yes	No	Yes	No	Yes	No	Yes	Partially (Clustering)
Collaviz	Yes	No	No	No	Yes	No	Yes	No	Yes	No	Yes	Partially (Clustering, Collaboration)
MiddleVR	Yes	No	No	No	Yes	No	Yes	No	Yes	No	Yes	Partially (Clustering, Collaboration)
Grappl	Yes	No	No	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No
Viargo	Yes	No	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Partially (Clustering)
CATHI Framework	Yes	No	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Partially (Clustering)
AMACONT	Partially (Few devices)	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	No

TABLE 3.2 – A classification of software solutions for the creation of plastic MR user interfaces according to the problem space of plasticity.

In Table 3.2, we provide a classification of these solutions according to the issues they can solve on the problem space of plastic MR user interfaces presented in Figure 3.5. As we can see, for now none solution is really designed to deal with all the plasticity issues. We can also highlight that for now, in the field of MR user interfaces, taking data into account during the adaptation process and considering redistribution as an adaptation mean are not very explored. Our goal with this thesis is to fill that gap by providing models ans development tools that handle the different issues of plasticity.

3.7 Conclusion

In Chapter 2, we have identified the design space for the development of MR user interfaces. From this survey, we can conclude that without appropriate tools the development of such applications can be difficult for a developer who wants to satisfy each user needs or each platform capabilities.

In this Chapter we have presented the plasticity concept: it is the capacity of an interactive system to withstand variations of both the system physical characteristics and the environment while preserving its usability. Considering such properties for the creation of a MR user interfaces is one way to handle the issues presented in Chapter 2. Indeed, considering plasticity for the development of a MR user interface can induce a lot of advantages for the developer such as the reduction of the development and maintenance times and costs and the possibility to distribute the application widely. It also benefits to the end-user as he will get an application that corresponds to his/her needs and that provide him/her usability continuity on his/her different interaction platforms.

In Section 3.6 we have shown that some software solutions exist for the creation of MR user interfaces but they do not solve all the issues induced by the plasticity concept. Therefore, our goal with this thesis is to propose new models integrated into a new software solution for the creation of MR user interfaces. In Section 3.5 we have identified the requirements that these models and this software solution have to cover.

The next Chapters are organized in order to present successively how our solution can target the different adaptation sources (Devices in Chapter 4, Users in Chapter 7 and Data in Chapter 8). Only Chapter 6 makes a focus on one adaptation mean: Redistribution.

Chapitre 4

Device Adaptation

4.1 Introduction

As detailed in Chapter 3, taking into account the hardware configuration as an adaptation source is a very important requirement for any MR application that wants to be distributed widely. Indeed, until recently most applications were developed for a specific target platform. Today uses have evolved, people have now access to a wide variety of different hardware platforms: desktop, mobile and immersive platforms. In the specific case of immersive ones, as explained in Sections 2.3.1 and 2.3.2, the hardware environment can really differ from a platform to another one. The set of possible output and input devices is very large. That is why developing manually a version for each possible configuration is no longer a viable option. One common way to solve this kind of issue is to support static reconfiguration of the system. However, such solution would require the intervention of the end-user for modifying the configuration of the system through a configuration file or through a graphical tool. For non expert users, performing this configuration can be difficult. A better solution consists in automatically detecting the end-user configuration in order to automatically adapt the application. Such solutions could also detect the variations of the hardware configuration at runtime. For instance, if the user plugs or unplugs a device at runtime.

In order to help the developers in the creation of plasticity MR user interfaces, we have initiated the creation of models, concepts and algorithms that aim to deal with the different plasticity issues. The focus of this Chapter according to our design space problem is detailed in Figure 4.1. This chapter only presents how recasting is handled, redistribution is discussed in the next Chapter. The device model and the application component model that we present in this Chapter have been published in [Lacoche et al., 2015a].

This Chapter is structured as follows. First, in Section 4.2, we detail the related work that deal with adaptation to the devices. Second, in Section 4.3 we give an overview of our models and we present how they can take the device configuration as an adaptation source. These models and concepts are integrated in a software solution: 3DPlasticToolkit. This Chapter presents the basis of this software solution with a focus on how it takes into account devices as an adaption source. To continue, in Section 4.4 we present our model for the description of input and output devices. In Section 4.5, we present our application component model and we describe how it can be used to perform interaction and content adaptations. In Section 4.6, we present the task system used in 3DPlasticToolkit and how we describe an application with a set of high level tasks. Next, in Section 4.7, we detail how dynamic recasting is handled with 3DPlasticToolkit with an internal adaptation process based on scoring mechanisms, and with a user interface that lets the end-user control the adaptation behaviors. We conclude in Section 4.8 and in the next Chapter we present three use cases where our solution is used to perform interaction adaptation and content visualization adaptation.

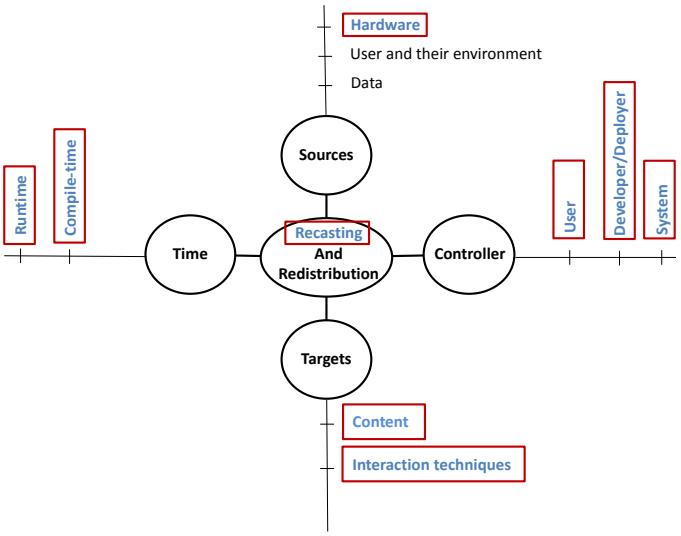


FIGURE 4.1 – Regarding our design space problem of plasticity for MR user interfaces, this chapter is interested in adapting the interaction techniques and the content visualization according to the devices. The basis of our models are also described in this Chapter so we also describe how we can handle the different adaptation times and adaptation controllers.

4.2 Related Work

For taking into account devices as an adaptation source, related work can be classified into two main categories. First, solutions that aim at classifying and modeling hardware devices. Second, solutions that use such classifications in order to adapt an application. Some solutions can be included in the two categories as they both propose a classification and an adaptation mechanism.

4.2.1 Device modeling

In order to take the device context as an adaptation source, a lot of solutions aim to categorize and model interaction and display devices. These categorizations can have multiple goals. First, it can help a designer or a developer to find the appropriate device for a particular situation during the conception of an application by giving him some guidelines. In that case, the application is developed for a particular device and is not compatible with equivalent ones. Second, such modelization can also be directly integrated into a software component in order to perform data comparison between the devices capabilities and the needs of an application. Such solutions can be used in order to perform static and dynamic adaptations to devices.

Several classifications exist about how to perform static or automatic input device selection. First, Buxton [Buxton, 1983] introduced a taxonomy that separates input devices into different axes such as the number of degrees of freedom (DOF), the property sensed (position, motion, pressure), the motor capacity used and the presence or not of a physical intermediary for the interaction. In the taxonomy shown in Figure 4.2, some common devices are classified such as mouses, trackers and joysticks. In an extension of this taxonomy, Mackinlay et al. [Mackinlay et al., 1990] established a difference between absolute and relative values and separated translation and rotational axes. This model also indicates which axes are acquired instead of just giving the number of DOF. Such taxonomies only expose information about the input data provided and nothing on how the data is acquired, its semantic or its limitations like values boundaries. Therefore, Lipscomp and Pique [Lipscomb and Pique, 1993] introduced a graph representation of interaction devices that gives the physical characteristics of each device and so a more precise description on how the data is acquired and their limitations. For example, it differentiates bound and unbound inputs as well as isotonic, isometric and elastic. It can specify if the device is on a table, held up or body mounted.

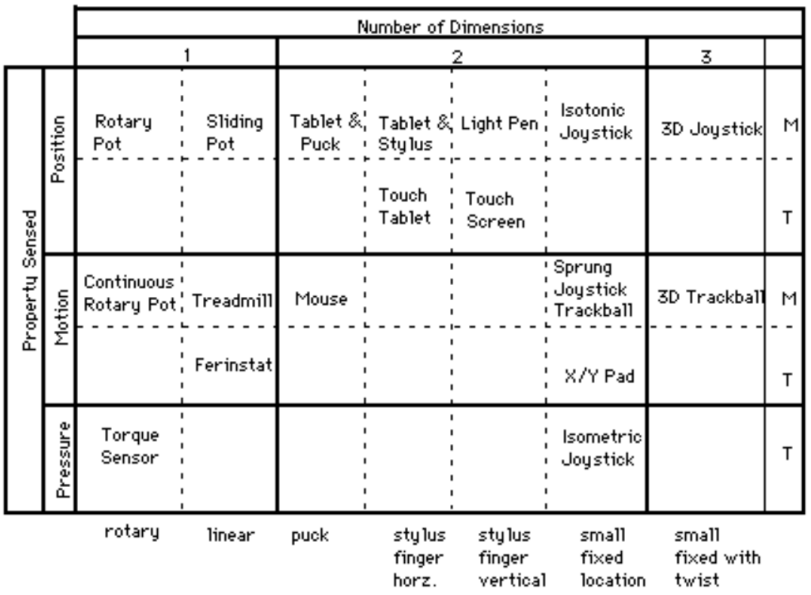


FIGURE 4.2 – The taxonomy of input devices proposed by Buxton [Buxton, 1983].

Recent devices are more complex than the ones available at the time of these taxonomies. Indeed, they may provide multiple inputs that are more complex than just a tracking value. For instance, the first Microsoft® Kinect™ is able to acquire 20 positions and rotations (skeleton joints), three image streams : depth, color and IR and can also perform gesture and voice recognition. Some devices can also provide outputs capabilities such as haptic, tactile or visual outputs. DEVAL [Ohlenburg et al., 2007] proposes a more actual model. As explained in Section 4.2.2, DEVAL is a device abstraction layer for VR and AR that hierarchically structures inputs and outputs units according to their properties. For example, it first differentiates continuous and discrete inputs. Then, these inputs are structured according to the modality they acquire and the number of degrees of freedom they provide. The DEVAL classification of inputs is given in Figure 4.3. A

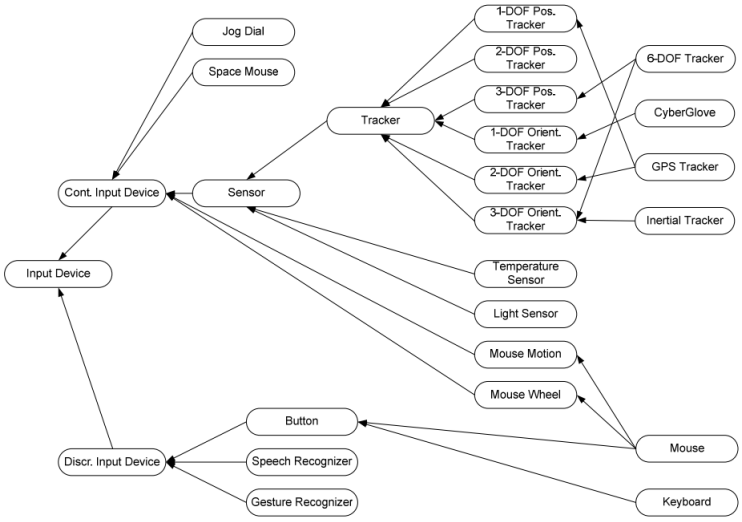


FIGURE 4.3 – The hierarchy for input devices proposed by DEVAL [Ohlenburg et al., 2007]. This hierarchy includes a wide variety of input device units including trackers, buttons and gesture recognizers.

device is then a composition of several input and output units. The model includes a wide variety

of device units including trackers, buttons, haptic feedback, speech and gesture recognition. In contrast to the other models, less common sensors are also included such as light and temperature sensors. Anyway, this model and the previous ones only define devices with input and output data while it would be interesting to also expose their physical properties. Indeed, describing the data is not enough, the physical properties of the devices in the real physical workspace and their internal properties, such as refresh rate or accuracy, must be known if possible. In case of automatic adaptation, these properties are needed to ensure usability continuity. For example, the position of a sensor impacts the reference coordinate system of any captured position. The model introduced by Lipscomp and Pique [Lipscomb and Pique, 1993] tries to fulfill this issue but, as said before, it does not provide enough properties to describe well today's devices. The device model proposed by Lindt [Lindt, 2009] extends DEVAL to add this kind of meta-data. For instance, in Figure 4.4, three 6-DOF trackers are differentiated by their accuracy, usage and update rate. To do so, three kinds of meta-data can be added to a device unit instance. First, static devices properties that do not change over the time like the weight of a HMD. Next, configurable devices properties depend on the device setup like the smoothing factor of a tracking device. Finally, runtime properties include performances and device states. The classification into three categories can be discussed because the associated category of a property may change over different devices. For instance, the resolution of an image acquired by a camera can be configurable for some devices such as the Microsoft® Kinect™ while for most of them it is a static value. Moreover, the set of properties introduced in the model does not include enough properties to precisely describe the capabilities of each device. For instance, values boundaries and devices position in the real world are not included. Another model that includes such meta-data is included in the MPEG-V standard [Han et al., 2012]. It includes a description of input and output devices even the less common ones such as wind, light and scent output devices. Nevertheless, it does not include description for display and sound output devices. Moreover, the range of meta-data included is not enough to really expose all devices capacities and limitations. The representation of the devices in the real world is also not included.

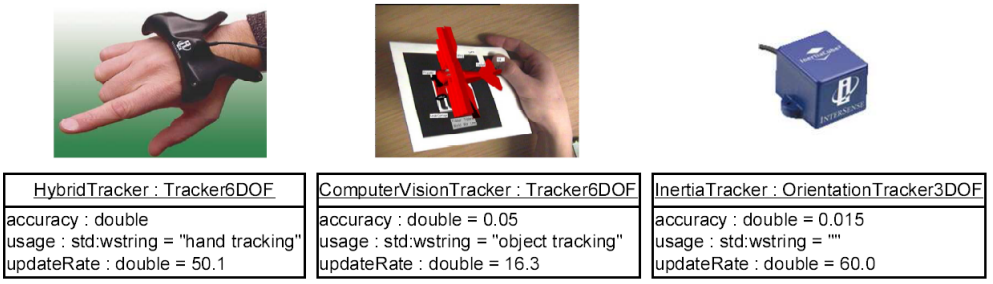


FIGURE 4.4 – In the extension of DEVAL [Ohlenburg et al., 2007] proposed by Lindt [Lindt, 2009], meta-data about internal and physical properties of inputs devices can be added. Here, three 6-DOF trackers are differentiated by their accuracy, usage and update rate.

From these different device models, we can conclude that none of them really provide enough information in order to adapt an application to all device properties. It would be interested to propose a new model that solve the issues of current models. Such a model must consider input and output devices, must provide devices capabilities and devices limitations and must also give the physical representation of devices .

4.2.2 Device adaptation solutions

With such device models, software solutions can take into account the device context of use in order to adapt the interaction techniques of an application or its content presentation. Here, we focus on recasting, solutions that consider also redistribution are described in the next Chapter. We separate the software solutions into two categories: solutions for 2D user interfaces and solutions for MR user interfaces.

2D Solutions

For dealing with plastic 2D user interfaces the most common approach consists in using *Model-Driven Engineering (MDE)*). The **CAMELEON** conceptual framework [Calvary et al., 2002b] proposes to structure the development process of a user interface into four steps where each step is represented by models:

- Task and concepts (T&C): a high level description of the user interface with the tasks that the user will perform and the concepts that he/she will manipulate.
- Abstract UI (AUI): it defines the interaction spaces and the navigation between these spaces. There is a correspondence between a workspace and a task. Furthermore, a workspace that corresponds to an abstract task can include the workspaces that correspond to the subtasks.
- Concrete UI (CUI): it concretises the AUI with concrete interaction objects including widgets and navigation interfaces.
- Final UI FUI: the generated user interface running on a computing platform by code interpretation or execution.

The reconfiguration of the user interface consists in applying transformations at each of these steps according to the context of use in order to ensure usability continuity. **UsiXML** [Limbourg et al., 2004] is an XML based markup language for the development of plastic user interfaces that is conform to the CAMELEON framework and can be used by designers and developers. It proposes a language for the creation of the different models at each development step of CAMELEON. It also introduces a language for the creation of rules for transforming the models according to the context of use. This context of use, including plaform, user and environment descriptions, can also be modeled with UsiXML.

In the field of pervasive computing, the **Dynamo Framework** introduced by Avouac et al. [Avouac et al., 2012] uses proxy models and interaction models to maintain a mediation chain that defines multimodal interaction techniques. The system is able to check the context (services, devices) at runtime and reconfigure itself dynamically. These models enable developers to focus on interaction techniques development independently from the concrete devices used. However, in order to avoid wrong associations between interaction techniques and devices, designers or developers have to create pre-defined mediation chains (interaction models). It needs *a priori* knowledge on how the devices will be used and is a lesser automatic approach than describing at a fine grain each device to perform the associations. Moreover, the framework does not include yet the possibility for the user to reconfigure the system and to express his preferences.

MR solutions

The first category of software solutions that aim to deal with devices as an adaptation source is the category of *abstraction layers*. Such solutions group devices into categories according to the data they can acquire or provide. This categorization directly exploits the taxonomies of devices introduced in Section 4.2.1. According to Lindt [Lindt, 2009], device abstraction layers have three main objectives:

- They encapsulate and hide the complexity of concrete interaction devices. They avoid the developer from using a complex SDK or library in order to use a particular device.
- They avoid an application from being dependent to particular devices.
- They support the exchange of a device by another one that can acquire or provide the same data with a static reconfiguration.

In this category, **OpenTracker** [Reitmayr and Schmalstieg, 2001] and **ICON** [Dragicevic and Fekete, 2001] are two abstraction layers based on dataflow architectures that can combine and filter input devices in order to connect them to different actions of an application. ICON has the advantage over OpenTracker to be dynamic, as the reconfiguration can be done at runtime. ICON also provides a graphical user interface in order to configure the system. Unfortunately, it does not handle natively any MR devices. Conversely, OpenTracker has been designed to handle common

MR devices such as trackers. One abstraction layer commonly used in MR is **VRPN** [Taylor et al., 2001]. VRPN gives access to a wide variety of peripherals such as trackers, buttons and force feedback devices. VRPN is based on a client-server architecture with a network-transparent interface: a server provides access to registered devices while the application is the client that uses these devices. VRPN is still very used because it already includes a lot of devices and because it is easy to extend and therefore new devices can be quickly integrated. Abstraction layers only give the possibility to statically configure the routing between concrete devices and application components. Most of the time, this configuration has to be done by a developer/deployer. Therefore, such solutions do not support the automatic replacement of a device by an equivalent one with the same capabilities. It needs the intervention of someone to reconfigure the system. If the new device does not provide the same capabilities it can result to the impossibility to ensure the same usability with a simple reconfiguration. Moreover, they do not include any built-in interaction techniques and component to modify the visual content of an application. Therefore, our solution needs to include a device abstraction layer to have an easy access to different devices but it is not enough to handle the different plasticity issues.

We can find in literature this kind of solution: *MR toolkits* that are built upon abstraction layers. First, **Viargo** [Valkov et al., 2012] is a generic interaction library. In the library, input devices are abstracted by device units that provide events to interaction metaphor components. These components process the events in order to update the state of the application. If a device is exchanged at runtime, the interaction metaphor is not disturbed as long as the new device events are compatible with it. It includes different built-in interaction techniques for navigating and manipulating. Nevertheless, Viargo only considers hardware as adaptation source and the interaction techniques as adaptation target. Moreover Viargo uses its internal scene graph (the "state system") that is synchronized with a 3D framework. This kind of method does not offer to developers the new capabilities and performances for graphics and physics of modern 3D frameworks. Next, as Viargo, the **Grappl library** [Green and Lo, 2004] focuses on adapting the interaction techniques to the hardware context. An extension has been proposed in order to also impact the content presentation with layout policies that can take into account the hardware context [Lee and Green, 2006]. Grappl proposes an association between a high level task and a set of compatible interaction techniques. At runtime, depending on the available devices, Grappl instantiates a compatible interaction technique for each needed task. The selected interaction technique is the one that best matches the task. To refine the choice, a set of parameters is associated to each task by the designer or by the developer such as the interaction style needed. However, Grappl does not solve the conflict that may happen when different interaction techniques provide the same interaction possibilities. In case of such a conflict it could be interesting to give the choice to the developer or to the user. To continue, even if the user interface is constructed at runtime, Grappl does not give any solution to deal with context modifications at runtime such as a device unplugged. Moreover, as Viargo, Grappl also uses its own internal graph system. Last, the most complete solution seems to be the **CATHI framework** [Lindt, 2009]. As Grappl, it also aims to adapt the application according to the hardware configuration. CATHI also creates the best user interface according to a set of high level needed tasks and to the current context of use. CATHI represents a 3D user interface as a graph of interconnected components. The designer selects the high level tasks to add to this graph. Then, according to the encountered context at runtime, the most adapted low level tasks are connected to the graph. A low level task represents an interaction technique. It is based on an interaction modality and determines a set of device units that is needed for this interaction modality to be usable. This low level task is instanciable if all these units are available at runtime. A scoring system is used to avoid conflicts between equivalent possible low level tasks. The scores are customized by developers or designers with rules that can take the context into account. At runtime the graph with the higher level score is selected as the current 3D user interface proposed to the user. Contrary to Grappl, CATHI can handle context modifications at runtime by recreating the interface graph when a modification happens. For now, the toolkit only takes into account device configuration and weather conditions as context information. For now, the only possibility given to the user to change the adaptation behaviors is to modify the set of rules, which can be difficult for non-expert users. Editing these rules is the role of the developer/deployer of the application. Moreover content presentation and redistribution are not covered by the framework yet.

The last kind of solutions consists in using *MDE*. Indeed, Gonzalez-Calleros et al. [Gonzalez-Calleros et al., 2009] propose to extend *model-based user interface* development from 2D to 3D in order to handle plasticity. They introduce a solution based on **UsiXML** [Limbourg et al., 2004] and **CAMELEON** [Calvary et al., 2002b]. With model transformations rules described with UsiXML, user and hardware adaptations can be integrated in the user interface development process. Nevertheless, the solution focuses on the creation of adaptable 3D widgets and the final user interface is generated as a VRML or X3D file. Therefore, the interaction part is limited, no solution is included to create interaction techniques independently from any concrete device. In the same way, Duval et al. [Duval et al., 2014] propose to use MDE for the development of 3D Collaborative Virtual Environments (3DCVE). With this solution the configuration of the 3D content and the deployment on various software and hardware platforms can be automated through code generation. However, the configuration is static, it does not includes mechanisms to take dynamically the context into account.

To finish, one important missing feature in these different solutions is the lack for the end-users to check and control the adaptation behaviors. Most solutions presented here can only give the adaptation control to the system or to the developer/deployer. Indeed, even if the created application is considered as the best one by the system, the user may want to try another interaction technique or another available device. We have to provide the end-user with the possibility to modify the application adaptations at runtime. In the same way, most solutions do not yet take user adaptation into account. As each user has his own profile and preferences, such a solution must take these information into account during the adaptation process. This issue is discussed in Chapter 7.

4.3 Overview

Our contributions are models, concepts and algorithms that are integrated in a software solution named 3DPlasticToolkit, aim at satisfying the requirements exposed in Section 3.5. An illustrated overview of the components of the toolkit is provided in Figure 4.5.

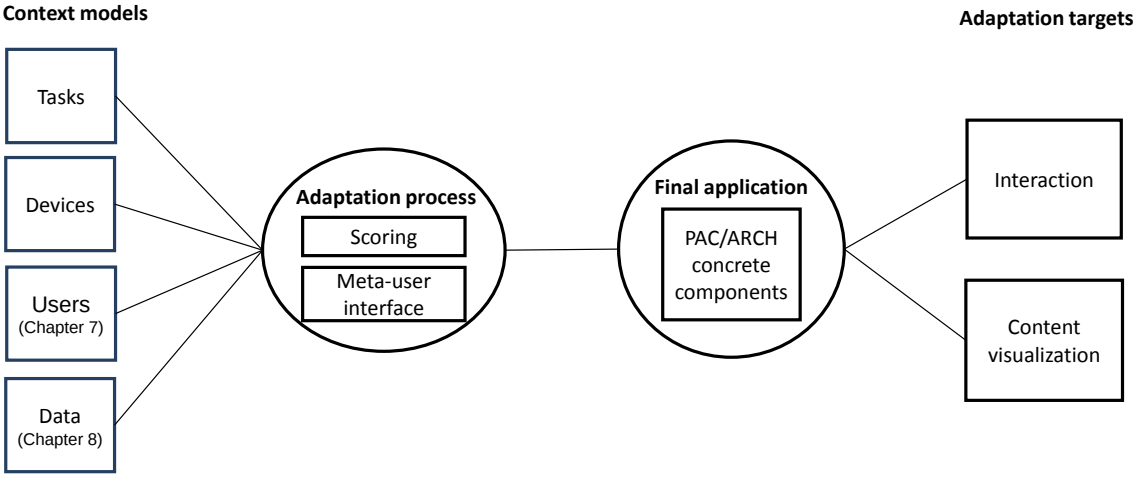


FIGURE 4.5 – The 3DPlasticToolkit overview.

In order to deal with the main adaptation sources and complete **R1**, these adaptation sources are modeled into the system and exposed to the adaptation process. They correspond to the context models represented in Figure 4.5. These models can be edited and extended by any developer. In this Chapter we describe how tasks and devices are modeled in 3DPlasticToolkit while Chapter 7 focuses on our user model and Chapter 8 on our data model.

The device model represents the hardware configuration at runtime. It exposes the different input and output capabilities handled by the available devices. More than just describing the data acquired or provided by devices, the model also exposes their properties, limitations and representations in the real world. These information are needed to ensure usability continuity.

The task model represents at a high level the application behaviour and possibilities. An application is represented as a collection of high level tasks. This collection is provided in a configuration file by the application developer or designer. Tasks can also be added and removed at runtime using the toolkit API. Tasks have to expose compatible concrete application components that will be possibly instantiated in the final application. A concrete application component is a software element that can be instantiated in the final application in order to accomplish a task. For instance, it can correspond to the code for a 3D widget or an interaction technique. The compatibility between a task and a concrete application component is ranked: each compatible component is associated with a score. This score can be modified at runtime according to the context. Additional properties can also be included into the tasks descriptions in the configuration file.

As shown in Figure 4.5, we propose a new model for creating application components to impact all the adaptation targets and therefore cover **R2**. This model proposes an extension of PAC [Coutaz, 1987] in order to create application components independent of any concrete interaction device and that can support alternative concrete representations. This concrete representation implements how the component will be represented in the 3D world. The model decomposes an application component into different facets in order to decouple component semantic, view and devices management. The model also ensures an independence from any targeted 3D framework in order to satisfy **R4**.

In order to take into account the adaptation sources and impact the different adaptation targets, as in the CATHI framework [Lindt, 2009], we propose to use a scoring algorithm that will drive the application component instantiations and modifications. As shown in Figure 4.5, this scoring algorithm is one part of our adaptation process. This adaptation process handles local adaptations in order to support recasting. In Chapter 6, we propose a redistribution process that allows us to also cover this adaptation means in 3DPlasticToolkit. Therefore **R3** is covered. This core component of our system receives the different events corresponding to the changes in the context of use and can react accordingly at runtime. Therefore dynamicity is supported and **R5** is covered. Thanks to this mechanism a good usability of the application is always ensured.

One important missing capability in the state-of-the-art solutions is the possibility for the end-user to check and modify the adaptation behaviors at runtime. To solve this issue and cover **R6**, our toolkit contains a built-in configuration user interface that can be shown and hidden at runtime: the meta-user interface. As shown in Figure 4.5, this is the second part of our adaptation process. For instance, the meta-user interface allows the end-user to replace concrete application components or switch from an interaction device to another one. As shown in Chapter 6, this integrated interface is also used to control the redistribution process of an application developed with our models. In Chapter 7 we also show how it can be used to modify the user model. Therefore, **R3** can be completely covered.

For now we only partially cover **R7** by providing some graphical authoring tools (device and task configuration), a collection of implemented interaction techniques and visual effects and some integrated devices.

4.4 A New Taxonomy for the Description of Interaction Devices

The first adaptation source that we consider and that we model in the system is the hardware configuration: this is the device model. In this Section, we present this model and its concrete implementation in 3DPlasticToolkit. We consider a platform as a hardware configuration composed of several devices and computing units on which an application can be launched. Input devices are considered as the devices that can collect data from the real world such as a position, a pressure on a button or a sound acquisition. Output devices restitute computer generated values to the real world such as an image on a screen or a vibration. This model exposes to our adaptation process described in Section 4.7 which devices are available, what are their capabilities and what are their limitations. This model is needed to perform device selection and adaptation in order to make an application available on a wide variety of platforms. It includes the properties to describe a device and its associated units. Our device model aims to solve the issues of the models presented in

Section 4.2.1 through the use of an accurate description of the devices capabilities and limitations, and of their representation in the real world.

Regarding the description, in the model, we consider an interaction device as a complex entity that may acquire input(s) and render output(s) and that has a representation in the real world. As shown in Figure 4.6, the model represents a device as a collection of inputs, outputs and physical objects entities. A device is defined by its name, the name of the SDK used to manage it, its index in the case we use different instances of the device, and a boolean that indicates if it is plugged or not.

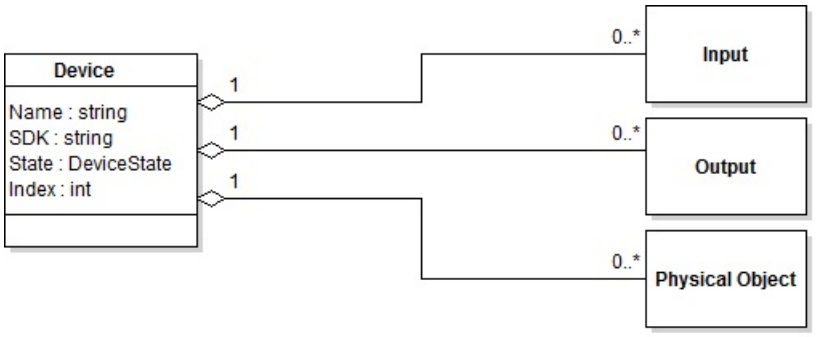


FIGURE 4.6 – A device is a set of inputs and outputs as well as a collection of physical objects for its real representation.

The set of physical objects describes the representation of the device in the real world. For example, it gives properties such as a 3D representation of the device if we want to represent it in the virtual world. Its position gives the possibility to a developer to automatically adapt the coordinate system of the tracking values. Moreover, all inputs and outputs are associated to a physical object, this information may help to select an input or an output unit used by an interaction technique. Indeed, for an interaction technique, inputs and outputs that correspond to the same physical object can be preferentially selected. The goal would be to minimize device switchings as well as the homing time of the GOMS model (Goals, Operators, Methods, and Selection rules) [Card et al., 1983]. The homing time corresponds to the duration that the user spends to move his hand from one device to another one during the interaction.

Regarding input description, the goal is to describe all possible acquired signals that are currently used in 3D applications. We established a list of three categories that gather these possible inputs:

- Real values: the most common value type used. It refers to continuous values acquired by trackers, sensors, touch-screens, etc. It can represent a position or a rotation as well as more original values such as a temperature or the lighting intensity of a room.

- Discrete values: taken into a set of predefined ones. For example a button press or a button release event, a symbolic gesture or a vocal command.

- Generic streams: they continuously provide arrays of values. This category is divided into multiple subtypes with more specific properties (image streams, EEG signals, sound acquisition...).

In the model, these different input types include a description of the acquired data, their properties and limitations, and information about how these data are acquired by the device. To do so, we propose to describe each input into two entities. The "data description" ensures the first need, and the "technology" ensures the second one.

For the two entities of the real value input type described in figure 4.7, the description reuses the most important properties of previously described taxonomies, especially [Lipscomb and Pique, 1993] and [Mackinlay et al., 1990], while adding some new ones. Regarding the new properties, three

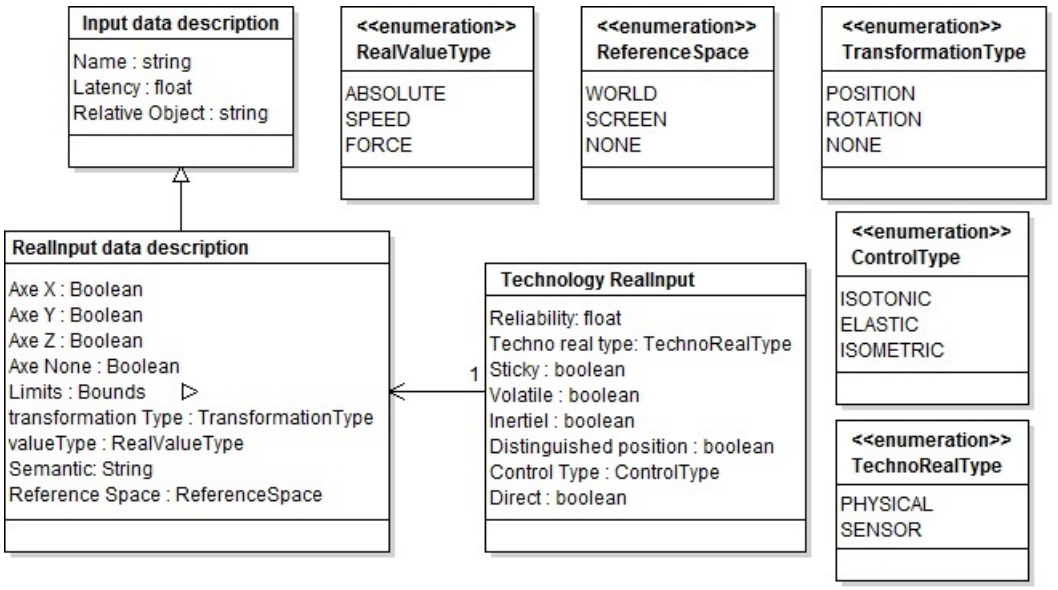


FIGURE 4.7 – The description of real value input with two entities: the "data description" and the "technology".

booleans describe the axis on which the value is defined as proposed by Mackinlay et al [Mackinlay et al., 1990]. A fourth axis called "none" is also included for values that are not expressed in a 3D coordinate system, for example a temperature. To continue, the semantics lets the developer know which real world data is acquired. For example, it can be the name of a body joint in the case of the Microsoft® Kinect™. In the "technology" entity, we included a reliability rank of the acquired data between 0 and 1, this property is present for all input types. This value may change at runtime, as some SDK are able to give a reliability score for each value acquired. With the "Techno Real Type" property, we differentiated the values acquired with a distant wireless sensor from the ones acquired with a physical object. For example with optical tracking there is no intermediary for the interaction contrary to a gamepad.

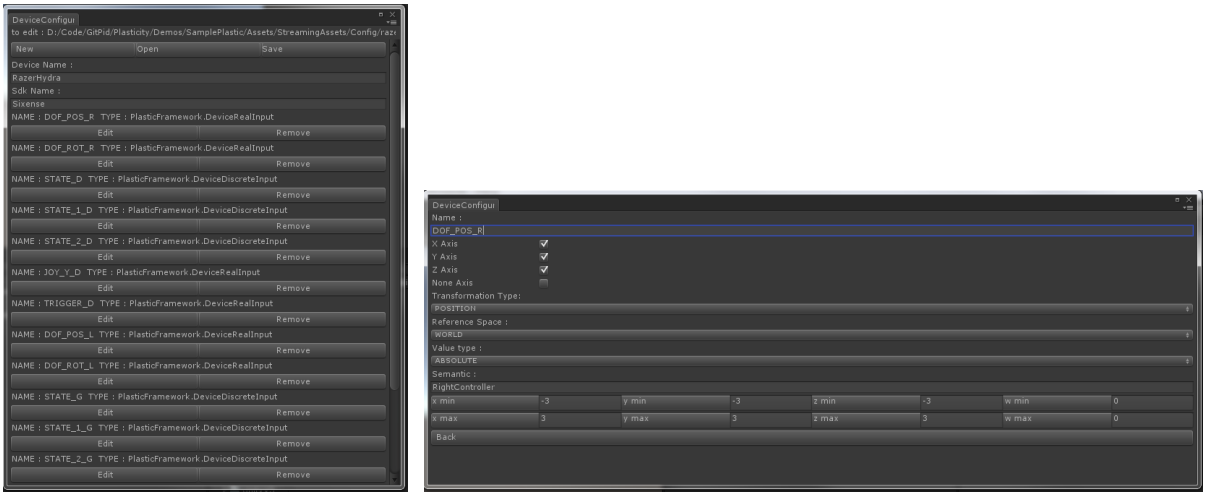
A description of output devices is also included. An output is described by one entity that provides the information about how the data is rendered to the real world. All output types but visual and sound are extracted from the MPEG-V standard [Han et al., 2012]:

- Visual outputs. They include screens such as monitors and HMDs. They also encompass punctual lights, for instance a gamepad led or a remotely controlled lighting system that modifies the lighting conditions of the real world.
- Sound outputs. They stimulate the auditory sense and include devices such as speakers and headphones.
- Tactile outputs. They stimulate the touch sense, for example the vibration of a gamepad.
- Force-feedback outputs. They apply a force in return of a user interaction in order to simulate a collision with a virtual object. Typical force devices are robotics arms.
- Temperature outputs. They modify the temperature of the real world or of a contact point. An example is to control the temperature according to the weather conditions in a virtual world.
- Wind output. They change the air speed and direction dynamically.
- Scent outputs. They stimulate the olfactory sense.

FIGURE 4.8 – The description of a force feedback output device

- Sprayer outputs. They can throw water on a user. It can be used to simulate a virtual rain effect in the real world.

Three properties are common to all output and input types: the name, the refresh rate and a possible associated relative object. For instance, the force feedback type properties are listed in Figure 4.8. The first three properties are extracted from the description of force feedback devices given by Florens et al. [Florens et al., 2007]. First, the continuous force is the maximum force that can be applied for an unlimited period without damaging the device. Secondly, the peak force is the maximum feasible force. Thirdly, the force resolution gives the quantization step of the force that can be applied. These values are expressed in newtons. Our description also contains a boolean for each DOF on which the device can apply a force.



(a) (b)

FIGURE 4.9 – The graphical tool for creating and editing devices configuration files. Here, the file of the Razer Hydra is currently edited. (a) The developer can edit each input, output, and physical object individually and add other ones if necessary. (b) The developer edits the data description that corresponds to the right controller position of the device.

Regarding the implementation of the model in 3DPlasticToolkit, the model must be extendable because new devices will still appear and they might include new properties not yet included in the current model. The model is described with UML class diagrams, so it is totally editable by any developer who wants to add new properties or new input or output types. In order to add a new device into the toolkit, the developer has to create a new class that inherits from the basic device class. In this new class, the developer has to complete some functions to fulfill the input data, trigger the outputs and tell the system when a new instance of the device is plugged or unplugged. These steps can be done with a device SDK. Two examples of code for the Razer Hydra device are given in Listings 4.1 and 4.2. In these two code examples, the only part that will be

```
1  public class RazerHydra : Device {
2  (...)
3  public override void update(){
4  {...)
5  // Get the position with the Razer Hydra API
6  Vector3 vecR =  SixenseInput.GetController(SixenseHands.RIGHT).Position ;
7  // Fulfill the data of the associated device unit
8  DeviceRealInput razerPosRight =  (DeviceRealInput) getInput("POS_R") ;
9  razerPosRight.Data.setValue(vecR) ;
10 // Get the Rotation with the Razer Hydra API
11 Quaternion quatR=   SixenseInput.GetController(SixenseHands.RIGHT).Rotation ;
12 // Fulfill the data of the associated device unit
13 DeviceRealInput razerRotRight = (DeviceRealInput) getInput("ROT_R") ;
14 razerRotRight.Data.setValue(quatR) ;
15 // Get the first button state with the Razer Hydra API
16 bool button1 = SixenseInput.GetController(SixenseHands.RIGHT).GetButton(SixenseButtons.ONE);
17 // Fulfill the data of the associated device unit
18 DeviceDiscreteInput razerButton1= (DeviceDiscreteInput) getInput("BUTTON1");
19 razerButton1.Data.setValue(button1) ;
20 }
21 }
```

Listing 4.1 – An excerpt of the update function of the Razer Hydra device class. Here, with the device SDK, the developer fulfills the data for the position and rotation of the right controller and also for one button. This example can be generalized for other devices. Indeed, for each device unit the process is the same. First, in lines 6, 11 and 16 the developer gets the data with the device SDK. Then, with our model API, he updates the data of the associated device input unit for the current frame.

changed from a device to another one are the calls to the function of the concrete devices SDK. In Listing 4.1 the inputs data are fulfilled with the data coming from the device SDK. The Listing 4.2 gives an example of the function that updates the device plug status. In this function, the developer declares to the device manager the index of the device instances that have been plugged or unplugged during the current frame. Then, the device manager automatically adds or removes these device instances from the available ones accordingly. Some devices cannot update their plug status at runtime. This is the case for some particular screens and sound outputs. Therefore, for now, the screen and sound configuration of the system must be manually edited in the application configuration file before execution. This is edited in the XML configuration file of 3DPlasticToolkit.

Regarding the device properties that correspond to the device and its device units description, they are fulfilled at runtime with an XML description file. For instance, an excerpt of the XML file of the Razer Hydra device is given in Listing 4.3. It presents the properties of the data description for the position of the tracker integrated in the right controller of the device. The developer can create and edit directly this XML or he can use a provided graphical tool. Indeed, in order to simplify the description of models and satisfy **R7** we have created a graphical tool for the creation and edition of devices configuration files. An insight of this graphical tool is shown in Figures 4.9a and 4.9b. In the device class the developer can also edit the dynamic properties at runtime. Static properties have to be fulfilled into the description before execution if needed such as the position of a motionless device in the real world. These properties can also be reported into the device SDK by the developer to perform its configuration.

4.5 A New Model for the representation of 3D Application Components

Our solution 3DPlasticToolkit decomposes a 3D user interface into different application components that can interact with each other. For instance, these components can be interaction techniques, widgets, visual effects, etc. According to the plasticity requirements detailed in Section 3.5, these components must be independent of any particular 3D framework, independent of any concrete interaction devices and support alternative concrete representations. Moreover, they must take into

```
1  public override void checkNewInstancePlugged(bool [] CurrentPlug , out bool [] NewPlug, out
       bool [] UnPlug){
2      int RAZER_MAX_INSTANCE = 2 ;
3      // Only two Razers can be plugged at the same time
4      NewPlug = new bool[RAZER_MAX_INSTANCE] ;
5      UnPlug  = new bool[RAZER_MAX_INSTANCE] ;
6      for (int i = 0 ; i < RAZER_MAX_INSTANCE ; i++){
7          NewPlug[i] = false ;
8          UnPlug[i] = false ;
9          if (SixenseInput.IsBaseConnected(i) && !CurrentPlug[i]){
10             NewPlug[i] = true ;
11         } else if (!SixenseInput.IsBaseConnected(i) && CurrentPlug[i]) {
12             UnPlug[i] = true ;
13         }
14     }
15 }
```

Listing 4.2 – The function that updates the plug status for all Razer Hydra instances. For a particular frame, the developer tells if some indexes of the device have been plugged or unplugged. To do so he uses the Razer Hydra SDK. As shown in lines 9 and 12, we use a function from the Razer Hydra API that tells if for a given index a device is plugged. As long as a device SDK can dynamically detect plug status, this function can be easily generalized.

```
1  <DeviceDescription Name="RazerHydra" SdkName="Sixense">
2  (...)
3  <DeviceRealInput>
4  <DataDescription Name="DOF_POS_R" X="True" Y="True" Z="True" None="False" TransformationType="
       POSITION" RealValueType="ABSOLUTE" SEMANTIC="RIGHTPOS" ReferenceSpace="WORLD">
5  <Bounds minX="-1.0" minY="-1.0" minZ="-1.0" minW="0" maxX="1.0" maxY="1.0" maxZ="1.0" maxW="0"
       />
6  </DataDescription>
7  <Technology>
8  (..)
9  </Technology>
10 </DeviceRealInput>
11 (...)
12 </DeviceDescription>
```

Listing 4.3 – A part of the Razer Hydra configuration file. Here is the data description for the position of the tracker integrated in the right controller of the device

account context changes at runtime and be configured accordingly.

Our application component model extends PAC [Coutaz, 1987] (Presentation-Abstraction-Control), by taking into account Compact [Calvary et al., 2004b] (COntext of user Mouldable PAC for plasticity) and ARCH [Arch, 1992] concepts in order to satisfy our needs. PAC is a multi-agent model that ensures a good decoupling between user interface semantics and its concrete implementation. It decomposes an interaction component into three facets:

- the Presentation: it is the concrete implementation of the component in charge of its input and output management,
- the Abstraction: it describes the semantics of the component and the function it can perform,
- the Control: it ensures the consistency between the presentation and the abstraction.

This model has not been designed to perform context adaptation: a PAC agent will be the same whatever the target platforms and users. Compact [Calvary et al., 2004b] is a specialization of PAC. It divides each facet into two parts, the physical part that is dependent from the current context and the logical one which is always the same. The model introduces a way to take into account the context of use. Indeed, the physical part of the control facet is in charge of the sensing of any context change and then of computing and applying a reaction. Both, PAC and Compact do not provide a good decoupling between the input and the output management because the Presentation facet includes both. This decoupling is needed in 3D because of the wider set of possible interaction devices. ARCH is a meta-model for other software models that proposes a generic separation between facets of interactive components. ARCH represents an interactive component as a set of facet branches dedicated to specific features such as presentation and data processing.

Our model has two mains differences compared to the PAC model. First, as it can be done with ARCH, it separates the original presentation facet into two branches as shown in Figure 4.11. The two branches are based on its rendering (rendering presentation) and device management (logical driver) core functions in order to decouple the two functions:

- The rendering presentation facet is the only facet depending on a 3D framework. It handles graphics output and physics. With the development of multiple compatible rendering presentations we can ensure the portability of the component over different operating systems and different 3D frameworks. For a given application component, this facet can also define its representation in the virtual world. For instance, the 3D aspect of a widget will be defined in this facet.
- The logical driver handles input and output devices management. If the application component does not need any device, it can be omitted. Its main use is for the development of interaction techniques. It describes the way the interaction technique is controlled according to a set of interaction device units. The work of the developer is to choose these device units in order to drive correctly the interaction technique. The logical driver describes all required inputs and outputs units according to a set of parameters taken from our device model described in Section 4.4. Some can be optional if they are not needed for a good usability. The logical driver can be instantiated if these units can be found at runtime. At runtime, the logical driver receives from concrete devices the input data that it needs and it can trigger the outputs. The device units may come from different concrete devices in order to perform device composition. The developer can also access each specific properties in order to ensure the same behaviour whatever the concrete device used. These properties are included in the device model described in Section 4.4. An example of logical driver is given in Figure 4.10.

The second difference compared to the PAC model consists in placing another facet on top of the control one. As proposed in Compact, this facet receives the context modifications at runtime and then is able to determine if a presentation facet is still possible in the current context. For instance, if a device is unplugged from the system, the supervision control may detect that the current logical driver is unusable. Contrary to Compact, the facet does not compute and apply any

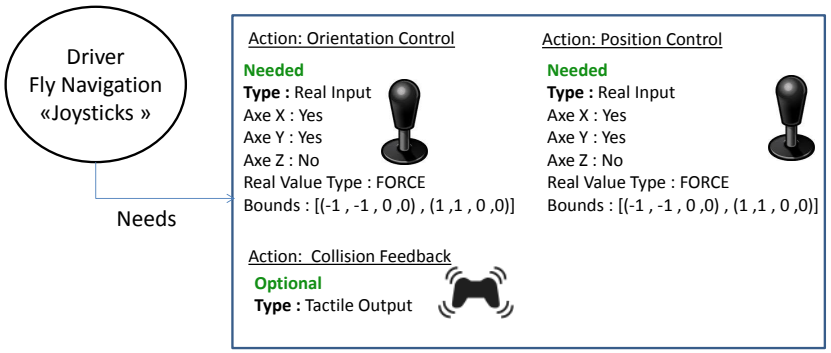


FIGURE 4.10 – An example of logical driver that controls a fly navigation interaction technique with joysticks. The different properties for the devices needed are extracted from the device model described in Section 4.4. The first needs are two real values that describe the two joysticks needed. The first one controls the translation of the point of view, and the second one its rotation. A need is defined with some properties extracted from the device model. As shown some device properties can be omitted, only the detailed ones will be used to select the available device units. The last one is tactile feedback, which is optional and can give a vibration feedback when the user collides any 3D object.

modification but it asks the adaptation engine for a modification. We consider that the modification must be global and not local as it can also impact other parts of the application. For instance, if the devices used by a component are replaced by other ones, they could be possibly connected to another component. The modification of an application component can be the replacement of one of the presentation facets (rendering presentation or logical driver), or the replacement of all the component. This component also contains all the types that can be instantiated as a logical driver or rendering presentation facet for the current application component. As shown in Figure 4.11 each compatibility is associated with a score: a score *Sld* for each compatible logical driver and a score *Srp* for each compatible rendering presentation. It allows the developer or the designer to rank the compatibilities in order to choose one facet over another if multiple are possible. These choices and the use of these scores are described in Section 4.7.

By using this approach we ensure a good decoupling between the application component semantics and its concrete implementation, the independence of the component over the targeted 3D framework and OS and over the concrete devices used. As multiple rendering presentation can be implemented, a same component can have different 3D aspects in the final application.

To illustrate our application model, the Figure 4.11 presents a 3D ray-based interaction technique used in the furniture planning application described in Section 5.2 for menu selections and objects manipulations. For creating this new component, the developer has to implement the different classes that correspond to our model. To do so, each of these new classes have to inherit from the basic model classes included in 3DPlasticToolkit: ComponentAbstraction, ComponentControl, ComponentSupervisionControl, ComponentRenderingPresentation and ComponentLogicalDriver. In the 3D Ray example, the rendering presentation facet creates the ray geometry, handles collision detection and performs scene graph modifications according to the control facet requests. It implements these different features with a particular 3D framework. The logical drivers handle how the ray is manipulated according to different device units. The first one is based on a 3D interaction device that can provide a 6-DOF tracker. The ray base is controlled in position and rotation by the data given by this tracker. A discrete input (such as a button event) is used to attach and detach an object to the ray extremity. Two other discrete inputs are used to change the length of the ray, the first one to increase it, the other one to decrease it. It also includes an optional tactile output in order to perform a vibration feedback when an object is caught. This implementation is shown in Figure 4.12a, the device used is a Razer Hydra[1]. The second logical driver is based on a

[1] http://sixense.com/razerhydra

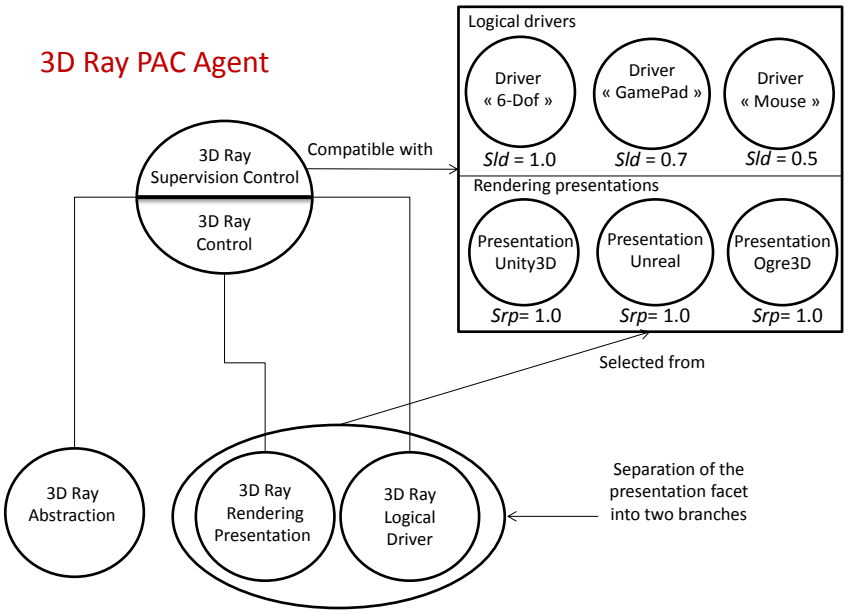


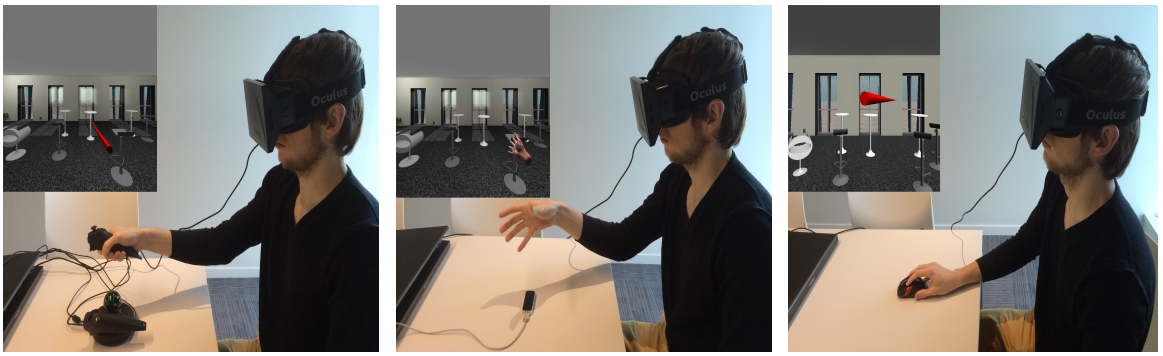
FIGURE 4.11 – The PAC agent of the 3D ray-based technique. This agent is compatible with three concrete logical drivers and three rendering presentations. As shown each compatibility is ranked with a score.

2-DOF force input and on a 1-DOF force input. For instance, it can correspond to two joysticks. The position of the ray base is constant and set at the center of the user view, just in front of the main camera. The 2-DOF input is used to control the rotation of the ray on the X and Y axes in order to control the ray target. The 1-DOF input is used to increase the ray length when a positive force is applied and to decrease it when a negative force is applied. A discrete input is still used for object catching. The last logical driver is based on an input of mouse type with a 2-DOF input in screen space, as shown in Figure 4.12c. The position is also constant at the center of the user view. To control the ray rotation, we compute the ray target as the intersection point between the far clipping plane and a ray defined by two points, the camera position and the 3D point that corresponds to the mouse position on the near clipping plane. A 1-DOF speed input that corresponds to the mouse wheel is used to increase and reduce its length and a discrete state (one button) for object catching. The Figure 4.12b gives another example of application component: a 3D cursor represented by a virtual hand. In that case, the logical driver used consists in controlling a 3D cursor in position and rotation by tracking the user's hand. To catch an object, the user has to close his hand.

With this model we can implement application components independently from 3D frameworks and from devices. In Section 4.7 we describe how the application components are instantiated and configured at runtime according to the context of use. One component is instantiated for each high level task defined as needed by the developer. The task model for the description of these high level tasks is described in the next Section.

4.6 The 3DPlasticToolkit task system

As in Grappl [Green and Lo, 2004] and CATHI [Lindt, 2009] we represent a 3D user interface as a composition of high level tasks. Both consider a high-level task component as a self-contained part of a 3D user interface. Both solutions focus on interaction tasks. An interaction task corresponds to an action performed by a user via a user interface in order to achieve a certain goal. In Grappl, each interaction task has a corresponding coding interface that compatible interaction techniques have to respect. In the CATHI framework, high level interaction tasks are connectable components connected to the application logic and to low level interaction tasks. These low level interaction



(a) (b) (c)

FIGURE 4.12 – An illustrative case of the possibilities offered by 3DPlasticToolkit. (a) On a laying-out scenario a user is manipulating the furnitures with a Razer Hydra and a 3D-ray based interaction technique (in red). (b) When the Razer Hydra is unplugged and a leap motion is plugged, the current interaction technique is replaced by cursor based-interaction technique represented with a virtual hand. Usability continuity is ensured with this adaptation. (c) Even without any 3D interaction device the user can still drive a 3D-ray based interaction technique with a mouse, mapping the mouse movements on the ray target.

tasks represent the concrete user interface.

Our task model does not only focus on interaction tasks. The tasks can refer to interaction techniques, widgets, visual effects, data-visualization metaphors or application logic components. They are elementary tasks that represent the behaviour of the application independently from any concrete application component. They represent the features that the developer wants to integrate into the application with a high granularity. Therefore, a task is represented by a name that describes its role in the final application. For instance, it can be "Selection", "Navigation", etc. Second a task can be parametrized with different attributes. These attributes allow the developer to more precisely describe the task. Last, a task is composed of events that are methods called when some particular actions related to the task occur in the application. For instance, in the selection task two possible events can be the selection of an object and the deselection of this same object. These events can be connected to other parts of the application.

For now our task model does not include the notions of sequences of events and actions that can occur in the application. Only simple events can be included in tasks. At runtime each task is associated with a concrete application component. An application component represents the concrete instantiation of a task in the application. Each task in the system derives from a basic task class and contains a list of compatible application components developed with the model presented in Section 4.5. Task events can be triggered automatically by a compatible application component and can be manually connected to other software components by the developer. Some example of task events are given in Section 5.2. The compatibility between tasks and components is ranked. Indeed, some application components can be considered more adapted than others. Therefore, a compatibility score is assigned to each application component. This compatibility list has to be edited in an XML file. For a given task, the developer has to give the list of the control facets names that correspond to the compatible application components. These control facets names are associated with the compatibility scores Sa. This XML file also contains the compatibility scores Srp and Sld between control facets, rendering presentations and logical drivers as described in Section 4.5. In this XML file we also give to the developer the possibility to change the score Sa of a component according to the main display type taken from our device model. Indeed, for instance, we consider that the compatibility between a task and an interaction technique can be different if the display is a flat screen or a HMD. An excerpt of this XML file for the creation of this list is given in Listing 4.4. It corresponds to the compatible application components for the selection and manipulation task given in Figure 4.13. For the 3D cursor component we show that its score can be changed according to the main display used. Here, if the display is a flat screen its compatibility

```
1 <TaskCompatibility taskName="TaskSelectionManipulation" componentName="Ray3DC" score="1.0">
2     <DriverCompatibility componentName="Ray3DC" driverName="3DRayGamePadDriver" score="0.3"/>
3     <DriverCompatibility componentName="Ray3DC" driverName="3DRay6DofDriver" score="1.5"/>
4     <DriverCompatibility componentName="Ray3DC" driverName="3DRayMouseDriver" score="0.5"/>
5 </TaskCompatibility>
6 <TaskCompatibility taskName="TaskSelectionManipulation" componentName="Cursor3DC" score="0.8">
7 <TaskCompatibility taskName="TaskSelectionManipulation"  componentName="Cursor3DC" displayType=
      "FLAT_SCREEN" score="0.3">
8 </TaskCompatibility>
9 <TaskCompatibility taskName="TaskSelectionManipulation" componentName="Proximity3DC" score="0.5
      ">
10 </TaskCompatibility>
```

Listing 4.4 – An excerpt of the XML file where the developer or the designer can edit the compatibilities between tasks and application components. Here, the selection and manipulation task is compatible with three interaction techniques: a 3D ray component with a score of 1.0, a 3D cursor component with a score of 0.8, and a 3D proximity component with a score of 0.5. Compatibilities between application components and logical drivers and rendering presentations. Here the Ray3D component is compatible with three logical drivers.

score is reduced, otherwise it takes the default one.

In order to associate the component names with concrete code instances, we use a factory design pattern in which the developer has to register his components. As the compatibility can also depend of the context of use, these scores can be edited at runtime, which can result to modifications in the final application. This compatibility list is exposed to the system to allocate the best application components according to the desired tasks and the current context of use. To illustrate this task model, the Figure 4.13 gives an example of a task commonly needed in MR applications: a selection and manipulation task. This task is compatible with three application components, a 3D ray-based interaction technique, a 3D cursor, and a 3D proximity manipulation technique.

The developer can also include some parameters into the task as key-value properties. At runtime, an application component will have access to its corresponding high level task and therefore its parameters. For instance, in the example of manipulation task given in Figure 4.13 we can parametrize the degrees of freedom on which objects can be manipulated. An application control task could be parameterized with a tree that defines the possible choices of a menu. Dependent tasks can also be defined by the developer or the designer in order to indicate if a task needs another one to be completely performed. For example, a 3D menu would need a selection interaction task in order to be achieved.

In order to add a new task to the system, the developer has to create a new class that inherits from a basic task class. To select which tasks will have to be performed in a particular application, the developer has to fulfill the XML configuration file of 3DPlasticToolkit. This part of this file also contains the tasks parameters. As for the device model, a graphical tool is provided to perform this configuration in order to make it usable by any designer. For now, key value properties in a string format can be extracted from the configuration file for each task. Custom properties can also be included into XML nodes in the configuration file. The parsing of these nodes must be implemented in the corresponding task class by the developer. As an example, an extract of the

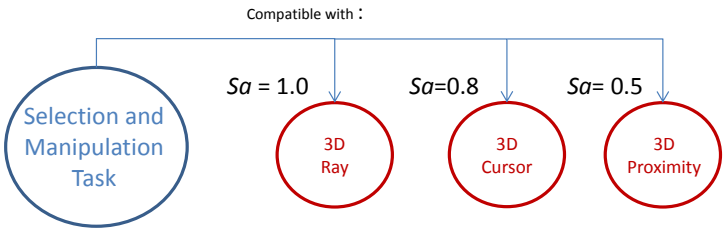


FIGURE 4.13 – An example of high level task: selection and manipulation. This task is compatible with three concrete application components. As shown, this compatibility is ranked with scores.

```
<TaskConfig>
<NeedTask userId="1" taskName="SelectionManipulation" taskId="0" ScoringModule= "User" >
</NeedTask>
<NeedTask userId="1" taskName="FurnitureControl" taskId="1" topTask="0" ScoringModule="Default"
    />
<NeedTask userId="1" taskName="Navigation" taskId="2" ScoringModule="User"/>
</NeedTask>
</TaskConfig>
```

Listing 4.5 – An extract of the 3DPlasticToolkit configuration file of the furniture planning application described in Section 5.2. Two high level tasks are defined as need: selection and manipulation, and application control. The "ScoringModule" field is discussed in Section 4.7 and the "userId" one in Chapter 7

configuration file for the furniture planning described in Section 5.2 is given in Listing 4.5. Two high level tasks are defined. First, a selection and manipulation task for menu selection and moving 3D objects into the scene. Second, an application task for adding furniture into the room with a menu. A menu needs a selection mechanism so this task is defined as dependent to the first one in the configuration file. As shown for each task there is a parameter named "ScoringModule". It gives the possibility to the developer to choose for each task how the compatibility scores will be taken into account. The possible choices and the impact on adaptations are described in Section 4.7. There is also a parameter named "userId" that associates to each task a particular user in order to perform user adaptation as explained in Chapter 7.

4.7 Dynamic Recasting in 3DPlasticToolkit

Recasting consists in locally adapting an application to the context of use without modifying its component distribution. According to the plasticity requirements our solution must be able to automatically adapt an application without an external intervention. It must also give the possibility for the end-user to change the adaptation behaviors. This is ensured in 3DPlasticToolkit with two main components: an automatic adaptation process based on scoring mechanisms and a meta-user interface to modify the adaptation behaviors.

4.7.1 An Adaptation Process based on Scoring Mechanisms

As for the CATHI framework [Lindt, 2009], our solution proposes an adaptation process that uses a scoring system combined with our application model described in section 4.5. The goal of these scores is to maximize the usability of the final application.

At runtime the construction of the 3D user interface consists in associating to each task the best triplet (application component, logical driver, rendering presentation). The goal of the adaptation engine is to find the triplet that maximizes the score of compatibility in order to create the application that uses the most suited devices and with the most adapted content presentation.

We propose an abstract way to compute these scores through what we named "scoring modules". Each task is associated to a particular scoring module as explained in Section 4.6. A scoring module implements a particular way to compute scores for an application component according to the context of use. Moreover, the 3DPlasticToolkit API also gives the possibilities to developers to integrate new scoring modules. Therefore, it gives them the flexibility to deeply control the adaptation process if they have specific needs. The first built-in scoring module that has been developed in 3DPlasticToolkit is the default one. It uses the different compatibility scores exposed in the application model and in the task model that are given by the developer or the designer. Other modules are discussed in Chapters 7 and 8 and some are given as perspectives of work.

Regarding this default scoring module, as explained before, for a given task the developer or the designer can rank each compatible application component with a score Sa. In the same way, for a given application component, a compatible rendering presentation will be ranked with a score Srp and a logical driver will be ranked with a score Sld. This module only considers these compatibility scores provided by the designer or the developer. The goal is to provide them the maximum control over the adaptation process. The score for a given triplet is computed as: $S = Sa + Sld + Srp$

The adaptation process uses these scoring modules in order to compute scores at every context modification. It ensures to detect not any longer usable application components or more adapted ones. In this process a task is classified as *done* if an application component is currently deployed for it in the application. If it is classified as *not done* no application component is deployed for this task because the adaption process did not find an adapted one yet. The life cycle of an application component starts when it is deployed in the application and ends when it is destroyed. The destruction of a component only affects usability continuity if no other component is deployed by the adaptation process to replace it. A device unit is defined *available* only if it is not used by another application component. Otherwise it is defined as *not available*. The process can be described as follows:

1. A context modification is detected. For example, it can be the connection of a new device, the add of a task, or the association of a task to a new detected user.

2. This modified context is transmitted through all supervision controls currently instantiated. For each one, we check if the associated logical driver is still convenient in the current context of use. It is still convenient if the devices that it uses are still plugged and available. If not, the application component is destroyed and the associated task is classified as *not done*.

3. For each not done task, we create a list of all convenient triplets (application component, logical driver, rendering presentation) that can achieve the given task. A triplet is possibly instantiable if the logical driver needed device units can be found in the list of connected devices and if they are available. The rendering presentations that do not correspond to the current used 3D framework are omitted. A score is attributed to each triplet according to the scoring module chosen for the given task. Then, if multiple triplets are found, the one that obtains the best positive score is used to instantiate the PAC agent with the suited logical driver and rendering presentation. The devices units associated with the logical driver are set as *not available*. The task is classified as done

4. For each done task that has not been processed in the previous step, we check if we can find a triplet more adapted than the current one. To do so, we create a list with all triplets that get a better score (positive) than the current one. If the list is empty, the current one is still the most adapted. Conversely, we destroy the current application component and we instantiate the new best choice. For now, this choice is directly applied but it could be only suggested to the user in order to produce a less disturbing effect.

In order to illustrate this adaptation process, a usage scenario given in Figure 4.14 shows the different steps of the adaptation process in 3DPlasticToolkit when context changes happen. In this example we focus on the instantiation of the application components for the following tasks: navigation and objects manipulation. In this example, the context change consists of modifications of the available devices. For this scenario, the Figures 4.12a, 4.12b and 4.12c represent how the adaptations impact the application and the end-user. In this scenario, when the application is launched, three devices are connected to the computer, a keyboard, a mouse and a Razer Hydra. The Unity3D game engine is used. In the XML configuration file of the application, the Oculus Rift HMD is chosen as the main display. Therefore, the associated device class configures the device SDK and the 3D rendering accordingly. The launch of the application induces the instantiation of a component for each task. The score assignation for the components is made with the default scoring module according to some of the scores that are detailed in Listing 4.4:

1. For the selection and manipulation task, according to the devices available the possible choices are:
   - (3DRay, 3DRay6DofDriver, 3DRayPUnity3D) with $S = 1.0 + 1.5 + 1.0 = 3.5$
   - (3DCursor, 3DCursor6DofDriver, 3DCursorPUnity3D) with $S = 0.8 + 1.5 + 1.0 = 3.3$
   - (3DProximity, 3DProximity6DofDriver, 3DProximityPUnity3D) with $S = 0.5 + 1.5 + 1.0 = 3.0$
   - (3DRay, 3DRayMouseDriver, 3DRayPUnity3D) with $S = 1.0 + 0.5 + 1.0 = 2.5$

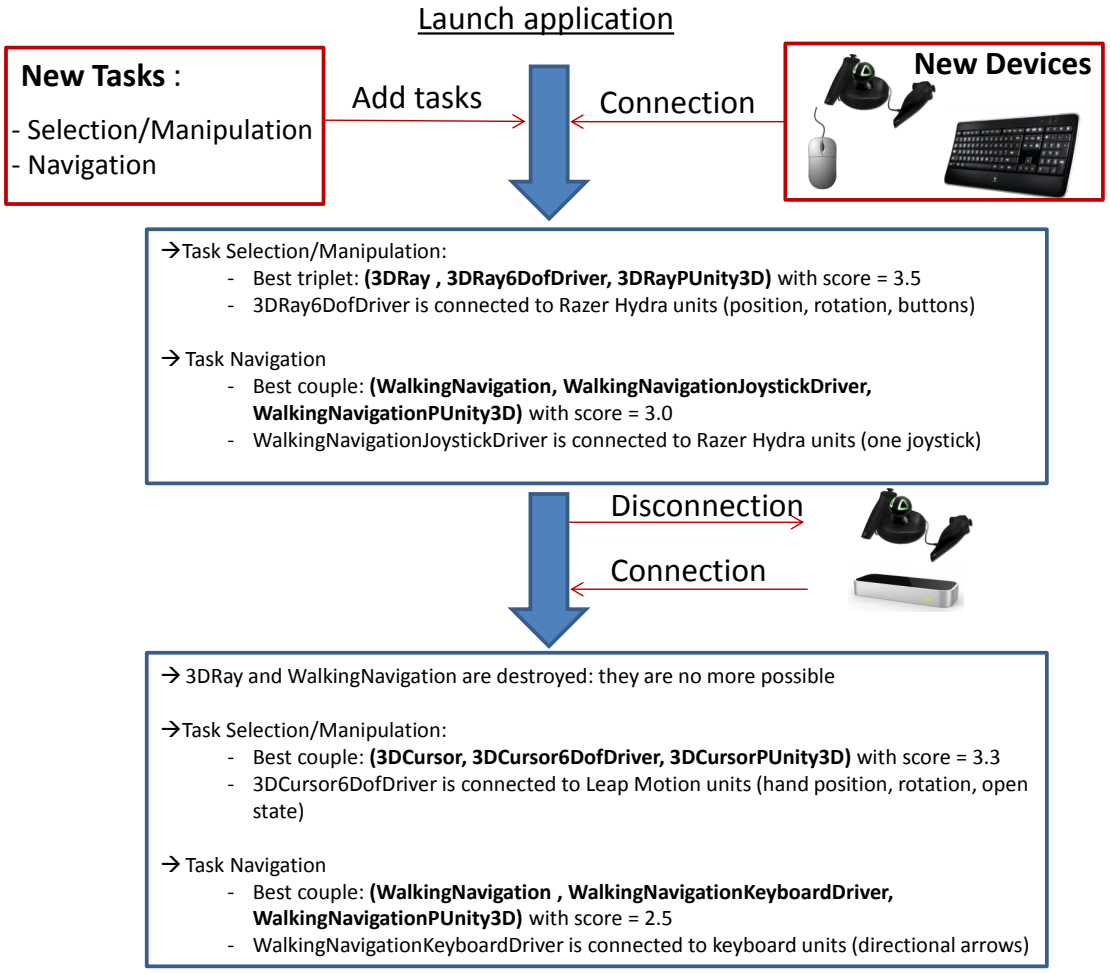


FIGURE 4.14 – A usage scenario of the furniture planning application developed with 3DPlasticToolkit. It demonstrates how the application can handle context changes: here the replacement of an interaction device by another one. This example represents the adaptation steps that correspond to the transition from the situation illustrated in Figure 4.12a to the one illustrated in Figure 4.12b

- (3DRay, 3DRayGamePadDriver, 3DRayPUnity3D) with $S = 1.0 + 0.3 + 1.0 = 2.3$
- (3DCursor, 3DCursorGamePadDriver, 3DCursorPUnity3D) with $S = 0.8 + 0.2 + 1.0 = 2.0$
- (3DProximity, 3DProximityGamePadDriver, 3DProximityPUnity3D) with $S = 0.5 + 0.2 + 1.0 = 1.7$

The combination chosen is (3DRay, 3DRayDriver6Dof, 3DRayPUnity3D) because it gets the best score (3.5). Therefore, the application component is instantiated. As shown in Figure 4.12a, the user can now manipulate the objects with a 3D Ray-based interaction technique controlled with the Razer Hydra.

2. For the navigation task, the possibilities are:
   - (WalkingNavigation, WalkingJoystickDriver, WalkingNavigationPUnity3D) with $S = 1.0 + 1.0 + 1.0 = 3.0$
   - (WalkingNavigation, WalkingKeyboardDriver, WalkingNavigationPUnity3D) with $S = 1.0 + 0.5 + 1.0 = 2.5$

   The chosen triplet is (WalkingNavigation, WalkingDriverJoystick, WalkingNavigationPUnity3D) because it gets the best score (3.0). It corresponds to an interaction technique based on a walking navigation metaphor controlled with one of the joysticks provided by the Razer Hydra. This metaphor can be compared to the "WALK" navigation type from X3D. The joystick is used to move the point of view forward and backward as well as changing its rotation around the up axis.

The logical driver instantiated for the navigation task also describes a tactile output as an optional output. A vibration is triggered when colliding a virtual object. As this output is optional, the logical driver can be instantiated without it. Indeed, the Razer Hydra does not provide any vibration capabilities.

After a few minutes, someone asks the user for the Razer Hydra. The user does not want to stop using the application because he has not finished to lay out the furnitures. In exchange for the Razer Hydra, a Leap Motion is given to him. As the Razer Hydra is disconnected, the two currently instantiated application components are destroyed because they are not adapted anymore. Then, with the connection of the Leap Motion the interface is rebuilt as follows:

1. For the selection and manipulation task, some triplet are no more convenient, the list of convenient one is updated:
   - (3DCursor, 3DCursor6DofDriver, 3DCursorPUnity3D) with $S = 0.8 + 1.5 + 1.0 = 3.3$
   - (3DProximity, 3DProximity6DofDriver, 3DProximityPUnity3D) with $S = 0.5 + 1.5 + 1.0 = 3.0$
   - (3DRay, 3DRayMouseDriver, 3DRayPUnity3D) with $S = 1.0 + 0.5 + 1.0 = 2.5$
   - (3DProximity, 3DProximityGamePadDriver, 3DProximityPUnity3D) with $S = 0.5 + 0.2 + 1.0 = 1.7$

   The triplet that gets the best score (3.3) is now (3DCursor, 3DCursor6DofDriver, 3DCursorPUnity3D). It corresponds to a 3D cursor controlled with a driver that implements a 6-DOF interaction as shown in Figure 4.12b. The position and rotation of the cursor are controlled through one hand detected by the Leap Motion. The open or close state of the hand is used as the signal for selection and unselection.
2. For the navigation task, no more joysticks are available. One triplet is possible and therefore is instantiated: (WalkingNavigation, WalkingKeyboardDriver, WalkingNavigationPUnity3D). The walking navigation component is now controlled with the directional keys of the keyboard. The up and down keys are used to move forward and backward while the left and right ones are used to rotate the point of view around the up axis.

The Figure 4.12c represents an extension of this scenario. It demonstrates that the user can still use the application without any 3D interaction device. Indeed, if the Leap Motion is unplugged no more 3D interaction device will be available. Nevertheless, as said in Section 4.5, the 3D-ray

based interaction technique can also be controlled with the mouse. The triplet that corresponds to this interaction is therefore automatically instantiated. The keyboard is still used to control the navigation.

This usage scenario demonstrates the ability of the proposed model to handle context modification in order to ensure usability continuity of any 3D user interface. Moreover, with the use of a scoring system this usability is maximized according to the current context of use. In the next section, we demonstrate how this adaptation mechanism can be configured at runtime by the end-user thanks to a an integrated graphical user interface: the meta-user interface.

4.7.2 Meta-User Interface: User Control over the Adaptation Process

In the previous Section we demonstrate that thanks to our automatic adaptation process the adaptation controller can be the system itself. However, one of the identified drawbacks in current MR solutions is the lack of control given to the user over the adaptation process. Indeed, the user has to be aware of any system adaptation and must be able to modify the result (requirement **R5**). Therefore, our solution proposes a built-in application component that implements a graphical user interface (GUI) that satisfies this need: the meta-user interface.

Meta-user interface have been already integrated into plastic 2D user interfaces. For instance in the Sedan-Bouillon plastic web site [Vanderdonckt et al., 2008], a meta-user interface is used to control the configuration of the applications. In the same paper, Vanderdonckt et al. define a meta-user interface as an integrated user interface that provides users with the means to configure, control, and evaluate the adaptation process. A meta-user interface may be plastic and may or may not negotiate the alternatives for adaptation with the user.

Our meta-user interface provides the end-user with a view of the current state of the system and gives him the possibility to modify it. This meta-user interface works with both 2D and 3D selection techniques integrated in 3DPlasticToolkit. Therefore, our meta-user interface is totally plastic. In our case the meta-user interface incorporates negotiation as the user can not only observe the adaptation process but he can also impact it with different possibilities. The following aspects of the application can be modified with this interface (some of these features are still currently in development):

- For each task, the user can see the current associated application component and can select another one to achieve it. Only the ones that can be instantiated in the current context of use are proposed. For instance, it can be used to switch from an interaction technique to another one.

- For each instantiated application component, the user can see the associated logical driver and can select another one to control it. Similarly to the interaction techniques, only the convenient logical drivers are proposed. For example, it can be used to change the kinds of devices that control an interaction technique.

- For each instantiated application component, the user can see the associated rendering presentation and can select another one. As well, only the possible ones are proposed. For example, it can be used to change the aspect of a 3D widget at runtime.

- For each logical driver, the user can see all associations between actions and device units. The user can change each associated device unit in a list of compatible and available ones. An example of this features is given in Figure 4.15. In this example, for the case of the 3D ray-based interaction technique controlled with the Razer Hydra, the user can change which controller manipulates the ray position. It can be the right or the left one.

- The user can change the current display used. For example a user could stop using an HMD and continue to interact on his/her monitor screen.

Any modification of the application performed by the user with the meta-user interface is then reported to the adaptation process as a context of use modification. Indeed, the modifications

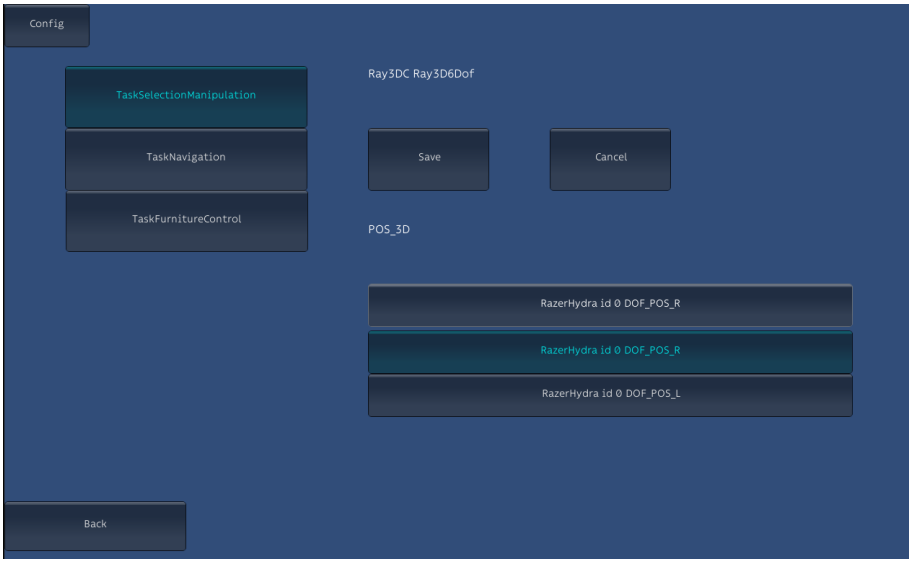


FIGURE 4.15 – An illustration of the meta-user interface. Here is an example where the user can change a device unit associated to a logical driver from a list of compatible ones. For the 3D ray-based interaction technique controlled with the Razer Hydra, the user can change which controller manipulates the ray position. It can be the one on the right or the one on the left.

that can be applied could change the device context of use. Some device units may be set as not available while others could be set as available and therefore be possibly used by other application components.

Other features of this meta-user interfaces are detailed in the next Chapters. For instance it is also used to control the redistribution process presented in Section 6 and to modify the user profile as explained in Section 7

4.8 Conclusion

In this Chapter we present the basis of our solution for the creation of plastic MR user interface. The focus of this Chapter is on adaptation to devices.

We propose a device model in order to precisely describe display and interaction devices that can be used by an application. With this model a MR user interface can be adapted according to the available devices and to their properties. Therefore, we partially cover **R1** in this Chapter. In order to fully cover this requirement some extensions of our solution are proposed in Sections 7 and 8. To continue, we also propose a model for the development of application components. These components deployed at runtime to accomplish high level tasks can correspond to interaction techniques, 3D widgets, visual effects, etc. By instantiating this kind of components 3DPlasticToolkit can target both interaction techniques and content visualization as adaptation sources. Therefore, **R2** is fully covered. These components can be developed independently from an OS or from a 3D framework, therefore **R4** can be covered. The recasting capability is ensured by two elements. First, a dynamic adaption process that checks the modifications of the context of use in order to deploy an application component for each task. This process maximizes the usability of the application with scoring mechanisms that use scores set by the developer. Second, an integrated user interface that allows the end-user to check and to modify the adaptation behaviors at runtime: the meta-user interface. For instance, he/she can use it at runtime in order to replace an interaction technique by another one. With these two elements recasting is handled, therefore **R3** is partially covered. In order to fully cover this requirement we also propose a redistribution process in Chapter 6. With this automatic adaption process, the possibility for the developer/deployer to edit different configuration files, and with the meta-user interface, we can say that **R5** and **R6** are fully covered. These different models, the adaptation process and the meta-user interface are integrated in a

software solutions for the creation of plastic MR user interfaces: 3DPlasticToolkit. For now, only one authoring tool is provided with 3DPlasticToolkit for the configuration of devices. We also plan to develop authoring tools for the creation of the 3DPlasticToolkit main configuration file or for configuring the compatibility scores. Therefore, we can say that **R7** is just partially covered

As detailed, our task model allows the developer to model the application behavior and possibilities at a high level. However it does not let him orchestrate the sequences of events and actions that can occur in the virtual environment. This is the goal of scenario engines such as SEVEN [Claude et al., 2014]. This engine can describe scenarios that handle collaboration between different users where each user has as an assigned role. Extending our task model with this kind of scenario engine and taking into account this notion of role in our user model could be perspectives for future work. In addition, as explained with our application component model the creation of a new component requires the creation of multiple classes. Indeed, as explained, multiple faces must be developed, one abstraction facet, one supervision control facet, one control facet and possibly multiple rendering presentations and logical drivers facets. The creation of these classes can be time consuming. Using a MDE approach for the creation of theses classes and to automatically connect their different functions could be a perspective of work to reduce the development time of application components.

In order to illustrate our solution and its benefits for the development of MR user interfaces, in the next Chapter we present three applications that have been developed with 3DPlasticToolkit. These concrete examples demonstrate how the interaction techniques and the content visualization can be adapted according to the device setup. These three uses cases are detailed before we present how we handle redistribution, user adaptation and data adaptation, because although the main focus of these applications is on adaptation to devices, nevertheless they also allow us to demonstrate the need for this redistribution adaptation means and for these two other adaptation sources.

Chapitre 5

Use Cases and Results for Device Adaptation

5.1 Introduction

In the previous Chapter, we present our models for the creation of MR user interfaces and how they can be used in order to adapt an application according to the hardware configuration. In this Chapter, we present three concrete applications that have been developed with 3DPlasticToolkit and that take this adaption source into account in order to adapt the two possible adaptation targets: interaction techniques and content visualization. In this Chapter, the content visualization modifications refer to changes of display modes and not on the choice of the adapted layout to present some data. This last point is discussed in Chapter 8. With the presentation of these three applications, our goal is to demonstrate how the use of 3DPlasticToolkit can really benefit to the development of MR user interfaces.

The first example that we present is a furniture planning application. The goal of this application is to help a customer to plan the use of premises, here a room rented for special events. The user has the capability to layout the room with furniture (add, remove, move) in order to help him to understand the potential of the free space. We demonstrate that this application can be used on a wide variety of platforms (desktop, mobile and immersive ones) thanks to 3DPlasticToolkit. A collaborative extension of this demo is also presented. In the second example, we present how 3DPlasticToolkit can be used in order to adapt the content visualization of a 3D application. This example focuses on a particular case of visual discomfort experienced by users with stereoscopic displays: frame cancellation. We propose two visual effects that can be applied with 3DPlasticToolkit in order to deal with visual discomfort. The last example has been developed for IEEE 3DUI contest 2016. It proposes a collaborative scenario where different users have to collaboratively manipulate an object in order to pass it through a labyrinth. In this application, thanks to 3DPlasticToolkit, each user automatically benefits from adaptation interaction techniques and from an adapted scene visualization according to his device setup.

These three use cases that present how plasticity can be used in order to adapt an application according to the hardware configuration have been published respectively in [Le Chenechal et al., 2015], [Lacoche et al., 2015b] and [Le Chénéchal et al., 2016b, Le Chénéchal et al., 2016a].

This Chapter is structured as follows. First, in Section 5.2, we present how 3DPlasticToolkit has been used to adapt the interaction techniques to the available devices in the furniture planning application. Second, in Section 5.3, we present our two visual effects that can be applied with 3DPlasticToolkit in order to deal with frame cancellation. In Section 5.4 we present our collaborative application that has been developed with 3DPlasticToolkit. In this application the interaction techniques and the scene visualization are adapted to the device setup of each user. In Section 5.5, we conclude and we also give some hints about what we have learned from the development of these three applications based on 3DPlasticToolkit

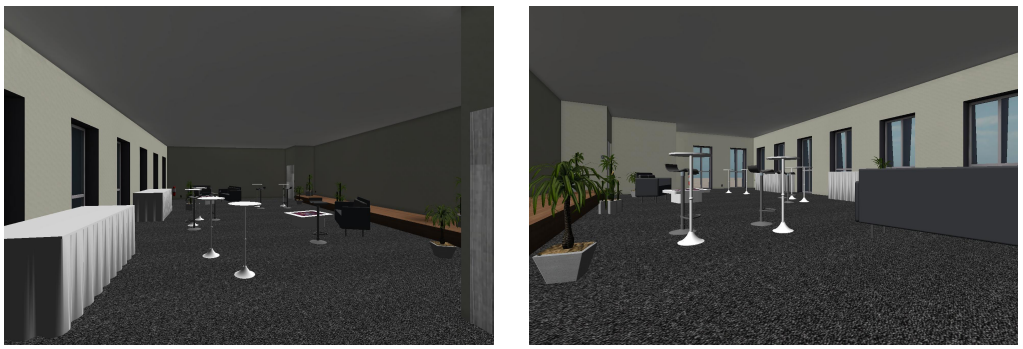


FIGURE 5.1 – Two screen-shots of our application showing a rented room that can be organized with furniture.

5.2 Adapting the Interaction Techniques: The Furniture Application Case Study

The furniture planning application is one of the first applications that we have created with 3DPlasticToolkit. This application enables us to demonstrate how the different features included in 3DPlasticToolkit can be used in order to adapt the interaction techniques according to the available devices. Indeed, we show how this application can be used on different platforms. With this demonstration our objective is to demonstrate the benefits that can bring 3DPlasticToolkit to such development. A video that describes this application can be found here: `https://www.dropbox.com/s/a7ugnevvhcgdua3/FurniturePlanningAdapt.mp4?dl=0`.

The goal of this application is to help a customer to plan the use of premises, here a room rented for special events as shown in Figure 5.1. As this room can be under construction or too far for a real guided tour, we propose to immerse the customer into a virtual version of the premises. The customer has the capability to layout the room with furniture (add, remove, move). These features help him/her to understand the potential of the free space and makes it possible for him/her to imagine and plan how the space will be used.

The different 3D models used in the demonstration such as the room and the furniture were already available. Our goal was to use these models in order to create an interactive application.

High level tasks

The first step in the development of this application with 3DPlasticToolkit consists in selecting the high level task that will describe this application. At the task level, the furniture planning application is composed of three different tasks:

- **The "SelectionManipulation" task**. This is the same task that the one detailed in Section 4.6. The goal of this task is to give the possibility to the end user to select and move furniture in the room.

- **The "Navigation" task**. This task has also been mentioned in Section 4.7. With this task the user will be able to move his point of view in the scene with a navigation interaction technique.

- **The "FurnitureControl" task**. This task has been especially created for this application. It contains different events that can be triggered by the associated application component such as adding a furniture, saving the current layout and loading a pre-defined one.

The configuration file of 3DPlasticToolkit is given in Listing 5.1. It defines as needed these three tasks and it also defines the current main display used as explained in Section 4.4, here a simple

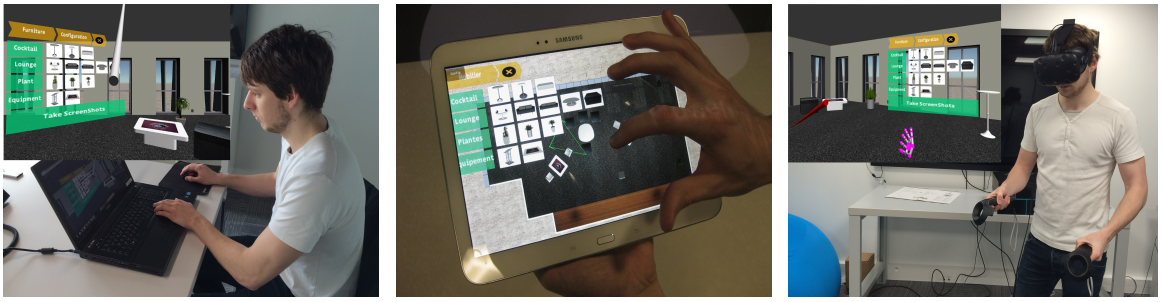
```
<3DPlasticToolkitConfig>
<TaskConfig>
<NeedTask taskName="SelectionManipulation" taskId="0"/>
<NeedTask taskName="FurnitureControl" taskId="1" topTask="0"/>
<NeedTask taskName="Navigation" taskId="2"/>
</TaskConfig>
<MainDisplay deviceName="ScreenMonitor" displayName="MONITOR"/>
</3DPlasticToolkitConfig>
```

Listing 5.1 – The XML task configuration file of the furniture planning application. Here, the applcation is defined by three high level tasks: Selection Manipulation, Navigation and Furniture Control. In this example, the defined main display is a screen monitor.

PC monitor.

The deployment of the application on multiple platforms

As explained in Section 4.7, in 3DPlasticToolkit, one application component is automatically associated to each task according to the context of use. For this application, concrete application components for navigation and selection/manipulation were already developed because they have been created for previous prototypes. It demonstrates the re-usability capabilities offered by 3DPlasticToolkit. Only the component that corresponds to the furniture control task has been specifically developed. Here, we describe the instantiated components for three very different platforms.



(a) (b) (c)

FIGURE 5.2 – Three different types of platforms on which the furniture application can be run: (a) a desktop platform (b) a mobile platform (a tablet), (c) an immersive platform (an HTC Vive).

First, as shown in Figure 5.2a, the application can be used on a desktop platform. This platform is simply composed of a monitor, a mouse and a keyboard. For the selection and manipulation task, a 3D ray-based interaction technique is proposed. The logical driver uses the mouse position to control the ray extremity and the buttons for selecting and grabbing as detailed in Section 4.5. For the navigation task, a walking navigation metaphor is deployed. The associated logical driver uses the arrows of the keyboard in order to translate and rotate the user's point of view. For the furniture application control task, a 3D menu is deployed and placed according to the user's point of view. This menu that can be shown and hidden gives access to the different events of the furniture control task for adding a furniture in the scene, loading the current layout and saving the current one.

Second, as shown in Figure 5.2b, the application can be used on a mobile platform. Here the platform is an Android tablet. In order to handle different exploitation systems, we use the portability capabilities offered by the current game engine used: Unity3D. For the selection and manipulation task, a 2D cursor interaction technique is deployed on the tablet. The chosen logical driver uses the multi-touch capabilities of the tablet. With this technique the user can translate the objects onto the floor with one finger and rotate them around the up axis with two fingers. For the navigation task, a pan and zoom navigation technique is deployed. The 3D menu for the application control task is also deployed on this platform. It is placed just in front of the camera and therefore it gives a 2D impression as shown on the Figure 5.2b.

Third, as shown in Figure 5.2c, the application can be used on an immersive platform. Here, we use an HTC Vive[1] that is composed of an HMD and two 6-DOF controllers with buttons and trackpads. For the selection and manipulation task, a virtual hand is used to select and catch the scene objects. The logical driver uses the position and the rotation of one of the controllers to set the hand's position and rotation. One button of the controller is used to close the hand and grab an object. For the navigation task, the user can navigate at scale one in the area defined by the head tracking zone. Moreover we combine it with a teleportation capability. With the second controller, the user can use a ray-based interaction technique to select a point in the scene where he wants to be teleported. As for the two previous examples, a 3D menu is also deployed for the furniture application control task.

Of course, the possible handled platforms are not limited to these three platforms. Indeed, for instance the application could also be used with a force feedback device and a stereoscopic screen, or with an HMD with a leap motion mounted on it or as shown in Section 6.5 with a CAVE system. Changing some devices directly at runtime is possible with this application thanks to 3DPlasticToolkit. It can give to the end user the possibility to totally interact differently with its application. For instance, in the case of the desktop platform, a razer hydra could be plugged at runtime. In that case, the users would not use anymore the mouse and the keyboard and he would be able to use 3D interactions with the two razer hydra controllers and to use the joysticks in order to navigate. For the "furniture application control task", for now we have just implemented an application component that corresponds to a 3D menu. A possible alternative could be to propose a concrete application component based on vocal recognition. Vocal commands would then be used in order to add a furniture, to load a pre-defined layout and to save the current one.

The collaborative extension

In an extension of this application, published as a research demonstration in [Le Chenechal et al., 2015], we have developed an asymmetric collaborative scenario between a customer immersed inside the room and an estate agent that guides and helps him. A video of this application can be found here: `https://www.dropbox.com/s/gu4sp3r7a7coyoj/FurniturePlanning.mp4?dl=0`. This collaboration is asymmetric because in this extension each role has its own interaction setup and its own interaction capabilities. The two users can communicate through microphones.

Both users have some common interaction capabilities that correspond to the tasks and applications components presented before. They can:

- **Select and manipulate** the scene objects ("SelectionManipulation" task),
- **Add, remove** furniture, **load** a pre-defined layout and **save** the current one ("Furniture-Control" Task).

Each user has also some specific interaction capabilities. First the customer can:

- **Make measurements** in the scene,
- **Navigate** in the scene ("Navigation" task presented before).

In the scenario that we have tested, the interaction setup of the customer is composed of an Oculus Rift DK2 HMD and a Razer Hydra. The user is totally immersed into the room with a first person view as shown in figure 5.3b. He uses a ray-based interaction technique to select and manipulate the objects. He can make the measurements using his 3D ray in a laser meter fashion.

Second, the guide can:

- **Send help** to the distant user on how to use the application,
- **Highlight objects** to catch the attention of the distant user.

[1] https://www.htcvive.com

(a) (b)

FIGURE 5.3 – The collaborative extension of the furniture planning application (a) The customer is immersed with a first person view in the shared virtual world with a HMD and interacts with a Razer Hydra. (b) The guide has a 3/4 top view of the scene and uses the zSpace interactive system.

The guide cannot navigate in the scene, he has just a 3/4 top view of the scene in order to see the entire room.

In our scenario, the interaction setup of the guide is the zSpace[2] interactive system composed of a 6-DOF tracker and a 24-inches co-located stereoscopic display. In the same way, with this setup as illustrated in Figure 5.3a, the guide uses a 3D ray-based interaction technique to select and manipulate the objects. Another button of the stylus is use to highlight the objects with the ray. A menu is also displayed to send help commands to the customer.

In this extension, the software framework that handles network synchronization is totally independent from 3DPlastictoolkit. Indeed, a software overlay of SmartFox Server[3] is used to manage collaboration. It ensures the synchronization of the shared VE between the different users as well as a consistent physical simulation. For awareness issues, the guide frustum and its 3D ray are represented in the shared environment. In order to handle concurrency when moving objects, the priority to move an object is given to the first user who grabs it. We could also have used the redistribution capabilities of 3DPlasticToolkit which are presented later in Section 6.

Conclusion and perspectives

This application enables us to demonstrate how 3DPlasticToolkit can be used in order to adapt the interaction techniques according to the available devices at runtime. As most of the components were already implemented before, the development of this application was very fast. The main work was to create the 3D content. For the collaborative extension, some of the components have been modified in order to handle some particular synchronization issues. However, in the same way the development of this extension has also been quick.

Here, we only demonstrate how the application can be used in a VR model. More work could be done to visualize and edit the created layouts in real situations with the help of AR. For example, visiting an under construction building with AR glasses to assess the quality of the current target layout. Interactions could be also adapted to AR devices.

To finish, the laying out scenario is not restrained to real estates need but could also be used for instance by heavy industries to imagine the future of factories.

[2] http://zspace.com/
[3] http://www.smartfoxserver.com/

5.3 Adapting the Visualization: The Frame-Cancellation Case Study

In this Section we demonstrate that 3DPlasticToolkit can also be used to adapt the content visualization of a 3D application. Here, we focus on a particular case of visual discomfort experienced by users with stereoscopic displays: frame cancellation. In order to deal with this discomfort we propose two visual effects that can be applied on 3D objects. We show that these effects can be easily applied with 3DPlasticToolkit. We also propose an alternative solution that consists in adapting a manipulation interaction technique. The results of this research work have been published in [Lacoche et al., 2015b].

Stereoscopic display is a sensorimotor contingency used for Virtual Reality (VR) applications in order to improve place and plausibility illusion [Slater, 2009]. Indeed, stereoscopy creates a three dimensional illusion by simulating stereopsis depth cue. One of the binocular depth cues introduced in Section 2.3.2. However, in some cases, stereoscopy may also lead to visual discomfort. When virtual objects in negative parallax (front of the screen) are overlapped by one of the screen borders (horizontals and/or verticals), two depth cues conflict. The first one, stereopsis, allows depth perception with the ability of our visual system to merge the two different images acquired by the two eyes. In negative parallaxes, this cue tells our visual system that the object is in front of the screen. The second cue is occlusion. If a screen edge is partially overlapping an object, the edge seems to be closer. Moreover, because of the limited display size, one part of an object can be clipped for one camera and not for the other [Mulder and Liere, 2000]. It is a particular case of binocular rivalry. Binocular rivalry occurs when the two eyes acquire two unrelated images. With horizontally aligned eyes, the parallax is only horizontal, and then only vertical edges are concerned by binocular rivalry. In case of a head tracking that considers head rotation around the 3 degrees of freedom, vertical parallax is also possible. Therefore, horizontal edges can also be concerned by binocular rivalry. This phenomenon, first described by Valyus [Valyus, 1966] is called "frame cancellation" and is described in Figure 5.4. This conflict reduces the illusion of depth in the perceived 3D image [Wartell, 2002] and creates an unpleasant effect sometimes called "eyestrain" for people watching the screen [Lipton and Akka, 2010]. These consequences of frame cancellation have been observed in [Ardouin et al., 2011]. As frame cancellation sometimes induces binocular rivalry, the problem is reinforced.

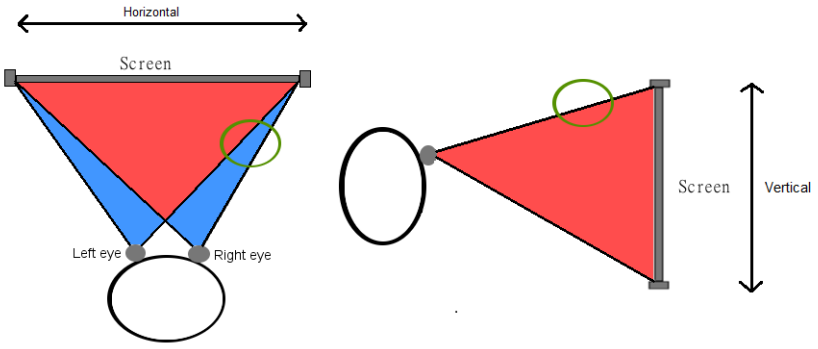


FIGURE 5.4 – Top view and side view of camera volumes with horizontally aligned eyes. In red, the area without conflict. In blue, the area where objects will be visible only by one camera (binocular rivalry). The green circle represents a virtual object subject to frame cancellation. In case of non horizontally aligned eyes, the blue area could also be visible on the side view.

Frame cancellation is mainly caused by the physical limitations of most available visual displays: they do not fill the entire field of view of the observers. CAVE^{TM} systems and head-mounted displays (HMD) are two display types that can avoid the problem by totally immersing the user in the virtual world thanks to a wide field of view. Nevertheless, when such a display is not available, we need to adapt the rendering and/or the interaction in order to reduce the frame cancellation effect. In these Section, we demonstrate that these adaptations can be directly performed with 3DPlasticToolkit. Indeed, the visual effects and the interaction technique are implemented into

application components developed with the model described in Section 4.5. This Section describes these different new effects and also how they can be automatically deployed when the main display is a simple stereoscopic screen.

Related Work

Adapting the rendering to deal with frame cancellation has been addressed for offline and real-time purposes. Indeed, 3D movies as well as interactive 3D applications can suffer from this problem.

One common approach consists in redefining the display window with virtual black bands on the screen borders when some objects are subject to the frame cancellation effect. The Floating Window introduced by Autodesk [Autodesk, 2008] for offline rendering and the Cadre Viewing approach proposed by Mulder et al. [Mulder and Liere, 2000] for real-time contexts are solutions based on vertical black bands. The virtual objects are then perceived in positive parallax related to the virtual window, and no part of a virtual object in negative parallax is then visible by only one eye. The main drawback of this approach is field of view reduction. Moreover, the occlusion issue with the horizontal screen borders is not addressed.

To continue, the Cyclopean Scale introduced by Ware et al. [Ware and Fleet, 1997] proposes to scale the virtual environment (VE) about a point between the observer's two eyes in order to always place it just behind the screen in terms of stereoscopic depth. This method avoids frame cancellation by always placing the VE in positive parallaxes or very small negative ones. Therefore, the Cyclopean Scale reduces the possibility of simulating popping-out effects offered by stereoscopic rendering. It is particularly limited in the case of co-located interactions.

One of the most relevant approaches has been proposed by Ardouin et al. [Ardouin et al., 2011]. The effect is called the Stereo Compatible Volume Clipping (SCVC) and consists in rendering only the part of the virtual objects that are in the conflict-free area: the Stereo Compatible Volume (SCV). This is done by clipping all the object's parts that intersect the blue area, as shown in Figure 5.4. One possible result is given in Figure 5.5. This technique always ensures two consistent right and left images, does not reduce the field of view, and is fully compatible with head tracking. However all conflicts are not solved. Indeed, the technique mainly focuses on solving the binocular rivalry issue, and no part of an object in negative parallax is visible by only one eye, but may still be perceived as roughly clipped by a horizontal or by a vertical screen border, which disrupts the object form perception.

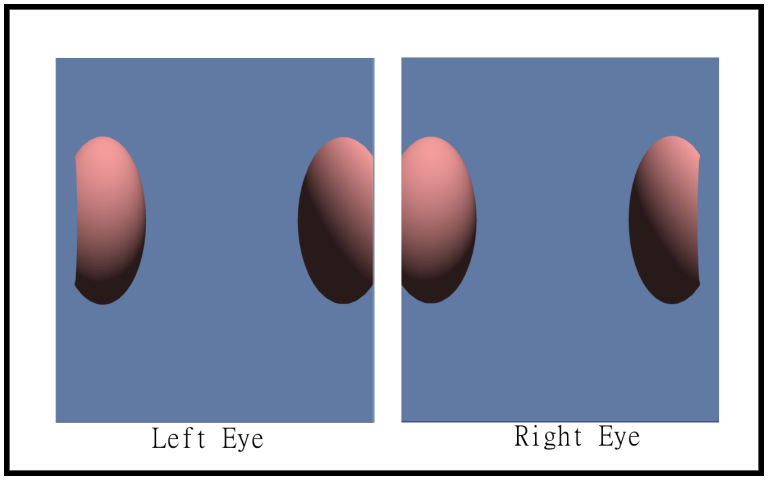


FIGURE 5.5 – SCVC approach illustration. The clipping of the 2 spheres in negative parallax avoids inconsistent left and right images

Most of the current methods focus on adapting the rendering to deal with frame cancellation. Nevertheless, we can also find an approach that consists in adapting an interaction technique to reduce the frame cancellation effect. Wartell et al. [Wartell et al., 1999] introduce a navigation technique for a large dataset based on scaling, panning, and rotating. During the navigation,

the technique performs automatic translation of the scene perpendicular to the projection plane in order to always minimize frame cancellation. The translation is computed by detecting the collision between the scene and a plane slightly above the projection plane, and parallel to it, with the help of the depth buffer. Such a method can be compared to the Cyclopean Scale because it also avoids important negative parallax values. Therefore, it reduces the possibility of simulating popping-out effects.

Two visual effects based on alpha blending to avoid frame cancellation

The main drawbacks of current methods are field of view reduction, scene scale alteration or a too rough clipping of virtual objects. Moreover no approach considers the horizontal edges of the screen in case of X-aligned eyes while they may also occlude virtual objects in negative parallax.

That is why we introduce two approaches that aim at solving these drawbacks by proposing a smoother technique based on alpha blending that will affect virtual objects in negative parallax.

Progressive Stereo Compatible Volume Clipping (PSCVC)

The main drawback of the stereo compatible volume clipping is the rough clipping of virtual objects. This clipping may lead the observer to a visual discomfort. Particularly when the clipping lets appear a virtual object behind the clipped one. That is why we propose to apply SCVC progressively by doing the clipping with an alpha blending method in order to create a smoother effect.

To compute the alpha value applied to each pixel (x, y), first we have to evaluate the distance $d(x, y)$ of the corresponding 3D point to the frame cancellation area. To do so, we compute the distance of this 3D point to the plane that is supposed to clip the object in SCVC. This plane selection is described in [Ardouin et al., 2011]. For non X-aligned eyes two planes can clip the object for a camera. In that case we compute two distances, one for each plane, then the minimal distance is taken to compute the alpha value (0: totally transparent, 1: totally opaque). The alpha factor that linearly decreases when the 3D point gets closer to the plane is computed with respect to the equation 5.1.

$$Alpha(x, y) = 1 - \frac{d(x, y) - distMinPlane}{distMaxPlane - distMinPlane} \tag{5.1}$$

where distMinPlane is the distance where the transparency begins (Maximale opacity) and distMaxPlane the distance where the transparency ends (Minimal opacity). Both values are expressed in meters, distMinPlane can take a negative value in order to really clip the virtual object after the SCVC planes. The alpha value is then clamped between 0.0 and 1.0. A result is given in figure 5.6 where we apply progressive SCVC on the two spheres subject to frame cancellation. For this image we have set distMinPlane to -0.02m and distMaxScreen to 0.05m, the sphere radius is 0.1m. Compared with SCVC in figure 5.5, we observe a smoother result that may lead to a better visual comfort in frame cancellation situations.

Virtual alpha blended window (VABW)

As stereo compatible volume clipping [Ardouin et al., 2011], progressive SCVC considers the two horizontal edges only in case of head-tracking with non X-aligned eyes. Even without head-tracking the horizontal edges may occlude virtual objects in negative parallax and cause stereo cues conflicts. The virtual alpha blended window aims to take into account this issue by considering vertical edges of the screen as well as the horizontal ones in all cases. This approach consists in applying a progressive alpha blending over the screen edges to the virtual objects in negative parallax for the two cameras. Contrary to SCVC and progressive SCVC, with this effect, when an object is close to a screen edge its rendering is affected for the two cameras. Indeed, we suppose that modifying only one view can lead to an increased visual discomfort by generating a difficulty to fuse the two images.

As for progressive SCVC, we propose to apply a linear decreasing alpha blending but this time on the two axes of the screen. Our method considers the screen as a space where all pixels

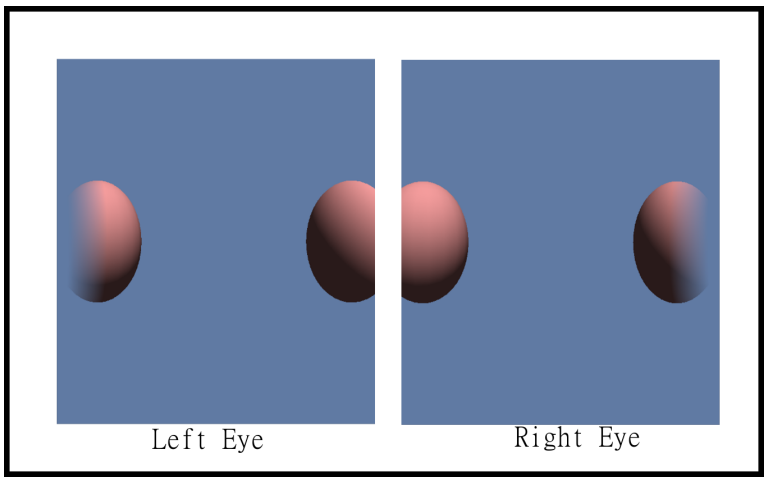


FIGURE 5.6 – Progressive SCVC applied to the virtual objects close the screen edges

coordinates take a value between [-1.0 , -1.0] (bottom left corner) and [1.0 , 1.0] (right top corner). The transparency is computed in order to get a square with rounded angles shape in this space. This shape is a "squircle" which is a particular case of a superellipse. A square shape would have been really coarse on the screen corners, and a circle shape would have to much erase the objects on these corners. For both cameras, for a pixel (x, y), we compute its alpha factor only if the corresponding position in the 3D world is front of the screen (negative parallax). Concerning the alpha factor computation, we first compute its distance to the squircle center. This computation is described in equation 5.2.

$$d(x, y) = \sqrt[4]{x^4 + y^4} \tag{5.2}$$

This value is then used to compute the alpha factor in equation 5.3.

$$Alpha(x, y) = 1 - \frac{d(x, y) - distMinScreen}{distMaxScreen - distMinScreen} \tag{5.3}$$

Where distMinScreen is the distance where the transparency begins (Maximale opacity) and distMaxScreen the distance where the transparency ends (Minimal opacity). The alpha value is then clamped between 0.0 and 1.0.

As the alpha computation only depends on the 3D positions of the scene objects and to the corresponding projected pixels positions, it is totally independent of a possible head tracking.

One possible result is given in figure 5.7 where we apply our filter on the two spheres subject to frame cancellation. For this rendering we have set distMinScreen to 0.8 and distMaxScreen to 0.97. We can see on this figure that the virtual objects are smoothly clipped by the horizontal and the vertical screen edges: this clipping would have been rougher with state of the art approaches.

Implementation in 3DPlasticToolkit

In order to deal with frame cancellation in 3DPlasticToolkit we added a built-in high level task: "FrameCancellationSolver". If the developer wants to handle frame cancellation in his/her application he/she just has to define this task as needed in the 3DPlasticToolkit configuration file. This task and its compatible components are shown in Figure 5.8.

Our different visual effects are implemented in the same application component which is defined as compatible with the "FrameCancellationSolver" task. This component is shown in Figure 5.8. The abstraction of this component is named "VisualEffectFrameCanellationA". The abstraction role is to collect the different information about the current stereoscopic configuration and compute the view frustum of each camera. Then these information are transmitted to the rendering presentation facet through the controller facet. The component is compatible with three rendering

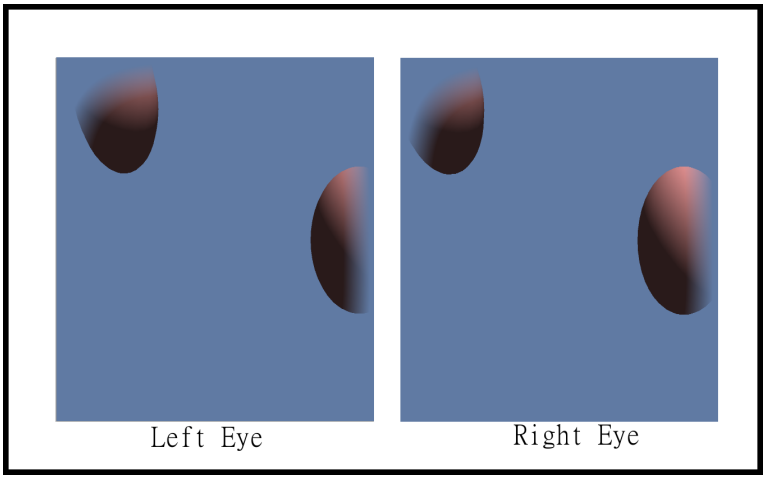


FIGURE 5.7 – Virtual alpha blended window effect applied to the virtual objects close the screen edges

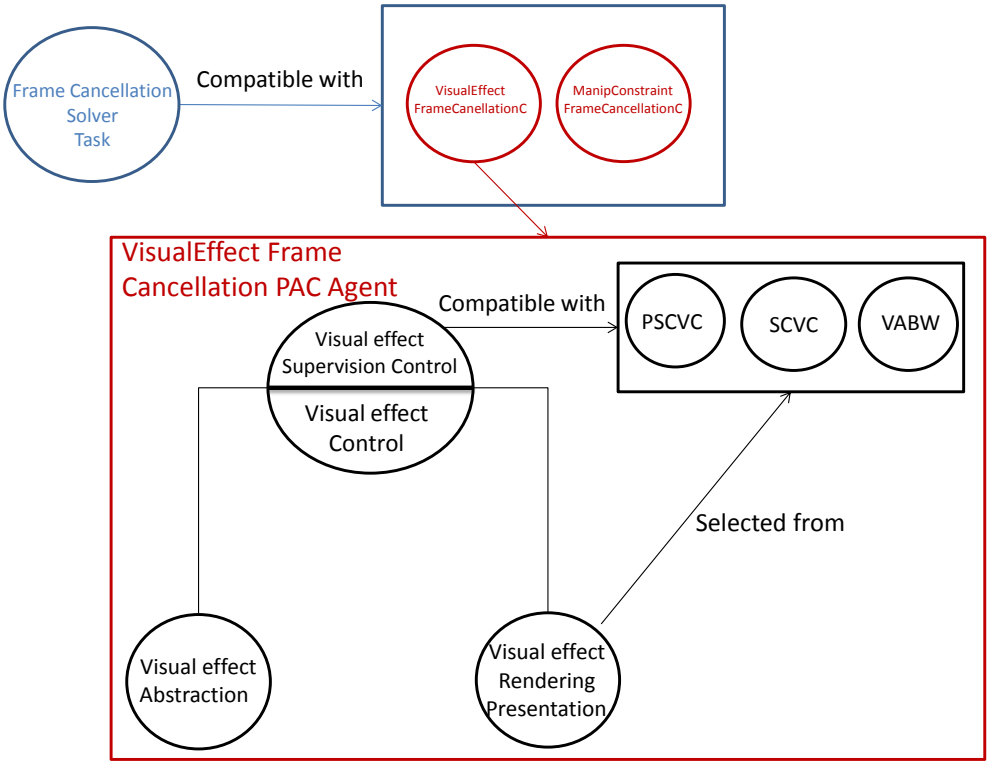


FIGURE 5.8 – The "FrameCancellationSolver" task and its compatible application components. Here, the application component that applies a visual effect is described.

presentation facets that implement three different effects: SCVC [Ardouin et al., 2011] from the state of the art, and PSCVC and VABW our two visual effects.

In order to be deployed only if the screen is a flat stereoscopic screen the scores are set to a positive value only in this case. This is done in the compatibility configuration file fulfilled by the developer as detailed in Section 4.7. In other cases the components get a negative score and they are not deployed. The part of the compatibility configuration file that set these scores is shown in Listing 5.2.

The three effects are implemented in each rendering presentation facet with Surface Shader in

```
1 <TaskCompatibility taskName="TaskFrameCancellationSolver" componentName="
      VisualEffectFrameCanellationC" displayType="FLAT" score="1.0">
2   <PresentationCompatibility componentName="VisualEffectFrameCanellationC" presentatioName="
        VisualEffectFrameCanellationSCVC" score="0.5"/>
3   <PresentationCompatibility componentName="VisualEffectFrameCanellationC" presentatioName="
        VisualEffectFrameCanellationPSCVC" score="0.8"/>
4   <PresentationCompatibility componentName="VisualEffectFrameCanellationC" presentatioName="
        VisualEffectFrameCanellationVABW" score="1.0"/>
5 </TaskCompatibility>
```

Listing 5.2 – The scores assigned to the visual effects that handle frame cancellation. The score of the full application component is positive only if the main display is a flat screen. Here, the biggest score is given to VABW effect.

Unity3D [4]and are fully compatible with simple materials like diffuse or specular and more complex ones such as bumped, reflective, etc. These effects are applied to every 3D objects in the scene. It is also possible to apply the effects only on particular objects by tagging them.

One interaction technique that uses collision detection to avoid frustum clipping

In order to avoid frame cancellation, in the context of an application that involves manipulations, we propose to adapt the user interaction on the virtual objects by constraining the position of the manipulated object into the Stereo Compatible Volume (SCV) described by Ardouin et al. [Ardouin et al., 2011]. Indeed, when manipulating virtual objects, the user may place them in a conflict area corresponding to the outside of the red zones in Figure 5.4. That is why, we propose a new approach to avoid such a situation during the object manipulation. Contrary to the method introduced by Wartell et al. [Wartell et al., 1999], ours does not reduce the possibilities of creating the popping out effect during the manipulation. This interaction technique is implemented in the second application component compatible with the "FrameCancellationSolver" task, which is named "ManipContraintFrameCancellation".

The goal of this component is to constrain the position of a manipulated object inside the SCV. With this component, we attach to a manipulated object an encapsulating rigid body controlled by a physics engine detecting collisions with the two cameras frustums modeled with collision planes. The result is a collision detection between the object and the intersection of the two frustums. The planes that define the SCV are updated in real time in the physical engine according to the user's head position. Then, according to the user's movements a collision detection algorithm ensures that the manipulated object never gets outside the area defined by these planes. A particular case to handle is when the user moves close to the screen. In that case the rigid body reacts as if it was pushed by user's head. Actually, it is pushed by the intersection planes of the two frustums, which ensures that the object is always free of stereoscopic conflicts.

If the size of a virtual object is taller than the screen's physical size, we can not move it in front of the screen since our algorithm is going to keep it behind the screen plane because of the collision with the frustums' planes. A solution to overcome this limitation can be a smart merge of this collision detection-based approach with one of the visual effects presented before. For example, the user could enable a scaling mode in order to increase the size of an object and see it with more details. In that case, the collision detection algorithm may be disabled and one of our two rendering effects may be enabled in order to reduce the frame cancellation effect, possibly caused by the scaling. An illustrative use case is provided Figure 5.9.

For now, in 3DPlasticToolkit the application component that implements this adaptation also includes the implementation of the manipulation. As perspective of work, this component must become independent from the manipulation and must be possibly compatible with the different manipulation techniques presented in Chapter 4. Another possibility could consist in making this component compatible with haptic devices thanks to the implementation of a particular logical driver. Indeed, it would let the end-user physically feel the constraint of the SCV with haptic feedbacks.

[4]http://docs.unity3d.com/Manual/SL-SurfaceShaders.html

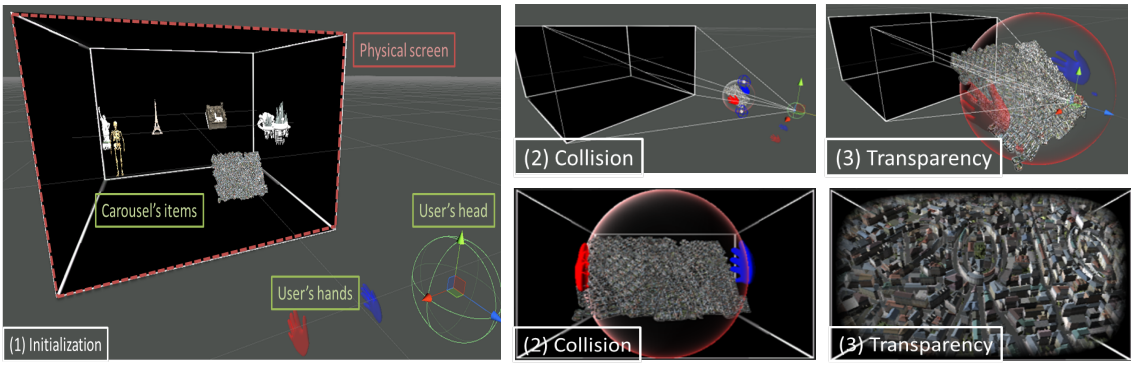


FIGURE 5.9 – Illustrative use case (inspired from [Gaucher et al., 2013]) of the manipulation technique smartly merging the two proposed approaches to handle frame cancellation. The user is interacting with a virtual carousel carrying several virtual objects (1). The user can turn the whole carousel and manipulate the front object. The object can be a map that the user can bring very close to him/her with the collision detection-based approach activated (2). Then, he/she can switch the manipulation mode to scale the map and perceive small details with the virtual alpha blended window approach activated (3). The first line shows the scene from an external point of view while the second line is captured from the user's point of view.

Conclusion

In this Section we demonstrate that 3DPlasticToolkit can be used in order to adapt the content visualization to deal with a visual discomfort created by stereoscopy: frame cancellation. An interaction technique adaptation is also proposed to deal with this issue. Each effect is implemented in an application component that is compatible with a built-in high level task that the developer can defined as needed in 3DPlasticToolkit if he/she wants frame cancellation to be handled.

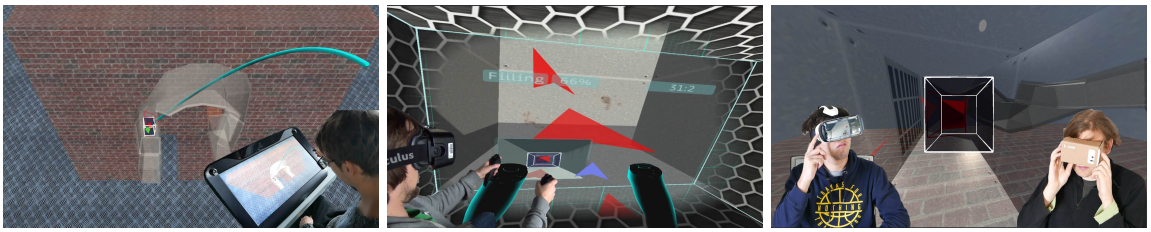
First, we propose two visual effects, PSCVC and VABW that are based on a progressive alpha blending that lets the objects disappear close to the physical screen edges. These effects are implemented in 3DPlasticToolkit with shaders and can be automatically deployed when the display detected is a flat stereoscopic screen.

Second, we introduce a new approach designed for 3D user interactions. Instead of adapting the rendering of objects in frame cancellation areas, the technique proposes to constrain the interaction in order to avoid this kind of situation. The technique is based on a collision detection system in order to constrain the manipulated object in the stereo compatible volume described by [Ardouin et al., 2011]. This adaptation is also implemented in a 3DPlasticToolkit application component.

In Appendix A, we present an evaluation of these different methods. A video of this evaluation can also be found here: `https://www.dropbox.com/s/o7bhfnrihdo313p/FrameCancellation.mp4?dl=0`.

5.4 Adapting the Visualization and the Interaction Techniques: the Co-Manipulation Case Study

In the context of the IEEE 3DUI contest 2016, the goal was to propose a set of interaction techniques in order to achieve collaborative manipulation tasks in VR. This contest consisted in a challenge between multiple teams for the proposition of a solution adapted to the manipulation tasks. The proposed tasks consisted in overcoming different obstacles by moving, rotating and scaling an object collaboratively. This kind of collaborative manipulation can be used to simulate industrial tasks such as in an automotive factory where cumbersome objects must be carried by several collaborators [Aguerreche et al., 2009]. Multiple constraints had to be respected such has integrating a collaboration between two are more users with MR devices (no desktop ones). The proposition also had to be innovative, easy to use and to learn and had to be tested on some provided 3D



(a) (b) (c)

FIGURE 5.10 – Collaborative manipulation of a virtual object (here, a cube) based on an asymmetric setting between two users who can be helped by two additional users. With 3DPlasticToolkit each user benefits from the adapted interaction technique, the adapted point of view and the adapted scene visualization according to his/her device setup in order to ensure an efficient collaboration. (a) The first participant has a global view of the scene and moves the object with a 3D bent ray. (b) The second user is placed inside the object and precisely rotates and scales it. (c) Two additional roles can be added. The first one helps to scale the object using a third person view of it. The other one is a spectator who switches between the other participants' viewpoints and helps them with oral communication.

scenes such as the labyrinth shown in Figure 5.10.

To do so, we proposed an asymmetric collaboration between two or more users with different devices (cf. Fig. 5.10). A presentation of this contribution in the video: `https://www.dropbox.com/s/w66xxgx5ivmc1rh/3DUIContest2016.mp4?dl=0`. This choice to propose an asymmetric collaboration and the needs to handle multiple devices and multiple interaction metaphors led to technical constraints that have been easy to solve with 3DPlasticToolkit. In this scenario each person benefits from a different role according to his/her available devices thanks to our plasticity mechanisms.

With the help of 3DPlasticToolkit, adapted interaction techniques are automatically associated to the suited users with our plasticity models. Therefore each user automatically benefits from the interaction technique and the point of view that matches his/her device setup. The concrete devices used in this demonstration can be exchanged by others with the same capacities thanks to 3DPlasticToolkit. In the same way the interaction technique of each user could also work with other types of devices if we implement other logical drivers. Some alternatives are proposed in this Section. Moreover, we demonstrate that 3DPlasticToolkit is also used in order to adapt the visualization of the scene for the users that wears an HMD. Indeed, this demonstration was also the opportunity to test a visual effect inspired from the ones presented in Section 5.3 in order to deal with cybersickness.

In this demonstration, the software framework that handles network synchronization is totally independent from 3DPlastictoolkit. Indeed, the same software overlay of SmartFox as the one discussed in Section 5.2 is used to manage collaboration.

Roles and interaction techniques

For this demonstration of 3DPlasticToolkit, we propose an asymmetric collaboration where each user benefits from interaction capabilities adapted to his/her interaction devices in order to move, rotate and scale a virtual object. As proposed by Pihno et al. [Pinho et al., 2002], our approach splits the DOF of the manipulated object between collaborators in order to maximize the system efficiency. Our approach is based on two main roles. First, the *Giant* has a global viewpoint of the scene and controls the object's translation. Second the *Ant* is inside the object and sets its scale and rotation. In our shared multi-scale VE, this approach allows the *Giant* to quickly move the object, while the *Ant* performs better accurate transformations. Two other uses can also be included. The helping user with a 3^{rd} person viewpoint of the object can also set the scale of the manipulated object by asymmetrically integrating its action with the *Ant*'s one. A spectator can also be included, he can share one of the other participants' viewpoints. The visualization of the scene is also adapted to each hardware setup. Indeed, the interaction technique for the *Ant* also

displays some feedbacks to help him/her during his/her manipulation task. As this user is also moved by the *Giant*, we also propose a visualization adaptation that consists in a visual effect that aims at reducing cybersickness. Indeed this situation can cause symptoms similar to motion sickness [Fernandes and Feiner, 2016] such as headaches, nausea, fatigue and disorientation, which can really decrease the user experience.

The collaborators benefit from complementary interaction techniques to perform collaborative manipulation tasks that need translating, rotating, and scaling. An example of manipulation task that we have tested is given in Figures 5.10a, 5.10b and 5.10c. The goal is to pass a cube through a labyrinth while maximizing the filling of this labyrinth with the manipulated object.

In 3DPlasticToolkit, the application of each user is exactly configured in the same way with two high level tasks:

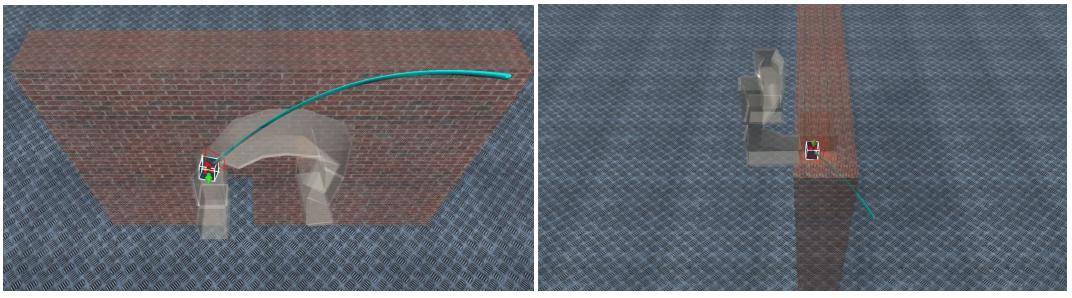
- **"CollaborativeMazeManipulation"**: this is the main task of the application. Each application component compatible with this task gives to the user the possibility to manipulate the object. As said, our collaboration is asymmetric, therefore each application component implements a particular role for the end user and does not give him/her the same interaction capabilities.

- **"CybersicknessSolver"**: as some users with HMDs may be moved in the VE during the simulation, it could disturb them and create a cybersickness effect. Therefore, we added a built-in high level task and compatible application components that aim at reducing cybersickness.

Global View: the *Giant*

As illustrated in Figure 5.11, the first user has a global view of the scene and can roughly manipulate the object in order to move it really fast in easy passages. This user can translate the object. As shown in Figure 5.10a, in our scenario this first user is interacting on a zSpace. This device is composed of a 3D stereoscopic display with head tracking and of a 3D tracked stylus for interacting. The zSpace screen is used to create a window to the VE.

In 3DPlasticToolkit, the application component compatible with the "CollaborativeMazeManipulation" task and that is deployed in this case is named "GiantManipulation". With this component, the user on the zSpace (*Giant*) can translate the object with a bent ray inspired from the interaction technique proposed by Riege et al. [Riege et al., 2006]. With the currently implemented logical driver, the ray is controlled in position and rotation by the stylus tracked by the zSpace. One button is used for object grabbing, and the other buttons are used to switch between four point of views: front, left, back and right. The ray is bent during the object translation in order to respect three constraints:



(a) (b)

FIGURE 5.11 – The *Giant* has a global view of the scene. He can translate the object with a 3D bent ray.

- The physical collider of the object avoids it to pass through other objects. The ray is bent accordingly.

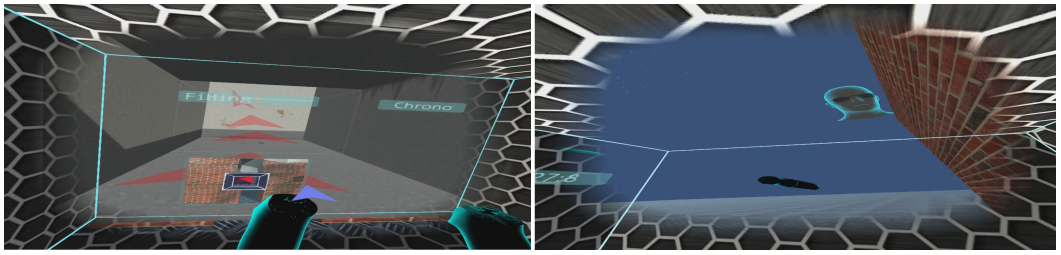
- We limit the ray extremity speed when an object is grabbed. The goal is to not disturb the distant user inside the manipulated object and reduce his/her cybersickness.

- Third, a last constraint is optional. We added an active help for the translation. It is a *magnetic path* that represents the perfect path to follow. The manipulated object can be connected to the closest point on this path and the ray is bent accordingly.

To make the others understand the *Giant*'s actions, his head, stylus and 3D ray are rendered in the shared environment.

For this user, nothing is deployed for the "CybersicknessSolver" task. Indeed, this user only sees the VE through a small stereoscopic screen. Therefore the user is not fully immersed in the VE and the cybersickness effect is reduced.

Micro View: the *Ant*

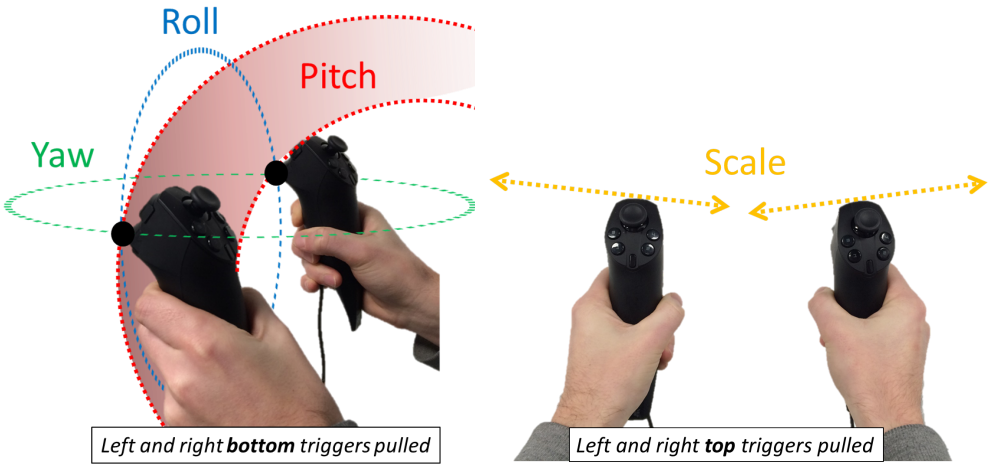


(a) (b)

FIGURE 5.12 – The *Ant* is placed inside the manipulated object. He can rotate and scale the object. He can also see where is the *Giant* (the blue head).

For this role, the user is placed inside the manipulated object as illustrated in Figure 5.12. This position enables him/her to manipulate the object with a fine accuracy. His/her role is essential to overcome difficult passages and maximize the courses filling by the object. He/she can scale the object and he/she can rotate it. His/her scale in the scene also offers him/her direct interaction possibilities such as pushing buttons to trigger different actions. The second user visualizes the scene with a Head-Mounted Display (HMD), here an Oculus Rift as shown in Figure 5.10b. He/she is interacting with a Razer Hydra composed of two 3D tracked controllers.

In 3DPlasticToolkit, the application component compatible with the "CollaborativeMazeManipulation" task and that is deployed in this case is named "AntManipulation". To be deployed with the presented logical driver, this component needs two available 6-DOF controllers with buttons. With this component the *Ant* is placed inside the manipulated object. The logical driver of the component enables him/her to scale and rotate with the two Razer Hydra controllers thanks to bimanual metaphors inspired from the work of Cutler et al. [Cutler et al., 1997]. These manipulations are performed with a fix reference: the object front face. This reference face can be changed with the Razer Hydra joysticks. With this application component, as shown in Figure 5.13a, the rotation is made with a modified version of the grab-and-twirl metaphor. Compared to the classical version, the pitch rotation is performed with a metaphor close to a plane yoke by orienting the two controllers to the top or to the bottom. The scale of the object is uniformly controlled with a grab-and-scale metaphor by bringing closer or further the two Razer Hydra controllers while pushing two corresponding buttons (cf. Fig.5.13b). Manipulations of the object are physically constrained, thus, it can not pass through an obstacle. Two visual feedbacks are rendered to make the *Ant* understand the distance between the manipulated object and possible obstacles. First, we render particles at the collision points. Second, a virtual grid visible in blue at bottom in Figure 5.10b, parallel to the user current front face, is displayed outside of the object.



(a) (b)

FIGURE 5.13 – The two metaphors used by the *Ant* to rotate and scale the manipulated object.

With our plasticity models and 3DPlasticToolkit, this role could be also available on other kinds of devices. For instance, it could be seamlessly adapted to a platform composed of an HMD and of a 6-DOF controller. The interaction would remain the same. Desktop devices could also be used. Indeed, the bent ray could be controlled with a 2D mouse as for the 3D ray-based interaction technique introduced in Section 4.5.

In some particular situations, local interactors can be integrated at scale one in the VE. For instance, in the example given in Figure 5.14, virtual buttons are available in the scene for triggering different actions such as opening doors. These virtual buttons are too smalls for the *Giant*. Therefore, as the application component also renders the Razer Hydra controllers in the VE. They can be used by the *Ant* to interact with these local interactors. Here, by touching one virtual button with one of the controller, the *Ant* can trigger the corresponding action. Our plasticity models are used to render these controllers automatically into the VE. We exploit the description of the device that is associated with the current logical driver. Indeed, as detailed in Section 4.4, with our device model we can get the list of physical objects from the device and each of these physical objects can be described with the path of 3D model that represents it. Here, we transmit these paths to the rendering presentation that loads and displays the associated models of the controllers. This is a good example of how the physical representation of a device included in our device model can be used to adapt the application. For example, if the Oculus Rift and the Razer Hydra were replaced by an HTC Vive, the 3D models that correspond to the controllers of the Vive would seamlessly replace the Razer Hydras ones in the VE.

To guide the *Ant* when he/she is placed in a closed environment such as the provided labyrinth, different spatial cues can help him/her. They are shown in Figure 5.10b. First, a World-In-Miniature [Stoakley et al., 1995a] shows a third person view that focuses on the manipulated object. Second, an arrow simulates a compass to show him/her the direction to follow. Third, in the labyrinth, the path to follow is also indicated with arrow signs.

Regarding the "CybersicknessSolver" task, the *Ant* is totally immersed in the VE and can be moved by the *Giant*. Therefore, here we deploy an application component that implements a visual effect that aims at reducing the cybersickness experienced by this user. The effect can be compared to the one propose by Fernandes and Feiner [Fernandes and Feiner, 2016], which consists in dynamically adapting the field of view dimension in order to reduce cybersickness. Our effect is also inspired from the visual effects that we have proposed and compared in Section 5.3 for dealing with frame cancellation. Our effect does not just reduce the field of view but it also makes the peripheral view of the user consistent with his/her head movement. Therefore, as shown in Figure 5.15, at the center of his/her view the user still views the VE while in the extremities of his/her viewport we display something consistent with his/her head movement. The transition is exactly the same as the one presented in Section 5.3 for the VABW effect. Indeed the VE rendering

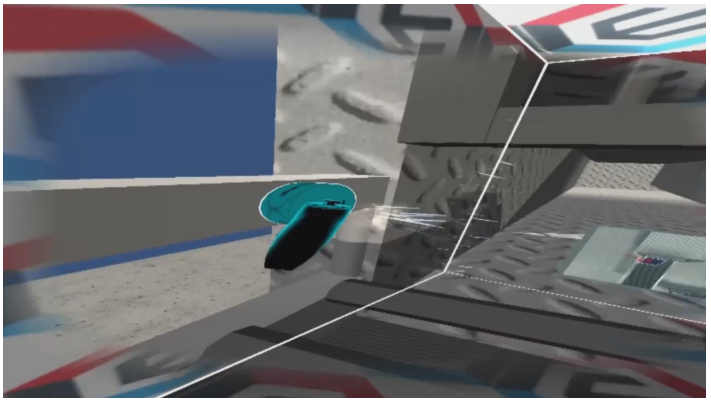


FIGURE 5.14 – As the *Ant* is placed at scale one in the shared VE, he/she has direct interaction capabilities with this VE. For instance, here it can push a virtual button in order to trigger an action in the scene (here, open a door). This kind of actions could not be performed by the *Giant*.

is progressively faded to head consistent area of the viewport by decreasing its opacity with the "squircle" distance formulate detailed in 5.2. This effect is applied before the viewport deformation applied by the HMD. We have experimented two kinds of contents on the peripheral view:

- **A grid displayed on a fixed object**. Here as shown on top right of Figure 5.10b, the user is placed inside a virtual object (a cube or a sphere) that is always fixed. The texture of this object is a simple grid. When the user is translated this object follow the user's head position. In this demonstration, as the user is placed inside the manipulate cube, the effect is directly applied on it.

- **The images from two additional cameras**. This case is possible if the HMD used is a video see-through HMD. We tested this effect with the OVRvision mentioned in Section 2.3.2. Therefore, as shown in Figure 5.15 the peripheral view will be fulfilled with the real world view, which will be consistent with the user's head movement.

With this kind of effect, the user's peripheral view is less disturbed by translations performed by the *Giant*. Some preliminary evaluations of this effect have been performed in another context and have shown good results. However, a formal evaluation of the effect needs to be performed if we really want to validate its efficiency. As shown in Figure 5.15, the area of the peripheral view is adjustable. Contrary to the effect proposed by Fernandes and Feiner [Fernandes and Feiner, 2016], for now, the area is fixed at runtime. Nevertheless, the impact of this area should also be investigated. One of our hypothesis is that the cybersickness feeling will decrease when we increase this area. If this hypothesis is verified it would let us dynamically adapt this area if we are able to measure the importance of cybersickness felt by the end user, for instance with physiological sensors.

Thanks to our plasticity models the devices used could be seamlessly replaced by equivalent ones. For instance, the combination Oculus Rift/Razer Hydra could be replaced by the HTC Vive that is composed of an HMD and two 6-DOF controllers with buttons. The Razer Hydra could also be replaced by the STEM system [5] that also provides similar tracked controllers. We could also imagine another logical driver or another application component in order to change the interaction technique of the *Ant*. For example, an haptic device could be used to change the position, the scale and the orientation of the manipulated object. In that case, haptic feedbacks would help the *Ant* to understand the presence of close obstacles. With our plasticity models and 3DPlasticToolkit, these adaptations can occur at runtime. It could allow the end-user to change dynamically his/her way to interact with the application only by replacing a device currently plugged by another one (equivalent or not).

[5] http://sixense.com/wireless

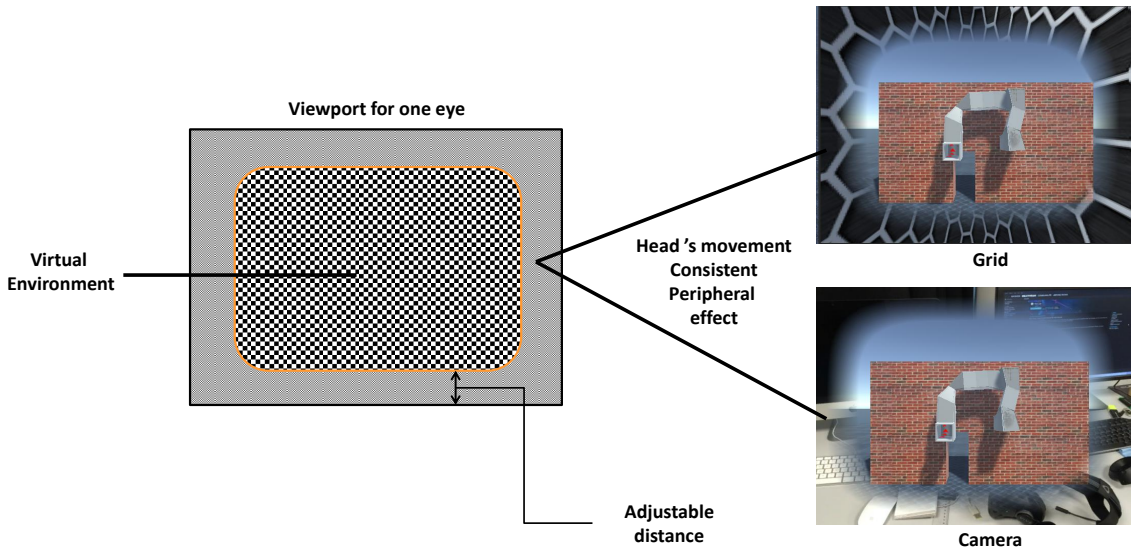


FIGURE 5.15 – Our peripheral effect that aims at reducing cybersickness. This effect makes the peripheral view of the user consistent with his/her head movement. It progressively blends the rendered VE on the screen extremities to a grid displayed on a fixed object or to the images acquired by the cameras of a video see-through HMD.

Third Helping User

This user has a third person view of the manipulated object. His/her role is to help the *Ant* to scale it. Therefore, the scaling capability is shared by these two users. This role and this interaction technique are designed for a user that does not have any 3D interaction device. The application component that corresponds to this technique and that is compatible with the "CollaborativeMazeManipulation" task is named "ScalerHelperManipulation". In our scenario, as shown in Figure 5.10c, the third user is interacting with a GearVR[6], an HMD with a 2D trackpad. The component "ScalerHelperManipulation" is automatically deployed for the "CollaborativeMazeManipulation" task by 3DPlasticToolkit when the current device used is a GearVR or an equivalent one. In 3DPlasticToolkit, the logical driver compatible with the application component that corresponds to this role modifies the scale with slide gestures on the GearVR trackpad.

The scale control is shared between the *Ant* and this third helping user. To solve this concurrency, we add the factors that the two users want to apply to the scale. The scale is modified by taking care of environmental physical constraints.

As explained in the previous Section, thanks to our automatic adaptation process, the application component that implements the anti-cybersickness filter is also deployed because in our scenario this user is also wearing an HMD (the GearVR). In the same way, a virtual sphere is placed around the user head with the transparency effect as shown in Figure 5.15.

For this role, as the task to perform is relatively simple, a lot of different other devices could be used thanks to 3DPlasticToolkit. For example, a logical driver could be implemented in order to use vocal commands in order to change the scale or simply uses two buttons or one joystick. For the display, another HMD or a simple screen could also be used.

Spectator

The last role is a spectator. The application component that corresponds to this technique and that is compatible with the "CollaborativeMazeManipulation" task is named "SpectatorManipulation". In our scenario shown in Figure 5.10c, the device used is a Google Cardboard [7]. The component

[6] http://www.samsung.com/fr/galaxynote4/gear-vr/
[7] https://www.google.com/get/cardboard/

"SpectatorManipulation" is automatically deployed for the "CollaborativeMazeManipulation" task by 3DPlasticToolkit when the current device used is a Google Cardboard or an equivalent one equipped with at least one button. This component for this user is really simple. It only lets the user switch between the other participants' viewpoints. Multiple spectators can be included in the shared VE. A spectator cannot really act on the manipulated object but he can help the other users by giving oral instructions. For changing the point of view, the user has to pull the Cardboard trigger. Indeed, the logical driver of the application component that corresponds to the spectator only needs a button to work. Therefore, thanks to 3DPlasticToolkit the spectator can be deployed on a wide variety of platforms, even the more simple ones. A desktop platform could have also been used as well as a simple tablet. Another kind of logical driver could have also been implemented, for example one that use vocal commands in order to change the point of view.

If a HMD is used as display such as here with the Google Cardboard, the component that corresponds to the anti-cybersickness filter is deployed as explained in the two previous Sections. This is also needed for the spectator as he/she can follow a point of view that can move.

Conclusion & Perspectives

This application lets us demonstrate that 3DPlasticToolkit can be used to develop collaborative applications context and to adapt the interaction techniques and the visualization of each participant according to their available devices. This demonstration has received the second price of the IEEE 3DUI contest 2016.

For this application our plasticity models and 3DPlasticToolkit have been used to develop each interaction technique independently from concrete devices. Therefore, the application can be seamlessly adapted on a wide variety of platform that are equivalent to the currently used ones. Moreover, with the development of other logical drivers or other application components, it could be possible to use other type of devices. For instance, haptic devices could be exploited in order to help the different users to feel the obstacles. It has not been exploited for the 3DUI contest but the possibility given by our models to change dynamically the application at runtime (with the meta-user interface or by plugging new devices) could really benefit to the application. Indeed, it would allow the end-user to try dynamically other ways to interact with the application.

5.5 Conclusion

In this Chapter, we present three applications that enable us to demonstrate how our plasticity models can be used in order to adapt the interaction techniques and the content visualization. The first application demonstrates how 3DPlasticToolkit has been used to develop a furniture planning application. This application can be run on a wide variety of device setups thanks to our solution. The second one presents two visual effects that can be applied with 3DPlasticToolkit. These effects aim to reduce a visual discomfort experienced by users with stereoscopic displays named frame cancellation. A formal evaluation of these effects has been performed and have shown good results. The last application proposes a collaborative scenario where multiple users collaboratively perform manipulations tasks. For this application, we propose an asymmetric approach. Indeed, with the use of 3DPlasticToolkit, each user automatically benefits from adapted interaction techniques and from an adapted visualization of the scene according to his available devices.

From our point of view, the use of 3DPlasticToolkit has really simplified and accelerated the developments of these applications. Nevertheless, feedbacks from other developers would be needed in order to demonstrate its efficiency. A formal evaluation of 3DPlasticToolkit is discussed in the perspectives of this thesis.

Nevertheless, with the development of these applications some perspectives of work for improving our solution have been identified. The re-usability of the components could be improved. Indeed, even with the similarity between some components used in the furniture planning application described in Section 5.2 and in the co-manipulation one 5.4, none have been used in both demos. A possible amelioration of this aspect could be to decompose application component into multiple ones. An interaction technique, a visual metaphor or a widget will then be a combination of multiple connected application components. However, such an approach would also increase the

number of classes to develop in order to create an interaction technique or a widget and therefore it would complexify developments. With the different demonstrations we also noticed the need to support user adaptation. Indeed, as some interactions can be compatible with the same device setup, the choice is given to the developer that edits the compatibility scores. In the same way, for the different visual effects that we propose, the developer also chooses which one is applied. The only way for the end-user to change the automatically chosen interaction technique or visual effect is to use the meta-user interface. A better solution could be to take into account the user properties and preferences during the automatic adaptation process. This aspect of 3DPlasticToolkit is discussed in Chapter 7.

Chapitre 6

Device Adaptation: the Specific Case of Redistribution

6.1 Introduction

In the previous Chapter we focused on recasting according to the device context. As said in Chapter 3, recasting is one means of adaptation of plasticity. The second means of adaptation is redistribution. A Distributed User Interface (DUI) is a user interface whose components are distributed across different dimensions such as platforms, displays and users [Elmqvist, 2011] [Melchior et al., 2009]. For instance, these components can be widgets, interactors, or content. The redistribution capability of an interactive system refers to its property to change statically or dynamically its components distribution [Calvary et al., 2004a]. It can include migration and replication mechanisms.

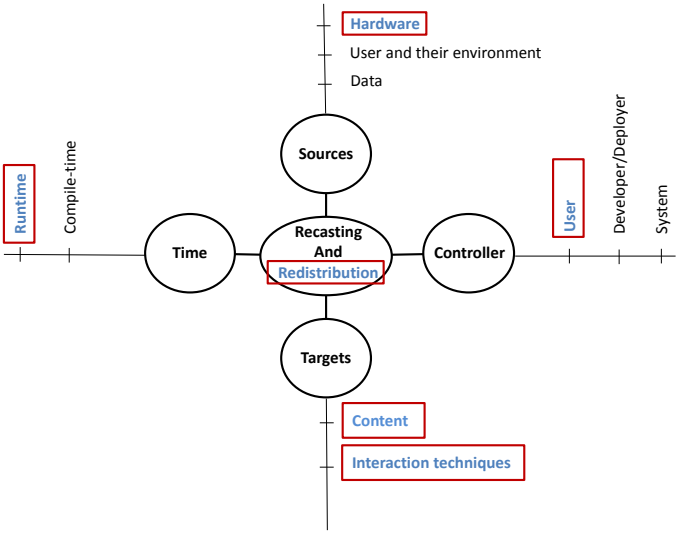


FIGURE 6.1 – Regarding our design space problem of plasticity for MR user interfaces, this chapter is interested in allowing the end user to change dynamically the distribution of a MR user interface.

The need for redistribution can be explained in different ways. Indeed, today, users have access to a wide variety of platforms such as mobile devices, desktop computers and immersive systems. Therefore, users are more frequently confronted with situations where they have to move from one platform to another [Demeure et al., 2008]. They need interaction continuity between these changes. Moreover, combining different platforms can give new interactions prospects to users. These possibilities directly refer to "distributed user interfaces" (DUI) and redistribution.

In this Chapter, as detailed in Figure 6.1, our focus is on allowing the end user to change dynamically the distribution of a MR user interface. Our contribution is D3PART (Dynamic 3D Plastic And Redistribuable Technology), a new model for developers for the design of MR user interfaces that can be dynamically redistributed across different dimensions: platform, user and display. The model is totally implemented into 3DPlasticToolkit. This model has been published in [Lacoche et al., 2016].

This Chapter is structured as follows: first we review the details of the redistribution concept and we present some related work. Second, we give an overview of D3PART. To continue, we describe the D3PART model and its integration into 3DPlasticToolkit. Then, we present how the redistribution process can be controlled with the meta user interface. Next, we present the three examples of redistribution between a tablet and an immersive multi-display system based on our furniture planning application presented before. Last, we give some directions for future work and we conclude about redistribution.

6.2 Definitions and Related Work

A DUI is a user interface which components are distributed across different dimensions [Elmqvist, 2011]. For MR user interfaces we consider three dimensions of distribution from the ones described in [Elmqvist, 2011] and [Melchior et al., 2009]:

- **Display**. The application content is displayed on one or multiple devices. Common examples in 3D for this kind of distribution are multiple display systems.

- **Platforms**. The application runs on a single computing platform or is distributed across multiple ones. These platforms may be heterogeneous (operating system, computing power, plugged devices). For 3D applications, in that category we can talk about cluster approaches that combine connected homogeneous computers to run a VR application with high performances. It can also concern interactive systems where the interactors of a same application are distributed across different platforms.

- **Users**. In that case the application is shared by multiple users. This dimension is directly linked to the two other ones as the different participants can use different displays and platforms. In 3D, this dimension directly refers to Collaborative Virtual Environments (CVE). The field of CVE includes concepts for sharing virtual worlds between different platforms and users.

Redistribution consists in changing the distribution of an interactive system on these different dimensions. According to Demeure et al. [Demeure et al., 2008], redistribution can be system-initiated (the system performs automatically the redistribution), user-initiated (the user initiates and parametrizes the redistribution), or mixed-initiated (the user and the system collaborate to perform the redistribution). According to Calvary et al. [Calvary et al., 2002b], redistribution can be performed on the fly (at runtime) or between sessions and the redistribution granularity may vary on four levels, application, workspace, domain concept and pixel:

- At **application level**, on the platform or user dimension, the application is fully replicated or fully migrated on a distant platform. The application may be adapted to its new context of use, which can include platform capabilities and user preferences. Full replication implies state synchronization to maintain consistency between the different instances of the application. On the contrary, for a full migration, each platform runs its own independent version and no synchronization is performed. For instance, Bandelloni and Paterno [Bandelloni and Paternò, 2004] present a bank 2D application that can fully migrate from a PDA to a PC while keeping the application runtime state during the process as shown in Figure 6.2a. The solution considers the different platforms capabilities in order to adapt the application's appearance and behaviour. In the same way, such example can also be found on the consumer market with the Nintendo® Wii U™[1]. With this console, the migration is not automatically done

[1] http://www.nintendo.com/wiiu/features/

but chosen by the user in the case that he does not or cannot play on his/her television anymore. The game totally migrates on the gamepad and the user can continue his/her game as he/she would have done on his/her television.

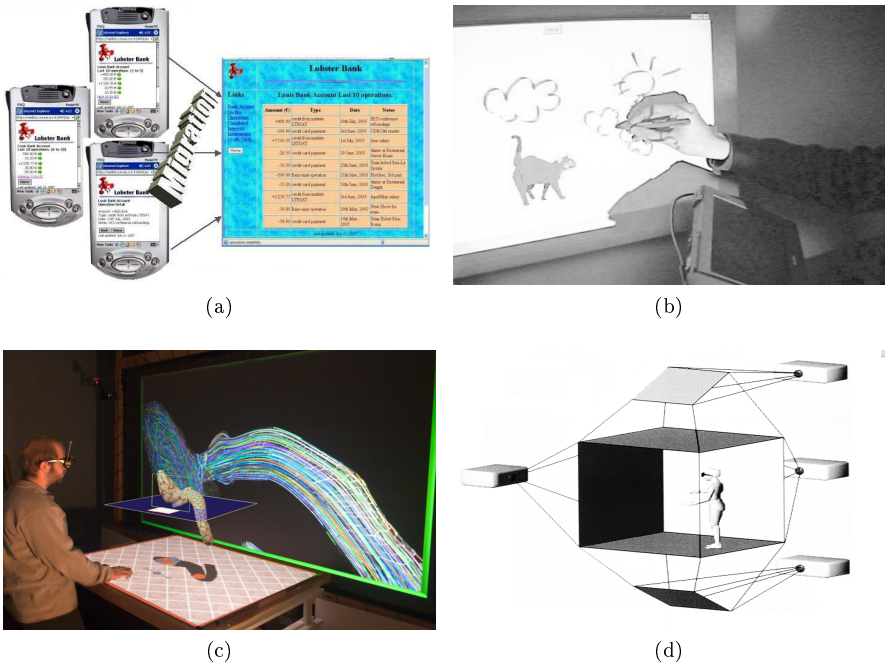
- At **workspace level**, workspaces can be redistributed on the platform, display and user dimensions. A workspace is an interaction space that groups together interactors that support the execution of a set of logically connected tasks. In graphical user interfaces, a workspace can be considered as a window. For instance, the painter metaphor [Rekimoto, 1997], shown in Figure 6.2b, includes two workspaces: the palettes of tools on a mobile device and the drawing area on an electronic white board. In the same way CamNote [Demeure et al., 2005] is a slides viewer that proposes to migrate a remote controller from a PC to a Smartphone. This controller allows the navigation in a presentation displayed by the PC.

- At **domain concept level**, physical interactors can be redistributed on the different dimensions. In 3D, it corresponds to the interaction techniques and widgets. A lot of example of this kind of distribution can be found in the field of MR user interfaces. For instance, BUILD IT [Rauterberg et al., 1998] is a tool dedicated to the design of factories. It is composed of two projective displays. A horizontal one allows the users to have a 2D view of the factory and provides them 2D interaction for object manipulation. A vertical display provides a perspective view of the result. In the same way, for data visualization, Slice WIM [Coffey et al., 2011], shown in Figure 6.2c, combines an interactive multi-touch table and a stereoscopic display in order to provide at the same time an overview of the data as well as a detail view. To continue, in [Medeiros et al., 2013], physical interactors for navigation, pointing and application control are distributed on a tablet in order to interact with content in an immersive system. In all cases the system distribution is hard-coded. It is not performed automatically as it has only been designed to work with these two platforms.

- At **pixel level**, view continuity is ensured across different displays thanks to a distribution on the display and the platform dimensions. In 3D, this kind of redistribution is performed for multiple display systems. In this case, an application can be distributed on a cluster of PCs and rendered on multiple displays with view continuity. A common example of such system in VR is the CAVE system [Cruz-Neira et al., 1992] shown in Figure 6.2d.

In order to handle redistribution on the different dimensions and at the different levels of granularity, solutions designed for 2D user interfaces can be found. For instance, VIGO [Klokmose and Beaudouin-Lafon, 2009] is an architecture that supports ubiquitous instrumental interaction among multiple devices and computers. It proposes an alternative to the MVC design pattern (Model-View-Controller) [Reenskaug, 1979] specifically designed to create distributed interfaces. The 4C reference framework [Demeure et al., 2008], introduced by Demeure et al., is divided in four dimensions: computation, communication, coordination, and configuration that capture the what, when, who, and how aspects of the distribution. It provides a meta-user interface in order to control the redistribution process. To continue, Melchior et al. [Melchior et al., 2009] propose a peer-to-peer architecture for the creation of DUIs. It includes mechanisms for widgets migrations and for the adaptations of the widgets representations and interactions according to the context of use. Moreover, ZOIL [Zöllner et al., 2011] is a software framework for the developement of post-WIMP ("Windows Icons Menus Pointer") distributed user interfaces. It proposes a client server architecture with a transparent persistent mechanism for the synchronization of a multi-user/multi-display/multi-device visual workspace. In the same way, the PolyChrome [Badam and Elmqvist, 2014] is a web application framework that supports the creation of distributed web-based applications for data visualization. The framework handles synchronous and asynchronous collaboration on multiple devices. It handles interaction sharing and synchronization among devices with a secure peer-to-peer (P2P) network. Persistent data and consistency are managed by a dedicated server.

In the field of MR user interfaces, solutions to create DUIs also exist but they mainly focus on specific cases and do not let the end-user change the system distribution at runtime. One specific case handled in 3D and cited before is the case of clusters of computers that manage multiple



(a) (b) (c) (d)

FIGURE 6.2 – Four examples of distribution at different levels.(a) A bank application that can be redistributed at the application level [Bandelloni and Paternò, 2004]. (b) The painter metaphor [Rekimoto, 1997] is an example of distribution at the workspace level. (c) Slice WIM [Coffey et al., 2011] is an example of distribution at the domain concept level. (d) A CAVE [Cruz-Neira et al., 1992] is an example of distribution at the pixel level.

display systems such as CAVEs, Holostages, or Workbenches. In that case the system distribution is performed on the platform and display dimensions. The VR Juggler [Bierbaum et al., 2005] framework and MiddleVR[2], already described in Section 4.2, propose cluster solutions based on client-server architectures. The second specific case handled in 3D is the field of CVE that needs a distribution at the platform and user levels. This implies a state synchronization between the different users' platforms in order to maintain a consistent application. Some architectures for CVE are reported in [Fleury et al., 2010]. They describe three main types of synchronization architecture: client-server, peer-to-peer and hybrid architecture which use both peer-to-peer connections and one or several servers.

In this Chapter, we introduce D3PART a solution that can handle redistribution on the platform, display and user dimensions that considers the 3D specificities. In our case, the redistribution is user-initiated and controlled with an integrated user interface. We focus on redistribution for MR user interfaces at application, workspace, and domain concept levels. Pixel level on clusters of PCs is not covered. Indeed, we consider that handling redistribution at the pixel level with high performances expectations is already a mature field of research while the other levels are less explored in 3D. With the integration of this model in 3DPlasticToolkit, any application developed with our software solution automatically benefits from redistribution capabilities.

6.3 Overview

D3PART (Dynamic 3D Plastic And Redistribuable Technology) is a new model for developers dedicated to the design of MR user interfaces that can be dynamically redistributed across different dimensions: platform, user and display. The model is totally implemented into 3DPlasticToolkit.

As shown in Figure 6.3, D3PART is placed on top of the models presented in Chapter 4 and includes a redistribution process that consists in distributing the high level interaction tasks and

[2]http://www.middlevr.com/middlevr-sdk

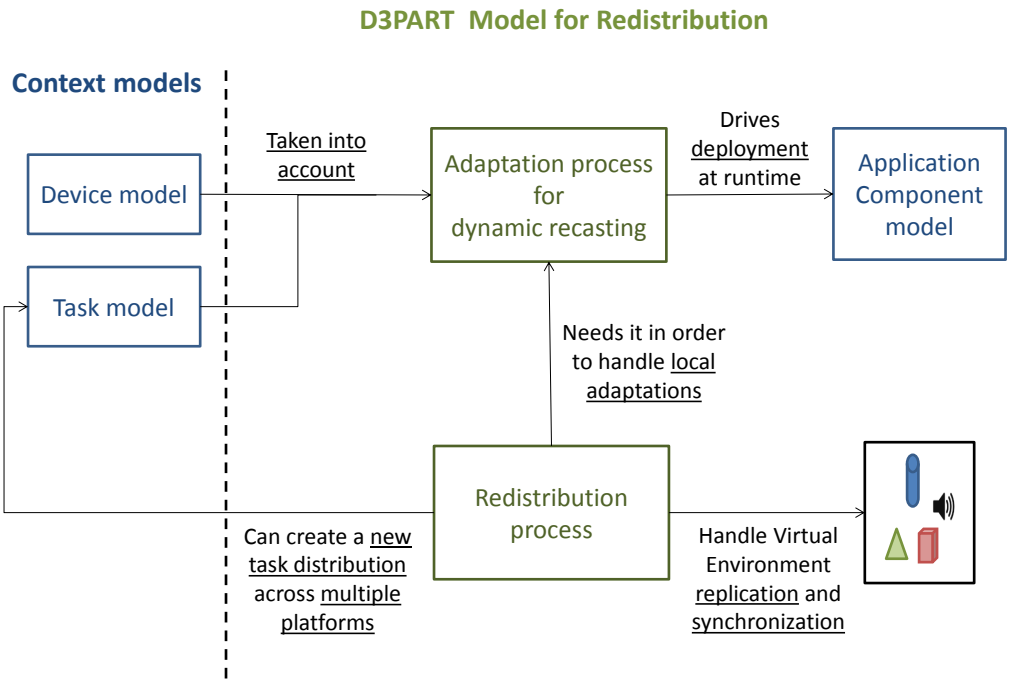


FIGURE 6.3 – The D3PART model consists of a redistribution process combined with the local adaptation mechanisms presented before.

the virtual environment of a 3D application across these different dimensions. D3PART is combined with our previously described adaptation process that ensures that a redistributed application will fit any local context of use. At runtime, we use a client-server architecture to automatically detect new platforms and also to synchronize the different instances of a redistributed application. The redistribution process is user-initiated. Indeed, an integrated user interface is provided to the end user in order to enable him to choose the new distribution of the system.

To illustrate our solution, we present three different scenarios of redistribution between a tablet and an immersive multi-display system for a furniture planning application. This prototype is developed with a toolkit that implements the D3PART model. In these examples, we show how the virtual environment and the interaction tasks can be distributed across the two platforms in order to combine them, to switch seamlessly from one platform to the other one and also to create a collaborative context of use.

6.4 Add Redistribution to 3DPlasticToolkit: D3PART

As explained in Chapter 4, 3DPlasticToolkit handles dynamic recasting according to the available devices and therefore a developer can create an application that can be adapted to the capabilities of a wide variety of platforms. D3PART aims to go further local adaptations and to bring to 3DPlasticToolkit the possibility to change the distribution of any developed application.

D3PART can be integrated in any application implemented with the models presented in Chapter 4. An application is composed of multiple tasks that represent at a high level the application behavior and possibilities. It is also composed of its 3D content (the virtual environment). At this point, thanks to our models the application supports recasting, it can be adapted locally on a wide variety of platforms. In order to add D3PART in such an application we propose a built-in high level task and its corresponding application component. This task and this component allow any developer to add redistribution capability to an application. To do so, the task has just to be defined as needed in the 3DPlasticToolkit configuration file. By defining this task as needed, the application can still support local adaptations (recasting) but it can also now be redistributed on

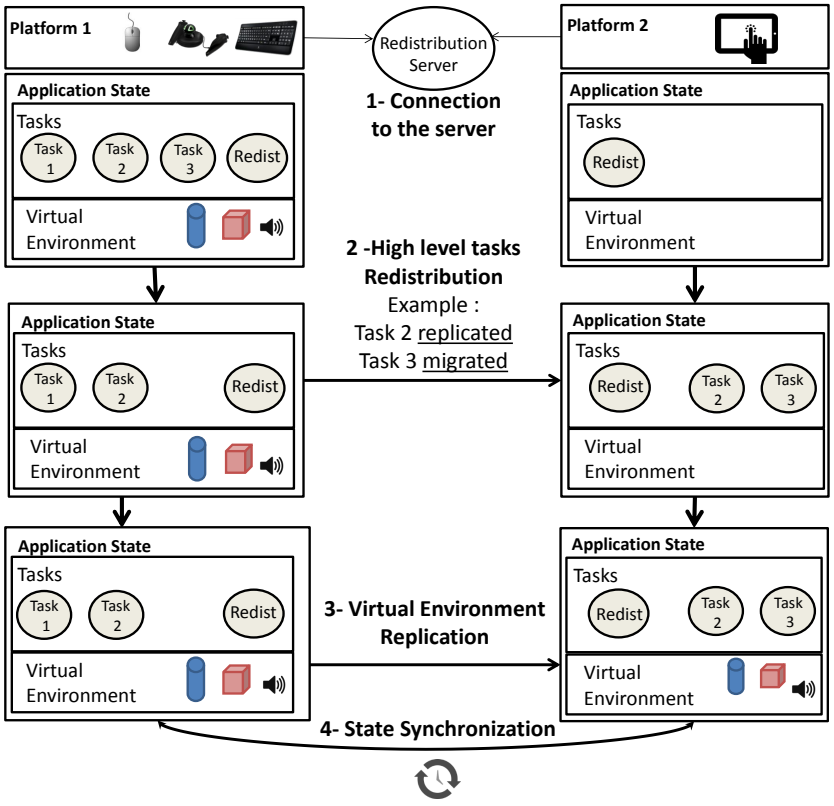


FIGURE 6.4 – With D3PART, the redistribution process is performed in four steps. First, the different platforms connect to the redistribution server. An empty application runs on these distant platforms. It only contains the redistribution task. Then, the user initiates a new distribution of the system with the meta-user interface. Here he chooses to replicate the task 2 and to migrate the task 3 to a second platform. For these two tasks, thanks to the dynamic recasting mechanism handled with our adaptation process, compatible application components are automatically deployed on the second platform that fits its capabilities. The third step consists in replicating the VE from the first platform to the second one. It includes 3D meshes, their materials, and sound assets. We ensure that the state of the application remains consistent during the redistribution. Then in the last step, the redistribution server ensures state synchronization between the two platforms while the applications are running.

multiple platforms, users and displays. The redistribution process has a strong dependency to the adaptation process (recasting) as it ensures usability continuity whatever the new chosen distribution. In the final application, a new distribution of the system will be intended by the end-user. The application component for redistribution is also defined with the extension of the PAC and ARCH models described in Section 4.5. No logical driver is defined as no specific interaction device is needed by this component. The abstraction facet contains the redistribution logic and the rendering presentation facet contains the parts that are dependent on the targeted 3D framework. This component implements a redistribution process that consists in distributing the high level tasks and the virtual environment across the different dimensions: platform, display and user.

Regarding the process, redistribution needs a connection mechanism between the different platforms. This is needed for platforms discovery and state synchronization. To do so, we use a client/-server architecture to which the different platforms can register. For now, this feature is implemented with the network capabilities of the targeted 3D framework. Therefore, it is integrated into the rendering presentation facet. We chose this solution in order to rapidly create prototypes. As proposed in the 4C reference framework [Demeure et al., 2008], this component implements an

```
<TaskConfig>
<NeedTask taskName="SelectionManipulation" taskId="0"/>
<NeedTask taskName="FurnitureControl" taskId="1" topTask="0"/>
<NeedTask taskName="Navigation" taskId="2"/>
<NeedTask taskName="Redistribution" taskId="3">
<ParamTask serverIp="127.0.0.1"/>
</NeedTask>
</TaskConfig>
```

Listing 6.1 – The XML task configuration file of the furniture planning application in the case that redistribution has to be handled. To do so, the redistribution task has to be added in the list of needed tasks. As shown, this task is parametrized with the redistribution server IP.

integrated user interface for platform registration and control redistribution process: the meta-user interface. In our case, the redistribution is performed at runtime and is user-initiated. Indeed, the meta-user interface is proposed to the end-user of the application. The interface can be shown and hidden at runtime with a graphical button or a device button depending on the context of use. The redistribution process is then performed in four different steps as shown in Figure 6.4.

The first step consists in connecting to the redistribution server. The IP address of the server can be given in the meta-user interface or the XML task configuration file as shown in listing 6.1. This step must be performed on the platform where the application is running and on each platform that must be available for redistribution. An empty application developed with 3DPlasticToolkit runs on these distant platforms. The only task defined as "needed" on these distant platforms is the redistribution one. Therefore, the corresponding application component is deployed on these platforms. The goal of this empty application is to be available as a redistribution target. It could receive a part of the main application.

The second step consists in configuring the desired redistribution with the meta-user interface. First, the user chooses the platform on which the application will be redistributed from a list of available ones. In our case, the basis of the redistribution process is made on the platform dimension. However, as each platform may manage another display and may be used by another person, user and display dimensions can also be targeted. Then, the user configures the high level tasks distribution across the two platforms. As shown in Figure 6.5: multiple choices are given to the user in the menu:

- **Full migration**: all tasks migrate. These tasks are now defined as needed on the distant platform and are deleted from the current one. Each platform runs an independent version of the application. It can be performed when the user wants to switch to another platform.

- **Partial migration**: the user chooses which task(s) will migrate to the distant platform. The selected tasks are defined as needed on the distant platform and deleted from the current one. The application is distributed and so shared between the two platforms. It can be performed to combine different platforms.

- **Partial replication**: the user replicates some tasks to the distant platform. The selected tasks are defined as needed on the distant platform and are still defined as needed on the current one. Therefore, he/she will be able to perform these tasks on the two platforms within the same shared application. In the same way, it can be used to combine multiple platforms.

- **Full replication**: all tasks are replicated and can be performed on different platforms in the same shared application. All tasks are now defined on the distant platform and are still defined as needed on the current one. This kind of redistribution can be used to start a collaboration with a user on a different platform.

Dependent tasks must be redistributed together. Therefore, they are grouped into the menu as shown in Figure 6.5. In this figure the furniture control task is dependent on the selection and manipulation task. In the meta-user interface we associate a warning icon to a high level task if it cannot be performed on the distant platform. To do so, we ask the distant platform if an application component can be deployed for each task according to the platform capabilities. The goal of this

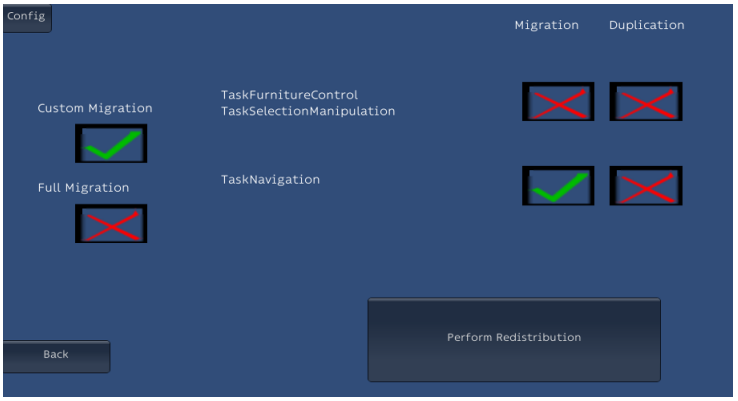


FIGURE 6.5 – The extension of the meta-user interface to control the redistribution process. Here, the user chooses the high level tasks that will be redistributed to the distant platform. In this example based on the furniture planning application, three tasks can be redistributed: navigation, selection/manipulation and furniture control. The last two ones are dependent. The user chooses a partial migration, only the navigation task will be redistributed. The two others tasks remain on his current platform.

feature is to warn the end user that the application can be degraded if this task is redistributed. On the other platform, thanks to the adaptation process included in 3DPlasticToolkit and described in Section 4.7, an application component is automatically associated with each redistributed task. As said, these components are chosen in order to fit the platform capabilities in terms of device availability. With this recasting mechanism, we ensure usability continuity during the redistribution process.

When the redistribution of tasks has been done, the third step consists in fully copying the virtual environment to the distant platform. The goal is to keep the application state during the redistribution to the target platform. This virtual environment includes 3D meshes, their materials, and sound assets. To perform this copy, we consider three solutions.

- Assets are known in the distant platform. Only the names are transmitted.
- Assets are not known but can be downloaded from a distant server. In this case, URLs are provided.
- Assets are not known. For instance in a case of a 3D painting application, the user is editing a new 3D content. Here, assets can be streamed over the network.

For now, our implementation only includes the first one. In order to implement the two others modes, we would exploit some related work on the streaming of virtual worlds through the network. For instance, Alliez et al. [Alliez and Gotsman, 2005] presents a survey of methods for compressing 3D meshes. To continue, Teler [Teler, 2001] presents a client-server architecture for streaming 3D scenes. In the same way, FLoD [Hu et al., 2008] is a framework for peer-to-peer 3D streaming.

The last step consists in synchronizing the different platforms. As for CVEs, a synchronization is performed in order to keep a consistent state between the instances of the same application running on different platforms. In case of full migration, no synchronization is performed because each platform runs an independent version. The synchronization is performed as long as all platforms are connected to the redistribution server. Two kinds of information are synchronized between the different instances of the application. First, the 3D objects transforms are synchronized in order to maintain a consistency between the different 3D worlds. In case of collaboration, to handle concurrency when moving objects, the priority to move an object is given to the first user who grabs it. Then, other users cannot grab and move this object until the first user has released it. Other mechanism could be integrated as well. Second, the events of high level tasks are also synchronized. The events constitute the logical implementation of the application and have to be

synchronously performed on each application instance. To do so, we use an observer design pattern. The redistribution application component observes all task events. When one event is triggered, it is transmitted with its corresponding parameters through the network as text messages in order to be triggered distantly. An example of an event given in the use case application in Section 6.5 is the addition of a 3D object into the scene.

With D3PART the integration of redistribution capacities is totally transparent and automatic for the developer. The developer has just to add the redistribution task as needed in the 3DPlasticToolkit configuration file. As explained, the process consists of distributing the high level tasks and the virtual environment across the different dimensions: platform, display and user. Therefore, the developer has just to focus on the task selection and on the creation of the content of the application. If needed, he can also create new high level tasks and implement new compatible application components with the help of the models described in Chapter 4. D3PART can target the different granularity levels. With the implementation of multiple compatible components for each task and multiple logical drivers, which use different kinds of devices, for each component, the developer ensures that an application will be usable on a wide variety of platforms.

In the next Section, we demonstrate how D3PART and 3DPlasticToolkit have been used in order to add redistribution capabilities to the furniture planning application described in Section 5.2.

6.5 Redistribution for the furniture planning application

In order to illustrate the redistribution possibilities offered by D3PART, we present different use cases that are based on a furniture planning application. This is the same application described in Section 5.2 which consists in laying-out an empty room with furniture. Its goal is to help people to plan the use of particular premises. With the integration of an implementation of D3PART into 3DPlasticToolkit, this application automatically benefits from redistribution capabilities. A video that shows the redistribution capabilities of this application can be found here: `https://www.dropbox.com/s/f39b1756js51jtb/Redistribution.mp4?dl=0`.

As explained previously, at the task level, the application is composed of three tasks. First, a navigation task is needed in order to navigate within the room. Second, we need an application control task (named furniture control) for adding furniture into the room with the help of a menu. Adding an object is defined as an event into the task. Last, we need a selection and manipulation task for moving furniture and for menu selections. These two last tasks are defined as dependent: indeed selection possibilities are needed for interacting with the menu. In the different cases of redistribution that we present we use two platforms. First, we use a mobile device which is an Android tablet. Then, we use an immersive system, the Immersia platform[3] that is a CAVE-like system [Cruz-Neira et al., 1992] with active stereo and with dimensions: 9.6m length × 3.1m height × 3.0m width. Our goal is to demonstrate that our system can handle redistribution between two very different platform: a lightweight mobile platform and an heavy immersive one. For Immersia, MiddleVR is used to handle the different screens and clustering. Even if they are not present in these examples other platforms could also be considered such as HMDs, desktop environments, etc. In the different examples, both systems runs approximately at 25 fps. The difference of frame rates does not impact the synchronization.

As described in section 6.4, the redistribution process starts with the connection of the tablet and of Immersia to the redistribution server. For all the presented cases, the application is first launched on a tablet. For now, the applications only runs locally on the tablet, therefore with the automatic adaptation process integrated in 3DPlasticToolkit, one concrete application component is deployed for each needed task. Each component is chosen in order to fit the platform capabilities. The application components deployed on this tablet are the same as the ones described in Section 5.2 for the mobile platform. First, for the furniture control task, a menu is instantiated with the list of furniture that can be added. According to its implementation the menu can be hidden if needed. For the manipulation task, a 2D cursor interaction technique is deployed on the tablet. The chosen Logical Driver exploits the multi-touch capabilities of the tablet. With this technique the user can translate the objects onto the floor with one finger and rotate them around the up

[3] http://www.irisa.fr/immersia/technical-description/

(a) (b)

FIGURE 6.6 – The redistributed World-In-Miniature: an example of redistribution that demonstrates how D3PART can be used for platform combination.

axis with two fingers. For the navigation task, a pan and zoom navigation technique is deployed. Here, this component places the camera in order to have a plan view of the scene on the tablet as shown in Figure 6.6a. With the multi-touch capabilities, the user can translate the point of view and can zoom within the scene while keeping the plan view of the room.

6.5.1 Redistribution for platform switching

Today, users are more frequently confronted with situations where they have to move from one platform to another [Demeure et al., 2008]. This is one scenario possible for our furniture planning application. This example demonstrates how the redistribution capabilities of our solution can ensure usability continuity during changes of hardware environment. In this scenario the redistribution is performed on the platform and display dimensions and at the application level.

First, the user is interacting on the tablet at his desk. With this tablet he can also work while mobile. All the tasks are available as corresponding application components are deployed as explained in the previous Section. However, the tablet only offers a 2D plan view of the result and the user would like to have a 3D view at scale one in order to better perceive the volumes. To do so, an immersive system is available: the Immersia platform. The meta-user interface allows the user to perform a full migration of his/her application to this platform. The application totally migrates to Immersia, all tasks and all contents, nothing remains on the tablet. The user can now be immersed at scale one and continue to fine-tune the layout of the room. Usability continuity is ensured thanks to the adaptation process included in 3DPlasticToolkit. Indeed, the application is adapted to the target platform as application components are chosen for each task according to the new platform capabilities. In that case, a 3D ray-based manipulation technique is deployed. The associated logical driver sets the position and the rotation of the ray with the tracked flystick and it uses its buttons for object selections and to change the ray length. For the navigation task, a walking navigation metaphor is deployed. The associated logical driver uses the tracked head position and the flystick joystick in order to move the point of view. For the furniture control task, a 3D movable menu is deployed. The 3D ray is used to select the menu items, to move it and also to hide and show it.

When the user has finished his work he may want to continue his work while mobile. Therefore, the meta-user interface is also available in Immersia and so the inverse process is also possible to migrate back to the tablet. For example, he would go showing the result to a colleague who could use another kind of VR setup such as an HMD.

6.5.2 Redistribution for platforms combination

In this Section, our example demonstrates how redistribution can be used in order to combine different platforms. In that case redistribution is performed on the display and platform dimensions

and at the domain concept level. Our example is based on the World-In-Miniature technique [Stoakley et al., 1995b] that provides the user with a handheld model of the virtual environment at a smaller scale. It can be used for manipulating virtual objects or for navigation. This miniature representation is directly rendered in the virtual world. Here, we propose to deploy this technique onto a tablet in order to control the furniture planning application in Immersia. The user will be able to interact with the tablet while being immersed at scale one in Immersia. This use case can be useful for novice users who are not confident with 3D interactions and may prefer more common multi-touch interactions. Indeed, the user can interact with the usual and easy-to-use multi-touch capacities of the tablet while being immersed at the same time in the 3D world with Immersia.

To do that, the user chooses a partial migration to Immersia. Only the navigation task migrates to the distant platform. Other tasks remain on the tablet. This choice is made with the meta-user-interface as shown in Figure 6.5. As described in Section 6.5.1, an interaction technique based on a walking metaphor controlled with head tracking and a joystick is deployed in Immersia for this navigation task. It places the point of view inside the room in order to immerse the user in it. At this time the application is distributed on two platforms and displays. First, as shown in Figure 6.6a, a redistributed World-In-Miniature is on the tablet. The virtual world is displayed at a lower scale with a plan view. Moreover, as said in Section 6.5.1, a 2D menu for furniture control and a multi-touch interaction for selection and manipulation are deployed. Second, as shown in Figure 6.6b, at the same time the user is immersed at scale one into the room in Immersia and can navigate in it. Our transparent synchronization mechanism ensures the consistency between the two parts of the application. Indeed, the synchronization of the 6 DoF transforms of the objects between the two platforms ensures consistency when the user moves an object on the tablet. As well, the command for adding an object into the room is also synchronized. Therefore when an object is added with the 2D menu on the tablet, the same object is also added in Immersia.

6.5.3 Redistribution for collaboration

In this example, we demonstrate that our redistribution process can be used in order to create a Collaborative Virtual Environment (CVE). Here, redistribution is performed on the user, platform and display dimensions and at the application level. Indeed, the replication capabilities included in our solution let any user to start at any time a collaboration with another person using a different platform. With this feature any application developed with 3DPlasticToolkit, including the furniture planning one, automatically benefits from collaboration capacities.

In this scenario, the first user has performed a first configuration of the empty room with his/her tablet and now wants to share his/her room configuration and wants to complete it with another user. Therefore, he performs a full replication from the tablet to the second user platform: Immersia. All tasks are replicated, navigation, selection and manipulation, and furniture control. Therefore, the two users have now the same interaction capabilities. The plasticity property handled by the system ensures usability continuity between the two platforms, the interaction capabilities remain the same. Indeed, the application components deployed for these different tasks are chosen according to each platform capabilities with the adaptation process included in 3DPlasticToolkit. They are the same than for the two scenarios described in the two previous sections. In this case, the collaboration is asymmetric as the two people are using different platforms and different interaction techniques as shown in Figure 6.7. A collaboration with two similar systems could also be performed. Here, the collaboration is co-located, both users are situated in the same place and can directly communicate about the result. However, the collaboration could also be distant. Indeed, our architectures makes possible to have distant connections to the redistribution server. With the virtual environment replication and the synchronization performed by the redistribution process, a high consistency between the two instances of the application is ensured. Both users are interacting in the same shared virtual environment. In order to provide awareness about the activity of the distant user, for now only the view frustum of each user is represented in the virtual environment. Future work could include different awareness mechanisms, for instance trying to make the distant user perceive his/her current context of use.

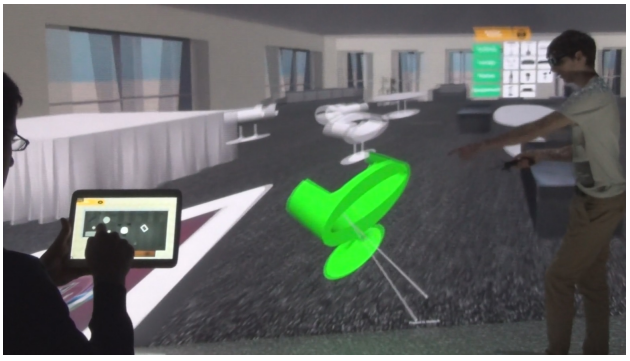


FIGURE 6.7 – An example of redistribution on the user dimension. For the furniture planning application, a full replication is performed from the tablet to Immersia The two users are now collaborating in the same shared virtual environment. Both users have the same interaction capabilities.

6.6 Perspectives

This work can be extended following three main directions.

First we want to consider **level of details** during the virtual environment replication. Indeed, as each platform may not have all the same computation capabilities, the rendering of the assets could be handled differently. In some cases it would be necessary to consider adaptive assets. For instance, a very complex 3D model cannot be rendered in the same way on a PC with multiple GPUs or on a mobile device. To solve this issue, we plan first to integrate information about the computing power of the platforms into the device model. Second, we plan to give the possibility to parametrize the choice of an asset according to the computation capabilities of the target platform. During the virtual environment replication process, only the assets that correspond to the distant platform computation capabilities would be transmitted.

Second, we want to explore the **automation of the redistribution process**. Our goal is to obtain also system-initiated redistribution or mixed-initiated redistribution. Indeed, for now the process is only user-initiated with the help of the meta-user interface. For instance, this kind of approach could consist in finding the right platform or the right user for each task according to the platforms capabilities and the user preferences. Our scoring system could be used to do so, for example by applying no longer locally, but instead by applying it on all available platforms. Indeed, for each task we could compare the scores obtained by the applications components on each available platforms and instantiate the best ones. Another possibility to trigger automatically the redistribution process could be to analyze the user behaviour. For example, we could detect when the user enters in Immersia with his tablet or when he drops it before coming in.

Third, for now the network synchronization between the different instances of the application is dependent on the 3D framework used and implemented in the rendering presentation facet of the redistribution application component. However, as future work, **the synchronization mechanisms could become independent of the 3D framework** and implemented in the abstraction facet of the redistribution application component. Our current implementation does not show apparent latency but being independent from the 3D framework would let us optimize the network load.

Last, we plan to **evaluate** the system in order to assess its interest, its usability and its acceptability for end users.

6.7 Conclusion

D3PART (Dynamic 3D Plastic And Redistribuable Technology) is a new model to handle redistribution for MR user interfaces which is integrated into 3DPlasticToolkit in order to fully cover our requirement **R3**. With D3PART, redistribution can be performed on the display, platform and user dimensions and can target three levels of granularity: application, workspace, and domain

concept levels. Our approach is based on a client-server architecture. Redistribution can be performed at runtime by the user with an integrated user interface: the meta-user interface. Dynamic recasting handled by 3DPlasticToolkit, with the included adaptation process, ensures usability continuity whatever the new distribution chosen. The distributed application will fit each target platform properties. Thanks to this approach, any application developed with the 3DPlasticToolkit automatically benefits from redistribution capabilities.

To illustrate these possibilities, we have presented three examples of redistribution on different dimensions and at different levels for a furniture planning application. These examples show how redistribution can be used to switch from a mobile platform to an immersive one, to combine these two platforms, and finally to create a collaborative context of use between them.

Chapitre 7

User Adaptation

7.1 Introduction

The second adaptation source that we take into account with our plasticity models is the user and his/her environment. For now, the choices of interaction techniques and of the content presentation are only impacted by the available devices and by the developer's decisions. Therefore, it could result to an interface that does not really match the user's needs and preferences. Indeed, the developer's choice will not always be the user's preference. As illustrated in Figure 7.1, the focus of this Chapter targets the adaptation to users. We have demonstrated in Section 3.3.2 that personalization consists in adapting an application to the user's properties, preferences and information about his/her environment. The two adaptation targets of our plasticity space problem, content visualization and interaction techniques, can be impacted by these different properties.

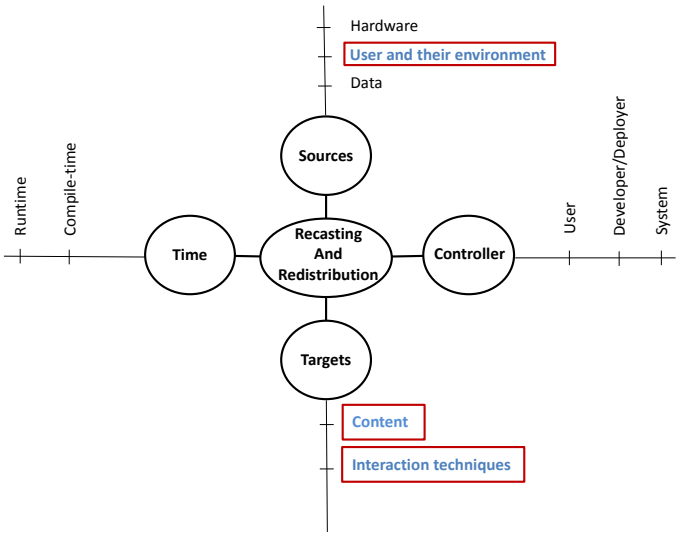


FIGURE 7.1 – Regarding our design space problem of plasticity for MR user interfaces, this chapter is focused on adapting the interaction techniques and the content visualization of an application according to the user properties.

For instance, in the experiment described in Section 5.3, we have shown that users have a global preference for some effects that deal with frame cancellation. Nevertheless, the decision is not unanimous and therefore it could be interesting to apply a different visual effect for each particular user. In the same way in the co-manipulation application described 5.4, no formal experiment has been performed but multiple interaction techniques have been tested during the development process especially for the user inside the manipulated object (the *Ant*). The user's feedbacks were

not always in the same direction. Therefore, it also demonstrates that adapting the interaction techniques to each particular user can be a good way to improve the user experience. In this Chapter, we also propose a preliminary experiment that compares the user preferences for selection techniques. One goal of this experiment is to identify the needs that our toolkit has to satisfy in order to consider user adaptation.

In order to cover this adaptation source in our model, three main issues have to be considered. First, we must create a user model that can include the different kinds of properties defined in Section 3.3.2: the profile, the preferences, the environment properties, the monitoring information. To do so, we also added a user model in 3DPlasticToolkit that is complementary to the device model detailed in Section 4. Second, we must also be able to update this user model at runtime. Two possibilities can be proposed, either the user can update himself/herself his/her properties, this is the explicit approach described in [dos Santos and Osorio, 2004], in 3DPlasticToolkit this is performed with the meta-user interface, or the system updates itself the model, which is the implicit approach detailed in [dos Santos and Osorio, 2004]. For now, the implicit approach is not fully implemented in 3DPlasticToolkit but our model covers the possibility to create a monitoring component that can track the user's behavior. Third and last, we must be able to take into account this user model in order to modify the application. Therefore, we have also implemented three new scoring modules that exploit our user model. They are named "UserPreferences", "UserAndDeveloper" and "MachineLearning". These modules update the application components compatibility scores and therefore provide the user with the MR user interface that matches his/her preferences. Two of these modules only use the preferences scores in the user model. However, with our preliminary experiment we have also investigated the use of machine learning in order to detect automatically the user preferences for interaction techniques according to the other kinds of properties of the user model. These results are integrated in the other scoring module.

This Chapter is structured as follows. First, in Section 7.2 we present different related work of software solutions that deal with user adaptation. In Section 7.3, we present a preliminary user study that we have performed in order to identify the needs that our toolkit has to satisfy to consider user adaptation. This preliminary study focuses on interaction techniques selection. The results of this preliminary study have been used in order to explore machine learning for the detection of user preferences. Then, in Section 7.4, we present our integration of automatic user adaptation in 3DPlasticToolkit. We present our user model, how it can be updated by the user and by the system, and how it is taken into account by the adaptation process with two different scoring modules. In this Section, we also present some examples of how user adaptations have performed in the demonstrators presented in Chapter 5. To finish, in Section 7.5 we conclude and give some perspectives of work.

7.2 Related Work

In the field of MR user interfaces, most solutions that handle user adaptation only target one of the possible adaptation targets: content visualization or interaction techniques.

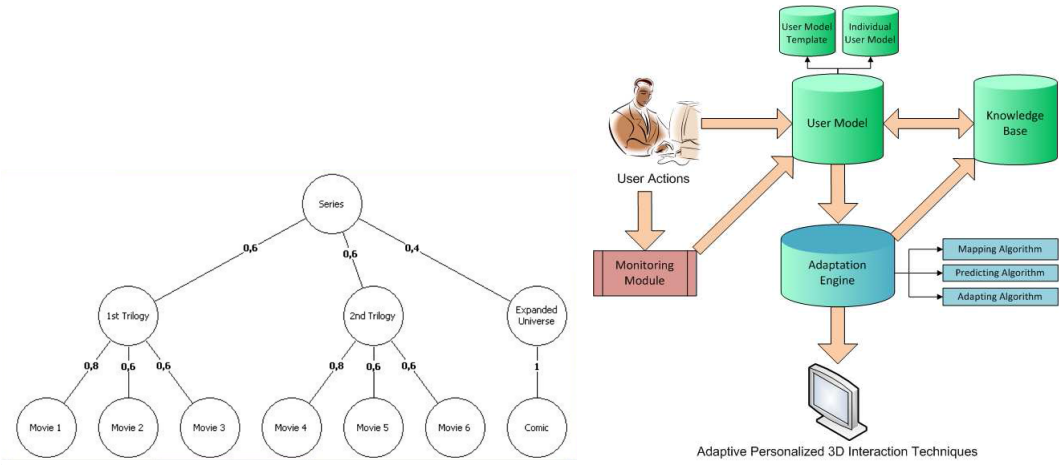
Content visualization adaptation

In this category, multiple solutions can be found. For example, Bonis et al. [Bonis et al., 2009] propose a software solution for the creation of virtual museums with personalized content including a user model based on a semantic graph. In this graph, nodes stand for objects and concepts, edges represent the relations between them, and the levels represent the degree of generalization. The user model is an instance of this graph as shown in Figure 7.2a, with all nodes being given a numerical value (positive or negative) that represents the degree of interest that the user is assumed to have for the object associated to the node. This user model is updated by monitoring the user's behaviour. For example, it observes the time spent by the user observing an object. The user model is then used in order to choose dynamically the museum content by focusing on the objects that get the better scores. To continue, in the field of personalized e-commerce 3D applications, Chittaro et al. [Chittaro et al., 2002] propose AWE3D, an architecture for the creation of adaptive 3D websites. This solution stores the user model on the server side. This model can include information about the user profile and his/her preferences. A rule-based system updates this model by using monitoring

information from VRML sensors. Another rule-based system adapts the application according to the user model. A similar approach is also proposed in AMACONT [Dachselt et al., 2006]. In [Chittaro et al., 2002], different examples are given on how to change products visibility and attractiveness according to the user model. For the same domain, Dos Santos et al. [dos Santos and Osorio, 2004] propose AdapTIVE (Adaptive Three-dimensional Intelligent and Virtual Environment) an architecture for the creation of personalized VEs. The user model contains information about the users's interests, preferences and behaviors. It can be edited explicitly by the user or implicitly by the system with a monitoring. Then, a generator of environment creates a VE that matches the user model.

Interaction techniques adaptation

Other solutions focus on adapting the interaction targets. Nevertheless, these kinds of solutions are less common. Octavia et al. [Octavia et al., 2009] propose a conceptual framework for the creation of 3D user interfaces with interaction techniques adapted to the user preferences. An overview of this framework is given in Figure 7.2b. First, this framework is composed of a user model that includes the user's interaction patterns, interests and characteristics. They define three levels of information included in this model. The general user model includes basic information used for adaptation to all users. The group user model provides specialized information to be applied to a group of users. For instance, group user models can be used in order to detect the user preferences according to his/her profile. The individual user model gives detailed information about one particular user. Second, the framework is composed of an adaptation engine that modifies the interaction techniques according to the user model. Interaction techniques can be replaced, disabled or enhanced with other modalities. They have investigated the creation of these different levels for a selection task



(a) (b)

FIGURE 7.2 – Two software solutions for user adaptation (a) In [Bonis et al., 2009] a user model is represented by a semantic graph with scores of interest (b) The conceptual framework for adaptation and personalization in VEs proposed by Octavia et al. [Octavia et al., 2009].

in a VE. Two selection techniques were compared: a ray-based interaction technique and a bubble cursor. Physiological data was measured in order to evaluate user frustration, user experience and mental workload. They did not find any trend for the creation of group user models. Nevertheless, they have shown in [Octavia et al., 2011] that employing general and individual user models can increase users' performances and can decrease users' frustrations. The general user model was created according to the scene parameters and an individual user model was created for each user after a first experiment with the two interaction techniques. Such approaches do not really give the possibility to adapt an application to the preferences of a new user. That is why more work should be done to create a solution that can detect the preferences of a new user according to the collected information about him/her. An interesting perspective of work have been proposed by Wingrave et al. [Wingrave et al., 2002]. They address interaction techniques exchangeability and

intrinsinc parameters tuning from users with two different selection techniques: Ray-casting and Occlusion selection. During the experiment, they did not find any trend, each user had his own preferences. That is why they propose a perspective of work that aims at correlating the user's profile and behaviour with his/her preferences. This approach consists in using "nuances" in an inverse reinforcement learning process to predict user behaviors. A nuance is defined by Wingrave et al. [Wingrave et al., 2001] as *"A repeatable action the user makes in their interactions in an environment, intentional or not, that are highly correlated with an intended action but not implicit in the interaction metaphor"*.

Analysis

To conclude, none of existing solutions considers personalization of both the content visualization and the interaction techniques. In the first category, solutions are mainly based on Web3D applications and on content selection. In the second category, no solution has succeeded to automatically detect the preferences for interaction techniques of a new user according to his/her properties. Our goal is also to find a way to detect the preferences of a user that has never used the application by only using his/her profile or by monitoring him/her. In this Chapter, we propose an integration of user adaptation in 3DPlasticToolkit that aims to target both categories. This integration focuses more on the choices of content visualization metaphors, visual effects and interaction techniques than on content selection. With a preliminary study we try to identify the needs that our toolkit has to satisfy to consider interaction techniques selection. The results of this study has also been exploited in order to explore the use of machine learning for detecting the preferences of a new detected user.

7.3 User Preferences for Mixed Reality User Interfaces: A Preliminary Study

As explained in the previous Section Octavia et al. [Octavia et al., 2011] explored the creation of user models with an experiment that compared two selection techniques on the same device setup. They have shown that choosing the adapted interaction technique according to a general model or an individual one can really be appreciated by the end-user. Nevertheless they did not succeed in building group user model for detecting the preferences of a new user.

Therefore, in this Section we propose a complementary experiment of this related work that aims to fulfill its lacks. Now we are going to describe our objectives.

1. **Verify consistency of performances and preferences on two device setups.** In their experiment they only analyzed the preferences on the same device setup. We wonder if the preferred interaction technique of a user on a given hardware configuration will be the same with a different setup. Therefore, we have compared the users preferences and performances for three 3D selection techniques on two device setups that mainly differ by their display types (stereoscopic screen and HMD).

2. **Verify consistency between performances and preferences.** Octavia et al. built the automatic choice algorithm by taking into account at the same time the preferences and the performances. Nevertheless we can wonder if the preferred interaction technique of a user always corresponds with the one with which he/she performed the best. Therefore, in our experiment we propose to collect and compare the qualitative preferences and the quantitative performances of the participants for three 3D selection techniques.

3. **Predict the preferences of a new detected user with machine learning**. In their work they did not succeed in creating a solution that can find the preferred interaction techniques of a new user according to his/her properties. Therefore, we propose to explore two ways based on machine learning to create such a solution:

   (a) **Predict the preferences according to a 2D preliminary selection task.** We propose a preliminary test based on a 2D selection task. The results of this test can be

used in two different ways. First, we want to know if there is a correlation between this test and the users performances during the 3D selection tasks. It could then be used as a preliminary test for any application in order to detect if the user would need any kind of interaction assistance at runtime. Second, we also want to use the results of this test in order to try to predict the user preferences by training a machine learning algorithm.

(b) **Predict the preferences according to the user profile.** We propose to ask more information about the user in order to build a more complete profile. Then, from these different collected profiles, we try to train a machine learning algorithm in order to predict the preferred technique of a new user.

In Section 7.3.1 we present our experiment, we present the results obtained by the different interaction techniques on our device setups and we try to reach our first two objectives. Then in Section 7.3.2, we try to reach the third objective by exploring the use of machine learning to predict user preferences.

7.3.1 Comparing three selection techniques on two device setups

Hardware

As said previously, one goal of this experiment was to compare the performances and preferences of users for the same application run on two different hardware configurations. The main difference between the two platforms was the display used as the control of the interaction techniques was approximately the same on the two platform with a 6-DOF controller. The two setups have already been presented in Chapter 5. First, the zSpace shown in Figure 7.4a is composed of a 6-DOF tracker with buttons (the stylus) and a 24-inches co-located stereoscopic display. Second, we used a setup based on an Oculus Rift DK2 HMD and a Razer Hydra as shown in Figure 7.4b. The Razer Hydra is composed of two 6-DOF controllers with buttons. Only one of them was used.

Participants

As one goal of the experiment was to use the results in order to train a machine learning algorithm, it was important to have a great number of participants. Therefore, our experimental group consisted of 51 subjects aged from 23 to 57 (age: M=34.8, SD=10.24). There were 38 males and 13 females. The subjects had various backgrounds, some of them even were not working in the technology industry. Most of them had a few knowledge about VR. In order to build the user profile of each participant that is integrated in the user model, each of them was asked to fulfill a preliminary questionnaire. The user profile consists in a set of parameters directly given by the user in order to define him/her as a person. The following informations were requested:

- gender,
- handedness,
- age,
- master Eye,
- visual problems (Yes/No),
- experience with virtual reality (Yes/No) (Hours per Week),
- experience with video games (Yes/No) (Hours per Week),
- experience with computers/tablets (Hours per week).

2D selection task

The first step consists in a 2D selection task based on Fitts's law. A view of a participant performing this test is given on the right of Figure 7.3. The Fitts law describes the relation between the target

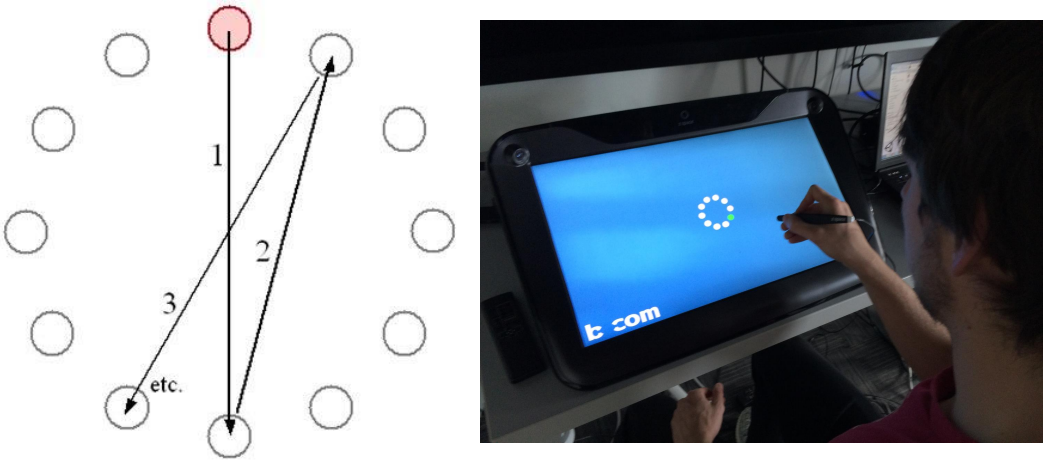


FIGURE 7.3 – ISO 9241-9 multi-directional pointing task. The user first selects the highlighted circle, then the next circles to select follow the pattern drawn with the arrows. On the image on the right, a participant is performing this selection test on the zSpace. As shown, the green disk is the one to select.

distance, width and time needed for a selection task. Fitts's law can be used in order to compare the performances of pointing devices by analizing the throughput. Our Fitts's law based tests follow the ISO9241-9 standard and the seven recommandations given by Soukoreff and Mackenzie [Soukoreff and MacKenzie, 2004] that are:

1. Use the Shannon formulation of the index of difficulty (ID).

$$ID = log2(\frac{A}{W} + 1) \tag{7.1}$$

   where A is distance between targets and W the targets width.

2. Use a wide and representative range of ID (between 2 and 8 bits). Each ID configuration must be presented to each subject many times (15-25) times. Movement time data should also be collected.

3. Subject's movement end-points must be collected. Obvious outliers such as double clicks may be removed from the data.

4. The end-points should be used to perform the adjustment for accuracy. Having these data you can define the effective target width We and the effective index of difficulty IDe. A large discrepancy between ID and IDe should be investigated.

5. least-squares linear regression is used to find the intercept and the slope of the Fitts' law equation. This test is needed to have a measure of the goodness of fit.

6. You can calculate the movement time prediction using the regression model and the ID value.

7. If devices or experiment conditions are to be compared, calculate the dependent measure throughput via the mean of means.

The standard describes the *Multi-directional pointing task* that can be used to evaluate pointing movements in many different directions. This is the pointing task that we have used in the first part of our experiment has shown in Figure 7.3. This first part of the experiment was performed by each user on the zSpace. No stereoscopy or head tracking were used. The targets were simply displayed in 2D on the screen of the device. The 6-DOF stylus and one of its button were used to control a 2D cursor to perform selections. The cursor position corresponded to the intersection

between the stylus and the screen. Therefore, to move it the user had to target the screen as shown in Figure 7.3. Two color feedbacks were provided. The disk to select was colored in green and when selected (clicked) the disk was colored in red. During this part of the experiment this multi-directional pointing task was performed three times by each participant with three different IDs (difficulty increases over time):

- disk diameter = 40 pixels ; distance = 200 pixels ; $ID = 2.58$
- disk diameter = 20 pixels ; distance = 200 pixels ; $ID = 3.46$
- disk diameter = 20 pixels ; distance = 500 pixels ; $ID = 4.7$

The goal of this first test is to get a first result on the interaction skill of each user for a basic task. The results can then be exploited in the user model as explained in Section 7.3.2

All the actions of each user were collected. Indeed, for instance, we collected the selection times, the cursor movements relative to the straight path between two targets, the number of clicks outside a disk or on a wrong one, the distance from a click to the center of click, etc.

(a) (b)

FIGURE 7.4 – The two device setups used for our user preferences study (a) The first setup is zSpace composed of a stereoscopic screen with head tracking and of a 6-DOF tracked stylus. As shown the biggest button of the stylus is used for object selection. (b) The second setup is composed of an Oculus Rift DK2 and a Razer Hydra. Only one 6-DOF controller of the Razer Hydra was used. As shown one of its triggers is used for object selection.

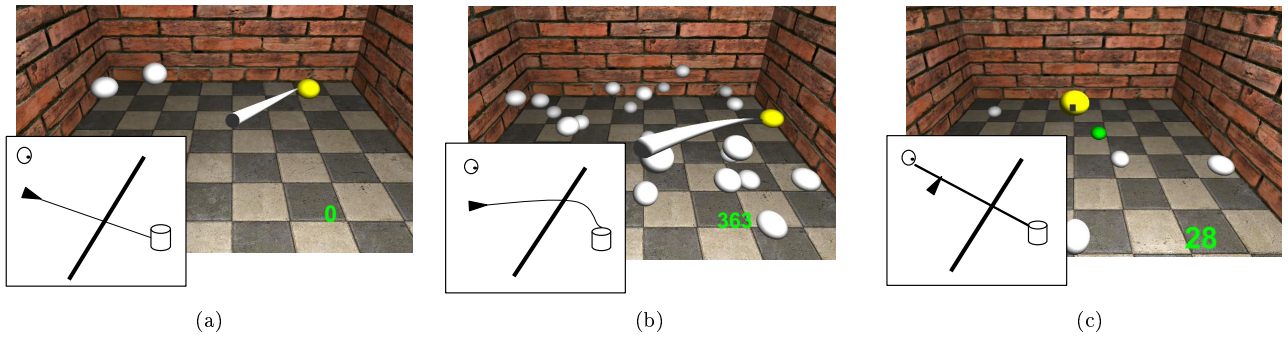
3D selection task

The second part of the experiment proposed a series of 3D selections to perform with the two interaction setups and with three different interaction techniques. In this part of the experiment participants were placed in a VE at scale one just in front of a box containing different spheres to select as shown in Figure 7.5. These spheres were all the same fixed size: 3cm of diameter. The three interaction techniques that we compared were:

- **3D Ray-Casting**: this first interaction technique is based on the ray-casting metaphor as described by Bowman et al. [Bowman et al., 2004]. A straight ray that comes from the user hand and intersects a scene object for selection as described in Figure 7.4a. The ray base is controlled in position and rotation with the values given by the stylus tracker or by one Razer Hydra controller. A button was used for selections.
- **3D Bendcast** this technique is very similar to the first one. It is also controlled with the stylus or by one Razer Hydra controller. The technique is described in Figure 7.5b. Our

implementation is similar to the one described by Cashion et al. [Cashion et al., 2013], the bendcast technique automatically bends to the closest target in order to help the user during the selection task. One difference in our implementation is that the ray bends only if the distance between the ray and the object is under a threshold. This threshold is chosen according to the targets sizes.

- **Occlusion**: as described by Pierce et al. [Pierce et al., 1997], the occlusion selection technique consists in placing the tip of your finger between your eye and the object you want to select. The occluded object is the one selected. In our implementation the stylus extremity or one Razer Hydra controller is used to drive the occluder and a button is also used to confirmed selection. The eye used by the technique corresponds to the user's master eye. The technique is described in figure 7.5c.



(a) (b) (c)

FIGURE 7.5 – The three interaction techniques compared in our user preferences study. (a) A 3D Ray-Casting interaction technique. The position and the orientation of the ray directly follow the 6-DOF controller. (b) A 3D Bendcast interaction technique. This technique is similar to the first one. Here, the ray bends in order to automatically sticks to a target if it is close enough (under a threshold). (c) The Occlusion selection technique. Here, the user has to hide an object with his controller extremity in order to point it. As shown, a small black cube is displayed in the VE at this extremity.

For each technique, two color feedbacks were provided to the user. A pointed sphere was colored in yellow, and a selected one in red. For this part of the experiment, some users started with the Oculus Rift while other users started with the zSpace. In order to pass to the next setup, the user had to perform all selections in four scene configurations and with the three interaction techniques. All the users did not try the techniques in the same order. Nevertheless the same order was applied on both setups for one particular participant. We applied a counterbalanced design: 2 *device setups* $*$ 6 *technique orders*. For each technique the process was the same. The users had to select the spheres in four scene configurations. For each configuration the user had to perform 10 selections. The sphere to display was colored in green as illustrated in Figure 7.5c. These scene configurations were depending on two parameters: two conditions of density (low and high) and two conditions of movement (mobile, stationary). For each interaction technique the scene configuration order was always the same: low density/stationary spheres, high density/stationary spheres, low density/mobile spheres and high density/mobile spheres. The goal was to increase the difficulty progressively. The instruction given to each user was to select the spheres as fast as possible and to avoid mistakes. In order to challenge them we displayed a score about their performances. This score took into account the selection times and was decreased when a wrong selection was made.

In order to get the preferences of all users, they had to fulfill different subjective questionnaires at different steps of the experiment. In these subjective questionnaires, participants had to grade the three techniques using a Likert-scale, from 1 (very low appreciation) to 7 (very high appreciation) according to one criteria: global appreciation. The goal was just to record which technique they were preferring. A first grade had to be given to an interaction technique after having performed all

selections for one scene configuration. Then, after having finished all selections on the first device setup, users were asked to give new grades to each technique, a global one and one for every scene configuration. The same process was performed on the second device setup, different grades for each technique were asked for this second setup. Last, at the end of the experiment, different grades for each setup and for each technique were requested. In this last step, the user had to give again different global grades for each technique on each device, and also to each technique for each scene configuration for each device setup. During this step, participants did not have access to the grades they gave earlier in the experiment. Each time, the given grades are final and cannot be modified later in the experiment. The goal of this last step was to allow the user to make comparison between the two hardware configurations. The user's actions and performances were also collected during this part of the experiment. One of our goals was to compare the preferences with the performances in order to determine if they are always consistent.

Results

For now, our analysis of the results does not separate the performances and the preferences according to the different scene configurations. Indeed, we have added the preferences scores and the performances scores of each technique over the selections performed for the four scene configurations proposed.

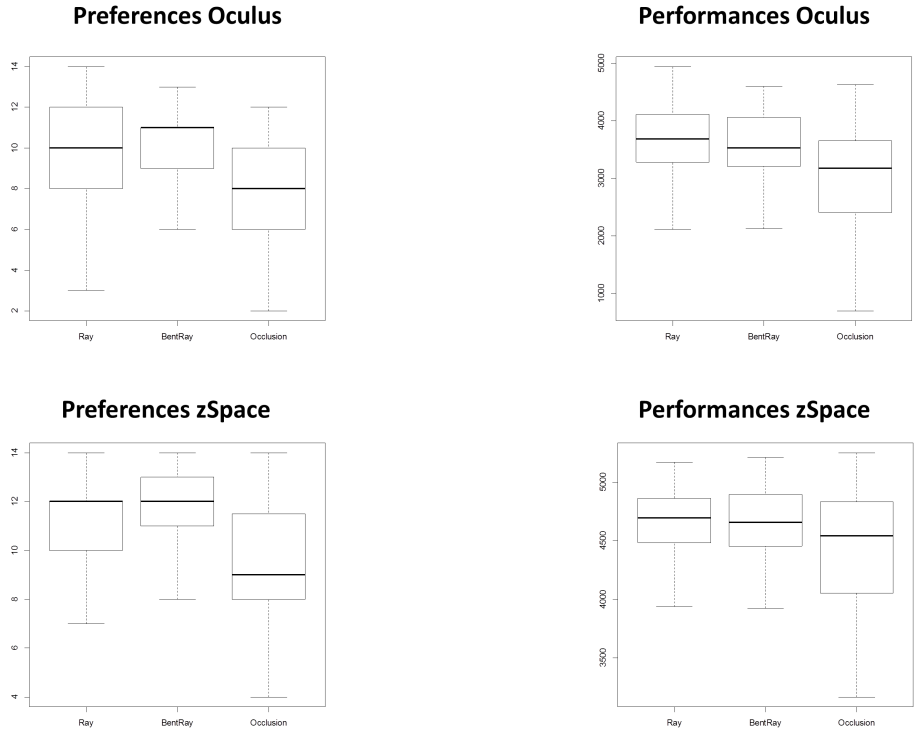


FIGURE 7.6 – Comparison of the preferences and performances results for the three techniques on the two device setups. The results for the 3D Ray-based interaction technique and the 3D Bentray are similar. However, the 3D Bentray seems to be preferred on the zSpace. We can also see that participants less preferred and less performed with the Occlusion selection technique.

First we compare the preferences and the performances of each technique on each device. With this comparison, our goal is to identify the technique that has to be selected by default if we do not know anything about the end user. This can be compared to the general user model presented in [Octavia et al., 2011]. For each technique and for each setup two different global grades between one and seven were given by the participants. The first one was requested after having finished the selections on one device setup. The second one was requested at the end of the experiment. These

Oculus

	Preferences				Performances		
	1st	2nd	3rd		1st	2nd	3rd
Ray	27	21	3		22	20	9
BentRay	26	21	4		22	21	8
Occlusion	9	10	32		7	10	34

Zspace

	Preferences				Performances		
	1st	2nd	3rd		1st	2nd	3rd
Ray	19	30	2		17	24	10
BentRay	35	11	5		24	17	10
Occlusion	11	12	27		9	10	31

TABLE 7.1 – For each technique number of times they have been ranked as first, second or third according to the preference or the performance. For the preferences, when two techniques got the same grade, they are both classified in the same unit. Results for the 3D Ray-based interaction technique and for the 3D Bentray are similar, even if this last one seems to be preferred on the zSpace. Moreover, participants less preferred and less performed with the Occlusion selection technique.

two grades have been added in order to have the global appreciation of the user for each technique. For the performances we have used the scores that we computed at runtime that correspond to a combination between the selections times and the number of errors. The more this score is important, the better the performances were.

The results of this comparison are compiled in Figure 7.6 and in Table 7.1. First, in Figure 7.6, we present a boxplot that presents a comparison of the global performances and preferences results of the three techniques on the two devices setups. Second, in Table 7.1, we present how many time each technique has been ranked at the first, second and third places for the preferences and performances. From this Figure and this Table we can say that the results for the 3D Ray-based interaction technique and the 3D Bentray are similar even if the 3D Bentray has been slightly more preferred on the zSpace. To continue, we also notice that participants less preferred and less performed with the Occlusion selection technique on the two device setups. On the Oculus Rift, an Analysis of Variance (global ANOVA) revealed that the technique used had a significant effect on the preferences ($F_{1,150} = 15.23, p < 0.005$). For this criterion, no significant result was found between the 3D Ray-based interaction technique and the 3D Bentray. Nevertheless, the Occlusion selection technique was significantly worse than the 3D Bentray ($F_{1,100} = 25.59, p < 0.005$) and significantly worse than the 3D-Ray based interaction technique ($F_{1,100} = 20.49, p < 0.005$). For this same device setup, an ANOVA revealed that the interaction technique also had a significant impact on the performances ($F_{2,150} = 10.54, p < 0.005$). No significant result between the 3D Ray-based interaction technique and the 3D Bentray was found. Nevertheless, for the performances, the Occlusion selection technique was significantly worse than the 3D Bentray ($F_{1,100} = 12.16, p < 0.005$) and significantly worse than the 3D-Ray based interaction technique ($F_{1,100} = 14.82, p < 0.005$). Similar observations can be made on the zSpace. Indeed, an ANOVA revealed that the technique used also had a significant effect on the preferences ($F_{2,150} = 23.98, p < 0.005$). No significant result was found between the 3D-Ray based interaction technique and the 3D Bentray. Nevertheless, the Occlusion selection technique was significantly worse than the 3D Bentray ($F_{1,100} = 36.76, p < 0.005$) and significantly worse than the 3D Ray-based interaction technique ($F_{1,100} = 25.67, p < 0.005$).

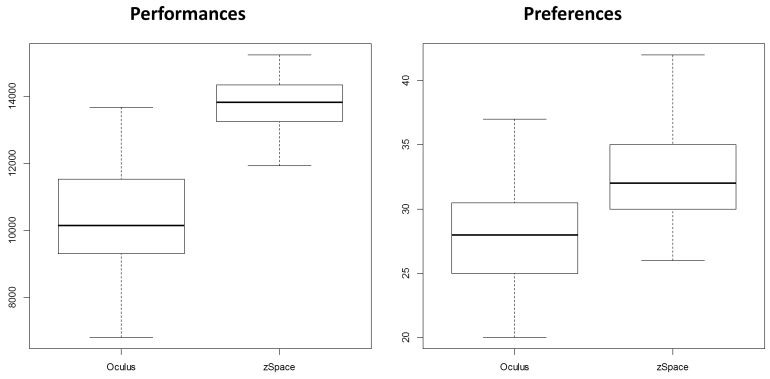


FIGURE 7.7 – Comparison of the global performances and of the global grades between the two device setups. As shown the performances have been globally better on the zSpace. In the same way the three interaction techniques have globally obtained better grades on the zSpace than with the Oculus rift.

For this same device setup, an ANOVA revealed that the interaction technique also had a significant impact on the performances ($F_{2,150} = 5,39p < 0.05$). No significant result between the 3D Ray-based interaction technique and the 3D Bentray was found. Nevertheless, for the performances, the Occlusion selection technique was significantly worse than the 3D Bentray ($F_{1,100} = 7.19, p < 0.05$) and significantly worse than the 3D-Ray based interaction technique ($F_{1,100} = 6.34, p < 0.05$).

As in most cases presented here the 3D Ray-based interaction technique and the 3D Bentray got similar results we can say that in general case it is complicated to choose a technique that will satisfy all users. This assessment justifies the need for user adaptation. Moreover, we have also shown that the 3D Bentray has been slightly more preferred on the zSpace. This is a first clue for demonstrating our first objective. Indeed, it is possible that the preferred interaction technique differed from a device setup to another one even if the two are similar.

Our **first objective** is to verify the consistency between the performances and the preferences on the two device setups. First, in order to demonstrate this objective we compare the scores and the qualitative grades obtained on both hardware configurations. To do so, for a given setup, all grades and all scores for all techniques have been added for each user. We compare the results obtained on the two setups. As shown in Figure 7.7 the performances have been globally better on the zSpace than with the Oculus Rift. One-way analysis of variance (ANOVA) showed a significant main effect of the device setup on the performances ($F_{1,100} = 131.0, p < 0.005$). In the same way the three interaction techniques have globally obtained better grades on the zSpace than with the Oculus rift. One-way analysis of variance (ANOVA) showed a significant main effect of the device setup on the grades ($F_{1,100} = 33.57, p < 0.005$). From these two results, we can say that participants significantly preferred to interact on the zSpace and significantly performed better on this same device setup. Therefore, we can say that even with the similarities between the two device setups (the same kind of 6-DOF controller was used to control the interaction techniques), the performances and global appreciations were different.

To continue, in order to demonstrate this objective we also want to know if the preferred interaction techniques or the ones that got the best performances are different on the two device setups. Therefore, we have computed the number of times the preferred interaction technique or the one that performed the best has changed from the first device setup to the second one. We obtained the following results:

- Regarding the preferences grades, 10 participants (19,6%) did not preferred the same interaction technique on both device setups. For these 10 participants, the grade of a preferred technique on a given device setup has decrease by an average of 14.70% on the other device setup.

- Regarding the performances scores, for 33 participants (64.7%)the interaction technique that obtained the best performance was not the same on the two device setups. For these 33

participants, the score of a technique that performed the best on a given device setup has decreased by an average of to 8.94% on the other device setup.

From these results, we can say that the interaction technique with which a user performs the best will not always be the same on all devices setup. The same observation can be made for the preferences. The technique that a user prefers will not always the be same on all hardware configurations. As the main difference between the two setups was the display type we can assume that this parameter has an important impact on the preferred technique and on the one that performs the best. This parameter has to be possibly taken into account in the adaptation mechanisms in 3DPlasticToolkit.

Our **second objective** is to verify the consistency between performances and preferences. In order to demonstrate this objective, we have computed on each setup the number of times the technique that obtained the best performance did not correspond to the one that the user preferred. In case of equality of grades at the first place, multiple techniques can be considered as the preferred one. We obtained the following results:

- With the Oculus, for 25 participants (49%), the preferred interaction technique did not correspond the one that performed the best. For these 25 participants, the average difference between the performance score of the technique that performed the best and the performance score of the preferred technique is 8.22%. The average difference between the grade of the preferred technique and the grade of the technique that performed the best is 22.32%.

- On the zSpace, for 21 participants (41%), the preferred interaction technique did not correspond the one that performed the best. For these 21 participants, the average difference between the performance score of the technique that performed the best and the performance score of the preferred technique is 6.59%. The average difference between the grade of the preferred technique and the grade of the technique that performed the best is 21.64%.

For the two device setups we obtain similar results. Approximately one on two users did not preferred the same technique that the one with which he/she performed the best. The average differences that we computed are important enough to conclude that when selecting a technique for a user a choice has to be made, in a lot of cases there will be a technique that will maximize his/her performances and another one that will maximize his/her preferences. Another possibility could also be to select a technique by making a compromise between performances and preferences.

One particular point of **objective 3.a** consists in demonstrating if there is a correlation between the results obtained in the preliminary 2D preliminary selection task and the results obtained in the 3D selection tasks. If this point can be demonstrated, such a 2D pre-test could be used in order to detect the skills of a user before a real 3D application. Therefore the application could be automatically adapted, for example it could be possible to give more instructions to a user with low skills. In order to demonstrate this point, we compare the global preferences for the 3D test obtained by the participants on each device setup with the global preferences obtained by the participants on the 2D preliminary test. For the 3D application, the selection times for the three techniques and for all scene configurations have been added. For the 2D application, the selection times obtained for the three different ID have also been added. We obtain the following results:

- First, we have computed the correlation factor between the selection times obtained on the 2D pre-test and the selection times obtained on the 3D application (times for the two setups have been added). We found a positive correlation between this data: $r = 0.303$. This result is significant $F_{1,49} = 4.95, p < 0.05$.

- Second, we have computed the correlation factor between the selection times obtained on the 2D pre-test and the selection times obtained on the 3D application on the zSpace. We found a positive correlation between this data: $r = 0.451$. This result is significant $F_{1,49} = 12.52, p < 0.005$.

- Second, we have computed the correlation factor between the selection times obtained on the 2D pre-test and the selection times obtained on the 3D application on the Oculus rift. We found a positive correlation between this data: $r = 0.234$. This result is not significant.

From these observations we can say that there is a small positive correlation between the performances obtained on the 2D pre-test and the ones obtained with the 3D application. Nevertheless, the results show that the positive correlation is more important with the performances on the zSpace. Therefore, we could say that if we want to anticipate the performances of a user with a pre-test, it is important that this pre-test is performed with the same device configuration. Moreover, the correlations were not important enough to totally demonstrate the efficiency of such a pre-test to anticipate the performances of a user in 3D selection tasks. In the next Section, we try to use the results of this pre-test in order to choose automatically a 3D selection technique.

7.3.2 Could we use Machine Learning to Detect User Preferences?

From this experiment we have obtained a dataset that includes the correspondences between each user profile and each user results from the 2D pre-test with his/her preferred technique and his/her best performing one on each setup. As explained in **objectives 3.a and 3b**, our goal is to determine if from this dataset we could automatically predict the interaction technique that would correspond to a new user.

This is the goal of supervised machine learning. Supervised machine learning aims to infer a function from labeled training data in order to predict unseen points [Mohri et al., 2012]. In order to create a mapping function, supervised learning algorithms are trained on labeled training data that include examples composed of an input object (a vector) and a label (the desired output). To do so, a loss error is minimized between the labels predicted by the function and the initial real labels. From this training a function that generalizes the problematics is created and in the ideal case this function will be able to predict the label of an unseen example. To evaluate the prediction robustness and accuracy of such method datasets are commonly separated in two parts. First the training dataset on which the algorithm is trained. Second, the testing dataset is composed of unseen points, not present in the training set on which the generalization ability is evaluated. The ratio *Good predictions/All predictions* is used to quantify the accuracy. Our objective is to exploit our dataset obtained from our preliminary experiment to create such function that could possibly detect the appropriate technique for a new user.

Here, we use a Support-Vector-Machine (SVM) classifier as supervised learning algorithm. A SVM classifier aims to construct from training examples a set of hyperplanes in a high dimensional space that separate the data into two classes (positive or negative) [Boser et al., 1992]. Recent approaches also exploit SVMs to deal with multi-classes problems. This is the kind of approach we use here as our data are labeled with three possible classes: the three interaction techniques (3D-Ray, 3D-Bentray or Occlusion). Our implementation is based on LIBSVM [Chang and Lin, 2011] that combines multiple binary classifiers in order to create a multi-classes classifier. Our choice is strongly motivated by the ability of SVMs to deal with small training unlike other algorithms such as neuronal networks and random forests. Indeed, our database only contains 51 examples, which would ne be enough for these classes of algorithms.

Each of our examples can be formulated as a pair $\{x_i; y_i\}$ in $\mathbb{R}^n \times \{0, 1, 2\}$. Here, x_i corresponds to the input feature vector and contains information extracted from the profile or from the 2D preliminary tests. In this Section, we use different compositions (different sizes and different values extracted from the profile or from the pre-test) of x_i as all compositions do not provide the same results depending on what we try to predict (preferences or performances on the zSpace or on the Oculus rift). In all cases, each value contained in x_i is normalized. Binary data such as "handedness" or "experience with VR" is set to 0 or 1. The dimensions of the different feature vectors used are kept fixed during the learning process as we use a linear kernel during the SVM optimization. y_i corresponds to the label, which is the interaction techniques associated to the profile (the preferred one or the one with the best performance), 0 corresponds to the 3D-Ray technique, 1 to the 3D Bentray and 2 to the Occlusion technique. For a given user, in case of equality between the qualitative grades obtained by multiple techniques, only one technique has been selected as the preferred one according to performances ranking.

In order to determine the prediction accuracy, in all cases we separate our data into two datasets: 40 examples in the training set and 11 examples in the testing set. During the training step, we have performed a k-fold cross validation in order to optimize one parameter of the SVM. This parameter named C tells to the SVM optimization problem how much it has to avoid misclassifying each

training example. This cross-validation method consists in randomly partitioning the training set into k subsamples. Then the model is trained on $(k-1)$ subsamples and tested on the remaining subsample. The process is repeated k times for each value of C we want to evaluate. During this cross validation, the metric to optimize in order to choose the best value of C is the ratio *Good predictions/Number of predictions*. Because of the small size of our training data, the generalization ability is a very challenging task. To enhance and guaranty this ability, we have decided to repeat the process 1000 times where each time the training set and the testing set are fulfilled randomly. The obtained results correspond to the average value of the ratio of good predictions on the testing set and the standard deviation for this same value obtained for the SVM classifiers evaluated on the 1000 separations. We test different SVM to try to predict: the preferred technique on the Oculus Rift, the best performing technique on the Oculus Rift, the preferred technique on the zSpace, the best performing technique on the zSpace. Indeed, we have seen in the previous Section that the results can differ for each of these configurations. For all cases, We compare the results obtained for the SVM with the result with a simple "majority" method. This method can be compared to the general user model proposed by Octavia et al. [Octavia et al., 2011]. It consisted in selecting for a new user the technique that has performed or preferred the most by the participants on one setup. Our goal is to obtain better ratios with our machine learning approach.

Prediction on the Oculus Rift

First, we evaluate our approach on the Oculus Rift. For evaluating the preferences and the performances with the data from the users profiles we use the following feature vector x_i:

$$x_i = \{Wear\ glasses\ (binary),\ VR\ experience\ (binary),\ Hour/week\ VR(real),\\ Video\ game\ experience\ (binary),\ Hour/week\ Videogame\ (real)\}$$

For evaluating the preferences and the performances with the data from the users profiles we use the following feature vector x_i:

$$x_i = \{Number\ Click\ Outside\ (real),\ Total\ Selection\ Time\ (real),\ Min\ Selection\ Time\ (real),\\ Click\ Mean\ Distance\ Center\ (real),\ Click\ Min\ Distance\ Center\ (real),\\ Click\ Max\ Distance\ Center\ (real), Distance\ Max\ Straight\ Path\ (real)\}$$

The results are compiled in Table 7.2.

As shown in the Table, for detecting the adapted technique of an unseen user on the Oculus Rift, using a SVM trained on the profile data or on the 2D pre-test data could slightly improve the prediction. Indeed, both method provide better performances than just applying the preferred technique or the best performing one for all new users. Our method has the advantage to possibly propose the three selection techniques to the end-user. Nevertheless, the differences are not important enough to really demonstrate the efficiency of the method. Indeed, with our approach between 43% and 51% of new users would not get the appropriate selection technique.

Prediction on the zSpace

Second, we evaluate our approach on the zSpace. For evaluating the preferences and the performances with the data from the users profiles we have changed the vector x_i for the zSpace. Indeed, we did not obtained similar ratios with the same one. This vector is the following one:

$$x_i = \{Age\ (real),\ VR\ experience\ (binary),\ Hour/week\ VR(real),\\ Video\ game\ experience\ (binary),\ Hour/week\ Videogame\ (real)\}$$

For evaluating the preferences and the performances we have also changed the vector x_i fulfilled with the data from the pre-test. This following feature vector x_i:

$$x_i = \{Number\ Click\ Outside\ (real),\ Total\ Selection\ Time\ (real),\ Min\ Selection\ Time\ (real),\\ Click\ Max\ Distance\ Center\ (real), Distance\ Mean\ Straight\ Path\ (real)\}$$

The results are compiled in Table 7.3.

	« Majority » method	SVM with profile features	SVM with pre-test features
Ratio of good predictions for the preferred technique	25/51: M = 49.02%	M = 57.17% SD = 12.99%	M= 52.95% SD = 13.85%
Ratio of good predictions for the best performing technique	22/51: M=43.14%	M = 55.60% SD= 13.97 %	M = 49.14% SD= 13.03 %

TABLE 7.2 – Comparison of the prediction ratios obtained with SVMs for the Oculus rift. M corresponds to the mean ratio of good predictions and SD to the standard deviation.

	« Majority » method	SVM with profile features	SVM with pre-test features
Ratio of good predictions for the preferred technique	24/51: M = 47.05%	M = 52.36% SD = 13.94%	M = 39.12% SD = 12.77%
Ratio of good predictions for the best performing technique	24/51: M=47.05%	M = 35.98% SD = 11.79%	M = 38.47% SD = 12.97%

TABLE 7.3 – Comparison of the prediction ratios obtained with SVMs for the zSpace. M corresponds to the mean ratio of good predictions and SD to the standard deviation.

As shown in the Table, the results obtained on the zSpace are worse than those on the Oculus Rift. Indeed, only the SVM based on the profile features for detecting the preferred technique gives a better ratio than the "Majority" method. However, the improvement between the two approaches is only 5.31%. For the three other cases, the results are not good enough to possibly detect the adapted technique for a new user.

Conclusion

To conclude on our machine learning approach, the results are encouraging enough to tell that it could be applied in order to detect the adapted technique of a new user. We have obtained better results with the profile features than for the pre-test features. In the same way, better results have been obtained on the Oculus Rift than on the zSpace. Regarding these differences, we think that the difficulty to predict the good technique on the zSpace can be explained by the fact that the differences of preferences and performances were less significant on this setup. Participants globally preferred the three techniques and had better performances on the zSpace. We think that the participants noticed less differences between the three techniques on this setup and therefore it is more difficult in that case to predict the adapted technique. To continue, the improvements of ratio of good predictions compared to the "Majority" method are not important enough to really apply our SVMs in their current states in order to detect the adapted interaction technique of a new user. Even with our better ratio, 43% of users would not get the adapted technique. Some work need to be done in order to demonstrate if these ratios can be improved.

Two main perspective of work can be identified in order to improve these results. First, the approach must be tested with a larger database. Indeed, we assume that our dataset composed of 51 participants was not big enough to really create accurate SVMs. Second, with more users in the database, it could be interesting to experiment the learning process with other algorithms such as the previously cited neural networks and random forests.

Last, the approach also suffers from generalization. Indeed, it can only detect the adapted technique between three interaction techniques for a selection task and for the two given setups. Other experiments would be needed if we want to create others SVMs to predict the adapted navigation interaction technique or the adapted manipulation technique. It could be possible to use SVMs to predict global characteristics from the user profile or from a preliminary test and then match these characteristics with the properties of some interaction techniques. For example, we could detect automatically a novice user with low skill and propose him/her automatically 3D menus with bigger icons and an assisted selection technique such as the 3D Bentray.

7.4 Extension Of 3DPlasticToolkit for User Adaptation

7.4.1 User Model Integration

Model contents and task association

The goal of the user model is to describe the users that will interact with the application and therefore perform user adaptations. As for the other configuration files of 3DPlasticToolkit, the configuration of the different user models is also performed with a XML file. One example of such a file is given in Listing 7.1. This user model must contain the four types of properties that we have defined in Section 3.3.2.

First, the **user profile** contains the different properties that characterize the user. These properties can be for example the user age, his/her gender, his/her level of expertise. In the user model they are represented as key-value properties. These properties are shown in the first part of the XML file presented in Listing 7.1.

Second, the user model contains the **user preferences**. These preferences are represented as scores that will be taken into account by the 3DPlasticToolkit adaptation process in order to instantiate the application components at runtime through one of the scoring modules presented in the next Section. Multiple scores can be contained in the user model:

- As proposed in Chapter 4, a score *Sa* represents the compatibility between a high level task and a concrete application component. The user can also express how he perceives this compatibility by including a score *Sau* in his model. For instance, it can tell to the system which interaction techniques or which visual effect the user prefers. As demonstrated in our user evaluation this compatibility can differ according the type of display used. Therefore, as shown at line 12 of Listing 7.1, this compatibility can be parametrized with a display type.

- In the application model, the scores *Sld* represent a compatibility between an application component and a logical driver. The user model includes scores *Sldu* to expose preferences

```
1 <UserConfig>
2 <User userId="1">
3 <Profile key="Name" value="Bernard"/>
4 <Profile key="Age" value="35"/>
5 <Profile key="Gender" value="M"/>
6 <Profile key="Expertise" value="Novice"/>
7 <Profile key="MasterEye" value="Right"/>
8 <UserPrefComponent Name="3DRay" Task="SelectionManipulation" score="1.0">
9    <UserPrefDriver Name="3DRay6DofDriver" score="1.5" />
10   <UserPrefDriver Name="3DRayMouseDriver" score="0.5" />
11 </UserPrefComponent>
12 <UserPrefComponent Name="3DCursor" displayType="HMD" Task="SelectionManipulation" score="0.8">
13  (...)
14 </UserPrefComponent>
15 <Environment key="Weather" value="Rainy"/>
16 <Environment key="Noise" value="Quiet"/>
17 </User>
18 </UserConfig>
```

Listing 7.1 – An example of user configuration file. Here one user is described. First the model includes the user profile. Here, it includes the user's name, his/her age, his/her gender, his/her expertise and his/her master eye. Second, his/her preferences are also included. For a given task a score is assigned to a compatible application component as shown in lines 8 and 12. As shown at line 12, this compatibility can be parametrized with a display type. Then, in lines 9 and 10, for a given application component, score for its compatible logical drivers and rendering presentations can also be assigned. Third, information about the environment of the user can also be added. For instance, here, the properties concern the weather and the sound atmosphere at the user's place.

for this compatibility. For example, the user may prefer using some kinds of devices to control a specific interaction technique.

- In the same way, the scores *Srp* represent a compatibility between an application component and a rendering presentation facet. The user model can also expose preferences for these facets with scores *Srpu*. For instance, it can express a preference for a particular representation of a 3D widget.

Third, information about the **user environment** can also be added. These properties refer to the physical and social properties of his/her environment. As for the user profile, in the user model they are represented as key-value properties. In the configuration file provided in Listing 7.1, two examples of such properties are given. These properties concern the weather and the sound atmosphere at the user's place. For instance, by knowing the sound atmosphere at the user's place it could be possible to change dynamically the sound volume of an application.

Last, at runtime the user model can also be updated with **monitoring information**. In that case, data about the interaction between the application are collected and reported in the user model. Our implementation of the user model in 3DPlasticToolkit allows the developer to collect such information with different data-structures. For example, in a 3D selection application such as the one presented in the previous Section, it could be interesting to monitor the user performances in order to automatically propose him/her another interaction technique if he/she has some difficulties.

As said, both Grappl and CATHI represent an interaction task as an action performed by a user via a user interface. However, these two solutions do not represent this relation into the system while it would be needed to perform user adaptations. Therefore, in 3DPlasticToolkit we represent this relation by associating each task to a particular user. Such associations are illustrated in listing 7.2 where all tasks are associated with the same user whose index is "1". With this association, the score of a triplet (application component, logical driver, rendering presentation) can be computed according to the user model as explained in the next Section. With this association, a deployed component can have access to all the properties of the user. It can be used to precisely adapt a concrete application component to a user. For instance, in Section 7.3 the occlusion selection technique uses the user's master eye in its selection process. As this property can be included

```
<TaskConfig>
<NeedTask userId="1" taskName="Navigation" taskId="0" topTask="0" ScoringModule="
    UserPreferences"/>
<NeedTask userId="1" taskName="Sketching" taskId="1" ScoringModule="UserAndDeveloper"/>
<NeedTask userId="1" taskName="Selection" taskId="2" ScoringModule= "LearningUser" >
</NeedTask>
</NeedTask>
</TaskConfig>
```

Listing 7.2 – The task description in the 3DPlasticToolkit configuration file. Here three tasks are associated with the three scoring modules presented in this Section. The "Navigation" task is associated with the scoring module that only takes the user preferences scores. The "Sketching" task is associated with the scoring module that combines the user's scores and the developer's ones. The "Selection" task is associated with the scoring module that uses a pre-trained machine learning algorithm to compute a triplet score.

in the user model (in the user profile part), the application component that corresponds to this interaction technique can directly have access to this information. As different users can be associated to different tasks in the same application, it could allow to take into account multiple users collaborating in the same application, for example with a multi-user stereoscopic display such as the Two-User Responsive Workbench presented by Agrawala et al. [Agrawala et al., 1997]. For now two possibilities are implemented to select the user associated to each task. First, it can be directly edited in the 3DPlasticToolkit file as shown in Listing 7.2. Second, it can also be edited with the meta-user interface. A login page could also be implemented in order to allow a user to load his/her profile from a distant server and associate it with all the high level tasks of an application. Another possibility could be to detect automatically the person that is using the application. For example, such a solution would use a face recognition algorithm in order to attach all current tasks to the detected user.

User model update

As explained before, there are two ways of updating a user model. First, the explicit way when the user edits himself/herself his/her properties. Second, the implicit way when the system automatically detects and updates some properties of the user model. For now, 3DPlasticToolkit fully includes the first way, the second one being partially implemented.

For the explicit way, there are two possibilities. First the user can directly edit the XML configuration file and fulfill his/her properties. Nevertheless, for a novice user without knowledge about 3DPlasticToolkit and about XML it can be a difficult task. Therefore, at runtime the meta-user interface can also be used to update the user model. Another possibility could be to edit this model between sessions and to load it at runtime through a login page as explained before.

In order to support the implicit way, some solutions are currently in development. For instance, as explained in Section 4.7.2, the application components, logical drivers and the rendering presentations can be modified with the meta-user interface. When a user changes one of these components in the meta-user interface and keeps the new one until the end of the application, the preferences scores of the two components could be swapped. The goal would be to learn from the user habits in order to automatically adapt the application another time. For now this feature is not implemented. Second, we included a "UserMonitoring" task and we plan to develop some compatible components for updating automatically the user model at runtime. It would allow the developers to develop their own monitoring components. With this approach some logical drivers could be implemented in order to detect some information about the user environment. For instance, a microphone could be used to detect the sound atmosphere. One example of component for this task that we have implemented is based on a weather sensor virtual device based on the web API: OpenWeatherMap [1]. Our application component that monitors the user uses a logical driver that exploits this virtual device in order to update the weather information in the user model. Then we have used this information in order to dynamically adapt the weather conditions in the VE by changing the skybox of the scene.

[1] http://openweathermap.org/api

```
1  public class ScoringModuleUserPreferences : ScoringModule
2  (...)
3  public float getTripletScore(Task task, string componentName, string logicalDriverName, string
       renderingPresentationName)
4  {
5      IndividualUser user = task.associatedUser;
6      float Sau = user.getCompatibilityControl(task.Name , componentName) ;
7      float Sldu = user.getCompatibilityLogicalDriver(logicalDriverName);
8      float Srpu = user.getCompatibilityRenderingPresentation(renderingPresentationName);
9      return Sau+Sldu+Srpu ;
10 }
11 (...)
12 }
```

Listing 7.3 – The implementation of the user preferences scoring module. The function that computes the score of a triplet adds the user preferences scores for the application component, for the logical driver and for the rendering presentation.

7.4.2 Scoring Module For User Adaptation

In order to take into account this user model in 3DPlasticToolkit three different scoring modules have been implemented. They are named "UserPreferences", UserAndDeveloper" and "Machine-Learning". As explained in Section 4.7, a scoring module implements a particular way to compute the scores of a triplet (application component, logical driver, rendering presentation) that can be associated to a task in the current context of use. The goal of this module is to order these different triplets for the adaption process. Then the adaption process automatically deploys the triplet that gets the better score.

As explained in the previous Section, in the user model, each user has his/her own preferences for the compatibilities between task and applications components, and between components and logical drivers and rendering presentations. Therefore, each user model contains the following scores: *Sau*, *Srpu* and Sldu.

The first scoring module named "UserPreferences" provides the maximal importance to these scores when ranking the different compatible components. Its name is "User Preferences". This is the one associated with the task "Navigation" in Listing 7.2. This module only uses the scores extracted from the user preferences. The goal is to provide the application that corresponds as much as possible to the user needs. The score for a given triplet is:

$$S = Sau + Sldu + Srpu \tag{7.2}$$

As these scores can be edited in the meta-user interface by the end-user, it gives him another possibility to change dynamically the application components currently deployed. Nevertheless in that case, the compatibility scores provided by the developper of the application are not taken into account. Therefore, in that case the developer has not really the control of adaptation process.

That is why the second scoring module named "UserAndDeveloper" combines both the scores in the user model (*Sau*, *Srpu* and *Sldu*) and the ones provided by the developer (*Sa*, *Srp* and *Sld*). This is the one associated with the task "Sketching" in Listing 7.2. The implementation of this module in 3DPlasticToolkit is given in Listing 7.3. It provides a good compromise between the user preferences and the developers and designers choices. This module proposes to add these different scores to compute the score of a triplet:

$$S = (Sa + Sau) + (Sld + Sldu) + (Srp + Srpu) \tag{7.3}$$

If needed, it could also be possible to weight the importance given to the developers scores and to the users ones. In that case, this scoring module can be parametrized with two values Wd and Wu that correspond respectively to weight for the developers scores and to the weight for user scores. In that case, the score of triplet is computed as:

$$S = Wd \times (Sa + Sld + Srp) + Wu \times (Sau + Sldu + Srpu) \tag{7.4}$$

With these two modules, the adaptation process can choose the interaction techniques and the content visualization that best match the user preferences and the developers needs. As explained

in the previous Section two methods are proposed to acquire the user preferences scores. A user can directly edit his/her score with the meta-user interface, or they are automatically updated when he/she replaces a component by another one with the same interface. The first solution can be quite difficult to understand for a novice user with poor knowledge about MR user interfaces. The second one forces the user to try multiple application components before being able to know which one he prefers. A better solution could consist in performing an automatic detection of the user preferences by analyzing his profile or his performances during the interaction.

This is what we have proposed in Section 7.3.2 with the use of machine learning. This is the one associated with the task "Selection" in Listing 7.2. For now, this scoring module is not implemented in 3DPlasticToolkit. This module computes the scores of application components according to a pre-trained SVM. This module can only be associated to the "Selection" task as it the only one for which we have trained SVMs on a dataset. In the same way, the scores are computed from the pre-trained SVM only if the main display is an HMD (as the Oculus rift) or a stereoscopic screen (as the zSpace). Of course other SVMs could also be trained with other displays and for other tasks and components. For computing the score of a triplet, the scores that correspond to the logical driver and to the rendering presentation are taken from the user preferences scores as for the "UserPreferences" scoring module. Only the score for the global component is computed from the prediction of the SVM. The scoring module has to be possibly configured by the developer in order to determine if it computes a prediction from finding the preferred technique or the best performing one for a given user. Therefore, the score of a triplet is computed as follows:

$$S = Sasvm + Sldu + Srpu \tag{7.5}$$

with $Sasvm = 1$ if the corresponding component is the one predicted by the SVM, however $Sasvm = 0$.

The main limitation of this module lies in its difficulty to be applied for other tasks.s, other displays and other components as it needs a dedicated pre-trained SVM. It also requires profiles containing the needed properties to make the prediction with the SVM (the user profile informations or the data from a preliminary test). Moreover, as explained before, the ratio of good predictions are not good enough to really ensure the efficiency of the method.

7.5 Conclusion

In this Chapter we present how we have integrated user adaptation in our plasticity models and in 3DPlasticToolkit. The goal is to include another adaptation source in order to complete **R1**.

With a preliminary experiment we have identified the different issues to consider for dealing with user as an adaptation source. The goal of this experiment was to compare the preferences and performances for three 3D selection techniques on two device setups. First we have shown that the preferred or the best performing interaction technique of a user can differ on two similar hardware configurations. Indeed, the main difference between our two setups was the display type, an HMD or a stereoscopic screen. We have also demonstrated that the preferred technique of a user is not always the one with which he/she performs the best. That is why, the developer has to determine if the chosen adapted technique will match the preferences of the user or his/her performances. From this experiment we propose a machine learning approach based on SVMs in order to detect the adapted technique for a new user from his/her profile or from the monitoring information obtained during a 2D selection pre-test. The results obtained with this approach are encouraging enough to tell that it could be possibly integrated in a user adaptation process. Nevertheless, the obtained ratio of good predictions are not good enough to really apply our SVMs in their current states to detect the adapted interaction user for a new user. As perspective of work, we could test our approach with more data or with another machine learning algorithm. Then, based on these different results we propose a user model that can include properties about the user profile, the user environment, the user preferences and also user monitoring information. As each high level task in 3DPlasticToolkit is associated with a particular user the deployed application components can be chosen according the user preferences and can be parametrized according to the user properties. For this deployment we propose three scoring modules. The first one only uses the preferences scores of the user in order to choose the application components that will match his/her preferences. Second,

we propose a scoring module that combines the preferences scores with the developer ones in order to make a compromise between the developer choices and the user choices. The last scoring module uses our machine learning approach in order to compute the score of an application component. More work should be done to improve the accuracy of our machine learning approach in order to really use it in an application.

Regarding the experiment, the performances and the preferences have been evaluated for similar types of 3D scenes. It would be interesting to see if the preferences and performances remain the same for a different virtual environment with other types of objects to select. More work could also be done in order to apply an automatic detection of the users preferences. Indeed, for now the main part of our approach depends on the scores provided by the end-user. More than just improving the accuracy of our machine learning approach, its generalization capability could also be improved. The current approach cannot be extended to other tasks and to other interaction techniques. Therefore, one objective could be to use machine learning in order to predict the user global characteristics from his/her user profile or from a preliminary test. These characteristics could then be matched with the properties of some interaction techniques in order to select the best one. Another possibility could also be to integrate a rule-based system in order to compute the preferences scores from the other properties of the user model.

Chapitre 8

Data Adaptation

8.1 Introduction

In a MR user interface a user is interacting with data, he/she can visualize them, modify them and manipulate them. This is particularly true for data-visualization applications but it is also true for other use cases. For instance, in a training application a user can have access to different kinds of documents such as procedures documents and diagrams. In the same way, for an AR application for assisted surgery, medical data are displayed to the end-user. As illustrated in Figure 8.1, the main focus of this Chapter is on the adaptation of the content visualization of an application according to the data structure and semantics.

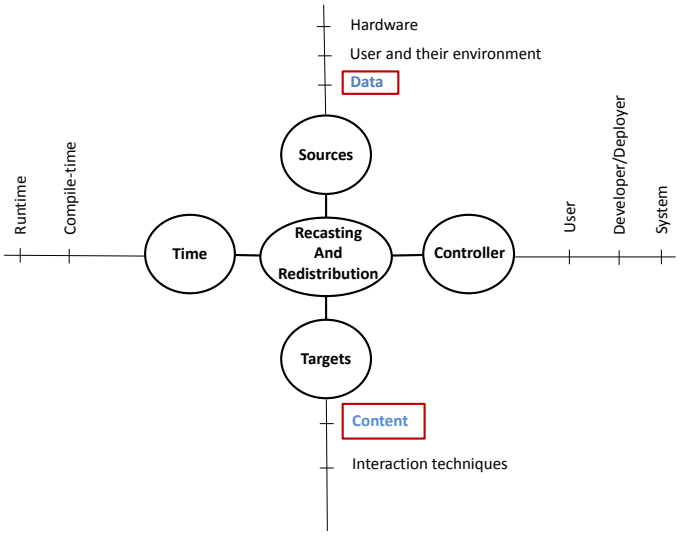


FIGURE 8.1 – Regarding our design space problem of plasticity for MR user interfaces, this chapter is interested in adapting the content visualization of an application according to data structure and semantics. At a lower level, we also show that the data can also be combined with the devices properties and the users properties.

As detailed in Chapter 3, considering the data as an adaptation source is one of the requirements for the development of plastic MR user interfaces. The data can really have an impact on the final aspect of an application, more specifically on its content visualization. First, the structure of the data can be taken into account in order to adapt a MR user interface. For instance the hierarchical structure of a database can automatically be used in order to choose a visual metaphor that can transpose the hierarchy in the virtual environment. In the same way, the data semantics can also be used in order to choose an adapted metaphor. For instance, a visual metaphor based on a virtual

museum can be used in order to display cultural artifacts [Bonis et al., 2009]while a virtual movie theater metaphor can be used to present a video on demand catalog such as in [Esnault et al., 2010].

In this Chapter we present a model for integrating data as an adaptation source in our plasticity models. We first propose an integration of a data model that can load large databases of different types and can expose information about data semantics through Web Ontology Language (OWL). This data model is currently implemented in 3DPlasticToolkit. Second we propose an update of the scoring mechanisms in order to choose an adapted visualization metaphor with adapted parameters to display the results of user data query. This update is based on an optimization function. For now, these scores computations are not implemented in 3DPlasticToolkit. Nevertheless we present a proof of concept that demonstrates different aspects of our solution with some implemented visualization metaphors. This proof of concept is based on an application for the visualization of cultural heritage data contained in the Topic-Topos. database[1].

This Chapter is structured as follows. First, in Section 8.2, we detail the related work that deals with adaptation to the data. Second, in Section 8.3 we present our integration of data modeling in our plasticity models and in 3DPlasticToolkit. To continue, in Section 8.4 we demonstrate how application components that correspond to data visualization metaphors can be deployed with our scoring mechanisms. In Section 8.5, we present a proof of concept demonstrating how our solution can handle data as an adaptation source for a cultural heritage application. Last, in Section 8.6 we conclude on this Chapter and we give some perspectives for future work.

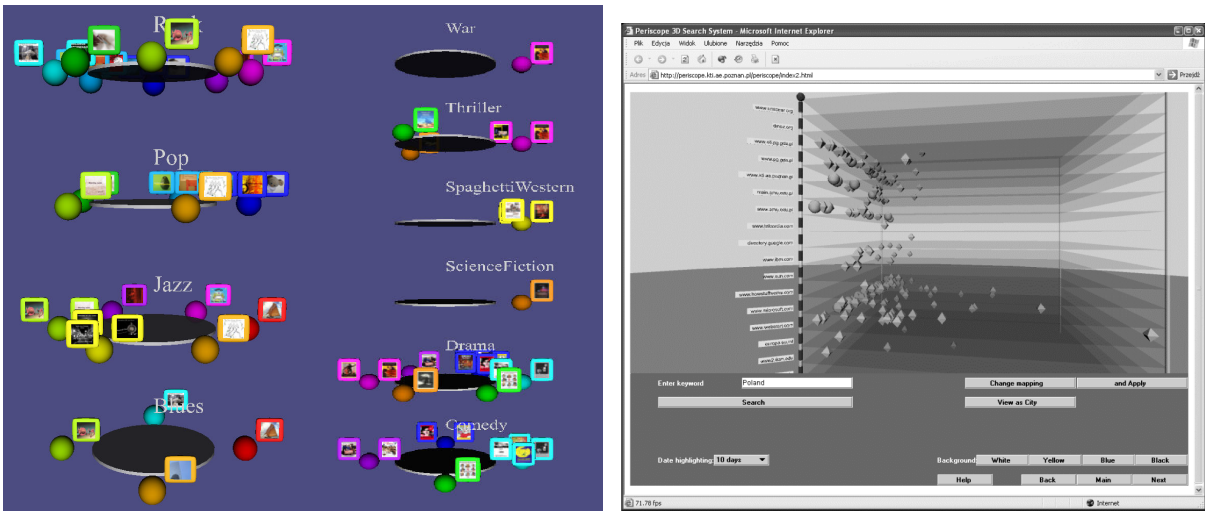
8.2 Related Work

In the field of MR user interfaces only few related work is interested in proposing solutions for dealing atomically with data in order to adapt an application. This is particularly true when we compare the number of such solutions with the number of approaches proposed to deal with adaptation to devices and to users.

In that category, we can for example cite the visualization framewrok proposed by Esnault et al. [Esnault et al., 2010]. It proposes a solution for the creation of 3D representations of databases through XSLT (eXtensible Stylesheet Language Transformations) stylesheets that exploit the extracted data structure. The generated visual representation can be related to the content such as a 3D museum for cultural artifacts or it can also be generic representation such as a carousel or a time-line. For instance, an example of generic metaphor is given in Figure 8.2a. It demonstrates that the same visualization metaphor (a tower of carousels) can be used for two different databases. To continue, in the context of web search results, AVE (Adaptive Visualization Environments) presented by Wiza et al. [Wiza et al., 2003] is a new approach for adapting the graphical presentation of search results according to the structure of the data returned by the search engine. The goal of this approach is to maximize the readability of the interface. Information about the data structure such as the overall number of documents returned, the quantity of keywords, the files types, can be used in order to choose a good visualization metaphor. A scoring mechanism finds the best interface and the best mapping of the data on this interface. In case of equality the choice is given to the end-user. The end-user can also modify different parameters of the final interface such as the color that represents particular types of documents. Three types of interfaces can be instantiated. First *holistic* interfaces are used for voluminous search results and present an entire result through groups of documents that share one or several criteria. Second, *analytical* interfaces present the search results in detail and are well suited for small amount of data. In such interfaces each 3D object represents a single document. Last, a *hybrid* interface presents aggregated data as well as detailed aspects. Nevertheless, the systems only allows the visualization of web search results and nothing is proposed to visualize data represented by 3D models. Content-related metaphors are not included either. In addition, the data semantics is not taken into account during the interface choice. An implementation of the AVE model is proposed in the Periscope system [Wiza et al., 2004] which proposes an intermediary layer between a user and indexing search engines.

In [Golemati et al., 2006], the authors propose a method to match visualization methods against the user, task, available devices and document contexts. To do so they propose a set of properties

[1] http://fr.topic-topos.com/



(a) (b)

FIGURE 8.2 – Two examples of automatic creation of data visualization interfaces. (a) The visualization of two different databases (Music and Movies) with the same visualization metaphor (a tower of carousels) [Esnault et al., 2010]. (b) An example of analytical interface created with the Periscope system [Wiza et al., 2004].

to describe users, tasks devices and documents (the context). In the same way, the visualization methods are described according to different properties. The system uses a rule-based systems in order to compute the score of a method according to its properties and according to the properties of the context elements. The method with the best score is chosen to visualize the data. Nevertheless no solution is proposed in order to fine tune the intrinsic parameters of a visualization methods according to the context of use. In this work, an example based on the visualization of historical archives is introduced.

To continue in order to adapt the visualization according to the data semantics. Bosca et al. [Bosca et al., 2007] propose a solution for the 3D visualization and exploration of OWL ontologies. For this visualization, concepts are rendered as spheres, instances as cubes and semantics relationships between entities are symbolized by arrowed lines. Nevertheless, it does not include any solution to construct a content-related visualization metaphor according to the ontology. Only the structure of the ontology can be visualized.

From all this work we can extract five main types of 3D visualization metaphors: content-related metaphors, generic exploration metaphors, temporal metaphors, maps and 3D diagrams. In order to atomically create such metaphors, two kinds of information about the data can be used: the semantic and the structure. For now, no solution can create the three types of interfaces according to the two kinds of information about the data. Moreover, for most solutions only the choice of the metaphor is impacted by the data properties, the intrinsic parameters of the chosen metaphor do not always depend on the data properties. Our solution aims to fulfill the lacks of this current related work.

8.3 An integration of data modeling

Our first goal in order to support data as an adaptation source is to include a data model in our solution. This data model must include and expose the two types of information that we identified in the related work. First, it must include semantics information about the data in order to describe the global topic of the database and the nature of each element. Second, the structural information must also be exposed by the model or possibly extracted from it. For instance, such information can range from a simple number to complex data structure for representing neighborly relations.

Our approach is separated into two parts as illustrated in Figure 8.3. First, the knowledge

information and the data elements are represented in the data model with ontologies and concrete instances with the use of Web Ontology Language (OWL). Second, we propose the possibility to implement different data loaders in order to support multiple types of databases. Of course OWL files can be loaded seamlessly but other types of databases can also be supported with the intervention of a developer, such as SQL databases.

OWL is a semantic markup language for publishing and sharing ontologies [Bechhofer, 2009]. OWL is developed as a vocabulary extension of RDF (the Resource Description Framework). An ontology is a shared conceptualization of knowledge in a particular domain. With the concept of classes included in OWL, resources with similar characteristics can be grouped. A list of properties for each class can be defined as well as relationships between the different classes. For instance we can imagine a class "Cultural Artifacts" and a list of its subclasses such as "Painting", "Sculpture" and "Book". These elements represent at a high level the concepts, semantics and knowledge about the database entities. The concrete instances of the database are represented with the concept of individual included in OWL. An individual is a concrete example of a class (a class extension) and is defined with properties instances. For instance, as possible individuals that derive from the class "Painting" we could imagine "The Mona Lisa" or "Guernica". With the use of OWL, we benefit from a widely used ontology standard that can expose data elements and their semantics and from which the data structure can be easily extracted.

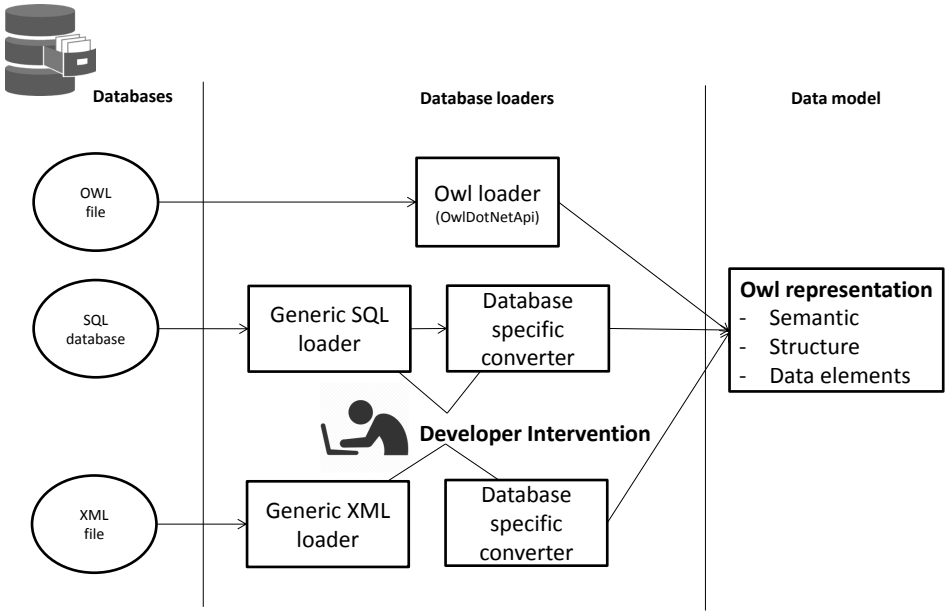


FIGURE 8.3 – The 3DPlasticToolkit data model based on a OWL representation and compatible other databases through the implementation of specific converters by the developer.

An application developed with 3DPlasticToolkit can be configured with the link to the data. For instance, the classic method to fulfill the data model is to load a OWL file as seen in Figure 8.3. This is performed with the global XML configuration file of 3DPlasticToolkit. An extract of this configuration file to configure the data model is given in Listing 8.1. In this example, a OWL file that contains the description of a cultural heritage database is loaded. Regarding the implementation, we have interfaced 3DPlasticToolkit with OwlDotNetApi[2] in order to load OWL files and to handle OWL data at runtime.

Nevertheless, as explained, we must also give the possibility to the developer to load any other kind of databases such as SQL databases or data organized in XML files. For instance, in the ap-

[2]https://github.com/bpellens/owldotnetapi

```
1 <Datamodel dataFile="D:/Data/CulturalArtifacts.owl"/>
```

Listing 8.1 – The data model configuration in the 3DPlasticToolkit configuration file. Here a OWL file is loaded.

plication described in Section 8.5, data are organized in a SQL database and a semantic knowledge is provided as a hierarchy of keywords included in a text file. Therefore, the process of loading an other type of database is performed in two steps as illustrated in Figure 8.3. First, the developer can create a generic module that loads in memory the data included in a given type of database. For instance, a generic module for loading a SQL database in memory can be developed. This loading module can then be reused in other applications for the same type of databases. After this step, the data and its structure are loaded in the memory but nothing describes its semantics. Therefore, a specific converter has to be implemented by the developer in order to convert these data to the OWL representation of our data model. In this converter basic semantic information can be extracted automatically from the fields names of the database or it can also load an additional file that exposes some semantic knowledge about the data. This converter is specific to a given database and has to be redeveloped if this database is replaced by a different one. The currently used converter can be configured in the 3DPlasticToolkit configuration file. For instance, in the application described in Section 8.5, this converter loads the hierarchy of keywords from a text file and converts the concepts as OWL classes in the data model. Then, the data elements are defined as OWL individuals and are associated to their corresponding classes according to their keywords.

With this data model included in our solution, semantic and structural information of the data visualized and manipulated at runtime can be exposed to the different application components and to the adaptation process. In the next Section we describe how this data model can be exploited in order to deploy and to adapt automatically data visualization metaphors.

8.4 Visualization metaphors deployments scores optimization

From our data model our goal is to exploit the information about the data structure and the data semantics in on order to adapt the content visualization of the application.

The adaptation of the content visualization is performed into two main steps as illustrated in Figure 8.4. First, it starts with data selection. In that step the user creates a query to select the data that he/she wants to visualize. In this query the user also indicates the type of the metaphor used to visualize the data. Five types of metaphor can be selected: temporal, cartography, generic exploration, contend-related and diagram. Second, we select a metaphor that corresponds to the type chosen by the user and that best matches the data query. To do so, we propose to optimize the intrinsic parameters of each compatible visualization metaphors according to the data properties. From this optimization a score of compatibility is computed and the metaphor with the best score is selected. This is seamlessly integrated in our adaptation process with our scoring mechanisms. Then if needed, the user can refine his request or perform an another one, and he/she can also change the created visualization interface with the help of the meta-user interface integrated in our solution. This process is not fully implemented in 3DPlasticToolkit. Indeed, for now, as detailed in Section 8.5 only simple queries can be expressed and there is only one concrete metaphor associated to each global type. The scoring mechanism is not included yet.

In the first step of the visualization process, the user gives a data selection query as well as the type of the metaphor that he/she wants to be used. To do so, we propose a high level task "DataSelection" as illustrated in Figure 8.4 for which a concrete application component will be deployed. A compatible component must give the possibility to select data elements according to three different levels:

- At the **concept level** the user selects to display all the elements in the ontology that correspond to a class,
- at the **property level** the user selects the elements that satisfy a common property, for instance he/she could want to visualize the elements created between two given dates,

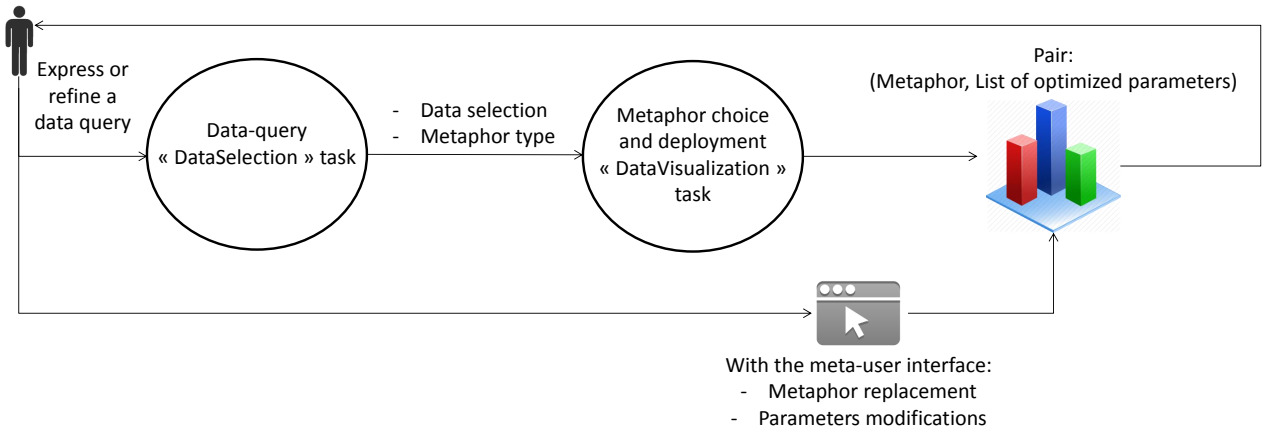


FIGURE 8.4 – The 3DPlasticToolkit data visualization process is composed of two main steps: data selection and visualization metaphor choice.

- at the **element level** the user selects individual elements.

Then, a compatible application component must also give the possibility to the end user to perform basic set operations for his query such as unions, negations and intersections. In order to allow the end-user to provide such a query we can benefit from the capacities of our solution to handle multiple types of devices. Therefore, as concrete application components that could be used in order to perform this query and provide the visualization metaphor type we could imagine:

- A 2D menu with a text input field could be used to enter and confirm the query with a mouse and a keyboard. Such a component could be used on desktop and mobile platforms.
- An application component based on speech recognition could be used in order to allow the user to express vocal queries. It could be used on any platform that has a microphone.
- A 3D menu could be implemented in order to configure a query with a graphical tool. For instance, it could be based on the ontology visualization metaphor proposed by Bosca et al. [Bosca et al., 2007]. Such a component could be used on many platform as long as selections mechanisms are deployed

Another possibility that could be used by an expert user could be an application component that allows him/her to provide more complex queries such a queries expressed with the SPARQL query language [3]. SPARQL is a semantic data query format that can be used to retrieve data included in RDF format such as OWL. Such an application component could be automatically deployed for expert users with the help of our user adaptation process described in Chapter 7.

From that query a list of data is created. It contains all elements that will be visualized by the end-user with an adapted visualization metaphor. In order to perform the choice of the adapted metaphor, two types of parameters are needed, the semantics information about the data and the structural properties. The data semantics information can be directly extracted from our data model based on OWL. Regarding the structural properties of the dataset, functions can be registered in the "DataSelection" task in order to compute from the selected elements a list that contains their structural properties. Such a function takes as inputs the data elements from the query and

[3] https://www.w3.org/TR/rdf-sparql-query/

extracts from them a given structural property. For instance, built-in functions can be implemented in the task such as computing the number of elements, the file types to visualize (pictures, text, 3D model etc.) or the spatial distribution of the data. If needed, the developer can also register its own functions in order to extract other structural properties that he would require for the visualization metaphors that he implemented himself/herself. All the registered functions are triggered before the metaphor choice in order to compute all the needed properties. Therefore, to perform the visualization, the dataset created with the query is associated with a set D of semantics and structural properties:

$$D = \{\bigcup_{i=1}^{n}(x_{i,1}, ..x_{i,m}), (y_1, .., y_k)\}$$

Where n corresponds to the number of elements in the dataset created from the query, m corresponds to the number of semantic properties associated to a given element in the dataset, x_i one property of an element, k is the number of structural properties y extracted from the dataset.

When the data query has been given by the user, the dataset of elements to visualize has been created and the different semantic and structural properties have been computed by the "DataSelection" task, a new task is defined as needed: "DataVisualization". This task is compatible with multiple application components that implement visualization metaphors. These metaphors correspond to the different types that we have detailed. For each type, we can image multiple compatible concrete application components implemented with our model presented in Chapter 4. For instance, for each type we can cite the following possible metaphors:

- 3D Diagrams: data-cube[Bach et al., 2014], point-cloud (Figure 8.2b), pie chart, etc.
- Generic exploration: carousel (Figure 8.2a), coverflow Figure 8.8 , city [Panas et al., 2003], etc.
- Cartography: Mercator projection map (Figure 8.6), Winkel projection map, globe, etc.
- Content related: museum[Bonis et al., 2009], movie theater [Esnault et al., 2010], store [dos Santos and Osório, 2004], library, etc.
- Temporal: timeline (Figure8.7), perspective wall [Mackinlay et al., 1991], etc.

Before the instantiation of a concrete application component with the "DataVisualization" task, all scores that correspond to the compatible metaphors are updated according to the semantic and structural properties of the dataset. The score of each compatible metaphor depends on the values of its parameters. For instance, as parameter we can imagine the level of zoom to select in order to see each element in detail, the size of each displayed element, the type of scale used in a diagram (linear, logarithmic, etc.). These parameters depend on the semantic and structural properties of the dataset to visualize. For a metaphor of index j, for each parameter $p_{i,j}$ with i the index of the parameter, the developer has to create a function $f_{i,j}$ that can provide the impact of the different possible values of $p_{i,j}$ on the metaphor compatibility score according to the structural and semantic properties of the dataset D. The developer also has to give the continuous or discrete range of the parameter $p_{i,j}$ between two values a and b. It gives:

$$\begin{cases} p_{i,j} \in [a, b] \\ \\ f_{i,j}(p_{i,j}, D) \end{cases}$$

If needed, the developer could also access in these functions $f_{i,j}$ to the user properties included in our user model detailed in Chapter 7. As well, the properties of the currently used devices could also be exploited by these functions. For example a zoom level could be computed according to the size of the main display. In case of dependent parameters it could also be possible to define a function f that computes an impact score according to multiple parameters. These functions could be edited directly in the implementation of the metaphor or we could also imagine to edit them in separate files with the creation of a specific language. It could allow the developer to edit

these functions without recompiling the code. For example, the impact a zoom level z to display an element in detail (a picture or a 3D model) for a cartographic metaphor such as illustrated in Figure 8.6 could be expressed as:

$$\begin{cases} z \in [1,10] \ with \ z \in \mathbb{N} \\ \\ f_{z,j}(z,D) = (\frac{z}{10} \times \frac{nbElem}{nbElemMax}) + (\frac{Res/ResMax}{z/10}) \end{cases}$$

Where z is the zoom level, an integer between 1 and 10 that respectively correspond to a very close view and to a very far view. $nbElement$ is included in the parameter D and corresponds to the number of elements to display and $nbElemMax$ the maximum number of elements that the metaphor can display. Res corresponds to number of pixel of the current display. This value is normalized by a value "$ResMax$". The function makes a compromise between two facts symbolized by the two operands of the equation. First, as expressed in the left operand of the right side of the equation, a high zoom level is required if there are many elements to display. Second, as expressed in right operand of the right side of the equation, with a higher resolution the elements can be displayed in detailed with a smaller zoom level.

To continue, the compatibility score of a visualization metaphor is the addition of the impact score of each of its parameters. The value is divided by the number of parameters. The scores of the metaphors that do not correspond to the type required by the user are set to zero. Then, for the remaining visualization metaphors, the computation of the compatibility score Sa of a given metaphor of index j is an optimization problem. The goal is to maximize the compatibility score of each metaphor by tuning their different parameters and then to deploy the metaphor that obtains the better score. For a given application component that implements a visualization metaphor, the choice of the best parameters $(p_{1,j}, ..., p_{n,j})$ and the computation of its compatibility score Sa can be described as:

$$\begin{cases} p_{i,j} = \underset{p}{\text{argmax}} \ f_{i,j}(p,D) \\ \\ Sa = \frac{\sum_{i=1}^{n} \ f_{i,j}(p_{i,j},D)}{n} \end{cases}$$

When the score Sa of each visualization has been computed, we deploy the application component that corresponds to the metaphor that obtained the best score. To do that, the computed score Sa is used by the "Default" scoring module described in Chapter 4 or by the "UserAndDeveloper" scoring module described in Chapter 7 in order to also take into account the users preferences scores. When the best application component is deployed it uses the previously computed parameters $(p_{1,j}, ..., p_{n,j})$ to configure itself.

After this step, the visualization interface is created. In that case, the metaphor and the parameters have been chosen by the system and by the developer choices. Regarding our requirement **R6**, the end-user must also be able to configure the adaptation process. As for the AVE model [Wiza et al., 2003], we aim to provide the end user with the possibility to configure the created visualization interface. Our goal is to extend our meta-user interface to do so. First, as explained in Chapter 4, the meta-user interface can be used in order to replace an application component by another compatible one. This feature could be used in order to allow the user to replace the currently deployed visualization metaphor by another one. Second, the different parameters that have been optimized could be modified with the meta-user interface. For example, for a 3D diagram that arranges the data on multiple axis, we could imagine changing the association between axis and properties. The meta-user interface needs to be edited to do so.

As shown in Figure 8.4, in the last step of the data visualization process the user can perform a new query on the all dataset or can refine its previous query. Indeed, the user may want to visualize different data or if he/she may want to visualize a subset of the currently displayed elements. In the case that he/she refines the query, we must ask the end-user if he/she wants to keep the same visualization metaphor or if he/she wants the new metaphor to still be chosen automatically. Indeed, we consider that changing automatically the visualization metaphor to observe a subset of the currently displayed elements could disturb the end-user. This choice has to be possibly given to the user by the concrete application component associated with the "DataSelectionTask".

These different visualization metaphors can be used at the same time with the different interaction tasks and the different interaction techniques that we described previously. For instance, the selection techniques presented in Chapter 7 could be used in order to select particular elements and get additional information about them. In the same way, a navigation interaction technique could be deployed in order to navigate within the data.

8.5 The Topic-Topos Database Visualization Case Study

For demonstrating the interest of our model and its efficiency we propose a proof of concept based on the visualization of cultural heritage database.

8.5.1 The Topic-Topos database and one of its possible use cases

The Topic-Topos database proposes an inventory of French patrimonial data. It includes 25 000 patrimonial elements that are arranged according to four main criteria:

- the "Topic" corresponds to the textual description of an element,
- the "Topos" refers to the geographical area of an element,
- the "Khronos" points to the time of an element,
- the "Thema" is the general theme of an element. Four main themes are possible: society, economy, space and mentality. These main themes are then refined through a hierarchy of elements.

The database also includes a picture for approximately all elements. For instance, in Figure 8.5 two elements of the database are represented. The particularity of this database is that each element can be described with one or multiple keywords. The goal of these keywords it to add a semantic knowledge for all patrimonial elements. Almost 1000 different keywords are used to annotate the database elements. These keywords are organized as a tree where the highest nodes represent more general themes and the lowest ones represent more precise information. Moreover, the database can also be edited by any user who wants to share his knowledge about cultural heritage.

From this database we want to propose an interactive 3D visualization tool in a context of virtual tourism. A possible use-case of such an application could be to plan the different steps of a discovering trip. It could help the end-users to plan their trip by selecting the elements they are interested in and according to the position, the time and the themes of each of these elements. Moreover, with the possibility of our models to adapt an application to devices, this tool could be used with a mobile platform or with augmented reality glasses during the trip in order access the elements descriptions included in the database.

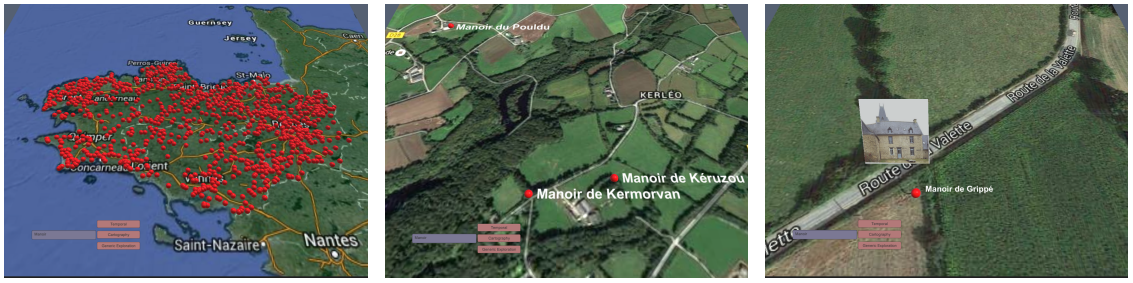
8.5.2 A first proof of concept for Topic-Topos database visualization

The Topic-Topos database is composed of two parts. First a SQL table includes all the elements and that contain fields such as "name", "keywords", "dating", "GPS position". This database is loaded through a generic SQL data loader in our data model. Second, an additional files provides a hierarchy of the keywords as a tree. These keywords represent the semantics concepts of the database. As explained in Section 8.3, this hierarchy is parsed by a specific converter in order to create classes and subclasses in the OWL ontology. Each element of the SQL database is defined as an individual of a class according to its associated keywords. For instance, an example of a class extracted from the database is "Housing" and "Manor", "House" and "Villa" are three of its subclasses.

Regarding the data visualization process, for the first step we have implemented an application component based on a text input field and on multiple virtual buttons in order to edit a query and give the metaphor type that he/she wants to be used. This graphical interface is shown in Figures 8.6, 8.7 and 8.8. In this three figures, the request is the same, the user wants to display all manors

FIGURE 8.5 – Two patrimonial elements taken from the Topic-Topos database. On the left the Benoist windmill in the town of Mont-Dol. On the right, a painting of church that can be seen in the town of Ploermel.

("Manoir" in french). For now, the query component only supports the concept level detailed in Section 8.4 and the union operator. In the same way, for now, only three global types can be selected: temporal, cartography and generic exploration. In the current implementation, only one visualization metaphor is compatible to each type. No optimization process is performed and the visualization metaphor that corresponds to the global type required by the user is automatically instantiated.



(a) (b) (c)

FIGURE 8.6 – The Mercator map visualization metaphor. The visualization of the elements changes according to the zoom level.

For the cartography type, we have implemented a Mercator map as illustrated in Figure 8.6. This map is based on the satellite images provided by Google and allows the user to zoom in or zoom out the map according to predefined zoom levels. The point of view can also be modified by the end-user. Different logical drivers have been implemented in order to control the map with different types of devices such as a mouse, a gesture sensor or a gamepad. As illustrated in Figure 8.6, the visualization of the elements change according to the zoom level. Indeed, as shown in Figure 8.6a, with a small zoom level, only spheres represent the different elements on the map. As illustrated in Figure 8.6b, when the user zoom in the map the names of the elements are also displayed. Last, as shown in Figure 8.6c, a picture is displayed for a high zoom level. For now, the zoom levels that trigger a change of how the elements are displayed are static and do not depend on the on the data properties. As future work, the parameters must be set with the optimization

process described in Section 8.4. In that case the data properties would be used to select the different zoom levels.

For the temporal type, we have implemented a timeline metaphor as illustrated in Figure 8.7. As for the map, the user can also move the timeline with different types of devices in order to see the elements of other dates. For this metaphor, the elements are simply placed on the timeline according to their date of creation included in the database. Both the name and the picture are displayed for each element. A possible parameter that could be optimized for this metaphor is the time resolution of time line. Indeed, this resolution could be chosen according to the time distribution of the elements to visualize.

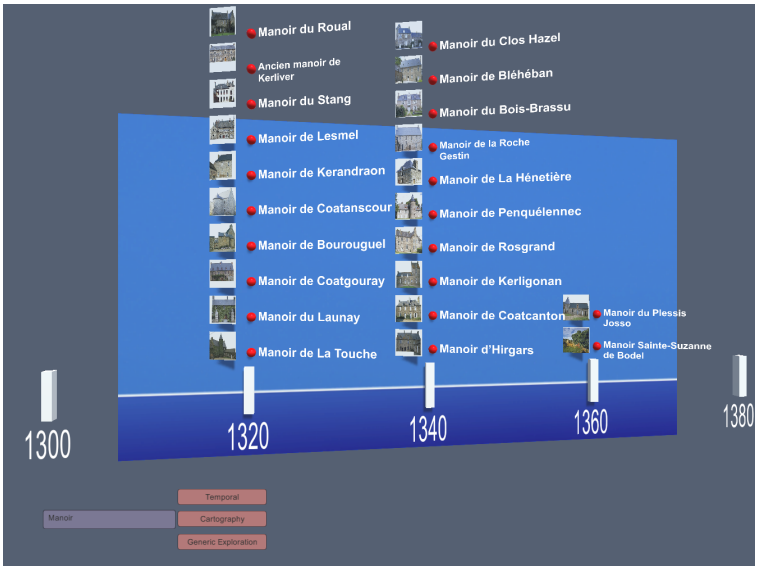


FIGURE 8.7 – The timeline visualization metaphor.

For the generic exploration type, we have implemented a coverflow visualization metaphor illustrated in Figure 8.8. This metaphor displays one central element with its picture and its name The user can scroll through the different items with different types of devices. For instance, a Leap Motion can be used in order to detect swipe gestures for changing the central element. A possible parameter that could be optimized for this metaphor is the scrolling speed. This speed could be chosen according to the number of elements to visualize. Indeed, in order to explore a very large dataset, an increased speed is needed.

As future evolution of these three visualization metaphors, we must integrate the possibility to select each individual element in order to display additional information such as the description of the elements included in the database. This could be performed with the different selection interaction techniques included in 3DPlasticToolkit and detailed in the previous Chapters.

With the possibilities to edit a data query and to visualize its results, this application demonstrates a first implementation of the basis of our model for taking data as an adaption source. This application allows any user to visualize the Topic-Topos cultural heritage database through different aspects in order to find interesting elements so that he/she can plan a discovering trip. The implementation of the model must be completed in order to evaluate its efficiency. This use case could be used for evaluating the model. Indeed, for instance, it could be interesting to evaluate the efficiency of our solution for planning a discovering trip by comparing our model with a classic 2D search engine.

8.6 Conclusion

In this Chapter we present a model for integrating data adaptation in our plasticity models. For now, this model is not fully implemented in 3DPlasticToolkit. Nevertheless, a first proof of concept illustrates the basis of this model and how it can be used to develop a cultural heritage visualization

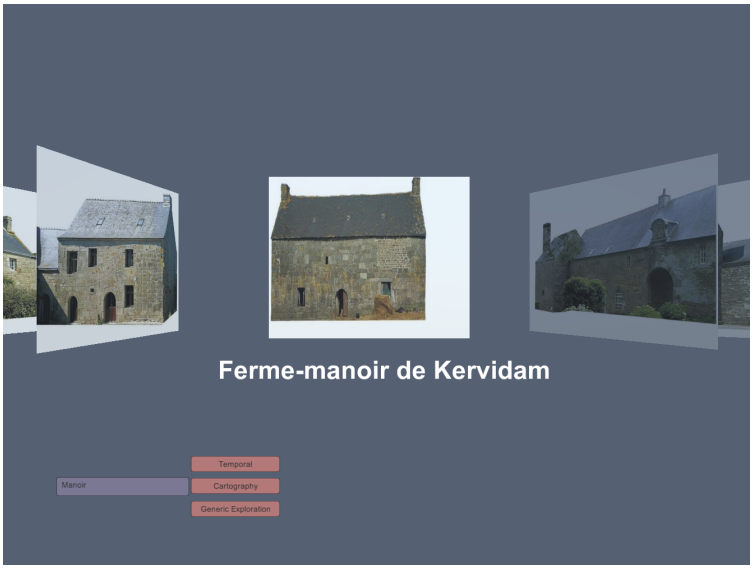


FIGURE 8.8 – Coverflow

application. When this model will be fully implemented in 3DPlasticToolkit we will be able to fully cover **R1**.

In this Chapter we first propose an integration of a data model in our solution. This model is based on OWL and aims at providing to the application the semantic and structural information about the manipulated data at runtime. OWL files can be seamlessly loaded with this model and we also ensure that any other type of databases can also be loaded. To do so, we propose a two steps conversion process that requires the intervention of the developer. Second, we propose a data visualization process composed of two steps. The first step allows the user to express a data query in order to select the data that he/she wants to visualize. Thanks to possibility offered by our models to handle multiple kinds of devices, this query can be edited through different ways such as directly with a keyboard or with the use of a speech recognizer. With this query, the user also provides a global type of metaphor that he/she wants to be used between five types: content-related metaphors, generic exploration metaphors, temporal metaphors, maps and 3D diagrams. In the second step, the semantic and structural properties of the selected data are used in order to choose the best visualization metaphor that corresponds to the global type given by the user. To do so, an optimization process is applied per compatible metaphor to select the best metaphor with the best parameters according to the data properties. For now, this data visualization process is not fully implemented in 3DPlasticToolkit. Nevertheless, we propose a first proof of concept of this model in a cultural heritage context that allows a user to edit queries with his/her keyboard and to visualize the results through three different visualization metaphors: a timeline, a Mercator map and a coverflow metaphor.

As perspective of work, of course we want to implement the whole model in 3DPlasticToolkit. Multiple visualization metaphors must be implemented in each category in order to provide end-users with a wide variety of visualization possibilities. A method to edit the impact score of each parameter must also be provided in order to perform the configuration of this part of the adaptation process without recompiling the code. As future work, it could also be interesting to illustrate the model in another context of use such as visualizing procedure data in a training process. Last, we plan to evaluate if the automatic choices of visualization metaphors always satisfy end-users. We want to see if the metaphors are always well chosen and if the frequent changes of metaphors disturb the end-user.

Chapitre 9

Conclusion and Perspectives

9.1 Summary

Our thesis focuses on plasticity for mixed reality user interfaces. It can concern virtual reality, augmented reality and augmented virtuality applications. Today, there is a growing interest for this domain and a lot of use cases can benefit from the use of MR. Nevertheless, developing such applications is more complex than developing classic 2D user interfaces.

The plasticity concept originally proposed in the context of 2D user interfaces is the capacity of an interactive system to withstand variations of both the system physical characteristics and the environment while preserving its usability. A plastic interactive system is able to detect the current context of use and is capable, if necessary, to change its appearance and behaviour in order to fit this current context. Considering plasticity during the creation of MR user interfaces can induce a lot of advantages such as the reduction of development and maintenances costs, the possibility to distribute widely an application, and the capacity to improve the attractiveness of an application according to the context of use.

The objective of this thesis is to identify the different requirements for the development of plastic MR user interfaces and to propose models, concepts and algorithms integrated in a software solution in order to cover these different requirements.

9.2 Results

In this context, we first propose a design space problem for the development of plastic MR user interfaces based on four axis that correspond to the adaptation time, the adaptation controller, the adaptation sources and the adaptation targets. At the center of this space problem, the two adaptations means are recasting and redistribution. This design space problem has been published in [Lacoche et al., 2014]. Based on this design space problem, we successively propose three models for dealing with the three identified adaptation sources: devices, users and data. By integrating these models into one software solution named 3DPlasticToolkit we can cover the different issues identified in the design space problem. A summary of our contributions according to this design space problem is illustrated in Figure 9.1.

For the creation of a MR user interface, one of the main issues is to consider devices as an adaptation source. Indeed, in MR the set of possible output and input devices is very large. Furthermore, this domain is very dynamic and new devices appear really often. Developing manually a version for each possible configuration is not a viable option. In order to cover this issue, we propose a device model for the description of interaction and display devices and an application component model for the development of MR applications components independently from devices and 3D frameworks. These models have been published in [Lacoche et al., 2015a]. From these two models, we support recasting with two elements. First, a dynamic adaption process that checks the modifications of the context of use in order to deploy the adapted application components at runtime. This process maximizes the usability of the application with scoring mechanisms that use scores given by the developer. Second, an integrated user interface allows the end-user to check and

to modify the adaptation behaviors at runtime: the meta-user interface. In order to demonstrate how these models can be used in concrete examples we present multiple applications developed with 3DPlasticToolkit that consider device as an adaptation source. For instance, we present a furniture planning application demonstrated at IEEE VR 2015 [Le Chenechal et al., 2015] and a collaborative manipulation demonstration presented for the IEEE 3UI contest 2016 [Le Chénéchal et al., 2016b]. A demonstration of how it can be used in order to deal with the "frame cancellation" visual discomfort has also been published in [Lacoche et al., 2015b].

To continue we also propose a model for allowing the end-user to change dynamically the distribution of his/her application across platforms/displays and user: this is redistribution. We propose D3PART (Dynamic 3D Plastic And Redistribuable Technology), a model specially designed to deal with redistribution for MR user interfaces. This model has been published in [Lacoche et al., 2016]. D3PART allows the user to configure a new distribution of the high level tasks with the meta-user interface and also handles the virtual environment replication. Combined with the dynamic recasting capability of our models, with the included adaptation process, usability continuity is ensured whatever the new distribution chosen. With D3PART any application developed with 3DPlasticToolkit automatically benefits from redistribution capabilities. We present three examples of redistribution for our furniture planning application. These examples show how redistribution can be used to switch from a mobile platform to an immersive one, to combine these two platforms, and finally to create a collaborative context of use between them.

From the different applications developed with our models that consider devices as an adaptation source, we have highlighted the need for also taking into account the users during the adaptation process: this is personalization. Indeed, the user preferences and properties have to be possibly taken into account during the choices of interaction techniques, visual effects, etc. In this thesis, we first try to understand the needs for taking into account the users preferences to adapt the interaction techniques and how we can detect automatically these preferences. To do so we present an experiment that compares different selection techniques on two device setups that only differ by their display type. We show that even with few differences between the two setups the users performances and the preferences are different. The results of this experiment are used in order to propose a machine learning approach based on SVMs in order to automatically detect the adapted selection technique for a new user. The results of this approach are really encouraging and more work needs to be done in order to demonstrate totally its efficiency. Then, based on these different results we propose a user model that can include properties about the user profile, the user environment, the user preferences and also user monitoring information. During the application components deployment these information can be taken into account in order to compute the different compatibility scores. The result of the integration of this user model and of the update of the scoring mechanisms is the possibility for 3DPlasticToolkit to dynamically adapt a MR user interface according to the user preferences.

The last focus of this thesis is on adaptation to data. Indeed, in most applications users manipulate, visualize and modify different kinds of data. The structure of these data and their semantic properties can really impact an application, mostly its content visualization. Therefore we propose a complementary model for taking into account data as an adaptation source. For now, this model is not totally implemented in 3DPlasticToolkit. Only preliminary proof of concepts have been developed. We first propose an integration of a data model which can load large databases of different types and can expose the information about data semantic through Web Ontology Language (OWL). Second we propose an update of the scoring mechanisms in order to choose an adapted visualization metaphor with adapted parameters to display the results of user data query. This mechanisms is based on an optimization function. An application component is also proposed in order to let the end-user expresses data queries. To illustrate this model we propose a proof of concept application for the visualization of cultural heritage data.

9.3 Perspectives

The different models, concepts, algorithms and software components presented in this thesis can be extended and also evaluated. Therefore, we have identified a list of future work that could be performed in order to complement the research contributions presented in this thesis.

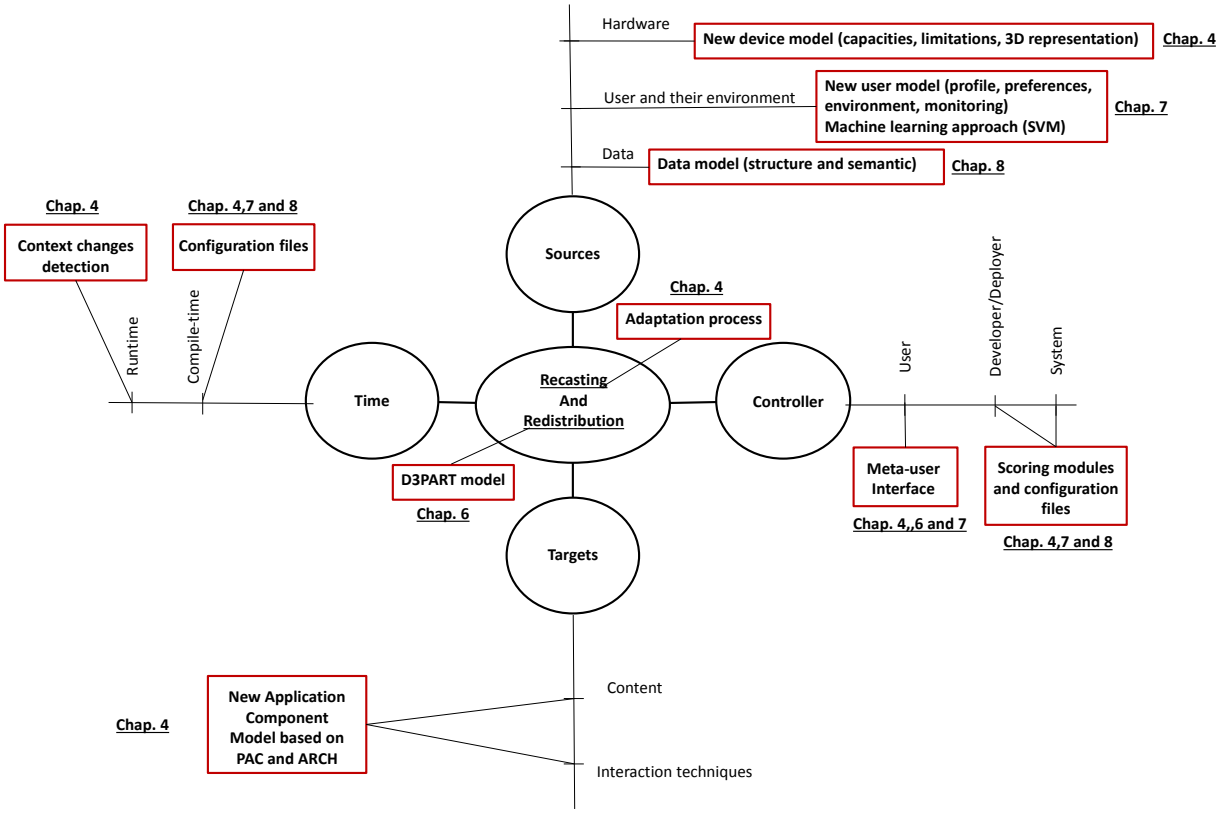


FIGURE 9.1 – A summary of our contributions for the creation of plastic MR user interfaces. We also give the associated Chapter for each of these contributions.

Improvement of association between logical drivers and device units

The accurate description of our device model aims to avoid not suitable alignments between devices capabilities and logical driver needs. In some cases this may not be enough to match the user needs. Indeed, the user may prefer to use other devices units for a given logical driver. Even if our meta-user interface provides a solution to modify these associations it can be tedious to do it each time the application is launched. A solution for future work could consist in using the interaction models proposed by Avouac et al. [Avouac et al., 2012] in order to pre-define the associations between some identified devices and one logical driver. To do so, an interaction model could be integrated in our user model. It would be included in the user preferences part of our mode. In order to update it, we could just save the modifications of associations that the user performs in the meta-user interface.

Considering system-intended redistribution

In D3PART presented in Chapter 6, for now the redistribution process is only user-initiated with the help of the meta-user interface. As perspective of work, we want to explore the automation of the redistribution process. Our goal is to obtain also system-initiated redistribution or mixed-initiated redistribution. For instance, it could consist in finding the right platform or the right user for each task according to the platforms capabilities and the user preferences. Our scoring system could be used to do so, for example by applying no longer locally, but instead by applying it on all available platforms. Another possibility to trigger automatically the redistribution process could be to analyze the user behaviour with the monitoring information of the user included in the user model presented in Chapter 7. For instance, in the use case presented in Section 6.5, if the user drops his/her tablet before entering in Immersia, we could initiate a full migration of the application from the mobile platform to the immersive one. Conversely, if the user enters in Immersia with the tablet, a partial migration could be intended in order to combine both platforms. The main issues here is user actions recognition that would require advanced computer vision algorithms such as

the work presented by Chéron et al. [Chéron et al., 2015].

Generalization of the machine learning approach for user preferences detection
In Chapter 7, we present a machine learning approach in order to detect the preferences of a user for a selection task. As detailed, the results of using such an approach are encouraging and we think that we larger a database of participants we would be able to precisely detect the preferences of a new user. Nevertheless, this approach also suffers from generalization. Indeed, it can only detect the adapted technique between three interaction technique for a selection task and for the two given setups. As future work, it could be possible to use machine learning in order to predict global characteristics from the user profile or from a preliminary test and then match these characteristics with the properties of some interaction techniques. An example that we have given in Chapter 7 could be to detect automatically a novice user with low skill and propose him/her automatically 3D menus with bigger icons and an assistive selection technique such as the 3D Bentray. The main difficulty of this perspective of work is to construct an experiment that would let us extract high level characteristics of users that can be matched with a wide variety of interaction techniques.

Considering LOD
On of the main bottlenecks for MR user interfaces concern the times to render complex 3D models. As keeping a low latency is essential for MR user interfaces, a particular attention has to be given to impact of the 3D models on performances. As each platform may not have all the same computation capabilities, the rendering of the assets could be handled differently. In some cases it would be necessary to consider adaptive assets (with level of details). To solve this issue, we plan first to integrate information about the computing power of the platforms into the device model described in Chapter 4. Second, we plan to give the possibility to parametrize the choice of an asset according to the computation capabilities of the target platform. This parametrization could be directly integrated in our data model detailed in Chapter 8. Then the adaptation process would be able to dynamically choose the adapted assets according to the platform capabilities. In the same way, in the redistribution process, during the virtual environment replication, only the assets that correspond to the distant platform computation capabilities would be transmitted.

3DPlasticToolkit completion
In order to be used by other developers 3DPlasticToolkit has to be completed in order to really bring an added value for the development of MR user interfaces. First the number of built-in high level tasks and built application components needs to be increased. Indeed, more interaction techniques need to be integrated as well as more visual effects and more data visualization metaphors. To simplify the creation of these components, as proposed in Chapter 4, a MDE approach could be used for the creation of the different classes and to automatically connect their different functions. Then, more devices must be integrated in 3DPlasticToolkit. Indeed, as demonstrated in this thesis with our different examples it already supports common devices such as HMDs, trackers, stereoscopic screen, depth sensors, etc. Nevertheless, our device model can also describe other kinds of devices that are less common such as haptic devices, wind output devices or temperature output devices. For now, our software solution does not include natively such devices. To continue, for now, our solution mainly focuses on helping developers for the creation of plastic MR user interfaces. Our goal is to also make it available for designers. Some authoring tools have been developed but they are not ready to use yet by designers. More work has to be done on these authoring tools if we want to fully cover **R7** and make our solution easy to use for designers.

3DPlasticToolkit validation
Once our software solution completed it has to be evaluated in order to investigate its efficiency. For developers and designers two aspects must be verified. First, we must verify if our solution is easy to use and efficient for the development of a new MR applications. To do so, feedbacks from developers and designers have to be collected about how easy are the configuration of a device, the configuration of the tasks and the use of the different authoring tools. Second, we must also investigate if the solution is easy to extend. For instance, we have to evaluate the capacity of developers to integrate new devices, to create new interaction techniques or to develop new scoring modules. We must also evaluate the impact of using 3DPlasticToolkit on the end-user. Indeed, multiple questions need to be answered regarding the integration of our plasticity mechanisms in a MR user interfaces. For instance, how do they react when the application is automatically adapted? Are the meta-user interface and the redistribution process are easy to understand and to use for the end-user? Do they always feel that usability continuity is ensured?

9.4 Conclusion

To conclude, in this thesis we propose different models integrated in 3DPlasticToolkit in order to create plastic MR user interfaces. Our contributions make it possible to take into account the users, the data and the devices as adaptations sources. These three adaptation sources can be taken into account in order to adapt the interaction techniques and the content visualization. The adaptations can be static and dynamic and controlled by the system, the end-user and the developer. Two adaptations means are supported: recasting and redistribution. We can say that our solution covers the different requirements of the design space problem of plasticity for MR user interfaces. A summary of our contributions according to this design space problem is illustrated in Figure 9.1.

3DPlasticToolkit already integrates a lot of display and interaction devices. As well it also proposes built-in interaction techniques, visualization metaphors and visual effects. Some authoring tools are also provided to configure the toolkit. Therefore, 3DPlasticToolkit is ready-to-use for any developer who wants to develop a MR user interface and who wants to benefit from the advantages of Plasticity. The different applications detailed in this thesis illustrate its efficiency. Furthermore, other applications based on 3DPlasticToolkit are still in development.

The proposition of these models offers a lot of perspectives of work in order to enhance, validate and complete them.

Publications

Refereed Conference Papers

- *Dealing with frame cancellation for stereoscopic displays in 3D user interfaces*, Jérémy Lacoche, Morgan Le Chénéchal, Sébastien Chalmé, Jérôme Royan, Thierry Duval, Valérie Gouranton, Eric Maisel, Bruno Arnaldi, in Proceedings of IEEE Symposium on 3D User Interfaces (3DUI), p73-80, Arles, France, March 2015.

- *Plasticity for 3D user interfaces: new models for devices and interaction techniques*, Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel, Jérôme Royan, in Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS), p28-33, Duisburg, Germany, June 2015.

- *D3PART: A new Model for Redistribution and Plasticity of 3D User Interfaces*, Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel, Jérôme Royan, in Proceedings of IEEE Symposium on 3D User Interfaces (3DUI), p23-26, Arles, France, March 2016.

Workshop Papers

- *A survey of plasticity in 3D user interfaces*, Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel, Jérôme Royan, in Proceedings of the IEEE Interaction Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), p19-26, Minneapolis, USA, March 2014.

- *When the Giant meets the Ant An Asymmetric Approach for Collaborative and Concurrent Object Manipulation in a Multi-Scale Environment*, Morgan Le Chénéchal Jérémy Lacoche, Jérôme Royan, Thierry Duval, Valérie Gouranton, Bruno Arnaldi, in Proceedings of IEEE International Workshop on Collaborative Virtual Environments (3DCVE), GreenVille, USA, March 2016.

Demonstration Papers

- *Laying out Spaces with Virtual Reality*, Morgan Le Chénéchal Jérémy Lacoche, Cyndie Martin, in Proceedings of IEEE Symposium on Virtual Reality (VR), p337-338, Arles, France, March 2015.

- *When the Giant meets the Ant An Asymmetric Approach for Collaborative Object Manipulation*, Morgan Le Chénéchal Jérémy Lacoche, Jérôme Royan, Thierry Duval, Valérie Gouranton, Bruno Arnaldi, Contest Demonstration at IEEE Symposium on 3D User Interfaces (3DUI), GreenVille, USA, March 2016.

Submitted Journal Papers

- *3DPlasticToolkit: Plasticity for 3D User Interfaces*, Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel, Jérôme Royan, in IEEE Transactions on Visualization and Computer Graphics (TVCG).
- *Providing Plasticity and Redistribution for 3D User Interfaces using the D3PART Model*, Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel, Jérôme Royan, in Journal on Multimodal User Interfaces (JMUI).

Informal National Communications

- *Redistribution et Plasticité pour les Interfaces Utilisateurs 3D: un Modèle Illustré*, Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Eric Maisel, Jérôme Royan Présentation à la $10^{\text{ème}}$ édition des journées de l'AFRV, Bordeaux, France, Octobre 2015.

Appendix A: Frame Cancellation Effects Evaluation

In this Appendix, we present the evaluation of the different effects presented in Section 5.3 for dealing with Frame Cancellation. This part is not included in the thesis as they are not correlated directly with plasticity.

In order to verify the efficiency of our two methods, we have evaluated both of them. The evaluation had two main objectives: first, to compare and evaluate our two visual effects PSCVC and VABW with SCVC [Ardouin et al., 2011], which seems to be the best state of the art approach. All effects have been compared to a simple rendering in order to see if displaying a treatment is really necessary to deal with frame cancellation. Then, we evaluated the avoidance of frame cancellation with collision detection in a manipulation task. To do so, we compared the technique with a non-constrained one in a basic scenario. The two experiments have been made by the same group of participants.

Participants

Our experimental group consisted of 21 subjects aged from 20 to 48 (age: M=34, SD=8). There were 14 males and 7 females. They had little knowledge about rendering techniques and had very limited experience with stereoscopy. The subjects have various backgrounds: PhD students, R&D engineers, managers and assistants.

The Simulator Sickness Questionnaire

We applied a Simulator Sickness Questionnaire (SSQ) at the beginning and end of the experiment [Kennedy et al., 1993]. The SSQ contains 16 physical symptoms rated on a categorical labeled scale (none, slight, moderate, severe). A factor analysis revealed that these symptoms can be placed into three general categories: oculomotor, disorientation, and nausea [Kennedy et al., 1993]. Weights are assigned to each of the categories and summed together to obtain a single score. The SSQ mean score was greater at the end than at the beginning of the experiment (start: M=5.7, SD=1.2; end: M=6.4, SD=1.3) mainly due to oculomotor symptoms. This increase was not significant. As the SSQ scores were low, we assumed that our VE did not produce strong cybersickness symptoms.

First step: Rendering techniques evaluation

For this first part of the evaluation, the application showed to the user different spheres of 15cm in diameter. The application background was blue and the spheres color was red, as shown in Figure 5.5. These colors were chosen in order to reduce the ghosting effect for the target display screen. Each sphere was different from the others only by the effect applied close to the screen edges. For the evaluation, four effects were compared, SCVC [Ardouin et al., 2011], PCVC, VABW and a last effect, which applied no specific treatment. During the evaluation, these spheres moved on a plane that was one meter in front of the screen and parallel to it. This distance was chosen in order to be included in the comfortable viewing zone for the current display setup with respect to the recommendations given by Chen et al. [Chen et al., 2011]. The sphere velocity ($0.2\ m/s$) was chosen according to subjective pre-test results. The user was positioned seated with his head aligned with

the screen center at about 2.80m distance to the screen. No head tracking was performed in order to modify the camera parameters; therefore, eyes were considered as X-aligned.

For the hardware configuration, we used a FaceLAB 5 eye-tracker system developed by Seeing Machines [1]. This system allowed us to track head and eyes in real time. This data was not analyzed in real-time, but collected for analysis after the evaluation. It operated non-intrusively, with no head-mounted device. This was advantageous as the subjects had to wear polarized 3D glasses and sometimes their corrective glasses. The display screen was a Planar LCD, 2.08m diagonal, 4K with passive stereoscopy. Moreover, an Xbox wireless controller was used to interact with the application.

For the **procedure**, the experiment started with a questionnaire and a Miles test [Miles, 1930] to evaluate dominant eye if unknown. In this test the observer creates a small opening with his hands. With both eyes, he looks at a distant object through this opening. The observer then alternates closing the eyes. The dominant eye is the one viewing the object. After this test, we calibrated the eye-tracker in precision mode for each subject.

Afterwards, for each side, the subjects saw each effect one after the other. Then, the subjects saw each effect for another side, and so on, one side after the other. They only had to visually follow the 3D sphere that moved about 1.80m in front of their face.

We applied a counterbalancing design with two between-groups variables: order of the side and order of the effects. *Order of the sides:* counterbalancing the order of the four sides leads to 24 orders. Each subject was assigned to a different order. For example [Down, Right, Down, Left] was assigned to subject 2. The participants did not know the order. *Order of the effects:* we called "series" an order of four effects. Counterbalancing the order of the four effects leads to 24 series. These series were assigned to the subjects. For example, the first series of subject 2 was [SCVC, NT, PSCVC, VABW], the forth series of this participant was [VABW, PSCVC, NT, SCVC], and the first series of subject 16 was [NT, PSCVC, VABW, SCVC]. The participants did not know the series assigned.

After each series, each subject was requested to fill out a subjective questionnaire to evaluate the effects. To do so, the subject remained free to see the effects again, once again one by one. The subject used the Xbox wireless controller. Each of the four buttons A, B, X, and Y of the controller allowed the subject to switch between the four effects. A corresponded to No Treatment (NT), B to Stereo Compatible Volume Clipping (SCVC), X to Progressive SCVC (PSCVC), and Y to the Virtual Alpha Blended Window (VABW). The name-effect relations were identical during all of the experiment and for all subjects. The participants were just told about the effects' letters and did not know their names and specificities.

Regarding the **data collection**, in the subjective questionnaire, participants had to grade the four effects using a Likert-scale, from 1 (very low appreciation) to 7 (very high appreciation) according to five subjective criteria. They were defined by Ardouin et al. ([Ardouin et al., 2011]): (a) Global appreciation, (b) Aesthetic, (c) Eye strain perceived, (d) Relief quality not at borders, and (e) Relief quality at borders. The participants could choose NA (no answer). To facilitate the comparisons, we calculated the score of eyestrain such that the lower the eyestrain the higher the score. We called this new variable; "comfort according to eye strain perceived". To compare the five criteria, we calculated a weighted average for each subjective criterion. As some participants choose NA, we divided each weighted average by its theoretical maximum score (7 points multiplied by number of numeric answers).

Although the FaceLAB system tracked many head and gaze variables, for the purposes of this paper, we analyzed the gaze fixation point. The fixation point was computed automatically by FaceLAB as the intersection between the unified gaze ray and the work plane (the planar screen), defined relative to the screen coordinate system. This coordinate system was defined by the screen bottom left corner: point (0,0) and the top right one: point (1,1). We analyzed the relative distances ($D_x = G_x - O_x$, $D_y = G_y - O_y$) between the gaze fixation point (G_x, G_y) and the 3D object positions (O_x, O_y) in the same coordinates system. If the absolute value of this relative distance was up to a fixed limit (equal to 0.2), we considered that the subjects deflected attention away from the object. We also measured the ratio of deflected fixations which is the number of deflected fixations divided by the total number of fixations (deflected and non-deflected).

[1] http://www.seeingmachines.com/

Our **hypothesis** was that our two visual effects, PSCVC and VABW, should give better results than the two other approaches, NT and SCVC effects:

- H1: Regarding Likert subjective questionnaire, we expect improved qualitative appreciations for the methods PSCVC and VABW. In particular, the global appreciation would be greater, and eyestrain would be lower.

- H2: The subjects could track the moving 3D objects close to the screen edges in a more comfortable manner with the methods PSCVC and VABW than the others. In others words, the subjects would be able to follow the 3D objects longuer at he screen edges with the methods PSCVC and VABW than the other ones. Regarding specific variable, the ratio of deflected fixations just at the screen edges would be less for PSCVC and VABW effects than for NT and SCVC effects.

Regarding the **subjective results**, Figure 9.2 shows the results concerning the grades (Likert-scale) obtained by the four different effects for each of the subjective criteria. The results for right and left sides were not significantly different as well as for top and bottom sides. Thus, only two graphics illustrate results.

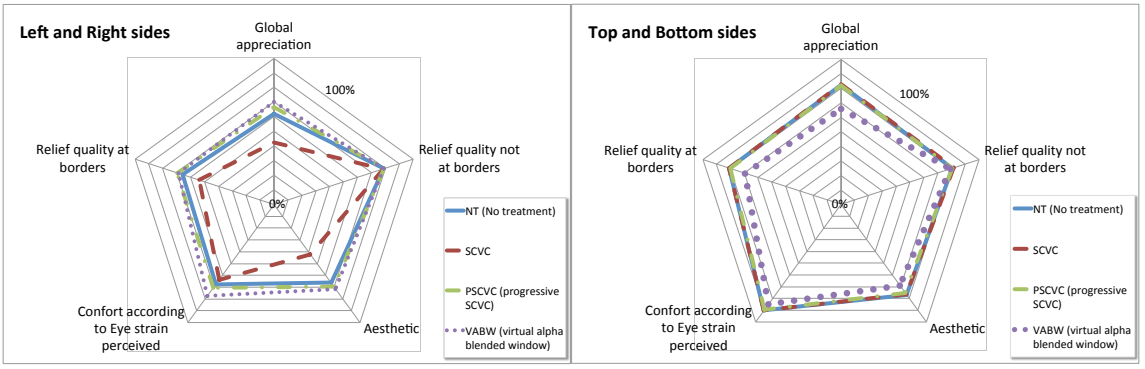


FIGURE 9.2 – Results for the subjective questionnaire for the four different techniques (1) Control (blue), (2) SCVC (red), and our two new methods (3) PSCVC (grey), and (4) VABW (purple) with respect to Likert-scale grading

The analysis of the subjects' responses gave two main results:

- For left and right sides: SCVC was the worst effect, VABW the best one,

- For up and bottom sides: NT, SCVC and PSCVC are equivalent and VABW was the worst effect.

For left and right sides, as we can see in Table 9.1, significant differences between the effects were found. SCVC was the worst effect for all the participants. SCVC was globally the least appreciated, the worst aesthetic effect, and presented the poorest relief quality at borders. A significant difference between the effects SCVC and VABW was found: SCVC provided stronger eyestrain than VABW. For top and bottom sides, few significant differences between the effects were found; in particular, VABW was globally less appreciated, as shown in Table 9.1. Regarding the order of preference, for each side, the participants were asked to order the effects from first to last place. Different effects could be equally placed. We analyzed the ranking order, in particular the frequency of effects chosen in the first two places. Significant difference between effects in the first two places was found ($\chi^2 = 18, df = 9, p < .05$). In comparison to the empirical probability distribution applied in the χ^2 test, SCVC was less preferred for right and left sides and PSCVC and VABW were more preferred for these two sides.

We calculated, for each side, the percent of times the participants ordered each effect in the first two places as shown in Figure 9.3.

χ2 Test		Right and Left sides			Top and Bottom sides		
Df = 2; p value (χ2 value) NS= No Significative difference		SCVC	PSCVC	VABW	SCVC	PSCVC	VABW
Global appreciation	NT	p<.001 (16,22)	NS	NS	NS	NS	p<.001 (21,52)
	SCVC		p<.001 (20,91)	p<.001 (25,54)		NS	p<.001 (23,81)
	PSCVC			NS			p<.001 (21,52)
Aesthetisc	NT	p<.001 (21,2)	NS	NS	NS	NS	NS
	SCVC		p<.001 (25,55)	p<.001 (33,59)		NS	NS
	PSCVC			NS			NS
Eye strain perceived	NT	NS	NS	NS	NS	NS	NS
	SCVC		NS	p<.05 (8,5)		NS	NS
	PSCVC			NS			NS
Relief quality at borders	NT	p<.05 (7,6)	NS	NS	NS	NS	p<.01 (13,62)
	SCVC		p<.01 (9,66)	p<.01 (10,72)		NS	p<.01 (13,62)
	PSCVC			NS			p<.01 (11,19)
Relief quality Not at borders	All effects	NS			NS		

TABLE 9.1 – Statistical results (χ^2 test) to compare the four methods

For right and left sides, PSCVC and VABW were the most appreciated effects. For top and bottom sides, NT and SCVC were the most appreciated. We observed a gap between PSCVC, SCVC and NT for top and bottom but this gap was not significant. Indeed, the virtual object aspect was the same for the three effects on these sides.

To continue, concerning **the eye-tracking results**, we compared, for each side, the ratio of deflected fixations. We used R software and applied a mixed model to analyze this ratio just at the screen edges. Regarding our hypothesis, the ratio of deflected fixations just at the screen edges would be less for the PSCVC and VABW effects than for NT and SCVC effects.

For the right side, we obtained a significant difference between effects ($F(3, 60) = 3.24, p < .05$). The ratio of deflected fixations was smaller with VABW than with the other effects. For left, top and bottom sides, we did not obtain any significant difference between methods.

For top and bottom sides, the result was not surprising. Indeed, as users' head rotation was not taken into account in the stereoscopic camera model, they did not perceive vertical parallax. Thus NT, SCVC and PSCVC effects were visually similar for top and bottom sides. The result for left and right sides was more surprising. Considering the subjective appreciations, we hoped for a larger difference between the effects. To understand the reason for these results, we analyzed the effect of the dominant eye.

Significant difference between dominant eyes was found at the screen edges ($F(1, 312) = 14.344, p < .001$). The ratio of deflected fixations was higher with left-eye dominant participants (M=0.50, SD=0.08) than with right-eye dominant (M=0.25, SD=0.06). The difference concerning each side is given in Figure 9.4. As a bit less than 2/3 of the participants (16) were right-eye dominant and 1/3 left-eye dominant (5), the effect of the dominant eye cannot be more discussed.

To **sum up** these results: our hypotheses were not all verified.

For the first hypothesis **H1**: PSCVC and VABW improve qualitative appreciations. We found that participants clearly preferred PSCVC and VABW for right and left sides. In particular for global appreciation. For top and bottom sides, we found that the participants preferred NT and SCVC, but differences between the four effects were low. As shown in Figure 9.2, PSCVC and VABW differ significantly from NT and SCVC for right and left sides. For top and bottom sides, in Figure 9.2 we also observe a little difference between VABW and the three other effects. These two observations can be explained by binocular rivalry only present at the vertical edges in our experiment. Indeed, as our two effects were preferred on the vertical edges and no treatment was preferred on the horizontal ones, we can deduce that in order to deal with frame cancellation,

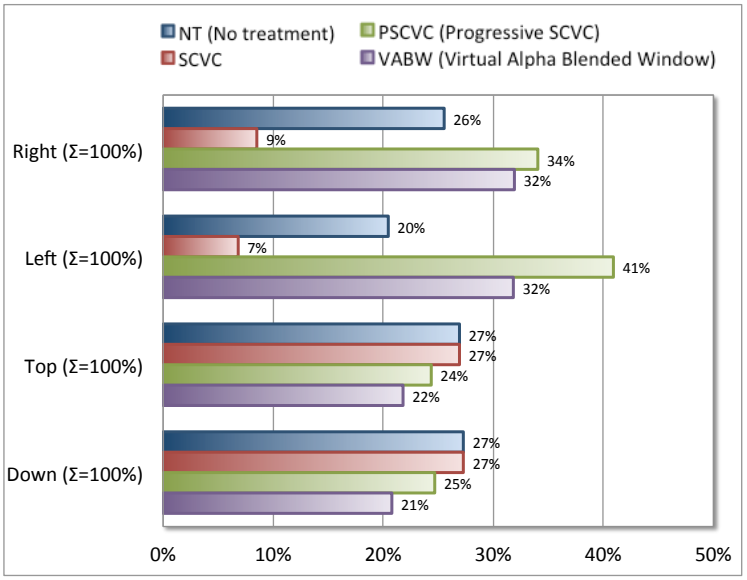


FIGURE 9.3 – For each side, percent of times the participants ordered each effect in the first two places

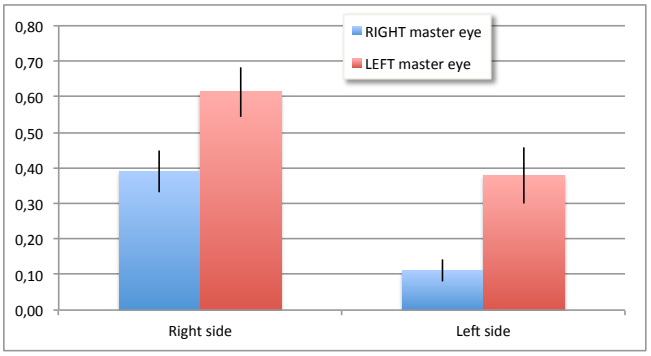


FIGURE 9.4 – Ratio of deflected fixations

binocular rivalry has to be especially considered. This consideration was done by the two effects we introduced and by the SCVC method. Regarding the results, the smooth aspect of our solutions seems to be preferred over the rough one proposed by SCVC. It is difficult to explain why NT was preferred over SCVC on the vertical edges because it contradicts the results obtained by Ardouin et al. [Ardouin et al., 2011]. We assume that the too rough clipping created a more disturbing effect than binocular rivalry and frame cancellation. However, this result could also be explained by the difference between our experimental setup and the one used by Ardouin et al. [Ardouin et al., 2011]. More work could be done in that way to compare the impact of different display setups and different VEs.

For the second hypothesis **H2**: PSCVC and VABW improve 3D object tracking at the screen edges. The number of deflected fixations just at the screen edges for right side was smaller for VABW. Thus, the participants could track the moving 3D objects close the screen's right edge in a more comfortable manner. We found that VABW was better than the other effects for right side. But, we did not obtain other significant results without taking into account the dominant eye. Indeed, the dominant eye played an important role in the deflecting movement. Left-eye dominant participants showed more deflected fixations than the others. We did not control eye dominance as an independent variable in the experiment, but our initial results suggest effects regarding the physiological reactions to 3D moving objects at the screen edges, which warrant future studies in this direction.

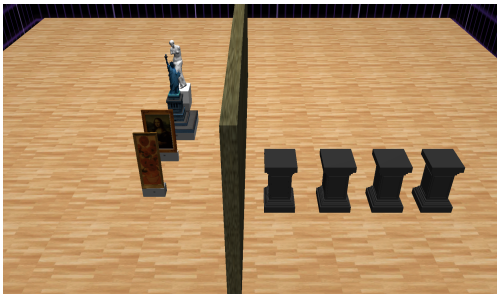


FIGURE 9.5 – The four masterpieces separated from the pedestals by a wall

Second step: Interaction technique evaluation

In the second experiment, the application displayed four artworks, four pedestals and a wall as shown in Figure 9.5. The goal was to put each artwork on its corresponding pedestal with a 3D-ray based manipulation technique. This technique is implemented as described by Bowman and Hodges [Bowman and Hodges, 1997] where a light ray extends from the user's hand detected by a 6-DoF tracker. When the ray intersects a scene object and the user presses a selection button, this object is attached to the ray extremity. The pedestals were aligned horizontally. Each artwork was identified by a number at its base. To find the corresponding pedestal of each one, the user had to sort them in ascending order. The lowest number corresponded to the leftmost pedestal, the highest to the rightmost one. In order to make sure that situations with frame cancellation occurred, we designed the experiment such that users had to bring the objects close to themselves during the trials. First, the wall was placed between the pedestal and the initial position of the artwork. It was a physical barrier that forced the user to move the objects in front of it to put them in the pedestals (the height of the wall was virtually infinite but not entirely represented in order to avoid an initial frame cancellation situation). Its length allowed us to control the minimal popping-out distance. Second, the numbers written on the bases of the artworks had two digits and were so small that they needed to be really close to the user to be well read.

As **hardware** for this experiment we used the zSpace interactive system which is composed of a passive stereo display (24-inches) with head tracking for co-location, and a stylus which provided a 6-Dof tracking and buttons that were used for object grabbing. Moreover, we added a numeric keypad to this system. At any time, a subject could use a key of this device to reset the position of the virtual objects, for example, when an object got lost. We did not use a stylus key for this feature in order to avoid accidental actions for novice users.

For the **procedure**, the experiment started with a training phase. For five minutes, the user trained himself how to move the virtual objects with the 3D-ray and how to place them on the different pedestals. Once this step was over, the participant performed two trials to compare our approach called Conflict-Free (CF) mode with the non-constrained one called Liberty (L) mode. We applied a counterbalanced design: 2 *modes* $\times$ 2 *series*. *Modes:* Half of the participants began with CF, half with L. The users were not told about the names and the characteristics of the two modes. *Series:* we defined two series of numbers on the artworks. Each subject was assigned to one of the four conditions. For example, the first subject followed L mode with series 1 then CF mode with series 2. After the two trials, each subject was requested to express their subjective impression in terms of usability and visual comfort.

Regarding the **data collection**, we collected the following data: number of manipulations, time to success, time and number of manipulations to put the first object in the pedestals, time and number of manipulations to put the fourth object in the pedestals (the first try).

Since CF constraints the object motion to keep it in a conflict-free area, we made two **hypothesis**:

- H1: Conversely to L mode, the participants in CF mode could not up-scale an object too much in front of them and could not generate frame-cancellation situations. We hoped that users would express more visual comfort with CF mode.

- H2: The participants in CF mode could not lose a part of an object culled by the cameras frustums so the performance should be better.

Regarding the **results**, only one significant difference between the two modes was found: the number of manipulations to put the fourth object in the pedestals ($F(1,40) = 42, p < 0.05$). The participants in CF made fewer manipulations (M=10.3, SD=2.1) than in L mode (M=12.1, SD=3.9). Thus, participants could complete their task in the same manner in the two modes. The L mode did not constrain the interaction of the participants. In contrast, the CF mode optimized the number of manipulations until the first try.

Even if the difference of number of manipulations to put the fourth object in the pedestals was significant, the participants did not express it often in the questionnaire at the end of the experiment. 2/3 of the participants (14) did not notice any functional differences between the two modes. In the 1/3 of the participants (7), a few did not appreciate the co-located manipulation technique without constraint because the objects did pop out too quickly: "*The object becomes too big too fast*". In contrast, a few participants did not appreciate the CF mode for the inverse reason; they were not able to pop out the virtual objects enough: "*It is less easy to pull the object in this mode*".

To **sump up**, the performances have improved a little with this mode (H2 verified). We assume that the CF mode helped and guided the users to complete the manipulation task. Future work may focus on comparing the two modes in another scenario in order to confirm the results on the interaction performances. Regarding H1, the participants did not express more visual comfort in CF mode in the qualitative results (H1 not verified). These results might be explained by the short duration (5-10 minutes) of the experiment. We assume that we would have found clearer results with a longer duration. Indeed, with a longer a duration, the participants would have generate more frame cancellation situations and observe visual comfort differences. More work should be done to evaluate the impact of the duration on the visual comfort with the two modes.

Bibliography

[Achibet et al., 2015] Achibet, M., Girard, A., Talvas, A., Marchal, M., and Lécuyer, A. (2015). Elastic-arm: Human-scale passive haptic feedback for augmenting interaction and perception in virtual environments. In *Virtual Reality (VR), 2015 IEEE*, pages 63–68. IEEE.

[Agrawala et al., 1997] Agrawala, M., Beers, A. C., McDowall, I., Fröhlich, B., Bolas, M., and Hanrahan, P. (1997). The two-user responsive workbench: support for collaboration through individual views of a shared space. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 327–332. ACM Press/Addison-Wesley Publishing Co.

[Aguerreche et al., 2009] Aguerreche, L., Duval, T., and Lécuyer, A. (2009). Short paper: 3-hand manipulation of virtual objects. In *JVRC 2009*, pages 4–p.

[Alliez and Gotsman, 2005] Alliez, P. and Gotsman, C. (2005). Recent advances in compression of 3d meshes. In *Advances in multiresolution for geometric modelling*, pages 3–26. Springer.

[Arch, 1992] Arch, A. (1992). A metamodel for the runtime architecture of an interactive system: The uims tool developers workshop. *SIGCHI Bull.*, 24(1):32–37.

[Ardouin et al., 2011] Ardouin, J., Lécuyer, A., Marchal, M., and Marchand, E. (2011). Design and Evaluation of Methods to Prevent Frame Cancellation in Real-Time Stereoscopic Rendering. In *IEEE Symposium on 3D User Interfaces, 3DUI 2011*, pages 95–98, Singapore, Singapore.

[Autodesk, 2008] Autodesk (2008). *Stereoscopic filmmaking whitepaper, The Business and Technology of Stereoscopic Filmmaking.*

[Avouac et al., 2012] Avouac, P.-A., Lalanda, P., and Nigay, L. (2012). Autonomic management of multimodal interaction: DynaMo in action. In *EICS 2012*, pages 35–44, Copenhagen, Denmark. ACM New York, NY, USA.

[Azuma, 1997] Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385.

[Bach et al., 2014] Bach, B., Pietriga, E., and Fekete, J.-D. (2014). Visualizing dynamic networks with matrix cubes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 877–886. ACM.

[Badam and Elmqvist, 2014] Badam, S. K. and Elmqvist, N. (2014). Polychrome: A cross-device framework for collaborative web visualization. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, pages 109–118. ACM.

[Bandelloni and Paternò, 2004] Bandelloni, R. and Paternò, F. (2004). Migratory user interfaces able to adapt to various interaction platforms. *International journal of human-computer studies*, 60(5):621–639.

[Bechhofer, 2009] Bechhofer, S. (2009). Owl: Web ontology language. In *Encyclopedia of Database Systems*, pages 2008–2009. Springer.

[Behr et al., 2009] Behr, J., Eschler, P., Jung, Y., and Zöllner, M. (2009). X3dom: a dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, pages 127–135. ACM.

[Bierbaum et al., 2005] Bierbaum, A., Hartling, P., Morillo, P., and Cruz-Neira, C. (2005). Implementing immersive clustering with vr juggler. In *Computational Science and Its Applications–ICCSA 2005*, pages 1119–1128. Springer.

[Bilasco et al., 2007] Bilasco, I. M., Villanova-Oliver, M., Gensel, J., and Martin, H. (2007). Semantic-based rules for 3d scene adaptation. In *Proceedings of the twelfth international conference on 3D web technology*, page 97–100.

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

[Bonis et al., 2009] Bonis, B., Stamos, J., Vosinakis, S., Andreou, I., and Panayiotopoulos, T. (2009). A platform for virtual museums with personalized content. *Multimedia tools and applications*, 42(2):139–159.

[Bosca et al., 2007] Bosca, A., Bonino, D., Comerio, M., Grega, S., and Corno, F. (2007). A reusable 3d visualization component for the semantic web. In *Proceedings of the twelfth international conference on 3D web technology*, pages 89–96. ACM.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

[Bowman and Hodges, 1997] Bowman, D. A. and Hodges, L. F. (1997). An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, pages 35–ff., New York, NY, USA. ACM.

[Bowman et al., 2004] Bowman, D. A., Kruijff, E., LaViola, J. J., and Poupyrev, I. (2004). *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[Bowman and McMahan, 2007] Bowman, D. A. and McMahan, R. P. (2007). Virtual reality: how much immersion is enough? *Computer*, 40(7):36–43.

[Brooks Jr et al., 1990] Brooks Jr, F. P., Ouh-Young, M., Batter, J. J., and Jerome Kilpatrick, P. (1990). Project gropehaptic displays for scientific visualization. In *ACM SIGGraph computer graphics*, volume 24, pages 177–185. ACM.

[Brusilovsky, 1996] Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User modeling and user-adapted interaction*, 6(2-3):87–129.

[Brusilovsky, 2001] Brusilovsky, P. (2001). Adaptive hypermedia. *User modeling and user-adapted interaction*, 11(1-2):87–110.

[Bryson and Levit, 1991] Bryson, S. and Levit, C. (1991). The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows. In *Proceedings of the 2Nd Conference on Visualization '91*, VIS '91, pages 17–24, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Burdea, 1996] Burdea, G. (1996). *Force and touch feedback for virtual reality*. Wiley New York.

[Burdea and Coiffet, 2003] Burdea, G. C. and Coiffet, P. (2003). *Virtual Reality Technology*. John Wiley & Sons, Inc., New York, NY, USA, 2 edition.

[Buxton, 1983] Buxton, W. (1983). Lexical and pragmatic considerations of input structures. *SIGGRAPH Comput. Graph.*, 17(1):31–37.

[Calvary, 2007] Calvary, G. (2007). *Plasticité des Interfaces Homme-Machine*. PhD thesis. Thèse Habilitation à Diriger des Recherches préparée au Laboratoire d'Informatique de Grenoble (LIG), Université Joseph Fourier.

[Calvary et al., 2004a] Calvary, G., Coutaz, J., Dâassi, O., Balme, L., and Demeure, A. (2004a). Towards a new generation of widgets for supporting software plasticity: the" comet". In *Engineering Human Computer Interaction and Interactive Systems*, pages 306–324. Springer.

[Calvary et al., 2004b] Calvary, G., Coutaz, J., Daassi, O., Balme, L., and Demeure, A. (2004b). Towards a new generation of widgets for supporting software plasticity: the 'comet'. In *EHCI-DSVIS'04*, pages 306–323. Hamburg, Germany.

[Calvary et al., 2002a] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., and Vanderdonckt, J. (2002a). Plasticity of user interfaces : A revised reference framework. In *First International Workshop on Task Models and Diagrams for User Interface Design TAMODIA'2002, Bucarest*, pages 127–134.

[Calvary et al., 2002b] Calvary, G., Coutaz, J., Thevenin, D. B., L., Florins, M., Limbourg, Q., Souchon, N., Vanderdonckt, J., Marucci, L., Paterno, F., and Santoro, C. (2002b). The CAMELEON Reference Framework. *Deliverable D1.1.*

[Card et al., 1983] Card, S. K., Newell, A., and Moran, T. P. (1983). *The Psychology of Human-Computer Interaction.* L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

[Carey and Bell, 1997] Carey, R. and Bell, G. (1997). *The annotated VRML 2.0 reference manual.* Addison-Wesley Longman Ltd.

[Cashion et al., 2013] Cashion, J., Wingrave, C., and LaViola, J. (2013). Optimal 3d selection technique assignment using real-time contextual analysis. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on*, pages 107–110.

[Castaneda and Navab, 2011] Castaneda, V. and Navab, N. (2011). Time-of-flight and kinect imaging. *Kinect Programming for Computer Vision.*

[Celentano et al., 2004] Celentano, A., Nodari, M., and Pittarello, F. (2004). Adaptive interaction in web3d virtual worlds. In *Proceedings of the Ninth International Conference on 3D Web Technology*, Web3D '04, page 41–50, New York, NY, USA. ACM.

[Cellary and Walczak, 2012] Cellary, W. and Walczak, K. (2012). *Interactive 3D MultiMedia Content.* Springer.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

[Chen et al., 2000] Chen, G., Kotz, D., et al. (2000). A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

[Chen et al., 2011] Chen, W., Fournier, J., Barkowsky, M., and Le Callet, P. (2011). New stereoscopic video shooting rule based on stereoscopic distortion parameters and comfortable viewing zone. In *Stereoscopic Displays and Applications XXII, SPIE 2011*, pages SPIE 7863, 78631O, San Francisco, United States.

[Chéron et al., 2015] Chéron, G., Laptev, I., and Schmid, C. (2015). P-cnn: pose-based cnn features for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3218–3226.

[Chevaillier et al., 2012] Chevaillier, P., Trinh, T.-H., Barange, M., De Loor, P., Devillers, F., Soler, J., and Querrec, R. (2012). Semantic modeling of virtual environments using MASCARET. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012 5th Workshop on*, page 1–8.

[Chittaro et al., 2002] Chittaro, L., Ranon, R., Artificial, I. S., and Realities, V. (2002). Dynamic generation of personalized VRML content: a general approach and its application to 3d e-commerce. In *In Proceedings of Web3D 2002: 7th International Conference on 3D Web*, page 145–154. Press.

[Claude et al., 2015] Claude, G., Gouranton, V., and Arnaldi, B. (2015). Roles in collaborative virtual environments for training. In *International Conference on Artificial Reality and Telexistence Eurographics Symposium on Virtual Environments (2015)*, pages 1–8.

[Claude et al., 2014] Claude, G., Gouranton, V., Bouville Berthelot, R., and Arnaldi, B. (2014). #SEVEN, a Sensor Effector Based Scenarios Model for Driving Collaborative Virtual Environment. In *ICAT-EGVE*.

[Cobb et al., 2008] Cobb, S., D'Cruz, M., Day, A., David, P., Gardeux, F., Broek, v. d. E., Voort, M., Meijer, F., Izkara, J. L., and Mavrikios, D. (2008). How is vr used to support training in industry? the intuition network of excellence working group on education and training.

[Coffey et al., 2011] Coffey, D., Malbraaten, N., Le, T., Borazjani, I., Sotiropoulos, F., and Keefe, D. F. (2011). Slice wim: A multi-surface, multi-touch interface for overview+detail exploration of volume datasets in virtual reality. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 191–198, New York, NY, USA. ACM.

[Coutaz, 1987] Coutaz, J. (1987). PAC, on object oriented model for dialog design. In *Interact'87*. 6 pages.

[Cruz-Neira et al., 1992] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., and Hart, J. C. (1992). The CAVE: Audio Visual Experience Automatic Virtual Environment. *Commun. ACM*, 35(6):64–72.

[Cutler et al., 1997] Cutler, L. D., Fröhlich, B., and Hanrahan, P. (1997). Two-handed direct manipulation on the responsive workbench. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, page 107–114.

[Dachselt et al., 2006] Dachselt, R., Hinz, M., and Pietschmann, S. (2006). Using the AMACONT architecture for flexible adaptation of 3d web applications. In *Proceedings of the Eleventh International Conference on 3D Web Technology*, Web3D '06, page 75–84, New York, NY, USA. ACM.

[De Sa and Zachmann, 1999] De Sa, A. G. and Zachmann, G. (1999). Virtual reality as a tool for verification of assembly and maintenance processes. *Computers & Graphics*, 23(3):389–403.

[Demeure et al., 2005] Demeure, A., Balme, L., and Calvary, G. (2005). CamNote: A plastic slides viewer. Plastic Services for Mobile Devices (PSMD), Workshop hel in conjunction with Interact'05, Rome, 12 Septembre 2005.

[Demeure et al., 2008] Demeure, A., Sottet, J.-S., Calvary, G., Coutaz, J., Ganneau, V., and Vanderdonckt, J. (2008). The 4c reference model for distributed user interfaces. In *Autonomic and Autonomous Systems, 2008. ICAS 2008. Fourth International Conference on*, pages 61–69. IEEE.

[Dey, 2001] Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7.

[dos Santos and Osorio, 2004] dos Santos, C. T. and Osorio, F. S. (2004). AdapTIVE: An intelligent virtual environment and its application in e-commerce. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, page 468–473.

[dos Santos and Osório, 2004] dos Santos, C. T. and Osório, F. S. (2004). An intelligent and adaptive virtual environment and its application in distance learning. In *Proceedings of the working conference on Advanced visual interfaces*, page 362–365.

[Dragicevic and Fekete, 2001] Dragicevic, P. and Fekete, J.-D. (2001). Input device selection and interaction configuration with ICON. In *People and Computers XV—Interaction without Frontiers*, page 543–558. Springer.

[Dubois et al., 1999] Dubois, E., Nigay, L., Troccaz, J., Chavanon, O., and Carrat, L. (1999). Classification space for augmented surgery. an augmented reality case study. In *Humancomputer interaction, INTERACT'99: IFIP TC. 13 International Conference on Human-Computer Interaction, 30th August-3rd September 1999, Edinburgh, UK*, volume 1, page 353.

[Dupont et al., 2010] Dupont, F., Duval, T., Fleury, C., Forest, J., Gouranton, V., Lando, P., Laurent, T., Lavoué, G., and Schmutz, A. (2010). Collaborative scientific visualization: the collaviz framework. In *JVRC 2010 (2010 Joint Virtual Reality Conference of EuroVR-EGVE-VEC)*.

[Duval et al., 2014] Duval, T., Blouin, A., and Jézéquel, J.-M. (2014). When model driven engineering meets virtual reality: Feedback from application to the collaviz framework. In *7th Workshop SEARIS*.

[Echtler et al., 2004] Echtler, F., Sturm, F., Kindermann, K., Klinker, G., Stilla, J., Trilk, J., and Najafi, H. (2004). The intelligent welding gun: Augmented reality for experimental vehicle construction. In *Virtual and augmented reality applications in manufacturing*, pages 333–360. Springer.

[Elmqvist, 2011] Elmqvist, N. (2011). Distributed user interfaces: State of the art. In *Distributed User Interfaces*, pages 1–12. Springer.

[Esnault et al., 2010] Esnault, N., Royan, J., Cozot, R., and Bouville, C. (2010). A flexible framework to personalize 3d web users experience. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, page 35–44, New York, NY, USA. ACM.

[Fernandes and Feiner, 2016] Fernandes, A. S. and Feiner, S. K. (2016). Combating vr sickness through subtle dynamic field-of-view modification. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 201–210.

[Fischer, 2001] Fischer, G. (2001). User modeling in human–computer interaction. *User modeling and user-adapted interaction*, 11(1-2):65–86.

[Fleury et al., 2010] Fleury, C., Duval, T., and Gouranton, V. (2010). Architectures and mechanisms to maintain efficiently consistency in collaborative virtual environments. In *SEARIS 2010 (IEEE VR 2010 Workshop on Software Engineering and Architectures for Realtime Interactive Systems)*.

[Fleury et al., 2012] Fleury, C., Duval, T., Gouranton, V., and Steed, A. (2012). Evaluation of Remote Collaborative Manipulation for Scientific Data Analysis. In *VRST 2012 - 18th Symposium on Virtual Reality Software and Technology*, pages 129–136, Toronto, Canada. ACM.

[Florens et al., 2007] Florens, J.-L., Hulin, T., Gil, J. J., and Davy, P. (2007). Force feedback device / force properties. In *Enaction and enactive interfaces : a handbook of terms*, pages 106–108. Enactive Systems Books.

[Foxlin et al., 2002] Foxlin, E. et al. (2002). Motion tracking requirements and technologies. *Handbook of virtual environment technology*, 8:163–210.

[Fuchs et al., 2003] Fuchs, P., Arnaldi, B., and Tisseau, J. (2003). La réalité virtuelle et ses applications. *Le traité de la réalité virtuelle - 2eme édition*, pages 3–51.

[Fuchs et al., 2011] Fuchs, P., Moreau, G., and Guitton, P. (2011). *Virtual Reality: Concepts and Technologies*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

[Gaucher et al., 2013] Gaucher, P., Argelaguet, F., Royan, J., and Lécuyer, A. (2013). A novel 3d carousel based on pseudo-haptic feedback and gestural interaction for virtual showcasing. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on*, pages 55–58. IEEE.

[Golemati et al., 2006] Golemati, M., Halatsis, C., Vassilakis, C., Katifori, A., and Lepouras, G. (2006). A context-based adaptive visualization environment. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pages 62–67. IEEE.

[Gonzalez-Calleros et al., 2009] Gonzalez-Calleros, J., Vanderdonckt, J., and Muoz-Arteaga, J. (2009). A structured approach to support 3d user interface development. In *Advances in Computer-Human Interactions, 2009. ACHI '09. Second International Conferences on*, pages 75–81.

[Green and Lo, 2004] Green, M. and Lo, J. (2004). The grappl 3d interaction technique library. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 16–23. ACM.

[Gregory, 2009] Gregory, J. (2009). *Game engine architecture*. CRC Press.

[Grolaux et al., 2001] Grolaux, D., Roy, P. V., and Vanderdonckt, J. (2001). QTk: An integrated model-based approach to designing executable user interfaces. In *DEPT. OF COMPUTER SCIENCE, UNIV. OF GLASGOW*, page 77–91. Springer Verlag.

[Guinan et al., 2000] Guinan, T., O'Hare, C., and Doikov, N. (2000). Enter: The personalisation and contextualisation of 3-dimensional worlds. In *Parallel and Distributed Processing, 2000. Proceedings. 8th Euromicro Workshop on*, page 142–148.

[Han et al., 2012] Han, S., Han, J.-J., Kim, J. D. K., and Yeong Kim, C. (2012). Connecting users to virtual worlds within MPEG-v standardization. *Signal Processing: Image Communication*.

[Hand, 1997] Hand, C. (1997). A survey of 3d interaction techniques. In *Computer graphics forum*, volume 16, page 269–281.

[Haouchine et al., 2013] Haouchine, N., Dequidt, J., Peterlik, I., Kerrien, E., Berger, M.-O., and Cotin, S. (2013). Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 199–208. IEEE.

[Hatala et al., 2004] Hatala, M., Kalantari, L., Wakkary, R., and Newby, K. (2004). Ontology and rule based retrieval of sound objects in augmented audio reality system for museum visitors. In *Proceedings of the 2004 ACM symposium on Applied computing*, page 1045–1050.

[Henry et al., 2012] Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663.

[Hirata and Sato, 1992] Hirata, Y. and Sato, M. (1992). 3-dimensional interface device for virtual work space. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, volume 2, pages 889–896. IEEE.

[Hirota et al., 1996] Hirota, G., Chen, D. T., Garrett, W. F., Livingston, M. A., et al. (1996). Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 429–438. ACM.

[Hoppe, 1996] Hoppe, H. (1996). Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, page 99–108. ACM.

[Hu et al., 2008] Hu, S.-Y., Huang, T.-H., Chang, S.-C., Sung, W.-L., Jiang, J.-R., and Chen, B.-Y. (2008). Flod: A framework for peer-to-peer 3d streaming. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE.

[Iwata et al., 2004] Iwata, H., Yano, H., Uemura, T., and Moriya, T. (2004). Food simulator: A haptic interface for biting. In *Virtual Reality, 2004. Proceedings. IEEE*, pages 51–57. IEEE.

[Jorgensen et al., 2000] Jorgensen, C., Wheeler, K., Stepniewski, S., and Norvig, P. (2000). Bioelectric control of a 757 class high fidelity aircraft simulation.

[Julier et al., 2002] Julier, S., Baillot, Y., Brown, D., and Lanzagorta, M. (2002). Information filtering for mobile augmented reality. *IEEE Comput. Graph. Appl.*, 22(5):12–15.

[Julier et al., 2000] Julier, S., Lanzagorta, M., Baillot, Y., Rosenblum, L., Feiner, S., Hollerer, T., and Sestito, S. (2000). Information filtering for mobile augmented reality. In *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, page 3–11.

[Kanbara and Yokoya, 2002] Kanbara, M. and Yokoya, N. (2002). Geometric and photometric registration for real-time augmented reality. In *Mixed and Augmented Reality, 2002. ISMAR 2002. Proceedings. International Symposium on*, page 279–280.

[Kennedy et al., 1993] Kennedy, R. S., Lane, N. E., Berbaum, K. S., and Lilienthal, M. G. (1993). Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The international journal of aviation psychology*, 3(3):203–220.

[Kim et al., 2004] Kim, H., Joslin, C., Di Giacomo, T., Garchery, S., and Magnenat-Thalmann, N. (2004). Adaptation mechanism for three dimensional content within the mpeg-21 framework. In *Computer Graphics International, 2004. Proceedings*, page 462–469.

[Klokmose and Beaudouin-Lafon, 2009] Klokmose, C. N. and Beaudouin-Lafon, M. (2009). VIGO: Instrumental Interaction in Multi-surface Environments. CHI 2009, pages 869–878, New York, NY, USA. ACM.

[Kobsa, 1993] Kobsa, A. (1993). User modeling: Recent work, prospects and hazards. *Human Factors in Information Technology*, 10:111–111.

[Kobsa, 1995] Kobsa, A. (1995). Supporting user interfaces for all through user modeling. *Advances in Human Factors/Ergonomics*, 20:155–157.

[Kobsa, 2001] Kobsa, A. (2001). Generic user modeling systems. *User modeling and user-adapted interaction*, 11(1-2):49–63.

[Lacoche et al., 2014] Lacoche, J., Duval, T., Arnaldi, B., Maisel, E., and Royan, J. (2014). A survey of plasticity in 3d user interfaces. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2014 IEEE 7th Workshop on*, pages 19–26. IEEE.

[Lacoche et al., 2015a] Lacoche, J., Duval, T., Arnaldi, B., Maisel, E., and Royan, J. (2015a). Plasticity for 3d user interfaces: new models for devices and interaction techniques. In *EICS 2015: 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 28–33. ACM.

[Lacoche et al., 2016] Lacoche, J., Duval, T., Arnaldi, B., Maisel, É., and Royan, J. (2016). D3part: A new model for redistribution and plasticity of 3d user interfaces. In *3DUI 2016: IEEE symposium on 3D User Interfaces Summit*, pages 23–36. IEEE.

[Lacoche et al., 2015b] Lacoche, J., Le Chénéchal, M., Chalmé, S., Royan, J., Duval, T., Gouranton, V., Maisel, É., and Arnaldi, B. (2015b). Dealing with frame cancellation for stereoscopic displays in 3d user interfaces. In *IEEE Symposium on 3D User Interfaces, 3DUI 2015*, pages 73–80. IEEE.

[Latoschik, 2002] Latoschik, M. E. (2002). Designing transition networks for multimodal VR-interactions using a markup language. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, ICMI '02, page 411–, Washington, DC, USA. IEEE Computer Society.

[Latoschik, 2005] Latoschik, M. E. (2005). A user interface framework for multimodal VR interactions. In *Proceedings of the 7th International Conference on Multimodal Interfaces*, ICMI '05, page 76–83, New York, NY, USA. ACM.

[Le Chenechal et al., 2015] Le Chenechal, M., Lacoche, J., Martin, C., and Royan, J. (2015). Laying out spaces with virtual reality. In *Virtual Reality (VR), 2015 IEEE*, pages 337–338. IEEE.

[Le Chénéchal et al., 2016a] Le Chénéchal, M., Lacoche, J., Royan, J., Duval, T., Gouranton, V., and Arnaldi, B. (2016a). When the giant meets the ant an asymmetric approach for collaborative and concurrent object manipulation in a multi-scale environment. In *3DCVE 2016: International Workshop on Collaborative Virtual Environments*, pages 1–4. IEEE.

[Le Chénéchal et al., 2016b] Le Chénéchal, M., Lacoche, J., Royan, J., Duval, T., Gouranton, V., and Arnaldi, B. (2016b). When the giant meets the ant an asymmetric approach for collaborative object manipulation. In *IEEE Symposium on 3D User Interfaces, 3DUI 2016*. IEEE.

[Lechner, 2013] Lechner, M. (2013). Arml 2.0 in the context of existing ar data formats. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2013 6th Workshop on*, pages 41–47. IEEE.

[Lecuyer et al., 2000] Lecuyer, A., Coquillart, S., Kheddar, A., Richard, P., and Coiffet, P. (2000). Pseudo-haptic feedback: can isometric input devices simulate force feedback? In *Virtual Reality, 2000. Proceedings. IEEE*, pages 83–90. IEEE.

[Lee and Green, 2005] Lee, W. L. and Green, M. (2005). A layout framework for 3d user interfaces. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '05, page 96–105, New York, NY, USA. ACM.

[Lee and Green, 2006] Lee, W. L. and Green, M. (2006). Automatic layout for 3d user interfaces construction. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, page 113–120.

[Limbourg et al., 2004] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and López-Jaquero, V. (2004). Usixml: a language supporting multi-path development of user interfaces. *EHCI/DS-VIS*, 3425:200–220.

[Lindt, 2005] Lindt, I. (2005). Exchangeability of 3d interaction techniques. In *Proceedings of the IEEE Workshop on New Directions in 3D User Interfaces*, page 93–94.

[Lindt, 2009] Lindt, I. (2009). *Adaptive 3D-User-Interfaces*. PhD thesis.

[Lipscomb and Pique, 1993] Lipscomb, J. S. and Pique, M. E. (1993). Analog input device physical characteristics. *SIGCHI Bull.*, 25(3):40–45.

[Lipton and Akka, 2010] Lipton, L. and Akka, B. (2010). Vertical surround parallax correction. US Patent 7,679,641.

[Livingston et al., 2011] Livingston, M. A., Rosenblum, L. J., Brown, D. G., Schmidt, G. S., Julier, S. J., Baillot, Y., Swan II, J. E., Ai, Z., and Maassel, P. (2011). Military applications of augmented reality. In *Handbook of augmented reality*, pages 671–706. Springer.

[Lops et al., 2011] Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer.

[Lotte et al., 2013] Lotte, F., Faller, J., Guger, C., Renard, Y., Pfurtscheller, G., Lécuyer, A., and Leeb, R. (2013). Combining BCI with Virtual Reality: Towards New Applications and Improved BCI. In Allison, B. Z., Dunne, S., Leeb, R., Millán, J. D. R., and Nijholt, A., editors, *Towards Practical Brain-Computer Interfaces:*. Springer.

[Lotte et al., 2008] Lotte, F., Renard, Y., and Lécuyer, A. (2008). Self-paced brain-computer interaction with virtual worlds: A quantitative and qualitative study "out of the lab". In *4th international brain computer interface workshop and training course*.

[Mackinlay et al., 1990] Mackinlay, J., Card, S. K., and Robertson, G. G. (1990). A semantic analysis of the design space of input devices. *Hum.-Comput. Interact.*, 5(2):145–190.

[Mackinlay et al., 1991] Mackinlay, J. D., Robertson, G. G., and Card, S. K. (1991). The perspective wall: Detail and context smoothly integrated. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176. ACM.

[Marchal et al., 2011] Marchal, M., Pettr, J., and Lécuyer, A. (2011). Joyman: A human-scale joystick for navigating in virtual worlds. In *3D User Interfaces (3DUI), 2011 IEEE Symposium on*, pages 19–26. IEEE.

[Marcotte, 2010] Marcotte, E. (2010). Responsive web design. *A list apart*, 306.

[Margery et al., 2002] Margery, D., Arnaldi, B., Chauffaut, A., Donikian, S., and Duval, T. (2002). Openmask:{Multi-Threaded| Modular} animation and simulation {Kernel| Kit}: a general introduction. In *VRIC*, pages 101–110.

[Margery et al., 1999] Margery, D., Arnaldi, B., and Plouzeau, N. (1999). *A general framework for cooperative manipulation in virtual environments*. Springer.

[Maynes-Aminzade, 2005] Maynes-Aminzade, D. (2005). Edible bits: Seamless interfaces between people, data and food. In *Conference on Human Factors in Computing Systems (CHI'05)-Extended Abstracts*, pages 2207–2210. Citeseer.

[Medeiros et al., 2013] Medeiros, D., Carvalho, F., Teixeira, L., Braz, P., Raposo, A., and Santos, I. (2013). Proposal and evaluation of a tablet-based tool for 3D virtual environments. *SBC*, 4(2):31.

[Mekni and Lemieux, 2014] Mekni, M. and Lemieux, A. (2014). Augmented reality: Applications, challenges and future trends. In *Applied Computational Science—Proceedings of the 13th International Conference on Applied Computer and Applied Computational Science (ACACOS '14) Kuala Lumpur, Malaysia*, pages 23–25.

[Melchior et al., 2009] Melchior, J., Grolaux, D., Vanderdonckt, J., and Van Roy, P. (2009). A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 69–78. ACM.

[Mercier-Ganady et al., 2014] Mercier-Ganady, J., Lotte, F., Loup-Escande, E., Marchal, M., and Lécuyer, A. (2014). The Mind-Mirror: See Your Brain in Action in Your Head Using EEG and Augmented Reality. In *IEEE Virtual Reality (VR)*, Minneapolis, United States. IEEE.

[Mestre et al., 2006] Mestre, D., Fuchs, P., Berthoz, A., and Vercher, J. (2006). Immersion et présence. *Le traité de la réalité virtuelle. Paris: Ecole des Mines de Paris*, pages 309–38.

[Miles, 1930] Miles, W. R. (1930). Ocular dominance in human adults. *The journal of general psychology*, 3(3):412–430.

[Milgram and Kishino, 1994] Milgram, P. and Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329.

[Mine et al., 1995] Mine, M. et al. (1995). Virtual environment interaction techniques. *UNC Chapel Hill computer science technical report TR95-018*, pages 507248–2.

[Mizell, 2001] Mizell, D. (2001). Boeing's wire bundle assembly project. *Fundamentals of wearable computers and augmented reality*, 5.

[Mohri et al., 2012] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

[Mora and Odobez, 2012] Mora, K. A. F. and Odobez, J.-M. (2012). Gaze estimation from multimodal kinect data. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 25–30. IEEE.

[Motti et al., 2013] Motti, L. G., Vigouroux, N., and Gorce, P. (2013). Interaction techniques for older adults using touchscreen devices : a literature review. In *25ème conférence francophone sur l'Interaction Homme-Machine, IHM'13*, Bordeaux, France. ACM.

[Mulder and Liere, 2000] Mulder, J. D. and Liere, R. V. (2000). Enhancing fish tank vr. In *In Proceedings of Virtual Reality*, pages 91–98. IEEE.

[Myers et al., 2000] Myers, B., Hudson, S. E., and Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):3–28.

[Nahon et al., 2015] Nahon, D., Subileau, G., and Capel, B. (2015). Never blind vr: enhancing the virtual reality headset experience with augmented virtuality. In *Virtual Reality (VR), 2015 IEEE*, pages 347–348.

[Nakamoto and Minh, 2007] Nakamoto, T. and Minh, H. P. D. (2007). Improvement of olfactory display using solenoid valves. In *Virtual Reality Conference, 2007. VR'07. IEEE*, pages 179–186. IEEE.

[Narumi et al., 2011] Narumi, T., Nishizaka, S., Kajinami, T., Tanikawa, T., and Hirose, M. (2011). Augmented reality flavors: Gustatory display based on edible marker and cross-modal interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 93–102, New York, NY, USA. ACM.

[Navab et al., 1999] Navab, N., Bani-Kashemi, A., and Mitschke, M. (1999). Merging visible and invisible: Two camera-augmented mobile c-arm (camc) applications. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*, pages 134–141. IEEE.

[Octavia et al., 2011] Octavia, J., Raymaekers, C., and Coninx, K. (2011). Adaptation in virtual environments: conceptual framework and user models. *Multimedia Tools and Applications*, 54(1):121–142.

[Octavia et al., 2009] Octavia, J. R., Raymaekers, C., and Coninx, K. (2009). A conceptual framework for adaptation and personalization in virtual environments. In *Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on*, page 284–288.

[Ohlenburg et al., 2007] Ohlenburg, J., Broll, W., and Lindt, I. (2007). DEVAL–a device abstraction layer for VR/AR. In *Universal Acess in Human Computer Interaction. Coping with Diversity*, page 497–506. Springer.

[Ohlenburg et al., 2004] Ohlenburg, J., Herbst, I., Lindt, I., Fröhlich, T., and Broll, W. (2004). The MORGAN framework: enabling dynamic multi-user AR and VR projects. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 166–169.

[Panas et al., 2003] Panas, T., Berrigan, R., and Grundy, J. (2003). A 3d metaphor for software production visualization. In *null*, page 314. IEEE.

[Pentenrieder et al., 2007] Pentenrieder, K., Bade, C., Doil, F., and Meier, P. (2007). Augmented reality-based factory planning-an application tailored to industrial needs. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 31–42. IEEE.

[Piekarski et al., 1999] Piekarski, W., Gunther, B., and Thomas, B. (1999). Integrating virtual and augmented realities in an outdoor application. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*, pages 45–54. IEEE.

[Pierce et al., 1997] Pierce, J. S., Forsberg, A. S., Conway, M. J., Hong, S., Zeleznik, R. C., and Mine, M. R. (1997). Image plane interaction techniques in 3d immersive environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 39–ff. ACM.

[Pinho et al., 2002] Pinho, M. S., Bowman, D. A., and Freitas, C. M. (2002). Cooperative object manipulation in immersive virtual environments: framework and techniques. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 171–178. ACM.

[Poupyrev et al., 1996] Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (1996). The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80. ACM.

[Rauterberg et al., 1998] Rauterberg, M., Fjeld, M., Krueger, H., Bichsel, M., Leonhardt, U., and Meier, M. (1998). Build-it: a planning tool for construction and design. In *CHI 98 Cconference Summary on Human Factors in Computing Systems*, pages 177–178. ACM.

[Reenskaug, 1979] Reenskaug, T. (1979). Models-views-controllers. *Technical note, Xerox PARC*, 32(55):6–2.

[Regenbrecht et al., 2004] Regenbrecht, H., Lum, T., Kohler, P., Ott, C., Wagner, M., Wilke, W., and Mueller, E. (2004). Using augmented virtuality for remote collaboration. *Presence: Teleoperators and virtual environments*, 13(3):338–354.

[Reitmayr and Schmalstieg, 2001] Reitmayr, G. and Schmalstieg, D. (2001). An open software architecture for virtual reality interaction. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, page 47–54, New York, NY, USA. ACM.

[Rekimoto, 1997] Rekimoto, J. (1997). Pick-and-drop: a direct manipulation technique for multiple computer environments. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39. ACM.

[Rekimoto and Nagao, 1995] Rekimoto, J. and Nagao, K. (1995). The world through the computer: Computer augmented interaction with real world environments. In *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, UIST '95, pages 29–36, New York, NY, USA. ACM.

[Renard et al., 2010] Renard, Y., Lotte, F., Gibert, G., Congedo, M., Maby, E., Delannoy, V., Bertrand, O., and Lécuyer, A. (2010). Openvibe: an open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments. *Presence: teleoperators and virtual environments*, 19(1):35–53.

[Rickel and Johnson, 1999] Rickel, J. and Johnson, W. L. (1999). Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied artificial intelligence*, 13(4-5):343–382.

[Riege et al., 2006] Riege, K., Holtkämper, T., Wesche, G., and Fröhlich, B. (2006). The bent pick ray: An extended pointing technique for multi-user interaction. In *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on*, pages 62–65. IEEE.

[Rolland et al., 2001] Rolland, J. P., Davis, L., and Baillot, Y. (2001). A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augmented reality*, 1:67–112.

[Ross, 2000] Ross, E. (2000). Intelligent user interfaces: Survey and research directions. *University of Bristol, Bristol, UK*.

[Royan et al., 2007] Royan, J., Gioia, P., Cavagna, R., and Bouville, C. (2007). Network-based visualization of 3d landscapes and city models. *IEEE Comput. Graph. Appl.*, 27(6):70–79.

[Sato et al., 2008] Sato, J., Ohtsu, K., Bannai, Y., and Okada, K.-i. (2008). Pulse ejection technique of scent to create dynamic perspective. In *18th International Conference on Artificial Reality and Telexistence*, pages 167–174. Citeseer.

[Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, page 85–90.

[Schön et al., 2004] Schön, B., O'Hare, C., Duffy, B. R., Martin, A. N., and Bradley, M. a. J. F. (2004). An agent-based approach to adaptive navigational support within 3d-environments. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 64–68. IEEE.

[Schroeder et al., 1992] Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. (1992). Decimation of triangle meshes. In *ACM SIGGRAPH Computer Graphics*, volume 26, page 65–70. ACM.

[Sherman and Craig, 2002] Sherman, W. R. and Craig, A. B. (2002). *Understanding virtual reality: Interface, application, and design*. Elsevier.

[Shilling and Shinn-Cunningham, 2002] Shilling, R. D. and Shinn-Cunningham, B. (2002). Virtual auditory displays. *Handbook of virtual environment technology*, pages 65–92.

[Shotton et al., 2013] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

[Slater, 2009] Slater, M. (2009). Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1535):3549–3557.

[Smith et al., 2013] Smith, N. G., Knabb, K., DeFanti, C., Weber, P., Schulze, J., Prudhomme, A., Kuester, F., Levy, T. E., and DeFanti, T. A. (2013). Artifactvis2: Managing real-time archaeological data in immersive 3d environments. In *Digital Heritage International Congress (DigitalHeritage), 2013*, volume 1, pages 363–370. IEEE.

[Song and Norman, 1993] Song, D. and Norman, M. L. (1993). Cosmic explorer: A virtual reality environment for exploring cosmic data. In *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*, pages 75–79. IEEE.

[Soukoreff and MacKenzie, 2004] Soukoreff, R. W. and MacKenzie, I. S. (2004). Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in hci. *International journal of human-computer studies*, 61(6):751–789.

[Sparacino, 2003] Sparacino, F. (2003). Sto (ry) chastics: a bayesian network architecture for user modeling and computational storytelling for interactive spaces. In *UbiComp 2003: Ubiquitous Computing*, page 54–72.

[Stoakley et al., 1995a] Stoakley, R., Conway, M. J., and Pausch, R. (1995a). Virtual reality on a WIM: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, page 265–272. ACM Press/Addison-Wesley Publishing Co.

[Stoakley et al., 1995b] Stoakley, R., Conway, M. J., and Pausch, R. (1995b). Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM Press/Addison-Wesley Publishing Co.

[Stricker et al., 2001] Stricker, D., Klinker, G., and Reiners, D. (2001). Augmented reality for exterior construction applications. *Fundamentals of wearable computers and augmented reality*, pages 379–427.

[Stuerzlinger and Smith, 2002] Stuerzlinger, W. and Smith, G. (2002). Efficient manipulation of object groups in virtual environments. In *Virtual Reality, 2002. Proceedings. IEEE*, pages 251–258. IEEE.

[Sutherland, 1968] Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 757–764. ACM.

[Taylor et al., 2001] Taylor, II, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T. (2001). VRPN: A device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, page 55–61, New York, NY, USA. ACM.

[Teler, 2001] Teler, E. (2001). *Streaming of complex 3d scenes for remote walkthroughs*. PhD thesis, The Hebrew University of Jerusalem.

[Thevenin and Coutaz, 1999] Thevenin, D. and Coutaz, J. (1999). Plasticity of user interfaces: Framework and research agenda. In *Proceedings of INTERACT*, volume 99, page 110–117.

[Valkov et al., 2012] Valkov, D., Bolte, B., Bruder, G., and Steinicke, F. (2012). Viargo - a generic virtual reality interaction library. In *5th Workshop SEARIS*.

[Valyus, 1966] Valyus, N. (1966). *Stereoscopy*. Focal library. Focal Press.

[Van Krevelen and Poelman, 2010] Van Krevelen, D. and Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*, 9(2):1.

[Vanderdonckt et al., 2008] Vanderdonckt, J., Calvary, G., Coutaz, J., and Stanciulescu, A. (2008). Multimodality for plastic user interfaces: Models, methods, and principles. In *Multimodal User Interfaces*, pages 61–84. Springer.

[Vasudevan et al., 2010] Vasudevan, R., Zhou, Z., Kurillo, G., Lobaton, E., Bajcsy, R., and Nahrstedt, K. (2010). Real-time stereo-vision system for 3d teleimmersive collaboration. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1208–1213.

[Vogt et al., 2006] Vogt, S., Khamene, A., and Sauer, F. (2006). Reality augmentation for medical procedures: System architecture, single camera marker tracking, and system evaluation. *International Journal of Computer Vision*, 70(2):179–190.

[Ware and Fleet, 1997] Ware, C. and Fleet, D. (1997). Integrating flying and fish tank metaphors with cyclopean scale. In *Computer Graphics International, 1997. Proceedings*, pages 39–46. IEEE.

[Wartell et al., 1999] Wartell, Z., Ribarsky, W., and Hodges, L. (1999). Third-person navigation of whole-planet terrain in a head-tracked stereoscopic environment. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 141–148.

[Wartell, 2002] Wartell, Z. J. (2002). Stereoscopic head-tracked displays: analysis and development of display algorithms.

[Weinschenk and Barker, 2000] Weinschenk, S. and Barker, D. T. (2000). *Designing Effective Speech Interfaces*. John Wiley & Sons, Inc., New York, NY, USA.

[Wingrave et al., 2001] Wingrave, C. A., Bowman, D. A., and Ramakrishnan, N. (2001). A first step towards nuance-oriented interfaces for virtual environments.

[Wingrave et al., 2002] Wingrave, C. A., Bowman, D. A., and Ramakrishnan, N. (2002). Towards preferences in virtual environment interfaces. In *Proceedings of the Workshop on Virtual Environments 2002*, EGVE '02, page 63–72, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Wiza et al., 2003] Wiza, W., Walczak, K., and Cellary, W. (2003). Ave—method for 3d visualization of search results. In *Web Engineering*, pages 204–207. Springer.

[Wiza et al., 2004] Wiza, W., Walczak, K., and Cellary, W. (2004). Periscope: a system for adaptive 3d visualization of search results. In *Proceedings of the ninth international conference on 3D Web technology*, pages 29–40. ACM.

[Wolpaw et al., 2002] Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., and Vaughan, T. M. (2002). Brain–computer interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791.

[Zhou et al., 2008] Zhou, F., Duh, H. B.-L., and Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 193–202, Washington, DC, USA. IEEE Computer Society.

[Zöllner et al., 2011] Zöllner, M., Jetter, H.-C., and Reiterer, H. (2011). *ZOIL: A design paradigm and software framework for post-WIMP distributed user interfaces*. Springer.

List of Figures

List of Tables

Listings

Résumé

Cette thèse s'intéresse à la plasticité des interfaces de Réalité Mixte (RM), c'est-à-dire les applications de Réalité Virtuelle (RV), Réalité Augmentée (RA) et de Virtualité Augmentée (AV). La plasticité d'un système interactif est sa capacité à s'adapter aux contraintes matérielles et environnementales dans le respect de son utilisabilité. La continuité de l'utilisabilité d'une interface plastique est assurée quel que soit le contexte d'usage. Nous proposons ainsi des modèles et une solution logicielle nommée 3DPlasticToolkit afin de permettre aux développeurs de créer des interfaces de réalité mixtes plastiques. Tout d'abord, nous proposons trois modèles pour modéliser les sources d'adaptation : un modèle pour représenter les dispositifs d'interaction et les dispositifs d'affichage, un modèle pour représenter les utilisateurs et leurs préférences et un modèle pour représenter la structure et la sémantique des données. Ces sources d'adaptation vont être prises en compte par un processus d'adaptation qui va déployer dans une application les composants applicatifs adaptés au contexte d'usage grâce à des mécanismes de notation. Le déploiement de ces composants va permettre d'adapter à la fois les techniques d'interaction de l'application et également la présentation de son contenu. Nous proposons également un processus de redistribution qui va permettre à l'utilisateur final de changer la distribution des composants de son système sur différentes dimensions : affichage, utilisateur et plateforme. Ce processus va ainsi permettre à l'utilisateur de changer de plateforme dynamiquement ou encore de combiner plusieurs plateformes. L'implémentation de ces modèles dans 3DPlasticToolkit permet de fournir aux développeurs une solution prête à l'usage qui peut gérer les périphériques actuels de réalité mixte et qui inclut un certain nombre de techniques d'interaction, d'effets visuels et de métaphores de visualisation de données.

Abstract

This PhD thesis focuses on plasticity for Mixed Reality (MR) User interfaces, which includes Virtual Reality (VR), Augmented Reality (AR) and Augmented Virtuality (AV) applications. Plasticity refers to the capacity of an interactive system to withstand variations of both the system physical characteristics and the environment while preserving its usability. Usability continuity of a plastic interface is ensured whatever the context of use. Therefore, we propose a set of software models, integrated in a software solution named 3DPlasticToolkit, that allow any developer to create plastic MR user interfaces. First, we propose three models for modeling adaptation sources: a model for the description of display devices and interaction devices, a model for the description of the users and their preferences, a model for the description of data structure and semantic. These adaptation sources are taken into account by an adaptation process that deploys application components adapted to the context of use thanks to a scoring system. The deployment of these application components lets the system adapt the interaction techniques of the application of its content presentation. We also propose a redistribution process that allows the end-user to change the distribution of his/her application components across multiple dimensions: display, user and platform. Thus, it allows the end-user to switch dynamically of platform or to combine multiple platforms. The implementation of these models in 3DPlasticToolkit provides developers with a ready to use solution for the development of plastic MR user interfaces. Indeed, the solution already integrates different display devices and interaction devices and also includes multiple interaction techniques, visual effects and data visualization metaphors.